National Semiconductor

400011

Data Communications
Local Area Networks
UARTs

Handbook

A Corporate Dedication to Quality and Reliability

National Semiconductor is an industry leader in the manufacture of high quality, high reliability integrated circuits. We have been the leading proponent of driving down IC defects and extending product lifetimes. From raw material through product design, manufacturing and shipping, our quality and reliability is second to none.

We are proud of our success . . . it sets a standard for others to achieve. Yet, our quest for perfection is ongoing so that you, our customer, can continue to rely on National Semiconductor Corporation to produce high quality products for your design systems.

Charles E. Sporck
President, Chief Executive Officer
National Semiconductor Corporation

Wir fühlen uns zu Qualität und Zuverlässigkeit verpflichtet

National Semiconductor Corporation ist führend bei der Herstellung von integrierten Schaltungen hoher Qualität und hoher Zuverlässigkeit. National Semiconductor war schon immer Vorreiter, wenn es galt, die Zahl von IC Ausfällen zu verringern und die Lebensdauern von Produkten zu verbessern. Vom Rohmaterial über Entwurf und Herstellung bis zur Auslieferung, die Qualität und die Zuverlässigkeit der Produkte von National Semiconductor sind unübertroffen.

Wir sind stolz auf unseren Erfolg, der Standards setzt, die für andere erstrebenswert sind. Auch ihre Ansprüche steigen ständig. Sie als unser Kunde können sich auch weiterhin auf National Semiconductor verlassen.

La Qualité et La Fiabilité:
Une Vocation Commune Chez National Semiconductor Corporation

National Semiconductor Corporation est un des leaders industriels qui fabrique des circuits intégrés d'une très grande qualité et d'une fiabilité exceptionelle. National a été le premier à vouloir faire chuter le nombre de circuits intégrés défectueux et a augmenter la durée de vie des produits. Depuis les matières premières, en passant par la conception du produit sa fabrication et son expédition, partout la qualité et la fiabilité chez National sont sans équivalents.

Nous sommes fiers de notre succès et le standard ainsi défini devrait devenir l'objectif à atteindre par les autres sociétés. Et nous continuons à vouloir faire progresser notre recherche de la perfection; il en résulte que vous, qui êtes notre client, pouvez toujours faire confiance à National Semiconductor Corporation, en produisànt des systèmes d'une très grande qualité standard.

Un Impegno Societario di Qualità e Affidabilità

National Semiconductor Corporation è un'industria al vertice nella costruzione di circuiti integrati di altà qualità ed affidabilità. National è stata il principale promotore per l'abbattimento della difettosità dei circuiti integrati e per l'allungamento della vita dei prodotti. Dal materiale grezzo attraverso tutte le fasi di progettazione, costruzione e spedizione, la qualità e affidabilità National non è seconda a nessuno.

Noi siamo orgogliosi del nostro successo che fissa per gli altri un traguardo da raggiungere. Il nostro desiderio di perfezione è d'altra parte illimitato e pertanto tu, nostro cliente, puoi continuare ad affidarti a National Semiconductor Corporation per la produzione dei tuoi sistemi con elevati livelli di qualità.

Charles E. Sporck
President, Chief Executive Officer
National Semiconductor Corporation

DATA COMMUNICATIONS LOCAL AREA NETWORKS UARTS

1990 Edition

Local Area Networks IEEE 802.3

High Speed Serial/IBM Data Communications

ISDN Components

UARTs

Modems

Transmission Line Drivers & Receivers

Physical Dimensions

1
2
3
4
5
6
7

TRADEMARKS

Following is the most current list of National Semiconductor Corporation's trademarks and registered trademarks.

Abuseable™	FAIRCAD™	MST™	SERIES/800™
Anadig™	Fairtech™	Naked-8™	Series 900™
ANS-R-TRAN™	FAST®	National®	Series 3000™
APPS™	5-Star Service™	National Semiconductor®	Series 32000®
ASPECT™	GENIX™	National Semiconductor	Shelf⌿Chek™
Auto-Chem Deflasher™	GNX™	Corp.®	SofChek™
BCP™	HAMR™	NAX 800™	SONIC™
BI-FET™	HandiScan™	Nitride Plus™	SPIRE™
BI-FET II™	HEX 3000™	Nitride Plus Oxide™	Staggered Refresh™
BI-LINE™	HPC™	NML™	STAR™
BIPLAN™	I3L®	NOBUS™	Starlink™
BLC™	ICM™	NSC800™	STARPLEX™
BLX™	INFOCHEX™	NSCISE™	Super-Block™
Brite-Lite™	Integral ISE™	NSX-16™	SuperChip™
BTL™	Intelisplay™	NS-XC-16™	SuperScript™
CheckTrack™	ISE™	NTERCOM™	SYS32™
CIM™	ISE/06™	NURAM™	TapePak®
CIMBUS™	ISE/08™	OXISS™	TDS™
CLASIC™	ISE/16™	P2CMOS™	TeleGate™
Clock⌿Chek™	ISE32™	PC Master™	The National Anthem®
COMBO™	ISOPLANAR™	Perfect Watch™	Time⌿Chek™
COMBO I™	ISOPLANAR-Z™	Pharma⌿Chek™	TINA™
COMBO II™	KeyScan™	PLAN™	TLC™
COPS™ microcontrollers	LMCMOS™	PLANAR™	Trapezoidal™
Datachecker®	M2CMOS™	Plus-2™	TRI-CODE™
DENSPAK™	Macrobus™	Polycraft™	TRI-POLY™
DIB™	Macrocomponent™	POSilink™	TRI-SAFE™
Digitalker®	MAXI-ROM®	POSitalker™	TRI-STATE®
DISCERN™	Meat⌿Chek™	Power + Control™	TURBOTRANSCEIVER™
DISTILL™	MenuMaster™	POWERplanar™	VIP™
DNR®	Microbus™ data bus	QUAD3000™	VR32™
DPVM™	MICRO-DAC™	QUIKLOOK™	WATCHDOG™
ELSTAR™	μtalker™	RAT™	XMOS™
Embedded System	Microtalker™	RTX16™	XPU™
Processor™	MICROWIRE™	SABR™	Z STAR™
E-Z-LINK™	MICROWIRE/PLUS™	Script⌿Chek™	883B/RETS™
FACT™	MOLE™	SCX™	883S/RETS™

DCA® is a registered trademark of Digital Communication Associates, Inc.
IBM®, PC®, XT®, AT®, PS/2® and MICROCHANNEL® are registered trademarks of International Business Machines Corporation.
PAL® is a registered trademark of and used under license from Monolithic Memories, Inc.
abel™ is a trademark of Data I/O Corporation.
BREIF™ and UnderWare™ are trademarks of UnderWare, Inc.
IRMA™ and SMART ALEC™ are trademarks of Digital Communication Associates, Inc.
NetWare™ is a trademark of Novell, Inc.
VAX™ is a trademark of Digital Equipment Corporation.
80286™ is a trademark of Intel Corporation.

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.

2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor

Introduction to VLSI Products
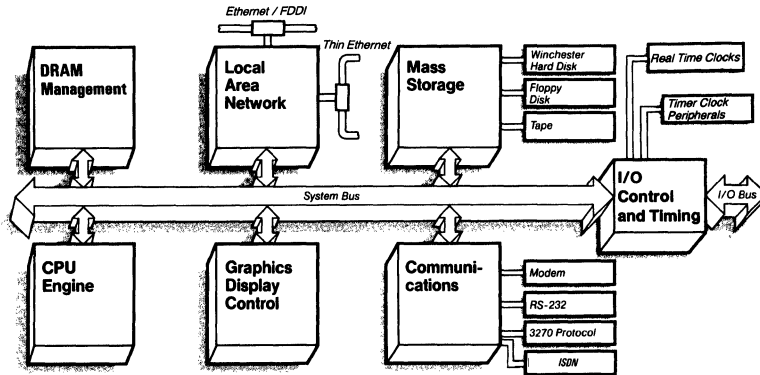
TL/XX/0058-1

National Semiconductor VLSI products include complex peripheral circuits designed to serve a variety of applications. The VLSI products are especially well suited for microcomputer and microprocessor systems such as graphics workstations, personal computers, and many others. National Semiconductor VLSI devices are fully described in a series of databooks and handbooks.

Among the books are the following titles:

MASS STORAGE

The National Semiconductor family of mass storage interface products offers the industry's highest performance and broadest range of products for Winchester hard disks, high performance ESDI and SCSI hard disks and floppy disks. Combined with CLASIC™, analog and high performance microcontroller devices, these products offer unparalleled solutions for integration.

DRAM MANAGEMENT

National Semiconductor offers the broadest range of DRAM controllers with the highest "No-waitstate" performance available on the market. For critical applications, National Semiconductor has developed several 16- and 32-bit Error Checking and Correction (ECC) devices to provide maximum data integrity.

MICROCONTROLLER

As one of the broadest cost/performance product offerings in the industry today, National's microcontrollers provide the intelligence required for high performance applications such as laser printers, ISDN terminal adapters, floppy disks and SCSI hard disks. Complete support tools are available, including applications specific software, Designer's Kits, emulators, simulators, and development systems. Whether the application demands 4-, 8- or 16-bit performance, National has the right embedded control solution.

LOCAL AREA NETWORKS, DATA COMMUNICATIONS, UARTS

National Semiconductor provides a complete three-chip solution for an entire IEEE 802.3 standard for Ethernet/Thin

Ethernet LANs. National Semiconductor offers a completely integrated solution for the IBM 370 class mainframes, System 3X and AS/400 systems for physical layer front end and processing of the IBM 3270/3299 "coaxial" and 5250 "twinaxial" protocols. National's family of UARTs provides high performance, low power serial data input/output interface.

INTERFACE

To drive the communications lines, National Semiconductor has drivers and receivers designed to meet all the major standards such as RS-232, RS-422, and RS-485.

GRAPHICS

The graphics chip set is designed to provide the highest level of performance with minimum demands and loading on the system CPU. The graphics system may be expanded to any number of color planes with virtually unlimited resolution.

REAL TIME CLOCKS

The RTC family provides a simple μP bus compatible interface to any system requiring accurate, reliable, on-going real time and calender functions.

EMBEDDED SYSTEMS PROCESSORS

National's Embedded System Processor™ family offers the most complete solution to 32-bit embedded processor needs via CPUs, slave processors, system peripherals, evaluation/development tools and software.

Our total product system solution approach includes the hardware, software, and development support products necessary for your design. Evaluation board, in-system emulator, software development tools, and third party software are available now.

National Semiconductor

ADVANCE INFORMATION

National Semiconductor FDDI Chip Set Introduction

Background

Fiber Distributed Data Interface (FDDI) is widely recognized as the future technology for the next generation of local area networks. The basic features of FDDI, including the use of fiber optic interconnect, transmission speeds of 100 Mbits/sec, and token ring protocol, position FDDI as a high performance network. As higher performance workstation platforms and distributed computing environments emerge, the need for high performance networking will drive an increasing number of network interfaces toward FDDI. Applications such as distributed databases and graphics applications are placing heavy burdens on existing network technology. FDDI also addresses the needs of interconnectivity. As the various lower speed networks in a corporation grow, the departments need to be connected into a corporate-wide or enterprise system network. FDDI provides a high-speed backbone service for interconnection of IEEE 802.3 Ethernet and 802.5 Token Ring Networks. A typical corporate-wide installation of FDDI is shown in *Figure 1*.

The NSC FDDI Chip Set

National Semiconductor's FDDI chipset represents a high performance, system level approach. Key to the architecture is the goal to provide high throughput and flexibility to interface to a variety of system configurations. Surface-mount packaging is used to reduce the footprint required for the networking electronics. Testability and diagnostics are built-in to aid in fault isolation. In addition, special features for bridging in backbone applications have been incorporated into the chip set. The following pages include details of the four devices which comprise National's FDDI solution.
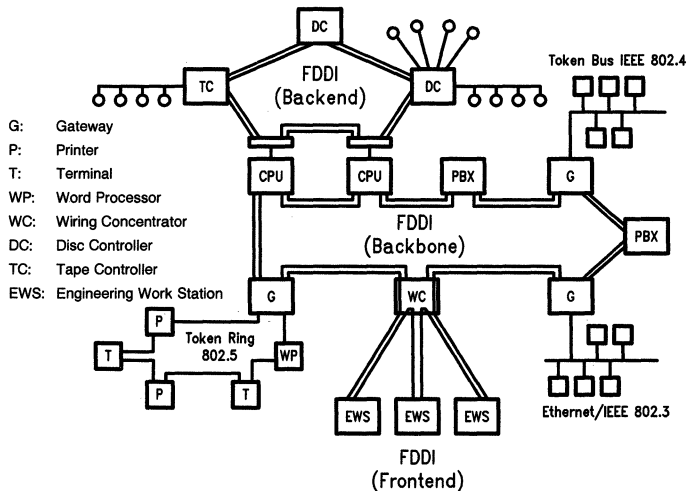
Features

- Single +5V supply
- Status counter for SMT
- Full bridging support
- Full duplex data path
- Low power budget
- Optimized for small footprint
- 100k ECL I/O on all serial bit paths
- Fast lock time on incoming bit stream
- No external counters required for SMT
- Rich set of diagnostic features
- Dedicated SMT microprocessor support
- Separate transmit and receive parallel bus
- Supports DAS, SAS, concentrators
- Point to point cascade option

G: Gateway
P: Printer
T: Terminal
WP: Word Processor
WC: Wiring Concentrator
DC: Disc Controller
TC: Tape Controller
EWS: Engineering Work Station

TL/XX/0180–1

FIGURE 1. Typical Corporate-Wide Installation

National Semiconductor
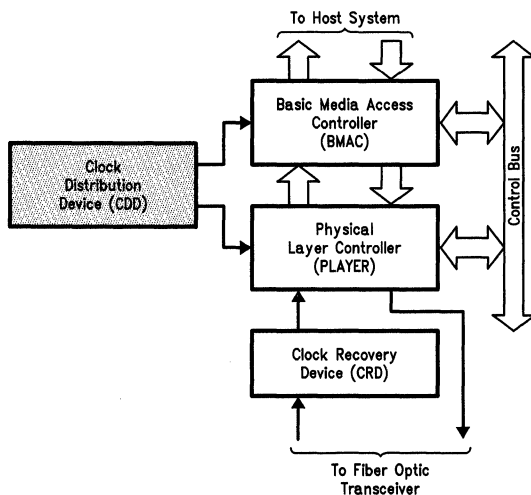
FDDI Clock Distribution Device

General Description

The Clock Distribution Device (CDD) is a clock generation and distribution device intended for use in FDDI (Fiber Distributed Data Interface) networks. The device provides the complete set of clocks required to convert byte wide data to serial format for fiber medium transmission and to move byte wide data between the PLAYER and BMAC functions in various station configurations. 12.5 MHz and 125 MHz differential ECL clocks are generated for the conversion of data to serial format and 12.5 MHz and 25 MHz TTL clocks are generated for the byte wide data transfers.

Features

- Provides 12.5 MHz and 25 MHz TTL clocks
- Provides 12.5 MHz and 125 MHz differential ECL clocks
- 5 phase TTL local byte clocks eliminates clock skew problem in concentrators
- Internal VCO requires no varactors, coils or adjustments
- Option for use of High Q external VCO
- 12.5 MHz clock generated from a 12.5 MHz crystal
- 28-pin PLCC package
- BiCMOS processing

FDDI Chip Set Diagram

TL/XX/0180-2

National Semiconductor
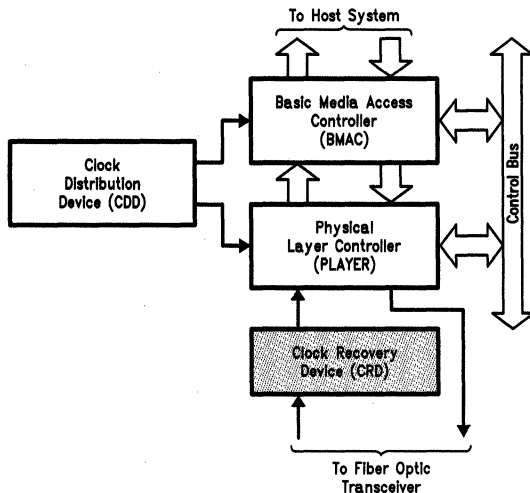
FDDI Clock Recovery Device

General Description

A clock recovery device has been designed for use in 100 Mb/s FDDI (Fiber Distributed Data Interface) networks. The device receives serial data from a Fiber Optic Receiver in differential ECL NRZI 4B/5B group code format and outputs resynchronized NRZI received data and a 125 MHz received clock in differential ECL format for use by the PLAYER device.

Features

■ Clock recovery at 100 Mb/s data rate
■ Internal 250 MHz VCO
  — No varactors or coils required
  — 1% VCO operating range
  — Crystal controlled
■ Precision window centering delay line
■ Excellent window truncation figures
■ User determined PLL loop filters
■ Single +5V supply
■ 28-pin PLCC package
■ BiCMOS Processing

FDDI Chip Set Diagram

To Host System

Basic Media Access Controller (BMAC)

Clock Distribution Device (CDD)

Physical Layer Controller (PLAYER)

Control Bus

Clock Recovery Device (CRD)

To Fiber Optic Transceiver

TL/XX/0180-3

National Semiconductor

FDDI Physical Layer Controller

General Description

The Physical Layer Controller (PLAYER) Devices are a part of National Semiconductor's Fiber Distributed Data Interface (FDDI) chip set solution. It implements one Physical Layer entity as defined by the ANSI X3T9.5 PHY standard. The PLAYER performs the 4B/5B encoding and decoding, serialization and deserialization of data, repeat filter and line state control and detection. It also contains a configuration switch. The PLAYER supports many types of station configuration as allowed by the standard.

Two versions of the PLAYER are available. The first supports single attachment stations and is packaged in an 84-pin PLCC. The second supports dual attachment stations and is packaged in a 132-pin quad flat pack.

Although tailored to the FDDI specification, the PLAYER is well suited for use in high speed point-to-point communication links over optical fibers or coaxial cable.

Features

■ Designed to meet ANSI X3T9.5 FDDI PHY standard
■ Low power CMOS-bipolar process
■ On-chip configuration switch
■ Parity and control bits for each byte
■ Single 5V supply
■ Full duplex operation
■ Single control interface
■ Internal loop back

FDDI Chip Set Diagram

TL/XX/0180–4

National Semiconductor
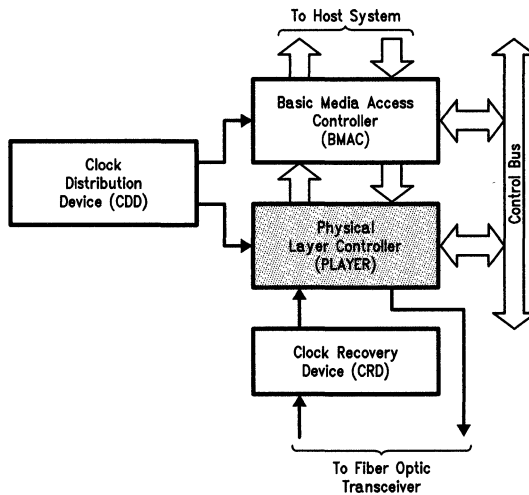
FDDI Basic Media Access Controller

General Description

The Basic Media Access Controller (BMAC) implements the functions defined by the ANSI X3T9.5 FDDI Media Access Control standard using low power CMOS technology. In addition to the functions required by the FDDI standard, the BMAC provides many additional features which enhance station performance, simplify network management and increase network availability.

The BMAC controls the transmitting, receiving, repeating and stripping of frames as well as the generation and checking of FCS codes. The transmit and receive state machines simplify interface software and network management by automatically generating CLAIM and BEACON frames and by framing data for transmission and removing delimiters on reception.

Duplicate address and multiple token error detection during normal operation quickly identifies otherwise hard to detect faults, thereby increasing network reliability and availability. Network Management is simplified further with unique on-chip statistical counters which measure network load and ring latency.

Features

- Designed to meet ANSI X3T9.5 FDDI MAC standard
- Low power CMOS
- Full duplex data path allows transmission to self
- Synchronous, Multiple Asynchronous and immediate service classes supported
- Individual, group and external addressing support
- Automatic Beacon and Claim frames generation
- On-chip statistical counters for easier network management

FDDI Chip Set Diagram

TL/XX/0180-5

National Semiconductor
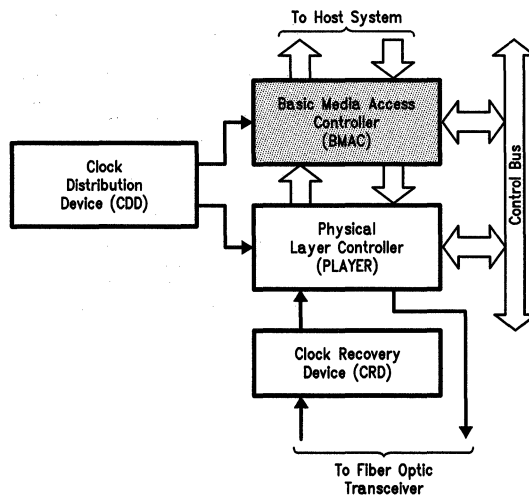
Product Status Definitions

Definition of Terms

Data Sheet Identification	Product Status	Definition
Advance Information	Formative or In Design	This data sheet contains the design specifications for product development. Specifications may change in any manner without notice.
Preliminary	First Production	This data sheet contains preliminary data, and supplementary data will be published at a later date. National Semiconductor Corporation reserves the right to make changes at any time without notice in order to improve design and supply the best possible product.
No Identification Noted	Full Production	This data sheet contains final specifications. National Semiconductor Corporation reserves the right to make changes at any time without notice in order to improve design and supply the best possible product.

National Semiconductor Corporation reserves the right to make changes without further notice to any products herein to improve reliability, function or design. National does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights, nor the rights of others.

Table of Contents

Table of Contents (Continued)

Table of Contents (Continued)

Alpha-Numeric Index

Alpha-Numeric Index (Continued)

Alpha-Numeric Index (Continued)

Alpha-Numeric Index (Continued)

Section 1
Local Area Networks
IEEE 802.3

1

Section 1 Contents

National Semiconductor

DP8390C/NS32490C Network Interface Controller

General Description

The DP8390C/NS32490C Network Interface Controller (NIC) is a microCMOS VLSI device designed to ease interfacing with CSMA/CD type local area networks including Ethernet, Thin Ethernet (Cheapernet) and StarLAN. The NIC implements all Media Access Control (MAC) layer functions for transmission and reception of packets in accordance with the IEEE 802.3 Standard. Unique dual DMA channels and an internal FIFO provide a simple yet efficient packet management design. To minimize system parts count and cost, all bus arbitration and memory support logic are integrated into the NIC.

The NIC is the heart of a three chip set that implements the complete IEEE 802.3 protocol and node electronics as shown below. The other two chips are the DP8391 Serial Network Interface (SNI) and the DP8392A Coaxial Transceiver Interface (CTI).

Features

- Compatible with IEEE 802.3/Ethernet II/Thin Ethernet/ StarLAN
- Interfaces with 8-, 16- and 32-bit microprocessor systems
- Implements simple, versatile buffer management
- Forms integral part of DP8390C, 91, 92 Ethernet/Thin Ethernet solution
- Requires single 5V supply
- Utilizes low power microCMOS process
- Includes
  - Two 16-bit DMA channels
  - 16-byte internal FIFO with programmable threshold
  - Network statistics storage
- Supports physical, multicast, and broadcast address filtering
- Provides 3 levels of loopback
- Utilizes independent system and network clocks

Table of Contents

1.0 System Diagram

IEEE 802.3 Compatible Ethernet/Thin Ethernet Local Area Network Chip Set
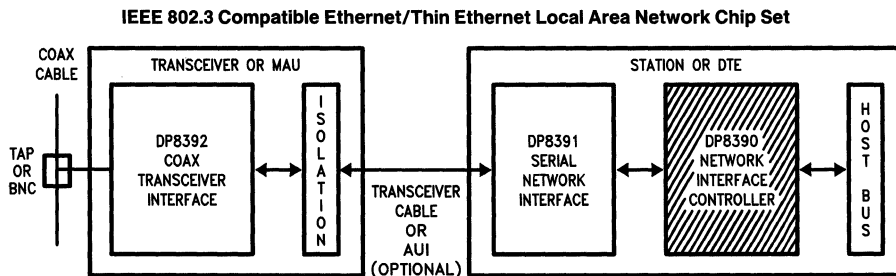
TL/F/8582-1

2.0 Block Diagram
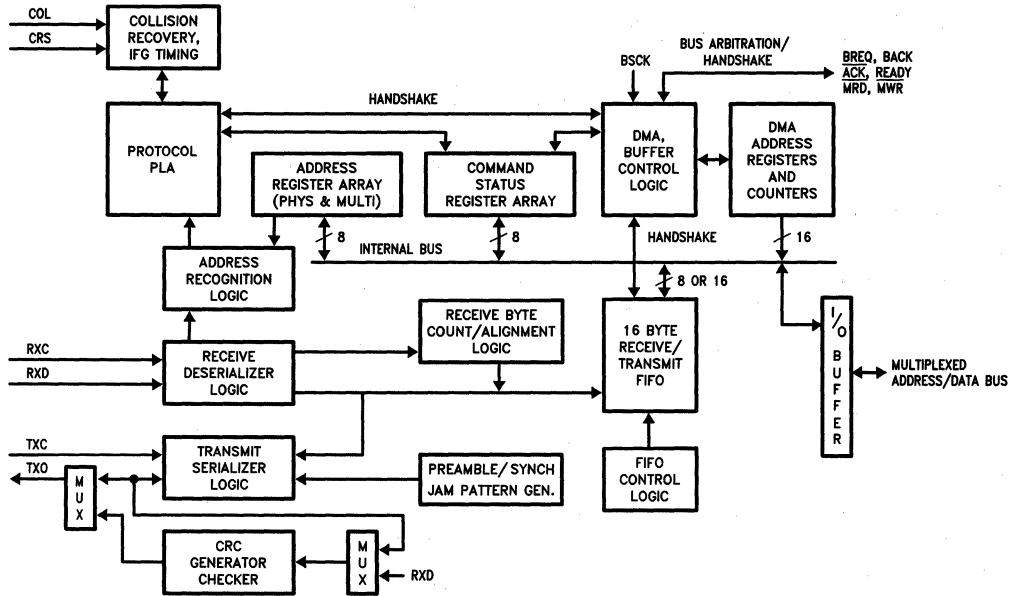
FIGURE 1

TL/F/8582–2

3.0 Functional Description

(Refer to *Figure 1*)

RECEIVE DESERIALIZER

The Receive Deserializer is activated when the input signal Carrier Sense is asserted to allow incoming bits to be shifted into the shift register by the receive clock. The serial receive data is also routed to the CRC generator/checker. The Receive Deserializer includes a synch detector which detects the SFD (Start of Frame Delimiter) to establish where byte boundaries within the serial bit stream are located. After every eight receive clocks, the byte wide data is transferred to the 16-byte FIFO and the Receive Byte Count is incremented. The first six bytes after the SFD are checked for valid comparison by the Address Recognition Logic. If the Address Recognition Logic does not recognize the packet, the FIFO is cleared.

CRC GENERATOR/CHECKER

During transmission, the CRC logic generates a local CRC field for the transmitted bit sequence. The CRC encodes all fields after the synch byte. The CRC is shifted out MSB first following the last transmit byte. During reception the CRC logic generates a CRC field from the incoming packet. This local CRC is serially compared to the incoming CRC appended to the end of the packet by the transmitting node. If the local and received CRC match, a specific pattern will be generated and decoded to indicate no data errors. Transmission errors result in a different pattern and are detected, resulting in rejection of a packet.

TRANSMIT SERIALIZER

The Transmit Serializer reads parallel data from the FIFO and serializes it for transmission. The serializer is clocked by

the transmit clock generated by the Serial Network Interface (DP8391). The serial data is also shifted into the CRC generator/checker. At the beginning of each transmission, the Preamble and Synch Generator append 62 bits of 1,0 preamble and a 1,1 synch pattern. After the last data byte of the packet has been serialized the 32-bit FCS field is shifted directly out of the CRC generator. In the event of a collision the Preamble and Synch generator is used to generate a 32-bit JAM pattern of all 1's.

ADDRESS RECOGNITION LOGIC

The address recognition logic compares the Destination Address Field (first 6 bytes of the received packet) to the Physical address registers stored in the Address Register Array. If any one of the six bytes does not match the pre-programmed physical address, the Protocol Control Logic rejects the packet. All multicast destination addresses are filtered using a hashing technique. (See register description.) If the multicast address indexes a bit that has been set in the filter bit array of the Multicast Address Register Array the packet is accepted, otherwise it is rejected by the Protocol Control Logic. Each destination address is also checked for all 1's which is the reserved broadcast address.

FIFO AND FIFO CONTROL LOGIC

The NIC features a 16-byte FIFO. During transmission the DMA writes data into the FIFO and the Transmit Serializer reads data from the FIFO and transmits it. During reception the Receive Deserializer writes data into the FIFO and the DMA reads data from the FIFO. The FIFO control logic is used to count the number of bytes in the FIFO so that after a preset level, the DMA can begin a bus access and write/read data to/from the FIFO before a FIFO underflow//overflow occurs.

3.0 Functional Description (Continued)

Because the NIC must buffer the Address field of each incoming packet to determine whether the packet matches its Physical Address Registers or maps to one of its Multicast Registers, the first local DMA transfer does not occur until 8 bytes have accumulated in the FIFO.

To assure that there is no overwriting of data in the FIFO, the FIFO logic flags a FIFO overrun as the 13th byte is written into the FIFO; this effectively shortens the FIFO to 13 bytes. In addition, the FIFO logic operates differently in Byte Mode than in Word Mode. In Byte Mode, a threshold is indicated when the n + 1 byte has entered the FIFO; thus, with an 8-byte threshold, the NIC issues Bus Request (BREQ) when the 9th byte has entered the FIFO. For Word Mode, BREQ is not generated until the n + 2 bytes have entered the FIFO. Thus, with a 4 word threshold (equivalent to an 8-byte threshold), BREQ is issued when the 10th byte has entered the FIFO.

PROTOCOL PLA

The protocol PLA is responsible for implementing the IEEE 802.3 protocol, including collision recovery with random backoff. The Protocol PLA also formats packets during transmission and strips preamble and synch during reception.

DMA AND BUFFER CONTROL LOGIC

The DMA and Buffer Control Logic is used to control two 16-bit DMA channels. During reception, the Local DMA stores packets in a receive buffer ring, located in buffer memory. During transmission the Local DMA uses programmed pointer and length registers to transfer a packet from local buffer memory to the FIFO. A second DMA channel is used as a slave DMA to transfer data between the local buffer memory and the host system. The Local DMA and Remote DMA are internally arbitrated, with the Local DMA channel having highest priority. Both DMA channels use a common external bus clock to generate all required bus timing. External arbitration is performed with a standard bus request, bus acknowledge handshake protocol.

4.0 Transmit/Receive Packet Encapsulation/Decapsulation

A standard IEEE 802.3 packet consists of the following fields: preamble, Start of Frame Delimiter (SFD), destination address, source address, length, data, and Frame Check Sequence (FCS). The typical format is shown in *Figure 2*. The packets are Manchester encoded and decoded by the DP8391 SNI and transferred serially to the NIC using NRZ data with a clock. All fields are of fixed length except for the data field. The NIC generates and appends the preamble, SFD and FCS field during transmission. The Preamble and SFD fields are stripped during reception. (The CRC is passed through to buffer memory during reception.)

PREAMBLE AND START OF FRAME DELIMITER (SFD)

The Manchester encoded alternating 1,0 preamble field is used by the SNI (DP8391) to acquire bit synchronization with an incoming packet. When transmitted each packet contains 62 bits of alternating 1,0 preamble. Some of this preamble will be lost as the packet travels through the network. The preamble field is stripped by the NIC. Byte alignment is performed with the Start of Frame Delimiter (SFD) pattern which consists of two consecutive 1's. The NIC does not treat the SFD pattern as a byte, it detects only the two bit pattern. This allows any preceding preamble within the SFD to be used for phase locking.

DESTINATION ADDRESS

The destination address indicates the destination of the packet on the network and is used to filter unwanted packets from reaching a node. There are three types of address formats supported by the NIC: physical, multicast, and broadcast. The physical address is a unique address that corresponds only to a single node. All physical addresses have an MSB of "0". These addresses are compared to the internally stored physical address registers. Each bit in the destination address must match in order for the NIC to accept the packet. Multicast addresses begin with an MSB of "1". The DP8390C filters multicast addresses using a standard hashing algorithm that maps all multicast addresses into a 6-bit value. This 6-bit value indexes a 64-bit array that filters the value. If the address consists of all 1's it is a broadcast address, indicating that the packet is intended for all nodes. A promiscuous mode allows reception of all packets: the destination address is not required to match any filters. Physical, broadcast, multicast, and promiscuous address modes can be selected.

SOURCE ADDRESS

The source address is the physical address of the node that sent the packet. Source addresses cannot be multicast or broadcast addresses. This field is simply passed to buffer memory.

LENGTH FIELD

The 2-byte length field indicates the number of bytes that are contained in the data field of the packet. This field is not interpreted by the NIC.

DATA FIELD

The data field consists of anywhere from 46 to 1500 bytes. Messages longer than 1500 bytes need to be broken into multiple packets. Messages shorter than 46 bytes will require appending a pad to bring the data field to the minimum length of 46 bytes. If the data field is padded, the number of valid data bytes is indicated in the length field. **The NIC does not strip or append pad bytes for short packets, or check for oversize packets.**

FCS FIELD

The Frame Check Sequence (FCS) is a 32-bit CRC field calculated and appended to a packet during transmission to allow detection of errors when a packet is received. During reception, error free packets result in a specific pattern in the CRC generator. Packets with improper CRC will be rejected. The AUTODIN II ($X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X^1 + 1$) polynomial is used for the CRC calculations.

PREAMBLE	SFD	DESTINATION	SOURCE	LENGTH	DATA	FCS
62b	2b	6B	6B	2B	46B–1500B	4B

RECEIVE OPERATIONS: STRIPPED BY NIC | TRANSFERRED VIA DMA

TRANSMIT OPERATIONS: APPENDED BY NIC | TRANSFERRED VIA DMA | CALCULATED + APPENDED BY NIC

B = BYTES
b = BITS

TL/F/8582–3
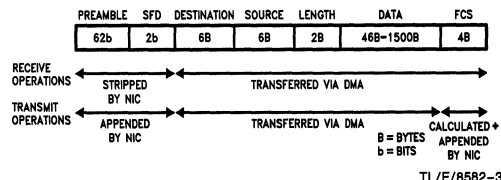
FIGURE 2

Connection Diagrams

Plastic Chip Carrier

PCC
68 PIN

TL/F/8582-5

Dual-In-Line Package

TL/F/8582-4
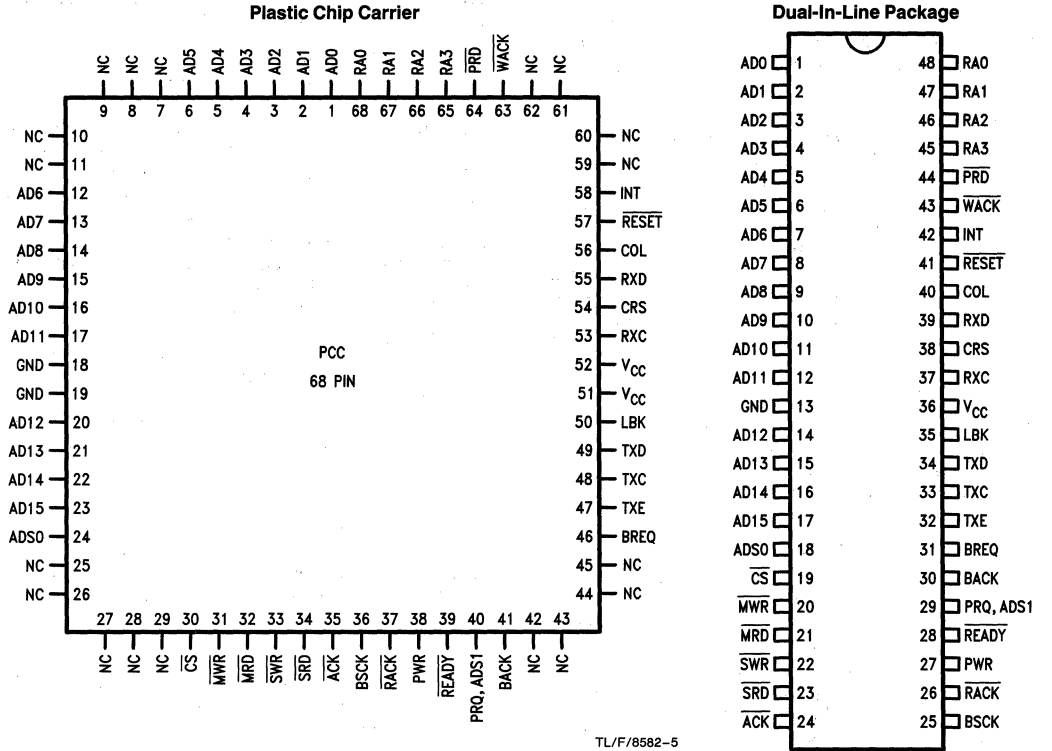
Order Number DP8390CN or DP8390CV
See NS Package Number N48A or V68A

5.0 Pin Descriptions

BUS INTERFACE PINS

Symbol	DIP Pin No	Function	Description
AD0–AD15	1–12 14–17	I/O,Z	**MULTIPLEXED ADDRESS/DATA BUS:** • Register Access, with DMA inactive, $\overline{CS}$ low and $\overline{ACK}$ returned from NIC, pins AD0–AD7 are used to read/write register data. AD8–AD15 float during I/O transfers. $\overline{SRD}$, $\overline{SWR}$ pins are used to select direction of transfer. • Bus Master with BACK input asserted. During t1 of memory cycle AD0–AD15 contain address. During t2, t3, t4 AD0–AD15 contain data (word transfer mode). During t2, t3, t4 AD0–AD7 contain data, AD8–AD15 contain address (byte transfer mode). Direction of transfer is indicated by NIC on $\overline{MWR}$, $\overline{MRD}$ lines.
ADS0	18	I/O,Z	**ADDRESS STROBE 0** • Input with DMA inactive and $\overline{CS}$ low, latches RA0–RA3 inputs on falling edge. If high, data present on RA0–RA3 will flow through latch. • Output when Bus Master, latches address bits (A0–A15) to external memory during DMA transfers.

5.0 Pin Descriptions (Continued)

BUS INTERFACE PINS (Continued)

Symbol	DIP Pin No	Function	Description
$\overline{\text{CS}}$	19	I	**CHIP SELECT:** Chip Select places controller in slave mode for μP access to internal registers. Must be valid through data portion of bus cycle. RA0–RA3 are used to select the internal register. $\overline{\text{SWR}}$ and $\overline{\text{SRD}}$ select direction of data transfer.
$\overline{\text{MWR}}$	20	O,Z	**MASTER WRITE STROBE:** Strobe for DMA transfers, active low during write cycles (t2, t3, tw) to buffer memory. Rising edge coincides with the presence of valid output data. TRI-STATE® until BACK asserted.
$\overline{\text{MRD}}$	21	O,Z	**MASTER READ STROBE:** Strobe for DMA transfers, active during read cycles (t2, t3, tw) to buffer memory. Input data must be valid on rising edge of $\overline{\text{MRD}}$. TRI-STATE until BACK asserted.
$\overline{\text{SWR}}$	22	I	**SLAVE WRITE STROBE:** Strobe from CPU to write an internal register selected by RA0–RA3.
$\overline{\text{SRD}}$	23	I	**SLAVE READ STROBE:** Strobe from CPU to read an internal register selected by RA0–RA3.
$\overline{\text{ACK}}$	24	O	**ACKNOWLEDGE:** Active low when NIC grants access to CPU. Used to insert WAIT states to CPU until NIC is synchronized for a register read or write operation.
RA0–RA3	45–48	I	**REGISTER ADDRESS:** These four pins are used to select a register to be read or written. The state of these inputs is ignored when the NIC is not in slave mode ($\overline{\text{CS}}$ high).
$\overline{\text{PRD}}$	44	O	**PORT READ:** Enables data from external latch onto local bus during a memory write cycle to local memory (remote write operation). This allows asynchronous transfer of data from the system memory to local memory.
$\overline{\text{WACK}}$	43	I	**WRITE ACKNOWLEDGE:** Issued from system to NIC to indicate that data has been written to the external latch. The NIC will begin a write cycle to place the data in local memory.
INT	42	O	**INTERRUPT:** Indicates that the NIC requires CPU attention after reception transmission or completion of DMA transfers. The interrupt is cleared by writing to the ISR. All interrupts are maskable.
$\overline{\text{RESET}}$	41	I	**RESET:** Reset is active low and places the NIC in a reset mode immediately, no packets are transmitted or received by the NIC until STA bit is set. Affects Command Register, Interrupt Mask Register, Data Configuration Register and Transmit Configuration Register. The NIC will execute reset within 10 BUSK cycles.
BREQ	31	O	**BUS REQUEST:** Bus Request is an active high signal used to request the bus for DMA transfers. This signal is automatically generated when the FIFO needs servicing.
BACK	30	I	**BUS ACKNOWLEDGE:** Bus Acknowledge is an active high signal indicating that the CPU has granted the bus to the NIC. If immediate bus access is desired, BREQ should be tied to BACK. **Tying BACK to V_{CC} will result in a deadlock.**
PRQ, ADS1	29	O,Z	**PORT REQUEST/ADDRESS STROBE 1** • 32-BIT MODE: If LAS is set in the Data Configuration Register, this line is programmed as ADS1. It is used to strobe addresses A16–A31 into external latches. (A16–A31 are the fixed addresses stored in RSAR0, RSAR1.) ADS1 will remain at TRI-STATE until BACK is received. • 16-BIT MODE: If LAS is not set in the Data Configuration Register, this line is programmed as PRQ and is used for Remote DMA Transfers. In this mode PRQ will be a standard logic output. **NOTE: This line will power up as TRI-STATE until the Data Configuration Register is programmed.**
$\overline{\text{READY}}$	28	I	**READY:** This pin is set high to insert wait states during a DMA transfer. The NIC will sample this signal at t3 during DMA transfers.

5.0 Pin Descriptions (Continued)

BUS INTERFACE PINS (Continued)

Symbol	DIP Pin No	Function	Description
$\overline{PWR}$	27	O	**PORT WRITE:** Strobe used to latch data from the NIC into external latch for transfer to host memory during Remote Read transfers. The rising edge of $\overline{PWR}$ coincides with the presence of valid data on the local bus.
$\overline{RACK}$	26	I	**READ ACKNOWLEDGE:** Indicates that the system DMA or host CPU has read the data placed in the external latch by the NIC. The NIC will begin a read cycle to update the latch.
BSCK	25	I	This clock is used to establish the period of the DMA memory cycle. Four clock cycles (t1, t2, t3, t4) are used per DMA cycle. DMA transfers can be extended by one BSCK increments using the $\overline{READY}$ input.

NETWORK INTERFACE PINS

Symbol	DIP Pin No	Function	Description
COL	40	I	**COLLISION DETECT:** This line becomes active when a collision has been detected on the coaxial cable. During transmission this line is monitored after preamble and synch have been transmitted. At the end of each transmission this line is monitored for CD heartbeat.
RXD	39	I	**RECEIVE DATA:** Serial NRZ data received from the ENDEC, clocked into the NIC on the rising edge of RXC.
CRS	38	I	**CARRIER SENSE:** This signal is provided by the ENDEC and indicates that carrier is present. This signal is active high.
RXC	37	I	**RECEIVE CLOCK:** Re-synchronized clock from the ENDEC used to clock data from the ENDEC into the NIC.
LBK	35	O	**LOOPBACK:** This output is set high when the NIC is programmed to perform a loopback through the StarLAN ENDEC.
TXD	34	O	**TRANSMIT DATA:** Serial NRZ Data output to the ENDEC. The data is valid on the rising edge of TXC.
TXC	33	I	**TRANSMIT CLOCK:** This clock is used to provide timing for internal operation and to shift bits out of the transmit serializer. TXC is nominally a 1 MHz clock provided by the ENDEC.
TXE	32	O	**TRANSMIT ENABLE:** This output becomes active when the first bit of the packet is valid on TXD and goes low after the last bit of the packet is clocked out of TXD. This signal connects directly to the ENDEC. This signal is active high.

POWER

Symbol	DIP Pin No	Function	Description
V_{CC}	36		+5V DC is required. It is suggested that a decoupling capacitor be connected between these pins. It is essential to provide a path to ground for the GND pin with the lowest possible impedance.
GND	13		

6.0 Direct Memory Access Control (DMA)

The DMA capabilities of the NIC greatly simplify use of the DP8390C in typical configurations. The local DMA channel transfers data between the FIFO and memory. On transmission, the packet is DMA'd from memory to the FIFO in bursts. Should a collision occur (up to 15 times), the packet is retransmitted with no processor intervention. On reception, packets are DMAed from the FIFO to the receive buffer ring (as explained below).

A remote DMA channel is also provided on the NIC to accomplish transfers between a buffer memory and system memory. The two DMA channels can alternatively be combined to form a single 32-bit address with 8- or 16-bit data.

DUAL DMA CONFIGURATION

An example configuration using both the local and remote DMA channels is shown below. Network activity is isolated on a local bus, where the NIC's local DMA channel performs burst transfers between the buffer memory and the NIC's FIFO. The Remote DMA transfers data between the buffer memory and the host memory via a bidirectional I/O port. The Remote DMA provides local addressing capability and is used as a slave DMA by the host. Host side addressing must be provided by a host DMA or the CPU. The NIC allows Local and Remote DMA operations to be interleaved.

SINGLE CHANNEL DMA OPERATION

If desirable, the two DMA channels can be combined to provide a 32-bit DMA address. The upper 16 bits of the 32-bit address are static and are used to point to a 64k byte (or 32k word) page of memory where packets are to be received and transmitted.

6.0 Direct Memory Access Control (DMA) (Continued)

Dual Bus System
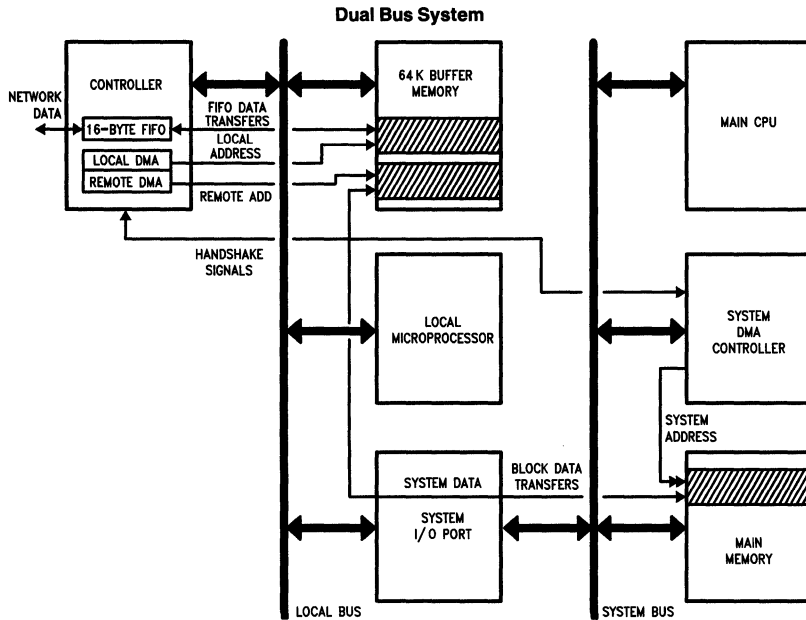
TL/F/8582–55

32-Bit DMA Operation
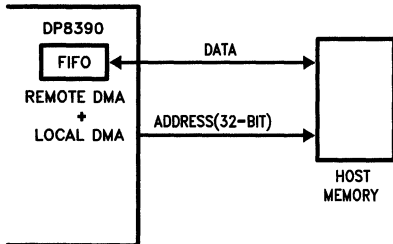
TL/F/8582–6

7.0 Packet Reception

The Local DMA receive channel uses a Buffer Ring Structure comprised of a series of contiguous fixed length 256 byte (128 word) buffers for storage of received packets. The location of the Receive Buffer Ring is programmed in two registers, a Page Start and a Page Stop Register. Ethernet packets consist of a distribution of shorter link control packets and longer data packets, the 256 byte buffer length provides a good compromise between short packets and longer packets to most efficiently use memory. In addition these buffers provide memory resources for storage of back-to-back packets in loaded networks. The assignment of buffers

NIC Receive Buffer Ring

TL/F/8582–7

1-9
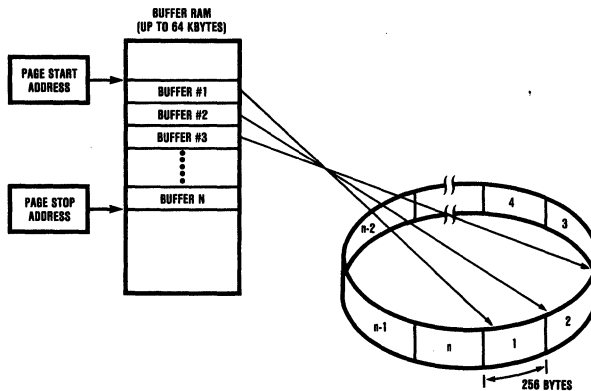
7.0 Packet Reception (Continued)

for storing packets is controlled by Buffer Management Logic in the NIC. The Buffer Management Logic provides three basic functions: linking receive buffers for long packets, recovery of buffers when a packet is rejected, and recirculation of buffer pages that have been read by the host.

At initialization, a portion of the 64k byte (or 32k word) address space is reserved for the receive buffer ring. Two eight bit registers, the Page Start Address Register (PSTART) and the Page Stop Address Register (PSTOP) define the physical boundaries of where the buffers reside. The NIC treats the list of buffers as a logical ring; whenever the DMA address reaches the Page Stop Address, the DMA is reset to the Page Start Address.

INITIALIZATION OF THE BUFFER RING

Two static registers and two working registers control the operation of the Buffer Ring. These are the Page Start Register, Page Stop Register (both described previously), the Current Page Register and the Boundary Pointer Register. The Current Page Register points to the first buffer used to store a packet and is used to restore the DMA for writing status to the Buffer Ring or for restoring the DMA address in the event of a Runt packet, a CRC, or Frame Alignment error. The Boundary Register points to the first packet in the Ring not yet read by the host. If the local DMA address ever reaches the Boundary, reception is aborted. The Boundary Pointer is also used to initialize the Remote DMA for removing a packet and is advanced when a packet is removed. A simple analogy to remember the function of these registers is that the Current Page Register acts as a Write Pointer and the Boundary Pointer acts as a Read Pointer.

Note 1: At initialization, the Page Start Register value should be loaded into both the Current Page Register and the Boundary Pointer Register.

Note 2: The Page Start Register must not be initialized to 00H.

Receive Buffer Ring At Initialization
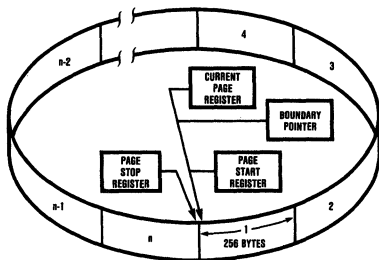
TL/F/8582–30

BEGINNING OF RECEPTION

When the first packet begins arriving the NIC begins storing the packet at the location pointed to by the Current Page Register. An offset of 4 bytes is saved in this first buffer to allow room for storing receive status corresponding to this packet.
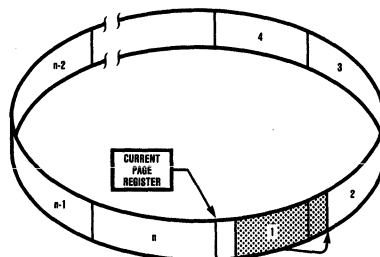
Received Packet Enters Buffer Pages
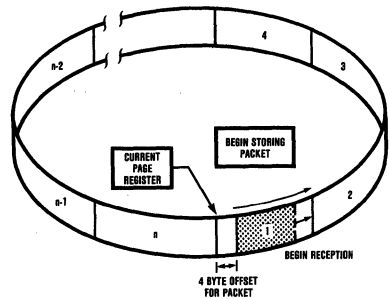
TL/F/8582–31

LINKING RECEIVE BUFFER PAGES

If the length of the packet exhausts the first 256 byte buffer, the DMA performs a forward link to the next buffer to store the remainder of the packet. For a maximal length packet the buffer logic will link six buffers to store the entire packet. Buffers cannot be skipped when linking, a packet will always be stored in contiguous buffers. Before the next buffer can be linked, the Buffer Management Logic performs two comparisons. The first comparison tests for equality between the DMA address of the next buffer and the contents of the Page Stop Register. If the buffer address equals the Page Stop Register, the buffer management logic will restore the DMA to the first buffer in the Receive Buffer Ring value programmed in the Page Start Address Register. The second comparison tests for equality between the DMA address of the next buffer address and the contents of the Boundary Pointer Register. If the two values are equal the reception is aborted. The Boundary Pointer Register can be used to protect against overwriting any area in the receive buffer ring that has not yet been read. When linking buffers, buffer management will never cross this pointer, effectively avoiding any overwrites. If the buffer address does not match either the Boundary Pointer or Page Stop Address, the link to the next buffer is performed.

Linking Buffers

Before the DMA can enter the next contiguous 256 byte buffer, the address is checked for equality to PSTOP and to the Boundary Pointer. If neither are reached, the DMA is allowed to use the next buffer.

Linking Receive Buffer Pages

1) Check for = to PSTOP
2) Check for = to Boundary

TL/F/8582–32

7.0 Packet Reception (Continued)

Received Packet Aborted if It Hits Boundary Pointer
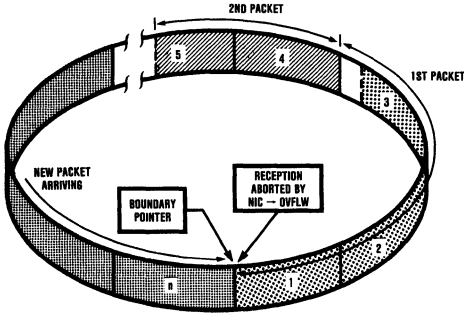
TL/F/8582–8

Buffer Ring Overflow

If the Buffer Ring has been filled and the DMA reaches the Boundary Pointer Address, reception of the incoming packet will be aborted by the NIC. Thus, the packets previously received and still contained in the Ring will not be destroyed.

In a heavily loaded network environment the local DMA may be disabled, preventing the NIC from buffering packets from the network. To guarantee this will not happen, a software reset must be issued during all Receive Buffer Ring overflows (indicated by the OVW bit in the Interrupt Status Register). **The following procedure is required to recover from a Receiver Buffer Ring Overflow.**

1. Issue the STOP mode command (Command Register = 21H). The NIC may not immediately enter the STOP mode. If it is currently processing a packet, the NIC will enter STOP mode only after finishing the packet. The NIC indicates that it has entered STOP mode by setting the RST bit in the Interrupt Status Register.

2. Clear the Remote Byte Counter Registers (RBCR0, RBCR1). The NIC requires these registers to be cleared before it sets the RST bit.

   Note: If the STP is set when a transmission is in progress, the RST bit may not be set. In this case, the NIC is guaranteed to be reset after the longest packet time (1500 bytes = 1.2 ms). For the DP8390C (but not for the DP8390B), the NIC will be reset within 2 microseconds after the STP bit is set and Loopback mode 1 is programmed.

3. Poll the Interrupt Status Register for the RST bit. When set, the NIC is in STOP mode.

4. Place the NIC in LOOPBACK (mode 1 or 2) by writing 02H or 04H to the Transmit Configuration Register. This step is required to properly enable the NIC onto an active network.

5. Issue the START mode command (Command Register = 22H). The local receive DMA is still inactive since the NIC is in LOOPBACK.

6. Remove at least one packet from the Receive Buffer Ring to accommodate additional incoming packets.

7. Take the NIC out of LOOPBACK by programming the Transmit Configuration Register back to its original value and resume normal operation.

   Note: If the Remote DMA channel is not used, you may eliminate step 6 and remove packets from the Receive Buffer Ring after step 1. This will reduce or eliminate the polling time incurred in step 3.

END OF PACKET OPERATIONS

At the end of the packet the NIC determines whether the received packet is to be accepted or rejected. It either branches to a routine to store the Buffer Header or to another routine that recovers the buffers used to store the packet.

SUCCESSFUL RECEPTION

If the packet is successfully received as shown, the DMA is restored to the first buffer used to store the packet (pointed to by the Current Page Register). The DMA then stores the Receive Status, a Pointer to where the next packet will be stored (Buffer 4) and the number of received bytes. Note that the remaining bytes in the last buffer are discarded and reception of the next packet begins on the next empty 256-byte buffer boundary. The Current Page Register is then initialized to the next available buffer in the Buffer Ring. (The location of the next buffer had been previously calculated and temporarily stored in an internal scratchpad register.)

Termination of Received Packet—Packet Accepted
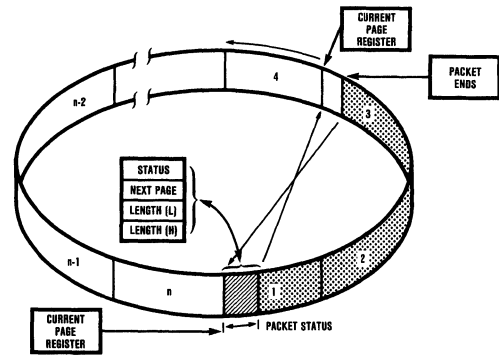
TL/F/8582–10

BUFFER RECOVERY FOR REJECTED PACKETS

If the packet is a runt packet or contains CRC or Frame Alignment errors, it is rejected. The buffer management logic resets the DMA back to the first buffer page used to store the packet (pointed to by CURR), recovering all buffers that had been used to store the rejected packet. This operation will not be performed if the NIC is programmed to accept either runt packets or packets with CRC or Frame Alignment errors. The received CRC is always stored in buffer memory after the last byte of received data for the packet.

Termination of Received Packet—Packet Rejected
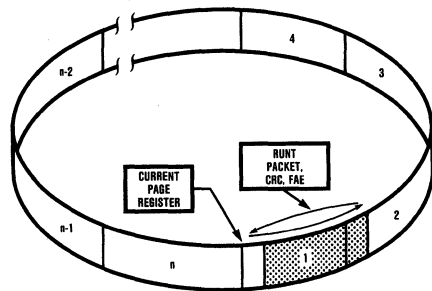
TL/F/8582–13

7.0 Packet Reception (Continued)

Error Recovery

If the packet is rejected as shown, the DMA is restored by the NIC by reprogramming the DMA starting address pointed to by the Current Page Register.

REMOVING PACKETS FROM THE RING

Packets are removed from the ring using the Remote DMA or an external device. When using the Remote DMA the Send Packet command can be used. This programs the Remote DMA to automatically remove the received packet pointed to by the Boundary Pointer. At the end of the transfer, the NIC moves the Boundary Pointer, freeing additional buffers for reception. The Boundary Pointer can also be moved manually by programming the Boundary Register. Care should be taken to keep the Boundary Pointer at least one buffer behind the Current Page Pointer.

The following is a suggested method for maintaining the Receive Buffer Ring pointers.

1. At initialization, set up a software variable (next_pkt) to indicate where the next packet will be read. At the beginning of each Remote Read DMA operation, the value of next_pkt will be loaded into RSAR0 and RSAR1.

2. When initializing the NIC set:
   BNDRY = PSTART
   CURR = PSTART + 1
   next_pkt = PSTART + 1

3. After a packet is DMAed from the Receive Buffer Ring, the Next Page Pointer (second byte in NIC buffer header) is used to update BNDRY and next_pkt.

   next_pkt = Next Page Pointer
   BNDRY = Next Page Pointer − 1
   If BNDRY < PSTART then BNDRY = PSTOP − 1

Note the size of the Receive Buffer Ring is reduced by one 256-byte buffer; this will not, however, impede the operation of the NIC.

In StarLAN applications using bus clock frequencies greater than 4 MHz, the NIC does not update the buffer header information properly because of the disparity between the network and bus clock speeds. The lower byte count is copied twice into the third and fourth locations of the buffer header and the upper byte count is not written. The upper byte count, however, can be calculated from the current next page pointer (second byte in the buffer header) and the previous next page pointer (stored in memory by the CPU). The following routine calculates the upper byte count and allows StarLAN applications to be insensitive to bus clock speeds. Next_pkt is defined similarly as above.

1st Received Packet Removed By Remote DMA
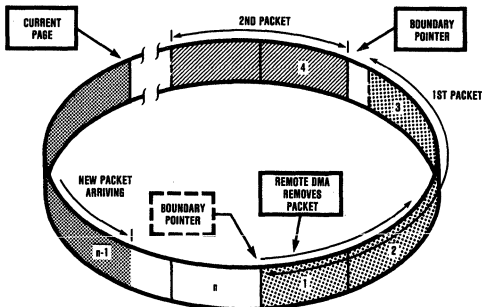
TL/F/8582–57

upper byte count = next page pointer − next_pkt − 1
if (upper byte count) < 0 then
upper byte count = (PSTOP − next_pkt) +
                   (next page pointer − PSTART) − 1
if (lower byte count) > 0 fch then
upper byte count = upper byte count + 1

STORAGE FORMAT FOR RECEIVED PACKETS

The following diagrams describe the format for how received packets are placed into memory by the local DMA channel. These modes are selected in the Data Configuration Register.

Storage Format

AD15 AD8	AD7 AD0
Next Packet Pointer	Receive Status
Receive Byte Count 1	Receive Byte Count 0
Byte 2	Byte 1

BOS = 0, WTS = 1 in Data Configuration Register.

This format used with Series 32000 808X type processors.

AD15 AD8	AD7 AD0
Next Packet Pointer	Receive Status
Receive Byte Count 0	Receive Byte Count 1
Byte 1	Byte 2

BOS = 1, WTS = 1 in Data Configuration Register.

This format used with 68000 type processors.

Note: The Receive Byte Count ordering remains the same for BOS=0 or 1.

AD7 AD0
Receive Status
Next Packet Pointer
Receive Byte Count 0
Receive Byte Count 1
Byte 0
Byte 1

BOS = 0, WTS = 0 in Data Configuration Register.

This format used with general 8-bit CPUs.

8.0 Packet Transmission

The Local DMA is also used during transmission of a packet. Three registers control the DMA transfer during transmission, a Transmit Page Start Address Register (TPSR) and the Transmit Byte Count Registers (TBCR0,1). When the NIC receives a command to transmit the packet pointed to by these registers, buffer memory data will be moved into the FIFO as required during transmission. The NIC will generate and append the preamble, synch and CRC fields.

8.0 Packet Transmission (Continued)

TRANSMIT PACKET ASSEMBLY

The NIC requires a contiguous assembled packet with the format shown. The transmit byte count includes the Destination Address, Source Address, Length Field and Data. It does not include preamble and CRC. When transmitting data smaller than 46 bytes, the packet must be padded to a minimum size of 64 bytes. The programmer is responsible for adding and stripping pad bytes.

General Transmit Packet Format

DESTINATION ADDRESS	6 BYTES
SOURCE ADDRESS	6 BYTES
TYPE LENGTH	2 BYTES
DATA	$\geq$ 46 BYTES
PAD (IF DATA < 46 BYTES)	

TX BYTE COUNT (TBCR0,1)
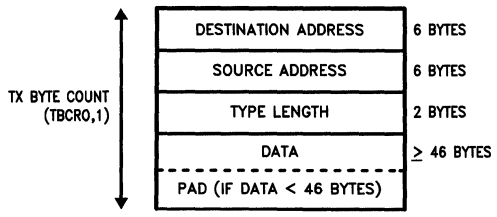
TL/F/8582–58

TRANSMISSION

Prior to transmission, the TPSR (Transmit Page Start Register) and TBCR0, TBCR1 (Transmit Byte Count Registers) must be initialized. To initiate transmission of the packet the TXP bit in the Command Register is set. The Transmit Status Register (TSR) is cleared and the NIC begins to prefetch transmit data from memory (unless the NIC is currently receiving). If the interframe gap has timed out the NIC will begin transmission.

CONDITIONS REQUIRED TO BEGIN TRANSMISSION

In order to transmit a packet, the following three conditions must be met:

1. The Interframe Gap Timer has timed out the first 6.4 μs of the Interframe Gap (See appendix for Interframe Gap Flowchart)

2. At least one byte has entered the FIFO. (This indicates that the burst transfer has been started)

3. If the NIC had collided, the backoff timer has expired.

In typical systems the NIC has already prefetched the first burst of bytes before the 6.4 μs timer expires. The time during which NIC transmits preamble can also be used to load the FIFO.

Note: If carrier sense is asserted before a byte has been loaded into the FIFO, the NIC will become a receiver.

COLLISION RECOVERY

During transmission, the Buffer Management logic monitors the transmit circuitry to determine if a collision has occurred. If a collision is detected, the Buffer Management logic will reset the FIFO and restore the Transmit DMA pointers for retransmission of the packet. The COL bit will be set in the TSR and the NCR (Number of Collisions Register) will be incremented. If 15 retransmissions each result in a collision the transmission will be aborted and the ABT bit in the TSR will be set.

Note: NCR reads as zeroes if excessive collisions are encountered.

TRANSMIT PACKET ASSEMBLY FORMAT

The following diagrams describe the format for how packets must be assembled prior to transmission for different byte ordering schemes. The various formats are selected in the Data Configuration Register.

D15	D8 D7	D0
DA1		DA0
DA3		DA2
DA5		DA4
SA1		DA0
SA3		DA2
SA5		DA4
T/L1		T/L0
DATA 1		DATA 0

BOS = 0, WTS = 1 in Data Configuration Register.

This format is used with Series 32000, 808X type processors.

D15	D8 D7	D0
DA0		DA1
DA2		DA3
DA4		DA5
SA0		SA1
SA2		SA3
SA4		SA5
T/L0		T/L1
DATA 0		DATA 1

BOS = 1, WTS = 1 in Data Configuration Register.

This format is used with 68000 type processors.

D7	D0
DA0	
DA1	
DA2	
DA3	
DA4	
DA5	
SA0	
SA1	
SA2	
SA3	

BOS = 0, WTS = 0 in Data Configuration Register.

This format is used with general 8-bit CPUs.

Note: All examples above will result in a transmission of a packet in order of DA0, DA1, DA2, DA3 . . . bits within each byte will be transmitted least significant bit first.

DA = Destination Address

SA = Source Address

T/L = Type/Length Field

9.0 Remote DMA

The Remote DMA channel is used to both assemble packets for transmission, and to remove received packets from the Receive Buffer Ring. It may also be used as a general purpose slave DMA channel for moving blocks of data or commands between host memory and local buffer memory. There are three modes of operation, Remote Write, Remote Read, or Send Packet.

Two register pairs are used to control the Remote DMA, a Remote Start Address (RSAR0, RSAR1) and a Remote Byte Count (RBCR0, RBCR1) register pair. The Start Address Register pair points to the beginning of the block to be moved while the Byte Count Register pair is used to indicate the number of bytes to be transferred. Full handshake logic is provided to move data between local buffer memory and a bidirectional I/O port.

REMOTE WRITE

A Remote Write transfer is used to move a block of data from the host into local buffer memory. The Remote DMA will read data from the I/O port and sequentially write it to local buffer memory beginning at the Remote Start Address. The DMA Address will be incremented and the Byte Counter will be decremented after each transfer. The DMA is terminated when the Remote Byte Count Register reaches a count of zero.

REMOTE READ

A Remote Read transfer is used to move a block of data from local buffer memory to the host. The Remote DMA will sequentially read data from the local buffer memory, beginning at the Remote Start Address, and write data to the I/O port. The DMA Address will be incremented and the Byte Counter will be decremented after each transfer. The DMA is terminated when the Remote Byte Count Register reaches zero.

SEND PACKET COMMAND

The Remote DMA channel can be automatically initialized to transfer a single packet from the Receive Buffer Ring. The CPU begins this transfer by issuing a "Send Packet" Command. The DMA will be initialized to the value of the Boundary Pointer Register and the Remote Byte Count Register pair (RBCR0, RBCR1) will be initialized to the value of the Receive Byte Count fields found in the Buffer Header of each packet. After the data is transferred, the Boundary Pointer is advanced to allow the buffers to be used for new receive packets. The Remote Read will terminate when the Byte Count equals zero. The Remote DMA is then prepared to read the next packet from the Receive Buffer Ring. If the DMA pointer crosses the Page Stop Register, it is reset to the Page Start Address. This allows the Remote DMA to remove packets that have wrapped around to the top of the Receive Buffer Ring.

Note 1: In order for the NIC to correctly execute the Send Packet Command, the upper Remote Byte Count Register (RBCR1) must first be loaded with 0FH.

Note 2: The Send Packet command cannot be used with 68000 type processors.

10.0 Internal Registers

All registers are 8-bit wide and mapped into two pages which are selected in the Command Register (PS0, PS1). Pins RA0–RA3 are used to address registers within each page. Page 0 registers are those registers which are commonly accessed during NIC operation while page 1 registers are used primarily for initialization. The registers are partitioned to avoid having to perform two write/read cycles to access commonly used registers.

Remote DMA Autoinitialization from Buffer Ring
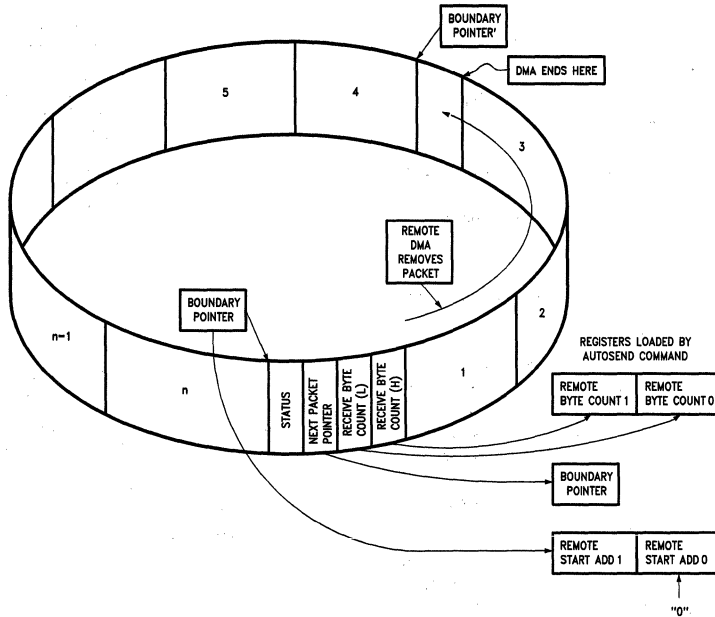
TL/F/8582–59

10.0 Internal Registers (Continued)

10.1 REGISTER ADDRESS MAPPING
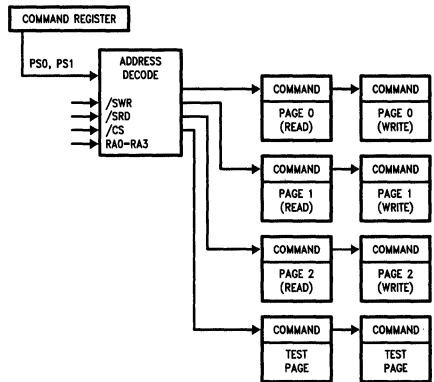
TL/F/8582-60

10.2 REGISTER ADDRESS ASSIGNMENTS

Page 0 Address Assignments (PS1 = 0, PS0 = 0)

RA0–RA3	RD	WR
00H	Command (CR)	Command (CR)
01H	Current Local DMA Address 0 (CLDA0)	Page Start Register (PSTART)
02H	Current Local DMA Address 1 (CLDA1)	Page Stop Register (PSTOP)
03H	Boundary Pointer (BNRY)	Boundary Pointer (BNRY)
04H	Transmit Status Register (TSR)	Transmit Page Start Address (TPSR)
05H	Number of Collisions Register (NCR)	Transmit Byte Count Register 0 (TBCR0)
06H	FIFO (FIFO)	Transmit Byte Count Register 1 (TBCR1)
07H	Interrupt Status Register (ISR)	Interrupt Status Register (ISR)
08H	Current Remote DMA Address 0 (CRDA0)	Remote Start Address Register 0 (RSAR0)
09H	Current Remote DMA Address 1 (CRDA1)	Remote Start Address Register 1 (RSAR1)
0AH	Reserved	Remote Byte Count Register 0 (RBCR0)
0BH	Reserved	Remote Byte Count Register 1 (RBCR1)
0CH	Receive Status Register (RSR)	Receive Configuration Register (RCR)
0DH	Tally Counter 0 (Frame Alignment Errors) (CNTR0)	Transmit Configuration Register (TCR)
0EH	Tally Counter 1 (CRC Errors) (CNTR1)	Data Configuration Register (DCR)
0FH	Tally Counter 2 (Missed Packet Errors) (CNTR2)	Interrupt Mask Register (IMR)

Page 1 Address Assignments (PS1 = 0, PS0 = 1)

RA0–RA3	RD	WR
00H	Command (CR)	Command (CR)
01H	Physical Address Register 0 (PAR0)	Physical Address Register 0 (PAR0)
02H	Physical Address Register 1 (PAR1)	Physical Address Register 1 (PAR1)
03H	Physical Address Register 2 (PAR2)	Physical Address Register 2 (PAR2)
04H	Physical Address Register 3 (PAR3)	Physical Address Register 3 (PAR3)
05H	Physical Address Register 4 (PAR4)	Physical Address Register 4 (PAR4)
06H	Physical Address Register 5 (PAR5)	Physical Address Register 5 (PAR5)
07H	Current Page Register (CURR)	Current Page Register (CURR)
08H	Multicast Address Register 0 (MAR0)	Multicast Address Register 0 (MAR0)
09H	Multicast Address Register 1 (MAR1)	Multicast Address Register 1 (MAR1)
0AH	Multicast Address Register 2 (MAR2)	Multicast Address Register 2 (MAR2)
0BH	Multicast Address Register 3 (MAR3)	Multicast Address Register 3 (MAR3)
0CH	Multicast Address Register 4 (MAR4)	Multicast Address Register 4 (MAR4)
0DH	Multicast Address Register 5 (MAR5)	Multicast Address Register 5 (MAR5)
0EH	Multicast Address Register 6 (MAR6)	Multicast Address Register 6 (MAR6)
0FH	Multicast Address Register 7 (MAR7)	Multicast Address Register 7 (MAR7)

1

10.0 Internal Registers (Continued)

Page 2 Address Assignments (PS1 = 1, PS0 = 0)

RA0–RA3	RD	WR
00H	Command (CR)	Command (CR)
01H	Page Start Register (PSTART)	Current Local DMA Address 0 (CLDA0)
02H	Page Stop Register (PSTOP)	Current Local DMA Address 1 (CLDA1)
03H	Remote Next Packet Pointer	Remote Next Packet Pointer
04H	Transmit Page Start Address (TPSR)	Reserved
05H	Local Next Packet Pointer	Local Next Packet Pointer
06H	Address Counter (Upper)	Address Counter (Upper)
07H	Address Counter (Lower)	Address Counter (Lower)

RA0–RA3	RD	WR
08H	Reserved	Reserved
09H	Reserved	Reserved
0AH	Reserved	Reserved
0BH	Reserved	Reserved
0CH	Receive Configuration Register (RCR)	Reserved
0DH	Transmit Configuration Register (TCR)	Reserved
0EH	Data Configuration Register (DCR)	Reserved
0FH	Interrupt Mask Register (IMR)	Reserved

Note: Page 2 registers should only be accessed for diagnostic purposes. They should not be modified during normal operation.

Page 3 should never be modified.

10.0 Internal Registers (Continued)

10.3 Register Descriptions

COMMAND REGISTER (CR)   00H (READ/WRITE)

The Command Register is used to initiate transmissions, enable or disable Remote DMA operations and to select register pages. To issue a command the microprocessor sets the corresponding bit(s) (RD2, RD1, RD0, TXP). Further commands may be overlapped, but with the following rules: (1) If a transmit command overlaps with a remote DMA operation, bits RD0, RD1, and RD2 must be maintained for the remote DMA command when setting the TXP bit. Note, if a remote DMA command is re-issued when giving the transmit command, the DMA will complete immediately if the remote byte count register have not been re-initialized. (2) If a remote DMA operation overlaps a transmission, RD0, RD1, and RD2 may be written with the desired values and a "0" written to the TXP bit. Writing a "0" to this bit has no effect. (3) A remote write DMA may not overlap remote read operation or visa versa. Either of these operations must either complete or be aborted before the other operation may start.

Bits PS1, PS0, RD2, and STP may be set any time.

7	6	5	4	3	2	1	0
PS1	PS0	RD2	RD1	RD0	TXP	STA	STP

Bit	Symbol	Description
D0	STP	**STOP:** Software reset command, takes the controller offline, no packets will be received or transmitted. Any reception or transmission in progress will continue to completion before entering the reset state. To exit this state, the STP bit must be reset and the STA bit must be set high. To perform a software reset, this bit should be set high. The software reset has executed only when indicated by the RST bit in the ISR being set to a 1. **STP powers up high.** Note: If the NIC has previously been in start mode and the STP is set, both the STP and STA bits will remain set.
D1	STA	**START:** This bit is used to activate the NIC after either power up, or when the NIC has been placed in a reset mode by software command or error. **STA powers up low.**
D2	TXP	**TRANSMIT PACKET:** This bit must be set to initiate transmission of a packet. TXP is internally reset either after the transmission is completed or aborted. This bit should be set only after the Transmit Byte Count and Transmit Page Start registers have been programmed. Note: Before the transmit command is given, the STA bit must be set and the STP bit reset.
D3, D4, D5	RD0, RD1, RD2	**REMOTE DMA COMMAND:** These three encoded bits control operation of the Remote DMA channel. RD2 can be set to abort any Remote DMA command in progress. The Remote Byte Count Registers should be cleared when a Remote DMA has been aborted. The Remote Start Addresses are not restored to the starting address if the Remote DMA is aborted. RD2 RD1 RD0 0 0 0 Not Allowed 0 0 1 Remote Read 0 1 0 Remote Write (Note 2) 0 1 1 Send Packet 1 X X Abort/Complete Remote DMA (Note 1) Note 1: If a remote DMA operation is aborted and the remote byte count has not decremented to zero, PRQ (pin 29, DIP) will remain high. A read acknowledge (RACK) on a write acknowledge (WACK) will reset PRQ low. Note 2: For proper operation of the Remote Write DMA, there are two steps which must be performed before using the Remote Write DMA. The steps are as follows: i) Write a non-zero value into RBCR0. ii) Set bits RD2, RD1, RD0 to 0, 0, 1. iii) Set RBCR0, 1 and RSAR0, 1 iv) Issue the Remote Write DMA Command (RD2, RD1, RD0 = 0, 1, 0)
D6, D7	PS0, PS1	**PAGE SELECT:** These two encoded bits select which register page is to be accessed with addresses RA0–3. PS1 PS0 0 0 Register Page 0 0 1 Register Page 1 1 0 Register Page 2 1 1 Reserved

10.0 Internal Registers (Continued)

10.3 Register Descriptions (Continued)

INTERRUPT STATUS REGISTER (ISR)   07H (READ/WRITE)

This register is accessed by the host processor to determine the cause of an interrupt. Any interrupt can be masked in the Interrupt Mask Register (IMR). Individual interrupt bits are cleared by writing a "1" into the corresponding bit of the ISR. The INT signal is active as long as any unmasked signal is set, and will not go low until all unmasked bits in this register have been cleared. The ISR must be cleared after power up by writing it with all 1's.

7	6	5	4	3	2	1	0
RST	RDC	CNT	OVW	TXE	RXE	PTX	PRX

Bit	Symbol	Description
D0	PRX	**PACKET RECEIVED:** Indicates packet received with no errors.
D1	PTX	**PACKET TRANSMITTED:** Indicates packet transmitted with no errors.
D2	RXE	**RECEIVE ERROR:** Indicates that a packet was received with one or more of the following errors: —CRC Error —Frame Alignment Error —FIFO Overrun —Missed Packet
D3	TXE	**TRANSMIT ERROR:** Set when packet transmitted with one or more of the following errors: —Excessive Collisions —FIFO Underrun
D4	OVW	**OVERWRITE WARNING:** Set when receive buffer ring storage resources have been exhausted. (Local DMA has reached Boundary Pointer).
D5	CNT	**COUNTER OVERFLOW:** Set when MSB of one or more of the Network Tally Counters has been set.
D6	RDC	**REMOTE DMA COMPLETE:** Set when Remote DMA operation has been completed.
D7	RST	**RESET STATUS:** Set when NIC enters reset state and cleared when a Start Command is issued to the CR. This bit is also set when a Receive Buffer Ring overflow occurs and is cleared when one or more packets have been removed from the ring. Writing to this bit has no effect. **NOTE:** This bit does not generate an interrupt, it is merely a status indicator.

10.0 Internal Registers (Continued)

10.3 Register Descriptions (Continued)

INTERRUPT MASK REGISTER (IMR)   0FH (WRITE)

The Interrupt Mask Register is used to mask interrupts. Each interrupt mask bit corresponds to a bit in the Interrupt Status Register (ISR). If an interrupt mask bit is set an interrupt will be issued whenever the corresponding bit in the ISR is set. If any bit in the IMR is set low, an interrupt will not occur when the bit in the ISR is set. **The IMR powers up all zeroes.**

7	6	5	4	3	2	1	0
—	RDCE	CNTE	OVWE	TXEE	RXEE	PTXE	PRXE

Bit	Symbol	Description
D0	PRXE	**PACKET RECEIVED INTERRUPT ENABLE** 0: Interrupt Disabled 1: Enables Interrupt when packet received.
D1	PTXE	**PACKET TRANSMITTED INTERRUPT ENABLE** 0: Interrupt Disabled 1: Enables Interrupt when packet is transmitted.
D2	RXEE	**RECEIVE ERROR INTERRUPT ENABLE** 0: Interrupt Disabled 1: Enables Interrupt when packet received with error.
D3	TXEE	**TRANSMIT ERROR INTERRUPT ENABLE** 0: Interrupt Disabled 1: Enables Interrupt when packet transmission results in error.
D4	OVWE	**OVERWRITE WARNING INTERRUPT ENABLE** 0: Interrupt Disabled 1: Enables Interrupt when Buffer Management Logic lacks sufficient buffers to store incoming packet.
D5	CNTE	**COUNTER OVERFLOW INTERRUPT ENABLE** 0: Interrupt Disabled 1: Enables Interrupt when MSB of one or more of the Network Statistics counters has been set.
D6	RDCE	**DMA COMPLETE INTERRUPT ENABLE** 0: Interrupt Disabled 1: Enables Interrupt when Remote DMA transfer has been completed.
D7	reserved	reserved

10.0 Internal Registers (Continued)

10.3 Register Descriptions (Continued)

DATA CONFIGURATION REGISTER (DCR) 0EH (WRITE)

This Register is used to program the NIC for 8- or 16-bit memory interface, select byte ordering in 16-bit applications and establish FIFO threshholds. **The DCR must be initialized prior to loading the Remote Byte Count Registers. LAS is set on power up.**

7	6	5	4	3	2	1	0
—	FT1	FT0	ARM	LS	LAS	BOS	WTS

Bit	Symbol	Description
D0	WTS	**WORD TRANSFER SELECT** 0: Selects byte-wide DMA transfers 1: Selects word-wide DMA transfers ; WTS establishes byte or word transfers for both Remote and Local DMA transfers **Note:** When word-wide mode is selected, up to 32k words are addressable; A0 remains low.
D1	BOS	**BYTE ORDER SELECT** 0: MS byte placed on AD15–AD8 and LS byte on AD7–AD0. (32000, 8086) 1: MS byte placed on AD7–AD0 and LS byte on AD15–AD8. (68000) ; Ignored when WTS is low
D2	LAS	**LONG ADDRESS SELECT** 0: Dual 16-bit DMA mode 1: Single 32-bit DMA mode ; When LAS is high, the contents of the Remote DMA registers RSAR0,1 are issued as A16–A31 Power up high.
D3	LS	**LOOPBACK SELECT** 0: Loopback mode selected. Bits D1, D2 of the TCR must also be programmed for Loopback operation. 1: Normal Operation.
D4	AR	**AUTO-INITIALIZE REMOTE** 0: Send Command not executed, all packets removed from Buffer Ring under program control. 1: Send Command executed, Remote DMA auto-initialized to remove packets from Buffer Ring. **Note:** Send Command cannot be used with 68000 type processors.
D5, D6	FT0, FT1	**FIFO THRESHHOLD SELECT:** Encoded FIFO threshhold. Establishes point at which bus is requested when filling or emptying the FIFO. During reception, the FIFO threshold indicates the number of bytes (or words) the FIFO has filled serially from the network before bus request (BREQ) is asserted. **Note:** FIFO threshold setting determines the DMA burst length. RECEIVE THRESHOLDS FT1 FT0 Word Wide Byte Wide 0 0 1 Word 2 Bytes 0 1 2 Words 4 Bytes 1 0 4 Words 8 Bytes 1 1 6 Words 12 Bytes During transmission, the FIFO threshold indicates the numer of bytes (or words) the FIFO has filled from the Local DMA before BREQ is asserted. Thus, the transmission threshold is 16 bytes less the receive threshold.

10.0 Internal Registers (Continued)

10.3 Register Descriptions (Continued)

TRANSMIT CONFIGURATION REGISTER (TCR)   0DH (WRITE)

The transmit configuration establishes the actions of the transmitter section of the NIC during transmission of a packet on the network. **LB1 and LB0 which select loopback mode power up as 0.**

7	6	5	4	3	2	1	0
—	—	—	OFST	ATD	LB1	LB0	CRC

Bit	Symbol	Description
D0	CRC	**INHIBIT CRC** 0: CRC appended by transmitter 1: CRC inhibited by transmitter ; In loopback mode CRC can be enabled or disabled to test the CRC logic.
D1, D2	LB0, LB1	**ENCODED LOOPBACK CONTROL:** These encoded configuration bits set the type of loopback that is to be performed. Note that loopback in mode 2 sets the LPBK pin high, this places the SNI in loopback mode and that D3 of the DCR must be set to zero for loopback operation. LB1 LB0 Mode 0 0 0 Normal Operation (LPBK = 0) Mode 1 0 1 Internal Loopback (LPBK = 0) Mode 2 1 0 External Loopback (LPBK = 1) Mode 3 1 1 External Loopback (LPBK = 0)
D3	ATD	**AUTO TRANSMIT DISABLE:** This bit allows another station to disable the NIC's transmitter by transmission of a particular multicast packet. The transmitter can be re-enabled by resetting this bit or by reception of a second particular multicast packet. 0: Normal Operation 1: Reception of multicast address hashing to bit 62 disables transmitter, reception of multicast address hashing to bit 63 enables transmitter.
D4	OFST	**COLLISION OFFSET ENABLE:** This bit modifies the backoff algorithm to allow prioritization of nodes. 0: Backoff Logic implements normal algorithm. 1: Forces Backoff algorithm modification to 0 to $2^{\min(3+n,10)}$ slot times for first three collisions, then follows standard backoff. (For first three collisions station has higher average backoff delay making a low priority mode.)
D5	reserved	reserved
D6	reserved	reserved
D7	reserved	reserved

10.0 Internal Registers (Continued)

10.3 Register Descriptions (Continued)

TRANSMIT STATUS REGISTER (TSR)   04H (READ)

This register records events that occur on the media during transmission of a packet. It is cleared when the next transmission is initiated by the host. All bits remain low unless the event that corresponds to a particular bit occurs during transmission. Each transmission should be followed by a read of this register. The contents of this register are not specified until after the first transmission.

7	6	5	4	3	2	1	0
OWC	CDH	FU	CRS	ABT	COL	—	PTX

Bit	Symbol	Description
D0	PTX	**PACKET TRANSMITTED:** Indicates transmission without error. (No excessive collisions or FIFO underrun) (ABT = "0", FU = "0").
D1	reserved	reserved
D2	COL	**TRANSMIT COLLIDED:** Indicates that the transmission collided at least once with another station on the network. The number of collisions is recorded in the Number of Collisions Registers (NCR).
D3	ABT	**TRANSMIT ABORTED:** Indicates the NIC aborted transmission because of excessive collisions. (Total number of transmissions including original transmission attempt equals 16).
D4	CRS	**CARRIER SENSE LOST:** This bit is set when carrier is lost during transmission of the packet. Carrier Sense is monitored from the end of Preamble/Synch until TXEN is dropped. Transmission is not aborted on loss of carrier.
D5	FU	**FIFO UNDERRUN:** If the NIC cannot gain access of the bus before the FIFO empties, this bit is set. Transmission of the packet will be aborted.
D6	CDH	**CD HEARTBEAT:** Failure of the transceiver to transmit a collision signal after transmission of a packet will set this bit. The Collision Detect (CD) heartbeat signal must commence during the first 6.4 μs of the Interframe Gap following a transmission. In certain collisions, the CD Heartbeat bit will be set even though the transceiver is not performing the CD heartbeat test.
D7	OWC	**OUT OF WINDOW COLLISION:** Indicates that a collision occurred after a slot time (51.2 μs). Transmissions rescheduled as in normal collisions.

10.0 Internal Registers (Continued)

10.3 Register Descriptions (Continued)

RECEIVE CONFIGURATION REGISTER (RCR)   0CH (WRITE)

This register determines operation of the NIC during reception of a packet and is used to program what types of packets to accept.

7	6	5	4	3	2	1	0
—	—	MON	PRO	AM	AB	AR	SEP

Bit	Symbol	Description
D0	SEP	**SAVE ERRORED PACKETS** 0: Packets with receive errors are rejected. 1: Packets with receive errors are accepted. Receive errors are CRC and Frame Alignment errors.
D1	AR	**ACCEPT RUNT PACKETS:** This bit allows the receiver to accept packets that are smaller than 64 bytes. The packet must be at least 8 bytes long to be accepted as a runt. 0: Packets with fewer than 64 bytes rejected. 1: Packets with fewer than 64 bytes accepted.
D2	AB	**ACCEPT BROADCAST:** Enables the receiver to accept a packet with an all 1's destination address. 0: Packets with broadcast destination address rejected. 1: Packets with broadcast destination address accepted.
D3	AM	**ACCEPT MULTICAST:** Enables the receiver to accept a packet with a multicast address, all multicast addresses must pass the hashing array. 0: Packets with multicast destination address not checked. 1: Packets with multicast destination address checked.
D4	PRO	**PROMISCUOUS PHYSICAL:** Enables the receiver to accept all packets with a physical address. 0: Physical address of node must match the station address programmed in PAR0–PAR5. 1: All packets with physical addresses accepted.
D5	MON	**MONITOR MODE:** Enables the receiver to check addresses and CRC on incoming packets without buffering to memory. The Missed Packet Tally counter will be incremented for each recognized packet. 0: Packets buffered to memory. 1: Packets checked for address match, good CRC and Frame Alignment but not buffered to memory.
D6	reserved	reserved
D7	reserved	reserved

Note: D2 and D3 are "OR'd" together, i.e., if D2 and D3 are set the NIC will accept broadcast and multicast addresses as well as its own physical address. To establish full promiscuous mode, bits D2, D3, and D4 should be set. In addition the multicast hashing array must be set to all 1's in order to accept all multicast addresses.

1

10.0 Internal Registers (Continued)

10.3 Register Descriptions (Continued)

RECEIVE STATUS REGISTER (RSR)   0CH (READ)

This register records status of the received packet, including information on errors and the type of address match, either physical or multicast. The contents of this register are written to buffer memory by the DMA after reception of a good packet. If packets with errors are to be saved the receive status is written to memory at the head of the erroneous packet if an erroneous packet is received. If packets with errors are to be rejected the RSR will not be written to memory. The contents will be cleared when the next packet arrives. CRC errors, Frame Alignment errors and missed packets are counted internally by the NIC which relinquishes the Host from reading the RSR in real time to record errors for Network Management Functions. The contents of this register are not specified until after the first reception.

7	6	5	4	3	2	1	0
DFR	DIS	PHY	MPA	FO	FAE	CRC	PRX

Bit	Symbol	Description
D0	PRX	**PACKET RECEIVED INTACT:** Indicates packet received without error. (Bits CRC, FAE, FO, and MPA are zero for the received packet.)
D1	CRC	**CRC ERROR:** Indicates packet received with CRC error. Increments Tally Counter (CNTR1). This bit will also be set for Frame Alignment errors.
D2	FAE	**FRAME ALIGNMENT ERROR:** Indicates that the incoming packet did not end on a byte boundary and the CRC did not match at last byte boundary. Increments Tally Counter (CNTR0).
D3	FO	**FIFO OVERRUN:** This bit is set when the FIFO is not serviced causing overflow during reception. Reception of the packet will be aborted.
D4	MPA	**MISSED PACKET:** Set when packet intended for node cannot be accepted by NIC because of a lack of receive buffers or if the controller is in monitor mode and did not buffer the packet to memory. Increments Tally Counter (CNTR2).
D5	PHY	**PHYSICAL/MULTICAST ADDRESS:** Indicates whether received packet had a physical or multicast address type. 0: Physical Address Match 1: Multicast/Broadcast Address Match
D6	DIS	**RECEIVER DISABLED:** Set when receiver disabled by entering Monitor mode. Reset when receiver is re-enabled when exiting Monitor mode.
D7	DFR	**DEFERRING:** Set when CRS or COL inputs are active. If the transceiver has asserted the CD line as a result of the jabber, this bit will stay set indicating the jabber condition.

Note: Following coding applies to CRC and FAE bits

FAE	CRC	Type of Error
0	0	No Error (Good CRC and <6 Dribble Bits)
0	1	CRC Error
1	0	Illegal, will not occur
1	1	Frame Alignment Error and CRC Error

10.0 Internal Registers (Continued)

10.4 DMA REGISTERS

DMA Registers

LOCAL DMA TRANSMIT REGISTERS

| 15 | 8 | 7 | 0 |

(TPSR) | PAGE START

(TBCR0,1) | TRANSMIT BYTE COUNT

LOCAL DMA RECEIVE REGISTERS

| 15 | 8 | 7 | 0 |

(PSTART) | PAGE START

(PSTOP) | PAGE STOP

(CURR) | CURRENT

(BRNY) | BOUNDARY

NOT READABLE | RECEIVE BYTE COUNT

(CLDA0,1) | CURRENT LOCAL DMA ADDRESS

REMOTE DMA REGISTERS

| 15 | 8 | 7 | 0 |

(RSAR0,1) | START ADDRESS

(RBCR0,1) | BYTE COUNT

(CRAD0,1) | CURRENT REMOTE DMA ADDRESS

LOCAL DMA CHANNEL

REMOTE DMA CHANNEL

TL/F/8582–61

The DMA Registers are partitioned into three groups; Transmit, Receive and Remote DMA Registers. The Transmit registers are used to initialize the Local DMA Channel for transmission of packets while the Receive Registers are used to initialize the Local DMA Channel for packet Reception. The Page Stop, Page Start, Current and Boundary Registers are used by the Buffer Management Logic to supervise the Receive Buffer Ring. The Remote DMA Registers are used to initialize the Remote DMA.

Note: In the figure above, registers are shown as 8 or 16 bits wide. Although some registers are 16-bit internal registers, all registers are accessed as 8-bit registers. Thus the 16-bit Transmit Byte Count Register is broken into two 8-bit registers, TBCR0 and TBCR1. Also TPSR, PSTART, PSTOP, CURR and BNRY only check or control the upper 8 bits of address information on the bus. Thus they are shifted to positions 15-8 in the diagram above.

10.5 TRANSMIT DMA REGISTERS

TRANSMIT PAGE START REGISTER (TPSR)

This register points to the assembled packet to be transmitted. Only the eight higher order addresses are specified since all transmit packets are assembled on 256-byte page boundaries. The bit assignment is shown below. The values placed in bits D7–D0 will be used to initialize the higher order address (A8–A15) of the Local DMA for transmission. The·lower order bits (A7–A0) are initialized to zero.

Bit Assignment

	7	6	5	4	3	2	1	0
TPSR	A15	A14	A13	A12	A11	A10	A9	A8

(A7–A0 Initialized to zero)

TRANSMIT BYTE COUNT REGISTER 0,1 (TBCR0, TBCR1)

These two registers indicate the length of the packet to be transmitted in bytes. The count must include the number of

bytes in the source, destination, length and data fields. The maximum number of transmit bytes allowed is 64k bytes. The NIC will not truncate transmissions longer than 1500 bytes. The bit assignment is shown below:

	7	6	5	4	3	2	1	0
TBCR1	L15	L14	L13	L12	L11	L10	L9	L8

	7	6	5	4	3	2	1	0
TBCR0	L7	L6	L5	L4	L3	L2	L1	L0

10.6 LOCAL DMA RECEIVE REGISTERS

PAGE START STOP REGISTERS (PSTART, PSTOP)

The Page Start and Page Stop Registers program the starting and stopping address of the Receive Buffer Ring. Since the NIC uses fixed 256-byte buffers aligned on page boundaries only the upper eight bits of the start and stop address are specified.

PSTART,PSTOP bit assignment

	7	6	5	4	3	2	1	0
PSTART, PSTOP	A15	A14	A13	A12	A11	A10	A9	A8

BOUNDARY (BNRY) REGISTER

This register is used to prevent overflow of the Receive Buffer Ring. Buffer management compares the contents of this register to the next buffer address when linking buffers together. If the contents of this register match the next buffer address the Local DMA operation is aborted.

	7	6	5	4	3	2	1	0
BNRY	A15	A14	A13	A12	A11	A10	A9	A8

10.0 Internal Registers (Continued)

	D7	D6	D5	D4	D3	D2	D1	D0
MAR0	FB7	FB6	FB5	FB4	FB3	FB2	FB1	FB0
MAR1	FB15	FB14	FB13	FB12	FB11	FB10	FB9	FB8
MAR2	FB23	FB22	FB21	FB20	FB19	FB18	FB17	FB16
MAR3	FB31	FB30	FB29	FB28	FB27	FB26	FB25	FB24
MAR4	FB39	FB38	FB37	FB36	FB35	FB34	FB33	FB32
MAR5	FB47	FB46	FB45	FB44	FB43	FB42	FB41	FB40
MAR6	FB55	FB54	FB53	FB52	FB51	FB50	FB49	FB48
MAR7	FB63	FB62	FB61	FB60	FB59	FB58	FB57	FB56

If address Y is found to hash to the value 32 (20H), then FB32 in MAR4 should be initialized to "1". This will cause the NIC to accept any multicast packet with the address Y.

NETWORK TALLY COUNTERS

Three 8-bit counters are provided for monitoring the number of CRC errors, Frame Alignment Errors and Missed Packets. The maximum count reached by any counter is 192 (C0H). These registers will be cleared when read by the CPU. The count is recorded in binary in CT0–CT7 of each Tally Register.

Frame Alignment Error Tally (CNTR0)

This counter is incremented every time a packet is received with a Frame Alignment Error. The packet must have been recognized by the address recognition logic. The counter is cleared after it is read by the processor.

	7	6	5	4	3	2	1	0
CNTR0	CT7	CT6	CT5	CT4	CT3	CT2	CT1	CT0

CRC Error Tally (CNTR1)

This counter is incremented every time a packet is received with a CRC error. The packet must first be recognized by the address recognition logic. The counter is cleared after it is read by the processor.

	7	6	5	4	3	2	1	0
CNTR1	CT7	CT6	CT5	CT4	CT3	CT2	CT1	CT0

Frames Lost Tally Register (CNTR2)

This counter is incremented if a packet cannot be received due to lack of buffer resources. In monitor mode, this counter will count the number of packets that pass the address recognition logic.

	7	6	5	4	3	2	1	0
CNTR2	CT7	CT6	CT5	CT4	CT3	CT2	CT1	CT0

FIFO

This is an eight bit register that allows the CPU to examine the contents of the FIFO after loopback. The FIFO will contain the last 8 data bytes transmitted in the loopback packet. Sequential reads from the FIFO will advance a pointer in the FIFO and allow reading of all 8 bytes.

	7	6	5	4	3	2	1	0
FIFO	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0

Note: The FIFO should only be read when the NIC has been programmed in loopback mode.

NUMBER OF COLLISIONS (NCR)

This register contains the number of collisions a node experiences when attempting to transmit a packet. If no collisions are experienced during a transmission attempt, the COL bit of the TSR will not be set and the contents of NCR will be zero. If there are excessive collisions, the ABT bit in the TSR will be set and the contents of NCR will be zero. The NCR is cleared after the TXP bit in the CR is set.

	7	6	5	4	3	2	1	0
NCR	—	—	—	—	NC3	NC2	NC1	NC0

11.0 Initialization Procedures

The NIC must be initialized prior to transmission or reception of packets from the network. Power on reset is applied to the NIC's reset pin. This clears/sets the following bits:

Register	Reset Bits	Set Bits
Command Register (CR)	TXP, STA	RD2, STP
Interrupt Status (ISR)		RST
Interrupt Mask (IMR)	All Bits	
Data Control (DCR)		LAS
Transmit Config. (TCR)	LB1, LB0	

The NIC remains in its reset state until a Start Command is issued. This guarantees that no packets are transmitted or received and that the NIC remains a bus slave until all appropriate internal registers have been programmed. After initialization the STP bit of the command register is reset and packets may be received and transmitted.

Initialization Sequence

The following initialization procedure is mandatory.

1) Program Command Register for Page 0 (Command Register = 21H)
2) Initialize Data Configuration Register (DCR)
3) Clear Remote Byte Count Registers (RBCR0, RBCR1)
4) Initialize Receive Configuration Register (RCR)
5) Place the NIC in LOOPBACK mode 1 or 2 (Transmit Configuration Register = 02H or 04H)
6) Initialize Receive Buffer Ring: Boundary Pointer (BNDRY), Page Start (PSTART), and Page Stop (PSTOP)
7) Clear Interrupt Status Register (ISR) by writing 0FFh to it.
8) Initialize Interrupt Mask Register (IMR)
9) Program Command Register for page 1 (Command Register = 61H)
   i) Initialize Physical Address Registers (PAR0-PAR5)
   ii) Initialize Multicast Address Registers (MAR0-MAR7)
   iii) Initialize CURRent pointer
10) Put NIC in START mode (Command Register = 22H). The local receive DMA is still not active since the NIC is in LOOPBACK.
11) Initialize the Transmit Configuration for the intended value. The NIC is now ready for transmission and reception.

1

11.0 Initialization Procedures

(Continued)

Before receiving packets, the user must specify the location of the Receive Buffer Ring. This is programmed in the Page Start and Page Stop Registers. In addition, the Boundary and Current Page Registers must be initialized to the value of the Page Start Register. These registers will be modified during reception of packets.

12.0 Loopback Diagnostics

Three forms of local loopback are provided on the NIC. The user has the ability to loopback through the deserializer on the DP8390C NIC, through the DP8391 SNI, and to the coax to check the link through the transceiver circuitry. **Because of the half duplex architecture of the NIC, loopback testing is a special mode of operation with the following restrictions:**

Restrictions During Loopback

The FIFO is split into two halves, one used for transmission the other for reception. Only 8-bit fields can be fetched from memory so two tests are required for 16-bit systems to verify integrity of the entire data path. During loopback the maximum latency from the assertion of BREQ to BACK is 2.0 μs. Systems that wish to use the loopback test yet do not meet this latency can limit the loopback packet to 7 bytes without experiencing underflow. Only the last 8 bytes of the loopback packet are retained in the FIFO. The last 8 bytes can be read through the FIFO register which will advance through the FIFO to allow reading the receive packet sequentially.

DESTINATION ADDRESS	= (6 bytes) Station Physical Address
SOURCE ADDRESS	
LENGTH	2 bytes
DATA	= 46 to 1500 bytes
CRC	Appended by NIC if CRC = "0" in TCR

When in word-wide mode with Byte Order Select set, the loopback packet must be assembled in the even byte locations as shown below. (The loopback only operates with byte wide transfers.)

LS BYTE (AD8-15)	MS BYTE (AD0-7)
	DESTINATION
	SOURCE
	LENGTH
~	~ DATA ~
	CRC

WTS = "1"   BOS = "1"   (DCR BITS)
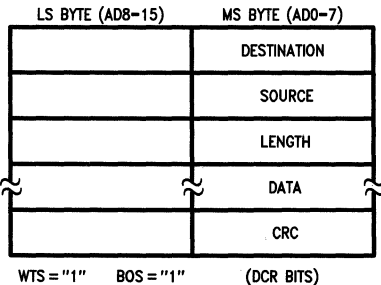
TL/F/8582-15

When in word-wide mode with Byte Order Select low, the following format must be used for the loopback packet.

MS BYTE (AD8-15)	LS BYTE (AD0-7)
DESTINATION	
SOURCE	
LENGTH	
~ DATA ~	~
CRC	

WTS = "1"   BOS = "0"   (DCR BITS)
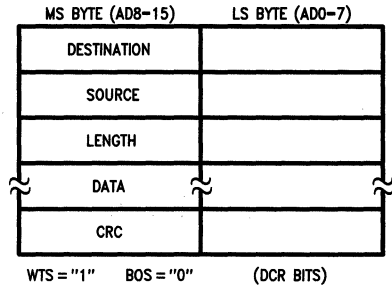
TL/F/8582-16

Note: When using loopback in word mode 2n bytes must be programmed in TBCR0, 1. Where n = actual number of bytes assembled in even or odd location.

To initiate a loopback the user first assembles the loopback packet then selects the type of loopback using the Transmit Configuration register bits LB0, LB1. The transmit configuration register must also be set to enable or disable CRC generation during transmission. The user then issues a normal transmit command to send the packet. During loopback the receiver checks for an address match and if CRC bit in the TCR is set, the receiver will also check the CRC. The last 8 bytes of the loopback packet are buffered and can be read out of the FIFO using the FIFO read port.

Loopback Modes

MODE 1: Loopback Through the Controller (LB1 = 0, LB0 = 1).

If the loopback is through the NIC then the serializer is simply linked to the deserializer and the receive clock is derived from the transmit clock.

MODE 2: Loopback Through the SNI (LB1 = 1, LB0 = 0).

If the loopback is to be performed through the SNI, the NIC provides a control (LPBK) that forces the SNI to loopback all signals.

MODE 3: Loopback to Coax (LB1 = 1, LB0 = 1).

Packets can be transmitted to the coax in loopback mode to check all of the transmit and receive paths and the coax itself.

Note: In MODE 1, CRS and COL lines are not indicated in any status register, but the NIC will still defer if these lines are active. In MODE 2, COL is masked and in MODE 3 CRS and COL are not masked. It is not possible to go directly between the loopback modes, it is necessary to return to normal operation (00H) when changing modes.

Reading the Loopback Packet

The last eight bytes of a received packet can be examined by 8 consecutive reads of the FIFO register. The FIFO pointer is incremented after the rising edge of the CPU's read strobe by internally synchronizing and advancing the pointer. This may take up to four bus clock cycles, if the pointer has not been incremented by the time the CPU reads the FIFO register again, the NIC will insert wait states

Note: The FIFO may only be read during Loopback. Reading the FIFO at any other time will cause the NIC to malfunction.

12.0 Loopback Diagnostics (Continued)

Alignment of the Received Packet in the FIFO

Reception of the packet in the FIFO begins at location zero, after the FIFO pointer reaches the last location in the FIFO, the pointer wraps to the top of the FIFO overwriting the previously received data. This process continues until the last byte is received. The NIC then appends the received byte count in the next two locations of the FIFO. The contents of the Upper Byte Count are also copied to the next FIFO location. The number of bytes used in the loopback packet determines the alignment of the packet in the FIFO. The alignment for a 64-byte packet is shown below.

FIFO LOCATION	FIFO CONTENTS		
0	LOWER BYTE COUNT	→	First Byte Read
1	UPPER BYTE COUNT	→	Second Byte Read
2	UPPER BYTE COUNT		•
3	LAST BYTE		•
4	CRC1		•
5	CRC2		•
6	CRC3		•
7	CRC4	→	Last Byte Read

For the following alignment in the FIFO the packet length should be $(N \times 8) + 5$ Bytes. Note that if the CRC bit in the TCR is set, CRC will not be appended by the transmitter. If the CRC is appended by the transmitter, the last four bytes, bytes N-3 to N, correspond to the CRC.

FIFO LOCATION	FIFO CONTENTS		
0	BYTE N-4	→	First Byte Read
1	BYTE N-3 (CRC1)	AR	Second Byte Read
2	BYTE N-2 (CRC2)		•
3	BYTE N-1 (CRC3)		•
4	BYTE N (CRC4)		•
5	LOWER BYTE COUNT		•
6	UPPER BYTE COUNT	→	Last Byte Read
7	UPPER BYTE COUNT		

LOOPBACK TESTS

Loopback capabilities are provided to allow certain tests to be performed to validate operation of the DP8390C NIC prior to transmitting and receiving packets on a live network. Typically these tests may be performed during power up of a node. The diagnostic provides support to verify the following:

1) Verify integrity of data path. Received data is checked against transmitted data.

2) Verify CRC logic's capability to generate good CRC on transmit, verify CRC on receive (good or bad CRC).

3) Verify that the Address Recognition Logic can
   a) Recognize address match packets
   b) Reject packets that fail to match an address

LOOPBACK OPERATION IN THE NIC

Loopback is a modified form of transmission using only half of the FIFO. This places certain restrictions on the use of loopback testing. When loopback mode is selected in the TCR, the FIFO is split. A packet should be assembled in memory with programming of TPSR and TBCR0, TBCR1 registers. When the transmit command is issued the following operations occur:

Transmitter Actions

1) Data is transferred from memory by the DMA until the FIFO is filled. For each transfer TBCR0 and TBCR1 are decremented. (Subsequent burst transfers are initiated when the number of bytes in the FIFO drops below the programmed threshold.)

2) The NIC generates 56 bits of preamble followed by an 8-bit synch pattern.

3) Data transferred from FIFO to serializer.

4) If CRC=1 in TCR, no CRC calculated by NIC, the last byte transmitted is the last byte from the FIFO (Allows software CRC to be appended). If CRC=0, NIC calculates and appends four bytes of CRC.

5) At end of Transmission PTX bit set in ISR.

Receiver Actions

1) Wait for synch, all preamble stripped.

2) Store packet in FIFO, increment receive byte count for each incoming byte.

3) If CRC=0 in TCR, receiver checks incoming packet for CRC errors. If CRC=1 in TCR, receiver does not check CRC errors, CRC error bit always set in RSR (for address matching packets).

4) At end of receive, receive byte count written into FIFO, receive status register is updated. The PRX bit is typically set in the RSR even if the address does not match. If CRC errors are forced, the packet must match the address filters in order for the CRC error bit in the RS to be set.

EXAMPLES

The following examples show what results can be expected from a properly operating NIC during loopback. The restrictions and results of each type of loopback are listed for reference. The loopback tests are divided into two sets of tests. One to verify the data path, CRC generation and byte count through all three paths. The second set of tests uses internal loopback to verify the receiver's CRC checking and address recognition. For all of the tests the DCR was programmed to 40h.

PATH	TCR	RCR	TSR	RSR	ISR
NIC Internal	02	00	53(1)	02(2)	02(3)

Note 1: Since carrier sense and collision detect inputs are blocked during internal loopback, carrier and CD heartbeat are not seen and the CRS and CDH bits are set.

Note 2: CRC errors are always indicated by receiver if CRC is appended by the transmitter.

Note 3: Only the PTX bit in the ISR is set, the PRX bit is only set if status is written to memory. In loopback this action does not occur and the PRX bit remains 0 for all loopback modes.

Note 4: All values are hex.

1

12.0 Loopback Diagnostics (Continued)

PATH	TCR	RCR	TSR	RSR	ISR
NIC External	04	00	43(1)	02	02

Note 1: CDH is set, CRS is not set since it is generated by the external encoder/decoder.

PATH	TCR	RCR	TSR	RSR	ISR
NIC External	06	00	03(1)	02	02(2)

Note 1: CDH and CRS should not be set. The TSR however, could also contain 01H,03H,07H and a variety of other values depending on whether collisions were encountered or the packet was deferred.

Note 2. Will contain 08H if packet is not transmittable.

Note 3: During external loopback the NIC is now exposed to network traffic, it is therefore possible for the contents of both the Receive portion of the FIFO and the RSR to be corrupted by any other packet on the network. Thus in a live network the contents of the FIFO and RSR should not be depended on. The NIC will still abide by the standard CSMA/CD protocol in external loopback mode. (i.e. The network will not be disturbed by the loopback packet).

Note 4: All values are hex.

CRC AND ADDRESS RECOGNITION

The next three tests exercise the address recognition logic and CRC. These tests should be performed using internal loopback only so that the NIC is isolated from interference from the network. These tests also require the capability to generate CRC in software.

The address recognition logic cannot be directly tested. The CRC and FAE bits in the RSR are only set if the address of the packet matches the address filters. If errors are expected to be set and they are not set, the packet has been rejected on the basis of an address mismatch. The following sequence of packets will test the address recognition logic. The DCR should be set to 40H, the TCR should be set to 03H with a software generated CRC.

Packet Contents			Results
Test	**Address**	**CRC**	**RSR**
Test A	Matching	Good	01(1)
Test B	Matching	Bad	02(2)
Test C	Non-Matching	Bad	01

Note 1: Status will read 21H if multicast address used.

Note 2: Status will read 22H if multicast address used.

Note 3: In test A, the RSR is set up. In test B the address is found to match since the CRC is flagged as bad. Test C proves that the address recognition logic can distinguish a bad address and does not notify the RSR of the bad CRC. The receiving CRC is proven to work in test A and test B.

Note 4: All values are hex.

NETWORK MANAGEMENT FUNCTIONS

Network management capabilities are required for maintenance and planning of a local area network. The NIC supports the minimum requirement for network management in hardware, the remaining requirements can be met with software counts. There are three events that software alone can not track during reception of packets: CRC errors, Frame Alignment errors, and missed packets.

Since errored packets can be rejected, the status associated with these packets is lost unless the CPU can access the Receive Status Register before the next packet arrives. In situations where another packet arrives very quickly, the CPU may have no opportunity to do this. The NIC counts the number of packets with CRC errors and Frame Alignment errors. 8-bit counters have been selected to reduce overhead. The counters will generate interrupts whenever their MSBs are set so that a software routine can accumulate the network statistics and reset the counters before overflow occurs. The counters are sticky so that when they reach a count of 192 (C0H) counting is halted. An additional counter is provided to count the number of packets NIC misses due to buffer overflow or being offline.

The structure of the counters is shown below:
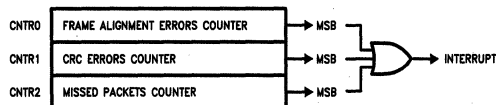
TL/F/8582-63

Additional information required for network management is available in the Receive and Transmit Status Registers. Transmit status is available after each transmission for information regarding events during transmission.

Typically, the following statistics might be gathered in software:

Traffic: Frames Sent OK
       Frames Received OK
       Multicast Frames Received
       Packets Lost Due to Lack of Resources
       Retries/Packet

Errors: CRC Errors
       Alignment Errors
       Excessive Collisions
       Packet with Length Errors
       Heartbeat Failure

13.0 Bus Arbitration and Timing

The NIC operates in three possible modes:

- BUS MASTER (WHILE PERFORMING DMA)
- BUS SLAVE (WHILE BEING ACCESSED BY CPU)
- IDLE
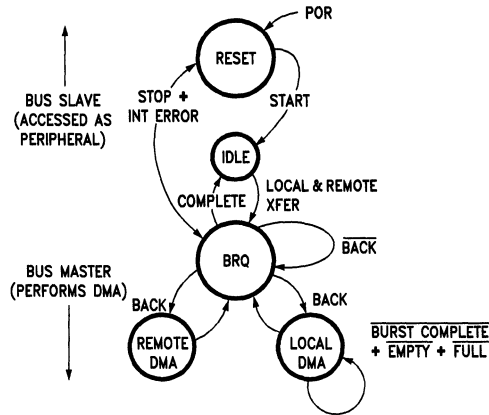
TL/F/8582-64

The NIC powers up as a bus slave in the Reset State, the receiver and transmitter are both disabled in this state. The reset state can be reentered under three conditions, soft reset (Stop Command), hard reset (RESET input) or an error that shuts down the receiver or transmitter (FIFO underflow or oveflow, receive buffer ring overflow). After initialization of registers, the NIC is issued a Start command and the NIC enters Idle state. Until the DMA is required the NIC remains in an idle state. The idle state is exited by a request from the FIFO in the case of receive or transmit, or from the Remote/DMA in the case of Remote DMA operation. After

acquiring the bus in a BREQ/BACK handshake the Remote or Local DMA transfer is completed and the NIC reenters the idle state.

DMA TRANSFERS TIMING

The DMA can be programmed for the following types of transfers:

16-Bit Address, 8-bit Data Transfer

16-Bit Address, 16-bit Data Transfer

32-Bit Address, 8-bit Data Transfer

32-Bit Address, 16-bit Data Transfer

All DMA transfers use BSCK for timing. 16-Bit Address modes require 4 BSCK cycles as shown below:

16-Bit Address, 8-Bit Data
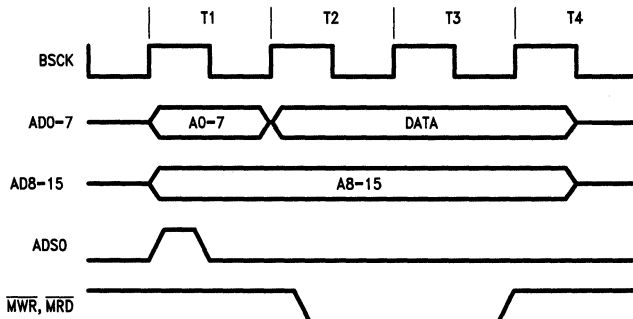
TL/F/8582-65

13.0 Bus Arbitration and Timing (Continued)

16-Bit Address, 16-Bit Data
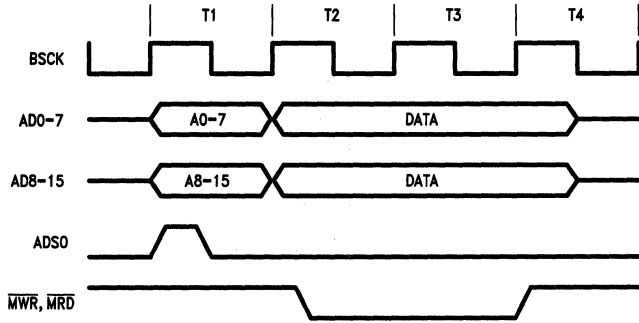
TL/F/8582-66

32-Bit Address, 8-Bit Data
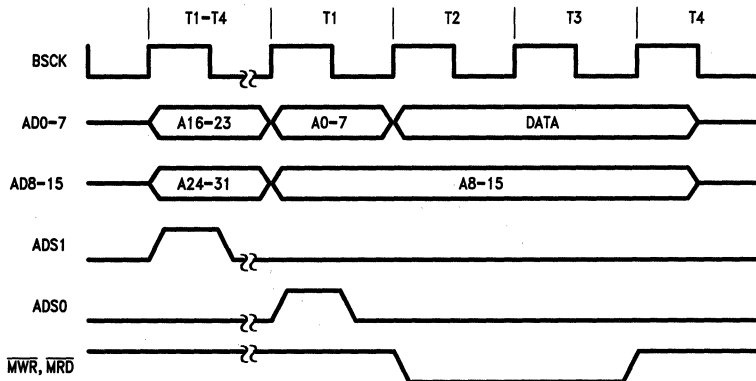
TL/F/8582-67

32-Bit Address, 16-Bit Data
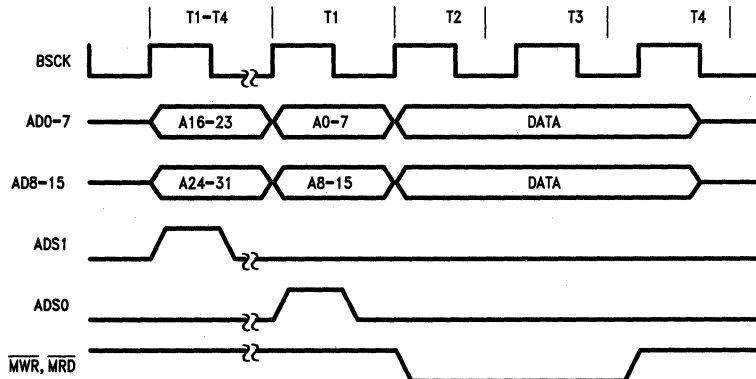
TL/F/8582-68

Note: In 32-bit address mode, ADS1 is at TRI-STATE after the first T1–T4 states; thus, a 4.7k pull-down resistor is required for 32-bit address mode.

13.0 Bus Arbitration and Timing (Continued)

When in 32-bit mode four additional BSCK cycles are required per burst. The first bus cycle (T1'–T4') of each burst is used to output the upper 16-bit addresses. This 16-bit address is programmed in RSAR0 and RSAR1 and points to a 64k page of system memory. All transmitted or received packets are constrained to reside within this 64k page.

FIFO BURST CONTROL

All Local DMA transfers are burst transfers, once the DMA requests the bus and the bus is acknowledged, the DMA will transfer an exact burst of bytes programmed in the Data Configuration Register (DCR) then relinquish the bus. If there are remaining bytes in the FIFO the next burst will not be initiated until the FIFO threshold is exceeded. If desired the DMA can empty/fill the FIFO when it acquires the bus. If BACK is removed during the transfer, the burst transfer will be aborted. **(DROPPING BACK DURING A DMA CYCLE IS NOT RECOMMENDED.)**
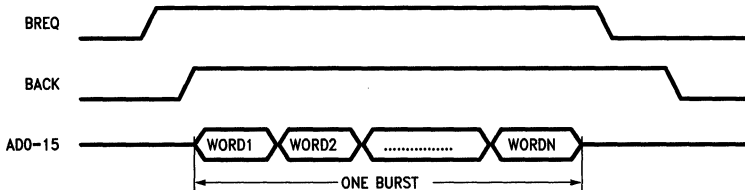
where N = 1, 2, 4, or 6 Words or N = 2, 4, 8, or 12 Bytes when in byte mode

TL/F/8582–69

INTERLEAVED LOCAL OPERATION

If a remote DMA transfer is initiated or in progress when a packet is being received or transmitted, the Remote DMA transfer will be interrupted for higher priority Local DMA transfers. When the Local DMA transfer is completed the Remote DMA will rearbitrate for the bus and continue its transfers. This is illustrated below:
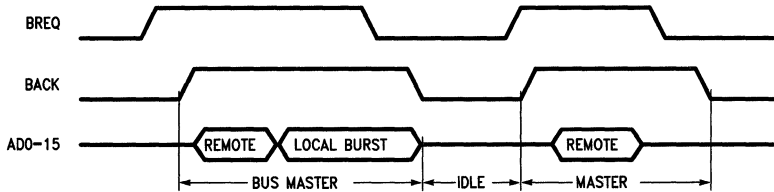
TL/F/8582–70

Note that if the FIFO requires service while a remote DMA is in progress, BREQ is not dropped and the Local DMA burst is appended to the Remote Transfer. When switching from a local transfer to a remote transfer, however, BREQ is dropped and raised again. This allows the CPU or other devices to fairly contend for the bus.

REMOTE DMA-BIDIRECTIONAL PORT CONTROL

The Remote DMA transfers data between the local buffer memory and a bidirectional port (memory to I/O transfer).

This transfer is arbited on a byte by byte basis versus the burst transfer used for Local DMA transfers. This bidirectional port is also read/written by the host. All transfers through this port are asynchronous. At any one time transfers are limited to one direction, either from the port to local buffer memory (Remote Write) or from local buffer memory to the port (Remote Read).

Bus Handshake Signals for Remote DMA Transfers
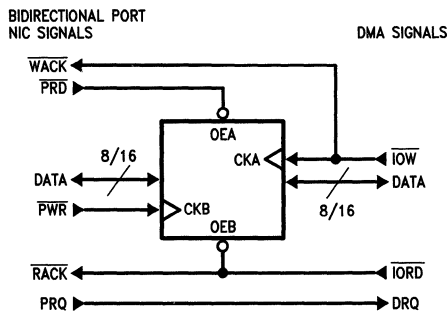
TL/F/8582–71

13.0 Bus Arbitration and Timing (Continued)

REMOTE READ TIMING

1) The DMA reads byte/word from local buffer memory and writes byte/word into latch, increments the DMA address and decrements the byte count (RBCR0,1).

2) A Request Line (PRQ) is asserted to inform the system that a byte is available.

3) The system reads the port, the read strobe (RACK) is used as an acknowledge by the Remote DMA and it goes back to step 1.

Steps 1–3 are repeated until the remote DMA is complete.

Note that in order for the Remote DMA to transfer a byte from memory to the latch, it must arbitrate access to the local bus via a BREQ, BACK handshake. After each byte or word is transferred to the latch, BREQ is dropped. If a Local DMA is in progress, the Remote DMA is held off until the local DMA is complete.
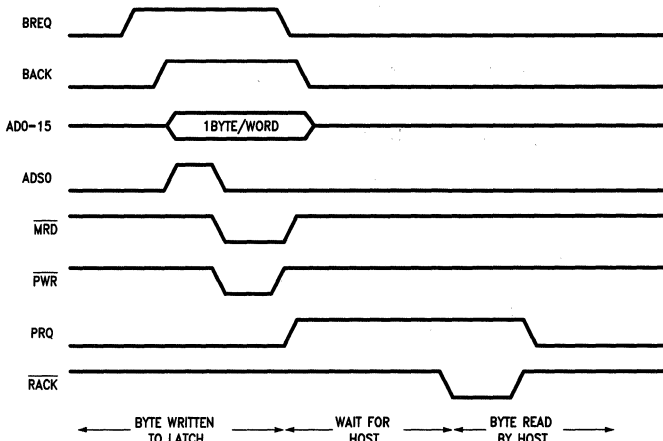
TL/F/8582–72

REMOTE WRITE TIMING

A Remote Write operation transfers data from the I/O port to the local buffer RAM. The NIC initiates a transfer by requesting a byte/word via the PRQ. The system transfers a byte/word to the latch via IOW, this write strobe is detected by the NIC and PRQ is removed. By removing the PRQ, the Remote DMA holds off further transfers into the latch until the current byte/word has been transferred from the latch, PRQ is reasserted and the next transfer can begin.

1) NIC asserts PRQ. System writes byte/word into latch. NIC removes PRQ.

2) Remote DMA reads contents of port and writes byte/word to local buffer memory, increments address and decrements byte count (RBCR0,1).

3) Go back to step 1.

Steps 1–3 are repeated until the remote DMA is complete.
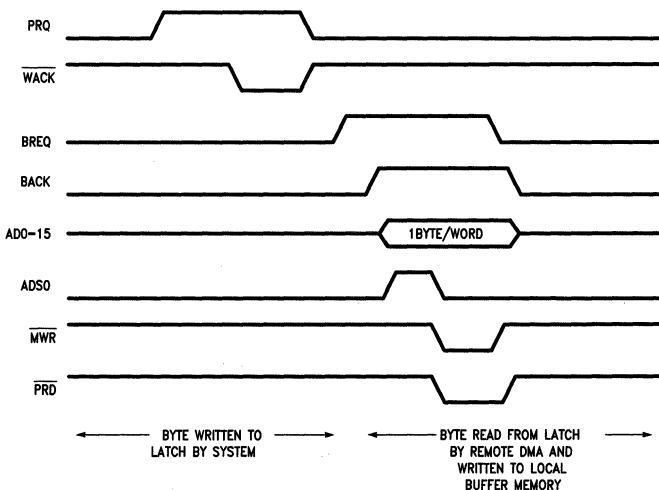
TL/F/8582–73

13.0 Bus Arbitration and Timing (Continued)

SLAVE MODE TIMING

When $\overline{CS}$ is low, the NIC becomes a bus slave. The CPU can then read or write any internal registers. All register access is byte wide. The timing for register access is shown below. The host CPU accesses internal registers with four address lines, RA0–RA3, $\overline{SRD}$ and $\overline{SWR}$ strobes.

ADS0 is used to latch the address when interfacing to a multiplexed, address data bus. Since the NIC may be a local bus master when the host CPU attempts to read or write to the controller, an $\overline{ACK}$ line is used to hold off the CPU until the NIC leaves master mode. Some number of BSCK cycles is also required to allow the NIC to synchronize to the read or write cycle.

Write to Register

RA0–RA3	REGISTER ADDRESS
ADS0	
AD0–AD7	DATA
$\overline{SWR}$	
$\overline{ACK}$	
$\overline{CS}$	

TL/F/8582–74

Read from Register

RA0–RA3	REGISTER ADDRESS
ADS0	
AD0–AD7	DATA
$\overline{SRD}$	
$\overline{ACK}$	
$\overline{CS}$	

TL/F/8582–75

1
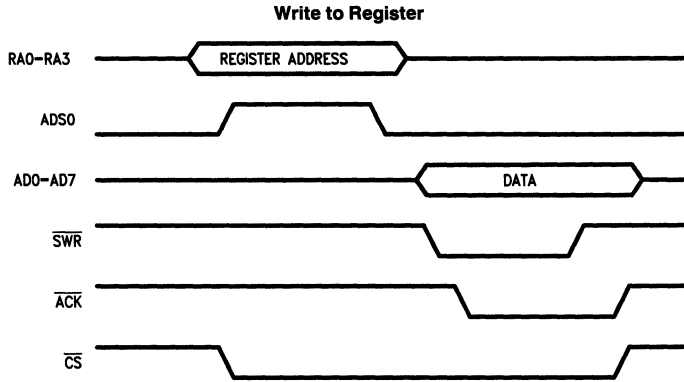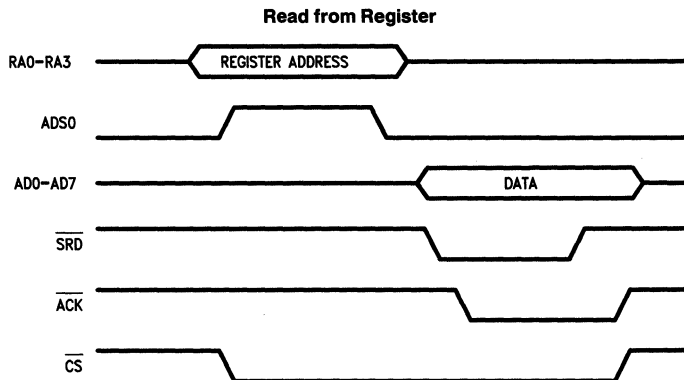
14.0 Preliminary Electrical Characteristics

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	$-0.5V$ to $+7.0V$
DC Input Voltage (V_{IN})	$-0.5V$ to $V_{CC} + 0.5V$
DC Output Voltage (V_{OUT})	$-0.5V$ to $V_{CC} + 0.5V$
Storage Temperature Range (T_{STG})	$-65°C$ to $+150°C$
Power Dissipation (PD)	500 mW
Lead Temp. (TL) (Soldering, 10 sec.)	260°C
ESD rating ($R_{ZAP} = 1.5k$, $C_{ZAP} = 120$ pF)	1600V

Preliminary DC Specifications $T_A = 0°C$ to $70°C$, $V_{CC} = 5V \pm 5\%$, unless otherwise specified

Symbol	Parameter	Conditions	Min	Max	Units
V_{OH}	Minimum High Level Output Voltage (Notes 1, 4)	$I_{OH} = -20 \mu A$ $I_{OH} = -2.0$ mA	$V_{CC} - 0.1$ 3.5		V V
V_{OL}	Minimum Low Level Output Voltage (Notes 1, 4)	$I_{OL} = 20 \mu A$ $I_{OL} = 2.0$ mA		0.1 0.4	V V
V_{IH}	Minimum High Level Input Voltage (Note 2)		2.0		V
V_{IH2}	Minimum High Level Input Voltage for RACK, WACK (Note 2)		2.7		V
V_{IL}	Minimum Low Level Input Voltage (Note 2)			0.8	V
V_{IL2}	Minimum Low Level Input Voltage For RACK, WACK (Note 2)			0.6	V
I_{IN}	Input Current	$V_I = V_{CC}$ or GND	-1.0	$+1.0$	μA
I_{OZ}	Maximum TRI-STATE Output Leakage Current	$V_{OUT} = V_{CC}$ or GND	-10	$+10$	μA
I_{CC}	Average Supply Current (Note 3)	TXCK = 10 MHz RXCK = 10 MHz BSCK = 20 MHz $I_{OUT} = 0 \mu A$ $V_{IN} = V_{CC}$ or GND		40	mA

Note 1: These levels are tested dynamically using a limited amount of functional test patterns, please refer to AC Test Load.

Note 2: Limited functional test patterns are performed at these input levels. The majority of functional tests are performed at levels of 0V and 3V.

Note 3: This is measured with a 0.1 μF bypass capacitor between V_{CC} and GND.

Note 4: The low drive CMOS compatible V_{OH} and V_{OL} limits are not tested directly. Detailed device characterization validates that this specification can be guaranteed by testing the high drive TTL compatible V_{OL} and V_{OH} specification.

15.0 Switching Characteristics AC Specs DP8390C Note: All Timing is Preliminary

Register Read (Latched Using ADS0)
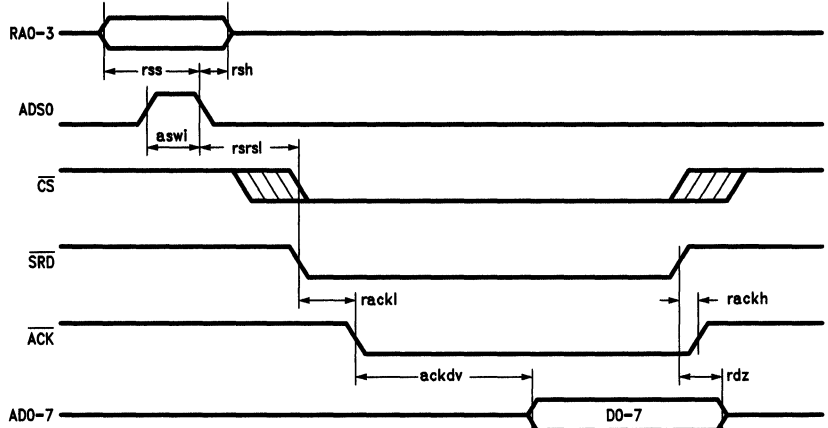
TL/F/8582-76

Symbol	Parameter	Min	Max	Units
rss	Register Select Setup to ADS0 Low	10		ns
rsh	Register Select Hold from ADS0 Low	13		ns
aswi	Address Strobe Width In	15		ns
ackdv	Acknowledge Low to Data Valid		55	ns
rdz	Read Strobe to Data TRI-STATE	15	70	ns
rackl	Read Strobe to $\overline{ACK}$ Low (Notes 1, 3)		n*bcyc + 30	ns
rackh	Read Strobe to $\overline{ACK}$ High		30	ns
rsrsl	Register Select to Slave Read Low, Latched RS0–3 (Note 2)	10		ns

Note 1: $\overline{ACK}$ is not generated until $\overline{CS}$ and $\overline{SRD}$ are low and the NIC has synchronized to the register access. The NIC will insert an integral number of Bus Clock cycles until it is synchronized. In Dual Bus systems additional cycles will be used for a local or remote DMA to complete. Wait states must be issued to the CPU until $\overline{ACK}$ is asserted low.

Note 2: $\overline{CS}$ may be asserted before or after $\overline{SRD}$. If $\overline{CS}$ is asserted after $\overline{SRD}$, rackl is referenced from falling edge of $\overline{CS}$. $\overline{CS}$ can be de-asserted concurrently with $\overline{SRD}$ or after $\overline{SRD}$ is de-asserted.

Note 3: These limits include the RC delay inherent in our test method. These signals typically turn off within 15 ns, enabling other devices to drive these lines with no contention.

15.0 Switching Characteristics (Continued)

Register Read (Non Latched, ADS0 = 1)
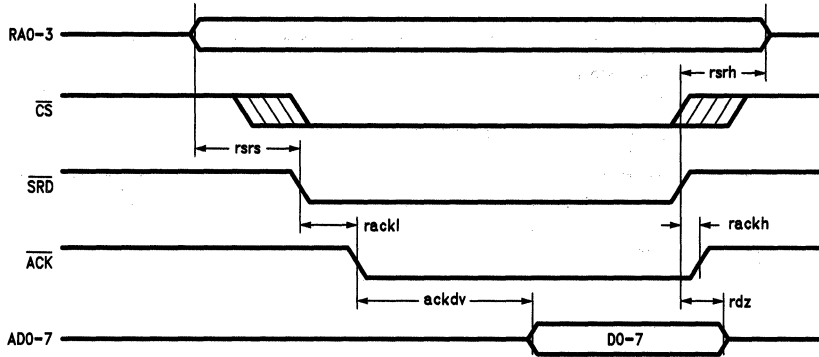
TL/F/8582-77

Symbol	Parameter	Min	Max	Units
rsrs	Register Select to Read Setup (Notes 1, 3)	10		ns
rsrh	Register Select Hold from Read	0		ns
ackdv	$\overline{ACK}$ Low to Valid Data		55	ns
rdz	Read Strobe to Data TRI-STATE (Note 2)	15	70	ns
rackl	Read Strobe to $\overline{ACK}$ Low (Note 3)		n*bcyc + 30	ns
rackh	Read Strobe to $\overline{ACK}$ High		30	ns

Note 1: rsrs includes flow-through time of latch.

Note 2: These limits include the RC delay inherent in our test method. These signals typically turn off within 15 ns enabling other devices to drive these lines with no contention.

Note 3: $\overline{CS}$ may be asserted before or after RA0-3, and $\overline{SRD}$, since address decode begins when $\overline{ACK}$ is asserted. If $\overline{CS}$ is asserted after RA0-3, and $\overline{SRD}$, rack1 is referenced from falling edge of $\overline{CS}$.

15.0 Switching Characteristics (Continued)

Register Write (Latched Using ADS0)
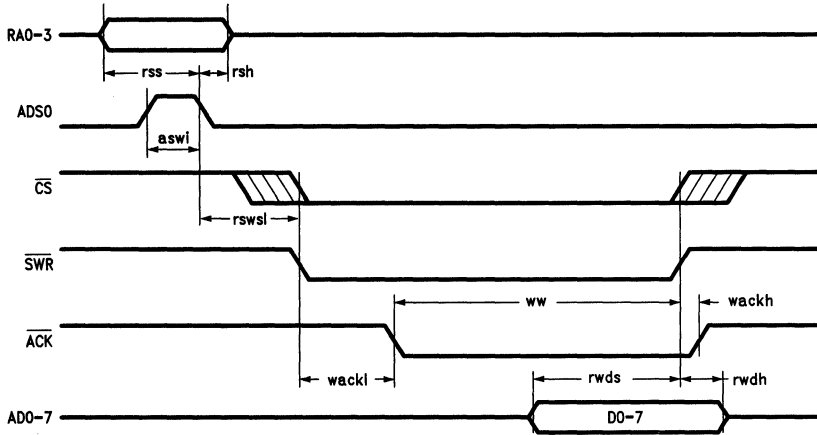
TL/F/8582–78

Symbol	Parameter	Min	Max	Units
rss	Register Select Setup to ADS0 Low	10		ns
rsh	Register Select Hold from ADS0 Low	17		ns
aswi	Address Strobe Width In	15		ns
rwds	Register Write Data Setup	20		ns
rwdh	Register Write Data Hold	21		ns
ww	Write Strobe Width from $\overline{ACK}$	50		ns
wackh	Write Strobe High to $\overline{ACK}$ High		30	ns
wackl	Write Low to $\overline{ACK}$ Low (Notes 1, 2)		n*bcyc + 30	ns
rswsl	Register Select to Write Strobe Low	10		ns

Note 1: $\overline{ACK}$ is not generated until $\overline{CS}$ and $\overline{SWR}$ are low and the NIC has synchronized to the register access. In Dual Bus Systems additional cycles will be used for a local DMA or Remote DMA to complete.

Note 2: $\overline{CS}$ may be asserted before or after $\overline{SWR}$. If $\overline{CS}$ is asserted after $\overline{SWR}$, wackl is referenced from falling edge of $\overline{CS}$.

15.0 Switching Characteristics (Continued)

Register Write (Non Latched, ADS0 = 1)
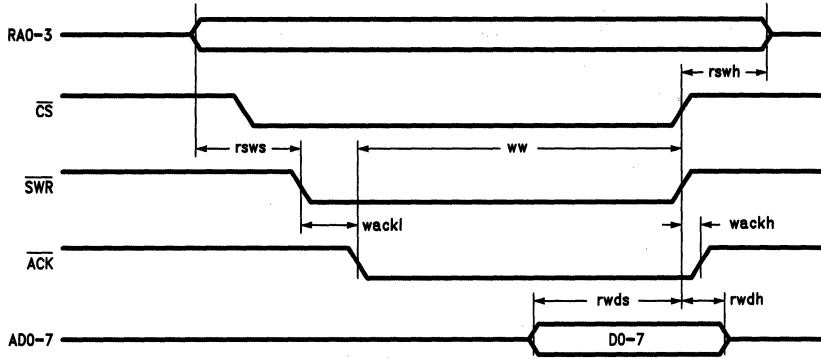
TL/F/8582-79

Symbol	Parameter	Min	Max	Units
rsws	Register Select to Write Setup (Note 1)	15		ns
rswh	Register Select Hold from Write	0		ns
rwds	Register Write Data Setup	20		ns
rwdh	Register Write Data Hold	21		ns
wackl	Write Low to $\overline{ACK}$ Low (Note 2)		$n*bcyc + 30$	ns
wackh	Write High to $\overline{ACK}$ High		30	ns
ww	Write Width from $\overline{ACK}$	50		ns

Note 1: Assumes ADS0 is high when RA0-3 changing.

Note 2: $\overline{ACK}$ is not generated until $\overline{CS}$ and $\overline{SWR}$ are low and the NIC has synchronized to the register access. In Dual Bus systems additional cycles will be used for a local DMA or remote DMA to complete.

15.0 Switching Characteristics (Continued)

DMA Control, Bus Arbitration
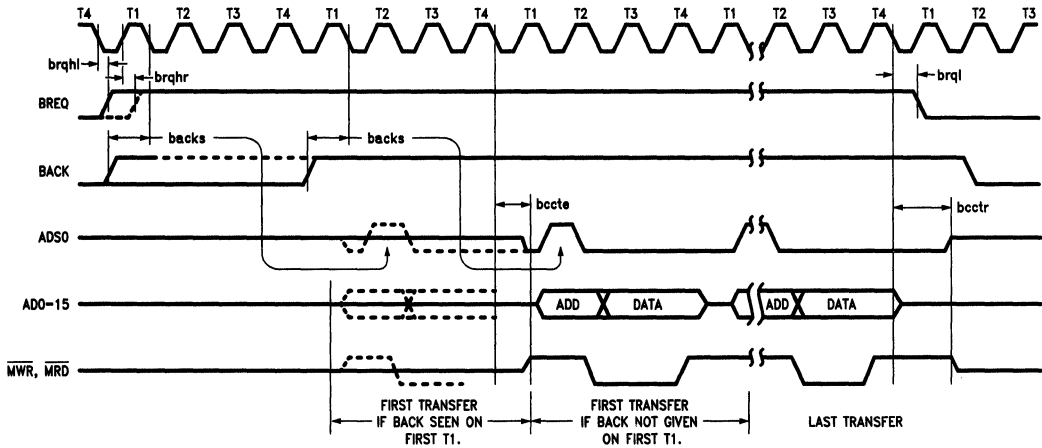
TL/F/8582–80

Symbol	Parameter	Min	Max	Units
brqhl	Bus Clock to Bus Request High for Local DMA		43	ns
brqhr	Bus Clock to Bus Request High for Remote DMA		38	ns
brql	Bus Request Low from Bus Clock		55	ns
backs	Acknowledge Setup to Bus Clock (Note 1)	2		ns
bccte	Bus Clock to Control Enable		60	ns
bcctr	Bus Clock to Control Release (Notes 2, 3)		70	ns

Note 1: BACK must be setup before T1 after BREQ is asserted. Missed setup will slip the beginning of the DMA by four bus clocks. The Bus Latency will influence the allowable FIFO threshold and transfer mode (empty/fill vs exact burst transfer).

Note 2: During remote DMA transfers only, a single bus transfer is performed. During local DMA operations burst mode transfers are performed.

Note 3: These limits include the RC delay inherent in our test method. These signals typically turn off within 15 ns enabling other devices to drive these lines with no contention.

15.0 Switching Characteristics (Continued)

DMA Address Generation
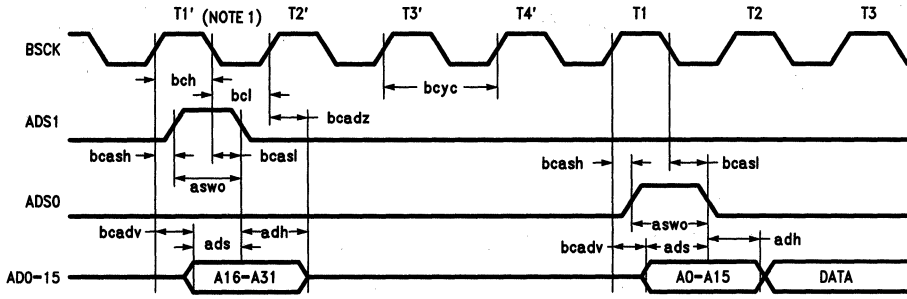
TL/F/8582–81

Symbol	Parameter	Min	Max	Units
bcyc	Bus Clock Cycle Time (Note 2)	50	1000	ns
bch	Bus Clock High Time	22.5		ns
bcl	Bus Clock Low Time	22.5		ns
bcash	Bus Clock to Address Strobe High		34	ns
bcasl	Bus Clock to Address Strobe Low		44	ns
aswo	Address Strobe Width Out	bch		ns
bcadv	Bus Clock to Address Valid		45	ns
bcadz	Bus Clock to Address TRI-STATE (Note 3)	15	55	ns
ads	Address Setup to ADS0/1 Low	bch − 15		ns
adh	Address Hold from ADS0/1 Low	bcl − 5		ns

Note 1: Cycles T1', T2', T3', T4' are only issued for the first transfer in a burst when 32-bit mode has been selected.

Note 2: The rate of bus clock must be high enough to support transfers to/from the FIFO at a rate greater than the serial network transfers from/to the FIFO.

Note 3: These limits include the RC delay inherent in our test method. These signals typically turn off within 15 ns, enabling other devices to drive these lines with no contention.

15.0 Switching Characteristics (Continued)

DMA Memory Read
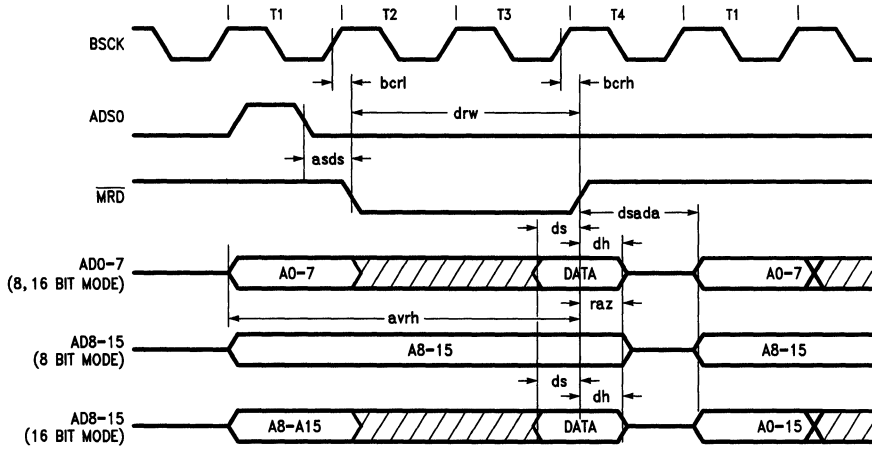
TL/F/8582-82

Symbol	Parameter	Min	Max	Units
bcrl	Bus Clock to Read Strobe Low		43	ns
bcrh	Bus Clock to Read Strobe High		40	ns
ds	Data Setup to Read Strobe High	25		ns
dh	Data Hold from Read Strobe High	0		ns
drw	DMA Read Strobe Width Out	2*bcyc − 15		ns
raz	Memory Read High to Address TRI-STATE (Notes 1, 2)		bch + 40	ns
asds	Address Strobe to Data Strobe		bcl + 10	ns
dsada	Data Strobe to Address Active	bcyc − 10		ns
avrh	Address Valid to Read Strobe High	3*bcyc − 15		ns

Note 1: During a burst A8–A15 are not TRI-STATE if byte wide transfers are selected. On the last transfer A8–A15 are TRI-STATE as shown above.

Note 2: These limits include the RC delay inherent in our test method. These signals typically turn off within bch + 15 ns, enabling other devices to drive these lines with no contention.

15.0 Switching Characteristics (Continued)

DMA Memory Write
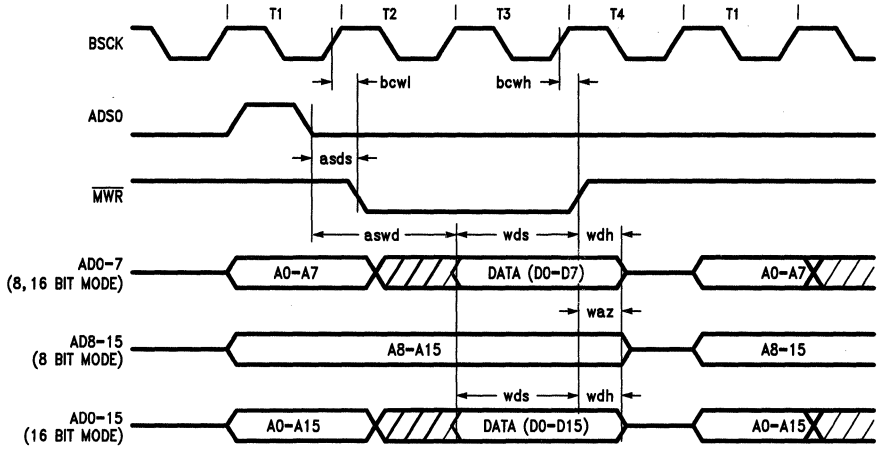
TL/F/8582–83

Symbol	Parameter	Min	Max	Units
bcwl	Bus Clock to Write Strobe Low		40	ns
bcwh	Bus Clock to Write Strobe High		40	ns
wds	Data Setup to $\overline{\text{WR}}$ High	2*bcyc − 30		ns
wdh	Data Hold from $\overline{\text{WR}}$ Low	bch + 7		ns
waz	Write Strobe to Address TRI-STATE (Notes 1, 2)		bch + 40	ns
asds	Address Strobe to Data Strobe		bcl + 10	ns
aswd	Address Strobe to Write Data Valid		bcl + 30	ns

Note 1: When using byte mode transfers A8–A15 are only TRI-STATE on the last transfer, waz timing is only valid for last transfer in a burst.

Note 2: These limits include the RC delay inherent in our test method. These signals typically turn off within bch + 15 ns, enabling other devices to drive these lines with no contention.

15.0 Switching Characteristics (Continued)

Wait State Insertion
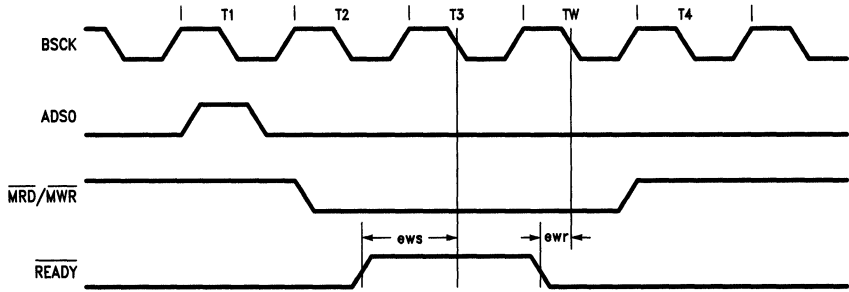
TL/F/8582–45

Symbol	Parameter	Min	Max	Units
ews	External Wait Setup to T3 ↓ Clock (Note 1)	10		ns
ewr	External Wait Release Time (Note 1)	15		ns

Note 1: The addition of wait states affects the count of deserialized bytes and is limited to a number of bus clock cycles depending on the bus clock and network rates. The allowable wait states are found in the table below. (Assumes 10 Mbit/sec data rate.)

BUSCK (MHz)	# of Wait States	
	Byte Transfer	Word Transfer
8	0	1
10	0	1
12	1	2
14	1	2
16	1	3
18	2	3
20	2	4

Table assumes 10 MHz network clock.

The number of allowable wait states in byte mode can be calculated using:

$$\#W_{(byte\ mode)} = \left(\frac{8\ tnw}{4.5\ tbsck} - 1 \right)$$

$\#W$ = Number of Wait States
tnw = Network Clock Period
tbsck = BSCK Period

The number of allowable wait states in word mode can be calculated using:

$$\#W_{(word\ mode)} = \left(\frac{5\ tnw}{2\ tbsck} - 1 \right)$$

1

15.0 Switching Characteristics (Continued)

Remote DMA (Read, Send Command)
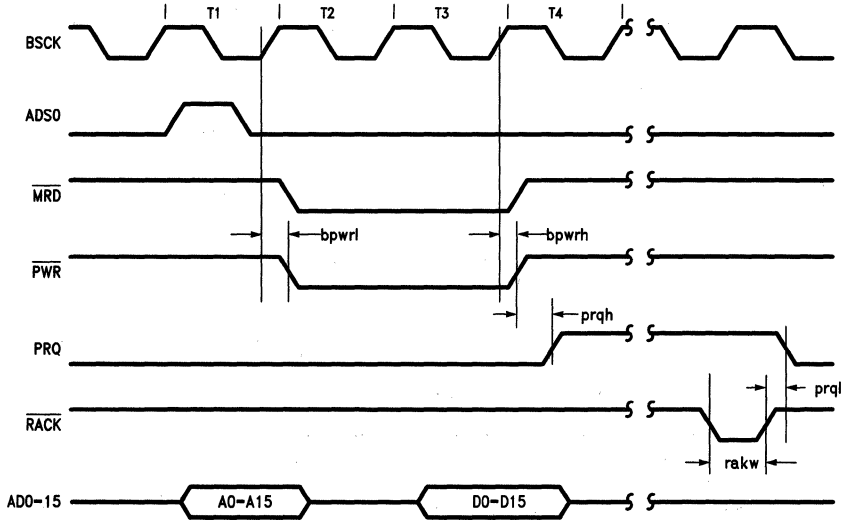
TL/F/8582-84

Symbol	Parameter	Min	Max	Units
bpwrl	Bus Clock to Port Write Low		43	ns
bpwrh	Bus Clock to Port Write High		40	ns
prqh	Port Write High to Port Request High (Note 1)		30	ns
prql	Port Request Low from Read Acknowledge High		45	ns
rakw	Remote Acknowledge Read Strobe Pulse Width	20		ns

Note 1: Start of next transfer is dependent on where $\overline{RACK}$ is generated relative to BSCK and whether a local DMA is pending.

15.0 Switching Characteristics (Continued)

Remote DMA (Read, Send Command) Recovery Time
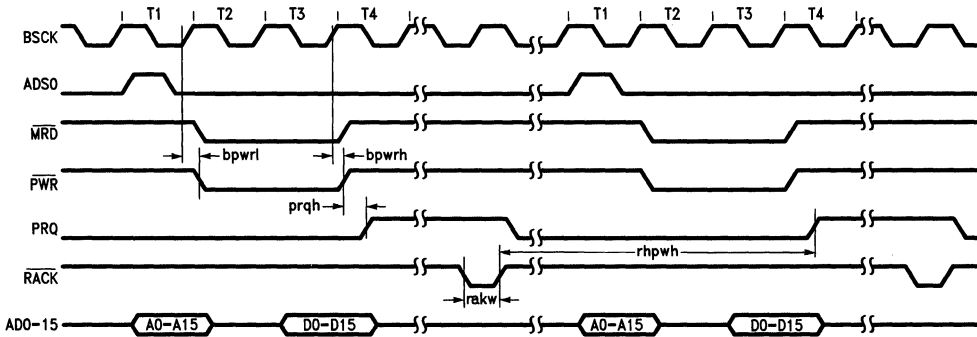
TL/F/8582–85

Symbol	Parameter	Min	Max	Units
bpwrl	Bus Clock to Port Write Low		43	ns
bpwrh	Bus Clock to Port Write High		40	ns
prqh	Port Write High to Port Request High (Note 1)		30	ns
prql	Port Request Low from Read Acknowledge High		45	ns
rakw	Remote Acknowledge Read Strobe Pulse Width	20		ns
rhpwh	Read Acknowledge High to Next Port Write Cycle (Notes 2,3,4)	11		BUSCK

Note 1: Start of next transfer is dependent on where $\overline{RACK}$ is generated relative to BSCK and whether a local DMA is pending.

Note 2: This is not a measured value but guaranteed by design.

Note 3: $\overline{RACK}$ must be high for a minimum of 7 BUSCK.

Note 4: Assumes no local DMA interleave, no $\overline{CS}$, and immediate BACK.

15.0 Switching Characteristics (Continued)

Remote DMA (Write Cycle)
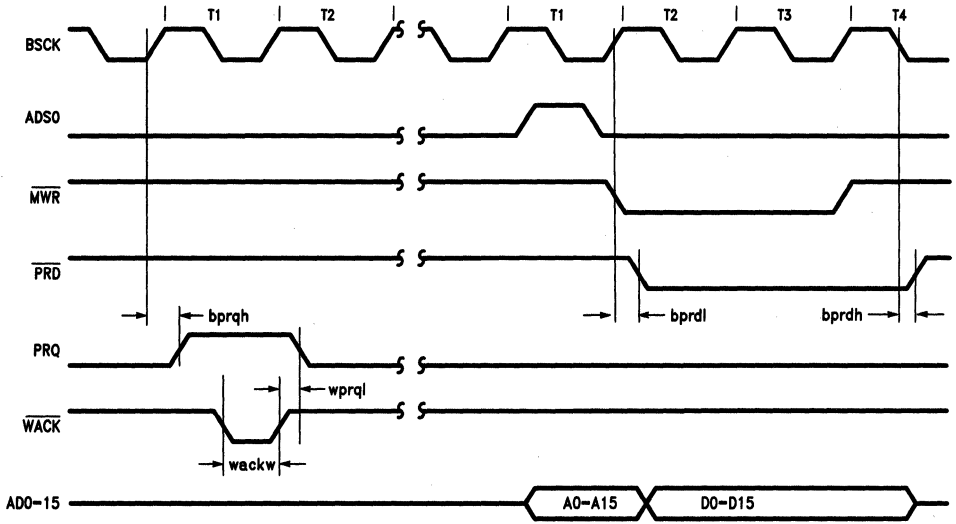
TL/F/8582–86

Symbol	Parameter	Min	Max	Units
bprqh	Bus Clock to Port Request High (Note 1)		42	ns
wprql	$\overline{\text{WACK}}$ to Port Request Low		45	ns
wackw	$\overline{\text{WACK}}$ Pulse Width	20		ns
bprdl	Bus Clock to Port Read Low (Note 2)		40	ns
bprdh	Bus Clock to Port Read High		40	ns

Note 1: The first port request is issued in response to the remote write command. It is subsequently issued on T1 clock cycles following completion of remote DMA cycles.

Note 2: The start of the remote DMA write following $\overline{\text{WACK}}$ is dependent on where $\overline{\text{WACK}}$ is issued relative to BUSCK and whether a local DMA is pending.

15.0 Switching Characteristics (Continued)

Remote DMA (Write Cycle) Recovery Time
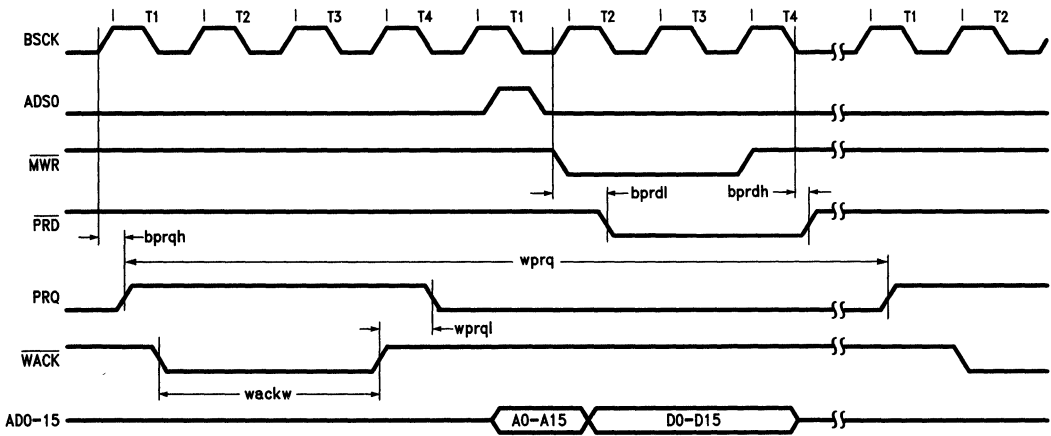
TL/F/8582-87

Symbol	Parameter	Min	Max	Units
bprqh	Bus Clock to Port Request High (Note 1)		40	ns
wprql	$\overline{\text{WACK}}$ to Port Request Low		45	ns
wackw	$\overline{\text{WACK}}$ Pulse Width	20		ns
bprdl	Bus Clock to Port Read Low (Note 2)		40	ns
bprdh	Bus Clock to Port Read High		40	ns
wprq	Remote Write Port Request to Port Request Time (Notes 3,4,5)	12		BUSCK

Note 1: The first port request is issued in response to the remote write command. It is subsequently issued on T1 clock cycles following completion of remote DMA cycles.

Note 2: The start of the remote DMA write following $\overline{\text{WACK}}$ is dependent on where $\overline{\text{WACK}}$ is issued relative to BUSCK and whether a local DMA is pending.

Note 3: Assuming wackw < 1 BUSCK, and no local DMA interleave, no $\overline{\text{CS}}$, immediate $\overline{\text{BACK}}$, and $\overline{\text{WACK}}$ goes high before T4.

Note 4: $\overline{\text{WACK}}$ must be high for a minimum of 7 BUSCK.

Note 5: This is not a measured value but guaranteed by design.

15.0 Switching Characteristics (Continued)

Serial Timing—Receive (Beginning of Frame)
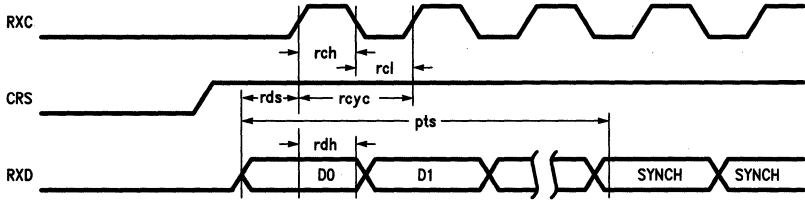
TL/F/8582–88

Symbol	Parameter	Min	Max	Units
rch	Receive Clock High Time	40		ns
rcl	Receive Clock Low Time	40		ns
rcyc	Receive Clock Cycle Time	800	1200	ns
rds	Receive Data Setup Time to Receive Clock High (Note 1)	20		ns
rdh	Receive Data Hold Time from Receive Clock High	17		ns
pts	First Preamble Bit to Synch (Note 2)	8		rcyc cycles

Note 1: All bits entering NIC must be properly decoded, if the PLL is still locking, the clock to the NIC should be disabled or CRS delayed. Any two sequential 1 data bits will be interpreted as Synch.

Note 2: This is a minimum requirement which allows reception of a packet.

Serial Timing—Receive (End of Frame)
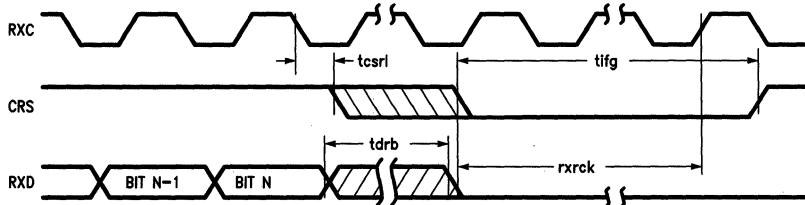
TL/F/8582–89

Symbol	Parameter	Min	Max	Units
rxrck	Minimum Number of Receive Clocks after CRS Low (Note 1)	5	–	rcyc cycles
tdrb	Maximum of Allowed Dribble Bits/Clocks (Note 2)		3	rcyc cycles
tifg	Receive Recovery Time (Notes 4,5)		40	rcyc cycles
tcrsl	Receive Clock to Carrier Sense Low (Note 3)	0	1	rcyc cycles

Note 1: The NIC requires a minimum number of receive clocks following the de-assertion of carrier sense (CRS). These additional clocks are provided by the DP8391 SNI. If other decoder/PLLs are being used additional clocks should be provided. Short clocks or glitches are not allowed.

Note 2: Up to 5 bits of dribble bits can be tolerated without resulting in a receive error.

Note 3: Guarantees to only load bit N, additional bits up to tdrb can be tolerated.

Note 4: This is the time required for the receive state machine to complete end of receive processing. This parameter is not measured but is guaranteed by design. This is not a measured parameter but is a design requirement.

Note 5: CRS must remain de-asserted for a minimum of 2 RXC cycles to be recognized as end of carrier.

15.0 Switching Characteristics (Continued)

Serial Timing—Transmit (Beginning of Frame)
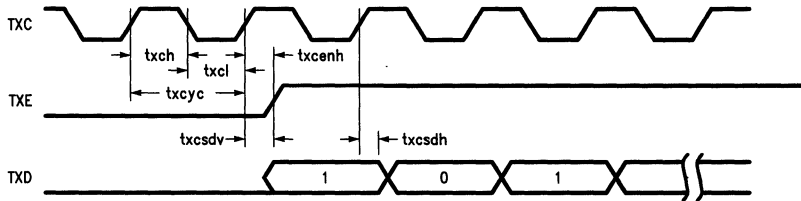
TL/F/8582-90

Symbol	Parameter	Min	Max	Units
txch	Transmit Clock High Time	36		ns
txcl	Transmit Clock Low Time	36		ns
txcyc	Transmit Clock Cycle Time	800	1200	ns
txcenh	Transmit Clock to Transmit Enable High (Note 1)		48	ns
txcsdv	Transmit Clock to Serial Data Valid		67	ns
txcsdh	Serial Data Hold Time from Transmit Clock High	10		ns

Note 1: The NIC issues TXEN coincident with the first bit of preamble. The first bit of preamble is always a 1.

Serial Timing—Transmit (End of Frame, CD Heartbeat)
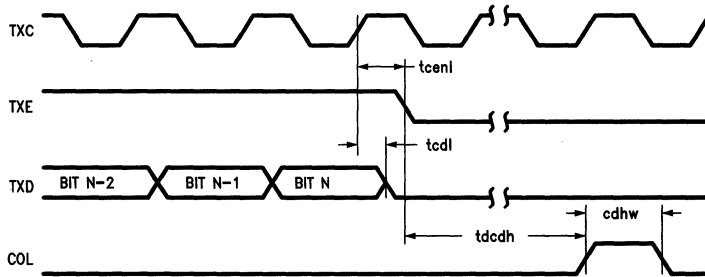
TL/F/8582-91

Symbol	Parameter	Min	Max	Units
tcdl	Transmit Clock to Data Low		55	ns
tcenl	Transmit Clock to TXEN Low		55	ns
tdcdh	TXEN Low to Start of Collision Detect Heartbeat (Note 1)	0	64	txcyc cycles
cdhw	Collision Detect Width	2		txcyc cycles

Note 1: If COL is not seen during the first 64 TX clock cycles following de-assertion of TXEN, the CDH bit in the TSR is set.

15.0 Switching Characteristics (Continued)

Serial Timing—Transmit (Collision)
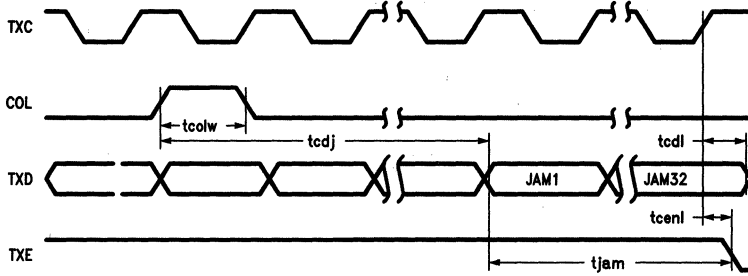
TL/F/8582–92

Symbol	Parameter	Min	Max	Units
tcolw	Collision Detect Width	2		txcyc cycles
tcdj	Delay from Collision to First Bit of Jam (Note 1)		8	txcyc cycles
tjam	Jam Period (Note 2)		32	txcyc cycles

Note 1: The NIC must synchronize to collision detect. If the NIC is in the middle of serializing a byte of data the remainder of the byte will be serialized. Thus the jam pattern will start anywhere from 1 to 8 TXC cycles after COL is asserted.

Note 2: The NIC always issues 32 bits of jam. The jam is all 1's data.

Reset Timing
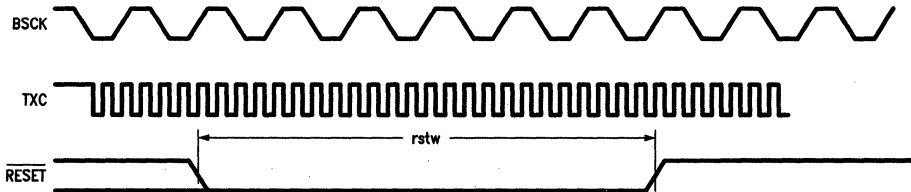
TL/F/8582–93

Symbol	Parameter	Min	Max	Units
rstw	Reset Pulse Width (Note 1)	8		BSCK Cycles or TXC Cycles (Note 2)

Note 1: The RESET pulse requires that BSCK and TXC be stable. On power up, $\overline{RESET}$ should not be raised until BSCK and TXC have become stable. Several registers are affected by $\overline{RESET}$. Consult the register descriptions for details.

Note 2: The slower of BSCK or TXC clocks will determine the minimum time for the $\overline{RESET}$ signal to be low.

If BSCK < TXC then $\overline{RESET}$ = 8 × BSCK

If TXC < BSCK then $\overline{RESET}$ = 8 × TXC

AC Timing Test Conditions

Input Pulse Levels	GND to 3.0V
Input Rise and Fall Times	5 ns
Input and Output Reference Levels	1.3V
TRI-STATE Reference Levels	Float (ΔV) $\pm$0.5V
Output Load (See Figure below)	
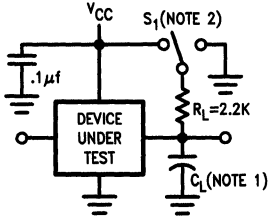
TL/F/8582–94

Note 1: C_L = 50 pF, includes scope and jig capacitance.

Note 2: S1 = Open for timing tests for push pull outputs.

S1 = V_{CC} for V_{OL} test.

S1 = GND for V_{OH} test.

S1 = V_{CC} for High Impedance to active low and active low to High Impedance measurements.

S1 = GND for High Impedance to active high and active high to High Impedance measurements.

Capacitance T_A = 25°C, f = 1 MHz

Parameter	Description	Typ	Max	Unit
C_{IN}	Input Capacitance	7	15	pF
C_{OUT}	Output Capacitance	7	15	pF

Note: This parameter is sampled and not 100% tested.

DERATING FACTOR

Output timings are measured with a purely capacitave load for 50 pF. The following correction factor can be used for other loads:

$C_L \geq$ 50 pf: +0.3 ns/pF (for all outputs except TXE, TXD, and LBK)

National Semiconductor

DP8391A/NS32491A Serial Network Interface

General Description

The DP8391A Serial Network Interface (SNI) provides the Manchester data encoding and decoding functions for IEEE 802.3 Ethernet/Cheapernet type local area networks. The SNI interfaces the DP8390 Network Interface Controller (NIC) to the Ethernet transceiver cable. When transmitting, the SNI converts non-return-to-zero (NRZ) data from the controller and clock pulses into Manchester encoding and sends the converted data differentially to the transceiver. The opposite process occurs on the receive path, where a digital phase-locked loop decodes 10 Mbit/s signals with as much as ±18 ns of jitter.

The DP8391A SNI is a functionally complete Manchester encoder/decoder including ECL like balanced driver and receivers, on board crystal oscillator, collision signal translator, and a diagnostic loopback circuit.

The SNI is part of a three chip set that implements the complete IEEE compatible network node electronics as shown below. The other two chips are the DP8392 Coax Transceiver Interface (CTI) and the DP8390 Network Interface Controller (NIC).

Incorporated into the CTI are the transceiver, collision and jabber functions. The Media Access Protocol and the buffer management tasks are performed by the NIC. There is an isolation requirement on signal and power lines between the CTI and the SNI. This is usually accomplished by using a set of miniature pulse transformers that come in a 16-pin plastic DIP for signal lines. Power isolation, however, is done by using a DC to DC converter.

Features

■ Compatible with Ethernet II, IEEE 802.3 10base5 and 10base2 (Cheapernet)

■ 10 Mb/s Manchester encoding/decoding with receive clock recovery

■ Patented digital phase locked loop (DPLL) decoder requires no precision external components

■ Decodes Manchester data with up to ±18 ns of jitter

■ Loopback capability for diagnostics

■ Externally selectable half or full step modes of operation at transmit output

■ Squelch circuits at the receive and collision inputs reject noise

■ High voltage protection at transceiver interface (16V)

■ TTL/MOS compatible controller interface

■ Connects directly to the transceiver (AUI) cable

Table of Contents

1.0 System Diagram

IEEE 802.3 Compatible Ethernet/Cheapernet Local Area Network Chip Set
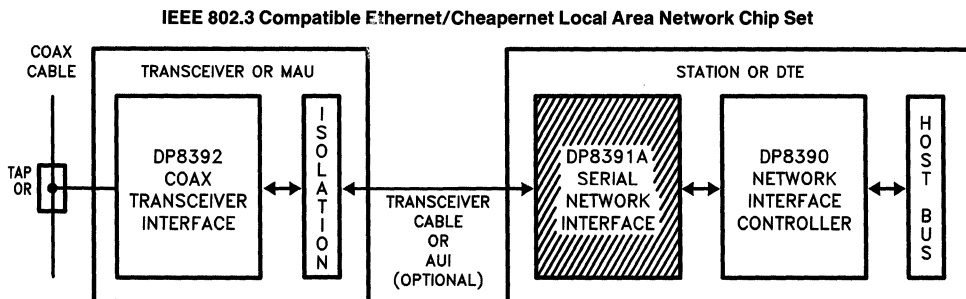
TL/F/9357–1

2.0 Block Diagram
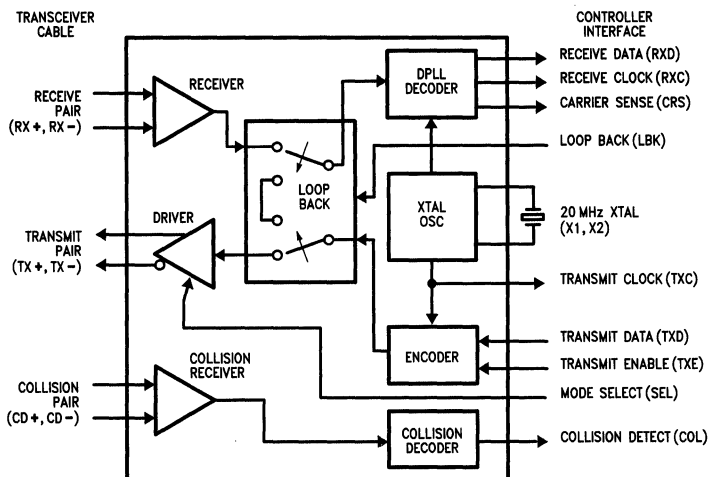
TL/F/9357-2

FIGURE 1

3.0 Functional Description

The SNI consists of five main logical blocks:

a) the oscillator—generates the 10 MHz transmit clock signal for system timing.

b) the Manchester encoder and differential output driver—accepts NRZ data from the controller, performs Manchester encoding, and transmits it differentially to the transceiver.

c) the Manchester decoder—receives Manchester data from the transceiver, converts it to NRZ data and clock pulses, and sends them to the controller.

d) the collision translator—indicates to the controller the presence of a valid 10 MHz signal at its input.

e) the loopback circuitry—when asserted, switches encoded data instead of receive input signals to the digital phase-locked loop.

3.1 OSCILLATOR

The oscillator is controlled by a 20 MHz parallel resonant crystal connected between X1 and X2 or by an external clock on X1. The 20 MHz output of the oscillator is divided by 2 to generate the 10 MHz transmit clock for the controller. The oscillator also provides internal clock signals to the encoding and decoding circuits.

Crystal Specification

Resonant frequency	20 MHz
Tolerance	$\pm 0.001\%$ at 25°C
Stability	$\pm 0.005\%$ 0–70°C
Type	AT-Cut
Circuit	Parallel Resonance

The 20 MHz crystal connection to the SNI requires special care. The IEEE 802.3 standard requires a 0.01% absolute accuracy on the transmitted signal frequency. Stray capacitance can shift the crystal's frequency out of range, causing

the transmitted frequency to exceed its 0.01% tolerance. The frequency marked on the crystal is usually measured with a fixed shunt capacitance (C_L) that is specified in the crystal's data sheet. This capacitance for 20 MHz crystals is typically 20 pF. The capacitance between the X1 and X2 pins of the SNI, of the PC board traces and the plated through holes plus any stray capacitance such as the socket capacitance, if one is used, should be estimated or measured. Once the total sum of these capacitances is determined, the value of additional external shunt capacitance required can be calculated. This capacitor can be a fixed 5% tolerance component. The frequency accuracy should be measured during the design phase at the transmit clock pin (TXC) for a given pc layout. *Figure 2* shows the crystal connection.
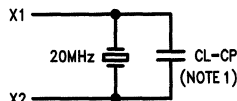
TL/F/9357-3

CL = Load capacitance specified by the crystal's manufacturer

CP = Total parasitic capacitance including:
  a) SNI input capacitance between X1 and X2 (typically 5 pF)
  b) PC board traces, plated through holes, socket capacitances

Note 1: When using a Viking (San Jose) VXB49N5 crystal, the external capacitor is not required, as the C_L of the crystal matches the input capacitance of the DP8391A.

FIGURE 2. Crystal Connection

3.2 MANCHESTER ENCODER AND DIFFERENTIAL DRIVER

The encoder combines clock and data information for the transceiver. Data encoding and transmission begins with the transmit enable input (TXE) going high. As long as TXE re-

3.0 Functional Description (Continued)

mains high, transmit data (TXD) is encoded out to the transmit-driver pair (TX±). The transmit enable and transmit data inputs must meet the setup and hold time requirements with respect to the rising edge of transmit clock. Transmission ends with the transmit enable input going low. The last transition is always positive at the transmit output pair. It will occur at the center of the bit cell if the last bit is one, or at the boundary of the bit cell if the last bit is zero.

The differential line driver provides ECL like signals to the transceiver with typically 5 ns rise and fall times. It can drive up to 50 meters of twisted pair AUI Ethernet transceiver cable. These outputs are source followers which need external 270Ω pulldown resistors to ground. Two different modes, full-step or half-step, can be selected with SEL input. With SEL low, transmit + is positive with respect to transmit − in the idle state. With SEL high, transmit + and transmit − are equal in the idle state, providing zero differential voltage to operate with transformer coupled loads. *Figures 4, 5* and *6* illustrate the transmit timing.

3.3 MANCHESTER DECODER

The decoder consists of a differential input circuitry and a digital phase-locked loop to separate Manchester encoded data stream into clock signals and NRZ data. The differential input should be externally terminated if the standard 78Ω transceiver drop cable is used. Two 39Ω resistors connected in series and one optional common mode bypass capacitor would accomplish this. A squelch circuit at the input rejects signals with pulse widths less than 5 ns (negative going), or with levels less than −175 mV. Signals more negative than −300 mV and with a duration greater than 30 ns are always decoded. This prevents noise at the input from falsely triggering the decoder in the absence of a valid signal. Once the input exceeds the squelch requirements,

carrier sense (CRS) is asserted. Receive data (RXD) and receive clock (RXC) become available typically within 6 bit times. At this point the digital phase-locked loop has locked to the incoming signal. The DP8391A decodes a data frame with up to ±18 ns of jitter correctly.

The decoder detects the end of a frame when the normal mid-bit transition on the differential input ceases. Within one and a half bit times after the last bit, carrier sense is de-asserted. Receive clock stays active for five more bit times before it goes low and remains low until the next frame. *Figures 7, 8* and *9* illustrate the receive timing.

3.4 COLLISION TRANSLATOR

The Ethernet transceiver detects collisions on the coax cable and generates a 10 MHz signal on the transceiver cable. The SNI's collision translator asserts the collision detect output (COL) to the DP8390 controller when a 10 MHz signal is present at the collision inputs. The controller uses this signal to back off transmission and recycle itself. The collision detect output is de-asserted within 350 ns after the 10 MHz input signal disappears.

The collision differential inputs (+ and −) should be terminated in exactly the same way as the receive inputs. The collision input also has a squelch circuit that rejects signals with pulse widths less than 5 ns (negative going), or with levels less than −175 mV. *Figure 10* illustrates the collision timing.

3.5 LOOPBACK FUNCTIONS

Logic high at loopback input (LBK) causes the SNI to route serial data from the transmit data input, through its encoder, returning it through the phase-locked-loop decoder to receive data output. In loopback mode, the transmit driver is in idle state and the receive and collision input circuitries are disabled.

4.0 Connection Diagram

Top View

FIGURE 3a

Order Number DP8391AN
See NS Package Number N24C

*Refer to the Oscillator section

TL/F/9357−4

PCC Connection Diagram

COLLISION PAIR

RECEIVE PAIR

39Ω

39Ω

0.01 μF

39Ω

39Ω

0.01 μF

RXC CRS RXD COL CD+ CD− RX+

4   3   2   1   28 27 26

SEL   5   25   RX−

GND   6   24   NC

      7   23   V_CC

      8   DP8391A   22

      9   SNI   21

LBK   10   20

X1   11   19   CAP

      12 13 14 15 16 17 18

      X2 TXD TXC TXE TX− TX+ NC

CL − CP *

20 MHz

0.001 μF MINIMUM

270Ω   270Ω

TL/F/9357−5

*Refer to the Oscillator section

FIGURE 3b
Order Number DP8391AV
NS Package Number V28A

1

5.0 Pin Descriptions

Pin No. (DIP)	Pin No. (PCC)	Name	I/O	Description
1	1	COL	O	**Collision Detect Output.** A TTL/MOS level active high output. A 10 MHz (+25%−15%) signal at the collision input will produce a logic high at COL output. When no signal is present at the collision input, COL output will go low.
2	2	RXD	O	**Receive Data Output.** A TTL/MOS level signal. This is the NRZ data output from the digital phase-locked loop. This signal should be sampled by the controller at the rising edge of receive clock.
3	3	CRS	O	**Carrier Sense.** A TTL/MOS level active high signal. It is asserted when valid data from the transceiver is present at the receive input. It is de-asserted one and a half bit times after the last bit at receive input.
4	4	RXC	O	**Receive Clock.** A TTL/MOS level recovered clock. When the phase-locked loop locks to a valid incoming signal a 10 MHz clock signal is activated on this output. This output remains low during idle (5 bit times after activity ceases at receive input).
5	5	SEL	I	**Mode Select.** A TTL level input. When high, transmit + and transmit − outputs are at the same voltage in idle state providing a "zero" differential. When low, transmit + is positive with respect to transmit − in idle state.
6	6–9	GND		**Negative Supply Pins.**
7	10	LBK	I	**Loopback.** A TTL level active high on this input enables the loopback mode.
8	11	X1	I	**Crystal or External Frequency Source Input (TTL).**
9	12	X2	O	**Crystal Feedback Output.** This output is used in the crystal connection only. It must be left open when driving X1 with an external frequency source.
10	13	TXD	I	**Transmit Data.** A TTL level input. This signal is sampled by the SNI at the rising edge of transmit clock when transmit enable input is high. The SNI combines transmit data and transmit clock signals into a Manchester encoded bit stream and sends it differentially to the transceiver.
11	14	TXC	O	**Transmit Clock.** A TTL/MOS level 10 MHz clock signal derived from the 20 MHz oscillator. This clock signal is always active.
12	15	TXE	I	**Transmit Enable.** A TTL level active high data encoder enable input. This signal is also sampled by the SNI at the rising edge of transmit clock.
13 14	16 17	TX− TX+	O	**Transmit Output.** Differential line driver which sends the encoded data to the transceiver. These outputs are source followers and require 270Ω pulldown resistors to GND.
15 16	18	NC		**No Connection.**
17	19	CAP	O	**Bypass Capacitor.** A ceramic capacitor (greater than 0.001 μF) must be connected from this pin to GND.
18 19	20–23	VCC		**Positive Supply Pins.** A 0.1 μF ceramic decoupling capacitor must be connected across VCC and GND as close to the device as possible.
20	24	NC		**No Connection.**
21 22	25 26	RX− RX+	I	**Receive Input.** Differential receive input pair from the transceiver.
23 24	27 28	CD− CD+	I	**Collision Input.** Differential collision input pair from the transceiver.

6.0 Absolute Maximum Ratings

Supply Voltage (V_{CC})	7V
Input Voltage (TTL)	0 to 5.5V
Input Voltage (differential)	-5.5 to $+16V$
Output Voltage (differential)	0 to 16V
Output Current (differential)	-40 mA
Storage Temperature	$-65°$ to 150°C
Lead Temperature (soldering, 10 sec)	300°C
Package Power Rating for DIP at 25°C	2.95W*
(PC Board Mounted)	
Derate Linearly at the rate of 23.8 mW/°C	
Package Power Rating for PCC at 25°C	1.92W*
Derate Linearly at the rate of 15.4 mW/°C	

*For actual power dissipation of the device please refer to Section 7.0.

ESD rating	2000V

Recommended Operating Conditions

Supply Voltage (V_{CC})	5V ± 5%
Ambient Temperature (DIP)	0° to 70°C
(PCC)	0° to 55°C

Note: *Absolute maximum ratings are those values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits.*

7.0 Electrical Characteristics

$V_{CC} = 5V \pm 5\%$, $T_A = 0°C$ to 70°C for DIP and 0°C to 55°C for PCC (Notes 1 & 2)

Symbol	Parameter	Test Conditions	Min	Max	Units
V_{IH}	Input High Voltage (TTL)		2.0		V
V_{IHX1a}	Input High Voltage (X1)	No Series Resistor	2.0	$V_{CC}-1.5$	V
V_{IHX1b}	Input High Voltage (X1)	1k Series Resistor	2.0	V_{CC}	V
V_{IL}	Input Low Voltage (TTL and X1)			0.8	V
I_{IH}	Input High Current (TTL) Input High Current (RX± CD±)	$V_{IN} = V_{CC}$ $V_{IN} = V_{CC}$		50 500	µA µA
I_{IL}	Input Low Current (TTL) Input Low Current (RX± CD±)	$V_{IN} = 0.5V$ $V_{IN} = 0.5V$		-300 -700	µA µA
V_{CL}	Input Clamp Voltage (TTL)	$I_{IN} = -12$ mA		-1.2	V
V_{OH}	Ouptut High Voltage (TTL/MOS)	$I_{OH} = -100$ µA	3.5		V
V_{OL}	Output Low Voltage (TTL/MOS)	$I_{OL} = 8$ mA		0.5	V
I_{OS}	Output Short Circuit Current (TTL/MOS)		-40	-200	mA
V_{OD}	Differential Output Voltage (TX±)	78Ω termination, and 270Ω from each to GND	±550	±1200	mV
V_{OB}	Diff. Output Voltage Imbalance (TX±)	same as above		±40	mV
V_{DS}	Diff. Squelch Threshold (RX± CD±)		-175	-300	mV
V_{CM}	Diff. Input Common Mode Voltage (RX± CD±)		-5.25	5.25	V
I_{CC}	Power Supply Current	10Mbit/s		270	mA

8.0 Switching Characteristics

$V_{CC} = 5V \pm 5\%$, $T_A = 0°C$ to 70°C for DIP and 0°C to 55°C for PCC (Note 2)

Symbol	Parameter	Figure	Min	Typ	Max	Units
OSCILLATOR SPECIFICATION						
t_{XTH}	X1 to Transmit Clock High	12	8		20	ns
t_{XTL}	X1 to Transmit Clock Low	12	8		20	ns
TRANSMIT SPECIFICATION						
t_{TCd}	Transmit Clock Duty Cycle at 50% (10 MHz)	12	42	50	58	%
t_{TCr}	Transmit Clock Rise Time (20% to 80%)	12			8	ns
t_{TCf}	Transmit Clock Fall Time (80% to 20%)	12			8	ns
t_{TDs}	Transmit Data Setup Time to Transmit Clock Rising Edge	4 & 12	20			ns
t_{TDh}	Transmit Data Hold Time from Transmit Clock Rising Edge	4 & 12	0			ns
t_{TEs}	Transmit Enable Setup Time to Trans. Clock Rising Edge	4 & 12	20			ns
t_{TEh}	Transmit Enable Hold Time from Trans. Clock Rising Edge	5 & 12	0			ns

Note 1: All currents into device pins are positive, all currents out of device pins are negative. All voltages are referenced to ground unless otherwise specified.

Note 2: All typicals are given for $V_{CC} = 5V$ and $T_A = 25°C$.

1

8.0 Switching Characteristics

V_{CC} = 5V ±5%, T_A = 0°C to 70°C for DIP and 0°C to 55°C for PCC (Note 2) (Continued)

Symbol	Parameter	Figure	Min	Typ	Max	Units
TRANSMIT SPECIFICATION (Continued)						
t_{TOd}	Transmit Output Delay from Transmit Clock Rising Edge	4 & 12			40	ns
t_{TOr}	Transmit Output Rise Time (20% to 80%)	12			7	ns
t_{TOf}	Transmit Output Fall Time (80% to 20%)	12			7	ns
t_{TOj}	Transmit Output Jitter	12		±0.25		ns
t_{TOh}	Transmit Output High Before Idle in Half Step Mode	5 & 12	200			ns
t_{TOi}	Transmit Output Idle Time in Half Step Mode	5 & 12			800	ns
RECEIVE SPECIFICATION						
t_{RCd}	Receive Clock Duty Cycle at 50% (10 MHz)	12	40	50	60	%
t_{RCr}	Receive Clock Rise Time (20% to 80%)	12			8	ns
t_{RCf}	Receive Clock Fall Time (80% to 20%)	12			8	ns
t_{RDr}	Receive Data Rise Time (20% to 80%)	12			8	ns
t_{RDf}	Receive Data Fall Time (80% to 20%)	12			8	ns
t_{RDs}	Receive Data Stable from Receive Clock Rising Edge	7 & 12	±40			ns
t_{CSon}	Carrier Sense Turn On Delay	7 & 12			50	ns
t_{CSoff}	Carrier Sense Turn Off Delay	8, 9 & 12			160	ns
t_{DAT}	Decoder Acquisition Time	7		0.6	1.80	μs
t_{Drej}	Differential Inputs Rejection Pulse Width (Squelch)	7	5		30	ns
t_{Rd}	Receive Throughput Delay	8 & 12			150	ns
COLLISION SPECIFICATION						
t_{COLon}	Collision Turn On Delay	10 & 12			50	ns
t_{COLoff}	Collision Turn Off Delay	10 & 12			350	ns
LOOPBACK SPECIFICATION						
t_{LBs}	Loopback Setup Time	11	20			ns
t_{LBh}	Loopback Hold Time	11	0			ns

Note 2: All typicals are given for V_{CC} = 5V and T_A = 25°C.

9.0 Timing and Load Diagrams
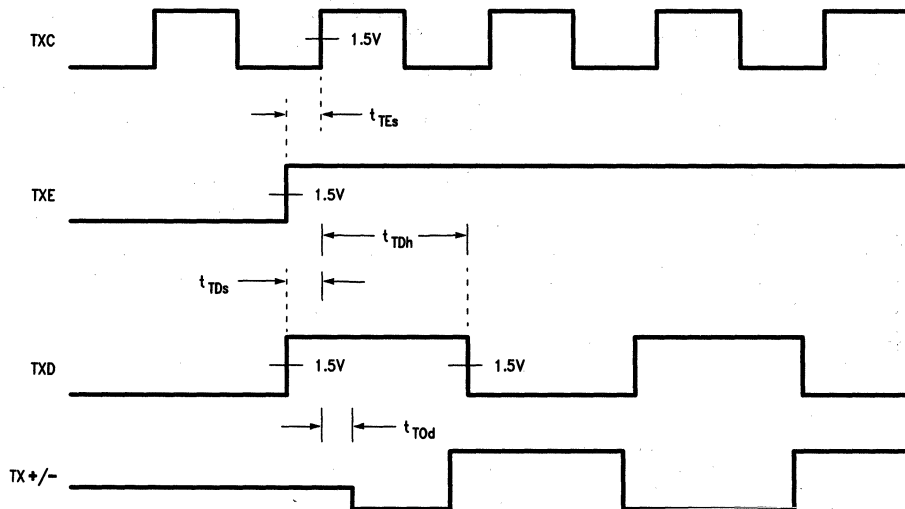
TL/F/9357–6

FIGURE 4. Transmit Timing - Start of Transmission

9.0 Timing and Load Diagrams (Continued)
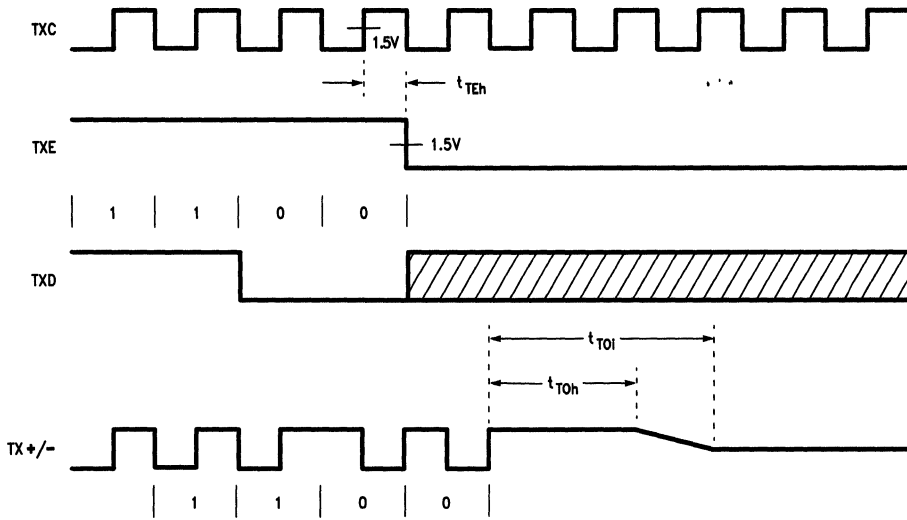
TL/F/9357–7

FIGURE 5. Transmit Timing - End of Transmission (last bit = 0)

TL/F/9357–8
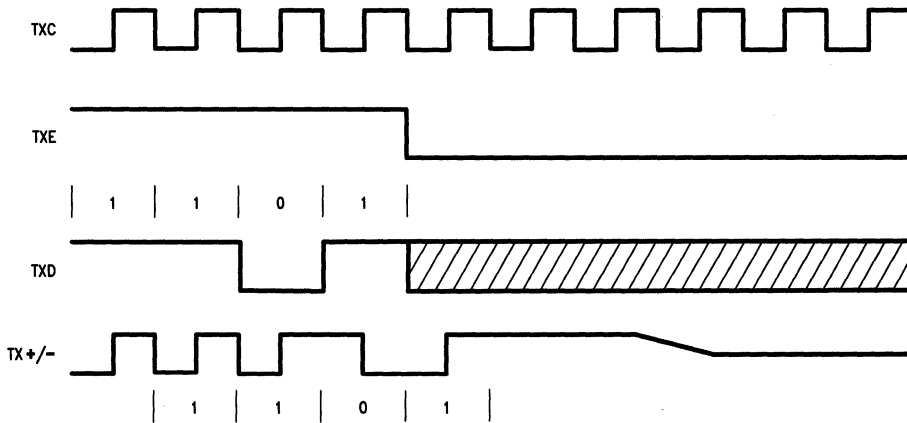
FIGURE 6. Transmit Timing - End of Transmission (last bit = 1)

1

9.0 Timing and Load Diagrams (Continued)

FIRST BIT DECODED

RX +/−

$t_{D\,rej}$

CRS

1.5v

$t_{CS\,on}$

RXC

1.5V

t_{DAT}

t_{RDs}

t_{RDs}

RXD

1.5v

| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
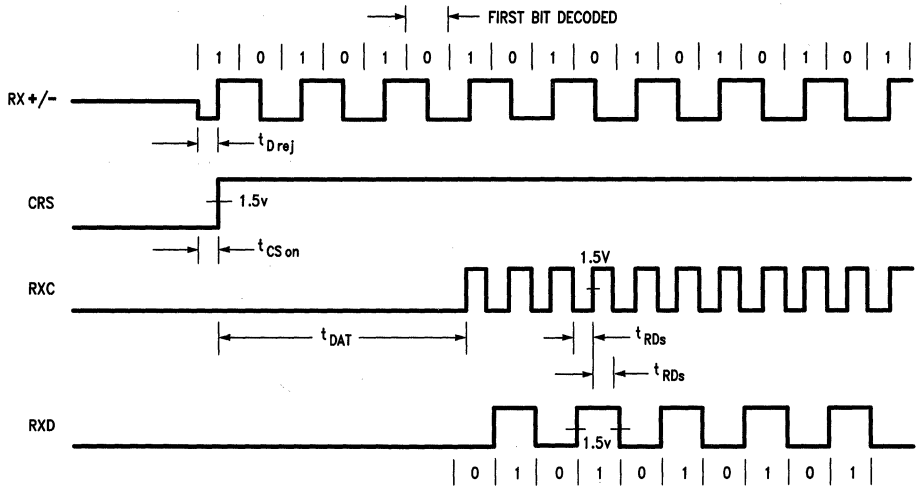
TL/F/9357−9

FIGURE 7. Receive Timing - Start of Packet

| 1 | 0 | 1 | 0 | 0 |

RX +/−

$t_{CS\,off}$

CRS

1.5v

t_{Rd}

RXC

5 EXTRA CLOCKS

RXD

| 1 | 0 | 1 | 0 | 0 |
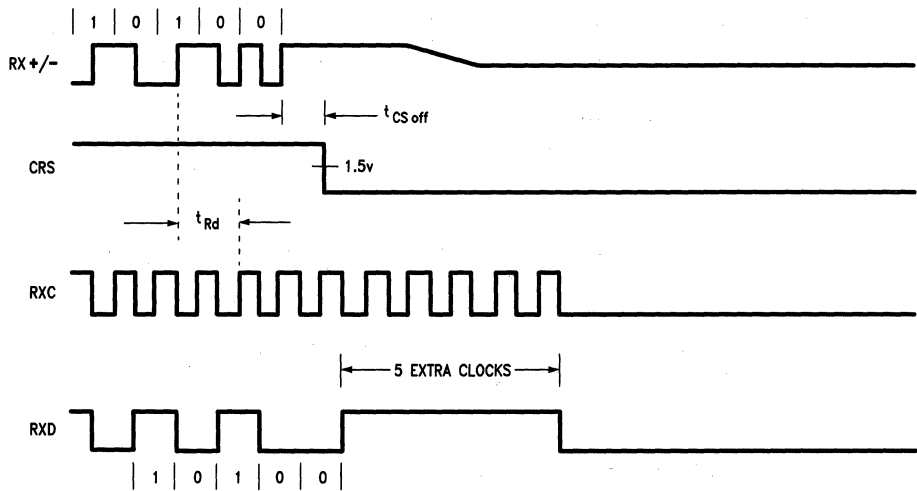
TL/F/9357−10

FIGURE 8. Receive Timing - End of Packet (last bit = 0)

9.0 Timing and Load Diagrams (Continued)
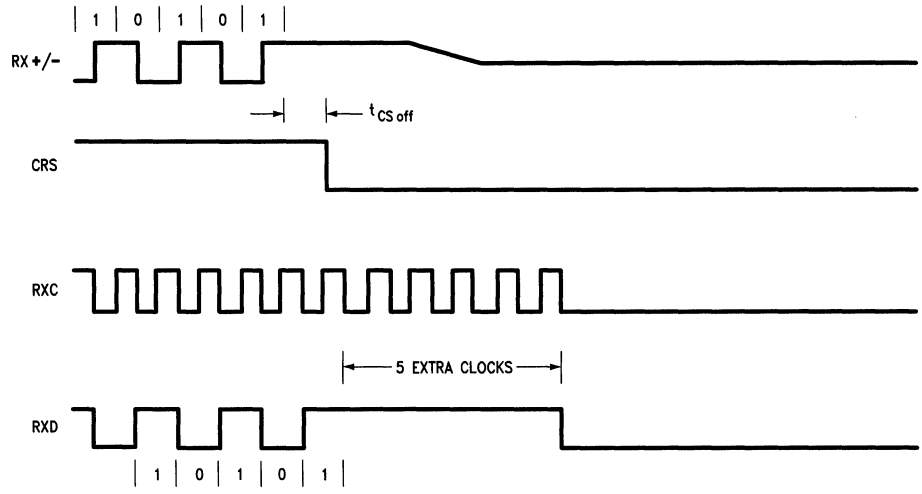
TL/F/9357–11

FIGURE 9. Receive Timing - End of Packet (last bit = 1)

TL/F/9357–12
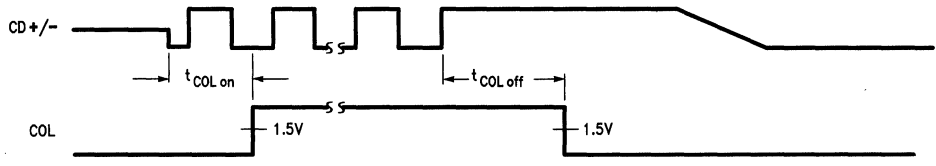
FIGURE 10. Collision Timing

TL/F/9357–13
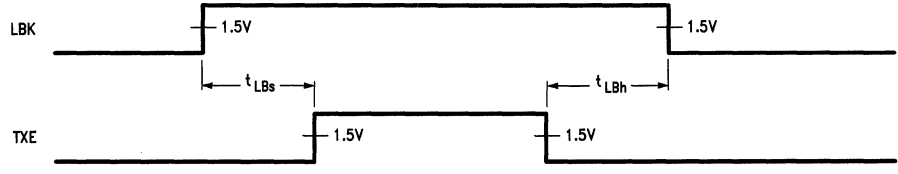
FIGURE 11. Loopback Timing

TL/F/9357–14
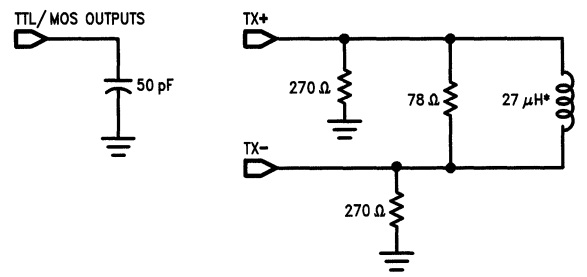
*27 µH transformer is used for testing purposes, 100 µH transformers (Valor, LT1101, or Pulse Engineering 64103) are recommended for application use.

FIGURE 12. Test Loads

National Semiconductor

DP83910
CMOS Serial Network Interface

General Description

The DP83910 CMOS Serial Network Interface (SNI) is a direct-pin equivalent of the bipolar DP8391 SNI and provides the Manchester data encoding and decoding functions for IEEE 802.3 Ethernet/Thin-Ethernet type local area networks. The SNI interfaces the DP8390 Network Interface Controller (NIC) to the DP8392 CTI or an Ethernet transceiver cable. When transmitting, the SNI converts non-return-to-zero (NRZ) data from the controller into Manchester data and sends the converted data differentially to the transceiver. Conversely, when receiving, a Phase Lock Loop decodes the 10 Mbit/s data from the transceiver into NRZ data for the controller.

The DP83910 operates in conjunction with the DP8392 Coaxial Transceiver Interface (CTI) and the DP8390 Network Interface Controller (NIC) to form a three-chip set that implements a complete IEEE 802.3 compatible network as shown below. The DP83910 is a functionally complete Manchester encoder/decoder including a balanced driver and receiver, on-board crystal oscillator, collision signal translator, and a diagnostic loopback feature. The DP83910, fabricated in low-power microCMOS, typically consumes less than 70 mA of current. However, as a result of being CMOS, the DP83910's differential signals must be isolated in both Ethernet and thin wire Ethernet.

Features

- Compatible with Ethernet I, IEEE 802.3, 10base5, and 10base2 (Thin-Ethernet)
- Designed to interface with 10 baseT transceivers
- Functional and pin-out duplicate of the DP8391
- 10 Mbits/s Manchester encoding/decoding with receive clock recovery
- Requires no precision components
- Decodes Manchester data with up to 20 ns of jitter
- Loopback capability for diagnostics
- Externally selectable half or full step modes of operation at transmit output
- Squelch circuitry at the receive and collision inputs to reject noise
- TTL/MOS compatible controller interface

1.0 System Diagram

IEEE 802.3 Compatible Ethernet/Thin-Ethernet/10 Baset Local Area Network Chip Set

TL/F/9365-1

National Semiconductor
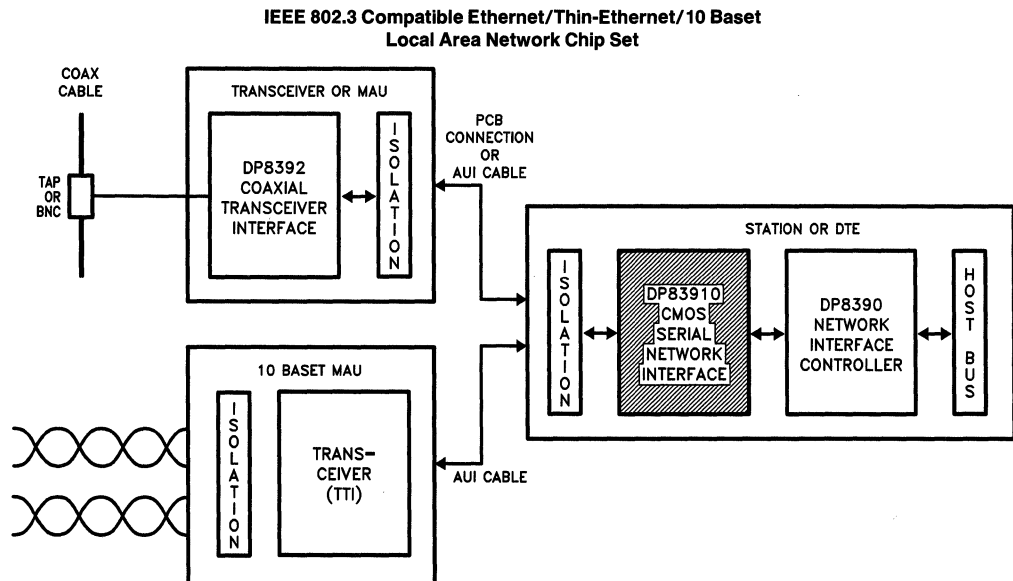
DP8392A/NS32492A Coaxial Transceiver Interface

General Description

The DP8392A Coaxial Transceiver Interface (CTI) is a coaxial cable line driver/receiver for Ethernet/Thin Ethernet (Cheapernet) type local area networks. The CTI is connected between the coaxial cable and the Data Terminal Equipment (DTE). In Ethernet applications the transceiver is usually mounted within a dedicated enclosure and is connected to the DTE via a transceiver cable. In Cheapernet applications, the CTI is typically located within the DTE and connects to the DTE through isolation transformers only. The CTI consists of a Receiver, Transmitter, Collision Detector, and a Jabber Timer. The Transmitter connects directly to a 50 ohm coaxial cable where it is used to drive the coax when transmitting. During transmission, a jabber timer is initiated to disable the CTI transmitter in the event of a longer than legal length data packet. Collision Detection circuitry monitors the signals on the coax to determine the presence of colliding packets and signals the DTE in the event of a collision.

The CTI is part of a three chip set that implements the complete IEEE 802.3 compatible network node electronics as shown below. The other two chips are the DP8391 Serial Network Interface (SNI) and the DP8390 Network Interface Controller (NIC).

The SNI provides the Manchester encoding and decoding functions; whereas the NIC handles the Media Access Protocol and the buffer management tasks. Isolation between the CTI and the SNI is an IEEE 802.3 requirement that can be easily satisfied on signal lines using a set of pulse transformers that come in a standard DIP. However, the power isolation for the CTI is done by DC-to-DC conversion through a power transformer.

Features

■ Compatible with Ethernet II, IEEE 802.3 10Base5 and 10Base2 (Cheapernet)
■ Integrates all transceiver electronics except signal & power isolation
■ Innovative design minimizes external component count
■ Jabber timer function integrated on chip
■ Externally selectable CD Heartbeat allows operation with IEEE 802.3 compatible repeaters
■ Precision circuitry implements receive mode collision detection
■ Squelch circuitry at all inputs rejects noise
■ Designed for rigorous reliability requirements of IEEE 802.3
■ Standard Outline 16-pin DIP uses a special leadframe that significantly reduces the operating die temperature

Table of Contents

1.0 System Diagram

IEEE 802.3 Compatible Ethernet/Cheapernet Local Area Network Chip Set

TL/F/7405–1

2.0 Block Diagram

DTE INTERFACE
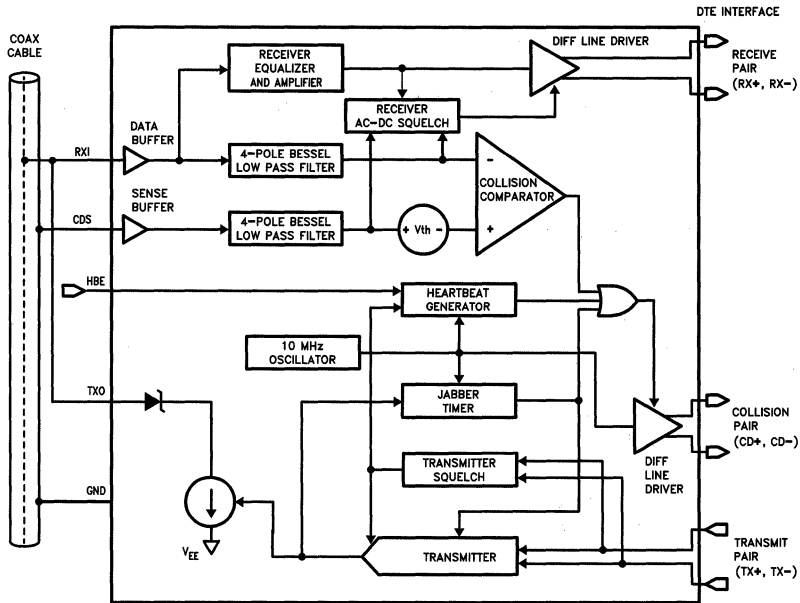
FIGURE 1. DP8392A Block Diagram

TL/F/7405–2

3.0 Functional Description

The CTI consists of four main logical blocks:

a) the Receiver - receives data from the coax and sends it to the DTE

b) the Transmitter - accepts data from the DTE and transmits it onto the coax

c) the Collision Detect circuitry - indicates to the DTE any collision on the coax

d) the Jabber Timer - disables the Transmitter in case of longer than legal length packets

3.1 RECEIVER FUNCTIONS

The Receiver includes an input buffer, a cable equalizer, a 4-pole Bessel low pass filter, a squelch circuit, and a differential line driver.

The buffer provides high input impedance and low input capacitance to minimize loading and reflections on the coax.

The equalizer is a high pass filter which compensates for the low pass effect of the cable. The composite result of the maximum length cable and the equalizer is a flatband response at the signal frequencies to minimize jitter.

The 4-pole Bessel low pass filter extracts the average DC level on the coax, which is used by both the Receiver squelch and the collision detection circuits.

The Receiver squelch circuit prevents noise on the coax from falsely triggering the Receiver in the absence of the signal. At the beginning of the packet, the Receiver turns on when the DC level from the low pass filter is lower than the DC squelch threshold. However, at the end of the packet, a quick Receiver turn off is needed to reject dribble bits. This is accomplished by an AC timing circuit that reacts to high level signals of greater than typically 200 ns in duration. The

Receiver then stays off only if within about 1 μs, the DC level from the low pass filter rises above the DC squelch threshold. *Figure 2* illustrates the Receiver timing.

The differential line driver provides ECL compatible signals to the DTE with typically 3 ns rise and fall times. In its idle state, its outputs go to differential zero to prevent DC standing current in the isolation transformer.

3.2 TRANSMITTER FUNCTIONS

The Transmitter has a differential input and an open collector output current driver. The differential input common mode voltage is established by the CTI and should not be altered by external circuitry. The transformer coupling of TX± will satisfy this condition. The driver meets all IEEE 802.3/Ethernet Specifications for signal levels. Controlled rise and fall times (25 ns V ±5 ns) minimize the higher harmonic components. The rise and fall times are matched to minimize jitter. The drive current levels of the DP8392A meet the tighter recommended limits of IEEE 802.3 and are set by a built-in bandgap reference and an external 1% resistor. An on chip isolation diode is provided to reduce the Transmitter's coax load capacitance. For Ethernet compatible applications, an external isolation diode (see *Figure 4*) may be added to further reduce coax load capacitance. In Cheapernet compatible applications the external diode is not required as the coax capacitive loading specifications are relaxed.

The Transmitter squelch circuit rejects signals with pulse widths less than typically 20 ns (negative going), or with levels less than −175 mV. The Transmitter turns off at the end of the packet if the signal stays higher than −175 mV for more than approximately 300 ns. *Figure 3* illustrates the Transmitter timing.

3.0 Functional Description (Continued)

3.3 COLLISION FUNCTIONS

The collision circuitry consists of two buffers, two 4-pole Bessel low pass filters (section 3.1), a comparator, a heartbeat generator, a 10 MHz oscillator, and a differential line driver.

Two identical buffers and 4-pole Bessel low pass filters extract the DC level on the center conductor (data) and the shield (sense) of the coax. These levels are monitored by the comparator. If the data level is more negative than the sense level by at least the collision threshold (Vth), the collision output is enabled.

At the end of every transmission, the heartbeat generator creates a pseudo collision for a short time to ensure that the collision circuitry is properly functioning. This burst on collision output occurs typically 1.1 μs after the transmission, and has a duration of about 1 μs. This function can be disabled externally with the HBE (Heartbeat Enable) pin to allow operation with repeaters.

The 10 MHz oscillator generates the signal for the collision and heartbeat functions. It is also used as the timebase for all the jabber functions. It does not require any external components.

The collision differential line driver transfers the 10 MHz signal to the CD± pair in the event of collision, jabber, or heartbeat conditions. This line driver also features zero differential idle state.

3.4 JABBER FUNCTIONS

The Jabber Timer monitors the Transmitter and inhibits transmission if the Transmitter is active for longer than 20 ms (fault). It also enables the collision output for the fault duration. After the fault is removed, The Jabber Timer waits for about 500 ms (unjab time) before re-enabling the Transmitter. The transmit input must stay inactive during the unjab time.

FIGURE 2. Receiver Timing

TL/F/7405–3
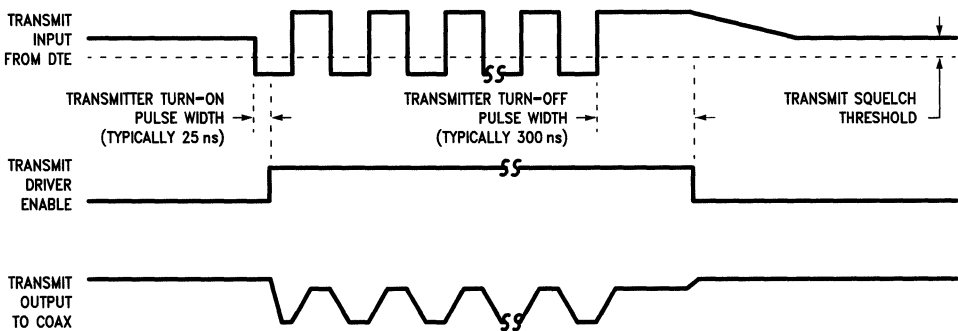
FIGURE 3. Transmitter Timing

TL/F/7405–4

4.0 Connection Diagram
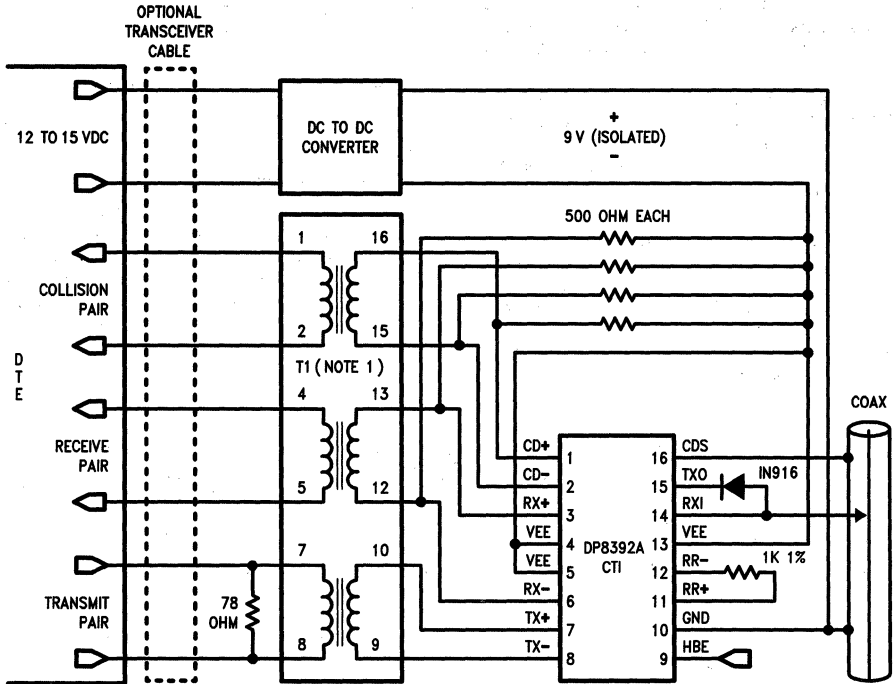
Note 1: T1 is a 1:1 pulse transformer, L = 100 μH
Pulse Engineering (San Diego) Part No. 64103
Valor Electronics (San Diego)
Part No. 1101 or equivalent

TL/F/7405–5

Top View

Order Number DP8392AN
See NS Package Number N16A

FIGURE 4

5.0 Pin Descriptions

Pin No.	Name	I/O	Description
1 2	CD+* CD−	O	**Collision Output.** Balanced differential line driver outputs from the collision detect circuitry. The 10 MHz signal from the internal oscillator is transferred to these outputs in the event of collision, excessive transmission (jabber), or during CD Heartbeat condition. These outputs are open emitters; pulldown resistors to VEE are required. When operating into a 78Ω transmission line, these resistors should be 500Ω. In Cheapernet applications, where the 78Ω drop cable is not used, higher resistor values (up to 1.5k) may be used to save power.
3 6	RX+* RX−	O	**Receive Output.** Balanced differential line driver outputs from the Receiver. These outputs also require 500Ω pulldown resistors.
7 8	TX+* TX−	I	**Transmit Input.** Balanced differential line receiver inputs to the Transmitter. The common mode voltage for these inputs is determined internally and must not be externally established. Signals meeting Transmitter squelch requirements are waveshaped and output at TXO.
9	HBE	I	**Heartbeat Enable.** This input enables CD Heartbeat when grounded, disables it when connected to VEE.
11 12	RR+ RR−	I	**External Resistor.** A fixed 1k 1% resistor connected between these pins establishes internal operating currents.
14	RXI	I	**Receive Input.** Connects directly to the coaxial cable. Signals meeting Receiver squelch requirements are equalized for inter-symbol distortion, amplified, and outputted at RX±.
15	TXO	O	**Transmit Output.** Connects either directly (Cheapernet) or via an isolation diode (Ethernet) to the coaxial cable.
16	CDS	I	**Collision Detect Sense.** Ground sense connection for the collision detect circuit. This pin should be connected separately to the shield to avoid ground drops from altering the receive mode collision threshold.
10	GND		**Positive Supply Pin.** A 0.1 μF ceramic decoupling capacitor must be connected across GND and VEE as close to the device as possible.
4 5 13	VEE		**Negative Supply Pins.** In order to make full use of the 3.5W power dissipation capability of this package, these pins should be connected to a large metal frame area on the PC board. Doing this will reduce the operating die temperature of the device thereby increasing the long term reliability.

* IEEE names for CD± = CI±, RX± = DI±, TX± = DO±

5.1 P.C. BOARD LAYOUT

The DP8392A package is uniquely designed to ensure that the device meets the 1 million hour Mean Time Between Failure (MTBF) requirement of the IEEE 802.3 standard. In order to fully utilize this heat dissipation design, the three VEE pins are to be connected to a copper plane which should be included in the printed circuit board layout. Refer to National Semiconductor application note AN-442 (Ethernet/Cheapernet Physical Layer Made Easy) for complete board layout instructions.

1

6.0 Absolute Maximum Ratings (Note 1)

Supply Voltage (V_{EE})   −12V

Package Power Rating at 25°C   3.5 Watts*
(PC Board Mounted)   See Section 5
Derate linearly at the rate of 28.6 mW/°C

Input Voltage   0 to −12V

Storage Temperature   −65° to 150°C

Lead Temp. (Soldering, 10 seconds)   260°C

*For actual power dissipation of the device please refer to section 7.0.

Recommended Operating Conditions

Supply Voltage (V_{EE})   −9v ±5%

Ambient Temperature   0° to 70°C

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

7.0 Electrical Characteristics $V_{EE} = -9V \pm 5\%$, $T_A = 0°$ to 70°C (Notes 2 & 3)

Symbol	Parameter	Min	Typ	Max	Units
I_{EE1}	Supply current out of V_{EE} pin—non transmitting		−85	−130	mA
I_{EE2}	Supply current out of V_{EE} pin—transmitting		−125	−180	mA
I_{RXI}	Receive input bias current (RXI)	−2		+25	μA
I_{TDC}	Transmit output dc current level (TXO)	37	41	45	mA
I_{TAC}	Transmit output ac current level (TXO)	±28		I_{TDC}	mA
V_{CD}	Collision threshold (Receive mode)	−1.45	−1.53	−1.58	V
V_{OD}	Differential output voltage (RX±, CD±)	±550		±1200	mV
V_{OC}	Common mode output voltage (RX±, CD±)	−1.5	−2.0	−2.5	V
V_{OB}	Diff. output voltage imbalance (RX±, CD±)			±40	mV
V_{TS}	Transmitter squelch threshold (TX±)	−175	−225	−300	mV
C_X	Input capacitance (RXI)		1.2		pF
R_{RXI}	Shunt resistance—non transmitting (RXI)	100			KΩ
R_{TXO}	Shunt resistance—transmitting (TXO)		10		KΩ

8.0 Switching Characteristics $V_{EE} = -9V \pm 5\%$, $T_A = 0°$ to 70°C (Note 3)

Symbol	Parameter	Fig	Min	Typ	Max	Units
t_{RON}	Receiver startup delay (RXI to RX±)	5 & 11		4		bits
t_{Rd}	Receiver propagation delay (RXI to RX±)	5 & 11		15	50	ns
t_{Rr}	Differential outputs rise time (RX±, CD±)	5 & 11		4		ns
t_{Rf}	Differential outputs fall time (RX±, CD±)	5 & 11		4		ns
t_{RJ}	Receiver & cable total jitter	10		±2		ns
t_{TST}	Transmitter startup delay (TX± to TXO)	6 & 11		1		bits
t_{Td}	Transmitter propagation delay (TX± to TXO)	6 & 11		25	50	ns
t_{Tr}	Transmitter rise time —10% to 90% (TXO)	6 & 11		25		ns
t_{Tf}	Transmitter fall time —90% to 10% (TXO)	6 & 11		25		ns
t_{TM}	t_{Tr} and t_{Tf} mismatch			0.5		ns
t_{TS}	Transmitter skew (TXO)			±0.5		ns
t_{TON}	Transmit turn-on pulse width at V_{TS} (TX±)	6 & 11		20		ns
t_{TOFF}	Transmit turn-off pulse width at V_{TS} (TX±)	6 & 11		250		ns
t_{CON}	Collision turn-on delay	7 & 11		7		bits
t_{COFF}	Collision turn-off delay	7 & 11			20	bits
f_{CD}	Collision frequency (CD±)	7 & 11	8.0		12.5	MHz
t_{CP}	Collision pulse width (CD±)	7 & 11	35		70	ns
t_{HON}	CD Heartbeat delay (TX± to CD±)	8 & 11	0.6		1.6	μs
t_{HW}	CD Heartbeat duration (CD±)	8 & 11	0.5	1.0	1.5	μs
t_{JA}	Jabber activation delay (TX± to TXO and CD±)	9 & 11	20	29	60	ms
t_{JR}	Jabber reset unjab time (TX± to TXO and CD±)	9 & 11	250	500	750	ms

Note 1: Absolute maximum ratings are those values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits.

Note 2: All currents into device pins are positive, all currents out of device pins are negative. All voltages referenced to ground unless otherwise specified.

Note 3: All typicals are given for $V_{EE} = -9V$ and $T_A = 25°C$.

9.0 Timing and Load Diagrams

TL/F/7405-6

FIGURE 5. Receiver Timing

TL/F/7405-7

FIGURE 6. Transmitter Timing
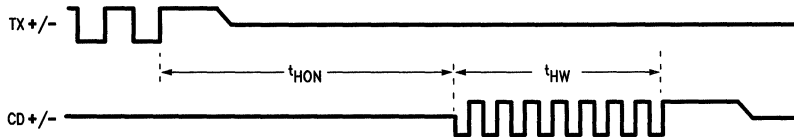
TL/F/7405-8

FIGURE 7. Collision Timing

TL/F/7405-9

FIGURE 8. Heartbeat Timing

9.0 Timing and Load Diagrams (Continued)

TX +/−

t_{JA}

t_{JR}

TXO

CD +/−

TL/F/7405−10

FIGURE 9. Jabber Timing

INPUT SIGNAL
WITH 30 ns RISE
AND FALL TIMES

R=1k

C=36 pF

RXI

DP8392A
RECEIVER

RX±
OUTPUT

R AND C NETWORK
SIMULATES WORST
CASE CABLE JITTER

±t_{RJ}

Input jitter ≤ ±1 ns

RX± Output jitter ≤ ±7 ns

Difference ≤ ±6 ns
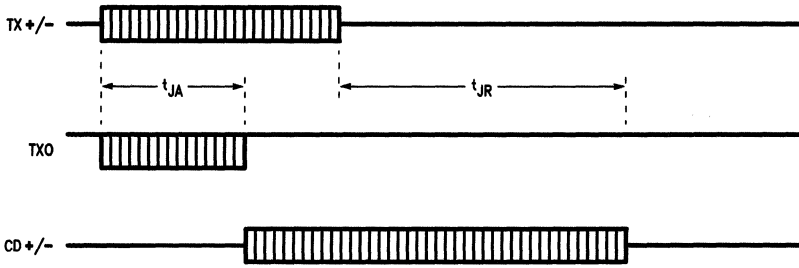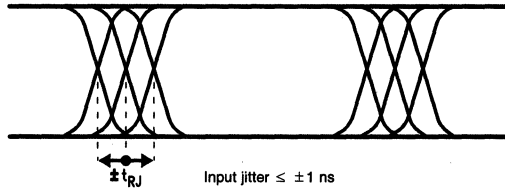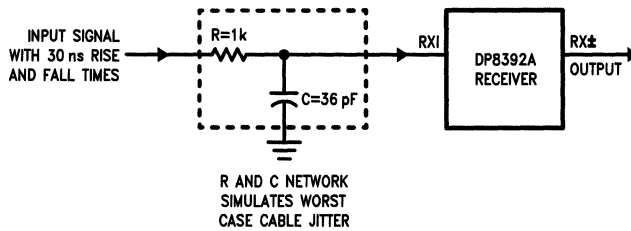
TL/F/7405−11

FIGURE 10. Receive Jitter Timing

TRANSMIT OUTPUT
(TXO)
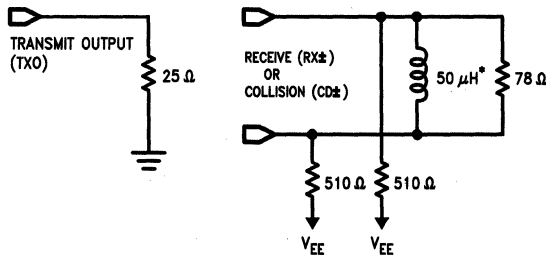
25 Ω

RECEIVE (RX±)
OR
COLLISION (CD±)

50 μH*

78 Ω

510 Ω

510 Ω

V_{EE}

V_{EE}

TL/F/7405−12

*The 50 μH inductance is for testing purposes. Pulse transformers with higher inductances are recommended (see *Figure 4*)

FIGURE 11. Test Loads

1-72

![National Semiconductor logo]

DP8392B/NS32492B Coaxial Transceiver Interface

General Description

The DP8392B Coaxial Transceiver Interface (CTI) is a coaxial cable line driver/receiver for Ethernet/Thin Ethernet (Cheapernet) type local area networks. The CTI is connected between the coaxial cable and the Data Terminal Equipment (DTE). In Ethernet applications the transceiver is usually mounted within a dedicated enclosure and is connected to the DTE via a transceiver cable. In Cheapernet applications, the CTI is typically located within the DTE and connects to the DTE through isolation transformers only. The CTI consists of a Receiver, Transmitter, Collision Detector, and a Jabber Timer. The Transmitter connects directly to a 50 ohm coaxial cable where it is used to drive the coax when transmitting. During transmission, a jabber timer is initiated to disable the CTI transmitter in the event of a longer than legal length data packet. Collision Detection circuitry monitors the signals on the coax to determine the presence of colliding packets and signals the DTE in the event of a collision.

The CTI is part of a three chip set that implements the complete IEEE 802.3 compatible network node electronics as shown below. The other two chips are the DP8391 Serial Network Interface (SNI) and the DP8390 Network Interface Controller (NIC).

The SNI provides the Manchester encoding and decoding functions; whereas the NIC handles the Media Access Protocol and the buffer management tasks. Isolation between the CTI and the SNI is an IEEE 802.3 requirement that can be easily satisfied on signal lines using a set of pulse transformers that come in a standard DIP. However, the power isolation for the CTI is done by DC-to-DC conversion through a power transformer.

Features

- Optimized for receive mode collision detection
- Compatible with Ethernet II, IEEE 802.3 10Base5 and 10Base2 (Cheapernet)
- Integrates all transceiver electronics except signal & power isolation
- Innovative design minimizes external component count
- Jabber timer function integrated on chip
- Externally selectable CD Heartbeat allows operation with IEEE 802.3 compatible repeaters
- Precision circuitry implements receive mode collision detection
- Squelch circuitry at all inputs rejects noise
- Designed for rigorous reliability requirements of IEEE 802.3
- Standard Outline 16-pin DIP uses a special leadframe that significantly reduces the operating die temperature

Table of Contents

1.0 System Diagram

IEEE 802.3 Compatible Ethernet/Cheapernet Local Area Network Chip Set

TL/F/10427–1

2.0 Block Diagram

COAX CABLE

DTE INTERFACE

RECEIVER EQUALIZER AND AMPLIFIER

DIFF LINE DRIVER

RECEIVE PAIR (RX+, RX−)

DATA BUFFER

RXI

RECEIVER AC−DC SQUELCH

4−POLE BESSEL LOW PASS FILTER

SENSE BUFFER

CDS

4−POLE BESSEL LOW PASS FILTER

Vth

COLLISION COMPARATOR

HBE

HEARTBEAT GENERATOR

10 MHz OSCILLATOR

JABBER TIMER

TXO

COLLISION PAIR (CD+, CD−)

TRANSMITTER SQUELCH

DIFF LINE DRIVER

GND

V_{EE}

TRANSMITTER

TRANSMIT PAIR (TX+, TX−)

TL/F/10427−2

FIGURE 1. DP8392B Block Diagram

3.0 Functional Description

The CTI consists of four main logical blocks:

a) the Receiver - receives data from the coax and sends it to the DTE

b) the Transmitter - accepts data from the DTE and transmits it onto the coax

c) the Collision Detect circuitry - indicates to the DTE any collision on the coax

d) the Jabber Timer - disables the Transmitter in case of longer than legal length packets

3.1 RECEIVER FUNCTIONS

The Receiver includes an input buffer, a cable equalizer, a 4-pole Bessel low pass filter, a squelch circuit, and a differential line driver.

The buffer provides high input impedance and low input capacitance to minimize loading and reflections on the coax.

The equalizer is a high pass filter which compensates for the low pass effect of the cable. The composite result of the maximum length cable and the equalizer is a flatband response at the signal frequencies to minimize jitter.

The 4-pole Bessel low pass filter extracts the average DC level on the coax, which is used by both the Receiver squelch and the collision detection circuits.

The Receiver squelch circuit prevents noise on the coax from falsely triggering the Receiver in the absence of the signal. At the beginning of the packet, the Receiver turns on when the DC level from the low pass filter is lower than the DC squelch threshold. However, at the end of the packet, a quick Receiver turn off is needed to reject dribble bits. This is accomplished by an AC timing circuit that reacts to high level signals of greater than typically 200 ns in duration. The

Receiver then stays off only if within about 1 μs, the DC level from the low pass filter rises above the DC squelch threshold. *Figure 2* illustrates the Receiver timing.

The differential line driver provides ECL compatible signals to the DTE with typically 3 ns rise and fall times. In its idle state, its outputs go to differential zero to prevent DC standing current in the isolation transformer.

3.2 TRANSMITTER FUNCTIONS

The Transmitter has a differential input and an open collector output current driver. The differential input common mode voltage is established by the CTI and should not be altered by external circuitry. The transformer coupling of TX± will satisfy this condition. The driver meets all IEEE 802.3/Ethernet Specifications for signal levels. Controlled rise and fall times (25 ns V ±5 ns) minimize the higher harmonic components. The rise and fall times are matched to minimize jitter. The drive current levels of the DP8392B meet the tighter recommended limits of IEEE 802.3 and are set by a built-in bandgap reference and an external 1% resistor. An on chip isolation diode is provided to reduce the Transmitter's coax load capacitance. For Ethernet compatible applications, an external isolation diode (see *Figure 4*) may be added to further reduce coax load capacitance. In Cheapernet compatible applications the external diode is not required as the coax capacitive loading specifications are relaxed.

The Transmitter squelch circuit rejects signals with pulse widths less than typically 20 ns (negative going), or with levels less than −175 mV. The Transmitter turns off at the end of the packet if the signal stays higher than −175 mV for more than approximately 300 ns. *Figure 3* illustrates the Transmitter timing.

3.0 Functional Description (Continued)

3.3 COLLISION FUNCTIONS

The collision circuitry consists of two buffers, two 4-pole Bessel low pass filters (section 3.1), a comparator, a heartbeat generator, a 10 MHz oscillator, and a differential line driver.

Two identical buffers and 4-pole Bessel low pass filters extract the DC level on the center conductor (data) and the shield (sense) of the coax. These levels are monitored by the comparator. If the data level is more negative than the sense level by at least the collision threshold (Vth), the collision output is enabled.

At the end of every transmission, the heartbeat generator creates a pseudo collision for a short time to ensure that the collision circuitry is properly functioning. This burst on collision output occurs typically 1.1 μs after the transmission, and has a duration of about 1 μs. This function can be disabled externally with the HBE (Heartbeat Enable) pin to allow operation with repeaters.

The 10 MHz oscillator generates the signal for the collision and heartbeat functions. It is also used as the timebase for all the jabber functions. It does not require any external components.

The collision differential line driver transfers the 10 MHz signal to the CD± pair in the event of collision, jabber, or heartbeat conditions. This line driver also features zero differential idle state.

3.4 JABBER FUNCTIONS

The Jabber Timer monitors the Transmitter and inhibits transmission if the Transmitter is active for longer than 20 ms (fault). It also enables the collision output for the fault duration. After the fault is removed, The Jabber Timer waits for about 500 ms (unjab time) before re-enabling the Transmitter. The transmit input must stay inactive during the unjab time.

TL/F/10427-3

FIGURE 2. Receiver Timing

TL/F/10427-4

FIGURE 3. Transmitter Timing

4.0 Connection Diagram

OPTIONAL TRANSCEIVER CABLE

12 TO 15 VDC

DC TO DC CONVERTER

+
9 V (ISOLATED)
−

500 OHM EACH

COLLISION PAIR

D T E

RECEIVE PAIR

TRANSMIT PAIR

T1 (NOTE 1)

78 OHM

COAX

CD+ 1 ... 16 CDS (Note 2)
CD− 2 ... 15 TXO   IN916 or IN415
RX+ 3 ... 14 RXI
VEE 4 DP8392B 13 VEE
VEE 5 CTI 12 RR−   1K 1%
RX− 6 ... 11 RR+
TX+ 7 ... 10 GND
TX− 8 ... 9 HBE

TL/F/10427−5

Note 1: T1 is a 1:1 pulse transformer, L = 100 μH
Pulse Engineering (San Diego) Part No. 64103
Valor Electronics (San Diego)
Part No. 1101 or equivalent

Note 2: This pin must be connected
to the coax shield

Top View

Order Number DP8392BN
See NS Package Number N16A

FIGURE 4

5.0 Pin Descriptions

Pin No.	Name	I/O	Description
1 2	CD+* CD−	O	**Collision Output.** Balanced differential line driver outputs from the collision detect circuitry. The 10 MHz signal from the internal oscillator is transferred to these outputs in the event of collision, excessive transmission (jabber), or during CD Heartbeat condition. These outputs are open emitters; pulldown resistors to VEE are required. When operating into a 78Ω transmission line, these resistors should be 500Ω. In Cheapernet applications, where the 78Ω drop cable is not used, higher resistor values (up to 1.5k) may be used to save power.
3 6	RX+* RX−	O	**Receive Output.** Balanced differential line driver outputs from the Receiver. These outputs also require 500Ω pulldown resistors.
7 8	TX+* TX−	I	**Transmit Input.** Balanced differential line receiver inputs to the Transmitter. The common mode voltage for these inputs is determined internally and must not be externally established. Signals meeting Transmitter squelch requirements are waveshaped and output at TXO.
9	HBE	I	**Heartbeat Enable.** This input enables CD Heartbeat when grounded, disables it when connected to VEE.
11 12	RR+ RR−	I	**External Resistor.** A fixed 1k 1% resistor connected between these pins establishes internal operating currents.
14	RXI	I	**Receive Input.** Connects directly to the coaxial cable. Signals meeting Receiver squelch requirements are equalized for inter-symbol distortion, amplified, and outputted at RX±.
15	TXO	O	**Transmit Output.** Connects either directly (Cheapernet) or via an isolation diode (Ethernet) to the coaxial cable.
16	CDS	I	**Collision Detect Sense.** Ground sense connection for the collision detect circuit. This pin should be connected separately to the shield to avoid ground drops from altering the receive mode collision threshold.
10	GND		**Positive Supply Pin.** A 0.1 μF ceramic decoupling capacitor must be connected across GND and VEE as close to the device as possible.
4 5 13	VEE		**Negative Supply Pins.** In order to make full use of the 3.5W power dissipation capability of this package, these pins should be connected to a large metal frame area on the PC board. Doing this will reduce the operating die temperature of the device thereby increasing the long term reliability.

* IEEE names for CD± = CI±, RX± = DI±, TX± = DO±

5.1 P.C. BOARD LAYOUT

The DP8392B package is uniquely designed to ensure that the device meets the 1 million hour Mean Time Between Failure (MTBF) requirement of the IEEE 802.3 standard. In order to fully utilize this heat dissipation design, the three VEE pins are to be connected to a copper plane which should be included in the printed circuit board layout. Refer to National Semiconductor application note AN-442 (Ethernet/Cheapernet Physical Layer Made Easy) for complete board layout instructions.

6.0 Absolute Maximum Ratings (Note 1)

Supply Voltage (V_{EE})	$-12V$
Package Power Rating at 25°C	3.5 Watts*
(PC Board Mounted)	See Section 5
Derate linearly at the rate of 28.6 mW/°C	
Input Voltage	0 to $-12V$
Storage Temperature	$-65°$ to $150°C$
Lead Temp. (Soldering, 10 seconds)	$260°C$

*For actual power dissipation of the device please refer to section 7.0.

Recommended Operating Conditions

Supply Voltage (V_{EE})	$-9v \pm 5\%$
Ambient Temperature	0° to 70°C

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

7.0 Electrical Characteristics $V_{EE} = -9V \pm 5\%$, $T_A = 0°$ to 70°C (Notes 2 & 3)

Symbol	Parameter	Min	Typ	Max	Units
I_{EE1}	Supply current out of V_{EE} pin—non transmitting		-85	-130	mA
I_{EE2}	Supply current out of V_{EE} pin—transmitting		-125	-180	mA
I_{RXI}	Receive input bias current (RXI)	-2		$+25$	μA
I_{TDC}	Transmit output dc current level (TXO)	37	41	45	mA
I_{TAC}	Transmit output ac current level (TXO)	± 28		I_{TDC}	mA
V_{CD}	Collision threshold (Receive mode)	-1.45	-1.53	-1.58	V
V_{OD}	Differential output voltage (RX$\pm$, CD$\pm$)	± 550		± 1200	mV
V_{OC}	Common mode output voltage (RX$\pm$, CD$\pm$)	-1.5	-2.0	-2.5	V
V_{OB}	Diff. output voltage imbalance (RX$\pm$, CD$\pm$)			± 40	mV
V_{TS}	Transmitter squelch threshold (TX$\pm$)	-175	-225	-300	mV
C_X	Input capacitance (RXI)		1.2		pF
R_{RXI}	Shunt resistance—non transmitting (RXI)	100			KΩ
R_{TXO}	Shunt resistance—transmitting (TXO)	7.5	10		KΩ

8.0 Switching Characteristics $V_{EE} = -9V \pm 5\%$, $T_A = 0°$ to 70°C (Note 3)

Symbol	Parameter	Fig	Min	Typ	Max	Units
t_{RON}	Receiver startup delay (RXI to RX$\pm$)	5 & 11		4	5	bits
t_{Rd}	Receiver propagation delay (RXI to RX$\pm$)	5 & 11		15	50	ns
t_{Rr}	Differential outputs rise time (RX$\pm$, CD$\pm$)	5 & 11		4	7	ns
t_{Rf}	Differential outputs fall time (RX$\pm$, CD$\pm$)	5 & 11		4	7	ns
t_{RJ}	Receiver & cable total jitter	10		± 2		ns
t_{TST}	Transmitter startup delay (TX$\pm$ to TXO)	6 & 11		1	2	bits
t_{Td}	Transmitter propagation delay (TX$\pm$ to TXO)	6 & 11	5	25	50	ns
t_{Tr}	Transmitter rise time —10% to 90% (TXO)	6 & 11	20	25	30	ns
t_{Tf}	Transmitter fall time —90% to 10% (TXO)	6 & 11	20	25	30	ns
t_{TM}	t_{Tr} and t_{Tf} mismatch			0.5		ns
t_{TS}	Transmitter skew (TXO)			± 0.5		ns
t_{TON}	Transmit turn-on pulse width at V_{TS} (TX$\pm$)	6 & 11	5	20	40	ns
t_{TOFF}	Transmit turn-off pulse width at V_{TS} (TX$\pm$)	6 & 11	130	250	340	ns
t_{CON}	Collision turn-on delay	7 & 11		7	13	bits
t_{COFF}	Collision turn-off delay	7 & 11			20	bits
f_{CD}	Collision frequency (CD$\pm$)	7 & 11	8.5		12.5	MHz
t_{CP}	Collision pulse width (CD$\pm$)	7 & 11	35		70	ns
t_{HON}	CD Heartbeat delay (TX$\pm$ to CD$\pm$)	8 & 11	0.6		1.6	μs
t_{HW}	CD Heartbeat duration (CD$\pm$)	8 & 11	0.5	1.0	1.5	μs
t_{JA}	Jabber activation delay (TX$\pm$ to TXO and CD$\pm$)	9 & 11	20	29	60	ms
t_{JR}	Jabber reset unjab time (TX$\pm$ to TXO and CD$\pm$)	9 & 11	250	500	750	ms

Note 1: Absolute maximum ratings are those values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits.

Note 2: All currents into device pins are positive, all currents out of device pins are negative. All voltages referenced to ground unless otherwise specified.

Note 3: All typicals are given for $V_{EE} = -9V$ and $T_A = 25°C$.

9.0 Timing and Load Diagrams

TL/F/10427–6

FIGURE 5. Receiver Timing

TL/F/10427–7

FIGURE 6. Transmitter Timing

TL/F/10427–8

TL/F/10427–9

FIGURE 7. Collision Timing
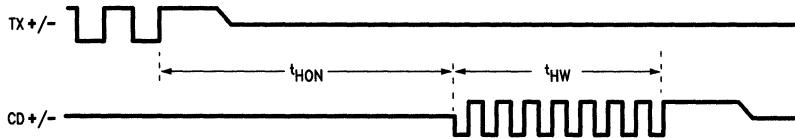
TL/F/10427–10

FIGURE 8. Heartbeat Timing

9.0 Timing and Load Diagrams (Continued)
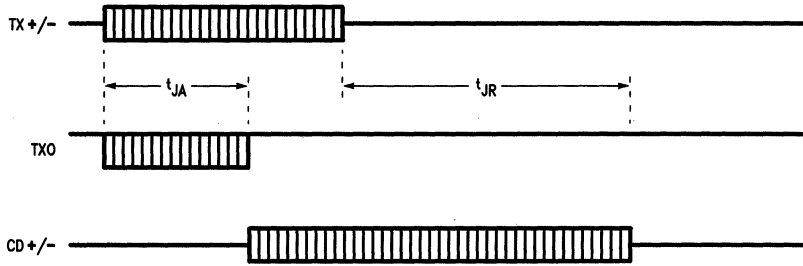
TL/F/10427–11

FIGURE 9. Jabber Timing

TL/F/10427–12
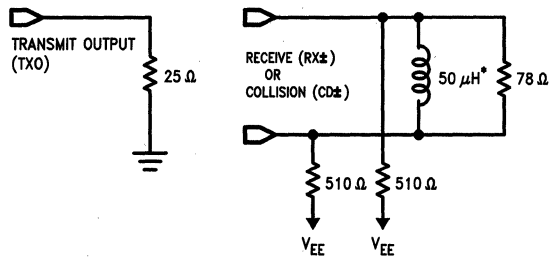
*The 50 µH inductance is for testing purposes. Pulse transformers with higher inductances are recommended (see *Figure 4*)

FIGURE 10. Test Loads

![National Semiconductor logo]

Twisted-Pair Interface

General Description

The Twisted-Pair Transceiver is used to connect IEEE 802.3 stations and repeaters to twisted-pair medium. It integrates all the transceiver functions of IEEE 802.3 10BASE-T standard.

With a full AUI interface, the Twisted-Pair Interface can be used both in stand-alone and embedded MAU applications. The primary functions are transmitter, receiver with smart squelch, collision detection, jabber timer, link test with status output, and SQE test (CD Heartbeat) with disable pin for repeater applications.

The Twisted-Pair Interface is part of a chip set that implements the complete IEEE 802.3 10BASE-T compatible network electronics. In repeater applications it can be used with the Repeater Interface Controller (RIC), and in node applications, it can be used with the DP8391/910 Serial Network Interface (SNI) and the DP8390 Network Interface Controller (NIC).

Features

- Compatible with IEEE 802.3 10BASE-T standard
- Integrates all transceiver electronics, including:
  — Transmitter
  — Receiver with Smart Squelch
  — Collision Detection
  — CD Heartbeat (SQE test) with disable pin
  — Jabber Timer Function
  — Link Integrity Test

- Full AUI compatible interface—can be used both in stand-alone and embedded MAU applications
- Link Status output
- Complete differential transmit and receive path for optimum jitter performance
- 24-pin narrow DIP package

Table of Contents

1.0 System Diagram

IEEE 802.3 Compatible Repeater Electronics

TL/F/10490–1

IEEE802.3 Compatible Node Electronics

TL/F/10490–2

National Semiconductor
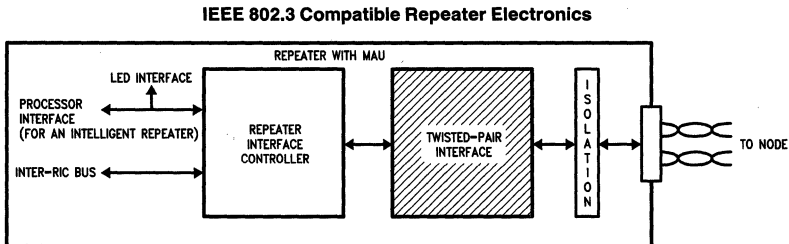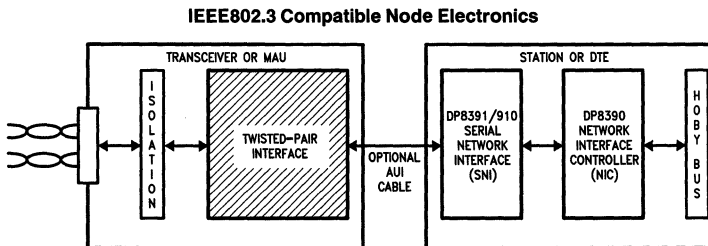
Systems-Oriented Network Interface Controller

Introduction

The SONIC™ (Systems-Oriented Network Interface Controller) is a second-generation IEEE 802.3 Controller designed to meet the demands of today's high-speed 32- and 16-bit systems. Its system interface operates up to 20 MHz, offering a 2-cycle DMA to transfer data up to 40 Mbytes/sec and typically consumes only 4% of the bus bandwidth. The SONIC employs a flexible linked-list Buffer Management system to operate in a variety of environments from PC-oriented adapters to high-performance 32-bit designs, and its bus interface works equally well with National/Intel or Motorola microprocessors. Furthermore, the SONIC integrates a fully-compatible IEEE 802.3 Encoder/Decoder (ENDEC) and when coupled with the DP8392 Coaxial Transceiver Interface or the Twisted Pair Interface, the SONIC provides a simple 2-chip solution for Ethernet.

For increased performance, the SONIC implements a unique buffer management scheme to efficiently process, receive and transmit packets in system memory. No intermediate packet copy is necessary. The receive buffer management uses three areas in memory for allocating additional resources, indicating status information, and buffering packet data. During reception, the SONIC stores packet in the buffer area, then indicates receive status and a packet pointer in the descriptor area. The system may allocate more resources to the SONIC by adding descriptors to the resource area. The Transmit Buffer Management uses two areas in memory: (1) for indicating status and control information and (2) for fetching the data of the packet. A transmit queue may be created by the system in the status and control area to allow multiple packets to be transmitted from a single transmit command. The data of the packet may reside on any arbitrary byte boundary and may exist in several non-contiguous locations.

Features

- 32-bit non-multiplexed address and data bus
- High-speed, 2-cycle DMA operating up to 20 MHz
- Linked-list Buffer Management maximizes flexibility
- Two Independent 32-byte Transmit and Receive FIFOs
- Compatible with National/Intel or Motorola Microprocessors
- Integrated IEEE 802.3 ENDEC
- Complete Address Filtering for up to 16 Physical or Multicast Addresses
- 32-bit General-Purpose Timer
- Full Loopback Diagnostics
- Fabricated in low-power CMOS
- 132 PQFP Package
- Fully supports the Layer Management 802.3 standard

Block Diagram

TL/F/10492–1

![National Semiconductor logo]
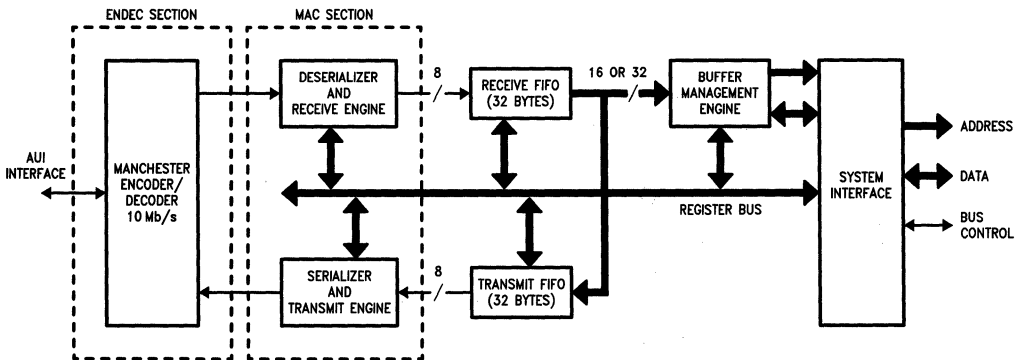
DP83901 Serial Network Interface Controller

General Description

The DP83901 Serial Network Interface Controller (SNIC) is a microCMOS VLSI device designed to ease interfacing with CSMA/CD type local area networks including Ethernet (10Base5), Thin Ethernet (10Base2), and Twisted Pair (10BaseT). The SNIC implements all Media Access Control layer functions for transmission and reception of packets in accordance with the IEEE802.3 standard. Unique dual DMA channels and an internal FIFO provide a simple yet efficient packet management design. To minimize system parts count and cost, all bus arbitration and memory support logic are integrated into the SNIC.

Also integrated into the SNIC is the Serial Network Interface. This provides the Manchester data encoding and decoding functions required by 802.3. The SNIC will interface directly to the Attachment Unit Interface (AUI). When transmitting, the SNIC produces differential data for the AUI. Conversely, when receiving, a phase-locked loop decodes the 10 Mbit/sec data.

An external transceiver may be connected directly to the SNIC's AUI interface. Transceivers for 10Base2 and 10BaseT are available from National Semiconductor. Alternatively, the SNIC may be connected directly to an AUI drop cable for 10Base5.

The SNIC is equivalent to the combination of the DP8390 and the DP83910 that are available from National Semiconductor.

Features

■ Implements simple, versatile buffer management
■ Combination of DP8390 Network Interface Controller and DP83910 Serial Network Interface
■ Compatible with 802.3 Ethernet 10Base5, 10Base2, and 10BaseT
■ Interfaces with 8-, 16-, and 32-bit μP systems
■ Connects directly to AUI interface
■ Single 5V supply
■ Utilizes low power microCMOS process
■ 68-pin PLCC

Block Diagram

TL/F/10469-1
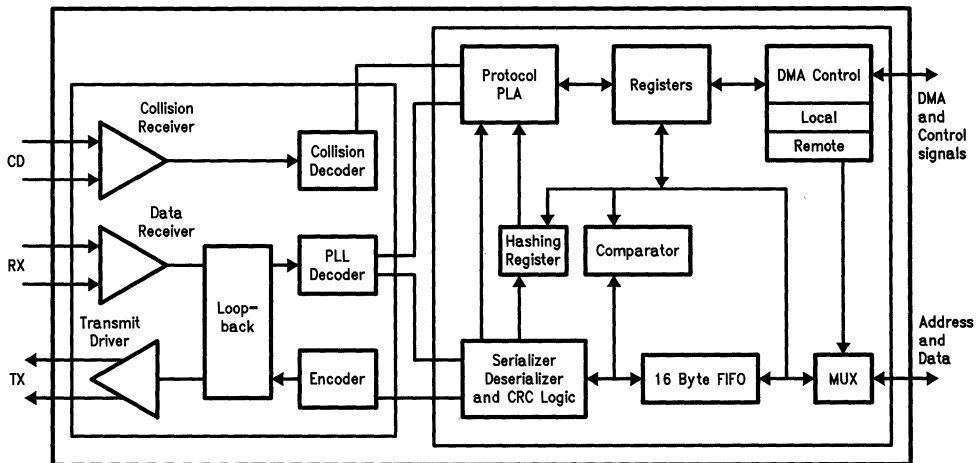
National Semiconductor

DP839EB-AT
16-Bit PCAT Ethernet Evaluation Board

General Description

The DP839EB-AT is designed as a high performance Ethernet adapter card which utilizes National Semiconductor's Ethernet chipset (DP8390, DP8391 or CMOS DP83910, DP8392). It provides a low-power thick (10Base5) or Thin (10Base2) Ethernet interface for the 16-bit PCAT bus.

Special features of the DP839EB-AT include shared buffer memory architecture, zero wait state shared memory arbitration, and word or byte wide transfers to/from the system bus. Also, the use of the CMOS DP83910 Serial Network Interface chip allows for a low power implementation. The shared memory is configurable to 8k x 16 or 32k x 16 and is mapped directly into the PCAT address space. This allows for highly efficient block data transfers between buffer memory and system memory. The zero wait state shared memory arbiter is designed to give the CPU immediate access to the buffer memory when the DP8390 NIC is not making a shared memory access. This increases system bus efficiency and allows optimal bus bandwidth.

The adapter's ability to perform byte or word wide shared memory transfers assures the system of full utilization of the 16-bit PCAT bus. In order to achieve this design, a state machine was designed which quickly notifies the CPU that 16-bit transfers may be used. On a typical block move cycle to the adapter card shared RAM, the first 16-bit move will

take place as two byte moves, then all successive moves, then all successive moves will take place as word transfers.

The DP839EB-AT is designed to make software interface and configuration as simple as possible. All variable card parameters (Interrupt number, Base Address, thick or thin Ethernet, and memory size) are software selectable, eliminating the need for hardware jumpers. Supplied demonstration code will provide all desired network functions and is coded in "C" for portability.

Features

- Efficient 16-bit shared buffer memory system bus interface
- Memory mapping supports 14 possible base addresses in real or extended memory
- Supports byte-wide or word-wide buffer memory transfers
- Zero wait state shared memory arbitration
- Software configurable for thick or thin wire Ethernet
- Low power (mostly) CMOS implementation
- No DMA channel required
- Full diagnostic software available, including Novell NetWare drivers
- Diagnostic LEDs

Block Diagram

TL/F/10471-1
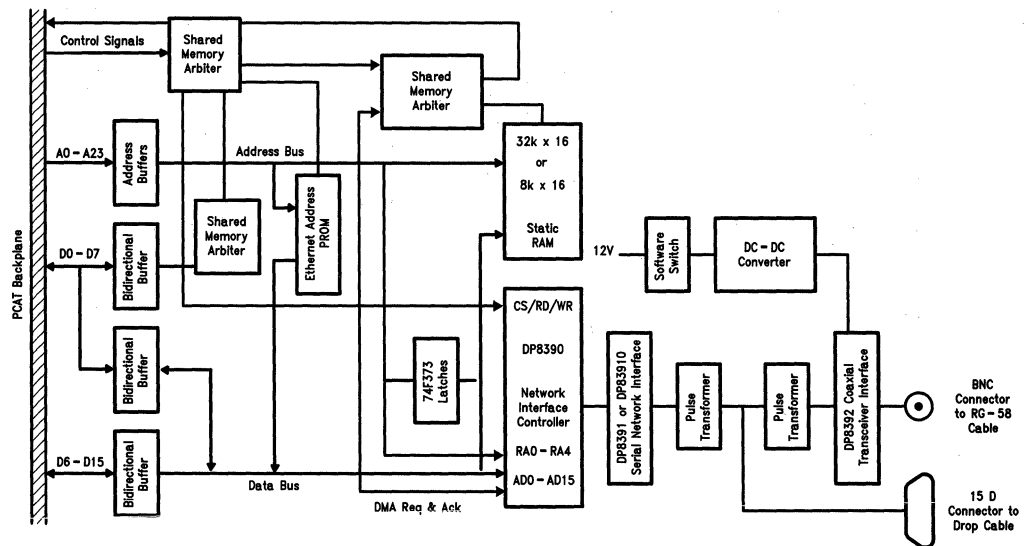
![National Semiconductor logo]

DP839EB-MC
16-Bit PS/2 Ethernet Evaluation Board

General Description

The DP839EB-MC is a 16-bit high performance Microchannel Ethernet adapter card for IBM's PS/2 computers. This board employs National Semiconductor's Ethernet chip set (DP8390, DP83910, DP8392) to provide a complete Ethernet and thin wire Ethernet solution.

This demonstration board is highlighted by the fact that its buffer memory, which is used to store receive and transmit packets, is directly accessible by both the DP8390 and the PS/2's CPU. By mapping the buffer RAM into memory, as opposed to I/O, the rate at which the host system can access the receive and transmit packet areas is greatly increased. Furthermore, the arbitration for this shared memory is designed to provide the CPU with zero wait state access by adhering to Microchannel's 30 ns request acknowledgement specification. This arbitration provides the maximum throughput available from the buffer memory to system memory, without disrupting the Network Interface Controller's (NIC) reception or transmission of packets.

In addition to offering a high performance Ethernet solution, the DP839EB/MC consumes only 1.44A of current from the PS/2's 5V power supply, making it a very power efficient discrete solution. This is well within the PS/2's specified amount of 1.6A. One final outstanding feature of the Microchannel Ethernet adapter card is that the configuration of the card as either an Ethernet or thin wire Ethernet adapter is performed through software via the DP839EM-MC's adapter definition file (ADF).

The DP839EB-MC comes with a demonstration program that performs both typical network functions and diagnostics. Also, the object and executable code for Novell file server and work station drivers are available.

Features

- 16-bit shared buffer memory architecture
- Zero wait state CPU accesses to the shared buffer RAM
- Minimal 1.44A drain from the 5V power supply
- Software configurable for Ethernet or thin wire Ethernet
- Configurable for 32k or 8k words of shared buffer RAM
- Physical layer diagnostic LEDs
- DP839EB-MC Adapter Definition File (ADF)
- Complete demonstration software package
- Novell file server and work station drivers (object and executable code)

Block Diagram

TL/F/10470–1
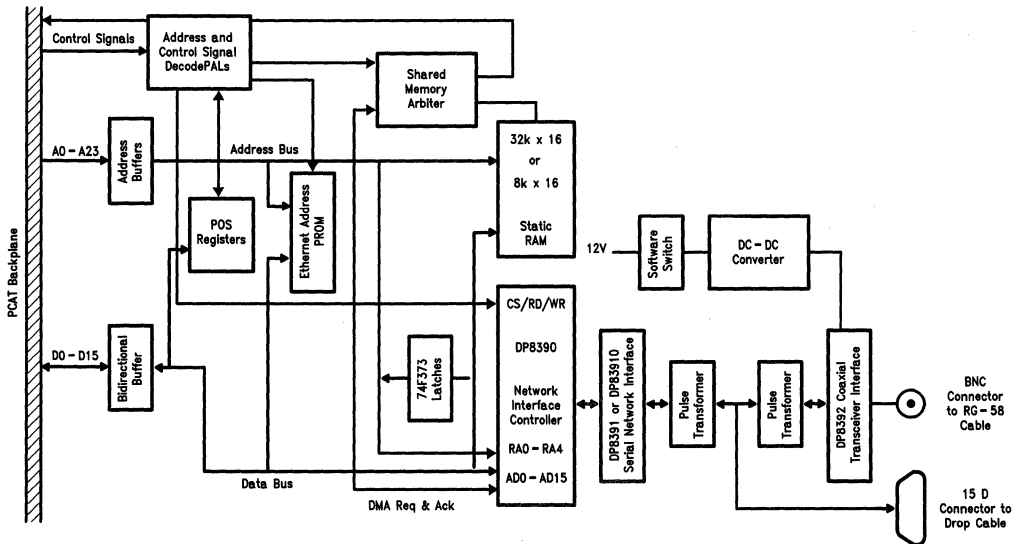
![National Semiconductor logo]

DP839EB-SE
16-Bit Mac SE Ethernet Evaluation Board

General Description

The DP839EB-SE is designed as a high performance low cost Ethernet adapter card for the Macintosh SE. This card utilizes National Semiconductor's Ethernet chipset. The DP839EB/SE provides a low-power thick (10BASE5) or thin (10BASE2) Ethernet interface for the Macintosh SE computer. Since the Mac SE provides a slot that is essentially Motorola 68000 μP signals, this board also is a good example of a general synchronous 68000 interface design using the National Ethernet Chip Set.

The DP839EB-SE is actually composed of two PCBs. One PCB lies on top of the SE's main board, and contains the Ethernet controller, buffer RAM, SNI, and bus logic. The second card, called the Connector Card, is mounted on the back of the Mac SE cabinet. It contains the DP8392 CTI, DC-DC converter, and pulse transformers.

The feature of the DP839EB-SE is an on-card shared packet buffer memory architecture that utilizes 16-bit wide RAM, either 16k or 64k bytes. This RAM is mapped into SE's 68000 memory space. The DP8390's bus clock is derived from the Mac SE's 16 MHz bus clock. This simplifies the shared RAM abitration logic since the Ethernet Controller and the CPU are synchronous to each other. This board provides a very low parts count solution, requiring only 14 ICs to completely implement the interface. Finally, the buffer RAM supports a byte write function which simplifies collecting packet fragments for transmission.

The DP839EB-SE utilizes the CMOS DP83910 Serial Network Interface chip and allows for a low power implementation of the cable interface. The DP8392 resides on the connector card which is mounted to the back of the Mac SE. The connector card contains a single thin/thick Ethernet selection switch, and also diagnostic LEDs for ease of connection debugging.

The DP839EB-SE is supplied with essentially the same demonstration/diagnostic program that is utilized on the DP839EB-NB. It provides network and diagnostic functions which are coded in 'C' using Apple's MPW 3.0 for portability.

Features

- Efficient 16-bit shared buffer memory with byte write function
- Supports byte, or 16 word transfers
- Fast synchronous shared memory arbitration
- Single jumper configurable for thick or thin Ethernet
- Low power fully CMOS implementation of 14 ICs
- No DMA channel required
- Full diagnostic software included
- Diagnostic LEDs

Block Diagram

TL/F/10472-1
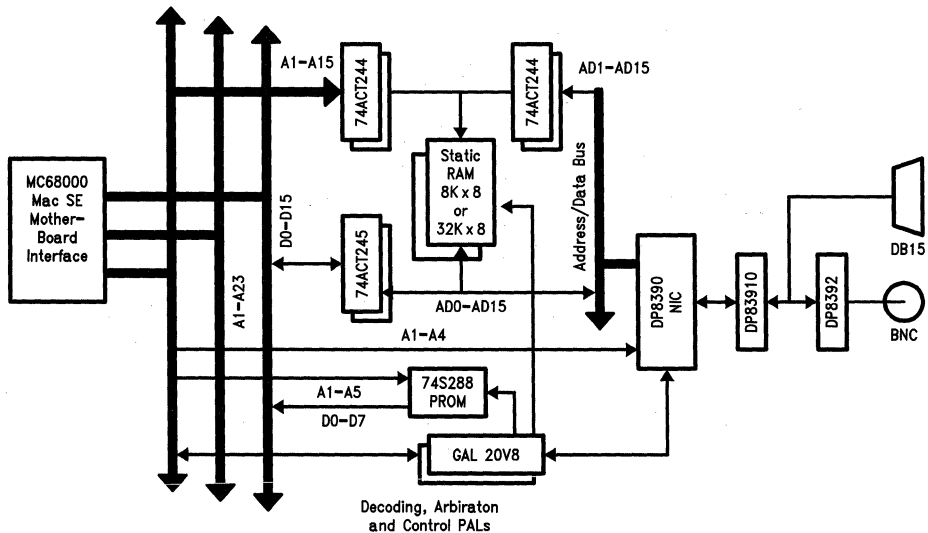
National Semiconductor

DP839EB-NB
32-Bit NuBus Ethernet Evaluation Board

General Description

The DP839EB-NB is designed as a high performance Ethernet adapter card which utilizes National Semiconductor's DP8390 Ethernet chipset (DP8390, DP8391 or CMOS DP83910, DP8392). This card provides a low-power Ethernet connection to thick (10Base5) or thin (10Base2) Ethernet for the NuBus equipped Macintosh computers (Mac II, IIx, IIcx, etc.).

The major feature of the DP839EB-NB is a shared buffer memory architecture that utilizes 16-bit wide RAM on the board. The shared memory is configurable for either 8k x 16 or 32k x 16 that is directly addressable by the NuBus as 32-bit words. Logic on the card utilizes a 5 clock transfer cycle. For a read this bus cycle first reads 16-bits from the RAM, then the next 16-bits, next logic assembles both 16-bit words into a single 32-bit word, and completes the transfer. On a RAM write the 32-bit quantity is split into two 16-bit quantities, and loaded into the RAM. This design allows for highly efficient block data transfers between buffer memory and system memory without the cost of a full 32-bit wide static RAM (4 byte-wide RAM chips).

The cable interface section utilizes the DP83910 and DP8392. It supports the use of either thin or thick Ethernet via the selection based on a single jumper.

The adapter fully supports Apple's implementation of Nu-Bus, including the Configuration ROM which also includes the Ethernet Address information. A state machine defines the bus cycles depending on device accessed, DP830 read/write or RAM read/write. The physical cable interface uses a single jumper to configure either thin or thick Ethernet operation.

The DP839EB-NB is supplied with demonstration/diagnostic code that provides network and diagnostic functions which is coded in "C" for portability.

Features

- Efficient 16-bit shared buffer memory with 32-bit system bus interface
- Supports byte, 16- or 32-bit word transfers
- Fast shared memory arbitration
- Single jumper configurable for thick or thin Ethernet implementation
- Low power CMOS implementation
- No DMA channel required
- Full diagnostic software included
- Diagnostic LED's

Block Diagram
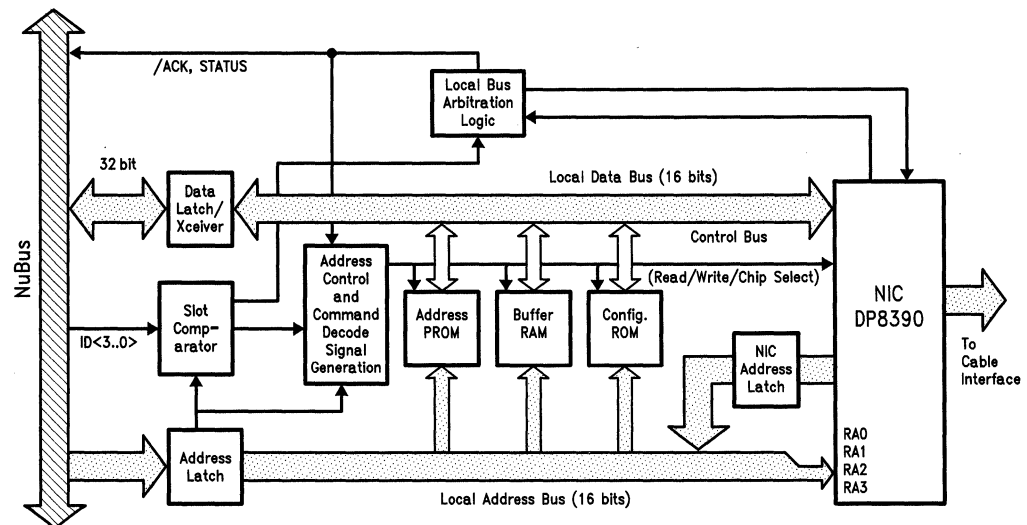
TL/F/10473-1

Low Power Ethernet with the CMOS DP83910 Serial Network Interface

INTRODUCTION

This application note discusses the features of, and implementation techniques for, National Semiconductor's CMOS Serial Network Interface (SNI), the DP83910. Also, a comparison of the CMOS SNI to National's bipolar SNI (DP8391) on several key issues will be provided. In general, the DP83910 provides a low power Attachment Unit Interface (AUI) for a Carrier-Sense Multiple Access with Collision Detect (CSMA/CD) Ethernet system. In fact, when used in conjunction with National Semiconductor's Network Interface Controller (NIC, DP8390) and Coaxial Transceiver Interface (CTI, DP8392), the DP83910 provides for a complete IEEE 802.3 Ethernet and/or thin wire Ethernet solution, as shown in *Figure 1*.

FUNCTIONAL DESCRIPTION OF THE DP83910

The CMOS SNI operates as an interface between an Ethernet transceiver and a local area network data controller. A functional block diagram of the DP83910 is shown in *Figure 2*. The primary function of this interface is to perform the encoding and decoding that is necessary for the differential pair Manchester encoded data of the transceiver and the Non-Return-to-Zero (NRZ) serial data of the NIC to be compatible with each other. In the case of a transmission, the SNI translates the NRZ serial data from a network controller's transmit data line into differential pair Manchester encoded data on a transceiver's transmit pair. In order to

perform this operation, the NRZ bit stream is first received by the Manchester encoder block of the SNI. Once the bit stream is encoded, it is transmitted out differentially on to the transmit differential pair through the transmit driver. When a reception takes place, the differential receive data from a transceiver is converted from Manchester encoded data into NRZ serial data and a receive clock, which are passed to the receive data and receive clock inputs of the Network Interface Controller. In executing this sequence, the DP83910's data receiver takes the Manchester data from the differential receive lines and passes it to the phase locked loop (PLL) decoder block. The PLL block then decodes the data and generates a data receive clock and a stream of NRZ serial data, which is presented to the NIC. In the case of National Semiconductor's Network Interface Controller, the DP8390, the serial NRZ signals are called TXD and RXD.

In addition to performing the Manchester encoding and decoding function, the DP83910 also provides several important network signals to the network controller. A diagram of the interface between National Semiconductor's NIC and the CMOS SNI can be found in *Figure 3*. The first of these signals is carrier sense (CRS), which indicates to the controller that data is present on the SNI's receive differential pair. Secondly, the SNI provides the network controller with a collision detection signal (COL), which informs the controller that a collision is taking place somewhere on the net-

IEEE 802.3 Compatible Ethernet/Thin Ethernet/10BaseT Local Area Network Chip Set
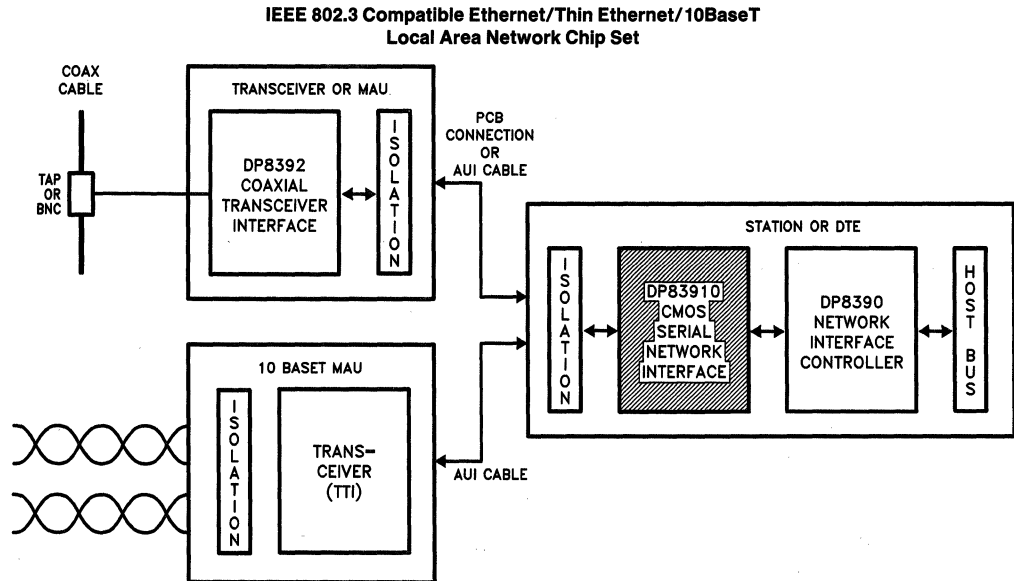
TL/F/10446–1

FIGURE 1. A Block Level Diagram of an Ethernet Node
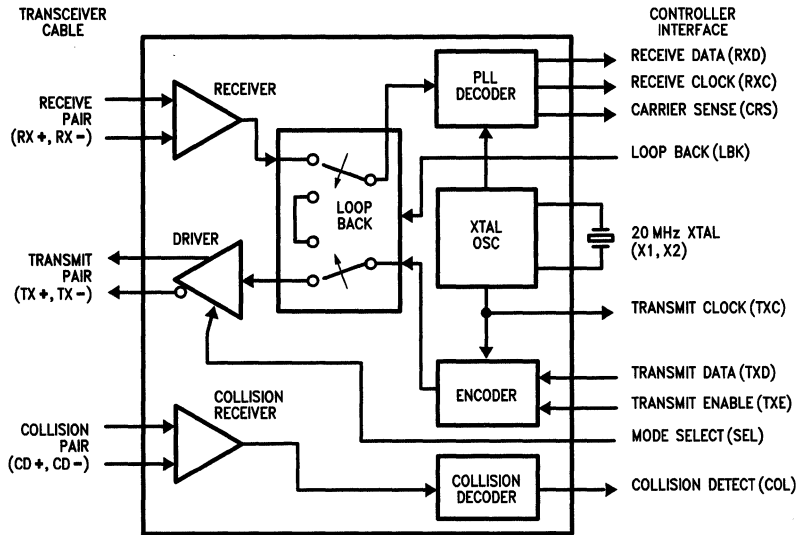
FIGURE 2. DP83910 Block Diagram

TL/F/10446–2
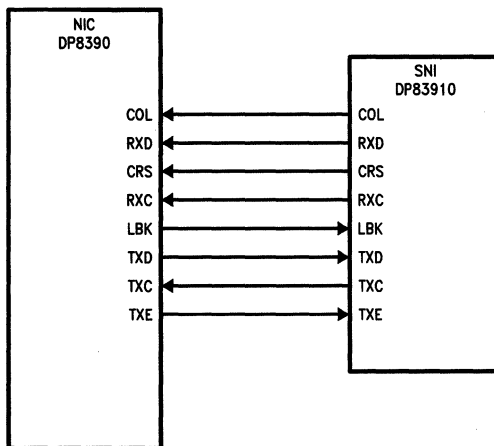
FIGURE 3. Interface between the DP8390 and DP83910

TL/F/10446–3

work. The SNI itself is informed of the collision when its collision receiver detects a 10 MHz signal on the differential collision input pair. Finally, the DP83910 provides both the receive and transmit clocks (RXC and TXC, respectively). The transmit clock is a divide by two derivative of the SNI's oscillator inputs (X1 and X2), while the receive clock is generated directly from the frequency of the input data to the PLL.

The DP83910 can also be placed in a loopback mode, in order to check the SNI's receive and transmit interface to the network controller. In loopback, as pictured above, the SNI's Manchester encoder block is essentially connected directly to the PLL decoder block. This allows for the validation of the Manchester encoding and decoding process without the variable of random network traffic. The SNI is placed in loopback mode when the loopback pin (LBK) is driven high.

COMPARING THE DP83910 WITH THE DP8391

The DP83910 is basically a CMOS version of the existing National Semiconductor bipolar SNI, the DP8391. The functionality of the two parts is identical. However, there are a few differences that exist between the two parts, in spite of the fact that they can be implemented as pin for pin compatible. The most fundamental difference between the two parts is the process under which each is manufactured. The DP83910 SNI is fabricated in a CMOS process, while the DP8391 is made in a bipolar process. As a result of this, the level of average power supply current needed by the DP83910 is approximately 75 percent less than the 270 mA required by the DP8391. Another significant difference between the two parts is the CMOS SNI's need for a pulse transformer to be placed between all of its differential signals and those of the transceiver, regardless of whether a drop cable or thin wire Ethernet configuration is being imple-

mented. This is necessary due to the fact that the CMOS process will not guarantee the IEEE 802.3 16V fail safe specification if no isolation is provided to the differential signals that go to the AUI cable. One consequence of the transformer requirement is that National Semiconductor defines the AUI interface at the transceiver side of the transformer and only guarantees the correct operation of the CMOS SNI when the pulse transformer is employed in the system.

In addition to the above process related differences, there are still two non-process related differences, which need to be mentioned. First, the phase locked loop in the bipolar SNI is digital, while the phase locked loop of the CMOS SNI is analog. This is functionally transparent when designing with the DP83910; however, it does provide for a significant savings in power consumption. Finally, it should be noted that pin 17 (TEST) on the bipolar SNI is required to be tied to ground through a capacitor, while the same pin on the CMOS SNI can either be implemented in the same manner or connected directly to ground. A list of all the above mentioned differences can be found in Table I.

DESIGNING WITH THE DP83910

In developing the DP83910, National Semiconductor performed extensive testing in its own Local Area Network Laboratory to assure that the CMOS SNI would provide an easily implemented low power controller/transceiver interface for Ethernet system designers. This development and testing assured that the DP83910 was IEEE 802.3 and Ethernet compatible, able to interface with industry standard transceivers (Ethernet, Twisted Pair Ethernet, and Fiber Optic Ethernet), and is capable of having the National Semiconductor DP8391 as a pin-for-pin replacement. In *Figures 4* and *5*, two methods of implementing the DP83910 with the DP8392 are demonstrated. One significant feature of both designs is that it is possible to directly substitute a DP8391 for the CMOS SNI and maintain the same functional quality.

The DP83910 Transmitter Operation

When operating as a transmitter, the DP83910 combines NRZ data received from the controller with a clock signal, which the SNI generates, and encodes them into a Manchester serial bit stream. This encoded signal then appears differentially at the SNI's TX± output. In Ethernet (10Base5) applications, this signal is sent to the transceiver or the Medium Attachment Unit (MAU) through an AUI transceiver cable. This cable, which can be up to 50 meters

in length, typically consists of four individually shielded twisted wire pairs (TX±, RX±, CD±, and PWR/GND), which are covered by an additional overall shield. The transmit signal pair, which has a differential characteristic impedance of 78Ω, should be terminated at the receiving end of the cable. It should be noted that each of the TX+ and TX− source follower outputs needs to be connected to ground through a 270Ω pull down resistor.

When employing the CMOS SNI, it is important to place a pulse transformer between the differential transmit pair on the DP83910 and the differential transmit signal on the AUI cable or CTI, as shown in *Figures 4* and *5*. This transformer is required in order to provide the necessary isolation for the CMOS SNI to meet the IEEE 802.3 16V fail safe specification. However, the pulse transformer does reduce the transmission of noise onto the transceiver cable. Also, it should be noted that more inductive transformers will decrease the magnitude of the undershoot. Furthermore, it is imperative that the designer guarantee the inductive load seen between the DP83910's AUI interface and the CTI receiver be greater than 27 μH. Transformers with 50 μH to 150 μH loading, such as the Pulse Engineering PE64103 and Nano Pulse NP5417, are recommended, since they will minimize the inductive undershoot on the SNI's TX± output pair and reduce the noise seen by the CTI's differential transmit input pair. It is important that the selected pulse transformer doesn't excessively increase the rise and fall time nor lower the output amplitude despite the fact that it reduces the undershoot.

The DP83910 provides both half and full step modes. The IEEE 802.3 standard requires the use of half step mode, in which the transmit output goes to differential zero in idle. In full step mode, the transmitter enters idle and stays at a fixed level. This will eventually allow the pulse transformer to completely saturate. The desired mode of operation is chosen through the Mode Select pin (SEL) on the SNI.

The DP83910 Data Receiver Operation

While performing reception, the CMOS SNI receives differential Manchester encoded serial data and converts it into NRZ serial data and a receive clock. The Manchester encoded data, which is received from the CTI or AUI cable, must be isolated before it reaches the SNI. Hence, the DP83910 requires that there be a pulse transformer on the SNI's side of the AUI interface. The actual employment of this transformer can be seen in both *Figures 4* and *5*. This

TABLE I. Comparison of the DP8391 and DP83910

	DP8391	DP83910
Process	Bipolar	CMOS
Power Consumption (Typical)	270 mA	70 mA
Pulse Transformer (At DTE Side of AUI Interface)	Optional	Required
Phase Locked Loop	Digital	Analog
Pin 17	PLL Filter/Capacitor Required	Test Pin/Capacitor Optional
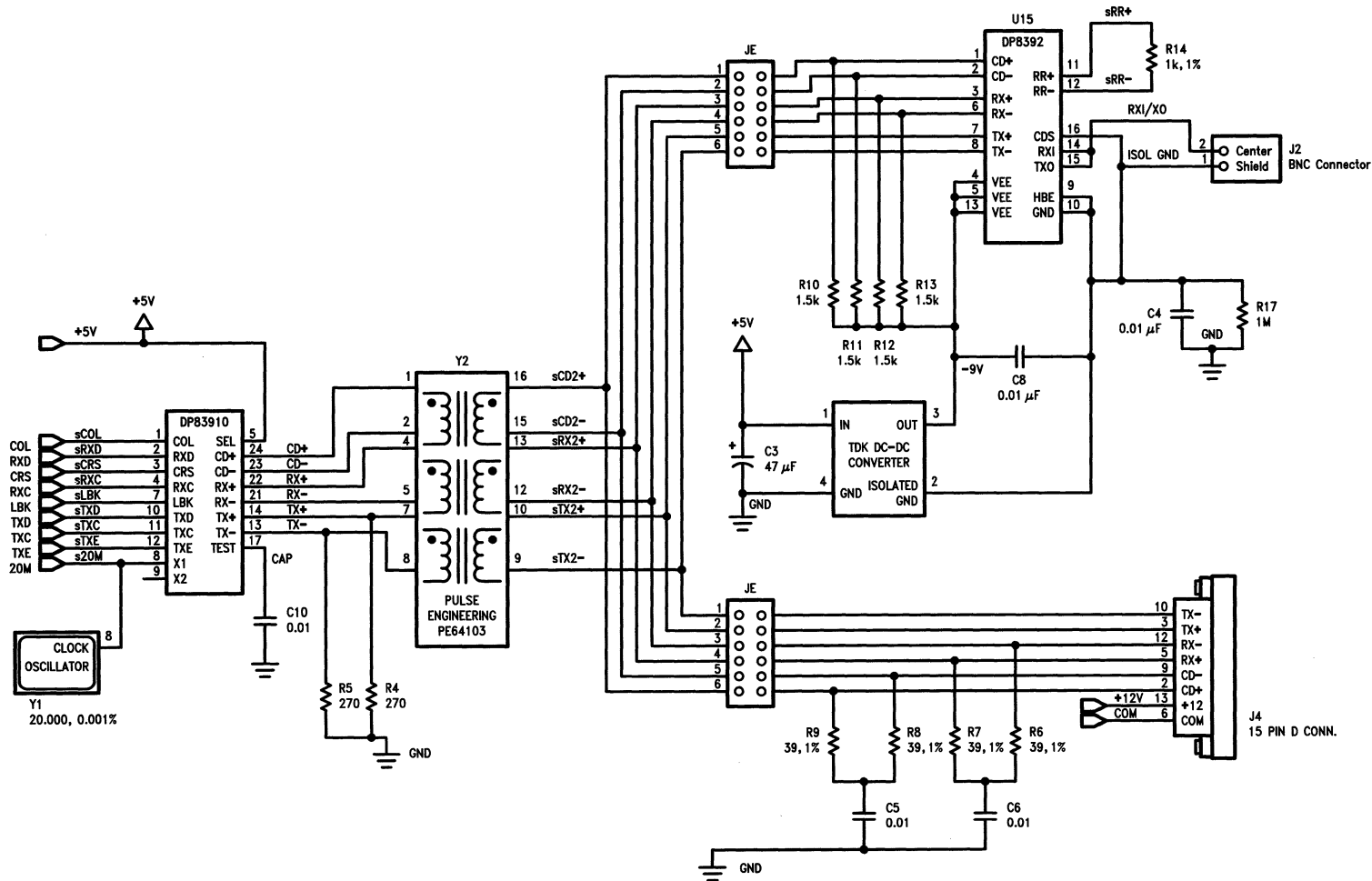
FIGURE 4. Interface for Ethernet and Thin Wire Ethernet

TL/F/10446–4
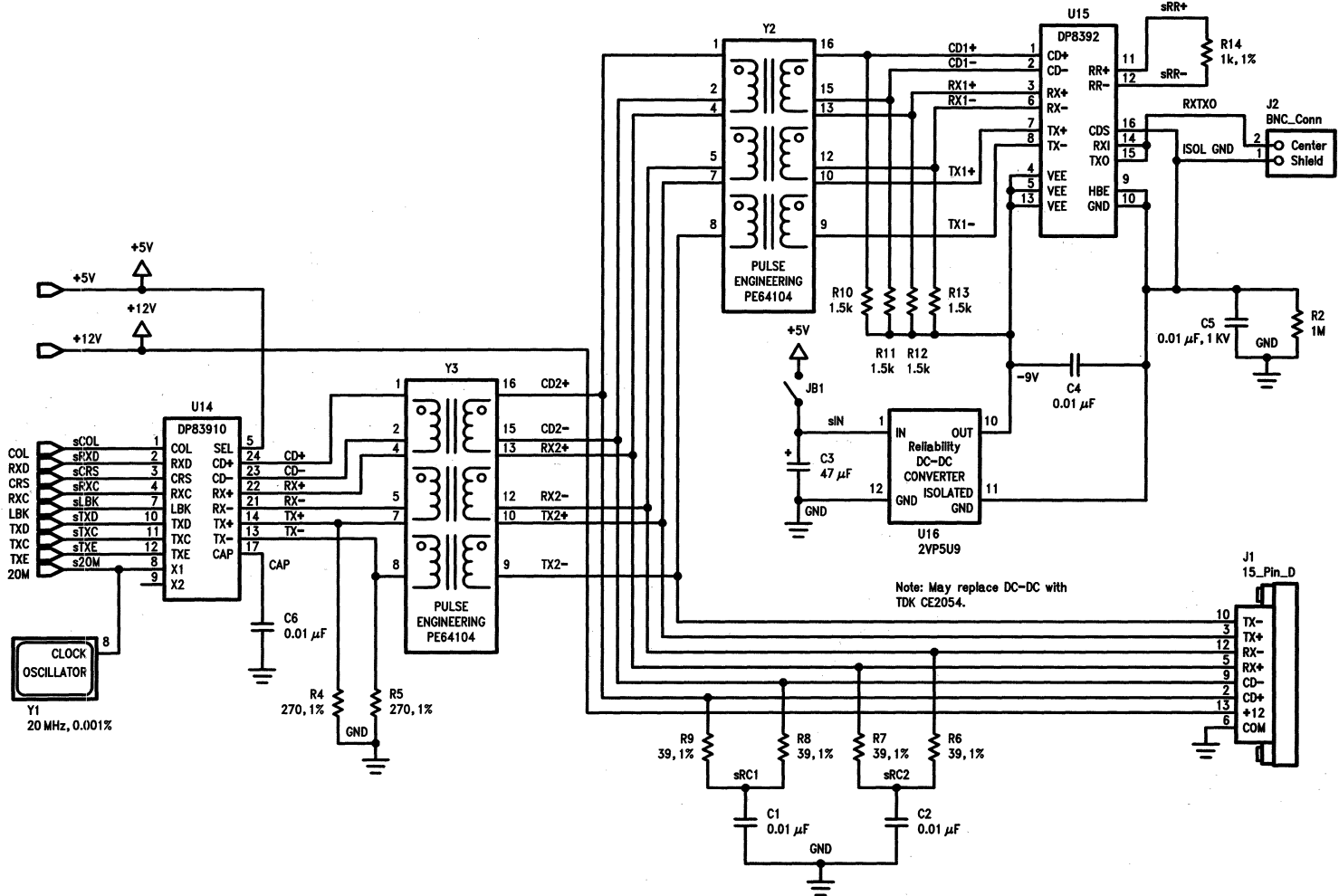
FIGURE 5. Single Switch Solution for Ethernet and Thin Wire Ethernet

TL/F/10446-6

transformer is mandatory and it forms part of the internal DC biasing circuit used for the differential receivers. Furthermore, the transformer is also needed to isolate the transceiver cable against the 16V voltage fault specification in the IEEE 802.3 standard. The performance of the differential receiver is not greatly affected by the selection of a pulse transformer. As a result, the pulse transformer selected for the transmitter design will also work correctly for the RX± data receiver. It should be noted here that the collision receiver is very similar to the data receiver and requires the same isolation. The collision input will be discussed more in the following section.

Once the data arrives at the receiver inputs of the SNI, it is amplified and then decoded by the analog phase locked loop, which can receive Manchester data with ±20 ns of random jitter. During the decoding process, the incoming signal is converted into NRZ data and a receive clock, which are sent to a network controller. Also, the differential data receiver has a built in filter to provide a static noise margin. This filter enables the SNI to reject signals that do not exceed the input squelch voltage and have less than a 30 ns pulse width.

Furthermore, since the DP83910 and pulse transformer constitute the AUI interface, the physical connection between the AUI and the MAU interfaces is defined as being on the MAU side of the pulse transformer. In light of this, it is permissible, when incorporating the CMOS SNI in a thin wire Ethernet application, to have a 78Ω resistance appear across the differential receive and collision inputs to the CTI, as shown in *Figure 5*.

The DP83910 Collision Pair Operation

In addition to the data receiver, the DP83910 also provides a differential receiver for the collision pair, which is driven by the transceiver. This 10 MHz active signal, from the AUI Interface, is converted to a TTL signal, digitally stretched, and sent to the controller as the Collision Detect Output (COL). Just as with the data receiver, the differential collision receiver has a built in filter that rejects pulses that do not exceed the input squelch voltage level and have a pulse width less than 30 ns.

Optimal Ethernet and Thin Wire Ethernet Interface

If it is necessary to design a LAN board that minimizes the number of switching devices (jumpers) to alternate between Ethernet and thin wire Ethernet, the solution in *Figure 5* could be employed. This solution, in contrast to the six jumper solution in *Figure 4*, requires only one switch, which enables and disables the power supply to the CTI. In the case of thin wire Ethernet, power would be supplied to the CTI, while during drop cable Ethernet operation the unused CTI would be powered down. Hence, no excessive power is required when thin wire Ethernet is not in use. Furthermore, since there is only one switch, it may be feasible to implement that switch with a transistor as opposed to a jumper. The advantage to using a transistor is that the Ethernet/thin wire Ethernet option can now be made to be software selectable. This is accomplished by developing a control signal, which the software can issue to switch the transistor. Also, in looking at *Figure 5*, it is seen that two pulse transformers are used. The first transformer (Y3) is required by the CMOS SNI, for the reasons previously mentioned. The second pulse transformer (Y2), however, is used to isolate the powered-down CTI from the AUI cable interface, when Ethernet is being used. As in *Figure 4*, the application in *Figure 5* allows the direct substitution of a bipolar SNI, the DP8391, for the CMOS SNI.

The DP83910 Oscillator Inputs

The oscillator inputs of the CMOS SNI can be driven with a crystal or an oscillator. In either case, the SNI oscillator must be driven with a 20 MHz signal that provides for the transmitted frequency to be accurate within 0.01% as specified in IEEE 802.3 standard. When using an oscillator, the output of the oscillator should be tied to input X1 of the SNI and the X2 input of the SNI should be left unconnected or grounded. However, the employment of a crystal to generate the 20 MHz signal at the SNI's oscillator inputs requires a great deal of care. The frequency of the crystal is usually measured with a fixed load capacitance (C_L, typically 20 pF), which is specified in the crystal's data sheet. In order to prevent any distortion in the transmitted frequency, the total capacitance across the crystal's leads should equal its specified load capacitance. The capacitance that is seen by the crystal's leads is the sum of the stray PC board capacitance (C_{PCB}) and the capacitance looking into the X1 and X2 inputs (C_{SNI}). If this capacitance is smaller than the crystal's load capacitance, a correctional capacitance (C_C) can be placed across the crystal's leads. This correctional capacitance would equal the difference between the crystal's load capacitance and the sum of the stray PC board capacitance and the SNI's X1 and X2 input capacitance. It should be noted that the input capacitance of the SNI that is seen across X1 and X2 is approximately a negligible 0.5 pF. *Figure 6* displays a possible crystal setup. The selected crystal should meet the following specifications:

Resonant frequency	20 MHz
Tolerance	±0.001% at 25°C
Stability	±0.005% at 0°C–70°C
Type	AT cut
Circuit	Parallel Resonance

$$C_C = C_L - (C_{PCB} + C_{SNI})$$
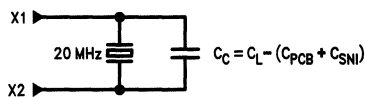
TL/F/10446–5

FIGURE 6. SNI Oscillator Input Circuit

Improving Transmitter Overshoot

Upon transitioning from a differential voltage of one polarity to another polarity (i.e., positive to negative), the magnitude of the differential transmit signal will reach a peak value. This peak at the transition points in the differential transmit waveform is referred to as the overshoot voltage. The overshoot voltage of the DP83910 is below the maximum allowable 1315 mV value that appears in the IEEE 802.3 standard. However, the IEEE standard also defines the overshoot voltage to be no greater than 1.12 times the nominal value (IEEE calls this nominal value V2). The DP83910 exceeds this particular segment of the overshoot specification, as shown in *Figure 7*. However, exceeding the allowable overshoot voltage value, as the CMOS SNI does, will have no functional affect on a system. Furthermore, the overshoot voltage can be altered to adhere to the IEEE 802.3 specification by placing a capacitor across the differential transmit pair at the primary (SNI side) of the required pulse transformer. This capacitor should be in the range of 40 pF to 50 pF and will not degrade the performance of the CMOS SNI or system in any way. It should also be mentioned that the DP8391, the bipolar SNI, will still be a pin-for-pin replacement for the CMOS SNI, in a design which employs the capacitor for improving the overshoot.
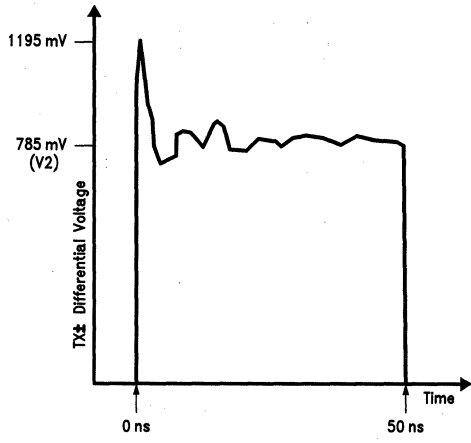
TL/F/10446–7

FIGURE 7. TX ± Differential Overshoot Voltage

Designing the DP8392 for Longer Cable Applications

National Semiconductor
Application Note 621
Mohammed Rajabzadeh

The IEEE 802.3 standard is designed for 500 meters of Ethernet cable and 185 meters of Cheapernet (RG58A/U) cable. To extend such segments to 1000 meters of Ethernet cable and 300 meters of Cheapernet cable requires utilization of Transmit mode collision detection. This method is described below.

COLLISION DETECTION SCHEMES

The collision circuitry monitors the coaxial DC level. If the level is more negative than the collision threshold, the collision output is enabled.

There are two different collision detection schemes that can be implemented with the CTI; Receive mode, and Transmit mode. The IEEE 802.3 standard allows the use of receive and transmit modes for non-repeater node applications. Repeaters are required to have to receive mode implementation. These different modes are defined as follows:

Receive Mode: Detects a collision between any two stations on the network with certainty at all times.

Transmit Mode: Detects a collision with certainty only when the station is transmitting.

Table I summarizes the receive and transmit mode definitions:

Receive Mode: The Receive mode scheme has a very simple truth table. However, the tight threshold limits make the design of it difficult. The threshold in this case has to be between the maximum DC level of one station (-1300 mV) and the minimum DC level of two far stations (-1581 mV). Several factors such as the termination resistor variation, signal skew, and input bias current of non-transmitting nodes contribute to this tight margin. On top of the -1300 mV minimum level, the impulse response of the internal low pass filter has to be added. The CTI incorporates a 4-pole Bessel filter in combination with a trimmed on board bandgap reference to provide this mode of collision detection.

Transmit Mode: In this case, collision has to be detected only when the station is transmitting. Thus, collision caused by two other nodes may or may not be detected. This feature relaxes the upper limit of the threshold. As a result of this, longer cable segments can be used. With the CTI, a resistor divider can be used at the Collision Detection Sense (CDS) pin to lower the threshold from receive to transmit mode.

COLLISION LEVELS—TRANSMIT MODE

Table II shows the parameter values that are used in calculating the collision levels in transmit mode.

TABLE I

Mode	Receive				Transmit			
No. of Stations	0	1	2	>2	0	1	2	>2
Transmitting	N	N	Y	Y	N	N	Y	Y
Non-Transmitting	N	N	Y	Y	N	N	M	Y

Y = Detects Collision
N = Does Not Detect Collision
M = Might Detect Collision

TABLE II. Assumptions and Definitions

R_T	= Termination Resistor at 20°C	= 50 ±1%Ω		802.3
t_T	= Temp. Coef. of the Terminator	= 0.0001/Deg.		ASSUMPTION
L	= Maximum Segment Length	= 300m	Cheapernet	802.3
		= 1000m	Ethernet	802.3
R_{DC}	= Maximum Cable DC Res. at 20°C	= 0.0489Ω/m	Cheapernet	BELDEN
		= 0.0100Ω/m	Ethernet	BELDEN
tc	= Temp. Coef. of Copper	= 0.004/°C		PHYSICS
Tm	= Maximum Cable Temp.	= 50°C		ASSUMPTION
SR	= Step Response at Max Cable Length	= 0.97	Cheapernet	NATIONAL
		= 0.94	Ethernet	NATIONAL
R_C	= Max. Connector Res./Station	= 0.0034Ω	Cheapernet	MIL SPEC
		= 0.0001Ω	Ethernet	ASSUMPTION
I_{B+}	= Max. Positive Bias Current	= 2 μA		802.3
I_{B-}	= Max. Negative Bias Current	= 25 μA		802.3
I_{max}	= Max. DC Drive Current	= 45 mA		802.3
I_{min}	= Min. DC Drive Current	= 37 mA		802.3
Ro	= Non Transmitting Output Impedance	= 100 kΩ		802.3
N	= Max Nodes per Segment	= 100	Cheapernet	802.3
		= 100	Ethernet	802.3
SK	= Skew Factor, Effect of Encoder Skew on DC Level			802.3
	= (SKEW×4)/100	= 0.02 for 0.5 ns Skew		
R_S	= Max. DC Loop Res. of a Segment			DEFINITION
R_L	= Load Resistance Seen by a Driver			DEFINITION
SEO	= Sending End Overshoot	= 0.10	Cheapernet	ASSUMPTION
		= 0.14	Ethernet	ASSUMPTION

The calculations below explain how the values for the resistor divider in *Figure 1* are obtained. First, collision levels V_{max} and V_{min} must be calculated. The V_{max} or "no detect" level is the maximum DC voltage generated by one node. The worst case here occurs when the transmitting node is at the center of a maximum length cable, and the collision is being detected either by itself or by a station right next to it. On the other hand, the V_{min} or "must detect" level is the minimum DC voltage generated by one minimum transmitting station and another minimum transmitting station at the other end of a maximum length cable.

The filter impulse response is not included in these calculation since it is mutually exclusive with the Sending End Overshoot. If the impulse response is larger than the Sending End Overshoot, the exceeding portion should be added on to the limits.

Maximum Non Collision Level V_{Max} (NO DETECT)—Transmit Mode
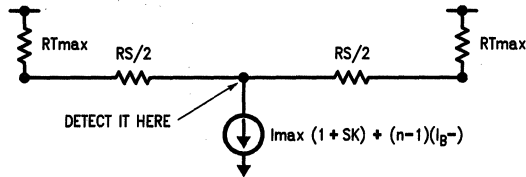
TL/F/10445–1

$R_{Tmax} = R_T \times 1.01 \times [(Tm - 20) \times t_T + 1]$

$R_S = R_{DC} \times L \times [(Tm - 20) \times t_c + 1] + N \times RC$

$R_L = (R_{Tmax} + R_S/2)/2$

$V_{Max} = [I_{Max} \times (1 + SK) + (N - 1)(I_{B-})] \times R_L \times (1 + SEO)$

CHEAPERNET Cable, 300 Meters, 100 Stations:

$R_{Tmax} = 50 \times 1.01 \times [(50 - 20) \times 0.0001 + 1]$
$= 50.652\,\Omega$

$R_S = 0.0489 \times 300 [(50 - 20) \times 0.004 + 1] + 100 \times 0.0034 = 16.770\,\Omega$

$R_L = (50.652 + 16.770/2)/2 = 29.519\,\Omega$

$V_{Max} = [45 \times 1.02 + 99 \times 0.025] \times 29.519 \times 1.10$
$= \mathbf{1571\,mV}$

ETHERNET Cable, 1000 Meters, 100 Stations:

$R_{Tmax} = 50 \times 1.01 \times [(50 - 20) \times 0.0001 + 1]$
$= 50.652\,\Omega$

$R_S = 0.01 \times 1000 [(50 - 20) \times 0.004 + 1] + 100 \times 0.0001 = 11.21\,\Omega$

$R_L = (50.652 + 11.21/2)/2 = 28.129\,\Omega$

$V_{Max} = [45 \times 1.02 + 99 \times 0.025] \times 28.129 \times 1.14$
$= \mathbf{1551\,mV}$

Minimum Collision Level V$_{Min}$ (MUST DETECT)—Transmit Mode
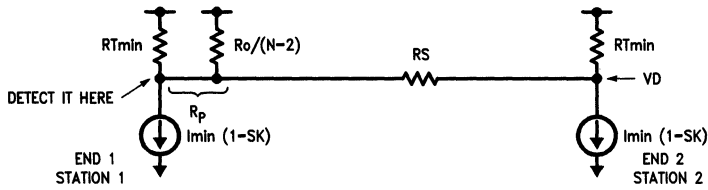
TL/F/10445–2

R_P = Near End Shunt Resistance
= $[R_O/(N - 2)]//R_{Tmin}$
$R_{T_{Min}}$ = $R_T \times 0.99$
VS1(1) = Station 1's DC Voltage at End 1
= $I_{Min} \times (1 - SK) \times [R_P//(R_S + R_{Tmin})]$
VS2(2) = Station 2's DC Voltage at End 2
= $I_{Min} \times (1 - SK) \times [R_{Tmin}//(R_S + R_P)]$
VS2(1) = Station 2's DC Voltage at End 1
= $VS2(2) \times [R_P/(R_S + R_P)] \times SR$
V$_{Min}$ = $VS1(1) + VS2(1)$

CHEAPERNET Cable, 300 Meters, 100 Stations:

R_P = $[100k/98]//(50 \times 0.99)$
= $1020//49.5$ = 47.209Ω
VS1(1) = $37 \times 0.98 \times [47.209//(16.770 + 49.5)]$
= 1000 mV
VS2(2) = $37 \times 0.98 \times [49.5//(16.770 + 47.209)]$
= 1012 mV
VS2(1) = $1012 \times [47.209/(47.209 + 16.770)] \times 0.97$
= 724 mV
V$_{Min}$ = $1000 + 724$ = **1724 mV**

ETHERNET Cable, 1000 Meters, 100 Stations:

R_P = $[100k/98]//(50 \times 0.99) = 1020//49.5$
= 47.209Ω
VS1(1) = $37 \times 0.98 \times [47.209//(11.21 + 49.5)]$
= 963 mV
VS2(2) = $37 \times 0.98 \times [49.5//(11.21 + 47.209)]$
= 972 mV
VS2(1) = $972 \times [47.209/(47.209 + 11.21)] \times 0.94$
= 738 mV
V$_{Min}$ = $963 + 738$ = **1701 mV**

CIRCUIT IMPLEMENTATION

Table III summarizes the design parameters.

TABLE III

Parameter	ETHERNET	CHEAPERNET
L	1000 Meter	300 Meter
N	100	100
V$_{Min}$	1701 mV	1724 mV
V$_{Max}$	1551 mV	1571 mV
R$_1$	125Ω ±1%	150Ω ±1%
R$_2$	10 kΩ ±1%	10 kΩ ±1%

Circuit implementation is shown in *Figure 1*
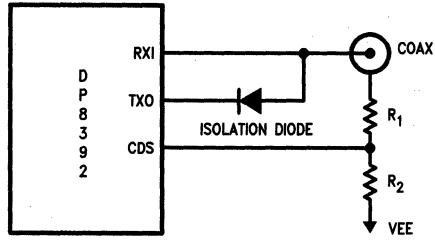
TL/F/10445–3

FIGURE 1

To check the design, subtract the additional offset generated by the resistor divider from these levels (V_{Max} and V_{Min}) and make sure that the internal 8392 collision levels (1450 mV to 1580 mV) are within this window. The supply voltage is assumed to be 9V$\pm$5%.

Ethernet

$1551\,\text{mV} - 8.55\text{V} (125\Omega/(10\,\text{k}\Omega + 125\Omega)) = 1445\,\text{mV}$

$1701\,\text{mV} - 9.45\text{V} (125\Omega/10\,\text{k}\Omega + 125\Omega)) = 1584\,\text{mV}$

Cheapernet

$1571\,\text{mV} - 8.55\text{V} (150\Omega/(10\,\text{k}\Omega + 150\Omega)) = 1445\,\text{mV}$

$1724\,\text{mV} - 9.45\text{V} (150\Omega/(10\,\text{k}\Omega + 150\Omega)) = 1584\,\text{mV}$

These calculations show that the resistor values are properly selected.

Interfacing the DP8392 to 93Ω and 75Ω Cable

The DP8392 Ethernet Coaxial Transceiver Interface (CTI) is designed primarily for 10BASE2 and 10BASE5 applications which use 50Ω coaxial cable. However, with minor modifications it is possible to use this transceiver with larger impedance cables. This article shows how to use the DP8392 with 75Ω or 93Ω cable. The trade off is that segment span is reduced to accommodate for higher series DC resistance of these cables. The CTI is a current driver. The two important factors that must be handled properly in using the chip with 75Ω and 93Ω cables are the dynamic range of the transmitter and collision detection levels.

DYNAMIC RANGE

The dynamic range of the transmitter is important in the following case:

Suppose two stations collide with one-another. To detect collisions properly, each station must sink at least as much DC current as it would in a non-collision case. This would mean that with the 93Ω cable when a collision occurs the chips should be able to sustain approximately −4V DC level. If the signals from the colliding stations are in phase the AC signal could be 8V peak to peak.

The DP8392's transmitter clamps before it pulls to −8V. However, when it clamps it also changes the duty cycle enough to sustain the −4V DC collision level.

An internal diode is included in series with the transmitter's output to isolate its capacitance and thereby minimizing the tap capacitance. For more dynamic range margin, it is recommended that external isolating diodes at the transmitter output not be used. It is also advisable to design the power supply to operate at the higher end of the 8.55V to 9.45V range.

COLLISION LEVELS—RECEIVE MODE

In order to understand the concerns with collision levels, it is necessary to calculate the levels for Cheapernet (10BASE2) 50Ω cable (RG58AU) as an example.

50Ω Cable Example (RG58A/U)

Table I shows the parameter values that are used in calculating the collision levels. Please note that all the levels in this article are for receive mode collision detection.

TABLE I. Assumptions and Definitions

R_T	= Termination Resistor at 20°C	= 50 ±1%	802.3
t_T	= Temp. Coef. of the Terminator	= 0.0001/°C	ASSUMPTION
L	= Maximum Segment Length	= 185m	802.3
R_{DC}	= Maximum Cable DC Res. at 20°C	= 0.0489Ω/m	BELDEN
tc	= Temp. Coef. of Copper	= 0.004/°C	PHYSICS
Tm	= Maximum Cable Temp.	= 50°C	ASSUMPTION
SR	= Step Response at Max Cable Length	= 0.98	NATIONAL
R_C	= Max Connector Res./Station	= 0.0034Ω	MIL SPEC
I_{B+}	= Max Positive Bias Current	= 2 μA	802.3
I_{B-}	= Max Negative Bias Current	= 25 μA	802.3
I_{max}	= Max DC Drive Current	= 45 mA	802.3
I_{min}	= Min. DC Drive Current	= 37 mA	802.3
R_O	= Non Transmitting Output Impedance	= 100 kΩ	802.3
N	= Max Nodes per Segment	= 30	802.3
SK	= Skew Factor, Effect of Encoder Skew on DC Level = (SKEW × 4)/100	= 0.02 for 0.5 ns Skew	802.3
R_S	= Max DC Loop Res. of a Segment		DEFINITION
R_L	= Load Resistance Seen by a Driver		DEFINITION
SEO	= Sending End Overshoot	= 0.08	ASSUMPTION

1

The collision levels that need to be calculated are V_{max} and V_{min}. The V_{max} or "no detect" level is the maximum DC voltage generated by one node. The worst case here occurs when the transmitting node is at the center of a maximum length cable, and the collision is being detected either by itself or by a station right next to it. On the other hand, the V_{min} or "must detect" level is the minimum DC voltage generated by two minimum stations transmitting at one end of a maximum length cable, and the collision is being detected by a node on the other side of the cable.

The filter impulse response is not included in these calculations since it is mutually exclusive with the Sending End Overshoot. If the impulse response is larger than the Sending End Overshoot, the exceeding portion should be added on to the limits.

Maximum Non Collision Level V_{max} (No-Detect)—Receive Mode—50Ω Cable
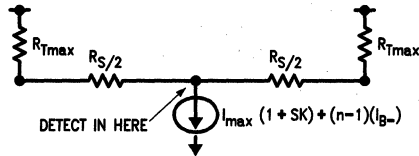
TL/F/10444–1

$$R_{Tmax} = R_T \times 1.01 \times [(Tm - 20) \times t_T + 1]$$
$$R_S = R_{DC} \times L \times [(Tm - 20) \times tc + 1] + N \times R_C$$
$$R_L = (R_{Tmax} + R_S/2)/2$$
$$V_{max} = [I_{max} \times (1 + SK) + (N - 1)(I_{B-})] \times R_L \times (1 + SEO)$$

$$R_{Tmax} = 50 \times 1.01 \times [(50 - 20) \times 0.0001 + 1] \qquad = 50.652\Omega$$
$$R_S = 0.0489 \times 185[(50 - 20) \times 0.004 + 1] + 30 \times 0.0034 \qquad = 10.234\Omega$$
$$R_L = (50.652 + 10.234/2)/2 \qquad = 27.885\Omega$$
$$V_{max} = [45 \times 1.02 + 29 \times 0.025] \times 27.885 \times 1.08 \qquad = \mathbf{1404\ mV}$$

Minimum Collision Level V_{min} (Must-Detect)—Receive Mode—50Ω Cable
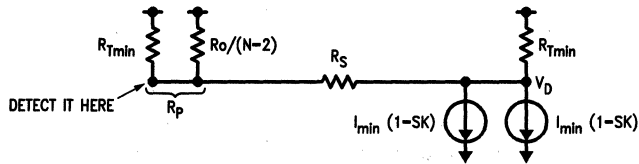
TL/F/10444–2

$$R_P = \text{NEAR END SHUNT RESISTANCE}$$
$$= [R_O/(N - 2)]//R_{Tmin}$$
$$R_{Tmin} = R_T \times 0.99$$
$$V_D = \text{TRANSMITTER'S END DC VOLTAGE}$$
$$= 2 \times I_{min} \times (1 - SK) \times [R_{Tmin}//(R_S + R_P)]$$
$$V_{min} = V_D \times [R_P/(R_S + R_P)] \times SR$$

$$R_P = [100k/28]//(50 \times 0.99) = 3571//49.5 \qquad = 48.823\Omega$$
$$V_D = 2 \times 37 \times 0.98 \times [49.5//(10.234 + 48.823)] \qquad = 1952\ mV$$
$$V_{min} = 1952 \times [48.823/(10.234 + 48.823)] \times 0.98 \qquad = \mathbf{1581\ mV}$$

The calculations show that the V_{max} and V_{min} are properly placed outside the collision threshold range of the DP8392 (1450 mV to 1580 mV).

93Ω Cable Collision Level Calculation

A few parameters need to be changed when using a different impedance cable. Here are those parameters for 93Ω cable (RG62A/U TYPE, BELDEN 9269);

TABLE II

R_T	= termination resistor at 20°C	= 93 ±1%
L	= maximum segment length	= 130m
R_{DC}	= maximum cable DC res. at 20°C	= 0.1437 Ω/m BELDEN

Considering the new values the V_{max} and V_{min} levels are;

Maximum Non Collision Level V_{max} (No Detect)—Receive Mode—93Ω Cable

R_{Tmax}	$= 93 \times 1.01 \times [(50 - 20) \times 0.0001 + 1]$	$= 94.212\Omega$
R_S	$= 0.1437 \times 130[(50 - 20) \times 0.004 + 1] + 30 \times 0.0034$	$= 21.025\Omega$
R_L	$= (94.212 + 21.025/2)/2$	$= 52.362\Omega$
V_{max}	$= [45 \times 1.02 + 29 \times 0.025] \times 52.362 \times 1.08$	**= 2636.692 mV**

Minimum Collision Level V_{min} (Must Detect)—Receive Mode—93Ω Cable

R_P	$= [100k/28]//(93 \times 0.99) = 3571//92.07$	$= 89.756\Omega$
V_D	$= 2 \times 37 \times 0.98 \times [92.070//(21.025 + 89.756)]$	$= 3646.396$ mV
V_{min}	$= 3646.396 \times [89.756/(21.025 + 89.756)] \times 0.98$	**= 2895.272 mV**

93Ω IMPLEMENTATION WITH DP8392

Figure 1 shows the connection diagram with 93Ω cable (100 meters and 30 stations). The design parameters defined below are summarized in Table III. The resistor divider ratio needs to be calculated to attenuate the receiver input signal. The two resistors R_1 and R_2 should center the calculated thresholds (2636 mV to 2895 mV) to the internal level of DP8392 (1450 mV to 1580 mV).

The resistor divider and the capacitor Cp, *Figure 1*, (Cp includes the RXI input capacitance, typically 1 pF, and the pc trace capacitance associated with it) form a low pass filter effect. It may be necessary to add the capacitor Cc (capacitor Cc creates a high pass effect) to compensate the low pass effect. The equation to calculate the capacitor Cc is;

$$Cc \times R_2 = Cp \times R_1$$

TABLE III

CABLE	BELDEN RG62A/U Type 93Ω Cable
L	130 meters
R_{DC}	0.1437 Ω/m
N	30
R_1	54.8k
R_2	45.2k

It is also necessary to add the resistor R_3 ($R_3 = R_1//R_2$) in series with the CDS pin. This will assure that the voltage drop due to the biasing currents into CDS and RXI pins are duplicated.

To check the design;

$[54.8k/(54.8k + 45.2k)] \times 2636$ mV = 1444 mV
$[54.8k/(54.8k + 45.2k)] \times 2895$ mV = 1586 mV

The DP8932's internal collision range is within this window.

75Ω CABLE IMPLEMENTATION

This method can also be successfully implemented for 80 meters of 75Ω cable (RG59/U BELDEN 8241). The collision thresholds are 2127.8 mV and 2339.6 mV. The corresponding R_1 and R_2 values are 67.8 kΩ and 32.2 kΩ respectively. Table IV summarizes the design parameters.

TABLE IV

CABLE	BELDEN RG59/U 75Ω Cable
L	80 meters
R_{DC}	0.1894 Ω/m
N	30
R_1	67.8k
R_2	32.2k
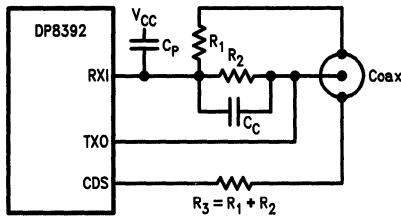
FIGURE 1

TL/F/10444–3

DP839EB Network Evaluation Board

National Semiconductor
Application Note 479

GENERAL DESCRIPTION

The DP839EB LAN Evaluation Board provides IBM PCs, IBM PC ATs and compatibles with Ethernet, Thin-Ethernet, Twisted Pair Ethernet and StarLAN connections. The DP839EB is compatible with the PC-bus and requires only a ½ size slot for installation.

The LAN Evaluation Board utilizes the National Semiconductor IEEE 802.3 chip set consisting of the DP8390 Network Interface Controller, the DP8391 Serial Network Interface Adapter, and the DP8392 Coaxial Transceiver Interface. The dual DMA capabilities of the DP8390, coupled with 8 kBytes of local buffer RAM allow the evaluation board to appear as an I/O port to the system.

The Evaluation Board provides the designer with a good example of an 8-bit controller implementation, that can be extrapolated to 16-bit buses.

In addition, software is included with the board to facilitate low level functional check out of the board, and as a guide to programming the board. The source code is provided, and can serve as a model for additional software development.

Additionally the DP839EB is compatible with Novell NetWare™, and TCP/IP software support is available through third party developers.

HARDWARE FEATURES

- Half-size IBM PC card form factor
- DP8390 IEEE 802.3 Chip Set utilizing dual DMA controller
- 8 kByte on board packet buffer
- Interfaces to PC using DMA, or to an AT via string I/O instructions
- Ethernet connector (15 pin D), Thin-Ethernet connector (BNC)
- StarLAN with optional daughter card
- Low power operation
- Utilizes DP8390, DP8391, and DP8392

SOFTWARE FEATURES

- No Software changes for conversion between Ethernet/Cheapernet and StarLAN
- Demonstration and diagnostic software available

NETWORK INTERFACE OPTIONS

The evaluation board supports three physical layer options: Ethernet, Cheapernet and StarLAN. When using Ethernet, a drop cable is connected to an external transceiver which is connected to a standard Ethernet network. (See *Figure 1*). When using Cheapernet, a low cost version of Ethernet, a transceiver is available on-board allowing direct connection to the network via the evaluation board. (See *Figure 2*). When using a StarLAN network, an optional daughter card replaces the SNI function and implements the required electronics to interface the DP8390 NIC to StarLAN. This configuration is illustrated in *Figure 3*. No software changes are needed for conversion between any of the described configurations.

HARDWARE DESCRIPTION

The block diagram shown in *Figure 4* illustrates the architecture of the Network Interface Adapter. The system/network interface is partitioned at the DP8390 Network Interface Controller (NIC). The NIC acts as both a master and a slave on the local bus. During reception or transmission of packets, the NIC is a master. When accessed by the PC, the NIC becomes a slave. The NIC utilizes a local 8-bit data bus connected to an 8k x 8 Static RAM for packet storage. The 8k x 8 RAM is partitioned into a transmit buffer and a receive buffer. All outgoing packets are first assembled in the packet buffer and then transmitted by the NIC. All incoming packets are placed in the packet buffer by the NIC and then transferred to the PC's memory. The transfer of data between the evaluation board and the PC is accomplished using the PC's DMA in conjunction with the NIC's Remote DMA. Two LS374 latches implement a bidirectional I/O port with the PC bus. The 8-bit transceiver (LS245) allows the PC to access the NIC's internal registers for programming. A 32 x 8 PROM located on the evaluation board contains the unique Physical Address assigned to each board.
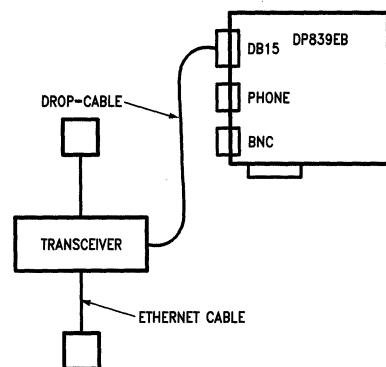
TL/F/9179-1

FIGURE 1. Ethernet Connection

TL/F/9179–2
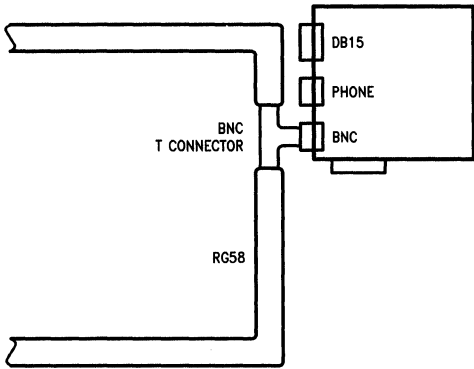
FIGURE 2. Cheapernet Connector

TL/F/9179–3
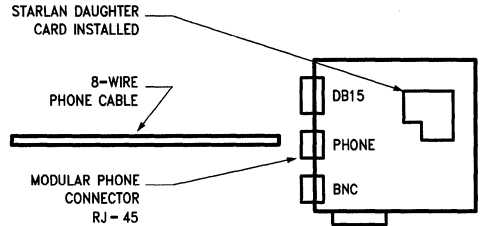
FIGURE 3. StarLAN Connector

TL/F/9179–4
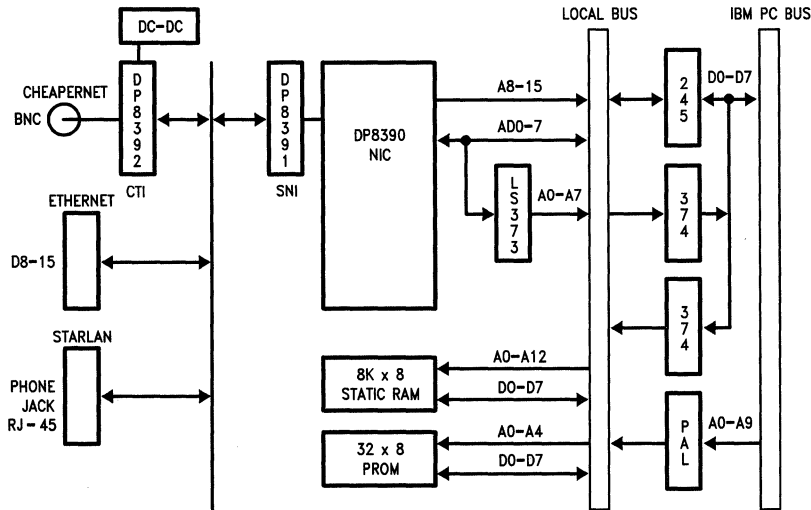
FIGURE 4

Since the NIC is accessing 8-bit memory, only a single de-multiplexing latch is required for the lower 8-bits of address. An LS373 is provided for this purpose.

A 20L8 PAL provides the address decoding and support for DMA handshaking and wait state generation.

SOFTWARE SUPPORT

The evaluation board provides a simple programming interface for development of software. Several software packages are provided for evaluation and development of networks using the DP8390 Chip set. SDEMO is a demonstration program that provides a low level interface to the DP8390 NIC for transmission and reception of packets. SDEMO supports register dumps and simple register modification. CONF is a conferencing program which supports simple message transfer. WORKSTAT and SERVER support file transfer between two nodes, one configured as a server and a second configured as a workstation. NLS, Network Load Simulator, is a program that simulates network loads based on statistical distributions of packet sizes,

bursts and intervals. NLS is useful for performance measurement and debug of software drivers. NES, Network Evaluation Software, consists of sample software drivers implementing a low level interface to the evaluation board.

LOCAL MEMORY MAP

The DP8390 NIC accesses an 8k x 8 buffer RAM located in its 64 kbyte memory space. This buffer RAM is used for temporary storage of receive and transmit packets. Data from this RAM is transferred between the host (the PC) and the evaluation board using the DP8390 NIC's remote DMA channel. An ID address PROM, containing the physical address of the evaluation board is also mapped into the memory space of the NIC.

Note: Partial decoding is performed on the PROM and RAM which will result in these devices appearing at other locations in the 64k memory space. The first occurrence of the PROM and RAM are used for programming purposes.

```
0000h
001fh          ┌──────────────────┐
   •           │       PROM        │
   •           │        •          │
   •           │        •          │
2000h          ├──────────────────┤
3fffh          │     8k x 8        │
   •           │   BUFFER RAM      │
   •           │        •          │
   •           │        •          │
ffffh          │        •          │
               └──────────────────┘
```

Address	Contents
00h	ADDRESS 0 (Physical Address Most Significant Byte)
01h	ADDRESS 1
02h	ADDRESS 2
03h	ADDRESS 3
04h	ADDRESS 4
05h	ADDRESS 5 (Physical Address Least Significant Byte)
06h	CHECKSUM (XOR OF ADDRESS 0-5) OPTIONAL
07h	REV. NUMBER
08h	MANUFACTURE LOT #
09h	MANUFACTURE DATE (MONTH)
10h	MANUFACTURE DATE (YEAR)
11h–1fh	RESERVED

PROM FORMAT

Each evaluation board is assigned a unique network (physical) address. This address is stored in a 74S288 32 x 8 PROM. The physical address is followed by a checksum. The checksum is calculated by exclusive OR-ing the 6 address bytes with each other. At initialization the software reads the PROM, verifies the checksum and loads the NIC's physical address registers. The following format is used in the PROM:

FIGURE 5

TL/F/9179-5

1-104

I/O SPACE

The I/O space and Ethernet/Cheapernet configurations are selected using the various I/O jumpers. There are 4 sets of jumpers that should be programmed prior to installation of the evaluation board into the PC environment. There are:

J4   I/O address, interrupt selection, DMA channel assignment

J1C–J7C, J7E   Select Ethernet or Cheapernet

Figure 5 depicts the location of the jumpers on the evaluation board.

The Factory Installed Configuration Is:

J4   I/O base = 300h

         Interrupt = IRQ3

         DMA = DREQ1, DACK1

J1C–J7C, J7E   Cheapernet selected

The evaluation board uses 32 I/O locations in the PC's I/O space. The base address is fixed at 300h and is not selectable using jumpers. (See Switch settings section.) The I/O map is shown below:

BASE + 00h	COMMAND REGISTER
01h	NIC REGISTER
02h	SPACE
03h	•
04h	•
05h	•
06h	•
07h	•
•	•
0fh	•
10h	I/O PORTS
•	•
1fh	•

NOTES: The NIC's Command Register is always mapped at Base + 0. The NIC registers are Base + 01 to Base + 0f; 0f will contain different registers depending on the value of bits PS0 and PS1 in the Command Register. These two bits select one of three register pages. For additional information consult the DP8390 data sheet.

The NIC uses the remote DMA channel to read/write data from/to the 8k x 8 Buffer RAM on the evaluation board. Typically a DMA channel on the PC is used in conjunction with the NIC's remote DMA. The I/O ports are then serviced by the DMA channel. If a DMA channel on the PC is not available, the NIC's DMA can still be used by accessing the I/O ports using programmed I/O. Reading the I/O port address will result in a $\overline{RACK}$ strobe to the NIC while writing the I/O port address will result in a $\overline{WACK}$ strobe to the NIC.

SWITCH SETTINGS

Jumper J4 allows assignment of I/O Address Bases, DMA channel assignments and Interrupt Request assignments. The jumper configuration is shown below and described in the following sections.

| IRQ2 | IRQ3 | IRQ4 | IRQ5 | DREQ1 | DREQ3 | IRQ5 | DACK1 | DACK3 | BASE2 | BASE3 |
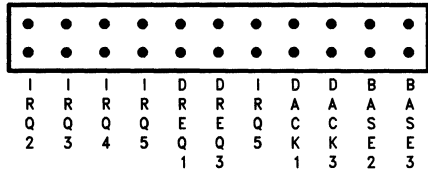
TL/F/9179–6

I/O BASE ADDRESS

The I/O Base Address for DP8390B boards is fixed at 300h and is not selectable.

INTERRUPTS

The NIC will generate interrupts based on received and transmitted packets, completion of DMA and other internal events. The interrupt can be connected to Interrupts 2, 3, 4 or 5 (IRQ 2, 3, 4, 5) via Jumper J4. Interrupt 5 is also provided as a software driven DMA Channel. If Interrupt 5 is being used as a DMA channel Interrupt 5 cannot be chosen for the NIC interrupt. The figures below illustrate the jumper positions for the various interrupt levels.

Interrupt 2

TL/F/9179–9

Interrupt 3
(Factory Installed)

TL/F/9179–10

Interrupt 4
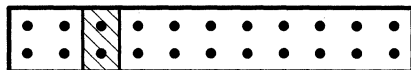
TL/F/9179–11

Interrupt 5
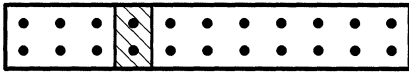
TL/F/9179-25

Note: Rev D demo software will not work unless the factory configuration for Jumper Block J4 is used.

FACTORY CONFIGURATION:
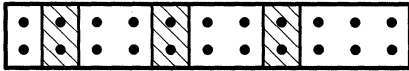
TL/F/9179-26

DMA

The evaluation board may use 1 DMA channel on the PC expansion bus. DMA channel 1 or 3 can be selected. The corresponding DACK line must also be installed on Jumper J4.

DMA Channel 1
(Factory Installed)
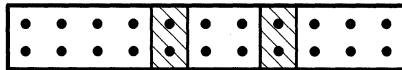
TL/F/9179-15

DMA Channel 3

TL/F/9179-16

If a DMA channel is not available an interrupt driven routine can be used to move data between the PC and the buffer memory on the evaluation board. Interrupt 5 is used for this function.

IRQ 5 for DMA
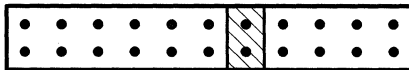
TL/F/9179-17

SELECTING ETHERNET OR CHEAPERNET

Two 10 Mbit/sec Interface options are available, a connection to an external transceiver via the DB-15 connector, or a direct interface to a BNC T-connector. Seven jumpers are used to select the appropriate option. These jumpers are labeled J1C-J7C and J7E.

For Cheapernet the following jumpers should be shorted:
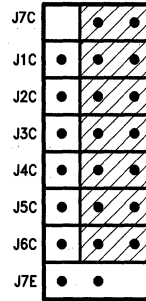
TL/F/9179-18

(Factory Installed)

For Ethernet the following jumpers should be shorted.
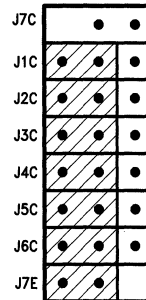
TL/F/9179-19

Double check the jumper positions prior to powering up the board.

OSCILLATOR

When the StarLAN daughter board is used, the 20 MHz oscillator must be disconnected by removing jumper JB. The StarLAN daughter board provides the clock to the NIC.

Ethernet, Cheapernet
(Factory Installed)
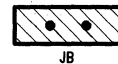
StarLAN
(Removed)

APPENDICES

The remainder of this document contains the evaluation board parts list, schematic and PAL descriptions.

TL/F/9179-21

PARTS LIST*

Item #	Description	Reference Designator	Qty
1	RES. CC 4.7K Ω 1/4W 5%	R1, R2, R3, R23	4
2	RES. CF 39 Ω 1/4W 5%	R6, R7, R8, R9	4
3	RES. CF 1.5K Ω 1/4W 5%	R10, R11, R12, R13	4
4	RES. CF 1M Ω 1/2W 5%	R17	1
5	RES. CF 270K Ω 1/4W 1%	R4, R5	2
6	RES. MK 1K Ω 1/4W 1%	R14	1
7	CAP. FILM 0.01μF 630V	C4	1
8	CAP. DIP TANT 100 μF 10V RD	C3, C21	2
9	CAP. DIP 0.47 μF 50V 0.3LS	C1, C7–C17, C19	13
10	CAP. CER 0.01 μF 50V 0.2LS	C5, C6	2
11	I.C. 74LS245	U3	1
12	I.C. 74LS374	U2, U6	2
13	I.C. 74LS373	U1	1
14	SRAM HM6264-1OO	U8	1
15	PROM 74S288	U4	1
16	PAL20L8	U16	1
17	TRANSFORMER PE64103	U14	1
18	OSCILLATOR 20.00 MHz	Y1	1
19	JUMPER, 2 POSITION	A/R	13
20	CONN. 15 POS D-SUB	J1	1
21	CONN. MODULAR JACK		1
22	CONN. BNC, R/A PCB MOUNT	J2	1
23	HEADER, 2 PIN SINGLE ROW	JB, J7C, J7E	3
24	HEADER, 3 PIN SINGLE ROW	J1C–J6C	6
25	HEADER, 44 PIN DOUBLE ROW	J4	0.5
26	SOCKET, 24 PIN DIP	U11	2
27	SOCKET, 24 PIN DIP (.300)	U16	1
28	SOCKET, 24 PIN (MACH)	U9	1
29	BRACKET, CNET	J1	1
30	SPACER, D-25 SET	J1	1
31	PCB		
32	DC-DC CONVERTER, 2VP5U9	U10	1
33	I.C. DP8390BN	U11	1
34	I.C. DP8391N	U9	1
35	I.C. DP8392AN	U15	1

*551 A201-01 REV D Board

1

```
PAL20L8
Decode and DMA Interface Logic for the DP839EB
DECODE1
A9    A8    A7    A6    A5    A4  PCRST /NMRD  NA13 /IORD  /IOWR GND
PRQ /DACK /WAIT AEN /RACK /WACK /CSX /CSN  /NICRST  /CSROM /ACK  VCC


CSN =  /AEN* A9* A8* /A7* /A6* /A5* /A4* IOWR +
       /AEN* A9* A8* /A7* /A6* /A5* /A4* IORD


NICRST = PCRST


RACK = /AEN* A9* A8* /A7* /A6* /A5* A4* PRQ* IORD +
       DACK*  IORD


WACK = /AEN* A9* A8* /A7* /A6* /A5* A4* PRQ* IOWR +
       DACK*  IOWR


CSX =  CSN* IORD +
       CSN* IOWR +
       /AEN* A9* A8* /A7* /A6* /A5* A4* IORD +
       /AEN* A9* A8* /A7* /A6* /A5* A4* IOWR +


IF (CSX)
WAIT = /ACK *  CSN  +
       /PRQ *  /CSN


CSROM = /NA13 * NMRD
```

DESCRIPTION

This PAL performs the I/O decodes for selecting the NIC, and the handshake signals for NIC's remote DMA. The PAL supports the DMA channels of the PC for remote DMA transfers with the NIC and also allows the use of string I/O between 80286 PC's and NIC's remote DMA.

DECODE1 fixes the I/O BASE of the card at 300h. NIC registers fall in the space 300h–30fh. To use the string I/O port, reads and writes are done to port 310h.

Wait states are inserted (WAIT) to the PC bus when register accesses are given and the NIC is busy performing DMA operations. When the NIC is ready, /ACK is given and no (more) wait states are inserted.

Wait states may also be inserted during remote DMA operations and 80286 machines using string I/O's. WAIT occurs during a remote read if the PC AT's /IORD goes low before the DP8390's PRQ goes high. Similarly, WAIT occurs during

a remote write if the PC AT's /IOWR goes low before the NIC's PRQ goes high.

NIC registers are accessed when CSN (Chip Select NIC) is asserted. The IORD and IOWR terms are included to ensure that the address lines are valid when CSN is given.

The RACK and WACK signals are used by the NIC's remote DMA channel to acknowledge the end of a single read or write operation through the remote DMA I/O ports. These ports are addressable by the PC DMA channel with DACK and IORD or IOWR, or by addressing the I/O location 310h (with string I/O's).

CSX is used to enable the TRI-STATE output of WAIT during a register access CSN), and during string I/O to the remote DMA's I/O port (CSX).

CSROM provides address decode for the address PROM. The card's unique Ethernet address is transferred to the system using the NIC's remote DMA.
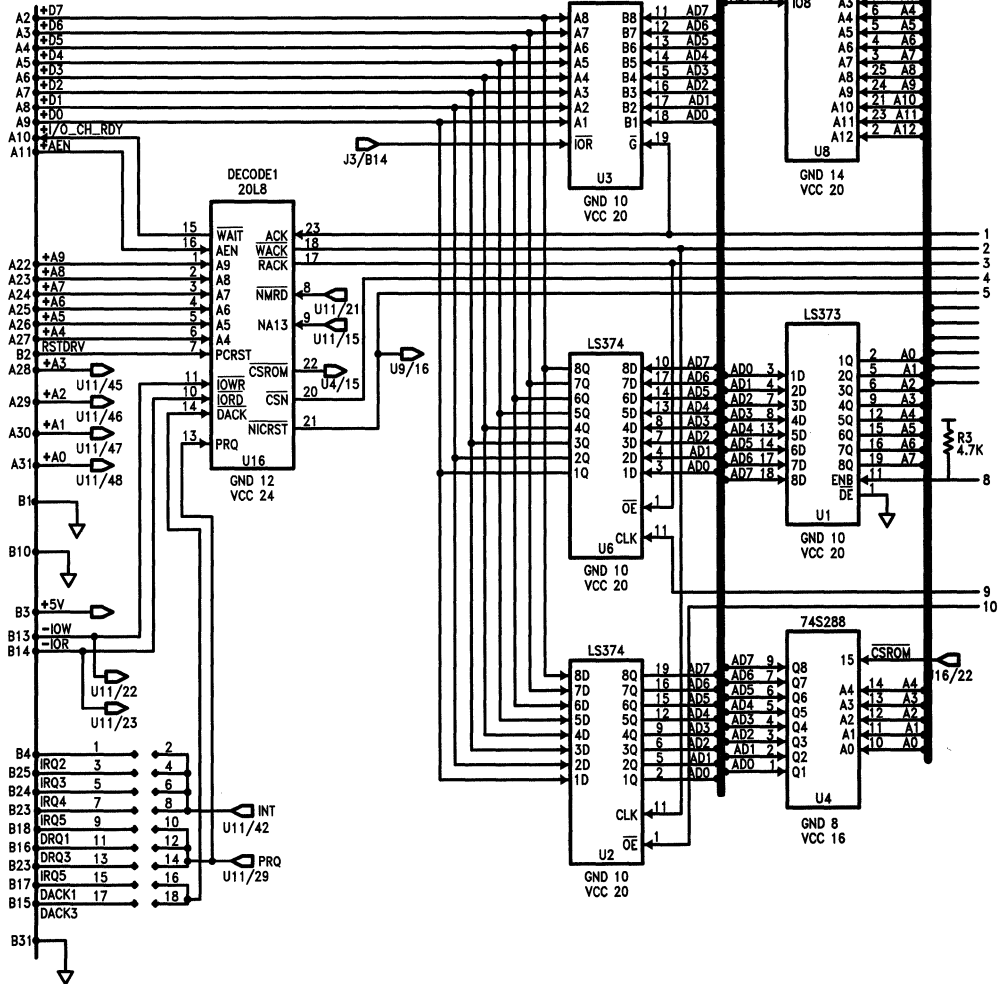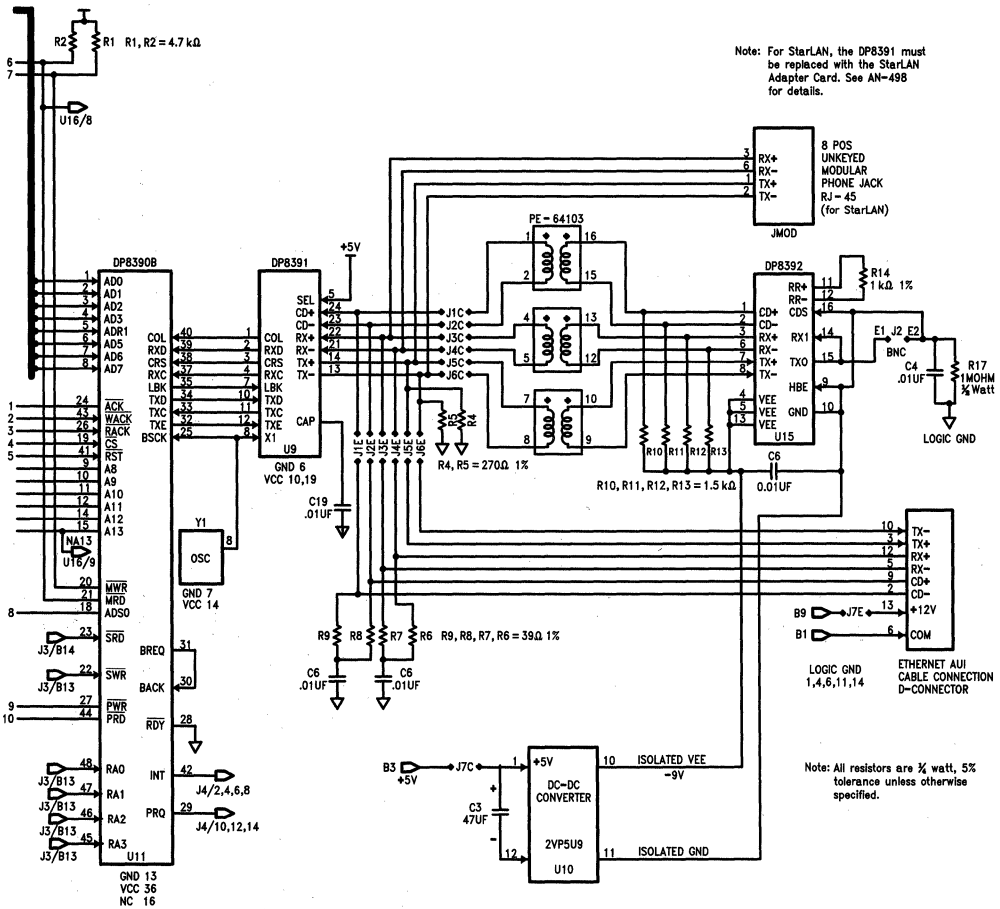
TL/F/9179-23

Note: For StarLAN, the DP8391 must
be replaced with the StarLAN
Adapter Card. See AN-498
for details.

R2 R1 R1, R2 = 4.7 kΩ

6
7
U16/8

8 POS
UNKEYED
MODULAR
PHONE JACK
RJ-45
(for StarLAN)

3 RX+
6 RX−
1 TX+
2 TX−
JMOD

PE-64103

DP8390B
1 AD0
2 AD1
3 AD2
4 AD3
5 AD4
6 AD5
7 AD6
8 AD7

COL 40
RXD 39
CRS 38
ADR1 37
RXC 36
LBK 35
TXD 34
TXC 33
TXE 32
BSCK 25

DP8391
SEL 5
CD+ 24
CD− 23
RX+ 22
RX− 21
TX+ 14
TX− 13

1 COL
2 RXD
3 CRS
4 RXC
5 LBK
10 TXD
11 TXC
12 TXE
8 X1
CAP

GND 6
VCC 10,19

DP8392
RR+ 11
RR− 12
CDS 16

1 CD+
2 CD−
3 RX+
6 RX+
7 RX−
8 TX−

RX1 14
TXO 15
HBE 7

VEE 4
VEE 5
VEE 13
GND 10
U15

R14
1 kΩ 1%

E1 J2 E2
BNC

C4
.01UF

R17
1MOHM
¾ Watt

LOGIC GND

24 ACK
43 WACK
26 RACK
19 CS
41 RST
9 A8
10 A9
11 A10
12 A11
14 A12
15 A13

NA13
U16/9

20 MWR
21 MRD
18 ADSO

J1C
J2C
J3C
J4C
J5C
J6C

R5 R4
R4, R5 = 270Ω 1%

R10 R11 R12 R13
R10, R11, R12, R13 = 1.5 kΩ

C6
0.01UF

Y1
OSC 8

GND 7
VCC 14

C19
.01UF

J1E
J2E
J3E
J4E
J5E
J6E

10 TX−
3 TX+
12 RX−
5 RX+
9 CD+
2 CD−

23 SRD
J3/B14

22 SWR
J3/B13

27 PWR
44 PRD

BREQ 31

BACK 30

RDY 28

B9 J7E 13 +12V
B1 6 COM

LOGIC GND
1,4,6,11,14

ETHERNET AUI
CABLE CONNECTION
D-CONNECTOR

9
10

R9 R8 R7 R6 R9, R8, R7, R6 = 39Ω 1%

C6
.01UF

C6
.01UF

48 RA0
J3/B13

47 RA1
J3/B13

46 RA2
J3/B13

45 RA3
J3/B13

INT 42
J4/2,4,6,8

PRQ 29
J4/10,12,14

U11
GND 13
VCC 36
NC 16

B3 J7C 1 +5V
+5V

DC-DC
CONVERTER

2VP5U9
U10

10 ISOLATED VEE
−9V

C3
47UF

12 11 ISOLATED GND

Note: All resistors are ¾ watt, 5%
tolerance unless otherwise
specified.

TL/F/9179-24

Ethernet/Cheapernet Physical Layer Made Easy with DP8391/92

With the integration of the node electronics of IEEE 802.3 compatible local area networks now on silicon, system design is simplified. This application note describes the differences between the Ethernet and Cheapernet versions of the standard, and provides design guidelines for implementing the node electronics with National Semiconductor's DP8390 LAN chip set.

INTRODUCTION

The DP8390 chip set is designed to provide the physical and media access control layer functions of local area networks as specified in IEEE 802.3 standard. This standard is based on the access method known as carrier-sense multiple access with collision detection (CSMA/CD). In this scheme, if a network station wants to transmit, it first "listens" to the medium; if someone else is transmitting, the station defers until the medium is clear before it begins to transmit. However, two or more stations could still begin transmitting at the same time and give rise to a collision. When this happens, the two nodes detect this condition, back off for a random amount of time before making another attempt.

The IEEE 802.3 standard supports two different versions for the media, 10BASE5 (commonly known as Ethernet) and 10BASE2 (Cheapernet). These can be used separately, or together in a hybrid form. Both versions have similar electrical specifications and can be implemented using the same transceiver chip (DP8392). Cheapernet is the low cost version and is user installable. The following table compares the two:

Parameter	10BASE5 (Ethernet)	10BASE2 (Cheapernet)
Data Rate	10 Mbit/s baseband	10 Mbits/s baseband
Segment Length	500 m	185 m
Network Span	2500 m	925 m
Nodes per Segment	100	30
Node Spacing	2.5 m (cable marked)	0.5 m min
Capacitance per Node	4 pF max	8 pF max
Cable	0.4 in diameter 50Ω Double Shielded Rugged N-Series Connectors	0.2 in diameter 50Ω (RG58A/U) Single Shielded Flexible BNC Connectors
Tranceiver Drop Cable	0.39 in diameter multiway cable with 15 pin D connectors 50 m max length	Not needed due to the high flexibility of the RG58A/U cable

Typical Connection Diagram for a Station

10BASE5 (Ethernet)

TL/F/8689–1

10BASE2 (Cheapernet)
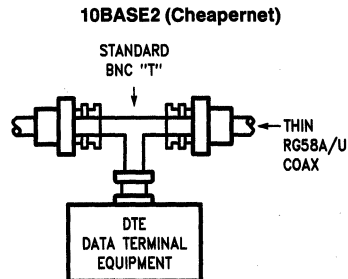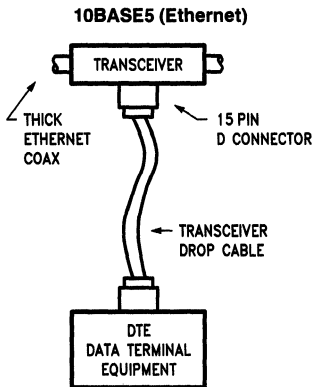
TL/F/8689–2

Although Cheapernet is intended for local use, several 185 meter segments can be joined together with simple repeaters to provide for a larger network span. Similarly, several Cheapernet segments can be tied into a longer Ethernet "backbone". In this hybrid configuration, the network combines all the benefits of Cheapernet, flexibility and low cost, with the ruggedness and the much larger geographic range of standard Ethernet. *Figure 1* illustrates a typical hybrid LAN configuration.
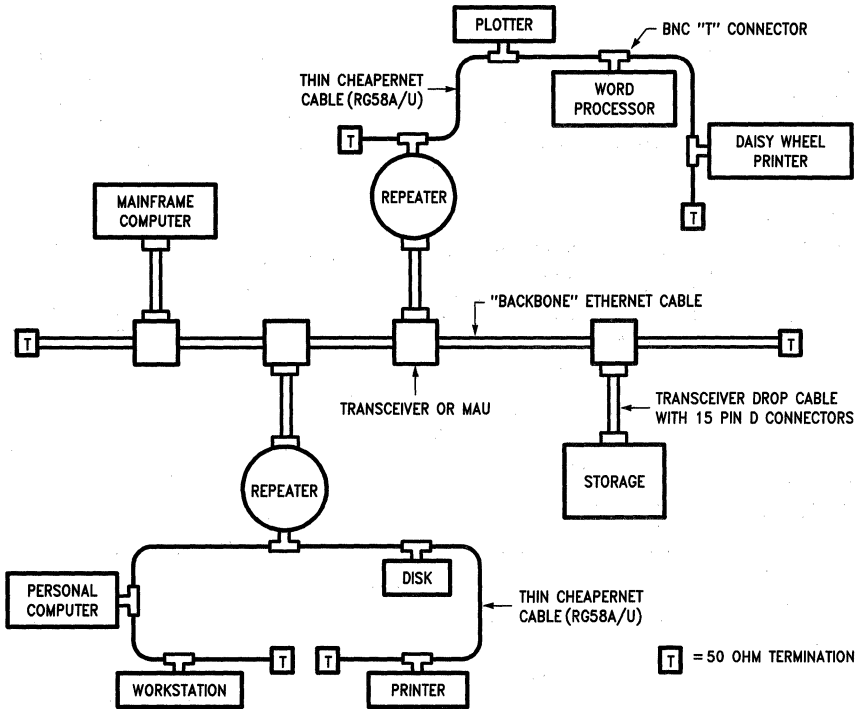
TL/F/8689–3

FIGURE 1. A Hybrid Ethernet/Cheapernet System

TRANSMITTING AND RECEIVING PACKETS WITH THE DP8390 CHIPSET

Node Block Diagram

TL/F/8689–4
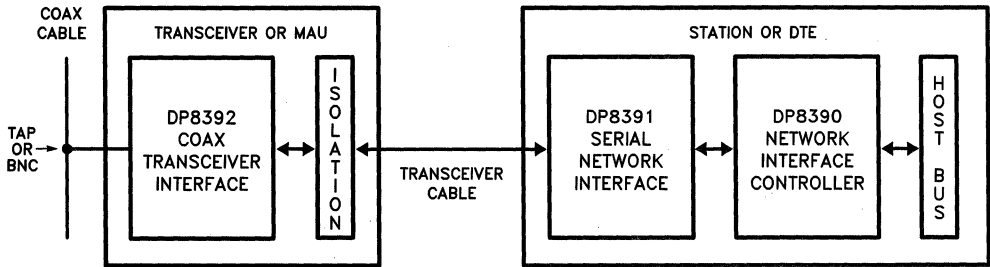
The node electronics is integrated into three chips, the DP8390 Network Interface Controller (NIC), the DP8391 Serial Network Interface (SNI), and the DP8392 Coaxial Transceiver Interface (CTI). To transmit a packet, the host processor issues a transmit command to the NIC, which normally transfers the data to a local buffer memory. The NIC then automatically handles the transmission of the packet (from the local buffer through an on-board FIFO to the SNI) according to the CSMA/CD protocol. The packet has to be in the following format:

PREAMBLE	SFD	DESTINATION	SOURCE	LENGTH	DATA	CRC
62-bits	2-bits	6-bytes	6-bytes	2-bytes	46–1500 bytes	4-bytes

PREAMBLE: This section consists of alternating 1 and 0 bits. As the packet travels through the network, some of these bits would be lost as most of the network components are allowed to provide an output some number of bits after being presented with a valid input.

START OF A FRAME DELIMITER (SFD): This field consists of two consecutive 1's to signal that the frame reception should begin.

DESTINATION AND SOURCE ADDRESSES: Each one of these frames is 6 bytes long and specifies the address of the corresponding node.

LENGTH: This 2 byte field indicates the number of bytes in the data field.

DATA: This field can be from 46 to 1500 bytes long. Messages shorter than 46 bytes require padding to bring the data field to the minimum length. If the data field is padded, the host can determine the number of valid data bytes by looking at the length field. Messages longer than 1500 bytes must be broken into multiple packets.

CRC: This field contains a Cyclic Redundancy Code calculation performed on the Destination address through the Data field for error control.

The shortest packet length thus adds up to be 512 bits long (excluding the preamble and the SFD). At 10 Mbit/sec this amounts to 51.2 μs, which is twice as much as the 25 μs maximum end-to-delay time that is allowed by the IEEE 802.3 protocol. This ensures that if a collision arises in the network, it would be recognized at all node locations.

The SNI combines the NRZ data packet received from the controller with a clock signal and encodes them into a serial bit stream using standard Manchester encoding. In this coding scheme, the first half of the bit cell contains the complementary data and the second half contains the true data. Thus a transition is always guaranteed in the middle of a bit cell.
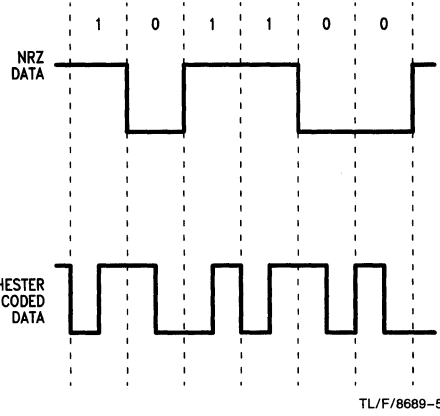
TL/F/8689–5

FIGURE 2. Manchester Coding

The encoded signal appears in differential form at the SNI's output. In 10BASE5 (Ethernet) applications, this signal is sent to the transceiver or the Medium Attachment Unit (MAU) through the twisted pair Tranceiver Drop cable (also known as the Attachment Unit Interface cable). This cable typically consists of four individually shielded twisted wire pairs with an overall shield covering these individually shielded pairs. The signal pairs, which have a differential characteristic impedance of 78Ω $\pm$5Ω, should be terminated at the receiving ends. The cable can be up to 50 meters in length and have a maximum delay of 257 ns. The shields of the individual pairs should be connected to the logic ground in the Data Terminal Equipment (DTE) and the outer shield to the chassis ground. *Figure 3* shows a picture of the cable and the corresponding pin assignments.
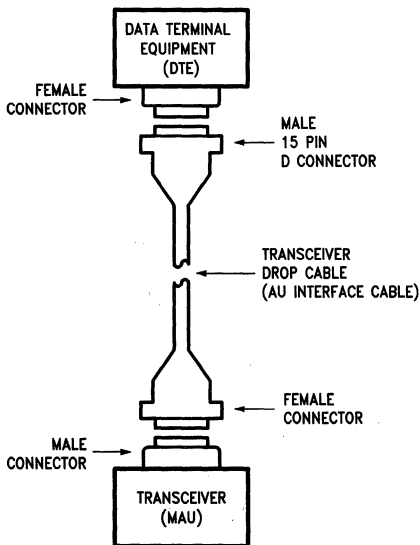
TL/F/8689–6

FIGURE 3. Transceiver Cable Pin Assignments

Pin	IEEE 802.3 Name	Pairs	DP8391/2 Name	Signal from DTE	Signal from MAU
3	DO + (Data Out +)	Transmit Pair	TX+	X	
10	DO − (Data Out −)		TX−	X	
11	DO S (DO Shield)			X	
5	DI + (Data In +)	Receive Pair	RX+		X
12	DI − (Data In −)		RX−		X
4	DI S (DI Shield)			X	
7	CO + (Control Out +)	Optional Pair		X	
15	CO − (Control Out −)			X	
8	CO S (CO Shield)			X	
2	CI + (Control In +)	Collision Pair	CD+		X
9	CI − (Control In −)		CD−		X
1	CI S (CI Shield)			X	
6	VC (Voltage Common)	Power Pair		X	
13	VP (Voltage Plus)			X	
14	VS (Voltage Shield)			X	
Shell	PG (Protective GND)			X	

The transmitted packet from the SNI as well as all other signals (receive, collision, and DC power) must be electrically isolated from the coax in the MAU. The isolation means provided must withstand 500 V_{AC} rms for one minute for 10BASE2 and 2000 V_{AC} rms for 10BASE5. In order to detect collisions reliably, the electrical isolation is not done at the coax; instead it is done on the side of the Attachment Unit Interface. The isolation for the three signal lines can be easily provided by using three pulse transformers that come in a standard 16 pin plastic DIP from several manufacturers (Pulse Engineering, Valor Electronics). The inductance value for these transformers vary from 50 μH to 150 μH with the larger inductance values slowing the rise and fall times, and the smaller ones causing more voltage droop.

The Manchester encoded data from the SNI now reaches the CTI's transmit input after passing through the isolation transformer. A noise filter at this input provides a static noise margin of -175 mV to -300 mV. These thresholds assure that differential Transmit (TX$\pm$) data signals less than -175 mV or narrower than 10 ns are always rejected, while signals greater than -300 mV and wider than 30 ns are always accepted. The -300 mV threshold provides sufficient margin since the differential drivers for the transceiver drop cable provide a minimum signal level of ±450 mV after inductive droop, and the maximum attenuation allowed for the drop cable is 3 dB at signal frequencies. Signals meeting the squelch requirements are waveshaped and outputted to the coax medium. This is done as follows:

The transmitter's output driver is a switching current source that drives a purely resistive load of 25Ω presented by the coax to produce a voltage swing of approximately 2V. This signal has to meet several critical electrical requirements:

RISE/FALL TIMES: The 10%–90% rise and fall times have to be 25 ns ±5 ns at 10 Mbit/sec. This spec helps to minimize electro-magnetic radiation by reducing the higher harmonic content of the signal and contributes to the smaller reflection levels on the coax. In addition, the rise and fall times are required to be matched to within 1 ns to minimize the overall jitter in the system.

DC LEVEL: The DC component of the signal has to be between -37 mA and -45 mA. The tolerance here is tight since collisions are detected by monitoring the average DC level on the coax.

AC LEVEL: The AC component of the signal has to be between ±28 mA and the DC level. This specification guarantees a minimum signal at the far end of the coax cable in the worst case condition.

The signal shown in Fig. 4 would be attenuated as it travels along the coax. The maximum cable attenuation per segment is 8.5 dB at 10 MHz and 6 dB at 5 MHz. This applies for both the 500 meters of Ethernet cable and the 185 meters of Cheapernet cable. With 10 Mbit/sec Manchester data, this cable attenuation results in approximately 7 ns of edge jitter in either direction. The CTI's receiver has to compensate for at least a portion of this jitter to meet the ±6 ns combined jitter budget. The receiver also should not overcompensate the signal in the case of a short cable. An equalizer filter in the CTI accomplishes this task. *Figure 5* shows a typical waveform seen at the far end of the cable and the corresponding differential output from the CTI's receiver.
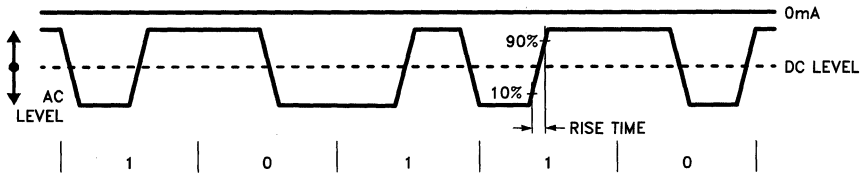
TL/F/8689–7

FIGURE 4. Coax Transmit Waveform

TL/F/8689–8
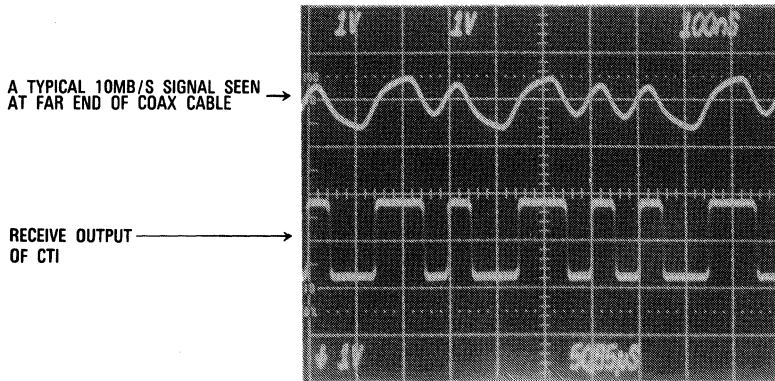
FIGURE 5. Oscilloscope Waveforms

TL/F/8689–9
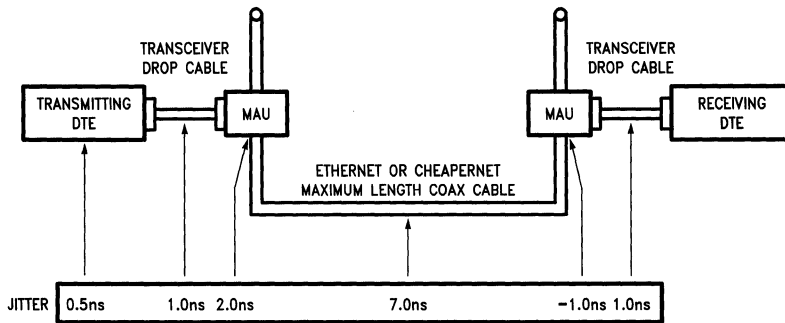
Total Jitter without Noise = 0.5 + 1.0 + 2.0 + 7.0 − 1.0 + 1.0 = 10.5 ns

Additional Jitter from Noise on Coax Cable = 5.0 ns

Additional Jitter from Noise on Drop Cables = 1.0 ns

Total System Jitter = 16.5 ns
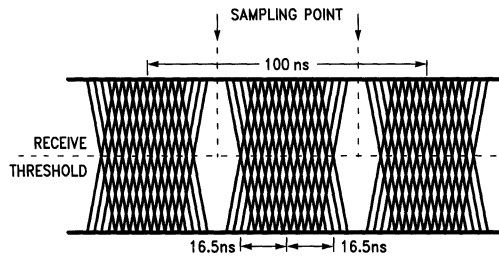
TL/F/8689–10

FIGURE 6. Typical Signal Waveform at SNI's Input

In addition to the equalizer, an AC/DC squelch circuit at the coax input prevents noise on the cable from falsely triggering the receiver in the absence of a valid signal. The Receive differential line from the CTI should be isolated before it reaches the SNI for Manchester decoding. This signal now could have accumulated as much as ±16.5 ns of jitter. *Figure 6* illustrates the jitter allocations for different network components and a typical signal waveform at the SNI's input. The digital phase-locked loop of the SNI can decode Manchester data with up to ±20 ns of random jitter which provides enough margin for implementation.

The SNI converts the Manchester received packet to NRZ data and clock pulses and sends them to the controller. Upon reception, the NIC checks the destination address, and if it is valid, verifies the CRC with the one generated on board and stores the packet in the local buffer memory. The packet is then moved to the host by the NIC, and when this is completed the buffer area is reclaimed for storing new packets. If a collision occurs during this transfer process, the CTI will detect it by sensing the average DC level on the coax and will send a 10 MHz collision signal to the SNI. The SNI will translate this information to the controller in TTL form, and the transmitting controllers will backoff for different times and retransmit later. Also in case of illegally long packets (longer than 20 ms), a jabber timer in the CTI will disable the coax driver so that the "jabbering" station will

not bring down the entire network. The collision pair is activated in this case to inform the controller of the faulty condition. After the fault is removed, the jabber timer holds for 500 ms before re-enabling the coax driver.

COLLISION DETECTION SCHEMES

There are two different collision detection schemes that can be implemented with the CTI; receive, transmit modes. The IEEE 802.3 standard allows the use of receive, transmit, and transhybrid modes for non-repeater nodes for both Ethernet and Cheapernet applications. Repeaters are required to have the receive mode implementation.

> **RECEIVE MODE:** Detects a collision between any two stations on the network with certainty at all times.

> **TRANSMIT MODE:** Detects collisions with certainty only when the station is transmitting.

RECEIVE MODE: The receive mode scheme has a very simple truth table; however, the tight threshold limits make the design of it difficult. The threshold in this case has to be between the maximum DC level of one station (−1300 mV) and the minimum DC level of two far end stations (−1581 mV). Several factors such as the termination resistor variation, coax center conductor resistance, driver current level variation, signal skew, and input bias current of non-transmitting nodes contribute to this tight margin. On

Truth Table for Various
Collision Detection Schemes

Mode	Receive				Transmit			
No. of Stations	0	1	2	>2	0	1	2	>2
Transmitting	N	N	Y	Y	N	N	Y	Y
Non-Transmitting	N	N	Y	Y	N	N	M	Y

Y = It will detect a collision,   N = It will not detect a collision,

M = It may detect a collision

top of the −1300 mV minimum level, the impulse response of the internal low pass filter has to be added. The CTI incorporates a 4 pole Bessel filter in combination with a trimmed on board bandgap reference to provide this mode of collision detection. However it would be difficult in receive mode to extend the cable length beyond the limits of the standard. It is also argued that it is not necessary for non-repeater nodes to detect collisions between other stations.

TRANSMIT MODE: In this case collisions have to be detected with certainty only when the station is transmitting. Thus, collisions caused by two other nodes may or may not be detected. This feature relaxes the upper limit of the threshold from −1581 mV to −1782 mV. As a result of this, longer cable segments can be used. With the CTI, a resistor divider can be used at the Collision Detect Sense pin (CDS) to lower the threshold from receive to transmit mode. Typical resistor values can be 120Ω from CDS to GND and 10k from CDS to V_{EE} (This moves the threshold by about −100 mV).

IMPLEMENTING A 10 BASE5 (ETHERNET) MAU WITH THE DP8392

The CTI provides all the MAU (transceiver) functions except for signal and power isolation. Signal isolation can easily be provided by a set of three pulse transformers that come in a single Dual-in-Line package. These are available from transformer vendors such as Pulse Engineering (PE64103) and Valor (LT1101). However, for the power isolation a DC to DC converter is required. The CTI requires a single −9 (±5%) volt supply. This power has to be derived from the power pair of the drop cable which is capable of providing 500 mA in the 12 (−6%) to 15 (+5%) volt range. The low supply current of the CTI makes the design of the DC to DC converter quite easy. Such converters are being developed in hybrid packages by transformer manufacturers (Pulse Engineering PE64430 and Reliability Inc. 2E12R9). They provide the necessary voltage isolation and the output regulation. One can also build a simple DC to DC converter with a two transistor self oscillating primary circuit and some regulation on the secondary as shown in *Figure 7*.

Several areas of the PC board layout require special care. The most critical of these is for the coax connection. Ethernet requires that the CTI capacitance be less than 2 pF on the coax with another 2 pF allocated for the tap mechanism. The Receive Input (RXI) and the Transmit Output (TXO) lines should be kept to an absolute minimum by mounting the CTI very close to the center pin of the tap. Also, for the external diode at TXO (see *Figure 8*), the designer must minimize any stray capacitance, particularly on the anode side of the diode. To do this, all metal lines, especially the ground and V_{EE} planes, should be kept as far as possible from the RXI and TXO lines.

In order to meet the stringent capacitive loading requirements on the coax, it is imperative that the CTI be directly soldered to the PC board without a socket. A special lead frame in the CTI package allows direct conduction of heat from the die through these leads to the PC board, thus reducing the operating die temperature significantly. For good heat conduction the V_{EE} pins (4, 5 and 13) should be connected to large metal traces or planes.

A separate voltage sense pin (CDS) is provided for accurate detection of collision levels on the coax. In receive mode, where the threshold margin is tight, this pin should be independently attached to the coax shield to minimize errors due to ground drops. A resistor divider network at this pin can be used for transmit mode operation as described earlier.

The differential transmit pair from the DTE should be terminated with a 78Ω differential resistive load. By splitting the termination resistor into two equal values and capacitively AC grounding the center node, the common mode impedance is reduced to about 20Ω, which helps to attenuate common mode transients.

To drive the 78Ω differential line with sufficient voltage swings, the CTI's collision and receive drivers need external 500Ω resistors to V_{EE}. By using external resistors, the power dissipation of the chip is reduced, enhancing long term reliability. The only precision component required for the CTI is one 1k 1% resistor. This resistor sets many important parameters of the chip such as the coax driving levels, output rise and fall times, 10 MHz collision oscillator frequency, jabber timing, and receiver AC squelch timing. It should be connected between pins 11 (RR+) and 12 (RR−).

The DP8392 features a heartbeat function which can be externally disabled using pin 9. This function activates the collision output for a short time (10 ±5 bit cells) at the end of every transmission. It is used to ensure the controller that the collision circuitry is intact and properly functioning. Pin 9 enables CD Heartbeat when grounded, and disables it when connected to V_{EE}.

The IEEE 802.3 standard requires a static discharge path to be provided between the shield of the coax cable and the DTE ground via a 1 MΩ, 0.25W resistor. The standard also requires the MAU to have low susceptibility levels to electro-magnetic interference. A 0.01 μF capacitor will provide a sufficient AC discharge path from the coaxial cable shield to the DTE ground. The individual shields should also be ca-pacitively coupled to the Voltage Common in MAU. A typical Ethernet MAU connection diagram using the CTI in receive mode with the CD Heartbeat enabled is shown in Figure 8.

TL/F/8689–11
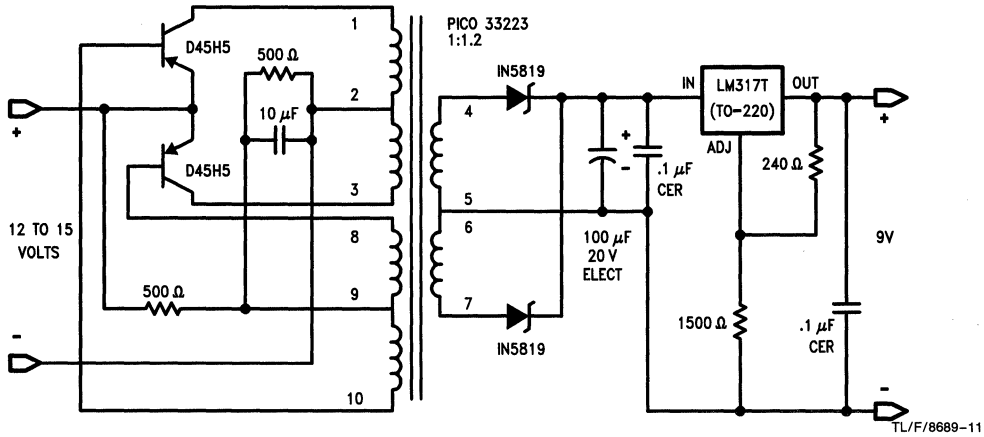
FIGURE 7. A Simple Low Cost DC to DC Converter

TL/F/8689–12
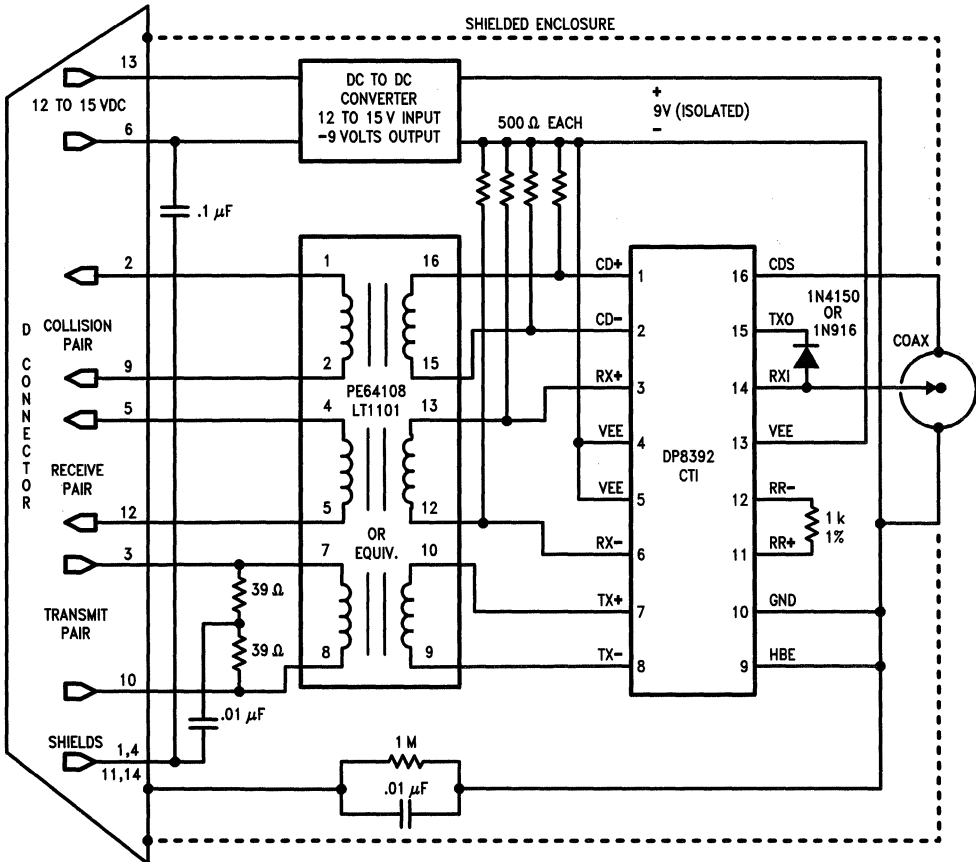
FIGURE 8. An Ethernet MAU Implementation with the CTI

AN-442

CHEAPERNET APPLICATION WITH THE DP8391 AND DP8392

The pin assignment of both the CTI and the SNI are designed to minimize the crossover of any printed circuit traces. Some of the components needed for an Ethernet like interface are not needed for Cheapernet. For instance, Cheapernet's relaxed load capacitance (8 pF, compared with 4 pF for Ethernet) obviates the need for an external capacitance isolation diode at TXO. Also, since the transceiver drop cable is not used in Cheapernet, there's no need for the 78Ω termination resistors. Moreover, without the 78Ω loading on the differential outputs, the pulldown resistors for both the CTI's collision and receive drivers and the SNI's transmit driver can be larger to save power. These resistors can be 1.5k instead of 500Ω for the CTI and 500Ω instead of 270Ω for the SNI.

The 20 MHz crystal connection to the SNI requires special care. The IEEE 802.3 standard requires a 0.01% absolute accuracy on the transmitted signal frequency. An external capacitor between the X1 and X2 pins is normally needed to get the required frequency range. Section 3.1 of the data sheet describes how to choose the value of this capacitor.

The SNI also provides loopback capability for fault diagnosis. In this mode, the Manchester encoded data is internally diverted to the decoder input and sent back to the controller. Thus both the encoding and the decoding circuits are tested. The transmit differential output driver and the differential input receiver circuits are disabled during loopback. This mode can be enabled by a TTL active high input at pin 7.

Two different modes, half step and full step, can be selected at the SNI's transmit output. The standards require half step mode of operation, where the output goes to differential zero during idle to eliminate large idle currents through the pulse transformers. On the other hand, the differential output remains in a fixed state during idle in full step mode. The SNI thus can be used with transceivers which work in either mode. The two different modes can be selected with a TTL input at pin 5.

Figure 9 shows a typical Cheapernet connection diagram using the DP8391 and the DP8392.
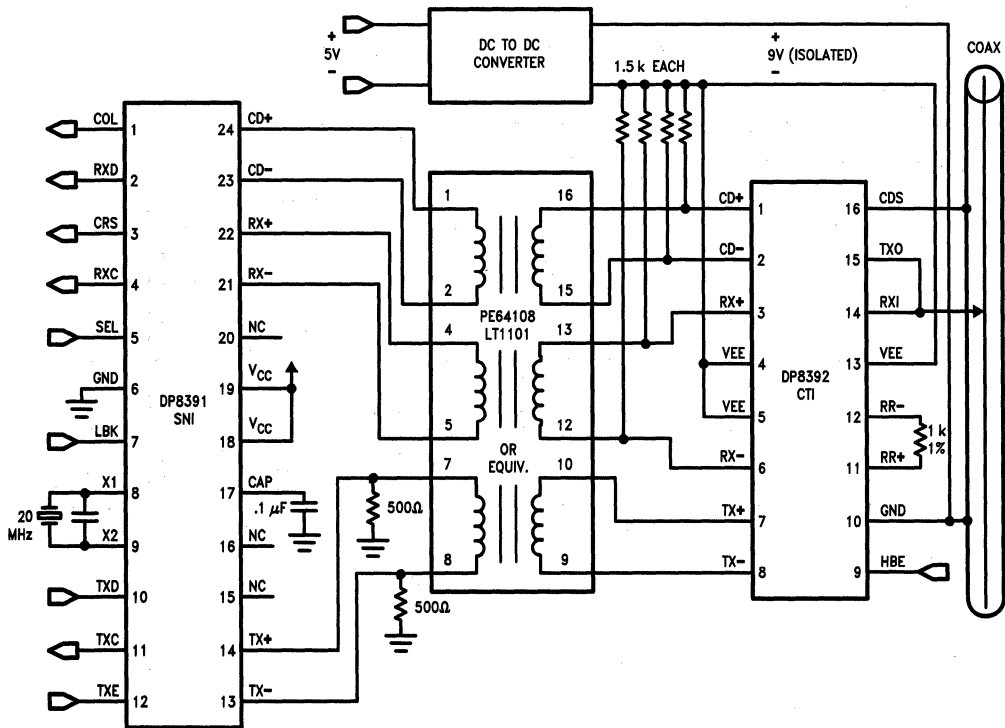
FIGURE 9. Cheapernet Connection Diagram

TL/F/8689–13

The power isolation is similar here as in the Ethernet application, except the DC input is now usually 5V instead of 12V. Hybrid DC to DC converters are also being developed for this application (Ex: Pulse Engineering PE64381). *Figure 10* shows a discrete implementation with 5V input and −9V output.

1-118
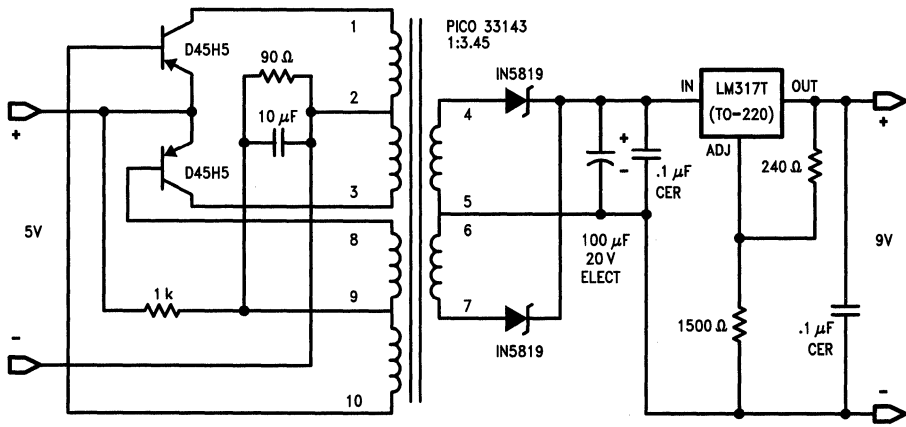
FIGURE 10. DC to DC Converter (5V to −9V)

TL/F/8689–14

DP8390 Network Interface Controller: An Introductory Guide

OVERVIEW

A general description of the DP8390 Network Interface Controller (NIC) is given in this application note. The emphasis is placed on how it operates, and how it can be used. This description should be read in conjunction with the DP8390 data sheet.

1.0 INTRODUCTION

The DP8390 Network Interface Controller provides all the Media Access Control layer functions required for transmission and reception of packets in accordance with the IEEE 802.3 CSMA/CD standard. The controller was designed to act as an advanced peripheral and serve as a complete interface between the system and the network. The on-board FIFO and DMA channels work together to form a straight-forward packet management scheme, providing (local) DMA transfers at up to 10 megabytes per second while tolerating typical bus latencies.

A second set of DMA channels (remote DMA) is provided on chip, and is integrated into the packet management scheme to aid in the system interface. The DP8390 was designed with the popular 8, 16 and 32 bit microprocessors in mind, and gives system designers several architectural options. The NIC is fabricated using National Semiconductor's double metal 2 micron microCMOS process, yielding high speed with very low power dissipation.

2.0 METHOD OF OPERATION

The NIC is used as a standard peripheral device and is controlled through an array of on-chip registers. These registers are used during initialization, packet transmission and reception, and remote DMA operations. At initialization, the physical address and multicast filters are set, the receiver, transmitter and data paths are configured, the DMA channels are prepared, and the appropriate interrupts are masked. The Command Register (CR) is used to initiate transmission and remote DMA operations.

Upon packet reception, end of packet transmission, remote DMA completion or error conditions, an interrupt is generated to indicate that an action should be taken. The processor's interrupt driven routine then reads the Interrupt Status Register (ISR) to determine the type of interrupt that occurred, and performs the appropriate actions.

3.0 PACKET TRANSMISSION

The NIC transmits packets in accordance with the CSMA/CD protocol, scheduling retransmission of packets up to 15 times on collisions according to the truncated binary exponential backoff algorithm. No additional processor intervention is required once the transmit command is given.
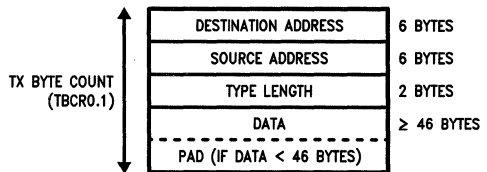
FIGURE 1. Transmit Packet Format

3.1 Transmission Setup

After a packet that conforms to the IEEE 820.3 specification is set up in memory, with 6 bytes of the destination address, followed by 6 bytes of the source address, followed by the data byte count and the data, it is ready for transmission (see *Figure 1*). To transmit a packet, the NIC is given the starting address of the packet (TPSR), the length of the packet (TPCR0, TBCR1), and then the PTX (transmit packet) bit of the Command Register is set to initiate the transmission (see *Figure 2*).
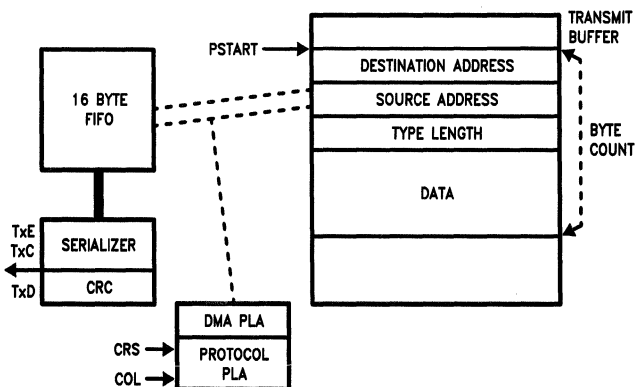
FIGURE 2. Packet Transmission

3.2 Transmission Process

Once the transmit command is given, if no reception is in progress, the transmit prefetch begins. The high speed local DMA channel bursts data into the NIC's FIFO. After the first DMA transfer of the prefetch burst, if no carrier is present on the network, and the NIC is not deferring, the TXE (transmit enable) signal is asserted and the transmission begins. After the 62 bits of preamble (alternating ONEs and ZEROs) and the start of frame delimiter (two ONEs) are sent out, the data in the FIFO is serialized, and sent out as NRZ data (pin TxD) with a clock (TxC), while the CRC is calculated. When the FIFO reaches a threshold (X bytes empty) a new DMA burst is initiated. This process continues until the byte count reaches zero. After the last byte is serialized, the four bytes of the calculated CRC are serialized and appended to complete the packet.

Should a collision occur, the current transmission stops, a jam sequence (32 Ones) transmitted (to ensure that every node senses a collision), and a retransmission of the packet is scheduled according to the truncated Binary Exponential Backoff Routine.

3.3 Transmission Status

After the transmission is complete, an interrupt is generated and either the PTX bit (complete packet transmitted) or the TXE bit (packet transmission aborted) of the ISR (Interrupt Status Register) is set. The interrupt driven routine then reads the TSR to find out details of the transmission. If the PTX bit is set, the TSR can reveal if a carrier was present when the transmission was initiated (DFR), if the carrier was lost during the transmission (CRS—this would point to a short somewhere on the network), if the collision detect circuitry is working properly (CDH), and if collision occurred (COL). Whenever a collision is encountered during transmission, the collision count register (NCR) is incremented. Should a collision occur outside the 512 bit window (slot time), the OWC (Out of Window Collision) bit of the TSR is set.

The TXE bit of the ISR is set if 16 collisions or a FIFO underrun occurs. If the transmission is aborted due to 16 collisions, the ABT bit of the TSR is set. (If this occurs it is likely that there is an open somewhere on the network.) If the local DMA channel can not fill the FIFO faster than data is sent to the network, the FU bit (FIFO Underrun) of the TSR is set and the transmission is also aborted. This is a result of a system bandwidth problem and points to a system design flaw. System bandwidth considerations are discussed further in Section 5.1.3.

4.0 PACKET RECEPTION

The bus topology used in CSMA/CD networks allows every node to receive every packet transmitted on the network. The receive filters determine which packets will be buffered to memory. Since every packet is not of interest, only packets having a destination address that passes the node's receive filters will be transferred into memory. The NIC offers many options for the receive filters and implements a complete packet management scheme for storage of incoming packets.

4.1 Reception Process

When a carrier is first sensed on the network (i.e. CRS signal is active), the controller sees the alternating ONE - ZERO preamble and begins checking for two consecutive ONEs, denoting the start of frame delimiter (SFD). Once the SFD is detected, the serial stream of data is deserialized and pushed into the FIFO, a byte at a time. As the data is being transferred into the FIFO, the first six bytes are checked against the receive address filters. If an address match occurs, the packet is DMAed from the FIFO into the receive buffer ring. If the address does not match, the packet is not buffered and the FIFO is reset.

Each time the FIFO threshold is reached, a DMA burst begins and continues for the proper number of transfers. DMA bursts continue until the end of the packet (Section 5.1.2). At the end of a reception, the NIC prepares for an immediate reception while writing the status of the previous reception to memory. An interrupt is issued to indicate that a packet was received, and is ready to be processed.

The CRC generator is free running and is reset whenever the SFD is detected. At every byte boundary the calculated value of the CRC is compared with the last four received bytes. When the CRS signal goes LOW, denoting the end of a packet, if the calculated CRC matches the received CRC on the last byte boundary, the packet is a good packet and is accepted. However, if the calculated and received CRCs do not match on the last byte boundary before CRS goes LOW, a CRC error is flagged (CRC bit of RSR set) and the packet is rejected, i.e. the receive buffer ring pointer (CURR) is not updated (Section 4.5). If the CRS signal does not go LOW on a byte boundary and a CRC error occurs, the incoming packet is misaligned, and a frame alignment error is flagged (FAE bit of RSR set). Frame alignment errors only occur with CRC errors.

4.2 Address Matches

The first bit received after the SFD indicates whether the incoming packet has a physical or multicast address. A ZERO indicates a physical address, that is, a unique map-
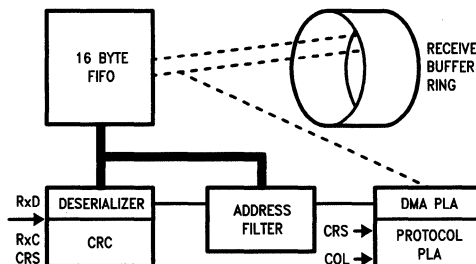
FIGURE 3. Packet Reception

TL/F/9141–3

ping between the received address and the node's 48 bit physical address as programmed at intialization (PAR0–PAR5). A ONE indicates a multicast address, meaning a packet intended for more than one node.

Multicast addressing is useful where one node needs to send a packet to multiple nodes, as in a query command. Multicast addressing provides a very fast way to perform address filtering in realtime, by using an on-chip hashing table. A hashing algorithm based on the CRC is used to map the multicast address into the 64 bit Multicast Address Filter (MAF0–7).

After the CRC has been calculated on the destination address, the upper six bits of the CRC are used as an index into the Multicast Address Filter (MAF). If the selected filter bit is ONE, the packet is accepted, if the MAF bit is ZERO the packet is not accepted.

A special multicast address is the broadcast address, which denotes a packet intended to be received by all nodes. The broadcast packet has an address of all ONEs (this address also maps into a bit in the MAF).

The DP8390 also provides the ability to accept all packets on the network with a physical address. Promiscuous mode causes any packet with a physical address to be buffered into memory. To receive all multicast packets it is nesessary to set all of the MAF bits to ONE.

4.3 Network Statistics

Three eight bit counters are provided for monitoring receive packet errors. After an address match occurs if a Frame Alignment or CRC error occurs, or if a packet is lost due to insufficient buffer resources (see below), the appropriate counter is incremented. These counters are cleared when read. The counters trigger an interrupt when they reach a value of 128 (if not masked) to force the processor to read (and thus clear) their contents. The counters have a maximum value of 192, providing a large latency between when the interrupt is asserted and when the counter overflows. When a CNT interrupt occurs, all three tally counters should be read and added into larger counters maintained by the processor.

4.4 Setting the Receive Configuration Register

The Receive Configuration Register (RCR) is used in conjunction with the physical and multicast addresses to determine which packets should be accepted and placed in the receive buffer ring. The RCR is initialized to accept physical, multicast and/or broadcast packets, or alternatively to place the receiver in promiscuous mode to accept all packets with a physical address. If the MON bit of the RCR is set, placing the receiver in monitor mode, the receiver still checks the addresses of incoming packets according to the set up address filter, and network statistics are still gathered, but packets are not buffered into memory.

The minimum packet size in standard 802.3 networks is 64 bytes long. Packets less than 64 bytes are considered runt packets and are normally rejected. However, in some applications it may be desirable to accept such packets. By setting the AR bit of the RCR, runt packets are accepted.

For diagnostic purposes it may be desirable to examine errored packets, and not overwrite them with good packets as is done in normal operation. By setting the SEP bit of the RCR, errored packets are saved and their status is written to memory.

4.5 Receive Buffer Ring

As packets are received they are placed into the receive buffer ring, and as they are processed they are removed from this ring. At initialization, an area of memory is allocated to act as the receive buffer ring, and the NIC's buffer management scheme then makes efficient use of this memory. The efficiency is helped significantly because the ring pointers are contained on chip, and the DMA channels can work at up to a 10 Mbyte/sec transfer rate. A second DMA channel, the remote DMA channel, is available for transferring packets out of the receive buffer ring.

The employed buffer management scheme effectively works as a large packet FIFO. This buffer management scheme is very appropriate for most networking applications because packets are generally processed in the order they are received.

Four pointers are used to control the ring; the (1) page start (PSTART) and (2) page stop (PSTOP) pointers to determine the size of the buffer ring, the (3) current page (CURR) pointer, to determine where the next packet will be loaded,
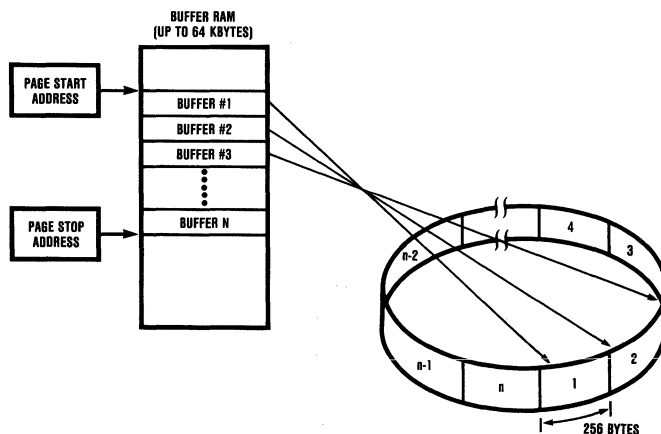
FIGURE 4. The Receive Buffer

TL/F/9141–4

TL/F/9141–5

TL/F/9141–6
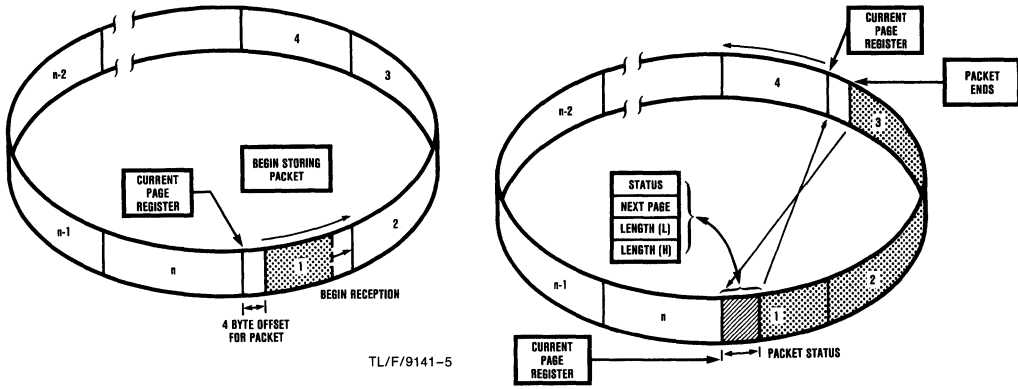
FIGURE 5. Receive Packet Buffering

and the (4) boundary (BNRY) pointer, to show where the next packet to be unloaded (or processed) lies. As packets are received, the boundary pointer follows the current page pointer around the ring. The page start and stop pointers remain unchanged during operation.

The receive buffer ring is divided into 256 byte buffers, and these buffers are linked together as required by the received packets (see *Figure 4*). Up to 256 of these buffers can be linked together in the receive buffer ring, yielding a maximum buffer size of 64K bytes. Since all NIC registers are 8 bits wide, the ring pointers refer to 256 byte boundaries within a 64K byte space.

At initialization, PSTART register is loaded with the beginning page address of the ring, and PSTOP is loaded with the ending page address of the ring.

On a valid reception, the packet is placed in the ring at the page pointed to by CURR plus a 4 byte offset (see *Figure 5*). The packet is transferred to the ring, a DMA burst at a time. When necessary, buffers are automatically linked together, until the complete packet is received. The last and first buffers of the ring buffer are linked just as the first and seconds buffers. At the end of a reception, the status from

the Receive Status Register (RSR), a pointer to the next packet, and the byte count of the current packet are written into the 4 byte offset.

If a receive error occurs (FAE, CRC) CURR is not updated at the end of a reception, so the next packet received overwrites the bad packet (see *Figure 6*). This feature can be disabled (by setting the save errored packet (SEP) bit in the RCR) to allow examination of errored packets.

At receiving nodes, collision fragments may be seen as runt packets. A runt packet is a packet less than 64 bytes (512 bits) long, and since a collision must occur in the first 512 bit times, the packet will be truncated to less than 64 bytes. After runt packets are received, the CURR is not updated, so the next packet received will overwrite the runt packet. This standard feature can also be suppressed by setting the AR bit in the TCR. This is useful when it is desirable to examine collision fragments, and in non-standard applications where smaller packets are desirable.

Once packets are in the receive ring they must be processed. However, the amount of processing that occurs while the packet is in the buffer ring varies according to the implementation. As packets are removed from the buffer ring, the boundary pointer (BNRY) must be updated. The BNRY always follows CURR around the ring (see *Figure 7*).

TL/F/9141–7
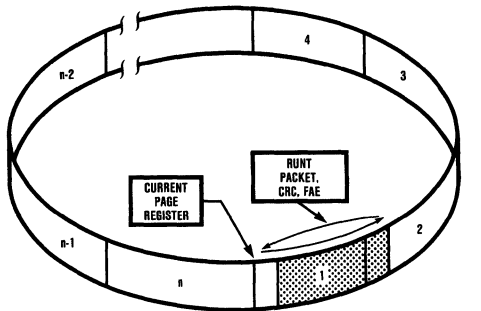
FIGURE 6. Packet Rejection

TL/F/9141–8
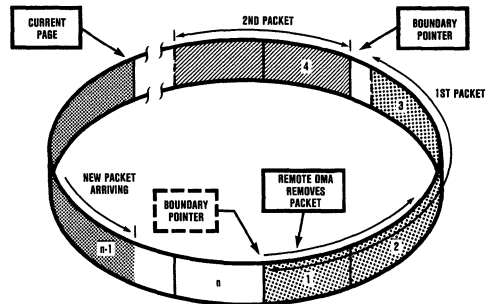
FIGURE 7. Removing Packets From Receive Buffer Ring

If the current local DMA address ever reaches BNRY, the ring is full. In this case, the current and any additional receptions are aborted and tallied until the BNRY pointer is updated. Packets already present in the ring will not be overwritten (see *Figure 8*). All missed packets will increment the missed packet tally counter. When enough memory is allocated for the receive buffer ring, the overwrite warning (setting of the OVW bit of the ISR) should seldom occur.
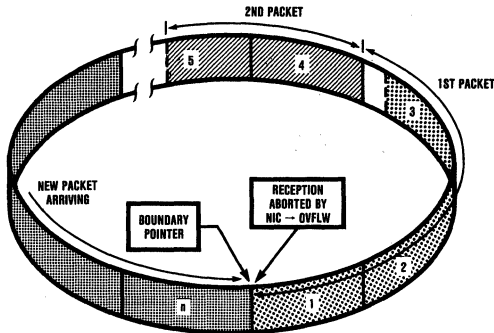
TL/F/9141–9

FIGURE 8. Receive Buffer Ring Overwrite Protection

A second set of DMA channels has been included on the DP8390 to aid in the transfer of packets out of the buffer ring. These Remote DMA channels can work in close co-operation with the receive buffer ring to provide a very effective system interface (§7).

If the BNRY is placed outside of the buffer ring, no overwrite protection will be present, and incoming packets may overwrite packets that have not been processed. This may be useful when evaluating the DP8390, but in normal operation it is not recommended.

When the CURR and BNRY pointers are equal, the buffer ring can either be completely empty or completely full. To ensure that the NIC does not misinterpret this condition, it is necessary to guarantee that the value of the BNRY pointer does not equal the value of the CURR pointer. It is recommended that the BNRY pointer be kept one less than CURR pointer when the ring is empty, and only be equal to CURR when the ring is full, as shown below.

1. Use a variable (NXTPKT) to indicate from where the next packet will be removed (possibly using Remote DMA)

2. At initialization set:
   BNRY = PSTART
   CURR = PSTART + 1
   NXTPKT = PSTART + 1

3. After each packet is removed from the ring, use the next packet pointer in the header information (the second byte of the header), HNXTPKT, and set:
   NXTPKT = HNXTPKT
   BNRY = HNXTPKT − 1
   If BNRY < PSTART then BNRY = PSTOP − 1

The above procedure is not necessary if the Send Packet Command is used to remove packets from the ring as explained in section 7.

5.0 SYSTEM/NETWORK INTERFACE

The DP8390 offers considerable flexibility when designing a system/network interface. This flexibility allows the designer to choose the appropriate price/performance combination while easing the actual design process.

5.1 Interfacing Considerations

Several features have been included on the NIC to allow it to easily be integrated into many systems. The size of the data paths, the byte ordering, and the bus latencies are all programmable. In addition, the clock the DMA channels use is not coupled to the network clock, so the NIC's DMA can easily be integrated into memory systems.

5.1.1 Data Path

The NIC can interface with 8, 16, and 32 bit microprocessors. The data paths are configurable for both byte- wide and word-wide transfers (bit WTS in DCR). When in word-wide mode, the byte ordering is programmable to accommodate both popular byte ordering schemes. All NIC registers are 8 bits wide to allow 8, 16 and 32 bit processors to access them with no additional hardware. If the NIC's 16 address lines (64K bytes) do not provide an adequate address space, the two DMA channels can be concatenated to form a 32 bit DMA address (bit LAS in DCR).

5.1.2 Local DMA

The DMA transfers between the FIFO and memory during transmission and reception occur in bursts. The bursts begin when the FIFO threshold is reached. Since only a single FIFO is required (because a node cannot receive and transmit simultaneously), the threshold takes on different meanings during transmission and reception. During reception the FIFO threshold refers to the number of bytes in the FIFO. During transmission the FIFO threshold refers to the number of empty bytes in the FIFO (16 - # bytes in FIFO). The FIFO threshold is set to 2, 4, 8 or 12 bytes (1, 2, 4 or 6 words) in the DCR (bits FT0, FT1).

The number of transfers that occur in a burst depends on whether the Exact Transfer or Empty/Fill mode is used (bit BMS in DCR). When in Exact Transfer mode, a number of bytes/words equal to the FIFO threshold will be transferred in each burst. The Empty/Fill mode continues the transfers until the FIFO is empty, during receptions, and full, during transmissions (see *Figure 8*).
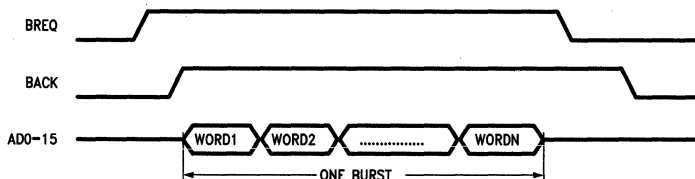
TL/F/9141–10

where N = 1, 2, 4 or 6 Words or N = 2, 4, 8, or 12 Bytes when in byte mode

FIGURE 8. Local DMA Burst

Before a burst can begin, the NIC must first arbitrate to become master of the bus. It requests the bus by activating the BREQ signal and waiting for acknowledgment with the BACK signal. Once the NIC becomes the master of the bus, the byte/word transfers may begin. The frequency of the DMA clock is not related to the network clock, and can be input (pin 25) as any frequency up to 20 MHz. For 10 Mbit/sec networks the DMA clock can be as slow as 6 MHz. This allows tailoring of the DMA channel, to the system. The local DMA channel can burst data into and out of the FIFO at up to 10 Mbyte/sec (8X the speed of standard Ethernet). This means that during transmission or reception the network interface could require as little as one eighth of the bus bandwidth.

5.1.3 Bus Analysis

Two parameters useful in analysis of bus systems are the Bus Latency and the Bus Utilization. The Bus Latency is the maximum time between the NIC assertion of BREQ and the system granting of BACK. This is of importance because of the finite size of the NIC's internal FIFO. If the bus latency becomes too great, the FIFO overflows during reception (FIFO overrun error), and becomes empty during transmission (FIFO underrun error). Both conditions result in an error that aborts the reception or transmission. In a well designed system these errors should never occur. The Bus Utilization is the fraction of time the NIC is the master of the bus. It is desirable to minimize the time the NIC occupies the bus, in order to maximize its use by the rest of the system. When designing a system it is necessary to guarantee the NIC a certain Bus Latency, and it is desirable to minimize the Bus Utilization required by the NIC.

Associated with each DMA burst is a DMA set up and recovery time. When a packet is being transferred either to or from memory it will be transferred in a series of bursts. If more byte/word transfers are accomplished in each burst, fewer bursts are required to transfer the complete packet, and less time is spent on DMA set up and recovery. Thus, when longer bursts are used, less bus bandwidth is required to complete the same packet transfer.

The Empty(/Fill) mode guarantees longer bursts because as the byte/word transfers are taking place, serialized data is still filling(/emptying) the FIFO, and these additional bytes/words must also be transferred out of(/into) the FIFO. The least NIC bus utilization occurs when the bursts are as long as possible. This occurs when the threshold is as high as possible, and Empty/Fill mode is used. The determination of the threshold is related to the maximum bus latency the system can guarantee the NIC.

If the NIC is required to guarantee other devices a certain bus latency, it can only remain master of the bus for a certain amount of time. In this case, the Exact Transfer burst mode is desirable because the NIC only remains master of the bus for a certain amount of time.

6.0 INTERFACE OPTIONS

The network interface can be incorporated into systems in several ways. The network interface can be controlled by either a system processor or a dedicated processor, and can utilize either system memory or buffer memory. This section covers the basic interface architectures.

6.1 Single Bus System

The least complex implementation places the NIC on the same bus as the processor (see *Figure 10*). The DP8390 acts as both a master and a slave on this bus; a master during DMA bursts, and a slave during NIC register accesses. This architecture is commonly seen on motherboards in personal computers and low cost workstations, but until recently without an integrated network interface. A major issue in such designs is the bus bandwidth for use by the processor. The DP8390 is particularly suitable for such applications because of its bus utilization characteristics. During transmissions and receptions, the only time the NIC becomes a bus master, the DP8390 can require as little as one- eighth the bus bandwidth. In addition, the previously mentioned bus tailoring features, ease its integration into such systems.
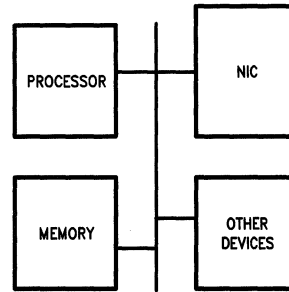
TL/F/9141–11

FIGURE 10. Single Bus Configuration

The design must only be able to guarantee the NIC a maximum bus latency ($< 9 \mu s$ for 10 Mbit/s networks), because of the finite size of the on-chip FIFO. In bus systems where the NIC is the highest priority device, this should present no problem. However, if the bus contains other devices such as Disk, DMA and Graphic controllers that require the bus for more than 10 μs during high priority or real time activities, meeting this maximum bus latency criteria could present a problem.

Likewise, many existing single bus systems make no provision for external devices to become bus masters, and if they do, it is only under several restrictions. In such cases, an interface without the mentioned bus latency restrictions is highly desirable.

6.2 Dual Port Memory

One popular method of increasing the apparent bus latency of an interface, has the added effect of shielding the system bus from the high priority network bandwidth. In this application, the Dual Port Memory (DPM) allows the system bus to access the memory through one port, while the network interface accesses it through the other port. In this way, all of the high priority network bandwidth is localized on a dedicated bus, with little effect on the system bus (see *Figure 11*).
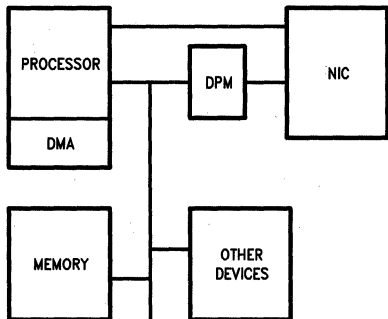
TL/F/9141–12

FIGURE 11. DPM Configuration

Dual Port Memories are typically smaller than the main memory and little, if any, processing can occur while the packets are in the DPM. Therefore, the processor (or if available, DMA controller) must transfer data between the DPM and the main memory before beginning packet processing. In this example, the DPM acts as a large packet FIFO.

Such configurations provide workable solutions, however, Dual Port Memories are inherently expensive. Aside from the extra complexity of the software, DPM contention logic is expensive, and dedicated DPM chips provide only 1k of memory and cost as much as advanced VLSI devices. In addition, some systems do not contain additional memory for such memory mapped interfaces.

6.3 Dual Port Memory Equivalent

The functional equivalent of a Dual Port Memory implementation can be realized for low cost with the DP8390. This configuration makes use of the NIC's Remote DMA capabilities and requires only a buffer memory, and a bidirectional I/O port (see *Figure 12*). The complete network interface, with 8k x 8 of buffer memory, easily fits onto a half size IBM-PC card (as in the Network Interface Adapter, NIA, for the IBM-PC.)
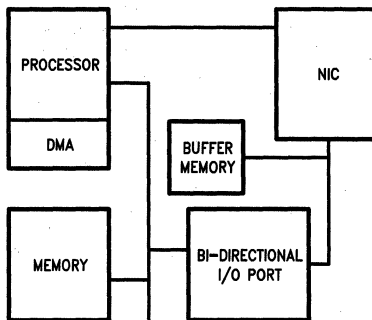
TL/F/9141–13

FIGURE 12. DPM Equivalent Configuration

The high priority network bandwidth is decoupled from the system bus, and the system interracts with the buffer memory using a lower priority bi-directional I/O port. For example, when a packet is received the local DMA channel transfers it into the buffer memory, part of which has been configured as the receive buffer ring. The remote DMA channel then transfers the packet on a byte by byte (or word by word) basis to the I/O port. At this point, as in the previous example, the processor (or if available, DMA channel), through a completely asynchronous protocol, transfers the packet into the main memory.

6.4 Dual Processor Configuration

For higher performance applications, it is desirable to off-load the lower-level packet processing functions from the main system (see *Figure 11*). A processor placed on a local bus with the NIC, memory and a bi-directional I/O port could accomplish these lower-level tasks, and communicate with the system processor through a higher level protocol. This processor could be responsible for sending acknowledgement packets, establishing and breaking logical links, assembling and disassembling files, executing remote procedure calls, etc.
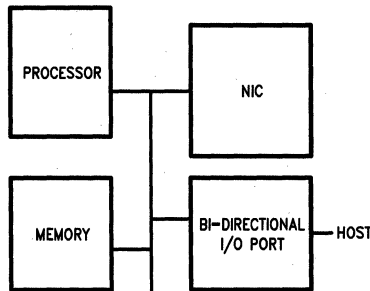
TL/F/9141–14

FIGURE 13. Dual Processor Configuration

7.0 REMOTE DMA

A second set of DMA channels is built into the DP8390 to aid in the system integration (as discussed above). Using a simple asynchronous protocol, the Remote DMA channels are used to transfer data between dedicated network memory, and common system memory. In normal operation, the remote DMA channels transfer data between the network memory and an I/O port, and the system transfers between the I/O port and the system memory. The system transfers are typically accomplished using either the processor, or a DMA controller.

The Remote DMA channels work in both directions; pending transmission packets are transferred into the network memory, and received packets are transferred out of the network memory. Transfers into the network memory are known as remote write operations, and transfers out of the network memory are known as remote read operations. A special remote read operation, send packet, automatically removes the next packet from the receive buffer ring.

7.1 Performing Remote DMA Operations

Before beginning a remote DMA operation, the controller must be informed of the network memory it will be using.

Both the starting address (RSAR0,1) and length (RBCR0,1) are set before initiating the remote DMA operation. The remote DMA operation begins by setting the appropriate bits in the Command Register (RD0–RD3). When the remote DMA operation is complete (all of the bytes transferred), the RDC bit (Remote DMA Complete) in the ISR (Interrupt Status Register) is set and the processor receives an interrupt, whereupon it takes the appropriate action. When the Send packet command is used, the controller automatically loads the starting address, and byte count from the receive buffer ring for the remote read operation, and upon completion updates the boundary pointer (BNRY) for the receive buffer ring. Only one remote DMA operation can be active at a time.

7.2 Hardware Considerations

The Remote DMA capabilities of the NIC were designed to require minimal external components and provide a simple implementation. An eight bit bi-directional port can be implemented using just two 374 latches (see the DP8390 Hardwre Design Guide). All of the control circuitry is provided on the DP8390. In addition, bus arbitration with the local DMA is accomplished within the NIC in such a way as to not lock out other devices on the bus (see the DP8390 Datasheet).

StarLAN With The DP839EB Evaluation Board

OVERVIEW

Because of the identical packet structures between Star-LAN (IEEE 802.3 1base5) and Ethernet (10base5), the DP8390 Network Interface Controller (NIC) will operate in all versions of IEEE 802.3 based networks. To evaluate the DP8390 in StarLAN applications, the DP839EB Evaluation Board can be used with a "daughter card" that replaces the Ethernet/Cheapernet front end with a StarLAN front end. The StarLAN front end consists of an RS-422/485 type transceiver and a 1 Mbit/s Manchester encoder/decoder (ENDEC), as shown below. The 82C550A, manufactured by Chips and Technology, and the MK5035N, manufactured by Mostek corporation, can provide the required ENDEC functions for the NIC.

CABLING

Since a significant number of StarLAN networks are expected to use existing twisted pair telephone wiring, DTEs will be connected to wall outlets, which in turn, will be connected to wiring closets where the StarLAN hubs will be located. The cabling used typically will consist of 26–22 gauge, unshielded twisted pairs with maximum cable length approximately 250 meters (800 ft) from Hub to DTE. If 5 levels of hub are used, the network may extend up to 2.5 Km.

TRANSCEIVER

The transceiver connects to two twisted pair phone wires, one for transmit, the other for receive and is isolated by two pulse transformers. Some pulse transformers also provide rise time limiting to reduce EMI. The transceiver circuitry is based on the DS8923 dual receiver/driver combination. Two of the receivers are used to provide receive and squelch functions.

RECEIVER/SQUELCH

Since the cabling may be bundled together and routed close to heavy electrical equipment, squelch circuitry is necessary to reject signals generated from crosstalk between adjacent wires and impulse noise from large equipment. Proper noise immunity may be implemented using a second-order Butterworth filter with a 2 MHz cutoff and setting a 600 mV squelch level. Because RS-422 receivers typically have 200 mV threshold levels, these inputs must be skewed to 600 mV. This may be implemented by using a resistor ladder which holds the inputs 600 mV apart (see Squelch Adjustment). When an incoming signal exceeds the 600 mV threshold, the receiver is enabled.

As shown in the Squelch Level Adjustment figure, two receivers are used for the receive/squelch function. One receiver sets the 600 mV input threshold and is used by the ENDEC to drive its internal squelch circuitry; the other receiver presents the actual unskewed data to be decoded.

TRANSMITTER

The transmitter is comprised of one RS-422 driver provided in the DS8923 dual line driver-receiver package. The driver is enabled using the external transceiver output of the Manchester ENDEC, which is asserted coincident with the first bit of valid data and is de-asserted two bit times following the last bit. This allows generation of the 2-bit idle signal, marking the end of the packet.

82C550A INTERFACE TO THE NIC

The 82C550A interfaces to the DP8390 via 5 inverters to provide the proper polarity of CRS, COL, TXE, RXC, and LBK. The normal mode (MODE = 1) is selected to allow an external transceiver to be used. The squelch level input, /RxDI must be connected to pin 1 of the DS8923 to attain

StarLAN Front End for the DP8390
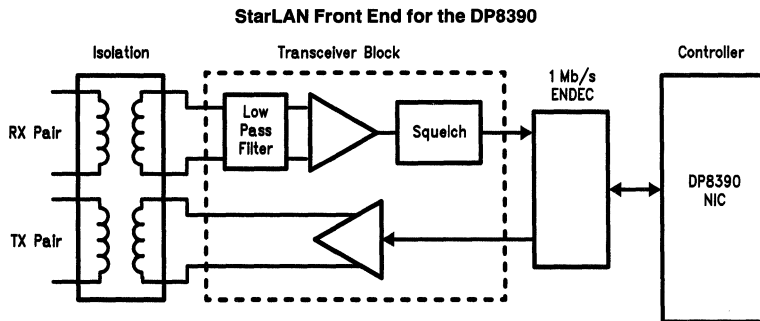
TL/F/9346–1

Squelch Level Adjustment
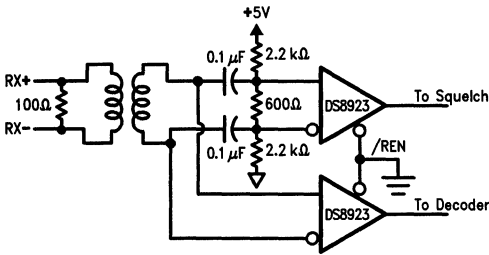
TL/F/9346-2

the proper input threshold (600 mV). The RxDI input contains the actual data to be decoded to NRZ.

During transmission, encoded data comes from the TxDO output and the external transceiver is enabled by the /TxDO output. The 1 MHz transmit clock is generated from the 16 MHz on-chip oscillator.

MK5035N INTERFACE TO THE NIC

The MK5035N interfaces directly to the NIC when CMODE is selected high. The MK5035N is functionally similar to the 82C550A; /RC and RD are the squelch and receive data inputs, and /XEN and XD are the external transceiver enable and transmit data outputs. XSEL input has been selected low to allow the use of an 8 MHz crystal.

BUILDING A StarLAN DAUGHTER CARD FOR THE DP839EB

The DP8391 Serial Network Interface of the DP839EB Evaluation Board has been socketed to allow insertion of a StarLAN daughter card in its place. Unused pins on the DP8391 have been wired with additional signals that are necessary for a StarLAN daughter card. The phone jack is connected to the receiver and transmit pairs. The schematic of a working daughter card is attached.

Daughter Card Pin Assignment

SNI Socket (U9) Pin	Connection
1	COL
2	RXD
3	CRS
4	RXC
6	GND
7	LBK
10	TXD
11	TXC
12	TXE
13	TX− (Pin 2 of Phone Jack)
14	TX+ (Pin 1 of Phone Jack)
16	/RST
19	+5V
21	RX− (Pin 6 of Phone Jack)
22	RX+ (Pin 3 of Phone Jack)

INSTALLATION OF THE DAUGHTER CARD

Once the daughter card has been assembled, the DP8391 Serial Network Interface chip (socketed) can be removed and replaced with the daughter card. Prior to installing the daughter card, the following jumpers must be removed: J1C–J7C, J1E–J7E, and JY (alternatively, JB some DP839EB boards have marked the oscillator jumper as JY or JB. This jumper lies just above the DP8391). All demo software that is provided with the DP839EB also works in StarLAN. The DP839EB is attached to the StarLAN network by connecting twisted pair phone cable between the 8-pin RJ-45 modular jack and the hub.

StarLAN Daughter Card Installation
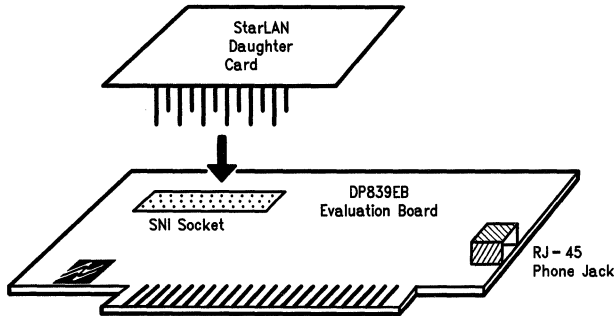
TL/F/9346-3

SUPPORTING DOCUMENTS

The following references can be used to obtain further information.

— DP8390N-1 Data Sheet
— Advanced Peripherals IEEE 802.3 Local Area Network Guide
— DP8390 Data Sheet Addendum, Sept. 1987
— IEEE 802.3 1Base5 ("StarLAN")
— 82C550A Data Sheet (a product of Chips and Technology Inc.)
— MK5035N Data Sheet (a product of Mostek Corporation)
— PT3589 pulse transformer Data Sheet (a product of VALOR Electronics)
— BH500-1436 pulse transformer Data Sheet (a product of BH Electronics)
— NP5413 pulse transformer Data Sheet (a product of Nano Pulse Inc.)

CONSIDERATIONS FOR USING REV. C DP8390N-1

(1) In order for the 4-byte packet header to be properly written by the DP8390, the DMA clock to Network clock may not be greater than 4:1; thus, in StarLAN applications, the DMA clock may not exceed 4 MHz.

Higher bus clock speeds (up to 8 MHz), however, can be achieved by manipulating the packet header under software control. If you are using a DMA clock which is greater than 4 MHz, the DP8390 occasionally copies the Lower Byte Count into the header twice, and fails to write the Upper Byte Count. The Upper byte count, however, may be calculated by subtracting the Next Page Pointer (second byte in the header) with the Next Page Pointer of the previous packet. See DP8390 Datasheet Addendum section 3.1.

(2) Due to the asynchronous nature between the local and remote DMAs, a race condition exists which may cause the local DMA to use the remote DMA's address counter or vice versa. This problem is fixed using a DMA clock synchronous to the transmit clock of the encoder, or a clock derived from the transmit clock.

(3) Because of problem (1) above, the "send packet" command will not operate at bus clock frequencies above 4 MHz. Instead, use the Remote Read DMA and update BNDRY under software control. Note that there is a special consideration for updating BNDRY as specified in section 3.0 of the DP8390 Data Sheet Addendum. BNDRY must always be kept at least one 256-byte buffer behind the CURR pointer.

(4) Rev. C parts will be marked as DP8390N-1 and will operate at a maximum bus clock of 8 MHz.

DAUGHTER CARD PARTS LIST

Resistors		
	120Ω (R3)	1
	560Ω (R4)	1
	2.2 KΩ (R1, 2)	2
	10 MΩ (R7 or R8)	1
Capacitors		
	30 pF (C3)	1
	15 pF (C7, 8 or 15, 16)	2
	0.1 μF (C1, 2)	2
	35 μF (C11)	1
Inductors		
	40 μH (L1)	1

DAUGHTER CARD PARTS LIST (Continued)

Crystal		
	8 MHz (for MK5035N) NDK AT-51	1
	16 MHz (for 82C550A) NDK AT-51	
ICs and Other		
	DS8923 (U2)	1
	MM74HC74 (U1)	1
	MM74HC04 (U7)	1
Manchester ENDECs		
	82C550A (U6)	1
	MK5035N	
Pulse Transformers		
	PT3589	1
	NP5413	
	BH500-1436	

LIST OF OTHER MANUFACTURERS

MANCHESTER ENCODER/DECODERS

82C550A
Chips and Technologies
Ken Buntaran, Technical Marketing Engineer
521 Cottonwood Drive
Milpitas, CA 95035
(408) 434-0600

MK5035N
Mostek Corporation
1310 Electronics Drive
Carrollton, TX 75006
(214) 466-6000

PULSE TRANSFORMERS

BH Electronics
John DeCramer, Engineering Manager
604 Michigan Road
Marshall, MN 56258
(507) 532-3211

Nano Pulse Industries, Inc.
440 Nibus Street
P.O. Box 9398
Brea, CA 92621
(714) 529-2600

Pulse Engineering, Inc.
Rey Bautista, Design Engineer
7250 Convoy Court
San Diego, CA 92111
(619) 268-2449

VALOR Electronics, Inc.
Ernest R. Jensen, Product Development
6750 Nancy Ridge Drive
San Diego, CA 92121
(619) 458-1471

PHONE JACK

Nova-Tronic, Inc.
Jeff Hines, Sales Manager
4701 Patrick Henry Drive #24
Santa Clara, CA 95054
(408) 727-9530

CRYSTALS

NDK-America
20300 Stevens Creek Blvd.
Cupertino, CA 95014
(408) 255-0831

StarLAN Front End Using the MK5035N
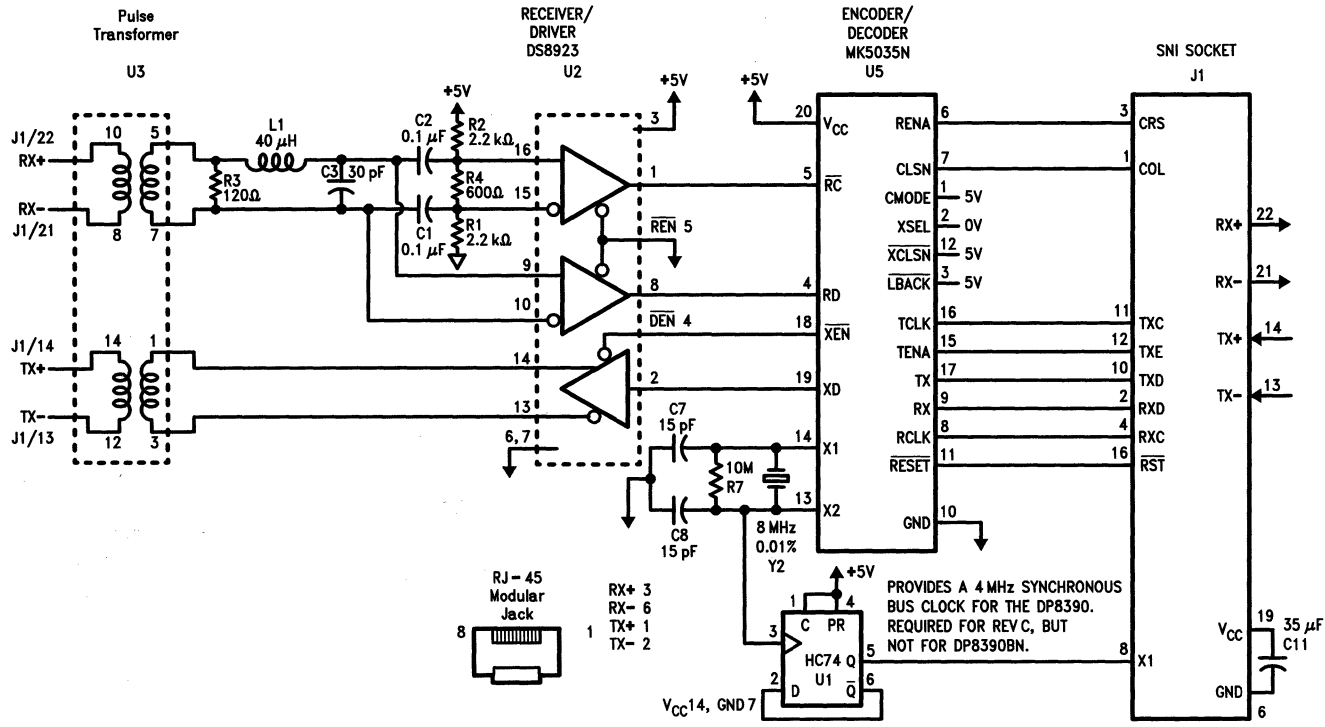
TL/F/9346-4

StarLAN Front End Using the 82C550A

TL/F/9346–5
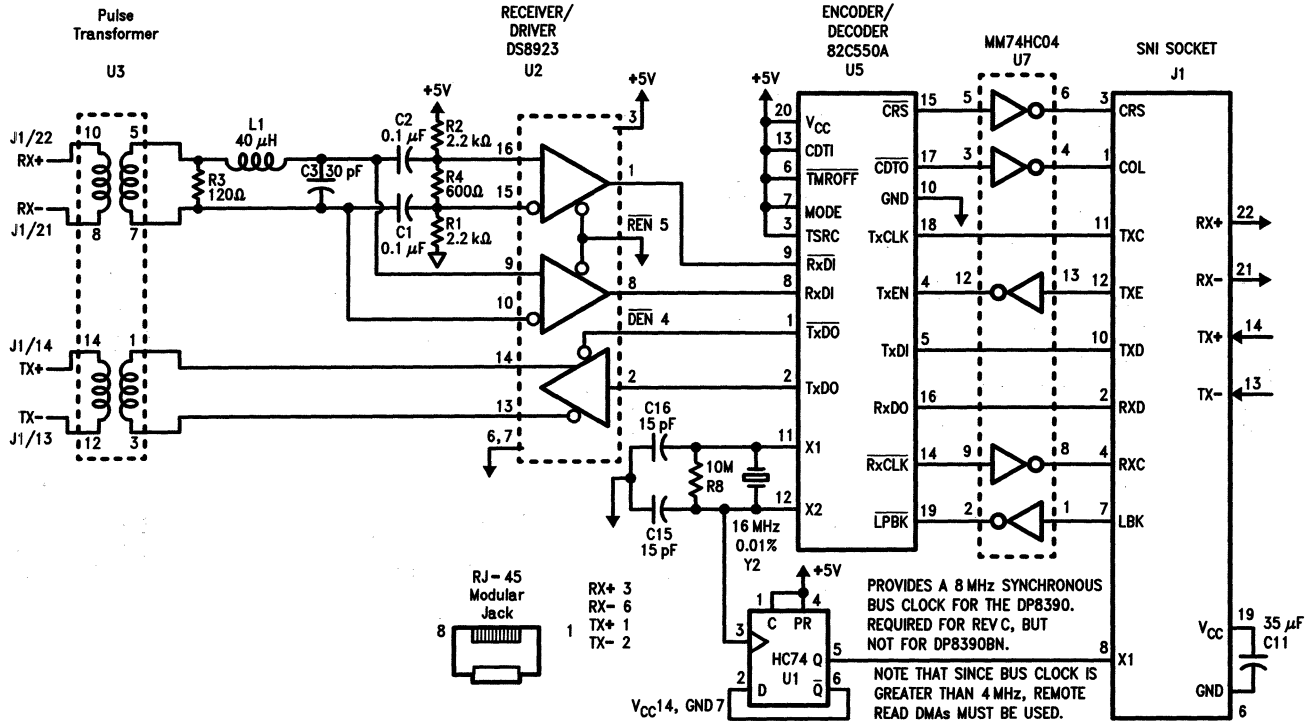
Reliability Data Summary for DP8392

REF: TEST LAB FILES

RDT25406	RDT26627
RDT25500	RDT26638
RDT26562	

ABSTRACT

DP8392 Coaxial Transceiver Interface parts from 8 lots were subjected to Operating Life Test, Temperature and Humidity Bias Test, Temperature Cycle Test, and Electrostatic Discharge Test.

PURPOSE OF TEST

Evaluation of new device and qualification of U.K. fab.

TEST SAMPLE DESCRIPTION/HISTORY

Lot	Device	Package	Date Code	Fab Location	Assembly Location
1	DP8392	N, 16 Leads	8509	NSSC	NSEB
2	DP8392	N, 16 Leads	8513	NSSC	NSEB
3	DP8392	N, 16 Leads	8526	NSSC	NSEB
4	DP8392	N, 16 Leads	8552	NSSC	NSEB
5	DP8392A(-4)	N, 16 Leads	8620	NSUK	
6	DP8392A(-5)	N, 16 Leads	8637	NSUK	
7	DP8392A(-5)	N, 16 Leads	8637	NSUK	
8	DP8392A(-5)	N, 16 Leads	8637	NSUK	

TESTS PERFORMED

Operating Life Test (OPL) (100°C; biased)

Operating Life Test (OPL) (125°C; biased)

Temperature and Humidity Bias Test (THBT) (85°C; 85% R.H.; biased)

Temperature Cycle Test (TMCL) (-40°C, $+125$°C; unbiased)

Electrostatic Discharge Test (ESD) (Human body model: $R = 1500\Omega$; $C = 120$ pF)

CONCLUSIONS

1. The DP8392AN exceeds the IEEE 802.3 specification of 1 million hours Mean Time Between Failure (MTBF).

2. U.K. fab results are comparable to those of Santa Clara. On ESD testing all pins passed at 1000V except for pin 7 (TX^+).

RESULTS

Test	Temperature	Lot	Fab	Time Point—Number of Failures				
				Hours				
				168	336	500	1000	2000
OPL	100°C	1	NSSC	0/50		0/50	0/50	
	100°C	2	NSSC	0/50		0/50	0/50	
	125°C	3	NSSC	0/74				
	125°C	4	NSSC	0/100		0/100	0/100	0/100
	100°C	5	NSUK	0/60				
	100°C	6	NSUK		0/33	0/33	0/33	0/33
	100°C	7	NSUK		0/31	0/31	0/31	0/31
	100°C	8	NSUK		0/33	0/31	0/31	0/31
THBT	85°C; 85% R.H.	1	NSSC	0/50		0/50	0/50	
		2	NSSC	0/50		0/50	0/50	
		3	NSSC	0/75		0/75	0/75	

Test	Temperature	Lot	Fab	Cycles				
				500	1000	2000	3000	
TMCL	-40°C, $+125$°C	4	NSSC	0/70	0/70	0/70	0/70	

ELECTROSTATIC DISCHARGE TEST (ESD) RESULTS

26 parts from 4 wafer lots were tested by the Human Body Model test condition; R = 1500Ω; C = 120 pF. First ground was held common, then V_{EE}. 5 positive and 5 negative pulses were applied for each pin/voltage combination.

Pin	Function	Voltage—Number of Failures	
		500V	**1000V**
1	CD+	0/26	0/20
2	CD−	0/26	0/20
3	RX+	0/26	0/20
4	V_{EE}	0/26	0/20
5	V_{EE}	0/26	0/20
6	RX−	0/26	0/20
7	TX+	6/26	13/20
8	TX−	0/26	0/20
9	HBE	0/26	0/20
10	GND	0/26	0/20
11	RR+	0/26	0/20
12	V_{EE}	0/26	0/20
13	V_{EE}	0/26	0/20
14	RXI	0/26	0/20
15	TXO	0/26	0/20
16	CDS	0/26	0/20

Further characterization has been done to determine individual pin ESD damage thresholds. In particular, for pin 7 (TX+), 80 parts from 4 wafer lots were tested. Pin 7 ESD damage thresholds varied from 200V–300V to 2000V–3000V, with a mean of 1800V.

MTBF (MEAN TIME BEFORE FAILURE) CONSIDERATIONS

Results total: 212, 000 device hours at 125°C, 0 failures

301,000 device hours at 100°C, 0 failures

Assume: E_a = 0.7 eV

P_d = 800 mW

θ_{ja} = 45°C/W

Chi-square statistics, 60% confidence

Then: $MTBF_{min}$ at 25°C ambient = 93,000,000 device hours.

$MTBF_{min}$ at 70°C ambient = 5,100,000 device hours.

![National Semiconductor logo] **National Semiconductor**

Repeater Interface Controller

General Description

The Repeater Interface Controller (RIC) fully implements the IEEE 802.3 repeater specification—the repeater, partition and jabber lockup protection state machines, TW1–TW6 and Transmit timers and Consecutive Collision counters.

The RIC has an on-chip Phase-Locked-Loop (PLL) for Manchester decoding, a Manchester encoder and a FIFO for preamble regeneration.

Each RIC can connect to 13 cable segments via transceivers. One port is fully AUI compatible, while the other 12 can connect to coaxial and twisted-pair transceivers. In addition, up to 8 RICs can be cascaded together to implement a larger repeater unit.

The RIC is configurable for specific applications. It provides port status information for LEDs and a simple interface for processors (intelligent repeater application).

Features

- IEEE 802.3 Compatible
- 13 ports per chip
- Each port may connect independently to a coax or twisted-pair transceiver
- Cascadable up to 8 chips
- 84-pin Plastic Leaded Chip Carrier (PLCC) package
- CMOS process for low power dissipation
- Single 5V supply
- On-chip FIFO, Manchester encoder and decoder
- Partition state machine for each port with separate timers (TW5, TW6) and collision counter (CC)
- Provides port status information (receive, collision, and partition for LEDs)
- Power-up configuration options:
  - TW1 and TW2 lengths
  - Consecutive collision count limit (32 or 64)
  - Interrupt sources (receive, collision, and partition)
  - Partition on loopback (DO to DI) failure
  - Disable partitioning (hard partition is unaffected)

- Simple processor interface for repeater management:
  - Interrupts on port activity (receive, collision and partition)
  - Write capability to each port to hard partition or disable transmitters
  - Read capability for extensive port information (status, partition type, port configuration)

Table of Contents

1.0 System Diagram

IEEE 802.3 Compatible Repeater Unit

TL/F/10489–1

Section 2

**High Speed Serial/IBM
Data Communications**

2

Section 2 Contents

![National Semiconductor logo] **National Semiconductor**

DP8340/NS32440 IBM 3270 Protocol Transmitter/Encoder

General Description

The DP8340/NS32440 generates a complete encoding of parallel data for high speed serial transmission which conforms to the protocol as defined by the IBM 3270 information display system standard. The DP8340/NS32440 converts parallel input data into a serial data stream. Although the IBM standard covers biphase serial data transmission over a coax line, the DP8340/NS32440 also adapts to general high speed serial data transmission over other than coax lines, at frequencies either higher or lower than the IBM standard.

The DP8340/NS32440 and its complementary chip, the DP8341 (receiver/decoder) have been designed to provide maximum flexibility in system designs. The separation of the transmitter/receiver functions provides convenient addition of more receivers at one end of a biphase line without the need of unused transmitters. This is specifically advantageous in control units where typical biphase data is multiplexed over many biphase lines and the number of receivers generally exceeds the number of transmitters.

Features

- Ten bits per data byte transmission
- Single-byte or multi-byte transmission
- Internal parity generation (even or odd)
- Internal crystal controlled oscillator used for the generation of all required chip timing frequencies
- Clock output directly drives receiver (DP8341) clock input
- Input data holding register
- Automatic clear status response feature
- Line drivers at data outputs provide easy interface to biphase coax line or general transmission lines
- < 2 ns driver output skew
- Bipolar technology provides TTL input/output compatibility
- Data outputs power up/down glitch free
- Internal power up clear and reset
- Single +5V power supply

Connection Diagrams

Plastic Chip Carrier

Dual-In-Line Package

TL/F/5251–1

Top View
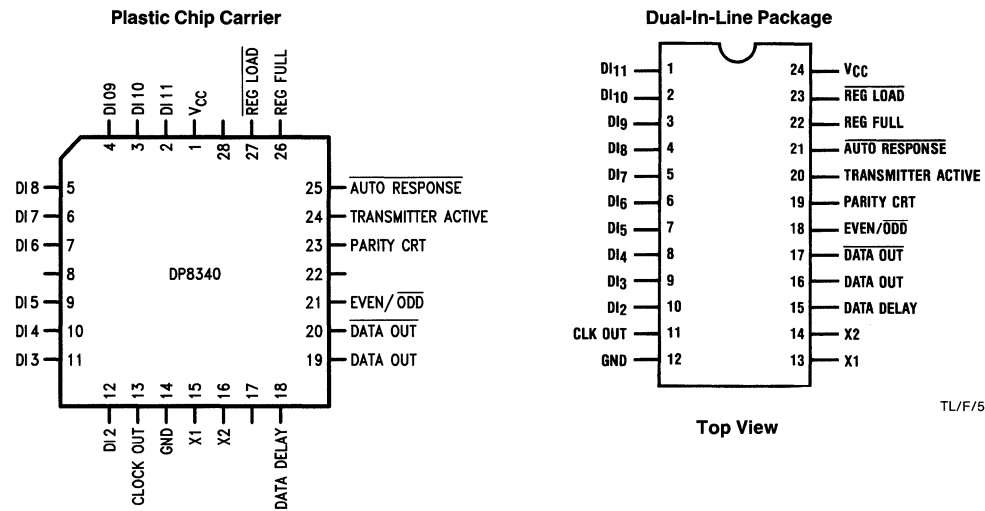
TL/F/5251–19

FIGURE 1

Order Number DP8340/NS32440 J, N or V
See NS Package Number J24A, N24A or V28A

2

Block Diagram
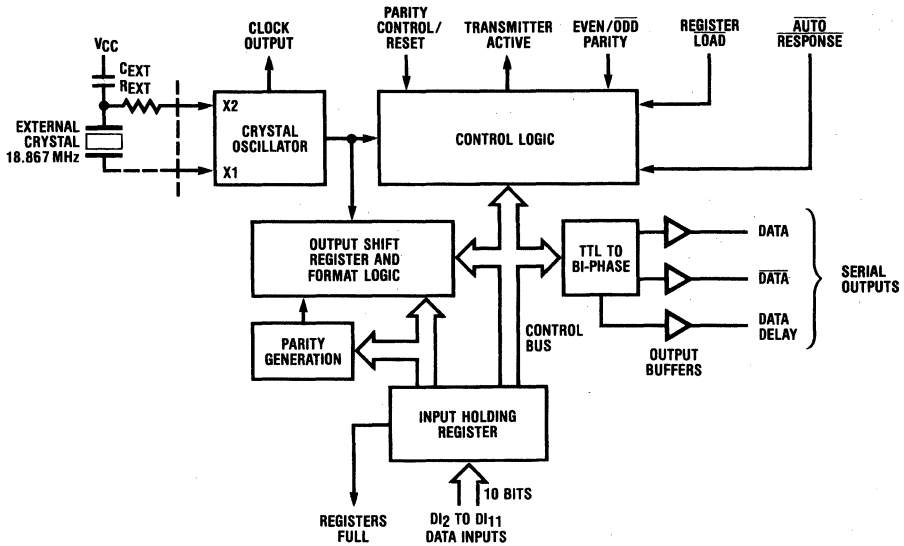
TL/F/5251–2

FIGURE 2. DP8340/NS32440 Serial Bi-Phase Transmitter/Encoder Block Diagram

Functional Description

Figure 2 is a block diagram of the DP8340/NS32440 bi-phase Transmitter/Encoder. The transmitter/encoder contains a crsytal oscillator whose input is a crystal with a frequency eight (8) times the data rate. A Clock Output is provided to drive the DP8341 receiver/decoder Clock Input and other system components at the oscillator frequency. Additionally, the oscillator drives the control logic and output shift register/format logic blocks.

Data is parallel loaded from the sytem data bus to the transmitter/encoder's input holding register. This data is in turn loaded by the transmitter/encoder to its output shift register if this register was empty at the time of the load. During this load, message formatting and parity are generated. The formatted message is then shifted out at the bit rate frequency to the TTL to biphase block which generates the proper data bit formatting. The three data outputs, DATA, $\overline{\text{DATA}}$, and DATA DELAY provide for flexible interface to the coax line with a minimum of external components.

The Control Logic block interfaces to all blocks to insure proper chip operation and sequencing. It controls the type of parity generation through the Even/Odd Parity input. An additional feature provided by the transmitter/encoder is generation of odd parity and placement in bit 10 position

while still maintaining even or odd parity in the bit 12 position. This is the format of data word bytes and other commands in the 3270 Standard. The Parity Control input is the pin which controls when this operation is in effect.

Another feature of the transmitter/encoder is the internal TT/AR (Transmission Turnaround/Auto Response) capability. After each Write type message from the control unit in the 3270 Standard, the receiving unit must respond with clean status (bits 2 through 11). With the transmitter/encoder, this function is accomplished simply by forcing the $\overline{\text{Auto-Response}}$ input to the Logic "0" state.

Operation of the transmitter/encoder is automatic. After the first data byte is loaded, the Transmitter Active output is set and the transmitter/encoder immediately formats the input data and serially shifts it out its data outputs. If the message is a multi-byte message, the internal format logic will modify the message data format for multibyte as long as the next byte is loaded to the input holding register before the last data bit of the previous data byte is transferred out of the internal output shift register. After all data is shifted out of the transmitter/encoder the Transmitter Active output will return to the inactive state.

Detailed Pin/Functional Description

Crystal Inputs X1 and X2

The oscillator is controlled by an external, parallel resonant crystal connected between the X1 and X2 pins. Normally, a fundamental mode crystal is used to determine the operating frequency of the osicllator; however, overtone mode crystals may be used.

Crystal Specifications (Parallel Resonant)

Type	AT-cut crystal
Tolerance	0.005% at 25°C
Stability	0.01% from 0°C to +70°C
Resonance	Fundamental (Parallel)
Maximum Series Resistance	Dependent on Frequency (For 18.867 MHz, 50Ω)
Load Capacitance	15 pF
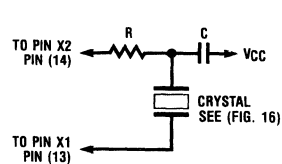
FREQ	R	C
10 MHz to 20 MHz	500Ω ±10%	30 pF
>20 MHz	120Ω ±10%	15 pF

TL/F/5251–3

FIGURE 3. Connection Diagram

If the DP8340/NS32440 transmitter is clocked by a system (clock crystal oscillator not used), pin 13 (X1 input) should be clocked directly using a Schottky series (74S) circuit. Pin 14 (X2 input) may be left open. The clocking frequency must be set at eight times the data bit rate. Maximum input frequency is 28 MHz. For the IBM 3270 Interface, this frequency is 18.867 MHz. At this frequency, the serial bit rate will be 2.358 Mbits/sec.

Clock Output

The Clock Output is a buffered output derived directly from the crystal oscillator block and clocks at the oscillator frequency. It is designed to directly drvie the DP8341 receiver/decoder Clock Input as well as other system components.

Registers Full

This output is used as a flag by the external operating system. A logic "1" (active state) on this output indicates that both the internal output shift register and the input holding register contain active data. No additional data should be loaded until this output returns to the logic "0" (inactive state).

Transmitter Active

This output will be in the logic "1" state while the transmitter/encoder is about to transmit or in the process of transmitting data. Otherwise, it will assume the logic "0" state indicating no data presently in either the input holding or output shift registers.

Register Load

The Register Load input is used to load data from the Data Inputs to the input holding register. The loading function is edge sensitive, the data present during the logic "0" state of this input is loaded, and the input data must be valid before the logic "0" to logic "1" transition. It is after this transition that the transmitter/encoder begins formatting of data for serial transmission.

Auto Response (TT/AR)

This input provides for automatic clear data transmission (all bits in logic "0") without the need of loading all zero's. When a logic "0" is forced on this inpiut the transmitter/encoder immediately responds with transmission of "clean status". This function is necessary after the completion of each write type command and in other functions in the 3270 specification. In the logic "1" state the transmitter/encoder transmits data entered on the Data Inputs.

Even/Odd Parity

This input sets the internal logic of the DP8340/NS32440 transmitter/encoder to generate either even or odd parity for the data byte in the bit 12 position. When this pin is in the logic "0" state odd parity is generated. In the logic "1" state even parity is generated. This feature is useful when the control unit is performing a loop back check and at the same time the controller wishes to verify proper data transmission with its receiver/decoder.

Parity Control/Reset

Depending on the type of message transmitted, it is at times necessary in the IBM 3270 specification to generate an additional parity bit in the bit 10 position. The bit generated is odd parity on the previous eight (8) bits of data. When the Parity Control input is in the logic "1" state the data entered at the Data Bit 10 position is placed in the transmitted word. With the Parity Control input in the logic "0" state the Data Bit 10 input is ignored and odd parity on the previous data bits is placed in the normal bit 10 position while overall word parity (bit 12) is even or odd (controlled by Even/Odd Parity input). This eliminates the need for external logic to generate the parity on the data bits.

Truth Table

Parity Control Input	Transmitted Data Bit 10
Logic "1"	Data entered on Data Input 10
Logic "0"	Odd Parity on 8-bit data byte

When this input is driven to a voltage that exceeds the power supply level (9V to 13V) the transmitter/encoder is reset.

Serial Outputs—DATA, DATA, and DATA DELAY

These three output pins provide for convenient application of data to the biphase Coax line (see *Figure 15* for application). The Data outputs are a direct bit representation of the biphase data while the DATA DELAY output provides the necessary increment to clearly define the four (4) DC levels of the pulse. The DATA and DATA outputs add flexibility to the DP8340/NS32440 transmitter/encoder for use in high speed differential line driving applications.

2

Functional Timing Waveforms—Message Format

Single Byte Transmission
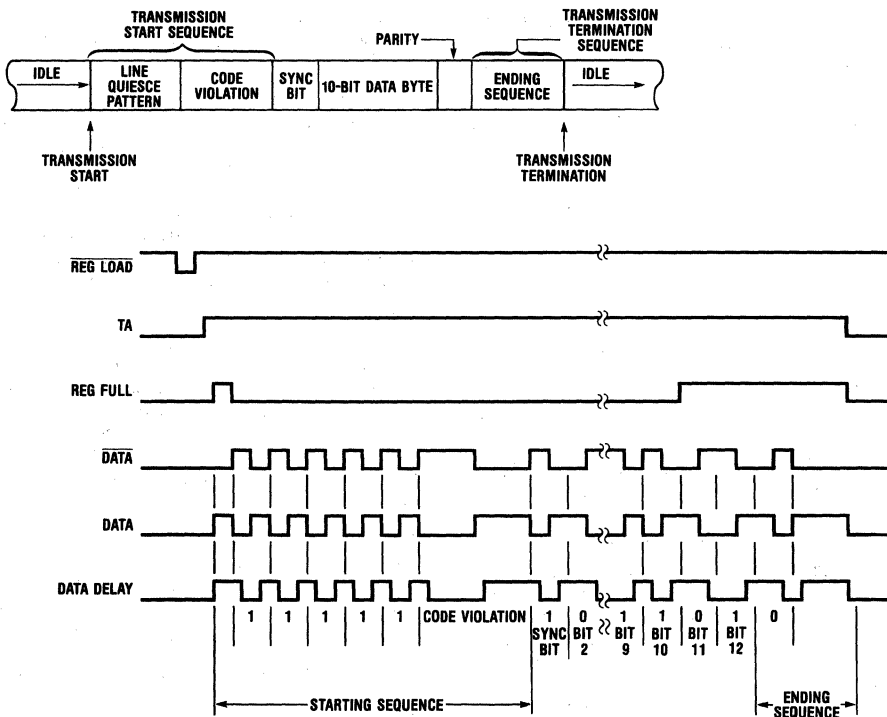
FIGURE 4. Overall Timing Waveforms for Single Byte

TL/F/5251–4

Multi-Byte Transmission
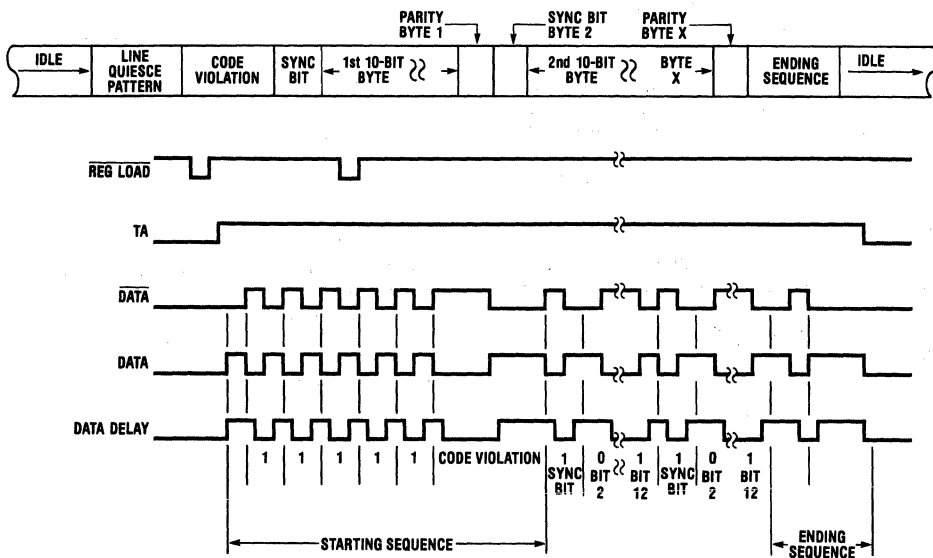
FIGURE 5. Overall Timing Waveforms for Multi-Byte

TL/F/5251–5

Absolute Maximum Ratings (Note 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage, V_{CC}	7V
Input Voltage	5.5V
Output Voltage	5.25V
Storage Temperature Range	$-65°C$ to $+150°C$
Lead Temperature (Soldering, 10 sec.)	300°C

Maximum Power Dissipation @25°C*
Cavity Package	2237 mW
Dual-In-Line Package	2500 mW
Plastic Chip Carrier	1720 mw

*Derate cavity package 14.9 mW/°C above 25°C; derate dual-in-line package 20 mW/°C above 25°C; derate PCC package 13.8 mW/°C above 25°C.

Operating Conditions

	Min	Max	Units
Supply Voltage, (V_{CC})	4.75	5.25	V
Ambient Temperature, T_A	0	$+70$	°C

Electrical Characteristics (Notes 2 and 3)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{IH}	Logic "1" Input Voltage (All Inputs Except X1 and X2)		2.0			V
V_{IL}	Logic "0" Input Voltage (All Inputs Except X1 and X2)				0.8	V
V_{CLAMP}	Input Clamp Voltage (All Inputs Except X1 and X2)	$I_{IN} = -12$ mA		-0.8	-1.2	V
I_{IH}	Logic "1" Input Current Register Load Input	$V_{CC} = 5.25V$ $V_{IN} = 5.25V$		0.3	120	μA
	All Others Except X1 and X2			0.1	40	μA
I_{IL}	Logic "0" Input Current Register Load Input	$V_{CC} = 5.25V$ $V_{IN} = 0.5V$		-15	-300	μA
	All Inputs Except X1 and X2			-5	-100	μA
V_{OH1}	Logic "1" All Outputs Except CLK OUT, DATA, $\overline{DATA}$, and DATA DELAY	$I_{OH} = -100\ \mu A$	3.2	3.9		V
		$I_{OH} = -1$ mA	2.5	3.4		V
V_{OH2}	Logic "1" for CKL OUT, DATA, $\overline{DATA}$ and DATA DELAY Outputs	$I_{OH} = -10$ mA	2.6	3.0		V
V_{OL1}	Logic "0" All Outputs Except CLK OUT, DATA, $\overline{DATA}$ and DATA DELAY Outputs	$I_{OL} = 5$ mA		0.35	0.5	V
V_{OL2}	Logic "0" for CLK OUT, DATA, $\overline{DATA}$ and DATA DELAY Outputs	$I_{OL} = 20$ mA		0.4	0.6	V
I_{OS1}	Short Circuit Current for All Outputs Except CLK OUT, DATA, $\overline{DATA}$, and DATA DELAY	$V_{OUT} = 0V$ (Note 4)	-10	-30	-100	mA
I_{OS2}	Short Circuit Current for DATA, $\overline{DATA}$, and DATA DELAY Outputs	$V_{OUT} = 0V$ (Note 4)	-50	-140	-350	mA
I_{OS3}	Short Circuit Current for CLK OUT	(Note 4)	-30	-90	-200	mA
I_{CC}	Power Supply Current	$V_{CC} = 5.25V$		170	250	mA

Timing Characteristics Oscillator Frequency = 18.867 MHz (Notes 2 and 3)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
t_{pd1}	$\overline{REG\ LOAD}$ to Transmitter Active (T_A) Positive Edge	Load Circuit 1 *Figure 7*		60	90	ns
t_{pd2}	$\overline{REG\ LOAD}$ to REG Full; Positive Edge	Load Circuit 1 *Figure 7*		45	75	ns
t_{pd3}	Register Full to T_A; Negative Edge	Load Circuit 1 *Figure 7*		40	70	ns
t_{pd4}	Positive Edge of $\overline{REG\ LOAD}$ to Positive Edge of DATA	Load Circuits 1 & 2 *Figure 9*		50	80	ns

2

Timing Characteristics Oscillator Frequency = 18.867 MHz (Notes 2 and 3) (Continued)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
t_{pd5}	REG LOAD to DATA; Positive Edge	Load Circuits 1 & 2 Figure 9, (Note 6)		380	475	ns
t_{pd6}	REG LOAD to DATA DELAY; Positive Edge	Load Circuits 1 & 2 Figure 9, (Note 6)		160	250	ns
t_{pd7}	Positive Edge of DATA to Negative Edge of DATA DELAY	Load Circuit 2 Figure 9, (Note 6)		100	115	ns
t_{pd8}	Positive Edge of DATA DELAY to Negative Edge of DATA	Load Circuit 2 Figure 9, (Note 6)		110	125	ns
t_{pd9}, t_{pd10}	Skew between DATA and DATA	Load Circuit 2 Figure 9		2	6	ns
t_{pd11}	Negative Edge of Auto Response to Positive Edge of TA	Load Circuit 1 Figure 10		70	110	ns
t_{pd12}	Maximum Time Delay to Load Second Byte after Positive Edge of REG FULL	Load Circuit 1 Figure 8, (Note 6)			$4 \times T - 50$	ns
t_{pd13}	X1 to CLK OUT; Positive Edge	Load Circuit 2 Figure 13		21	30	ns
t_{pd14}	X1 to CLK OUT; Negative Edge	Load Circuit 2 Figure 13		23	33	ns
t_{pd15}	Negative Edge of AR to Positive Edge of REG FULL	Load Circuit 1 Figure 10		45	75	ns
t_{pd16}	Skew between TA and REG FULL during Auto Response	Load Circuit 1 Figure 10		50	80	ns
t_{pd17}	REG LOAD to REG FULL; Positive Edge for Second Byte	Load Circuit 1 Figure 14		45	75	ns
t_{pw1}	REG LOAD Pulse Width	Figure 12	40			ns
t_{pw2}	First REG FULL Pulse Width (Note 5)	Load Circuit 1 Figure 7, (Note 6)		$8 \times T + 60$	$8 \times T + 100$	ns
t_{pw3}	REG FULL Pulse Width prior to Ending Sequence (Note 5)	Load Circuit 1, Figure 7, (Note 6)		$5 \times B$		ns
t_{pw4}	Pulse Width for Auto Response	Figure 10	40			ns
t_S	Data Setup Time prior to REG LOAD Positive Edge, Hold Time (t_H) = 0 ns	Figure 12		15	25	ns
t_{r1}	Rise Time for DATA, DATA, and DATA DELAY Output Waveform	Load Circuit 2 Figure 11		7	13	ns
t_{f1}	Fall Time for DATA, DATA, and DATA DELAY Output Waveform	Load Circuit 2 Figure 11		5	11	ns
t_{r2}	Rise Time for TA and REG FULL	Load Circuit 1 Figure 15		20	30	ns
t_{f2}	Fall Time for TA and REG FULL	Load Circuit 1 Figure 15		15	25	ns
f_{MAX}	Data Rate Frequency (Clock Input must be 8X this Frequency)	(Note 7)	DC		3.5	Mbits/s

Note 1: "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

Note 2: Unless otherwise specified, min./max. limits apply across the 0°C to +70°C temperature range and the 4.75V to 5.25V power supply range. All typical values are for T_A = 25°C and V_{CC} = 5.0V.

Note 3: All currents into device pins are shown as positive; all currents out of device pins are shown as negative; all voltages are referenced to ground, unless otherwise specified. All values shown as max. or min. are so classified on absolute basis.

Note 4: Only one output should be shorted at a time. Output should not be shorted for more than one second at a time.

Note 5: T = 1/(Oscillator Frequency), unit for T should be ns. B = 8T

Note 6: Oscillator Frequency Dependent.

Note 7: For the IBM 3270 Interface, the data rate frequency is 2.358 Mbits/s. 28 MHz clock frequency corresponds to 3.75% jitter when referenced to Figure 10 of DP8341 Datasheet.

Load Circuit 1

V_{CC}

$R_L = 2k$

15 pF

TL/F/5251-6

Load Circuit 2

V_{CC}

$R_L = 2k$

50 pF
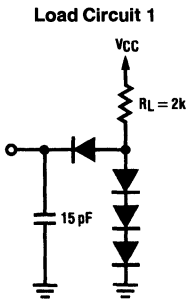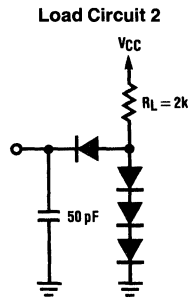
TL/F/5251-7

FIGURE 6. Test Load Circuits

Timing Waveforms

REG LOAD   50%   3V

TA t_{pd1} V_{OH} 50% V_{OL}

REG FULL t_{pd2} t_{pd3} V_{OH} 50% V_{OL}

t_{pw2} t_{pw3}
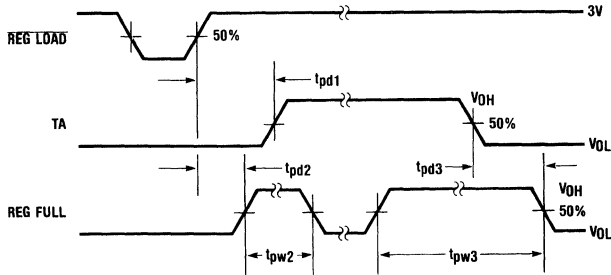
TL/F/5251-8

FIGURE 7. Timing Waveforms for Single Byte Transfer

REG LOAD   50%   3V

REG FULL t_{pd12} V_{OH} 50% V_{OL}

WINDOW
TO LOAD MULTI-BYTE DATA
$17\frac{1}{2} \times B$
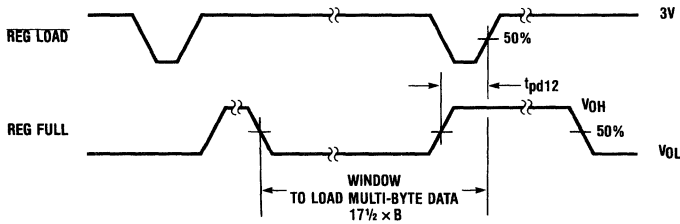
TL/F/5251-9

FIGURE 8. Maximum Window to Load Multi-Byte Data

REG LOAD   50%   3V

DATA t_{pd4} V_{OH} 50% V_{OL}

DATA t_{pd5} t_{pd9} t_{pd10} V_{OH} 50% V_{OL}

DATA DELAY t_{pd6} t_{pd7} t_{pd8} V_{OH} 50% V_{OL}
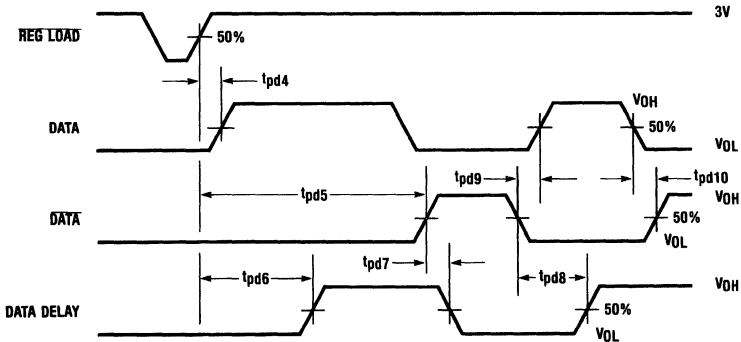
TL/F/5251-10

FIGURE 9. Timing Waveforms for Three Serial Outputs

2

Timing Waveforms (Continued)
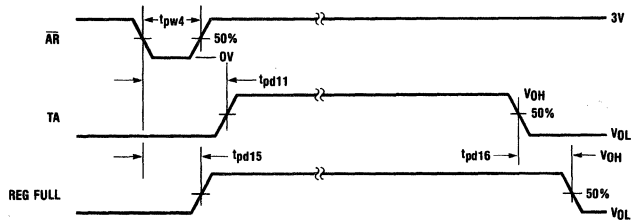
FIGURE 10. Timing Waveforms for $\overline{\text{Auto-Response}}$
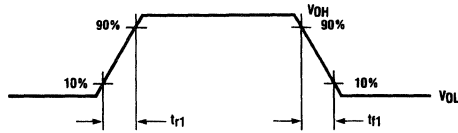
TL/F/5251–11

FIGURE 11. Output Waveform for DATA, $\overline{\text{DATA}}$, DATA DELAY (Load Circuit 2)
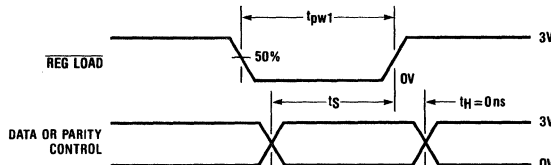
TL/F/5251–12

FIGURE 12. Register Load Waveform Requirement
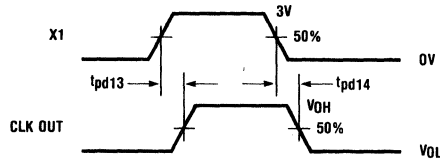
TL/F/5251–13

FIGURE 13. Timing Waveforms for Clock Pulse
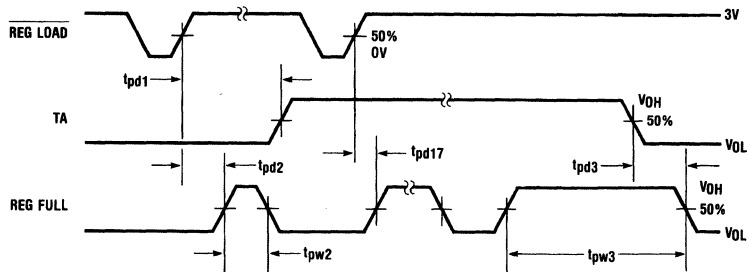
TL/F/5251–14

FIGURE 14. Timing Waveforms for Two Byte Transfer
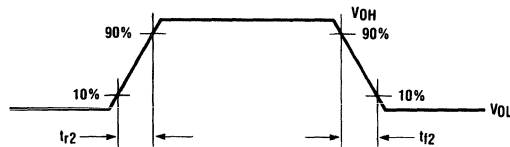
TL/F/5251–15

FIGURE 15. Rise and Fall Time Measurement for TA and REG Full

TL/F/5251–16

Typical Applications
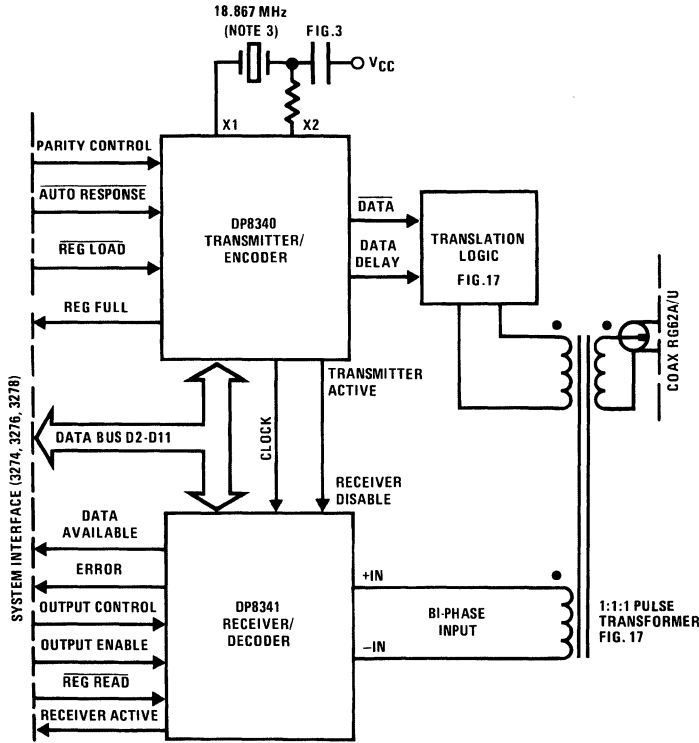
FIGURE 16. Typical Applications for IBM 3270 Interface

TL/F/5251–17

TL/F/5251–18
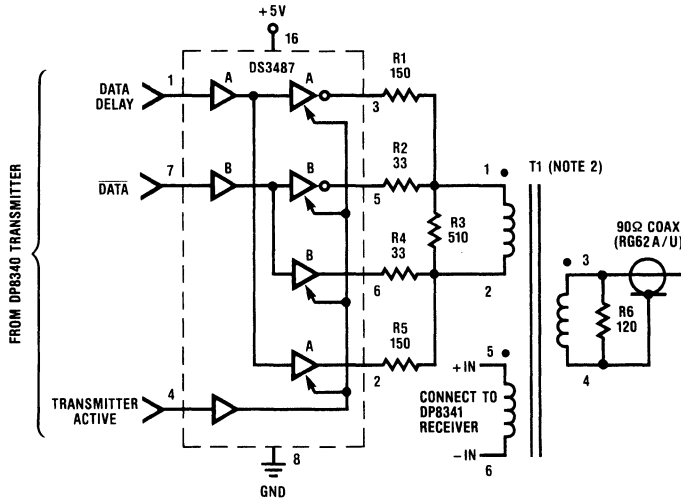
Note 1: Resistance values are in Ω, $\pm 5\%$, $\frac{1}{4}$ W

Note 2: T1 is a 1:1:1 pulse transformer, $L_{MIN} = 500 \mu H$ for 18 MHz system clock. Pulse Engineering Part No. 5762/Surface Mount, 5762M/PE-85762. Technitrol Part No. 11LHA, Valor Electronics Part No. CT1501 or equivalent transformers.

Note 3: Crystal manufacturer's Midland Ross Corp. NEL Unit Part No. NE-18A (C2560N) @ 18.867 MHz and the Viking Group of San Jose, CA Part No. VXB46NS @ 18.867 MHz.

FIGURE 17. Translation Logic

National
Semiconductor

DP8341/NS32441 IBM 3270 Protocol Receiver/Decoder

General Description

The DP8341/NS32441 provides complete decoding of data for high speed serial data communications. In specific, the DP8341/NS32441 recognizes serial data that conforms to the IBM 3270 Information Display System Standard and converts it into ten (10) bits of parallel data. Although this standard covers biphase serial data transmission over a coax line, this device easily adapts to generalized high speed serial data transmission on other than coax lines at frequencies either higher or lower than the IBM 3270 standard.

The DP8341/NS32441 receiver and its complementary chip, the DP8340 transmitter, are designed to provide maximum flexibility in system designs. The separation of transmitter and receiver functions allows addition of more receivers at one end of the biphase line without the necessity of adding unused transmitters. This is advantageous specifically in control units where typically biphase data is multiplexed over many biphase lines and the number of receivers generally outnumber the number of transmitters. The separation of transmitter and receiver function provides an additional advantage in flexibility of data bus organization. The data bus outputs of the receiver are TRI-STATE®, thus enabling the bus configuration to be organized as either a common transmit/receive (bi-directional) bus or as separate transmit and receive busses for higher speed.

Features

■ DP8341/NS32441 receivers ten (10) bit data bytes and conforms to the IBM 3270 Interface Display System Standard

■ Separate receiver and transmitter provide maximum system design flexibility

■ Even parity detection

■ High sensitivity input on receiver easily interfaces to coax line

■ Standard TTL data input on receiver provides generalized transmission line interface and also provides hysteresis

■ Data holding register

■ Multi-byte or single byte transfers

■ TRI-STATE receiver data outputs provide flexibility for common or separated transmit/receive data bus operation

■ Data transmission error detection or receiver provides for both error detection and error type definition

■ Bi-polar technology provides TTL input/output compatibility with excellent drive characteristics

■ Single +5V power supply operation

Connection Diagrams

Plastic Chip Carrier
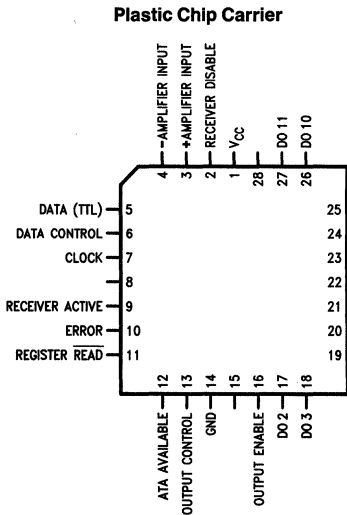
Order Number DP8341V or NS32441V
See NS Package Number V28A

TL/F/5238–1

Dual-In-Line Package

RECEIVER DISABLE	1	24 — VCC
+AMPLIFIER INPUT	2	23 — DO11
−AMPLIFIER INPUT	3	22 — DO10
DATA (TTL)	4	21 — DO9
DATA CONTROL	5	20 — DO8
CLOCK	6	19 — DO7
RECEIVER ACTIVE	7	18 — DO6
ERROR	8	17 — DO5
REGISTER READ	9	16 — DO4
DATA AVAILABLE	10	15 — DO3
OUTPUT CONTROL	11	14 — DO2
GND	12	13 — OUTPUT ENABLE
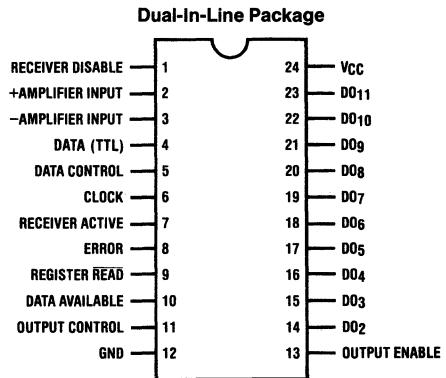
TL/F/5238–2

Top View

Order Number DP8341J or DP8341N
See NS Package Number J24A or N24A

FIGURE 1

Block Diagram
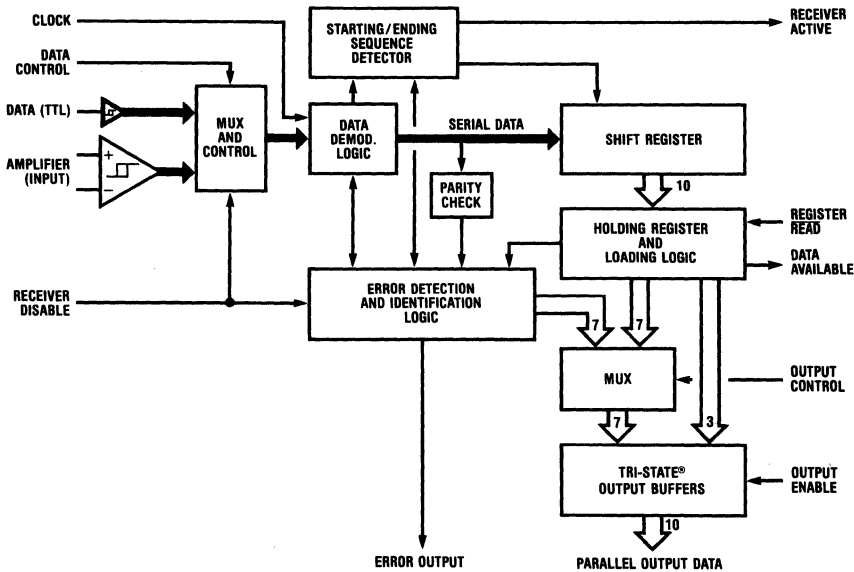
TL/F/5238–3

FIGURE 2. DP8341/NS32441 Serial Bi-Phase Receiver/Decoder Block Diagram

Block Diagram Functional Description

Figure 2 is a block diagram of the DP8341/NS32441. This chip is essentially a serial in/parallel out shift register. However, the serial input data must conform to a very specific format (see *Figures 3–5*). The message will not be recognized unless the format of the starting sequence is correct. Deviations from the format in the data, sync bit, parity or ending sequence will cause an error to be detected, terminating the message.

Data enters the receiver through the differential input amplifier or the TTL Data input. The differential amplifier is a high sensitivity input which may be used by connecting it directly to a transformer coupled coax line, or other transmission medium. The TTL Data input provides 400 mV of hysteresis and recognizes TTL logic levels. The data then enters the demodulation block.

The data demodulation block samples the data at eight (8) times the data rate and provides signals for detecting the starting sequence, ending sequence, and errors. Detection of the starting sequence sets the Receiver Active output high and enables the input shift register.

As the ten bits of data are shifted into the shift register, the receiver will verify that even parity is maintained on the data bits and the sync bit. After one complete data byte is received, the contents of the input shift register is parallel loaded to the holding register, assuming the holding register is empty, and the Data Available output is set. If the holding register is full, this load will be delayed until that register has been read. If another data byte is received when the shift

register and the holding register are full a Data Overflow Error will be detected, terminating the message. Data is read from the holding register through the TRI-STATE Output Buffers. The Output Enable input is the TRI-STATE control for these outputs and the Register Read input signals the receiver that the read has been completed.

When the receiver detects an ending sequence the Receiver Active output will be reset to a logic "0" indicating the message has been terminated. A message will also terminate when an error is detected. The Receiver Active output used in conjunction with the Error output allows quick response to the transmitting unit when an error free message has been received.

The Error Detection and Identification block insures that valid data reaches the outputs of the receiver. Detection of an error sets the Error output to a logic "1" and resets the Receiver Active output to a logic "0" terminating the message. The error type may be read from the data bus outputs by setting the Output Control input to logic "0" and enabling the TRI-STATE outputs. The data bit outputs have assigned error definitions (see error code definition table). The Error output will return to a logic "0" when the next starting sequence is received, or when the error is read (Output Control to logic "0" and a Register Read performed).

The Receiver Disable input is used to disable both the amplifier and TTL Data receiver inputs. It will typically be connected directly to the Transmitter Active output of the DP8340 transmitter circuit (see *Figure 12*).

2

Detailed Functional Pin Description

RECEIVER DISABLE

This input is used to disable the receiver's data inputs. The Receiver Disable input will typically be connected to the Transmitter Active output of the DP8340. However, at the system controller it is necessary for both the transmitter and receiver to be active at the same time in the loop-back check condition. This variation can be accomplished with the addition of minimal external logic.

Truth Table

Receiver Disable	Data Inputs
Logic "0"	Active
Logic "1"	Disabled

AMPLIFIER INPUTS

The receiver has a differential input amplifier which may be directly connected to the transformer coupled coax line. The amplifier may also be connected to a differential type TTL line. The amplifier has 20 mV of hysteresis.

DATA INPUT

This input can be used either as an alternate data input or as a power-up check input. If the system designer prefers to use his own amplifier, instead of the one provided on the receiver, then this TTL input may be used. Using this pin as an alternate data input allows self-test of the peripheral system without disturbing the transmission line.

DATA CONTROL

This input is the control pin that selects which of the inputs are used for data entry to the receiver.

Truth Table

Data Control	Data Input To
Logic "0"	Data Input
Logic "1"	Amplifier Inputs

Note: This input is also used for testing. When the input voltage is raised to 7.5V the chip resets.

CLOCK INPUT

The input is the internal clock of the receiver. It must be set at eight (8) times the line data bit rate. For the IBM 3270 Standard, this frequency is 18.87 MHz or a data bit rate of 2.358 MHz. The crystal-controlled oscillator provided in the DP8340 transmitter also operates at this frequency. The Clock Output of the transmitter is designed to directly drive the receiver's Clock Input. In addition, the receiver is designed to operate correctly to a data bit rate of 3.5 MHz.

RECEIVER ACTIVE

The purpose of this output is to inform the external system when the DP8341/NS32441 is in the process of receiving a message. This output will transition to a logic "1" state after the receipt of a valid starting sequence and transition to logic "0" when a valid ending sequence is received or an error is detected. This output combined with the Error output will inform the operating system of the end of an error free data transmission.

ERROR

The Error output transitions to a logic "1" when an error is detected. Detection of an error causes the Receiver Active and the Data Available outputs to transition to a logic "0". The Error output returns to a logic "0" after the error register has been read or when the next starting sequence is detected.

REGISTER READ

The Register Read input when driven to the logic "0" state signals the receiver that data in the holding register is being read by the external operating system. The data present in the holding register will continue to remain valid until the Register Read input returns to the logic "1" condition. At this time, if an additional byte is present in the input shift register it will be transferred to the holding register, otherwise the data will remain valid in the holding register. The Data Available output will be in the logic "0" state for a short interval while a new byte is transferred to the holding register after a register read.

DATA AVAILABLE

This output indicates the existence of a data byte within the output holding register. It may also indicate the presence of a data byte in both the holding register and the input shift register. This output will transition to the logic "1" state as soon as data is available and return to the logic "0" state after each data byte has been read. However, even after the last data byte has been read and the Data Available output has assumed the logic "0" state, the last data byte read from the holding register will remain until new data has been received.

Detailed Functional Pin Description (Continued)

OUTPUT CONTROL

The Output Control input determines the type of information appearing at the data outputs. In the logic "1" state data will appear, in the logic "0" state error codes are present.

Truth Table

Output Control	Data Outputs
Logic "0"	Error Codes
Logic "1"	Data

OUTPUT ENABLE

The Output Enable input controls the state of the TRI-STATE Data outputs.

Truth Table

Output Enable	TRI-STATE Data Outputs
Logic "0"	Disabled
Logic "1"	Active

DATA OUTPUTS

The DP8341 has a ten (10) bit TRI-STATE data bus. Seven bits are multiplexed with error bits. The error bits are de-fined in the table below. The Output Control input is the multiplexer control for the Data/Error bits.

Error Code Definition

Data Bit	Error Type
DO2	Data Overflow (Byte not removed from holding register when it and the input shift register are both full and new data is received)
DO3	Parity Error (Odd parity detected)
DO4	Transmit Check conditions (existence of errors on any or all of the following data bits: DO3, DO5, and DO6
DO5	An invalid ending sequence
DO6	Loss of mid-bit transition detected at other than normal ending sequence time
DO7	New starting sequence detected before data byte in holding register has been read
DO8	Receiver disabled during receiver active mode

Message Format

Single Byte Transmission
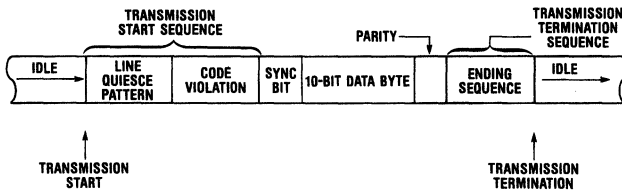
Multi-Byte Transmission
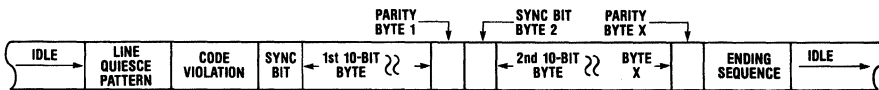
FIGURE 3. IBM 3270 Message Format

TL/F/5238-4

Message Format (Continued)
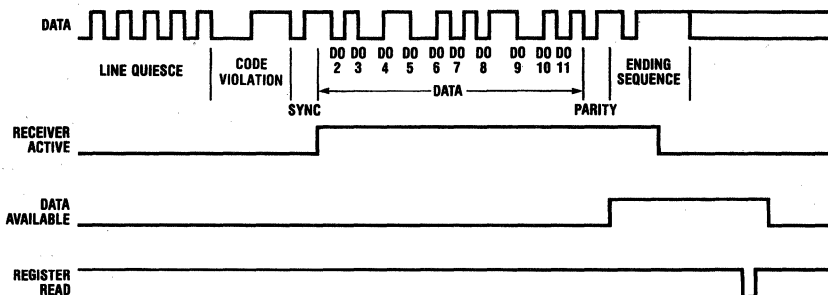
FIGURE 4a. Single Byte Message

TL/F/5238-5
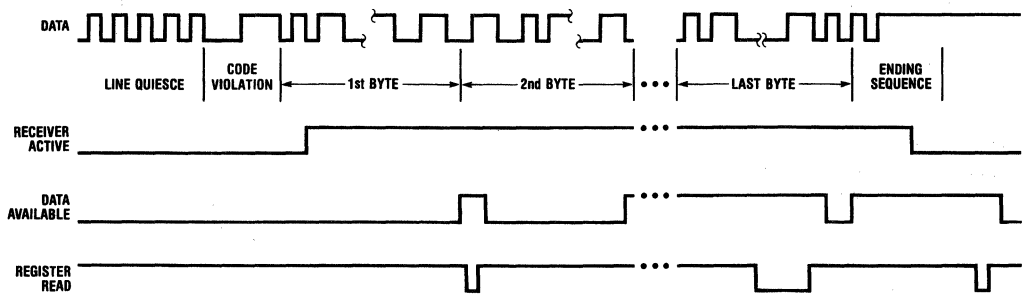
FIGURE 4b. Multi-Byte Message

TL/F/5238-6
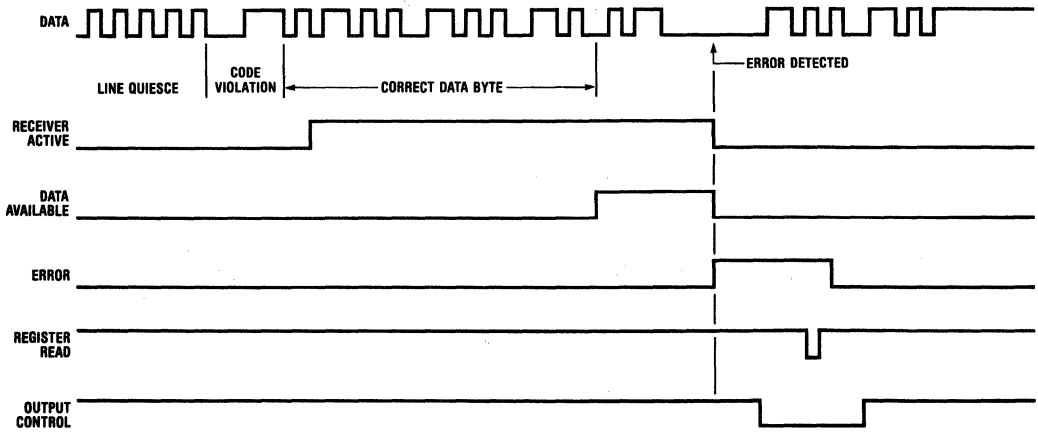
FIGURE 5. Message with Error

TL/F/5238-7

Absolute Maximum Ratings (Note 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage, V_{CC}	7V
Input Voltage	+5.5V
Output Voltage	5.25V
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 10 seconds)	300°C

Maximum Power Dissipation* at 25°C

Cavity Package	2040 mW
Dual-In-Line Package	2237 mW
Plastic Chip Carrier	1690 mW

*Derate cavity package 13.6 mW/°C above 25°C; derate PCC package 13.5 mW/°C above 25°C; derate Dual-In-Line package 17.9 mW/°C above 25°C.

Operating Conditions

	Min	Max	Units
Supply Voltage, (V_{CC})	4.75	5.25	V
Ambient Temperature, (T_A)	0	+70	°C

Electrical Characteristics (Notes 2, 3, and 5)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{IH}	Input High Level		2.0			V
V_{IL}	Input Low Level				0.8	V
V_{IH}–V_{IL}	Data Input Hysteresis (TTL, Pin 4)		2.0	0.4		V
V_{CLAMP}	Input Clamp Voltage	$I_{IN} = -12$ mA		−0.8	−1.2	V
I_{IH}	Logic "1" Input Current	$V_{CC} = 5.25V$, $V_{IN} = 5.25V$		2	40	μA
I_{IL}	Logic "0" Input Current	$V_{CC} = 5.25V$, $V_{IN} = 0.5V$		−20	−250	μA
V_{OH}	Logic "1" Output Voltage	$I_{OH} = -100$ μA	3.2	3.9		V
		$I_{OH} = -1$ mA	2.5	3.2		V
V_{OL}	Logic "0" Output Voltage	$I_{OL} = 5$ mA		0.35	0.5	V
I_{OS}	Output Short Circuit Current	$V_{CC} = 5V$, $V_{OUT} = 0V$ (Note 4)	−10	−20	−100	mA
I_{OZ}	TRI-STATE Output Current	$V_{CC} = 5.25V$, $V_O = 2.5V$	−40	1	+40	μA
		$V_{CC} = 5.25V$, $V_O = 0.5V$	−40	−5	+40	μA
A_{HYS}	Amplifier Input Hysteresis		5	20	30	mV
I_{CC}	Power Supply Current	$V_{CC} = 5.25V$		160	250	mA

Timing Characteristics (Notes 2, 6, 7, and 8)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
T_{D1}	Output Data to Data Available Positive Edge		5	20	40	ns
T_{D2}	Register Read Positive Edge to Data Available Negative Edge		10	25	45	ns
T_{D3}	Error Positive Edge to Data Available Negative Edge		10	30	50	ns
T_{D4}	Error Positive Edge to Receiver Active Negative Edge		5	20	40	ns
T_{D5}	Register Read Positive Edge to Error Negative Edge		20	45	75	ns
T_{D6}	Delay from Output Control to Error Bits from Data Bits		5	20	50	ns
T_{D7}	Delay from Output Control to Data Bits from Error Bits		5	20	50	ns
T_{D8}	First Sync Bit Positive Edge to Receiver Active Positive Edge			$3.5 \times T$ +70		ns

2

Timing Characteristics (Notes 2, 6, 7, and 8) (Continued)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
T_{D9}	Receiver Active Positive Edge to First Data Available Positive Edge			$92 \times T$		ns
T_{D10}	Negative Edge of Ending Sequence to Receiver Active Negative Edge			$11.5 \times T$ $+ 50$		ns
t_{D11}	Data Control Set-Up Multiplexer Time Prior to Receiving Data through Selected Input		40	30		ns
T_{PW1}	Register Read (Data) Pulse Width		40	30		ns
T_{PW2}	Register Read (Error) Pulse Width		40	30		ns
T_{PW3}	Data Available Logic "0" State between Data Bytes		25	45		ns
T_S	Output Control Set-Up Time Prior to Register Read Negative Edge		0	-5		ns
T_H	Output Control Hold Time After the Register Read Positive Edge		0	-5		ns
T_{ZE}	Delay from Output Enable to Logic "1" or Logic "0" from High Impedance State	Load Circuit 2		25	35	ns
T_{EZ}	Delay from Output Enable to High Impedance State from Logic "1" or Logic "0"	Load Circuit 2		25	35	ns
F_{MAX}	Data Bit Frequency (Clock Input must be $8 \times$ the Data Bit Frequency)	(Note 9)	DC		3.5	MBits/s

Note 1: "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

Note 2: Unless otherwise specified, min./max. limits apply across the 0°C to +70°C temperature range and the 4.75V to 5.25V power supply range. All typical values are for $T_A = 25°C$ and $V_{CC} = 5.0V$.

Note 3: All currents into device pins are shown as positive; all currents out of device pins are shown as negative; all voltages are referenced to ground, unless otherwise specified. All values shown as max. or min. are so classified on absolute value basis.

Note 4: Only one output at a time should be shorted.

Note 5: Input characteristics do not apply to amplifier inputs (pins 2 and 3).

Note 6: Unless otherwise specified, all AC measurements are referenced to the 1.5V level of the input to the 1.5V level of the output and load circuit 1 is used.

Note 7: AC tests are done with input pulses supplied by generators having the following characteristics: $Z_{OUT} = 50\Omega$ and $T_r \leq 5$ ns, $T_f \leq 5$ ns.

Note 8: $T = 1/$(clock input frequency), units for "T" should be ns.

Note 9: 28 MHz clock frequency corresponds to 3.75% jitter when referenced to *Figure 10*.

Load Circuit 1   Load Circuit 2
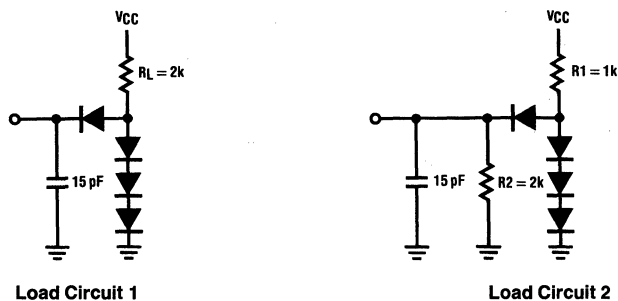
TL/F/5238-8

FIGURE 6. Test Load Circuits

Timing Waveforms

OUTPUT ENABLE
1.5V
1.5V
T_{ZE}
T_{EZ}

DO$_2$-DO$_{11}$ OUTPUTS (OUTPUT CONTROL = HI)
$(Z + 0.5)V$
$(Z - 0.5)V$
$(V_{OH} - 0.5)V$
$(V_{OH} + 0.5)V$

T_{D1}

DATA AVAILABLE

t_{pw1}
T_{D2}

REGISTER READ
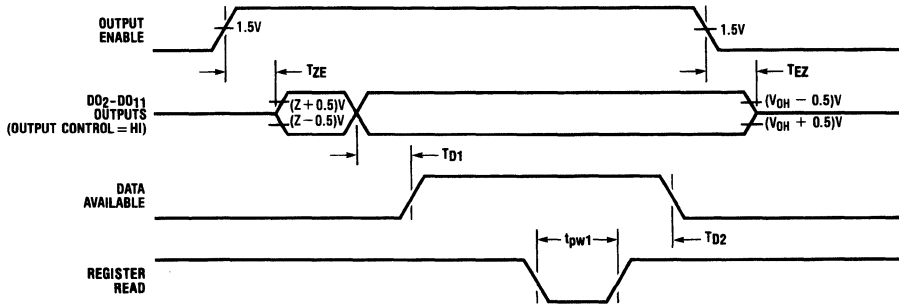
TL/F/5238–9

FIGURE 7. Data Sequence Timing

DATA AVAILABLE

T_{D3}

RECEIVER ACTIVE

T_{D4}

ERROR

T_{D5}

REGISTER READ

T_S
T_{pw2}
T_H

OUTPUT CONTROL

T_{D6}
T_{D7}

DO$_2$-DO$_8$
DATA BITS
ERROR BITS
DATA BITS
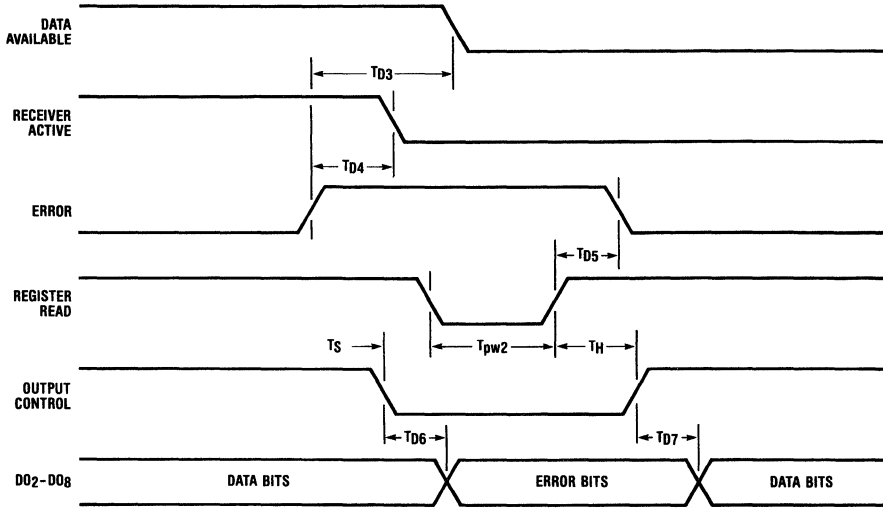
TL/F/5238–10

FIGURE 8. Error Sequence Timing

1 1 1 CODE VIOLATION 1 DATA 0 MCV MCV

T_{D8}
T_{D10}

RECEIVER ACTIVE

T_{D9}

DATA AVAILABLE

TL/F/5238–11

FIGURE 9. Message Timing

2

Timing Waveforms (Continued)

$$8T \pm (T - 25 \text{ ns})$$

$$4T \pm (T - 25 \text{ ns})$$

$$T = \frac{1}{\text{CLOCK INPUT FREQUENCY}}$$

1.3V MAX. 40 mV MIN. V_{IN+}

V_{IN-}

−40 mV MIN. −1.3V MAX. V_{IN+}
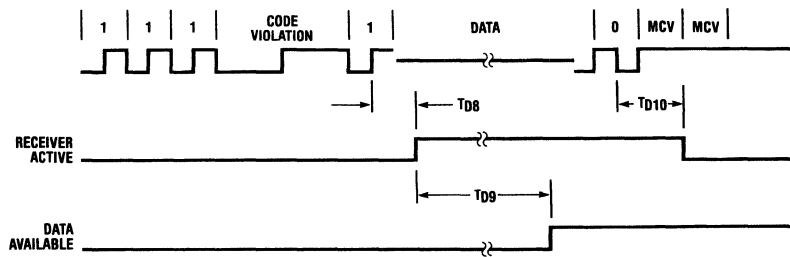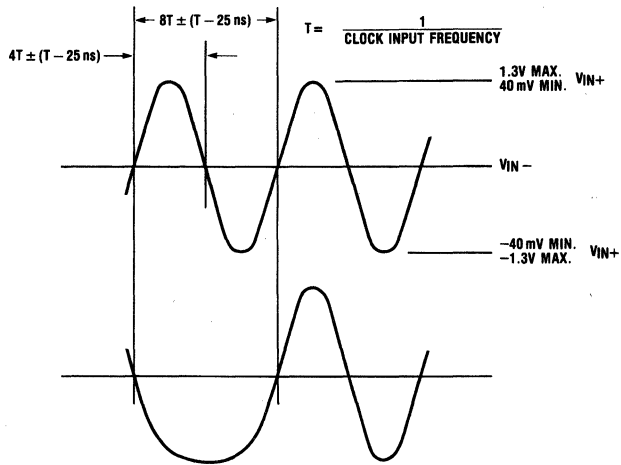
TL/F/5238−12

FIGURE 10. Data Waveform Constraints: Amplifier Inputs

$$8T \pm (T - 25 \text{ ns})$$

$$4T \pm (T - 25 \text{ ns})$$

$$T = \frac{1}{\text{CLOCK INPUT FREQUENCY}}$$
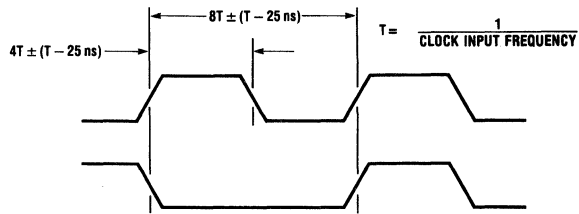
Note: $|T_r - T_f| \leq 10$ ns

TL/F/5238−13

FIGURE 11. Data Waveform Constraints: Data Input (TTL)

Typical Applications
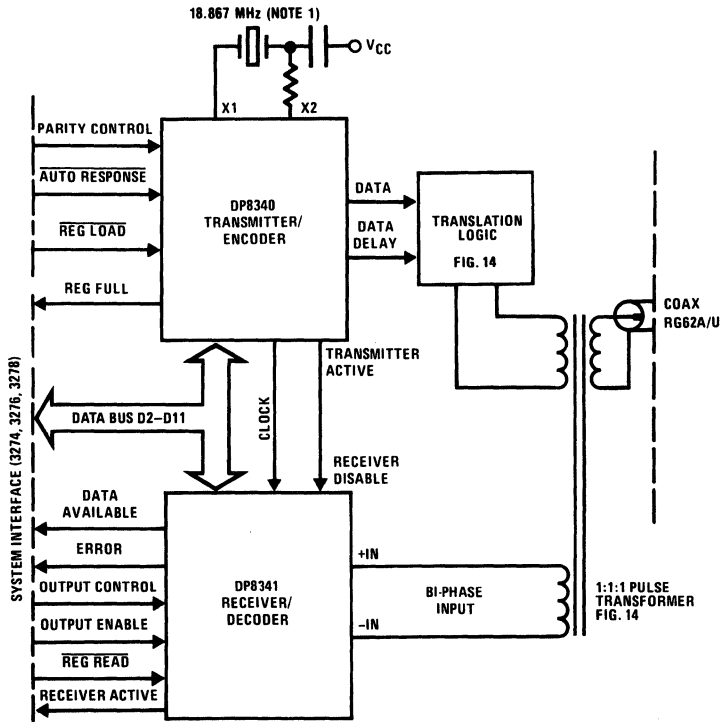
TL/F/5238-14

Note 3: Crystal manufacturers: Midland Ross Corp.

NEL Unit Part No. NE18A (C2560N) @ 18.867 MHz

The Viking Group Part No. VXB-46NS @ 18,867 MHz. Located in San Jose, CA.

FIGURE 12. Typical Application for IBM 3270 Interface
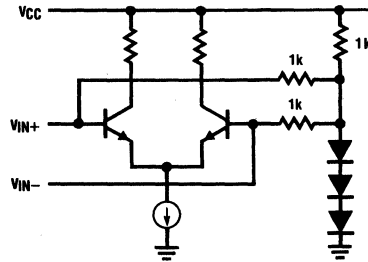
TL/F/5238-15

FIGURE 13. Equivalent Circuit for DP8341/NS32441 Input Amplifier

Typical Applications (Continued)
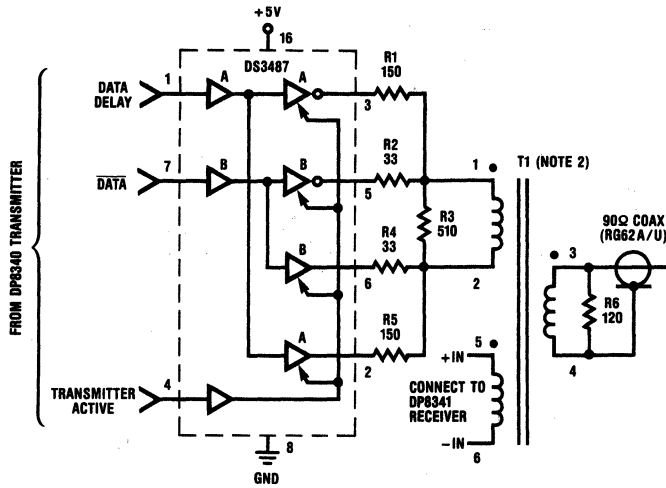
Note 1: Resistance values are in Ω, ±5%, ¼W

Note 2: T1 is a 1:1:1 pulse transformer, L_{MIN} = 500 μH for 18 MHz system clock
Pulse Engineering Part No. 5762/Surface Mount, 5762M/PE-85762
Valor Electronics Part No. CT1501
Technitrol Part No. 11LHA or equivalent transformers

TL/F/5238–16

FIGURE 14. Translation Logic
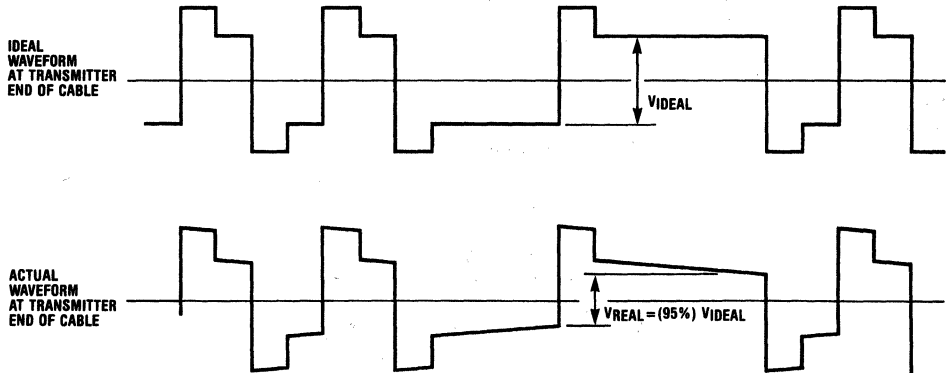
TL/F/5238–17

*To maintain loss at 95% of ideal signal, select transformer inductance such that:

$$L_{(MIN)} = \frac{10,000}{f_{CLK}}$$

f_{CLK} = System Clock Frequency (e.g., 18.87 MHz)

EXAMPLE:

$$L = \frac{10,000}{18.87 \times 10^6} \rightarrow L(MIN) = 530 \mu H$$

Note 1: Less inductance will cause greater amplitude attenuation

Note 2: Greater inductance may decrease signal rise time slightly and increase ringing, but these effects are generally negligible.

FIGURE 15. Transformer Selection

![National Semiconductor logo] **National Semiconductor**

DP8342/NS32442
High-Speed 8-Bit Serial Transmitter/Encoder

General Description

The DP8342/NS32442 generates a complete encoding of parallel data for high speed serial transmission. It generates a five bit starting sequence, three bit code violation, followed by a syn bit and eight bit per byte of data plus a parity bit. A three-bit ending code signals the termination of the transmission. The DP8342/NS32442 adapts to generalized high speed serial data transmission as well as the coax lines at a maximum data rate of 3.5 MHz.

The DP8342/NS32442 and its complementary chip, the DP8343 (receiver/decoder) have been designed to provide maximum flexibility in system designs. The separation of the transmitter receiver functions provides convenient addition of more receivers at one end of a biphase line without the need of unused transmitters. This is specifically advantageous in control units where typical biphase data is multiplexed over many biphase lines and the number of receivers generally exceeds the number of transmitters.

Features

- Eight bits per data byte transmission
- Single-byte or multi-byte transmission
- Internal parity generation (even or odd)
- Internal crystal controlled oscillator used for the generation of all required chip timing frequencies
- Clock output directly drives receiver (DP8343) clock input
- Input data hold register
- Automatic clear status response feature
- Line drivers at data outputs provide easy interface to bi-phase coax line or general transmission media
- <2 ns driver output skew
- Bipolar technology provides TTL input/output compatibility
- Data outputs power up/down glitch free
- Internal power up clear and reset
- Single +5V power supply

Connection Diagram

Dual-In-Line Package

Pin	Signal		Pin	Signal
1	OUTPUT ENABLE		24	VCC
2	BYTE CLK		23	REG LOAD
3	BIT 8		22	REG FULL
4	BIT 7		21	AUTO RESPONSE
5	BIT 6		20	TRANSMITTER ACTIVE
6	BIT 5		19	RESET
7	BIT 4		18	EVEN/ODD
8	BIT 3		17	DATA OUT
9	BIT 2		16	DATA OUT
10	BIT 1		15	DATA DELAY
11	CLK OUT		14	X2
12	GND		13	X1
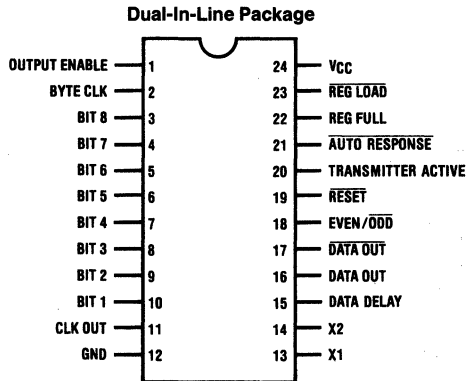
TL/F/5236–1

FIGURE 1

**Order Number DP8342J, NS32442J
or DP8342J, NS32442N
See NS Package Number J24A or N24A**
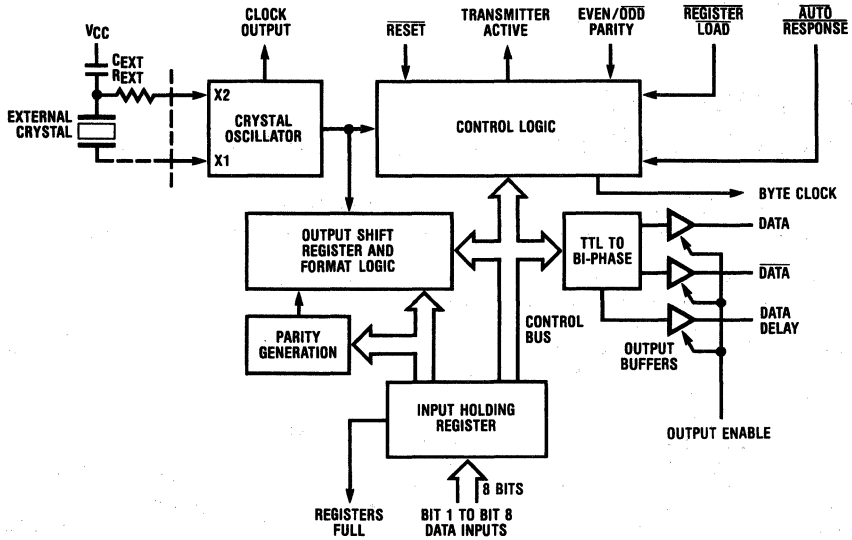
Block Diagram

FIGURE 2

TL/F/5236–2

Functional Description

Figure 2 is a block diagram of the DP8342/NS32442 Biphase Transmitter/Encoder. The transmitter/encoder contains a crystal oscillator whose input is a crystal with a frequency eight (8) times the data rate. A Clock Output is provided to drive the DP8342/NS32442 receiver/decoder Clock Input and other system components at the oscillator frequency. Additionally, the oscillator drives the control logic and output shift register/format logic blocks.

Data is parallel loaded from the system data bus to the transmitter/encoder's input holding register. This data is in turn loaded by the transmitter/encoder to its output shift register if this register was empty at the time of the load. During this load, message formatting and parity are generated. The formatted message is then shifted out at the bit rate frequency to the TTL to Biphase block which generates the proper data bit formatting. The data outputs, DATA, $\overline{\text{DATA}}$, and DATA DELAY provide for flexible interface to the transmission medium with little or no external components.

The control Logic block interfaces to all blocks to insure proper chip operation and sequencing. It controls the type of parity generation through the Even/$\overline{\text{Odd}}$ Parity input. An additional feature provided by the transmitter/encoder is

the Reset and Output-TRI-STATE® capability. Another feature of the DP8342/NS32442 is the Byte Clock output which keeps track of the number of bytes transferred.

The transmitter/encoder is also capable of internal TT/AR (Transmission Turnaround/Auto Response). When the Auto-Response ($\overline{\text{AR}}$) input is forced to the logic "0" state, the transmitter/encoder responds with clean status (all zeros on data bits).

Operation of the transmitter/encoder is automatic. After the first data byte is loaded, the Transmitter Active output is set and the transmitter/encoder immediately formats the input data and serially shifts it out its data outputs. If the message is a mutli-byte message, the internal format logic will modify the message data format for multibyte as long as the next byte is loaded to the input holding format logic will modify the message data format for multibyte as long as the next byte is loaded to the input holding register before the last data bit of the previous data byte is transferred out of the internal output shift register. After all data is shifted out the transmitter/encoder the Transmitter Active output will return to the inactive state.

Detailed Pin/Functional Description

CRYSTAL INPUTS X1 AND X2

The oscillator is controlled by an external, parallel resonant crystal connected between the X1 and X2 pins. Normally, a fundamental mode crystal is used to determine the operating frequency of the oscillator; however, over-tone mode crystals may be used.

CRYSTAL SPECIFICATIONS (PARALLEL RESONANT)

Type	<20 MHz AT-cut or > 20 MHz BT-cut
Tolerance	0.005% at 25°C
Stability	0.01% from 0°C to +70°C
Resonance	Fundamental (Parallel)
Maximum Series Resistance	Dependent on Frequency (For 20 MHz, 50Ω)
Load Capacitance	15 pF

Connection Diagram

TO PIN 22 PIN (14)   R   C   Vcc

CRYSTAL (FIG. 18)

TO PIN X1 PIN (13)
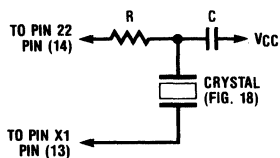
TL/F/5236–3

Freq	R	C
10 MHz–20 MHz	500Ω	30 pF
>20 MHz	120Ω	15 pF

If the DP8342/NS32442 transmitter is clocked by a system clock (crystal oscillator not used), pin 13 (X1 input) should be clock directly using a Schottky series (74S) circuit. Pin 14 (X2 input) may be left open. The clocking frequency must be set at eight times the data bit rate. Maximum input frequency is 28 MHz.

CLOCK OUTPUT

The Clock Output is a buffered output derived directly from the crystal oscillator block and clocks at the oscillator frequency. It is designed to directly drive the DP8343 receiver/decoder Clock Input as well as other system components.

REGISTERS FULL

This output is used as a flag by the external operating system. A logic "1" (active state) on this output indicates that both the internal output shift register and the input holding register contain active data. No additional data should be loaded until this output returns to the logic "0" state (inactive state).

TRANSMITTER ACTIVE

This output will be in the logic "1" state while the transmitter/encoder is about to transmit or is in the process of transmitting data. Otherwise, it will assume the logic "0" state indicating no data presently in either the input holding or output shift registers.

REGISTER LOAD

The Register Load input is used to load data from the Data Inputs to the input holding register. The loading function is level sensitive, the data present during the logic "0" state of this input is loaded, and the input data must be valid before the logic "0" to logic "1" transition. It is after this transition that the transmitter/encoder begins formatting of data for serial transmission.

AUTO RESPONSE (TT/AR)

This input provides for automatic clear data transmission (all bits in logic "0") without the need of loading all zero's. When a logic "0" is forced on this input the transmitter/encoder immediately responds with transmission of "clean status". When this input is in the logic "1" state the transmitter/encoder transmits data entered on the Data Inputs.

EVEN/ODD PARITY

This input sets the internal logic of the DP8342/NS32442 transmitter/encoder to generate either even or odd parity for the data byte in the bit 10 position. When this pin is in the logic "0" state odd parity is generated. In the logic "1" state even parity is generated. This feature is useful when the control unit is performing a loop back check and at the same time the controller wishes to verify proper data transmission with its receiver/decoder.

SERIAL OUTPUTS—DATA, DATA, AND DATA DELAY

These three output pins provide for convenient application of data to the Bi-Phase transmission line. The Data outputs are a direct bit representation of the Biphase data while the Data Delay output provides the necessary increment to clearly define the four (4) DC levels of the pulse. The DATA and DATA outputs add flexibility to the DP8342/NS32442 transmitter/encoder for use in high speed differential line driving applications. The typical DATA to DATA skew is 2 ns.

RESET

When a logic "0" is forced on this input, all outputs except Clock Output are latched low.

OUTPUT ENABLE

When a logic "0" is forced on this input the three serial data outputs are in the high impedence state.

BYTE CLOCK

This pin registers a pulse at the end of each byte transmission. The number of pulses registered corresponds to the number of bytes transmitted.

2

Message Format

Single Byte Transmission
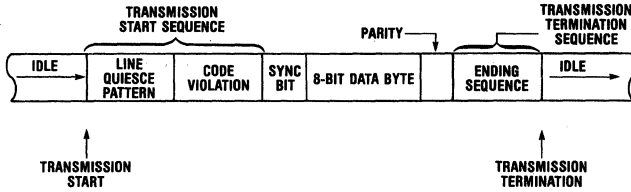
Multi-Byte Transmission
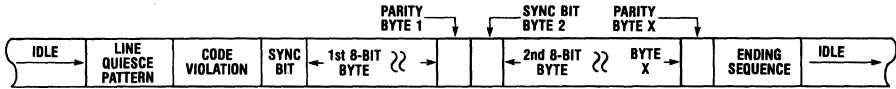
FIGURE 3

TL/F/5236-4

Functional Timing Waveforms
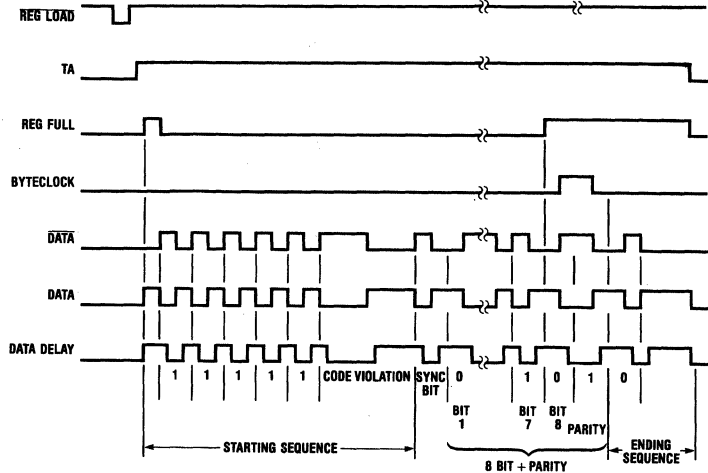
FIGURE 4. Overall Timing Waveforms for Single Byte

TL/F/5236-5
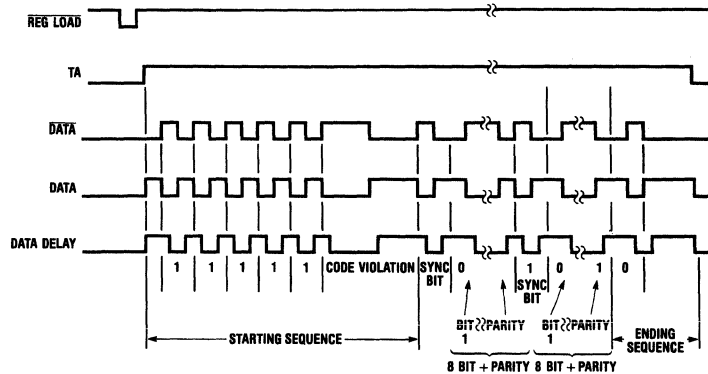
FIGURE 5. Overall Timing Waveforms for Multi-Byte

TL/F/5236-6

Absolute Maximum Ratings (Note 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage, V_{CC}	7V
Input Voltage	5.5V
Output Voltage	5.25V
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C

Maximum Power Dissipation* at 25°C
Cavity Package	2237 mW
Dual-In-Line package	2500 mW

*Derate cavity package 14.9 mW/°C above 25°C; derate dual in line package 20 mW/°C above 25°C.

Operating Conditions

	Min	Max	Units
Supply Voltage, (V_{CC})	4.75	5.25	V
Ambient Temperature, T_A	0	+70	°C

Electrical Characteristics (Notes 2 and 3)

Symbol	Parameter		Conditions	Min	Typ	Max	Units
V_{IH}	Logic "1" Input Voltage (All Inputs Except X1 and X2)		$V_{CC} = 5V$	2.0			V
V_{IL}	Logic "0" Input Voltage (All Inputs Except X1 and X2)		$V_{CC} = 5V$			0.8	V
V_{CLAMP}	Input Clamp Voltage (All Inputs Except X1 and X2)		$I_{IN} = -12$ mA		−0.8	−1.2	V
I_{IH}	Logic "1" Input Current	Register Load Input	$V_{CC} = 5.25V$ $V_{IN} = 5.25V$		0.3	120	µA
		All Others Except X1 and X2			0.1	40	µA
I_{IL}	Logic "0" Input Current	Register Load Input	$V_{CC} = 5.25V$ $V_{IN} = 0.5V$		−15	−300	µA
		All Inputs Except X1 and X2			−5	−100	µA
V_{OH1}	Logic "1" All Outputs Except CLK OUT, DATA, $\overline{DATA}$, and DATA DELAY		$I_{OH} = -100$ µA $V_{CC} = 4.75V$	3.2	3.9		V
			$I_{OH} = -1$ mA	2.5	3.4		V
V_{OH2}	Logic "1" for CLK OUT, DATA, $\overline{DATA}$, and DATA DELAY Outputs		$V_{CC} = 4.75V$ $I_{OH} = -10$ mA	2.6	3.0		V
V_{OL1}	Logic "0" All Outputs Except CLK OUT, DATA, $\overline{DATA}$, and DATA DELAY		$V_{CC} = 4.75V$ $I_{OL} = 5$ mA		0.35	0.5	V
V_{OL2}	Logic "0" for CLK OUT, DATA $\overline{DATA}$, and DATA DELAY Outputs		$V_{CC} = 4.75V$ $I_{OL} = 20$ mA		0.4	0.6	V
I_{OS1}	Output Short Circuit Current for All Except CLK OUT, DATA, $\overline{DATA}$, and DATA DELAY Outputs		(Note 5) $V_{OUT} = 0V$	−10	−30	−100	mA
I_{OS2}	Output Short Circuit Current DATA, $\overline{DATA}$, and DATA DELAY Outputs		(Note 5) $V_{OUT} = 0V$	−50	−140	−350	mA
I_{OS3}	Output Short Circuit Current for CLK OUT		(Note 5) $V_{OUT} = 0V$	−30	−90	−200	mA
I_{CC}	Power Supply Current		$V_{CC} = 5.25V$		170	250	mA

Timing Characteristics $V_{CC} = 5V \pm 5\%$, $T_A = 0°C$ to 70°C, Oscillator Frequency = 28 MHz (Notes 2 and 3)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
t_{pd1}	$\overline{REG\ LOAD}$ to Transmitter Active (TA) Positive Edge	Load Circuit 1 Figure 6		60	90	ns
t_{pd2}	$\overline{REG\ LOAD}$ to Register Full; Positive Edge	Load Circuit 1 Figure 6		45	75	ns
t_{pd3}	TA to Register Full; Negative Edge	Load Circuit 1 Figure 6		40	70	ns
t_{pd4}	Positive Edge of $\overline{REG\ LOAD}$ to Positive Edge of DATA	Load Circuit 2 Figure 9		50	80	ns
t_{pd5}	$\overline{REG\ LOAD}$ to $\overline{DATA}$; Positive Edge	Load Circuit 2 Figure 9		280	380	ns
t_{pd6}	$\overline{REG\ LOAD}$ to DATA DELAY; Positive Edge	Load Circuit 2 Figure 9		150	240	ns

2

Timing Characteristics (Continued)

$V_{CC} = 5V \pm 5\%$, $T_A = 0°C$ to $70°C$, Oscillator Frequency $= 28$ MHz (Notes 2 and 3)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
t_{pd7}	Positive Edge of $\overline{DATA}$ to Negative Edge of DATA DELAY	Load Circuit 2 *Figure 9*		70	85	ns
t_{pd8}	Positive Edge of DATA DELAY to Negative Edge of $\overline{DATA}$	Load Circuit 2 *Figure 9*		80	95	ns
t_{pd9}, t_{pd10}	Skew between DATA and $\overline{DATA}$	Load Circuit 2 *Figure 9*		2	6	ns
t_{pd11}	Negative Edge of Auto Response $(\overline{AR})$ to Positive Edge of TA	Load Circuit 1 *Figure 10*		70	100	ns
t_{pd12}	Maximum Time Delay to Load Second Byte after Positive Edge of REG FULL	Load Circuit 1 *Figure 8*, (Note 7)			$4 \times T - 50$	ns
t_{pd13}	X1 to CLK OUT; Positive Edge	Load Circuit 2 *Figure 11*		21	30	ns
t_{pd14}	X1 to CLK OUT; Negative Edge	Load Circuit 2 *Figure 11*		23	33	ns
t_{pd15}	Negative Edge of $\overline{AR}$ to Positive Edge of REG FULL	Load Circuit 1 *Figure 10*		45	75	ns
t_{pd16}	Skew between TA and REG FULL during $\overline{\text{Auto Response}}$	Load Circuit 1 *Figure 10*		50	80	ns
t_{pd17}	$\overline{\text{REG LOAD}}$ to REG FULL; Positive Edge for Second Byte	Load Circuit 1 *Figure 7*		45	75	ns
t_{pd18}	REG FULL to BYTE CLK; Negative Edge	Load Circuit 1 *Figure 7*		60	90	ns
t_{pd19}	REG FULL to BYTE CLK; Positive Edge	Load Circuit 1 *Figure 7*		145	180	ns
t_{ZH}	Output Enable to DATA, $\overline{DATA}$, or DATA DELAY outputs: HiZ to High	$C_L = 50$ pF *Figures 16, 17*		25	45	ns
t_{ZL}	Output Enable to DATA, $\overline{DATA}$, or DATA DELAY Outputs; HiZ to High	$C_L = 50$ pF *Figures 16, 17*		15	30	ns
t_{HZ}	Output Enable to DATA, $\overline{DATA}$, or DATA DELAY Outputs; High to HiZ	$C_L = 15$ pF *Figures 16, 17*		65	100	ns
t_{LZ}	Output Enable to DATA, $\overline{DATA}$, or DATA DELAY Outputs; Low to HiZ	$C_L = 15$ pF *Figures 16, 17*		45	70	ns
t_{pw1}	$\overline{\text{REG LOAD}}$ Pulse Width	*Figure 12*	40			ns
t_{pw2}	First REG FULL Pulse Width (Note 6)	Load Circuit 1 *Figure 7*, (Note 7)		$8 \times T + 60$	$8 \times T + 100$	ns
t_{pw3}	REG FULL Pulse Width Prior to Ending Sequence (Note 6)	Load Circuit 1 *Figure 7*		$5 \times B$		ns
t_{pw4}	Pulse Width for $\overline{\text{Auto Response}}$	*Figure 10*	40			ns
t_{pu5}	Pulse Width for BYTE CLK	Load Circuit 1 *Figure 7*, (Note 7)		$8 \times T + 30$	$8 \times T + 80$	ns
t_s	Data Setup Time prior to $\overline{\text{REG LOAD}}$ Positive Edge; Hold Time = 0 ns	*Figure 12*		15	23	ns
t_{r1}	Rise Time for DATA, $\overline{DATA}$, and DATA DELAY Output Waveform	Load Circuit 2 *Figure 13*		7	13	ns
t_{f1}	Fall Time for DATA, $\overline{DATA}$, and DATA DELAY Output Waveform	Load Circuit 2 *Figure 13*		5	11	ns
t_{r2}	Rise Time for TA and REG FULL	Load Circuit 1 *Figure 14*		20	30	ns
t_{f2}	Fall Time for TA and REG FULL	Load Circuit 1 *Figure 14*		15	25	ns

Timing Characteristics (Continued)

$V_{CC} = 5V \pm 5\%$, $T_A = 0°C$ to $70°C$, Oscillator Frequency = 28 MHz (Notes 2 and 3)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
f_{MAX}	Data Rate Frequency (Clock Input must be $8\times$ this Frequency)		DC		3.5	Mbits/s
C_{IN}	Input Capacitance—Any Input	(Note 4)		5	15	pF

Note 1: "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

Note 2: Unless otherwise specified, min/max limits apply across the 0°C to +70°C temperature range and the 4.75V to 5.25V power supply range. All typical values are for $T_A = 25°C$ and $V_{CC} = 5.0V$.

Note 3: All currents into device pins are shown as positive; all currents out of device pins are shown as negative; all voltages are referenced to ground, unless otherwise specified. All values shown as max or min are so classified on absolute basis.

Note 4: Input capacitance is guaranteed by periodic testing. $f_{TEST} = 10$ kHz at 300 mV, $T_A = 25°C$.

Note 5: Only one output should be shorted at a time.

Note 6: T = 1/(Oscillator Frequency). Unit for T should be in ns. B = 8T.

Note 7: Oscillator Frequency Dependent.

Timing Waveforms (Continued)
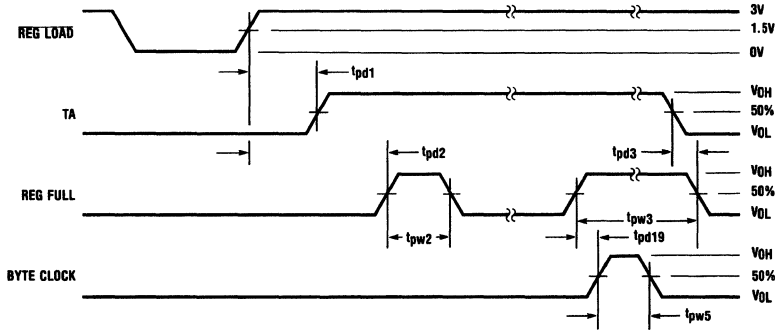
TL/F/5236–7

FIGURE 6. Single Byte Transfer
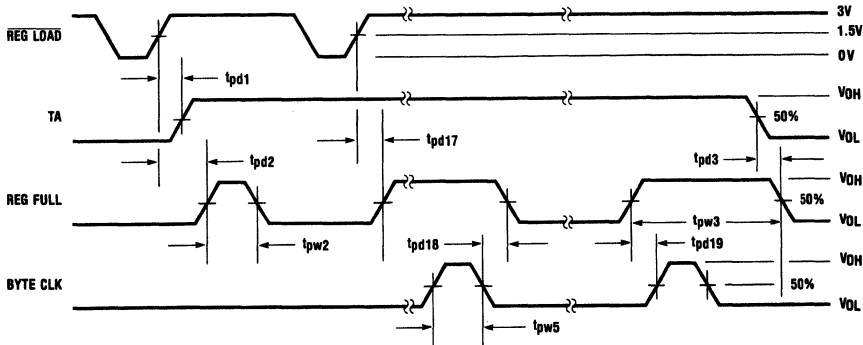
TL/F/5236–8

FIGURE 7. Two-Byte Transfer
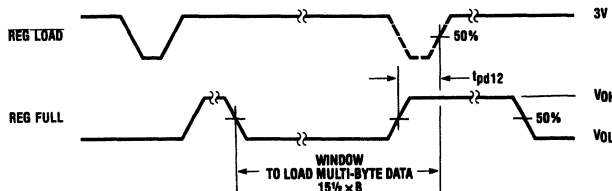
TL/F/5236–9

FIGURE 8. Maximum Window to Load Multi-Byte Data

Functional Timing Waveforms (Continued)
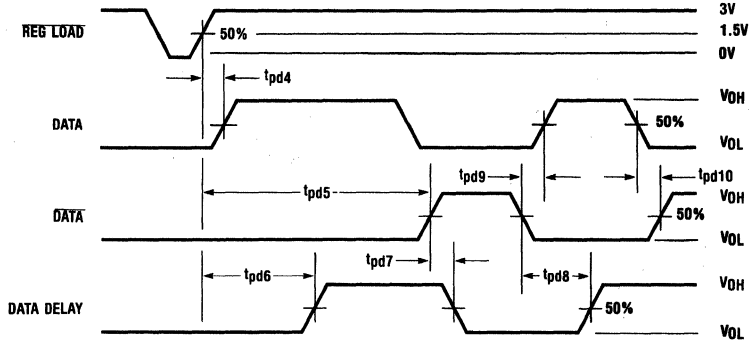
FIGURE 9. Three Serial Outputs

TL/F/5236–10
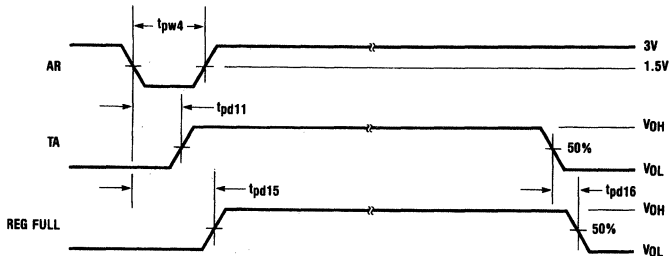
FIGURE 10. Auto-Response
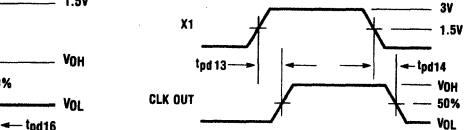
TL/F/5236–11

FIGURE 11. Clock Pulse

TL/F/5236–12
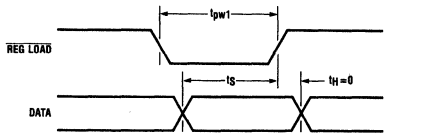
FIGURE 12. REG LOAD

TL/F/5236–13
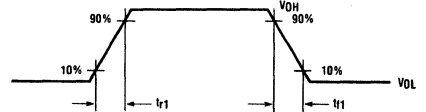
FIGURE 13. Output Waveform for DATA, DATA,
DATA DELAY (Load Circuit 2)

TL/F/5236–14
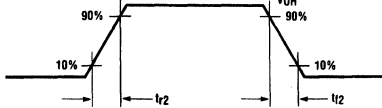
FIGURE 14. Rise and Fall Time Measurement
for TA and REG FULL

TL/F/5236–15
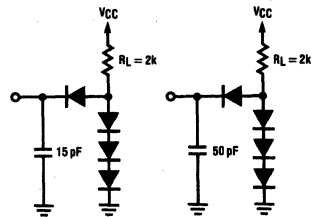
Load Circuit 1   Load Circuit 2
FIGURE 15. Test Load Circuits

TL/F/5236–16

Timing Waveforms (Continued)
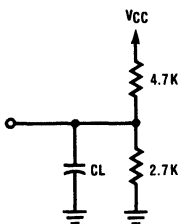
TL/F/5236–17

FIGURE 16. Load Circuit for Output TRI-STATE Test
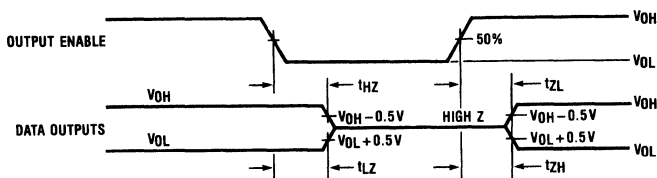
TL/F/5236–18

FIGURE 17. TRI-STATE Test

Typical Applications
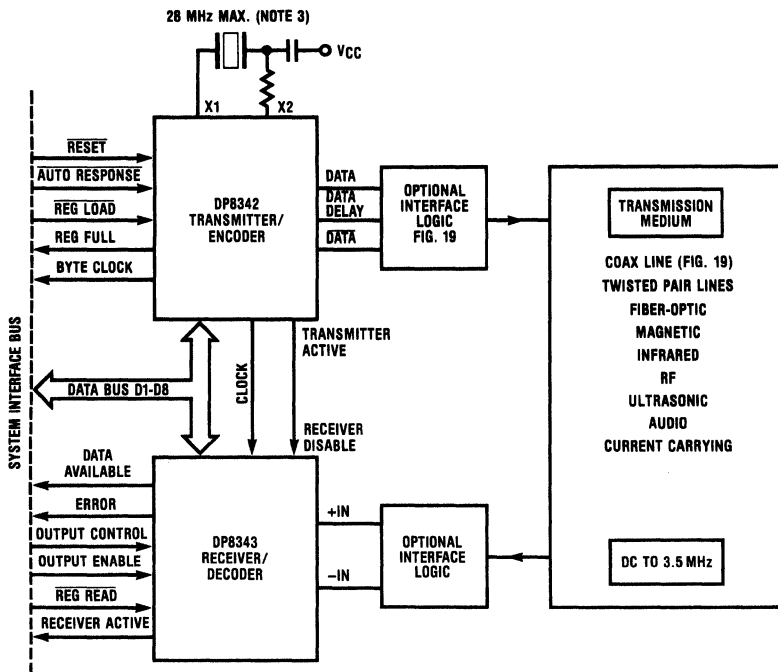
TL/F/5236–19

FIGURE 18

Typical Applications (Continued)
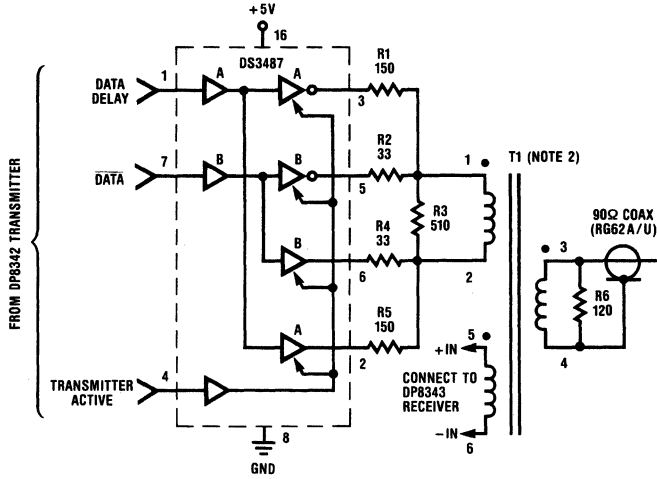
TL/F/5236–20

Note 1: Resistance values are in Ω, ±5%, ¼W.

Note 2: T1 is a 1:1:1 pulse transformer, L = 500 μH for 18 MHz to 28 MHz system clock. Pulse Engineering Part No. 5762; Technitrol Part No. 11LHA, Valor Electronics Part No. CT1501, or equivalent transformer.

Note 3: Crystal manufacturer Midland Ross Corp. NEL Unit Part No. NE-18A at 28 MHz.

FIGURE 19. Interface Logic for a Coax Transmission Line
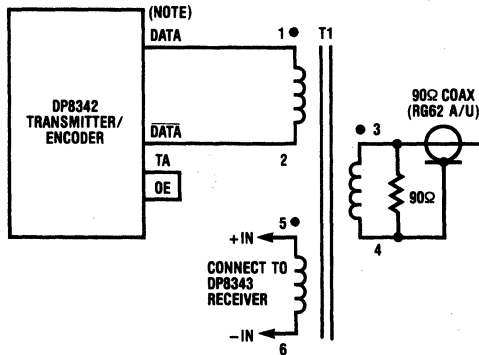
TL/F/5236–21

Note: Data rates up to 3.5 Mbits/s at 5000′ still apply.

FIGURE 20. Direct Interface for a Coax Transmission Line (Non-IBM Voltage Levels)

![National Semiconductor logo] **National Semiconductor**

DP8343/NS32443
High-Speed 8-Bit Serial Receiver/Decoder

General Description

The DP8343/NS32443 provides complete decoding of data for high speed serial data communications. In specific, the DP8343/NS32443 receiver recognizes biphase serial data sent from its complementary chip, the DP8342 transmitter, and converts it into 8 bits of parallel data. These devices are easily adapted to generalized high speed serial data transmission systems that operate at bit rates up to 3.5 MHz.

The DP8343/NS32443 receiver and the DP8342 transmitter are designed to provide maximum flexibility in system designs. The separation of transmitter and receiver functions allows addition of more receivers at one end of the biphase line without the necessity of adding unused transmitters. This is advantageous in control units where the data is typically multiplexed over many lines and the number of receivers generally exceeds the number of transmitters. The separation of transmitter and receiver function provides an additional advantage in flexibility of data bus organization. The data bus outputs of the receiver are TRI-STATE®, thus enabling the bus configuration to be organized as either a common transmit/receive (bi-directional) bus or as separate transmit and receive busses for higher speed.

Features

- DP8343/NS32443 receives 8-bit data bytes
- Separate receiver and transmitter provide maximum system design flexibility
- Even parity detection
- High sensitivity input on receiver easily interfaces to coax line
- Standard TTL data input on receiver provides generalized transmission line interface and also provides hysteresis
- Data holding register
- Multi-byte or single byte transfers
- TRI-STATE receiver date outputs provide flexibility for common or separated transmit/receive data bus operation
- Data transmission error detection on receiver provides for both error detection and error type definition
- Bipolar technology provides TTL input/output compatibility with excellent drive characteristics
- Single +5V power supply operation

Connection Diagram

Dual-In-Line Package

RECEIVER DISABLE	1	24 — VCC
+AMPLIFIER INPUT	2	23 — DATA CLOCK
−AMPLIFIER INPUT	3	22 — SERIAL DATA
DATA (TTL)	4	21 — BIT 8
DATA CONTROL	5	20 — BIT 7
CLOCK	6	19 — BIT 6
RECEIVER ACTIVE	7	18 — BIT 5
ERROR	8	17 — BIT 4
REGISTER READ	9	16 — BIT 3
DATA AVAILABLE	10	15 — BIT 2
OUTPUT CONTROL	11	14 — BIT 1
GND	12	13 — OUTPUT ENABLE
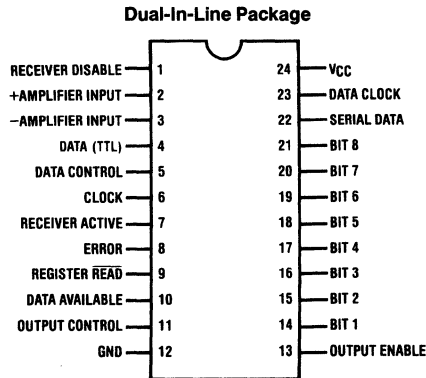
TL/F/5237–1

FIGURE 1
Order Number DP8343/NS32443J
or DP8343/NS32443N
See NS Package Number J24A or N24A
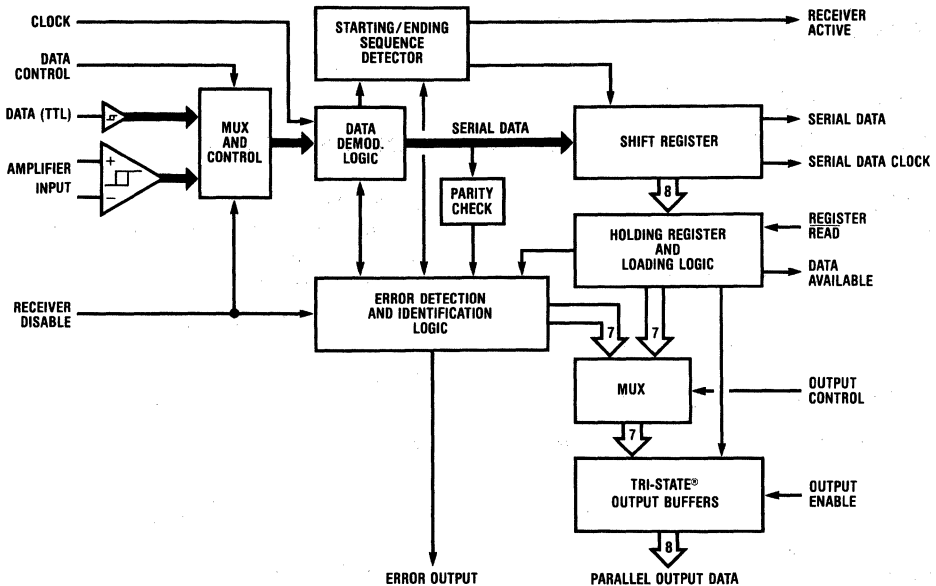
Block Diagram

FIGURE 2. DP8343/NS32443 Biphase Receiver

TL/F/5237–2

Functional Description

Figure 2 is a block diagram of the DP8343/NS32443 receiver. This chip is essentially a serial in/parallel out shift register. However, the serial input data must conform to a very specific format (see *Figures 3–6*). The message will not be recognized unless the format of the starting sequence is correct. Deviations from the format in the data, sync bit, parity or ending sequence will cause an error to be detected, terminating the message.

Data enters the receiver through the differential input amplifier or the TTL Data input. The differential amplifier is a high sensitivity input which may be used by connecting it directly to a transformer coupled coax line, or other transmission medium. The TTL Data input provides 400 mV of hysteresis and recognizes TTL logic levels. The data then enters the demodulation block.

The data demodulation block samples the data at eight (8) times the data rate and provides signals for detecting the starting sequence, ending sequence, and errors. Detection of the starting sequence sets the Receiver Active output high and enables the input shift register.

As the eight bits of data are shifted into the shift register, the receiver will verify that even parity is maintained on the data bits and the sync bit. Serial Data and Serial Data Clock, the inputs to the shift register, are provided for use with external error detecting schemes. After one complete data byte is received, the contents of the input shift register is parallel loaded to the holding register, assuming the holding register is empty, and the Data Available output is set. If the holding register is full, this load will be delayed until that register has

been read or the start of another data byte is received, in which case a Data Overflow Error will be detected, terminating the message. Data is read from the holding register through the TRI-STATE Output Buffers. The Output Enable input is the TRI-STATE control for these outputs and the Register Read input signals the receiver that the read has been completed.

When the receiver detects an ending sequence the Receiver Active output will be reset to a logic "0" indicating the message has been terminated. A message will also terminate when an error is detected. The Receiver Active output used in conjunction with the Error output allows quick response to the transmitting unit when an error free message has been received.

The Error Detection and Identification block insures that valid data reaches the outputs of the receiver. Detection of an error sets the Error output to a logic "1" and resets the Receiver Active output to a logic "0" terminating the message. The error type may be read from the data bus outputs by setting the Output Control input to logic "0" and enabling the TRI-STATE outputs. The data bit outputs have assigned error definitions (see error code definition table). The Error output will return to a logic "0" when the next starting sequence is received, or when the error is read (Output Control to logic "0" and a Register Read performed).

The Receiver Disable input is used to disable both the amplifier and TTL Data receiver inputs. It will typically be connected directly to the Transmitter Active output of the DP8342 transmitter circuit.

Detailed Functional Pin Description

RECEIVER DISABLE

This input is used to disable the receiver's data inputs. The Receiver Disable input will typically be connected to the Transmitter Active output of the DP8342. However, at the system controller it may be necessary for both the transmitter and receiver to be active at the same time. This variation can be accomplished with the addition of minimal external logic.

Truth Table

Receiver Disable	Data Inputs
Logic "0"	Active
Logic "1"	Disabled

AMPLIFIER INPUTS

The receiver has a differential input amplifier which may be directly connected to the transformer coupled coax line. The amplifier may also be connected to a differential type TTL line. The amplifier has 20 mV of hysteresis.

DATA INPUT

This input can be used either as an alternate data input or as a power-up check input. If the system designer prefers to use his own amplifier, instead of the one provided on the receiver, then this TTL input may be used. Using this pin as an alternate data input allows self-test of the peripheral system without disturbing the transmission line.

DATA CONTROL

This input is the control pin that selects which of the inputs are used for data entry to the receiver.

Truth Table

Data Control	Data Input To
Logic "0"	Data Input
Logic "1"	Amplifier Inputs

Note: This input is also used for testing. When the input voltage is raised to 7.5V the chip resets.

CLOCK INPUT

This input is the internal clock of the receiver. It must be set at eight (8) times the line data bit rate. The crystal-controlled oscillator provided in the DP8342 transmitter also operates at this frequency. The Clock Output of the transmitter is designed to directly drive the receiver's Clock Input. In addition, the receiver is designed to operate correctly to a data bit rate of 3.5 MHz.

RECEIVER ACTIVE

The purpose of this output is to inform the external system when the DP8343/NS32443 is in the process of receiving a message. This output will transition to a logic "1" state after a receipt of a valid starting sequence and transition to logic "0" when a valid ending sequence is received or an error is detected. This output combined with the Error output will inform the operating system of the end of an error free data transmission.

ERROR

The Error output transitions to a logic "1" when an error is detected. Detection of an error causes the Receiver Active and the Data Available outputs to transition to a logic "0". The Error output returns to a logic "0" after the error register has been read or when the next starting sequence is detected.

REGISTER READ

The Register Read input when driven to the logic "0" state signals the receiver that data in the holding register is being read by the external operating system. The data present in the holding register will continue to remain valid until the Register Read input returns to the logic "1" condition. At this time, if an additional byte is present in the input shift register it will be transferred to the holding register, otherwise the data will remain valid in the holding register. The Data Available output will be in the logic "0" state for a short interval while a new byte is transferred to the holding register after a register read.

DATA AVAILABLE

This output indicates the existence of a data byte within the output holding register. It may also indicate the presence of a data byte in both the holding register and the input shift register. This output will transition to the logic "1" state as soon as data is available and return to the logic "0" state after each data byte has been read. However, even after the last data byte has been read and the Data Available output has assumed the logic "0" state, the last data byte read from the holding register will remain until new data has been received.

OUTPUT CONTROL

The Output Control input determines the type of information appearing at the data outputs. In the logic "1" state data will appear, in the logic "0" state error codes are present.

Truth Table

Output Control	Data Outputs
Logic "0"	Error Codes
Logic "1"	Data

OUTPUT ENABLE

The Output Enable input controls the state of the TRI-STATE Data outputs.

Truth Table

Output Enable	TRI-STATE Data Outputs
Logic "0"	Disabled
Logic "1"	Active

DATA OUTPUTS

The DP8343/NS32443 has an 8-bit TRI-STATE data bus. Seven bits are multiplexed with error bits. The error bits are defined in the following table. The Output Control input is the multiplexer control for the Data/Error bits.

2

Message Format

Single Byte Transmission
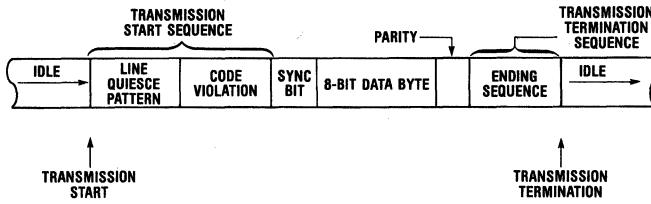
Multi-Byte Transmission
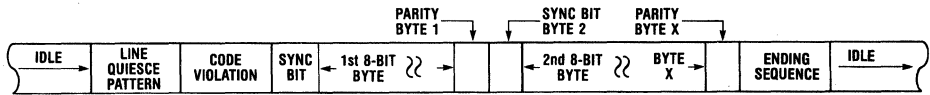
FIGURE 3

TL/F/5237-3
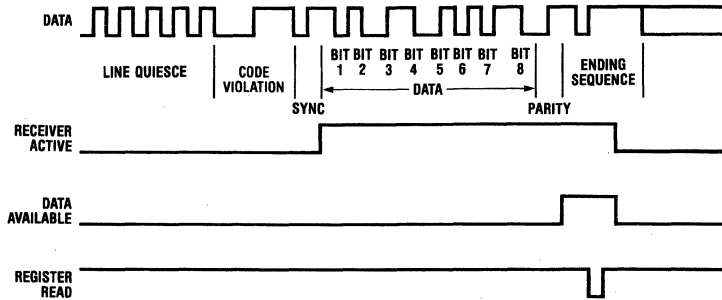
FIGURE 4a. Single Byte (8-Bit) Message

TL/F/5237-4
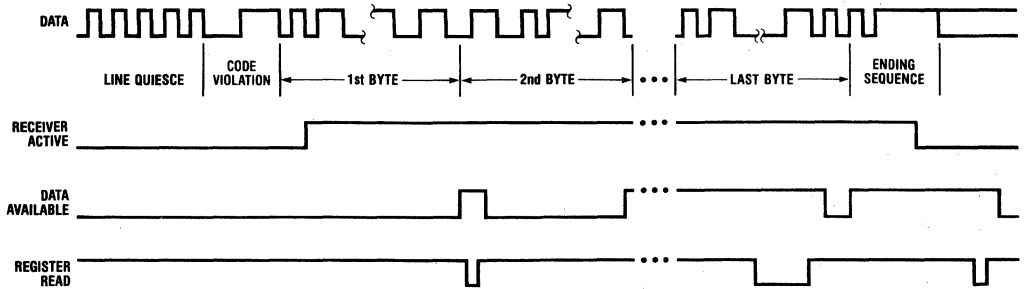
FIGURE 4b. Multi-Byte Message

TL/F/5237-5

Error Code Definition

Data Bit DP8343	Error Type
Bit 1	Data Overflow (Byte not removed from holding register when it and the input shift register are both full and new data is received)
Bit 2	Parity Error (Odd parity detected)
Bit 3	Transmit Check conditions (existence of errors on any or all of the following data bits: Bit 2, Bit 4, and Bit 5)
Bit 4	An invalid ending sequence
Bit 5	Loss of mid-bit transition detected at other than normal ending sequence time
Bit 6	New starting sequence detected before data byte in holding register has been read
Bit 7	Receiver disabled during receiver active mode

SERIAL DATA

The Serial Data output is the serial data coming into the input shift register.

DATA CLOCK

The Data Clock output is the clock to the input shift register.

Message Format (Continued)
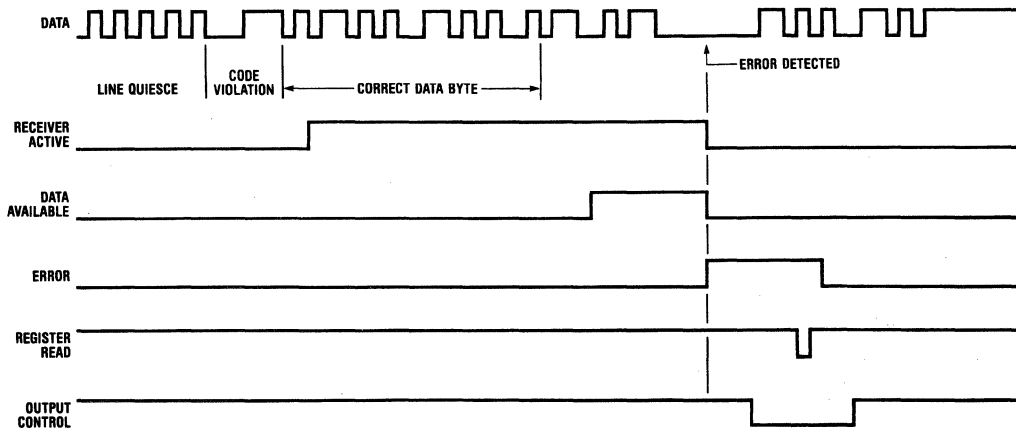
TL/F/5237-6

FIGURE 5. Message with Error

TL/F/5237-7
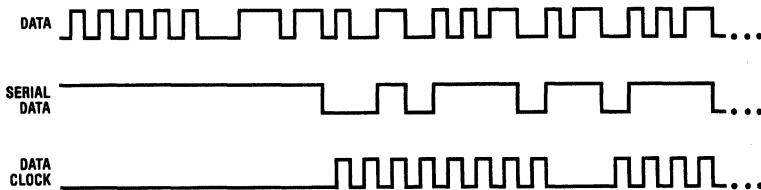
FIGURE 6. Data Clock and Serial Data

Absolute Maximum Ratings (Note 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage, (V_{CC})	7.0V
Input Voltage	5.5V
Output Voltage	5.25V

Storage Temperature Range	$-65°C$ to $+150°C$
Lead Temperature (Soldering, 10 sec.)	300°C

Operating Conditions

	Min	Max	Units
Supply Voltage, (V_{CC})	4.75	5.25	V
Ambient Temperature, T_A	0	$+70$	°C

Electrical Characteristics (Notes 2, 3 and 5)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{IH}	Input High Level		2.0			V
V_{IL}	Input Low Level				0.8	V
V_{IH}-V_{IL}	Data Input Hysteresis (TTL, Pin 4)		0.2	0.4		V
V_{CLAMP}	Input Clamp Voltage	$I_{IN} = -12$ mA		-0.8	-1.2	V
I_{IH}	Logic "1" Input Current	$V_{CC} = 5.25V$, $V_{IN} = 5.25V$		2	40	μA
I_{IL}	Logic "0" Input Current	$V_{CC} = 5.25V$, $V_{IN} = 0.5V$		-20	-250	μA
V_{OH}	Logic "1" Output Voltage	$I_{OH} = -100$ μA	3.2	3.9		V
		$I_{OH} = -1$ mA	2.5	3.2		V
V_{OL}	Logic "0" Output Voltage	$I_{OL} = 5$ mA		0.35	0.5	V
I_{OS}	Output Short Circuit Current	$V_{CC} = 5V$, $V_{OUT} = 0V$ (Note 4)	-10	-20	-100	mA
I_{OZ}	TRI-STATE Output Current	$V_{CC} = 5.25V$, $V_O = 2.5V$	-40	1	$+40$	μA
		$V_{CC} = 5.25V$, $V_O = 0.5V$	-40	-5	$+40$	μA
A_{HYS}	Amplifier Input Hysteresis		5	20	30	mV
I_{CC}	Power Supply Current	$V_{CC} = 5.25V$		160	250	mA

Timing Characteristics (Notes 2, 6, 7, and 8)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
T_{D1}	Output Data to Data Available Positive Edge		5	20	40	ns
T_{D2}	Register Read Positive Edge to Data Available Negative Edge		10	25	45	ns
T_{D3}	Error Positive Edge to Data Available Negative Edge		10	30	50	ns
T_{D4}	Error Positive Edge to Receiver Active Negative Edge		5	20	40	ns
T_{D5}	Register Read Positive Edge to Error Negative Edge		20	45	75	ns
T_{D6}	Delay from Output Control to Error Bits from Data Bits		5	20	50	ns
T_{D7}	Delay from Output Control to Data Bits from Error Bits		5	20	50	ns
T_{D8}	First Sync Bit Positive Edge to Receiver Active Positive Edge			$3.5 \times T$ $+70$		ns
T_{D9}	Receiver Active Positive Edge to First Data Available Positive Edge			$76 \times T$		ns
T_{D10}	Negative Edge of Ending Sequence to Receiver Active Negative Edge			$11.5 \times T$ $+50$		ns
T_{D11}	Data Control Set-up Multiplexer Time Prior to Receiving Data through Selected Input		40	30		ns
T_{D12}	Serial Data Set-Up Prior to Data Clock Positive Edge			$3 \times T$		ns

Timing Characteristics (Notes 2, 6, 7, and 8) (Continued)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
T_{PW1}	Register Read (Data) Pulse Width		30	40		ns
T_{PW2}	Register Read (Error) Pulse Width		40	30		ns
T_{PW3}	Data Available Logic "0" State between Data Bytes		25	45		ns
T_S	Output Control Set-Up Time Prior to Register Read Negative Edge		0	-5		ns
T_H	Output Control Hold Time after the Register Read Positive Edge		0	-5		ns
T_{ZE}	Delay from Output Enable to Logic "1" or Logic "0" from High Impedance State	Load Circuit 2		25	35	ns
T_{EZ}	Delay from Output Enable to High Impedance State from Logic "1" or Logic "0"	Load Circuit 2		25	35	ns
F_{MAX}	Data Bit Frequency (Clock Input must be 8 × the Data Bit Frequency)		DC		3.5	MBits/s

Note 1: "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

Note 2: Unless otherwise specified, min./max. limits apply across the 0°C to +70°C temperature range and the 4.75V to 5.25V power supply range. All typical values are for T_A = 25°C and V_{CC} = 5.0V.

Note 3: All currents into device pins are shown as positive; all currents out of device pins are shown as negative; all voltages are referenced to ground, unless otherwise specified. All values shown as max. or min. are so classified on absolute value basis.

Note 4: Only one output at a time should be shorted.

Note 5: Input characteristics do not apply to amplifier inputs (pins 2 & 3).

Note 6: Unless otherwise specified, all AC measurements are referenced to the 1.5V level of the input to the 1.5V level of the output and load circuit 1 is used.

Note 7: AC tests are done with input pulses supplied by generators having the following characteristics: Z_{OUT} = 5Ω, T_r ≤ 5 ns, and T_f ≤ 5 ns.

Note 8: T = 1/(clock input frequency), units for "T" should be ns.

Test Load Circuits
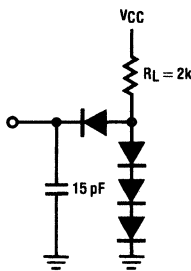
Load Circuit 1

TL/F/5237–8
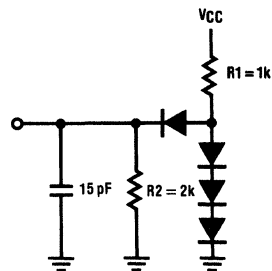
Load Circuit 2

TL/F/5237–9

FIGURE 7

Timing Waveforms
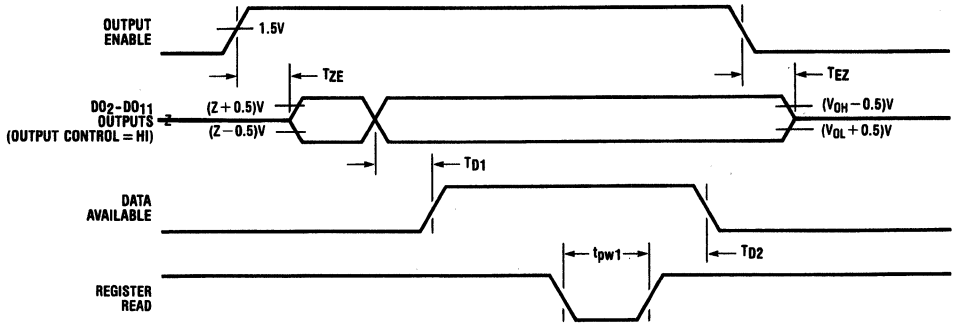
FIGURE 8. Data Sequence Timing

TL/F/5237–10
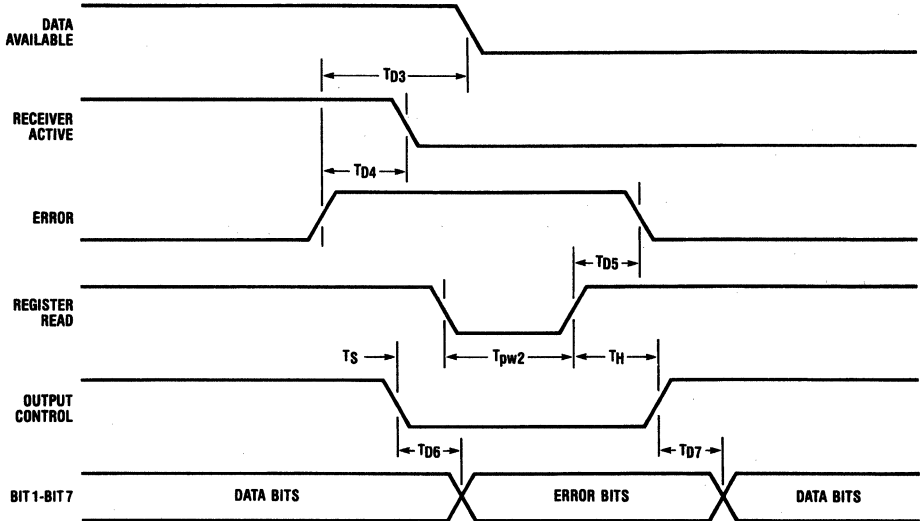
FIGURE 9. Error Sequence Timing

TL/F/5237–11
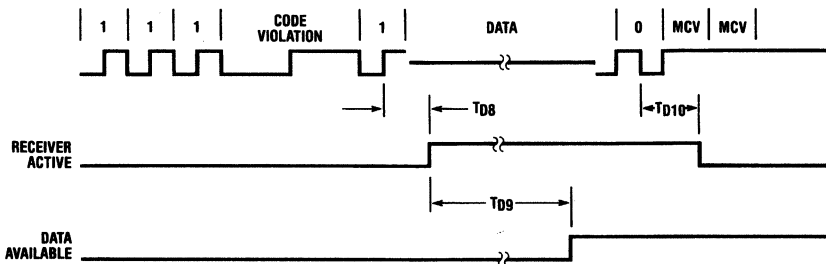
FIGURE 10. Message Timing

TL/F/5237–12

Timing Waveforms (Continued)

SERIAL DATA

T_{D12}
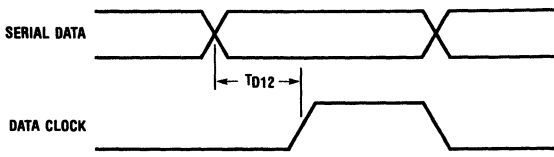
DATA CLOCK

TL/F/5237-13

FIGURE 11. Data Clock and Serial Data Timing

$8T \pm (T - 25\,ns)$

$4T \pm (T - 25\,ns)$

$T = \dfrac{1}{\text{CLOCK INPUT FREQUENCY}}$

1.3V MAX.
40 mV MIN. V_{IN+}

$V_{IN} -$

−40 mV MIN.
−1.3V MAX. V_{IN+}
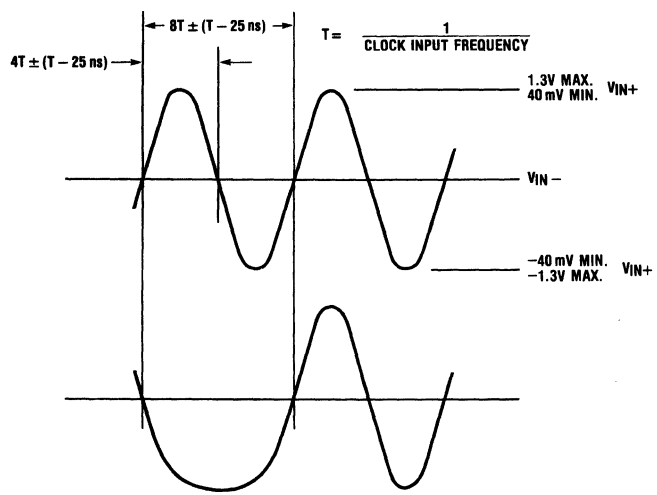
TL/F/5237-14

FIGURE 12. Data Waveform Constraints: Amplifier Inputs

$8T \pm (T - 25\,ns)$

$4T \pm (T - 25\,ns)$

$T = \dfrac{1}{\text{CLOCK INPUT FREQUENCY}}$
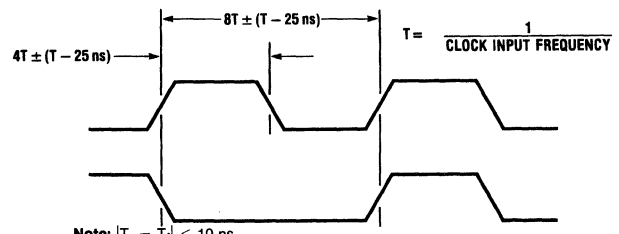
Note: $|T_r - T_f| \leq 10\,ns$

TL/F/5237-15

FIGURE 13. Data Waveform Constraints: Data Input (TTL)

V_{CC}

1k

1k

1k

V_{IN+}

V_{IN-}
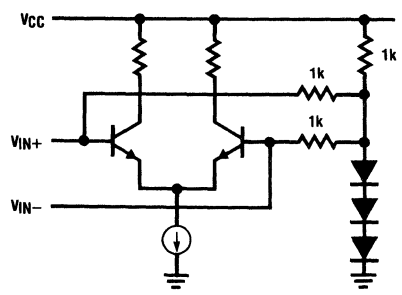
TL/F/5237-16

FIGURE 14. Equivalent Circuit for DP8343/NS32443 Input Amplifier

Typical Applications

28 MHz MAX. (NOTE 1)

V_CC

X1   X2

RESET
AUTO RESPONSE
REG LOAD
REG FULL
BYTE CLOCK

DP8342
TRANSMITTER/
ENCODER

DATA
DATA
DELAY
DATA

OPTIONAL
INTERFACE
LOGIC
(FIG. 16)

TRANSMISSION
MEDIUM

COAX LINE (FIG. 16)
TWISTED PAIR LINES
FIBER-OPTIC
MAGNETIC
INFRARED
RF
ULTRASONIC
AUDIO
CURRENT CARRYING

DC TO 3.5 MHz

SYSTEM INTERFACE BUS

DATA BUS D1-D8

CLOCK

TRANSMITTER
ACTIVE

RECEIVER
DISABLE

DATA
AVAILABLE
ERROR
OUTPUT CONTROL
OUTPUT ENABLE
REG READ
RECEIVER ACTIVE

DP8343
RECEIVER/
DECODER

+IN

−IN

OPTIONAL
INTERFACE
LOGIC

TL/F/5237–17

Note 1: Crystal manufacturer Midland Ross Corp., NEL Unit Part No. NE-18A @ 28 MHz

FIGURE 15
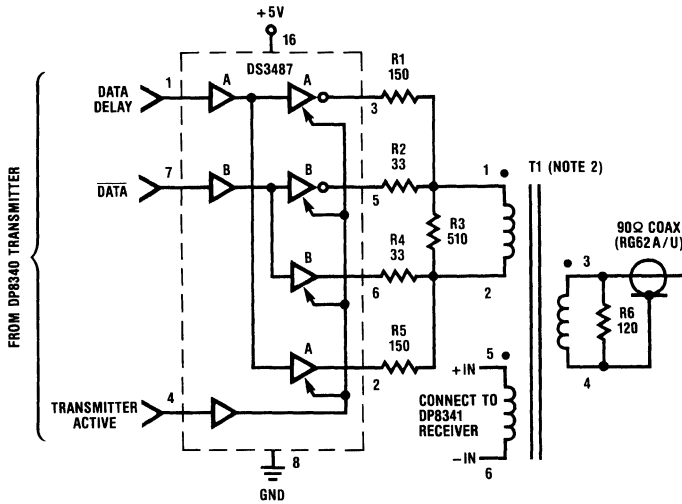
TL/F/5237–18

Note 1: Resistance values are in Ω, $\pm 5\%$, $1/4$W.

Note 2: T1 is a 1:1:1 pulse transformer, $L_{MIN} = 500$ μH for 18 MHz system clock.
Pulse Engineering Part No. 5762,
Valor Electronics Part No. CT1501
Technitrol Part No. 11LHA or equivalent transformers.

FIGURE 16. Interface Logic for a Coax Transmission Line

TL/F/5237–19

TL/F/5237–20

*To maintain loss at 95% of ideal signal, select transformer inductance such that:

$$L_{(MIN)} = \frac{10,000}{f_{CLK}}$$

f_{CLK} = System Clock Frequency (e.g., 18.87 MHz)

Note 1: Less inductance will cause greater amplitude attenuation.

Note 2: Greater inductance may decrease signal rise time slightly and incease ringing, but these effects are generally negligible.

Example:

$$L = \frac{10,000}{18.87 \times 10^6} \rightarrow L(MIN) = 530 \ \mu H$$

FIGURE 17. Transformer Selection

The BIPLAN™ DP8342/DP8343 Biphase Local Area Network

THE BIPLAN

The BIPLAN is a star local area network designed to demonstrate the capabilities of National Semiconductor's DP8342/43 transmitter/encoder and receiver/decoder chips. These chips are eight bit versions of the DP8340/41 ten bit parts designed to conform to the IBM 3270 protocol. These eight bit devices are ideal for general purpose high speed serial communication. They enable communication at any data rate up to 3.5 megabits/sec over a variety of transmission media with a minimum of external components and easily interface to an eight bit data bus. These devices automatically provide line conditioning, manchester encoding and error checking minimizing transmission errors while enhancing noise immunity and reliability.

The LAN system described here is a star network (see *Figure 1*) supporting up to 256 nodes with either fiber-optic links or coax links or both (simultaneously) at distances up to 2 miles and a data rate of 3.5 megabits/sec.

To demonstrate this LAN system, a PC board has been developed which can be configured as a master or a slave (the slave hardware is a subset of the master hardware). As a master, the board will support 8 fiber-optic slaves and four coax slaves and can be expanded to support up to 128 fiber-optic slaves and 128 coax slaves simultaneously (see *Figure 2*). The network interface will communicate with its host either serially (RS-232) or through an eight bit parallel port (multibus). Some features of the network system include:

— 3.5 Megabits/sec data rate

— Distance between slaves—2 miles for coax

— Simultaneous support for 128 fiber-optic slaves and 128 coax slaves

— Protocol insures data integrity—All transfers acknowledged

— 1–254 byte transfers, up to four 254 byte pages per data packet

— 1 kbyte transmit and receive buffers.
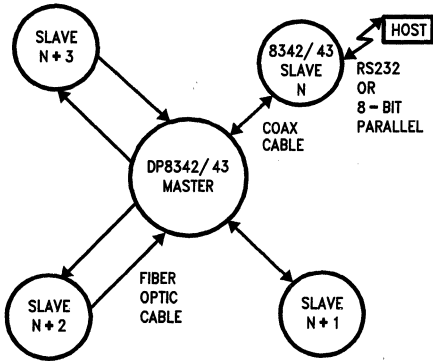
TL/F/9339–1
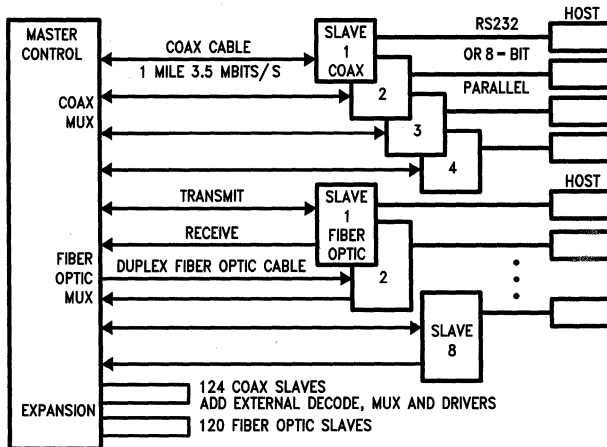
FIGURE 1

TL/F/9339–2

FIGURE 2. Master/Slave Network Configuration

NETWORK PROTOCOL

The central node controls access to the network and is therefore termed "master" and satellite nodes are "slaves" since they provide no network control. The master polls each slave sequentially to offer access to the network. Since the master polls one slave at a time and no slave may transmit unless polled, there is no possibility of contention. If a slave is not ready to transmit data, it responds to a poll with an "auto-response" and the master polls the next slave (see *Figure 3*). Both the poll and the auto-response (AR) are a single byte transmission with all zero data bits (message types will be discussed in detail later). A disabled or disconnected slave will cause the master to time out and poll the next slave.

POLLING

* MASTER OR SLAVE MAY HAVE A HOST WHICH IS A TERMINAL, CPU, ETC.
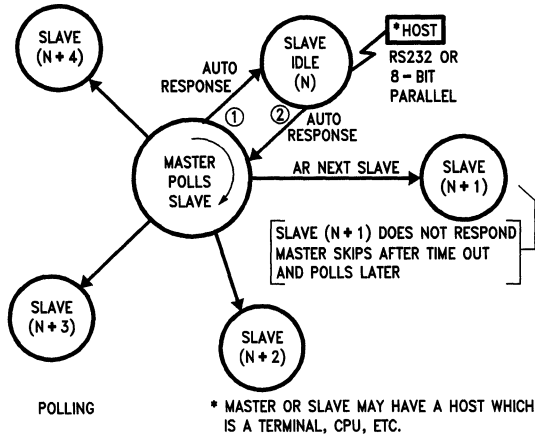
TL/F/9339–3

FIGURE 3

If a slave is ready to transmit, it responds to a poll with a "request to transmit" indicating the number of pages to be transmitted and a destination address. The master sends this "request to transmit" to the destination slave. If the destination slave is not ready to receive data, it sends a "no-permission to transmit" and the master sends it to the source slave deferring data transfer until the destination is ready to receive it (see *Figure 4*). This pre-interrogation prevents wasted data transfers thus improving system throughput. It also allows each node to prepare its DMA circuitry to transfer a block of data.

REQUEST TO TRANSMIT
(SOURCE READY DESTINATION NOT)
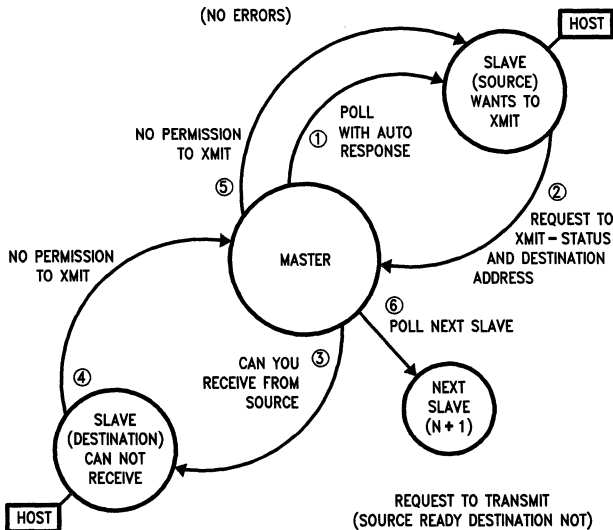
TL/F/9339–4

FIGURE 4

In the case where the destination slave is ready to receive data, it sends a "permission to transmit" and prepares to receive data. The master, on receiving this "permission to transmit" sends it to the source slave and prepares for a transparent transfer of data from source slave to destination slave. The source transmits data and if it reaches the destination without error, the destination slave sends an acknowledge byte. The data/acknowledge cycle continues until the last page of data is transferred. At this point the destination slave sends a special acknowledge called an EOT ("end of transmission") terminating the communication sequence and releasing the master to poll the next slave (see *Figure 5*).
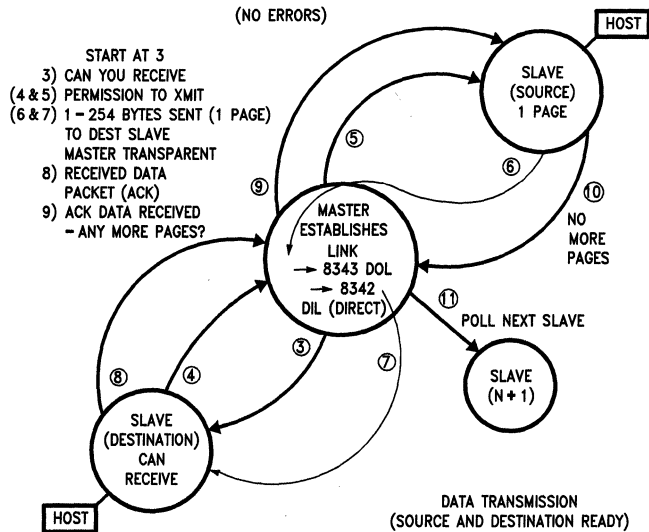
FIGURE 5

TL/F/9339-5

ERROR HANDLING

Recovery from transmission error is handled in the following way. If any node (either master or slave) detects an error while receiving any message from the network (data or control), it sends an error message to the sending node. On receiving an error message, a node retransmits its last message (examples are shown in *Figure 6*). This may continue to a limit of five retransmission attempts per communication sequence. An error message is simply the error flag register of the DP8343 receiver indicating the type of error that occurred. The receiver provides the following types of internal error checking:

— Data overflow

— Parity error

— Transmit check

— Invalid ending sequence

— Loss of mid-bit transition

— New starting sequence before read

— Receiver disabled while active

Error on Poll

* SLAVE DETECTS ERROR

TL/F/9339–6

Error on Auto Response

* MASTER DETECTS ERROR

TL/F/9339–7

Error on Data Transfer

* DESTINATION SLAVE DETECTS ERROR

TL/F/9339–8

FIGURE 6
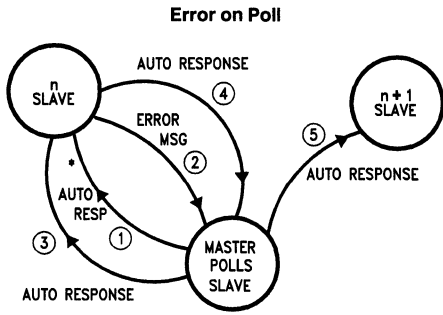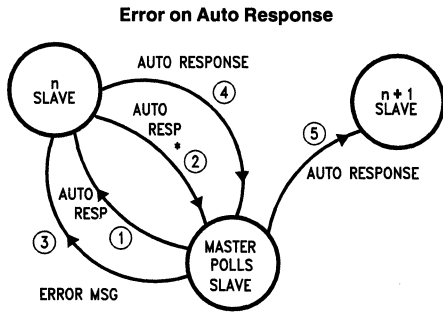
An exception to this rule is during a transparent data transfer (from source slave through transparent master to destination slave), if the master detects an error, it will not send an error message to the source slave. Instead, the master forces a parity error on the next data byte causing the destination to detect a parity error in the data. The destination slave sends an error message to the master and the master then sends the error message to the source slave which re-attempts the data transfer (see *Figure 7*). This method of forcing a parity error at the master informs the destination slave of the error condition immediately without having to compare byte counts and enables quicker recovery.

The complete network protocol is summarized by the flow chart in *Figure 8*.
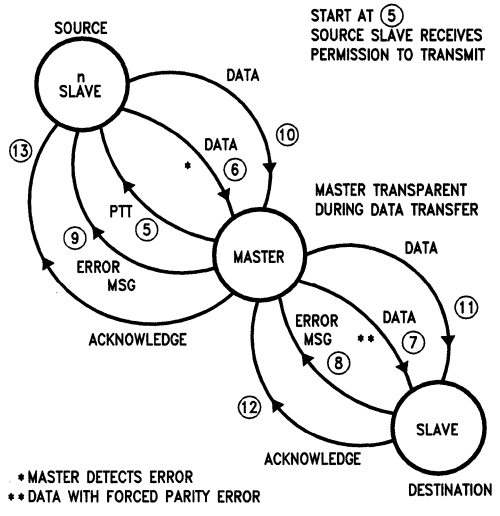
* MASTER DETECTS ERROR
** DATA WITH FORCED PARITY ERROR

TL/F/9339–9

FIGURE 7. Error on Data Transfer
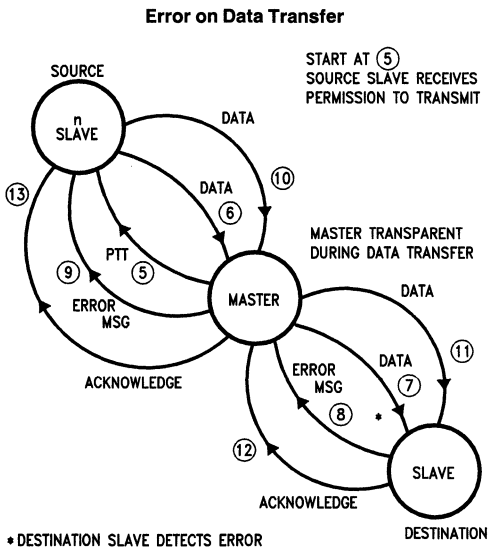
INTERFACE / DESIGN

THE BIPLAN
PROTOCOL FLOW CHART

TYPICAL SEQUENCES FOR HANDLING ERRORS

(MASTER POLLS SLAVE AGAIN)

a

SLAVE DETECTS ERROR — Y → SLAVE SENDS ERRMSG (TO MASTER)

a' | N

b

SLAVE DETECTS ERROR → MASTER SENDS ERRMSG

b'

(SLAVE SENDS AR AGAIN)

MASTER POLLS SALVE WITH AR ← NEXT SLAVE

a

SLAVE WANTS ACCESS ? — NO → SLAVE SENDS AR (b)

YES

SLAVE (SOURCE) SENDS RTT AND DEST ADDRESS (TO MASTER)

MASTER SENDS RTT AND DEST ADDRS TO DEST

DEST SLAVE RESPONDS — NO → MASTER SENDS NO RTT TO SOURCE

YES

DEST SLAVE RESPONDS — NO → DEST SLAVE SENDS NO RTT TO MASTER

YES

DESTINATION SLAVE SENDS PTT TO MASTER

MASTER SENDS PTT TO SOURCE
MASTER ESTABLISHES TRANSPARENT LINK

AR – AUTO RESPONSE
RTT – REQUEST TO TRANSMIT
PTT – PERMISSION TO TRANSMIT
ACKMSG – ACKNOWLEDGE MESSAGE
ERRMSG – ERROR MESSAGE

(NEW PAGE OF DATA)

MASTER ESTABLISHES TRANSPARENT LINK

SOURCE SENDS DATA PACKET TO DEST SLAVE

(SAME PAGE OF DATA)

(LIMIT 3 ATTEMPTS)

MASTER DETECTS ERROR — YES → MASTER FORCES PARITY ERROR

MASTER SENDS ERRMSG TO SOURCE

NO

DEST SLAVE DETS ERROR — YES → DEST SLAVE SENDS ERRMSG TO MASTER

NO

SOURCE SENDS AR

DEST SENDS ACK MESSAGE TO MASTER → MASTER SENDS ACK MESSAGE TO SOURCE → LAST PAGE OF DATA — YES

NO

FIGURE 8. BIPLAN15B

TL/F/9339–10

MESSAGE TYPES

There are two major types of messages on the network: control messages and data messages. Control messages are one or two bytes in length and include the following types: poll, auto-response, request to transmit, permission to transmit, acknowledge and error message (see *Figure 9*). Data messages are 3 to 256 bytes in length and include 1 to 254 bytes of data. Once a node is granted access to the network, it is allowed to transmit up to 4 such data messages (or pages) so that any number of data bytes from 1 to 1016 bytes (4 pages) can be transferred per access. All messages, data as well as control, begin with a status byte as defined in *Figure 9*.

Data communication rates including overhead for the protocol are shown in *Figure 10*. Note that the effective data rate is optimum for large data packets as the overhead becomes a less significant portion of the total time for the data transfer.

Data Communication Rates Source Slave to Destination Slave

Neglecting Cable Propagation Delays
(RG 62/AU Coax = 1.2 ns/ft = 3.6 ns/meter)

First Page		Next Two to Four Pages	
No. of Bytes	Time (μs)	No. of Bytes	Time (μs)
1	260	1	115
10	286	10	143
100	550	100	400
254	1000	254	840

Total Time for Transfer of: 1 Page (254 Bytes)—1000 μs
2 Pages (508 Bytes)—1840 μs
3 Pages (762 Bytes)—2680 μs
4 Pages (1016 Bytes)—3520 μs or
2.3 Mbits/sec

LATENCY

No Network Traffic—(40)(N)μs

(N) is the number of slaves on the network

FIGURE 10

THE BIPLAN
MESSAGE TYPES

(1) POLL ┌── PRE‑AMBLE S 00000000 P POST‑AMBLE ──┐ (AUTO RESPONSE)

LINE QUIESE | DATA | ENDING SEQUENCE

SYNC   PARITY
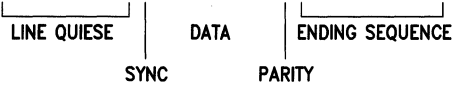
(2) SLAVE RESPONSE TO POLL (NOT REQUESTING ACCESS)
    SAME (AUTO RESPONSE)

(3) REQUEST TO TRANSMIT

PRE‑AMBLE S 0XXX0000 P | S XXXXXXXX P POST‑AMBLE

STATUS BYTE   DESTINATIION ADDRESS

(4) PERMISSION TO TRANSMIT

PRE‑AMBLE S 0XXX0001 P S XXXXXXXX P POST‑AMBLE

STATUS BYTE   DESTINATIION ADDRESS

(5) DATA

PRE‑AMBLE S 0XXX0010 P S XXXXXXXX P S XXXXXXXX P S XXXXXXXX P POST‑AMBLE

STATUS BYTE   SOURCE AD   DATA BYTE 1   LAST DATA BYTE

(6) ACKNOWLEDGE   PRE‑AMBLE S 0XXX0X11 P POST‑AMBLE

STATUS BYTE

STATUS BYTE
0XXXXXXX   1XXXXXXX

NO ERROR – 0	
ERROR – 1	
PAGE NUMBER –	
REQ TO TRANSMIT – 000	NOT USED
PERM TO TRANSMIT – 001	MESSAGE TYPE
DATA – 010	
ACK (NO ERROR) – 011	
END OF XMISSION – 111	

ERROR MESSAGE

ERROR TYPE
(FROM DP8343 ERROR FLAG REG)

0000000 – COMM SEQUENCE ERROR
0000001 – DATA OVERFLOW
0000010 – PARITY
0000100 – TRANSMIT CHECK
0001000 – INVALID ENDING SEQ
0010000 – LOSS OF MID – BIT XSTION
0100000 – NEW SEQ BEFORE READ
1000000 – RCVR DISABLE WHILE ACTIVE

FIGURE 9. BIPLAN17

TL/F/9339–11

DESCRIPTION OF HARDWARE

A block diagram of the network interface showing the major functional elements and the bus structure is shown in *Figure 11*. This represents both the master and the slave except the master contains additional circuitry for multiplexing and de-multiplexing the biphase signal. The CPU (NSC800) provides the intelligence necessary to communicate with the host (8-bit parallel or RS-232 serial) and to implement the network protocol. The DP8342/43 transmitter and receiver both have 2 byte buffers so the CPU can transmit and receive one or two bytes at any time without critical timing requirements. However, the CPU is too slow to accommodate the 350 kbytes/sec data rate during multi-byte (more than 2) transfers. A DMA state-machine controls the transfer of these fast multi-byte messages. Thus, the CPU handles all the control transfers on the network (all 1 or 2 bytes) but the high-speed data must be transferred (to or from the DP8342/43) using direct-memory-access.
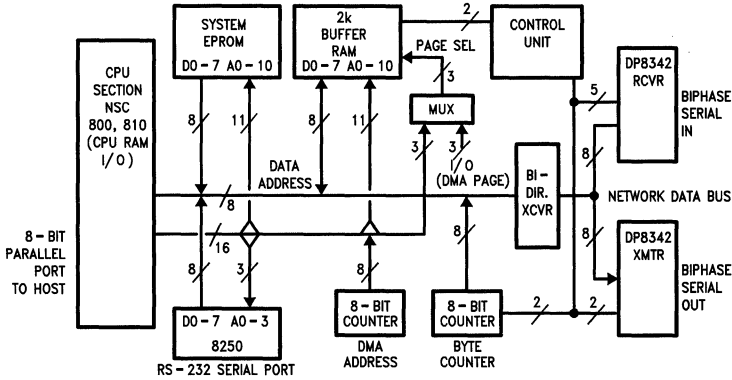
TL/F/9339–12

FIGURE 11. Network Interface Bus Structure (Slave)

The state-machine controlling the DMA sequences consists of EPROMS and latches. The CPU commands the state machine using two I/O pins called "receive request" and "transmit request". A 2 kbyte buffer stores transmit and receive data and is segmented in to eight pages of 256 bytes each. Four pages are allocated for transmit data and four for receive data. When a node has been granted permission to transmit, the CPU loads the appropriate page, loads the DMA counters and asserts "transmit request". Similarly, if a node has granted permission to transmit, it prepares to receive data by loading the appropriate page, initializing the DMA counters and asserting "receive request". The master may assert both "receive request" and "transmit request" simultaneously to effect a "repeat request" (transparent mode where data from the receiver is loaded directly to the transmitter). *Figure 12* shows the control signals involved in the DMA sequences including those required to handshake with the transmitter and receiver.
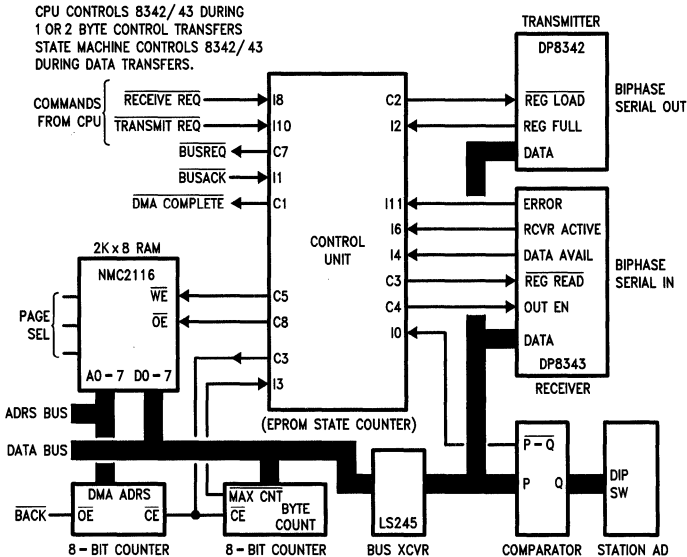
TL/F/9339–13

FIGURE 12. DMA Section

An EPROM implementation was selected for the state machine in the interest of flexibility and to keep its operation as lucid as possible. The state machine has three main functions: transmit data (DMA read), receive data (DMA write) and repeat data. *Figure 13* is the state diagram showing how it accomplishes these functions. *Figures 14, 15* and *16* show how the state machine handshakes with the DP8342/43 in getting data on and off the bus.
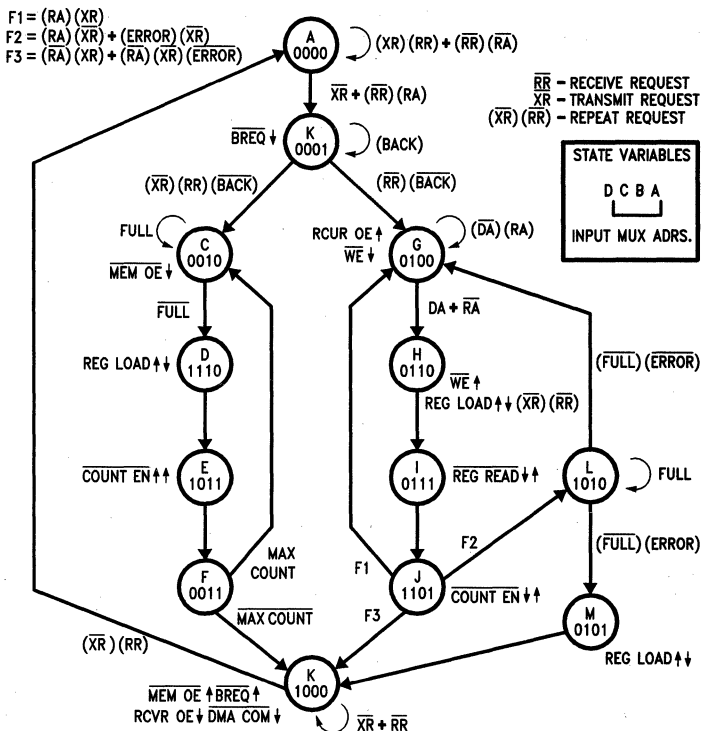
FIGURE 13. Control Unit State Diagram

TL/F/9339–14
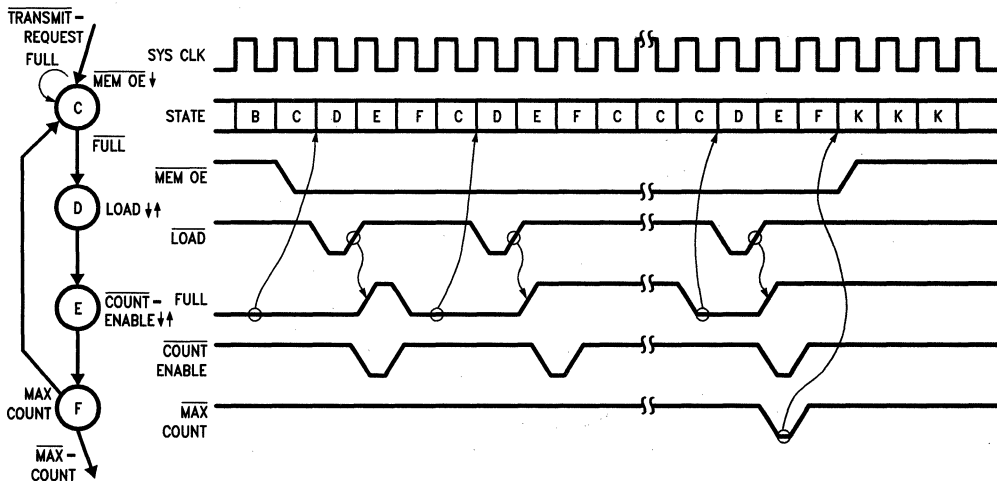
TL/F/9339–15

TRANSMIT REQUEST (FROM CPU) INITIATES DATA TRANSMIT SEQUENCE. CONTROLLER WAITS (IN STATE C) UNTIL REG FULL IS LOW BEFORE DOING EACH REG LOAD (IN STATE D). MAX COUNT TERMINATES THE SEQUENCE.

FIGURE 14. DMA Transmit Timing (Read from Memory)
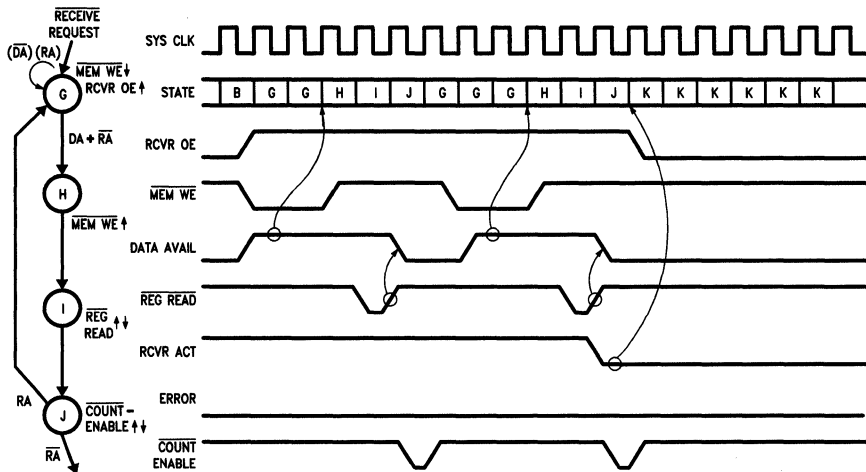
TL/F/9339–16

RECEIVER ACTIVE (RA) INITIATES DMA WRITE SEQUENCE ONLY IF REC REQ (RR) IS LOW. FOR EACH WRITE CYCLE, WRITE EN IS KEPT LOW UNTIL DATA AVAIL (DA) GOES HIGH. THIS SIGNIFIES THAT VALID RECEIVE DATA IS ON THE BUS AND THE WRITE CYCLE IS COMPLETED.

FIGURE 15. DMA Receive Timing (Write to Memory)
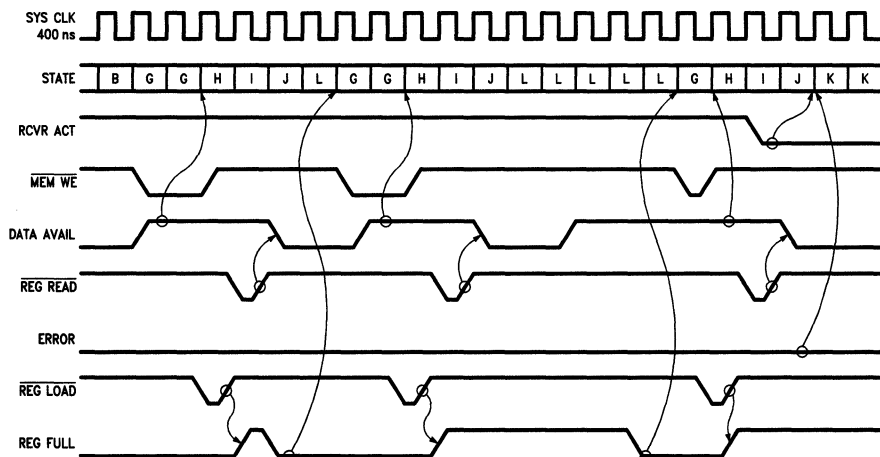
TL/F/9339–17

REPEAT SEQUENCE IS INITIATED ON RECEIVER ACTIVE IF BOTH REC REQ AND XMT REQ ARE LOW. FOR THE FIRST BYTE, THE CONTROLLER WAITS (IN STATE G) FOR DATA AVAILABLE BEFORE DOING A LOAD. FOR SUBSEQUENT BYTES, THE CONTROLLER WAITS (IN STATE L) FOR REGISTER FULL TO GO LOW THEN WAITS FOR DATA AVAILABLE BEFORE DOING EACH LOAD.

FIGURE 16. Repeat Timing (Master Transparent Mode)

2

MASTER MULTIPLEXING/DE-MULTIPLEXING

The network master must communicate with multiple slaves so that some method of multiplexing and de-multiplexing the high-speed biphase signals is necessary. For receiving data we must accommodate analog signals from the coaxial links and TTL signals from the fiber-optic receivers. This is easy since the DP8343 receiver has both TTL and analog inputs and an internal mux to select the input. A TTL multiplexer was used to select one of eight fiber-optic channels (expandable off board to 128 channels). For the coax channels, instead of using line receivers and then multiplexing the TTL signals, an analog mux was used to select one of four coax channels (also expandable to 128 channels). This method allows us to take advantage of the excellent input characteristics of the line receiver within the DP8343 receiver and minimizes the number of external components (see *Figure 17*). For transmitting data, the master must select one of eight fiber-optic drivers or one of four coaxial line drivers. *Figure 18* shows how this is done.

The hardware has been designed for maximum flexibility and to provide a friendly environment for developing communication software. What has been implemented in the present system is the first two layers of the ISO/OSI model of local area networks (the physical layer and the data link layer). That is, the system provides the transfer of data from one station to another assembling frames and handling transmission errors. There is sufficient bandwidth remaining on the network interface CPU to implement any host system interface.

BIPLAN23
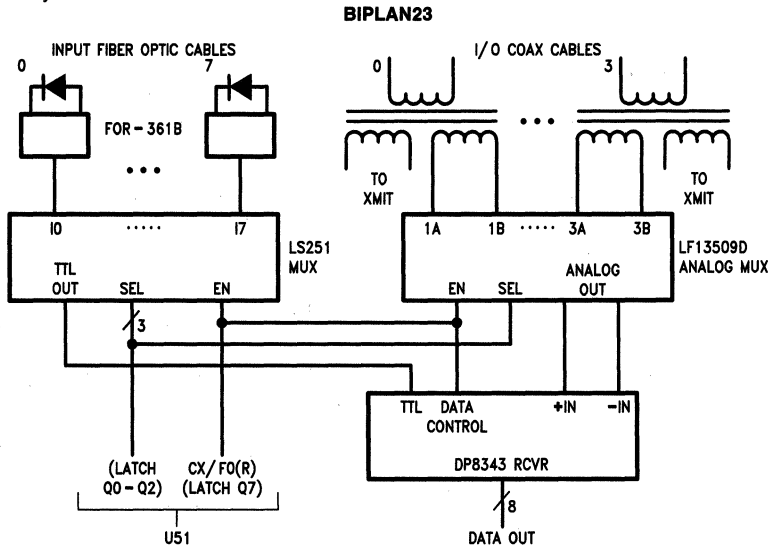
TL/F/9339–18

FIGURE 17. Receiving Mux (Master)
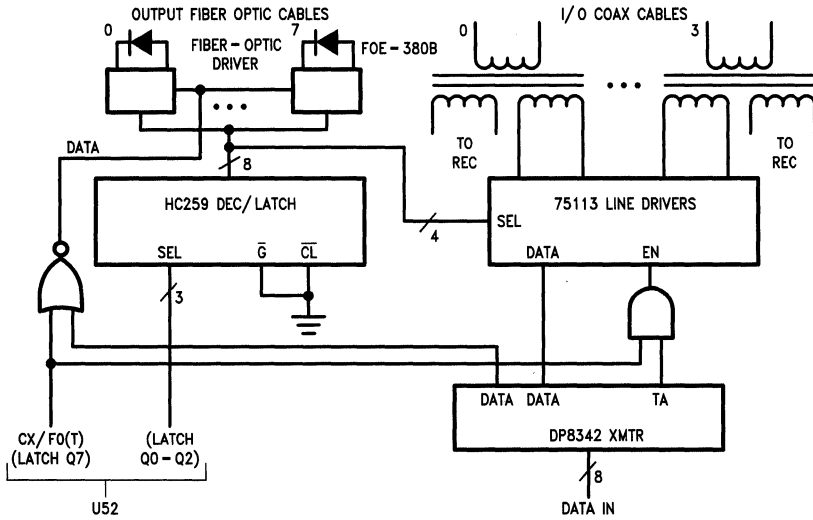
BIPLAN24

FIGURE 18. Transmitter Select (Master)

TL/F/9339–19

THE BIPLAN DP8342/DP8343 BIPHASE LOCAL AREA NETWORK

TL/F/9339–21

2

TL/F/9339–22

THE BIPLAN DP8342/DP8343 BIPHASE LOCAL AREA NETWORK

5V

(U16/20) TA
CX/FO (T)
(U52/6)

U39

Vcc 2YP 12
1B 2YS 13
CC 2ZS 14
1A 2ZP 15
2A 1ZP 1
2C 1ZS 2
1C 1YS 3
1YP 4
M U32
DS75113
LINE DRIVER

(U30/4) OC
(U30/5) Q1

−12V
12V
−VEE
Vcc
EN
A0
A1
DE
DA

S1A 4
S1B 13
S2A 12
SAB 6
S3A 11
S3B 7
S4A 5
S4B 10
M U31
ANALOG MUX
LF13509D
G2△

CX/FO (R)
(U51/6)
(U51/3) A
(U51/4) B

5V
W20

DATA
(U16/16)

Vcc 2YP 12
1B 2YS 13
CC 2ZS 14
1A 2ZP 15
2A 1ZP 1
2C 1ZS 2
1C 1YS 3
1YP 4
M U33
DS75113
LINE DRIVER

(U30/6) Q2
(U30/7) Q3

R5 91
T1
U34
R5 91
T3
R6 91
T2
R6 91
U35 M
T4

BNC C10 0
C10 1
4 CH COAX
BNC C10 2
C10 3

W8
(U16/17) DATA
(U16/17) DATA

LS32 (U4/12) C3
U14 I12 (U3/7)
ERROR (W23/2) 5V

U15

(W21/1) ERROR
U26

(W27/1) DA
U26

CA U14
U26

U13
RD

(U6/15) I4
(U23/5) XR
(U23/2) RR
U39

READ Vcc 24
ERROR DC 5
RA GND 12
RA
OUT CONT G2
OUT EN TTL
B7 IN
B6
B5 DP8343
B4 RCVR
B3 U17
B2
B1 −IN
B0 +IN
RCVR
DIS CLK

W9
(U51)
A
B
C
M Y U40
LS251 MUX
STB
GND

5V 16
I0
I1
I2
I3
I4
I5
I6
I7

FOR 0 FOR 0 FOR 0 FOR 0
M M M M
5V 5V 5V 5V

FOR 361B−2
FIBER−OPTIC
RECEIVERS

FOR 4 FOR 4 FOR 4 FOR 4
M M M M
5V 5V 5V 5V

J2−48 47
J2−50
J2−49
J2−49

TL/F/9339−26

2

TL/F/9339–23

Notes:

*—01 IS A SEPARATE GROUND BUS FROM U41–U44 TO EDGE (P1)

**—V1 IS A SEPARATE V$_{CC}$ BUS FROM U41–U44 TO EDGE (P1)

Δ—G2 IS ANOTHER GROUND BUS FROM U31 AND U17 TO EDGE (P1)

M—FOR MASTER SYSTEM ONLY.

R—FOR RING SYSTEM ONLY.

THE BIPLAN DP8342/DP8343 BIPHASE
LOCAL AREA NETWORK
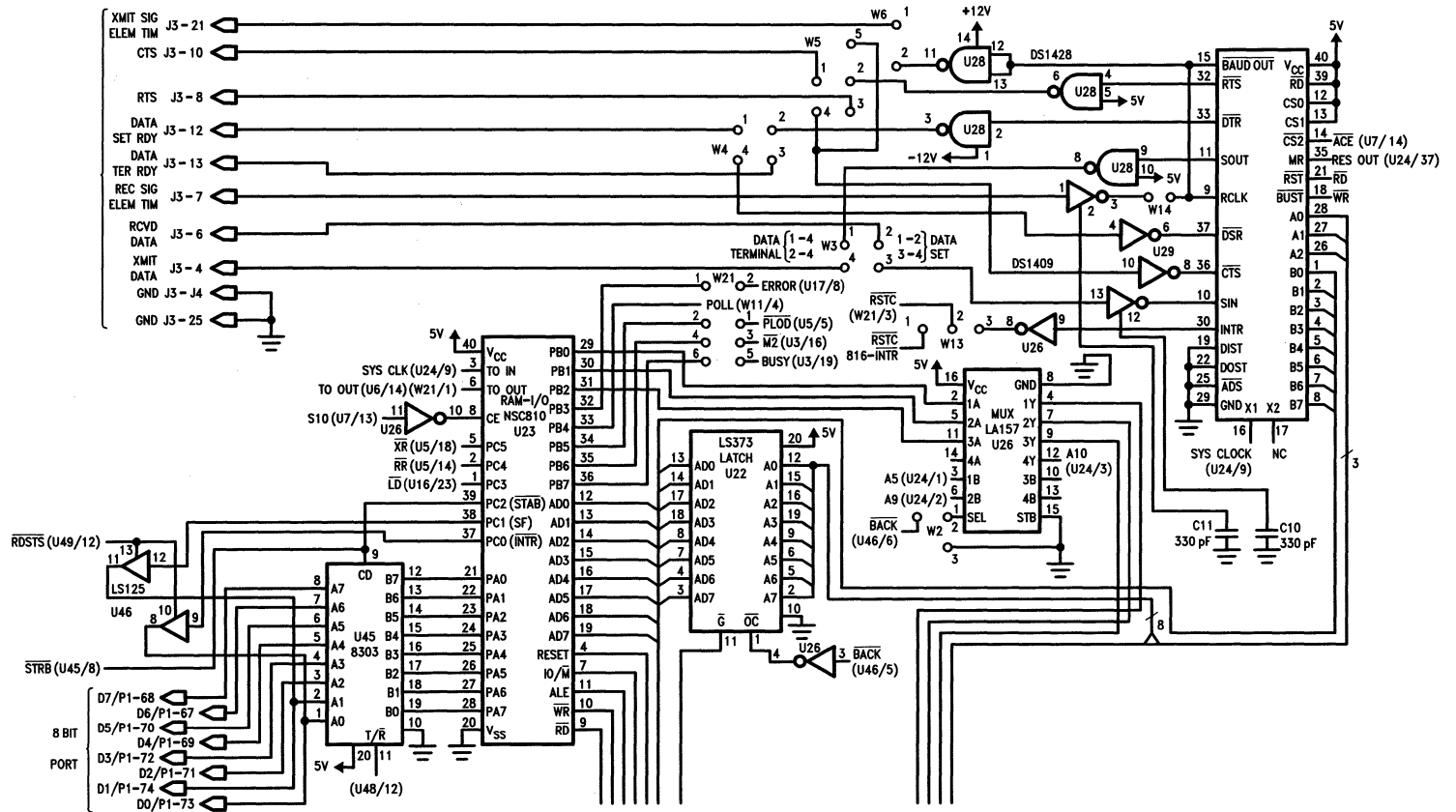
TL/F/9339–24

TL/F/9339–25
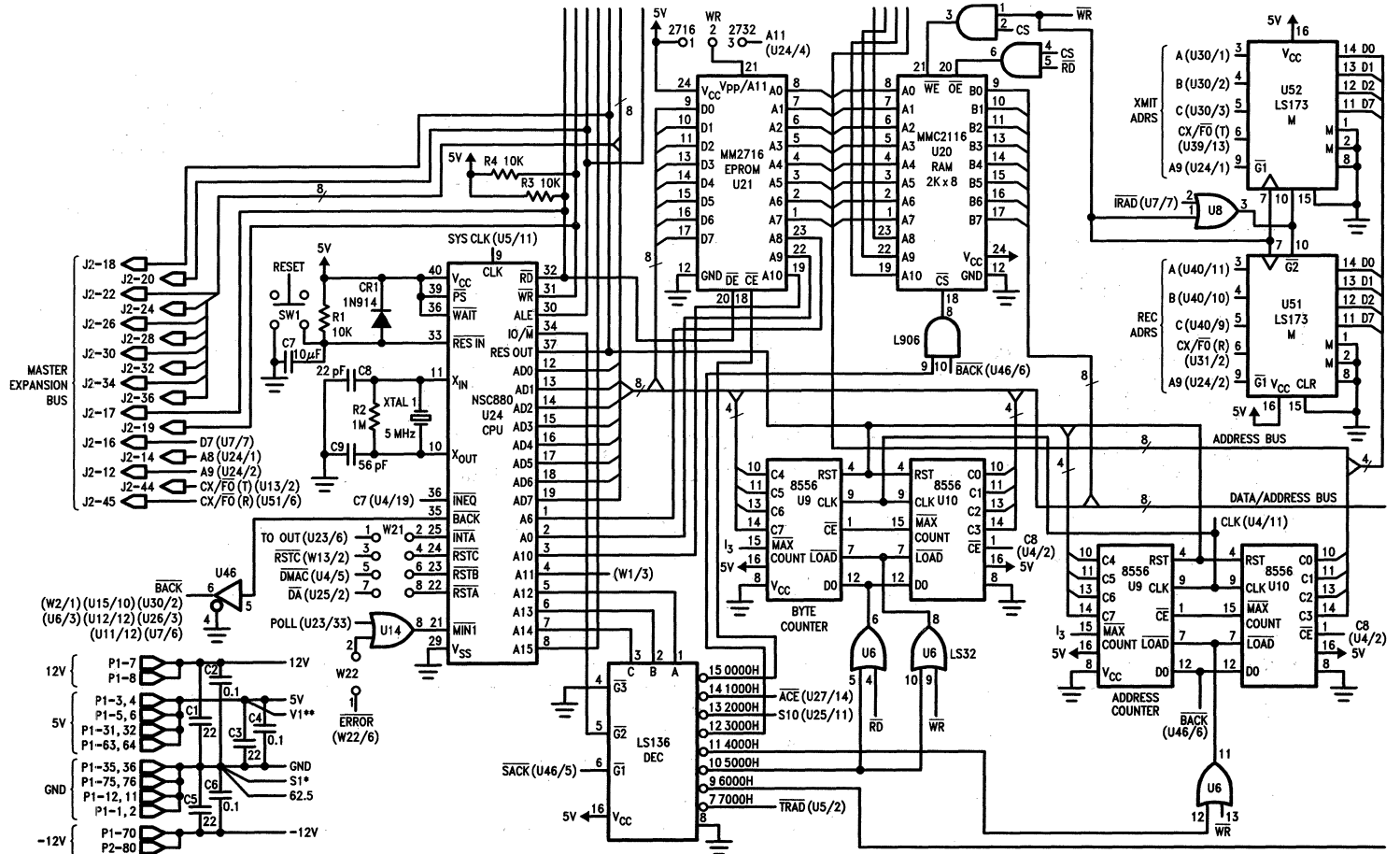
Notes:
*—G1 IS A SEPARATE GROUND BUS FROM U41–U44 TO EDGE (P1)
**—V1 IS A SEPARATE V$_{CC}$ BUS FROM U41–U44 TO EDGE (P1)
Δ—G2 IS ANOTHER GROUND BUS FROM U31 AND U17 TO EDGE (P1)
M—FOR MASTER SYSTEM ONLY.
R—FOR RING SYSTEM ONLY.

THE BIPLAN
Biphase Local Area Network
PARTS LIST

Device	Type	Device	Type
U1, U2	MM2716Q	FOE 0–FOE 7 (M)	FOE–380B
U3, U4, U5	DM74LS374	FOR 0–FOR 7 (M)	FOR-361B
U6	DM74LS151		
U7	DM74LS138		
U8, U14	DM74LS32	R1, R3, R4, R17–R19	10K
U9–U12	DM8556	R2	1M
U13	DM74LS02	R5–R8	91
U15, U39	DM74LS08	R9–R16 (M)	36Ω ½W
U16	DP8342	R20	120
U17	DP8343	RP1 (R)	10K x 8
U20	NMC2116N-25L		
U21	MM2716/2732		
U22	DM74LS373	C1, C3, C5	22 μF, 20V
U23	NSC810	C13, C15, C17, C19	0.1 μF
U24	NSC800	C7	10 μF, 10V
U26	DM74LS04	C8	22 pF
U27 (S)	INS8250A	C9	56 pF
U28 (S)	DS1488	C10, C11	330 pF
U29 (S)	DS1489	C12	15 pF
U30 (M)	MM74HC259	C14, C16, C18, C20	3.3 μf, 10V
U31 (M)	LF13509		
U32–U33 (M)	DS75113		
U36	DM74LS245	SW1	Push Button SW
U37 (R)	MM74HC688	SW2	8 Wide Dip SW
U38	DM74LS157		
U40 (M)	DM74LS251	XTAL1	CPU OSC (5 MHz)
U41–U44 (M)	DS75451	XTAL2	*Bi-Phase OSC (287 MHz)
U45 (P)	DM8303		
U46	DM74LS125	T1–T4	*Pulse Transformers
U47 (P)	DM8136		
U48 (P)	DM74LS00		
U49 (P)	DM74LS10		
U51, U52 (M)	DM74LS173		

Notes:
(M)—Master Only
(R)—Ring Configuration Only
(S)—Serial (RS-232) Link to Host
(P)—8-Bit Parallel Link to Host
*(See DP8342 Data Sheet)

2

DP8342/DP8343 HIGH SPEED INTERFACE FOR REMOTE DATA ACQUISITION

CONTINUOUS CONVERSION AND TRANSMIT 16-CHANNEL UNIDIRECTIONAL A/D SYSTEM APPLICATION

** T1 and T2 pulse transformers Pulse Eng. 5762 or Technitrol 11LHA

Note 1: V_{REF} = 5 V_{DC} for this application.

Note 2: Crystal manufacturer Midland Ross Net Unit—See data sheet for spec.

Note 3: Optional V_{REF} circuit.

Note 4: GND inputs of unused gates.

	5 V_{DC} Pin	GND Pin
DP8342	24	12
DP8343	24	12
MM74C374	20	10
DM74LS74	14	7
MM74C00	14	7
MM74C93	4	11
DM74LS00	14	7
MM74HC04	14	7

NSC800™ is a trademark of National Semiconductor Corporation.

TL/F/9339–20

![National Semiconductor logo] **National Semiconductor**
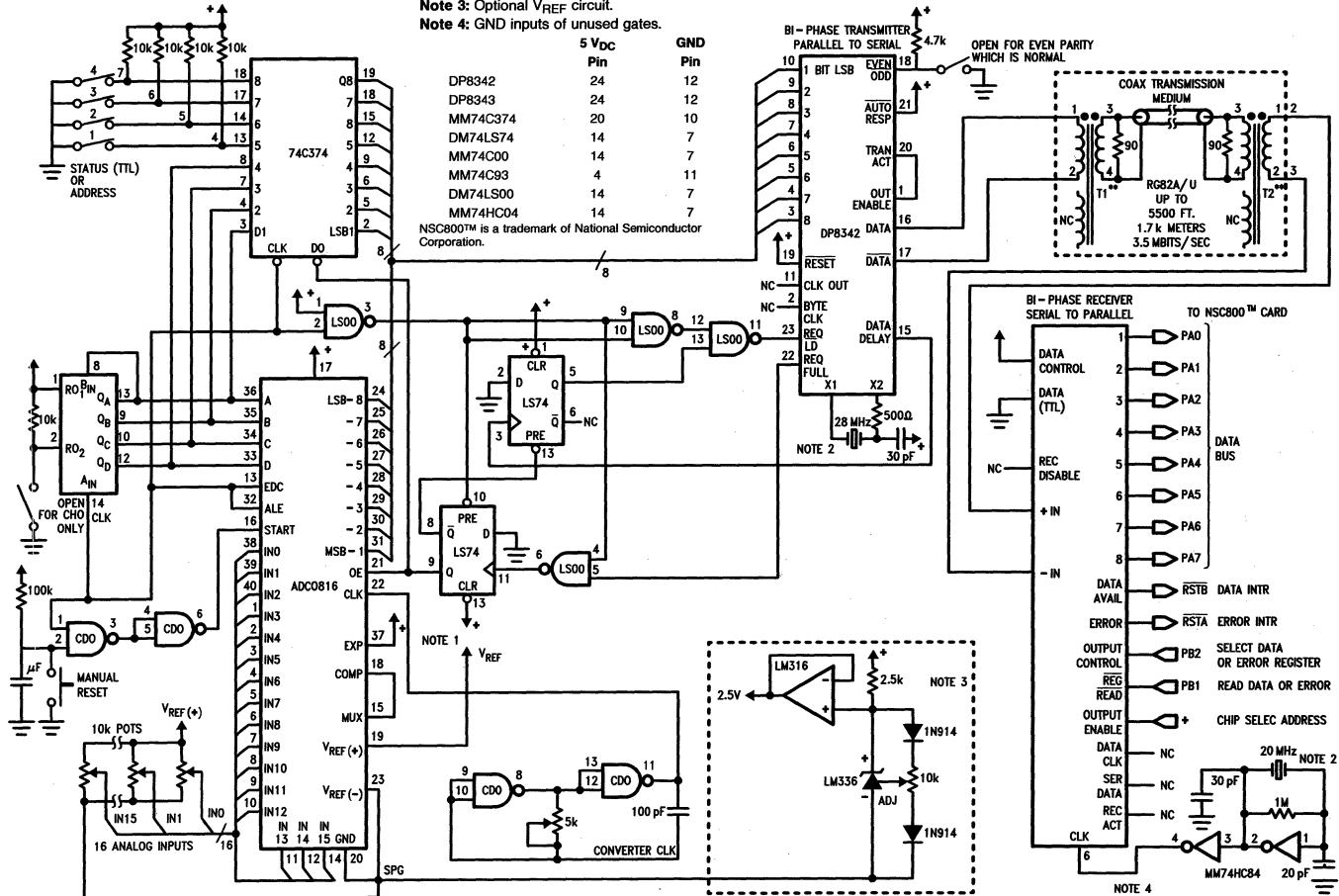
DP8344A Biphase Communications Processor—BCP®

General Description

The DP8344A BCP is a communications processor designed to efficiently process IBM 3270, 3299 and 5250 communications protocols. A general purpose 8-bit protocol is also supported.

The BCP integrates a 20 MHz 8-bit Harvard architecture RISC processor, and an intelligent, software-configurable transceiver on the same low power microCMOS chip. The transceiver is capable of operating without significant processor interaction, releasing processor power for other tasks. Fast and flexible interrupt and subroutine capabilities with on-chip stacks make this power readily available.

The transceiver is mapped into the processor's register space, communicating with the processor via an asynchronous interface which enables both sections of the chip to run from different clock sources. The transmitter and receiver run at the same basic clock frequency although the receiver extracts a clock from the incoming data stream to ensure timing accuracy.

The BCP is designed to stand alone and is capable of implementing a complete communications interface, using the processor's spare power to control the complete system. Alternatively, the BCP can be interfaced to another processor with an on-chip interface controller arbitrating access to data memory. Access to program memory is also possible, providing the ability to download BCP code.

A simple line interface connects the BCP to the communications line. The receiver includes an on-chip analog comparator, suitable for use in a transformer-coupled environment, although a TTL-level serial input is also provided for applications where an external comparator is preferred.

A typical system is shown below. Both coax and twinax line interfaces are shown, as well as an example of the (optional) remote processor interface.

Features

Transceiver
- Software configurable for 3270, 3299, 5250 and general 8-bit protocols
- Fully registered status and control
- On-chip analog line receiver

Processor
- 20 MHz clock (50 ns T-states)
- Max. instruction cycle: 200 ns
- 33 instruction types (50 total opcodes)
- ALU and barrel shifter
- 64k x 8 data memory address range
- 64k x 16 program memory address range (note: typical system requires <2k program memory)
- Programmable wait states
- Soft-loadable program memory
- Interrupt and subroutine capability
- Stand alone or host operation
- Flexible bus interface with on-chip arbitration logic

General
- Low power microCMOS; typ. I_{CC} = 25 mA at 20 MHz
- 84-pin plastic leaded chip carrier (PLCC) package

Block Diagram

Typical BCP System

Note: A typical system will require < 2k program memory

FIGURE 1

TL/F/9336–51

Table of Contents

List of Illustrations

2

List of Illustrations (Continued)

List of Tables

2

Note: To match Timing table number with appropriate Timing illustration, Tables 5-1 and 5-2 are purposely omitted.

1.0 Communications Processor Introduction

The increased demand for computer connectivity has driven National Semiconductor to develop the next generation of special purpose microprocessors. The DP8344A is the first example of a "Communications Processor" for the IBM environment. It integrates a very fast, full function microprocessor with highly specialized transceiver circuitry. The combination of speed, power, and features allows the designer to easily implement a state-of-the-art communications interface. Typical applications for a communications processor are terminal emulation boards for PCs, stand-alone terminals, printer interfaces, and cluster controllers.

The transceiver is designed to simplify the handling of specific communication protocols. This feature makes it possible to quickly develop interfaces and software with little concern for the "housekeeping" details of the protocol being used.

1.1 COMMUNICATIONS PROTOCOLS

A communication protocol is a set of rules which defines the physical, electrical, and software specifications required to successfully transfer data between two systems.

The physical specification includes the network architecture, as well as the type of connecting medium, the connectors used, and the maximum distance between connections. Networks may be configured in "loops," "stars," or "daisy chains," and they often use standard coaxial or twisted-pair cable.

The electrical specification includes the polarity and amplitude of the signal, the frequency (bit rate), and encoding technique. One common method of encoding is called "bi-phase" or "Manchester II." This technique combines the clock and data information into one transmission by encoding data as a "mid-bit" transition. *Figure 1-1* shows how the data transition is related to the bit boundary in a typical transmission. The polarity of the "mid-bit" transition en-

codes the data value, other transitions lie on bit boundaries. Bit boundaries are not always indicated by transitions, so techniques employing start sequences and sync bits are used with bi-phase transmissions to ensure proper frame alignment and synchronization.

The software specification covers the use of start sequences and sync bits, as well as defining the message format. Parity bits may be used to ensure data integrity. The message format is the "language" that is used to exchange information across the connecting medium. It defines command and control words, response times, and expected responses.

The DP8344A Bi-phase Communications Processor supports both the IBM 3270 and 5250 communication protocols, as well as IBM 3299 and a general purpose 8-bit protocol. The specialized transceiver is combined with a microprocessor whose instruction set is optimized for use in a communications environment. This makes the DP8344 a powerful single-chip solution to a wide range of communication applications.

An example of an IBM 3270 message is shown in *Figure 1-2*. The transmission begins with a very specific start sequence and sync pulse for synchronization. This is followed by the data, command, and parity bits. Finally, the end sequence defines the end of the transmission.

The IBM 3270 and 5250 are two widely used protocols. The 3270 protocol was developed for the 370 class mainframe, and it employs coaxial cable in a "star" configuration. The 5250 protocol was developed for the System/3x machines, and it uses a "daisy-chain" of twin-ax cable. A good overview of both of these environments may be found in the "Multi-Protocol Adapter System User Guide" from National Semiconductor, and in the Transceiver section of this document.
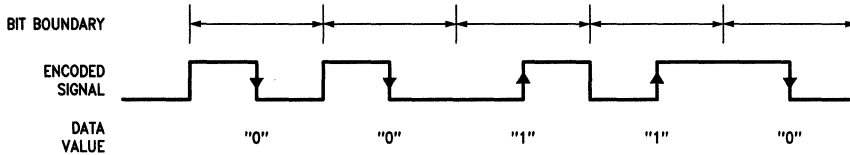
TL/F/9336–B7

FIGURE 1-1. Biphase Encoding

TL/F/9336–B8
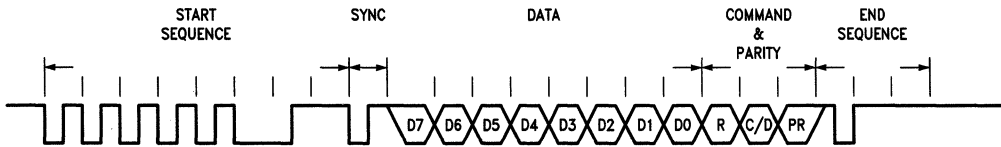
FIGURE 1-2. IBM 3270 Message Format

1.0 Communications Processor Introduction (Continued)

1.2 INTERNAL ARCHITECTURE INTRODUCTION

The DP8344A Biphase Communications Processor (BCP) is divided into three major functional blocks: the Transceiver, the Central Processing Unit (CPU), and the Remote Interface and Arbitration System, RIAS. *Figure 1-3* shows how these blocks are related to each other and to other system components.

The transceiver consists of an asynchronous transmitter and receiver which can communicate across a serial data path. The transmitter takes parallel data from the CPU and appends to it the appropriate framing information. The resulting message is shifted out and is available as a serial data stream on two output pins. The receiver shifts in serial messages, strips off the framing information, and makes the data available in parallel form to the CPU. The framing information supplied by the BCP provides the proper message format for several popular communication protocols. These include IBM 3270, 3299, and 5250, as well as a general purpose 8-bit mode.

The transceiver clock may be derived from the internal oscillator, either directly or through internal divide-down circuitry. There is also an input for an external transceiver clock, thus allowing complete flexibility in the choice of data rates.

The receiver input can come from three possible sources. There is a built-in differential amplifier which is suitable for most line interfaces, a single-ended digital input for use with an external comparator, and an internal loopback path for self testing. Refer to the Transceiver section for a detailed description of all transmitter and receiver functions, and to the application note on coax interfaces for the proper use of the differential amplifier.

The CPU is a general purpose, 8-bit microprocessor capable of 20 MHz operation. It has a reduced instruction set which is optimized for transceiver and data handling performance. It also has a full function arithmetic/logic unit (ALU) which performs addition, subtraction, Boolean operations, rotations and shifts. Separate instruction and data memory systems are supported, each with 16-bit address buses, for a total of 64k address space in each.

There are 44 internal registers accessible to the CPU. These include special configuration and control registers for the transceiver and processor, four 16-bit indices to data memory, and 20 8-bit general purpose registers. There is also a 16-bit timer and a 16-byte deep LIFO data stack which are accessible in the register address space. For more detailed information, see the specific sections on the Register set, the Timer, and the ALU.

The BCP can operate independently or with another processor as the host system. If such a system is required, communication with the BCP is possible by sharing data memory. The Remote Interface controls bus arbitration and access to data memory, as well as program up-loading and execution. For example, it is possible for a host system to load the BCP's instruction memory and begin program execution, then pass data back and forth through data memory accesses. The section on the Remote Interface and Arbitration System provides all of the necessary timing and control information to implement an interface between a BCP and a remote system.

As shown in *Figure 1-4,* the BCP uses two entirely separate memory systems, one for program storage and the other for data storage. This type of memory arrangement is referred to as Harvard architecture. Each system has 16 address lines, for a maximum of 64k words in each, and its own set of data lines. The instruction (program) memory is two bytes (16 bits) wide, and the data memory is one byte (8 bits) wide.

In order to reduce the number of pins required for these signals, the address and data lines for data memory are multiplexed together. This requires an external latch and the Address Latch Enable signal (ALE) for de-multiplexing.
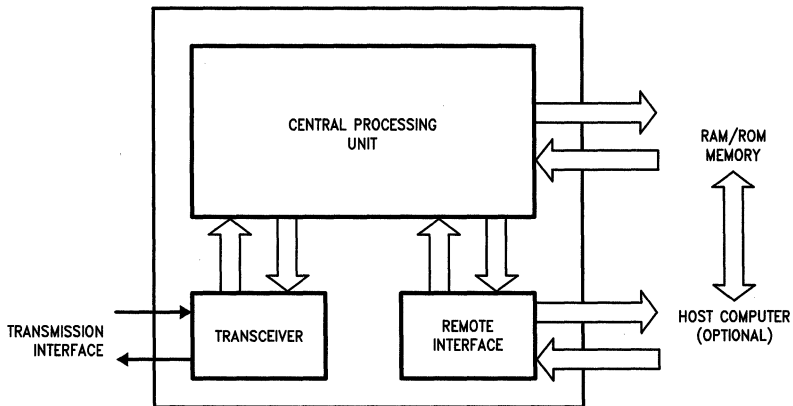
FIGURE 1-3. Simplified Block Diagram

TL/F/9336–B9

1.0 Communications Processor Introduction (Continued)

Simultaneous access to both data and program memory, and instruction pipelining greatly enhance the speed performance of the BCP, making it well suited for real-time processing. The pipeline allows the next instruction to be retrieved from program memory while the current instruction is being executed.

1.3 TIMING INTRODUCTION

The timing of all CPU operations, instruction execution and memory access is related to the CPU clock. This clock is usually generated by a crystal and the internal oscillator, with optional divide by two circuitry. The period of the resulting CPU clock is referred to as a T-state; for example, a 20 MHz CPU clock yields a 50 ns T-state. Most CPU functions, such as arithmetic and logical operations, shifts and rotates, and register moves, require only two T-states. Branching instructions and data memory accesses require three to four T-states.

Each memory system has a separate, programmable number of wait states to allow the use of slower memory devices. Instruction memory wait states are inserted into all instructions, as shown in *Figure 1-5,* thus they affect the overall speed of program execution. Instruction memory wait states can also apply when the Remote Interface is loading a program into instruction memory. Data memory wait states are only inserted into data memory access instructions, hence there is less degradation in overall program execution. Refer to the Timing section for detailed examples of all BCP instruction and data memory timing.
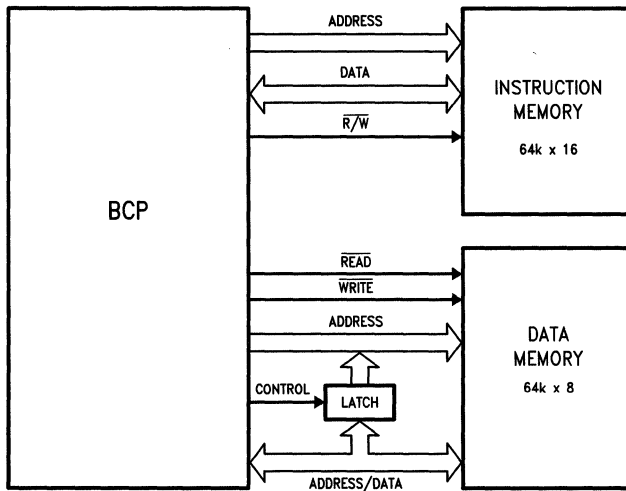
FIGURE 1-4. Memory Configuration

TL/F/9336–C1
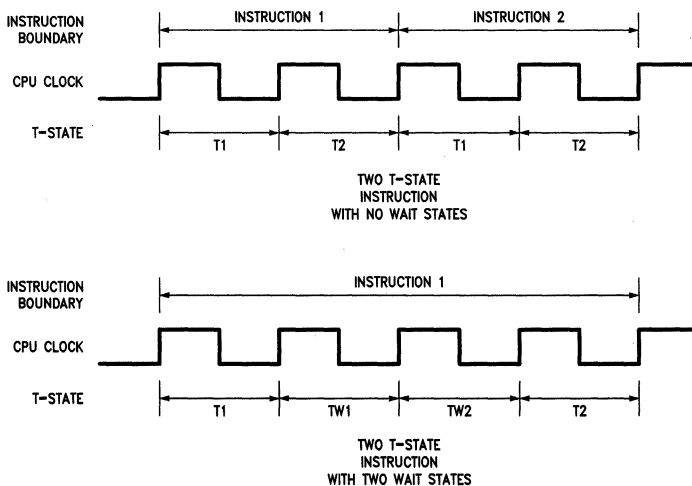
FIGURE 1-5. Effect of Memory Wait States on Timing

TL/F/9336–C2

1.0 Communications Processor Introduction (Continued)

1.4 DATA FLOW

The CPU registers are all dual port, that is, they have separate input and output paths. This arrangement allows a single register to function as both a source and a destination within the same instruction.

Figures 1-6a through *1-6f* show the internal data flow path for the BCP. The CPU registers are a central element to this path. When a register functions as an output, its contents are placed on the Source bus. When a register is an input, data from the Destination bus is written into that register.

The other key element in the data path is the ALU. This unit does all of the arithmetic and data manipulation operations, but it also has bus multiplexing capabilities. Both the Data Memory bus and a portion of the Instruction Memory bus are routed to this unit and serve as alternative sources of data. Since the data flow is always through this unit, most data moves may include arithmetic manipulations with no penalty in execution time.

Figure 1-6a shows the data path for all arithmetic instructions and register to register moves. The source register contents are placed on the Source bus, routed through the
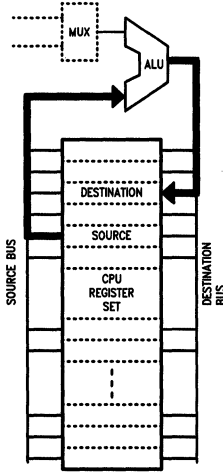
TL/F/9336-C3
FIGURE 1-6a. Register to Register
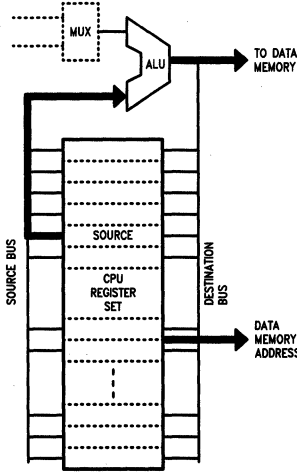
TL/F/9336-C4
FIGURE 1-6b. Data Memory WRITE
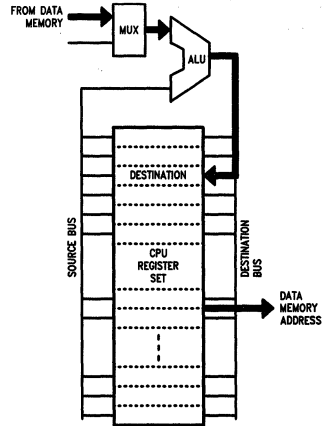
TL/F/9336-C5
FIGURE 1-6c. Data Memory READ
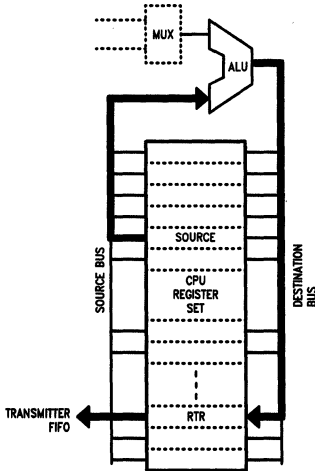
TL/F/9336-C6
FIGURE 1-6d. WRITE to Transmitter
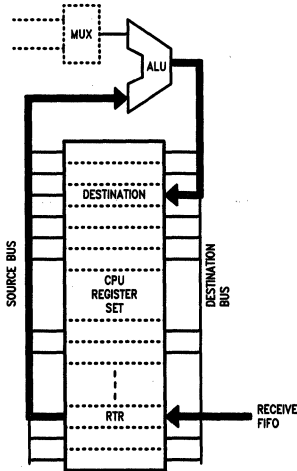
TL/F/9336-C7
FIGURE 1-6e. READ from Receiver
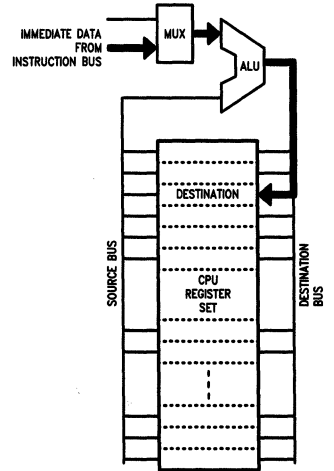
TL/F/9336-C8
FIGURE 1-6f. Load Immediate Data

1.0 Communications Processor Introduction (Continued)

ALU/MUX, and then placed on the destination bus. This data is then stored into the appropriate destination register.

Figures 1-6b and *1-6c* show the data path for data memory accesses. For a WRITE operation, the source register contents follow the same path through the ALU/MUX, but the Destination bus is routed to output pins and on to data memory. For a READ operation, incoming data is routed onto the Destination bus by the ALU/MUX, and then stored in a register. The address for all data memory accesses is provided by one of four 16-bit index registers which can operate in a variety of automatic increment and decrement modes.

Transfer of the data byte between the CPU and the Transceiver is accomplished through a register location. This register, {RTR}, appears as a normal CPU register, but writing to it automatically transfers data to the transmitter FIFO, and reading from it retrieves data from the receiver FIFO. These paths are illustrated in *Figures 1-6d* and *1-6e*.

It is also possible to load immediate data into a CPU register. This data is supplied by the program and is usually a constant such as a pointer or character. As shown in *Figure 1-6f*, a portion of the Instruction bus is routed through the ALU/MUX for this purpose.

1.5 REMOTE INTERFACE AND ARBITRATION SYSTEM INTRODUCTION

The BCP is designed to serve as a complete, stand alone communications interface. Alternately, it can be interfaced with another processor by means of the Remote Interface and Arbitration System. Communication between the BCP and the remote processor is possible by sharing data memory. Harvard architecture allows the remote system to access any BCP data memory location while the BCP continues to fetch and execute instructions, thereby minimizing performance degradation.

Figure 1-7 shows a simplified remote processor interface. This includes tri-state buffers on the address and data buses of the BCP's Data Memory, and all of the control and handshaking signals required to communicate between the BCP and the host system.

There is an 8-bit control register, Remote Interface Control {RIC}, accessible only to the remote system, which is used to control a variety of features, including the types of memory accesses, interface speeds, single step program execution, CPU start/stop, instruction memory loads, and so forth. Detailed information on all interface options is provided in the section on Remote Interface and Arbitration System, and in the related Reference section.
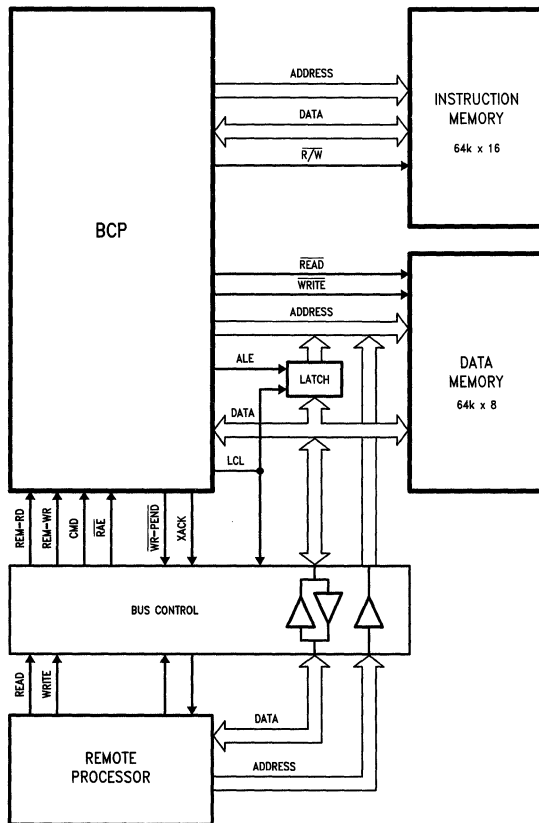
TL/F/9336-C9

FIGURE 1-7. Basic Remote Interface

2.0 CPU Description

The CPU is a general purpose, 8-bit microprocessor capable of 20 MHz operation. It contains a large register set for standard CPU operations and control of the transceiver. The reduced instruction set is optimized for the communications environment. The following sections are an architectural and functional description of the DP8344A CPU.

2.1 CPU ARCHITECTURAL DESCRIPTION

2.1.1 Register Set

This section describes the BCP's internal CPU registers. It is a general overview of the register structure and the functions mapped into the CPU register space. It is not a detailed or exhaustive description of every bit. For such a description, please refer to Section 6.2, Register Set Reference. Also, the Remote Interface Configuration register, {RIC}, is not accessible to the BCP (being accessible only by the remote system) and is described in Section 6.3, Remote Interface Reference.

The register set of the BCP provides for a compliment of both special function and general purpose registers. The special function registers provide access to on-chip peripherals (transceiver, timer, interrupt control, etc.) while the general purpose registers maximize CPU throughput by minimizing accesses to external data memory. The CPU can address a total of 44 8-bit registers, providing access to:

- 20 general purpose registers
- 8 configuration and control registers
- 4 transceiver access registers
- 2 8-bit accumulators
- 4 16-bit pointers
- 16-bit timer
- 16 byte data stack
- address and data stack pointers

The CPU addresses internal registers with a 5-bit field, addressing 32 locations generically named R0 through R31. The first twelve locations (R0–R11) are further organized by function as two groups of banked registers (A and B) as shown in *Figure 2-1.* Each group contains both a main and an alternate bank. Only one bank is active for group A and one for bank B and thus accessible during program execution. Switching between the banks is performed by the exchange instruction EXX which selects whether Main A or Alternate A occupies R0–R3 and whether Main B or Alternate B occupies R4–R11.
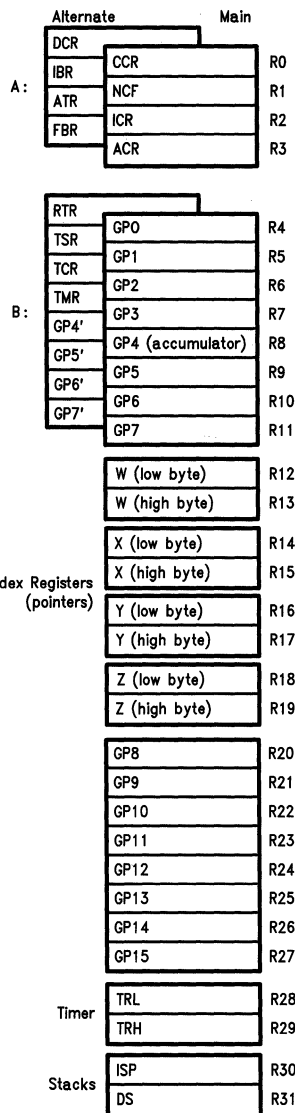
TL/F/9336–32

FIGURE 2-1. Register Map

2.0 CPU Description (Continued)

Registers in the R0–R11 address space are allocated in a manner that minimizes the need to switch banks:

Main A: CPU control and transceiver status

Alternate A: CPU and transceiver configuration

Main B: 8 general purpose

Alternate B: 4 transceiver access, 4 general purpose

Most of the BCP's instructions with register operand(s) can access all 32 register locations. Only instructions with an immediate operand are limited to the first sixteen register locations (R0–R15). These instructions, however, still have access to all registers required for transceiver operation, CPU status and control registers, 12 general purpose registers, and two of the index registers.

The general purpose registers are used for the majority of BCP operations. There are 8 general purpose registers in Main Bank B (R4–R11), 4 in Alternate Bank B (R8–R11), and 8 more (R20–R27) that are always accessible but are outside the limited register range. Since these registers are internal to the BCP, they can be accessed without data memory wait states, speeding up processing time. The index registers may also be used as general purpose registers if required.

For those instructions that require two operands, an accumulator (R8, one in each bank) serves as the second operand. The result of such an operation is stored back in the accumulator only if it is specified as the destination, thus allowing three operand operations such as R5 + R8 → R20. See Section 2.1.3 Instruction Set for further explanation.

Most registers have a predetermined state following a reset to the BCP. Refer to Section 6.2, Register Set Reference for a detailed summary.

2.1.1.1 Banked Registers

The CPU register set was designed to optimize CPU performance in an environment which supports multiple tasks. Generally the most important and time critical of these tasks will be maintaining the serial link (servicing the transceiver section) which often requires real time processing of commands and data. Therefore, all transceiver functions have been mapped into special function registers which the CPU can access quickly and easily. Switching between this task and other tasks has been facilitated by dedicating a register bank (Alternate B) to transceiver functions. Alternate Bank B provides access to all transceiver status, control, and data, in addition to four general purpose registers for protocol related storage. Main Bank B contains eight general purpose registers for use by other tasks. Having general purpose registers in both B banks allows for quick context switching and also helps eliminate some of the overhead of saving general purpose registers. The main objective of this banked register structure is to expedite servicing of the transceiver as a background (interrupt driven) task allowing the CPU to efficiently interleave that function with other background and foreground operations.

To facilitate using the transceiver in a polled fashion (instead of using interrupts), many of the status flags necessary to handshake with the transceiver are built into the conditional jump instructions, with others available in the Main A bank (normally active) so that Alternate Bank B does not have to be switched in to poll the transceiver. Timer and BIRQ tasks may also be run using polling techniques to Main A bank.

In general, the registers have been arranged within the banks so as to minimize the need to switch banks. The power-up state is Alternate bank A, Alternate bank B allowing access to configuration registers. Again, the banks switch by using the EXX instruction which explicitly specifies which bank is active (Main or Alternate) for each register group (A and B). The EXX instruction allows selecting any of four possible bank settings with a single two T-state instruction. This instruction also has the option of enabling or disabling the maskable interrupts.

The contents of the special function registers can be divided into several groups for general discussion—timing/control, interrupt control, the transceiver, the condition codes, the index registers, the timer, the stacks, and remote interface.

2.1.1.2 Timing/Control Registers

The BCP provides a means to configure its external timing through setting bits in the Device Control Register, {DCR}, and the Auxiliary Control Register, {ACR}. One of the first configuration registers to be initialized on power-up/reset is {DCR} which defines the hardware environment in which the BCP is functioning. Specifically, {DCR} controls the clock select logic for both the CPU and transceiver, in addition to the number of wait states to be used for instruction and data memory accesses.

The BCP allows either one clock source operation for the CPU and the transceiver from the on-chip oscillator, or an independent clock source can run the transceiver from the eXternal Transceiver CLocK input, X-TCLK. The Transceiver Clock Select bits, [TCS1,0], select the clock source for the transceiver which is either the on-chip Oscillator CLocK, OCLK, or X-TCLK. Options for selecting divisions of the on-chip oscillator frequency are also provided (see the description of {DCR} in Section 6.2, Register Set Reference. The CPU Clock Select bit, [CCS], allows the CPU to run at the OCLK frequency or at half that speed. The clock output at the pin CLK-OUT, however, is never divided and always reflects the crystal frequency OCLK. The frequency selected for the transceiver (referred to as TCLK) should always be eight times the desired serial data rate. The frequency selected for the CPU defines the length of each T-state (e.g., 20 MHz implies 50 ns T-states).

There are two independent fields for defining wait states, one for instruction memory access (n_{IW}) and one for data memory access (n_{DW}). These fields specify to the BCP how many wait states to insert to meet the access time requirements of both memory systems. The Instruction memory Wait-state select bits, [IW1,0], and the Data memory Wait-state select bits, [DW2–0], control the number of inserted wait states for instruction and data memory, respectively.

After a reset, the maximum number of wait states are set in {DCR}, n_{IW} = 3 T-states and n_{DW} = 7 T-states. Wait-states are discussed in more detail in Section 2.2.2, Timing. For a complete discussion on choosing your memory and determining the number of wait states required, please refer to the application note *Choosing Your RAM for the Biphase Communication Processor.*

2.0 CPU Description (Continued)

Another control bit in the {ACR} register is the Clock Out Disable bit, [COD]. When [COD] is asserted, the buffered clock output at pin CLK-OUT is tri-stated.

2.1.1.3 Interrupt Control Registers

The configuration bank (Alternate Bank A) includes an Interrupt Base Register, {IBR}, which defines the high byte of all interrupt and trap vector addresses. Thus, the interrupt vector table can be located in any 256 byte page of the 64k range of instruction addresses. The interrupt base is normally initialized once on reset before interrupts are enabled or any traps are executed. Since $\overline{NMI}$ is nonmaskable and may occur before {IBR} is initialized, the power-up/reset value of {IBR} (00h) should be used to accommodate $\overline{NMI}$ during initialization. In other words, if $\overline{NMI}$ is used in the system, the absolute address 001Ch (the $\overline{NMI}$ vector) should contain a jump to an $\overline{NMI}$ service routine.

The Interrupt Control Register, {ICR}, provides individual masks [IM4–0] for each of the maskable interrupts. The Global Interrupt Enable bit, [GIE], located in {ACR} works in conjunction with these individual masks to control each of the maskable interrupts.

The external pin called $\overline{BIRQ}$ is a Bidirectional Interrupt ReQuest. $\overline{BIRQ}$ is defined as an input or an output by the Bidirectional Interrupt Control bit, [BIC], in {ACR}. [IM3] functions as BIRQ's interrupt mask if $\overline{BIRQ}$ is an input as defines by [BIC]. When [BIC] defines BIRQ as an output, [IM3] controls the output state of $\overline{BIRQ}$.

Section 2.2.3, Interrupts provides a further description of these registers.

2.1.1.4 Timer Registers

The timer block interfaces with the CPU via two registers, TimeR Low byte, {TRL}, and TimeR High byte, {TRH}, which form the input/output ports to the timer. Writing to {TRL} and {TRH} stores the low and high byte, respectively, of a 16-bit time-out value into two holding registers. The word stored in the holding registers is the value that the timer will be loaded with via [TLD]. Also, the timer will automatically reload this word upon timing out. Reading {TRL} and {TRH} provides access to the count down status of the timer.

Control of timer operation is maintained via three bits in the Auxiliary Control Register {ACR}. Timer STart [TST], bit 7 in {ACR}, is the start/stop control bit. Writing a one to [TST] allows the timer to start counting down from its current value. When low, the timer stops and the timer interrupt is cleared. Timer Load [TLD], bit 6 in {ACR}, is the load control of the timer. After writing the desired values into {TRL} and {TRH}, writing a one to [TLD] will load the 16-bit word in the holding registers into the timer and initialize the timer clock to zero in preparation to start counting. Upon completing the load operation, [TLD] is automatically cleared. Timer Clock Selection [TCS], bit 5 in {ACR}, determines the clock frequency of the timer count down. When low, the timer divides the CPU clock by sixteen to form the clock for the down counter. When [TCS] is high, the timer divides the CPU clock by two. The input clock to the timer is the CPU clock and should not be confused with the oscillator clock, OCLK. The rate of the CPU clock will be either equal to OCLK or one-half of OCLK depending on the value of bit 7 in the Device Control Register, {DCR}.

When the timer reaches a count of zero, the timer interrupt is generated, the Time Out flag, [TO], (bit 7 in the Condition Code Register {CCR}), goes high, and the timer reloads the 16-bit word stored in the holding registers to recycle through a count down. The timer interrupt and [TO] can be cleared by either writing a one to [TO] in {CCR} or stopping the timer by writing a zero to [TST] in {ACR}. Refer to Section 2.1.2, Timer for more information on the timer operation.

2.1.1.5 Transceiver Registers

Two registers in the Alternate A bank initialize transceiver functions. The Auxiliary Transceiver Register, {ATR}, specifies a station address used by the address recognition logic within the transceiver when using the non-promiscuous 5250 and 8-bit protocol modes. In 5250 modes, {ATR} also defines how long the TX-ACT pin stays asserted after the end of a transmitted message. The Fill Bit Register, {FBR}, specifies the number of optional fill bits inserted between frames in a multiframe 5250 message.

{ICR} contains the Receiver Interrupt Select bits, [RIS1,0]. These bits determine the receiver interrupt source selection. The source may be either Receiver FIFO Full, Data Available, or Receiver Active.

The Receive/Transmit Register, {RTR}, is the input/output port to both the transmitter and receiver FIFO's. It appears to the BCP CPU like any other register. The {RTR} register provides the least significant eight bits of data in both received and transmitted messages.

The Transceiver Mode Register, {TMR}, contains bits used to set the configuration of the transceiver. As long as the Transceiver RESet bit, [TRES], is high, the transceiver remains in reset. Internal LOOP-back operation of the transceiver can be selected by asserting [LOOP]. The RePeat ENable bit, [RPEN], allows the receiver to be active at the same time as the transmitter. When the Receiver INvert bit, [RIN], is set, all data sent to the receiver is inverted. The Transmitter INvert bit, [TIN], is analogous to [RIN] except it is for the transmitter. The protocol that the transceiver is using is selected with the Protocol Select bits, [PS2–0].

The Transceiver Command Register, {TCR}, controls the workings of the transmitter. To generate 5.5 line quiesce pulses at the start of a transmission rather than 5, the Advance Transmitter Active bit, [ATA], must be set high. Parity is automatically generated on a transmission and the Odd Word Parity bit, [OWP], determines whether that parity is even or odd. Bits 2–0 of {TCR} make up part of the Transmitter FIFO [TF10–8] along with {RTR}. Whenever a write is made to {RTR}, [TF10–8] are automatically pushed on the FIFO with the 8 bits written to {RTR}.

Other bits in {TCR} control the operation of the on-chip receiver. The number of line quiesce bits the receiver must detect to recognize a valid message is determined by the Receive Line Quiesce bit, [RLQ]. The BCP has its own internal analog comparator, but an off-chip one may be connected to DATA-IN. The receiver source is determined by the Select Line Receiver bit, [SLR]. To view transceiver errors in the Error Code Register, {ECR}, the Select Error Codes, [SEC], bit in {TCR} must be set high. When [SEC] is high, Alternate Bank B R4 is remapped from {RTR} to {ECR} so that {ECR} can be read.

2.0 CPU Description (Continued)

Just as [TF10–8] bits get pushed onto the transmitter FIFO when a write to {RTR} occurs, the Receiver FIFO bits, [RF10–8], in the Transceiver Status Register, {TSR}, reflect the state of the top word of the receive FIFO. {TSR} also contains flags that show Transmit FIFO Full, [TFF], Transmitter Active, [TA], Receiver Error, [RE], Receiver Active, [RA], and Data AVailable, [DAV]. These flags may be polled to determine the state of the transceiver. For instance, during a Receiver Active interrupt, the BCP can query the [DAV] bit to determine whether data is ready in the receiver FIFO yet.

The Error Code Register, {ECR}, contains flags for receiver errors. As previously stated, the [SEC] bit in {TRC} must be set high to read this register. Reading {ECR} or resetting the transceiver with [TRES] will clear all the errors that are present. The receiver OVerFlow flag, [OVF], is set when the receiver attempts to add another word to the FIFO when it is full. If internally checked parity and parity transmitted with a 3270 message conflict, then the PARity error bit, [PAR], is set high. The Invalid Ending Sequence bit, [IES], is set when the ending sequence in a 3270, 3299, or 8-bit message is incorrect. When the expected mid-bit transition in the Manchester waveform does not occur, a Loss of Mid-Bit Transition occurs ([LMBT]). Finally, if the transmitter is activated while the receiver is active, the Receiver DISabled while active flag, [RDIS], will be set unless [RPEN] is asserted.

The second register in Main A bank is called the Network Command Flag register, {NCF}, and contains information about the transceiver which is useful for polling the transceiver (during other tasks for example) to see if it needs servicing. These flags include bits to indicate Transmit FIFO Empty [TFE], Receive FIFO Full [RFF], Line Active [LA], and a Line Turn Around [LTA]. [LTA] indicates that a message has been received without error and a valid ending sequence has occurred. These flags facilitate polling of the transceiver section when transceiver interrupts are not used. Also included in this register is a bit called [DEME] (Data Error/Message End). In 3270/3299 modes, this bit indicates a mismatch between received and locally generated byte parity. In 5250 modes, [DEME] decodes an end of message indicator (111 in the address field). Three other bits: Received Auto Response [RAR], Acknowledge [ACK] and Poll [POLL] are decoded from a received message (at the output of the receive FIFO) and are valid only in 3270/3299 modes where response time is critical.

Section 3.0 Transceiver provides comprehensive coverage of this on-chip peripheral.

2.1.1.6 Condition Codes/Remote Handshaking Register

The ALU condition codes are available in the Condition Code Register {CCR}. The [Z] bit is set when a zero result is generated by an arithmetic, logical, or shift instruction. Similarly, [N] indicates the Negative result of the same operations. An oVerflow condition from an arithmetic instruction sets the [V] bit in {CCR}. The Carry bit [C] indicates a carry or borrow result from an arithmetic instruction. See Section 2.2.2, ALU for more information.

The Condition Code Register, {CCR}, also contains [BIRQ], a status bit which reflects the logic level of the bidirectional interrupt input pin BIRQ. Hence, this pin can be used as a general purpose input/output port as well as a bidirectional

interrupt request as defined by bits in {ACR} and {ICR}. If a remote CPU is present and shares data memory (dual port memory) with the BCP, handshaking can be accomplished by using the two status bits in {CCR} called [RR] and [RW], which indicate Remote Read and Remote Write accesses, respectively.

In {ACR}, a lock bit, [LOH], is available to lock out all host accesses. When this bit is set, all host accesses are disabled. Locking out remote accesses is often done during interrupts to ensure quick response times.

The Remote Interface Configuration register, {RIC}, is not available to the BCP internally. The Remote Interface Reference section provides further detail on {RIC} and interfacing a remote processor.

2.1.1.7 Index Registers

Four index registers called IW, IX, IY, and IZ provide 16-bit addressing for both data memory and instruction memory. Each of these index registers is actually a pair of 8-bit registers which are individually addressable just like any other CPU register. They occupy register addresses R12 through R19. Thus, the first two pointers IW and IX (comprising R12–R15) can be accessed with immediate mode instructions (which can access only R0 to R15). Refer to Section 2.1.3.2, Addressing Modes to see how the index registers are formed from R12–R19.

Accessing data memory requires the use of one of the four index registers. All such instructions allow you to specify which pointer is to be used, except the immediate-relative moves: MOVE rs,[IZ+n] and MOVE [IZ+n],rd. These instructions always use the IZ pointer. Register indirect operations have options to alter the value of the index register; the options include pre-increment, post-increment, and post-decrement. These options facilitate block moves, searches, etc. Refer to Section 2.1.3, Instruction Set for more information about data moves.

Since the BCP's ALU is 8 bits wide, all code that manipulates the index registers must act on them eight bits at a time.

The index registers can also be used in register indirect jumps (LJMP [lr]), useful in implementing relocatable code. Any one of the index registers can be specified to provide the 16-bit instruction address for the indirect jump.

2.1.1.8 Stack Registers

The last two register addresses (R30,R31) are dedicated to provide access to the two on-chip stacks—the data stack and the address stack. The data stack is 8 bits wide and 16 words deep. It is a Last In First Out (LIFO) type and provides high speed storage for variables, pointers, etc. The address stack is 23 bits wide and 12 words deep, providing twelve levels of nesting of subroutines and interrupts. It is also a LIFO structure and stores processor status as well as return addresses from CALL instructions, TRAP instructions, and interrupts. The seven bits of processor status consist of the four ALU flags, ([C], [N], [V], and [Z]), the current bank setting (two bits), and [GIE].

Stack pointers for both the on-chip stacks are provided in R30, the Internal Stack Pointer register, {ISP}. The lower four bits are the pointer for the data stack and the upper four bits are the pointer for the address stack. Both internal stacks are circular. For example if 16 bytes are written to

2

2.0 CPU Description (Continued)

the data stack, the next byte pushed will overwrite the first. {ISP} can be read and written to like any other register, but after a write, the BCP must execute one instruction before reading the stack whose pointer was modified.

The Data Stack register, {DS}, is the input/output port for the data stack. This port is accessed like any other register, but a write to it will "push" a byte onto the stack and a read from it will "pop" a byte from the stack. The data stack pointer is updated when a read or write of {DS} occurs.

Information bits in the instruction address stack are not mapped into the CPU's register space and, therefore, are not directly accessible. A remote system running a monitor program can access this information by forcing the BCP to single-step through a return instruction and then reading the program counter. Since the stack pointers are writeable, the remote system can access any location (return address) in the address stack to trace program flow and then restore the stack pointer to its original position.

2.1.2 Timer

The BCP has an internal 16-bit timer that can be used in a variety of ways. The timer counts independently of the CPU, eliminating the waste of valuable processor bandwidth. The timer can be used in a polled or interrupt driven configuration for user software flexibility.

The timer interfaces with the CPU via two registers, TimeR Low byte, {TRL}, and TimeR High byte, {TRH}, which form the input/output ports to the timer. Writing to {TRL} and {TRH} stores the low and high byte, respectively, of a 16-bit time-out value into two holding registers. The word stored in the holding registers is the value that the timer will be load-

ed with via [TLD]. Also, the timer will automatically reload this word upon timing out. Reading {TRL} and {TRH} provides access to the count down status of the timer.

Control of timer operation is maintained via three bits in the Auxiliary Control Register {ACR}. Timer STart [TST], bit 7 in {ACR}, is the start/stop control bit. Writing a one to [TST] allows the timer to start counting down from its current value. When low, the timer stops and the timer interrupt is cleared. Timer Load [TLD], bit 6 in {ACR}, is the load control of the timer. After writing the desired values into {TRL} and {TRH}, writing a one to [TLD] will load the 16-bit word in the holding registers into the timer and initialize the timer clock to zero in preparation to start counting. Upon completing the load operation, [TLD] is automatically cleared. Timer Clock Selection [TCS], bit 5 in {ACR}, determines the clock frequency of the timer count down. When low, the timer divides the CPU clock by sixteen to form the clock for the down counter. When [TCS] is high, the timer divides the CPU clock by two. The input clock to the timer is the CPU clock and should not be confused with the oscillator clock, OCLK. The rate of the CPU clock will be either equal to OCLK or one-half of OCLK depending on the value of bit 7 in the Device Control Register, {DCR}.

When the timer reaches a count of zero, the timer interrupt is generated, the Time Out flag, [TO], (bit 7 in the Condition Code Register {CCR}), goes high, and the timer reloads the 16-bit word stored in the holding registers to recycle through a count down. The timer interrupt and [TO] can be cleared by either writing a one to [TO] in {CCR} or stopping the timer by writing a zero to [TST] in {ACR}. A block diagram of the timer is shown in *Figure 2-2*.
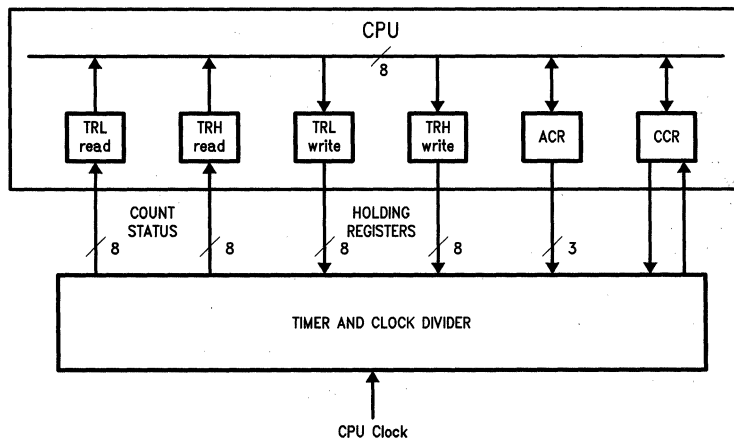
TL/F/9336–D1

FIGURE 2-2. Timer Block Diagram

2.0 CPU Description (Continued)

2.1.2.1 Timer Operation

After the desired 16-bit time-out value is written into {TRL} and {TRH}, the start, load, and clock selection can be achieved in a single write to {ACR}. A restriction exists on changing the timer clock frequency in that [TCS] should not be changed while the timer is running (i.e., [TST] is high). After a write to {ACR} to load and start the timer, the timer begins counting down at the selected frequency from the value in {TRL} and {TRH}. Upon reaching a count of zero, the timer interrupt is generated and, the timer reloads the current word from {TRL} and {TRH} to cycle through a countdown again. The timing waveforms shown in *Figure 2-3* show a write to {ACR} that loads, starts, selects CPU clock rate/2 for the countdown rate, and asserts the Global Interrupt Enable [GIE]. Prior to the write to {ACR}, {TRL} and {TRH} were loaded with 00h and 01h respectively, the timer interrupt was unmasked in the Interrupt Control Register {ICR} by clearing bit 4, and zero instruction wait states were selected in {DCR}. Since the write to {ACR} asserted [GIE], the timer interrupt is enabled and the CPU will vector to the timer interrupt service routine address when the timer reaches a count of zero. The timer interrupt is the lowest priority interrupt and is latched and maintained until it is cleared in software. (See CPU Interrupts section). For very long time intervals, time-outs can be accumulated under software control by writing a one to [TO] in {CCR} allowing the timer to recycle its count down with no other intervention. For time-outs attainable with one count down, stopping the timer will clear the interrupt and [TO]. When the timer interrupt is enabled, the call to the interrupt service routine occurs at different instruction boundaries depending on when the timer interrupt occurs in the instruction cycle. If the timer times out prior to T2, where T2 is the last T-state of an instruction cycle, the call to the interrupt service routine will occur in the next instruction. When the time-out occurs in T2, the call to the interrupt service routine will not occur in the next instruction. It occurs in the second instruction following T2.

The count status of the timer can be monitored by reading {TRL} and/or {TRH}. When the registers are read, the output of the timer, not the value in the input holding registers, is presented to the ALU. Some applications might require monitoring the count status of the timer while it is counting down. Since the timer can time-out between reads of {TRL} and {TRH}, the software should take this fact into consideration. To read back what was written to {TRL} and {TRH}, the timer must first be loaded via [TLD] without starting the timer followed by a one instruction delay before reading {TRL} and {TRH} to allow the output registers to be updated from the load operation.

To determine the time-out delay for a given value in {TRL} and {TRH} other than 0000h, the following equation can be used:

$$TD = (\text{value in } \{TRH\}\{TRL\}) * T * k$$

where:

k = 2 when [TCS] = 1 or 16 when [TCS] = 0

T = The period of the CPU clock

TD = The amount of time delay after the end of the instruction that asserts [TST] in {ACR}

When the value of 0000h is loaded in the timer, the maximum time-out is obtained and is calculated as follows:

$$TD = 65536 * T * k$$

With the CPU running full speed with an 18.8 MHz crystal, the maximum single loop time delay attainable would be 55.6 ms ([TCS] = 0). The minimum time delay with the same constraints is 106 ns ([TCS] = 1). For accumulating time-out intervals, the total time delay is simply the number of loops accumulated multiplied by the calculated time delay. The equations above do not account for any overhead for processing the timer interrupt. The added overhead of processing the interrupt may need to be included for precision timing.

2

TL/F/9336–D2
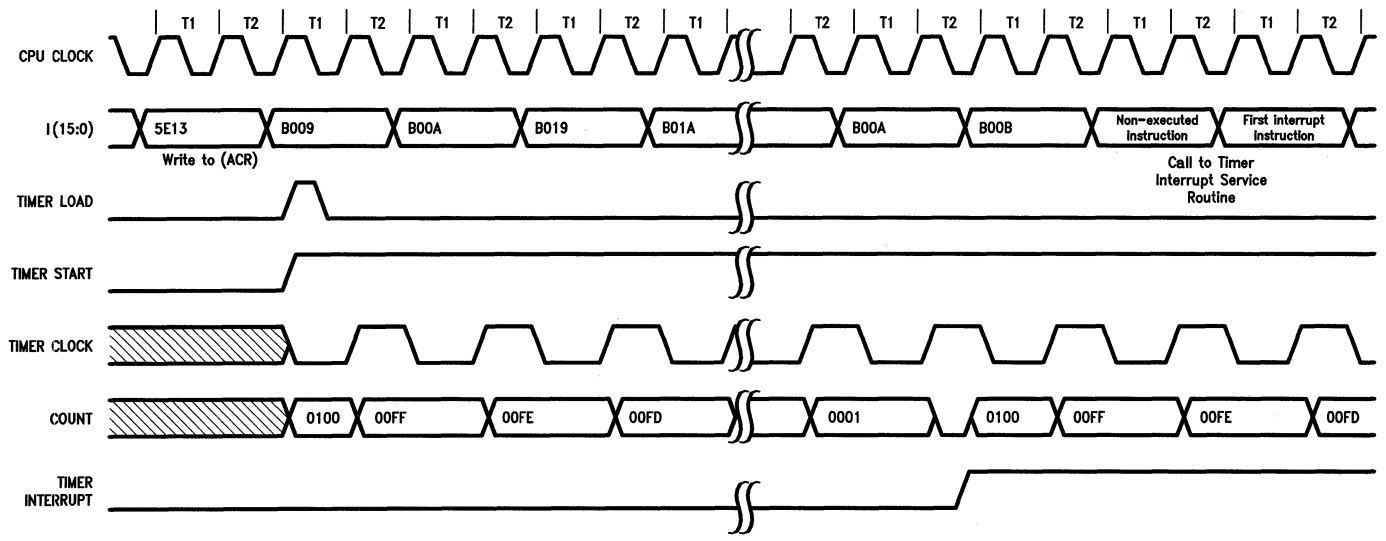
FIGURE 2-3. Timer Interrupt Diagram

2.0 CPU Description (Continued)

2.1.3 Instruction Set

The followng paragraphs introduce the BCP's architecture by discussing addressing modes and briefly discussing the Instruction Set. For detailed explanations and examples of each instruction, refer to the Instruction Set Reference Section.

2.1.3.1 Harvard Architecture Implications

The BCP utilizes a true Harvard Architecture, where the instruction and data memory are organized into two independent memory banks, each with their own address and data buses. Both the Instruction Address Bus and the Instruction Bus are 16 bits wide with the Instruction Address Bus addressing memory by words. (A word of memory is 16 bits long; i.e., 1 word = 2 bytes.) Most of the instructions are one word long. The exceptions are two words long, containing a word of instruction followed by a word of immediate data. The combination of word sized instructions and a word based instruction address bus eliminates the typical instruction alignment problems faced by many CPU's.

The Data Address Bus is 16 bits wide (with the low order 8 bits multiplexed on the Data Bus), and the Data Bus is 8 bits wide (i.e., one byte wide). The Data Address Bus addresses memory by bytes. Most of the BCP's instructions operate on byte-sized operands.

Note that although both instruction addresses and data addresses are 16 bits long, these addresses are for two different buses and, therefore, have two different numerical meanings, (i.e., byte address or word address.) Each instruction determines whether the meaning of a 16-bit address is that of an instruction word address or a data byte address. Little confusion exists though because only the program flow instructions interpret 16-bit addresses as instruction addresses.

2.1.3.2 Addressing Modes

An addressing mode is the mechanism by which an instruction accesses its operand(s). The BCP's architecture supports five basic addressing modes: register, immediate, indexed, immediate-relative, and register-relative. The first two allow instructions to execute the fastest because they require no memory access beyond instruction fetch. The remaining three addressing modes point to data or instruction memory. Typical of a RISC processor, most of the instructions only support the first three addressing modes, with one of the operands always limited to the register addressing mode.

Register Addressing Modes

There are two terminologies for the register addressing modes: Register and Limited Register. Instructions that allow Register operands can access all the registers in the CPU. Note that only 32 of the 44 CPU registers are available at any given point in time because the lower 12 register locations (R0–R11) access one of two switchable register banks each. (See Section 2.1.1.1, Banked Registers for more information on the CPU register banks.) Instructions that allow the Limited Register operands can access just the first 28 registers of the CPU. Again, note that only 16 of these 28 registers are available at any given point in time. Table 2-1 shows the notations used for the Register and Limited Register operands. Some instructions also imply the use of certain registers, for example the accumulators. This is noted in the discussions of those instructions.

Immediate Addressing Modes

The two types of the immediate addressing modes available are: Immediate numbers and Absolute numbers. Immediate numbers are 8 bits of data, (one data byte), that code directly into the instruction word. Immediate numbers may represent data, data address displacements, or relative instruction addresses. Absolute numbers are 16-bit numbers. They code into the second word of two word instructions and they represent absolute instruction addresses. Table 2-2 shows the notations used for both of these addressing modes.

TABLE 2-1. Register Addressing Mode Notations

Notation	Type of Register Operand	Registers Allowed
Rs	Source Register	R0–R31
Rd	Destination Register	R0–R31
Rsd	Register is both a Source & Destination	R0–R31
rs	Limited Source Register	R0–R15
rd	Limited Destination Register	R0–R15
rsd	Limited Register is both a Source & Destination	R0–R15

TABLE 2-2. Immediate Addressing Mode Notations

Notation	Type of Immediate Operand	Size
n	Immediate Number	8 Bits
nn	Absolute Number	16 Bits

2.0 CPU Description (Continued)

Indexed Addressing Modes

Indexed operands involve one of four possible CPU register pairs referred to as the index registers. *Figure 2-4* illustrates how the index registers map into the CPU Register Set. Note that the index registers are 16 bits wide.

Index registers allow for indirect memory addressing and usually contain data memory addresses, although, the LJMP instruction can use index registers to hold instruction memory addresses. Most of the instructions that allow memory indirect addressing, (i.e. the use of index registers), also allow pre-incrementing, post-incrementing, or post-decrementing of the index register contents during instruction execution, if desired. Table 2-3 lists the notations used for the index register modes.

Index Register	CPU Register Pair Forming Index Register (MSB) (LSB)

IW

R13	R12

15   8 7   0

IX

R15	R14

15   8 7   0

IY

R17	R16

15   8 7   0

IZ

R19	R18

15   8 7   0
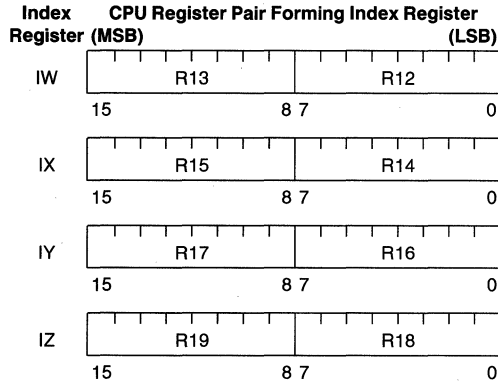
FIGURE 2-4. Index Register Map

Immediate-Relative and Register-Relative Address Modes

The Immediate-Relative mode adds an unsigned 8-bit immediate number to the index register IZ forming a data byte address. The Register-Relative mode adds the unsigned 8-bit value in the current accumulator, A, to any one of the index registers forming a data byte address. Both of these indirect memory addressing modes are available only on the MOVE instruction. Table 2-4 shows the notation used for these two addressing modes.

2.1.3.3 Instruction Set Overview

The BCP's RISC instruction set contains seven categories of instructions: Data Movement, Integer Arithmetic, Logic, Shift-Rotate, Comparison, Program Flow, and Miscellaneous.

Data Movement Instructions

The MOVE instruction is responsible for all the data transfer operations that the BCP can perform. Moving one byte at a time, five different types of transfer are allowed: register to register, data memory to register, register to data memory, instruction memory to register, and instruction memory to data memory. Table 2-5 lists all the variations of the MOVE instruction.

TABLE 2-3. Index Register Addressing Mode Notations

Notation	Meaning
[Ir]	Index Register, Contents Not Changed
[Ir−]	Index Register, Contents Post-Decremented
[Ir+]	Index Register, Contents Post-Incremented
[+Ir]	Index Register, Contents Pre-Incremented
[mIr]	General Notation Indicating that Any of the Above Modes Is Allowed

Note: [ ] denotes indirect memory addressing and is part of the instruction syntax.

TABLE 2-4. Relative Index Register Mode Notations

Notation	Type of Action Performed to Calculate a Data Memory Address
[IZ + n]	IZ + Immediate Number (unsigned) $\longrightarrow$ Data Memory Address
[Ir + A]	Index Register + Current Accumulator (unsigned) $\longrightarrow$ Data Memory Address

Note: [ ] denotes indirect memory addressing and is part of the instruction syntax.

TABLE 2-5. Data Movement Instructions

Syntax	Instruction Operation	Addressing Modes
MOVE Rs, Rd	register $\longrightarrow$ register	Register, Register
MOVE Rs, [mIr]	register $\longrightarrow$ data memory	Register, Indexed
MOVE [mIr], Rd	data memory $\longrightarrow$ register	Indexed, Register
MOVE Rs, [Ir + A]	register $\longrightarrow$ data memory	Register, Register-Relative
MOVE [Ir + A], Rd	data memory $\longrightarrow$ register	Register-Relative, Register
MOVE rs, [IZ + n]	register $\longrightarrow$ data memory	Limited Register, Immediate-Relative
MOVE [IZ + n], rd	data memory $\longrightarrow$ register	Immediate-Relative, Limited Register
MOVE n, rd	instruction memory $\longrightarrow$ register	Immediate, Limited Register
MOVE n, [Ir]	instruction memory $\longrightarrow$ data memory	Immediate, Indexed

2.0 CPU Description (Continued)

Integer Arithmetic Instructions

The integer arithmetic instructions operate on 8-bit signed (two's complement) binary numbers. Two arithmetic functions are supported: Add and Subtract. Three versions of the Add and Subtract instructions exist: operand ± accumulator, operand ± accumulator ± carry, and immediate operand ± operand. The first two versions support both the register and indexed addressing modes for the destination operand. These two versions also allow the specification of a separate register or data address for the destination operand and so that the sources may retain their integrity; (i.e., true three-operand instructions). Note that the currently active "B" register bank selects which accumulator is used in these instructions. The third version, immediate operand ± operand, only supports the register addressing mode for the destination operand with the register as both a source and the destination. Table 2-6 lists the integer arithmetic instructions along with their variations.

Logic Instructions

The logic instructions operate on 8-bit binary data. A full set of logic functions is supported by the BCP: AND, OR, eXclusive OR, and Complement. All the logic functions except complement allow either an immediate operand or the currently active accumulator as an implied operand. Complement only allows one register operand which is both the source and destination. The other logic instructions include the following addressing modes: register, indexed, and immediate. As with the integer arithmetic instructions, the integrity of the sources may be maintained by specifying a destination register which is different from the source. Table 2-7 lists all the logic instructions.

TABLE 2-6. Integer Arithmetic Instructions

Syntax		Instruction Operation	Addressing Modes
ADD	n, rsd	register + n $\rightarrow$ register	Immediate, Limited Register
ADDA	Rs, Rd	Rs + accumulator $\rightarrow$ Rd	Register, Register
ADDA	Rs, [mlr]	Rs + accumulator $\rightarrow$ data memory	Register, Indexed
ADCA	Rs, Rd	Rs + accumulator + carry $\rightarrow$ Rd	Register, Register
ADCA	Rs, [mlr]	Rs + accumulator + carry $\rightarrow$ data memory	Register, Indexed
SUB	n, rsd	register − n $\rightarrow$ register	Immediate, Limited Register
SUBA	Rs, Rd	Rs − accumulator $\rightarrow$ Rd	Register, Register
SUBA	Rs, [mlr]	Rs − accumulator $\rightarrow$ data memory	Register, Indexed
SBCA	Rs, Rd	Rs − accumulator − carry $\rightarrow$ Rd	Register, Register
SBCA	Rs, [mlr]	Rs − accumulator − carry $\rightarrow$ data memory	Register, Indexed

TABLE 2-7. Logic Instructions

Syntax		Instruction Operation	Addressing Modes
AND	n, rsd	register & n $\rightarrow$ register	Immediate, Limited Register
ANDA	Rs, Rd	Rs & accumulator $\rightarrow$ Rd	Register, Register
ANDA	Rs, [mlr]	Rs & accumulator $\rightarrow$ data memory	Register, Indexed
OR	n, rsd	register | n $\rightarrow$ register	Immediate, Limited Register
ORA	Rs, Rd	Rs | accumulator $\rightarrow$ Rd	Register, Register
ORA	Rs, [mlr]	Rs | accumulator $\rightarrow$ data memory	Register, Indexed
XOR	n, rsd	register $\oplus$ n $\rightarrow$ register	Immediate, Limited Register
XORA	Rs, Rd	Rs $\oplus$ accumulator $\rightarrow$ Rd	Register, Register
XORA	Rs, [mlr]	Rs $\oplus$ accumulator $\rightarrow$ data memory	Register, Indexed
CPL	Rsd	$\overline{\text{register}}$ $\rightarrow$ register	Register

Note: & = logical AND operation
 | = logical OR operation
 $\oplus$ = logical exclusive OR operation
 $\bar{r}$ = one's complement

2.0 CPU Description (Continued)

Shift and Rotate Instructions

The shift and rotate instructions operate on any of the 8-bit CPU registers. The BCP supports shift left, shift right, and rotate operations. Table 2-8 lists the shift and rotate instructions.

Comparison Instructions

The BCP utilizes two comparison instructions. The CMP instruction performs a two's complement subtraction between a register and immediate data. The BIT instruction tests selected bits in a register by ANDing it with immediate data. Neither instruction stores its results, only the ALU flags are affected. Table 2-9 lists both of the comparison instructions.

Program Flow Instructions

The BCP has a wide array of program flow instructions: unconditional jumps, calls and returns; conditional jumps, calls, and returns; relative or absolute instruction addressing on jumps and calls; a specialized register field decoding
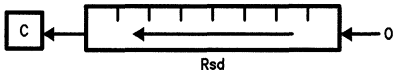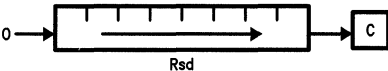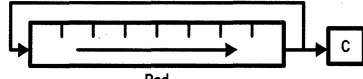
jump; and software interrupt capabilities. These instructions redirect program flow by changing the Program Counter.

The unconditional jump instructions support both relative instruction addressing, the (JuMP instruction), and absolute instruction addressing, (the Long JuMP instruction), using the following addressing modes: Immediate, Register, Absolute, and Indexed. Table 2-10 lists the unconditional jump instructions and their variations.

The conditional jump instructions support both relative instruction addressing and absolute instruction addressing using the Immediate and Absolute addressing modes. The conditional relative jump instruction tests flags in the Condition Code Register, {CCR}, and the Transceiver Status Register, {TSR}. Two possible syntaxes are supported for the conditional relative jump instruction; see Table 2-11.

Table 2-12 lists the various flags "f" that the conditional JMP instruction can test and Table 2-13 lists the various conditions "cc" that the Jcc instruction can test for. Keep in

TABLE 2-8. Shift and Rotate Instructions

Syntax	Instruction Operation	Addressing Mode
SHL Rsd,b		Register
SHR Rsd,b		Register
ROT Rsd,b		Register

Note: "b" = the number of bit shifts/rotates to perform.

TABLE 2-9. Comparison Instructions

Syntax	Instruction Operation	Addressing Mode
CMP rs, n	register − n	Limited Register
BIT rs, n	register & n	Limited Register

Note: & = logical AND operation

TABLE 2-10. Unconditional Jump Instructions

Syntax	Instruction Operation	Operand Range	Addressing Mode
JMP n	PC + n (sign extended) → PC	−128, +127	Immediate
JMP Rs	PC + Rs (sign extended) → PC	−128, +127	Register
LJMP nn	nn → PC	0, 64k	Absolute
LJMP [lr]	lr → PC	0, 64k	Indexed

Note: PC = Program Counter; contents initially points to instruction following jump.

2.0 CPU Description (Continued)

mind that the Jcc instruction is just an optional syntax for the conditional JMP instruction.

The example in *Figure 2-5* demonstrates two possible ways to code the conditional relative jump instruction when testing for a false [Z] flag in {CCR}. In the example, assume that the symbol "Z" equals "000" binary, that the symbol "NS" equals "0" binary, and that the symbol "SKIP.IT" points to the desired instruction with which to begin execution if [Z] is false.

On the other hand, the conditional absolute jump instruction, LJMP, can test any bit in any currently active CPU register. Table 2-14 shows the conditional long jump instruction syntax.

```
JMP   Z,NS,SKIP.IT   ;If [Z]=0 goto SKIP.IT
      -or-
JNZ   SKIP.IT         ;If [Z]=0 goto SKIP.IT
```

FIGURE 2-5. Coding Examples of Equivalent Conditional Jump Instructions

TABLE 2-11. Conditional Relative Jump Instruction

Syntax	Instruction Operation	Operand Range	Addressing Mode
JMP f,s,n	If the flag "f" is in the state "s" then PC + n (sign extended) → PC	−128, +127	Immediate
Jcc n	If the condition "cc" is met then PC + n (sign extended) → PC	−128, +127	Immediate

Note: PC = Program Counter; contents initially points to instruction following jump.

TABLE 2-12. "f" Flags

"f"(Binary)	Flag	Flag Name	Register Containing Flag
000	Z	Zero	{CCR}
001	C	Carry	{CCR}
010	V	Overflow	{CCR}
011	N	Negative	{CCR}
100	RA	Receiver Active	{TSR}
101	RE	Receiver Error	{TSR}
110	DAV	Data Available	{TSR}
111	TFF	Transmitter FIFO Full	{TSR}

TABLE 2-13. "cc" Conditions Tested

"cc" Field	Condition Tested for	Flag "f"'s Condition
Z	Zero	[Z] = 1
NZ	Not Zero	[Z] = 0
EQ	Equal	[Z] = 1
NEQ	Not Equal	[Z] = 0
C	Carry	[C] = 1
NC	No Carry	[C] = 0
V	Overflow	[V] = 1
NV	No Overflow	[V] = 0
N	Negative	[N] = 1
P	Positive	[N] = 0
RA	Receiver Active	[RA] = 1
NRA	Not Receiver Active	[RA] = 0
RE	Receiver Error	[RE] = 1
NRE	No Receiver Error	[RE] = 0
DA	Data Available	[DAV] = 1
NDA	No Data Available	[DAV] = 0
TFF	Transmitter FIFO FULL	[TFF] = 1
NTFF	Transmitter FIFO Not Full	[TFF] = 0

TABLE 2-14. Conditional Absolute Jump Instruction

Syntax	Instruction Operation	Operand Range	Addressing Mode
LJMP Rs,p,s,nn	If the bit of register "Rs" in position "p" is in the state "s" then nn → PC	0, 64k	Register, Absolute

Note: PC = Program Counter

2.0 CPU Description (Continued)

The BCP also has a specialized relative jump instruction called relative Jump with Rotate and Mask on source register, JRMK. This instruction facilitates the decoding of register fields often involved in communications processing. JRMK does this by rotating and masking a copy of its register operand to form a signed program counter displacement which usually points into a jump table. Table 2-15 shows the syntax and operation of the JRMK instruction.

JRMK's masking, (setting to zero), the least significant bit of the displacement allows the construction of a jump table using either one or two word instructions; for instance, a table of JMP and/or LJMP instructions, respectively. The example in *Figure 2-6* demonstrates the JRMK instruction decoding the address frame of the 3299 Terminal Multiplex-er protocol which is located in the Receive/Transmit Register, {RTR[4–2]}.

The BCP has two unconditional call instructions; CALL, which supports relative instruction addressing and LCALL, (Long CALL), which supports absolute instruction addressing. These instructions push the following information onto the CPU's internal Address Stack: the address of the next instruction; the status of the Global Interrupt Enable flag, [GIE]; the status of the ALU flags [Z], [C], [N], and [V]; and the status of which register banks are currently active. Table 2-16 lists the two unconditional call instructions. Note that the Address Stack is only twelve positions deep; therefore, the BCP allows twelve levels of nested subroutine invocations, (this includes both interrupts and calls).

TABLE 2-15. JRMK Instruction

Syntax	Instruction Operation	Displacement Range	Addressing Mode
JRMK Rs, b, m	(a) Rotate a copy of register "Rs" "b" bits to the right. (b) Mask the most significant "m" bits and the least significant bit of the above result. (c) PC + resulting displacement (sign extended) → PC.	−128, +126	Register

Note: PC = Program Counter; contents initially points to instruction following jump.

```
Example Code
JRMK   RTR,1,4    ;decode terminal address
LJMP   ADDR.0     ;jump to device handler #0
LJMP   ADDR.1     ;jump to device handler #1
. . .
LJMP   ADDR.7     ;jump to device handler #7
```

	Instruction Execution	JRMK Displacement Register Contents							
(a)	Copy {RTR} into JRMK's displacement register:	x	x	x	A2	A1	A0	y	y
(b)	Rotate displacement register 1 bit to the right:	y	x	x	x	A2	A1	A0	y
(c)	AND result with "00001110" binary mask:	0	0	0	0	A2	A1	A0	0
(d)	Sign extend resulting displacement and add it to the program counter, (PC). If the bits A2 A1 A0 equal "0 0 1" binary then + 2 is added to the Program Counter; (i.e., PC + 2 → PC).	0	0	0	0	0	0	1	0
(e)	Execute the instruction pointed to by the PC, which in this example is: LJMP ADDR.1								

FIGURE 2-6. JRMK Instruction Example

TABLE 2-16. Unconditional Call Instructions

Syntax	Instruction Operation	Operand Range	Addressing Mode
CALL n	PC & [GIE] & ALU flags & reg. bank selection → Address Stack PC + n (sign extended) → PC	−128, +127	Immediate
LCALL nn	PC & [GIE] & ALU flags & reg. bank selection → Address Stack nn → PC	0, 64k	Absolute

Note: PC = Program Counter; contents initially points to instruction following call.
    [GIE] = Global Interrupt Enable bit.
    & = concatenation operator, combines operands together forming one long operand.

2.0 CPU Description (Continued)

The BCP has one conditional call instruction capable of testing any bit in any currently active CPU register. This call only supports absolute instruction addressing. Table 2-17 shows the conditional call instruction syntax and operation.

The return instruction complements the above call instructions. Two versions of the return instruction exist, the uncondtional return and the conditional return. When the unconditional return instruction is executed, it pops the last address on the CPU's Address Stack into the program counter and it can optionally affect the [GIE] bit, the ALU flags, and the register bank selection. Table 2-18 shows the syntax and operation of the unconditional return instruction.

The conditional return instruction functions the same as the unconditional return instruction if a desired condition is met. As with the conditional jump instruction, the conditional return instruction has two possible syntaxes. Table 2-19 lists the syntax for the conditional return. The "f" flags and the "cc" conditions for the return instruction are the same as for the conditional jump instruction, therefore refer to Table 2-12 and Table 2-13 for the listing of "f" and "cc", respectively.

TABLE 2-17. Conditional Call Instruction

Syntax	Instruction Operation	Operand Range	Addressing Mode

LCALL Rs, p, s, nn	If the bit of register "Rs" in position "p" is in the state "s" then
	    PC & [GIE] & ALU flags &
	    reg. bank selection → Address Stack
	    nn → PC
	End if
		0, 64k	Register, Absolute

Note: PC = Program Counter; contents initially points to instruction following call.
    [GIE] = Global Interrupt Enable bit
    & = concatenation operator, combines operands together forming one long operand.

TABLE 2-18. Unconditional Return Instruction

Syntax	Instruction Operation

RET {g {, rf}}	Case "g" of
	        0: leave [GIE] unaffected, (default)
	        1: restore [GIE] from Address Stack
	        2: set [GIE]
	        3: clear [GIE]
	End case
	If "rf" = 1 then
	        restore ALU flags from Address Stack
	        restore register bank selection from Address Stack
	Else (the default)
	        leave the ALU flags and register bank selections unchanged
	End if
	Address Stack → PC

Note: PC = Program Counter
    [GIE] = Global Interrupt Enable bit
    {} = surrounds optional operands that are not part of the instruction syntax.
    Optional operands may either be specified or omitted.

TABLE 2-19. Conditional Return Instruction

Syntax	Instruction Operand
RETF f, s {, {g}, {, rf}}	If the flag "f" is in the state "s" then perform a RET {g {, rf}}
Rcc {g {, rf}}	If the condition "cc" is met then perform a RET {g {,rf}}

Note: See Table XVIII for an explanation of "RET {g {, rf}}"
    {} = surrounds optional operands that are not part of the instruction syntax.
    Optional operands may either be specified or omitted.
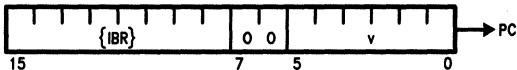
2.0 CPU Description (Continued)

In addition to the above jump, call and return program flow instructions, the BCP is capable of generating software interrupts via the TRAP instruction. This instruction generates a call to any one of 64 possible interrupt table addresses based on its vector number operand. This allows both the simulation of hardware interrupts and the construction of special software interrupts, if desired. The actual interrupt table entry address is determined by concatenating the Interrupt Base Register, {IBR}, to an 8-bit representation of the vector number operand in the TRAP instruction. This instruction may also clear the [GIE] bit, if desired. Table 2-20 shows the syntax and operation of the TRAP instruction.

Miscellaneous Instructions

As stated in the "CPU Register Set" section, the BCP has 44 registers with 24 of them arranged into four register banks: Main Bank A, Alternate Bank A, Main Bank B, and Alternate Bank B. The exchange instruction, EXX, selects which register banks are currently available to the CPU, for example either Main Bank A or Alternate Bank A. The deselected register banks retain their current values. The EXX instruction can also alter the state of [GIE], if desired. Table 2-21 shows the EXX instruction syntax and operation.

TABLE 2-20. TRAP Instruction

Syntax	Instruction Operation	Operand Range
TRAP v {, g'}	PC & [GIE] & ALU flags & reg. Bank Selection ⟶ Address Stack If "g'" = 1 then clear [GIE] Form PC address as shown below: {IBR} ┆ 0 0 ┆ v ┆ ⟶ PC 15 7 5 0	0, 63

Note: PC = Program Counter; contents initially points to instruction following call.
 [GIE] = Global Interrupt Enable bit
 IBR = Interrupt Base Register
 & = concatenation operator, combines operands together forming one long operand.
 { } = surrounds optional operands that are not part of the instruction syntax.
 Optional operands may either be specified or omitted.

TABLE 2-21. EXX Instruction

Syntax	Instruction Operation
EXX ba, bb {, g}	Case "ba" of 0: activate Main Bank A 1: activate Alternate Bank A End case Case "bb" of 0: activate Main Bank B 1: activate Alternate Bank B End case Case "g" of 0: leave [GIE] unaffected, (default) 1: (reserved) 2: set [GIE] 3: clear [GIE] End case

Note: [GIE] = Global Interrupt Enable bit
 { } = surrounds optional operands that are not part of the instruction syntax.
 Optional operands may either be specified or omitted.

2.0 CPU Description (Continued)

2.2 CPU FUNCTIONAL DESCRIPTION

2.2.1 ALU

The BCP provides a full function high speed 8-bit Arithmetic Logic Unit (ALU) with full carry look ahead, signed arithmetic, and overflow decision capabilities. The ALU can perform six arithmetic, nine logic, one rotate and two shift operations on binary data. Full access is provided to all CPU registers as both source and destination operands, and using the indirect addressing mode, results may be placed directly into data memory. All operations which have an internal destination (register addressing) are completed in two (2) T-states. External destination operations (indirect addressing to data memory) complete in three (3) T-states.

Arithmetic operations include addition with or without carry, and subtraction with or without borrow (represented by carry). Subtractions are performed using 2's complement addition to accommodate signed operands. The subtrahend is converted to its 2's complement equivalent by the ALU and then added to the minuend. The result is left in 2's complement form.

The remaining ALU operations include full logic, shift and rotate operations. The logic functions include Complement, AND, OR, Exclusive-OR, Compare and Bit Test. Zero through seven bit right and left shift operations are provided, along with a zero through seven bit right rotate operation. Note that the shift and rotate operations may only be performed on a register, which is both the source and destination. (See the Instruction Set Overview section for detailed descriptions of these operations.)

The BCP ALU provides the programmer with four instruction result status bits for conditional operations. These bits (known as condition code flags) indicate the status (or condition) of the destination byte produced by certain instructions. Not all instructions have an affect on every status flag. (See the Instruction Set Reference section for the specific details on what status flags a given instruction affects.) These flags are held in the Condition Code Register, {CCR}, see Figure 2-7.

7	6	5	4	3	2	1	0
TO	RR	RW	BIRQ	N	V	C	Z

where:

N = Negative

C = Carry

V = Overflow

Z = Zero

FIGURE 2-7. Condition Code Register ALU Flags

If an instruction is documented as affecting a given flag, then the flags are set (to 1) or cleared (to 0) under the following conditions:

[N]— The Negative flag is set if the most significant bit (MSB) of the result is one (1), otherwise it is cleared. This flag represents the sign of the result if it is interpreted as a 2's complement number.

[C] — The Carry flag is set if:

a) An addition operation generates a carry, see Figure 2-8a.

b) A subtract or compare operation generates a borrow, see Figure 2-8b.

c) The last bit shifted out during a shift operation (in either direction) is a one (1), see Figure 2-9.

d) The last bit rotated by the rotate operation is a one (1), see Figure 2-10.

In all other conditions [C] is cleared.

[V]— Overflow is set whenever the result of an arithmetic or compare operation on signed operands is not representable by the operand size, thereby producing an incorrect result. For example, the addition of the two signed negative numbers in Figure 2-8a would set [V] since the correct representation of the result, both sign and magnitude, is not possible in 8 bits. On the other hand, in Figure 2-8b and 2-8c [V] would be cleared because the results are correctly represented in both sign and magnitude. It is important to remember that Overflow is only meaningful in signed arithmetic and that it is the programmer's responsibility to determine if a given operation involves signed or unsigned values.

[Z]— The Zero flag is set only when an operation produces an all bits cleared result (i.e., a zero). In all other conditions [Z] is cleared.

```
     11101010          10111010          11011100
  +  10001100       -  11000100       +  01100011
1 ←  01110110     1 →  11110110     1 ←  00111111

    [C] = 1           [C] = 1           [C] = 1
    [V] = 1           [V] = 0           [V] = 0
      (a)               (b)               (c)
```
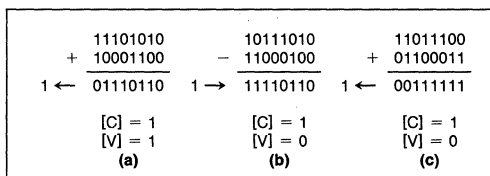
FIGURE 2.8. Carry and Overflow Calculations

Shift Left

Shift Right
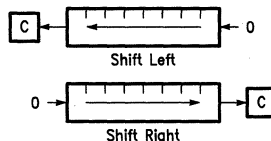
TL/F/9336–D3

FIGURE 2-9. Shifts' Effect on Carry
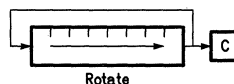
Rotate

TL/F/9336–D4

FIGURE 2-10. Rotate's Effect on Carry

2

2.0 CPU Description (Continued)

Several conditions apply to these flags, independent of their operation and the way they are calculated. These conditions are:

1. A flag's previous state is retained when an instruction has no affect on that flag.

2. Direct reading and writing of all ALU flags is possible via the {CCR} register.

3. Currrent flag values are saved onto the address stack during interrupt and call operations, and can be restored to their original values if a return instruction with the re-store flags option is executed.

4. Flag status is calculated in parallel with the instruction result, therefore no time penalty is associated with flag operation.

When performing single byte arithmetic (i.e., the values are completely represented in one byte) the Add (ADD,ADDA) and Subtract (SUB,SUBA) instructions should be used, but when performing multi-byte arithmetic the Add with Carry (ADCA) and Subtract with Carry (SBCA) instructions should be used. This is because the carry (in an add operation) or the borrow (in a subtract operation) must be carried forward to the higher order bytes. *Figure 2-11* demonstrates an instruction sequence for a 16-bit add and an instruction sequence for a 16-bit subtract.

Assume the 16-bit variable X is represented by the register pair R4(MSB), R5(LSB), and that the 16-bit variable Y is represented by the register pair R6(MSB), R7(LSB).

To perform the assignment Y = X + Y:

```
MOVE    R7,A    ;GET LSB OF Y
ADDA    R5,R7   ;Y(LSB)=X(LSB)+Y(LSB)
MOVE    R6,A    ;GET MSB OF Y
ADCA    R4,R6   ;Y(MSB)=X(MSB)+Y(MSB)
                        +CARRY
```

To perform the assignment Y = X − Y:

```
MOVE    R7,A    ;GET LSB OF Y
SUBA    R5,R7   ;Y(LSB)=X(LSB)−Y(LSB)
MOVE    R6,A    ;GET MSB OF Y
SBCA    R4,R6   ;Y(MSB)=X(MSB)−Y(MSB)
                        −CARRY
```

**FIGURE 2-11. Multi-Byte Arithmetic
Instruction Sequences**

When using the ALU to perform comparisons, the programmer has two options. If the compare is to a constant value then the CMP instruction can be used, else one of the subtract instructions must be used. When determining the results of any compare, the programmer must keep in mind whether they are comparing signed or unsigned values. Table 2-22 lists the Boolean condition that must be met for unsigned comparisons and Table 2-23 lists the Boolean condition that must be met for signed comparisons.

TABLE 2-22

Unsigned Comparison Results	
Comparison: x − y	Boolean Condition
x < y	C
x ≤ y	C $\mid$ Z
x = y	Z
x ≥ y	$\overline{C}$
x > y	$\overline{C}$ & $\overline{Z}$

Note: & = logical AND

$\quad\mid$ = logical OR

$\quad\overline{z}$ = one's complement

TABLE 2-23

Signed Comparison Results	
Comparison: x − y	Boolean Condition
x < y	$(N\&\overline{V}) \mid (\overline{N}\&V)$
x ≤ y	$Z \mid (N\&\overline{V}) \mid (\overline{N}\&V)$
x = y	Z
X ≥ y	$(N\&V) \mid (\overline{N}\&\overline{V})$
x > y	$(N\&V\&\overline{Z}) \mid (\overline{N}\&\overline{V}\&\overline{Z})$

Note: & = logical AND

$\quad\mid$ = logical OR

$\quad\overline{z}$ = one's complement

2.2.2 Timing

Timing on the BCP is controlled by an internal oscillator and circuitry that generates the internal timing signals. This circuitry in the CPU is referred to as Timing Control. The internal timing of the CPU is synchronized to an internal clock called the CPU clock, CPU-CLK. A period of CPU-CLK is referred to as a T-state. The clock for the BCP is provided by a crystal connected between X1 and X2 or from a clock source connected to X1. This clock will be referred to as the oscillator clock, OCLK. The frequency of OCLK is divided in half when the CPU clock select bit, [CCS], in the Device Control Register, {DCR}, is set to a one. Either OCLK or OCLK/2 is used by Timing Control to generate CPU-CLK and other synchronous signals used to control the CPU timing.

After the BCP is reset, [CCS] is high and CPU-CLK is generated from OCLK/2. Since the output of the divider that creates OCLK/2 can be high or low after reset, CPU-CLK can also be in a high or low state. Therefore, the exact number of clock cycles to the start of the first instruction cannot be determined. Automatic test equipment can synchronize to the BCP by asserting $\overline{RESET}$ as shown in *Figure 2-12*. The falling edge of $\overline{RESET}$ generates a clear signal which causes CPU-CLK to fall. The next rising edge of X1 removes the clear signal from CPU-CLK. The second rising edge of X1 will cause CPU-CLK to rise and the relationship between X1 and CPU-CLK can be determined from this point.

Writing a zero to [CCS] causes CPU-CLK to switch from OCLK/2 to OCLK. The transition from OCLK to OCLK/2 occurs following the end of the instruction that writes to

2.0 CPU Description (Continued)

[CCS] as shown in *Figure 2-13*. The switch occurs on the falling edge of X1 when CPU-CLK is low. CPU-CLK can be changed back to OCLK/2 by writing a one to [CCS]. The point at which CPU-CLK changes depends on whether there has been an odd or even number of T-states since [CCS] was set low. The change would require a maximum of two T-states and a minimum of one T-state following the end of the instruction that writes to [CCS].

The CPU is a RISC processor with a limited number of instructions which execute in a short period of time. The maximum instruction cycle time is four T-states and the minimum is two T-states. Five types of instruction timing are used in

the CPU: two T-state, three T-state program control, three T-state data memory access, four T-state program control, and four T-state two word program control. The first T-state of each instruction is T1 and the last T-state is T2. Intermediate T-states required to complete the instruction are referred to as TX.

The instruction clock output, ICLK, defines the instruction boundaries. ICLK rises at the beginning of each instruction and falls one-half T-state after the next address is generated on the instruction address bus, IA. Thus, ICLK indicates the start of each instruction and when the next instruction address is valid.
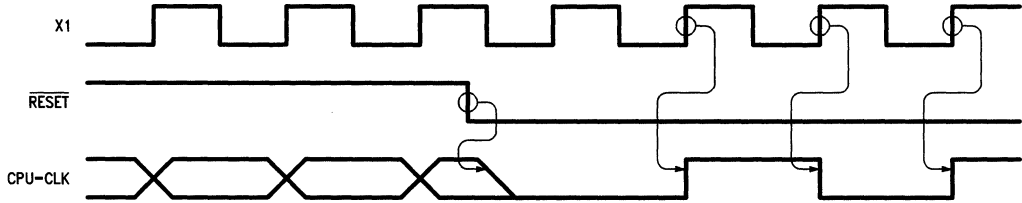
TL/F/9336–D5

FIGURE 2-12. CPU-CLK Synchronization with X1

TL/F/9336–D6
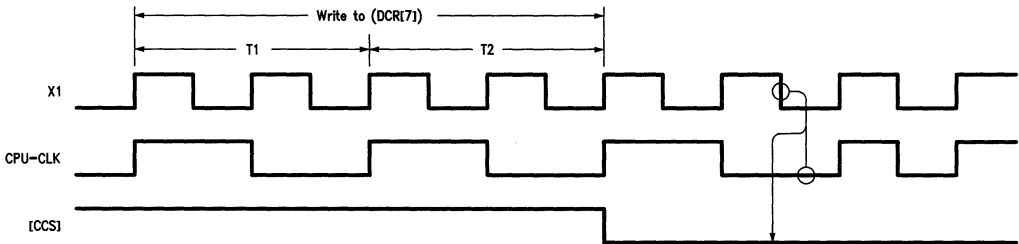
FIGURE 2-13. Changing from OCLK/2 to OCLK

2.0 CPU Description (Continued)

Figure 2-14 shows the relationship between CPU-CLK, ICLK, and IA for a two T-state instruction. The rising edge of CPU-CLK generates ICLK at the start of T1. The next falling edge of CPU-CLK increments the instruction address which appears on IA. ICLK falls one-half T-state later. The instruction completes during T2 which ends with ICLK rising, signifying the beginning of the next instruction.

The three T-state program control instruction is similar and is shown in *Figure 2-15*. An additional T-state, TX, is added between T1 and T2. ICLK rises at the beginning of T1 as before but falls at the end of TX. The next instruction address is generated one-half T-state before the end of TX and the instruction ends with T2.

The three T-state data memory access instruction timing is shown in *Figure 2-16*. Again, TX is inserted between T1 and T2. ICLK rises at the beginning of the instruction and falls at the end of T1. The next instruction address appears on IA one-half clock cycle before ICLK falls. The address latch enable output, ALE, rises halfway through T1 and falls half-way through TX. The BCP has a 16-bit data memory address bus and an 8-bit data bus. The data bus is multiplexed with the lower 8 bits of the address bus and ALE is used to latch the lower 8 bits of the address during a data memory access. The upper 8 bits of the address become valid one-half T-state after the beginning of T1 and go invalid one-half T-state after the end of T2. The lower 8 bits of the address become valid on the address-data bus, AD, when ALE rises and goes invalid one-half T-state after ALE falls. *Figure 2-16* shows a write to data memory in which case AD switches from address to data at the beginning of T2. The data is held valid until one-half T-state after the end of T2. The write strobe, $\overline{\text{WRITE}}$, falls at the beginning of T2 and rises at the end of T2. A read of data memory is shown in *Figure 2-17*. The read timing is the same as a write except one-half T-state after ALE falls AD goes into a high impedance state allowing data to enter the BCP from data memory. AD returns to an active state at the end of T2. The read strobe, $\overline{\text{READ}}$, timing is identical to $\overline{\text{WRITE}}$.
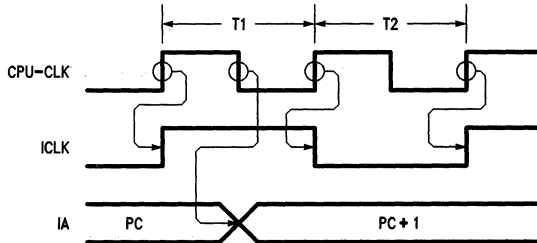
FIGURE 2-14. Two T-state Instruction

TL/F/9336–D7
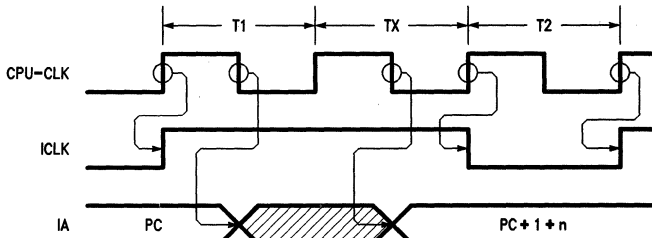
FIGURE 2-15. Three T-state Program Control Instruction

TL/F/9336–D8

2.0 CPU Description (Continued)
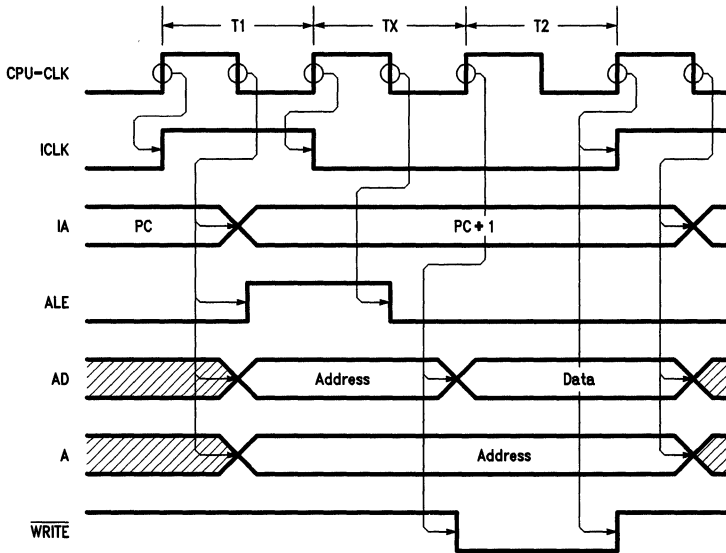
TL/F/9336-D9

FIGURE 2-16. Three T-state Data Memory Write Instruction

TL/F/9336-E1
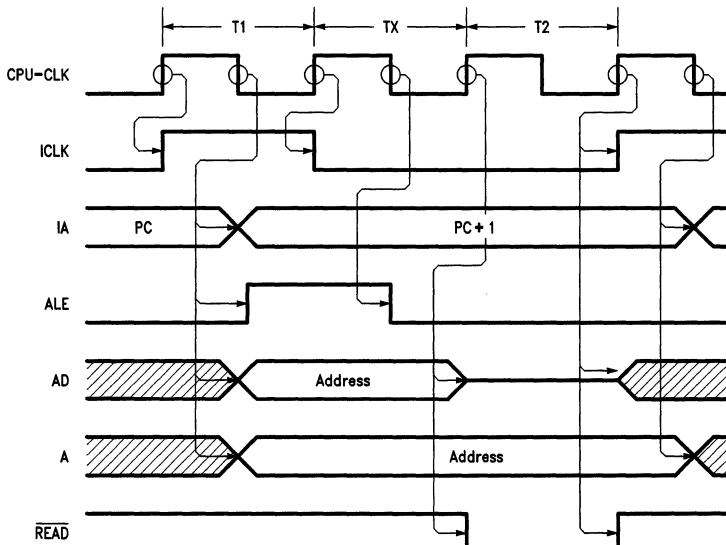
FIGURE 2-17. Three T-state Data Memory Read Instruction

2.0 CPU Description (Continued)

The four T-state program control instruction timing is shown in *Figure 2-18*. The instruction has two TX states inserted between T1 and T2. ICLK rises at the beginning of T1 and falls at the end of the second TX. The next instruction address becomes valid halfway through the second TX. The four T-state two word program control instruction timing is the same as two consecutive two T-state instructions and is shown in *Figure 2-19*.

This timing describes the minimum cycle time required by each type of instruction. The BCP can be slowed down by changing the number of wait states selected in the Device Control Register, {DCR}. The BCP can be programmed for up to three instruction memory wait states (instruction wait states) and seven data memory wait states (data wait

states). Instruction wait states affect all instruction types while data wait states affect only data memory access instructions. Bits three and four in {DCR} control the number of instruction wait states and bits zero, one and two are used to select the number of data wait states. The relationships between the control bits and the number of wait states selected are shown in Table 2-24 and Table 2-25. The BCP is configured with three instruction wait states and seven data wait states after reset. A write to {DCR[4,3]} to change the number of instruction wait states takes effect on the following instruction if that instruction is a three T-state or four T-state program control instruction. For the other instruction types, the new number of instruction wait states will take effect on the instruction following the instruction
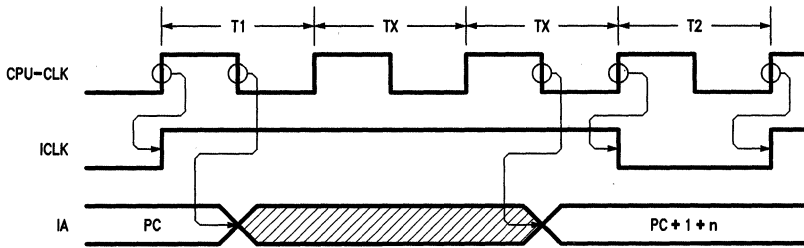
TL/F/9336–E2

FIGURE 2-18. Four T-state Program Control Instruction
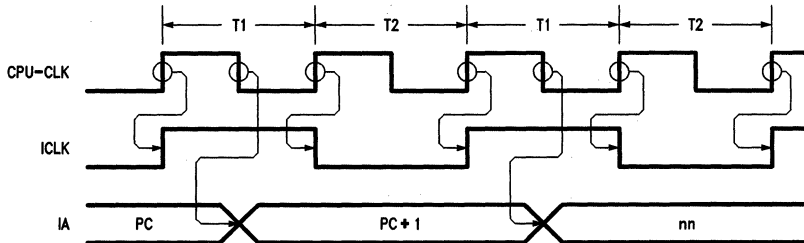
TL/F/9336–E3

FIGURE 2-19. Four T-state Two Word Instruction

TABLE 2-24. Data Memory Wait States

{DCR[2–0]}	Data Wait States
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

TABLE 2-25. Instruction Memory Wait States

{DCR[4,3]}	Instruction Wait States
00	0
01	1
10	2
11	3

2.0 CPU Description (Continued)

after the write to {DCR}. A write to {DCR[2–0]} to change the number of data wait states will take effect on the next data memory access instruction even if it immediately follows the write to {DCR}. Both instruction and data wait states cause the insertion of additional T-states prior to T2 and these T-states are referred to as TW. The purpose of instruction wait states is to increase the time from instruction address generation to the beginning of the next instruction cycle. Data wait states increase the time from data memory address generation to the removal of the strobe at the end of data memory access instructions. Therefore, instruction and data wait states are counted concurrently in a data memory access instruction and TX of a data memory access instruction is counted as one instruction wait state. The actual number of wait states added to a data memory access is calculated as the maximum between the number

of data wait states and one less than the number of instruction wait states. *Figure 2-20* shows a write of data memory with one wait state. This could be accomplished by selecting two instruction wait states or one data wait state. The effect of the wait state is to increase the time the write strobe is active and the data is valid on AD. The same situation for a read of data memory is shown in *Figure 2-21.* A two T-state instruction with two instruction wait states is shown in *Figure 2-22* and a four T-state instruction with one instruction wait state is shown in *Figure 2-23.* As stated earlier, instruction wait states are inserted before T2. Adding wait states to a four T-state two word instruction causes the wait states to count twice when calculating total instruction cycle time. The wait states are added to each of the two words of the instruction.
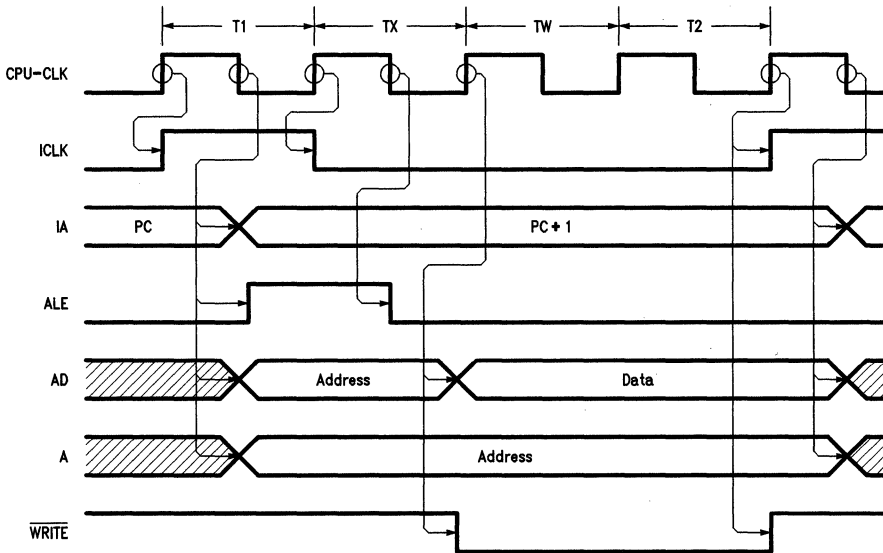
TL/F/9336–E4

FIGURE 2-20. Data Memory Write with One Wait State

2.0 CPU Description (Continued)
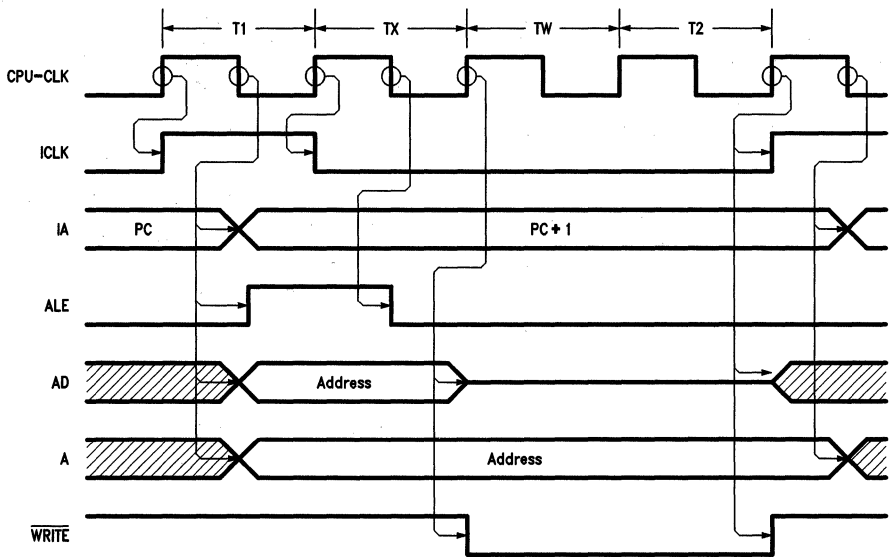
FIGURE 2-21. Data Memory Read with One Wait State

TL/F/9336–E5
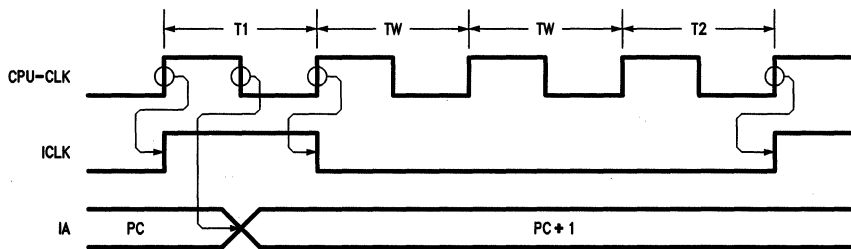
FIGURE 2-22. Two T-state Instruction with Two Wait States

TL/F/9336–E6
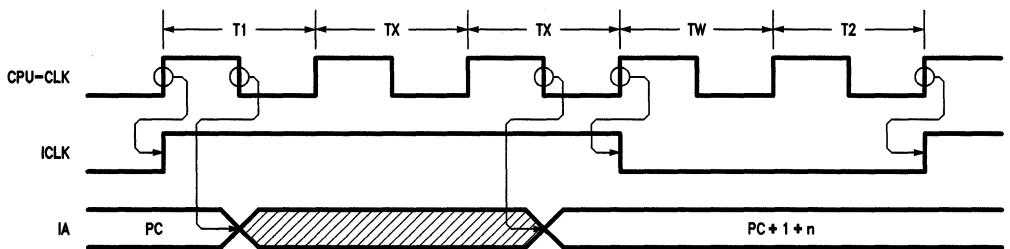
FIGURE 2-23. Four T-state Instruction with One Wait State

TL/F/9336–E7

2.0 CPU Description (Continued)

The $\overline{\text{WAIT}}$ pin can also be used to add wait states to BCP instruction execution. The CPU will be waited as long as $\overline{\text{WAIT}}$ is low. To wait a given instruction, $\overline{\text{WAIT}}$ must be asserted low one-half T-state prior to the beginning of T2 in the instruction to be affected. *Figure 2-24* shows $\overline{\text{WAIT}}$ asserted during a write to data memory. In order to wait this instruction, $\overline{\text{WAIT}}$ must fall prior to the falling edge of CPU-CLK in TX. One wait state is added to the access and $\overline{\text{WAIT}}$ rises prior to the falling edge of CPU-CLK in TW which al-

lows the access to finish. If $\overline{\text{WAIT}}$ had remained low, the access would have been held off indefinitely. Programmed wait states would delay when $\overline{\text{WAIT}}$ must be asserted since they would delay the beginning of T2. *Figures 2-25* through *Figure 2-27* depict the use of $\overline{\text{WAIT}}$ with three other instruction types. In all three cases, $\overline{\text{WAIT}}$ is asserted one-half T-state prior to when T2 would normally begin. Also, it is evident that the effect of $\overline{\text{WAIT}}$ on instruction timing is identical to adding programmed wait states.
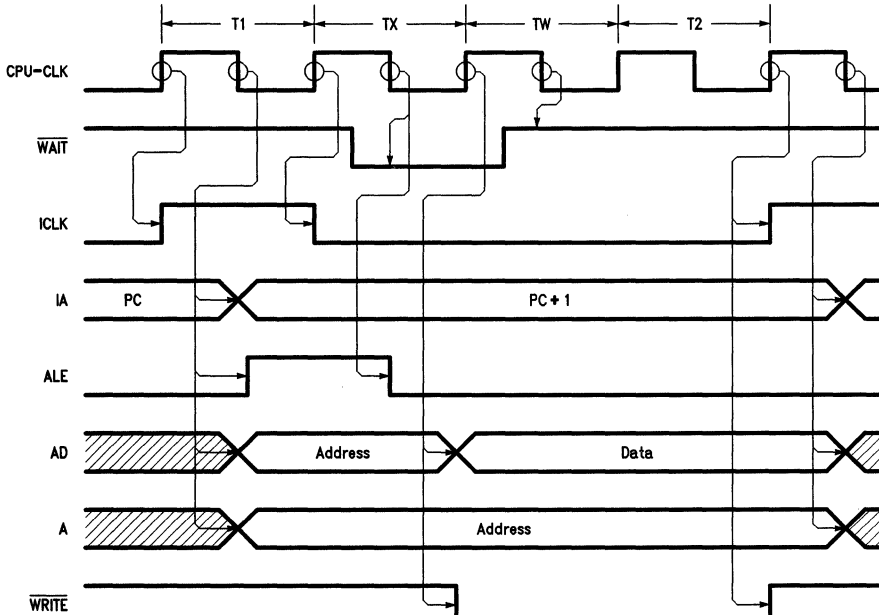
TL/F/9336–E8

FIGURE 2-24. Data Memory Access $\overline{\text{WAIT}}$ Timing

TL/F/9336–E9
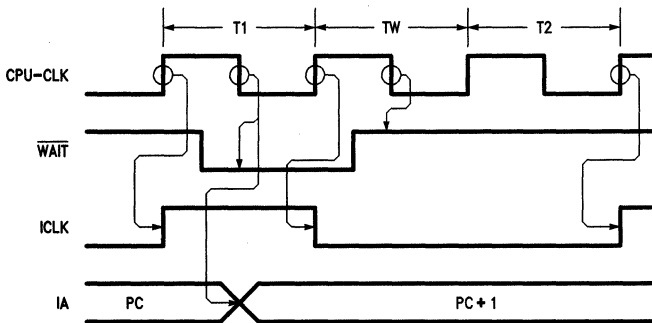
FIGURE 2-25. Two T-state Instruction $\overline{\text{WAIT}}$ Timing

2

2.0 CPU Description (Continued)
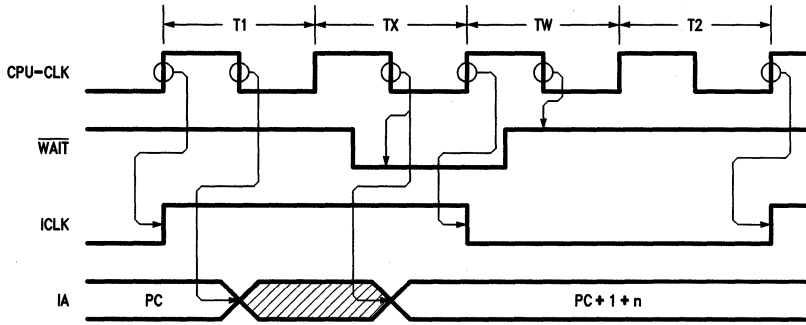
TL/F/9336–F1

FIGURE 2-26. Three T-state Program Control Instruction $\overline{WAIT}$ Timing
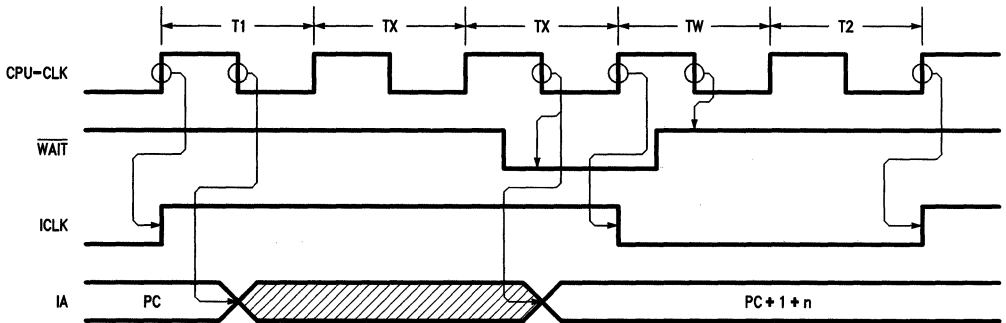
TL/F/9336–F2

FIGURE 2-27. Four T-state Program Control Instruction $\overline{WAIT}$ Timing

$\overline{LOCK}$ is another input which affects BCP instruction timing. $\overline{LOCK}$ prevents the BCP from accessing data memory. When asserted low, $\overline{LOCK}$ will cause the BCP to wait when it executes a data memory access instruction. The BCP will be waited until $\overline{LOCK}$ is taken high. To prevent a given access of data memory, $\overline{LOCK}$ must be asserted low one-half T-state prior to the beginning of the instruction accessing data memory. *Figure 2-28* shows $\overline{LOCK}$ being used to wait a write to data memory. $\overline{LOCK}$ falls prior to the falling edge of CPU-CLK before T1. In order to guarantee at least one wait state, $\overline{LOCK}$ is held low until after the falling edge of CPU-CLK in T1. This causes the insertion of TW into the cycle prior to TX. ALE remains high and the address is delayed on AD until $\overline{LOCK}$ is removed. After $\overline{LOCK}$ rises the access concludes normally with ALE falling halfway through TX and $\overline{WRITE}$ occurring during T2. Note that $\overline{LOCK}$ waits the access at a different point in the cycle than programmed wait

states or $\overline{WAIT}$. Additional wait states could occur from these sources prior to T2. *Figure 2-29* shows an example of $\overline{LOCK}$ holding off a write to data memory with one programmed wait state.

With timing similar to $\overline{LOCK}$, the BCP will be delayed from making a data memory access by an access from the remote system. If the remote system is accessing the Remote Interface Configuration register, {RIC}, or data memory, the BCP will be waited by the Remote Interface and Arbitration System, RIAS, until the remote access is finished. The length of time the BCP is waited depends on the speed of the remote system and the type of remote access. The wait states are added prior to TX in the same manner as for $\overline{LOCK}$ shown in *Figure 2-28*. A more detailed description of the operation of RIAS can be found in Section 4.0, Remote Interface and Arbitration System.

2.0 CPU Description (Continued)
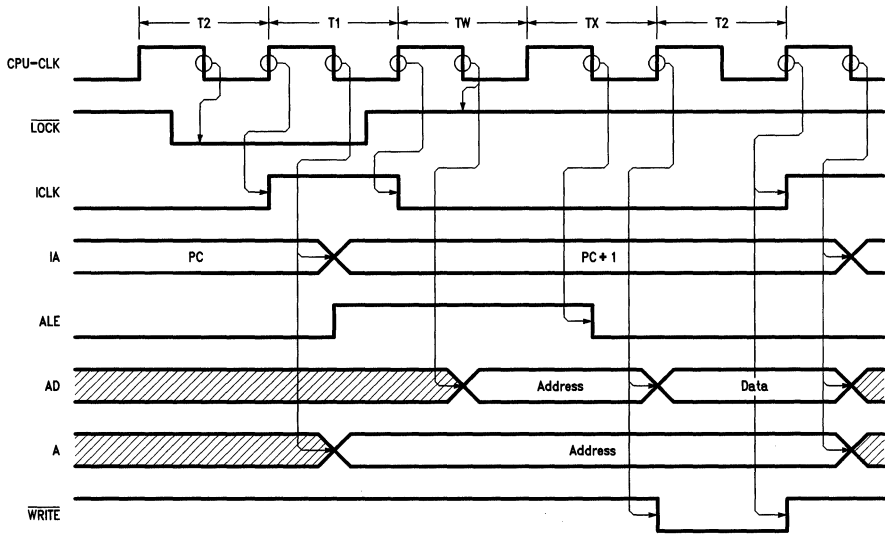
TL/F/9336-F3

FIGURE 2-28. $\overline{LOCK}$ Timing

TL/F/9336-F4
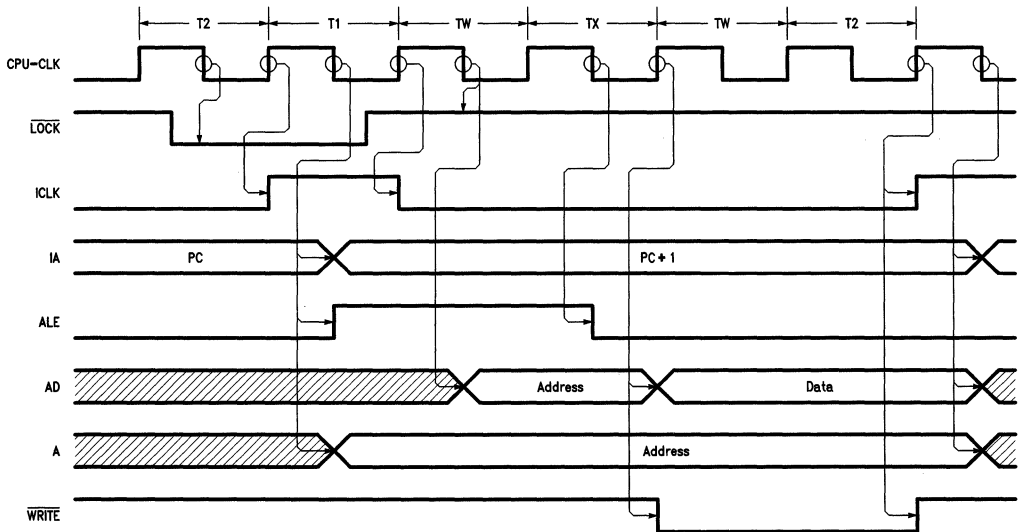
FIGURE 2-29. $\overline{LOCK}$ Timing with One Wait State

2.0 CPU Description (Continued)

The CPU will be stopped after $\overline{RESET}$ is asserted low. The CPU can be externally controlled by changing the state of the start bit, [STRT], in {RIC}. The CPU starts executing instructions from the current address in the program control register when a one is written to [STRT] and stops when [STRT] is cleared. The CPU will complete the current instruction before stopping. Controlling the CPU from {RIC} requires a processor to access {RIC}. If no external processor is present, the CPU can be made to start automatically after reset by holding $\overline{REM\text{-}WR}$ and $\overline{REM\text{-}RD}$ low and $\overline{RAE}$ high while $\overline{RESET}$ is transitioning from low to high. The CPU "kick-starts" and will begin executing instructions from address zero. The timing for kick-starting the CPU is shown in *Figure 2-30*. ICLK rises on the rising edge of CPU-CLK one T-state after $\overline{RESET}$ is de-asserted. The falling edge of ICLK signifies the beginning of the first instruction fetch. Three instruction wait states and T2 precede the first instruction.

A functional state diagram describing the timing of the CPU is shown in *Figure 2-31*. The functional state diagram is similar to a flow chart, except that transitions to a new state (states are denoted as rectangular boxes) can only occur on the rising edge of the CPU-CLK. A state box can specify several actions, and each action is separated by a horizontal line. A signal name listed in a state box indicates that that pin will be asserted high when Timing Control has entered that state. When the signal is omitted from a box, it is asserted low. (Note: this requires using the inversion of a signal in some cases.) Decision blocks are shown as diamonds and their meaning is the same as in a flow chart. The functional state diagram is a generalized approach to determining instruction flow while allowing for any combination of wait states and control signals. Timing Control always starts from a reset in the state IDLE. After $\overline{RESET}$ goes high, Timing Control remains in IDLE until [STRT] is written high. If the BCP kick-starts, Timing Control enters TST on the next rising edge of CPU-CLK. Timing Control starts with a dummy

instruction cycle in order to fetch the first instruction. ICLK goes high in T1 and the instruction wait state counter is loaded. ICLK falls when either T2 or TW is entered as determined by the value of i_{IW} and $\overline{WAIT}$. The normal instruction flow begins after T2 at B on the diagram. As an example, consider a three T-state data memory write instruction with one data wait state. The instruction cycle path for this instruction would begin at T1 following the decision block for data memory access. In T1, ICLK is asserted high, the instruction wait state counter is loaded, and a bus request to RIAS is generated. Also, ALE is asserted high on the falling edge of CPU-CLK during T1. A branch decision is now made based on the state of $\overline{LOCK}$ and the response from RIAS to the bus request. Assuming that $\overline{LOCK}$ is not asserted and a remote access is not in progress, Timing Control enters TX on the next rising edge of CPU-CLK. In TX, the data wait state counter is loaded and the instruction wait state counter is decremented. In this example, the instruction wait state counter is at zero and is not counting. The data wait state counter is loaded with one. ALE goes low on the falling edge of CPU-CLK during TX. The next decision block checks for a read of data memory. This example is a write to data memory so the decision is no and the branch is to the right. The wait state conditions are evaluated in the following decision block. i_{DW} is one and Timing Control enters TW on the next rising edge of CPU-CLK. $\overline{WRITE}$ is asserted low when TW is entered and the data wait state counter is decremented to zero. The decision on i_{DW}, i_{IW}, and $\overline{WAIT}$ is now true and T2 is entered on the next rising edge of CPU-CLK. $\overline{WRITE}$ remains low. The CPU will stop execution if [STRT] is low at B in the diagram. Otherwise, the next instruction will be executed beginning at A. To summarize, this instruction went through the following states: T1, TX, TW, and T2. The complete instruction cycle is shown in *Figure 2-20*. Any instruction cycle can be analyzed in a similar manner using this functional state diagram.
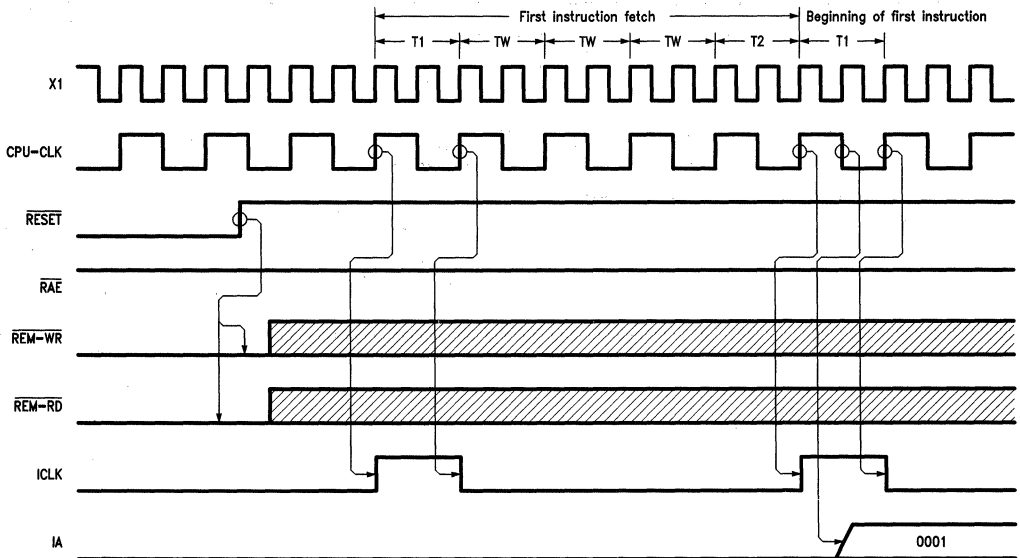
FIGURE 2-30. CPU Start-Up Timing

TL/F/9336–F5
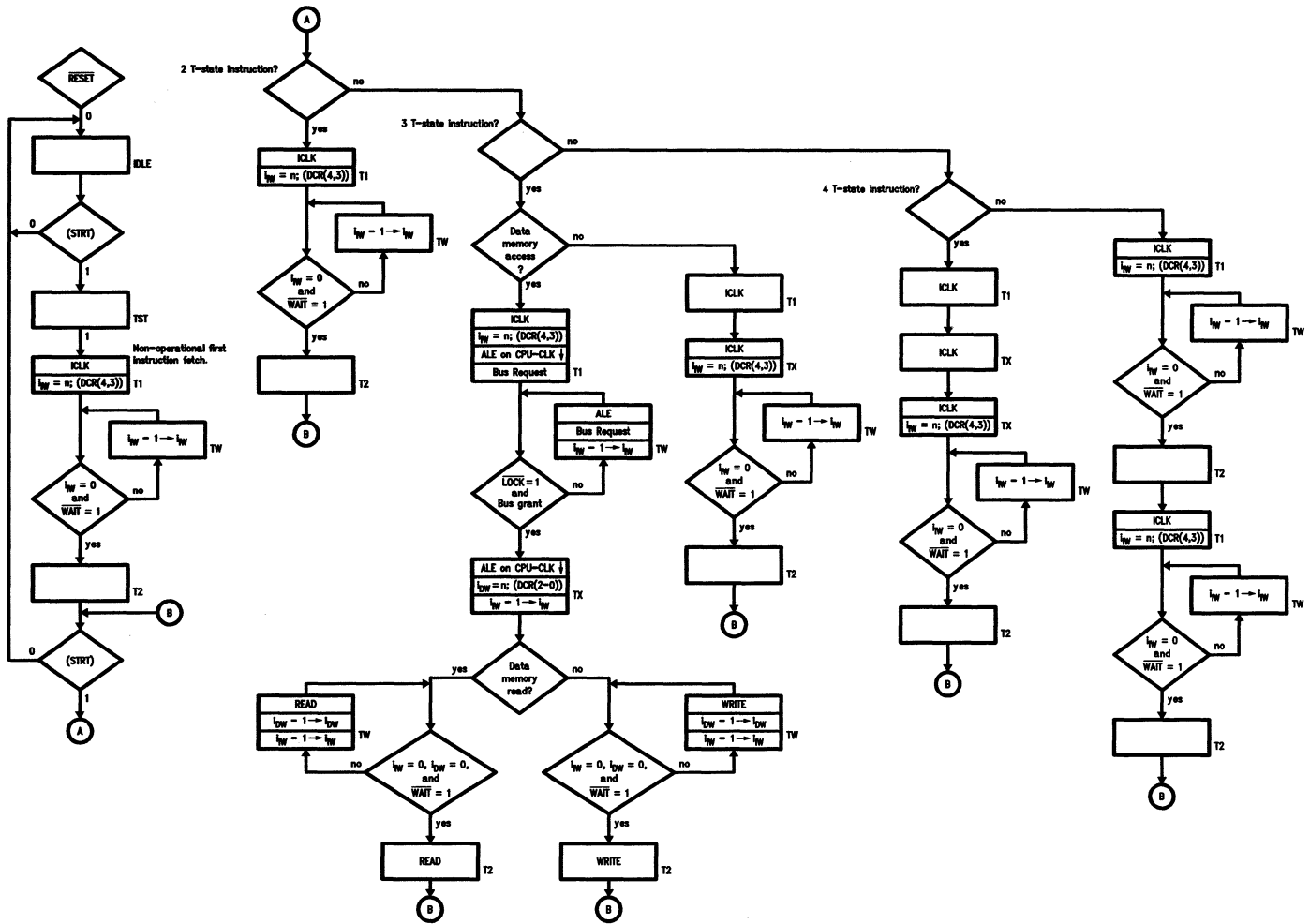
TL/F/9336–F6

FIGURE 2-31. Functional State Diagram of CPU Timing

2.0 CPU Description (Continued)

2.2.3 Interrupts

The DP8344A has two external and four internal interrupt sources. The external interrupt sources are the Non-Maskable Interrupt pin, $\overline{NMI}$, and the Bi-directional Interrupt Request pin, $\overline{BIRQ}$.

External

A non-maskable interrupt is detected by the CPU when a falling edge is detected at the $\overline{NMI}$ pin. The interrupt is automatically cleared internally when the CPU recognizes the interrupt.

$\overline{BIRQ}$ can function as both an interrupt into the DP8344 and as an output which can be used to interrupt other devices. $\overline{BIRQ}$ is configured as an input or output according to the state of [BIC] in the Auxiliary Control Register, {ACR}. $\overline{BIRQ}$ is an input if [BIC] is a zero and an output when [BIC] is a one. The reset state of [BIC] is a zero, causing $\overline{BIRQ}$ to be an input after the BCP is reset. [BIRQ] in the Condition Code Register, {CCR}, is a read only bit which mirrors the state of $\overline{BIRQ}$ regardless of whether $\overline{BIRQ}$ is configured as an input or output. This bit is updated at the beginning of T1 of each instruction.

When $\overline{BIRQ}$ is configured as an input, an interrupt will occur if the pin is held low. $\overline{BIRQ}$ must be held low until the interrupt is recognized or the interrupt will not be processed. Due to the prioritizing of interrupts as described below, $\overline{BIRQ}$ may not be recognized by the CPU until higher priority interrupts have been serviced. $\overline{BIRQ}$ will be recognized after higher priority interrupts have been processed. The low state on $\overline{BIRQ}$ should be removed after the CPU recognizes the interrupt or the interrupt will be processed multiple times.

When $\overline{BIRQ}$ is configured as an output, its state is controlled by [IM3] in the Interrupt Control Register, {ICR}. Changing the state of this bit will change $\overline{BIRQ}$ at the beginning of T1 of the instruction following the write to [IM3]. Note that [BIRQ] in {CCR} is also updated at the beginning of T1. Therefore, there is a one instruction cycle delay from when [IM3] changes to when the new value of $\overline{BIRQ}$ is made available in [BIRQ]. [BIS] in the Remote Interface Configuration register, {RIC}, mirrors the state of [IM3]. When $\overline{BIRQ}$ is an output, writing a one to [BIS] will change the state of [IME] thus changing $\overline{BIRQ}$ and allowing a remote processor to acknowledge an interrupt from the BCP. $\overline{BIRQ}$ will change state two T-states after the end of the write to [BIS]. Writing a one to [BIS] will have no effect on [IM3] when $\overline{BIRQ}$ is an input. Table 2-26 summarizes the relationship between $\overline{BIRQ}$ and its associated register bits.

TABLE 2-26. $\overline{BIRQ}$ Control Summary

[BIC]	[IM3]	[BiS]	[BIRQ]	$\overline{BIRQ}$
0	0	0	$\overline{BIRQ}$	Input, Active
0	1	1	$\overline{BIRQ}$	Input, Masked
1	0	0	0	Output, 0
1	1	1	1	Output, 1

Internal

The internal interrupts consist of the Transmitter FIFO Empty, TFE, interrupt, the Line Turn Around, LTA, interrupt, the Time Out, TO, interrupt, and a user selectable receiver in-

terrupt source. The receiver interrupt source is selected from either the Receiver FIFO, Full, RFF, interrupt, the Data Available, DA, interrupt, or the Receiver Active, RA, interrupt. The receiver interrupt is selected using bits [RIS1] and [RIS0] in the Interrupt Control Register, {ICR}. See the Section 3.0, Transceiver for a description of these interrupts.

Masking

The BCP uses two levels of interrupt masking: a global interrupt mask which affects all interrupts except $\overline{NMI}$ and individual interrupt mask bits. Global enabling and disabling of the interrupts is performed by changing the state of the Global Interrupt Enable bit, [GIE], in {ACR}. The maskable interrupts are disabled when [GIE] is a zero and enabled when [GIE] is a one. [GIE] is a zero after the BCP is reset. [GIE] is a read/write register bit and may be changed by using any instruction that can write to {ACR}. In addition, the RET, RETF, and EXX instructions have option fields which can be used to alter the state of [GIE]. The EXX instruction can set or clear [GIE] as well as leaving it unchanged. The RET and RETF instructions can restore [GIE] to the value that was saved on the address stack at the time the interrupt was recognized. These instructions also provide the options of clearing or setting [GIE] or leaving it unchanged. [GIE] is set to a zero when an interrupt is recognized by the CPU. It is necessary to set [GIE] to a one if interrupts are to be recognized within an interrupt routine.

The individual interrupt mask bits are located in {ICR}. When set to a one, bits [IM0], [IM1], [IM2], [IM3], and [IM4] in {ICR} mask the receiver interrupt, TFE interrupt, LTA interrupt, $\overline{BIRQ}$ interrupt, and TO interrupt, respectively. To enable an interrupt, its mask bit must be set to a zero. The interrupts and associated mask bits are shown in Table 2-27. These bits are set to a one when the DP8344 is reset.

Masking interrupts with [GIE] or the mask bits in {ICR} prevents the CPU from acknowledging interrupts but does not prevent the interrupts from occurring. Therefore, if an interrupt is asserted, it will be processed as soon as it is unmasked by changing [GIE] to a one and/or changing the appropriate mask bit in {ICR} to a zero.

Priorites

When more than one interrupt is unmasked and asserted, the CPU processes the interrupt with the highest priority first. $\overline{NMI}$ has the highest priority followed by the receiver interrupt, TFE, LTA, $\overline{BIRQ}$, and TO. Each time the interrupts are sampled, the highest priority interrupt is processed first, regardless of how long a lower priority interrupt has been active. Interrupt priority is summarized in Table 2-27.

TABLE 2-27. {ICR} Interrupt Mask Bits and Interrupt Priority

Interrupt	Mask Bit	Priority
$\overline{NMI}$	—	Highest
RFF, DA, RA	[IMO]	
TFE	[IM1]	
LTA	[IM2]	
$\overline{BIRQ}$	[iM3]	
TO	[IM4]	Lowest

2.0 CPU Description (Continued)

A call to the interrupt address is generated when an interrupt is detected by the CPU. The address for each interrupt is constructed by concatenating the Interrupt Base Register, {IBR}, contents with the individual interrupt code as shown in Table 2-28. There is room between the interrupt addresses for a maximum of four instruction words.

TABLE 2-28. Interrupt Vector Generation

Interrupt	Code
NMI	111
RFF, DA, RA	001
TFE	010
LTA	011
BIRQ	100
TO	101

Interrupt Vector

{IBR} Contents		0	0	0	Code		0	0
15		8		5		2		0

Interrupts are sampled by each falling edge of the CPU clock with the last falling edge prior to the start of the next instruction determining whether an interrupt will be processed. The timing of a typical interrupt event is shown in *Figure 2-32*. The interrupt occurs during the current instruction and is sampled by the falling edge of the CPU clock. The next instruction is not operated on and its address is stored in the internal address stack along with [GIE], the ALU flags, and the register bank positions. The address stack is twelve words deep. A two T-state internal call is now executed in place of the non-executed instruction. This call will cause a branch to the interrupt address that is generated in the first half of T-state T1. Also, [GIE] is cleared at the end of the first half of T-state T1. The internal call to the interrupt address is subject to instruction wait states as configured in {DCR}.

2.2.4 Oscillator

The crystal oscillator is an on-chip amplifier which may be used with an external crystal to generate accurate CPU and transceiver clocks. The input to this amplifier is X1, pin 33. The output of the amplifier is X2, pin 34. When X1 and X2 are connected to a crystal and external capacitors *(Figure 2-33)*, the combined circuit forms a Pierce crystal oscillator with the crystal operating at parallel resonance. Crystals that oscillate over the frequency range of 2 MHz to 20 MHz may be used. The recommended crystal parameters for operation with the oscillator are given in Table 2-29. The external capacitor values should be chosen to provide the manufacturer's specified load capacitance for the crystal when combined with the parasitic capacitance of the trace, socket, and package. As an example, a crystal with a specified load capacitance of 20 pF used in a circuit with 13 pF per pin parasitic capacitance will require external capacitor values of 27 pF each. This provides an equivalent capacitance of 40 pF on each side of the crystal, and has a 20 pF series equivalent value across the crystal.

As an alternative to the crystal oscillator, an external clock source may be used. In this case, the external clock source should be connected to X1 and no external circuitry should be connected to X2 *(Figure 2-34)*. The DP8344 can supply a clock source, equal in frequency to the crystal oscillator or external clock source, to other circuitry via pin 35, the CLKOUT output. This output is a buffered version of the signal at X1.

TABLE 2-29

Recommended Crystal Parameters
AT Cut, Parallel Resonant
Fundamental Mode
Load Capacitor = 20 pF
Series Resistance < 20Ω
Frequency Tolerance 0.005% at 25°C
Stability 0.01% 0°–70°C
Drive Level 0.5 mW Typical
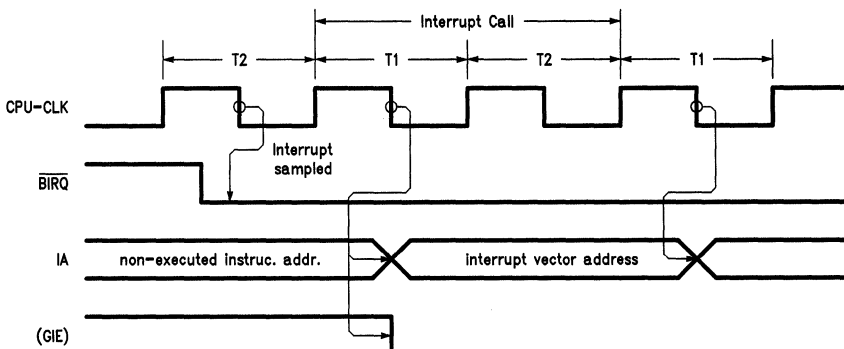
FIGURE 2-32. Interrupt Timing

TL/F/9336–F7

2

2.0 CPU Description (Continued)
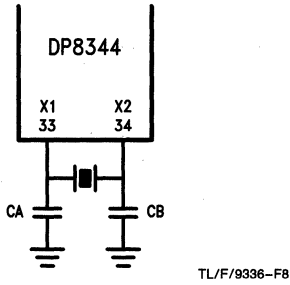
FIGURE 2-33. DP8344A Operation with Crystal

TL/F/9336–F8
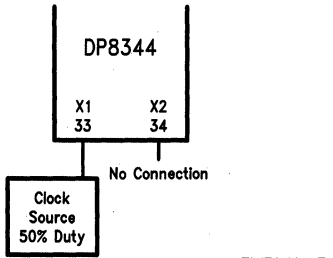
FIGURE 2-34. DP8344A Operation with External Clock

TL/F/9336–F9

3.0 Transceiver

3.1 TRANSCEIVER ARCHITECTURAL DESCRIPTION

The transceiver section operates as an on-chip, independent peripheral, implementing all the necessary formatting required to support the physical layer of the following serial communications protocols:

- IBM 3270 (including 3299)
- IBM 5250
- NSC general purpose 8-bit

The CPU and transceiver are tightly coupled through the CPU register space, with the transceiver appearing to the CPU as a group of special function registers and three dedicated interrupts. The transceiver consists of separate transmitter and receiver logic sections, each capable of independent operation, communicating with the CPU via an asynchronous interface. This interface is software configurable for both polled and interrupt-driven interaction, allowing the system designer to optimize his product for the specific application.

The transceiver connects to the line through an external line interface circuit which provides the required DC and AC drive characteristics appropriate to the application. A block diagram of such an interface is shown in *Figure 3-1*. An on-chip differential analog comparator, optimized for use in a transformer coupled coax interface, is provided at the input to the receiver. Alternatively, if an external comparator is necessary, the input signal may be routed to the DATA-IN pin.
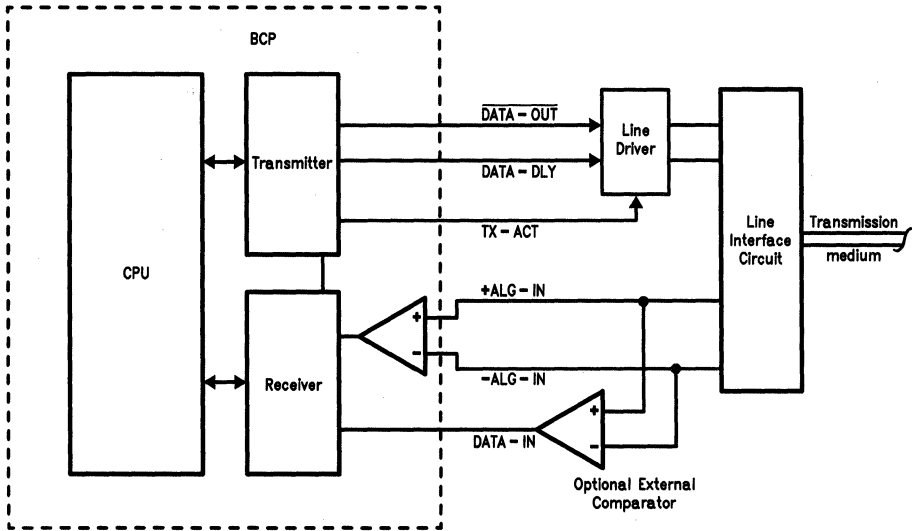
FIGURE 3-1. System Block Diagram, Showing Details of the Line Interface

TL/F/9336–33

3.0 Transceiver (Continued)

The transceiver has several modes of operation. It can be configured for single line, half-duplex operation in which the receiver is disabled while the transmitter is active. Alternatively, both receiver and transmitter can be active at the same time for multi-channel (such as repeater) or loopback operation. The transceiver has both internal and external loopback capabilities, facilitating testing of both the software and external hardware. At all times, both transmitter and receiver operate according to the same protocol definition.

3.1.1 Protocols

In all protocols, data is transmitted serially in discrete messages containing one or more frames, each representing a single word of information. Biphase (Manchester II) encoding is used, in which the data stream is divided into discrete time intervals (bit-times) denoted by a level transition in the center of the bit-time. For the IBM 3270, 3299 and NSC general purpose 8-bit protocols, a mid-bit transition from low to high represents a biphase "1", and a mid-bit transition from high to low represents a biphase "0". For the 5250 protocol, the definition of biphase logic levels is exactly reversed, i.e. a biphase "1" is represented by a high to low transition. Depending on the bit sequence, there may or may not be a transition on the bit-time boundary. The biphase encoding of a simple bit sequence is illustrated in *Figure 3-2(a)*.

Each transmission begins with a unique start sequence consisting of 5 biphase encoded "1's", (referred to as "line quiesce pulses") followed by a 3 bit-time code violation and the sync bit of the first frame, *Figure 3-2(b)*. The three bit-time code violation does not conform to the rules of Manchester encoding and forms a unique recognition pattern for bit time synchronization by the receiver logic. The first bit of any frame is the sync bit, a biphase "1". The frame is then formatted according to the requirements of the protocol. If a multi-frame message is being transmitted, additional frames are appended to the end of the first frame—except for the 5250 protocol, where there may be an optional number of "fill bits" (biphase "0") between each frame.

Depending on the protocol, when all data has been transmitted, the end of a message will be indicated either by the transmission of an ending sequence, or (for 5250) simply by the cessation of transitions on the differential line. Later model 5250 equipment has incorporated a "line hold" at the end of the message. The line hold maintains the final differential state on the line for several bit times to eliminate noise or reflections that could be interpreted as a continuance of the message. The ending sequence for all but 5250 protocols consists of a single biphase "0" followed by a low to high transition on the bit-time boundary and two bit-times with no transitions (two mini-code violation), *Figure 3-2(c)*.

The various protocol framing formats are shown in *Figures 3-3 through 3-5*. The diagrams use a bit pattern drawing convention which, for clarity, shows the bit-time boundaries but not the biphase transitions in the center of the bit times. The timing relationship between the biphase encoded bit stream and the bit pattern diagrams is consistent with *Figure 3-2*.
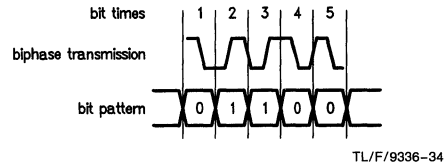
(a) Biphase Encoding

TL/F/9336–34
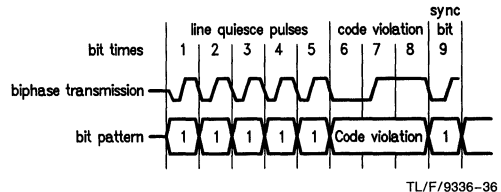
(b) Starting Sequence

TL/F/9336–36
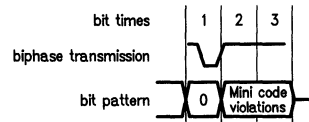
(c) Ending Sequence

TL/F/9336–35

FIGURE 3-2. Biphase Encoding

3.1.1.1 IBM 3270

The framing format of the IBM 3270 coax protocol is shown in *Figures 3-3(a)* and *(b)*, for both single and multi-frame messages. Each message begins with a starting sequence and ends with an ending sequence, as shown in *Figures 3-2(b)* and *(c)*. Each 12-bit frame begins with a sync bit (B1) followed by an 8-bit data byte (MSB first), a 2-bit control field, and the frame delimiter bit (B12), representing even parity on the previous 11 bits. The bit rate on the coax line is 2.3587 MHz.

3.1.1.2 IBM 3299

Adding 3299 multiplexers to the 3270 environment requires an address to be transmitted along with each message from the controller to the multiplexer. The IBM 3299 Terminal Multiplexer protocol provides this capability by defining an additional 8-bit frame as the first frame of every message sent from the controller, as shown in *Figure 3-3(c)*. This frame contains a 6-bit data field along with the normal sync and word parity bits. The protocol currently utilizes bits B2–B4 as an address field that directs the message through the multiplexor hardware. Following the address frame, the rest of the message follows standard 3270 convention. The bit rate, 2.3587 MHz, is the same as standard 3270.

3.1.1.3 IBM 5250

The framing format of the IBM 5250 twinax protocol is shown in *Figure 3-4*, for both single and multi-frame messages. Each message begins with the starting sequence shown in *Figure 3-2(b)*, and ends with 3 fill bits (biphase "0"). A 16-bit frame is employed, consisting of a sync bit
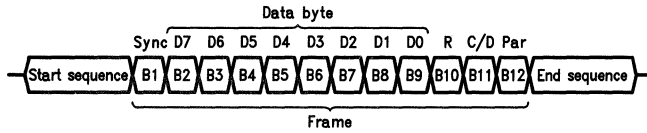
TL/F/9336–37

(a) 3270 Single-Byte Message
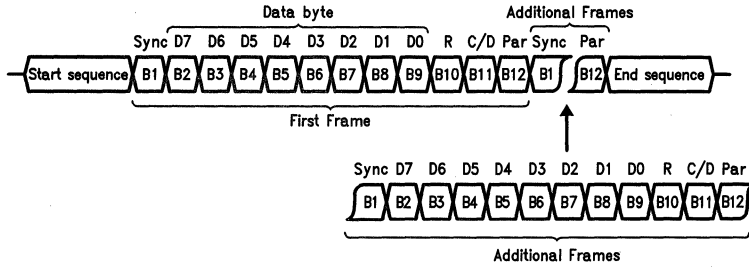
TL/F/9336–38

(b) 3270 Multi-Byte Message

TL/F/9336–39
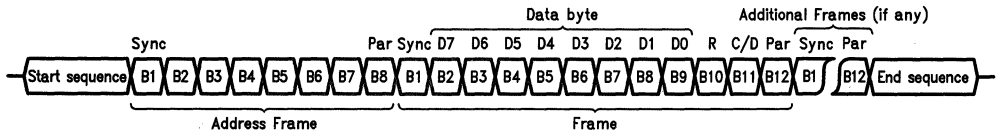
(c) 3299 Controller/Multiplexer Message

FIGURE 3-3. 3270/3299 Protocol Framing Format
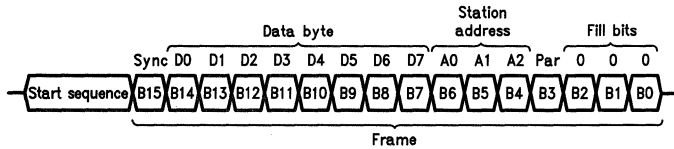
TL/F/9336–40

(a) 5250 Single-Byte Message
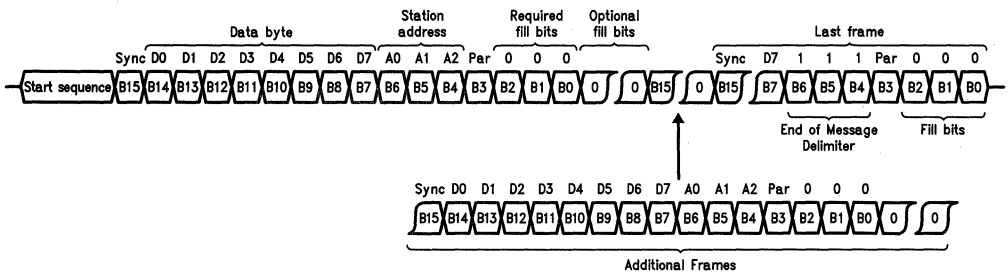
TL/F/9336–41

(b) 5250 Multi-Byte Message

FIGURE 3-4. 5250 Protocol Framing Format

3.0 Transceiver (Continued)

(B15); an 8-bit data byte (B7–B14) (LSB first); a 3-bit station address field (B4–B6); and the last bit (B3) representing even word parity on the previous 12 bits. Following the parity bit, 3 biphase "0" fill bits (B0–B2) are transmitted. Following these required fill bits, up to 240 additional fill bits can be inserted between frames before the next sync bit and the start of the next frame of a multi-byte message. The bit rate on the twinax line is 1 MHz.

3.1.1.4 NSC General Purpose 8-Bit

The framing format of the general purpose 8-bit protocol is shown in *Figure 3-5*, for both single and multi-frame messages. It is identical to that used by the National Semiconductor DP8342 transmitter and DP8343 receiver chips. Each message begins with a starting sequence and ends with an ending sequence, as shown in *Figures 3-2(b)* and *(c)*. A 10-bit frame is employed, consisting of the sync bit (B1); an 8-bit data byte (B2–B9) (LSB first); and the last bit of the frame (B10) representing even word parity on the previous 9 bits. For multiplexed applications, the first frame can be designated as an address frame, with all 8 bits available for the logical address. (See General Purpose 8-bit Modes in this section.)

3.2 TRANSCEIVER FUNCTIONAL DESCRIPTION

A block diagram of the transceiver, revealing external inputs and outputs and details of the CPU interface, is shown in *Figure 3-6*. The transmitter and receiver are largely independent of each other, sharing only the clock, reset and protocol select signals. The transceiver is mapped into the CPU register space, thus the status of the transceiver can always be polled. In addition, the CPU/Transceiver interface can be configured for an interrupt-driven environment. (See Transceiver Interrupts in this section.)

Both transmitter and receiver are reset by a common Transceiver Reset bit, [TRES], allowing the CPU to independently reset the transceiver at any time. The Transceiver is also reset whenever the CPU reset is asserted, including the required power-up reset. When [TRES] is asserted, both transmitter and receiver FIFO's are emptied resulting in the Transmit FIFO Empty flag [TFE] being asserted and the Data Available flag [DAV] cleared. Other flags cleared by [TRES] are Transmit FIFO Full [TFF] and Transmitter Active [TA] in the transmitter and Line Active [LA], Receiver Active [RA], Receiver Error [RE], Receive FIFO Full [RFF], Data Error or Message End [DEME], [POLL], [ACK], and [RAR] command flags in the receiver. When [TRES] is asserted, external pin TX-ACT is cleared, $\overline{\text{DATA-OUT}}$ goes high, and DATA-DLY goes to a state equal to the complement of Transmitter INvert [TIN] in {TMR}. When [TRES] is asserted under software control, it is necessary to wait at least one instruction after asserting [TRES] before seeing the resulting reset state of the affected flags in the CPU. The transmitter and receiver are clocked by a common Transceiver Clock, TCLK, at a frequency equal to eight times the required serial data rate. TCLK can either be obtained from the on-chip oscillator divided by 1, 2 or 4, or from an external clock applied to the X-TCLK pin. TCLK selection is controlled by two Transceiver Clock Select bits, [TCS 1–0] located in the Device Control Register, {DCR}. [TCS 1–0] should only be changed when the transceiver is inactive.

Since the TCLK source can be asynchronous with respect to the CPU clock, the CPU/Transceiver interface can be asynchronous. All flags from the Transceiver are therefore latched at the start of all instructions, and parallel data is transferred through 3 word FIFOs in both the transmitter and receiver.

Protocol selection is controlled by three Protocol Select bits, [PS2–0] in the Transceiver Mode Register, {TMR} (see Table 3-1). Enough flexibility is provided for the BCP to operate in all required positions in the network. It is not pos-

TL/F/9336–42

(a) 8-Bit Single-Byte Message

TL/F/9336–43

(b) 8-Bit Multi-Byte Message
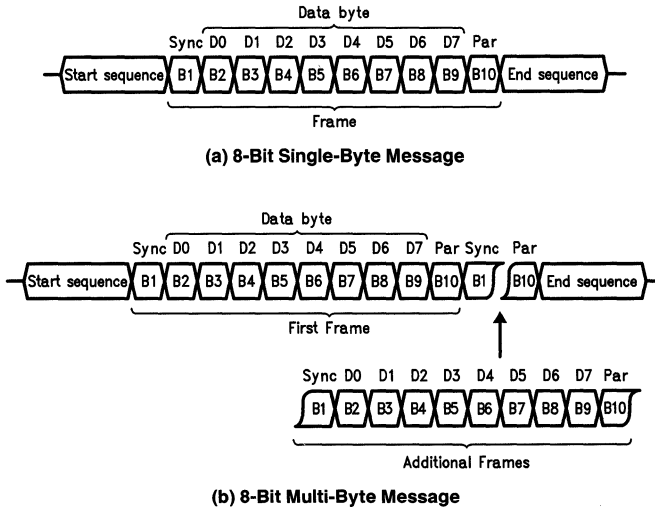
FIGURE 3-5. General Purpose 8-Bit Protocol Framing Format

3.0 Transceiver (Continued)

sible for the transmitter and receiver to operate with different protocols at the same time. The protocol mode should only be changed when both transmitter and receiver are inactive.

If both transmitter and receiver are connected to the same line, they should be configured to operate sequentially (half-duplex). This mode of operation is achieved by clearing the RePeater ENable control bit [RPEN] in {TMR}. In this mode, an active transmitter will disable the receiver, preventing simultaneous operation of transmitter and receiver. If the transmitter FIFO is loaded while the receiver is actively processing an incoming signal, the receiver will be disabled and flag the CPU that a "Receiver Disabled While Active" error has occurred. (See Receiver Errors in this sec-

tion.) On power-up/reset the transceiver defaults to this half-duplex mode.

By asserting the Repeat Enable flag [RPEN], the receiver is not disabled by the transmitter, allowing both transmitter and receiver to be active at the same time. This feature provides for the implementation of a repeater function or loopback for test purposes.

The transmitter output can be connected to the receiver input, implementing a local (on-chip) loopback, by asserting [LOOP]. [RPEN] must also be asserted to enable both the transmitter and receiver at the same time. With [LOOP] asserted, the output TX-ACT is disabled, keeping the external line driver in TRI-STATE. The internal flag [TA] is still enabled, as are the serial data outputs.

TABLE 3-1. Protocol Mode Definition

PS2–0	Protocol Mode	Comments
0 0 0	3270	Standard IBM 3270 protocol.
0 0 1	3299 Multiplexer	Receiver expects first frame to be address frame. Transmitter uses standard 3270, no address frame.
0 1 0	3299 Controller	Transmitter generates address frame as first frame. Receiver expects standard 3270, no address frame.
0 1 1	3299 Repeater	Both transmitter and receiver operate with first frame as address frame.
1 0 0	5250	Non-promiscuous mode. [DAV] asserted only when first frame address matches {ATR}.
1 0 1	5250 Promiscuous	[DAV] asserted on all valid received data without regard to address field.
1 1 0	8-Bit	General-purpose 8-bit protocol with first frame address. Non-promiscuous mode. [DAV] asserted only when first frame address matches {ATR}.
1 1 1	8-Bit Promiscuous	[DAV] asserted on all valid received frames.

DP8344A
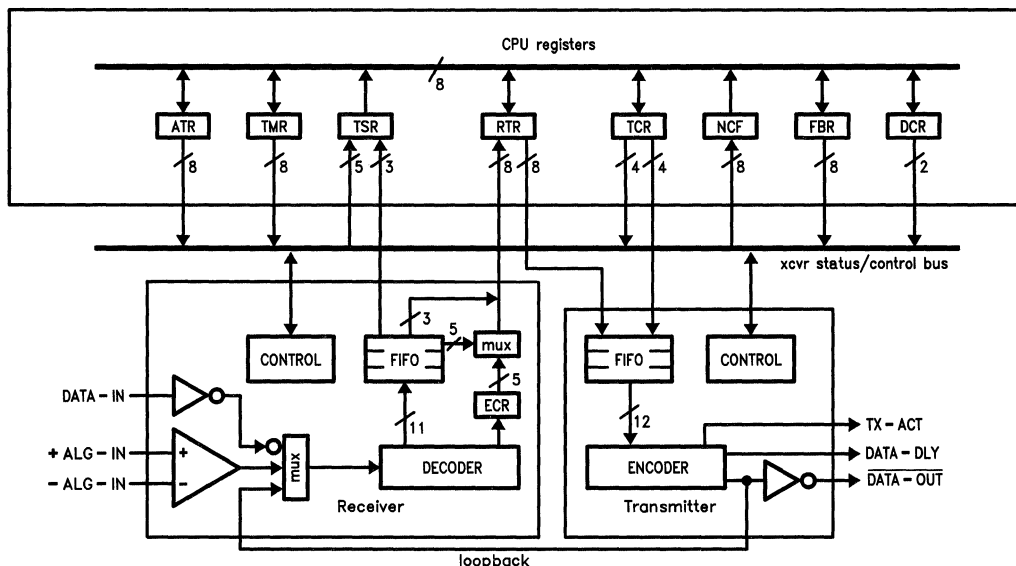
TL/F/9336–44

KEY TO REGISTERS

RTR	Receive/Transmit Register	ATR	Auxiliary Transceiver Register
TSR	Transceiver Status Register	NCF	Network Command Register
TCR	Transceiver Command Register	FBR	Fill-Bit Register
TMR	Transceiver Mode Register	DCR	Device Control Register

FIGURE 3-6. Block Diagram of Transceiver, Showing CPU Interface

3.0 Transceiver (Continued)

3.2.1 Transmitter

The transmitter accepts parallel data from the CPU, formats it according to the desired protocol and transmits it as a serial biphase-encoded bit stream. A block diagram of the transmitter logic is shown in *Figure 3-6*. Two biphase outputs, DATA-OUT, DATA-DLY, and the external line driver enable, TX-ACT, provide the data and control signals for the external line interface circuitry. The two biphase outputs are valid only when TX-ACT is asserted (high) and provide the necessary phase relationship to generate the "predistortion" waveform common to all of the transceiver protocols. See *Figure 3-7* for the timing relationships of these outputs as well as the output of the line driver. For a recommended 3270/3299 coax interface, see Section 3.2.5.1 3270 Line Interface. For a recommended 5250 twinax interface see Section 3.2.5.2 5250 Line Interface.

The capability is provided to invert DATA-OUT and DATA-DLY via the Transmitter Invert bit, [TIN], located in the Transceiver Mode Register, {TMR}. In addition, the timing relationship between TX-ACT and the two biphase outputs can be modified with the Advance Transmitter Active control, [ATA]. When [ATA] is cleared low (the power-up condition), the transmitter generates exactly five line quiesce bits at the start of each message, as shown in *Figure 3-7*. If [ATA] is asserted high, the transmitter generates a sixth line quiesce bit, adding one biphase bit time to the start sequence transmission. The line driver enable, TX-ACT, is asserted halfway through this bit time, allowing an additional half-bit to precede the first full line quiesce of the transmitted waveform. Also, the state of DATA-DLY is such that no predistortion results on the line during this first half line quiesce. This modified start sequence is depicted in the dotted lines shown in *Figure 3-7* and is used to limit the initial transient voltage amplitude when the message begins.

Data is loaded into the transmitter by writing to the Receive/Transmit Register {RTR}, causing the first location of the FIFO to be loaded with a 12-bit word (8 bits from {RTR} and 4 bits from the Transceiver Command Register {TCR}. The data byte to be transmitted is loaded into {RTR}, and {TCR} contains additional information required by the protocol. It is important to note that if {TCR} is to be changed, it must be loaded before {RTR}. A multi-frame transmission is accomplished by sequentially loading the FIFO with the required data, the transmitter taking care of all necessary frame formatting.

If the FIFO was previously empty, indicated by the Transmit FIFO Empty flag [TFE] being asserted, the first word loaded into the FIFO will asynchronously propagate to the last location in approximately 40 ns, leaving the first two locations empty. It is therefore possible to load up the FIFO with three sequential instructions, at which time the Transmit FIFO Full

flag [TFF] will be asserted. If {RTR} is written while [TFF] is high, the first location of the FIFO will be over-written and that data will be destroyed.

When the first word is loaded into the FIFO, the transmitter starts up from idle, asserting TX-ACT and the Transmitter Active flag [TA], and begins generating the start sequence. After a delay of approximately 16 TCLK cycles (2 biphase bit times), the word in the last location of the FIFO is loaded into the encoder and prepared for transmission. If the FIFO was full, [TFF] will be de-asserted when the encoder is loaded, allowing an additional word to be loaded into the FIFO.

When the last word in the FIFO has been loaded into the encoder, [TFE] goes high, indicating that the FIFO is empty. To ensure the continuation of a multi-frame message, more data must then be loaded into the FIFO before the encoder starts the transmission of the last bit of the current frame (the frame parity bit for 3270, 3299, and 8-bit modes; the last of the three mandatory fill bits for 5250). This maximum load time from [TFE] can be calculated by subtracting two from the number of bits in each frame of the respective protocol, and multiplying that result by the bit rate. This number represents the best case time to load—the worst case value is dependent on CPU performance. Since the CPU samples the transceiver flags and interrupts at instruction boundaries, the CPU clock rate, wait states (from programmed wait states, asserting the WAIT pin, or remote access cycles), and the type of instruction currently being executed can affect when the flag or interrupt is first presented to the CPU.

If there is no further data to transmit (or if the load window is missed), the ending sequence (3270/3299/8-bit) is generated and the transmitter returns to idle, de-asserting TX-ACT and [TA]. In 5250 mode, the three required fill bits are sent and TX-ACT and [TA] are de-asserted at a time dependent on the value of bits 7 through 3 of the Auxiliary Transceiver Register {ATR}. If {ATR[7–3]} = 00000, TX-ACT and [TA] are de-asserted at the end of the third required fill bit resulting in no additional "line hold" at the end of the message. Each increment of {ATR[7–3]} results in an additional half bit time of line hold up to a maximum of 15.5 bit times.

Data should not be loaded into the FIFO after the transmitter is committed to ending the message and before the [TA] flag is deasserted. If this occurs, the load will be missed by the transmitter control logic and the word(s) will remain in the FIFO. This condition exists when [TA] and [TFE] are both low at the same time, and can be cleared by resetting the transceiver (asserting [TRES]) or by loading more data into the FIFO, in which case the first frame(s) transmitted will contain the word(s) left in the FIFO from the previous message.

3.0 Transceiver (Continued)

dotted lines indicate waveforms
with [ATA] set high
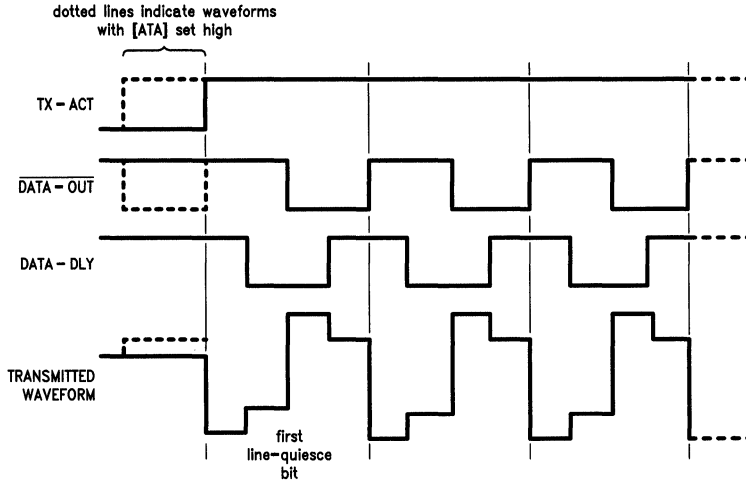
TL/F/9336–45

FIGURE 3-7. Transmitter Output

3.2.2 Receiver

The receiver accepts a serial biphase-encoded bit stream, strips off the framing information, checks for errors and reformats the data for parallel transfer to the CPU. The block diagram in *Figure 3-6* depicts the data flow from the serial input(s) to the FIFO's parallel outputs. Note that the FIFO outputs are multiplexed with the Error Code Register {ECR} outputs.

The receiver and transmitter share the same TCLK, though in the receiver this clock is used only to establish the sampling rate for the incoming biphase encoded data. All control timing is derived from a clock signal extracted from this data. Several status flags and interrupts are made available to the CPU to handle the asynchronous nature of the incoming data stream. See *Figure 3-8* for the timing relationships of these flags and interrupts relative to the incoming data.

The input source to the decoder can be either the on-chip analog line receiver, the DATA-IN input or the output of the transmitter (for on-chip loopback operation). Two bits, the Select Line Receiver [SLR] and Loopback [LOOP], control this selection. For interfacing to the on-chip analog line receiver, see Section 3.2.5.1, 3270 Line Interface. An example of an external comparator circuit for interfacing to twinax cable in 5250 environments is contained in Section 3.2.5.2, 5250 Line Interface. The selected serial data input can be inverted via the Receiver Invert [RIN] control bit.

The receiver continually monitors the line, sampling at a frequency equal to eight times the expected data rate. The Line Active flag [LA] is asserted whenever an input transition is detected and will remain asserted as long as another input transition is detected within 16 TCLK cycles. If another transition is not detected in this time frame, [LA] will be de-asserted. The propagation delay from the occurrence of the edge to [LA] being set is approximately 1 transceiver clock cycle. This function is independent of the mode of operation of the transceiver; [LA] will continue to respond to input signal transitions, even if the transmitter is activated and the receiver disabled.

If the receiver is not disabled by the transmitter or by asserting [TRES], the decoder will adjust its internal timing to the incoming transitions, attempting to synchronize to valid biphase-encoded data. When synchronization occurs, the biphase clock will be extracted and the serial NRZ (Non-Return to Zero) data will be analyzed for a valid start sequence, see *Figure 3-2(b)*. The minimum number of line quiesce bits required by the receiver logic is selectable via the Receiver Line Quiesce [RLQ] control bit. If this bit is set high (the power-up condition), three line quiesce bits are required; if set low, only two are needed. Once the start sequence has been recognized, the receiver asserts the Receiver Active flag [RA] and enables the error detection circuitry. The propagation delay from the occurrence of the mid-bit edge of the sync bit in the starting sequence to [RA] being set is approximately 3 transceiver clock cycles.

The NRZ serial bit stream is now clocked into a serial to parallel shift register and analyzed according to the expected data pattern as defined by the protocol. If no errors are detected by the word parity bit, the parallel data (up to a total of 11-bits, depending on the protocol) is passed to the first location of the FIFO. It then propagates asynchronously to the last location in approximately 40 ns, at which time the Data Available flag [DAV] is asserted, indicating to the CPU that valid data is available in the FIFO. The propagation delay from the occurrence of the mid-bit edge of the parity bit of the frame to [DAV] being set is approximately 5 transceiver clock cycles.

Of the possible 11-bits in the last location of the FIFO, 8-bits (data byte) are mapped into {RTR} and the remaining bits (if any) are mapped into the Transceiver Status Register {TSR [2–0]}. The CPU accesses the data byte by reading {RTR}, and the 5250 address field or 3270 control bits by reading {TSR}. When reading the FIFO, it is important to note that {TSR} must be read before {RTR}, since reading {RTR} advances the FIFO. Once [DAV] has been recognized as set by the CPU, the data can be read by any instruction with {RTR} as the source. All instructions with {RTR} as the source (except BIT, CMP, JRMK, JMP reg-

2

3.0 Transceiver (Continued)
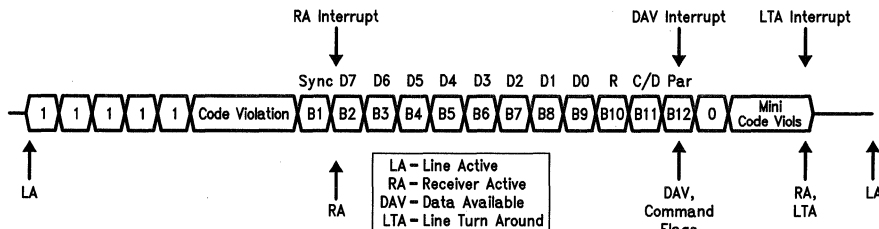
FIGURE 3-8. Timing of Receiver Flags Relative to Incoming Data

TL/F/9336–46

ister, LJMP conditional, and LCALL conditional) will result in popping the last location of the FIFO, presenting a new word (if present) for future CPU access. Data in the FIFO will propagate from one location to the next in approximately 10–15 ns, therefore the CPU is easily able to unload the FIFO with a set of consecutive instructions.

If the received bit stream is a multi-byte message, the receiver will continue to process the data and load the FIFO. After the third load (if the CPU has not accessed the FIFO), the Receive FIFO Full flag [RFF] will be asserted. The propagation delay from the occurrence of the mid-bit edge of the parity bit of the frame to [RFF] being set is approximately 5 transceiver clock cycles. If there are more than 3 frames in the incoming message, the CPU has approximately one frame time (sync bit to start of parity bit) to start unloading the FIFO. Failure to do so will result in an overflow error condition and a resulting loss of data (see Receiver Errors).

If there are no errors detected, the receiver will continue to process the incoming frames until the end of message is detected. The receiver will then return to an inactive state, clearing [RA] and asserting the Line Turn-Around flag, [LTA] indicating that a message was received with no errors. The propagation delay from the occurrence of the edge starting the first minicode violation to [RA] cleared and [LTA] set is approximately 17 transceiver clock cycles in 3270, 3299, and 8-bit modes. In 5250 modes, the assertion of [LTA] and clearing of [RA] are dependent on how the transmission line ends after the transmission of the three required fill bits (see 5250 Modes). For the 3270 and 3299 protocols, [LTA] can be used to initiate an immediate transmitter FIFO load; for the other protocols, an appropriate response delay time may be needed. [LTA] is cleared by loading the transmitter's FIFO, writing a one to [LTA] in the Network Command flag register, or by asserting [TRES].

Receiver Errors

If the Receiver Active flag, [RA], is asserted by the receiver logic, the selected receiver input source is continuously checked for errors, which are reported to the CPU by asserting the Receiver Error flag, [RE], and setting the appropriate receiver error flag in the Error Code Register {ECR}. If a condition occurs which results in multiple errors being created, only the first error detected will be latched into {ECR}. Once an error has been detected and the appropriate error flag has been set, the receiver is disabled, clearing [RA] and preventing the Line Turn-Around flag and interrupt

[LTA] from being asserted. The Line Active flag [LA] remains asserted if signal transitions continue to be detected on the input.

5 error flags are provided in {ECR}:

7	6	5	4	3	2	1	0
rsv	rsv	rsv	OVF	PAR	IES	LMBT	RDIS

[OVF] **Overflow**—Asserted when the decoder writes to the first location of the FIFO while [RFF] is asserted. The word in the first location will be over-written; there will be no effect on the last two locations.

[PAR] **Parity Error**—Asserted when a received frame fails an even (word) parity check.

[IES] **Invalid Ending Sequence**—Asserted during an expected end sequence when an error occurs in the mini code-violation. Not valid in 5250 modes.

[LMBT] **Loss of Mid-Bit Transition**—Asserted when the expected biphase-encoded mid-bit transition does not occur within the expected window. Indicates a loss of receiver synchronization.

[RDIS] **Receiver Disabled While Active**—Asserted when an active receiver is disabled by the transmitter being activated.

To determine which error has occurred, the CPU must read {ECR}. This is accomplished by asserting the Select Error Codes control bit, [SEC], and reading {RTR}. The {ECR} is only 5 bits wide, therefore the upper 3 bits are still the output of the receive FIFO (see Figure 3-6). All instructions with {ECR} as the source (except BIT, CMP, JRMK, JMP register, LJMP conditional, and LCALL conditional) will clear the error condition and return the receiver to idle, allowing the receiver to again monitor the incoming data stream for a new start sequence. The [SEC] control bit must be de-asserted to read the FIFO's data from {RTR}.

If data is present in the FIFO when the error occurs, the Data Available flag [DAV] is de-asserted when the error is detected and re-asserted when {ECR} is read. Data present in the FIFO before the error occurred is still available to the CPU. The flexibility is provided, therefore, to read the error type and still recover data loaded into the FIFO before the error occurred. The Transceiver Reset, [TRES] can be asserted at any time, clearing both Transceiver FIFOs and the error flags.

3.0 Transceiver (Continued)

3.2.3 Transceiver Interrupts

The transceiver has access to 3 CPU interrupt vectors, one each for the transmitter and receiver, and a third, the Line Turn-Around interrupt, providing a fast turn around capability between receiver and transmitter. The receiver interrupt is the CPU's highest priority interrupt (excluding NMI), followed by the transmitter and Line Turn-Around interrupts, respectively. The three interrupt vector addresses and a full description of the interrupts are given in Table 3-2.

The receiver interrupt is user-selectable from 4 possible sources (only 3 used at present) by specifying a 2-bit field, the Receiver Interrupt Select bits [RIS1-0] in the Interrupt Control Register {ICR}. A full description is given in Table 3-3.

The RFF + RE interrupt occurs only when the receive FIFO is full (or an error is detected). If the number of frames in a received message is not exactly divisible by 3, one or two words could be left in the FIFO at the end of the message, since the CPU would receive no indication of the presence of that data, it is recommended that this interrupt be used together with the line turn-around interrupt, whose service routine can include a test for whether any data is present in the receive FIFO.

For additional information concerning interrupts, refer to Sections 2.1.1.3, Interrupt Control Registers, and 2.2.3, Interrupts.

3.2.4 Protocol Modes

3270/3299 Modes

As shown in Table 3-1, the transceiver can operate in 4 different 3270/3299 modes, to accommodate applications of the BCP in different positions in the network. The 3270 mode is designed for use in a device or a controller which is not in a multiplexed environment. For a multiplexed network, the 3299 multiplexer and controller modes are designed for each end of the controller to multiplexer connection, the 3299 repeater mode being used for an in-line repeater situated between controller and multiplexer.

For information on how parallel data loaded into the transmit FIFO and unloaded from the receive FIFO maps into the serial bit positions, see *Figure 3-9*.

To transmit a frame, {TCR [3–0]} must first be set up with the correct control information, after which the data byte can be written to {RTR}. The resulting composite 12-bit word is loaded into the transmit FIFO where it propagates through to the last location to be loaded into the encoder and formatted for transmission.

When formatting a 3270 frame, {TCR [2]} controls whether the transmitter is required to format a data frame or a command frame. If {TCR [2]} is low, the transmitter logic calcu-

TABLE 3-2. Transceiver Interrupts

Interrupt	Vector Address	Description
Receiver	000100	User selectable from 4 possible sources, see Table 3-3.
Transmitter	001000	Set when [TFE] asserted, indicating that the transmit FIFO is empty, cleared by writing to {RTR}. **Note:** [TRES] causes [TFE] to be asserted.
Line Turn-Around	001100	Set when a valid end sequence is detected, cleared by writing to {RTR}, writing a one to [LTA], or asserting [TRES]. In 5250 modes, interrupt is set when the last fill bit has been received and no further input transitions are detected. Will not be set in 5250 or 8-bit non-promiscuous modes unless an address match was received.

The interrupt vector is obtained by concatenating {IBR} with the vector address as shown:

	IBR					0	0		vector address				interrupt vector			

15   8   5   0
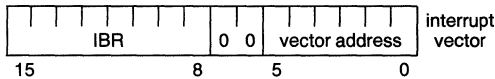
TABLE 3-3. Receiver Interrupts

Interrupt	RIS1,0	Description
RFF + RE	0 0	Set when [RFF] or [RE] asserted. If activated by [RFF], indicating that the receive FIFO is full, interrupt is cleared by reading from {RTR}. If activated by [RE], indicating that an error has been detected, interrupt is cleared by reading from {ECR}.
DAV + RE	0 1	Set when [DAV] or [RE] asserted. If activated by [DAV], indicating that valid data is present in the receive FIFO, interrupt is cleared by reading from {RTR}. If activated by [RE], indicating that an error has been detected, interrupt is cleared by reading from {ECR}.
Not Used	1 0	Reserved for future product enhancement.
RA	1 1	Set when [RA] asserted, indicating the receipt of a valid start sequence, cleared by reading {ECR} or {RTR}.

All receiver interrupts can be cleared by asserting [TRES].

3.0 Transceiver (Continued)

lates odd parity on the data byte (B2–B9) and transmits this value for B10. If {TCR [2]} is high, B10 takes the state of {TCR [0]}. Odd Word Parity [OWP] controls the type of parity calculated on B1–B11 and transmitted as B12, the frame delimiter. If [OWP] is high, odd parity is output; otherwise even parity is transmitted. In this manner the system designer is provided with maximum flexibility in defining the transmitted 3270 control bits (B10–B12).

When data is written to {RTR}, the least significant 4 bits of {TCR} are loaded into the FIFO along with the data being

written to {RTR}. The same {TCR} contents can therefore be used for more than one frame of a multi-frame transmission, or changed for each frame.

When a 3270 frame is received and decoded, the decoder loads the parallel data into the receive FIFO where it propagates through to the last location and is mapped into {RTR} and {TSR}. Bits B2–B11 are exactly as received; Byte Parity [BP] is odd parity on B2–B9, calculated in the decoder. Reading {RTR} will advance the receive FIFO, therefore {TSR} must be read first if this information is to be utilized.
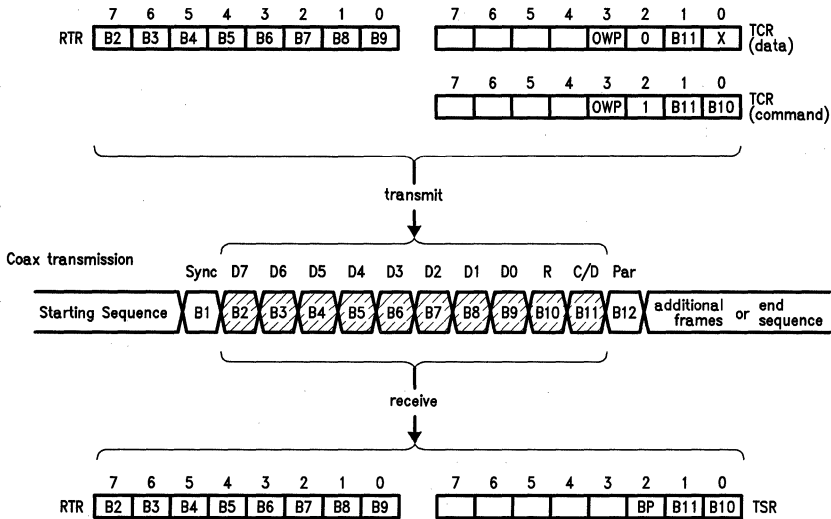
TL/F/9336–47

(a) 3270 Data and Command Frames

TL/F/9336–48
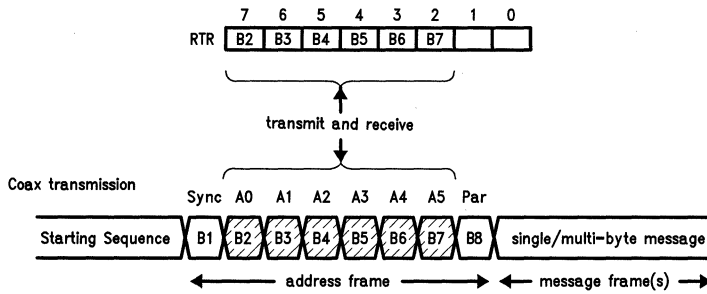
(b) 3299 Address Frame
FIGURE 3-9. 3270/3299 Frame Assembly/Disassembly Procedure

3.0 Transceiver (Continued)

When formatting a 3299 address frame, the procedure is the same as for a 3270 frame, with {RTR [7–2]} defining the address to be transmitted. The only bit in {TCR} which has any functional meaning in this mode is [OWP], which controls the type of parity required on B1–B8. Similarly, when the receiver de-formats a 3299 address frame, the received address bits are loaded into {RTR [7–2]}; {RTR [1–0]} and {TSR [2–0]} are undefined.

The POLL, POLL/ACK and TT/AR flags in the Network Command Flag Register are valid only in 3270 and 3299 (excluding the 3299 address frame) modes. These flags are decodes of their respective coax commands as defined in Table 3-4. The Data Error or Message End [DEME] flag (also in the {NCF} register) indicates different information depending on the selected protocol. In 3270 and 3299, [DEME] is set when B10 of the received frame does not match the locally generated odd parity on bits B2–B9 of the received frame. [DEME] is not part of the receiver error logic, it functions only as a status flag to the CPU. These flags are decoded from the last location in the FIFO and are valid only when [DAV] is asserted; they are cleared by reading {RTR} and must be checked before advancing the receiver FIFO.

5250 Modes

The biphase data is inverted in the 5250 protocol relative to 3270/3299 (see the Protocol section—IBM 5250). Depending on the external line interface circuitry, the transceiver's biphase inputs and outputs may need to be inverted by asserting the [RIN] (Receiver INvert) and [TIN] (Transmitter INvert) control bits in {TMR}.

For information on how data must be organized in {TCR} and {RTR} for input to the transmitter, and how data extracted from a received frame is organized by the receiver and mapped into {TSR} and {RTR}, see *Figure 3-10*.

To transmit a 5250 message, the least significant 4 bits of {TCR} must first be set up with the correct address and parity control information. The station address field (B4–B6) is defined by {TCR[2–0]}, and [OWP] controls the type of parity (even or odd) calculated on B4–B15 and transmitted as B3. When the 8-bit data byte is written to {RTR}, the resulting composite 12-bit word is loaded into the transmit FIFO, starting the transmitter. The same {TCR} contents can be used for more than one frame of a multi-frame transmission, or changed for each frame.

The 5250 protocol defines bits B0–B2 as fill bits which the transmitter automatically appends to the parity bit (B3) to

TABLE 3-4. Decode of 3270 Coax Commands

Received Word										Flag	Description
B2	B3	B4	B5	B6	B7	B8	B9	B10	B11		
0	0	0	0	0	0	0	0	0	0	RAR	TT/AR (Clean Status) Received
X	X	X	1	0	0	0	1	X	1	ACK	POLL/ACK Command Received
X	X	X	0	0	0	0	1	X	1	POLL	POLL Command Received

All flags cleared by reading {RTR}.
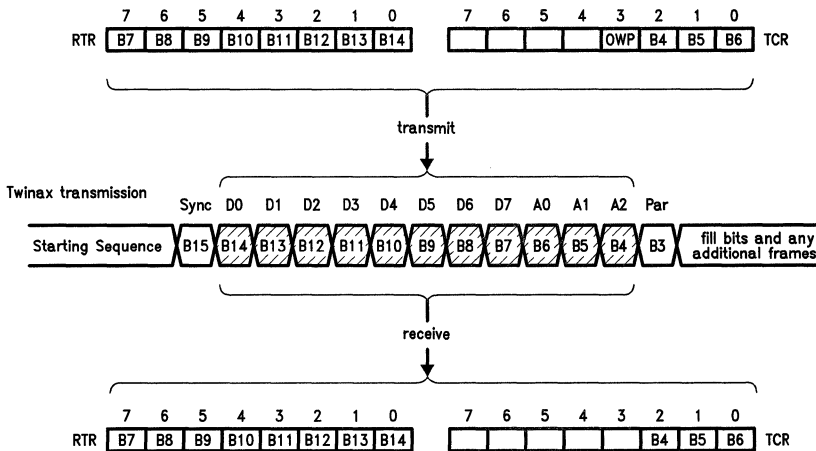
FIGURE 3-10. 5250 Frame Assembly/Disassembly Description

TL/F/9336–49

3.0 Transceiver (Continued)

form the 16-bit frame. Additional fill bits may be inserted between frames of a multi-frame transmission by loading the fill bit register, {FBR}, with the one's complement of the number of fill bits to be transmitted. A value of FF (hex), corresponds to the addition of no extra fill bits. At the conclusion of a message the transmitter will return to the idle state after transmitting the 3 fill bits of the last frame (no additional fill bits will be transmitted).

As shown in Table 3-1, the transceiver can operate in 2 different 5250 modes, designated "promiscuous" and "non-promiscuous". The transmitter operates in the same manner in both modes.

In the promiscuous mode, the receiver passes all received data to the CPU via the FIFO, regardless of the station address. The CPU must determine which station is being addressed by reading {TSR [2–0]} before reading {RTR}.

In the non-promiscuous mode, the station address field (B4–B6) of the first frame must match the 3 least significant bits of the Auxiliary Transceiver Register, {ATR [2–0]}, before the receiver will pass the data on to the CPU. If no match is detected in the first frame of a message, and if no errors were found on that frame, the receiver will reset to idle, looking for a valid start sequence. If an address match is detected in the first frame of a message, the received data is passed on to the CPU. For the remainder of the message all received frames are decoded in the same manner as the promiscuous mode.

To maintain maximum flexibility, the receiver logic does not interpret the station address or command fields in determining the end of a 5250 message. The message typically ends with no further line transitions after the third fill bit of the last frame. This end of message must be distinguished from a loss of synchronization between frames of a multi-byte transmission condition by looking for line activity some time after the loss of synchronization occurs. When the loss of synchronization occurs during fill bit reception, the receiver monitors the Line Active flag, [LA], for up to 11 biphase bit times (11 μs at the 1 MHz data rate). If [LA] goes inactive at any point during this period, the receiver returns to the idle state, de-asserting [RA] and asserting [LTA]. If, however, [LA] is still asserted at the end of this window, the receiver interprets this as a real loss of synchronization and flags the [LMBT] error condition to the CPU. (See Receiver Errors in this section.)

In the 5250 modes, the Data-Error-or-Message-End [DEME] flag is a decode of the 111 station address (the end of message delimiter) and is valid only when [DAV] is asserted. This function allows the CPU to quickly determine when the end of message has been received.

The transmitter has the flexibility of holding TX-ACT active at the end of a 5250 message, thus reducing line reflections and ringing during this critical time period. The amount of hold time is programmable from 0 μs to 15.5 μs in 500 ns increments (assuming TCLK is 8 MHz), and is set by writing the selected value to the upper 5-bits of the Auxiliary Transceiver Register, {ATR [7–3]}.

General Purpose 8-Bit Modes

As shown in Table 3-1, the transceiver can operate in 2 different 8-bit modes, designated "promiscuous" and "non-promiscuous". In the non-promiscuous mode, the first frame data byte (B2–B9) must match the contents of {ATR[7–0]} before the receiver will load the FIFO and assert [DAV]. If no match is made on the first frame, and if no errors were found on that frame, the receiver will go back to idle, looking for a valid start sequence. The address comparator logic is not enabled in the promiscuous mode, and therefore all received frames are passed through the receive FIFO to the CPU. The transmitter operates in the same manner in both modes.

The serial bit positions relative to the parallel data loaded into the transmit FIFO and presented to the CPU by the receiver FIFO are shown in *Figure 3-11*. To transmit a frame, the data byte is written to {RTR}, loading the transmit FIFO where it propagates through to the last location to be loaded into the encoder and formatted for transmission. Only [OWP] in {TCR} is loaded into the transmitter FIFO in both protocol modes; {TCR [2–0]} are don't cares. B10 is defined by a parity calculation on B1–B9; odd if [OWP] is high and even if [OWP] is low.

When a frame is received, the decoder loads the processed data into the receive FIFO where it propagates through to the last location and is mapped into {RTR}. All bits are exactly as received. Reading the data is accomplished by reading {RTR}. {TSR [2–0]} are undefined in the 8-bit modes.
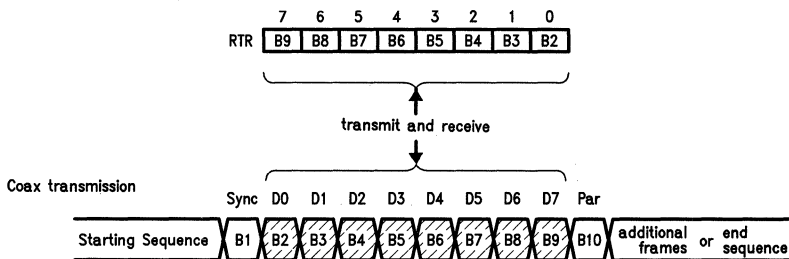
TL/F/9336–50

FIGURE 3-11. General Purpose 8-Bit Frame Assembly/Disassembly Procedure

3.0 Transceiver (Continued)

3.2.5 Line Interface

3.2.5.1 3270 Line Interface

In the 3270 environment, data is transmitted between a control unit and a device via a single coax cable or twisted pair cable. The coax type is RG62AU with a maximum length of 1.5 kilometers. The twisted pair cable has become more prevalent to reduce cabling and routing costs. Typically, a 24 AWG unshielded twisted pair is used to achieve the cost reduction goals. The length of the twisted pair cable is a minimum of 100 feet to a maximum of 900 feet. The 3270 protocol utilizes a transformer to isolate the peripheral from the cabling system.

An effective line interface design must be able to accept either coax or twisted pair cabling and compensate for noise, jitter and reflections in the cabling system. There must be an adequate amount of jitter tolerance to offset the effects of filtering and noise. Some filtering is needed to reduce ambient noise caused by surrounding hardware. Such filtering must not introduce transients that the receiver comparator translates into data jitter.

An effective driver design should also attempt to compensate for the filtering effects of the cable. Higher data frequencies become attenuated more than lower frequency signals as cable length is increased, yielding greater disparity in the amplitudes of these signals. This effect generates greater jitter at the receiver. The 3270 signal format allows for a high voltage (predistorted) magnitude and a low voltage (nondistorted) magnitude within each data bit time. Increasing the predistorted-to-nondistorted signal level ratio counteracts the filtering phenomenon because the lower frequency signals contain less predistortion than do higher frequency signals. Thus, the amplitude of the higher frequency signals is "boosted" more than the lower frequency signals. Unfortunately, a low signal level is more susceptible to reflection-induced errors at short cable length. Proper impedance matching and slower edge rates must be utilized to eliminate as much reflection as possible at these lengths.

Additionally, shielded or balanced operation must be adequately supported. Shielded operation implies the use of coax cable, where balanced implies the use of twisted pair cable. Proper termination should be employed, and a termination slightly greater than the characteristic impedance of the line may actually provide more desirable waveforms

than a perfectly matched termination. Board layout should make the comparator lines as short as possible. Lines should be placed closely together to avoid the introduction of differential noise. These lines should not pass near "noisy" lines. A ground plane should isolate all "noisy" lines.

BCP Design

The line interface design for the receiver is shown in *Figure 3-12.* An offset of approximately 17 mV separates the comparator inputs, making the receiver more immune to ambient noise present on the circuit board. A 2:1:1 (arranged as a 3:1) transformer increases any voltage sensitivity lost by introducing the offset. A bandpass filter is employed to reduce edge rate to the comparator and eliminate ambient noise. The bandwidth (30 kHz to 30 MHz) was chosen to provide sufficient attenuation for noise while producing minimum data jitter.

The driver design, *Figure 3-13,* incorporates a National Semiconductor DS3487 and a resistor network to generate the proper signal levels. The predistorted-to-nondistorted ratio was chosen to be about 4.5 to 1. The coax/twisted pair front end, *Figure 3-14,* includes an ADC brand connector to switch between coax and twisted pair cable. The coax interface has the shield capacitively coupled to ground. The 510Ω resistor and the filter loading produce a termination of about 95Ω. The twisted pair interface balances both lines and possesses an input impedance of about 100Ω. This termination is somewhat higher than the characteristic impedance (about 96Ω) of twisted pair. Terminations of this type produce reflections that do not tend to generate mid-bit errors. Such terminations have the benefit of creating a larger voltage at the receiver over longer cable lengths. For a more detailed explanation of the 3270 line interface, see Application Note "A Combined Coax/Twisted Pair 3270 Line Interface for the DP8344 Biphase Communications Processor".

3.2.5.2 5250 Line Interface

The 5250 environment utilizes twinax in a multi-drop configuration, where eight devices can be "daisy-chained" over a total distance of 5,000 feet and eleven splices, (each physical device is considered a splice). Twinax connectors are bulky and expensive, but are very sturdy. Twinaxial cable is a shielded twisted pair that is nearly $\frac{1}{3}$ of an inch thick.

Legend

Ⓐ To coax/twisted pair front end
Ⓑ To line driver circuitry
Ⓒ To BCP comparator
* Includes board capacitance
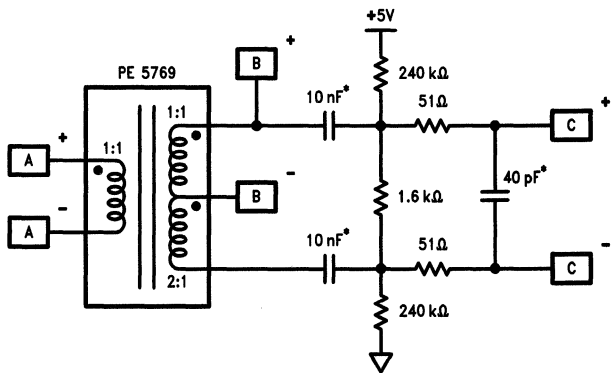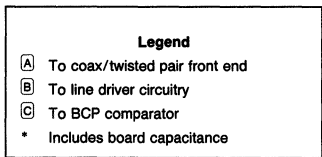
FIGURE 3-12. BCP Receiver Design

TL/F/9336–G1

3.0 Transceiver (Continued)
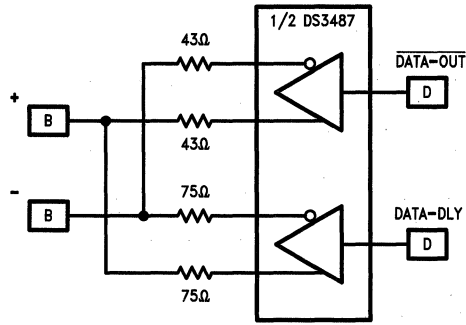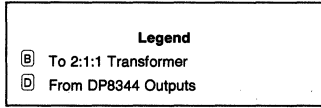
FIGURE 3-13. BCP Driver Design

Legend
B To 2:1:1 Transformer
D From DP8344 Outputs

TL/F/9336-G2

Legend
A To 2:1:1 Transformer
Switch Open — Twisted Pair
Switch Closed — Coax
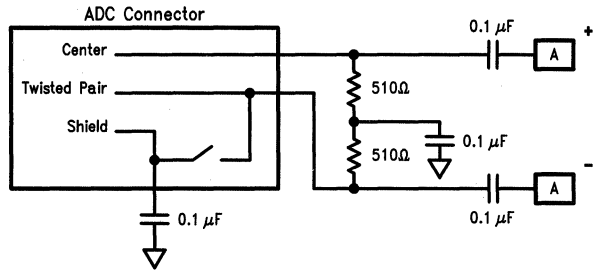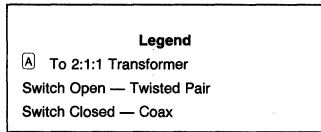
TL/F/9336-G3

FIGURE 3-14. BCP Coax/Twisted Pair Front End

The cable shield must be continuous throughout the transmission system, and be grounded at the system unit and each station. Since twinax connectors have exposed metal connected to their shield grounds, care must be taken not to expose them to noise sources. The polarity of the two inner conductors must also be maintained throughout the transmission system.

The transmission system is implemented in a balanced current mode; every receiver/transmitter pair is directly coupled to the twinax at all times. Data is impressed on the transmission line by unbalancing the line voltage with the driver current. The system requires passive termination at both ends of the transmission line. The termination resistance value is given by:

$R_t = Z_{O/2}$; where

R_t: Termination Resistance

Z_O: Characteristic Impedance

In practice, termination is accomplished by connecting both conductors to the shield via 54.9Ω, 1% resistors; hence the characteristic impedance of the twinax cable of 107Ω ±5% at 1.0 MHz. Intermediate stations must not terminate the line; each is configured for "pass-through" instead of "terminate" mode. Stations do not have to be powered on to pass twinax signals on to other stations; all of the receiver/transmitter pairs are DC coupled. Consequently, devices must never output any signals on the twinax line during power-up or down that could be construed as data, or interfere with valid data transmission between other devices.

Driver Circuits for the DP8344A

The transmitter interface on the DP8344A is sufficiently general to allow use in 3270, 5250, and 8-bit transmission systems. Because of this generality, some external hardware is needed to adapt the outputs to form the signals necessary to drive the twinax line. The chip provides three signals: DATA-OUT, DATA-DLY and TX-ACT. DATA-OUT is biphase serial data (inverted). DATA-DLY is the biphase serial data output (non-inverted) delayed one-quarter bit-time. TX-ACT, or transmitter active, signals that serial data is being transmitted when asserted. DATA-OUT and DATA-DLY can be used to form the A and B phase signals with their three levels by the circuit shown in Figure 3-15. TX-ACT is used as an external transmitter enable. The BCP can invert the sense of the DATA-OUT and DATA-DLY signals by asserting [TIN] {TMR[3]}. This feature allows both 3270 and 5250 type biphase data to be generated, and/or utilization of inverting on non-inverting transmitter stages.

Drivers for the 5250 environment may not place any signals on the transmission system when not activated. The power-on and off conditions of drivers must be prevented from causing noise on the system since other devices may be in operation. Figure 3-15 shows a "DC power good" signal enabling the driver circuit. This signal will lock out conduction in the drivers if the supply voltage is out of tolerance.

Twinax signals can be viewed as consisting of two distinct phases, phase A and phase B, each with three levels, off,

3.0 Transceiver (Continued)

high and low. The off level corresponds with 0 mA current being driven, the high level is nominally 62.5 mA, +20% −30%, and the low level is nominally 12.5 mA, +20% −30%. When these currents are applied to a properly terminated transmission line the resultant voltages impressed at the driver are: off level is 0V, low level is 0.32V ±20%, high level is 1.6V ±20%. The interface must provide for switching of the A and B phases and the three levels. A bimodal constant current source for each phase can be built that has a TTL level interface for the BCP.

Receiver Circuits

The pseudo-differential mode of the twinax signals make receiver design requirements somewhat different than the coax 3270 world. Hence, the analog receiver on the BCP is not well suited to receiving twinax data. The BCP provides both analog inputs to an on-board comparator circuit as well as a TTL level serial data input, DATA-IN. The sense of this serial data can be inverted by the BCP by asserting [RIN], {TMR[4]}.

The external receiver circuit must be designed with care to ensure reliable decoding of the bit-stream in the worst environment. Signals as small as 100 mV must be detected. In order to receive the worst case signals, the input level switching threshold or hysteresis for the receiver should be nominally 29 mV ±20%. This value allows the steady state, worst case signal level of 100 mV ±66% of its amplitude before transitioning.

To achieve this, a differential comparator with complementary outputs can be applied, such as the National LM361. The complementary outputs are useful in setting the hysteresis or switching threshold to the appropriate levels. The LM361 also provides excellent common mode noise rejection and a low input offset voltage. Low input leakage current allows the design of an extremely sensitive receiver, without loading the transmission line excessively.

In addition to good analog design techniques, a low pass filter with a roll-off of approximately 1 MHz should be applied to both the A and B phases. This filter essentially conducts high frequency noise to the opposite phase, effectively making the noise common mode and easily rejectable.

Layout considerations for the LM361 include proper bypassing of the ±12V supplies at the chip itself, with as short as possible traces from the pins to 0.1 μF ceramic capacitors. Using surface mount chip capacitors reduces lead inductance and is therefore preferable in this case. Keeping the input traces as short and even in length is also important. The intent is to minimize inductance effects as well and standardize those effects on both inputs. The LM361 should have as much ground plane under and around it as possible. Trace widths for the input signals especially should be as wide as possible; 0.1 inch is usually sufficient. Finally, keep all associated discrete components nearby with short routing and good ground/supply connections.

For a more detailed explanation of the 5250 line interface, see application note "Interfacing the DP8344 to Twinax."
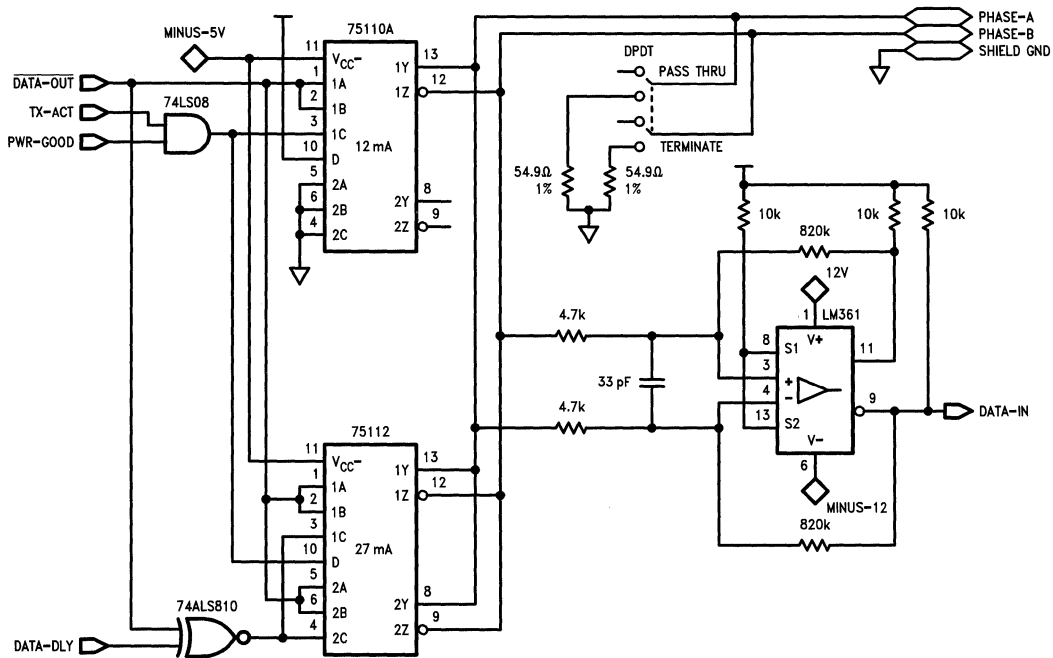
TL/F/9336–G4

FIGURE 3-15. 5250 Line Interface Schematic

4.0 Remote Interface and Arbitration System (RIAS)

INTRODUCTION

Communication with the BCP is based on the BCP's ability to share its data memory. A microprocessor (or any intelligent device) can read and write to any BCP data location while the BCP CPU is executing instructions. This capability is part of the BCP's Remote Interface and Arbitration System (RIAS). Sharing data memory is possible because RIAS's arbitration logic allocates use of the BCP's data and address buses. RIAS has been designed so that accesses of BCP data memory by another device minimally impact its performance as well as the BCP's. In addition to data memory accesses, RIAS allows another device to control how BCP programs are loaded, started and debugged.

4.1 RIAS ARCHITECTURAL DESCRIPTION

Interfacing to the BCP is accomplished with the control signals listed in Table 4-1. *Figure 4-1* shows the BCP interfaced to Instruction Memory, Data Memory, and an intelligent device, termed the Remote Processor (RP). Instruction and Data are separate memory systems with separate address buses and data paths. This arrangement allows continuous instruction fetches without interleaved data accesses. Instruction Memory (IMEM) is interfaced to the BCP through the Instruction (I) and Instruction Address (IA) buses. IMEM is 16 bits wide and can address up to 64k memory. Data Memory (DMEM) is eight bits wide and can also address up to 64k memory. The DMEM address is formed by the 8-bit upper byte (A bus) and the 8-bit lower byte (AD bus). The AD bus must be externally latched because it also serves as the path for data between the BCP and DMEM.

The Remote Processor's address and data buses are connected to the BCP's address and data buses through the bus control circuitry. The RP's address lines decode a chip select for the BCP called Remote Access Enable ($\overline{\text{RAE}}$). Basically, the BCP's Data Memory has been memory mapped into the RP's memory. A Remote Access of the BCP occurs when $\overline{\text{REM-RD}}$ or $\overline{\text{REM-WR}}$, along with $\overline{\text{RAE}}$ is asserted low. $\overline{\text{REM-RD}}$ and $\overline{\text{REM-WR}}$ can be directly connected to the Remote Processor's read and write lines, or for more complicated systems the $\overline{\text{REM-RD}}$ and $\overline{\text{REM-WR}}$ signals may be controlled by a combination of address decode and the RP's read and write signals. To the RP, an access of the BCP will appear as any other memory system access. This configuration allows the RP to read and write Data Memory, read and write the BCP's Program Counter, and read and write BCP Instruction Memory. These functions are selected by control bits in the Remote Interface Configuration register {RIC}. This register can be accessed only by the RP and not by the BCP CPU. If the Remote Processor executes a remote access with the Command input (CMD) high, {RIC} is accessed through the BCP's AD bus.

In *Figure 4-1*, the Remote Processor's address lines are decoded to form the CMD input. When a remote access takes place with CMD low, the memory system designated in {RIC} is accessed. *Figure 4-2* shows the contents of {RIC}. The two least significant bits are the Memory Select bits [MS1–0] which designate the type of remote access: to Data Memory, the Program Counter, or Instruction Memory. This register also contains the BCP start bit [STRT], three interface select bits [FBW, LR, LW], the Single-Step bit [SS], and the Bi-directional Interrupt Status bit [BIS]. Refer to the RIAS Reference Section for a more detailed description of the contents of this register and the function of each bit.
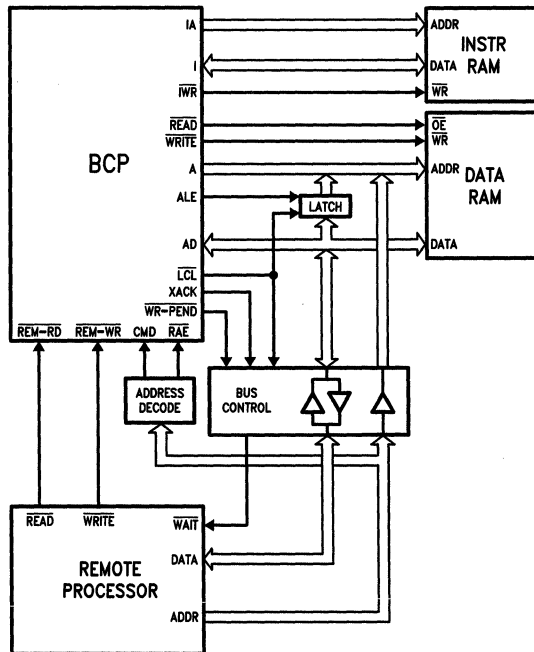
TL/F/9336–19

FIGURE 4-1. BCP/Remote Processor Interface

4.0 Remote Interface and Arbitration System (RIAS) (Continued)

TABLE 4-1. RIAS Inputs and Outputs

Signal	In/Out	Pin	Reset State	Function
CMD	In	45	X	CoMmanD input. When high, remote accesses are directed to the Remote Interface Configuration register, {RIC}. When low, remote accesses are directed to Data Memory, Instruction Memory or the Program Counter as determined by {RIC [1,0]}.
$\overline{\text{LCL}}$	Out	31	0	LoCaL. Normally low, goes high when the BCP relinquishes the data and address bus to service a remote access.
$\overline{\text{LOCK}}$	In	44	X	Asserting this input Low will **LOCK** out local (BCP) accesses to Data Memory. Once the remote processor has been granted the bus, $\overline{\text{LOCK}}$ gives it sole access to the bus and BCP accesses are "waited".
$\overline{\text{RAE}}$	In	46	X	Remote Access Enable. Setting this input low allows host access of BCP functions and memory.
$\overline{\text{REM-RD}}$	In	47	X	REMote ReaD. When low along with $\overline{\text{RAE}}$, a remote read cycle is requested; serviced by the BCP when the data bus becomes available.
$\overline{\text{REM-WR}}$	In	48	X	REMote WRite. When low along with $\overline{\text{RAE}}$, a remote write cycle is requested; serviced by the BCP when the data bus becomes available.
$\overline{\text{WR-PEND}}$	Out	49	1	WRite PENDing. In a system configuration where remote write cycles are latched, $\overline{\text{WR-PEND}}$ will go low, indicating that the latches contain valid data which have yet to be serviced by the BCP.
XACK	Out	50	1	Transfer ACKnowledge. Normally high, goes low on $\overline{\text{REM-RD}}$ or $\overline{\text{REM-WR}}$ going low (if $\overline{\text{RAE}}$ low) returning high when the transfer is complete. Normally used as a "wait" signal to a remote processor. (In the Latched Write mode, XACK will only transition if a second remote access begins before the first one completes.)
$\overline{\text{WAIT}}$	In	54	X	Asserting this input low will add wait states to both remote accesses and to the BCP instruction cycle. $\overline{\text{WAIT}}$ will extend a remote access until it is set high.

```
 7   6   5   4   3    2    1    0
┌───┬───┬────┬───┬───┬─────┬─────┬─────┐
│BIS│SS │FBW │LR │LW │STRT │MS1  │MS0  │ RIC
└───┴───┴────┴───┴───┴─────┴─────┴─────┘
```

BIS   —Bidirectional Interrupt Status
SS   —Single-Step
FBW   —Fast Buffered Write mode
LR   —Latched Read mode
LW   —Latched Write mode
STRT   —BCP CPU start/stop
MS1–0 —Memory Selection
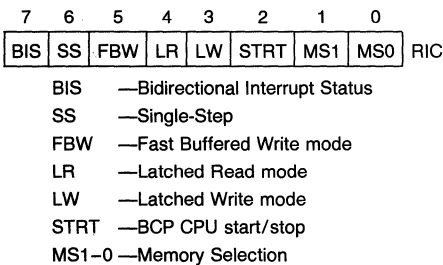
FIGURE 4-2. Remote Interface Control Register

4.1.1 Remote Arbitration Phases

The BCP CPU and RIAS share the internal CPU-CLK. This clock is derived from the X1 crystal input. It can be divided by two by setting [CCS] = 1 in {DCR} or run undivided by setting [CCS] = 0. The frequency at which the Remote Processor is run need not bear any relationship to the CPU-CLK. A remote access is treated as an asynchronous event and data is handshaked between the Remote Processor and the BCP.

The two key handshake signals involved in the BCP/RP interface are Transfer Acknowledge (XACK) and Local ($\overline{\text{LCL}}$). Internally, two more signals control the access timing: $\overline{\text{INT-READ}}$ and $\overline{\text{INT-WRITE}}$. The timing for a generic Remote Access is shown in *Figure 4-3*. A remote access is

TL/F/9336–20
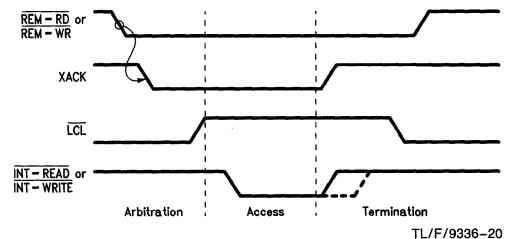
FIGURE 4-3. Generic Remote Access ($\overline{\text{RAE}}$ = 0)

initiated by the RP asserting $\overline{\text{REM-RD}}$ or $\overline{\text{REM-WR}}$ with $\overline{\text{RAE}}$ low. There is no set-up/hold time relationship between $\overline{\text{RAE}}$ and $\overline{\text{REM-RD}}$ or $\overline{\text{REM-WR}}$. These signals are internally gated together such that if RAE (REM-RD + REM-WR) is true, a remote access will begin. A short delay later, XACK will fall. This signal can be fed back to the RP's wait line to extend its read or write cycle, if necessary. When the BCP's

4.0 Remote Interface and Arbitration System (RIAS) (Continued)

arbitration logic determines that the BCP is not using data memory, $\overline{LCL}$ rises, relinquishing control of the address and data buses to the RP. The remote access can be delayed at most one BCP instruction (providing [LOR] is not set high). If the CPU is executing a string of data memory accesses, RIAS has an opportunity to break in at the completion of every instruction. The time period between $\overline{REM\text{-}RD}$ or $\overline{REM\text{-}WR}$ being asserted (with $\overline{RAE}$ low) and $\overline{LCL}$ rising is called the Arbitration Phase. It is a minimum of one T-state, but can be increased if the BCP CPU is accessing Data Memory (local access) or if the BCP has set the Lock Out Remote bit [LOR].

The CMD pin is internally latched on the first falling edge of the CPU-CLK after a remote access has been initiated by asserting $\overline{RAE}$ low along with asserting $\overline{REM\text{-}RD}$ or $\overline{REM\text{-}WR}$ low. If the remote interface is asynchronous, the CMD signal must be valid simultaneously or before $\overline{RAE}$ is asserted low along with $\overline{REM\text{-}RD}$ or $\overline{REM\text{-}WR}$ being asserted low. The value of CMD is only sampled once during each remote access and will remain in effect for the duration of the remote access.

After the Arbitration Phase has ended, the Access Phase begins. Either Data Memory, Instruction Memory, the Program Counter, or {RIC} is read or written in this phase. Either $\overline{INT\text{-}READ}$ or $\overline{INT\text{-}WRITE}$ will fall one T-state after $\overline{LCL}$ rises. These two signals provide the timing for the different types of accesses. $\overline{INT\text{-}READ}$ times the transitions on the AD bus for Remote Reads and forms the external $\overline{READ}$ line. $\overline{INT\text{-}WRITE}$ clocks data into the PC and {RIC} and forms the $\overline{IWR}$ and $\overline{WRITE}$ lines. $\overline{INT\text{-}READ}$ and $\overline{INT\text{-}WRITE}$ rise with XACK, or shortly after.

The duration of the Access Phase depends on the type of memory being accessed. Data Memory and Instruction Memory accesses are subject to any programmed wait states and all remote accesses are waited by asserting $\overline{WAIT}$ low. The minimum time in the Access Phase is 2 T-states.

The rising edge of XACK indicates the Access Phase has ended and the Termination Phase has begun. If the RP was doing a read operation, this edge indicates that valid data is available to the RP. During the Termination Phase the BCP is regaining control of the buses. $\overline{LCL}$ falls one T-state after XACK and since the RP is no longer being waited, it can deassert $\overline{REM\text{-}RD}$ or $\overline{REM\text{-}WR}$. The duration of this phase is a minimum of one T-state, but can be extended depending on the interface mode chosen in {RIC}.

4.1.2 Access Types

There are four types of accesses an RP can make of the BCP:

—Remote Interface Control Register {RIC}
—Data Memory (DMEM)
—Program Counter (PC)
—Instruction Memory (IMEM)

An access of {RIC} is accomplished by asserting $\overline{RAE}$ and $\overline{REM\text{-}RD}$ or $\overline{REM\text{-}WR}$ with the CMD pin asserted high. The Remote Interface Configuration register is accessed through the AD bus as shown in Figure 4-4(c). A read or write of {RIC} can take place while the BCP CPU is executing instructions. Timing for this access is shown in Figures 4-4(a) and (b). Note that in the Remote Read Figure 4-4(a), AD does not transition. This is because the contents of {RIC} are active on the bus by default. The AD bus is in

TRI-STATE during a Remote Write Figure 4-4(b) while $\overline{LCL}$ is high. The byte being written to {RIC} is latched on the rising edge of XACK and can be seen on AD after $\overline{LCL}$ falls.

The Access Phase, in this case, is always two T-states (unless $\overline{WAIT}$ is low) because {RIC} is not subject to any programmed wait states.
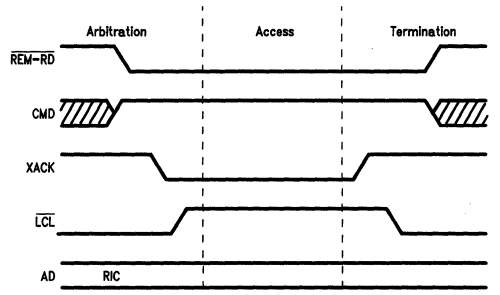
TL/F/9336-80

(a) Remote Read Timing ($\overline{RAE}$ = 0)
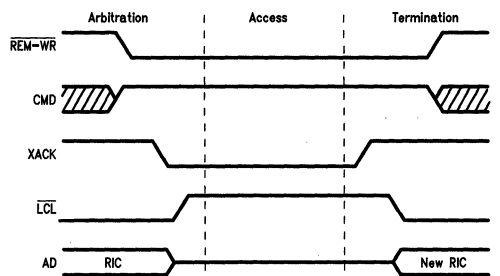
TL/F/9336-81

(b) Remote Write Timing ($\overline{RAE}$ = 0)
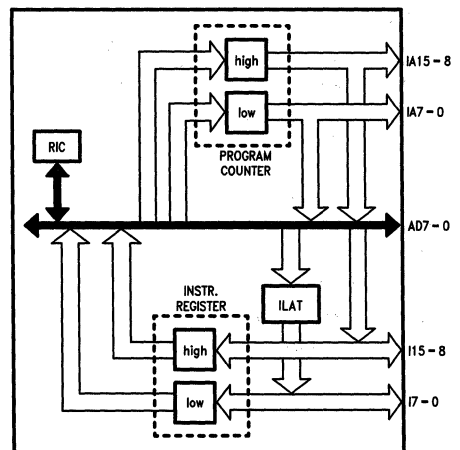
TL/F/9336-82

(c) RIC to AD Connectivity

FIGURE 4-4. Generic RIC Access

4.0 Remote Interface and Arbitration System (RIAS) (Continued)

Remote Accesses other than to {RIC} are accomplished with the CMD pin low in conjunction with asserting $\overline{RAE}$ low along with $\overline{REM\text{-}WR}$ or $\overline{REM\text{-}RD}$ being taken low. The type of access performed is defined by the Memory Select bits in {RIC}, as shown in *Figure 4-5*.

7	6	5	4	3	2	1	0
BIS	SS	FBW	LR	LW	ST	MS1	MS0
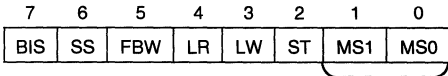
Memory Select Bits
00 - Data Memory
01 - Instruction Memory
10 - PC low byte
11 - PC high byte

FIGURE 4-5. Memory Select Bits in {RIC}

Reads or writes of Data Memory (DMEM) are preceded by setting the Memory Select bits in {RIC} for a DMEM access: [MS1,0] = 00. After that, the RP simply reads or writes to BCP Data Memory as many times as it needs to. A DMEM access, as well as a {RIC} access, can be made while the BCP CPU is executing instructions. All other accesses must be executed with the BCP CPU stopped.

The timing for a Data Memory read and write are shown in *Figure 4-6*. The access is initiated by asserting $\overline{RAE}$ and $\overline{REM\text{-}RD}$ or $\overline{REM\text{-}WR}$ while CMD is low. The BCP responds by bringing its address and data lines into TRI-STATE and allowing the RP to control DMEM. $\overline{READ}$ is asserted in the Access Phase of a Remote Read *Figure 4-6(a)*. It will stay low for a minimum of one T-state, but can be extended by adding programmable data wait states or by taking $\overline{WAIT}$ low. $\overline{WRITE}$ is asserted in the Access Phase with a remote write. It too is a minimum of one T-state and can be increased by adding programmable wait states or by taking $\overline{WAIT}$ low.

Figure 4-7(c) shows the data path from the Program Counter to the AD bus. Both high and low PC bytes can be written or read through AD. The RP has independent control of the high and low bytes of the Program Counter—the byte being accessed is specified in the Memory Select bits. The high byte of the PC is accessed by setting [MS1–0] = 11. Setting [MS1–0] = 10 allows access to the low byte of the PC. After the Memory Select bits are set by a Remote Write to {RIC}, the byte selected can be read or written by the RP by executing a Remote Access with CMD low. Remote accesses to both the high and low bytes of the PC, as well as the instruction memory access must be executed with the BCP CPU idle. Four accesses by the RP are necessary to read or write both the high and low bytes of the PC. Timing for a PC access is shown in *Figure 4-7(a)* and *(b)*. The PC becomes valid on a Remote Read *(a)* one T-state after $\overline{LCL}$ rises and one T-state before XACK rises. AD is in TRI-STATE while $\overline{LCL}$ is high for a Remote Write *(b)*. Time in the Access Phase is two T-states if $\overline{WAIT}$ is not asserted.

Instruction memory (IMEM) is accessed through another internal path: from AD to the I bus, shown in *Figure 4-8(c)*. The memory is accessed first low byte, then high byte. Low and high bytes of the 16-bit I bus are alternately accessed for Remote Reads. An 8-bit holding register, ILAT, retains

TL/F/9336–83

(a) Remote Read Timing ($\overline{RAE}$ = 0)

TL/F/9336–84
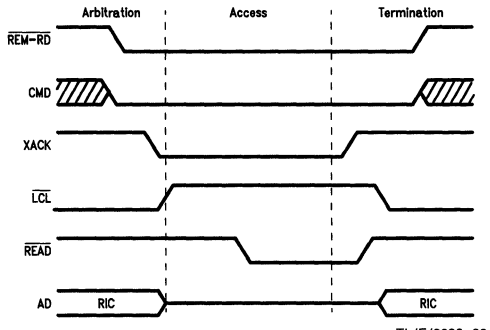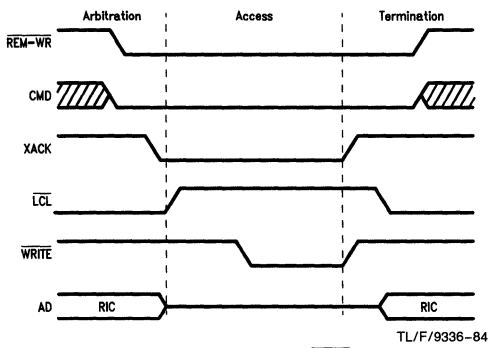
(b) Remote Write Timing ($\overline{RAE}$ = 0)

FIGURE 4-6. Generic DMEM Access

the low byte until the high byte is written by the Remote Processor for the write to IMEM. The BCP increments the PC after the high byte has been accessed.

Timing for an IMEM access is shown in *Figure 4-8(a)* and *(b)*. As before, the Memory Select bits are first set to instruction memory: [MS1–0] = 01. It is only necessary to set [MS1–0] once for repeated IMEM accesses. (Instruction Memory is the power-up Memory Selection state.) A simple state machine keeps track of which instruction byte is expected next—low or high byte. The state machine powers up looking for the low instruction byte and every IMEM access causes this state machine to switch to the alternate byte. Accesses other than to IMEM will not cause the state machine to switch to the alternate byte and only a BCP reset will force the state machine to the "low byte state".

Figure 4-8(a) shows a Remote Read of Instruction memory. Both the low byte, then the high byte can be seen on back to back remote reads. An instruction byte becomes active on the AD bus one T-state after $\overline{LCL}$ rises and is valid when XACK rises. This time period will be a minimum of one T-state, but can be extended up to three more T-states by instruction wait states.
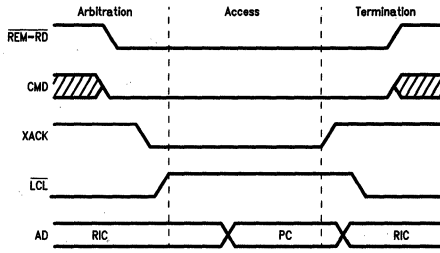
2

4.0 Remote Interface and Arbitration System (RIAS) (Continued)

(a) Remote Read Timing ($\overline{RAE}$ = 0)
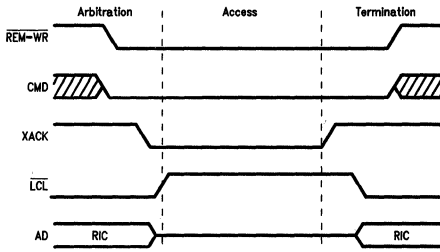
TL/F/9336-85

(b) Remote Write Timing ($\overline{RAE}$ = 0)
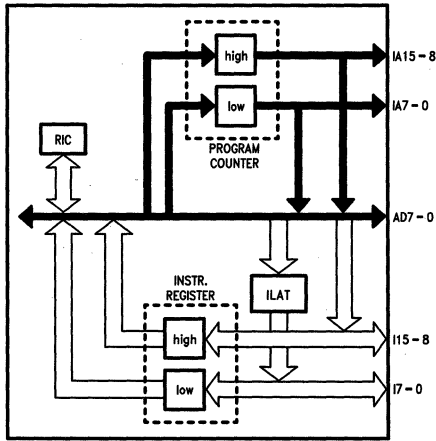
TL/F/9336-86

(c) IA to AD Connectivity

TL/F/9336-87

FIGURE 4-7. Generic PC Access

4.0 Remote Interface and Arbitration System (RIAS) (Continued)
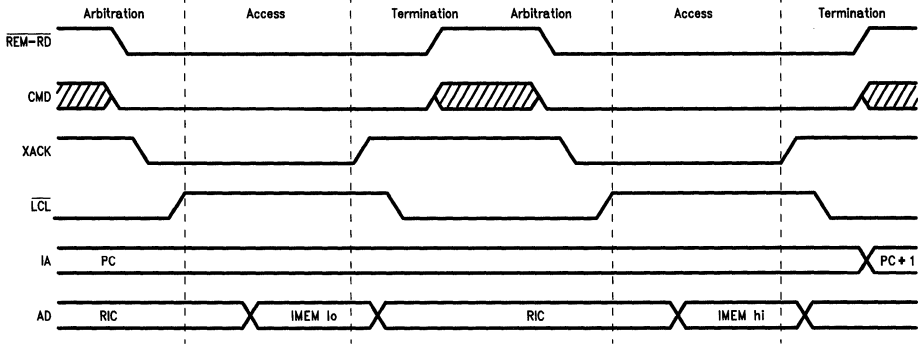
TL/F/9336–88

(a) Remote Read Timing ($\overline{RAE} = 0$)
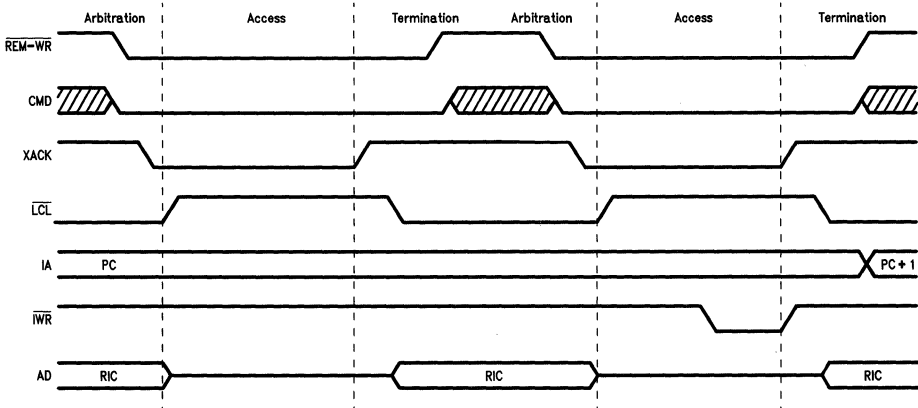
TL/F/9336–89

(b) Remote Write Timing ($\overline{RAE} = 0$)
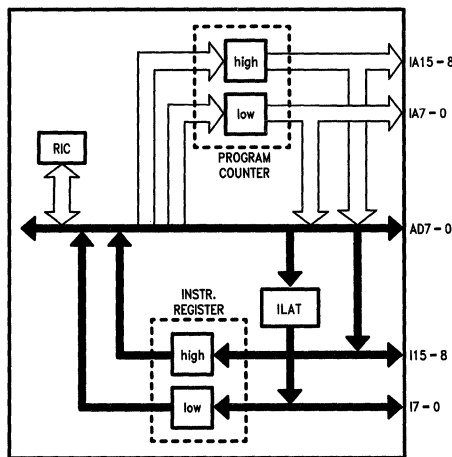
TL/F/9336–90

(c) I to AD Connectivity

FIGURE 4-8. Generic IMEM Access

2

4.0 Remote Interface and Arbitration System (RIAS) (Continued)

In addition, $\overline{\text{WAIT}}$ can delay the rising edge of XACK indefinitely. One T-state after XACK rises, {RIC} will once again be active on AD. Timing is similar for a Remote Write. AD is in TRI-STATE while $\overline{\text{LCL}}$ is high. $\overline{\text{LCL}}$ is asserted for a minimum of three T-states, but can be extended by instruction wait states and the $\overline{\text{WAIT}}$ pin. $\overline{\text{IWR}}$ clocks the instruction into memory during the write of the high byte. The Instruction Address (PC) is incremented about one T-state after $\overline{\text{LCL}}$ falls on a high byte access for both Remote Reads and Writes.

Soft-loading Instruction Memory is accomplished by first setting the BCP Program Counter to the starting address of the program to be loaded. The Memory Select bits are then set to IMEM. BCP instructions can then be moved from the Remote Processor to the BCP—low byte, high byte—until the entire program is loaded.

4.1.3 Interface Modes

The Remote Interface and Arbitration System will support TRI-STATE buffers or latches between the Remote Processor and the BCP. The choice between buffers and latches depends on the type of system that is being interfaced to. Latches will help prevent the faster system from slowing to the speed of the slower system. Buffers can be used if the Remote Processor (RP) requires that data be handshaked between the systems.

Figure 4-9 shows the timing of Remote Reads via a buffer *(a)* and a latch *(b)* (called a Buffered Read and Latched Read). The main difference in these modes is in the Termination Phase. The Buffered Read handshakes the data back to the RP. When the BCP deasserts XACK, data is valid and the RP can deassert $\overline{\text{REM-RD}}$. Only after $\overline{\text{REM-RD}}$ goes high is $\overline{\text{LCL}}$ removed. In the Latched Read *Figure 4-9(b)* XACK rises at the same time, but the Termination Phase completes without waiting for the rising edge of $\overline{\text{REM-RD}}$. One half T-state after XACK rises, $\overline{\text{INT-READ}}$ ris-

es and one half T-state later $\overline{\text{LCL}}$ falls. The BCP can use the buses one T-state after $\overline{\text{LCL}}$ falls. The minimum time (no wait states, no arbitration delay) the BCP CPU could be prevented from using the bus is four T-states in the Latched Read Mode.

A Buffered Read prevents the BCP CPU from using the bus during the time RP is allocated the buses. This time period begins when $\overline{\text{LCL}}$ rises and ends when $\overline{\text{REM-RD}}$ is removed. If the $\overline{\text{REM-RD}}$ is asserted longer than the minimum Buffered Read execution time (four T-states), then the BCP may be unnecessarily prevented from using the buses. Therefore, if there are no overriding reasons to use the Buffered Read Mode, the Latched Read Mode is preferable.

There are three Remote Write Modes—two require buffers and one requires latches. The timing for the writes utilizing buffers is shown in *Figure 4-10*. The Slow Buffered Write *(a)* is handshaked in the same manner as the Buffered Read and thus has the same timing. The Fast Buffered Write has similar timing to the Latched Read. This timing similarity exists because the BCP terminates the remote access without waiting for the RP to deassert $\overline{\text{REM-WR}}$.

In both cases, XACK falls a short delay after $\overline{\text{REM-WR}}$ falls and $\overline{\text{LCL}}$ rises when the RP is given the buses. One T-state after $\overline{\text{LCL}}$ rises, $\overline{\text{INT-WRITE}}$ falls. The termination in the Slow Buffered Write mode keys off $\overline{\text{REM-WR}}$ rising, as shown in *Figure 4-10(a)*. $\overline{\text{INT-WRITE}}$ rises a prop-delay later and $\overline{\text{LCL}}$ falls one T-state later. The Fast Buffered Write, shown in *Figure 4-10(b)*, begins the Termination Phase with the rising edge of XACK. $\overline{\text{INT-WRITE}}$ rises at the same time as XACK, and $\overline{\text{LCL}}$ falls one T-state later. The BCP can begin a local access one T-state after $\overline{\text{LCL}}$ transitions.

A Fast Buffered Write is preferable to the Slow Buffered Write if RP's write cycles are slow compared to the minimum Fast Buffered Write execution time. The Fast Buffered Write assumes, though, that data is available to the BCP by the time $\overline{\text{INT-WRITE}}$ rises.

TL/F/9336–91

(a) Buffered Read

TL/F/9336–92

(b) Latched Read
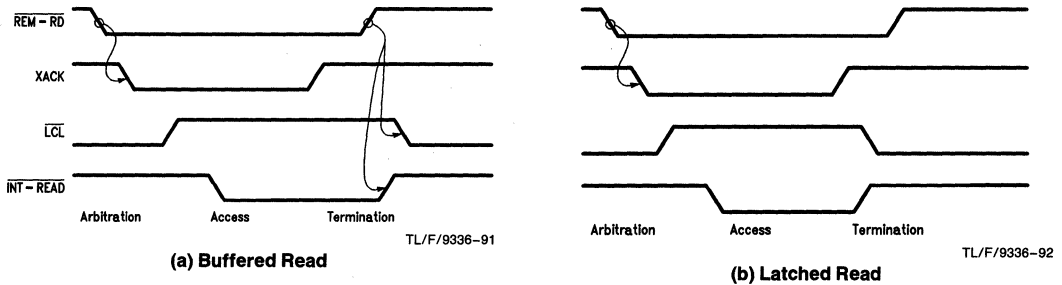
FIGURE 4-9. Read from Remote Processor

4.0 Remote Interface and Arbitration System (RIAS) (Continued)

TL/F/9336-93

(a) Slow Buffered Write

TL/F/9336-94
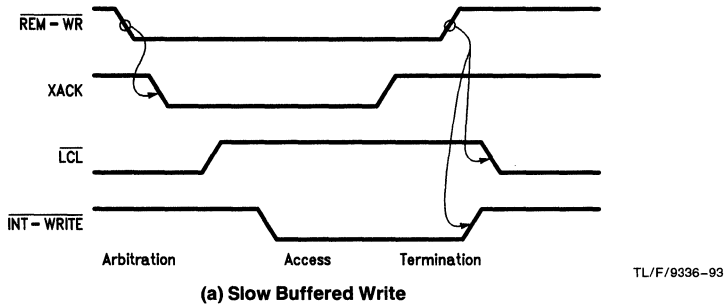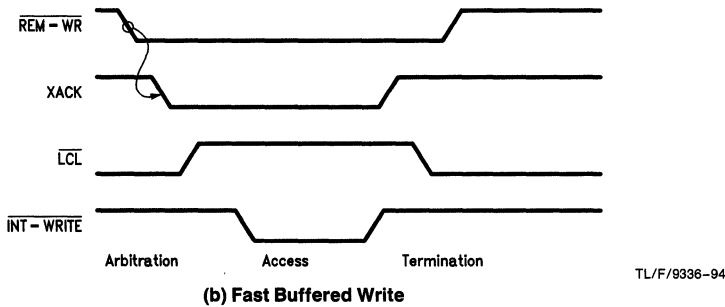
(b) Fast Buffered Write

FIGURE 4-10. Buffered Write from Remote Processor

In both Buffered Write Modes, XACK is asserted to wait the RP. The Latched Write Mode makes it possible for the RP to write to the BCP without getting waited. The timing for the Latched Write Mode is shown in *Figure 4-11*. When the Remote Processor writes to the BCP, its address and data buses are externally latched on the rising edge of REM-WR. Even though REM-WR has been asserted XACK does not

TL/F/9336-95

FIGURE 4-11. Latched Write from Remote Processor

switch. The BCP only begins remote access execution after the trailing edge of REM-WR. Since the RP is not requesting data back from the BCP, it can continue execution without waiting for the BCP to complete the remote access. After REM-WR is deasserted, WR-PEND is taken low to prevent overwrite of the latches. A minimum of two T-states later LCL switches and AD, A, and the external address latch go into TRI-STATE, allowing the latches which contain the remote address and data to become active. If the RP attempts to initiate another access before the current write is complete, XACK is taken low to wait the RP and the address and the data are safe because WR-PEND prevents the latches from opening. The Access Phase ends when INT-WRITE rises and the data is written. One T-state later, LCL falls and one T-state after that WR-PEND rises. If another access is pending, it can begin in the next T-state. This is indicated by XACK rising when WR-PEND rises.

A minimum BCP/RP interface utilizes four TRI-STATE buffers or latches. A block diagram of this interface is shown in *Figure 4-12*. The blocks A, B, C, and D indicate the location of buffers or latches. Blocks A and B isolate 16 bits of the RP's address bus from the BCP's Data Address bus. Two more blocks, C and D, bidirectionally isolate 8 bits of the RP's data bus from the BCP AD bus.

2

4.0 Remote Interface and Arbitration System (RIAS) (Continued)

FIGURE 4-12. Minimum BCP/Remote Processor Interface

TL/F/9336–96

The BCP Remote Arbitrator State Machine (RASM) must know what hardware interfaces to the RP in order to time the remote accesses correctly. To accomplish this, three Interface Mode bits in {RIC} are used to define the hardware interface. These bits are the Latched Write bit [LW], the Latched Read bit [LR] and the Fast Buffered Write bit [FBW]. See *Figure 4-13*.

7	6	5	4	3	2	1	0
BIS	SS	FBW	LR	LW	ST	MS1	MS0
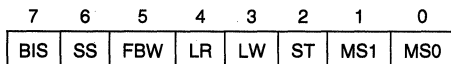
Interface Mode Bits

– 0 – - Buffered Read

– 1 – - Latched Read

0 – 0 - Slow Buffered Write

1 – 0 - Fast Buffered Write

X – 1 - Latched Write

FIGURE 4-13. Interface Mode Bits

All combinations of Remote Reads or Writes with buffers or latches can be configured via the Interface Mode bits. A Buffered Read is accomplished by using a buffer for block D and setting [LR] = 0. Conversely, using a latch for block D and setting [LR] = 1 configures the RASM for Latched Reads. Using buffers for blocks A, B, and C and setting [LW] = 0 allows either a Slow or Fast Buffered Write. Setting [FBW] = 0 configures RASM for a Slow Buffered Write

and [FBW] = 1 designates a Fast Buffered Write. A Latched Write is accomplished by using latches for blocks A, B, and C and setting [LW] = 1.

4.1.4 Execution Control

The BCP can be started and stopped in two ways. If the BCP is not interfaced to another processor, it can be started by pulsing $\overline{RESET}$ low while both $\overline{REM\text{-}RD}$ and $\overline{REM\text{-}WR}$ are low. Execution then begins at location zero. If there is a Remote Processor interfaced to the BCP, a write to {RIC} which sets the start bit [STRT] high will begin execution at the current PC location. Writing a zero to [STRT] stops execution after the current instruction is completed. A Single-Step is accomplished by writing a one to the Single-Step bit [SS] in {RIC}. This will execute the instruction at the current PC, increment the PC, and then return to idle. [SS] returns low after the single-stepped instruction has completed. [SS] is a write only bit and will always appear low when {RIC} is read.

Two pins ($\overline{WAIT}$ and $\overline{LOCK}$), and one register bit, [LOR], can also affect the BCP CPU or RIAS execution. The $\overline{WAIT}$ pin can be used to add wait states to a remote access. When $\overline{WAIT}$ must be asserted low to add wait states is dependent on which remote access mode is being used. The information needed to calculate when $\overline{WAIT}$ must be asserted to add wait states, is contained within the individual descriptions of the modes in the next section (4.2 RIAS Functional Description).

4.0 Remote Interface and Arbitration System (RIAS) (Continued)

Programmed wait states delay when $\overline{WAIT}$ must be asserted since programmed wait states are inserted before $\overline{WAIT}$ is tested to see if any more wait states should be added. $\overline{LOCK}$ prevents local accesses of Data Memory. If $\overline{LOCK}$ is asserted a half T-state before T1 of a BCP instruction cycle, further local accesses will be prevented by waiting the Timing Control Unit. The Timing Control Unit (TCU) is the BCP CPU sub system responsible for timing each instruction. For a more detailed description of the operation of $\overline{LOCK}$, refer to the CPU Timing section. [LOR] allows the BCP to prevent remote accesses. Once [LOR], located in {ACR}, is set high, further remote accesses are waited by XACK remaining low.

Though the BCP CPU runs independently of RIAS there is some interaction between the two systems. [LOR] is one such interaction. In addition, two bits allow the BCP CPU to keep track of remote accesses. These bits are the Remote Write bit [RW] and the Remote Read bit [RR], and are located in {CCR[6–5]}. Each bit goes high when its respective remote access to DMEM reaches its Termination Phase. Once one of these bits has been set, it will remain high until a "1" is written to that bit to reset it low.

4.2 RIAS FUNCTIONAL DESCRIPTION

In this section, the operation of the Remote Arbitration State Machine (RASM), is described in detail. Discussed, among other things, are the sequence of events in a remote access, arbitration of the data buses, timing of external signals, when inputs are sampled, and when wait states are added. Each of the five Interface Modes is described in functional state machine form. Although each interface mode is broken out in a separate flow chart, they are all part of a single state machine (RASM). Thus the first state in each flow chart is actually the same state.

The functional state machine form is similar to a flow chart, except that transitions to a new state (states are denoted as rectangular boxes) can only occur on the rising edge of the internal CPU clock (CPU-CLK). CPU-CLK is high during the first half of its cycle. A state box can specify several actions, and each action is separated by a horizontal line. A signal name listed in a state box indicates that that pin will be asserted high when RASM has entered that state. Signals not listed are assumed low.

Note: This sometimes necessitates using the inversion of the external pin name.

This same rule applies to the A and AD buses. By default, these buses are active. The A bus will have the upper byte of the last used data address.

The AD bus will display {RIC}. When one of these buses appears in a state box, the condition specified will be in effect only during that state. Decision blocks are shown as diamonds and their meaning is the same as in a flow chart. The hexagon box is used to denote a conditional state—not synchronous with the clock. When the path following a decision block encounters a conditional state, the action specified inside the hexagon box is executed immediately.

Also provided is a memory arbitration example in the form of a timing diagram for each of the five modes. These examples show back to back local accesses punctuated by a remote access. Both the state of RASM and the Timing Control Unit are listed for every clock at the top of each timing diagram. The RASM states listed correspond to the flow charts. The Timing Control Unit states are described in Section 2.2.2, Timing portion of the data sheet.

4.2.1 Buffered Read

The unique feature of this mode is the extension of the read until $\overline{REM\text{-}RD}$ is deasserted high. The complete flow chart for the Buffered Read mode is shown in *Figure 4-14*. Until a Remote Read is initiated (RAE*REM-RD true), the state machine (RASM) loops in state RS_{A1}. If a Remote Read is initiated and [LOR] is set high, RASM will move to state RS_{A2}. Likewise, if a Remote Read is initiated while the buses have been granted locally (i.e., Local Bus Request = 1), RASM will move to state RS_{A2}. The state machine will loop in state RS_{A2} as long as [LOR] is set high or the buses are granted locally. If the BCP CPU needs to access Data Memory while in either RS_A state (and $\overline{LOCK}$ is high), it can still do so. A local access is requested by the Timing Control Unit asserting the Local Bus Request (LCL-BREQ) signal. A local bus grant will be given by RASM if the buses are not being used (as is the case in the RS_A states).

XACK is taken low as soon as RAE*REM-RD is true, regardless of an ongoing local access. If [LOR] is low, RASM will move into RS_B on the next clock after RAE*REM-RD is true and there is no local bus request. No further local bus requests will be granted until the remote access is complete and RASM returns to RS_A.

On the next CPU-CLK, RASM enters RS_C and $\overline{LCL}$ is taken high while XACK remains low. The wait state counters, i_{IW} and i_{DW}, are loaded in this state from [IW1–0] and [DW2–0], respectively, in {DCR}. The A bus (and AD if the access is to Data Memory) now goes into TRI-STATE and the Access Phase begins.

The state machine can move into one of several states, depending on the state of CMD and [MS1–0], on the next clock. XACK remains low and $\overline{LCL}$ remains high in all the possible next states. If CMD is high, the access is to {RIC} and the next state will be RS_{D1}. Since the default state of AD is {RIC}, it will not transition in this state.

The five other next states all have CMD low and depend on the Memory Select bits. If [MS1–0] is 10 or 11 the state machine will enter either RS_{D2} or RS_{D3} and the low or high bytes of the Program Counter, respectively, will be read.

[MS1–0] = 00 designates a Data Memory access and moves RASM into RS_{D4}. $\overline{READ}$ will be asserted in this state and A and AD continue to be in TRI-STATE. This allows the Remote Processor to drive the Data Memory address for the read. Since DMEM is subject to wait states, RS_{D4} is looped upon until all the wait states have been inserted.
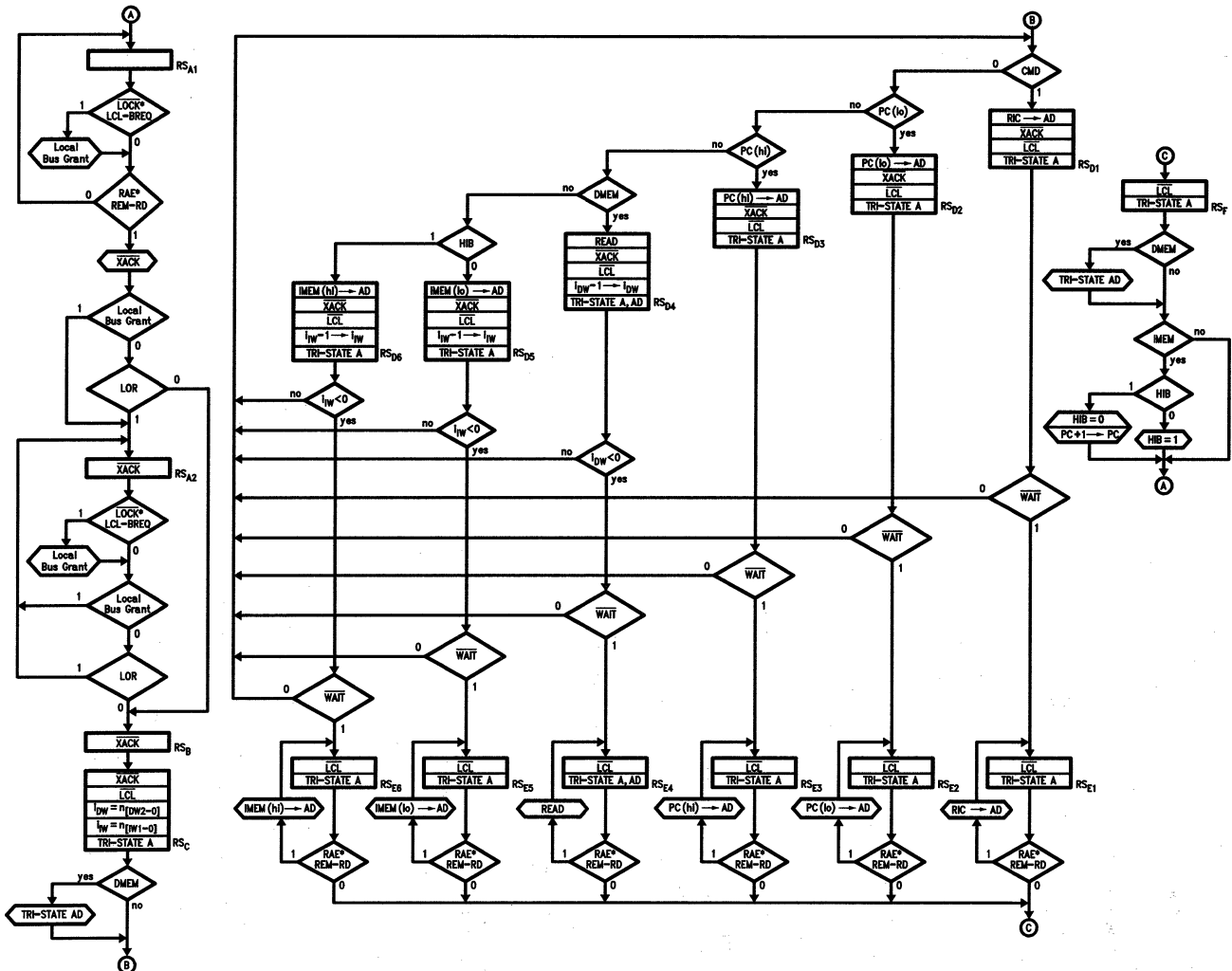
FIGURE 4-14. Flow Chart of Buffered Read Mode

TL/F/9336–97
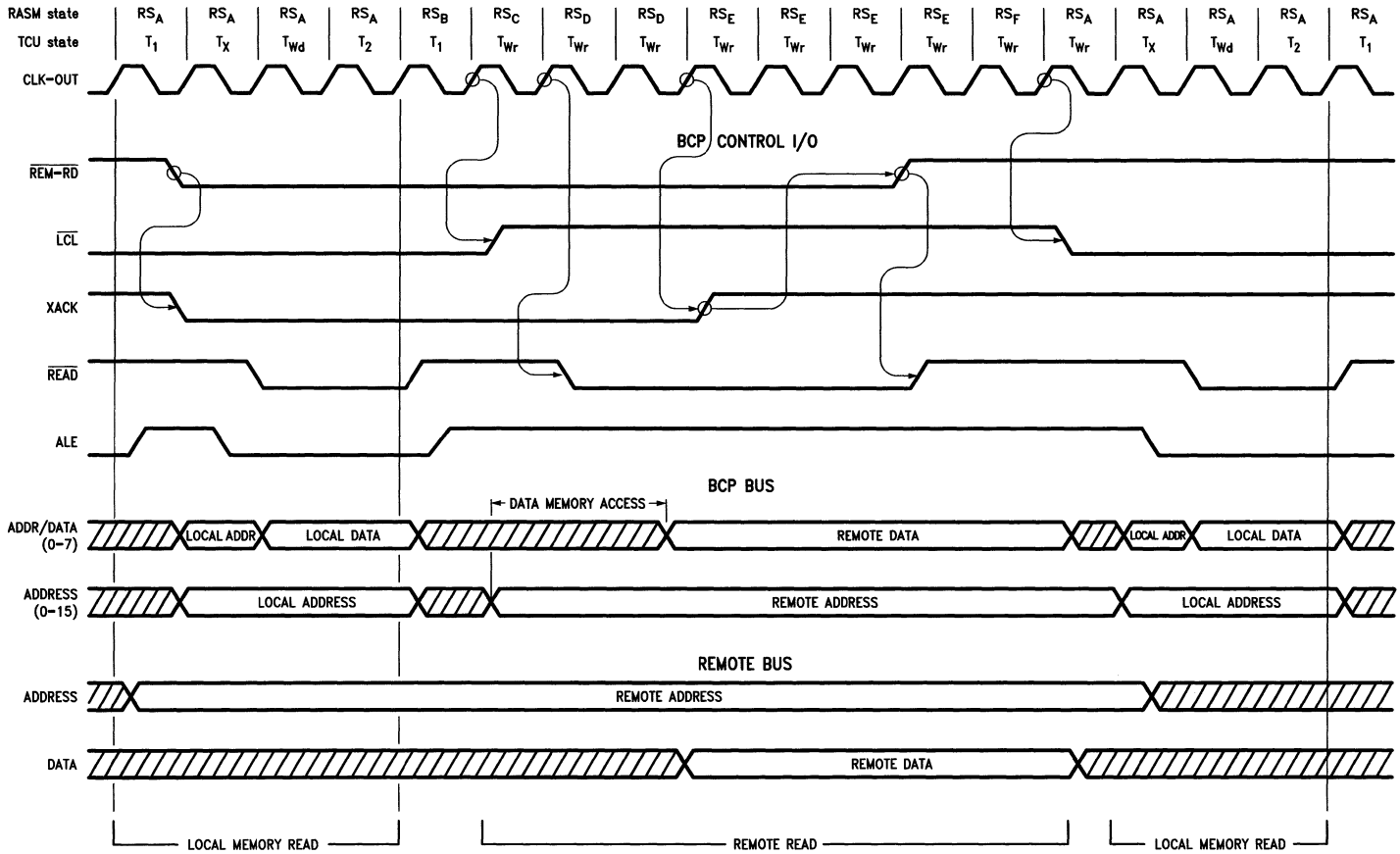
TL/F/9336–27

Register Configuration:

—One Wait-State Programmed for Data-Memory
—Zero Wait-States Programmed for Instruction-Memory
—{RIC} Contents: XXX0X100
—[LOR] = 0

Other BCP Control Signals:

$\overline{RAE}$ = 0
CMD = 0
$\overline{REM\text{-}WR}$ = 1
$\overline{LOCK}$ = 1

FIGURE 4-15. Buffered Read of Data Memory by Remote Processor

4.0 Remote Interface and Arbitration System (RIAS) (Continued)

The last possible Memory Selection is Instruction Memory, [MS1–0] = 01. The two possible next states for an IMEM access depend on if RASM is expecting the low byte or high byte. Instruction words are accessed low then high byte and RASM powers up expecting the low Instruction byte. The internal flag that keeps track of the next expected Instruction byte is called the High Instruction Byte flag (HIB). If HIB is low, the next state is RS$_{D5}$ and the low instruction byte is MUXed to the AD bus. If HIB is high, the high instruction byte is MUXed to AD and RS$_{D6}$ is entered. An IMEM access, like a DMEM access, is subject to wait states and these states will be looped on until all programmed instruction memory wait states have been inserted.

After all of the programmed wait states are inserted in the RS$_D$ states, more wait states may be added by asserting WAIT low a half T-state before the end of the last programmed wait state. If there are no programmed wait states, WAIT must be asserted low a half T-state before the end of RS$_D$ to add wait states. If WAIT remains low, the remote access is extended indefinitely. All the RS$_D$ states move to their corresponding RS$_E$ states on the CPU-CLK after the programmed wait state conditions are met and WAIT is high. The RS$_E$ states are looped upon until RAE* REM-RD is deasserted. LCL remains high in all RS$_E$ states and A remains in TRI-STATE. AD will also stay in TRI-STATE if the access was to DMEM. XACK is taken back high to indicate that data is now valid on the read. If XACK is connected to a Remote Processor wait pin, it is no longer waited and can now terminate its read cycle. This state begins the Termination Phase. The action specified in the conditional box is only executed while RAE*REM-RD is asserted—a clock edge is not necessary.

On the CPU-CLK after RAE*REM-RD is deasserted, RASM enters RS$_F$, where LCL is high and the TRI-STATE condition in RS$_E$ remains in effect. The next clock brings the state machine back to RS$_A$ state where it will loop until another

Remote Access is initiated. If the access was to IMEM, then the last action of the remote access before returning to RS$_A$ is to switch HIB and increment the PC if the high byte was read.

The example in *Figure 4-15* shows the BCP executing the first of two consecutive Data Memory reads when REM-RD goes low. In response, XACK goes low waiting the remote processor. At the end of the first instruction, although the BCP begins its second read by taking ALE high, the RASM now takes control of the bus and takes LCL high at the end of T$_1$. A one T-state delay is built into this transfer to ensure that READ has been deasserted before the data bus is switched. The Timing Control Unit is now waited, inserting remote access wait states, T$_{Wr}$, as RASM takes over.

The remote address is permitted one T-state to settle on the BCP address bus before READ goes low, XACK then returns high one T-state plus the programmed Data Memory wait state, T$_{Wd}$ later, having satisfied the memory access time. The Remote Processor will respond by deasserting REM-RD high to which the BCP in turn responds by deasserting READ high. Following READ being deasserted high, the BCP waits till the end of the next T-state before taking LCL low, again ensuring that the read cycle has concluded before the bus is switched. Control is then returned to the Timing Control Unit and the local memory read continues.

4.2.2 Latched Read

This mode differs from the Buffered Read mode in the way the access is terminated. A latched Read cycle ends after the data being read is valid and the termination doesn't wait for the trailing edge of REM-RD. Therefore the Arbitration and Access Phases of the Latched Read mode are the same as for the Buffered Read mode. The complete flow chart for the Latched Read mode is shown in *Figure 4-16*. Until a Remote Read is initiated (RAE*REM-RD true), the state machine (RASM) loops in state RS$_{A1}$. If a Remote

4.0 Remote Interface and Arbitration System (RIAS) (Continued)

Read is initiated and [LOR] is set high, RASM will move to state RS_{A2}. Likewise, if a Remote Read is initiated while the buses have been granted locally (i.e., Local Bus Grant = 1), RASM will move to state RS_{A2}. The state machine will loop in state RS_{A2}, as long as [LOR] is set high or the buses are granted locally. If the BCP CPU needs to access Data Memory while in either RS_A state (and $\overline{LOCK}$ is high), it can still do so. A local access is requested by the Timing Control Unit asserting the Local Bus Request (LCL-BREQ) signal. A local bus grant will be given by RASM if the buses are not being used (as is the case in RS_A).

XACK is taken low as soon as RAE*REM-RD is true, regardless of an ongoing local access. If [LOR] is low, RASM will move into RS_B on the next clock after RAE*REM-RD is asserted and there is no local bus request. No further local bus requests will be granted until RASM enters the Termination Phase. If the BCP CPU initiates a Data Memory access after RS_A, the Timing Control Unit will be waited and the BCP CPU will remain in state T_{Wr} until the remote access reaches the Termination Phase.

On the next clock, RASM enters RS_C and $\overline{LCL}$ is taken high while XACK remains low. The wait state counters, i_{IW} and i_{DW}, are loaded in this state from [IW1-0] and [DW2-0], respectively, in {DCR}. The A bus (and AD if the access is to Data Memory) now goes into TRI-STATE and the Access Phase begins.

The state machine can move into one of several states, depending on the state of CMD and [MS1-0], on the next clock. XACK remains low and $\overline{LCL}$ remains high in all the possible next states. If CMD is high, the access is to {RIC} and the next state will be RS_{D1}. Since the default state of AD is {RIC}, it will not transition in this state. The five other next states all have CMD low and depend on the Memory Select bits. If [MS1-0] is 10 or 11 the state machine will enter either RS_{D2} or RS_{D3} and the low or high bytes of the Program Counter, respectively, will be read.

[MS1-0] = 00 designates a Data Memory access and moves RASM into RS_{D4}. $\overline{READ}$ will be asserted low in this state and A and AD continue to be tri-stated. This allows the Remote Processor to drive the Data Memory address for the read. Since DMEM is subject to wait states, RS_{D4} is looped upon until all the wait states have been inserted.

The last possible Memory Selection is Instruction Memory, [MS1-0] = 01. The two possible next states for the IMEM access depend on if RASM is expecting the low byte or high byte. Instruction words are accessed low byte then high byte and RASM powers up expecting the low Instruction byte. The internal flag that keeps track of the next expected Instruction byte is called the High Instruction Byte flag (HIB). If HIB is low, the next state is RS_{D5} and the low instruction byte is MUXed to the AD bus. If HIB is high, the high instruction byte is MUXed to AD and RS_{D6} is entered. An IMEM access, like a DMEM access, is subject to wait states and these states will be looped on until all programmed instruction memory wait states have been inserted.

After all of the programmed wait states are inserted in the RS_D states, more wait states may be added by asserting $\overline{WAIT}$ low a half T-state before the end of the last programmed wait state. If there are no programmed wait states $\overline{WAIT}$ must be asserted low a half T-state before the end of RS_D to add wait states. If $\overline{WAIT}$ remains low, the remote access is extended indefinitely.

All the RS_D states move to their corresponding RS_E states on the CPU-CLK after the programmed wait state conditions are met and $\overline{WAIT}$ is high. $\overline{LCL}$ remains high in all RS_E states and A remains in TRI-STATE (and AD if the access is to Data Memory). XACK returns high in this state, indicating that data is valid so that it can be externally latched. The action specific to each RS_D state remains in effect during the first half of the RS_E cycle (i.e. $\overline{READ}$ is asserted in the first half of RS_{E4}). This half T-state of hold time is provided to guarantee data is latched when XACK goes high. This state begins the Termination Phase.
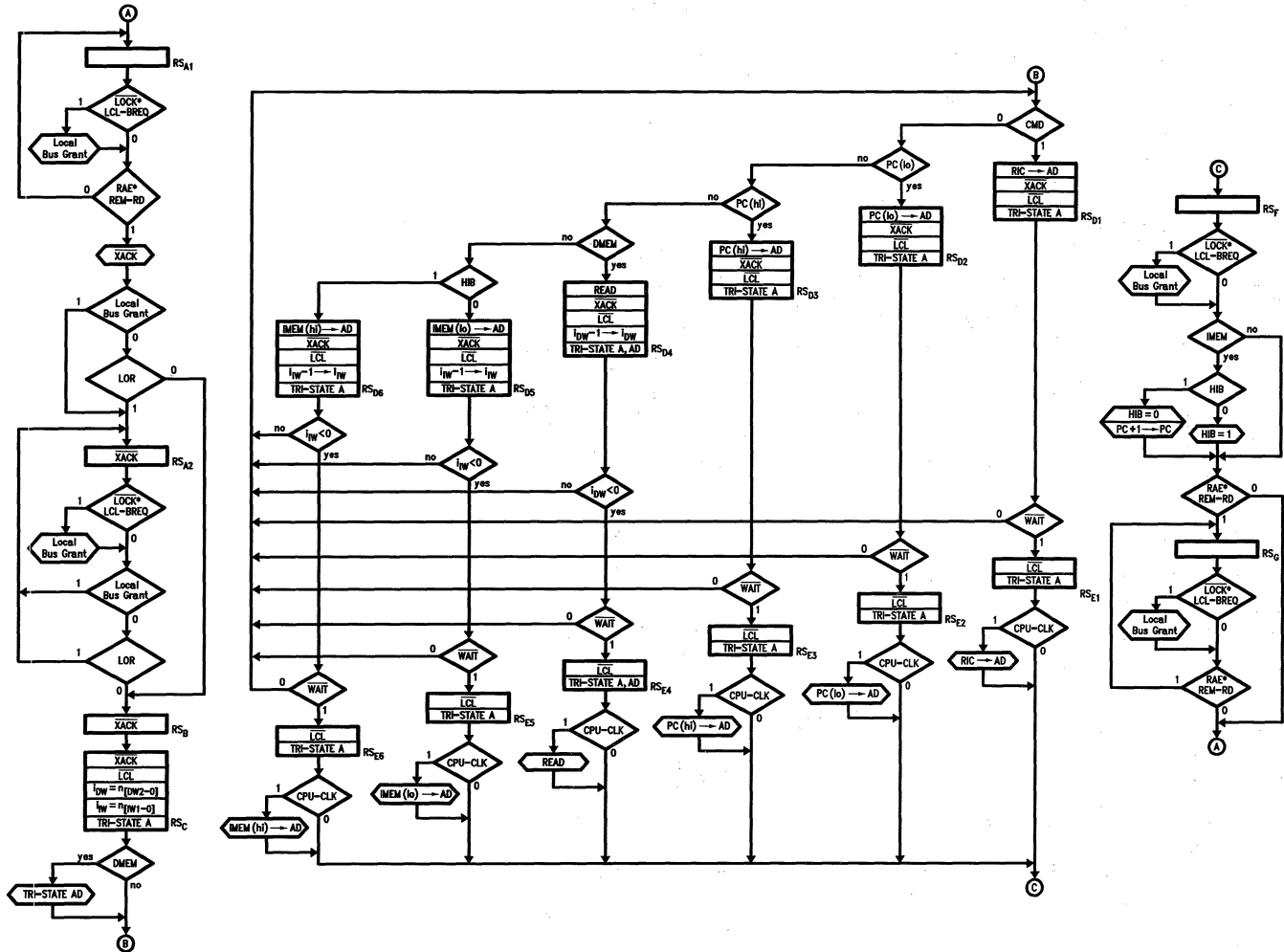
FIGURE 4-16. Flow Chart of Latched Read Mode
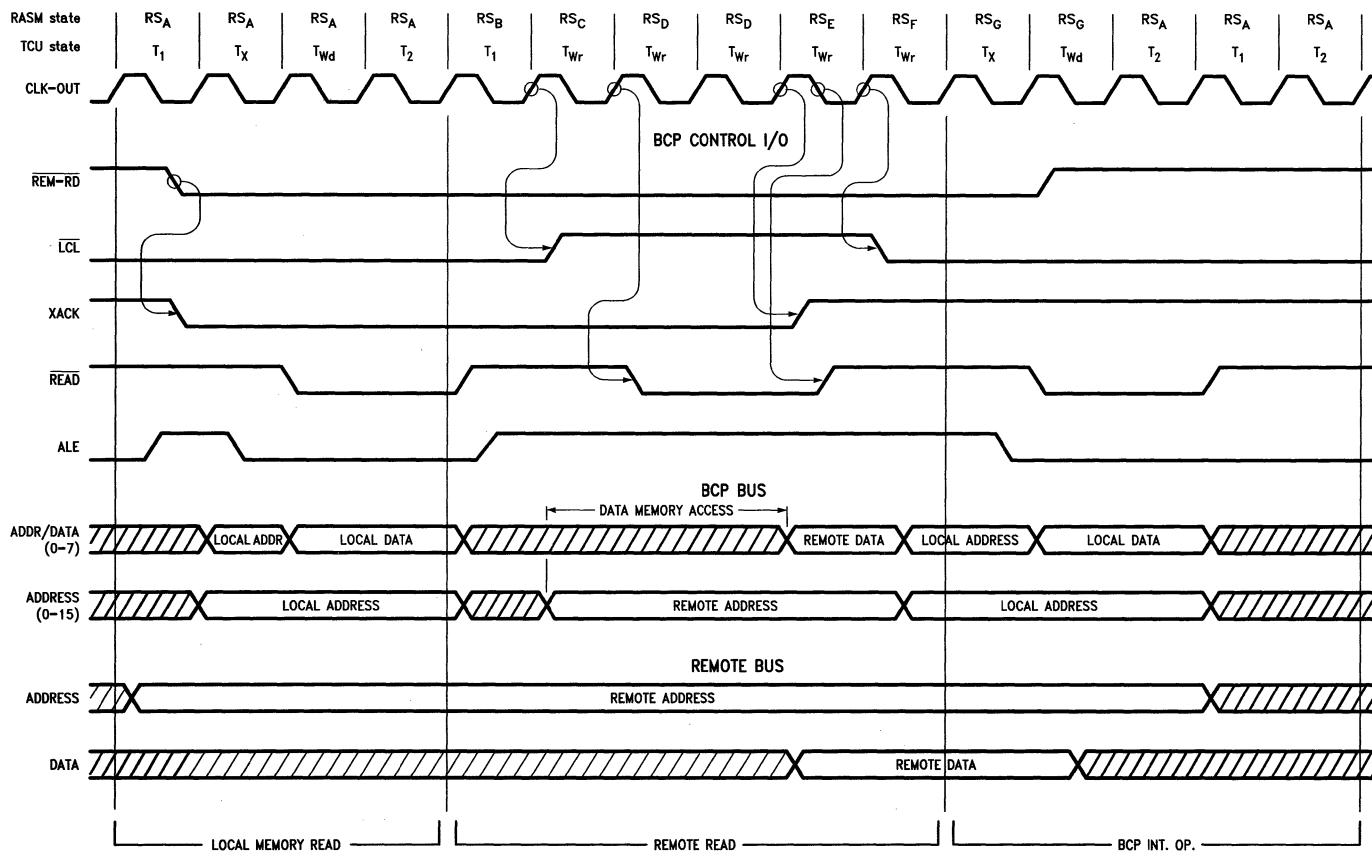
TL/F/9336-98

TL/F/9336–30

Register Configuration:

—One Wait-State Programmed for Data-Memory
—Zero Wait-States Programmed for Instruction-Memory
—{RIC} Contents: XXX1X100
—[LOR] = 0

Other BCP Control Signals:

$\overline{RAE}$	= 0
CMD	= 0
$\overline{REM\text{-}WR}$	= 1
$\overline{LOCK}$	= 1

FIGURE 4-17. Latched Read of Data Memory by Remote Processor

2

4.0 Remote Interface and Arbitration System (RIAS) (Continued)

On the next clock the state machine will enter RS_F and $\overline{LCL}$ will return low. Once the state machine enters RS_F, the Remote Processor is no longer using the buses and the BCP CPU will be granted the buses if LCL-BREQ is asserted. If a local bus request is made, a local bus grant will be given to the Timing Control Unit. If the preceding access was a read of IMEM, then HIB is switched and if the access was to the high byte of IMEM then the PC is incremented. If RAE* REM-RD is deasserted at this point, the next clock will bring RASM back to RS_A where it will loop until another Remote Access is initiated. RS_G is entered if RAE*REM-RD is still true. RASM will loop in RS_G until RAE*REM-RD is no longer active at which time the state machine will return to RS_A.

In *Figure 4-17*, the BCP is executing the first of two Data Memory reads when $\overline{REM\text{-}RD}$ goes low. In response, XACK goes low, waiting the Remote Processor. At the end of the first instruction, although the BCP begins its second write by taking ALE high, the RASM now takes control of the bus and deasserts $\overline{LCL}$ high at the end of T_1. A one T-state delay is built into this transfer to ensure that $\overline{READ}$ has been deasserted high before the data bus is switched. The Timing Control Unit is now waited, inserting remote access wait states, T_{Wr}, as RASM takes over.

The remote address is permitted one T-state to settle on the BCP address bus before $\overline{READ}$ goes low, XACK then returns high one T-state plus the programmed Data Memory wait state, T_{Wd} later, having satisfied the memory access time. $\overline{READ}$ returns high a half T-state later, ensuring sufficient hold time, followed by $\overline{LCL}$ being reasserted low after an additional half T-state, transferring bus control back to the BCP. The Remote Processor responds to XACK returning high by deasserting $\overline{REM\text{-}RD}$ high, although by this time the BCP is well into its own memory read.

4.2.3 Slow Buffered Write

The timing for this mode is the same as the Buffered Read mode. The complete flow chart for the Slow Buffered Write mode is shown in *Figure 4-18*. Until a Remote Write is initiated (RAE*REM-WR true), the state machine (RASM) loops in state RS_{A1}. If a Remote Write is initiated and [LOR] is set high, RASM will move to state RS_{A2}. Likewise, if a Remote Write is initiated while the buses have been granted locally (i.e., Local Bus Grant = 1), RASM will move to state RS_{A2}. The state machine will loop in state RS_{A2} as long as [LOR] is set high or the buses are granted locally. If the BCP CPU needs to access Data Memory while in either RS_A state (and $\overline{LOCK}$ is high), it can still do so. A local access is requested by the Timing Control Unit asserting the Local Bus Request (LCL-BREQ) signal. A local bus grant will be given by RASM if the buses are not being used (as is the case in the RS_A state).

XACK is taken low as soon as RAE*REM-WR is true, regardless of an ongoing local access. RASM will move into RS_B on the next clock after RAE*REM-WR is asserted and there is no local bus request and [LOR] = 0. No further local bus requests will be granted until the remote access is complete and RASM returns to RS_A. If the BCP CPU initiates a Data Memory access after RS_A, the Timing Control Unit will be waited and the BCP CPU will remain in state T_{Wr} until completion of the remote access.

On the next CPU-CLK, RASM enters RS_C and $\overline{LCL}$ is taken high while XACK remains low. The wait state counters, i_{IW}

and i_{DW}, are loaded in this state from [IW1–0] and [DW2–0], respectively, in {DCR}. The A and AD buses now go into TRI-STATE and the Access Phase begins. The state machine can move into one of several states, depending on the state of CMD and [MS1–0], on the next clock. XACK remains low and $\overline{LCL}$ remains high in all the possible next states. If CMD is high, the access is to {RIC} and the next state will be RS_{D1}. The path from AD to {RIC} opens in this state. Any remote access mode changes made by this write will not take effect until one T-state after the completion of the present write.

The five other next states all have CMD low and depend on the Memory Select bits. If [MS1–0] is 10 or 11, the state machine will enter either RS_{D2} or RS_{D3} and the low or high bytes of the Program Counter, respectively, will be written.

[MS1–0] equal to 00 designates a Data Memory access and moves RASM into RS_{D4}. $\overline{WRITE}$ will be asserted in this state and A and AD continue to be tri-stated. This allows the Remote Processor to drive the Data Memory address and data buses for the write. Since DMEM is subject to wait states, RS_{D4} is looped upon until all the programmed data memory wait states have been inserted.

The last possible Memory Selection is Instruction Memory, [MS1–0] = 01. The two possible next states for IMEM depend on whether RASM is expecting the low byte or high byte. Instruction words are accessed low byte, then high byte and RASM powers up expecting the low instruction byte. The internal flag that keeps track of the next expected Instruction byte is called the High Instruction Byte flag (HIB). If HIB is low, the next state is RS_{D5} and the low instruction byte is written into the holding register, ILAT. If HIB is high, the high instruction byte is moved to I15–8 and the value in ILAT is moved to I7–0. At the same time, $\overline{IWR}$ is asserted low, beginning the write to instruction memory. An IMEM access, like a DMEM access, is subject to wait states and these states will be looped on until all programmed Instruction Memory wait states have been inserted.

After all of the programmed wait states are inserted in the RS_D states, more wait states may be added by asserting $\overline{WAIT}$ low a half T-state before the end of the last programmed wait state. If there are no programmed wait states, $\overline{WAIT}$ must be asserted low a half T-state before the end of RS_D to add wait states. If $\overline{WAIT}$ remains low, the remote access is extended indefinitely. All the RS_D states move to their corresponding RS_E states on the CPU-CLK after the programmed wait state conditions are met and $\overline{WAIT}$ is high. The RS_E states are looped upon until RAE* REM-WR is deasserted. $\overline{LCL}$ remains high in all RS_E states, but XACK is taken back high to indicate that the remote access can be terminated. If XACK is connected to a Remote Processor wait pin, it can now terminate its write cycle. This state begins the Termination Phase. The action specified in the conditional box is only executed while RAE*REM-WR is asserted—a clock edge is not necessary.

On the CPU-CLK after RAE*REM-WR is deasserted, RASM enters RS_F, where $\overline{LCL}$ remains high and the BCP A and AD buses are still in TRI-STATE. The next clock brings the state machine back to RS_A state where it will loop until another Remote Access is initiated. If the access was to IMEM, then the last action of the remote access before returning to RS_A is to switch HIB and increment the PC if the high byte was written.
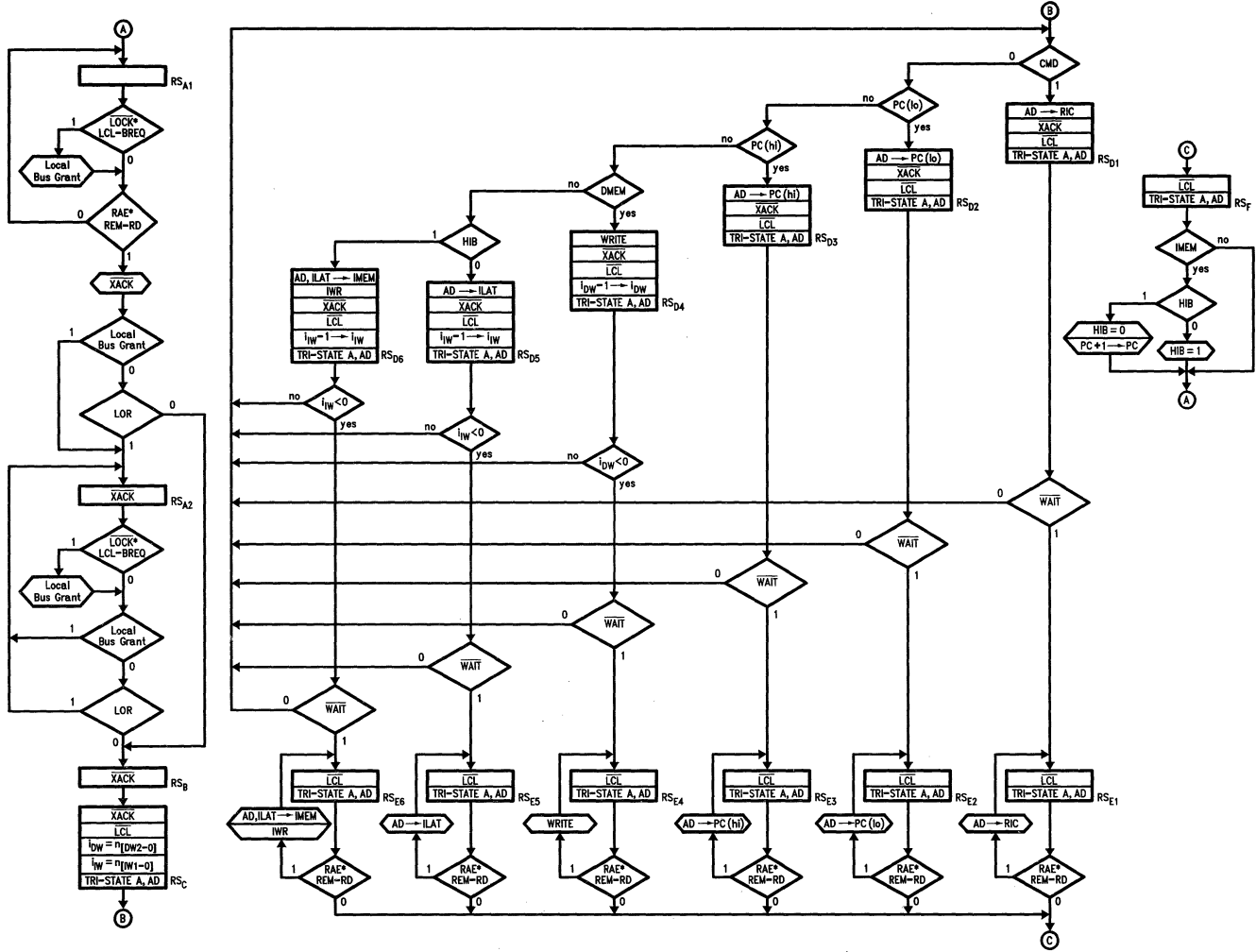
FIGURE 4-18. Flow Chart of Slow Buffered Write Mode

TL/F/9336–99

4.0 Remote Interface and Arbitration System (RIAS) (Continued)

In *Figure 4-19,* the BCP is executing the first of two consecutive Slow Buffered Writes to Data Memory when REM-WR goes low. In response, XACK goes low, waiting the Remote Processor. At the end of the first instruction, although the BCP begins its second write by taking ALE high, RASM now Takes control of the bus and deasserts LCL high at the end of T_1. A one T-state delay is built into this transfer to ensure that WRITE has been deasserted high before the data bus is switched. The Timing Control Unit is now waited, inserting remote access wait states, T_{Wr}, as RASM takes over.

The remote address is permitted one T-state to settle on the BCP address bus before WRITE goes low, XACK then returns high one T-state plus the programmed Data Memory wait state, T_{Wd} later, having satisfied the memory access time. The Remote Processor will respond by deasserting REM-WR high to which the BCP in turn responds by deasserting WRITE high. Following WRITE being deasserted high, the BCP waits till the end of the next T-state before asserting LCL low, again ensuring that the write cycle has concluded before the bus is switched. Control is then returned to the Timing Control Unit and the local memory write continues.

4.2.4 Fast Buffered Write

The timing for the Fast Buffered Write mode is very similar to the timing of the Latched Read. The major difference is the additional half clock that AD is active in the Latched Read mode that is not present in the Fast Buffered Write mode. The Fast Buffered Write cycle ends after the data is written and the termination doesn't wait for the trailing edge of REM-WR. Therefore the Arbitration and Access Phases of the Fast Buffered Write mode are the same as for the Latched Read mode.

The complete flow chart for the Fast Buffered Write mode is shown in *Figure 4-20.* Until a Remote Write is initiated (RAE*REM-WR true), the state machine (RASM) loops in state RS_{A1}. If a Remote Write is initiated and [LOR] is set high, RASM will move to state RS_{A2}. Likewise, if a Remote

Write is initiated while the buses have been granted locally (i.e., Local Bus Grant = 1), RASM will move to state RS_{A2}. The state machine will loop in state RS_{A2} as long as [LOR] is set high or the buses are granted locally. If the BCP CPU needs to access Data Memory while in either RS_A state (and LOCK is high), it can still do so. A local access is requested by the Timing Control Unit asserting the Local Bus Request (LCL-BREQ) signal. A local bus grant will be given by RASM if the buses are not being used (as is the case in the RS_A states).

XACK is taken low as soon as RAE*REM-WR is true, regardless of an ongoing local access. If [LOR] is low, RASM will move into RS_B on the next clock after RAE*REM-WR is asserted and there is no local bus request. No further local bus requests will be granted until the BCP enters the Termination Phase. If the BCP CPU initiates a Data Memory access after RS_A, the Timing Control Unit will be waited and the BCP CPU will remain in state T_{Wr} until the remote access reaches the Termination Phase.

On the next CPU-CLK, RASM enters RS_C and LCL is taken high while XACK remains low. The wait state counters, i_{IW} and i_{DW}, are loaded in this state from [IW1–0] and [DW2–0], respectively, in {DCR}. The A and AD buses now go into TRI-STATE and the Access Phase begins.

The state machine can move into one of several states depending on the state of CMD and [MS1–0] on the next clock. XACK and LCL in all the possible next states. If CMD is high, the access is to {RIC} and the next state will be RS_{D1}. The path from AD to {RIC} opens in this state. Any remote access mode changes made by this write will not take effect until one T-state after the completion of the present write.

The five other next states all have CMD low and depend on the Memory Select bits. If [MS1–0] is 10 or 11 the state machine will enter either RS_{D2} or RS_{D3} and the low or high bytes of the Program Counter, respectively, will be written. [MS1–0] = 00 designates a Data Memory access and moves RASM into RS_{D4}. WRITE will be asserted in this

| RASM state | RS$_A$ | RS$_A$ | RS$_A$ | RS$_A$ | RS$_B$ | RS$_C$ | RS$_D$ | RS$_D$ | RS$_E$ | RS$_E$ | RS$_E$ | RS$_E$ | RS$_F$ | RS$_A$ | RS$_A$ | RS$_A$ | RS$_A$ | RS$_A$ |
| TCU state | T$_1$ | T$_X$ | T$_{Wd}$ | T$_2$ | T$_1$ | T$_{Wr}$ | T$_{Wr}$ | T$_{Wr}$ | T$_{Wr}$ | T$_{Wr}$ | T$_{Wr}$ | T$_{Wr}$ | T$_{Wr}$ | T$_{Wr}$ | T$_X$ | T$_{Wd}$ | T$_2$ | T$_1$ |

CLK-OUT

REM-WR

BCP CONTROL I/O

LCL

XACK

WRITE

ALE

BCP BUS

ADDR/DATA (0–7) LOCAL ADDR LOCAL DATA REMOTE DATA LOCAL ADDRESS LOCAL DATA

ADDRESS (0–15) LOCAL ADDRESS REMOTE ADDRESS LOCAL ADDRESS

REMOTE BUS

ADDRESS REMOTE ADDRESS

DATA REMOTE DATA

LOCAL MEMORY WRITE — REMOTE WRITE — LOCAL MEMORY WRITE

TL/F/9336–28
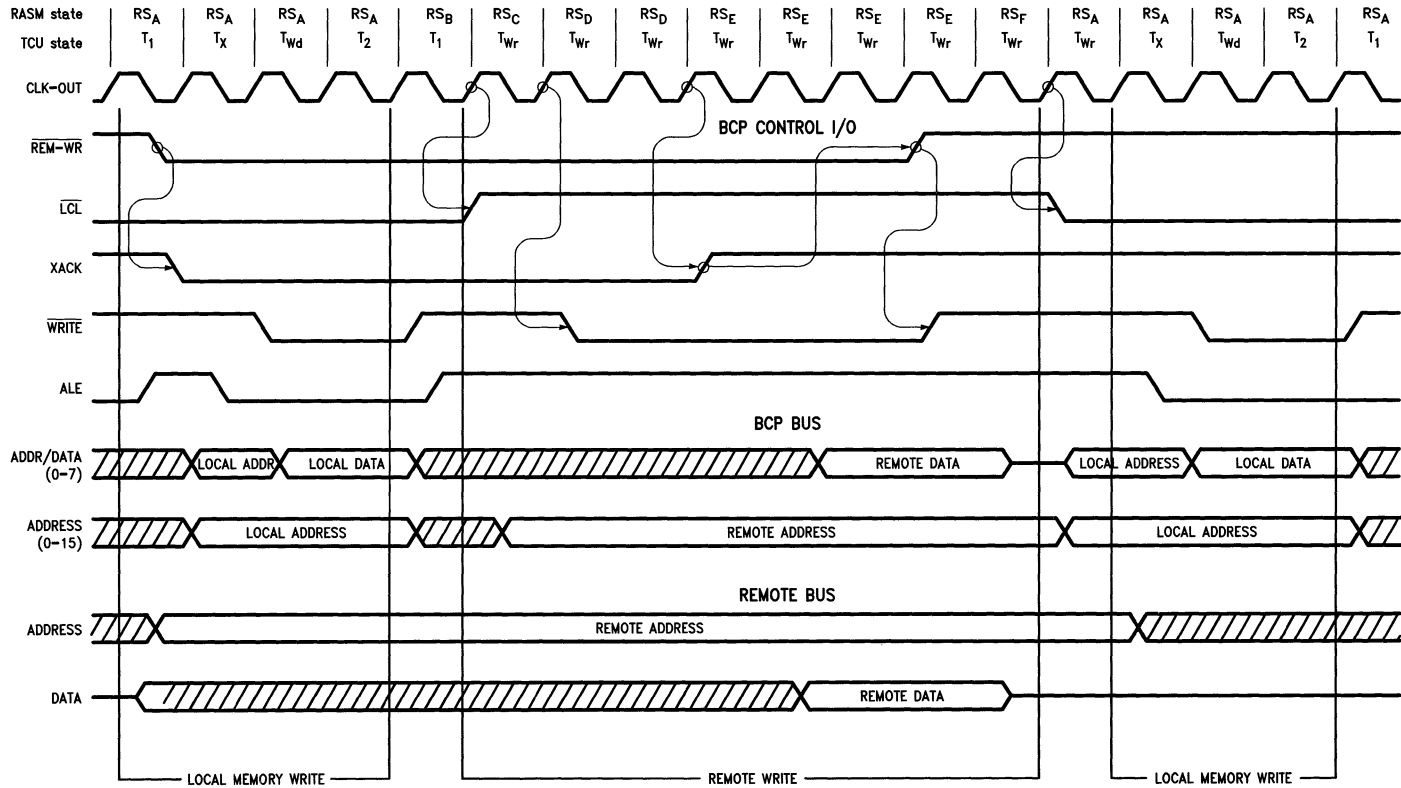
Register Configuration:
—One Wait-State Programmed for Data-Memory
—Zero Wait-States Programmed for Instruction-Memory
—{RIC} Contents: XX0X0100
—[LOR] = 0

Other BCP Control Signals:

$\overline{\text{RAE}}$	= 0
CMD	= 0
$\overline{\text{REM-RD}}$	= 1
$\overline{\text{LOCK}}$	= 1

FIGURE 4-19. Slow Buffered Write to Data Memory by Remote Processor

4.0 Remote Interface and Arbitration System (RIAS) (Continued)

state and A and AD continue to be tri-stated. This allows the Remote Processor to drive the Data Memory address and data buses for the write. Since DMEM is subject to wait states, RS_{D4} is looped upon until all the programmed Data Memory wait states have been inserted.

The last possible Memory Selection is Instruction Memory, [MS1–0] = 01. The two possible next states for IMEM depend on whether RASM is expecting the low byte or high byte. Instruction words are accessed low byte then high byte and RASM powers up expecting the low Instruction byte. The internal flag that keeps track of the next expected Instruction byte is called the High Instruction Byte flag (HIB). If HIB is low, the next state is RS_{D5} and the low instruction byte is written into the holding register, ILAT. If HIB is high, the high instruction byte is moved to I15–8 and ILAT is moved to I7–0. At the same time $\overline{IWR}$ is asserted low, beginning the write to instruction memory. An IMEM access, like a DMEM access, is subject to wait states and these states will be looped on until all programmed instruction memory wait states have been inserted.

After all of the programmed wait states are inserted into RS_D states, more wait states may be added by asserting $\overline{WAIT}$ low a half T-state before the end of the last programmed wait state. If there are no programmed wait states $\overline{WAIT}$ must be asserted low a half T-state before the end of RS_D to add wait states. If $\overline{WAIT}$ remains low, the remote access is extended indefinitely. All the RS_D states converge to state RS_E on the next CPU-CLK after the programmed wait state conditions are met and $\overline{WAIT}$ is high. $\overline{LCL}$ remains high in all RS_E states and A and AD remain in TRI-STATE as well. XACK returns high in this state, indicating that the data is written and the cycle can be terminated by the RP. This state begins the Termination Phase.

On the next clock the state machine will enter RS_F and $\overline{LCL}$ will return low. Once the state machine enters RS_F, the Remote Processor is no longer using the buses and the BCP CPU can make an access to Data Memory by asserting LCL-BREQ. If a local bus request is made, a local bus grant will be given to the Timing Control Unit. If the preceding access was a write of IMEM, then HIB is switched and if the access was to the high byte of IMEM then the PC is incremented. If RAE*REM-WR is deasserted at this point, the next clock will bring RASM back to RS_A where it will loop until another remote access is initiated. RS_G is entered if RAE*REM-WR is still true. RASM will loop in RS_G until RAE*REM-WR is no longer active at which time the state machine will return to RS_A.

In *Figure 4-21*, the BCP is executing the first of two Data Memory writes when $\overline{REM-WR}$ goes low. In response, XACK goes low, waiting the Remote Processor. At the end of the first instruction, although the BCP begins its second write by taking ALE high, RASM now takes control of the bus and deasserts $\overline{LCL}$ high at the end of T_1. A one T-state delay is built into this transfer to ensure that $\overline{WRITE}$ has been deasserted high before the data bus is switched. The Timing Control Unit is now waited, inserting remote access wait states, T_{Wr}, as RASM takes over.

The remote access is permitted one T-state to settle on the BCP address bus before $\overline{WRITE}$ goes low, XACK then returns high one T-state plus the programmed Data Memory wait state, T_{Wd} later, having satisfied the memory access time. $\overline{WRITE}$ returns high at the same time, and one T-state later $\overline{LCL}$ returns low, transferring bus control back to the BCP. The remote processor responds to XACK returning high by deasserting $\overline{REM-WR}$ high, although by this time the BCP is well into its own memory write.
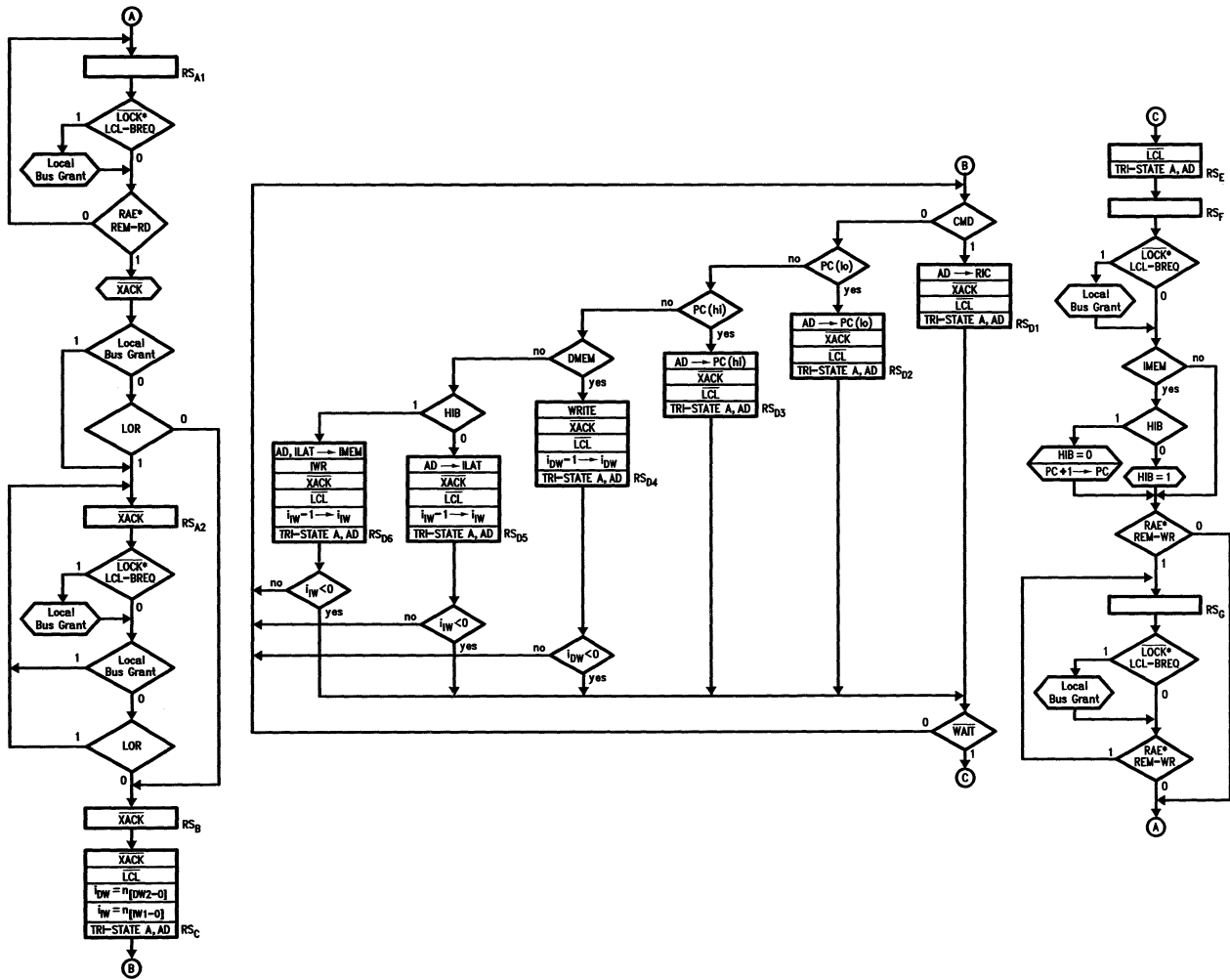
FIGURE 4-20. Flow Chart of Fast Buffered Write Mode

TL/F/9336–A0
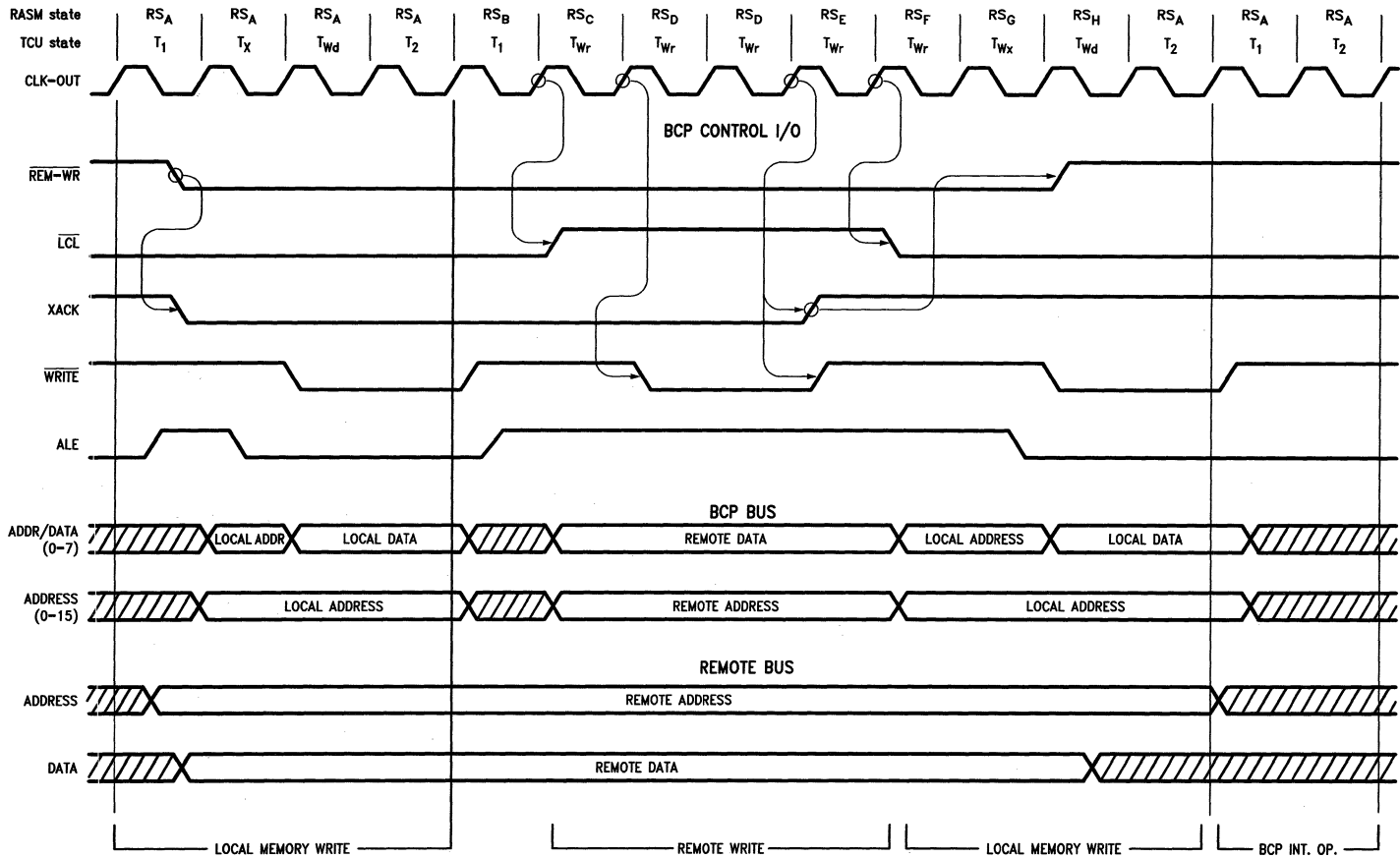
TL/F/9336-29

Register Configuration:

—One Wait-State Programmed for Data-Memory
—Zero Wait-States Programmed for Instruction-Memory
—{RIC} Contents: XX1X0100
—[LOR] = 0

Other BCP Control Signals:

$\overline{RAE}$	= 0
$\overline{CMD}$	= 0
$\overline{REM\text{-}RD}$	= 1
$\overline{LOCK}$	= 1

FIGURE 4-21. Fast Buffered Write to Data Memory by Remote Processor

4.0 Remote Interface and Arbitration System (RIAS) (Continued)

4.2.5 Latched Write

This mode executes a write without waiting the Remote Processor—XACK isn't normally taken low. The complete flow chart for the Latched Write mode is shown in *Figure 4-22*. Until a Remote Write is initiated (RAE*REM-WR true), the state machine (RASM) loops in state RS_A. If the BCP CPU needs to access Data Memory at this time (and $\overline{LOCK}$ is high), it can still do so. A local access is requested by the Timing Control Unit asserting the Local Bus Request (LCL-BREQ) signal. A local bus grant will be given by RASM if the buses are not being used (as is the case in RS_A).

RASM will move into RS_B on the next clock after RAE*REM-WR is asserted. XACK is not taken low and therefore the RP is not waited. The state machine will loop in RS_B until the RP terminates its write cycle—until RAE*REM-WR is no longer true. The external address and data latches are typically latched on the trailing edge of $\overline{REM\text{-}WR}$. A local bus request will still be serviced in this state.

Next, RASM enters RS_C and $\overline{WR\text{-}PEND}$ is asserted to prevent overwrite of the external latches. Since the RP has completed its write cycle, another write or read can happen at any time. Any Remote Read cycle (RAE*REM-RD) or Remote Write cycle (RAE*REM-WR) occurring after the state machine enters RS_C will take XACK low. A local access initiated before or during this state must be completed before RASM can move to RS_D. Once RS_D is entered, though, no further local bus requests will be granted until RASM enters the Termination Phase. If the BCP CPU initiates a Data Memory access after RS_C, the Timing Control Unit will be waited and the BCP CPU will remain in state T_{Wr} until the RASM enters RS_H.

On the next clock, the state machine enters RS_E and $\overline{LCL}$ is taken high. $\overline{WR\text{-}PEND}$ continues to be asserted low in this state and the data and instruction wait state counters, i_{DW} and i_{IW}, are loaded from [DW2–0] and [IW1–0], respectively, in {DCR}. Any remote accesses now occurring will take XACK low and wait the Remote Processor.

The state machine will move into one of several states on the next clock, depending on the state of CMD and [MS1–0]. $\overline{WR\text{-}PEND}$ remains low and $\overline{LCL}$ remains high in all the possible next states. If CMD is high, the access is to {RIC} and the next state will be RS_{F1}. The path from AD to {RIC} opens in this state. Any remote access mode changes made by this write will not take effect until one T-state after the completion of the present write.

The five other next states all have CMD low and depend on the Memory Select bits. If [MS1–0] is 10 or 11 the state machine will enter either RS_{F2} or RS_{F3} and the low or high bytes of the Program Counter, respectively, will be loaded.

[MS1–0] = 00 designates a Data Memory access and moves RASM into RS_{F4}. $\overline{WRITE}$ will be asserted low in this state and A and AD continue to be tri-stated. This allows the Remote Processor to drive the Data Memory address and data for the write. Since DMEM is subject to wait states, RS_{F4} is looped upon until all the programmed Data Memory wait states have been inserted.

The last possible Memory Selection is Instruction Memory, [MS1–0] = 01. The two possible next states for IMEM depend on if RASM is expecting the low byte or high byte. Instruction words are accessed low byte then high byte and

RASM powers up expecting the low Instruction byte. The internal flag that keeps track of the next expected Instruction byte is called the High Instruction Byte flag (HIB). If HIB is low, the next state is RS_{F5} and the low instruction byte is written into the holding register, ILAT. If HIB is high, the high instruction byte is moved to I15–8 and the value in ILAT is moved to I7–0. At the same time, $\overline{IWR}$ is asserted low and the write to Instruction Memory is begun. An IMEM access, like a DMEM access, is subject to wait states and these states will be looped on until all programmed instruction memory wait states have been inserted.

All the RS_F states converge to a single decision box that tests $\overline{WAIT}$. If $\overline{WAIT}$ is low then the state machine loops back to RS_F, otherwise RASM will move on to RS_G. $\overline{LCL}$ remains high and $\overline{WR\text{-}PEND}$ remains low in this state but the actions specific to the RS_F states have ended (i.e. $\overline{WRITE}$ will no longer be asserted low).

The next CPU-CLK moves RASM into RS_H, the last state in the state machine. $\overline{LCL}$ returns low but $\overline{WR\text{-}PEND}$ is still low. XACK will be taken low if a Remote Access is initiated. If the just completed access was to IMEM, HIB will be switched. Also, the PC will be incremented if the high byte was written. A local access will be granted if LCL-BREQ is asserted in this state.

If another Remote Write is pending, the state machine takes the path to RS_B where that write will be processed. A pending Remote Read will return to the RS_A in either the Buffered or Latched Read sections (not shown in *Figure 4-22*) of the state machine. And if no Remote Access is pending, the machine will loop in RS_A until the next access is initiated.

In *Figure 4-23*, the BCP is executing the first of two Data Memory writes when $\overline{REM\text{-}WR}$ goes low. The BCP takes no action until $\overline{REM\text{-}WR}$ goes back high, latching the data and making a remote access request. The BCP responds to this by taking $\overline{WR\text{-}PEND}$ low. At the end of the first instruction, although the BCP begins its second write by taking ALE high, RASM now takes control of the bus and deasserts $\overline{LCL}$ high at the end of T_1. A one T-state delay is built into this transfer to ensure that $\overline{WRITE}$ has been deasserted high before the data bus is switched. Timing Control Unit is now waited, inserting remote access wait states, T_{Wr}, as RASM takes over.

The remote address is permitted one T-state to settle on the BCP address bus before $\overline{WRITE}$ goes low. $\overline{WRITE}$ then returns high one T-state plus the programmed Data Memory wait state, T_{Wd}, later, having satisfied the memory access time, and one T-state later $\overline{LCL}$ is reasserted low, transferring bus control back to the BCP.

In this example, $\overline{REM\text{-}WR}$ goes low again during the remote write cycle which, since $\overline{WR\text{-}PEND}$ is still low, causes XACK to go low to wait the Remote Processor. Then $\overline{LCL}$ goes low, allowing the second data byte to be latched on the next trailing edge of $\overline{REM\text{-}WR}$. One T-state later. XACK and $\overline{WR\text{-}PEND}$ go back high at the same time.

The BCP is now shown executing a local memory write, with remote data still pending in the latch. At the end of this instruction, the BCP begins executing a series of internal operations which do not require the bus. RASM therefore takes over and, without waiting the Timing Control Unit, executes the Remote Write.
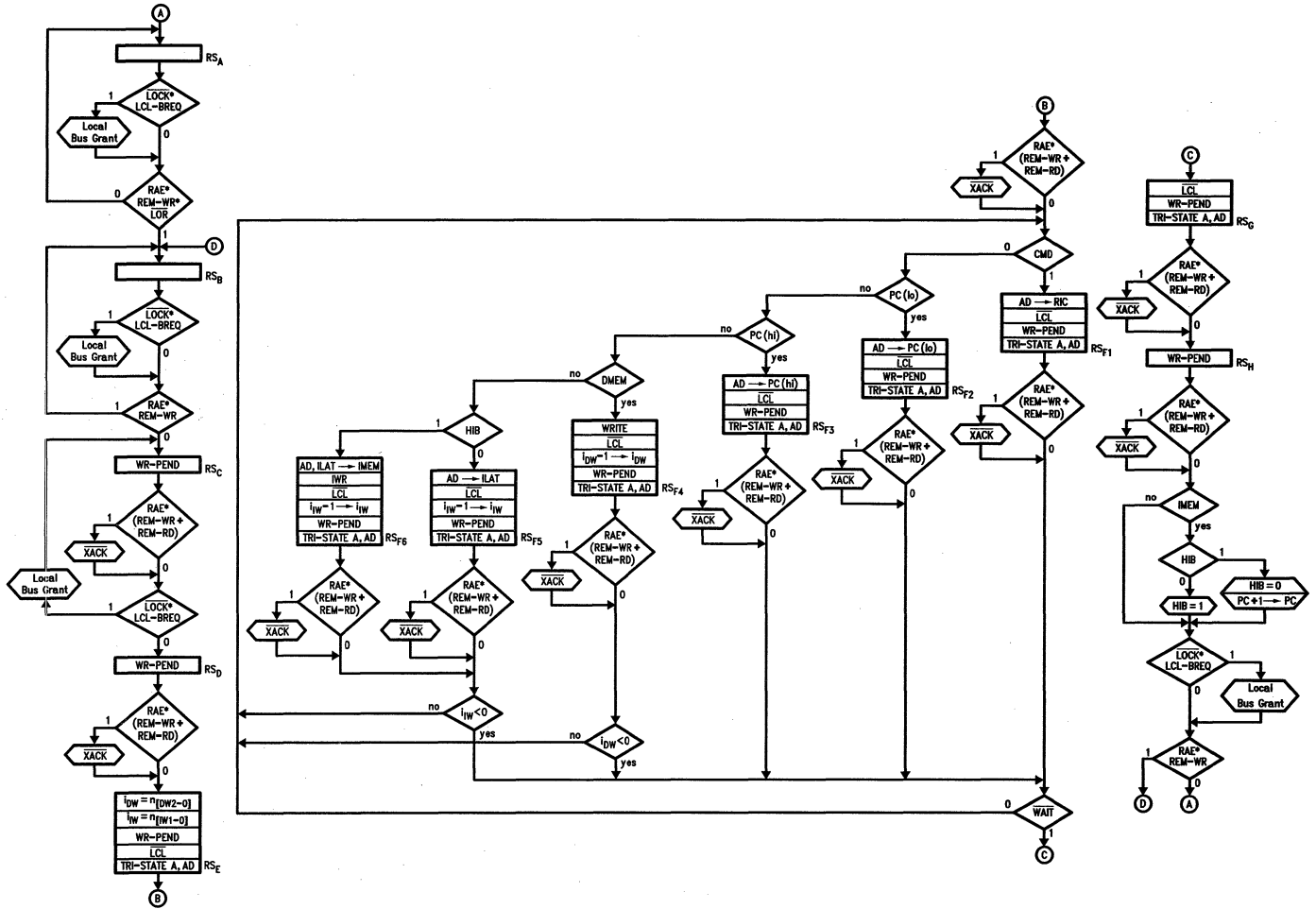
FIGURE 4-22. Flow Chart of Latched Write Mode
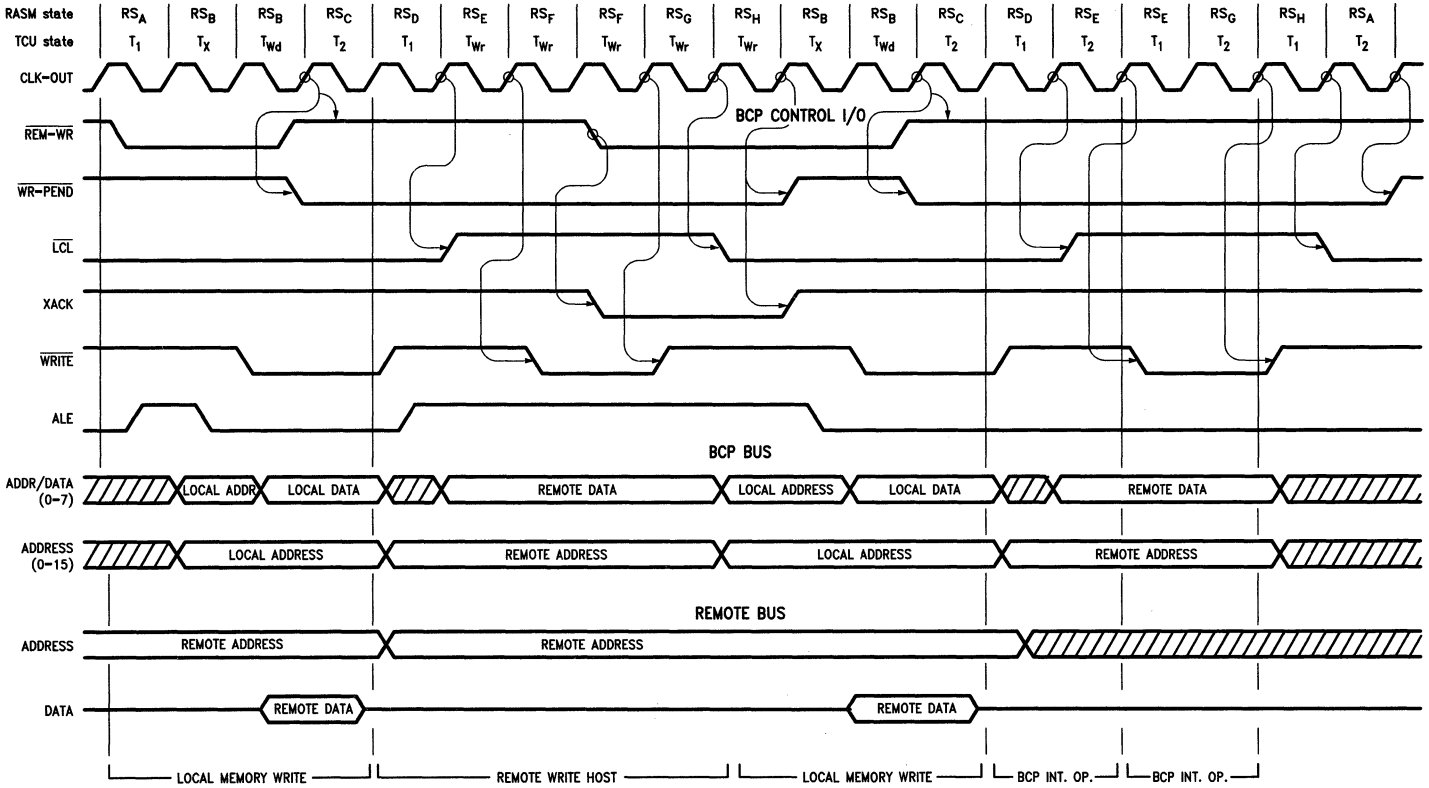
TL/F/9336–A1

TL/F/9336-31

Register Configuration:
—One Wait-State Programmed for Data-Memory
—Zero Wait-States Programmed for Instruction-Memory
—{RIC} Contents: XXXX1100
—[LOR] = 0

Other BCP Control Signals:

$\overline{RAE}$	= 0
CMD	= 0
$\overline{REM\text{-}RD}$	= 1
$\overline{LOCK}$	= 1

FIGURE 4-23. Latched Write to Data Memory by Remote Processor

4.0 Remote Interface and Arbitration System (RIAS) (Continued)

4.2.6 Remote Rest Time

For the BCP to operate properly, remote accesses to the BCP must be separated by a minimal amount of time. This minimal amount of time has been termed "rest time".

There are two causes for remote rest time. The first cause is implied in the functional state machine forms for remote accesses and can be explained as follows: At the beginning of every T-state the validity of a remote access is sampled for that T-state. To guarantee that the BCP recognizes the end of a remote cycle, the time between remote accesses must be a minimum of one T-state plus set up and hold times.

In the case of Latched Read and Fast Buffered Write, the validity of a remote access is not sampled on the first rising edge of the CPU-CLK following XACK rising. However, on all subsequent rising edges of the CPU-CLK the validity of the remote access is sampled. As a result, if the remote processor can terminate its remote access quickly after XACK rises (within a T-state), up to a T-state may be added to the above equation for Latched Read and Fast Buffered Write modes (i.e., a second remote access should not begin for two T-states plus set up and hold times after XACK rises in Latched Read and Fast Buffered Write modes). On the other hand, if the remote processor does not terminate its remote access within a T-state of XACK rising, the above equation (one T-state plus set up and hold times between remote accesses) remains valid for Latched Read and Fast Buffered Write modes.

If these specifications are not adhered to, the BCP may sample the very end of one valid remote access and one T-state later sample the very beginning of a second remote access. Thus, the BCP will treat the second access as a continuation of the first remote access and will not perform the second read/write. The second access will be ignored. (Reference *Figure 4-24* for the timing diagrams which demonstrate how two remote accesses can be mistaken as one.)

The second source of remote rest time is due to the manner in which the BCP samples the CMD signal. CMD is sampled once at the beginning of each remote access. Due to the manner in which CMD is sampled, CMD will not be sampled again if a second remote access begins within 1.5 T-states plus a hold time, after the BCP recognizes the end of the first remote access. If this happens, the BCP will use the value of CMD from the previous remote access during the second remote access. If the value of CMD is the same for both accesses, the second access will proceed as intended. However, if the value of CMD is different for the two remote accesses, the second remote access will read/write the wrong location.

The reader should note that the timing of the second source of rest time begins at the same time that the BCP first samples the end of the previous remote access. Thus when the first source of rest time ends, the second source of rest time begins. (Reference *Figure 4-25* for timing diagrams for rest time in all modes except Latched Write mode).

Latched Write Mode

Latched Write mode is a special case of rest time and needs to be discussed separately from the other modes. The first cause of rest time affects every mode including Latched Write. In regards to the second source of rest time, Latched Write mode was designed to allow a second remote access to start while a write is still pending (i.e., $\overline{\text{WR-PEND}}$ = 0). Thus, when $\overline{\text{WR-PEND}}$ rises (signaling the end of the previous write) the value of CMD is sampled for the second remote access. This allows Latched Write to avoid the second cause of rest time discussed above.

However, if a remote access begins within one half a T-state after $\overline{\text{WR-PEND}}$ rises, CMD will not be sampled again. For this case, if the value of CMD changes just after $\overline{\text{WR-PEND}}$ rose and at the same time the remote access begins, the BCP will read/write the wrong location. (Reference *Figure 4-26* for timing diagrams of rest time for latched write mode.)

4.0 Remote Interface and Arbitration System (RIAS) (Continued)
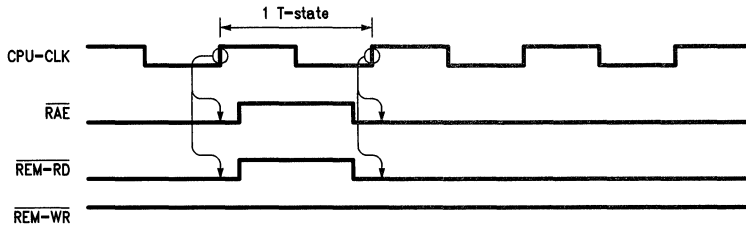
TL/F/9336–G5

(a) This timing diagram shows two remote accesses within one T-state. The first set of arrows shows the BCP sampling a valid remote read. The next time the BCP samples the validity of the remote access is shown by the second set of arrows (1 T-state later). In this case, it will sample the second remote access and mistake it as a continuation of the first remote access.
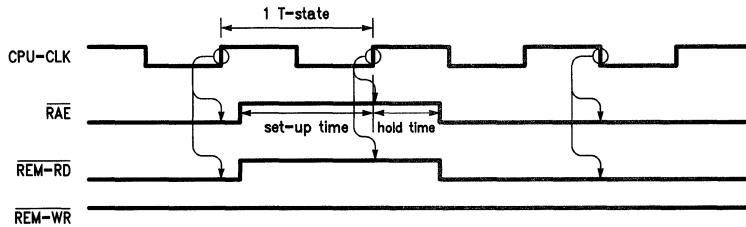
TL/F/9336–G6

(b) This timing diagram shows the timing necessary for the BCP to recognize both accesses as separate accesses. The first set of arrows shows the BCP sampling a valid remote read. One T-state later at the second set of arrows the BCP will sample the end of the first remote access. Another T-state later at the third set of arrows the BCP will sample the beginning of the second remote access.

FIGURE 4-24. Mistaking Two Remote Accesses as Only One
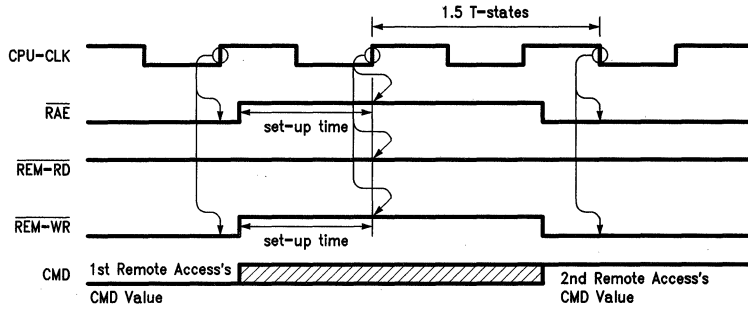
TL/F/9336–G7

(a) This timing diagram shows the second remote access violating rest time. The first set of arrows shows the BCP sampling a valid remote write. The second set of arrows (1 T-state later), shows the BCP sampling the end of the first remote access. If a second remote access starts before the position of the third set of arrows (another 1.5 T-states later), the value of CMD will not be sampled. The value of CMD has changed from the first remote access, so the BCP will write to the wrong location during the second access.
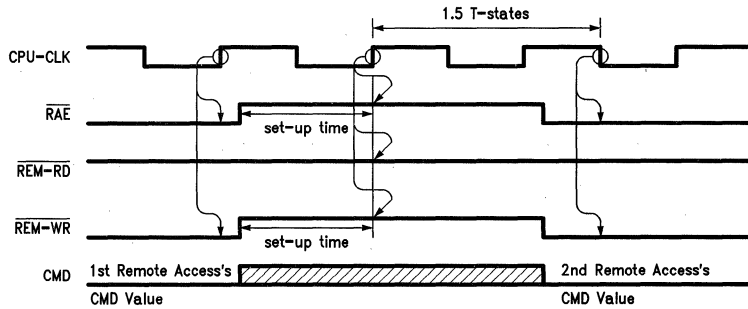
TL/F/9336–G8

(b) This timing diagram shows the second remote access violating rest time. The first set of arrows shows the BCP sampling a valid remote write. The second set of arrows (1 T-state later), shows the BCP sampling the end of the first remote access. If a second remote access starts before the position of the third set of arrows (another 1.5 T-states later), the value of CMD will not be sampled. The value of CMD does not change from the first remote access, so the BCP will write to the intended location during the second remote access.
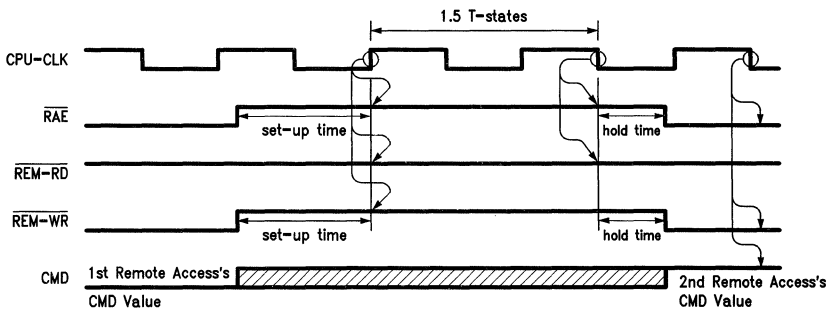
TL/F/9336–G9

(c) This timing diagram shows the timing needed to avoid violating rest time for all modes except latched write. The first set of arrows shows the BCP sampling the end of the first remote access. The second set of arrows (1.5 T-states later), shows the BCP recognizing no remote access has started and the value of CMD will be sampled for the next remote access. The third set of arrows shows the BCP sampling the correct value of CMD for the second remote access.

FIGURE 4-25. Remote Rest Time for All Modes except Latched Write

4.0 Remote Interface and Arbitration System (RIAS) (Continued)
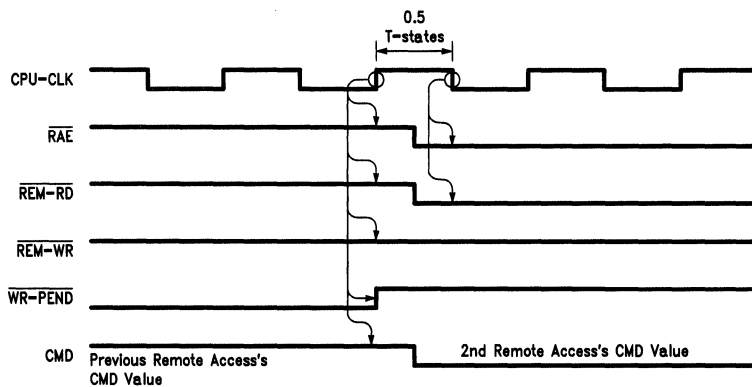
TL/F/9336–H1

(a) This timing diagram shows a remote access violating remote rest time. The first set of arrows shows the BCP sampling the value of CMD when WR-PEND rises. If a remote access begins after WR-PEND rises and before the position of the second set of arrows (0.5 T-states later), the value of CMD will not be sampled again. The value of CMD has changed since WR-PEND rose, so the BCP will read the wrong location.
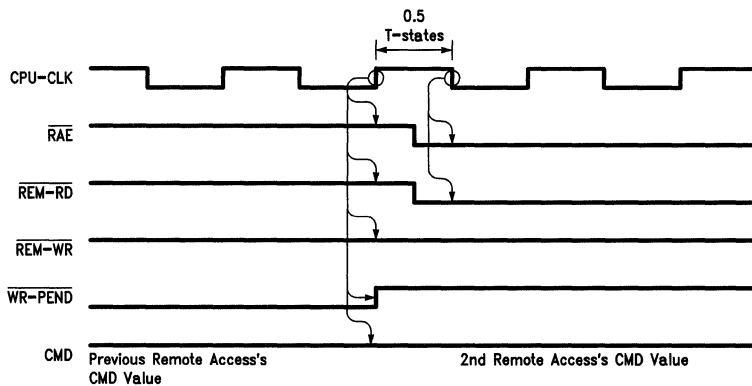
TL/F/9336–H2

(b) This timing diagram shows a remote access violating remote rest time. The first set of arrows shows the BCP sampling the value of CMD when WR-PEND rises. If a remote access begins after WR-PEND rises and before the position of the second set of arrows (0.5 T-states later), the value of CMD will not be sampled again. The value of CMD has not changed since WR-PEND rose, so the BCP will read the intended location.

FIGURE 4-26. Rest Time for Latched Write Mode

4.0 Remote Interface and Arbitration System (RIAS) (Continued)
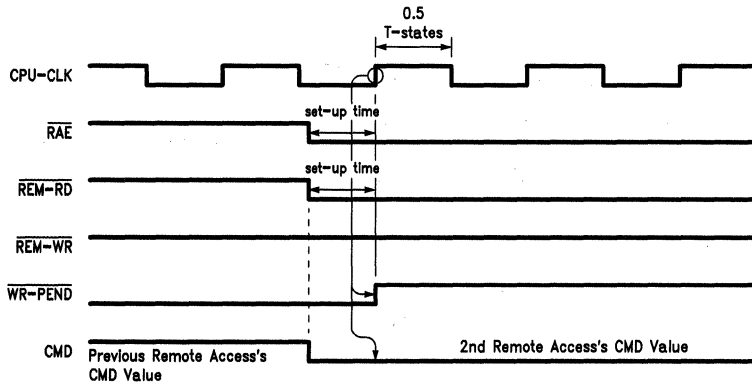
TL/F/9336–H3

(c) This timing diagram shows a remote access setting up in time for $\overline{WR\text{-}PEND}$ rising to latch in the proper value of CMD. The only set of arrows shows the BCP sampling the second remote access's CMD value when $\overline{WR\text{-}PEND}$ rises. The value of CMD will not be sampled again. The BCP will carry out the second remote access as it was intended.
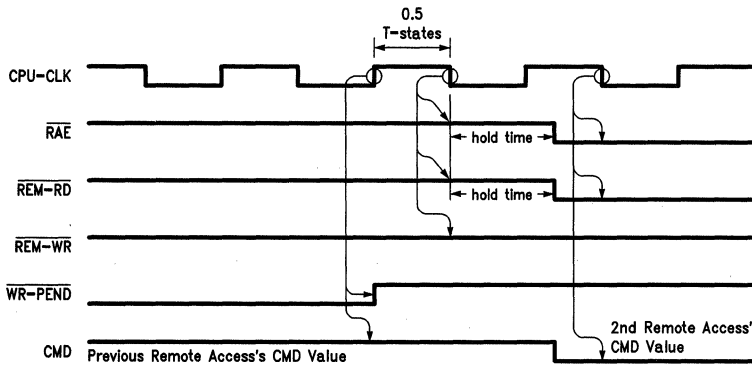
TL/F/9336–H4

(d) This timing diagram shows a remote access starting after a half T-state plus a hold time since $\overline{WR\text{-}PEND}$ rose. The first set of arrows shows the BCP sampling the value of CMD when $\overline{WR\text{-}PEND}$ rises. The second set of arrows shows the BCP recognizing that no remote access has started and the value of CMD will be sampled for the next remote access. The third set of arrows shows the BCP sampling the correct value of CMD for the second remote access. The BCP will carry out the second remote access as it was intended.

FIGURE 4-26. Rest Time for Latched Write Mode (Continued)

5.0 Device Specifications

Plastic Chip Carrier
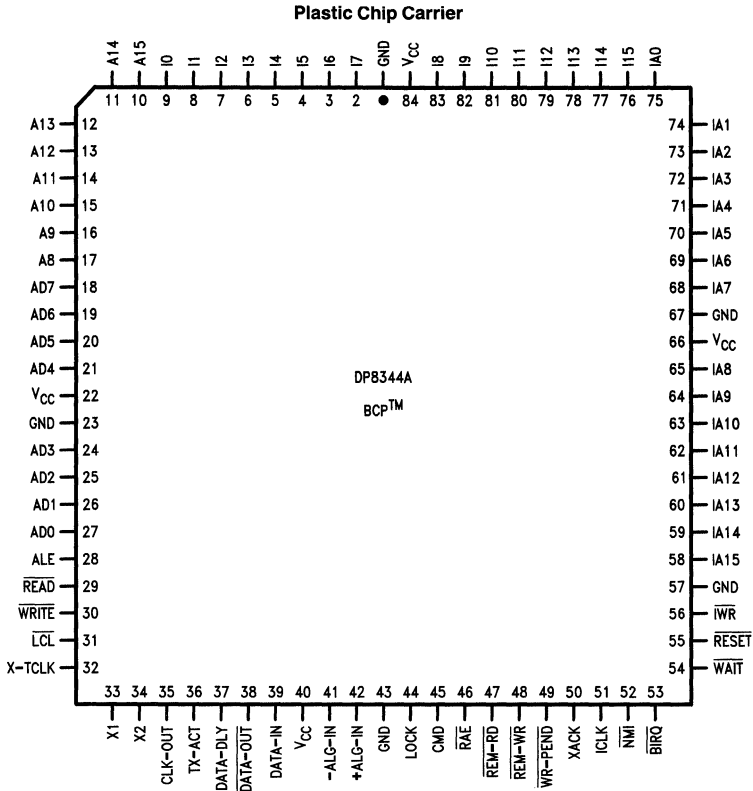
FIGURE 5-1. Top View

Order Number DP8344A
See NS Package Number V84A

TL/F/9336-2

5.1 PIN DESCRIPTIONS

Signal	In/Out	Pin	Reset State	Description
5.1.1 TIMING/CONTROL SIGNALS				
X1	In	33	X	Input and output of the on-chip crystal oscillator amplifier. Connect a crystal
X2	Out	34	$\overline{X1}$	across these pins, or apply an external clock to X1, with X2 left open.
CLK-OUT	Out	35	X1	Buffered **CL**oc**K** oscillator **OUT**put, at the crystal frequency.
X-TCLK	In	32	X	**EX**ternal **T**ransceiver **CL**oc**K** input.
$\overline{WAIT}$	In	54	X	CPU **WAIT**. When active, waits processor and remote interface controller.
$\overline{RESET}$	In	55	0	Master **RESET**. Parallel reset to all sections of the chip.
5.1.2 INSTRUCTION MEMORY INTERFACE **Instruction Address Bus:**				
IA15 (MSB)	Out	58	0	16-bit **I**nstruction memory **A**ddress bus.
IA14	Out	59	0	
IA13	Out	60	0	
IA12	Out	61	0	
IA11	Out	62	0	
IA10	Out	63	0	

5.0 Device Specifications (Continued)

Signal	In/Out	Pin	Reset State	Description
5.1.2 INSTRUCTION MEMORY INTERFACE (Continued)				
Instruction Address Bus: (Continued)				
IA9	Out	64	0	16-bit Instruction memory Address bus.
IA8	Out	65	0	
IA7	Out	68	0	
IA6	Out	69	0	
IA5	Out	70	0	
IA4	Out	71	0	
IA3	Out	72	0	
IA2	Out	73	0	
IA1	Out	74	0	
IA0 (LSB)	Out	75	0	
Instruction Bus:				
I15 (MSB)	In/Out	76	In	16-bit Instruction memory data bus.
I14	In/Out	77	In	
I13	In/Out	78	In	
I12	In/Out	79	In	
I11	In/Out	80	In	
I10	In/Out	81	In	
I9	In/Out	82	In	
I8	In/Out	83	In	
I7	In/Out	2	In	
I6	In/Out	3	In	
I5	In/Out	4	In	
I4	In/Out	5	In	
I3	In/Out	6	In	
I2	In/Out	7	In	
I1	In/Out	8	In	
I0 (LSB)	In/Out	9	In	
Timing Control:				
$\overline{\text{IWR}}$	Out	56	1	Instruction **WR**ite. Instruction memory write strobe.
ICLK	Out	51	0	Instruction **CL**o**cK**. Delimits instruction fetch cycles. Rises during the first half of T1, signifying the start of an instruction cycle, and falls when the next instruction address is valid.
5.1.3 DATA MEMORY INTERFACE				
Address Bus:				
A15 (MSB)	Out	10	X	High byte of 16-bit memory Address.
A14	Out	11	X	
A13	Out	12	X	
A12	Out	13	X	
A11	Out	14	X	
A10	Out	15	X	
A9	Out	16	X	
A8	Out	17	X	
Multiplexed Address/Data Bus:				
AD7	In/Out	18	1	Low byte of 16-bit data memory Address, multiplexed with 8-bit Data bus.
AD6	In/Out	19	0	
AD5	In/Out	20	0	
AD4	In/Out	21	0	
AD3	In/Out	24	0	
AD2	In/Out	25	0	
AD1	In/Out	26	0	
AD0 (LSB)	In/Out	27	1	

5.0 Device Specifications (Continued)

Signal	In/Out	Pin	Reset State	Description
5.1.3 DATA MEMORY INTERFACE (Continued) **Timing/Control:**				
ALE	Out	28	0	**A**ddress **L**atch **E**nable. Demultiplexes AD bus. Address should be latched on the falling edge.
READ	Out	29	1	Data memory **READ** strobe. Data is latched on the rising edge.
WRITE	Out	30	1	Data memory **WRITE** strobe. Data is presented on the rising edge.
5.1.4 TRANSCEIVER INTERFACE				
DATA-IN	In	39	X	Logic level serial **DATA IN**put.
+ALG-IN	In	42	X	Non-inverting **A**na**L**o**G IN**put for biphase serial data.
−ALG-IN	In	41	X	Inverting **A**na**L**o**G IN**put for biphase serial data.
DATA-OUT	Out	38	1	Biphase serial **DATA OUT**put (inverted).
DATA-DLY	Out	37	1	Biphase serial **DATA** output **D**e**L**a**Y**ed by one-quarter bit time.
TX-ACT	Out	36	0	**T**ransmitter **ACT**ive. Normally low, goes high to indicate serial data is being transmitted. Used to enable external line drive circuitry.
5.1.5 REMOTE INTERFACE				
RAE	In	46	X	**R**emote **A**ccess **E**nable. A "chip-select" input to allow host access of BCP functions and memory.
CMD	In	45	X	**C**o**M**man**D** input. When high, remote accesses are directed to the Remote Interface Configuration register {RIC}. When low, remote accesses are directed to data-memory, instruction-memory or program counter as determined by {RIC}.
REM-RD	In	47	X	**REM**ote **R**ea**D**. When active along with RAE, a remote read cycle is requested; serviced by the BCP when the data bus becomes available.
REM-WR	In	48	X	**REM**ote **WR**ite. When active along with RAE, a remote write cycle is requested; serviced by the BCP when the data bus becomes available.
XACK	Out	50	1	Transfer **ACK**nowledge. Normally high, goes low on REM-RD or REM-WR going low (if RAE low), returning high when the transfer is complete. Normally used as a "wait" signal to a remote processor.
WR-PEND	Out	49	1	**WR**ite **PEND**ing. In a system configuration where remote write cycles are latched, indicates when the latches contain valid data which is yet to be serviced by the BCP.
LOCK	In	44	X	The remote processor uses this input to **LOCK** out local (BCP) accesses to data-memory. Once the remote processor has been granted the bus, LOCK gives it sole access to the bus and BCP accesses are "waited".
LCL	Out	31	0	**L**o**C**a**L**. Normally low, goes high when the BCP relinquishes the data and address bus to service a Remote Access.
5.1.6 EXTERNAL INTERRUPTS				
BIRQ	In/Out	53	In	**B**i-directional **I**nterrupt **R**e**Q**uest. As an input, can be used as an active low interrupt input (maskable and level-sensitive). As an output, can be used to generate remote system interrupts, reset via {RIC}.
NMI	In	52	X	**N**on-**M**askable **I**nterrupt. Negative edge sensitive interrupt input.

2

5.0 Device Specifications (Continued)

5.2 ABSOLUTE MAXIMUM RATINGS (Notes 1 & 2)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	$-0.5V$ to $+7.0V$
DC Input Voltage (V_{IN}) or	$-0.5V$ to $V_{CC} + 0.5V$
DC Input Diode Current	±20 mA
DC Output Voltage (V_{OUT}) or	$-0.5V$ to $V_{CC} + 0.5V$
DC Output Current, per Pin (I_{OUT})	±20 mA
DC V_{CC} or GND Current, per Pin	±50 mA
Storage Temperature Range (T_{STG})	$-65°C$ to $+150°C$
Power Dissipation (PD)	500 mW

Lead Temperature (Soldering, 10 sec)	260°C
ESD Tolerance: $C_{ZAP} = 120$ pF,	
$R_{ZAP} = 1500\Omega$	2.0 kV

5.3 OPERATING CONDITIONS

	Min	Max	Units
Supply Voltage (V_{CC})	4.5	5.5	V
DC Input or Output Voltage			
(V_{IN}, V_{OUT})	0.0	V_{CC}	V
Operating Temp. Range (T_A)	0	70	°C
Input Rise or Fall Times (t_r, t_f)		500	ns
Oscillator Crystal R_S		20	Ω
V_{CC} Power Up Ramp	6		ms

DC ELECTRICAL CHARACTERISTICS $V_{CC} = 5V \pm 10\%$ (unless otherwise specified)

Symbol	Parameter	Conditions	Guaranteed Limits 0–70°C	Units
V_{IH}	Minimum High Level Input Voltage	X1	3.8	V
		X2 (Note 3)		
		DATA-IN	2.3	V
		$\overline{RESET}$	2.3	V
		All Other Inputs except $-$ALG-IN, $+$ALG-IN	2.0	V
V_{IL}	Maximum Low Level Input Voltage	X1	1.5	V
		X2 (Note 3)		
		DATA-IN	0.6	V
		All Other Inputs except $-$ALG-IN, $+$ALG-IN	0.8	V
$V_{IH}-V_{IL}$	Minimum DATA-IN Hysteresis		0.4	V
V_{SENS}	Minimum Analog Input IN$+$, IN$-$ Differential Sensitivity	*Figure 5-8b*	25	mV
V_{BIAS}	Common Mode Analog Input Bias Voltage	User Provided Bias Voltage	Min 2.25 Max 2.75	V V
V_{OH}	Minimum High Level Output Voltage IA, A, AD All Other Outputs	$V_{IN} = V_{IH}$ or V_{IL} $\|I_{OUT}\| = 20$ μA $\|I_{OUT}\| = 4.0$ mA, $V_{CC} = 4.5V$ $\|I_{OUT}\| = 1.0$ mA, $V_{CC} = 4.5V$	$V_{CC} - 0.1$ 3.5 3.5	V V V
V_{OL}	Maximum Low Level Output Voltage IA, A, AD All Other Outputs	$V_{IN} = V_{IH}$ or V_{IL} $\|I_{OUT}\| = 20$ μA $\|I_{OUT}\| = 4.0$ mA, $V_{CC} = 4.5V$ $\|I_{OUT}\| = 1.0$ mA, $V_{CC} = 4.5V$	0.1 0.4 0.4	V V V
I_{IN}	Maximum Input Current	$V_{IN} = V_{CC}$ or GND $-$ALG-IN, $+$ALG-IN X1 (Note 3) All Others	±10 ±20 ±10	μA μA μA
I_{OZ}	Maximum TRI-STATE® Output Leakage Current	$V_{OUT} = V_{CC}$ or GND	±10	μA
I_{CC}	Maximum Operating Supply Current Total to 4 V_{CC} Pins (Note 4)	$V_{IN} = V_{CC}$ or GND TCLK = 8 MHz, CPU-CLK = 16 MHz Xcvr and CPU Operating Xcvr Idle, CPU Waited $V_{IN} = V_{CC}$ or GND TCLK = 20 MHz, CPU-CLK = 20 MHz Xcvr and CPU Operating Xcvr Idle, CPU Waited	75 30 85 31	mA mA mA mA

Note 1: Absolute Maximum Ratings are those values beyond which damage to the device may occur.

Note 2: Unless otherwise specified, all voltages are referenced to ground.

Note 3: X2 is an internal node with ESD protection. Do not use other than with crystal oscillator application.

Note 4: No DC loading, with X1 driven, no crystal. AC load per Test Circuit for Output Tests.

5.0 Device Specifications (Continued)

5.5 SWITCHING CHARACTERISTICS

The following specifications apply for V_{CC} = 4.5V to 5.5V, T_A = 0°C to 70°C.

5.5.1 Definitions

The timing specifications for the BCP are provided in the following tables and figures. The tables consist of five sections which are the following: the timing parameter symbol, the parameter ID#, the parameter description, the formula for the parameter, and the timing specification for the parameter. Below each table is a figure containing the waveforms for the parameters in the table.

The parameter symbol is composed of the type of timing specification and the signal or signals involved. Note that the symbols are unique only within a given table. The following symbol conventions are used for the type of timing specification.

t_W — Pulse width specification

t_{PD} — Propagation delay specification

t_H — Hold time specification

t_{SU} — Setup time specification

t_{ZA} — High impedance to active delay specification (enable time)

t_{AZ} — Active to high impedance delay specification (disable time)

t_{ACC} — Access time specification

t_T — Clock period specification

The parameter ID# is used to cross reference the timing parameter to the appropriate timing relationship in the accompanying figure. The waveforms in the figures are shown with the CPU clock running full speed ([CCS] = 0). For this case, CPU-CLK and CLK-OUT are equivalent. If CPU-CLK/2 is selected ([CCS] = 1), the effect on the waveforms with CLK-OUT is for CLK-OUT to double in frequency. The same is true for waveforms with X1. Note that CLK-OUT is always running at the crystal frequency and it is the CPU-CLK that is changing to half speed.

The parameter description defines the timing relationship being specified. BCP pin references are capitalized in the description.

Many of the timing specifications are dependent on variables such as operating frequency and number of programmed wait states. The formula for the parameter allows an accurate timing specification to be calculated for any combination of these variables. The formula represents the part of the timing specification that is synchronized to the internal CPU clock. This value is calculated and then added to the value specified under the Min or Max column to create the minimum or maximum guaranteed timing specification for the parameter.

The following acronyms are used in the tables:

DMEM refers to data memory

IMEM refers to instruction memory

RIC refers to the Remote Interface Control register

PC refers to the BCP Program Counter

T refers to the CPU clock period in ns

C refers to the transceiver clock period in ns

n_{IW} is the number of instruction memory wait states programmed in DCR

n_{DW} is the number of data memory wait states programmed in DCR

n_{LW} is the number of remote wait states due to a BCP local data memory access

n_{RW} is the number of CPU wait states due to a remote access

MAX(A,B) means take the greater value of A or B

The following table is an example of the format used for the timing specifications. In this example, $t_{W\text{-}RD}$ indicates a pulse width specification for the output pin $\overline{READ}$. The ID# for locating the parameter in the timing waveforms is 10. The formula for this specification involves data and instruction memory wait states and the CPU clock period. For the case of 3 data memory wait states and 0 instruction memory wait states and a CPU clock period of 50 ns, the $\overline{READ}$ low minimum pulse width would be calculated as:

$(MAX(3,0-1)+1)T+(-10) = 4T - 10 = 190$ ns

For the case of 1 data memory wait state and 3 instruction memory wait states and a CPU clock period of 50 ns, the $\overline{READ}$ low minimum pulse width would be calculated as:

$(MAX(1,3-1)+1)T+(-10) = 3T - 10 = 140$ ns

Symbol	ID#	Parameter	Formula	Min	Max	Units
$t_{W\text{-}RD}$	10	$\overline{Read}$ Low	$(MAX(n_{DW},n_{IW}-1)+1)T+$	-10		ns

2

5.0 Device Specifications (Continued)

Test Circuit for Output Tests

Note 1: S_1 = V_{CC} for t_{PZL}, and t_{PLZ} measurements

S_1 = GND for t_{PZH}, and t_{PHZ} measurements

S_1 = Open for push pull outputs

Note 2: R_L = 1.1k for 4 mA outputs

R_L = 4.4k for 1 mA outputs

Note 3: C_L includes scope and jig capacitance.
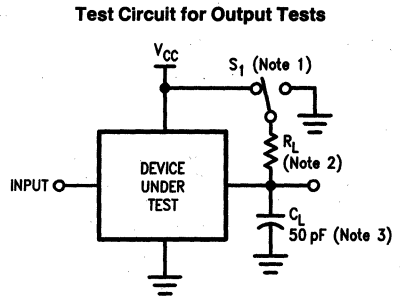
TL/F/9336-A2

Propagation Delay Waveforms Except for Oscillator
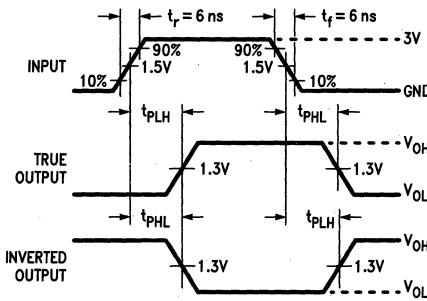
TL/F/9336-A3

Propagation Delay Waveform for Oscillator

TL/F/9336-A4

Setup and Hold Time Waveforms

TL/F/9336-A6

Note 1: Waveform for negative edge sensitive circuits will be inverted.
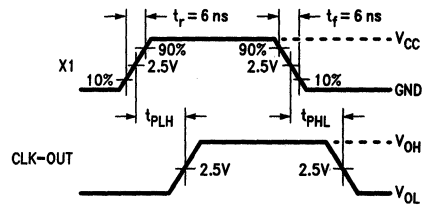
Input Pulse Width Waveforms

TL/F/9336-A5

TRI-STATE Output Enable and Disable Waveforms
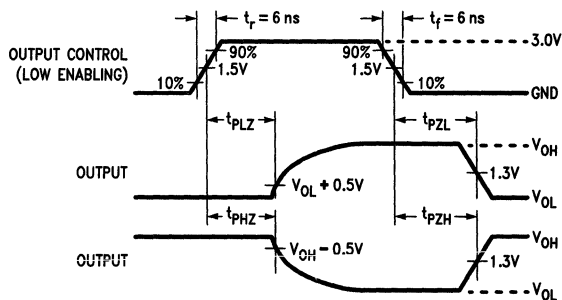
TL/F/9336-A7

FIGURE 5-2. Switching Characteristic Measurement Waveforms

5.0 Device Specifications (Continued)

TABLE 5-3. Data Memory Read Timing (Note 1)

Symbol	ID #	Parameter	Formula	Min	Max	Units
$t_{W\text{-}ALE}$	1	ALE High	$(n_{RW}+1)T+$	-10	12	ns
$t_{PD\text{-}AAD\text{-}ALE}$	2	A, AD (Data Address) Valid to ALE Falling	$T+$	-28		ns
$t_{PD\text{-}ALE\text{-}AD}$	3	ALE Falling to AD (Data Address) Invalid	$0.5T+$	0		ns
$t_{H\text{-}RD\text{-}DATA}$	4	Data Valid after $\overline{READ}$ Rising		0		ns
$t_{AZ\text{-}RD\text{-}AD}$	5	$\overline{READ}$ Falling to AD Disabled			35	ns
$t_{SU\text{-}RD\text{-}DATA}$	6	$\overline{READ}$ Falling to AD (Data) SetUp	$(MAX(n_{DW},n_{IW}-1)+1)T+$		-26	ns
$t_{ZA\text{-}RD\text{-}AD}$	7	$\overline{READ}$ Rising to AD Enabled		7		ns
$t_{PD\text{-}AAD\text{-}RD}$	8	A, AD (Data Address) Valid before $\overline{READ}$ Falling	$1.5T+$	-32		ns
$t_{W\text{-}RD}$	9	$\overline{READ}$ Low	$(MAX(n_{DW},n_{IW}-1)+1)T+$	-10	10	ns
$t_{ACC\text{-}D}$	10	Data Memory Read Time	$(MAX(n_{DW},n_{IW}-1)+2.5)T+$		-52	ns
$t_{SU\text{-}AD\text{-}DATA}$	11	AD Disabled to AD (Data) Setup	$(MAX(n_{DW},n_{IW}-1)+1)T+$	-50		ns
$t_{PD\text{-}ALE\text{-}AAD}$	12	ALE Rising to A, AD (Data Address) Valid	$(n_{RW})T+$		26	ns
$t_{PD\text{-}RD\text{-}A}$	13	$\overline{READ}$ Rising to A Invalid	$0.5T+$	0		ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

TL/F/9336–52

FIGURE 5-3. Data Memory Read Timing

2

5.0 Device Specifications (Continued)

TABLE 5-4. Data Memory Write Timing (Note 1)

Symbol	ID#	Parameter	Formula	Min	Max	Units
$t_{W\text{-}ALE}$	1	ALE High	$(n_{RW}+1)T+$	-10	12	ns
$t_{PD\text{-}AAD\text{-}ALE}$	2	A, AD (Data Address) Valid to ALE Falling	$T+$	-28		ns
$t_{PD\text{-}ALE\text{-}AD}$	3	ALE Falling to AD (Data Address) Invalid	$0.5T+$	-2		ns
$t_{PD\text{-}DATA\text{-}WR}$	4	AD (Data) Valid to $\overline{WRITE}$ Rising	$(MAX(n_{DW},n_{IW}-1)+1)T+$	-20		ns
$t_{PD\text{-}AAD\text{-}WR}$	5	A, AD (Data Address) Valid to $\overline{WRITE}$ Falling	$1.5T+$	-28		ns
$t_{PD\text{-}WR\text{-}DATA}$	6	$\overline{WRITE}$ Falling to AD (Data) Valid			19	ns
$t_{PD\text{-}WR\text{-}DATAz}$	7	$\overline{WRITE}$ Rising to AD (Data) Invalid	$0.5T+$	5		ns
$t_{W\text{-}WR}$	8	$\overline{WRITE}$ Low	$(MAX(n_{DW},n_{IW}-1)+1)T+$	-10	10	ns
$t_{PD\text{-}ALE\text{-}AAD}$	9	ALE Rising to A, AD (Data Address) Valid	$(n_{RW})T+$		26	ns
$t_{PD\text{-}WR\text{-}A}$	10	$\overline{WRITE}$ Rising to A Invalid	$0.5T+$	-2		ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.
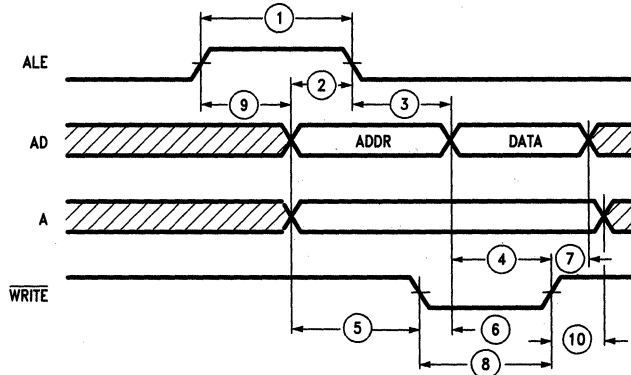
TL/F/9336–53

FIGURE 5-4. Data Memory Write Timing

5.0 Device Specifications (Continued)

TABLE 5-5. Instruction Memory Read Timing (Note 1)

Symbol	ID#	Parameter	Formula	Min	Max	Units
t_{ACC-I}	1	Instruction Memory Read Time	$(n_{IW} + 1.5)T +$		-24	ns
t_{H-IA-I}	2	IA Invalid to I Invalid		0		ns
$t_{PD-ICLK-IA}$	3	ICLK Rising to IA Invalid	$0.5T +$	-17		ns
$t_{PD-IA-ICLK}$	4	Next IA Valid before ICLK Falling	$0.5T +$	-12		ns
$t_{PD-IAz-ICLK}$		IA Invalid before ICLK Falling			17	ns
$t_{SU-I-ICLK}$	5	I Valid before ICLK Rising		27		ns
$t_{H-I-ICLK}$	6	I Invalid before ICLK Falling	$0.5T +$		25	ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

TL/F/9336–A9

(a) Instruction Memory Read Timing

TL/F/9336–54

(b) Instruction ICLK Timing
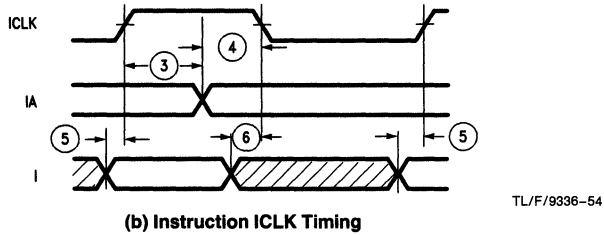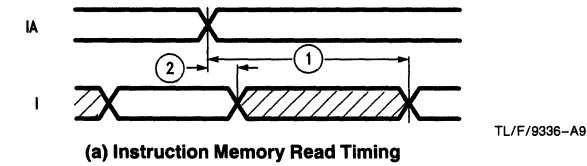
FIGURE 5-5. Instruction Memory Timing

5.0 Device Specifications (Continued)

TABLE 5-6. Clock Timing (Note 1)

Symbol	ID#	Parameter	Formula	Min	Max	Units
t_{T-X1}	1	X1 Period (Note 2)		50	500	ns
$t_{PD-XI-CO}$	2	X1 to CLK-OUT (Note 2)			49	ns
$t_{PD-CO-ICLKr}$	3	CLK-OUT Rising to ICLK Rising			20	ns
$t_{PD-CO-ICLKf}$	4	CLK-OUT Rising to ICLK Falling (Note 3)			20	ns
t_{T-XT}	5	X-TCLK Period (Note 4)		50	500	ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

Note 2: Measurement thresholds at 2.5V.

Note 3: The falling edge of ICLK occurs only after the next IA becomes valid. The CLK-OUT cycle in which this occurs depends on the instruction being executed.

Note 4: There is no relationship between X1 and X-TCLK. X-TCLK is fully asynchronous.
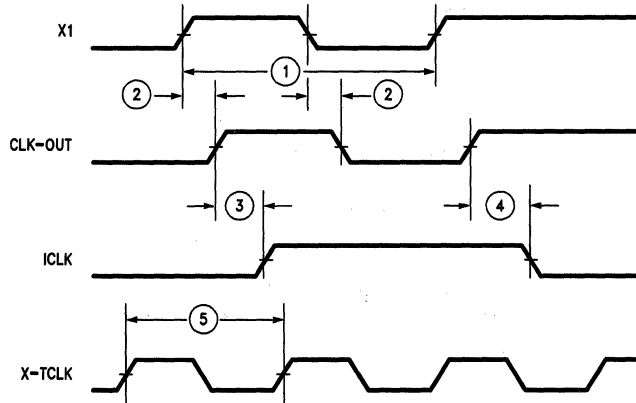
TL/F/9336-55

FIGURE 5-6. Clock Timing

5.0 Device Specifications (Continued)

TABLE 5-7. Transceiver Timing (Note 1)

Symbol	ID #	Parameter	Formula	Min	Max	Units
$t_{PD-X1-TA}$	1	X1 Rising to TX-ACT Rising/Falling		18	87	ns
$t_{PD-XTCLK-TA}$	2	X-TCLK Rising to TX-ACT Rising/Falling		13	67	ns
$t_{PD-TA-DO}$	3	TX-ACT Rising/Falling to DATA-OUT Falling/Rising (Note 2)		−12	12	ns
$t_{W-DO-HB}$	4	DATA-OUT Half Bit Cell Width	4C+	−10	10	ns
$t_{W-DO-FB}$	5	DATA-OUT Full Bit Cell Width	8C+	−10	10	ns
$t_{PD-DO-DD}$	6	DATA-OUT Falling/Rising to DATA-DLY Rising/Falling (Note 3)	2C+	−10	10	ns
$t_{PD-TA-DD}$	7	TX-ACT Falling to DATA-DLY Rising/Falling (Note 4)		−2		ns
$t_{PD-DOr-DDf}$	8	DATA-OUT Rising to DATA-DLY Falling (Note 5)	2C+	−19	5	ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

Note 2: (a) shows the beginning of a transmission for all protocols with [ATA] = 0 (solid line), and [ATA] = 1 (dashed line). (b) shows the ending of a 5250 protocol transmission with no line hold ({ATR[7–3]} = 00000, solid line), and with one half bit time line hold ({ATR[7–3]} = 00001, dashed line). When TX-ACT falls, $t_{PD-TA-DO}$ is valid only in 5250 modes with [TIN] = 1 and 3270/3299/8-bit modes with [TIN] = 0. In other modes, DATA-OUT is already high when TX-ACT falls so there is no transition.

Note 3: Valid only when TX-ACT = 1 excluding first and last quarter bit time of a frame. Between frames (i.e. TX-ACT = 0), DATA-DLY = [TIN] and DATA-OUT = 1 in all protocols.

Note 4: Valid in 5250 mode with [TIN] = 1 and {ATR[7–3]} = 00000; 5250 mode with [TIN] = 0 and {ATR[7–3]} = XXXXX.

Note 5: Valid in 5250 mode with [TIN] = 1 and {ATR[7–3]} ≠ 00000. Applies only to the last transition of DATA-OUT and DATA-DLY prior to TX-ACT falling.
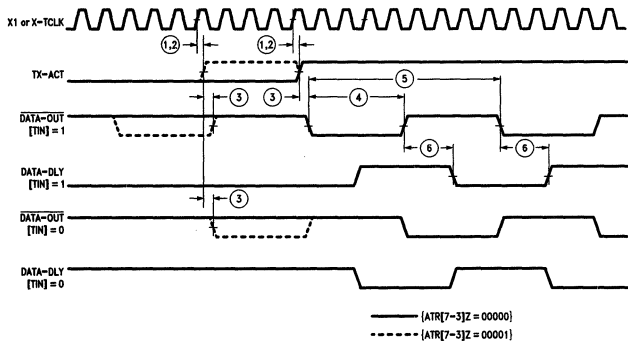
TL/F/9336–56

(a) Transmission Beginning Timing

TL/F/9336–57
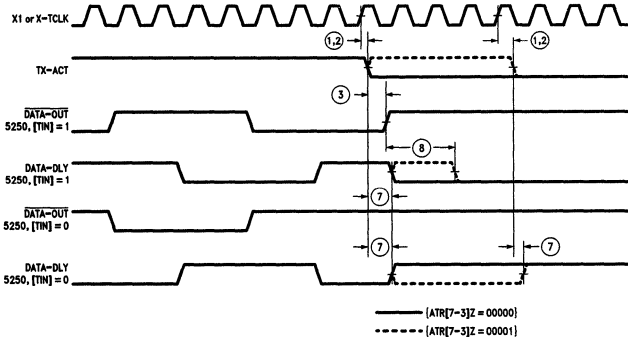
(b) Transmission Ending Timing

FIGURE 5-7. Transceiver Timing

5.0 Device Specifications (Continued)

TABLE 5-8. Analog and DATA-IN Timing (Note 1)

Symbol	ID#	Parameter	Formula	Min	Max	Units
$t_{W-DI-hb}$	1	DATA-IN Data, Half Bit Width	3C+	12		ns
			5C+		−12	ns
$t_{W-DI-fb}$	2	DATA-IN Data, Full Bit Width	7C+	12		ns
			9C+		−12	ns
$t_{W-AI-hb}$	3	Analog Data, Half Bit Width (−ALG-IN or +ALG-IN)	3C+	20		ns
			5C+		−20	ns
$t_{W-AI-fb}$	4	Analog Data, Full Bit Width (−ALG-IN or +ALG-IN)	7C+	20		ns
			9C+		−20	ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.
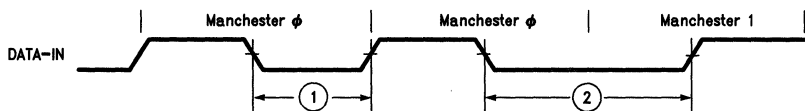
TL/F/9336–58

(a) DATA-IN Jitter Timing (3270)

TL/F/9336–59
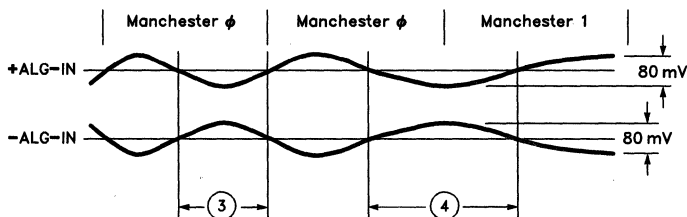
(b) Analog Jitter Timing (3270)

FIGURE 5-8. Analog and DATA-IN Timing

5.0 Device Specifications (Continued)

TABLE 5-9. Interrupt Timing (Note 1)

Symbol	ID #	Parameter	Formula	Min	Max	Units
$t_{SU\text{-}NMI\text{-}CO}$	1	$\overline{NMI}$ Falling before CLK-OUT Falling		7		ns
$t_{W\text{-}NMI}$	2	$\overline{NMI}$ Low	2T+	0		ns
$t_{SU\text{-}BQ\text{-}CO}$	3	$\overline{BIRQ}$ (Input) Falling before CLK-OUT Falling		10		ns
$t_{PD\text{-}ICLK\text{-}BQ}$	4	ICLK Rising to $\overline{BIRQ}$ (Output) Rising/Falling			33	ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

(a) Interrupt Timing

TL/F/9336–60

(b) $\overline{BIRQ}$ Output Timing

TL/F/9336–61
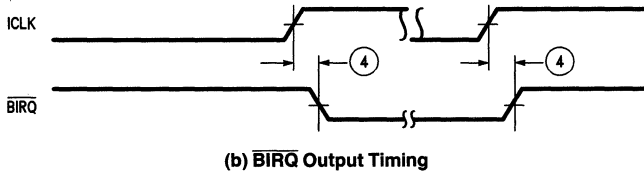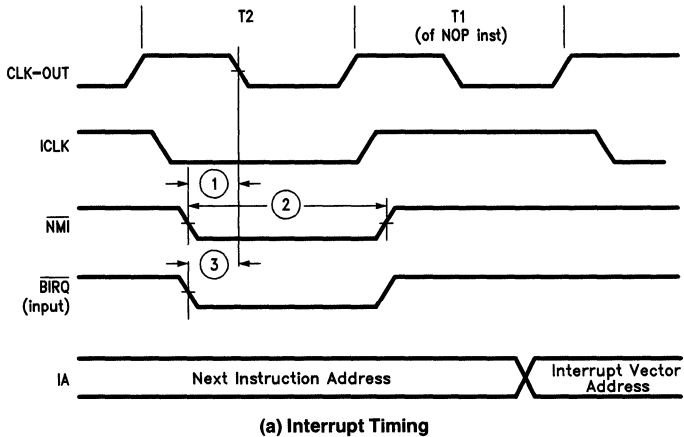
FIGURE 5-9. Interrupt Timing

5.0 Device Specifications (Continued)

TABLE 5-10. Control Pin Timing (Note 1)

Symbol	ID#	Parameter	Formula	Min	Max	Units
t_{W-RST}	1	$\overline{RESET}$ Low	$10T+$	0		ns
$t_{PD-RST-ICLK}$	2	$\overline{RESET}$ Rising to ICLK Rising	$4T+$		0	ns
$t_{SU-ALE-WT}$	3	$\overline{WAIT}$ Low after ALE High to Extend Cycle	$(MAX(n_{DW},n_{IW}-1)+1)T+$		-30	ns
$t_{H-WT-ALE}$	4	$\overline{WAIT}$ Rising after ALE Falling (Note 2)		0		ns
			$(MAX(n_{DW},n_{IW}-1)+1)T+$		-28	ns
$t_{PD-WT-RDWR}$	5	$\overline{WAIT}$ Rising to $\overline{READ}$ or $\overline{WRITE}$ Rising	$1.5T+$	-22		ns
			$2.5T+$		2	ns
$t_{SU-RRW-RST}$	6	$\overline{REM-RD}$, $\overline{REM-WR}$ Low to $\overline{RESET}$ Rising for BCP to Start		8		ns
$t_{H-RST-RRW}$	7	$\overline{REM-RD}$, $\overline{REM-WR}$ Low after $\overline{RESET}$ Rising for BCP to Start		10		ns
$t_{SU-LK-ICLK}$	8	$\overline{LOCK}$ Low before ICLK High (Note 3)	$0.5T+$	23		ns
$t_{PD-LK-ALE}$	9	$\overline{LOCK}$ High to ALE Low	$T+$	-1		ns
			$3T+$		25	ns
$t_{SU-WT-ICLK}$	10	$\overline{WAIT}$ Low after ICLK Rising to Extend Cycle (Note 4)	$(MAX(n_{DW},n_{IW}-1)+0.5)T+$		-29	ns
$t_{H-WT-ICLK}$	11	$\overline{WAIT}$ High after ICLK Rising (Notes 2, 4)	$(MAX(n_{DW},n_{IW}-1)+0.5)T+$	1		ns
			$(MAX(n_{DW},n_{IW}-1)+1.5)T+$		-28	ns
$t_{H-LK-ICLK}$	12	$\overline{LOCK}$ Rising after ICLK High	$0.5T+$	2		ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

Note 2: The maximum value for this parameter is the lastest $\overline{WAIT}$ can be removed without adding an additional T-state. The formula assumes a minimum externally generated wait of one T-state.

Note 3: If $t_{SU-LK-ICLK}$ is not met, the maximum time from $\overline{LOCK}$ low till no more local accesses is $T(MAX(n_{DW}, n_{IW}-1)+3)$.

Note 4: The formula(s) apply to a 2 T-state instruction. For a three T-state instruction, add one T-state. For a four T-state instruction, add two T-states.

5.0 Device Specifications (Continued)

(a) Reset Timing

TL/F/9336-62

(b) BCP Access WAIT Timing

TL/F/9336-63

(c) LOCK Timing

TL/F/9336-64

(d) Instruction WAIT Timing

TL/F/9336-A8
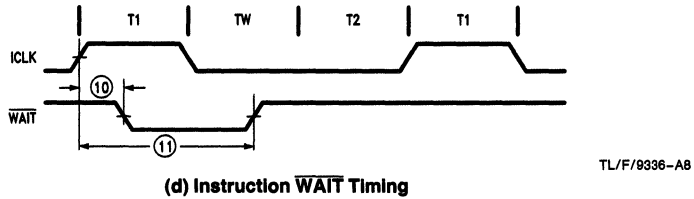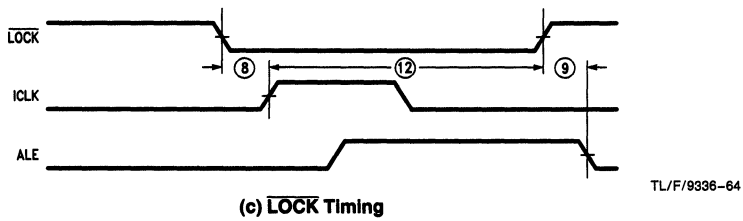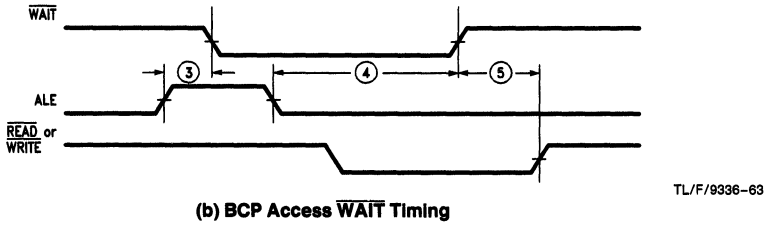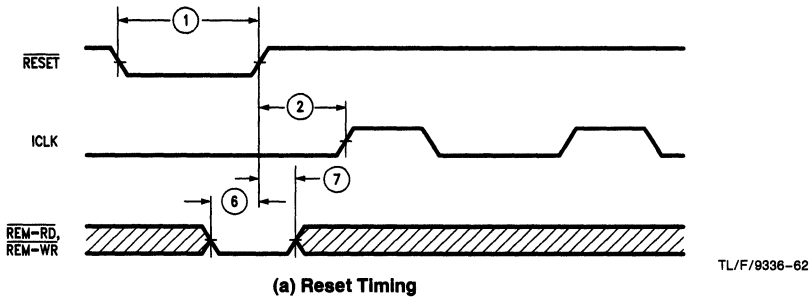
FIGURE 5-10. Control Pin Timing

5.0 Device Specifications (Continued)

TABLE 5-11. Buffered Read of PC, RIC (Note 1)

Symbol	ID#	Parameter	Formula	Min	Max	Units
$t_{SU-RRR-CO}$	1	$\overline{RAE}$, $\overline{REM-RD}$ Falling before CLK-OUT Rising		24		ns
$t_{H-RRR-X}$	2	$\overline{RAE}$, $\overline{REM-RD}$ Rising after XACK Rising (Note 2)		0		ns
			2T+		−44	ns
$t_{SU-CMD-RRR}$	3	CMD Valid before $\overline{RAE}$, $\overline{REM-RD}$ Falling		0		ns
$t_{H-CMD-RRR}$	4	CMD Invalid after $\overline{RAE}$, $\overline{REM-RD}$ Falling	T+	26		ns
$t_{PD-RRR-X}$	5	$\overline{RAE}$, $\overline{REM-RD}$ Falling to XACK Falling			37	ns
$t_{PD-X-LCL}$	6	XACK Falling to $\overline{LCL}$ Rising	$(n_{LW}+1)T+$	−5		ns
$t_{PD-LCL-X}$	7	$\overline{LCL}$ Rising to XACK Rising	2T+	−10	8	ns
$t_{PD-RRR-LCL}$	8	$\overline{RAE}$, $\overline{REM-RD}$ Rising to $\overline{LCL}$ Falling	T+	6		ns
$t_{AZ-LCL-A}$	9	A Disabled after $\overline{LCL}$ Rising			13	ns
$t_{ZA-A-LCL}$	10	A Enabled before $\overline{LCL}$ Falling			5	ns
$t_{PD-LCL-PC}$	11	$\overline{LCL}$ Rising to AD (PC) Valid	T+		31	ns
$t_{PD-PC-X}$	12	AD (PC, RIC) Valid before XACK Rising	T+	−30		ns
$t_{PD-RRR-PC}$	13	$\overline{RAE}$, $\overline{REM-RD}$ Rising to AD (PC) Invalid		8		ns
t_{W-PC}	14	AD (PC, RIC) Valid Time	T+	−2		ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

Note 2: The maximum value for this parameter is the latest $\overline{RAE}$, $\overline{REM-RD}$ can be removed without adding a T-state to the remote access.
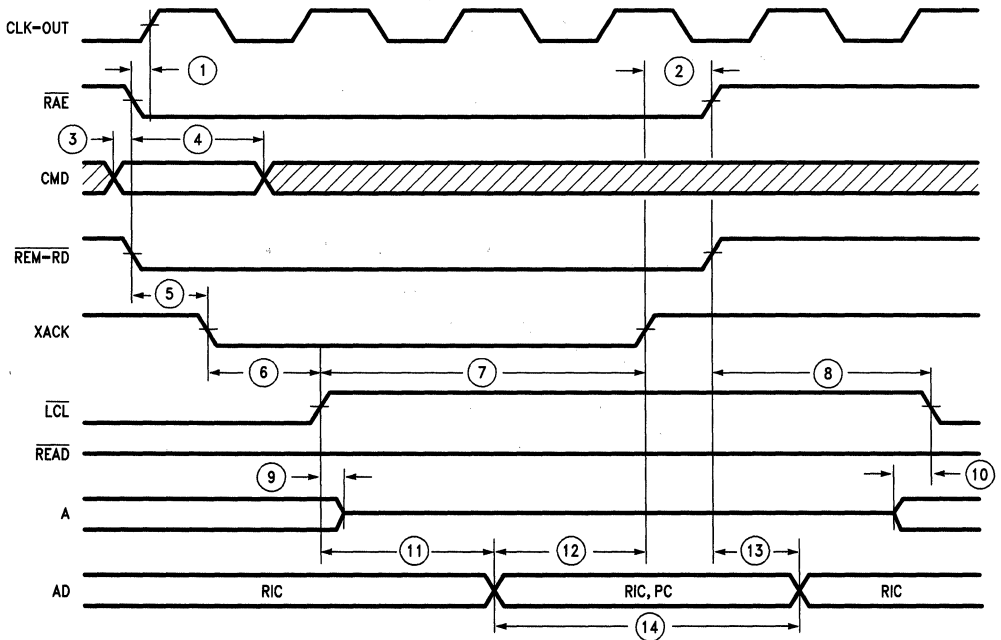
FIGURE 5-11. Buffered Read of PC, RIC

TL/F/9336–65

5.0 Device Specifications (Continued)

TABLE 5-12. Buffered Read of DMEM (Note 1)

Symbol	ID #	Parameter	Formula	Min	Max	Units
$t_{SU-RRR-CO}$	1	$\overline{RAE}$, $\overline{REM-RD}$ Falling before CLK-OUT Rising		24		ns
$t_{H-RRR-X}$	2	$\overline{RAE}$, $\overline{REM-RD}$ Rising after XACK Rising (Note 2)		0		ns
			2T+		−44	ns
$t_{SU-CMD-RRR}$	3	CMD Valid before $\overline{RAE}$, $\overline{REM-RD}$ Falling		0		ns
$t_{H-CMD-RRR}$	4	CMD Invalid after $\overline{RAE}$, $\overline{REM-RD}$ Falling	T+	26		ns
$t_{PD-RRR-X}$	5	$\overline{RAE}$, $\overline{REM-RD}$ Falling to XACK Falling			37	ns
$t_{PD-X-LCL}$	6	XACK Falling to $\overline{LCL}$ Rising	$(n_{LW}+1)T+$	−5		ns
$t_{PD-LCL-X}$	7	$\overline{LCL}$ Rising to XACK Rising	$(n_{DW}+2)T+$	−10	8	ns
$t_{PD-RRR-LCL}$	8	$\overline{RAE}$, $\overline{REM-RD}$ Rising to $\overline{LCL}$ Falling	T+	6		ns
$t_{PD-LCL-RD}$	9	$\overline{LCL}$ Rising to $\overline{READ}$ Falling	T+	−4	16	ns
$t_{PD-RD-X}$	10	$\overline{READ}$ Falling to XACK Rising	$(n_{DW}+1)T+$	−17		ns
$t_{PD-RRR-RD}$	11	$\overline{RAE}$, $\overline{REM-RD}$ Rising to $\overline{READ}$ Rising		5	36	ns
$t_{AZ-LCL-AAD}$	12	A, AD Disabled after $\overline{LCL}$ Rising			27	ns
$t_{ZA-AAD-LCL}$	13	A, AD Enabled before $\overline{LCL}$ Falling			5	ns
t_{W-RD}	14	Read Low	$(n_{DW}+1)T+$	−2		ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

Note 2: The maximum value for this parameter is the latest $\overline{RAE}$, $\overline{REM-RD}$ can be removed without adding a T-state to the remote access.
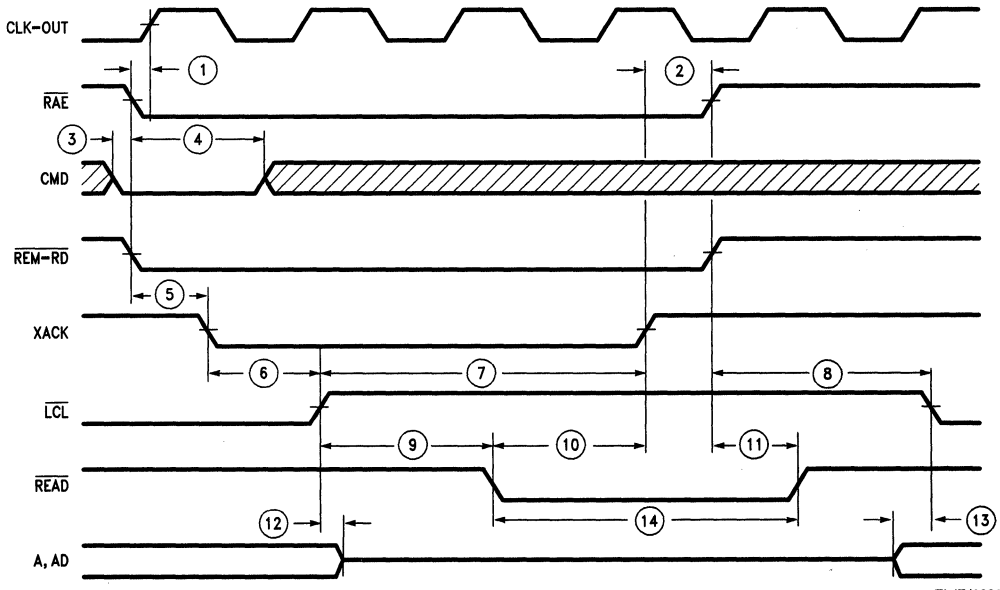
TL/F/9336–66

FIGURE 5-12. Buffered Read of DMEM

5.0 Device Specifications (Continued)

TABLE 5-13. Buffered Read of IMEM (Note 1)

Symbol	ID#	Parameter	Formula	Min	Max	Units
$t_{SU-RRR-CO}$	1	$\overline{RAE}$, $\overline{REM-RD}$ Falling before CLK-OUT Rising		24		ns
$t_{H-RRR-X}$	2	$\overline{RAE}$, $\overline{REM-RD}$ Rising after XACK Rising (Note 2)		0		ns
			$2T+$		-44	ns
$t_{SU-CMD-RRR}$	3	CMD Valid before $\overline{RAE}$, $\overline{REM-RD}$ Falling		0		ns
$t_{H-CMD-RRR}$	4	CMD Invalid after $\overline{RAE}$, $\overline{REM-RD}$ Falling	$T+$	26		ns
$t_{PD-RRR-X}$	5	$\overline{RAE}$, $\overline{REM-RD}$ Falling to XACK Falling			37	ns
$t_{PD-X-LCL}$	6	XACK Falling to $\overline{LCL}$ Rising	$T+$	-5		ns
$t_{PD-LCL-X}$	7	$\overline{LCL}$ Rising to XACK Rising	$(n_{IW}+2)T+$	-10	8	ns
$t_{PD-RRR-LCL}$	8	$\overline{RAE}$, $\overline{REM-RD}$ Rising to $\overline{LCL}$ Falling	$T+$	6		ns
$t_{AZ-LCL-A}$	9	A Disabled after $\overline{LCL}$ Rising			13	ns
$t_{ZA-A-LCL}$	10	A Enabled before $\overline{LCL}$ Falling			5	ns
$t_{PD-IMEM-X}$	11	AD (IMEM) Valid before XACK Rising	$(n_{IW}+1)T+$	-32		ns
$t_{PD-RRR-IMEM}$	12	AD (IMEM) Invalid after $\overline{RAE}$, $\overline{REM-RD}$ Rising		10		ns
$t_{PD-LCL-IMEM}$	13	$\overline{LCL}$ Rising to AD (IMEM) Valid	$T+$		33	ns
t_{W-IMEM}	14	(IMEM) Valid	$(n_{IW}+1)T+$	1		ns
$t_{PD-IA-LCL}$	15	Next IA Valid to $\overline{LCL}$ Falling (Note 3)	$0.5T+$	-3	20	ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

Note 2: The maximum value for this parameter is the latest $\overline{RAE}$, $\overline{REM-RD}$ can be removed without adding a T-state to the remote access.

Note 3: Two remote reads from instruction memory are necessary to read a 16-bit instruction word from IMEM—low byte followed by high byte. The timing for the two reads are the same except that IA is incremented after the high instruction memory byte is read.
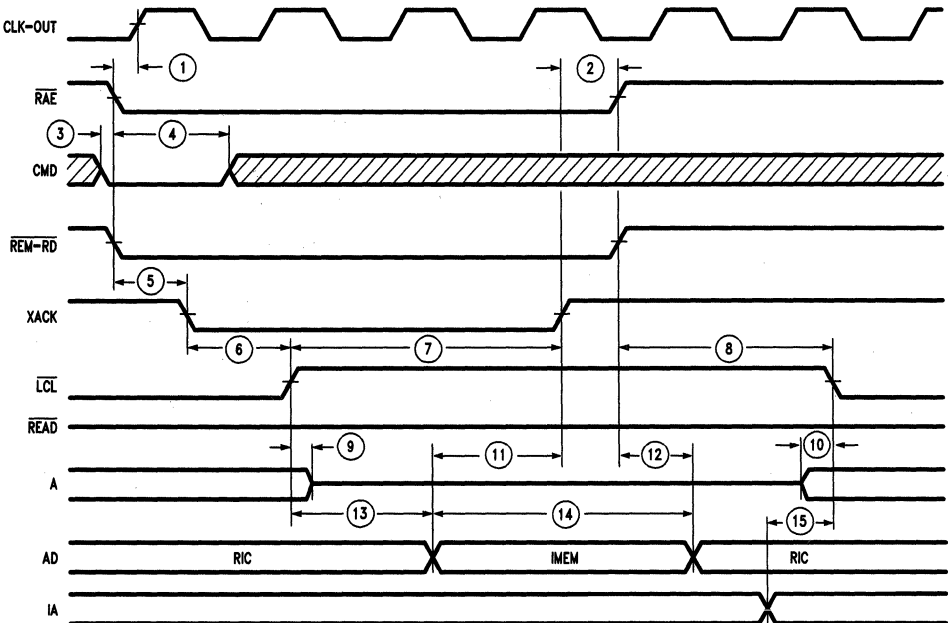
TL/F/9336–67

FIGURE 5-13. Buffered Read of IMEM

5.0 Device Specifications (Continued)

TABLE 5-14. Latched Read of PC, RIC (Note 1)

Symbol	ID #	Parameter	Formula	Min	Max	Units
$t_{SU-RRR-CO}$	1	$\overline{RAE}$, $\overline{REM-RD}$ Falling before CLK-OUT Rising		24		ns
$t_{H-RRR-X}$	2	$\overline{RAE}$, $\overline{REM-RD}$ Rising after XACK Rising		0		ns
$t_{SU-CMD-RRR}$	3	CMD Valid before $\overline{RAE}$, $\overline{REM-RD}$ Falling		0		ns
$t_{H-CMD-RRR}$	4	CMD Invalid after $\overline{RAE}$, $\overline{REM-RD}$ Falling	$T+$	26		ns
$t_{PD-RRR-X}$	5	$\overline{RAE}$, $\overline{REM-RD}$ Falling to XACK Falling			37	ns
$t_{PD-Xf-LCLr}$	6	XACK Falling to $\overline{LCL}$ Rising	$(n_{LW}+1)T+$	-5		ns
$t_{PD-LCL-X}$	7	$\overline{LCL}$ Rising to XACK Rising	$2T+$	-10	8	ns
$t_{PD-Xr-LCLf}$	8	XACK Rising to $\overline{LCL}$ Falling	$T+$	-11	11	ns
$t_{AZ-LCL-A}$	9	A Disabled after $\overline{LCL}$ Rising			13	ns
$t_{ZA-A-LCL}$	10	A Enabled before $\overline{LCL}$ Falling			5	ns
$t_{PC-LCL-PC}$	11	$\overline{LCL}$ Rising to AD (PC) Valid	$T+$		31	ns
$t_{PD-PC-X}$	12	AD (PC) Valid before XACK Rising	$T+$	-30		ns
$t_{PD-X-PC}$	13	XACK Rising to AD (PC) Invalid	$0.5T+$	2		ns
t_{W-PC}	14	AD (PC, RIC) Valid	$1.5T+$	-12		ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.
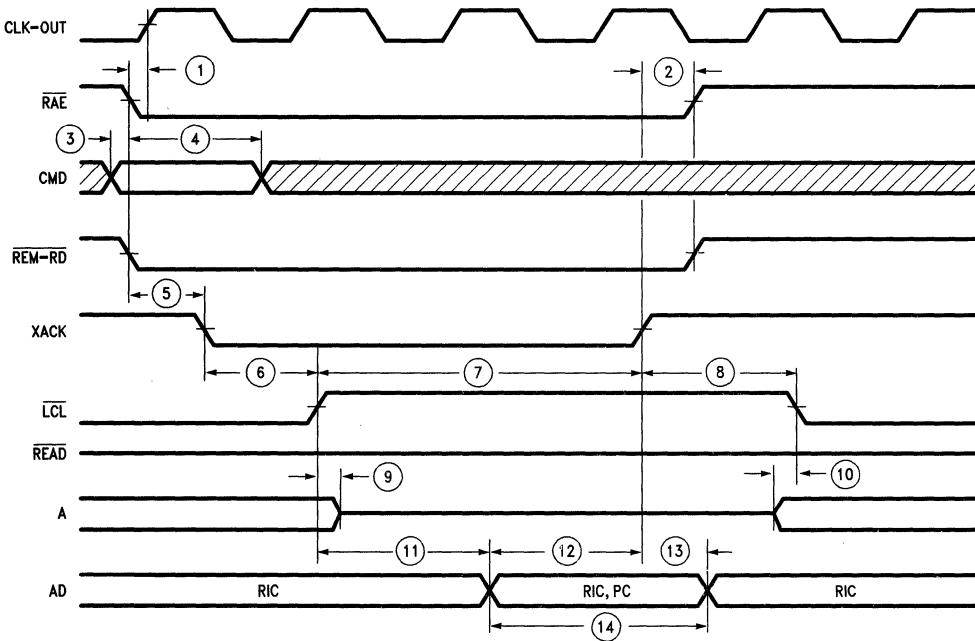
TL/F/9336–68

FIGURE 5-14. Latched Read of PC, RIC

5.0 Device Specifications (Continued)

TABLE 5-15. Latched Read of DMEM (Note 1)

Symbol	ID#	Parameter	Formula	Min	Max	Units
$t_{SU\text{-}RRR\text{-}CO}$	1	$\overline{RAE}$, $\overline{REM\text{-}RD}$ Falling before CLK-OUT Rising		24		ns
$t_{H\text{-}RRR\text{-}X}$	2	$\overline{RAE}$, $\overline{REM\text{-}RD}$ Rising after XACK Rising		0		ns
$t_{SU\text{-}CMD\text{-}RRR}$	3	CMD Valid before $\overline{RAE}$, $\overline{REM\text{-}RD}$ Falling		0		ns
$t_{H\text{-}CMD\text{-}RRR}$	4	CMD Invalid after $\overline{RAE}$, $\overline{REM\text{-}RD}$ Falling	$T+$	26		ns
$t_{PD\text{-}RRR\text{-}X}$	5	$\overline{RAE}$, $\overline{REM\text{-}RD}$ Falling to XACK Falling			37	ns
$t_{PD\text{-}Xf\text{-}LCLr}$	6	XACK Falling to $\overline{LCL}$ Rising	$(n_{LW}+1)T+$	-5		ns
$t_{PD\text{-}LCL\text{-}X}$	7	$\overline{LCL}$ Rising to XACK Rising	$(n_{DW}+2)T+$	-10	8	ns
$t_{PD\text{-}Xr\text{-}LCLf}$	8	XACK Rising to $\overline{LCL}$ Falling	$T+$	-11	11	ns
$t_{PC\text{-}LCL\text{-}RD}$	9	$\overline{LCL}$ Rising to $\overline{READ}$ Falling	$T+$	-4	16	ns
$t_{PD\text{-}RD\text{-}X}$	10	$\overline{READ}$ Falling before XACK Rising	$(n_{DW}+1)T+$	-17		ns
$t_{PD\text{-}X\text{-}RD}$	11	XACK Rising to $\overline{READ}$ Rising	$0.5T+$	-7	12	ns
$t_{AZ\text{-}LCL\text{-}AAD}$	12	A, AD Disabled after $\overline{LCL}$ Rising			27	ns
$t_{ZA\text{-}AAD\text{-}LCL}$	13	A, AD Enabled before $\overline{LCL}$ Falling			5	ns
$t_{W\text{-}RD}$	14	$\overline{READ}$ Low	$(n_{DW}+1.5)T+$	-13		ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.
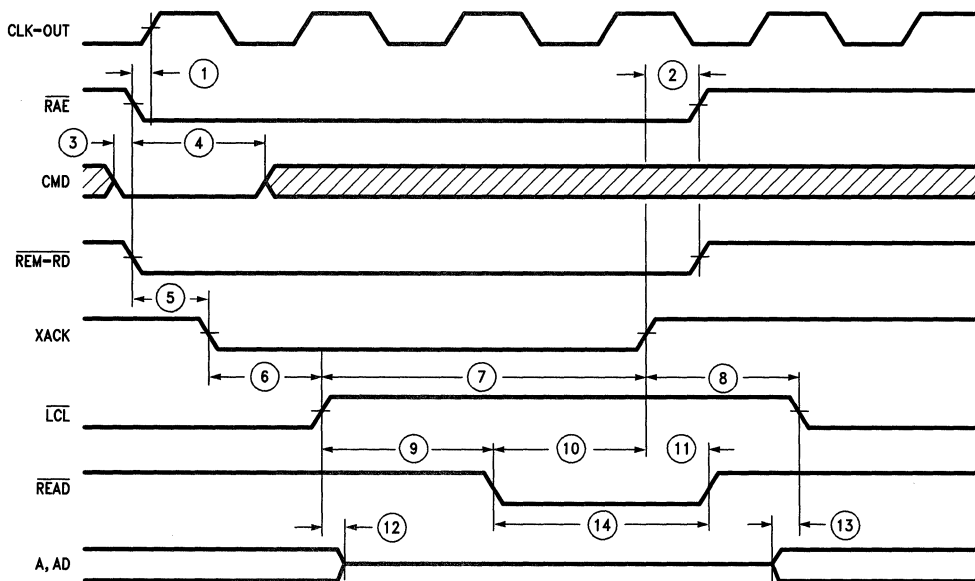
FIGURE 5-15. Latched Read of DMEM

TL/F/9336–69

5.0 Device Specifications (Continued)

TABLE 5-16. Latched Read of IMEM (Note 1)

Symbol	ID #	Parameter	Formula	Min	Max	Units
$t_{SU-RRR-CO}$	1	$\overline{RAE}$, $\overline{REM-RD}$ Falling before CLK-OUT Rising		24		ns
$t_{H-RRR-X}$	2	$\overline{RAE}$, $\overline{REM-RD}$ Rising after XACK Rising		0		ns
$t_{SU-CMD-RRR}$	3	CMD Valid before $\overline{RAE}$, $\overline{REM-RD}$ Falling		0		ns
$t_{H-CMD-RRR}$	4	CMD Invalid after $\overline{RAE}$, $\overline{REM-RD}$ Falling	$T+$	26		ns
$t_{PD-RRR-X}$	5	$\overline{RAE}$, $\overline{REM-RD}$ Falling to XACK Falling			37	ns
$t_{PD-Xf-LCLr}$	6	XACK Falling to $\overline{LCL}$ Rising	$T+$	-5		ns
$t_{PD-LCL-X}$	7	$\overline{LCL}$ Rising to XACK Rising	$(n_{IW}+2)T+$	-10	8	ns
$t_{PD-Xr-LCLf}$	8	XACK Rising to $\overline{LCL}$ Falling	$T+$	-11	11	ns
$t_{AZ-LCL-A}$	9	A Disabled after $\overline{LCL}$ Rising			13	ns
$t_{ZA-A-LCL}$	10	A Enabled before $\overline{LCL}$ Falling			5	ns
$t_{PD-LCL-IMEM}$	11	$\overline{LCL}$ Rising to AD (IMEM) Valid	$T+$		33	ns
$t_{PD-IMEM-X}$	12	AD (IMEM) Valid to XACK Rising	$(n_{IW}+1)T+$	-32		ns
$t_{PD-X-IMEM}$	13	XACK Rising to AD (IMEM) Invalid	$0.5T+$	9		ns
$t_{PD-LCL-IA}$	14	$\overline{LCL}$ Falling to Next IA Valid (Note 2)	$0.5T+$	-25	5	ns
t_{W-IMEM}	15	IMEM Valid	$(n_{IW}+1.5)T+$	-6		ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

Note 2: Two remote reads from instruction memory are necessary to read a 16-bit instruction word from IMEM—low byte followed by high byte. The timing for the two reads are the same except that IA is incremented after the high instruction memory byte is read.
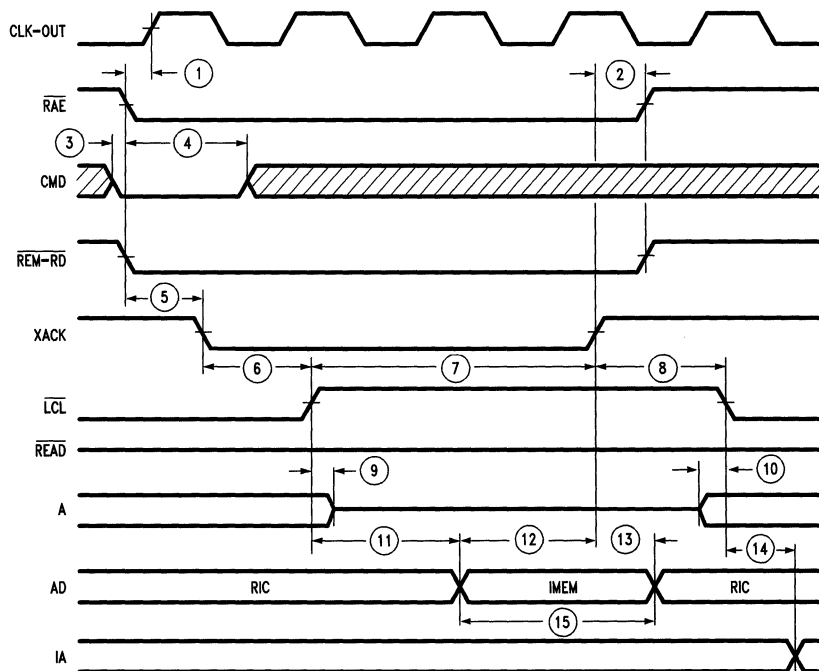
FIGURE 5-16. Latched Read of IMEM

TL/F/9336–70

5.0 Device Specifications (Continued)

TABLE 5-17. Slow Buffered Write of PC, RIC (Note 1)

Symbol	ID#	Parameter	Formula	Min	Max	Units
$t_{SU-RRW-CO}$	1	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling before CLK-OUT Rising		27		ns
$t_{H-RRW-X}$	2	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Rising after XACK Rising (Note 2)		0		ns
			$2T+$		-45	ns
$t_{SU-CMD-RRW}$	3	CMD Valid before $\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling		0		ns
$t_{H-CMD-RRW}$	4	CMD Invalid after $\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling	$T+$	26		ns
$t_{PD-RRW-X}$	5	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling to XACK Falling			40	ns
$t_{PD-X-LCL}$	6	XACK Falling to $\overline{LCL}$ Rising	$(n_{LW}+1)T+$	-5		ns
$t_{PD-LCL-X}$	7	$\overline{LCL}$ Rising to XACK Rising	$2T+$	-10	8	ns
$t_{PD-RRW-LCL}$	8	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Rising to $\overline{LCL}$ Falling	$T+$	6		ns
$t_{AZ-LCL-AAD}$	9	A, AD Disabled after $\overline{LCL}$ Rising			27	ns
$t_{ZA-AAD-LCL}$	10	A, AD Enabled before $\overline{LCL}$ Falling			5	ns
$t_{SU-RDAT-RRW}$	11	AD (Data) Valid before $\overline{RAE}$, $\overline{REM\text{-}WR}$ Rising		15		ns
$t_{H-RDAT-RRW}$	12	AD (Data) Invalid after $\overline{RAE}$, $\overline{REM\text{-}WR}$ Rising		15		ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

Note 2: The maximum value for this parameter is the latest $\overline{RAE}$, $\overline{REM\text{-}WR}$ can be removed without adding a T-state to the remote access.
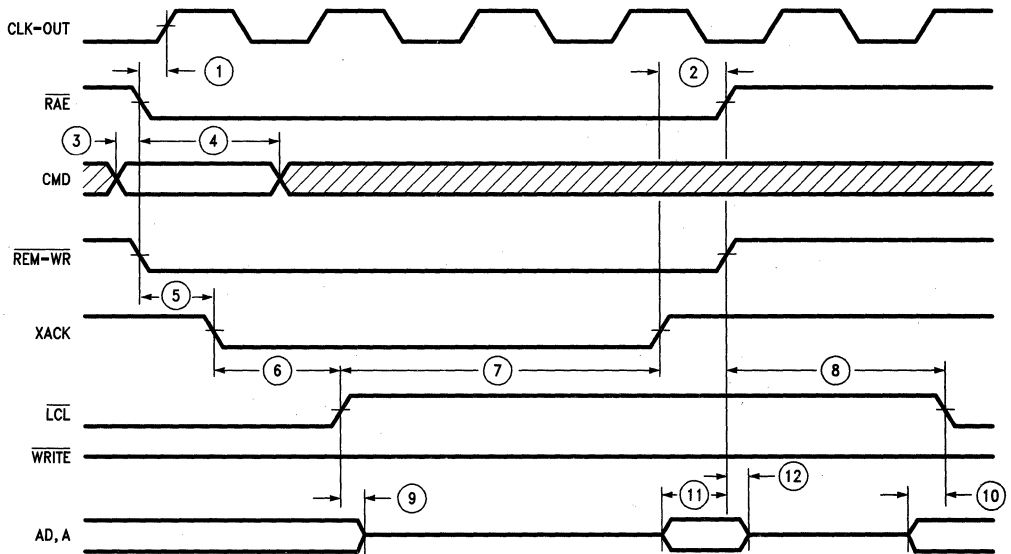
FIGURE 5-17. Slow Buffered Write of PC, RIC

TL/F/9336–71

5.0 Device Specifications (Continued)

TABLE 5-18. Slow Buffered Write of DMEM (Note 1)

Symbol	ID #	Parameter	Formula	Min	Max	Units
$t_{SU-RRW-CO}$	1	$\overline{RAE}$, $\overline{REM-WR}$ Falling before CLK-OUT Rising		27		ns
$t_{H-RRW-X}$	2	$\overline{RAE}$, $\overline{REM-WR}$ Rising after XACK Rising (Note 2)		0		ns
			$2T+$		-45	ns
$t_{SU-CMD-RRW}$	3	CMD Valid before $\overline{RAE}$, $\overline{REM-WR}$ Falling		0		ns
$t_{H-CMD-RRW}$	4	CMD Invalid after $\overline{RAE}$, $\overline{REM-WR}$ Falling	$T+$	26		ns
$t_{PD-RRW-X}$	5	$\overline{RAE}$, $\overline{REM-WR}$ Falling to XACK Falling			40	ns
$t_{PD-X-LCL}$	6	XACK Falling to $\overline{LCL}$ Rising	$(n_{LW}+1)T+$	-5		ns
$t_{PD-LCL-X}$	7	$\overline{LCL}$ Rising to XACK Rising	$(n_{DW}+2)T+$	-10	8	ns
$t_{PD-RRW-LCL}$	8	$\overline{RAE}$, $\overline{REM-WR}$ to $\overline{LCL}$ Falling	$T+$	6		ns
$t_{PD-LCL-WR}$	9	$\overline{LCL}$ Rising to $\overline{WRITE}$ Falling	$T+$	0		ns
$t_{PD-WR-X}$	10	$\overline{WRITE}$ Falling to XACK Rising	$(n_{DW}+1)T+$	-22		ns
$t_{PD-RRW-WR}$	11	$\overline{RAE}$, $\overline{REM-WR}$ Rising to $\overline{WRITE}$ Rising		7	43	ns
$t_{AZ-LCL-AAD}$	12	A, AD Disabled after $\overline{LCL}$ Rising			27	ns
$t_{ZA-AAD-LCL}$	13	A, AD Enabled before $\overline{LCL}$ Falling			5	ns
t_{W-WR}	14	$\overline{WRITE}$ Low	$(n_{DW}+1)T+$	-2		ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

Note 2: The maximum value for this parameter is the latest $\overline{RAE}$, $\overline{REM-WR}$ can be removed without adding a T-state to the remote access.
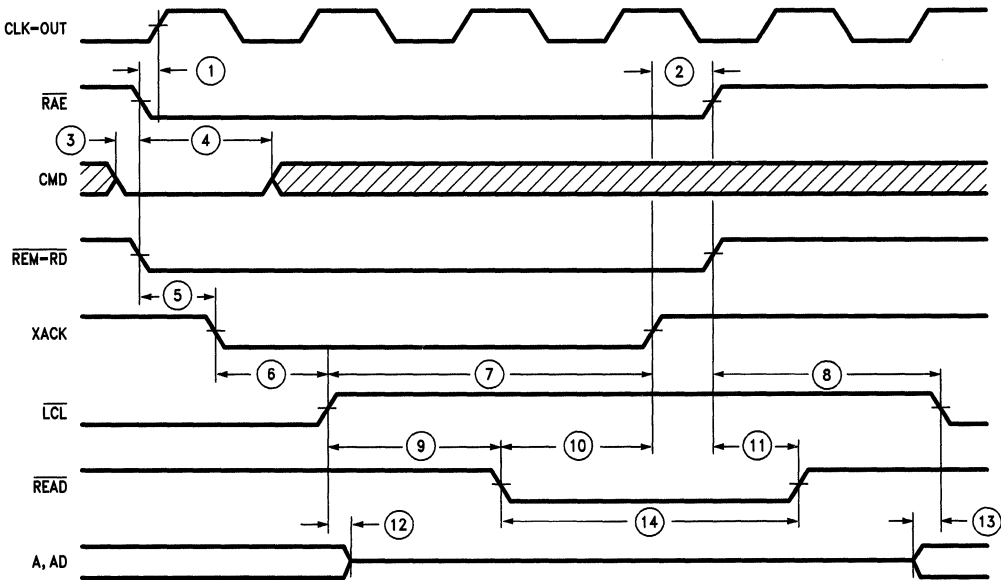
TL/F/9336-72

FIGURE 5-18. Slow Buffered Write of DMEM

5.0 Device Specifications (Continued)

TABLE 5-19. Slow Buffered Write of IMEM (Notes 1, 2)

Symbol	ID#	Parameter	Formula	Min	Max	Units
$t_{SU-RRW-CO}$	1	$\overline{RAE}$, $\overline{REM-WR}$ Falling before CLK-OUT Rising		27		ns
$t_{H-RRW-X}$	2	$\overline{RAE}$, $\overline{REM-WR}$ Rising after XACK Rising (Note 3)		0		ns
			$2T+$		-45	ns
$t_{SU-CMD-RRW}$	3	CMD Valid before $\overline{RAE}$, $\overline{REM-WR}$ Falling		0		ns
$t_{H-CMD-RRW}$	4	CMD Invalid after $\overline{RAE}$, $\overline{REM-WR}$ Falling	$T+$	26		ns
$t_{PD-RRW-X}$	5	$\overline{RAE}$, $\overline{REM-WR}$ Falling to XACK Falling			40	ns
$t_{PD-X-LCL}$	6	XACK Falling to $\overline{LCL}$ Rising	$T+$	-5		ns
$t_{PD-LCL-X}$	7	$\overline{LCL}$ Rising to XACK Rising	$(n_{IW}+2)T+$	-10	8	ns
$t_{PD-RRW-LCL}$	8	$\overline{RAE}$, $\overline{REM-WR}$ to $\overline{LCL}$ Falling	$T+$	6		ns
$t_{AZ-LCL-AAD}$	9	A, AD Disabled after $\overline{LCL}$ Rising			27	ns
$t_{AZ-AAD-LCL}$	10	A, AD Enabled before $\overline{LCL}$ Falling			5	ns
$t_{PD-RDAT-I}$	11	AD (Data) Valid to I Valid			44	ns
$t_{H-RDAT-RRW}$	12	AD (Data) Invalid after $\overline{RAE}$, $\overline{REM-WR}$ Rising		14		ns
$t_{PD-IA-LCL}$	13	Next IA Valid to $\overline{LCL}$ Falling	$0.5T+$	-3	20	ns
$t_{PD-LCL-IWR}$	14	$\overline{LCL}$ Rising to $\overline{IWR}$ Falling	$T+$	1		ns
$t_{PD-IWR-X}$	15	$\overline{IWR}$ Falling before XACK Rising	$(n_{IW}+1)T+$	-30		ns
$t_{PD-RRW-IWR}$	16	$\overline{RAE}$, $\overline{REM-WR}$ Rising to $\overline{IWR}$ Rising		8		ns
$t_{ZA-IWR-I}$	17	$\overline{IWR}$ Falling to I Enabled		0		ns
$t_{AZ-IWR-I}$	18	$\overline{IWR}$ Rising to I Disabled		28	52	ns
$t_{PD-I-IWR}$	19	I Valid before $\overline{IWR}$ Rising	$(n_{IW}+1)T+$	-10		ns
t_{W-IWR}	20	$\overline{IWR}$ Low	$(n_{IW}+1)T+$	-2		ns
$t_{PD-I-IA}$	21	I Disabled to IA Invalid	$0.5T+$	-64		ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

Note 2: Two remote writes to instruction memory are necessary to store a 16-bit instruction word to IMEM—low byte followed by high byte. The timing for the 2nd write is shown in the following diagram. The timing of the first write is the same as a write of the PC or RIC.

Note 3: The maximum value for this parameter is the latest $\overline{RAE}$, $\overline{REM-WR}$ can be removed without adding a T-state to the remote access.

5.0 Device Specifications (Continued)
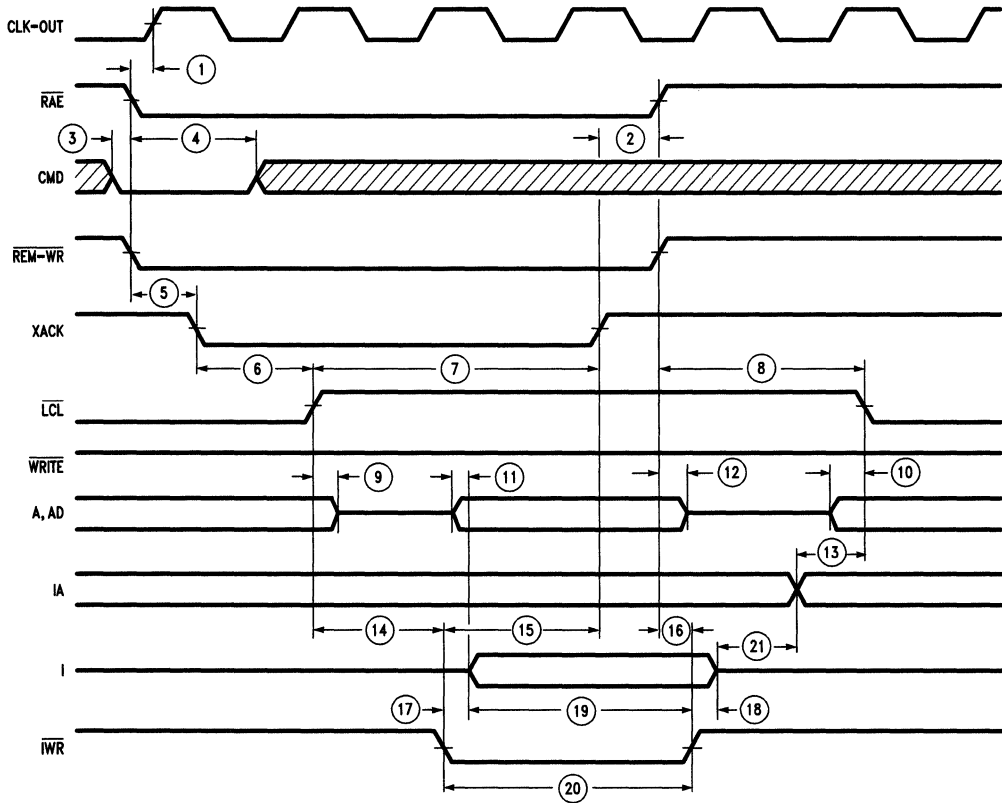
TL/F/9336–73

FIGURE 5-19. Slow Buffered Write of IMEM

5.0 Device Specifications (Continued)

TABLE 5-20. Fast Buffered Write of RIC, PC (Note 1)

Symbol	ID#	Parameter	Formula	Min	Max	Units
$t_{SU\text{-}RRW\text{-}CO}$	1	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling before CLK-OUT Rising		27		ns
$t_{H\text{-}RRW\text{-}X}$	2	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Rising after XACK Rising		0		ns
$t_{SU\text{-}CMD\text{-}RRW}$	3	CMD Valid before $\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling		0		ns
$t_{H\text{-}CMD\text{-}RRW}$	4	CMD Invalid after $\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling	$T+$	26		ns
$t_{PD\text{-}RRW\text{-}X}$	5	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling to XACK Falling			40	ns
$t_{PD\text{-}Xf\text{-}LCLr}$	6	XACK Falling to $\overline{LCL}$ Rising	$(n_{LW}+1)T+$	-5		ns
$t_{PD\text{-}LCL\text{-}X}$	7	$\overline{LCL}$ Rising to XACK Rising	$2T+$	-10	8	ns
$t_{PD\text{-}Xr\text{-}LCLf}$	8	XACK Rising to $\overline{LCL}$ Falling	$T+$	-11	11	ns
$t_{AZ\text{-}LCL\text{-}AAD}$	9	A, AD Disabled after $\overline{LCL}$ Rising			27	ns
$t_{AZ\text{-}AAD\text{-}LCL}$	10	A, AD Enabled before $\overline{LCL}$ Falling			5	ns
$t_{SU\text{-}RDAT\text{-}X}$	11	AD (Data) Valid before XACK Rising		21		ns
$t_{H\text{-}RDAT\text{-}X}$	12	AD (Data) Invalid after XACK Rising		3		ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.
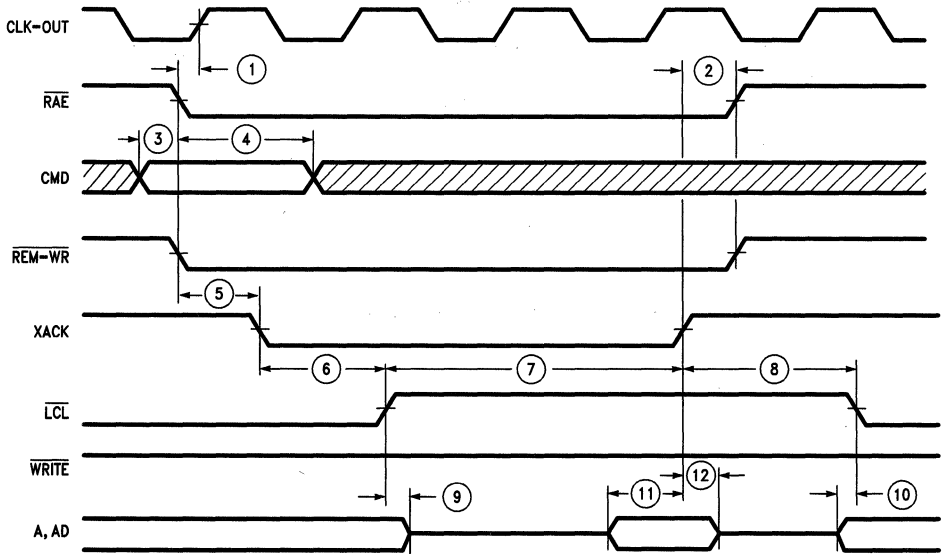
TL/F/9336–74

FIGURE 5-20. Fast Buffered Write of RIC, PC

5.0 Device Specifications (Continued)

TABLE 5-21. Fast Buffered Write of DMEM (Note 1)

Symbol	ID#	Parameter	Formula	Min	Max	Units
$t_{SU-RRW-CO}$	1	$\overline{RAE}$, $\overline{REM-WR}$ Falling before CLK-OUT Rising		27		ns
$t_{H-RRW-X}$	2	$\overline{RAE}$, $\overline{REM-WR}$ Rising after XACK Rising		0		ns
$t_{SU-CMD-RRW}$	3	CMD Valid before $\overline{RAE}$, $\overline{REM-WR}$ Falling		0		ns
$t_{H-CMD-RRW}$	4	CMD Invalid after $\overline{RAE}$, $\overline{REM-WR}$ Falling	T+	26		ns
$t_{PD-RRW-X}$	5	$\overline{RAE}$, $\overline{REM-WR}$ Falling to XACK Falling			40	ns
$t_{PD-Xf-LCLr}$	6	XACK Falling to $\overline{LCL}$ Rising	$(n_{LW}+1)T+$	-5		ns
$t_{PD-LCL-X}$	7	$\overline{LCL}$ Rising to XACK Rising	$(n_{DW}+2)T+$	-10	8	ns
$t_{PD-Xr-LCLf}$	8	XACK Rising to $\overline{LCL}$ Falling	T+	-11	11	ns
$t_{PD-LCL-WR}$	9	$\overline{LCL}$ Rising to $\overline{WRITE}$ Falling	T+	-1		ns
$t_{PD-WR-X}$	10	$\overline{WRITE}$ Falling to XACK Rising	$(n_{DW}+1)T+$	-22		ns
$t_{PD-X-WR}$	11	XACK Rising to $\overline{WRITE}$ Rising		0	20	ns
$t_{AZ-LCL-AAD}$	12	A, AD Disabled after $\overline{LCL}$ Rising			27	ns
$t_{ZA-AAD-LCL}$	13	A, AD Enabled before $\overline{LCL}$ Falling			5	ns
t_{W-WR}	14	$\overline{WRITE}$ Low	$(n_{DW}+1)T+$	-10		ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

TL/F/9336–75
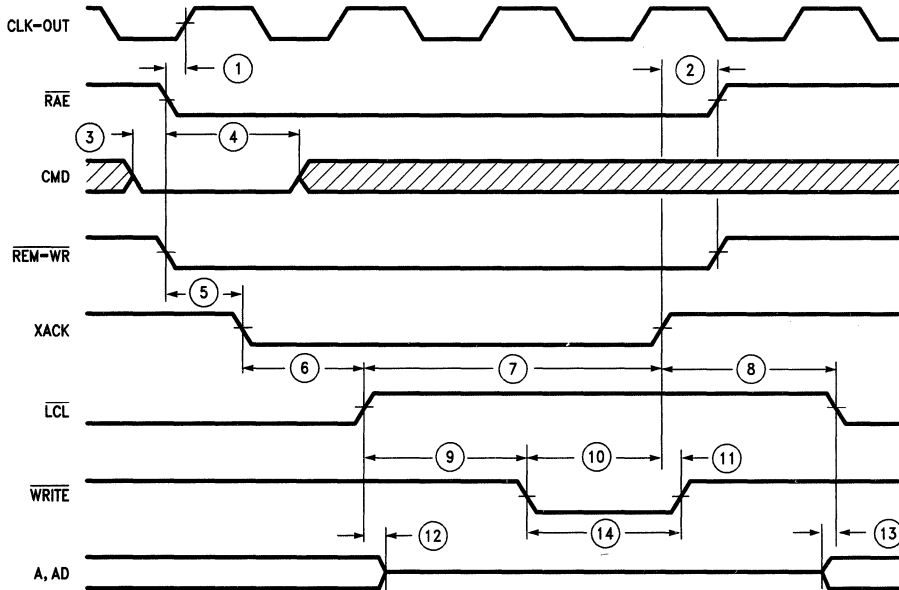
FIGURE 5-21. Fast Buffered Write of DMEM

2

5.0 Device Specifications (Continued)

TABLE 5-22. Fast Buffered Write of IMEM (Notes 1, 2)

Symbol	ID#	Parameter	Formula	Min	Max	Units
$t_{SU-RRW-CO}$	1	$\overline{RAE}$, $\overline{REM-WR}$ Falling before CLK-OUT Rising		27		ns
$t_{H-RRW-X}$	2	$\overline{RAE}$, $\overline{REM-WR}$ Rising after XACK Rising		0		ns
$t_{SU-CMD-RRW}$	3	CMD Valid before $\overline{RAE}$, $\overline{REM-WR}$ Falling		0		ns
$t_{H-CMD-RRW}$	4	CMD Invalid after $\overline{RAE}$, $\overline{REM-WR}$ Falling	T+	26		ns
$t_{PD-RRW-X}$	5	$\overline{RAE}$, $\overline{REM-WR}$ Falling to XACK Falling			40	ns
$t_{PD-Xf-LCLr}$	6	XACK Falling to $\overline{LCL}$ Rising	T+	−5		ns
$t_{PD-LCL-X}$	7	$\overline{LCL}$ Rising to XACK Rising	$(n_{IW}+2)$T+	−10	8	ns
$t_{PD-Xr-LCLf}$	8	XACK Rising to $\overline{LCL}$ Falling	T+	−11	11	ns
$t_{AZ-LCL-AAD}$	9	A, AD Disabled after $\overline{LCL}$ Rising			27	ns
$t_{ZA-AAD-LCL}$	10	A, AD Enabled before $\overline{LCL}$ Falling			5	ns
$t_{PD-RDAT-I}$	11	AD (Data) Valid to I Valid			44	ns
$t_{H-RDAT-X}$	12	AD (Data) Invalid after XACK Rising		3		ns
$t_{PD-IWR-X}$	13	$\overline{IWR}$ Falling before XACK Rising	$(n_{IW}+1)$T+	−30		ns
$t_{PD-LCL-IA}$	14	$\overline{LCL}$ Falling to next IA Valid	0.5T+	−25	5	ns
$t_{PD-LCL-IWR}$	15	$\overline{LCL}$ Rising to $\overline{IWR}$ Falling	T+	1		ns
$t_{PD-X-IWR}$	16	XACK Rising to $\overline{IWR}$ Rising		0		ns
$t_{ZA-IWR-I}$	17	$\overline{IWR}$ Falling to I Enabled		0		ns
$t_{AZ-IWR-I}$	18	$\overline{IWR}$ Rising to I Disabled		28	54	ns
$t_{PD-I-IWR}$	19	I Valid before $\overline{IWR}$ Rising	$(n_{IW}+1)$T+	−22		ns
t_{W-IWR}	20	$\overline{IWR}$ Low Time	$(n_{IW}+1)$T+	−12		ns
$t_{PD-I-IA}$	21	I Disabled to IA Invalid	1.5T+	−89		ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

Note 2: Two remote writes to instruction memory are necessary to store a 16-bit instruction word to IMEM—low byte followed by high byte. The timing of the 2nd write is shown in the following diagram. The timing of the first write is the same as a write of the PC or RIC as shown in *Figure 5-20*.

5.0 Device Specifications (Continued)
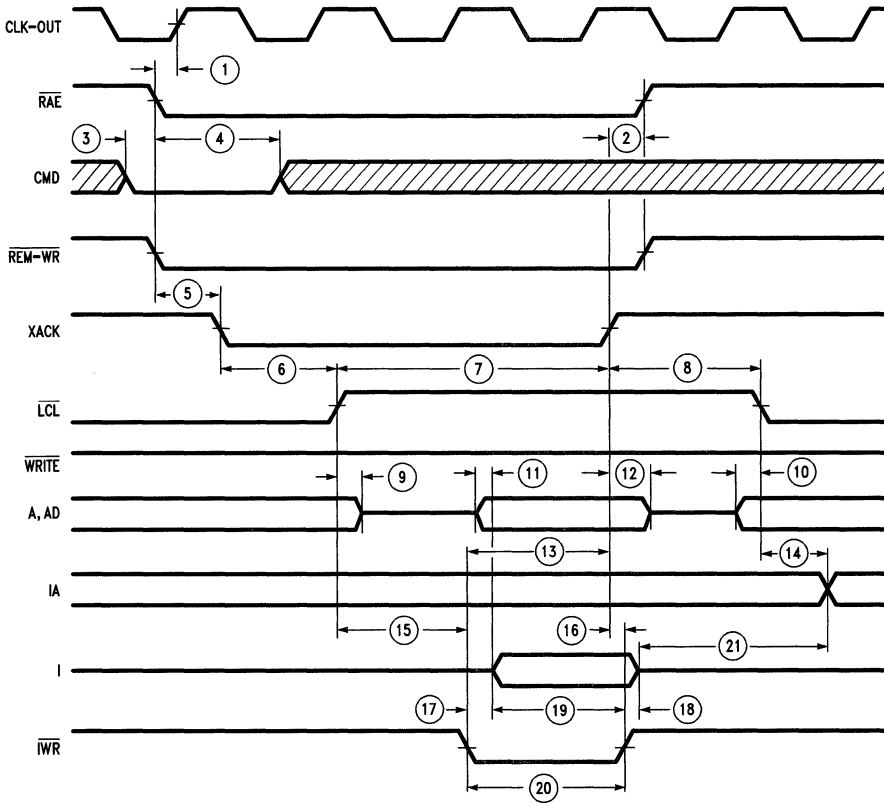
TL/F/9336–76

FIGURE 5-22. Fast Buffered Write of IMEM

5.0 Device Specifications (Continued)

TABLE 5-23. Latched Write of PC, RIC (Note 1)

Symbol	ID#	Parameter	Formula	Min	Max	Units
$t_{SU-RRW-CO}$	1	$\overline{RAE}$, $\overline{REM-WR}$ Falling before CLK-OUT Rising		27		ns
$t_{H-RRW-CO}$	2	$\overline{RAE}$, $\overline{REM-WR}$ Rising after CLK-OUT Rising (Note 2)	0.5T+	6		ns
			T+		−20	ns
$t_{H-RRW-X}$	3	$\overline{RAE}$, $\overline{REM-WR}$ Rising after XACK Rising		0		ns
$t_{SU-CMD-RRW}$	4	CMD Valid before $\overline{RAE}$, $\overline{REM-WR}$ Falling		0		ns
$t_{H-CMD-RRW}$	5	CMD Invalid after $\overline{RAE}$, $\overline{REM-WR}$ Falling	T+	26		ns
$t_{PD-RRW-X}$	6	$\overline{RAE}$, $\overline{REM-WR}$ Falling to XACK Falling			40	ns
$t_{SU-RDAT-LCL}$	7	AD (Data) Valid after $\overline{LCL}$ Rising	2T+		−24	ns
$t_{H-RDAT-LCL}$	8	AD (Data) Invalid after $\overline{LCL}$ Rising	2T+	2		ns
$t_{AZ-LCL-AAD}$	9	A, AD Disabled after $\overline{LCL}$ Rising			27	ns
$t_{ZA-AAD-LCL}$	10	A, AD Enabled before $\overline{LCL}$ Falling			5	ns
$t_{PD-RRW-WPND}$	11	$\overline{RAE}$, $\overline{REM-WR}$ Rising to $\overline{WR-PEND}$ Falling		8		ns
			T+		47	ns
$t_{SU-CMD-WPND}$	12	CMD Valid before $\overline{WR-PEND}$ Rising		25		ns
$t_{H-CMD-WPND}$	13	CMD Invalid after $\overline{WR-PEND}$ Rising		0		ns
$t_{SU-RRWr-CO}$	14	$\overline{RAE}$, $\overline{REM-WR}$ Rising before CLK-OUT Rising		20		ns
$t_{PD-X-WPND}$	15	XACK Rising to $\overline{WR-PEND}$ Rising			13	ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

Note 2: The maximum value for this parameter is the latest $\overline{RAE}$, $\overline{REM-WR}$ can be removed without delaying the remote access by one T-state.
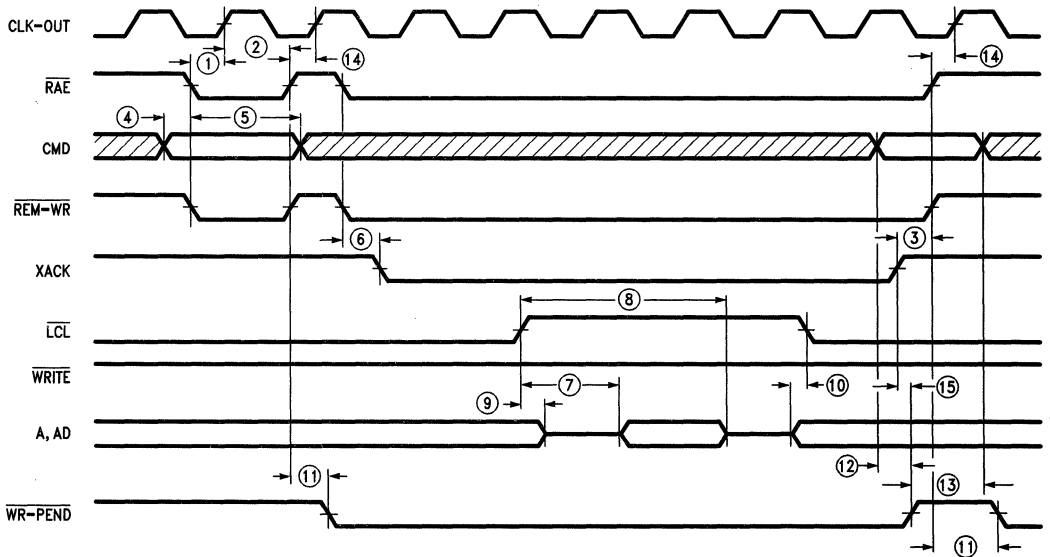
TL/F/9336–77

FIGURE 5-23. Latched Write of PC, RIC

5.0 Device Specifications (Continued)

TABLE 5-24. Latched Write of DMEM (Note 1)

Symbol	ID#	Parameter	Formula	Min	Max	Units
$t_{SU-RRW-CO}$	1	$\overline{RAE}$, $\overline{REM-WR}$ Falling before CLK-OUT Rising		27		ns
$t_{H-RRW-CO}$	2	$\overline{RAE}$, $\overline{REM-WR}$ Rising after CLK-OUT Rising (Note 2)	0.5T+	6		ns
			T+		−20	ns
$t_{H-RRW-X}$	3	$\overline{RAE}$, $\overline{REM-WR}$ Rising after XACK Rising		0		ns
$t_{SU-CMD-RRW}$	4	CMD Valid before $\overline{RAE}$, $\overline{REM-WR}$ Falling		0		ns
$t_{H-CMD-RRW}$	5	CMD Invalid after $\overline{RAE}$, $\overline{REM-WR}$ Falling	T+	26		ns
$t_{PD-RRW-X}$	6	$\overline{RAE}$, $\overline{REM-WR}$ Falling to XACK Falling			40	ns
$t_{PD-LCL-WR}$	7	$\overline{LCL}$ Rising to $\overline{WRITE}$ Falling	T+	−1		ns
$t_{PD-WR-LCL}$	8	$\overline{WRITE}$ Rising to $\overline{LCL}$ Falling	T+	−20		ns
$t_{AZ-LCL-AAD}$	9	A, AD Disabled after $\overline{LCL}$ Rising			27	ns
$t_{ZA-AAD-LCL}$	10	A, AD Enabled before $\overline{LCL}$ Falling			5	ns
t_{W-WR}	11	$\overline{WRITE}$ Low Time	$(n_{DW}+1)T+$	−10		ns
$t_{PD-RRW-WPND}$	12	$\overline{RAE}$, $\overline{REM-WR}$ Rising to $\overline{WR-PEND}$ Falling		8		ns
			T+		47	ns
$t_{SU-CMD-WPND}$	13	CMD Valid before $\overline{WR-PEND}$ Rising		25		ns
$t_{H-CMD-WPND}$	14	CMD Invalid after $\overline{WR-PEND}$ Rising		0		ns
$t_{SU-RRWr-CO}$	15	$\overline{RAE}$, $\overline{REM-WR}$ Rising before CLK-OUT Rising		20		ns
$t_{PD-X-WPND}$	16	XACK Rising to $\overline{WR-PEND}$ Rising			13	ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

Note 2: The maximum value for this parameter is the latest $\overline{RAE}$, $\overline{REM-WR}$ can be removed without delaying the remote access by one T-state.
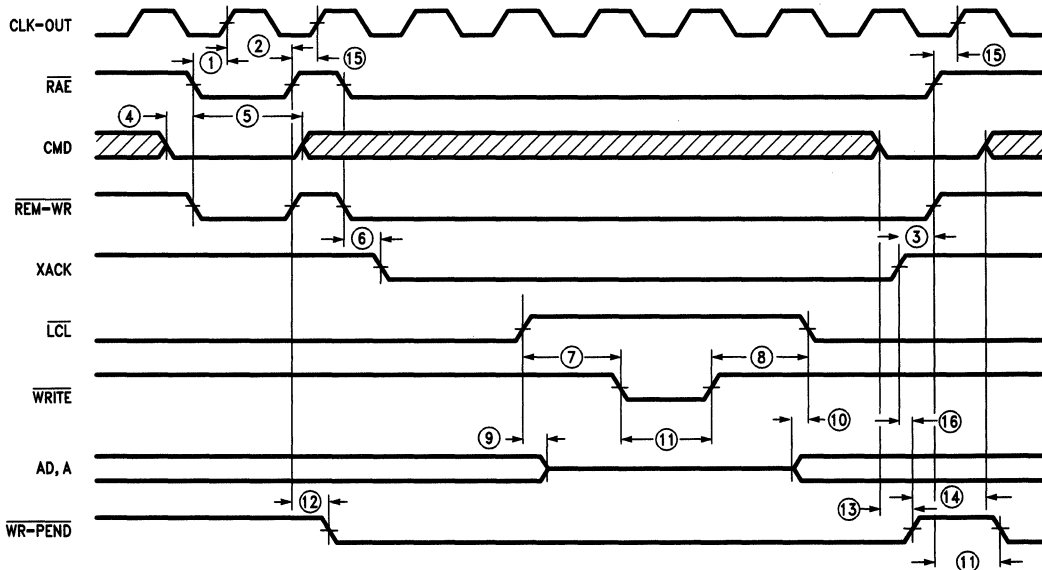
TL/F/9336–78

FIGURE 5-24. Latched Write of DMEM

2

2-181

5.0 Device Specifications (Continued)

TABLE 5-25. Latched Write of IMEM (Notes 1, 2)

Symbol	ID#	Parameter	Formula	Min	Max	Units
$t_{SU-RRW-CO}$	1	$\overline{RAE}$, $\overline{REM-WR}$ Falling before CLK-OUT Rising		27		ns
$t_{H-RRW-CO}$	2	$\overline{RAE}$, $\overline{REM-WR}$ Rising after CLK-OUT Rising (Note 3)	0.5T+	6		ns
			T+		−20	ns
$t_{H-RRW-X}$	3	$\overline{RAE}$, $\overline{REM-WR}$ Rising after XACK Rising		0		ns
$t_{SU-CMD-RRW}$	4	CMD Valid before $\overline{RAE}$, $\overline{REM-WR}$ Falling		0		ns
$t_{H-CMD-RRW}$	5	CMD Invalid after $\overline{RAE}$, $\overline{REM-WR}$ Falling	T+	26		ns
$t_{PD-RRW-X}$	6	$\overline{RAE}$, $\overline{REM-WR}$ Falling to XACK Falling			40	ns
$t_{AZ-LCL-AAD}$	7	A, AD Disabled after $\overline{LCL}$ Rising			27	ns
$t_{ZA-AAD-LCL}$	8	A, AD Enabled before $\overline{LCL}$ Falling			5	ns
$t_{PD-RDAT-I}$	9	AD (Data) Valid to I Valid			44	ns
$t_{H-RDAT-IWR}$	10	AD (Data) Invalid after $\overline{IWR}$ Rising		−4		ns
$t_{PD-RRW-WPND}$	11	$\overline{RAE}$, $\overline{REM-WR}$ Rising to $\overline{WR-PEND}$ Falling			8	ns
			T+		47	ns
$t_{PD-LCL-IA}$	12	$\overline{LCL}$ Falling to Next IA Valid	0.5T+	−25	5	ns
$t_{ZA-IWRH}$	13	$\overline{IWR}$ Falling to I Enabled			0	ns
$t_{AZ-IWR-I}$	14	$\overline{IWR}$ Rising to I Disabled		28	54	ns
$t_{PD-I-IWR}$	15	I Valid before $\overline{IWR}$ Rising	$(n_{IW}+1)$T+	−26		ns
$t_{PD-LCL-IWR}$	16	$\overline{LCL}$ Rising to $\overline{IWR}$ Falling	T+	1		ns
$t_{PD-IWR-LCL}$	17	$\overline{IWR}$ Rising to $\overline{LCL}$ Falling	T+	−29		ns
t_{W-IWR}	18	$\overline{IWR}$ Low Time	$(n_{IW}+1)$T+	−12		ns
$t_{SU-CMD-WPND}$	19	CMD Valid before $\overline{WR-PEND}$ Rising		25		ns
$t_{H-CMD-WPND}$	20	CMD Invalid after $\overline{WR-PEND}$ Rising		0		ns
$t_{PD-I-IA}$	21	I Disabled to IA Invalid	1.5T+	−89		ns
$t_{SU-RRWr-CO}$	22	$\overline{RAE}$, $\overline{REM-WR}$ Rising before CLK-OUT Rising		20		ns
$t_{PD-X-WPND}$	23	XACK Rising to $\overline{WR-PEND}$ Rising			13	ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

Note 2: Two remote writes to instruction memory are necessary to store a 16-bit instruction word to IMEM—low byte followed by high byte. The timing of the 2nd write is shown in the following diagram. The first write is the same as a write of the PC or RIC as shown in *Figure 5-23*.

Note 3: The maximum value for this parameter is the latest $\overline{RAE}$, $\overline{REM-WR}$ can be removed without delaying the remote access by one T-state.

5.0 Device Specifications (Continued)
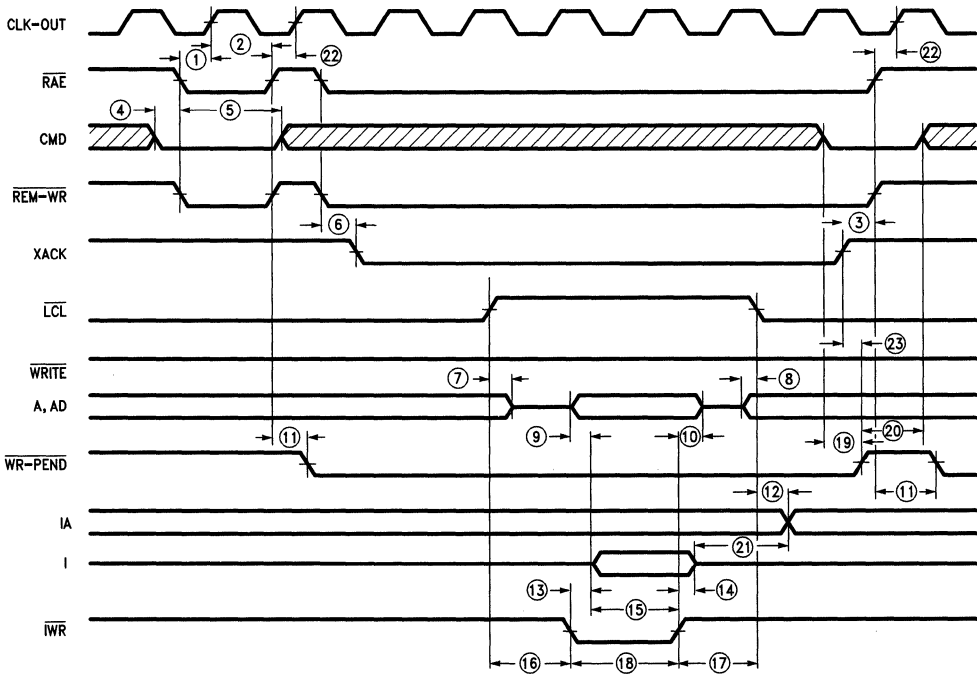
TL/F/9336–79

FIGURE 5-25. Latched Write of IMEM

5.0 Device Specifications (Continued)

TABLE 5-26. Remote Rest Time (Note 1)

Symbol	ID#	Parameter	Formula	Min	Max	Units
$t_{SU-BR-RR-CO}$	1	$\overline{REM-RD}$ Rising before CLK-OUT Rising (Buffered Read Mode)		21		ns
t_{H-BR}	2	CLK-OUT Rising after $\overline{REM-RD}$ Rising to $\overline{REM-RD}$ or $\overline{REM-WR}$ Falling (Buffered Read Mode)	1.5T +	10		ns
$t_{SU-LR-RR-CO}$	3	$\overline{REM-RD}$ Rising before CLK-OUT Rising (Latched Read Mode)		16		ns
t_{H-LR}	4	CLK-OUT Rising after $\overline{REM-RD}$ Rising to $\overline{REM-RD}$ or $\overline{REM-WR}$ Falling (Latched Read Mode)	1.5T +	10		ns
$t_{SU-SBW-RW-CO}$	5	$\overline{REM-WR}$ Rising before CLK-OUT Rising (Slow Buffered Write Mode)		23		ns
t_{H-SBW}	6	CLK-OUT Rising after $\overline{REM-WR}$ Rising to $\overline{REM-RD}$ or $\overline{REM-WR}$ Falling (Slow Buffered Write Mode)	1.5T +	10		ns
$t_{SU-FBW-RW-CO}$	7	$\overline{REM-WR}$ Rising before CLK-OUT Rising (Fast Buffered Write Mode)		23		ns
t_{H-FBW}	8	CLK-OUT Rising after $\overline{REM-WR}$ Rising to $\overline{REM-RD}$ or $\overline{REM-WR}$ Falling (Fast Buffered Write Mode)	1.5T +	10		ns
$t_{SU-LW-RW-CO}$	9	$\overline{REM-WR}$ Rising before CLK-OUT Rising (Latched Write Mode)		20		ns
t_{H-LW}	10	CLK-OUT Rising after $\overline{REM-WR}$ Rising to $\overline{REM-RD}$ or $\overline{REM-WR}$ Falling (Latched Write Mode)		10		ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.
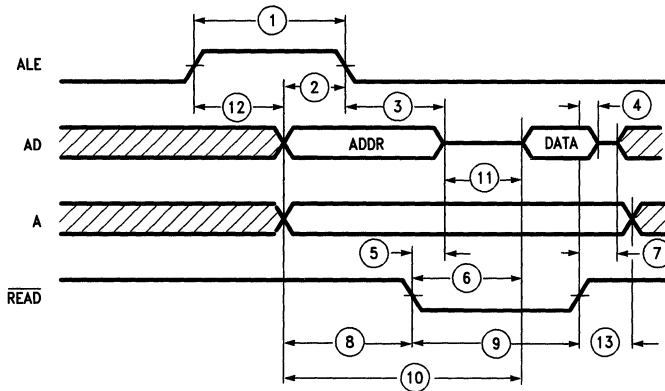
5.0 Device Specifications (Continued)

TL/F/9336–B0

(a) REM-RD Rest Time (Buffered Read Mode)

TL/F/9336–B1

(b) REM-RD Rest Time (Latched Read Mode)

TL/F/9336–B2

(c) REM-WR Rest Time (Slow Buffered Write Mode)

TL/F/9336–B3

(d) REM-WR Rest Time (Fast Buffered Write Mode)

TL/F/9336–B4

(e) REM-WR Rest Time (Latched Write Mode)
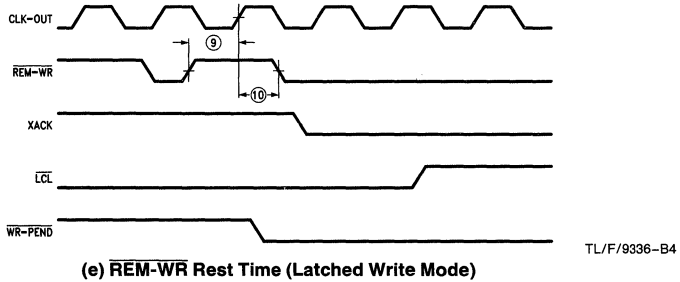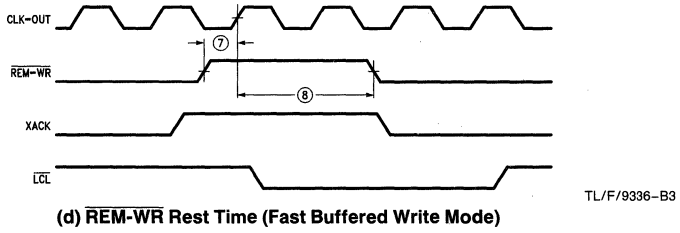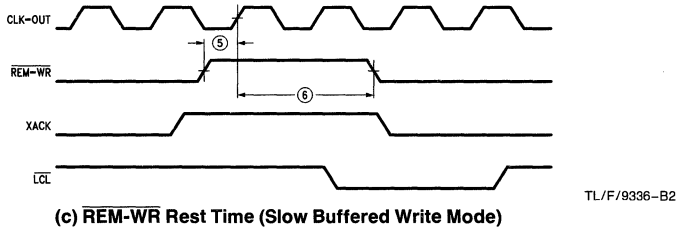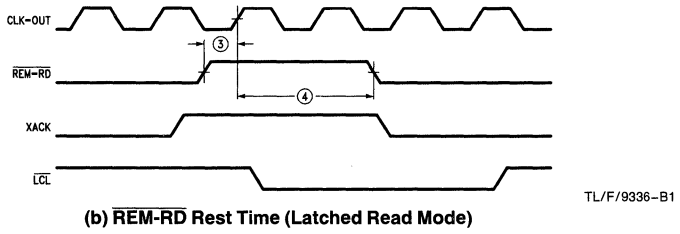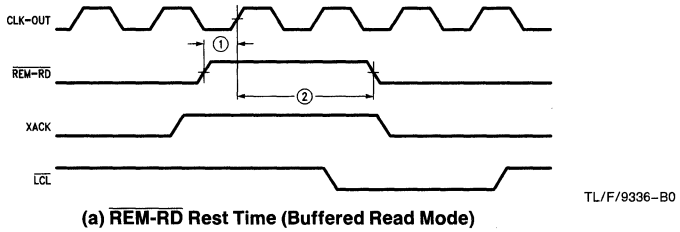
FIGURE 5-26. Remote Rest Time

2

5.0 Device Specifications (Continued)

TABLE 5-27. Remote Interface $\overline{\text{WAIT}}$ Timing (Note 1)

Symbol	ID#	Parameter	Formula	Min	Max	Units
$t_{SU\text{-}WT\text{-}LCL}$	1	$\overline{\text{WAIT}}$ Falling after $\overline{\text{LCL}}$ Rising to Extend Cycle (Buffered Read, Latched Read, Slow Buffered Write, Fast Buffered Write and Latched Write of PC, RIC)	$1.5\text{T}+$		-38	ns
		$\overline{\text{WAIT}}$ Falling after $\overline{\text{LCL}}$ Rising to Extend Cycle (Buffered Read, Latched Read, Slow Buffered Write, Fast Buffered Write and Latched Write of DMEM)	$(n_{DW}+1.5)\text{T}+$		-38	ns
		$\overline{\text{WAIT}}$ Falling after $\overline{\text{LCL}}$ Rising to Extend Cycle (Buffered Read, Latched Read, Slow Buffered Write, Fast Buffered Write and Latched Write of IMEM)	$(n_{IW}+1.5)\text{T}+$		-38	ns
$t_{H\text{-}WT\text{-}LCL}$	2	$\overline{\text{WAIT}}$ Rising after $\overline{\text{LCL}}$ Rising (Buffered Read, Latched Read, Slow Buffered Write, Fast Buffered Write and Latched Write of PC, RIC) (Note 2)	$1.5\text{T}+$	-3		ns
			$2.5\text{T}+$		-37	ns
		$\overline{\text{WAIT}}$ Rising after $\overline{\text{LCL}}$ Rising (Buffered Read, Latched Read, Slow Buffered Write, Fast Buffered Write and Latched Write of DMEM) (Note 2)	$(n_{DW}+1.5)\text{T}+$	-3		ns
			$(n_{DW}+2.5)\text{T}+$		-37	ns
		$\overline{\text{WAIT}}$ Rising after $\overline{\text{LCL}}$ Rising (Buffered Read, Latched Read, Slow Buffered Write, Fast Buffered Write and Latched Write of IMEM) (Note 2)	$(n_{IW}+1.5)\text{T}+$	-3		ns
			$(n_{IW}+2.5)\text{T}+$		-37	ns
$t_{SU\text{-}WT\text{-}RD}$	3	$\overline{\text{WAIT}}$ Falling after $\overline{\text{READ}}$ Falling to Extend Cycle (Buffered Read and Latched Read)	$(n_{DW}+0.5)\text{T}+$		-44	ns
$t_{SU\text{-}WT\text{-}WR}$	3	$\overline{\text{WAIT}}$ Falling after $\overline{\text{WRITE}}$ Falling to Extend Cycle (Slow Buffered Write, Fast Buffered Write and Latched Write)	$(n_{DW}+0.5)\text{T}+$		-49	ns
$t_{SU\text{-}WT\text{-}IWR}$	3	$\overline{\text{WAIT}}$ Falling after $\overline{\text{IWR}}$ Falling to Extend Cycle (Slow Buffered Write, Fast Buffered Write and Latched Write)	$(n_{IW}+0.5)\text{T}+$		-56	ns
$t_{H\text{-}WT\text{-}RD}$	4	$\overline{\text{WAIT}}$ Rising after $\overline{\text{READ}}$ Falling (Buffered Read and Latched Read) (Note 2)	$(n_{DW}+0.5)\text{T}+$	-8		ns
			$(n_{DW}+1.5)\text{T}+$		-43	ns
$t_{H\text{-}WT\text{-}WR}$	4	$\overline{\text{WAIT}}$ Rising after $\overline{\text{WRITE}}$ Falling (Slow Buffered Write, Fast Buffered Write and Latched Write) (Note 2)	$(n_{DW}+0.5)\text{T}+$	-11		ns
			$(n_{DW}+1.5)\text{T}+$		-48	ns
$t_{H\text{-}WT\text{-}IWR}$	4	$\overline{\text{WAIT}}$ Rising after $\overline{\text{IWR}}$ Falling (Slow Buffered Write, Fast Buffered Write and Latched Write) (Note 2)	$(n_{IW}+0.5)\text{T}+$	-12		ns
			$(n_{IW}+1.5)\text{T}+$		-55	ns
$t_{PD\text{-}WT\text{-}X}$	5	$\overline{\text{WAIT}}$ Rising to XACK Rising (Buffered Read, Latched Read, Slow Buffered Write and Fast Buffered Write)	$0.5\text{T}+$	2		ns
			$1.5\text{T}+$		34	ns
$t_{PD\text{-}WT\text{-}LCL}$	6	$\overline{\text{WAIT}}$ Rising to $\overline{\text{LCL}}$ Falling (Latched Write)	$1.5\text{T}+$	4		ns
			$2.5\text{T}+$		33	ns
$t_{PD\text{-}WT\text{-}WR}$	7	$\overline{\text{WAIT}}$ Rising to $\overline{\text{WRITE}}$ Rising (Latched Write)	$0.5\text{T}+$	8		ns
			$1.5\text{T}+$		45	ns
$t_{PD\text{-}WT\text{-}IWR}$	7	$\overline{\text{WAIT}}$ Rising to $\overline{\text{IWR}}$ Rising (Latched Write)	$0.5\text{T}+$	10		ns
			$1.5\text{T}+$		54	ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

Note 2: The maximum value for this parameter is the latest $\overline{\text{WAIT}}$ can be removed without adding an additional T-state. The formula assumes a minimum external wait of one T-state.

5.0 Device Specifications (Continued)
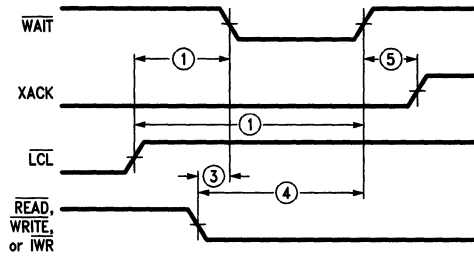
TL/F/9336–B5

(a) Buffered Read, Latched Read, Slow Buffered Write and Fast Buffered Write
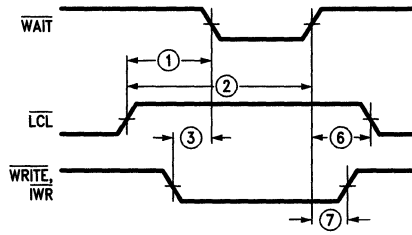
TL/F/9336–B6

(b) Latched Write

FIGURE 5-27. Remote Interface $\overline{\text{WAIT}}$ Timing

6.0 Reference Section

6.1 INSTRUCTION SET REFERENCE

The Instruction Set Reference section contains detailed information on the syntax and operation of each BCP instruction. The instructions are arranged in alphabetical order by mnemonic for easy access. Although this section is primarily intended as a reference for the assembly language programmer, previous assembly language experience is not a prerequisite. The intent of this instruction set reference is to include all the pertinent information regarding each instruction on the page(s) describing that instruction. The only exceptions to this rule concern the instruction addressing modes and the bus timing diagrams. The discussion of the instruction addressing modes occurs at the beginning of the BCP Instruction Set Overview section and, therefore, will not be repeated here. The figures for the bus timing diagrams are located at the end of this introduction rather than constantly repeating them under each instruction. The information that is contained under each instruction is divided into eight categories titled: Syntax, Affected Flags, Description, Example, Instruction Format, T-states, Bus timing, and Operation. The following paragraphs explain what information each category conveys and any special nomenclature that a category may use.

Syntax

This category illustrates the assembler syntax for each instruction. Multiple lines are used when a given instruction supports more than one type of addressing mode, or if it has an optional mnemonic. All capital letters, commas (,), math symbols ($+$, $-$), and brackets ([ ]) are entered into the assembler exactly as shown. Braces ({ }) surround an instruction's optional operands and their associated syntax. The text between the braces may either be entered in with or omitted from the instruction. The braces themselves should not be entered into the assembler because they are not part of the assembler syntax. Lower case characters and operands that begin with the capital R represent symbols. These must be replaced with actual register names, numbers, or equated registers and numbers. Table 6-1 lists all the symbols and their associated meanings.

Affected Flags

If an instruction sets or clears any of the ALU flags, (i.e., Negative [N], Zero [Z], Carry [C], and/or Overflow [V]), then those flags affected are listed under this category.

Description

The Description category contains a verbal discussion about the operation of an instruction, the operands it allows, and any notes highlighting special considerations the prorammer should keep in mind when using the instruction.

Example

Each instruction has one or more coding examples designed to show its typical usage(s). For clarity, register name abbreviations are often used instead of the register numbers, (i.e., RTR is used in place of R4). Each example assumes that the ".EQU" assembler directive has been previously executed to establish these relationships. Information relating register abbreviations to register names, numbers, and purpose is located in the CPU Registers section.

Instruction Format

This category illustrates the formation of an instruction's machine code for each operand variation. Assembly or disassembly of any instruction can be accomplished using these figures.

T-states

The T-state category lists the number of CPU clock cycles required for each instruction, including operand variations and conditional considerations. Using this information, actual execution times may be calculated. For example, if the conditional relative jump instruction's condition is not met, the CPU's clock cycle is 18.867 MHz ([CCS]=0), and no instruction wait states are requested ([IW1−0]=00), then Jcc's execution time is calculated as shown below:

$$t_{execution} = 1/(\text{CPU clock frequency}) \times \text{T-states}$$
$$= 1/(18.867 \times 10^6 \text{ Hz}) \times 2$$
$$= (53 \times 10^{-9}\text{s}) \times 2$$
$$= 106 \text{ ns}$$

See the section BCP Timing for more information on calculating instruction execution times.

Bus Timing

This category refers the user to the Bus Timing *Figures 6-1* to *6-6* on the following pages. These figures illustrate the relationship between software instruction execution and some of the BCP's hardware signals.

Operation

The operation category illustrates each instruction's operation in a symbolic coding format. Most of the operand names used in this format come directly from each instruction's syntax. The exceptions to this rule deal with implied operands. Instructions that imply the use of the accumulators use the name "accumulator" as an operand. Instructions that manipulate the Program Counter use the symbol "PC". Instructions that "push" onto or "pop" off of the internal Address Stack specify "Address Stack" as an operand. Instructions that save or restore the ALU flags and the register bank selections use those terms as operands. Two specialized operator symbols are used in the symbolic coding format, the arrow " → " and the concatenation operator "&". The arrow indicates the movement of data from one operand to another. For instance, after the operation "Rs → Rd" is performed the content of Rd has been replaced with the content of Rs. The concatenation operator "&" simply indicates that the operands surrounding an "&" are attached together forming one new operand. For example, "PC & [GIE] & ALU flags & register bank selections → Address Stack" means that the Program Counter, the Global Interrupt Enable bit, the ALU flags and the register bank selections are combined into one operand and pushed onto the internal Address Stack. Three conditional structures are utilized in the symbolic coding format: the "Two Line If" structure, the "Blocked If" structure, and the "Blocked Case" structure. In the "Two Line If" structure, if the *condition* is met then the *operation* is performed, otherwise the *operation* is not performed.

"Two Line If" structure:

    If *condition*

       then *operation*

6.0 Reference Section (Continued)

In the "Blocked If" structure, if the *condition* is met then all the *operations* between the "If" statement and the "End if" statement are performed.

"Blocked If" structure:

> If *condition* then
> > *operation*
> > *operation*
> > *etc* . . .
>
> End if

In the "Blocked Case" structure, the *operation* preceded by the equivalent numeric value of the *operand* is executed. For example, if the *operand*'s value is equal to "1" then the *operation* preceded by "1:" is executed.

"Blocked Case" structure:

> Case *operand* of
> > 0: *operation*
> > 1: *operation*
> > 2: *etc* . . .
>
> End case

Two reference tables have been added to the back of the Instruction Set Reference section. The first table, Table 6-2, lists all the instructions with their associated T-states, Affected Flags, and Bus Timing figure numbers in a compact format. The second table, Table 6-3, lists all the instructions in opcode order to facilitate disassembly.

TABLE 6-1. Notational Conventions for Instruction Set

Symbol	Represents	Meaning	Length
n	0 to 255 +127 to −128	Unsigned Number Signed Number	8 Bits
nn	0 to 65535	Unsigned Number	16 Bits
Rs	R0–R31	Source Register	
Rd	R0–R31	Destination Register	
Rsd	R0–R31	Combination Source/Destination Register	
rs	R0–R15	Limited Source Register	
rd	R0–R15	Limited Destination Register	
rsd	R0–R15	Limited Combination Source/Destination Register	
lr	IW, IX, IY, IZ	Index Register	
mlr		Index Register in One of the Following Address Modes:	
	lr−	Post Decrement	
	lr	No Change	
	lr+	Post Increment	
	+lr	Pre-Increment	
b	0–7	Shift Field	3 Bits
m	0–7	Mask Field	3 Bits
p	0–7	Position Field	3 Bits
s	0–1	State Field	1 Bit
f	0–7	Flag Reference Field	3 Bits
cc		Condition Code Instruction Extensions	
v	0–63	Vector Field	6 Bits
g	0–3	Global Interrupt Enable Flag [GIE] Status Control	2 Bits
g′	0–1	Global Interrupt Enable Flag [GIE] Limited Status Control	1 Bit
rf	0–1	Register Bank and ALU Flag Status Control	1 Bit
ba	0–1	Register Bank A Select	1 Bit
bb	0–1	Register Bank B Select	1 Bit

2

6.0 Reference Section (Continued)
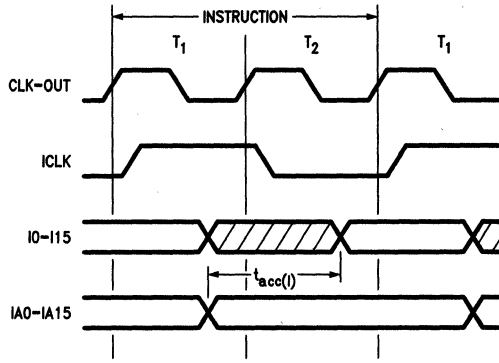
TL/F/9336–21

**FIGURE 6-1. Instruction-Memory Bus Timing for 2 T-state Instructions
(No Instruction Wait States [IW1–0] = 00, CPU Running at Full Speed [CCS] = 0)**
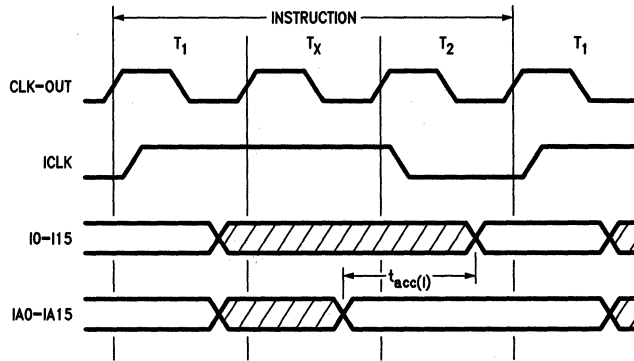
TL/F/9336–22

**FIGURE 6-2. Instruction-Memory Bus Timing for 3 T-state Instructions
(No Instruction Wait States [IW1–0] = 00, CPU Running at Full Speed [CCS] = 0)**

6.0 Reference Section (Continued)
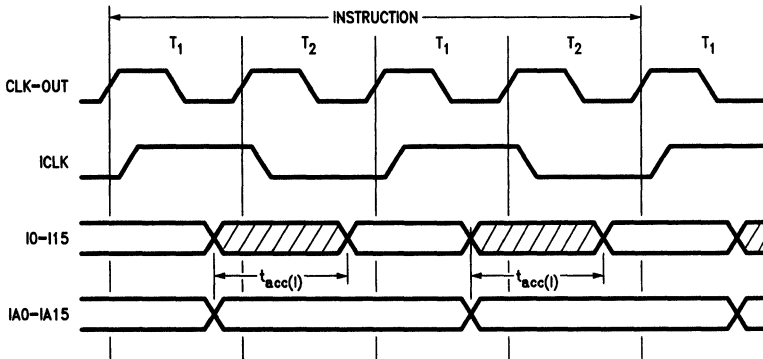
TL/F/9336–23

FIGURE 6-3. Instruction-Memory Bus Timing for (2 + 2) T-state Instructions
(No Instruction Wait States [IW1–0] = 00, CPU Running at Full Speed [CCS] = 0)
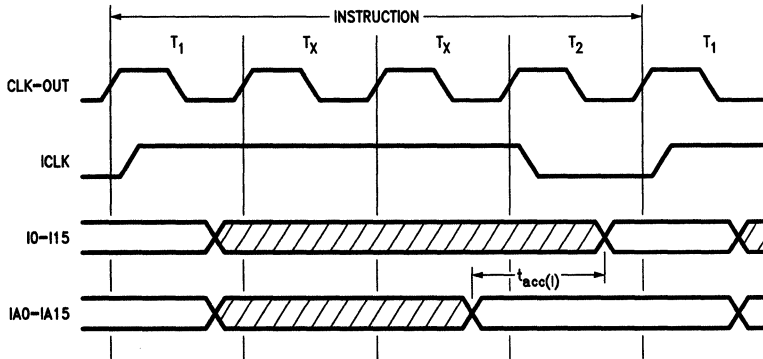
TL/F/9336–24

FIGURE 6-4. Instruction-Memory Bus Timing for 4 T-state Instructions
(No Instruction Wait States [IW1–0] = 00, CPU Running at Full Speed [CCS] = 0)

2

6.0 Reference Section (Continued)
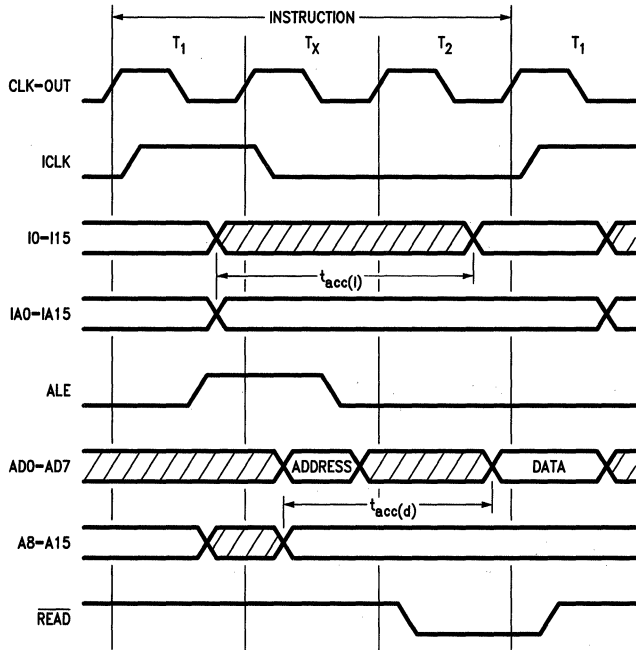
TL/F/9336–25

FIGURE 6-5. Instruction/Data Memory Bus Timing for Data Memory Read
(No Instruction or Data Memory Wait States, CPU Running at Full Speed [CCS] = 0)
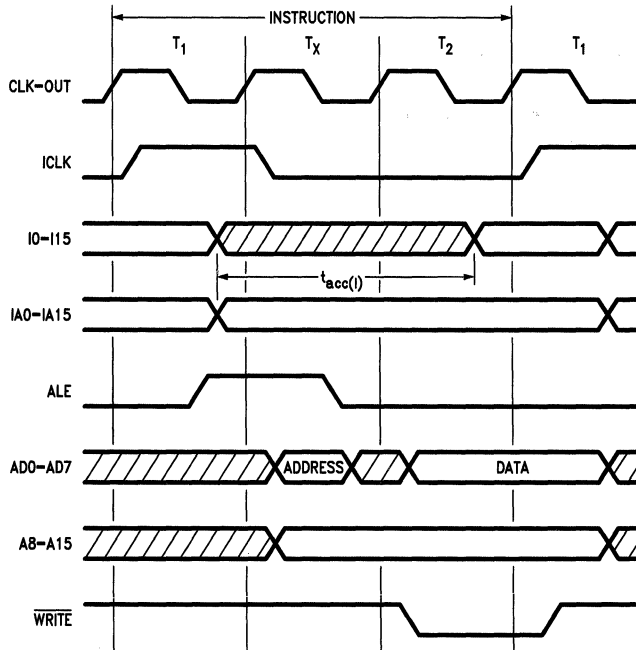
TL/F/9336–26

FIGURE 6-6. Instruction/Data Memory Bus Timing for Data Memory Write
(No Instruction or Data Memory Wait States, CPU Running at Full Speed [CCS] = 0)

6.0 Reference Section (Continued)

ADCA   Add with Carry and Accumulator

Syntax
ADCA   Rs, Rd   —register, register
ADCA   Rs, [mlr]   —register, indexed

Affected Flags
N, Z, C, V

Description
Adds the source register Rs, the active accumulator, and the carry flag together, placing the result into the destination specified. The destination may be either a register, Rd, or data memory via an index register mode, [mlr]. Note that register bank selection determines which accumulator is active.

Example
Add the constant 109 to the index register IW, (which is 16 bits wide).

```
SUBA  A, A         ;Clear the accumulator
ADD   109, R12     ;Add 109 to low byte of IW
ADCA  R13, R13     ;Add carry to high byte of IW
```

Instruction Format
ADCA   Rs, Rd
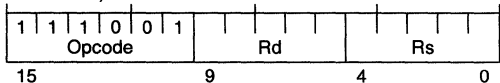
1	1	1	1	0	0	1									
	Opcode							Rd				Rs			
15					9				4						0

ADCA   Rs, [mlr]

1	0	1	1	0	0	0	1								
		Opcode						m	lr		Rs				
15							8		6	4					0

```
00 - post-decrement      00 - IW
01 - no change           01 - IX
10 - post increment      10 - IY
11 - pre-increment       11 - IZ
```
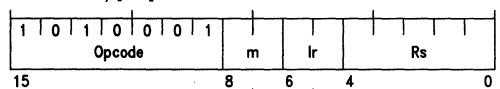
TL/F/9336–5

T-states
ADCA   Rs, Rd   —2
ADCA   Rs, [mlr]   —3

Bus Timing
ADCA   Rs, Rd   —Figure 6-1
ADCA   Rs, [mlr]   —Figure 6-6

Operation
ADCA   Rs, Rd
Rs + accumulator + carry bit → Rd

ADCA   Rs, [mlr]
Rs + accumulator + carry bit → data memory

ADD   Add Immediate

Syntax
ADD   n, rsd   —immediate, limited register

Affected Flags
N, Z, C, V

Description
Adds the immediate value n to the register rsd and places the result back into the register rsd. Note that only the active registers R0–R15 may be specified for rsd. The value of n is limited to 8 bits; (unsigned range: 0 to 255, signed range: $+127$ to -128).

Example
Add the constant -3 to register 10.

```
ADD    -3, R10        ;R10 + (-3) → R10
```

Instruction Format

0	0	0	0	0											
	Opcode							n					rsd		
15				11								3			0
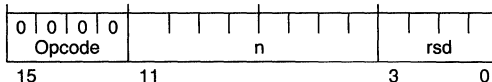
T-States
2

Bus Timing
Figure 6-1

Operation
rsd + n → rsd

6.0 Reference Section (Continued)

ADDA   Add with Accumulator

Syntax

ADDA   Rs, Rd   —register, register
ADDA   Rs, [mlr]   —register, indexed

Affected Flags

N, Z, C, V

Description

Adds the source register Rs to the active accumulator and places the result into the destination specified. The destination may be either a register, Rd, or data memory via an index register mode, [mlr]. Note that register bank selection determines which accumulator is active.

Example

In the first example, the value 4 is placed into the currently active accumulator, that accumulator is added to the contents of register 20, and then the result is placed into register 21.

```
MOVE   4, A        ;Place constant into accum
ADDA   R20, R21    ;R20 + accum → R21
```

In the second example, the alternate accumulator of register bank B is selected and then added to register 20. The result is placed into the data memory pointed to by the index register IZ and then the value of IZ is incremented by one.

```
EXX    0, 1        ;Select alt accumulator
ADDA   R20, [IZ+]  ;R20 + accum → data mem
                   ;and increment data pointer
```

Instruction Format

ADDA   Rs, Rd
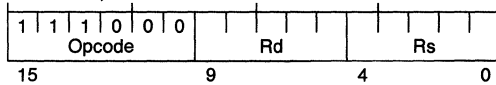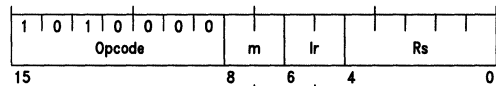
1	1	1	1	0	0	0									
		Opcode							Rd				Rs		

15   9   4   0

ADDA   Rs, [mlr]

1	0	1	1	0	0	0	0								
		Opcode						m	lr			Rs			

15   8   6   4   0

```
00 - post-decrement      00 - IW
01 - no change           01 - IX
10 - post increment      10 - IY
11 - pre-increment       11 - IZ
```

TL/F/9336-6

T-states

ADDA   Rs, Rd   —2
ADDA   Rs, [mlr]   —3

Bus Timing

ADDA   Rs, Rd   —Figure 6-1
ADDA   Rs, [mlr]   —Figure 6-6

Operation

ADDA   Rs, Rd
Rs + accumulator → Rd

ADDA   Rs, [mlr]
Rs + accumulator → data memory

AND   And Immediate

Syntax

AND   n, rsd   —immediate, limited register

Affected Flags

N, Z

Description

Logically ANDs the immediate value n to the register rsd and places the result back into the register rsd. Note that only the active registers R0–R15 may be specified for rsd. The value of n is 8 bits wide.

Example

Unmask both the Transmitter and Receiver interrupts via the Interrupt Control Register {ICR}, R2. Leave the other interrupts unaffected.

```
EXX   0,0            ;select main register banks
AND   11111100B,R2   ;unmask transmitter and
                     ; receiver interrupts
```

Instruction Format

0	1	1	0	0											
	Opcode							n					rsd		

15   11   3   0

T-states

2

Bus Timing

Figure 6-1

Operation

rsd   AND   n → rsd

6.0 Reference Section (Continued)

ANDA   And with Accumulator

Syntax

ANDA   Rs, Rd   —register, register
ANDA   Rs, [mlr]   —register, indexed

Affected Flags

N, Z

Description

Logically ANDs the source register Rs to the active accumulator and places the result into the destination specified. The destination may be either a register, Rd, or data memory via an index register mode, [mlr]. Note that register bank selection determines which accumulator is active.

Example

This example demonstrates a way to quickly unload all 11 bits of the three words in the Receiver FIFO when the FIFO is full. The example assumes that the index register IZ points to the location in data memory where the information should be stored.

```
        EXX    1,1          ;select alternate banks
        MOVE   00000111B, A ;place the {TSR} mask
                            ; into the accumulator
;  Pop the first word from the receiver FIFO
        ANDA   TSR, [IZ+]   ;read bits 8, 9, & 10
        MOVE   RTR, [IZ+]   ;pop bits 0-7
;  Pop the second word from the receiver FIFO
        ANDA   TSR, [IZ+]
        MOVE   RTR, [IZ+]
;  Pop the third word from the receiver FIFO
        ANDA   TSR, [IZ+]
        MOVE   RTR, [IZ+]
```

Instruction Format

ANDA   Rs, Rd
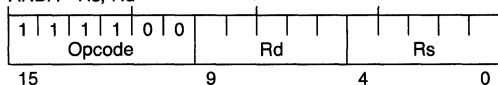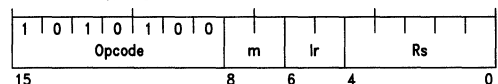
1	1	1	1	1	0	0								
		Opcode						Rd				Rs		

15   9   4   0

ANDA   Rs,[mlr]

1	0	1	0	1	1	0	0						
			Opcode					m	lr			Rs	

15   8   6   4   0

```
00 - post-decrement      00 - IW
01 - no change           01 - IX
10 - post increment      10 - IY
11 - pre-increment       11 - IZ
```

TL/F/9336-7

T-states

ANDA   Rs, Rd   —2
ANDA   Rs, [mlr]   —3

Bus Timing

ANDA   Rs, Rd   —*Figure 6-1*
ANDA   Rs, [mlr]   —*Figure 6-6*

Operation

ANDA   Rs, Rd
Rs   AND accumulator → Rd

ANDA   Rs, [mlr]
Rs   AND accumulator → data memory

BIT   Bit Test

Syntax

BIT   rs, n   —limited register, immediate

Affected Flags

N, Z

Description

Performs a bit level test by logically ANDing the source register rs to the immediate value n. The affected flags are updated, but the result is not saved. Note that only the active registers R0–R15 may be specified for rs. The value n is 8 bits wide.

Example

Poll the Transmitter FIFO Empty flag [TFE] in the Network Command Flag register {NCF}, R1, waiting for the Transmitter to send the current FIFO data.

```
        EXX    0,1                  ;select main A, alt B
Poll:   BIT    NCF,10000000B        ;All data sent yet?
        JZ     Poll                 ; No, poll TFE
        . . .                       ; Yes, send next byte(s)
```

Instruction Format

0	1	1	1	1										
		Opcode						n					rs	

15   11   3   0

T-states

2
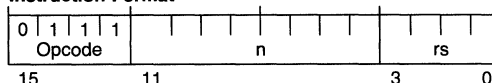
Bus Timing

Figure 6-1

Operation

rs   AND   n

6.0 Reference Section (Continued)

CALL   Unconditional Relative Call

Syntax

CALL   n   —immediate

Affected Flags

None

Description

Pushes the Program Counter, the ALU flags, the Global Interrupt Enable bit [GIE], and the current register bank selections onto the internal Address Stack; then unconditionally transfers control to the instruction at the memory address calculated by adding the contents of the Program Counter to the immediate value n, (sign extended to 16 bits). Since the immediate value n is an 8-bit two's complement displacement, the unconditional relative call's range is from +127 to −128 relative to the Program Counter. Note that the Program Counter initially contains the memory address of the next instruction following the call.

Example

Transfer control to the subroutine "Send.it". Note that "Send.it" must be within +127/−128 words relative to the PC.

    CALL   Send.it

Instruction Format

1	1	0	0	1	1	0	0								
Opcode								n							

15   7   0

T-states

3
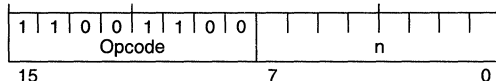
Bus Timing

Figure 6-2

Operation

PC & [GIE] & ALU flags & register bank selections
  —→ Address Stack
PC + n(sign extended) —→ PC

CMP   Compare

Syntax

CMP   rs, n   —limited register, immediate

Affected Flags

N, Z, C, V

Description

Compares the immediate value n with the source register rs by subtracting n from rs. The affected flags are updated, but the result is not saved. Note that only the active registers R0–R15 may be specified for rs. The value of n is limited to 8 bits; (unsigned range: 0 to 255, signed range: +127 to −128).

Example

Compare the data byte in register 11 to the ASCII character "A".

    CMP   R11,"A"   ;If:
    JC   Less_than_A   ;   data<"A"
    JEQ   Equal_to_A   ;   data="A"
    . . .   ;else data>"A"

Compare the contents of register 8 to the value 25.

    CMP   R8,25   ;if:
    BIT   CCR,00000011B   ;   data > 25
    JZ   Greater_than   ;   Goto Greater_than

Comparing of Unsigned Values	
Comparison	**Flag(s) to Test**
LT (<)	C
LEQ (< =)	C|Z
EQ (=)	Z
GEQ (> =)	$\overline{C}$
GT (>)	$\overline{C}\,\&\,\overline{Z}$

Note: & = logical AND
    | = logical OR

Instruction Format

0	0	1	1	1											
Opcode					n								rs		

15   11   3   0

T-states

2
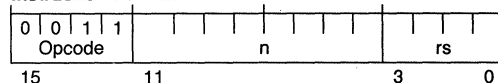
Bus Timing

Figure 6-1

Operation

rs − n

6.0 Reference Section (Continued)

CPL   Complement

Syntax

CPL   Rsd   —register

Affected Flags

N, Z

Description

Logically complements the contents of the register Rsd, placing the result back into that register.

Example

Load the fill-bit count passed from the host into the Transmitter's Fill-Bit Register {FBR}, R3, and then perform the required one's complement of the fill-bit count. In this example, register 20 contains the fill-bit count.

```
EXX     1,1        ;select alternate banks
MOVE    R20, FBR   ;load {FBR}
CPL     FBR        ;complement fill-bit count
```

Instruction Format

1	0	1	0	1	1	1	1	0	0	0	0				
			Opcode										Rsd		

15   4   0

T-states

2
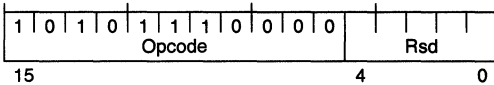
Bus Timing

Figure 6-1

Operation

$\overline{Rsd}$ → Rsd

EXX   Exchange Register Banks

Syntax

EXX   ba, bb {,g}

Affected Flags

None

Description

Selects which CPU register banks are active by exchanging between the main and alternate register sets for each bank. Bank A controls R0–R3 and Bank B controls R4–R11. The table below shows the four possible register bank configurations. Note that deactivated registers retain their current values. The Global Interrupt Enable bit [GIE] can be set or cleared, if desired.

Register Bank Configurations

ba	bb	Active Register Banks
0	0	Main A, Main B
0	1	Main A, Alternate B
1	0	Alternate A, Main B
1	1	Alternate A, Alternate B

Example

Activate the main register set of Bank A, the alternate register set of Bank B, and leave the Global Interrupt Enable bit [GIE] unchanged.

```
EXX     0,1        ;select main A, alt B reg banks
```

Instruction Format

1	0	1	0	1	1	1	1	0	1		g	ba	bb	0	0	0
			Opcode													

15   6   4   3   2   0

00—GIE not affected
01—reserved
10—Set GIE
11—Clear GIE

T-states

2
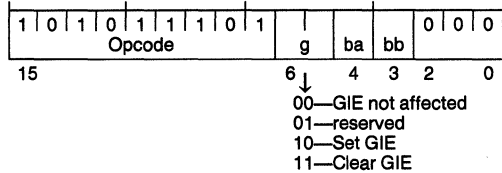
Bus Timing

Figure 6-1

Operation

```
Case ba of
   0:  activate main Bank A
   1:  activate alternate Bank A
End case
Case bb of
   0:  activate main Bank B
   1:  activate alternate Bank B
End case
Case g of
   0:  leave [GIE] unaffected, (default)
   1:  (reserved)
   2:  set [GIE]
   3:  clear [GIE]
End case
```

2

6.0 Reference Section (Continued)

JMP   Conditional Relative Jump

Jcc

Syntax

JMP   f, s, n   —immediate
Jcc   n   —immediate (optional syntax)

Affected Flags
None

Description

Conditionally transfers control to the instruction at the memory address calculated by adding the contents of the Program Counter to the immediate value n, (sign extended to 16 bits), if the state of the flag referenced by f is equal to the state of the bit s; or, optionally, if the condition cc is met. See the tables below for the flags that f can reference and the conditions that cc may specify. Since the immediate value n is an 8-bit two's complement displacement, the conditional relative jump's range is from +127 to −128 relative to the Program Counter. Note that the Program Counter initially contains the memory address of the next instruction following the jump.

Example

This example demonstrates both syntaxes of the conditional relative jump instruction testing for a non-zero result from a previous instruction; (i.e., [Z]=0). If the condition is met then control transfers to the instruction labeled "Loop.back"; else the next instruction following the jump is executed.

```
      JMP    000B,0,Loop.back      ;jump on not zero
      . . .
      JNZ    Loop.back             ;jump on not zero
```

Condition Specification Table for "cc"

cc	Meaning	Condition Tested for
Z	Zero	[Z] = 1
NZ	Not Zero	[Z] = 0
EQ	Equal	[Z] = 1
NEQ	Not Equal	[Z] = 0
C	Carry	[C] = 1
NC	No Carry	[C] = 0
V	Overflow	[V] = 1
NV	No Overflow	[V] = 0
N	Negative	[N] = 1
P	Positive	[N] = 0
RA	Receiver Active	[RA] = 1
NRA	Not Receiver Active	[RA] = 0
RE	Receiver Error	[RE] = 1
NRE	No Receiver Error	[RE] = 0
DA	Data Available	[DAV] = 1
NDA	No Data Available	[DAV] = 0
TFF	Transmitter FIFO Full	[TFF] = 1
NTFF	Transmitter FIFO Not Full	[TFF] = 0

Instruction Format
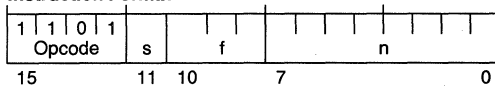
1	1	0	1	1	Opcode	s	f	n
15					11	10	7	0

T-states
2 if condition is not met
3 if condition is met

Bus Timing
Figure 6-1 if condition is not met
Figure 6-2 if condition is met

Operation

JMP   f, s, n
If flag f is in state s
    then PC + n(sign extended) → PC

Jcc   n
If cc condition is true
    then PC + n(sign extended) → PC

Flag Reference Table for "f"

f	(binary)	Flag Reference	
0	(000)	[Z]	in {CCR}
1	(001)	[C]	in {CCR}
2	(010)	[V]	in {CCR}
3	(011)	[N]	in {CCR}
4	(100)	[RA]	in {TSR}
5	(101)	[RE]	in {TSR}
6*	(110)	[DAV]	in {TSR}
7	(111)	[TFF]	in {TSR}

*Note: The value of f for [DAV] differs from the numeric value for the position of [DAV] in {TSR}.

6.0 Reference Section (Continued)

JMP   Unconditional Relative Jump

Syntax

JMP   n   —immediate
JMP   Rs   —register

Affected Flags
None

Description

Unconditionally transfers control to the instruction at the memory address calculated by adding the contents of the Program Counter to either the immediate value n or the contents of the source register Rs, (both sign extended to 16 bits). Since the immediate value n and the contents of Rs are 8-bit two's complement displacements, the unconditional relative jump's range is from +127 to −128 relative to the Program Counter. Note that the Program Counter initially contains the memory address of the next instruction following the jump.

Example

Transfer control to the instruction labeled "Init_Xmit", which is within +127/−128 words relative to the PC.

    JMP   Init_Xmit   ;go initialize Transmitter

Instruction Format
JMP   n

1	1	0	0	1	0	1	1								
		Opcode									n				

15   7   0

JMP   Rs

1	1	0	0	1	1	0	1	1	0	0					
			Opcode									Rs			

15   4   0
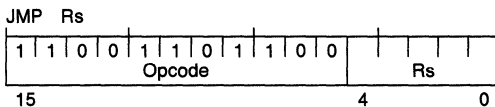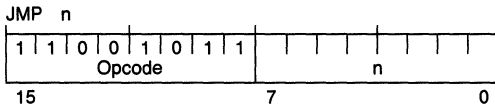
T-states

JMP   n   —3
JMP   Rs   —4

Bus Timing

JMP   n   —*Figure 6-2*
JMP   Rs   —*Figure 6-4*

Operation

JMP   n
PC + n(sign extended) → PC

JMP   Rs
PC + Rs(sign extended) → PC

2

6.0 Reference Section (Continued)

JRMK   Relative Jump with Rotate and Mask on Register

Syntax

JRMK   Rs, b, m   —register

Affected Flags

None

Description

Transfers control to the instruction at the memory address calculated by adding the contents of the Program Counter to a specially formed displacement. The displacement is formed by rotating a copy of the source register Rs the value of b bits to the right, masking (setting to zero) the most significant m bits, masking the least significant bit, and then sign extending the result to 16 bits. Typically, the JRMK instruction transfers control into a jump table. The LSB of the displacement is always set to zero so that the jump table may contain two word instructions, (e.g., LJMP). The range of JRMK is from +126 to −128 relative to the Program Counter. Note that the Program Counter initially contains the memory address of the next instruction following JRMK. The source register Rs may specify any active CPU register. The rotate value b may be from 0 to 7, where 0 causes no bit rotation to occur. The mask value m may be from 0 to 7; where m=0 causes only the LSB of the displacement to be masked, m=1 causes the MSB and the LSB to be masked, m=2 causes bits 7–6 and the LSB to be masked, etc …

Example

This example demonstrates the decoding of the address frame of the 3299 Terminal Multiplexer protocol. In the address frame, only the bits 4–2 contain the address of the Logical Unit.

```
EXX     0,1        ;select main A, alt B
JRMK    RTR,1,4    ;decode device address
LJMP    ADDR.0     ;jump to device handler #0
LJMP    ADDR.1     ;jump to device handler #1
LJMP    ADDR.2     ;jump to device handler #2
 . . .
LJMP    ADDR.7     ;jump to device handler #7
```

Instruction Format

1	0	0	0	0											
	Opcode					m			b				Rs		

15   10   7   4   0
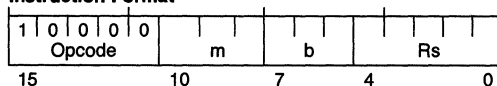
T-states

4

Bus Timing

Figure 6-4

Operation

Copy Rs to a temporary register:
  Rs → register

Rotate the register b bits to the right:
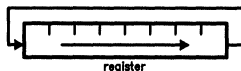
register

TL/F/9336–8

Mask the most significant m bits and the LSB:

m

register AND 0 ... 0 1 ... 1 0 → register

Modify the Program Counter:
  PC + register(sign extended) → PC

6.0 Reference Section (Continued)

LCALL Conditional Long Call

Syntax

LCALL   Rs, p, s, nn   —register, absolute

Affected Flags

None

Description

If the bit in position p of register Rs is equal to the bit s, then push the Program Counter, the ALU flags, the Global Interrupt Enable bit [GIE], and the current register bank selections onto the internal Address Stack. Following the push, transfer control to the instruction at the absolute memory address nn. The operand Rs may specify any active CPU register. The value of p may be from 0 to 7, where 0 corresponds to the LSB of Rs and 7 corresponds to the MSB of Rs. The absolute value nn is 16 bits long, (range: 0 to 64k), therefore, all of instruction memory can be addressed.

Example

Call the "Load.Xmit" subroutine when the Transmitter FIFO Empty flag, [TFE], of the Network Command Flag register {NCF} is "1".

```
EXX    0,0                  ;select main A, alt B
LCALL  NCF,7,1, Load.Xmit   ;If [TFE] = 1 call
```

Instruction Format

1	0	0	0	0	1	1	1		s		p			Rs	
			Opcode												

15   8 7   4   0

							nn								

15   0

T-states

(2 + 2)
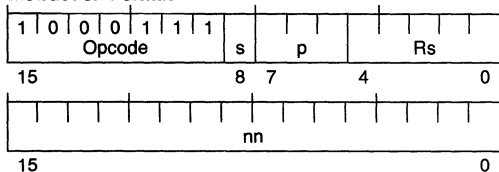
Bus Timing

Figure 6-3

Operation

If Rs[p] = s then
   PC & [GIE] & ALU flags & register bank selections
       → Address Stack
   nn → PC
End if

LCALL Unconditional Long Call

Syntax

LCALL   nn   —absolute

Affected Flags

None

Description

Pushes the Program Counter, the ALU flags, the Global Interrupt Enable bit [GIE], and the current register bank selections onto the internal Address Stack; then unconditionally transfers control to the instruction at the absolute memory address nn. The value of nn is 16 bits long, (range: 0 to 64k), therefore, all of instruction memory can be addressed.

Example

Transfer control to the subroutine "Send.it.all", which could be located anywhere in instruction memory.

```
LCALL    Send.it.all
```

Instruction Format

1	1	0	0	1	1	1	0	1	0	0	0	0	0	0	0
							Opcode								

15   0

							nn								

15   0

T-states

(2 + 2)
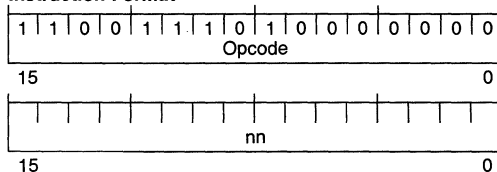
Bus Timing

Figure 6-3

Operation

PC & [GIE] & ALU flags & register bank selections
    → Address Stack
nn → PC

2

6.0 Reference Section (Continued)

LJMP   Conditional Long Jump

Syntax

LJMP   Rs, p, s, nn   —register, absolute

Affected Flags

None

Description

Conditionally transfers control to the instruction at the absolute memory address nn if the bit in position p of register Rs is equal to the state of the bit s. The operand Rs may specify any active CPU register. The value of p may be from 0 to 7, where 0 corresponds to the LSB of Rs and 7 corresponds to the MSB of Rs. The absolute value nn is 16 bits long, (range: 0 to 64k), therefore, all of instruction memory can be addressed.

Example

Long Jump to one of the receiver error handling routines based on the contents of the Error Code Register {ECR}.

```
    EXX     0,1,3           ;select main A, alt B
                            ; and clear [GIE]
    OR      01000000B,TSR   ;set [SEC] in {TSR}
    MOVE    ECR, R11        ;read {ECR}
; Determine error condition
    LJMP    R11, 0, 1, Software_error
    LJMP    R11, 1, 1, Loss_of_Midbit
    LJMP    R11, 2, 1, Invalid_Ending_Seq
    LJMP    R11, 3, 1, Parity_error
    LJMP    R11, 4, 1, Software_error
```

Instruction Format

1	0	0	0	0	1	1	0		s		p			Rs	
		Opcode													
15							8	7			4				0

							nn								
15															0

T-states

(2 + 2)
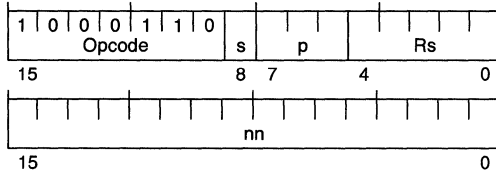
Bus Timing

Figure 6-3

Operation

If Rs[p] = s
  then nn → PC

LJMP   Unconditional Long Jump

Syntax

LJMP   nn   —absolute
LJMP   [Ir]   —indexed

Affected Flags

None

Description

Unconditionally transfers control to the instruction at the memory address specified by the operand. The operand may either specify an absolute instruction address nn, (16 bits long), or an index register Ir, which contains an instruction address. Long Jump's addressing range is from 0 to 64k; (i.e., all of instruction memory can be addressed).

Example

Transfer control to the instruction labeled "Reset.System", which may be located anywhere in instruction memory.

```
    LJMP    Reset.System    ;go reset the system
```

Instruction Format

LJMP   nn

1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0
				Opcode											
15															0

							nn								
15															0

LJMP   [Ir]

1	1	0	0	1	1	0	1		Ir		0	0	0	0	0
			Opcode												
15								6		4					0

```
00—IW
01—IX
10—IY
11—IZ
```
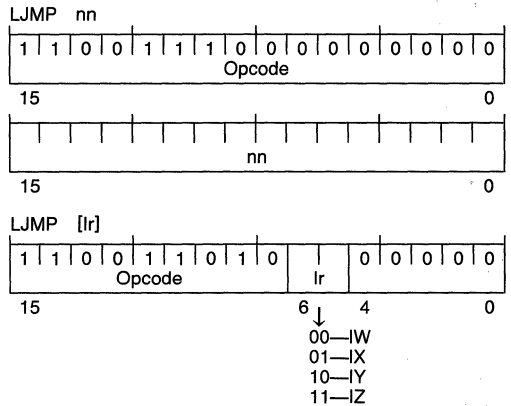
T-states

LJMP   nn   —(2 + 2)
LJMP   [Ir]   —2

Bus Timing

LJMP   nn   —*Figure 6-3*
LJMP   [Ir]   —*Figure 6-1*

Operation

LJMP   nn
nn → PC
LJMP   [Ir]
Ir → PC

6.0 Reference Section (Continued)

MOVE   Move Data Memory

Syntax

MOVE   [mlr], Rd   —indexed, register
MOVE   [lr + A], Rd   —register-relative, register
MOVE   [IZ + n], rd   —immediate-relative, limited register

Affected Flags

None

Description

Moves a data memory byte into the destination register specified. The data memory source operand may specify any one of the index register modes; [mlr], [lr + A], [IZ + n]. The index register-relative mode, [lr + A], forms its data memory address by adding the contents of the index register lr to the unsigned 8-bit value contained in the currently active accumulator. The immediate-relative mode, [IZ + n], forms its data memory address by adding the contents of the index register IZ to the unsigned 8-bit immediate value n. The destination register operand Rd may specify any active CPU register; where as, the destination register operand rd is limited to the active registers R0–R15.

Example

The first example loads the current accumulator by "popping" an external data stack, which is pointed to by the index register IX.

    MOVE   [+ IX], A   ;pop accum from ext. stack

The second example demonstrates the random access of a data byte within a logical record contained in memory. The index register IY contains the base address of the logical record.

    ADDA   R9, A   ;calculate offset into record
    MOVE   [IY + A], R20 ;get data byte from record

In the final example, the 4th element of an Error Count table is transmitted to a host. The index register IZ points to the 1st entry of the table.

    EXX   0,1   ;select main A, alt B
    MOVE   [IZ + 3], RTR ;transmit 4th element

Instruction Format

MOVE   [mlr], Rd

1	1	0	0	0	0	0	0	m	lr	Rd
		Opcode								

15   8   6   4   0

```
00 = post-decrement      00 = IW
01 = no change           01 = IX
10 = post increment      10 = IY
11 = pre-increment       11 = IZ
```

TL/F/9336–9

MOVE   [lr + A], Rd

1	1	0	0	0	0	1	0	0	0	lr	Rd
		Opcode									

15   6 ↓   4   0

```
00—IW
01—IX
10—IY
11—IZ
```

MOVE   [IZ + n], rd

1	0	0	1	n	rd
	Opcode				

15   11   3   0

T-states

3
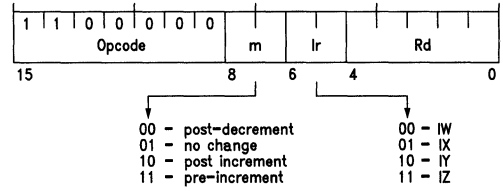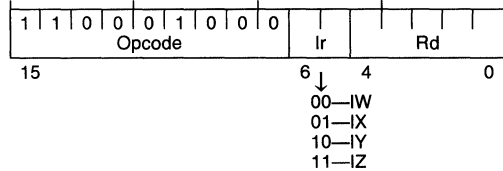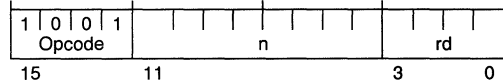
Bus Timing

Figure 6-5

Operation

MOVE   [mlr], Rd
data memory $\longrightarrow$ Rd

MOVE   [lr + A], Rd
data memory $\longrightarrow$ Rd

MOVE   [IZ + n], rd
data memory $\longrightarrow$ rd

2

6.0 Reference Section (Continued)

MOVE   Move Immediate

Syntax

MOVE   n, rd   —immediate, limited register

MOVE   n, [lr]   —immediate, indexed

Affected Flags

None

Description

Moves the immediate value n into the destination specified. The destination may be either a register, rd, (limited to the active registers R0–R15), or data memory via an index register, lr. The value n is 8 bits wide.

Example

Load the current accumulator with the value of 4.

MOVE   4, A   ;Load accumulator

Instruction Format

MOVE   n, rd

1	0	1	1	1											
Opcode						n						rd			

15   11   3   0

MOVE   n, [lr]

1	0	0	0	1	0										
Opcode						n[7–5]			lr		n[4–0]				

15   9   6 ↓   4   0

00—IW
01—IX
10—IY
11—IZ

T-states

MOVE   n, rd   —2

MOVE   n, [lr]   —3

Bus Timing

MOVE   n, rd   —*Figure 6-1*

MOVE   n, [lr]   —*Figure 6-6*

Operation

MOVE   n, rd

n → rd

MOVE   n, [lr]

n → data memory

6.0 Reference Section (Continued)

MOVE   Move Register

Syntax

MOVE	Rs, Rd	—register, register
MOVE	Rs, [mlr]	—register, indexed
MOVE	Rs, [lr+A]	—register, register-relative
MOVE	rs, [IZ+n]	—limited register, immediate-relative

Affected Flags
None

Description

Moves the contents of the source register into the destination specified. The source register operand Rs may specify any active CPU register; where as the source register operand rs is limited to the active registers R0–R15. The destination operand may specify either any active CPU register, Rd, or data memory via one of the index register modes; [mlr], [lr+A], [IZ+n]. The index register-relative mode, [lr+A], forms its data memory address by adding the contents of the index register lr to the unsigned 8-bit value contained in the currently active accumulator. The immediate-relative mode, [IZ+n], forms its data memory address by adding the contents of the index register IZ to the unsigned 8-bit immediate value n.

Example

The first example loads the Transmitter FIFO with a data byte in register 20.

```
EXX    0,1           ;select main A, alt B
MOVE   R20, RTR      ;Load the Transmitter FIFO
```

The second example "pushes" the current accumulator's contents onto an external data stack, which is pointed to by the index register IX.

```
MOVE   A, [IX−]      ;push accum to ext. stack
```

The third example demonstrates the random access of a data byte within a logical record contained in memory. The index register IY contains the base address of the logical record.

```
ADDA   R9, A         ;calculate offset into record
MOVE   R20, [IY+A]   ;update data byte in record
```

In the final example, the 4th element of an Error Count table is updated with a new value contained in the current accumulator. The index register IZ points to the 1st entry of the table.

```
MOVE   A, [IZ+3]     ;update 4th element of table
```

Instruction Format
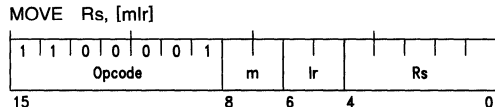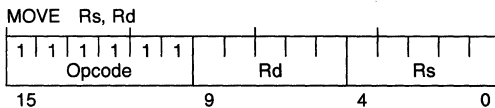
MOVE   Rs, Rd

1	1	1	1	1	1	1	1								
			Opcode						Rd				Rs		
15							9				4				0

MOVE   Rs, [mlr]

1	1	0	0	0	0	0	1								
			Opcode					m		lr			Rs		
15							8		6		4				0

```
00 - post-decrement          00 - IW
01 - no change               01 - IX
10 - post increment          10 - IY
11 - pre-increment           11 - IZ
```

TL/F/9336–10

MOVE   Rs, [lr+A]

1	1	0	0	0	0	1	1	0	0	1					
					Opcode							lr		Rs	
15									6		4				0

```
00—IW
01—IX
10—IY
11—IZ
```

MOVE   rs, [Z+n]

0	0	0	0	1											
		Opcode							n					rs	
15				11									3		0
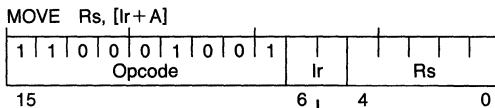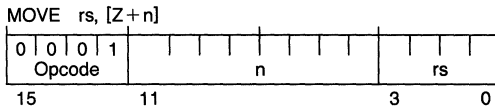
T-states

MOVE	Rs, Rd	—2
MOVE	Rs, [mlr]	—3
MOVE	Rs, [lr+A]	—3
MOVE	rs, [IZ+n]	—3

Bus Timing

MOVE	Rs, Rd	—Figure 6-1
MOVE	Rs, [mlr]	—Figure 6-6
MOVE	Rs, [lr+A]	—Figure 6-6
MOVE	rs, [IZ+n]	—Figure 6-6

Operation

MOVE	Rs, Rd	—Rs → Rd
MOVE	Rs, [mlr]	—Rs → data memory
MOVE	Rs, [lr+A]	—Rs → data memory
MOVE	rs, [IZ+n]	—rs → data memory

6.0 Reference Section (Continued)

OR   OR Immediate

Syntax

OR   n, rsd   —immediate, limited register

Affected Flags

N, Z

Description

Logically ORs the immediate value n to the register rsd and places the result back into the register rsd. Note that only the active registers R0–R15 may be specified for rsd. The value of n is 8 bits wide.

Example

Mask both the Transmitter and Receiver interrupts via the Interrupt Control Register {ICR}, R2. Leave the other interrupts unaffected.

```
EXX    0,0              ;select main reg banks
OR     00000011B, ICR   ;mask transmitter and
                        ; receiver interrupts
```

Instruction Format

0	1	0	1												
	Opcode						n						rsd		
15			11								3				0

T-states

2
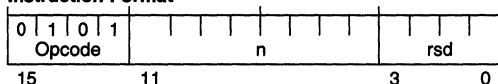
Bus Timing

Figure 6-1

Operation

rsd   OR   n → rsd

ORA   OR with Accumulator

Syntax

ORA   Rs, Rd   —register, register

ORA   Rs, [mlr]   —register, indexed

Affected Flags

N, Z

Description

Logically ORs the source register Rs to the active accumulator and places the result into the destination specified. The destination may be either a register, Rd, or data memory via an index register mode, [mlr]. Note that register bank selection determines which accumulator is active.

Example

Write an 11-bit word to the Transmitter's FIFO. This example assumes that the index register IZ points to the location of the data in memory.

```
TCR.settings:  .EQU   00101000B
. . .
EXX    1,1              ;select main A, alt B
MOVE   TCR.settings,A   ;load accumulator w/mask
MOVE   [IZ+],R20        ;load bits 8, 9, & 10
ORA    R20,TCR          ;write bits 8, 9, 10 to {TCR}
MOVE   [IZ+],RTR        ;push 11-bit word to FIFO
```

Instruction Format

ORA   Rs, Rd
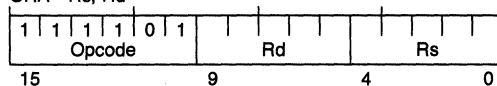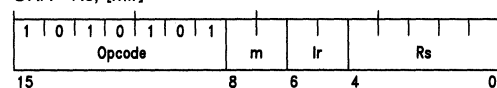
1	1	1	1	1	0	1									
		Opcode							Rd				Rs		
15						9				4					0

ORA   Rs, [mlr]

1	0	1	1	0	1	0	1								
		Opcode						m		lr			Rs		
15							8		6		4				0

```
00 – post-decrement          00 – IW
01 – no change               01 – IX
10 – post increment          10 – IY
11 – pre-increment           11 – IZ
```

TL/F/9336–11

T-states

| ORA   Rs, Rd | —2 |
| ORA   Rs, [mlr] | —3 |

Bus Timing

| ORA   Rs, Rd | —*Figure 6-1* |
| ORA   Rs, [mlr] | —*Figure 6-6* |

Operation

ORA   Rs, Rd

Rs   OR   accumulator → Rd

ORA   Rs, [mlr]

Rs   OR   accumulator → data memory

6.0 Reference Section (Continued)

RETF Conditional Return

Rcc

Syntax

RETF   f, s{,{g} {,rf}}
Rcc   {g{,rf}}   —(optional syntax)

Affected Flags

If rf = 1 then N, Z, C, and V

Description

Conditionally returns control to the last instruction address pushed onto the internal Address Stack by popping that address into the Program Counter, if the state of the flag referenced by f is equal to the state of the bit s; or, optionally, if the condition cc is met. See the tables on the following page for the flags that f can reference and the conditions that cc may specify. The conditional return instruction also has two optional operands, g and rf. The value of g determines if the Global Interrupt Enable bit [GIE] is left unchanged (g=0), restored from the Address Stack (g=1), set (g=2), or cleared (g=3). If the g operand is omitted then g=0 is assumed. The second optional operand, rf, determines if the ALU flags and register bank selections are left unchanged (rf=0), or restored from the Address Stack (rf=1). If the rf operand is omitted then rf=0 is assumed.

Example

This example demonstrates both syntaxes of the conditional return instruction testing for a carry result from a previous instruction; (i.e., [C] = 1). If the condition is met then the return occurs, else the next instruction following the return is executed. The current environment is left unchanged.

```
RETF   001B,1      ; If [C] = 1 then return
. . .
RC                 ; If [C] = 1 then return
```

Instruction Format
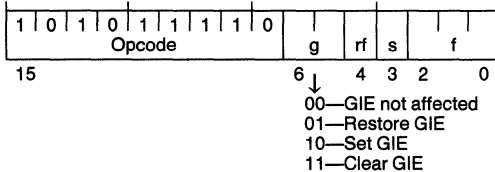
1	0	1	0	1	1	1	1	1	0						
			Opcode							g	rf	s		f	
15									6		4	3	2		0

00—GIE not affected
01—Restore GIE
10—Set GIE
11—Clear GIE

T-states

2 if condition is not met

3 if condition is met

Bus Timing

Figure 6-1 if condition is not met
Figure 6-2 if condition is met

Operation

If flag f is in state s then
  Case g of
    0: leave [GIE] unaffected, (default)
    1: restore [GIE] from Address Stack
    2: set [GIE]
    3: clear [GIE]
  End case
  If rf = 1 then
    restore ALU flags from Address Stack
    restore register bank selection from Address Stack
  End if
  Address Stack → PC
End if

Condition Specification Table for "cc"

cc	Meaning	Condition Tested for
Z	Zero	[Z] = 1
NZ	Not Zero	[Z] = 0
EQ	Equal	[Z] = 1
NEQ	Not Equal	[Z] = 0
C	Carry	[C] = 1
NC	No Carry	[C] = 0
V	Overflow	[V] = 1
NV	No Overflow	[V] = 0
N	Negative	[N] = 1
P	Positive	[N] = 0
RA	Receiver Active	[RA] = 1
NRA	Not Receiver Active	[RA] = 0
RE	Receiver Error	[RE] = 1
NRE	No Receiver Error	[RE] = 0
DA	Data Available	[DAV] = 1
NDA	No Data Available	[DAV] = 0
TFF	Transmitter FIFO Full	[TFF] = 1
NTFF	Transmitter FIFO Not Full	[TFF] = 0

Flag Reference Table for "f"

f	(binary)	Flag Referenced	
0	(000)	[Z]	in {CCR}
1	(001)	[C]	in {CCR}
2	(010)	[V]	in {CCR}
3	(011)	[N]	in {CCR}
4	(100)	[RA]	in {TSR}
5	(101)	[RE]	in {TSR}
6*	(110)	[DAV]	in {TSR}
7	(111)	[TFF]	in {TSR}

*Note: The value of f for [DAV] differs from the numeric value for the position of [DAV] in {TSR}.

6.0 Reference Section (Continued)

RET   Unconditional Return

Syntax

RET   {g {,rf}}

Affected Flags

If rf = 1 then N, Z, C, and V

Description

Unconditionally returns control to the last instruction address pushed onto the internal Address Stack by popping that address into the Program Counter. The unconditional return instruction also has two optional operands, g and rf. The value of g determines if the Global Interrupt Enable bit [GIE] is left unchanged (g = 0), restored from the Address Stack (g = 1), set (g = 2), or cleared (g = 3). If the g operand is omitted then g = 0 is assumed. The second optional operand, rf, determines if the ALU flags and register bank selections are left unchanged (rf = 0), or restored from the Address Stack (rf = 1). If the rf operand is omitted then rf = 0 is assumed.

Example

Return from an interrupt.

    RET   1,1   ;Restore environment & return

Instruction Format

1	0	1	0	0	1	1	1	1	1	1			0	0	0	0
			Opcode								g	rf				

15   6 ↓   4   3   0

00—GIE not affected
01—Restore GIE
10—Set GIE
11—Clear GIE

T-states

2
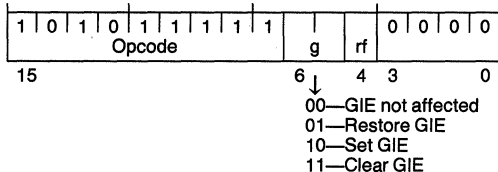
Bus Timing

Figure 6-1

Operation

Case g of
    0: leave [GIE] unaffected, (default)
    1: restore [GIE] from Address Stack
    2: set [GIE]
    3: clear [GIE]
End case
If rf = 1 then
    restore ALU flags from Address Stack
    restore register bank selection from Address Stack
End if
Address Stack → PC

ROT   Rotate

Syntax

ROT   Rsd, b   —register

Affected Flags

N, Z, C

Description

Rotates the contents of the register Rsd b bits to the right and places the result back into that register. The bits that are shifted out of the LSB are shifted back into the MSB, (and copied into the Carry flag). The value b may specify from 0 to 7 bit rotates.

Example

Add 3 to the Address Stack Pointer contained in the Internal Stack Pointer register {ISP}, R30.

    MOVE   ISP, R8   ;get {ISP}
    ROT   R8, 4   ;shift [ASP] to low order nibble
    ADD   3, R8   ;add 3 to [ASP]
    ROT   R8, 4   ;shift [ASP] to high order nibble
    MOVE   R8, ISP   ;store new {ISP}

Instruction Format

1	1	0	0	0	1	0	1	0					
		Opcode							b		Rsd		

15   7   4   0

T-states

2
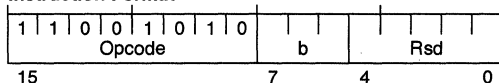
Bus Timing

Figure 6-1

Operation
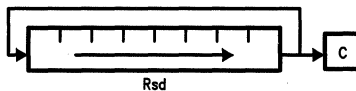
TL/F9336–12

6.0 Reference Section (Continued)

SBCA Subtract with Carry and Accumulator

Syntax

SBCA Rs, Rd   —register, register
SBCA Rs, [mlr]   —register, indexed

Affected Flags

N, Z, C, V

Description

Subtracts the active accumulator and the carry flag from the source register Rs, placing the result into the destination specified. The destination may be either a register, Rd, or data memory via an index register mode, [mlr]. Negative results are represented using the two's complement format. Note that register bank selection determines which accumulator is active.

Example

Subtract the constant 109 from the index register IW, (which is 16 bits wide).

```
SUBA   A, A         ;Clear the accumulator
SUB    109, R12     ;low byte of IW—109
SBCA   R13, R13     ;high byte of IW—borrow
```

Instruction Format
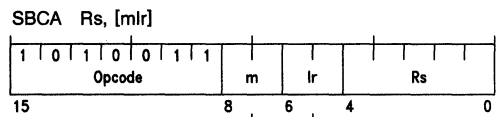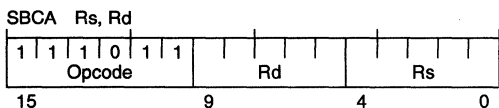
SBCA Rs, Rd

1	1	1	1	0	1	1									
		Opcode						Rd				Rs			
15					9				4				0		

SBCA Rs, [mlr]

1	0	1	1	0	0	1	1								
		Opcode						m	lr		Rs				
15							8		6		4				0

```
00 - post-decrement        00 - IW
01 - no change             01 - IX
10 - post increment        10 - IY
11 - pre-increment         11 - IZ
```

TL/F9336–13

T-states

SBCA Rs, Rd   —2
SBCA Rs, [mlr]   —3

Bus Timing

SBCA Rs, Rd   —Figure 6-1
SBCA Rs, [mlr]   —Figure 6-6

Operation

SBCA Rs, Rd
Rs − accumulator − carry bit → Rd

SBCA Rs, [mlr]
Rs − accumulator − carry bit → data memory

SHL Shift Left

Syntax

SHL Rsd, b   —register

Affected Flags

N, Z, C

Description

Shifts the contents of the register Rsd b bits to the left and places the result back into that register. Zeros are shifted in from the right, (i.e., from the LSB). The value b may specify from 0 to 7 bit shifts. The Carry flag contains the last bit shifted out.

Example

Place a new internal Address Stack Pointer into the Internal Stack Pointer register {ISP}, R30. Assume that the new [ASP] is located in register 20.

```
MOVE   ISP,R8           ;read {ISP} for [DSP]
AND    00001111B,R8     ;save [DSP] only
SHL    R20,4            ;left justify [ASP]
ORA    R20,ISP          ;combine [ASP] + [DSP],
                        ; then place into {ISP}
```

Instruction Format

1	1	0	0	1	1	0	0	1							
		Opcode							(8-b)			Rsd			
15							7		4				0		

T-states

2
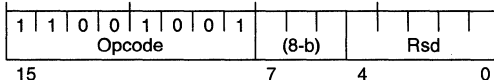
Bus Timing

Figure 6-1

Operation
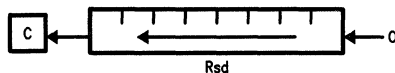
TL/F/9336–14

6.0 Reference Section (Continued)

SHR   Shift Right

Syntax

SHR   Rsd, b   —register

Affected Flags

N, Z, C

Description

Shifts the contents of the register Rsd b bits to the right and places the result back into that register. Zeros are shifted in from the left, (i.e., from the MSB). The value b may specify from 0 to 7 bit shifts. The Carry flag contains the last bit shifted out.

Example

Right justify the Address Stack Pointer from the Internal Stack Pointer register {ISP}, R30.

    MOVE   ISP, R20   ;Load [ASP] from {ISP}
    SHR   R20,4   ;right justify [ASP]

Instruction Format

1	1	0	0	1	0	0	0		b			Rsd	
			Opcode										

15   7   4   0

T-states

2
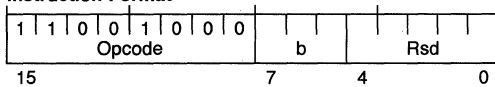
Bus Timing

Figure 6-1

Operation
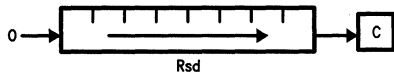
TL/F/9336–15

SUB   Subtract Immediate

Syntax

SUB n, rsd   —immediate, limited register
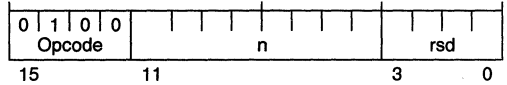
Affected Flags

N, Z, C, V

Description

Subtracts the immediate value n from the register rsd and places the result back into the register rsd. Note that only the active registers R0–R15 may be specified for rsd. The value of n is limited to 8 bits; (signed range: +127 to −128). Negative numbers are represented using the two's complement format.

Example

Subtract the constant 3 from register 10.

    SUB   3, R10   ; R10 − 3 → R10

Instruction Format

0	0	1	0				n					rsd	
	Opcode												

15   11   3   0

T-states

2
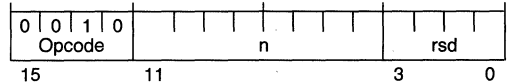
Bus Timing

Figure 6-1

Operation

rsd − n → rsd

6.0 Reference Section (Continued)

SUBA   Subtract with Accumulator

Syntax

SUBA   Rs, Rd   —register, register
SUBA   Rs, [mlr]   —register, indexed

Affected Flags

N, Z, C, V

Description

Subtracts the active accumulator from the source register Rs and places the result into the destination specified. The destination may be either a register, Rd, or data memory via an index register mode, [mlr]. Negative numbers are represented using the two's complement format. Note that register bank selection determines which accumulator is active.

Example

In the first example, the value 4 is placed into the currently active accumulator, that accumulator is subtracted from the contents of register 20, and then the result is placed into register 21.

    MOVE   4, A   ;Place constant into accum
    SUBA   R20, R21   ;R20 − accum → R21

In the second example, the alternate accumulator of register bank B is selected and then subtracted from register 20. The result is placed into the data memory pointed to by the index register IZ and then the value of IZ is incremented by one.

    EXX   0, 1   ;Select alt accumulator
    SUBA   R20, [IZ+] ;R20 − accum → data mem
                      ;and increment data pointer

Instruction Format

SUBA Rs, Rd
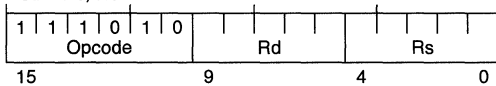
1	1	1	1	0	1	0							
		Opcode						Rd				Rs	

15   9   4   0

SUBA Rs, [mlr]

1	0	1	1	0	0	1	0							
		Opcode						m	Ir			Rs		

15   8   6   4   0

```
00 = post-decrement      00 - IW
01 = no change           01 - IX
10 = post increment      10 - IY
11 = pre-increment       11 - IZ
```
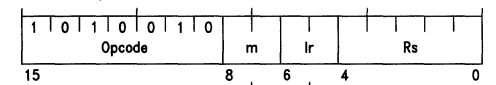
TL/F/9336–16

T-states

SUBA Rs, Rd   —2
SUBA Rs, [mlr]   —3

Bus Timing

SUBA Rs, Rd   —Figure 6-1
SUBA Rs, [mlr]   —Figure 6-6

Operation

SUBA Rs, Rd
Rs − accumulator → Rd

SUBA Rs, [mlr]
Rs − accumulator → data memory

TRAP   Software Interrupt

Syntax

TRAP v {,g'}

Affected Flags

None

Description

Pushes the Program Counter, the Global Interrupt Enable bit [GIE], the ALU flags, and the current register bank selections onto the internal Address Stack; then unconditionally transfers control to the instruction at the memory address created by concatenating the contents of the Interrupt Base Register {IBR} to the value of v extended with zeros to 8 bits. If the value of g' is equal to "1" then the Global Interrupt Enable bit [GIE] will be cleared. If the g' operand is omitted, then g' = 0 is assumed. The vector number v points to one of 64 Interrupt Table entries; (range: 0 to 63). Since some of the Interrupt Table entries are used by the hardware interrupts, the TRAP instruction can simulate hardware interrupts. The following table lists the hardware interrupts and their associated vector numbers:

Hardware Interrupt Vector Table

Interrupt	v	(Binary)
NMI	28	(011100)
RFF/DA/RA	4	(000100)
TFE	8	(001000)
LTA	12	(001100)
BIRQ	16	(010000)
TO	20	(010100)

Example

Simulate the Transmitter FIFO Empty interrupt.

    TRAP   8, 1   ;TFE interrupt simulation

Instruction Format

1	1	0	0	1	1	1	1	1	1					
		Opcode								g'			v	

15   6   5   0

T-states

2
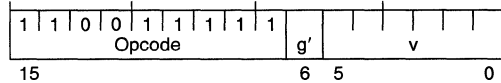
Bus Timing

Figure 6-1

Operation

PC & [GIE] & ALU flags & register bank selections
   → Address Stack

if g' = 1
   then clear [GIE]

Create PC address by concatenating the {IBR} register to the vector number v as shown below:

				{IBR}				0	0				v			→ PC

15   7   5   0
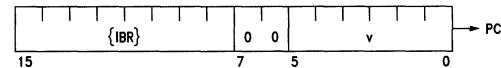
TL/F/9336–17

6.0 Reference Section (Continued)

XOR Exclusive OR Immediate

Syntax
XOR n, rsd   —immediate, limited register

Affected Flags
N, Z

Description
Logically exclusive ORs the immediate value n to the register rsd and places the result back into the register rsd. Note that only the active registers R0–R15 may be specified for rsd. The value of n is 8 bits wide.

Example
Encode/decode a data byte in register 15.

    XOR code_pattern, R15   ;encode/decode

Instruction Format

0	1	1	1	0												
Opcode							n						rsd			

15   11   3   0
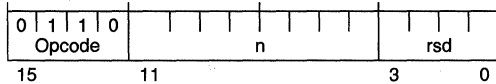
T-states
2

Bus Timing
Figure 6-1

Operation
rsd XOR n $\rightarrow$ rsd

XORA Exclusive OR with Accumulator

Syntax
XORA   Rs, Rd   —register, register
XORA   Rs, [mlr] —register, indexed

Affected Flags
N, Z

Description
Logically exclusive ORs the source register Rs to the active accumulator and places the result into the destination specified. The destination may be either a register, Rd, or data memory via an index register mode, [mlr]. Note that register bank selection determines which accumulator is active.

Example
Decode the data byte just received and place it into data memory. This example assumes that the accumulator contains the "key" and that the index register IY points to the location where the information should be stored.

    EXX   1,1   ;select alternate banks
    XORA   RTR, [IY+] ;decode received byte and
                      ; save it

Instruction Format
XORA Rs, Rd
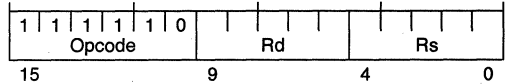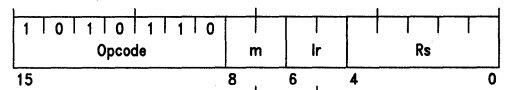
1	1	1	1	1	1	0										
Opcode								Rd				Rs				

15   9   4   0

XORA Rs, [mlr]

1	0	1	1	0	1	1	0									
Opcode								m		lr		Rs				

15   8   6   4   0

00 – post-decrement   00 – IW
01 – no change   01 – IX
10 – post increment   10 – IY
11 – pre-increment   11 – IZ

TL/F/9336–18

T-states
XORA Rs, Rd   —2
XORA Rs, [mlr] —3

Bus Timing
XORA Rs, Rd   —*Figure 6-1*
XORA Rs, [mlr] —*Figure 6-6*

Operation
XORA Rs, Rd
Rs XOR accumulator $\rightarrow$ Rd

XORA Rs, [mlr]
Rs XOR accumulator $\rightarrow$ data memory

6.0 Reference Section (Continued)

TABLE 6-2. Instructions vs T-states, Affected Flags, and Bus Timing

Instruction		T-states	Affected Flags	Timing Figure	Instruction		T-states	Affected Flags	Timing Figure
ADCA	Rs, Rd	2	N,Z,C,V	6-1	MOVE	Rs, Rd	2		6-1
ADCA	Rs, [mlr]	3	N,Z,C,V	6-6	MOVE	Rs, [mlr]	3		6-6
ADD	n, rsd	2	N,Z,C,V	6-1	MOVE	Rs, [lr + A]	3		6-6
ADDA	Rs, Rd	2	N,Z,C,V	6-1	MOVE	rs, [IZ + n]	3		6-6
ADDA	Rs, [mlr]	3	N,Z,C,V	6-6	MOVE	[mlr], Rd	3		6-5
AND	n, rsd	2	N,Z	6-1	MOVE	[lr + A], Rd	3		6-5
ANDA	Rs, Rd	2	N,Z	6-1	MOVE	[IZ + n], rd	3		6-5
ANDA	Rs, [mlr]	3	N,Z	6-6	OR	n, rsd	2	N,Z	6-1
BIT	rs, n	2	N,Z	6-1	ORA	Rs, Rd	2	N,Z	6-1
CALL	n	3		6-2	ORA	Rs, [mlr]	3	N,Z	6-6
CMP	rs, n	2	N,Z,C,V	6-1	Rcc	{g{,rf}}	2 false		6-1
CPL	Rsd	2	N,Z	6-1			3 true	N,Z,C,V*	6-2
EXX	ba, bb {,g}	2		6-1	RET	{g{,rf}}	2	N,Z,C,V*	6-1
Jcc	n	2 false		6-1	RETF	f, s {,{g} {,rf}}	2 false		6-1
		3 true		6-2			3 true	N,Z,C,V*	6-2
JMP	f, s, n	2 false		6-1	ROT	Rsd, b	2	N,Z,C	6-1
		3 true		6-2	SBCA	Rs, Rd	2	N,Z,C,V	6-1
JMP	n	3		6-2	SBCA	Rs, [mlr]	3	N,Z,C,V	6-6
JMP	Rs	4		6-4	SHL	Rsd, b	2	N,Z,C	6-1
JRMK	Rs, b, m	4		6-4	SHR	Rsd, b	2	N,Z,C	6-1
LCALL	nn	(2+2)		6-3	SUB	n, rsd	2	N,Z,C,V	6-1
LCALL	Rs, p, s, nn	(2+2)		6-3	SUBA	Rs, Rd	2	N,Z,C,V	6-1
LJMP	nn	(2+2)		6-3	SUBA	Rs, [mlr]	3	N,Z,C,V	6-6
LJMP	[lr]	2		6-1	TRAP	v {,g'}	2		6-1
LJMP	Rs, p, s, nn	(2+2)		6-3	XOR	n, rsd	2	N,Z	6-1
MOVE	n, rd	2		6-1	XORA	Rs, Rd	2	N,Z	6-1
MOVE	n, [lr]	3		6-4	XORA	Rs, [mlr]	3	N,Z	6-6

*Note: If rf = 1 then N, Z, C, and V are affected.

2

6.0 Reference Section (Continued)

TABLE 6-3. Instruction Opcodes

Hex	Opcode		Instruction	
0000–0FFF	`0 0 0 0` Opcode | n | rsd (bits 15, 11, 3, 0)		ADD	n, rsd
1000–1FFF	`0 0 0 1` Opcode | n | rs (bits 15, 11, 3, 0)		MOVE	rs, [IZ + n]
2000–2FFF	`0 0 1 0` Opcode | n | rsd (bits 15, 11, 3, 0)		SUB	n, rsd
3000–3FFF	`0 0 1 1` Opcode | n | rs (bits 15, 11, 3, 0)		CMP	rs, n
4000–4FFF	`0 1 0 0` Opcode | n | rsd (bits 15, 11, 3, 0)		AND	n, rsd
5000–5FFF	`0 1 0 1` Opcode | n | rsd (bits 15, 11, 3, 0)		OR	n, rsd
6000–6FFF	`0 1 1 0` Opcode | n | rsd (bits 15, 11, 3, 0)		XOR	n, rsd
7000–7FFF	`0 1 1 1` Opcode | n | rs (bits 15, 11, 3, 0)		BIT	rs, n
8000–87FF	`1 0 0 0 0` Opcode | m | b | Rs (bits 15, 10, 7, 4, 0)		JRMK	Rs, b, m

KEY

mlr

00	lr−
01	lr
10	lr+
11	+lr

lr

00	IW
01	IX
10	IY
11	IZ

g

00	NCHG
01	RI
10	EI
11	DI

g'

0	NCHG
1	DI

ba/bb

0	MAIN
1	ALT

f

000	[Z]
001	[C]
010	[V]
011	[N]
100	[RA]
101	[RE]
110	[DAV]
111	[TFF]

6.0 Reference Section (Continued)

TABLE 6-3. Instruction Opcodes (Continued)

Hex	Opcode	Instruction
8800–8BFF	`1 0 0 0 1 0` Opcode / `n[7–5]` / `lr` / `n[4–0]` 15　　9　6　4　　0	MOVE　n, [lr]
8C00–8DFF	`1 0 0 0 1 1 0` Opcode / `s` / `p` / `Rs` 15　　　8 7　4　　　0	LJMP　Rs, p, s, nn
0000–FFFF	`nn` 15　　　　　　　　　0	
8E00–8FFF	`1 0 0 0 1 1 1` Opcode / `s` / `p` / `Rs` 15　　　8 7　4　　　0	LCALL　Rs, p, s, nn
0000–FFFF	`nn` 15　　　　　　　　　0	
9000–9FFF	`1 0 0 1` Opcode / `n` / `rd` 15　　11　　　　3　　0	MOVE　[IZ + n], rd
A000–A1FF	`1 0 1 0 0 0 0 0` Opcode / `m` / `lr` / `Rs` 15　　　　8　6　4　　0	ADDA　Rs, [mlr]
A200–A3FF	`1 0 1 0 0 0 0 1` Opcode / `m` / `lr` / `Rs` 15　　　　8　6　4　　0	ADCA　Rs, [mlr]
A400–A5FF	`1 0 1 0 0 0 1 0` Opcode / `m` / `lr` / `Rs` 15　　　　8　6　4　　0	SUBA　Rs, [mlr]

KEY

mlr
00	lr −
01	lr
10	lr +
11	+ lr

lr
00	IW
01	IX
10	IY
11	IZ

g
00	NCHG
01	RI
10	EI
11	DI

g'
0	NCHG
1	DI

ba/bb
0	MAIN
1	ALT

f
000	[Z]
001	[C]
010	[V]
011	[N]
100	[RA]
101	[RE]
110	[DAV]
111	[TFF]

2

6.0 Reference Section (Continued)

TABLE 6-3. Instruction Opcodes (Continued)

Hex	Opcode	Instruction
A600–A7FF	`1 0 1 1 0 0 1 1` Opcode / `m` / `lr` / `Rs` — bits 15, 8, 6, 4, 0	SBCA Rs, [mlr]
A800–A9FF	`1 0 1 1 0 1 0 0` Opcode / `m` / `lr` / `Rs` — bits 15, 8, 6, 4, 0	ANDA Rs, [mlr]
AA00–ABFF	`1 0 1 1 0 1 0 1` Opcode / `m` / `lr` / `Rs` — bits 15, 8, 6, 4, 0	ORA Rs, [mlr]
AC00–ADFF	`1 0 1 1 0 1 1 0` Opcode / `m` / `lr` / `Rs` — bits 15, 8, 6, 4, 0	XORA Rs, [mlr]
AE00–AE1F	`1 0 1 1 0 1 1 1 0 0 0 0` Opcode / `Rs` — bits 15, 4, 0	CPL Rsd
AE80–AEF8	`1 0 1 1 0 1 1 1 0 1` Opcode / `g` `ba` `bb` `0 0 0` — bits 15, 6, 4, 3, 2, 0	EXx ba, bb {,g}
AF00–AF7F	`1 0 1 1 0 1 1 1 1 1 0` Opcode / `g` `rf` `s` `f` — bits 15, 6, 4, 3, 2, 0	RETF f,s{,{g}{,rf}} Rcc {g{,rf}}
AF80–AFF0	`1 0 1 1 0 1 1 1 1 1 1` Opcode / `g` `rf` `0 0 0 0` — bits 15, 6, 4, 3, 0	RET {g{,rf}}
B000–BFFF	`1 0 1 1 1` Opcode / `n` / `rd` — bits 15, 11, 3, 0	MOVE n, rd

KEY

mlr

00	lr−
01	lr
10	lr+
11	+lr

lr

00	IW
01	IX
10	IY
11	IZ

g

00	NCHG
01	RI
10	EI
11	DI

g'

0	NCHG
1	DI

ba/bb

0	MAIN
1	ALT

f

000	[Z]
001	[C]
010	[V]
011	[N]
100	[RA]
101	[RE]
110	[DAV]
111	[TFF]

6.0 Reference Section (Continued)

TABLE 6-3. Instruction Opcodes (Continued)

Hex	Opcode	Instruction	
C000–C1FF	`1 1 0 0 0 0 0` Opcode `m lr Rd` 15　8　6　4　0	MOVE	[mlr], Rd
C200–C3FF	`1 1 0 0 0 0 1` Opcode `m lr Rs` 15　8　6　4　0	MOVE	Rs, [mlr]
C400–C47F	`1 1 0 0 0 1 0 0 0` Opcode `lr Rd` 15　6　4　0	MOVE	[lr+A], Rd
C480–C4FF	`1 1 0 0 0 1 0 0 1` Opcode `lr Rs` 15　6　4　0	MOVE	Rs, [lr+A]
C800–C8FF	`1 1 0 0 1 0 0 0` Opcode `b Rsd` 15　7　4　0	SHR	Rsd, b
C900–C9FF	`1 1 0 0 1 0 0 1` Opcode `(8-b) Rsd` 15　7　4　0	SHL	Rsd, b
CA00–CAFF	`1 1 0 0 1 0 1 0` Opcode `b Rsd` 15　7　4　0	ROT	Rsd, b
CB00–CBFF	`1 1 0 0 1 0 1 1` Opcode `n` 15　7　0	JMP	n
CC00–CCFF	`1 1 0 0 1 1 0 0` Opcode `n` 15　7　0	CALL	n

KEY

mlr

00	lr−
01	lr
10	lr+
11	+lr

lr

00	IW
01	IX
10	IY
11	IZ

g

00	NCHG
01	RI
10	EI
11	DI

g′

0	NCHG
1	DI

ba/bb

0	MAIN
1	ALT

f

000	[Z]
001	[C]
010	[V]
011	[N]
100	[RA]
101	[RE]
110	[DAV]
111	[TFF]

2

6.0 Reference Section (Continued)

TABLE 6-3. Instruction Opcodes (Continued)

Hex	Opcode	Instruction	
CD00–CD60	`1 1 0 0 1 1 0 1 0` `lr` `0 0 0 0 0` — Opcode (15...6) lr (4) (0)	LJMP	[lr]
CD80–CD9F	`1 1 0 0 1 1 0 1 1 0 0` `Rs` — Opcode (15...4) Rs (0)	JMP	Rs
CE00 0000–FFFF	`1 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0` — Opcode (15...0) `nn` (15...0)	LJMP	nn
CE80 0000–FFFF	`1 1 0 0 1 1 1 0 1 0 0 0 0 0 0 0` — Opcode (15...0) `nn` (15...0)	LCALL	nn
CF80–CFFF	`1 1 0 0 1 1 1 1 1` `g'` `v` — Opcode (15...6) g' (5) v (0)	TRAP	v{,g'}
D000–DFFF	`1 1 0 1` `s` `f` `n` — Opcode (15...11) s (10) f (7) n (0)	JMP Jcc	f, s, n n

KEY

mlr

00	lr−
01	lr
10	lr+
11	+lr

lr

00	IW
01	IX
10	IY
11	IZ

g

00	NCHG
01	RI
10	EI
11	DI

g'

0	NCHG
1	DI

ba/bb

0	MAIN
1	ALT

f

000	[Z]
001	[C]
010	[V]
011	[N]
100	[RA]
101	[RE]
110	[DAV]
111	[TFF]

6.0 Reference Section (Continued)

TABLE 6-3. Instruction Opcodes (Continued)

Hex	Opcode	Instruction	
E000–E3FF	`1 1 1 1 0 0 0` Opcode / Rd / Rs — 15 9 4 0	ADDA	Rs, Rd
E400–E7FF	`1 1 1 1 0 0 1` Opcode / Rd / Rs — 15 9 4 0	ADCA	Rs, Rd
E800–EBFF	`1 1 1 1 0 1 0` Opcode / Rd / Rs — 15 9 4 0	SUBA	Rs, Rd
EC00–EFFF	`1 1 1 1 0 1 1` Opcode / Rd / Rs — 15 9 4 0	SBCA	Rs, Rd
F000–F3FF	`1 1 1 1 1 0 0` Opcode / Rd / Rs — 15 9 4 0	ANDA	Rs, Rd
F400–F7FF	`1 1 1 1 1 0 1` Opcode / Rd / Rs — 15 9 4 0	ORA	Rs, Rd
F800–FBFF	`1 1 1 1 1 1 0` Opcode / Rd / Rs — 15 9 4 0	XORA	Rs, Rd
FC00–FFFF	`1 1 1 1 1 1 1` Opcode / Rd / Rs — 15 9 4 0	MOVE	Rs, Rd

KEY

mlr

00	lr−
01	lr
10	lr+
11	+lr

lr

00	IW
01	IX
10	IY
11	IZ

g

00	NCHG
01	RI
10	EI
11	DI

g'

0	NCHG
1	DI

ba/bb

0	MAIN
1	ALT

f

000	[Z]
001	[C]
010	[V]
011	[N]
100	[RA]
101	[RE]
110	[DAV]
111	[TFF]

2

6.0 Reference Section (Continued)

6.2 REGISTER SET REFERENCE

The register set reference contains detailed information on the bit definitions of all special function registers that are addressable in the CPU. This reference section presents the information in three forms: a bit index, a register description and bit definition tables. The bit index is an alphabetical listing of all status/control bits in the CPU-addressable function registers, with a brief summary of the function. The register description is a list of all CPU-addressable special function registers in alphabetical order. The bit definition tables describe the location and function of all control and status bits in the various CPU-addressable special function registers. These tables are arranged by function.

6.2.1 Bit Index

An alphabetical listing of all status/control bits in the CPU-addressable special function registers, with a brief summary of function. Detailed definitions are provided in Section 6.2.3, Bit Definition Tables.

Bit	Name	Location		Function
ACK	poll/ACKnowledge	NCF	[1]	Receiver Status
ASP3-0	Address Stack Pointer	ISP	[7-4]	Stacks
AT7-0	Auxilliary Transceiver control	ATR	[7-0]	Receiver Control
ATA	Advance Transmitter Active	TCR	[4]	Transmitter Control
BIC	Bi-directional Interrupt Control	ACR	[4]	Interrupt Control
BIRQ	Bi-directional Interrupt ReQuest	CCR	[4]	Interrupt Control
C	Carry	CCR	[1]	Arithmetic Flag
CCS	CPU Clock Select	DCR	[7]	Timing Control
COD	Clock Out Disable	ACR	[2]	Timing Control
DAV	Data AVailable	TSR	[3]	Receiver Status
DEME	Data Error or Message End	NCF	[3]	Receiver Status
DS7-0	Data Stack	DS	[7-0]	Stacks
DSP3-0	Data Stack Pointer	ISP	[3-0]	Stacks
DW2-0	Data memory Wait-state select	DCR	[2-0]	Timing Control
FB7-0	Fill Bits	FBR	[7-0]	Transmitter Control
GIE	Global Interrupt Enable	ACR	[0]	Interrupt Control
IES	Invalid Ending Sequence	ECR	[2]	Receiver Error Code
IM4-0	Interrupt Mask select	ICR	[4-0]	Interrupt Control
IV15-8	Interrupt Vector	IBR	[7-0]	Interrupt Control
IW1,0	Instruction memory Wait-state select	DCR	[4,3]	Timing Control
LA	Line Active	NCF	[5]	Receiver Status
LMBT	Loss of Mid Bit Transition	ECR	[1]	Receiver Error Code
LOR	Lock Out Remote	ACR	[1]	Remote Interface
LOOP	internal LOOP-back	TMR	[6]	Transceiver Control
LTA	Line Turn Around	NCF	[4]	Receiver Status
N	Negative	CCR	[3]	Arithmetic Flag
OVF	receiver OVerFlow	ECR	[4]	Receiver Error Code
OWP	Odd Word Parity	TCR	[3]	Transmitter Control
PAR	PARity error	ECR	[3]	Receiver Error Code
POLL	POLL	NCF	[0]	Receiver Status
PS2-0	Protocol Select	TMR	[2-0]	Transceiver Control
RA	Receiver Active	TSR	[4]	Receiver Status
RAR	Received Auto-Response	NCF	[2]	Receiver Status
RDIS	Receiver DISabled while active	ECR	[0]	Receiver Error Code
RE	Receiver Error	TSR	[5]	Receiver Status
RF10-8	Receive FIFO	TSR	[2-0]	Receiver Control
RFF	Receive FIFO Full	NCF	[6]	Receiver Status
RIN	Receiver INvert	TMR	[4]	Receiver Control
RIS1,0	Receiver Interrupt Select	ICR	[7,6]	Interrupt Control
RLQ	Receive Line Quiesce	TCR	[7]	Receiver Control
RPEN	RePeat ENable	TMR	[5]	Receiver Control
RR	Remote Read	CCR	[6]	Remote Interface
RTF7-0	Receive/Transmit FIFO	RTR	[7-0]	Transceiver Control
RW	Remote Write	CCR	[5]	Remote Interface
SEC	Select Error Codes	TCR	[6]	Receiver Control
SLR	Select Line Receiver	TCR	[5]	Receiver Control
TA	Transmitter Active	TSR	[6]	Transmitter Status
TCS1,0	Transceiver Clock Select	DCR	[6,5]	Transceiver Control
TF10-8	Transmit FIFO	TCR	[2-0]	Transmitter Control

6.0 Reference Section (Continued)

6.2.1 Bit Index (Continued)

An alphabetical listing of all status/control bits in the CPU-addressable special function registers, with a brief summary of function. Detailed definitions are provided in Section 6.2.3, Bit Definition Tables.

Bit	Name	Location		Function
TFE	Transmit FIFO Empty	NCF	[7]	Transmitter Status
TFF	Transmit FIFO Full	TSR	[7]	Transmitter Status
TIN	Transmitter INvert	TMR	[3]	Transmitter Control
TLD	Timer LoaD	ACR	[6]	Timer
TM7-0	TiMer	TRL	[7-0]	Timer
TM15-8	TiMer	TRH	[7-0]	Timer
TMC	TiMer Clock select	ACR	[5]	Timer
TO	Time Out flag	CCR	[7]	Timer
TRES	Transceiver RESet	TMR	[7]	Transceiver Control
TST	Timer StarT	ACR	[7]	Timer
V	oVerflow	CCR	[2]	Arithmetic Flag
Z	Zero	CCR	[0]	Arithmetic Flag

6.2.2 Register Description

A list of all CPU-addressable special function registers, in alphabetical order.

The Remote Interface Configuration register {RIC}, which is addressable only by the remote system, is not included. See Section 6.3, Remote Interface Reference for details of the function of this register.

Each register is listed together with its address, the type of access available, and a functional description of each bit. Further details on each bit can be found in Section 6.2.3, Bit Definition Tables.

ACR   AUXILIARY CONTROL REGISTER
[Main R3; read/write]

7	6	5	4	3	2	1	0
TST	TLD	TMC	BIC	rsv	COD	LOR	GIE

rsv . . . state is undefined at all times.

TST   — **Timer StarT** . . . When high, the timer is enabled and will count down from it's current value.

When low, timer is disabled. Timer is stopped by writing a 0 to [TST].

TLD   — **Timer LoaD** . . . When high, generates timer load pulse. Cleared when load complete.

TMC   — **TiMer Clock select** . . . Selects timer clock frequency. Should not be written when [TST] is high. Can be written at same time as [TST] and [TLD].

TMC	Timer Clock
0	(CPU-CLK)/16
1	(CPU-CLK)/2

BIC   — **Bi-directional Interrupt Control** . . . Controls direction of BIRQ.

BIC	BIRQ
0	Input
1	Output

COD   — **Clock Out Disable** . . . When high, CLK-OUT output is at TRI-STATE.

LOR   — **Lock Out Remote** . . . When high, a remote system is prevented from accessing the BCP.

GIE   — **Global Interrupt Enable** . . . When low, disables all maskable interrupts. When high, works with [IM4-0] to enable maskable interrupts.

2

6.0 Reference Section (Continued)

ATR   AUXILIARY TRANSCEIVER REGISTER
[Alternate R2; read/write]

7	6	5	4	3	2	1	0
AT7	AT6	AT5	AT4	AT3	AT2	AT1	AT0

AT7–0 — **Auxiliary Transceiver** ... In 5250 protocol modes, bits 2–0 define the receive station address, and bits 7–3 control the amount of time TX-ACT stays asserted after the last fill bit.

In 8-bit protocol modes, bits 7–0 define the receive station address.

For further information, see Section 3.0 Transceiver.

ATR 7–3	TX-ACT Hold Time (μs) (if TCLK = 8 MHz)
0 0 0 0 0	0
0 0 0 0 1	0.5
0 0 0 1 0	1.0
0 0 0 1 1	1.5
↓	↓
1 1 1 1 1	15.5

CCR   CONDITION CODE REGISTER
[Main R0; bits 0–3, 5–7 read/write, bit 4 read only]

7	6	5	4	3	2	1	0
TO	RR	RW	BIRQ	N	V	C	Z

TO   — **Time Out flag** ... Set high when timer counts to zero. Cleared by writing a 1 to this location or by stopping timer (by writing a 0 to [TST]).

RR   — **Remote Read** ... Set on the trailing edge of a $\overline{\text{REM-RD}}$ pulse, if $\overline{\text{RAE}}$ is asserted and {RIC} is pointing to Data Memory. Cleared by writing a 1 to this location.

RW   — **Remote Write** ... Set on the trailing edge of a $\overline{\text{REM-WR}}$ pulse, if $\overline{\text{RAE}}$ is asserted and {RIC} is pointing to Data Memory. Cleared by writing a 1 to this location.

BIRQ   — **Bi-directional Interrupt ReQuest** ... [Read only]. Reflects the logic level of the Bi-directional interrupt pin, $\overline{\text{BIRQ}}$. Updated at the beginning of each instruction cycle.

N   — **Negative** ... A high level indicates a negative result generated by an arithmetic, logical or shift instruction.

V   — **oVerflow** ... A high level indicates an overflow condition generated by an arithmetic instruction.

C   — **Carry** ... A high level indicates a carry or borrow generated by an arithmetic instruction. During a shift/rotate operation the state of the last bit shifted out appears in this location.

Z   — **Zero** ... A high level indicates a zero result generated by an arithmetic, logical or shift instruction. Further information: Section 2.2.1 ALU.

6.0 Reference Section (Continued)

DCR DEVICE CONTROL REGISTER
[Alternate R0; read/write]

7	6	5	4	3	2	1	0
CCS	TCS1	TCS0	IW1	IW0	DW2	DW1	DW0

CCS — **CPU Clock Select** ... Selects CPU clock frequency. OCLK represents the frequency of the on-chip oscillator, or the externally applied clock on input X1.

CCS	CPU CLK
0	OCLK
1	OCLK/2

TCS1,0 — **Transceiver Clock Select** ... Selects transceiver clock, TCLK, frequency.

OCLK represents the frequency of the on-chip oscillator, or the externally applied clock on input X1. X-TCLK is the external transceiver clock input.

TCS1,0	TCLK
0 0	OCLK
0 1	OCLK/2
1 0	OCLK/4
1 1	X-TCLK

IW1,0 — **Instruction memory Wait-state select** ... Selects from 0 to 3 wait states for accessing instruction memory.

DW2-0 — **Data memory Wait-state select** ... Selects from 0 to 7 wait states for accessing data memory.

DS DATA STACK
[Main R31; read/write]

7	6	5	4	3	2	1	0
DS7	DS6	DS5	DS4	DS3	DS2	DS1	DS0

DS7-0 — **Data Stack** ... Data stack input/output port. Stack is 16 bytes deep. Further information: Section 2.1.1.8 Stock Registers.

2

6.0 Reference Section (Continued)

ECR   ERROR CODE REGISTER
[Alternate R4 with [SEC] high; read only]

7	6	5	4	3	2	1	0
rsv	rsv	rsv	OVF	PAR	IES	LMBT	RDIS

rsv . . . state is undefined at all times.

OVF — **Receiver oVerFlow** . . . Set when the receiver has processed 3 words and another complete frame is received before the FIFO is read by the CPU. Cleared by reading {ECR} or by asserting [TRES].

PAR — **PARity error** . . . Set when bad (odd) overall word parity is detected in any receive frame. Cleared by reading {ECR} or by asserting [TRES].

IES — **Invalid Ending Sequence** . . . Set when the "mini-code violation" is not correct during a 3270, 3299, or 8-bit ending sequence. Cleared by reading {ECR} or by asserting [TRES].

LMBT — **Loss of Mid-Bit Transition** . . . Set when the expected Manchester Code mid-bit transition does not occur within the allowed window. Cleared by reading {ECR} or by asserting [TRES].

RDIS — **Receiver DISabled while active** . . . Set when transmitter is activated while receiver is active, without RPEN being asserted. Cleared by reading {ECR} or by asserting [TRES]. Further information: Section 3.2 Transceiver Functional Description.

FBR   FILL-BIT REGISTER
[Alternate R3; read/write]

7	6	5	4	3	2	1	0
FB7	FB6	FB5	FB4	FB3	FB2	FB1	FB0

FB7–0 — **Fill Bits** . . . 5250 fill-bit control. Further information: Section 3.0 Transceiver.

6.0 Reference Section (Continued)

IBR   INTERRUPT BASE REGISTER
[Alternate R1; read/write]

7	6	5	4	3	2	1	0
IV15	IV14	IV13	IV12	IV11	IV10	IV9	IV8
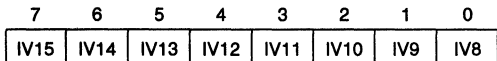
IV15–8— **Interrupt Vector** . . . High byte of interrupt and trap vectors. Further information: Section 2.2.3, Interrupts.
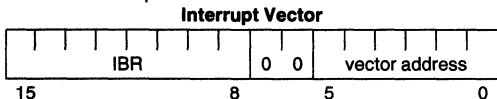
Interrupt Vector

IBR	0 0	vector address
15	8 5	0

The interrupt vector is obtained by concatenating {IBR} with the vector address:

Interrupt	Vector Address	Priority	
NMI	0 1 1 1 0 0	—	
Receiver	0 0 0 1 0 0	1	high
Transmitter	0 0 1 0 0 0	2	↑
Line Turn Around	0 0 1 1 0 0	3	
Bi-directional	0 1 0 0 0 0	4	↓
Timer	0 1 0 1 0 0	5	low

ICR   INTERRUPT CONTROL REGISTER
[Main R2; read/write]

7	6	5	4	3	2	1	0
RIS1	RIS0	rsv	IM4	IM3	IM2	IM1	IM0
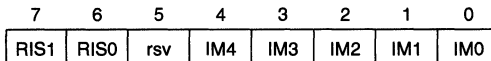
rsv . . . state is undefined at all times

RIS1,0 — **Receiver Interrupt Select** . . . Defines the source of the Receiver Interrupt.

RIS1,0	Interrupt Source
0 0	RFF + RE
0 1	DAV + RE
1 0	(unused)
1 1	RA

"+" indicates logical "or"

Further information: Section 3.2.3 Transceiver Interrupts.

IM4–0 — **Interrupt Masks** . . . Each bit, when set high, masks an interrupt. IM3 functions as an interrupt mask only if BIRQ is defined as an input. When BIRQ is defined as an output, IM3 controls the state of BIRQ.

IM4–0	Interrupt
0 0 0 0 0	No Mask
X X X X 1	Receiver
X X X 1 X	Transmitter
X X 1 X X	Line Turn-Around
X 1 X X X	Bi-Directional
1 X X X X	Timer

Further information: Section 2.2.3 Interrupts.

2

6.0 Reference Section (Continued)

ISP INTERNAL STACK POINTER
[Main R30; read/write]

7	6	5	4	3	2	1	0
ASP3	ASP2	ASP1	ASP0	DSP3	DSP2	DSP1	DSP0

ASP3–0 — **Address Stack Pointer** . . . Input/output port of the address stack pointer. Further information: Section 2.1.1.8 Stack Registers.

DSP3–0 — **Data Stack Pointer** . . . Input/output port of the data stack pointer. Further information: Section 2.1.1.8 Stack Registers.

NCF NETWORK COMMAND FLAG REGISTER
[Main R1; read only]

7	6	5	4	3	2	1	0
TFE	RFF	LA	LTA	DEME	RAR	ACK	POLL

TFE — **Transmit FIFO Empty** . . . Set high when the FIFO is empty. Cleared by writing to {RTR}.

RFF — **Receive FIFO Full** . . . Set high when the Receive FIFO contains 3 received words. Cleared by reading to {RTR}.

LA — **Line Active** . . . Indicates activity on the receiver input. Set high on any transition; cleared after detecting no input transitions for 16 TCLK periods.

LTA — **Line Turn Around** . . . Set high when end of message is received. Cleared by writing to {RTR}, writing a "1" to this location, or by asserting [TRES].

DEME — **Data Error or Message End** . . . In 3270 & 3299 modes, asserted when a byte parity error is detected. In 5250 modes, asserted when the [111] station address is decoded and [DAV] is asserted. Cleared by reading {RTR}. Undefined in 8-bit modes and in the first frame of 3299 modes.

RAR — **Received Auto-Response** . . . Set high when a 3270 Auto-Response message is decoded and [DAV] is asserted. Cleared by reading {RTR}. Undefined in 5250 and 8-bit modes and in the first frame of 3299 modes.

ACK — **Poll/ACKnowledge** . . . Set high when a 3270 poll/ack command is decoded and [DAV] is asserted. Cleared by reading {RTR}. Undefined in 5250 and 8-bit modes and in the first frame of 3299 modes.

POLL — **POLL** . . . Set high when a 3270 poll command is decoded and [DAV] is asserted. Cleared by reading {RTR}. Undefined in 5250 and 8-bit modes and in the first frame of 3299 modes. Further information: Section 3.0 Transceiver.

6.0 Reference Section (Continued)

RTR   RECEIVE/TRANSMIT REGISTER
[Alternate R4; read/write]

7	6	5	4	3	2	1	0
RTF7	RTF6	RTF5	RTF4	RTF3	RTF2	RTF1	RTF0

RTF7–0 — **Receive Transmit FIFO's** . . . Input/output port to the least significant eight bits of receive and transmit FIFO's. [OWP], [TF10–8] and [RTF7–0] are pushed onto the transmit FIFO on moves into {RTR}. [RF10–8] and [RTF7–0] are popped from receiver FIFO on moves out of {RTR}. Further information: Section 3.0 Transceiver.

TCR   TRANSCEIVER COMMAND REGISTER
[Alternate R6; read/write]

7	6	5	4	3	2	1	0
RLQ	SEC	SLR	ATA	OWP	TF10	TF9	TF8

RLQ — **Receive Line Quiesce** . . . Selects number of line quiesce bits the receiver looks for.

RLQ	Number of Quiesces
0	2
1	3

SEC — **Select Error Codes** . . . When high {ECR} is switched into {RTR} location.

SLR — **Select Line Receiver** . . . Selects the receiver input source.

SLR	Source
0	DATA-IN
1	On-chip analog line receiver

ATA — **Advance Transmitter Active** . . . When high, TX-ACT is advanced one half bit time so that the transmitter can generate 5.5 line quiesce pulses.

OWP — **Odd Word Parity** . . . Controls transmitter word parity.

OWP	Word Parity
0	Even
1	Odd

TF10–8 — **Transmit FIFO** . . . [OWP], [TF10–8] and [RTF7–0] are pushed onto transmit FIFO on moves into {RTR}.

Further information: Section 3.0 Transceiver.

2

TMR   TRANSCEIVER MODE REGISTER
[Alternate R7; read/write]

7	6	5	4	3	2	1	0
TRES	LOOP	RPEN	RIN	TIN	PS2	PS1	PS0

TRES — **Transceiver RESet** ... Resets transceiver when high. Transceiver can also be reset by $\overline{\text{RESET}}$, without affecting [TRES].

LOOP — **Internal LOOP-back** ... When high, TX-ACT is disabled (held at 0) and transmitter serial data is internally directed to the receiver serial data input.

RPEN — **RePeat ENable** ... When high, the receiver can be active at the same time as the transmitter.

RIN — **Receiver INvert** ... When high, the receiver serial data is inverted.

TIN — **Transmitter INvert** ... When high the transmitter serial data outputs are inverted.

PS2–0 — **Protocol Select** ... Selects protocol for both transmitter and receiver.

PS2–0	Protocol
0 0 0	3270
0 0 1	3299 multiplexer
0 1 0	3299 controller
0 1 1	3299 repeater
1 0 0	5250
1 0 1	5250 promiscuous
1 1 0	8-bit
1 1 1	8-bit promiscuous

Further information: Section 3.0 Transceiver.

TRH   TIMER REGISTER — HIGH
[Main R29; read/write]

7	6	5	4	3	2	1	0
TM15	TM14	TM13	TM12	TM11	TM10	TM9	TM8

TM15–8 — **TiMer** ... Input/output port of high byte of timer. Further information: Section 2.1.1.4 Timer Registers.

6.0 Reference Section (Continued)

TRL   TIMER REGISTER—LOW
[Main R28; read/write]

7	6	5	4	3	2	1	0
TM7	TM6	TM5	TM4	TM3	TM2	TM1	TM0

TM7–0— **TiMer** . . . Input/output port of low byte of timer. Further information: Section 2.1.1.4 Timer Registers.

TSR   TRANSCEIVER STATUS REGISTER
[Alternate R5; read only]

7	6	5	4	3	2	1	0
TFF	TA	RE	RA	DAV	RF10	RF9	RF8

TFF — **Transmit FIFO Full** . . . Set high when the transmit FIFO is full. {RTR} must not be written to when [TFF] is high.

TA — **Transmitter Active** . . . Reflects the state of TX-ACT, indicating that data is being transmitted. Unlike TX-ACT, however, [TA] is not disabled by [LOOP].

RE — **Receiver Error** . . . Set high when a receiver error is detected. Cleared by reading {ECR} or by asserting [TRES].

RA — **Receiver Active** . . . Set high when a valid starting sequence is received. Cleared when either an end of message or an error is detected. In 5250 modes, [RA] is cleared at the same time as [LA].

DAV — **Data AVailable** . . . Set high when valid data is available in {RTR} and {TSR}. Cleared by reading {RTR}, or when an error is detected.

RF10–8— **Receive FIFO** . . . [RF10–8] and [RTF7–0] reflect the state of the top word of the receive FIFO.

Further information: Section 3.0 Transceiver.

2

6.0 Reference Section (Continued)

6.2.3 Bit Definition Tables

The following tables describe the location and function of all control and status bits in the various BCP addressable special function registers. The Remote Interface Configuration register, {RIC}, which is addressable only by a remote processor is not included.

6.2.3.1 Processor

	Bit	Name	Location	Reset State	Function
Timing/ Control	CCS	CPU Clock Select	DCR [7]	1	Selects CPU clock frequency. **CCS | CPU CLK** 0 | OCLK 1 | OCLK/2 Where OCLK is the frequency of the on-chip oscillator, or the externally applied clock on input X1.
	DW2–0	Data memory Wait-state select	DCR [2–0]	111	Selects from 0 to 7 wait states for accessing data memory.
	IW1,0	Instruction memory Wait-state select	DCR [4,3]	11	Selects from 0 to 3 wait states for accessing instruction memory.
	COD	Clock Out Disable	ACR [2]	0	When high, CLK-OUT is at TRI-STATE.
Remote Interface	LOR*	Lock Out Remote	ACR [1]	0	When high, a remote processor is prevented from accessing the BCP or its memory.
	RR*	Remote Read	CCR [6]	0	Set on the trailing edge of a $\overline{\text{REM-RD}}$ pulse, if $\overline{\text{RAE}}$ is asserted and {RIC} is pointing to Data Memory. Cleared by writing a 1 to [RR].
	RW*	Remote Write	CCR [5]	0	Set on the trailing edge of a $\overline{\text{REM-WR}}$ pulse, if $\overline{\text{RAE}}$ is asserted and {RIC} is pointing to Data Memory. Cleared by writing a 1 to [RW].
Interrupt Control	BIC	Bi-directional Interrupt Control	ACR [4]	0	Controls the direction of $\overline{\text{BIRQ}}$. **BIC | $\overline{\text{BIRQ}}$** 0 | Input 1 | Output
	BIRQ	Bi-directional Interrupt ReQuest	CCR [4]	X	[Read Only]. Reflects the logic level of the $\overline{\text{BIRQ}}$ input. Updated at the beginning of each instruction cycle.
	GIE	Global Interrupt Enable	ACR [0]	0	When low, disables all maskable interrupts. When high, works with [IM4–0] to enable maskable interrupts.
	IM4–0	Interrupt Mask select	ICR [4–0]	11111	Each bit, when set high, masks an interrupt. **IM4–0 | Interrupt | Priority** 0 0 0 0 0 | No Mask | — X X X X 1 | Receiver | 1 High X X X 1 X | Transmitter | 2 ↑ X X 1 X X | Line Turn-Around | 3 X 1 X X X | Bi-Directional | 4 ↓ 1 X X X X | Timer | 5 Low IM3 functions as an interrupt mask only when $\overline{\text{BIRQ}}$ is defined as an input. When $\overline{\text{BIRQ}}$ is defined as an output, IM3 controls the state of $\overline{\text{BIRQ}}$.

*These bits represent the only visibility and control that the processor has into the operation of the remote interface controller. The Remote Interface Configuration register, {RIC}, accessible only by a remote processor, provides further control functions. See Remote Interface section for more information.

6.0 Reference Section (Continued)

6.2.3 Bit Definition Tables (Continued)

The following tables describe the location and function of all control and status bits in the various BCP addressable special function registers. The Remote Interface Configuration register, {RIC}, which is addressable only by a remote processor is not included.

6.2.3.1 Processor (Continued)

	Bit	Name	Location	Reset State	Function
Interrupt Control (Continued)	IV15–8	Interrupt Vector	IBR [7–0]	0000 0000	High byte of interrupt and trap vectors. The interrupt vector is obtained by concatenating {IBR} with the vector address: **Interrupt** / **Vector Address** $\overline{\text{NMI}}$ — 0 1 1 1 0 0 Receiver — 0 0 0 1 0 0 Transmitter — 0 0 1 0 0 0 Line Turn Around — 0 0 1 1 0 0 Bi-Directional — 0 1 0 0 0 0 Timer — 0 1 0 1 0 0 **Interrupt Vector** [IBR | 0 0 | vector address] 15 8 5 0
	RIS1,0	Receiver Interrupt Select	ICR [7,6]	11	Defines the source of the receiver interrupt. **RIS1,0** / **Interrupt Source** 0 0 — RFF + RE 0 1 — DAV + RE 1 0 — (unused) 1 1 — RA
Address and Data Stacks	ASP3–0	Address Stack Pointer	ISP [7–4]	0000	Address stack pointer. Writing to this location changes the value of the pointer.
	DSP3–0	Data Stack Pointer	ISP [3–0]	0000	Data stack pointer. Writing to this location changes the value of the pointer.
	DS7–0	Data Stack	DS [7–0]	XXXX XXXX	Data Stack Input/Output port. Stack is 16 bytes deep.
Arithmetic Flags	C	Carry	CCR [1]	0	A high level indicates a carry or borrow, generated by an arithmetic instruction. During a shift/rotate operation the state of the last bit shifted out appears in this location.
	N	Negative	CCR [3]	0	A high level indicates a negative result generated by an arithmetic, logical, or shift instruction.
	V	oVerflow	CCR [2]	0	A high level indicates an overflow condition, generated by an arithmetic instruction.
	Z	Zero	CCR [0]	0	A high level indicates a zero result generated by an arithmetic, logical, or shift instruction.

2

6.0 Reference Section (Continued)

6.2.3. Bit Definition Tables (Continued)

The following tables describe the location and function of all control and status bits in the various BCP addressable special function registers. The Remote Interface Configuration register, {RIC}, which is addressable only by a remote processor is not included.

6.2.3.1 Processor (Continued)

	Bit	Name	Location	Reset State	Function
Timer	TLD	**Timer LoaD**	ACR [6]	0	Set high to load timer. Cleared automatically when load complete.
	TM15–8	**TiMer**	TRH [7–0]	XXXX XXXX	Input/output port of high byte of timer.
	TM7–0	**TiMer**	TRL [7–0]	XXXX XXXX	Input/output port of low byte of timer.
	TMC	**Timer Clock select**	ACR [5]	0	Selects timer clock frequency. Must not be written when [TST] high. Can be written at same time as [TST] and [TLD]. **TMC** | **Timer Clock** 0 | CPU-CLK/16 1 | CPU-CLK/2
	TO	**Time Out flag**	CCR [7]	0	Set high when timer counts down to zero. Cleared by writing a 1 to [TO] or by stopping the timer (by writing a 0 to [TST]).
	TST	**Timer StarT**	ACR [7]	0	When high, timer is enabled and will count down from its current value. Timer is stopped by writing a 0 to this location.

6.2.3.2 Transceiver

Table includes control and status bits only. It does not include definitions of bit fields provided for the formatting (de-formatting) of data frames. For further information see the Transceiver section.

	Bit	Name	Location	Reset State	Function
Transceiver Control	LOOP	internal **LOOP**-back	TMR [6]	0	When high, TX-ACT is disabled (held at 0) and transmitter serial data is internally directed to the receiver serial data input.
	PS2–0	**Protocol Select**	TMR [2–0]	000	Selects protocol for both transmitter and receiver. **PS2–0** | **Protocol** 0 0 0 | 3270 0 0 1 | 3299 Multiplexer 0 1 0 | 3299 Controller 0 1 1 | 3299 Repeater 1 0 0 | 5250 1 0 1 | 5250 Promiscuous 1 1 0 | 8-bit 1 1 1 | 8-bit Promiscuous
	RTF7–0	**Receive/Transmit FIFOs**	RTR [7–0]	XXXX XXXX	Input/output port of the least significant 8 bits of receive and transmit FIFOs. [OWP], [TF10–8] and [RTF7–0] are pushed onto the transmit FIFO on moves to {RTR}. [RF10–8] and [RTF7–0] are popped from receive FIFO on moves from {RTR}.

6.0 Reference Section (Continued)

6.2.3 Bit Definition Tables (Continued)

6.2.3.2 Transceiver (Continued)

Table includes control and status bits only. It does not include definitions of bit fields provided for the formatting (de-formatting) data frames. For further information see the Transceiver section.

	Bit	Name	Location	Reset State	Function
Transceiver Control (Continued)	TCS1,0	Transceiver Clock Select	DCR [6,5]	10	Selects transceiver clock, TCLK, source. **TCS1,0 / TCLK** 0 0 / OCLK 0 1 / OCLK/2 1 0 / OCLK/4 1 1 / X-TCLK OCLK is the frequency of the on-chip oscillator, or the externally applied clock on input X1. X-TCLK is the external transceiver clock input.
	TRES	Transceiver RESet	TMR [7]	0	Resets transceiver when high. Transceiver can also be reset by $\overline{\text{RESET}}$, without affecting [TRES].
Transmitter Control	ATA	Advance Transmitter Active	TCR [4]	0	When high, TX-ACT is advanced one half bit time so that the transmitter can generate 5.5 line quiesce pulses.
	AT7–3	Auxiliary Transceiver control	ATR [7–3]	XXXXX	In 5250 modes. Controls the time TX-ACT is held after the last fill bit. **AT7–3 / TX-ACT Hold Time (μs) (If TCLK = 8 MHz)** 00000 / 0 00001 / 0.5 00010 / 1 ↓ / ↓ 11111 / 15.5
	FB7–0	Fill Bit select	FBR [7–0]	XXXX XXXX	The value in this register contains the 1's complement of the number of additional 5250 fill bits selected.
	OWP	Odd Word Parity	TCR [3]	0	Controls transmitter word parity. **OWP / Word Parity** 0 / Even 1 / Odd
	TF10–8	Transmit FIFO	TCR [2–0]	000	[OWP], [TF10–8] and [RTF7–0] are pushed onto the transmit FIFO on moves to {RTR}.
	TIN	Transmitter INvert	TMR [3]	0	When high, the transmitter serial data outputs are inverted.
Receiver Control	AT7–0	Auxiliary Transceiver control	ATR [7–0]	XXXX XXXX	In 5250 modes, [AT2–0] contains the station address. In 8-bit modes, [AT7–0] contains the station address.
	RF10–8	Receive FIFO	TSR [2–0]	XXX	Reflects the state of the most significant 3 bits in the top location of the receive FIFO.
	RIN	Receiver INvert	TMR [4]	0	When high, the receiver serial data is inverted.
	RLQ	Receive Line Quiesce	TCR [7]	1	Selects number of line quiesce bits the receiver requires before it will indicate receipt of a valid start sequence. **RLQ / Number of Line Quiesce Pulses** 0 / 2 1 / 3
	RPEN	RePeat ENable	TMR [5]	0	When high, the receiver can be active at the same time as the transmitter.
	SEC	Select Error Codes	TCR [6]	0	When high, {ECR} is switched into {RTR} location.

2

6.0 Reference Section (Continued)

6.2.3 Bit Definition Tables (Continued)

6.2.3.2 Transceiver (Continued)

Table includes control and status bits only. It does not include definitions of bit fields provided for the formatting (de-formatting) data frames. For further information see the Transceiver section.

	Bit	Name	Location	Reset State	Function
Receiver Control (Continued)	SLR	Select Line Receiver	TCR [5]	0	Selects the receiver input source. SLR | Source 0 | DATA-IN 1 | On-Chip Analog Line Receiver
Transmitter Status	TA	Transmitter Active	TSR [6]	0	Reflects the state of TX-ACT, indicating that data is being transmitted. Is not disabled by [LOOP].
	TFE	Transmit FIFO Empty	NCF [7]	1	Set high when the FIFO is empty. Cleared by writing to {RTR}.
	TFF	Transmit FIFO Full	TSR [7]	0	Set high when the FIFO is full. {RTR} must not be written when [TFF] is high.
Receiver Status	ACK	poll/ ACKnowledge	NCF [1]	0	Set high when a 3270 poll/ack command is decoded and [DAV] is asserted. Cleared by reading {RTR}. Undefined in 5250 and 8-bit modes and in the first frame of 3299 modes.
	DAV	Data AVailable	TSR [3]	0	Set high when valid data is available in {RTR} and {TSR}. Cleared by reading {RTR}, or when an error is detected.
	DEME	Data Error or Message End	NCF [3]	0	In 3270 or 3299 modes, asserted when a byte parity error is detected. In 5250 modes, asserted when the [111] station address is decoded and [DAV] is asserted. Undefined in 8-bit modes and first frame of 3299 modes.
	LA	Line Active	NCF [5]	0	Indicates activity on the receiver input. Set high on any transition; cleared after no input transitions are detected for 16 TCLK periods.
	LTA	Line Turn Around	NCF [4]	0	Set high when an end of message is detected. Cleared by writing to {RTR}, writing a "1" to [LTA] or by asserting [TRES].
	POLL	POLL	NCF [0]	0	Set high when a 3270 Poll command is decoded and [DAV] is asserted. Cleared by reading {RTR}. Undefined in 5250 and 8-bit modes and in the first frame of 3299 modes.
	RA	Receiver Active	TSR [4]	0	Set high when a valid start sequence is received. Cleared when either an end of message or an error is detected.
	RAR	Received Auto-Response	NCF [2]	0	Set high when a 3270 Auto-Response message is decoded and [DAV] is asserted. Cleared by reading {RTR}. Undefined in 5250 and 8-bit modes and in the first frame of 3299 modes.
	RE	Receiver Error	TSR [5]	0	Set high when an error is detected. Cleared by reading {ECR} or by asserting [TRES].
	RFF	Receive FIFO Full	NCF [6]	0	Set high when the receive FIFO contains 3 received words. Cleared by reading {RTR}.

6.0 Reference Section (Continued)

6.2.3 Bit Definition Tables (Continued)

6.2.3.2 Transceiver (Continued)

Table includes control and status bits only. It does not include definitions of bit fields provided for the formatting (de-formatting) data frames. For further information see the Transceiver section.

	Bit	Name	Location	Reset State	Function
Receiver Error Codes	IES	Invalid Ending Sequence	ECR [2]	0	Set when the first mini-code violation is not correct during a 3270, 3299 or 8-bit ending sequence. Cleared by reading {ECR} or asserting [TRES].
	LMBT	Loss of Mid-Bit Transition	ECR [1]	0	Set when the expected Manchester Code mid-bit transition does not occur within the allowed window. Cleared by reading {ECR} or by asserting [TRES].
	OVF	receiver OVerFlow	ECR [4]	0	Set when the receiver has processed 3 words and another complete frame is received before the FIFO is read by the CPU. Cleared by reading {ECR} or asserting [TRES].
	PAR	PARity error	ECR [3]	0	Set when bad (odd) overall word parity is detected in any receive frame. Cleared by reading {ECR} or asserting [TRES].
	RDIS	Receiver DISabled while active	ECR [0]	0	Set when transmitter is activated by writing to {RTR} while receiver is still active, without [RPEN] first being asserted. Cleared by reading {ECR} or asserting [TRES].

6.3 REMOTE INTERFACE CONFIGURATION REGISTER

This register can be accessed only by the remote system. To do this, CMD and $\overline{RAE}$ must be asserted and the [LOR] bit in the {ACR} register must be low.

7	6	5	4	3	2	1	0	
BIS	SS	FW	LR	LW	STRT	MS1	MS0	RIC

BIS Bidirectional Interrupt Status . . . Mirrors the state of IM3 ({ICR} bit 3), enabling the remote system to poll and determine the status of the $\overline{BIRQ}$ I/O. When $\overline{BIRQ}$ is an output, the remote system can change the state of this output by writing a one to BIS. This can be used as an interrupt acknowledge, whenever $\overline{BIRQ}$ is used as a remote interrupt.

SS Single-Step . . . Writing a 1 with STRT low, the BCP will single-step by executing the current instruction and advancing the PC. On power up/reset this bit is low.

FW Fast Write . . . When high, with LW low, selects fast write mode for the buffered interface. When low selects slow write mode. On power up/reset this bit is low (LW will also be low, so buffered write mode is selected).

LR Latched Read . . . When high selects latched read mode, when low selects buffered read mode. On power up/reset this bit is low.

LW Latched Write . . . When high selects latched write mode, when low selects buffered write mode. On power up/reset this bit is low (FW will also be low, so slow buffered write mode is selected).

STRT STaRT . . . The remote system can start and stop the BCP using this bit. On power-up/reset this bit is low (BCP stopped). When set, the BCP begins executing at the current Program Counter address. When cleared, the BCP finishes executing the current instruction, then halts to an idle mode.

In some applications, where there is no remote system, or the remote system is not an intelligent device, it may be desirable to have the BCP power-up/reset running rather than stopped at address 0000H. This can be accomplished by asserting $\overline{REM-RD}$, $\overline{REM-WR}$ and $\overline{RESET}$, with $\overline{RAE}$ de-asserted.

MS1,0 Memory Select 1,0 . . . These two bits determine what the remote system is accessing in the BCP system, according to the following table:

MS1	MS0	Selected Function
0	0	Data Memory
0	1	Instruction Memory
1	0	Program Counter (Low Byte)
1	1	Program Counter (High Byte)

The BCP must be idle for the remote system to read/write Instruction memory or the Program Counter.

All remote accesses are treated the same (independent of where the access is directed using MS0 and MS1), as defined by the configuration bits LW, LR, FW.

If the remote system and the BCP request data memory access simultaneously, the BCP will win first access. If the locks ([LOR], $\overline{LOCK}$) are not set, the remote system and BCP will alternate access cycles thereafter.

On power-up/reset, MS1,0 points to instruction memory.

2

6.0 Reference Section (Continued)

6.4 DEVELOPMENT TOOLS

National Semiconductor provides tools specifically created for the development of products that use the DP8344. These tools consist of the DP8344 BCP Assembler System, the DP8344 BCP Demonstration/Development Kit, and the DP8344 BCP Multi-Protocol Adapter (MPA) Design/Evaluation Kit.

6.4.1 Assembler System

The Assembler System is an MS-DOS compatible program used to translate the DP8344's instruction set into a directly executable machine language. The system contains a macro cross assembler, link editor and librarian. The macro cross assembler provides nested macro definitions and expansions, to automate common instruction sequences, and source file inclusion nested conditional assembly, which allows the assembler to make intelligent decisions concerning instruction sequence based on user directives. The linker allows relocatable object sections to be combined in any desired order. It can also generate a load map which details each section's contribution to the linked module. The librarian allows for the creation of libraries from frequently accessed object modules.

6.4.2 Demonstration/Development Kit

The Demonstration/Development kit is a cost effective development tool that performs functions similar to an in-circuit emulator. The kit, developed by Capstone Technology, Inc., Fremont, California, consists of a DP8344 based development board, a monitor/debugger software package, National Semiconductor's DP8344 video training tapes, and all required documentation. The development board is a full size PC card that contains a 22 square inch area for logic prototype wiring. The monitor/debugger program displays internal register contents and status information. It also provides functions such as execution break points and single stepping.

6.4.3 Multi-Protocol Adapter (MPA)
Design/Evaluation Kit

The Multi-Protocol Adapter (MPA) is a PC expansion card that emulates a 3270 or 5250 display terminal and supports industry standard PC emulation software. The MPA comes in a design/evaluation kit that includes the hardware, schematics and PAL equations, and software including all the DP8344 source code. This kit was produced to provide a blueprint for PC emulation products and a cornerstone for all 3270 and 5250 product development using the DP8344. The code was developed in a modular fashion so it can be adapted to any 3270 or 5250 application.

6.5 THIRD PARTY SUPPLIERS

The following section is intended to make the DP8344 Customer aware of products, supplied by companies other than National Semiconductor, that are available for use in developing DP8344 systems. While National Semiconductor has supported these ventures and has become familiar with many of these products, we do not provide technical support, or in any way guarantee the functionality of these products.

6.5.1 Crystal Supplier

The recommended crystal parameters for operation with the DP8344 are given in Section 2.2.4. Any crystal meeting these specifications will work correctly with the DP8344. NEL Frequency Controls, Inc., Burlington, Wisconsin, has developed crystals, the NEL C2570N and NEL C2571N, specifically for the DP8344 which meet these specifications. The C2570N and C2571N are both 18.8696 MHz fundamental mode AT cut quartz crystals. The C2571N has a hold down pin for case ground and a third mechanical tie down. NEL Frequency Controls, Inc. is located at:

NEL Frequency Controls, Inc.
357 Beloit Street
Burlington, Wisconsin 53105
(414) 763-3591

6.5.2 System Development Tools

The DP8344, with its higher level of integration and processing power, has opened the IBM mainframe connectivity market to a wider range of product manufacturers, who until now found the initial cost and time to market prohibitive. This wider base of manufacturers created the opportunity for a more extensive line of development tools that dealt not only with the use of the DP8344 but also with the implementation of the 3270 and 5250 protocols. While National Semiconductor is dedicated to providing the Customer with the proper tools in both areas, we also have aided and encouraged a number of third party suppliers to offer additional development tools. This has further provided an avenue for faster and more reliable product development in this product area. The development tools discussed in this section are controller emulators and line monitors for the IBM 3270/3299 and 5250 protocols.

A controller emulator is a device that emulates an IBM 3x74 cluster controller or a System 3x controller. With the DP8344 both of these controllers can be emulated with the same piece of hardware. The controller emulator allows the designer to issue individual commands or sequences of commands to a peripheral. This is very useful in characterizing existing equipment and testing of products under development. Capstone Technology offers such a product. Their Extended Interactive Controller, part #CT-109, is a single PC expansion card that can emulate both 3270 and 5250 control devices (the 3x74 and System 3X, respectively). Newleaf Technologies, Ltd., Cobham, Surrey, England, and Azure Technology, Inc., Franklin, Mass., also supply products in this area. Newleaf Technology offers the COLT52, a twinax controller emulator, and Azure Technology offers a controller made with their CoaxScope and TwinaxScope line monitors.

A line monitor is a device that monitors all the activity on the coax or twinax cable. The activity includes both the commands from the controller and the responses from the peripheral. These devices typically decode the commands and present them in an easy to read format. The individual transmissions are time stamped to provide the designer with response time information. The line monitors are very useful in characterizing communications traffic and in determining the source of problems during development or in the field. Azure Technology offers both a 3270/3299 (Coax) and 5250 (Twinax) line monitor. Their Coax Scope and Twinax Scope are single PC expansion cards that can record, decode and display activity on the 3270 coax and 5250 twinax

6.0 Reference Section (Continued)

line respectively. These devices also allow the play back of the recorded controller information. Capstone Technology also supplies a line monitor. The CT101C, Network Analysis Monitor (NAM), is a coax line monitor.

These companies can be contacted at the following locations:

Azure Technology, Inc.
38 Pond Street
Franklin, Massachusettes 02038
(508) 520-3800

Capstone Technology
853 Brown Rd., Suite 207
Fremont, California 94539
(415) 657-3901

New Leaf Technology, Ltd.
24A High Street
Cobham
Surrey
KT113EB
ENGLAND
(0932) 66466

The North American distributor for Newleaf Technology is GTI, Ltd.

GTI, Ltd.
One Huntington Quad
Suite 2C14
Melville, New York 11747
(516) 420-1590

6.6 GLOSSARY

3270—An IBM **communication protocol** originally developed for the 370 class mainframe that implements a star topology using a single **coax** cable per slave device. In this master-slave protocol, all communication is initiated by the **controller** (master) and responses are returned by the terminal or other attached device (slave). The data is transmitted using **biphase encoding** at a bit rate of 2.3587 MHz.

3299—A **communications protocol** that is the **3270** protocol with an eight bit address frame added to the beginning of each controller transmission between the **start sequence** and the first coax word. Currently, IBM only uses three bits of the address field which allows up to eight devices to communicate with the **controller** through a **multiplexer**.

5250—An IBM **communications protocol** originally developed for the Series 3 that became widely used on the System 34/36/38 family of minicomputers and currently the AS/400. It uses a **multidrop** bus topology on **twin-ax** cable. This protocol is a master-slave type. The data is transmitted using **bi-phase encoding** at a bit rate of 1 MHz.

accumulator—The implied source register of one operand for some arithmetic operations. In the **BCP**, R8 in the currently enabled bank acts as the accumulator.

ALU—The Arithmetic Logic Unit, a component of the CPU that performs all arithmetic (addition and subtraction), logical (AND, OR, XOR, compare, bit test, and complement), rotational, and shifting operations.

ALU flags—Bits that indicate the result of certain ALU functions.

banked registers—Two or more sets of CPU registers that occupy the same register space, but only one of which is accessible at a time.

barrel shifter—Dedicated hardware for shifting and rotating.

BCP—An abbreviation for Biphase Communications Processor, the National Semiconductor DP8344.

bi-phase—In this communications signal encoding technique, the data is divided into discrete bit time intervals denoted by a transition in the center of the bit time. This technique combines the clock and data information into one transmission. In **3270** and **3299** protocols, a **mid-bit** transition from low to high represents a bi-phase 1, and a **mid-bit** transition from high to low represents a bi-phase 0. For the **5250** protocol, the definition of biphase logic levels is reversed. Biphase encoding is also called **Manchester II encoding.**

BIRQ—The Bidirectional Interrupt ReQuest. Without any other notation, BIRQ will refer to the BIRQ interrupt itself. BIRQ with a bar on top of it ($\overline{\text{BIRQ}}$) is used where the pin is referenced. BIRQ in brackets ([BIRQ]) is bit 4 in the {CCR} register.

coax—(1) RG-62A/U 93Ω coaxial cable that is used in **3270** protocol systems. (2) Sometimes, this term is used to refer to the **3270** protocol itself.

code violation—A violation of the **bi-phase** encoding format that is part of the **start sequence**. In **3270, 3299,** and the **general purpose 8-bit mode**, the code violation is 1½ bit times low and then 1½ bit times high. In the **5250** protocol, the signal levels are reversed.

communications protocol—A set of rules which defines the physical, electrical, control, and formatting specifications required to successfully transfer data between two systems.

context switch—Switching between two theoretically independent functions that should not affect each other except under specified circumstances.

controller—The master device that initiates all communication to the slave device and controls the manner in which the slave presents the information. It acts as the interface, both physically and logically, between the slave terminals and printers and a host processor.

CPU-CLK—The clock that the operation of the **BCP's** CPU is synchronized to. The period of this clock which defines **T-state** boundaries is either that of **OCLK** or one-half of **OCLK** depending on the configuration of the **BCP**. The timer clock is also derived from CPU-CLK.

CUT—Control Unit Terminal. A mode of the **controller** where attached devices have limited intelligence and are perceived to be hardware extensions of the **controller**. The **controller** directs all printer, screen, and keyboard activity.

DFT—Distributed Function Terminal. A **controller** mode that supports multiple logical terminals in the same device. The **controller** communicates in higher level commands via data placed in the buffer. The slave device has a greater amount of intelligence than the **CUT** mode device and is responsible for the terminal operation.

direct coupled—The connection of the **transceiver** to the transmission cable in a manner that does not isolate it from DC voltages. Contrast this with **transformer coupled.**

2

6.0 Reference Section (Continued)

dual port memory—A memory architecture that allows two different processors to access the same memory range alternately.

ending sequence—A defined sequence of bits signifying the end of a transmission. In **3270** and **3299**, it consists of a **bi-phase** 0 followed by a low to high transition on the bit time boundary and two **mini-code violations.**

FIFO—A section of memory or, as in the case of the **BCP** transceiver, a set of registers that are accessed in a First-In First-Out method. In other words, the first data placed in the FIFO by a write will be the first data removed by a read.

fill bits—Fill bits are **bi-phase** 0's used only in the **5250** protocol. A minimum of three fill bits are required between each frame of a **multi-frame message.** This number may be increased by the controller to approximately 243 per the SetMode command. There are always only three fill bits after the last frame of the transmission.

general purpose 8-bit mode—A generic communications mode similar to 3270 and 5250 frame formatting using 8-bit serial data and **bi-phase** signal encoding. The **BCP** supports both **promiscuous** and **non-promiscuous** modes.

Harvard architecture—A computer architecture where the instruction and data memory are organized into two independent memory banks, each with their own address and data buses.

hold time—The amount of time the line is driven at the end of **5250** transmissions to suppress noise on the cabling system.

ICLK—The clock that identifies the start of each instruction when it rises and indicates when the next instruction address is valid when it falls.

Immediate addressing mode—An addressing method where one operand, the data for Move instructions and the address for Jump instructions, is contained in the instruction itself.

Immediate-relative addressing mode—An addressing method that adds an unsigned 8-bit immediate number to the index register IZ to form the data memory address of an operand.

Indexed addressing mode—An addressing method that uses the contents of an index register as the data memory address for one of the operands in an instruction.

Interrupt latency—The time from when an interrupt first occurs until it begins executing at its interrupt vector.

jitter—Timing variations for signals of different harmonic content that move the edges of a transmitted signal in time causing uncertainty in their decoding.

jitter tolerance—The total amount of time an edge of a transmitted bit may move and still have its data bit decoded correctly.

LIFO—A sequence of registers or memory locations that are accessed in a Last-In First-Out method; in other words, the last data written into the LIFO will be the first to be removed by a read. Also known as a **stack.**

limited register set—In the **BCP**, the first 16 register address locations (R0–R11 in both banks and R12–R15) that can be used in all instructions.

line hold—The act of driving the transmission line during **5250** transmissions at the end of a message to allow the receivers to unsync. This insures that the receivers will not see line noise as the start of another frame when the line floats.

line interface—All the circuitry between the **BCP** and the communications cable medium.

line reflection—Energy from a transmission that is not absorbed by a load impedance and can cause interference in that signal.

Manchester II encoding—See **bi-phase** encoding.

mask—(1) A mechanism that allows the program to specify whether interrupts will be accepted by the CPU. (2) To disable the accepting of an interrupt by the CPU.

mid-bit—In **bi-phase** encoding, the transition in the center of a bit time.

mini-code violation—A violation of the **bi-phase** encoding format that is part of the **ending sequence** in 3270, 3299, and the **general purpose 8-bit mode.** The mini-code violation has no **mid-bit** transition being high for the entire bit time. There is no mini-code violation in **5250.**

multidrop—A communication method where all the slave devices are attached to the same cable and respond to **controller** commands and data only when their own address frame precedes the transmitted frame.

multi-frame message—Several bytes of data together in the same uninterrupted message that have only one **start sequence** and one **ending sequence.**

multiplexer—A device that receives 3299 protocol transmissions from a **controller**, strips off the address field, and determines over which of eight ports to transmit the message in 3270 format. The device then directs the response from the terminal back to the **controller.**

non-promiscuous—A receiver mode that only enables a data available interrupt when the address frame of the message matches that previously specified. The **5250** and **general purpose 8-bit modes** of the **BCP** support both **promiscuous** and non-promiscuous modes.

NRZ—Non Return to Zero. A data format that uses a high level to represent a data 1 and a low level to represent a data 0. The signal level does not return to a zero level in each bit time. See also **NRZI.**

NRZI—Non Return to Zero Inverted. A data format similar to **NRZ** but with the signal levels reversed.

OCLK—The external Oscillator CLocK connected to the **BCP.** This frequency, from a crystal or a clock, cannot be changed by the **BCP** itself. **CPU-CLK** is derived from OCLK; in addition, the **transceiver** can be configured so that **TCLK** is derived from OCLK.

parity—A one bit code, usually following data, that makes the total number of 1's in a data word odd or even, including the parity bit itself. It is included as an error checking mechanism.

POLL—A command issued by a **controller** to determine changes in terminal status, such as keyboard activity or keylock.

POLL/ACK (PACK)—A command issued by a **controller** to indicate to the terminal that the controller has recognized the non-zero status response of the terminal to its **POLL,** hence its full name poll/acknowledge.

6.0 Reference Section (Continued)

pop—To remove data from a **stack**.

predistortion—The initial voltage step in a **Manchester encoded** bit used to change frequency components of the signal to limit introducing **jitter.**

promiscuous—A receiver mode that enables a data available interrupt regardless of the contents of the transmission address frame. The **5250** and **general purpose 8-bit** modes of the **BCP** support both promiscuous and **non-promiscuous** modes.

push—To place data onto a **stack.**

quiesce pulse—A **bi-phase** 1 bit that is placed at the beginning of a transmission to charge the cable in preparation for the transmission of data. In addition, the quiesce pulses are used as part of the identifying **start sequence**. Typically, five quiesce pulses are placed there.

register addressing mode—An addressing method that uses only operands contained in registers.

register-relative addressing mode—An instruction addressing mode that adds the unsigned 8-bit value in the current **accumulator** to any one of the index registers forming a data memory address for one of the instruction's operands.

remote access—An access to **dual port memory** by a device other than the **BCP.**

repeater—A device used to extend the communication distance between a **controller** and a slave device by receiving the message and re-transmitting it.

RIAS—The Remote Interface and Arbitration System that allows a remote processor and the **BCP** to share the same memory with arbitration of any conflict while the **BCP** is running. A remote processor may also stop and start the **BCP** as well as read and write the Program Counter.

soft-loadable—A feature of a processor system that allows another processor to provide it with instructions and data.

stack—See **LIFO.**

start sequence—A unique arrangement of bits that begin each transmission to ensure proper frame alignment and synchronization. Each transmission begins with five **bi-phase** encoded 1's **quiesce pulses**, a **code violation**, and the **sync bit** of the first frame.

station address—The identification number of a **5250** terminal or other slave device that will specify which device on a **multidrop** line a message is sent to.

sync bit—A **bi-phase** 1 that is placed as the first bit of a frame.

T-state—The period of **CPU-CLK.**

TCLK—The Transceiver CLocK that runs both the transmitter and receiver at a frequency equal to eight times the required serial data rate. The clock can be obtained from a scaled **OCLK** or from **X-TCLK.**

time-out—An interrupt that occurs when the timer reaches a count of zero.

transceiver—The TRANSmitter used for sending messages and the reCEIVER used for reading messages.

transformer coupled—The isolation of the **transceiver** from the transmission cable through the use of a transformer. Contrast this with **direct coupled.**

trap—A **BCP** instruction that forces a software interrupt.

TT/AR—Transmission Turn-around / Auto Response. An acknowledgement by the terminal or other slave device that a write command has successfully been received or that a **POLL** command status response is all zero.

twin-ax—(1) The shielded pair cable that is used in a **5250** communications systems. (2) Sometimes used to refer to the IBM **5250** communications protocol itself.

unmask—Enable the accepting of an interrupt by the CPU.

wait state—Additional **T-states** that may be added to a memory access to increase the time from address generation to the beginning of either a memory read or write. The **BCP** may add as many as seven data wait states and three instruction wait states.

X-TCLK—The eXternal Transceiver CLocK. An independent clock source that the **BCP transceiver** operation may synchronize to rather than from **OCLK.**

2

Interfacing Memory to the DP8344A

As with most other aspects of a design, choosing memory is a cost vs. performance trade off. Maximum performance is achieved running no wait-states with fast, expensive memory. Slower, less expensive memory can be used, but wait-states must be added, slowing down the BCP. Therefore one needs to choose the slowest memory possible while still meeting design specifications. While this article assumes RAM is used for instruction and data memory, the information is relevant to memory devices in general.

The BCP needs separate data and instruction RAM, each with their own requirements. Instruction read time is the major constraint when choosing instruction RAM. Instruction read time, t_I, as shown in *Figure 1*, is measured from when the instruction address becomes valid to when the next instruction is latched into the BCP. Instruction read time for various clock frequencies and wait states are given in Table I. Clock frequency and wait state combinations other than those given in the table can be calculated by the following equation:

$$t_I = 10^3 (1.5 + n_{IW})/f_{CPU} - 24$$

where t_I is the instruction read time (ns), n_{IW} is the number of instruction memory wait states, and f_{CPU} is the clock frequency (MHz) at which the CPU is running. The RAM chosen needs to have a faster access time than the read time for the desired combination of clock frequency and wait states. However, instruction read time is not the only timing consideration when choosing instruction RAM. If the BCP is used in an application which requires full speed softloading of instruction RAM, there are two other timing relationships which require evaluation. These are data setup time and write pulse width. The relevant BCP timing parameters are I valid before $\overline{IWR}$ rising, $t_{PD-I-IWR}$, and $\overline{IWR}$ low time, t_{W-IWR}. The value of these timing parameters depends on the Remote Interface mode of operation. More detailed information can be found in the Electrical Specifications and the Remote Interface and Arbitration System sections of the BCP data manual. Note that in a typical application of the BCP, softloading occurs after reset with the BCP operating with CLK/2 and full wait states. Under these conditions the instruction read time value is the critical parameter for choosing the instruction RAM.

TL/F/10447-1
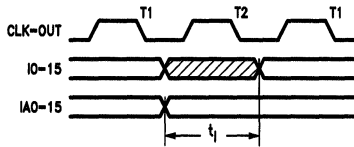
FIGURE 1. Instruction Read Time

TABLE I. Instruction Read Times, t_I (ns)

CPU Clock Freq. (MHz)	Wait States		
	0	1	2
9.43	135	241	347
18.86	55	108	161
20.00	51	101	151

The selection of data memory RAM requires the evaluation of several important timing parameters. The RAM access time, strobe width, and data setup times are three of the most critical timing parameters and must all be matched to equivalent BCP timing parameters. The RAM access time should be compared to the data read time of the BCP.

Data read time, t_D, *(Figure 2)* is measured from when the data address is valid to when data from the RAM is latched into the BCP. Table II gives data read times. The equation for calculating data read time is similar to the one given for instruction read time:

$$t_D = 10^3 (2.5 + MAX(n_{DW}, n_{IW} - 1))/f_{CPU} - 52$$

where t_D is the data read time (ns), n_{DW} is the number of data memory wait states, n_{IW} is the number of instruction memory wait states, and f_{CPU} is the clock frequency (MHz) at which the CPU is running. Since the lower address byte (AD) is externally latched, the latch propagation delay needs to be subtracted from the available read time when determining the required RAM access time.
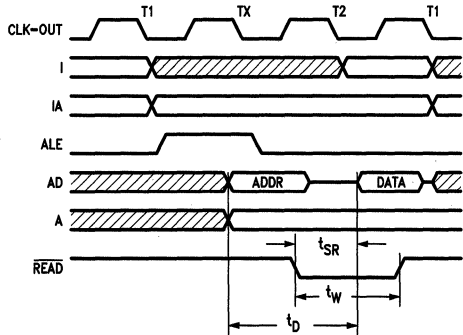
TL/F/10447-2

FIGURE 2. Data Memory Read Timing

TABLE II. Data Read Time, t_D (ns)

CPU Clock Freq. (MHz)	Wait States MAX(n_{DW}, n_{IW} − 1)		
	0	1	2
9.43	213	319	425
18.86	80	133	186
20.00	73	123	173

Another important timing parameter is the RAM strobe width. The BCP $\overline{READ}$ and $\overline{WRITE}$ outputs will typically be used to strobe data out of and into the RAM. The signal relationships for a data memory access are shown in *Figure 2* for a read and in *Figure 3* for a write. Table III contains $\overline{READ}$ and $\overline{WRITE}$ pulse width values for various clock frequencies and wait state combinations. The equation for calculating $\overline{READ}$ and $\overline{WRITE}$ pulse width is:

$$t_W = 10^3(1 + MAX(n_{DW}, n_{IW} - 1)/f_{CPU} - 10$$

where t_W is the pulse width (ns), n_{DW} is the number of data memory wait states, n_{IW} is the number of instruction memory wait states, and f_{CPU} is the clock frequency (MHz) at

which the CPU is running. The RAM chosen should require shorter strobe widths than the pulse width listed in Table III for the desired combination of clock frequency and wait states.
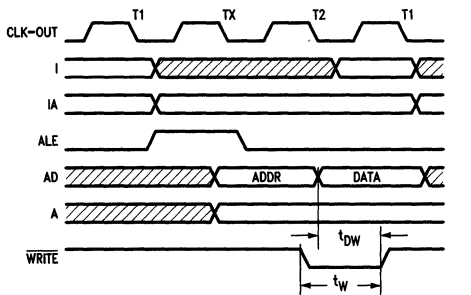
TL/F/10447–3

FIGURE 3. Data Memory Write Timing

TABLE III. $\overline{READ}$ and $\overline{WRITE}$ Pulse Width, t_W (ns)

CPU Clock Freq. (MHz)	Wait States MAX(n_{DW}, n_{IW} − 1)		
	0	1	2
9.43	96	202	308
18.86	43	96	149
20.00	40	90	140

The last important consideration when choosing the data memory RAM is setup times into the BCP on a read and into the RAM on a write. In a typical application, $\overline{READ}$ is connected to the output enable pin on the RAM. When reading from the RAM, the data becomes valid when $\overline{READ}$ falls and activates the RAM outputs. The data must become valid fast enough to meet the setup time required by the BCP. This setup time t_{SR}, as shown in *Figure 2*, is listed in Table IV for various combinations of clock frequencies and wait states. It can be calculated from the following equation:

$$t_{SR} = 10^3(1 + MAX(n_{DW}, n_{IW} - 1)/f_{CPU} - 26$$

where t_{SR} is the maximum time allowed for the data to become valid (ns), n_{DW} is the number of data memory wait states, n_{IW} is the number of instruction memory wait states, and f_{CPU} is the clock frequency (MHz) at which the CPU is running. The data memory RAM used needs to have a faster output enable time than the time listed in Table IV for the desired combination of clock frequency and wait states.

TABLE IV. Data Read Setup Time, t_{SR} (ns)

CPU Clock Freq. (MHz)	Wait States MAX(n_{DW}, n_{IW} − 1)		
	0	1	2
9.43	80	186	292
18.86	27	80	133
20.00	24	74	124

When writing to data memory, the data must be valid in time to meet the setup time requirement of the RAM. In a typical application, this time is measured from the data becoming valid out of the BCP to $\overline{WRITE}$ going high. *Figure 3* shows this timing relationship, t_{DW}, and Table V contains times for various combinations of clock frequencies and wait states. The equation for calculating this time is:

$$t_{DW} = 10^3(1 + MAX(n_{DW}, n_{IW} - 1))/f_{CPU} - 20$$

where t_{DW} is the minimum data valid time before $\overline{WRITE}$ rising (ns), n_{DW} is the number of data memory wait states, n_{IW} is the number of instruction memory wait states, and f_{CPU} is the clock frequency (MHz) at which the CPU is running. This time should be at least as long as the data setup time of the RAM.

TABLE V. Data Write Valid Time, t_{DW} (ns)

CPU Clock Freq. (MHz)	Wait States MAX(n_{DW}, n_{IW} − 1)		
	0	1	2
9.43	94	200	306
18.86	41	94	147
20.00	30	80	130

Instruction RAM has the greatest effect on execution speed. Each added instruction memory wait state slows the BCP by about 40% as compared to running with no instruction memory wait states. Each added data memory wait state slows a data access by 33% as compared to running with no data memory wait states. RAM costs are coming down, but higher speed RAM still carries a price premium. So there is the trade-off.

2

A Combined Coax-Twisted Pair 3270 Line Interface for the DP8344 Biphase Communications Processor

National Semiconductor
Application Note 624
Tim Davis and David Weinman

This paper will discuss the design of an improved 3270 transceiver interface for the National Semiconductor DP8344 combining increased error-free performance and the ability to communicate over both coax and twisted pair transmission lines. At this date, the largest installed base of terminals is the 3270 protocol terminal which primarily utilizes coax cabling. Because of phone wire's easy accessibility and lower cost, twisted pair cabling has become popular among end users for new terminal installations. In the past, baluns have been used to augment existing coax interfaces, but their poor performance and cost considerations leave designers seeking new solutions. In addition, the integration of coax and twisted pair on the same board has become a market requirement, but this is a considerable design challenge. A brief summary of the interface concepts, a discussion of the proposed design, and a description of the results are included in this application note.

CONCEPTS

Coax cable is normally driven on the center conductor with the shield grounded. Conversely, unshielded twisted pair cable is driven on both lines. Because of the way that each is driven, coax operation is often called unbalanced and twisted pair operation balanced.

Transmission line characteristics of coax and twisted pair cables can be envisioned as essentially those of a low-pass filter with a length-dependent bandwidth.[1] In 3270 systems, different data combinations generate dissimilar transmission frequencies because of the Manchester format.[2] These two factors combine to produce data pulse widths that vary according to the data transmitted and the length and type of cable used. This pulse-width variation is often described as "data jitter."[3]

In addition to line filtering, noise can cause jitter. Coax cable employs a shield to isolate the signal from external noise. Electromagnetically balanced lines minimize differential noise in unshielded twisted pair cable. In other words, the twisted pair wires are theoretically equidistant from any noise source, and all noise superimposed on the signal should be the common-mode type. Although these methods diminish most noise, they are not totally effective, and environmental interference from other nearby wiring and circuitry may still cause problems.

Besides the effects of jitter, reflections can produce undesirable signal characteristics that introduce errors. These reflections may be caused by cable discontinuities, connectors, or improper driver and receiver matching. Signal edge rates may aggravate reflection problems since faster edges tend to produce reflections that may dramatically distort the signal.[3] Most reflection difficulties occur over short cable (less than 150 ft.) because at these distances reflections suffer little attenuation and can significantly distort the signal. Since the timing of the reflections is a function of cable length, it may be possible to operate at some short distance and not at some greater length.

An effective receiver design must address each of the above concerns. To counteract the effects of line filtering and noise, there must be a large amount of jitter tolerance. Some filtering is needed to reduce the effects of environmental noise caused by terminals, computers, and other proximate circuitry. At the same time, such filtering must not introduce transients that the receiver comparator translates into data jitter.

Like the receiver design, a successful driver design should compensate for the filtering effects of the cable. As cable length is increased, higher data frequencies become attenuated more than lower frequency signals, yielding greater disparity in the amplitudes of these signals.[4] This effect generates greater jitter at the receiver. The 3270 signal format allows for a high voltage (predistorted) magnitude followed by a low voltage (nondistorted) magnitude within each data half-bit time.[2] Increasing the predistorted-to-nondistorted signal level ratio counteracts the filtering phenomenon because the lower frequency signals contain less predistortion than do higher frequency signals. Thus, the amplitude of the higher frequency components are greater than the lower frequency components at the transmitter. Implementation of this compensation technique is limited because nondistorted signal levels are more susceptible to reflection-induced errors at short cable lengths. Consequently, proper impedance matching and slower edge rates must be utilized to eliminate as much reflection as possible at these lengths.

Besides improved performance, both unbalanced and balanced operation must be adequately supported. Electromagnetic isolation for coaxial cabling can be provided by a properly grounded shield. Electrically and geometrically symmetric lines must be maintained for twisted pair operation. For both cable types, proper termination should be employed, although terminations slightly greater than the characteristic impedance of the line may actually provide a larger received signal with insignificant reflection.[3] In the board layout, the comparator traces should be as short as possible. Lines should be placed close together along their entire path to avoid the introduction of differential noise. These traces should not pass near high frequency lines and should be isolated by a ground plane.

BCP LINE INTERFACE DESIGN

An extensive characterization of the BCP comparator was done to facilitate this interface design. The proposed design enhances some of the BCP transceiver's characteristics and incorporates the aforementioned suggestions.

The interface design takes into account the common comparator attributes of power supply rejection, variable switching offset, finite voltage sensitivity, and fast edge rate sensitivity. V_{CC} noise can affect the comparator output when the inputs are biased to the same voltage. This particular type of biasing may render portions of the comparator susceptible to supply noise. Variable switching offset and finite voltage sensitivity cause the receiver decoding circuitry to see a

substantial amount of data jitter when signal amplitudes approach the sensitivity limits of the comparator. At these signal magnitudes, considerable variation in the output of the comparator is observed. Finally, edge sensitivity may allow a fast edge to introduce errors as charge is coupled through the inputs during a rapid predistorted-to-nondistorted level transition, especially as the nondistorted level is reduced in magnitude.

The receiver interface design *(Figure 1)* addresses each of the BCP comparator's characteristics. A small offset (about 17 mV) separates the inputs to eliminate V_{CC}-coupled noise. This offet is relatively large compared to possible fabrication variations, resulting in a more consistent, device-independent operation. The offset has the added benefit of making the comparator more immune to ambient noise that may be present on the circuit board. A 2:1:1 transformer (arranged as a 3:1) restores any voltage sensitivity lost by introducing the offset. A bandpass filter is employed to reduce the edge rate of the signal at the comparator and to eliminate environmental noise. The bandwidth (30 kHz to 30 MHz) was chosen to provide sufficient noise attenuation while producing minimum data jitter. Refer to Appendix 2 for a derivation of the filter equations.

Like many present 3270 circuits, the driver design *(Figure 2)* utilizes a National Semiconductor DS3487 and a resistor network to generate the proper signal levels. The predistorted-to-nondistorted ratio was chosen to be about 4.5 to 1. This ratio was observed to offer good noise immunity at short cable lengths (less than 150 feet) and error-free transmission to an IBM 3174 controller at long cable lengths (greater than 5000 feet).

To allow for two interfaces in the same circuit design, the coax/twisted pair front end *(Figure 3)* includes an ADC Telecommunications brand TPC connector to switch between coax and twisted pair cable. This connector allows different male connectors for coax and twisted pair cable to switch in different interfaces for the particular cable type. The coax interface has only the shield capacitively coupled to ground. The 510Ω resistor and the filter loading produce a termination of about 95Ω. The twisted pair interface balances both lines and possesses an input impedance of about 100Ω. This termination is somewhat higher than the characteristic impedance (about 96Ω) of twisted pair. Terminations of this type produce reflections that do not tend to generate mid-bit errors, as well as having the benefit of creating a larger voltage at the receiver over longer cable lengths.

Legend
Ⓐ To coax/twisted pair front end
Ⓑ To line driver circuitry
*Includes board capacitance
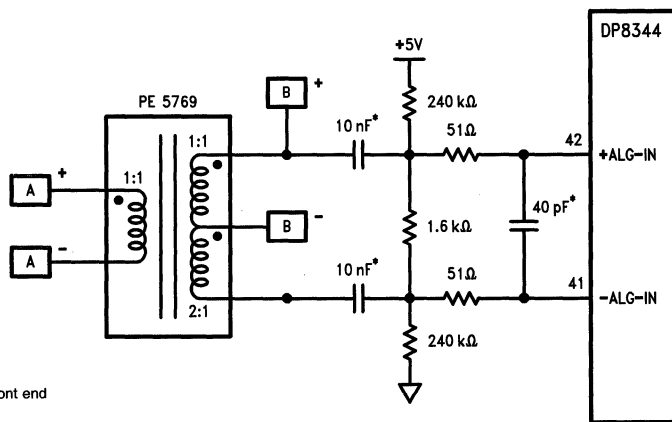
TL/F/10448–1

FIGURE 1. BCP Receiver Filter Design

Legend
Ⓑ To 2:1:1 Transformer
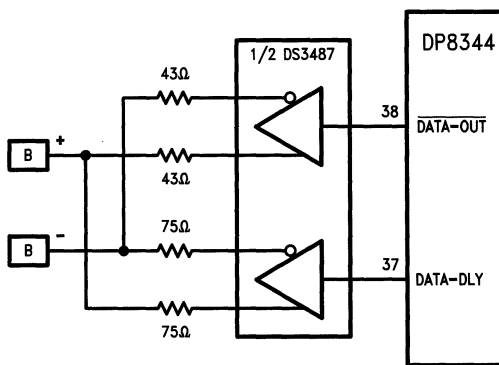
TL/F/10448–2

FIGURE 2. BCP Driver Design

ADC Connector

Center

Twisted Pair

Shield

510Ω

510Ω

0.1 μF

0.1 μF

0.1 μF

0.1 μF

A +

A −
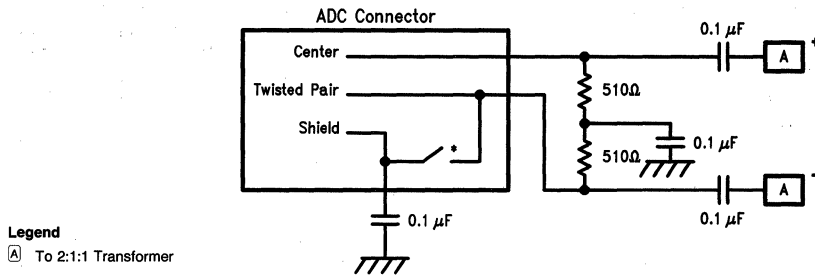
Legend

Ⓐ To 2:1:1 Transformer

TL/F/10448–3

*Connector closes switch for coax and opens switch for twisted pair.

FIGURE 3. BCP Coax/Twisted Pair Front End

RESULTS AND COMPARISONS

The evaluation involved producing multiple data transfers between an IBM 3174-81R and the device under test during a live 3270 session. The preferred method of testing would be to transfer extremely large files to the host. Since terminals and muxes cannot transfer files and all devices being tested needed to be evaluated under similar conditions, a screen-oriented approach was taken for testing. The screen-oriented approach involved using common methods for forcing the controller to send an entire screen of characters to the device. Procedural specifics are included in Appendix 1.

Performance of the BCP interface typically extended over 8000 feet of RG62A/U coax and 1750 feet of AT&T DIW 4 pair/24 AWG unshielded twisted pair. This operation met or exceeded many of the current 3270 solutions. The performance of other 3270 products was obtained from production stock of competitors' equipment and should be taken as typical operation. Although these long distances are possible, it is recommended that companies specify their products to IBM's PAI[2] specifications of 5000 feet of coax cable. The extra long distance capability of the new interface will assure the designer a comfortable guardband of performance. Similarly, a 50% margin on the unshielded twisted pair capability will give approximately a 900 foot specification.

It should be noted that the BCP receiver detects errors before the controller does. This is because of comparator skew, a mechanism that occurs when the amplitude of the signal approaches the sensitivity of the comparator. At these small levels, propagation symmetry for high-to-low and low-to-high transitions is lost. The failure mechanisms of competitors include insufficient receiver jitter tolerance, filter transients, and comparator skew. Operational distance may be extended by the utilization of transformers with higher turn ratios as long as considerations are taken for impedance matching, driver loading, and component quality toler-

ances (higher turn ratios may demand circuits with very low tolerance percentages).

There are also economical advantages in using the BCP comparator. The number of active and passive components required to build the line interface is small compared to competing solutions. The proposed design is extremely cost competitive with current media solutions.

CONCLUSION

An effective and economical 3270 interface solution has been demonstrated using only passive components and a line driver. Guidelines have also been suggested to facilitate the design and layout of such an interface. Criteria concerning board layout and noise suppression must be considered to be at least as important as the components themselves; for example, adjustments should be made for variations in board capacitances and inductances. With only slight modification of the components given for this design, it is thus likely that optimum performance can be obtained for a specific layout. Implementation of these design principles should prove advantageous for the development of an efficient and competitive 3270 line interface.

REFERENCES

1. H.P. Neff, Jr., *Basic Electromagnetic Fields,* New York: Harper and Row, 1981, Chapter 13.

2. *IBM 3174/3274 Control Unit to Device Product Attachment Information,* Communication Products Information Development, International Business Machines Corporation, Research Triangle Park, NC, October 1986.

3. K.M. True, *The Interface Handbook: Line Drivers and Receivers,* Semiconductor Components Group, Fairchild Camera and Instrument Corporation, Mountain View, CA, 1975. Chapters 3 and 4.

4. N.S. Nahman, "A Discussion on the Transient Analysis of Coaxial Cables Considering High Frequency Losses," *IRE Trans. Circuit Theory,* vol. CT-9, pp. 144–152, June 1962.

APPENDIX 1:
TEST PROCEDURE FOR LONG AND SHORT DISTANCE TESTING

1. Enter Test mode on the 3174 controller.

2. Clear the error counters.

3. Hit the Clear key rapidly 30 times. This will repaint the screen with the test menu very rapidly. This is a quick and easy method to cause an entire screen of characters to be sent to either an emulation card or a terminal over the coax.

4. Exit Test mode.

5. LOGON to a session on the host.

6. Issue the FILELIST command.

7. Hit the Clear key 20 times. After the controller clears the screen, it will repaint the FILELIST menu each time. This will again cause an entire screen of characters to be sent over the coax to the device under test.

8. XEDIT a 40k file text file.

9. Page through the entire file forwards once, then backwards once. Again, this will cause a varied stream of transmissions to be sent to the device under test.

10. LOGOFF the session.

11. Enter Test mode again.

12. Check for errors on the error test screen.

APPENDIX 2:
DERIVATION OF FILTER EQUATIONS

The basic operation of the filter can be understood by studying the figure below. The actual circuit includes the effects of the terminating resistors, DC isolation capacitors, and the transformer; furthermore, a thorough investigation of bandwidth and gain characteristics should employ the use of a circuit simulator such as SPICE.
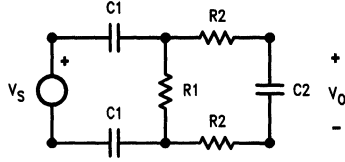
TL/F/10448–4

Simple loop analysis yields the following transfer function for the filter:

$$\frac{V_O}{V_S} = \frac{\dfrac{1}{2\,R2\,C2}(S)}{S^2 + S\left[\dfrac{R1\,C1 + C2\,(4\,R2 + 2\,R1)}{2\,R1\,R2\,C1\,C2}\right] + \dfrac{1}{R1\,R2\,C1\,C2}}$$

If it is assumed R1 $\gg$ R2 and C1 $\gg$ C2, we can then simplify the equation and solve for the poles to obtain the following form:

$$|f| = \frac{\dfrac{1}{2\,R2\,C2} \pm \sqrt{\dfrac{1}{4R2^2\,C2^2} - 4\left(\dfrac{1}{R1\,R2\,C1\,C2}\right)}}{4\pi}$$

After splitting the above equation to solve each pole and using a binomial expansion to simplify each pole's equation, we get:

$$f_l \approx \frac{1}{\pi R1\,C1} \approx 20\text{ kHz}$$

(vs. 30 kHz from simulation and testing)

$$f_h \approx \frac{1}{4\pi\,R2\,C2} \approx 40\text{ MHz}$$

(vs. 30 MHz from simulation and testing)

2

Interfacing the DP8344 to Twinax

National Semiconductor
Application Note 516
Thomas J. Quigley

OVERVIEW

The DP8344, or **Biphase Communications Processor** from National Semiconductor's Advanced Peripherals group brings a new level of system integration and simplicity to the IBM® connectivity world. Combining a 20 MHz RISC architecture CPU with a flexible multi-protocol transceiver and remote interface, the BCP is well suited for IBM 3270, 3299 and 5250 protocol interfaces. This Application Note will show how to interface the BCP to twinax, as well as provide some basics about the IBM 5250 environment.

5250 ENVIRONMENT

The IBM 5250 environment encompasses a family of devices that attach to the IBM System/34, 36 and 38 mid-size computer systems. System unit model numbers include the 5360, 5362, 5364, 5381, and 5382, and remote controller models 5294 and 5251 model 12. The system units have integral work station controllers and some may support up to 256 native mode twinax devices locally. Native mode twinax devices are ones that connect to one of these host computers or their remote control units via a multi-drop, high speed serial link utilizing the 5250 data stream. This serial link is primarily implemented with twinaxial cable but may be also found using telephone grade twisted pair. Native mode 5250 devices include mono-chrome, color and graphics terminals, as well as a wide range of printers and personal computer emulation devices.

TWINAX AS A TRANSMISSION MEDIA

The 5250 environment utilizes twinax in a multi-drop configuration, where eight devices can be "daisy-chained" over a total distance of 5000 ft. and eleven splices, (each physical device is considered a splice) see *Figure 1*. Twinax can be routed in plenums or conduits, and can be hung from poles between buildings (lightning arrestors are recommended for this). Twinax connectors are bulky and expensive, but are very sturdy. Different sorts may be purchased from IBM or a variety of third party vendors, including Amphenol. Twinax should not be spliced; to connect cables together both cables should be equipped with male connectors and a quick-disconnect adapter should be used to join them (Amphenol #82-5588).

Twinaxial cable is a shielded twisted pair that is nearly 1/3 of an inch thick. This hefty cable can be either vinyl or teflon jacketed and has two internal conductors encased in a stiff polethylene core. The cable is available from BELDEN (type #9307) and other vendors, and is significantly more expensive than coax.

The cable shield must be continuous throughout the transmission system, and be grounded at the system unit and each station. Since twinax connectors have exposed metal connected to their shield grounds, care must be taken not to expose them to noise sources. The polarity of the two inner conductors must also be maintained throughout the transmission system.

The transmission system is implemented in a balanced current mode; every receiver/transmitter pair is directly coupled to the twinax at all times. Data is impressed on the transmission line by unbalancing the line voltage with the driver current. The system requires passive termination at both ends of the transmission line. The termination resistance value is given by:

$R_t = Z_{O/2}$; where

R_t: Termination Resistance

Z_O: Characteristic Impedance

In practice, termination is accomplished by connecting both conductors to the shield via 54.9Ω, 1% resistors; hence the characteristic impedance of the twinax cable of 107Ω ±5% at 1.0 MHz. Intermediate stations must not terminate the line; each is configured for "pass-through" instead of "terminate" mode. Stations do not have to be powered on to pass twinax signals on to other stations; all of the receiver/transmitter pairs are DC coupled. Consequently, devices must never output any signals on the twinax line during power-up or down that could be construed as data, or interfere with valid data transmission between other devices.

WAVEFORMS

The bit rate utilized in the 5250 protocol is 1 MHz ±2% for most terminals, printers and controllers. The IBM 3196 dis-

STATION '0'   STATION '1'   STATION '7'
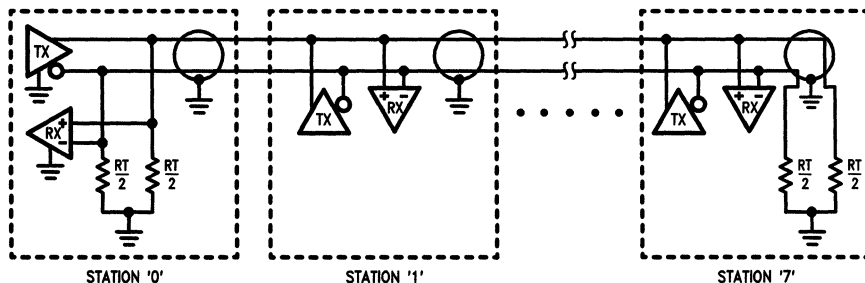
TL/F/9635-1

FIGURE 1. Multi-Drop Transmission Lines

The eight stations shown include the host device as a station. The first and last stations are terminated while intermediate stations are not.

play station has a bit rate of 1.0368 MHz ±0.01%. The data are encoded in biphase, NRZI (non-return to zero inverted) manner; a "1" bit is represented by a positive to negative transition, a "0" is a negative to positive transition in the center of a bit cell. This is opposite from the somewhat more familiar 3270 coax method. The biphase NRZI data is encoded in a pseudo-differential manner; i.e. the signal on the "A" conductor is subtracted from the signal on "B" to form the waveform shown in *Figure 2*. Signals A and B are not differentially driven; one phase lags the other in time by 180°. *Figures 3* and *4* show actual signals taken at the driver and receiver after 5000 ft. of twinax, respectively.

The signal on either the A or B phase is a negative going pulse with an amplitude of −0.32V ±20% and duration of 500 ±20 ns. During the first 250 ±20 ns, a predistortion or pre-emphasis pulse is added to the waveform yielding an amplitude of −1.6V ±20%. When a signal on the A phase is considered together with its B phase counterpart, the resultant waveform represents a bit cell or bit time, comprised of two half-bit times. A bit cell is 1 μs ±20 ns in duration and must have a mid bit transition. The mid bit transition is the synchronizing element of the waveform and is key to maintaining transmission integrity throughout the system.

FIGURE 2. Twinax Waveforms

The signal on phase A is shown at the initiation of the line quiesce/line violation sequence.

Phase B is shown for that sequence, delayed in time by 500 ns.

The NRZI data recovered from the transmission.
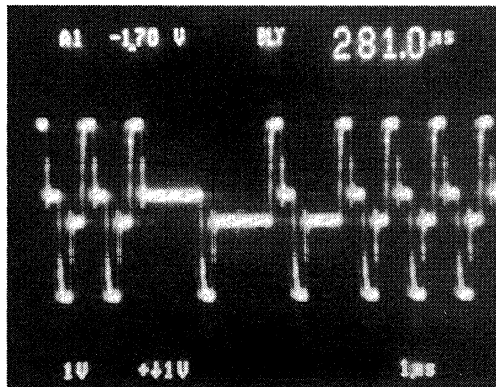
TL/F/9635–2

TL/F/9635–3

FIGURE 3. Signal at the Driver

The signal shown was taken with channel 1 of an oscilloscope connected to phase B, channel 2 connected to A, and then channel 2 inverted and added to channel 1.
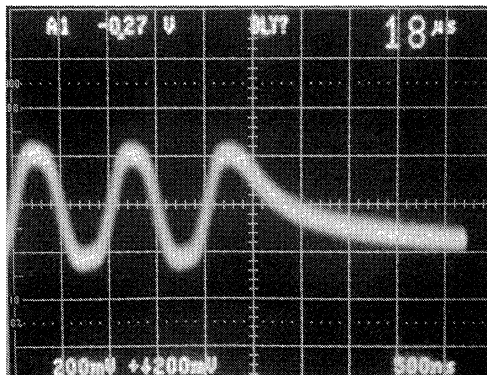
TL/F/9635–4

FIGURE 4. Signal at the Receiver

The signal shown was viewed in the same manner as *Figure 3*. The severe attenuation is due to the filtering effects of 5000 ft. of twinax cable.

2

As previously mentioned, the maximum length of a twinax line is 5000 ft. and the maximum number of splices in the line is eleven. Devices count as splices, so that with eight devices on line, there can be four other splices. The signal 5000 ft. and eleven splices from the controller has a minimum amplitude of 100 mV and a slower edge rate. The bit cell transitions have a period of 1 μs $\pm$30 ns.

5250 BIT STREAM

The 5250 Bit stream used between the host control units and stations on the twinax line consists of three separate parts; a bit synchronization pattern, a frame synchronization pattern, and one or more command or data frames. The bit sync pattern is typically five one bit cells. This pattern serves to charge the distributed capacitance of the transmission line in preparation for data transmission and to synchronize receivers on the line to the bit stream. Following the bit sync or line quiesce pattern is the frame sync or line violation. This is a violation of the biphase, NRZI data mid bit transition rule. A positive going half bit, 1.5 times normal duration, followed by a negative going signal, again 1.5 times normal width, allows the receiving circuitry to establish frame sync.

Frames are 16 bits in length and begin with a sync or start bit that is always a 1. The next 8 bits comprise the command or data frame, followed by the station address field of three bits, a parity bit establishing even parity over the start, data and address fields, and ending with a minimum of three fill bits (fill bits are always zero). A message consists of a bit sync, frame sync, and some number of frames up to 256 in total. A variable amount of inter-frame fill bits may be used to control the pacing of the data flow. The SET MODE command from the host controller sets the number of bytes of zero fill sent by attached devices between data frames. The zero fill count is usually set to zero. The number of zero fill bits injected between frames by the BCP is set by the Fill Bit select register {FBR}. This register contains the one's complement of the number of BITS sent, not bytes.

Message routing is accomplished through use of the three-bit address field and some basic protocol rules. As mentioned above, there is a maximum of eight devices on a given twinax line. One device is designated the controller or host and the remaining seven are slave devices. All communication on the twinax line is host initiated and half duplex. Each of the seven devices is assigned a unique station address from zero to six. Address seven is used for an End Of Message delimiter, or EOM. The first or only frame of a message from controller to device must contain the address of the device. Succeeding frames do not have to contain the same address for the original device to remain selected, although they usually do.

The last frame in a sequence must contain the EOM delimiter. For responses from the device to the controller, the responding device places its own address in the address field in frames 1 to (n − 1),where n ≤ 256, and places the EOM delimiter in the address field of frame n. However, if the response to the controller is only one frame, the EOM delimiter is used. The controller assumes that the responding devices was the one addressed in the initiating command.

Responses to the host must begin in 60 ±20 μs, although some specifications state a 45 ±15 μs response time. In practice, controllers do not change their time out values per device type so that anywhere from 30 μs to 80 μs response times are appropriate.

DRIVER CIRCUITS FOR THE DP8344

The transmitter interface on the DP8344 is sufficiently general to allow use in 3270, 5250, and 8-bit transmission systems. Because of this generality, some external hardware is needed to adapt the outputs to form the signals necessary to drive the twinax line. The chip provides three signals: $\overline{\text{DATA-OUT}}$, DATA-DLY, and TX-ACT. $\overline{\text{DATA-OUT}}$ is biphase serial data (inverted). DATA-DLY is the biphase serial data output (non-inverted) delayed one-quarter bit-time. TX-ACT, or transmitter active, signals that serial data is being transmitted when asserted. $\overline{\text{DATA-OUT}}$ and DATA-DLY can be used to form the A and B phase signals by the circuit shown in *Figure 5*. TX-ACT is used as an external transmitter enable. The BCP can invert the sense of the $\overline{\text{DATA-OUT}}$ and DATA-DLY signals by asserting TIN {TMR[3]}. This feature allows both 3270 and 5250 type biphase data to be generated, and/or utilization of inverting or non-inverting transmitter stages.
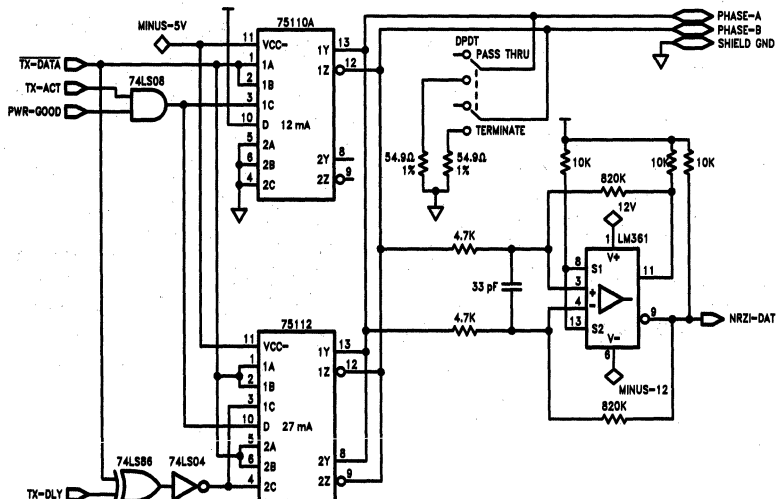
FIGURE 5. Schematic

TL/F/9635–5

The current mode drive method used by native twinax devices has both distinct advantages and disadvantages. Current mode drivers require less power to drive properly terminated, low-impedance lines than voltage mode drivers. Large output current surges associated with voltage mode drivers during pulse transition are also avoided. Unwanted current surges can contribute to both crosstalk and radiated emission problems. When data rate is increased, the surge time (representing the energy required to charge the distributed capacitance of the transmission line) represents a larger percentage of the driver's duty cycle and results in increased total power dissipation and performance degradation.

A disadvantage of current mode drive is that DC coupling is required. This implies that system grounds are tied together from station to station. Ground potential differences result in ground currents that can be significant. AC coupling removes the DC component and allows stations to float with respect to the host ground potential. AC coupling can also be more expensive to implement.

Drivers for the 5250 environment may not place any signals on the transmission system when not activated. The power-on and off conditions of drivers must be prevented from causing noise on the system since other devices may be in operation. *Figure 5* shows a "DC power good" signal enabling the driver circuit. This signal will lock out conduction in the drivers if the supply voltage is out of tolerance.

Twinax signals can be viewed as consisting of two distinct phases, phase A and phase B, each with three levels, off, high and low. The off level corresponds with 0 mA current being driven, the high level is nominally 62.5 mA, +20%−30%, and the low level is nominally 12.5 mA, +20%−30%. When these currents are applied to a properly terminated transmission line the resultant voltages impressed at the driver are: off level is 0V, low level is 0.32V ±20%, high level is 1.6V ±20%. The interface must provide for switching of the A and B phases and the three levels. A bi-modal constant current source for each phase can be built that has a TTL level interface for the BCP.

An integrated solution can be constructed with a few current mode driver parts available from National and Texas Instruments. The 75110A and 75112 can be combined to provide both the A and B phases and the bi-modal current drive required as in *Figure 5*. The external logic used adapts the coax oriented BCP outputs to the twinax interface circuit, and prevents spurious transmissions during power-up or down. The serial NRZ data is inverted prior to being output by the BCP by setting TIN, {TMR[3]}.

RECEIVER CIRCUITS

The pseudo-differential mode of the twinax signals make receiver design requirements somewhat different than the coax 3270 world. Hence, the analog receiver on the BCP is not well suited to receiving twinax data. The BCP provides both analog inputs to an on-board comparator circuit as well as a TTL level serial data input, TTL-IN. The sense of this serial data can be inverted by the BCP by asserting RIN, {TMR[4]}.

The external receiver circuit must be designed with care to ensure reliable decoding of the bit-stream in the worst environments. Signals as small as 100 mV must be detected. In order to receive the worst case signals, the input level switching threshold or hysteresis for the receiver should be

nominally 29 mV ±20%. This value allows the steady state, worst case signal level of 100 mV 66% of its amplitude before transitioning.

To achieve this, a differential comparator with complementary outputs can be applied, such as the National LM361. The complementary outputs are useful in setting the hysteresis or switching threshold to the appropriate levels. The LM361 also provides excellent common mode noise rejection and a low input offset voltage. Low input leakage current allows the design of an extremely sensitive receiver, without loading the transmission line excessively.

In addition to good analog design techniques, a low pass filter with a roll-off of approximately 1 MHz should be applied to both the A and B phases. This filter essentially conducts high frequency noise to the opposite phase, effectively making the noise common mode and easily rejectable.

Layout considerations for the LM361 include proper bypassing of the ±12V supplies at the chip itself, with as short as possible traces from the pins to 0.1 μF ceramic capacitors. Using surface mount chip capacitors reduces lead inductance and is therefore preferable in this case. Keeping the input traces as short and even in length is also important. The intent is to minimize inductance effects as well as standardize those effects on both inputs. The LM361 should have as much ground plane under and around it as possible. Trace widths for the input signals especially should be as wide as possible; 0.1 inch is usually sufficient. Finally, keep all associated discrete components nearby with short routing and good ground/supply connections.

Design equations for the LM361 in a 5250 application are shown here for example. The hysteresis voltage, V_h, can be expressed the following way:

$$V_h = V_{rio} + ((R_{in}/(R_{in} + R_f) \times V_{ol}) - (R_{in}/(R_{in} + R_f) \times V_{ol}))$$

where

V_h — Hysteresis Voltages, Volts

R_{in} — Series Input Resistance, Ohms

R_f — Feedback Resistance, Ohms

C_{in} — Input Capacitance, Farads

V_{rio} — Receiver Input Offset Voltage, Volts

V_{oh} — Output Voltage High, Volts

V_{ol} — Output Voltage Low, Volts

The input filter values can be found through this relationship:

$$V_{cin} = V_{in1} - V_{in2/1} + jwC_{in} (R_{in1} + R_{in2})$$

where $R_{in1} = R_{in2} = R_{in}$:

$Fro = w/2\pi$

$Fro = 1/(2\pi \times R_{in} \times C_{in})$

$C_{in} = 1/(2\pi \times R_{in} \times Fro)$

where

V_{in1}, V_{in2} — Phase A and B signal voltages, Volts

V_{cin} — Voltage across C_{in}, or the output of the filter, Volts

R_{in1}, R_{in2} — Input resistor values, $R_{in1} = R_{in2}$, Ohms

Fro — Roll-Off Frequency, Hz

W — Frequency, Radians

2

The roll-off frequency, Fro, should be set nominally to 1 MHz to allow for transitions at the transmission bit rate. The transition rate when both phases are taken together is 2 MHz, but then R_{in1} and R_{in2} must be considered, so:

$$Fro2 = 1/(2\pi \times (R_{in1} + R_{in2}) \times C_{in})$$

or,

$$Fro2 = 1/(2\pi \times 2 \times R_{in} \times C_{in})$$

where $Fro2 = 2 \times Fro$, yielding the same results.

The following table shows the range of values expected:

TABLE I

Value	Maximum	Minimum	Nominal	Units	Tolerance
R_{IN}	4.935E+03	4.465E+03	4.700E+03	Ω	0.05
R_F	8.295E+05	7.505E+05	7.900E+05	Ω	0.05
C_{IN}	4.4556E−11	2.6875E−11	3.3863E−11	F	
V_{OH}	5.250E+00	4.750E+00	5.000E+00	V	
V_{OL}	4.000E−01	2.000E−01	3.000E−01	V	
V_{IN+}	1.920E+00	1.000E−01		V	
$V_{IN−}$	1.920E+00	1.000E−01		V	
V_{RIO}	5.000E−03	0.000E+00	1.000E−03	V	
R	6.533E−03	5.354E−03	5.914E−03	Ω	
Fro	1.200E+06	8.000E+05	1.000E+06	Hz	0.2
V_H	3.368E−02	2.691E−02	2.880E−02	V	
Xc	7.4025E+03	2.9767E+03	4.7000E+03	Ω	

The BCP has a number of advanced features that give designers much flexibility to adapt products to a wide range of IBM environments. Besides the basic multi-protocol capability of the BCP, the designer may select the inbound and outbound serial data polarity, the number of received and transmitted line quiesces, and in 5250 modes, a programmable extension of the TX-ACT signal after transmission.

The polarity selection on the serial data stream is useful in building single products that handle both 3270 and 5250 protocols. The 3270 biphase data is inverted with respect to 5250.

Selecting the number of line quiesces on the inbound serial data changes the number of line quiesce bits that the receiver requires before a line violation to form a valid start sequence. This flexibility allows the BCP to operate in extremely noisy environments, allowing more time for the transmission line to charge at the beginning of a transmission. The selection of the transmitted line quiesce pattern is not generally used in the 5250 arena, but has applications in 3270. Changing the number of line quiesces at the start of a line quiesce pattern may be used by some equipment to implement additional repeater functions, or for certain inflexible receivers to sync up.

The most important advanced feature of the BCP for 5250 applications is the programmable TX-ACT extension. This feature allows the designer to vary the length of time that the TX-ACT signal from the BCP is active after the end of a transmission. This can be used to drive one phase of the

twinax line in the low state for up to 15.5 μs. Holding the line low is useful in certain environments where ringing and reflections are a problem, such as twisted pair applications. Driving the line after transmitting assures that receivers see no transitions on the twinax line for the specified duration. The transmitter circuit shown in *Figure 5* can be used to hold either the A or B phase by using the serial inversion capability of the BCP in addition to swapping the A and B phases. Choosing which phase to hold active is up to the designer; 5250 devices use both. Some products hold the A phase, which means that another transition is added after the last half bit time including the high and low states, with the low state helf for the duration, see *Figure 6*. Alternatively, some products hold the B phase. Holding the B phase does not require an extra transition and hence is inherently quieter.
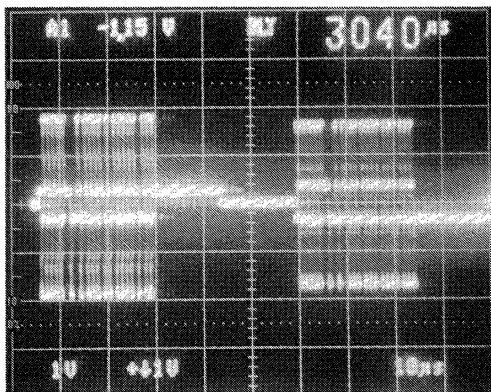
TL/F/9635–6

FIGURE 6. Line Hold Options

The signal was viewed in the same manner as *Figures 3* and *4*. The lefthand portion of the signal is a transmitting device utilizing line hold on phase A. The right hand side shows the IBM style (phase B) line hold.

To set the TX-ACT hold feature, the upper five bits of the Auxilliary Transceiver Register, {ATR [3–7]}, are loaded with one of thirty-two possible values. The values loaded select a TX-ACT hold time between 0 μs and 15.5 μs in 500 ns increments.

SOFTWARE INTERFACE

The BCP was designed to simplify designing IBM communications interfaces by providing the specific hardware necessary in a highly integrated fashion. The power and flexibility of the BCP, though, is most evident in the software that is written for it. Software design for the BCP deserves careful attention.

When designing a software architecture for 5250 terminal emulation, for example, one concern the designer faces is how to assure timely responses to the controller's commands. The BCP offers two general schemes for handling the real time response requirements of the 5250 data stream: interrupt driven transceiver interface mode, and polled transceiver interface mode. Both modes have strengths that make them desirable. The excellent interrupt

response and latency times of the BCP make interrupts very useful in most 5250 applications.

Although factors such as data and instruction memory wait states and remote processors waiting BCP data memory accesses can degrade interrupt response times, the minimum latency is 2.5 T-states. The BCP samples all interrupt sources by the falling edge of the CPU clock; the last falling edge prior to the start of the next instruction determines whether an interrupt will be processed. When an interrupt is recognized, the next instruction in the present stream is not executed, but its address is pushed on the address stack. A two T-state call to the vector generated by the interrupt type and the contents of {IBR} is executed and [GIE] (Global Interrupt Enable) is cleared. If the clock edge is missed by the interrupt request or if the current instruction is longer than 2 bytes, the interrupt latency is extended.

Running in an interrupt driven environment can be complex when multiple sessions are maintained by the same piece of code. The software has the added overhead of determining the appropriate thread or session and handling the interrupt accordingly. For a multi-session 5250 product, the transceiver interrupt service routines must determine which session is currently selected through protocol inferences and internal semaphores to keep the threads separate and intact.

In a polled environment, the biggest difficulty in designing software is maintaining appropriate polling intervals. Polling too often wastes CPU bandwidth, not polling frequently enough loses data and jeopardizes communication integrity. Standard practice in servicing polled devices is to count CPU clock cycles in the program flow to keep track of when to poll. A program change can result in lengthy recalculations of polling intervals and requalifications of program functionality. Using the programmable timer on board the BCP to set the polling interval alleviates the need to count instructions when code is changed or added. In both polled or interrupt environments, the latency effects of remote processors waiting memory accesses must be limited to a known length of time and figured into both polling intervals and worst case interrupt latency calculations. Using the programmable timer on the BCP makes both writing and maintaining polled software easier.

SOFTWARE ARCHITECTURE FOR 5250 EMULATION

The 5250 data rate is much lower than that of the 3270 data stream, hence it is possible for the BCP to emulate all seven 5250 sessions with a CPU frequency of 8 MHz. Choosing a 16 MHz crystal allows the transceiver to share the CPU clock at OCLK/2, eliminating an extra oscillator circuit. The 8 MHz rate yields a 125 ns T-state, or 250 ns for most instructions. Interrupt latency is typically one instruction (assuming no wait states or remote accesses) which is suitable for 5250 operation. If more speed is desired, the CPU could be switched to 16 MHz operation.

A MULTI-MODE TRANSCEIVER

The BCP provides two 5250 protocol modes, promiscuous and non-promiscuous. These two modes afford the designer a real option only when the end product will attach to one 5250 address at a time. The non-promiscuous mode is configured with an address in the {ATR} register and only re-

ceives messages whose first frame address matches that address, or an error occurs in the first frame of the message. Filtering out unwanted transmissions to other addresses leaves more CPU time for other non-protocol related tasks, but limits the device to one address at a time. The promiscuous mode allows messages to any and all addresses to be received. Resetting the transceiver during a message destined to another device forces the transceiver to begin looking for a start sequence again, effectively discarding the entire unwanted message. Because of its flexibility, the promiscuous mode is used in this illustration.

REAL TIME CONSIDERATIONS

Choosing a scheme for servicing the transceiver is basic to the design of any emulation device. The BCP provides both polled and interrupt driven modes to handle the real time demands of the chosen protocol. In this example, the interrupt driven approach is used. This implies the extra overhead of setting up interrupt vectors and initializing the interrupt masks appropriately. This approach eliminates the need to figure polling intervals within the context of other CPU tasks.

5250 CONFIGURATION

Configuring a complex device like the BCP can be difficult until a level of familiarity with the device is reached. To help the 5250 product designer through an initial configuration, a register by register description follows, along with the reasons for each configuration choice. Certainly, most applications will use different configurations than the one shown here. The purpose is to illustrate one possible setup for a 5250 emulation device.

There are two major divisions in the BCP's configuration registers: CPU specific and transceiver specific ones.

CPU SPECIFIC CONFIGURATION REGISTERS:

{DCR}—**Device Control Register**—This register controls the clocks and wait states for instruction and data memory. Using a value of H#A0 sets the CPU clock to the OCLK/2 rate, the transceiver to OCLK/2, and no wait states for either memory bank. As described above, the choice of a 16 MHz crystal and configuring this way allows 8 MHz operation now, with a simple software change for straight 16 MHz operation in the future.

{ACR}—**Auxiliary Control Register**—Loading this register with H#20 sets the timer clock source to CPU-CLK/2, sets [BIC], the Bidirectional Interrupt Control to configure BIRQ as an input, allows remote accesses with [LOR] cleared, and disables all maskable interrupts through [GIE] low. When interrupts are unmasked in {ICR}, [GIE] must be set high to allow interrupts to operate. [GIE] can be set and cleared by writing to it, or through a number of instructions including RET and EXX.

{IBR}—**Interrupt Base Register**—This register must be set to the appropriate base of the interrupt vector table located in data RAM. The DP8344 development card and monitor software expect [IBR] to be at H#1F, making the table begin at H#1F00. The monitor software can be used without the interrupt table at H#1F00, but doing so is simplest for this illustration.

{ICR}—**Interrupt Control Register**—This register contains both CPU and transceiver specific controls. From the

CPU point of view, the interrupt masks are located here. In this illustration, the system requires receiver, transmitter, BIRQ, and timer interrupts, so that in operation those interrupt bits should be unmasked. For initialization purposes, though, interrupts should be masked until their vectors are installed and the interrupt task is ready to be started. Therefore, loading [ICR] with H#7F is prudent. This also sets the receiver interrupt source, but that will be discussed in the next section.

TRANSCEIVER CONFIGURATION REGISTERS:

{TMR}—**Transceiver Mode Register**—This register controls the protocol selection, transceiver reset, loopback, and bit stream inversion. Loading this register with H#0D sets up the receiver in 5250 promiscuous mode, inverts serial data out, does not invert incoming serial data, does not allow the transmitter and receiver to be active at the same time, disables loopback, and does not reset the transceiver. Choosing to set [RIN] low assumes that serial data will be presented to the chip in NRZI form. Not allowing the receiver and transmitter to operate concurrently is not an issue in 5250 emulation, since there is no defined repeater function in the protocol as in 3270 (3299). Bits [B5, 6], [RPEN] and [LOOP] are primarily useful in self testing, where [LOOP] routes the transmitted data stream into the receiver and simultaneous operation is desirable. Please note that for loopback operation, [RIN] must equal [TIN]. [TRES] is used regularly in operation, but should be left off when not specifically needed.

{TCR}—**Transceiver Command Register**—This register has both configuration and operation orientated bits, including the transmitted address and parity bits. For this configuration, the register should be set to H#00 and the specific address needed summed into the three LSBs, as appropriate. The [SEC] or Select Error Codes bit is used to enable the {ECR} register through the {RTR} transceiver FIFO port, and should be asserted only when an error has been detected and needs to be read. [SLR], or Select Line Receiver is set low to enable the TTL-IN pin as the serial data in source. The BCP's on chip comparator is best suited to transformer coupled environments, and National's LM361 high speed differential comparator works very well for the twinax line interface. [ATA], or Advance Transmitter Active is normally used in the 3270 modes to change the form of the first line quiesce bit for transmission. Some twinax products use a long first line quiesce bit, although it is not necessary. The lower four bits in {TCR} are used to form the frame transmitted when data is written into {RTR}, the transceiver FIFO port. Writing into {RTR} starts the transmitter and/or loads the transmit FIFO. The least significant three bits in {TCR} form the address field in that transmitted frame, and B3, [OWP] controls the type of parity that is calculated and sent with that frame. [OWP] set to zero calculates even parity over the eight data bits, address and sync bit as defined in the IBM 5250 PAI.

{ATR}—**Auxiliary Transceiver Register**—Since this application is configured for promiscuous mode, the {ATR} register serves only to set the line hold function time. In non-promiscuous mode, the three least significant bits of this register are the selected address. Setting this register to H#50 allows a 5 μs hold time and clears the address field to 0, since promiscuous mode is used.

{**FBR**}—**Fill Bit Register**—This register controls the number of biphase zeros inserted between concatenated frames when transmitting. This register should be set upon reception of the SET MODE instruction from the host. {FBR} contains the one's complement of the number of inter-frame fill bits so that H#FF sends no extra fill bits.

{**ICR**}—**Interrupt Control Register**—As discussed in the CPU configuration section, this register sets [RIS] or Receiver Interrupt Select as well as the interrupt mask. Setting the register to H#7F selects [DA + ERR], Data Available or transceiver ERRor, as the interrupt source. This interrupt is asserted when either a valid frame has been clocked into the receive FIFO or an error has occurred. Other interrupt options are available including: [RA], Receiver Active; and [RFF + ERR], Receive FIFO Full or transceiver ERRor. For 5250 protocols the [DA + ERR] is most efficient. The [RFF + ERR] interrupt will not assert until the FIFO is full . . . regardless of whether the incoming message is single or multi-frame. [RA] provides plenty of notice that a frame is incoming, but due to the speed of the BCP, this advanced warning is not generally needed. [DA + ERR] provides a notification just after the parity bit has been decoded from the incoming frame which is almost 3 μs prior to the end of the frame. With the CPU running at 8 MHz, that allows typically nine instructions ([(4 * 3) − 3)]) for interrupt latency, trap and bank switch after interrupting.

MULTI-SESSION POWER

Handling multiple sessions in software is not trivial, and making the receiver service routines interrupt driven complicates the task further. The BCP is so fast, that at 8 MHz handling a multi-frame message by interrupting on the first frame and polling for succeeding frames is very inefficient. To maximize bandwidth for non-protocol related tasks, the CPU should handle each frame separately on interrupt and exit. To do this, a number of global state variables must be maintained. Since the alternate B register bank is primarily used for transceiver functions anyway, dedicating the other registers in that bank permanently as state variables is acceptable in most cases; doing so speeds and simplifies access to them. Defining the following registers as:

enables the software to keep track of the various states the protocol must handle.

The active address bits in GP5' allow individual addresses to be active, or any combination of addresses. When interrupted by a message to a non-selected address, [TRES] is toggled to reset the receiver until the beginning of the next message is detected. [B7] is used to determine if any particular address is "selected" and in the process of receiving data. The selected flag is set and cleared according to specific protocol rules set up in the IBM PAI.

Register GP6' contains the selected address storage [B0–2], where the address of the device expecting at least one other frame is stored when exiting the interrupt service routine, so that upon interruption caused by the reception of that frame, the address is still available. The received_EOM flag, [B3] is set when a message is decoded that contains B#111 or EOM delimiter. It is stored in this global status register to allow the protocol to determine the end of a transmission. In most multi-byte transmissions, the number of data frames expected is dictated by the protocol. However, ACTIVATE WRITE commands to printers can have any number of data frames associated with them up to 256. In this situation, the activated flag, [B4] is set to signal a variable length stream. Certain host devices also concatenate commands within messages, obscuring the determination of end of message. This scheme allows the software to keep track during such scenarios. The multi-count bits, [B6–7] are used in addition to the EOM delimiter to determine the end of a transmission. The number of additional frames expected in a given multi-byte command is written into these bits (note that a maximum of three bytes can be planned for in this way). When the count is terminated and no EOM delimiter is present, the algorithm then assumes a multi-command message is in progress. [B5] is unused.

Register GP7' is used to store the received data or error code for passage to other routines. The data can be passed on the stack, but dedicating a register to this function simplifies transactions in this case. Keeping track of received data is of utmost importance to communications devices.

GP5' — Active Address and Select Register

7 ... 0

- address 0 select
- address 1 select
- address 2 select
- address 3 select
- address 4 select
- address 5 select
- address 6 select
- device selected flag

TL/F/9635-7

GP6' — Global Status Register

7 ... 0

- selected address B0
- selected address B1
- selected address B2
- received_EOM flag
- activated flag
- unused
- multi-count B0
- multi-count B1
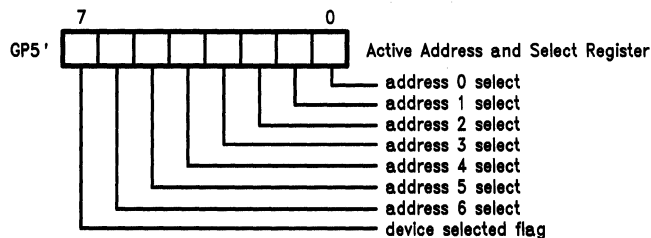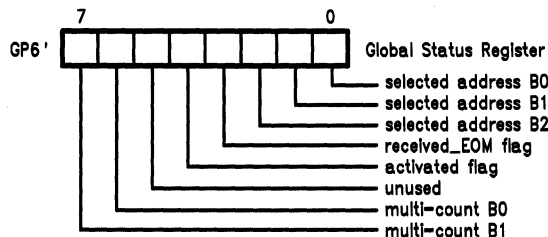
TL/F/9635-8

GP7'—Bits [0–7] Received Data or Error Register

RECEIVER INTERRUPT

The receiver interrupt algorithm handles any or all seven addresses possible on the twinax line. The same code is used for each address by utilizing a page oriented memory scheme. Session specific variables are stored in memory pages of 256 bytes each. All session control pages, or SCPs are on 256-byte even boundaries. By setting the high order byte of a BCP index register to point to a particular page or SCP, the low order byte then references an offset within that page. Setting up data memory in such a way that the first SCP begins at an address of B#xxxx x000 0000 0000 further enhances the usefulness of this construct. In this scheme, the high byte of the SCP base pointer can be used to set the particular SCP merely by summing the received or selected address into the lower three bits of the base register.

NORMAL OPERATION

In normal operation, the configuration described thus far is used in the following manner: After initializing the registers, data structures are initialized, and interrupt routines should be activated. This application utilizes the receiver, transmitter, timer, and bi-directional interrupts. Since {IBR} is set to H#1F, the interrupt table is located at H#1F00. A LJMP to the receiver interrupt routine should be installed at location H#1F104, the transmitter interrupt vector at H#1F08, the BIRQ interrupt vector at H#1F10, and the Timer interrupt vector at H#1F14. Un-masking the receiver interrupt and BIRQ at start up allows the device to come on-line.

When interrupt by the receiver, the receiver interrupt service routine first checks the [ERR] flag in {TSR [B5]}. If no errors have been flagged, the received__EOM flag is either set or cleared. This is accomplished by comparing {TSR [B0–2]} with the B#111 EOM delimiter. A test of the selected flag, {GP5' [B7]} determines if any of the active addresses are selected. Assuming that the system is just coming on line, none of the devices would be selected. If the frame is addressed to an active device, the SCP for that device is set, and the command is parsed. Parsing the command sets the appropriate state flags, so that upon exiting, the interrupt routine will be prepared for the next frame. Once parsed, the command can be further decoded and handled. If the command is queue-able, the command is pushed on the internal command queue, and the receiver interrupt routine exits. If the command requires an immediate response, then the response is formulated, the timer interrupt is setup, and the routine is exited.

The timer interrupt is used in responding to the host by waiting an appropriate time to invoke the transmit routine. The typical response delay is 45 ±15 μs after the last valid fill bit received in the command frame. Some printers and terminals are allowed a full 60 ±20 μs to respond. In either case, simply looping is very inefficient. The immediate response routine simply sets the timer for the appropriate delay and unmasks the timer.

In the transmit routine, the data to be sent is referenced by a pointer and an associated count. The routine loads the appropriate address in the three LSBs of {TCR}, and writes the data to be sent into {RTR}. This starts the transmitter. If the data count is greater than the transmit FIFO depth (three bytes), the Transmit FIFO Empty interrupt [TFE] is

setup. This vectors to code that refills the FIFO and re-enables that interrupt again, if needed. This operation must be carried out before the transmitter is finished the last frame in the FIFO or the message will end prematurely.

The last frame transmitted must contain the EOM delimiter. It can be loaded into {TCR} and data into {RTR} while the transmitter is running without affecting the current frame. In other words, the transmit FIFO is 12 bits wide, including address and parity with data; the address field is clocked along with the data field. In this way, multi-byte response may be made in efficient manner.

ERROR HANDLING

In 5250 environments, the time immediately after the end of message is most susceptible to transmission errors. The BCP's receiver does not detect an error after the end of a message unless transitions on the line continue for a complete frame time or resemble a valid sync bit of a multi-frame transmission. If the twinax line is still active at the end of what could be an error frame, the receiver posts the LMBT error. For example, if noise on the twinax line continues for up to 11 μs after the three required fill bits, the receiver will reset without flagging an error. If noise resembles a start bit, the receiver now expects a new frame and will post an error if a loss of synchronization occurs. If the noisy environment is such that transitions on the receiver's input continue for 11 μs, or the receiver really has lost sync on a real frame, the error is posted.

Basically, the receiver samples [LA] in addition to the loss of synchronization indication to determine when to reset or to post an error. After a loss of synchronization in the fill bit portion of a frame, if the [LA] flag's time-out of 2 μs is reached prior to the end of what could be the next frame, the receiver will reset. If the transitions prevent [LA] from timing out for an entire 11 μs frame time, a LMBT error is posted. This method for resetting the receiver is superior in that not only are the spurious loss of mid bit errors eliminated, the receiver performs better in noisy environments than other designs.

SUMMARY

The IBM 5250 twinax environment is less understood and in some ways more complex than the 3270 environment to many developers. This application note has attempted to explain some basics about twinax as a transmission medium, the hardware necessary to interface the DP8344 to that medium, and some of the features of the BCP that make that task easier. Schematics are included in this document to illustrate possible designs. Details of the twinax waveforms were discussed and figures included to illustrate some of the more relevant features. Also, some different software approaches to handling the transceiver interface were discussed.

REFERENCES

5250 Information Display to System/36 and System/38 System Units Product Attachment Information, IBM, November 1986.

Transmission Line Characteristics, Bill Fowler, National Semiconductor Application Note AN-108.

Basic Electromagnetic Theory, D.T. Paris, F.K. Hurd McGraw-Hill Inc., 1969.

APPENDIX A: EXAMPLE CODE

The following code was assembled with the HILEVEL assembler. Table II shows the correlation between HILEVEL mnemonics and the mnemonics used in National data sheets for the DP8344V.

TABLE II

HILEVEL	National Semiconductor
MOVE Rs,Rd	MOVE Rs,Rd
LD Ptr,Rd{,Mde}	MOVE [mIr],Rd
ST Rs,Ptr{,Mde}	MOVE Rs,[mIr]
LDAX Ptr,Rd	MOVE [Ir + A],Rd
STAX Rs,Ptr	MOVE Rs,[Ir + A]
LDNZ n,Rd	MOVE [IZ + n],rd
STNZ Rs,n	MOVE rs,[IZ + n]
LDI n,Rd	MOVE n,rd
STI n,Ptr	MOVE n,[Ir]
ADD Rs,Rd	ADDA Rs,Rd
ADDRI Rs,Ptr{,Mde}	ADDA Rs,[mIr]
ADDI n,Rsd	ADD n,rsd
ADC Rs,Rd	ADCA Rs,Rd
ADCRI Rs,Ptr{,Mde}	ADCA Rs,[mIr]
SUBT Rs,Rd	SUBA Rs,Rd
SUBRI Rs,Ptr{,Mde}	SUBA Rs,[mIr]
SUBI n,Rsd	SUB n,rsd
SBC Rs,Rd	SBCA Rs,Rd
SBCRI Rs,Ptr{,Mde}	SBCA Rs,[mIr]
AND Rs,Rd	ANDA Rs,Rd
ANDRI Rs,Ptr{,Mde}	ANDA Rs,[mIr]
ANDI n,Rsd	AND n,rsd
OR Rs,Rd	ORA Rs,Rd
ORRI Rs,Ptr{,Mde}	ORA Rs,[mIr]
ORI n,Rsd	OR n,rsd
XOR Rs,Rd	XORA Rs,Rd
XORRI Rs,Ptr{,Mde}	XORA Rs,[mIr]
XORI n,Rsd	XOR n,rsd
CMP Rs,n	CMP rs,n
CPL Rsd	CPL Rsd
BIT Rs,n	BIT rs,n
SRL Rsd,n	SHR Rsd,b
SLA Rsd,n	SHL Rsd,b
ROT Rsd,n	ROT Rsd,b

TL/F/9635–9

2

```
JMP   n                   JMP   n
LJMP  n                   LJMP  nn
JMPR  Rs                  JMP   Rs
JMPI  Ptr                 LJMP  [Ir]
JRMK  Rs,n,m              JRMK  Rs,b,m
JMPB  Rs,s,p,n            LJMP  Rs,p,s,nn
JMPF  s,f,n               JMP   f,s,n
                         Jcc   n    - opt. syntax for JMP f-


CALL  n                   CALL  n
LCALL n                   LCALL      nn
CALLB Rs,s,p,n            LCALL      Rs,p,s,nn

RET   {g{,rf}}            RET   {g {,rf}}
RETF  s,f{,g{,rf}}        RETF  f,s,{,{g} {,rf}}
                         Rcc   {g {,rf}}   - opt. syntax -


EXX   a,b{,g}            EXX   ba,bb,{,g}


TRAP  n{,g}             TRAP  n {,gU}
```

Table 2.

```
Addr    Line

          1            .REL
          2     TAB    8
          3     WIDTH  132
          4     LIST   S,F
          5     TITLE  RXINT
          6     ;----------------------------------------------------------------
          7     ;      RXINT - 9/21/87
          8     ;
          9     ;      pseudo code
         10     ;
         11     ;bool       selected;                     /* station is selected
         12     ;byte       seladdr;                      /* address of selected station
         13     ;byte       multicount;                   /* number of frames in this multi
         14     ;bool       activated;                    /* command has been activated
         15     ;
         16     ;rxint()
         17     ;byte       data;                         /* data storage
         18     ;bool       rx_eom;                       /* received EOM
         19     ;bool       lta;                          /* line turn around flag
         20     ;{
         21     ;      if (error) {
         22     ;              if (logerror()== true) return;     /* receiver errors
         23     ;              }
         24     ;      else {
         25     ;              if (TSR == EOM) rx_eom = true;     /* set received EOM flag
         26     ;              else rx_eom = false;
         27     ;
         28     ;              if (!selected) {
```

TL/F/9635-10

2-256

```
 29    ;                       if (active) {
 30    ;                               if (!rx_eom) {
 31    ;                                       seladdr = (TSR * EOM);
 32    ;                                       IZ = (SCPBASE + seladdr); /* set SCP to appropriate session */
 33    ;                                       data = rtr;
 34    ;                               else {
 35    ;                                       proto_error(); /* should not get here
 36    ;                                       reset_xcvr();
 37    ;                                       return();
 38    ;                                       }
 39    ;                               }
 40    ;                       else {
 41    ;                               reset_xcvr();          /* not of interest
 42    ;                               return();
 43    ;                               }
 44    ;                       if (multiframe) {              /* activate write, etc...
 45    ;                               multicount = parse(data); /* set number of frames */
 46    ;                               selected = true;       /* only way to select   */
 47    ;                               queue(data);
 48    ;                               }
 49    ;                       else {                         /* not multi
 50    ;                               if ((var = single_decode(data)) == queable)
 51    ;                                       queue(data);
 52    ;                               else if (var == immed) immediate(data);
 53    ;                       }
 54    ;               else {                                 /* selected        */
 55    ;                       IZ = (SCPBASE + seladdr);

Addr    Line RXINT

 56    ;                       data = rtr
 57    ;               if (activated) {                       /* in the middle of transmission
 58    ;                   act_data(data);
 59    ;                   if (rx_eom) {                      /* end of message
 60    ;                       selected = false;
 61    ;                       activated = false;
 62    ;                       }
 63    ;                   return();
 64    ;                   }
 65    ;                   if (multicount > 0) {
 66    ;                           queue(data);
 67    ;                           if (multicount-= 0) {
 68    ;                                   if (rx_eom) selected = false;
 69    ;                                   }
 70    ;                           }
 71    ;                   else {
 72    ;                           if (multiframe) {
 73    ;                                   multicount = parse(data);
 74    ;                                   queue(data);
 75    ;                                   }
 76    ;                           else {
 77    ;                                   if ((var = single_decode(data)) == queable)
 78    ;                                           queue(data);
```

TL/F/9635-11

2

```
 79   ;                                      else if (var == immed) immediate(data);
 80   ;                                      if (rx_eom) selected = false;
 81   ;                                      }
 82   ;                                  }
 83   ;                              }
 84   ;                      }
 85   ;              }
 86   ;          return();
 87   ;}
 88   ;logerror()
 89   ;{
 90   ; bool result;
 91   ;      switch (error_type) {
 92   ;      case RDIS:
 93   ;          result = err_rdis();                    /* receiver diabled while active
 94   ;          break;
 95   ;      case LMBT:
 96   ;          result = err_lmbt();                    /* loss of midbit error
 97   ;          break;
 98   ;      case PARR:
 99   ;          result = err_parr();                    /* parity error
100   ;          break;
101   ;      case OVF:
102   ;          result = err_ovf();                     /* receiver FIFO overrun
103   ;          break;
104   ;      default:
105   ;          result = err_unknown();                 /* strange error handler
106   ;          break;
107   ;      }
108   ;      return(result);
109   ; }
110   ;
```

Addr   Line RXINT

```
111   ;err_lmbt()
112   ;{
113   ;      if (!DA && !selected && !delay(LA)) return(false); /* delay of 6 usec
114   ;      else {
115   ;          log();                              /* bump error counters
116   ;          return(true);                       /* admit defeat
117   ;          }
118   ; }
119   ;-----------------------------------------------------------------
120   ;      name:           RXINT
121   ;      description:    receiver interrupt handler
122   ;
123   ;              received datum is sent to other routines thru gp7'
124   ;              SCP is set appropriately in IZ
125   ;              GP5P - active addresses:bits 0-6
126   ;                     selected flag:   bit 7
127   ;              GP6P - multicount:       bit 7-6
128   ;                     unused:          bit 5
```

TL/F/9635-12

```
                 129   ;                        activated:    bit 4
                 130   ;                        rx_eom flag:  bit 3
                 131   ;                        seladdr:      bits 2-0
                 132   ;              GP7P - received data
                 133   ;
                 134   ;      entry:     DA interrupt, GP5', GP6'
                 135   ;      exit:      ACC',GP7' ARE DESTROYED
                 136   ;      history:   tjq  9/16/87   create
                 137   ;-------------------------------------------------------------
                 138          PUBLIC  RCVRINT
                 139
                 140          EXTRN   PARSE,QUEUE,IMMEDECODE,RESXCVR
                 141          EXTRN   MIDERRL,MIDERRH,OVFERRL,OVFERRH,PARERRL,PARERRH
                 142          EXTRN   RXERRL,RXERRH,RSPCTL,RSPCTH,BASESCP,IESERRL,IESERRH
                 143
                 144
                 145   SELERR: EQU    B#01000000       ; select the error register
                 146   RXEOM:  EQU    B#00001000       ; rxeom flag
                 147   EOM:    EQU    B#00000111       ; EOM delimeter
                 148   MULTI:  EQU    B#11000000       ; multicount
                 149   SELECT: EQU    B#10000000       ; selected flag
                 150   LTA:    EQU    B#101            ;       "
                 151   CFLAG:  EQU    B#00000010       ; CARRY FLAG
                 152
00000            153   RCVRINT:
                 154          EXX     MA,AB,DI         ; SET APPROPRIATE BANK
00000 AEE8       154
00001 D500       155          JMPF    NS,RERR,NOERROR
00002 CC00       156          CALL    RXERROR          ; ERROR IN FRAME
00003 D900       157          JMPF    S,C,EXIT         ; ABORT
00004 D900       158   NOERROR:
00004 B07B       159          LDI     EOM,ACC          ; LOAD MASK
                 160          AND     TSR,GP7          ; FORM ADDRESS
00005 F165       160
                 161          CMP     GP7,EOM          ; TEST
00006 307B       161
                 162          JMPF    NS,Z,C1RXINT     ; IF NOT EQUAL, JUMP

 Addr     Line RXINT

0000B 508A       163          ORI     RXEOM,GP6        ; ELSE SET EOM FLAG
00009 CB00       164          JMP     C2RXINT          ;
0000A CB00       165   C1RXINT:
0000A 4F7A       166          ANDI    RXEOM*,GP6       ; CLEAR IT
                 167   ;
                 168   ; DECIDE IF WE'RE ALREADY SELECTED
                 169   ;
0000B            170   C2RXINT:
                 171          JMPB    GP5,S,B7,DEVSELECT ; IF ALREADY SELECTED
0000B BDE9       171
0000C 0000       171
                 172   ;
                 173   ; NOT SELECTED...DECIDE IF ADDRESS IS ACTIVE, IE; VALID FOR US
```

TL/F/9635–13

2

```
           174   ;
0000D      175   DEVTABLE:                        ; ELSE, SEE IF ACTIVE
           176          JRMK    TSR,ROT6,MSK3    ; JUMP BASED ON THE ADDRESS FIELD*4
0000D 83C5 176
           177          JMPB    GP5,NS,B0,RSTRX  ; ADDR 0 - IF NOT ACTIVE, RESET RX
0000E 8C09 177
0000F 0000 177
           178          LJMP    LOADSCP          ; ACTIVE DEVICE, SET scp
00010 CE00 178
00011 0000 178
           179          JMPB    GP5,NS,B1,RSTRX  ; ADDR 1 - IF NOT ACTIVE, RESET RX
00012 8C29 179
00013 0000 179
           180          LJMP    LOADSCP          ; ACTIVE DEVICE, SET scp
00014 CE00 180
00015 0000 180
           181          JMPB    GP5,NS,B2,RSTRX  ; ADDR 2 - IF NOT ACTIVE, RESET RX
00016 8C49 181
00017 0000 181
           182          LJMP    LOADSCP          ; ACTIVE DEVICE,
00018 CE00 182
00019 0000 182
           183          JMPB    GP5,NS,B3,RSTRX  ; ADDR 3 - IF NOT ACTIVE,
0001A 8C69 183
0001B 0000 183
           184          LJMP    LOADSCP          ; ACTIVE DEVICE,
0001C CE00 184
0001D 0000 184
           185          JMPB    GP5,NS,B4,RSTRX  ; ADDR 4 - IF NOT ACTIVE,
0001E 8C89 185
0001F 0000 185
           186          LJMP    LOADSCP          ; ACTIVE DEVICE,
00020 CE00 186
00021 0000 186
           187          JMPB    GP5,NS,B5,RSTRX  ; ADDR 5 - IF NOT ACTIVE,
00022 8CA9 187
00023 0000 187
           188          LJMP    LOADSCP          ; ACTIVE DEVICE,
00024 CE00 188
00025 0000 188
           189          JMPB    GP5,NS,B6,RSTRX  ; ADDR 6 - IF NOT ACTIVE,
00026 8CC9 189

 Addr      Line RXINT

00027 0000 189
           190          LJMP    LOADSCP          ; ACTIVE DEVICE,
00028 CE00 190
00029 0000 190
           191          LCALL   RESXCVR          ; ADDR 7 - RECEIVED EOM ...WE'RE NOT INTERESTED
0002A CE80 191
0002B 0000 191
0002C CB00 192   JMP     EXIT             ; QUIT
```

TL/F/9635–14

```
              193    ;
              194    ; LOAD THE SCP POINTER, IZ
              195    ;
0002D         196    LOADSCP:
              197            XOR     ACC,ACC         ; CLEAR
0002D F908    197
              198            MOVE    ACC,ZLO         ; LOW BYTE
0002E FE48    198
0002F B008    199            LDI     BASESCP,ACC     ; SET UP UPPER BYTE OF SCP POINTER
              200            MOVE    ACC,ZHI         ;
00030 FE68    200
00031 B07B    201            LDI     EOM,ACC         ; EOM MASK
              202            AND     TSR,ACC         ; LEAVE IN ACC
00032 F105    202
              203            ADD     ZHI,ZHI         ; ADD INTO Z POINTER
00033 E273    203
              204    ;
              205    ; DECODE THE COMMAND FRAME
              206    ;
00034         207    DECODE:
              208            MOVE    RTR,GP7         ; GET RX DATA
00034 FD64    208
              209            JMPB    GP7,S,B0,MULTIFRM; IF MULTIFRAME
00035 BD0B    209
00036 0000    209
              210            LCALL   IMMEDECODE      ; ELSE, IMMEDIATE ACTION REQUIRED
00037 CEB0    210
0003B 0000    210
00039 CB00    211            JMP     EXIT
0003A CB00    212    MULTIFRM:
              213            LCALL   PARSE           ; SET MULTI COUNT
0003A CEB0    213
0003B 0000    213
0003C 5809    214            ORI     H#80,GP5        ; SELECTED = TRUE
0003D 4FBA    215            ANDI    EOM*,GP6        ; CLEAR SELECTED ADDRESS
0003E B07B    216            LDI     EOM,ACC         ; MASK ADDRESS
              217            AND     TSR,ACC         ; LEAVE IN ACC
0003F F105    217
              218            OR      GP6,GP6         ; SET NEW ADDRESS
00040 F54A    218
              219            LCALL   QUEUE           ; PLACE ON QUEUE
00041 CEB0    219
00042 0000    219
00043 CB00    220            JMP     EXIT            ;
              221    ;
              222    ; THIS CODE IS BRANCHED TO IF THE DEVICE IS SELECTED
              223    ;      FIRST, SET SCP BASED ON SELECTED ADDRESS

  Addr   Line RXINT

              224    ;
00044         225    DEVSELECT:
              226            XOR     ACC,ACC         ; CLEAR ACC
```

TL/F/9635–15

2

```
00044 F908    226
              227            MOVE    ACC,ZLO       ; CLEAR LOW BYTE OF POINTER
00045 FE48    227
00046 B008    228            LDI     BASESCP,ACC   ; BASE OF SESSION CONTROL PAGE
              229            MOVE    ACC,ZHI       ; UPPER BYTE
00047 FE68    229
00048 B07B    230            LDI     EOM,ACC       ; MASK ADDRESS
              231            AND     GP6,ACC       ; LEAVE IN ACC
00049 F10A    231
              232            ADD     ZHI,ZHI       ; FORM SCP POINTER
0004A E273    232
              233     ;
              234     ; NOW DECIDE ABOUT MULTIFRAME POSSIBILITIES
              235     ;
              236            MOVE    RTR,GP7       ; GET DATA
0004B FD64    236
0004C BC08    237            LDI     MULTI,ACC     ; MULTI MASK
              238            AND     GP6,ACC       ; COUNT IN UPPER NIBBLE
0004D F10A    238
              239            SRL     ACC,ROT6      ; POSITION IN LOWER NIBBLE
0004E C8C8    239
0004F D800    240            JMPF    S,Z,NEWCOMM   ; NOT in A MULTIBYTE
              241            LCALL   QUEUE         ; MULTI, SO PUSH ON QUEUE
00050 CEB0    241
00051 0000    241
00052 201B    242            SUBI    H#01,ACC      ; DECREMENT MULTICOUNT
00053 D800    243            JMPF    S,Z,TERMULTI  ; IF ZERO, MULTI HAS TERMINATED
              244     ;
              245     ; MULTI STILL IN PROGRESS
              246     ;
00054 43FA    247            ANDI    MULTI*,GP6    ; CLEAR OUT OLD COUNT
              248            SLA     ACC,ROT6      ; REPOSITION COUNT
00055 C948    248
              249            OR      GP6,GP6       ; SUM INTO STATUS
00056 F54A    249
00057 CB00    250            JMP     EXIT
              251     ;
              252     ; MULTICOUNT HAS REACHED ZERO, SO TERMINATE
              253     ;
0005B         254     TERMULTI:
0005B 43FA    255            ANDI    MULTI*,GP6    ; CLEAR OLD COUNT TO ZERO
              256            JMPB    GP6,NS,B3,C1TERM; IF NOT EOM,
00059 8C6A    256
0005A 0000    256
0005B 47F9    257            ANDI    SELECT*,GP5   ; ELSE, SELECT = FALSE
0005C CB00    258            JMP     RSTRX         ; RESET THE TRANSCEIVER
0005D CB00    259     C1TERM:
0005D CB00    260            JMP     EXIT
              261     ;
              262     ; NEW COMMAND; MULTI OR SINGLE
              263     ;
0005E         264     NEWCOMM:
```

TL/F/9635–16

```
Addr      Line RXINT

               265          JMPB    GP7,NS,B0,SINGLE; IF NEW COMMAND IS NOT MULTI,
0005E BC0B     265
0005F 0000     265
               266          LCALL   PARSE          ; IS MULTI, SET COUNT
00060 CE80     266
00061 9000     266
               267          LCALL   QUEUE          ; PUSH ON QUEUE
00062 CE80     267
00063 0000     267
00064 CB00     268          JMP     EXIT           ; QUIT, TIL NEXT FRAME
               269    ;
               270    ; NEW COMMAND IS SINGLE AND/OR NEEDS IMMEDIATE RESPONSE
               271    ;
00065          272    SINGLE:
               273          LCALL   IMMEDECODE     ; SINGLE...GO DO IT
00065 CE80     273
00066 0000     273
               274          JMPB    GP6,NS,B3,EXIT ; IF NOT EOM...
00067 8C6A     274
00068 0000     274
00069 47F9     275          ANDI    SELECT*,GP5    ; CLEAR SELECTED BIT
0006A 47F9     276    RSTRX:
               277          LCALL   RESXCVR        ; RESET, CLEAR DATA OUT
0006A CE80     277
0006B 0000     277
0006C 0000     278    EXIT:
0006C AFB0     279          RET     RI,RF          ; RETURN GRACEFULLY
               280
               281    ;----------------------------------------------------------------
               282    ;    name:         RXERROR
               283    ;    description:  receiver ERROR handler
               284    ;
               285    ;    entry:        DA + ERR interrupt, GP5', GP6'
               286    ;    exit:         ACC',GP7' ARE DESTROYED
               287    ;    history:      tjq 9/16/87   create
               288    ;----------------------------------------------------------------
               289    ;
               290    ; RECEIVER ERROR HANDLER
               291    ;
0006D          292    RXERROR:
0006D 5406     293          ORI     SELERR,TCR     ; SET ECR BIT
               294          MOVE    RTR,GP7        ; GET ERROR TYPE
0006E FD64     294
0006F 4BF6     295          ANDI    SELERR*,TCR    ; RESET TCR
               296          JMPB    GP7,S,B1,LMBTERR; LOSS OF MIDBIT
00070 8D2B     296
00071 0000     296
               297          JMPB    GP7,S,B3,PARERR ; PARITY
00072 8D6B     297
00073 0000     297
               298          JMPB    GP7,S,B4,OVFERR ; OVERFLOW
```

TL/F/9635–17

2

```
00074 8D8B    298
00075 0000    298
00076 0000    299    ILLEGAL:
00076 B00B    300           LDI    ILLEGAL,ACC    ; WHAT ERROR IS THIS?


   Addr       Line RXINT


00077 CB00    301           JMP    BUMPERR        ; SHOULD NOT GET HERE!!
0007B CB00    302    LMBTERR:
0007B DE00    303           JMPF   S,DA,CLEARC    ; if DA, THEN NO ERROR
              304           JMPB   GP5,S,B7,LOGIT ; IF SELECTED, POST
00079 8DE9    304
0007A 0000    304
0007B CC00    305           CALL   SDLY           ; DELAY FOR 6 USEC
              306           JMPB   NCF,NS,B5,CLEARC; IF NOT ACTIVE - DISCARD, ELSE POST
0007C 8CA1    306
0007D 0000    306
0007E 0000    307    LOGIT:
0007E B008    308           LDI    MIDERRL,ACC    ; LOSS OF MIDBIT
0007F CB00    309           JMP    BUMPERR        ; INCREMENT COUNTER
00080 CB00    310    PARERR:
00080 B008    311           LDI    PARERRL,ACC    ; PARITY
00081 CB00    312           JMP    BUMPERR
00082 CB00    313    OVFERR:
00082 B008    314           LDI    OVFERRL,ACC    ; OVERFLOW...VERY BAD!
00083 B008    315    BUMPERR:
              316           ADD    ZLO,YLO        ; FORM NEW POINTER
00083 E212    316
00084 B018    317           LDI    H#01,ACC       ; INCREMENT
              318           LD     PTRY,GP6       ; FETCH OLD COUNT
00085 COCA    318
              319           ADDRI  GP6,PTRY,POSTD ; WRITE OUT NEW
00086 A04A    319
00087 D100    320           JMPF   NS,C,RXEXIT    ; GET OUT
              321           LD     PTRY,GP6       ; FETCH UPPER BYTE
00088 COCA    321
              322           ADDRI  GP6,PTRY       ;
00089 AOCA    322
0008A 5020    323           ORI    CFLAG,CCR      ; SET CARRY
0008B 5020    324    RXEXIT:
0008B AFB0    325           RET                   ; DO NOT restore flags
0008C AFB0    326    CLEARC:
0008C 4FD0    327           ANDI   CFLAG*,CCR     ; CLEAR CARRY
0008D CB00    328           JMP    RXEXIT
              329    ;-------------------------------------------------------------
              330    ;      name:        SDLY
              331    ;      description: delay routine, MULTIPLES OF 4.8usec,
              332    ;                   1.4 usec OVERHEAD, MAX OF 410usec
              333    ;      entry:       delay count on stack
              334    ;      exit:        acc destroyed
              335    ;      WARNING:     DONT CALL THIS WITH COUNT = 0!
              336    ;      history:     tjq 9/16/87   create
              337    ;-------------------------------------------------------------
```

TL/F/9635–18

```
         338
0008E    339    SDLY:
         340         EXX    MA,MB,NAI    ; BANK, ALLOW INTERRUPTS
0008E AE80 340
         341         MOVE   DS,ACC       ; GET COUNT
0008F FD1F 341
         342         MOVE   GP7,DS       ; PUSH GP7 REGISTERS USED
00090 FFEB 342
         343         MOVE   GP6,DS
 Addr    Line RXINT

00091 FFEA 343
         344         MOVE   ACC,GP7      ; USE GP7 FOR COUNT ALSO
00092 FD68 344
00093 FD68 345    SDLYLP1:
00093 B03A 346         LDI    H#03,GP6     ; LOAD FOR 4.8usec COUNTS
00094 B03A 347    SDLYLP2:
00094 201A 348         SUBI   H#01,GP6     ; DECREMENT COUNT
00095 D000 349         JMPF   NS,Z,SDLYLP2 ; CONTINUE UNTIL EXHAUSTED
00096 201B 350         SUBI   H#01,GP7     ; DECREMENT OUTER COUNT
00097 D000 351         JMPF   NS,Z,SDLYLP1 ; CONTINUE IF NOT ZERO
         352         MOVE   DS,GP6       ; POP REG
00098 FD5F 352
         353         MOVE   DS,GP7       ;
00099 FD7F 353
0009A AFB0 354         RET    RI,RF        ; RETURN, RESTORE FLAGS
         355
         356
         357         END
```

Assembly Phase complete.
   0 error(s) detected.

TL/F/9635-19

2

DP8344 BCP Stand-Alone Soft-Load System

National Semiconductor
Application Note 504
Jim Margeson

INTRODUCTION

The DP8344 Biphase Communications Processor (BCP) is a 20 MHz Harvard architecture microprocessor with an on-chip transmitter and receiver. The BCP can be used to implement several biphase communication protocols: IBM 3270, IBM 3299, IBM 5250, and National's general purpose 8-bit protocol. This application note shows how DP8344 software can be loaded from EPROM into instruction RAM. It is particularly valuable in stand-alone systems where the BCP is not interfaced to a host processor. Possible applications include: protocol converters, multiplexers, high-speed remote data acquisition systems and remote process control systems.
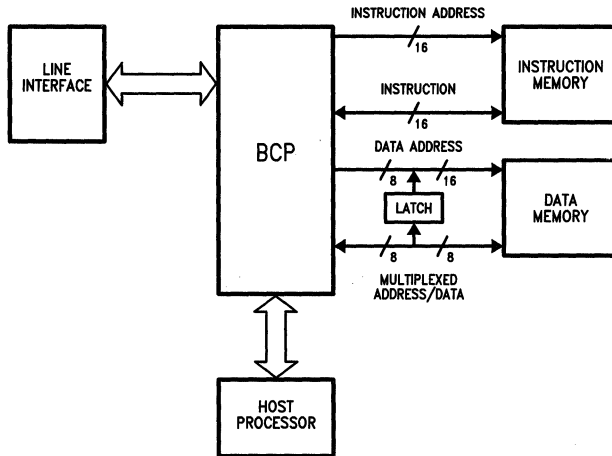
TL/F/9403–1

FIGURE 1. BCP System with Host Processor

TL/F/9403–2
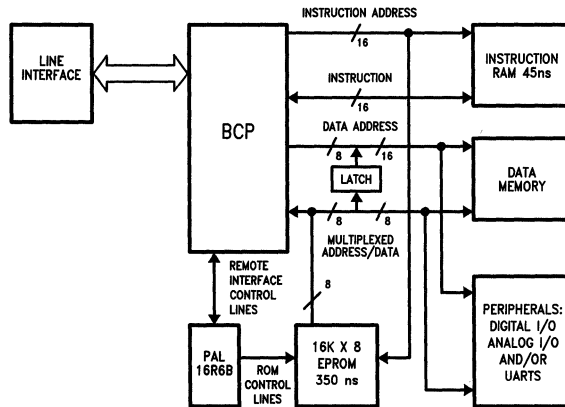
FIGURE 2. BCP Stand-Alone System with EPROM Soft Load Circuit
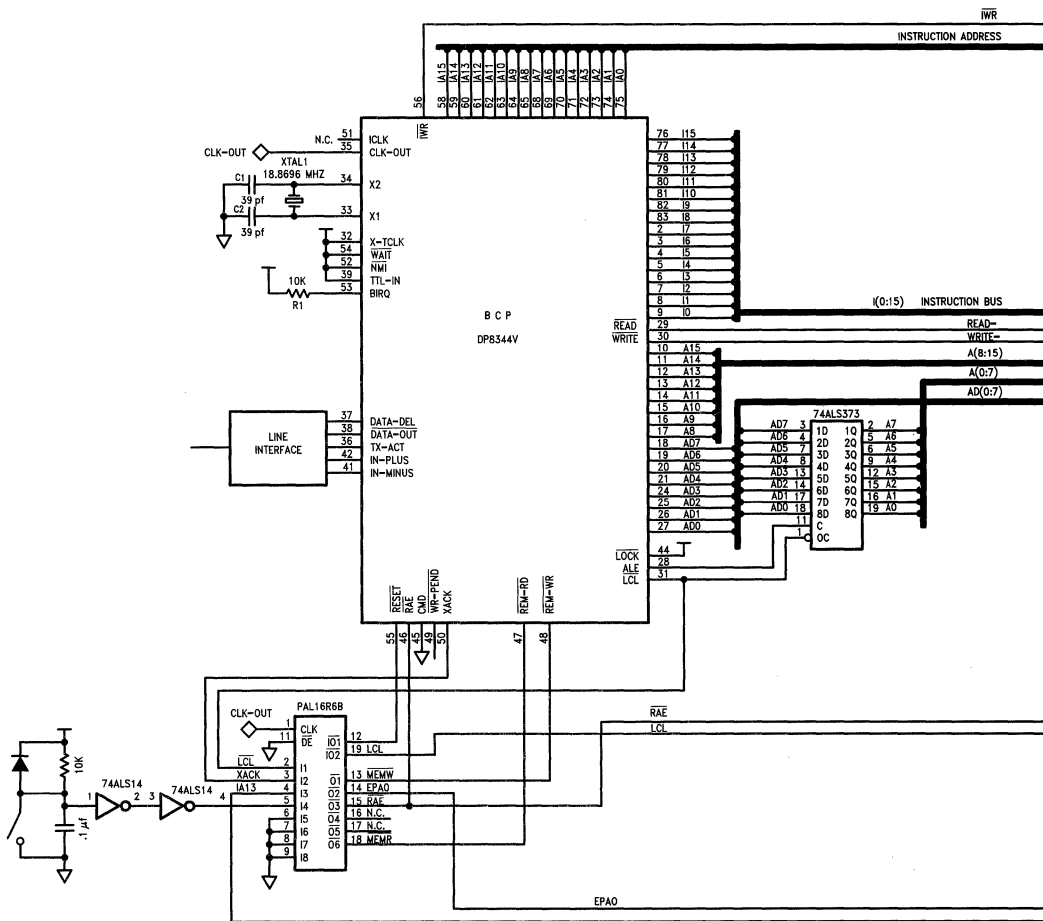
FIGURE 3. Schematic

TL/F/9403–3
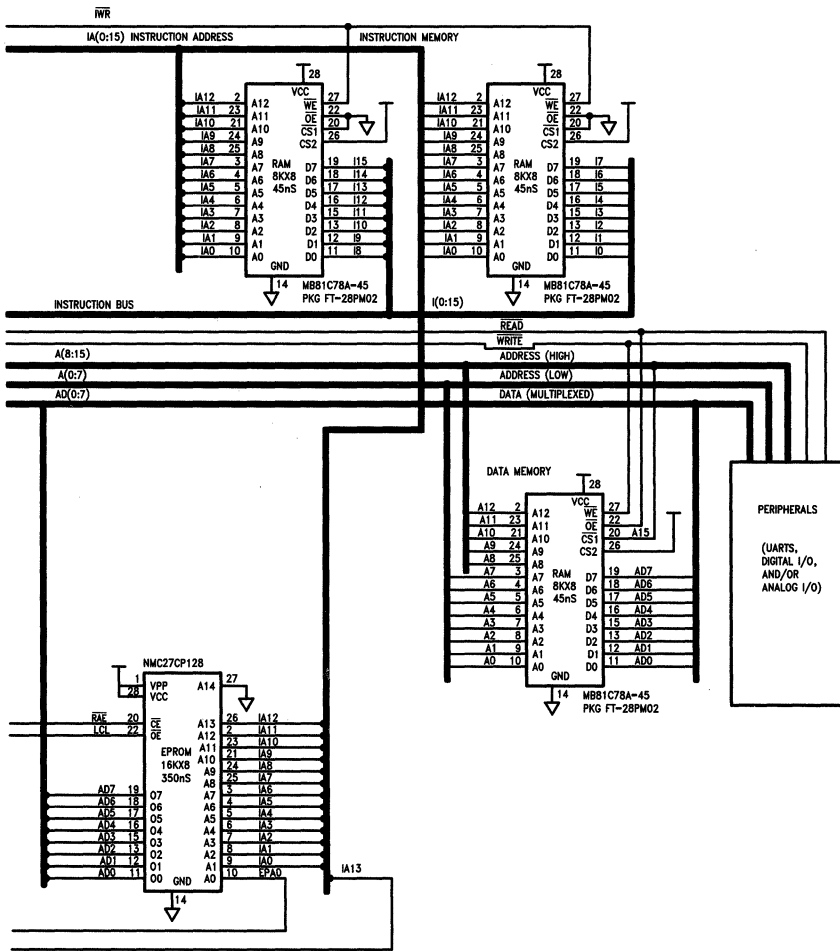
FIGURE 3. Schematic (Continued)

TL/F/9403–4

WHY EPROM SOFT-LOAD?

In a stand-alone application, the BCP instruction code must be kept in non-volatile memory. Instruction memory with 45 ns access time is required to run the BCP at full speed.

EPROM at this speed can be quite expensive, much more than 45 ns RAM or 350 ns EPROM. RAM with 45 ns access time can be used for instruction memory if a scheme is employed to load the BCP code into the RAM from slow (350 ns), inexpensive EPROM, upon power-up.

In non-stand-alone applications, a host processor would communicate with the BCP through the BCP's built-in remote interface (Figure 1). In such a system, BCP code would be loaded from the host into the BCP's instruction RAM using the remote interface. In a stand-alone system, however, the BCP is not interfaced to a host; the program is loaded from EPROM through the remote interface. As shown in Figure 2 a PAL® sequencer controls the loading of the program, generating handshaking signals similar to those of a typical host processor. When the load is complete, the sequencer tells the BCP to begin execution of the program.

HOW THE SOFT-LOAD CIRCUIT WORKS

The BCP, as configured in this system, comes up halted after reset (Figure 3). The program counter is set to zero, and the remote interface is configured to receive 16-bit instructions in 8-bit pieces and write them into instruction memory. The BCP has the feature that it can be configured to come up stopped or to begin program execution after a reset has occurred. If the following conditions are true when reset is de-asserted then the processor will begin running: RAE~ (Remote Access Enable, active low) = High, REMWR~ (Remote Write, active low) = low, REMRD~ (Remote Read, active low) = low. Otherwise, it will come up halted.

The PAL sequencer begins the software load by writing the low byte of the first instruction to the remote interface. A simplified flowchart of the sequence operation is shown in Figure 4.

This byte comes from address 0000H of the EPROM. The corresponding locations of EPROM and RAM are shown in Figure 5. The least significant address line of the EPROM is controlled by the sequencer; the other address lines are driven by the instruction address bus of the BCP. The instruction address bus reflects the contents of the BCP's program counter (PC), which contains the destination of the instruction currently being loaded. After the low byte of the first instruction is written to the remote interface, the sequencer brings the least significant address line of the EPROM high. Now location 0001H of the EPROM is addressed, and the high byte of the first instruction is written to the remote interface. At this point the BCP writes both bytes into address 0000H of instruction RAM, and increments its program counter.
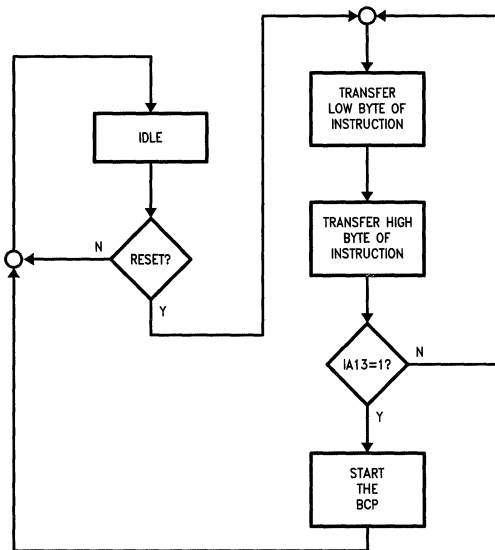
TL/F/9403–5

FIGURE 4. Sequencer Operation

2

EPROM Address	Instruction Memory Address	
0	0	(Low Byte)
1	0	(High Byte)
2	1	(Low Byte)
3	1	(High Byte)
4	2	(Low Byte)
5	2	(High Byte)
•	•	•
•	•	•
•	•	•
•	•	•
16382	8190	(Low Byte)
16383	8191	(High Byte)

FIGURE 5. EPROM to RAM Address Mapping

The first 16-bit instruction has been transferred; the second is done in a similar manner. The sequencer brings the least significant address line of the EPROM low again. The PC now contains 0001H, which is output on the instruction address bus. Location 0002H of the EPROM is addressed, and the low byte of the second instruction is written to the remote interface. The sequencer then brings the least significant address line of the EPROM high (to address location 0003H) and the high byte of the second instruction is transferred. The BCP writes the second 16-bit instruction to location 0001H of instruction RAM. This process is repeated until the last instruction is transferred.

The sequencer senses that the load is complete when instruction address line 13 comes high. This occurs when the program counter is incremented to a value of 4000H, indicating that 8K instruction words have been transferred. At this point the BCP must be started. To achieve this, the sequencer resets the BCP again, while holding RAE~ high, REMRD~ low, and REMWR~ low. A reset during these conditions brings the processor up running, and also clears the program counter. The BCP begins execution at instruction address 0000H and the sequencer and EPROM go into an inactive state, transparent to the software being executed. A detailed version of the sequencer flowchart is shown in *Figure 6*. A hardware compiler/minimizer was used to obtain the equations shown in *Figure 7*. These equations were used to program a National PAL16R6B. Typical timing waveforms of the soft-load are shown in *Figure 8*.
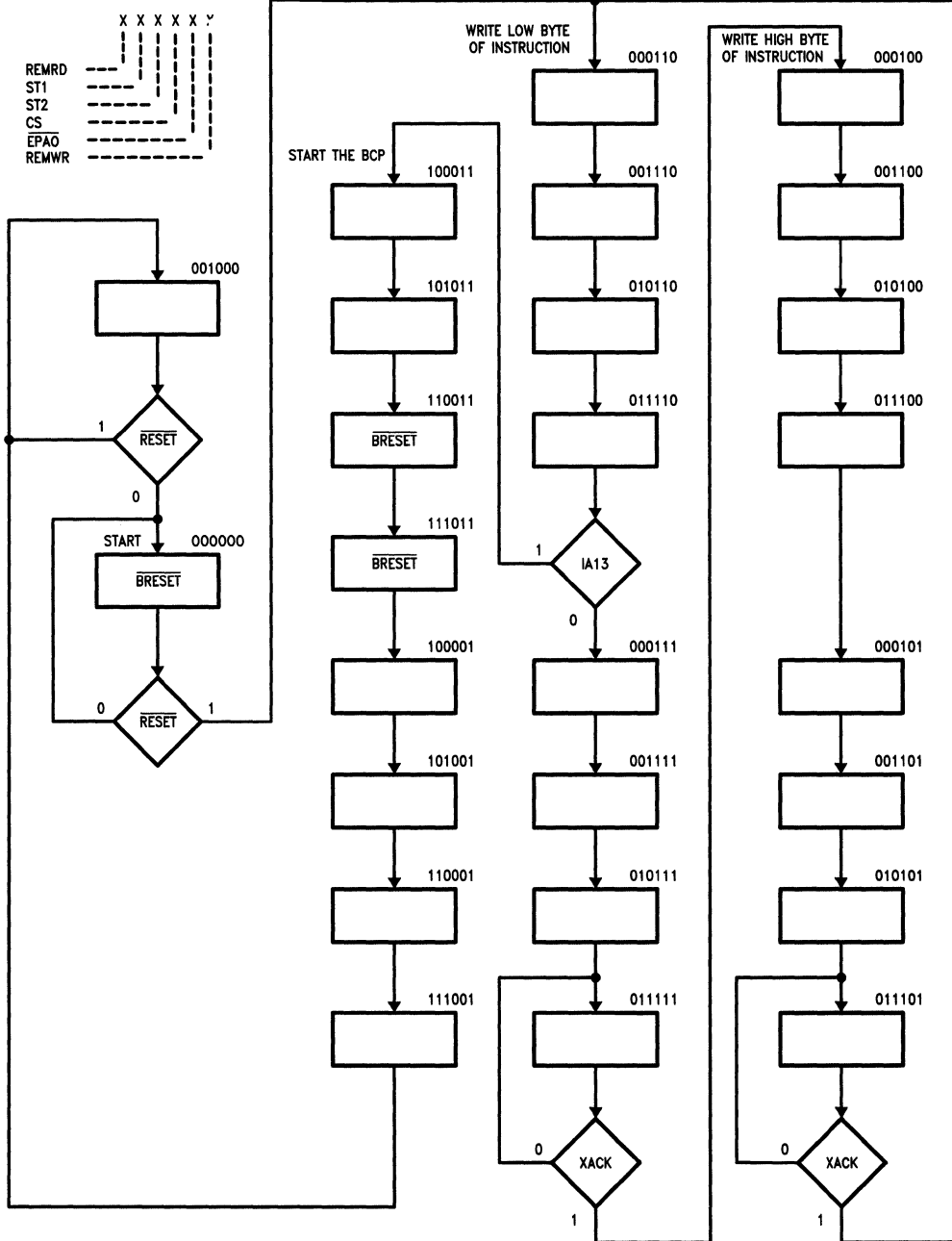
FIGURE 6. Sequencer Flowchart

TL/F/9403–8

2

There are several advantages to using the remote interface to load the BCP software. If a scheme like the one in *Figure 9* was used to load the program directly from EPROM to instruction RAM, much more hardware would be required and the access time of the RAM would need to be shorter. Two EPROMs would have to be used instead of one because the transfer would be 16 bits wide instead of 8 bits. In this case the BCP's program counter could not be used to increment through the memory locations, thus an external 13-bit counter would be needed. TRI-STATE® buffers would isolate the RAM and EPROM from the instruction data and instruction address busses during soft-load. These buffers would add propagation delays to memory accesses demanding that faster RAM be used. Soft-loading through the remote interface requires fewer I.C.'s and does not degrade the performance of the processor.

```
DMPAL16R6B;   SOFTLOAD
CK LCL XACK IA13 RESET NC6 NC7 NC8 IWR GND
/OE /BRESET /REMWR /EPAO /CS /ST2 /ST1 /REMRD /LCLINV VCC
/REMRD := RESET*      /REMRD*           CS*/EPAO*/REMWR
       + RESET*       /REMRD*      ST2* CS*       /REMWR
       + RESET*       /REMRD* ST1*      CS*       /REMWR
       + RESET*IA13* REMRD*/ST1*/ST2*/CS*/EPAO* REMWR
/ST1   := RESET*       REMRD*/ST1* ST2*/CS
       + RESET*        REMRD* ST1*/ST2*/CS
       + RESET*       /REMRD* ST1*/ST2* CS*       /REMWR
       + RESET*       /REMRD*/ST1* ST2* CS*       /REMWR
       + RESET*/XACK*REMRD*      /ST2*/CS*       /REMWR
/ST2   := RESET*       REMRD*      ST2*/CS
       + RESET*/XACK*REMRD*/ST1*     /CS*       /REMWR
       + RESET*       /REMRD*      ST2* CS*       /REMWR
       + RESET*       /REMRD*/ST1*      CS* EPAO*/REMWR
       + RESET*        REMRD* ST1*/ST2* CS* EPAO* REMWR
/CS    := RESET*       REMRD*           /CS*       /REMWR
       + RESET*        REMRD* ST1*      /CS
       + RESET*        REMRD*           /CS* EPAO
       + RESET*        REMRD*      ST2*/CS
       + RESET*        REMRD* ST1* ST2*      EPAO* REMWR
       + RESET*/IA13*REMRD*           /CS
*/EPAO := RESET*        REMRD*      ST2*/CS*/EPAO
       + RESET*/XACK*REMRD*           /CS*/EPAO
       + RESET*        REMRD*           /CS*/EPAO* REMWR
       + RESET*        REMRD* ST1*      /CS*/EPAO
       + RESET*       /REMRD* ST1*       CS*/EPAO*/REMWR
       + RESET*       /REMRD*      ST2* CS*/EPAO*/REMWR
       + RESET*XACK* REMRD*/ST1*/ST2*/CS*EPAO*/REMWR
       + RESET*        REMRD* ST1* ST2* CS*EPAO* REMWR
/REMWR := RESET*       /REMRD*           ST2*/CS*       /REMWR
       + RESET*        REMRD* ST1*      /CS*       /REMWR
       + RESET*       /REMRD*           CS*/EPAO*/REMWR
       + RESET*       /REMRD*      ST2* CS*       /REMWR
       + RESET*        REMRD*/ST1*/ST2*/CS*          REMWR
       + RESET*       /REMRD* ST1*       CS*       /REMWR
       + RESET*/XACK*REMRD*           /CS*       /REMWR
/BRESET = /RESET +    /REMRD*/ST1*      CS*/EPAO*/REMWR
/LCLINV = LCL
```

FIGURE 7

Timing at Beginning of Instruction Load

RESET

BRESET

RAE

REMRD

REMWR

XACK

LCL

IWR

ICLK
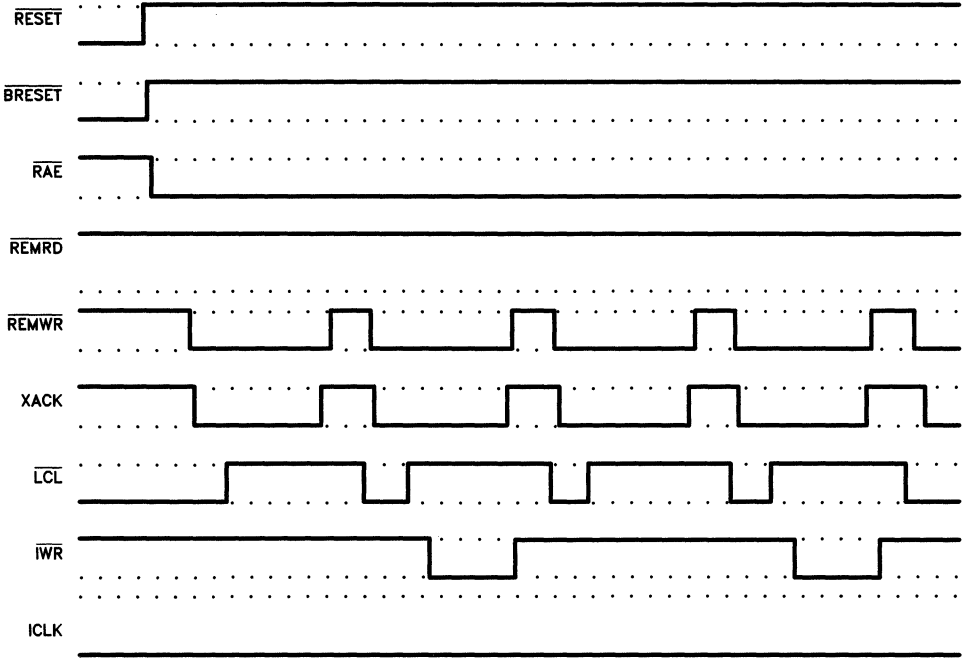
TL/F/9403–6

Timing at End of Instruction Load

RESET

BRESET

RAE

REMRD

REMWR

XACK

LCL

IWR

ICLK
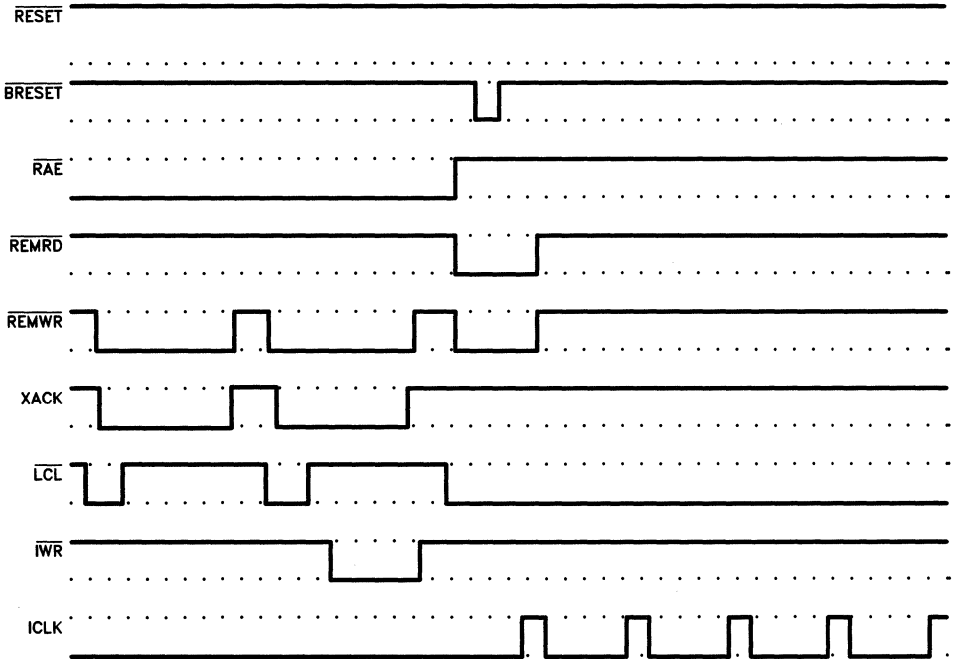
TL/F/9403–9

FIGURE 8. Example of Timing Waveforms
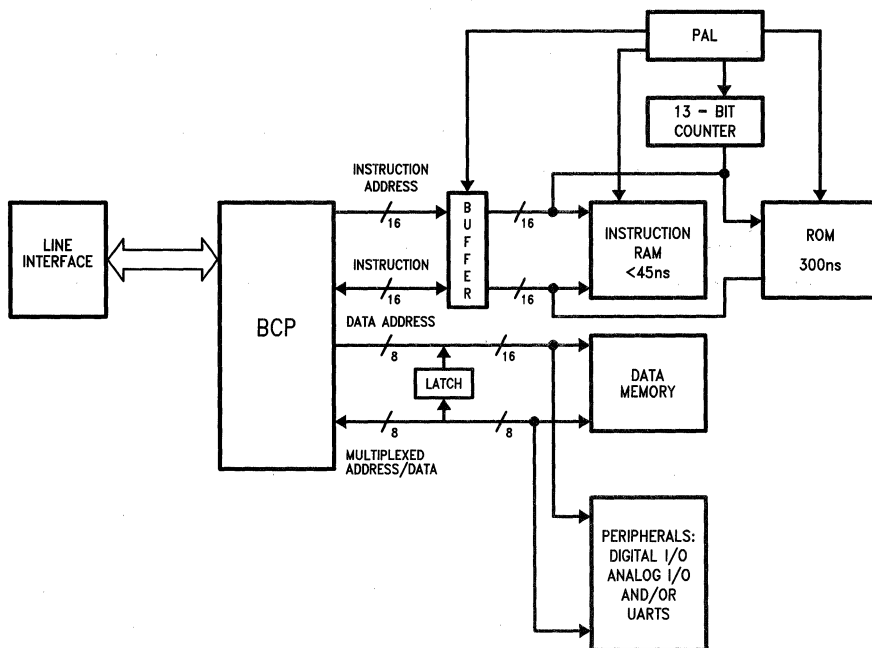
TL/F/9403-7

FIGURE 9. Another Method of Soft-Loading (A Non-Ideal Solution)

MODIFYING THE SOFT-LOAD SYSTEM FOR LARGER MEMORY

The soft-load system as documented loads 8K x 16 bits of instruction memory. Large programs may require more memory; smaller, lower cost systems may use less. The soft-load system can easily be altered to load larger or smaller instruction memory by changing one connection.

Connecting a different instruction address line to pin 4 of the PAL changes how much instruction memory is loaded: These connections are shown in *Figure 10.*

Instruction Memory Size:	Connect Pin 4 of PAL to:
32k x 16	IA15
16k x 16	IA14
8k x 16	IA13
4k x 16	IA12
2k x 16	IA11

FIGURE 10. Connections for Altering Instruction Memory Size

USING THE CAPSTONE CT-104 DEVELOPMENT BOARD TO EVALUATE THE SOFT-LOAD APPLICATION

A DP8344 biphase Communications Process development board is available from Capstone Technology Inc., of Fremont, California. The board is designed to reside in an IBM® PC. A breadboard area is provided on the board so that custom circuitry can be added. It can be converted into a stand-alone soft-load system by wire-wrapping three addi-

tional I.C.'s into the breadboard area. A diagram of the CT-104 board with the additional components is shown in *Figure 11.* Note that most of the prototyping area remains available, enabling the addition of other circuitry specific to the application being developed. A parts list is shown in *Figure 12.* The PAL16R6 is programmed with the equations shown in *Figure 7.* U22 and U23 must be removed from the CT-104 board and be replaced with specially wired 20-pin headers. The wiring on these headers, shown in *Figure 13,* provides access to the RESET~ signal and disables the unused interface circuitry on the board. Pin 11 of the header that replaces U23 must be wired to pin 13 of the 74LS14. A wiring list is shown in *Figure 14.* Power supply connections must be added because the board can no longer reside in the PC. Development of a stand-alone soft-load application can be done easily and quickly by using the CT-104 board because minimal circuit construction is required.

SUMMARY

The soft-load circuit uses the BCP's remote interface to load BCP code from slow EPROM to fast RAM, with a minimum of extra hardware. This method is useful in systems where there is no host processor directly interfaced to the BCP and the full processing speed of the BCP is needed.

The circuit can easily be modified to load different sizes of memory. The Capstone Technology, Inc. CT-104 development board can easily be converted to a stand-alone soft-load system for evaluation of the application.
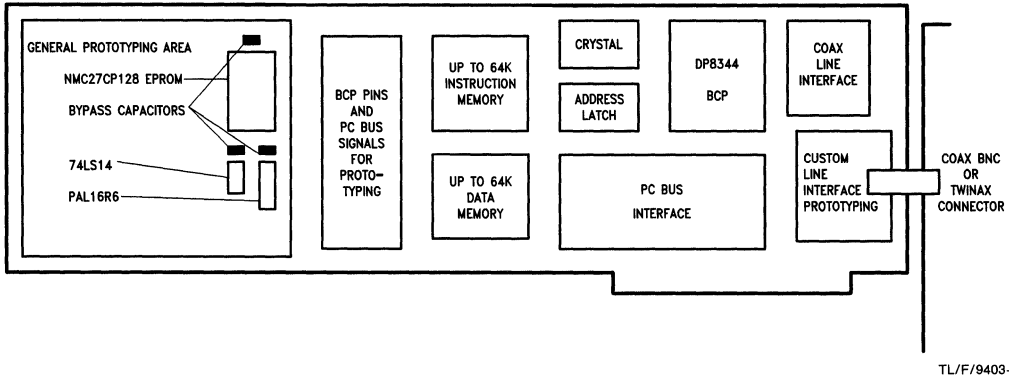
TL/F/9403-10

FIGURE 11. CT-104 Development Board with Soft-Load Circuitry

NMC27CP128 350 ns access time or faster
PAL16R6B
DM74LS14N
28-pin wire-wrap socket
20-pin wire-wrap socket
14-pin wire-wrap socket
3 Bypass capacitors, 0.1 μF
2 50-pin wire-wrap strips, 2 pins wide
2 20-pin headers

FIGURE 12. Parts List for Conversion of CT-104 Board
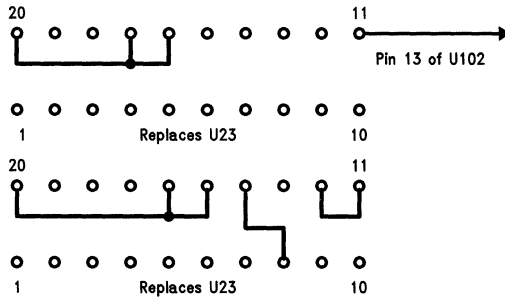
TL/F/9403-11

FIGURE 13. Header Wiring for Conversion of CT-104 Board

Pin	Unit	to	Pin	Unit		Pin	Unit	to	Pin	Unit
1	U100		—	VCC		28	U100		—	VCC
2	U100		12	W1		1	U101		17	W2
3	U100		7	W1		2	U101		11	W2
4	U100		6	W1		3	U101		7	W2
5	U100		5	W1		4	U101		14	W1
6	U100		4	W1		5	U101		10	U102
7	U100		3	W1		6	U101		—	GND
8	U100		2	W1		7	U101		—	GND
9	U100		1	W1		8	U101		—	GND
10	U100		14	U101		9	U101		50	W1
11	U100		33	W1		10	U101		—	GND
12	U100		34	W1		11	U101		49	W2
13	U100		35	W1		12	U101		8	W2
14	U100		—	GND		13	U101		48	W2
15	U100		36	W1		15	U101		46	W2
16	U100		37	W1		18	U101		47	W2
17	U100		38	W1		20	U101		—	VCC
18	U100		39	W1		1	U102		—	GND
19	U100		40	W1		3	U102		—	GND
20	U100		46	W2		5	U102		—	GND
21	U100		10	W1		7	U102		—	GND
22	U100		19	U101		9	U102		—	GND
23	U100		11	W1		11	U102		12	U102
24	U100		9	W1		13	U102		11	U23 HEADER
25	U100		8	W1		14	U102		—	VCC
26	U100		13	W1		45	W2		—	GND
27	U100		—	VCC						

FIGURE 14. Wiring List for Conversion of CT-104 Board

"Interrupts"—A Powerful Tool of the Biphase Communications Processor

When you have only 5.5 μs to respond you have to act fast. This is the amount of time specified in the IBM 3270 Product Attachment Information document as the maximum time allowed to respond to a message in a 3270 environment. This 5.5 μs is why the DP8344 interrupts are specifically tailored for the task of managing a communications line and feature very short latency times. This article contains information that will help the user to take better advantage of the extensive interrupt capability found in the DP8344.

The DP8344 has two external and four internal interrupt sources. The external interrupt sources are the Non-Maskable Interrupt pin, ($\overline{\text{NMI}}$), and the Bi-directional Interrupt Request pin ($\overline{\text{BIRQ}}$). A Non-Maskable Interrupt is detected by the CPU when $\overline{\text{NMI}}$ receives a falling edge. The falling edge is captured internally and the interrupt is processed when it is detected by the CPU as described later. $\overline{\text{BIRQ}}$ can function as both an interrupt into the DP8344 and as an output which can be used to interrupt other devices. When $\overline{\text{BIRQ}}$ is configured as an input an interrupt will occur if the pin is held low. Note that $\overline{\text{BIRQ}}$ is not edge sensitive and if the pin is taken back high before the interrupt is processed by the CPU then no interrupt will occur.

The internal interrupts consist of the Transmitter FIFO Empty (TFE) interrupt, the Line Turn Around (LTA) interrupt, the Time Out (TO) interrupt, and a user selectable receiver interrupt source.

The receiver interrupt source is selected from either the Receiver FIFO Full (RFF) interrupt, the Data Available (DA) interrupt, or the Receiver Active (RA) interrupt. The RFF interrupt occurs when the receive FIFO is full or if the receiver detects an error condition. This interrupt enables the user to handle packets of data as opposed to handling every data word individually. It also allows the program to spend additional time performing other tasks. However, since the RFF interrupt is only asserted when the receive FIFO is full, the LTA interrupt should be used in conjunction with RFF to allow the program to check the FIFO for additional words at the end of a message. The DA interrupt indicates valid data is present in the receive FIFO and also occurs if the receiver detects an error condition. It should be used when it is desirable to handle each data word individually. The DA interrupt also allows the program to utilize the time between receiving each data word for performing other tasks. The RA interrupt is asserted when the receiver detects a valid start sequence. It provides the user with an early indication of data coming into the receiver. This allows the program time to perform any necessary overhead activity before handling the receiver data. The RA interrupt is asserted approximately 90 transceiver clock cycles prior to data becoming available in the receive FIFO when using 3270 mode. Consequently, if the transceiver and CPU are operating at the same clock frequency, approximately 90 clock cycles (T-states) are available for interrupt latency and taking care of overhead prior to handling the received data.

A TFE interrupt occurs when the last word in the transmit FIFO is loaded into the encoder. This interrupt allows a program to continue working on another task while the transmitter is sending data. It is especially useful when sending a long message. When the transmit FIFO becomes empty the program is alerted by the TFE interrupt and may continue the message by loading additional words into the FIFO. This approach frees up a significant amount of processing time. For example, after the transmit FIFO is loaded it takes the transmitter approximately 264 transceiver clock cycles to send the starting sequence and two data words in 3270 mode. With the CPU operating at the transceiver clock frequency, the program has approximately 264 T-states available before the TFE interrupt will occur.

Once the TFE interrupt occurs the CPU has approximately 80 transceiver clock cycles to load the transmit FIFO in order to continue a multiframe message in 3270 mode. If the CPU is operating at the transceiver clock frequency, the program has approximately 80 T-states to accomplish the load operation. Since the load to the Receive/Transmit Register, {RTR}, only takes 2 T-states, 78 T-states are available for interrupt latency and processing overhead after the interrupt occurs.

The LTA interrupt provides an easy means for determining the end of a message. This allows a program to quickly begin transmitting after the end of a reception. The LTA interrupt indicates that the receiver detected a valid end sequence in 3270 mode of operation. In 5250 operating mode, the LTA interrupt occurs when the last fill bit has been received and no further input transitions are detected by the receiver. However, a LTA interrupt does not occur in 5250 or 8-bit non-promiscuous modes of operation unless an address match was decoded by the receiver.

The TO interrupt occurs when the CPU timer counts down to zero. The timer provides a flexible means for timing events. It is a sixteen bit counter which can be loaded by accessing CPU registers {TRH} and {TRL} and is controlled by the [TCS], [TLD] and [TST] bits in the Auxiliary Control Register, {ACR}.

After an interrupt occurs the event that generated it must be handled in order to clear the interrupt. The exception to this is $\overline{\text{NMI}}$. Since it is falling edge triggered, it is cleared internally when the CPU processes the interrupt. The actions necessary to clear the interrupts are listed in Table I.

In the case where $\overline{\text{BIRQ}}$ is asserted, the response will be dependent on the system design. Ordinarily, this response would involve some hardware handshaking such as reading or writing a specific data memory location. When internal interrupts become asserted there are specific actions which must be taken by a program to clear these interrupts. The RFF interrupt is cleared when the receive FIFO is no longer full and any errors detected by the receiver are cleared. Data is read from the receive FIFO by reading {RTR}. Reading the Error Code Register, {ECR}, clears any errors detected by the receiver. The DA interrupt is cleared when the receive FIFO is empty and any errors detected by the receiver are cleared. The RA interrupt is cleared by reading {RTR} or {ECR}. All three receiver interrupts are cleared when the transceiver is reset. In many cases, resetting the transceiver is the preferable response to an error detected

2

TABLE I. Clearing Interrupts

Interrupt	How to Clear Interrupt
$\overline{\text{NMI}}$	Internally Cleared When Recognized by the CPU.
RFF	Read {RTR} When Receive FIFO is Full. Read {ECR} When an Error Occurs. Read {ECR} and {RTR} When an Error Occurs and Receive FIFO is Full. Reset the Transceiver. Reset the DP8344.
DA	Read {RTR} When Receive FIFO is Not Empty. Read {ECR} When an Error Occurs. Read {ECR} and {RTR} When an Error Occurs and Receive FIFO is Not Empty. Reset the Transceiver. Reset the DP8344.
RA	Read {RTR} or {ECR}. Reset the Transceiver. Reset the DP8344.
TFE	Write to {RTR}.
LTA	Write to {RTR}. Reset the Transceiver. Reset the DP8344. Write a One to {NCF} Bit 4.
$\overline{\text{BIRQ}}$	System Dependent.
TO	Write a One to {CCR} Bit 7. Stop the Timer. Reset the DP8344.

by the receiver. The TFE interrupt is cleared by writing to {RTR}. Unlike the receiver interrupts, the TFE interrupt is asserted when the transceiver is reset. The LTA interrupt is also cleared by writing to {RTR} or resetting the transceiver. In addition, it may be cleared by writing a one to bit 4 of the Network Command Flags register, {NCF}. The last internal interrupt is TO. It is cleared by writing a one to bit 7 in the Condition-Code Register, {CCR} or by stopping the timer. Note that the timer reloads itself and continues to count after the interrupt has been generated regardless of whether a one is written to bit 7 in {CCR}.

With the exception of $\overline{\text{NMI}}$, all of the interrupts are disabled when the DP8344 is reset. In order to make use of the interrupts they must be enabled in software. Software enabling and disabling of the interrupts is performed by changing the state of the Global Interrupt Enable, [GIE], bit in {ACR} and the state of the individual interrupt mask bits in the Interrupt Control Register, {ICR}.

[GIE] is a read/write register bit and so may be changed by using any instruction that can write to {ACR}. In addition, the RET, RETF, and EXX instructions have option fields which can be used to alter the state of [GIE]. RET and RETF are the return instructions in the DP8344 and EXX is used to exchange register banks. The EXX instruction can set or clear [GIE] as well as leaving it unchanged. The RET and RETF instructions can restore [GIE] to the value that

was saved on the address stack at the time the interrupt was recognized. They also provide the options of clearing or setting [GIE] or leaving it unchanged. [GIE] is cleared when an interrupt is recognized by the CPU in order to prevent other interrupts from occurring during an interrupt service routine. The [GIE] options described above facilitate enabling and disabling interrupts when returning from an interrupt service routine. The restore option is especially useful with the $\overline{\text{NMI}}$. Since a Non-Maskable Interrupt can occur whether [GIE] is set or cleared, the restore [GIE] option can be used in the return instruction to put [GIE] back to its state prior to the interrupt occurring.

As the name implies, [GIE] affects all the maskable interrupts. However, in order to use any of these interrupts they must be unmasked by changing the state of their associated mask bit in {ICR}. When set high, bits [IM0], [IM1], [IM2], [IM3], and [IM4] in {ICR} mask the receiver interrupt, TFE interrupt, LTA interrupt, $\overline{\text{BIRQ}}$ interrupt, and TO interrupt respectively. To enable an interrupt, its mask bit must be set low. The interrupts and associated mask bits are shown in Table II. These bits are set high when the DP8344 is reset. Bits [RIS1] and [RIS0] in {ICR} are used to select the source of the receiver interrupt as shown in Table III. Note that only one of these interrupts can be active as the source of the receiver interrupt.

TABLE II. {ICR} Interrupt Mask Bits and Interrupt Priority

Interrupt	Mask Bit	Priority
NMI	—	Highest
RFF, DA, RA	IM0	
TFE	IM1	
LTA	IM2	
BIRQ	IM3	
TO	IM4	Lowest

TABLE III. {ICR} Receiver Interrupt Select Bits

RIS1	RIS0	Receiver Interrupt Source
0	0	RFF
0	1	DA
1	0	Reserved
1	1	RA

As stated earlier, [GIE] is cleared when an interrupt is recognized by the CPU. This prevents other interrupts from occurring in the interrupt service routine. In cases where it is desirable to allow nesting of interrupts, [GIE] should be set high within the interrupt routine. An example of nesting interrupts is using the RA interrupt in the main program and switching to the RFF or DA interrupt in the RA interrupt routine. Note that the internal address stack is twelve words deep and there is no recovery from a stack overflow. Therefore, care should be taken when nesting interrupts.

When more than one interrupt is unmasked and asserted, the CPU processes the interrupt with the highest priority first. NMI has the highest priority followed by the receiver interrupt, TFE, LTA, BIRQ, and TO. Therefore, if DA and BIRQ were both active, DA would be processed first followed by BIRQ. However, if a higher priority interrupt occurred while the DA interrupt was being handled then it would be processed before BIRQ. Each time the interrupts are sampled, the highest priority interrupt is processed first, regardless of how long a lower priority interrupt has been active. Interrupt priority is summarized in Table II.

A call to the interrupt address is generated when an interrupt is detected by the CPU. The address for each interrupt is constructed by concatenating the Interrupt Base Register, {IBR}, contents with the individual interrupt code as shown in Table IV. There is room between the interrupt addresses for a maximum of four instruction words. Normally, at each interrupt address there would be a jump instruction to an

interrupt service routine. The return instruction at the end of the interrupt service routine would then return to the address at which the interrupt occurred. By changing {IBR} it is possible to locate the interrupt jump table in memory wherever it is convenient or for one program to use more than one interrupt jump table.

TABLE IV. Interrupt Vector Generation

Interrupt	Code
NMI	111
RFF, DA, RA	001
TFE	010
LTA	011
BIRQ	100
TO	101

Interrupt Vector

{IBR} Contents	0 0 0	Code	0 0
15	8	4 2	0

As mentioned previously, the interrupts are sampled in the CPU prior to the start of each instruction. To be precise, they are sampled by each falling edge of the CPU clock with the last falling edge prior to the start of the next instruction determining whether an interrupt will be processed. The timing of a typical interrupt event is shown in *Figure 1*. The interrupt occurs during the current instruction and is sampled by the falling edge of the CPU clock. The next instruction is not operated on and its address is stored in the internal address stack. In addition, the current state of [GIE] and the states of the ALU flags and bank positions are stored in the internal address stack. A 2 T-state call is now executed in place of the non-executed instruction. This call will cause a branch to the interrupt address that is generated in the first half of T-state T1. [GIE] is then cleared during the first half of T-state T2. From this description it is evident that the shortest interrupt latency is 2.5 T-states. This assumes that an interrupt occurs during the first half of T2 and is sampled by the next falling edge of the CPU clock. However, a number of factors can increase the interrupt latency. If the interrupt misses the setup time to the falling edge of the last CPU clock the response time will increase by a minimum of 2 T-states. This increase is caused by the execution of one additional instruction. Of course, if the additional instruction takes more than 2 T-states to execute the interrupt latency will be greater.

minimum interrupt latency
(~125 ns @ 20 MHz)

T1   T2   T1   T2   T1   T2

CPU CLK

Interrupt
sampled

Interrupt

Instruction
Address Bus | Non-executed Instruction Address | Interrupt Vector Address

Instruction
Bus | Non-executed instruction | First interrupt instruction

[GIE]
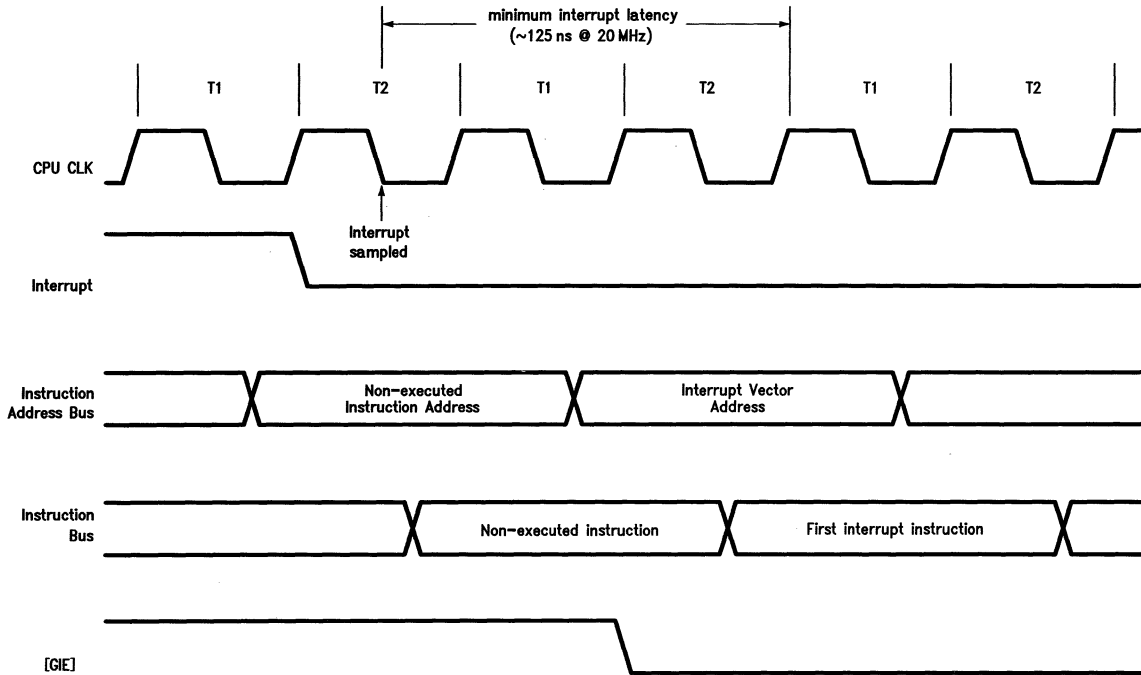
TL/F/9361–1

FIGURE 1. Minimum Interrupt Timing

Running the DP8344 with wait states will also increase interrupt latency. Instruction memory wait states increase latency by increasing the length of each instruction, including the call to the interrupt service routine. Data memory wait states will increase interrupt latency if an interrupt must wait for an instruction which accesses data memory to execute before it can be processed. A less obvious factor that can increase interrupt latency is data memory accesses by the remote system. If the DP8344 is attempting a data memory access and the remote system already has control of the data memory bus, the CPU will be waited. If an interrupt occurs at this time it will not be processed until the DP8344 is able to complete the instruction which is accessing data memory. This implies that a system with a lot of data memory arbitration occurring between the DP8344 and the remote system may have a longer average interrupt latency. The worst case interrupt latency will occur when the external $\overline{\text{LOCK}}$ or $\overline{\text{WAIT}}$ pins are asserted. Clearly, if the CPU is stopped by the assertion of the $\overline{\text{WAIT}}$ pin any interrupts occurring will not be processed until the CPU is released from the wait state. Asserting the $\overline{\text{LOCK}}$ pin would have the same affect if the DP8344 attempts to make a data memory access. Note that interrupts are not disabled or cleared when the CPU is stopped by the remote system deasserting [STRT] in the Remote Interface Configuration, {RIC}, register. When the CPU is restarted any asserted interrupts will be processed. From the above discussion it is evident that calculating the interrupt latency is not trivial and will be dependent on the program and the system.

The interrupts on the DP8344 are powerful tools for controlling events in a time critical environment. They are one of the many reasons why the DP8344 Bi-phase Communications Processor provides a superior solution to managing communications interfaces.

2

JRMK Speeds Command Decoding

The Biphase Communications Processor (BCP) has several features that make it ideal to use in a high speed communications environment. The relative Jump with Rotate and MasK on register command, JRMK, is designed to allow quick and efficient decoding of register fields. Fast decoding of command, data, and address fields allows the BCP to spend most of an interrupt handler's code and time on the protocol's actual instruction execution, instead of on decoding it. This helps meet the stringent 5.5 μs turn around times demanded in 3270 communications.

JRMK rotates and masks a copy of its source register to form a signed program counter offset which is often used to point to a jump table. The JRMK instruction first makes a copy of the source register. All actions will be performed on this copy, not on the original. The register then is rotated to the right zero to seven places. Next, JRMK masks (zeros out) the LSB in addition to as many bits as the mask field indicates, starting at the MSB. Finally, JRMK adds this result to the Program Counter (PC), providing a relative range of +128, −126 instruction words. In practice, relative jumps (JMP) and long jumps (LJMP) are usually placed in the table, but there are no restrictions on which instructions may fit in. Each entry has a minimum space of two instruction words allowing LJMP's to fit. *Figure 1* demonstrates the BCP's internal execution of a JRMK instruction.

Example Code

JRMK   RTR,3,3   ;decode feature address

Instruction Execution

(a) Copy {RTR} into JRMK's displacement register

(b) Rotate displacement register 3 bits right

(c) AND result with "00011110"

(d) Sign extend resulting displacement and add it to the program counter, (PC). If the bits F4–F1 equal "0001" then +2 is added to the PC.

JRMK Displacement Register Contents

	F4	F3	F2	F1	x	x	x	x
(a)	F4	F3	F2	F1	x	x	x	x
(b)	x	x	x	F4	F3	F2	F1	x
(c)	0	0	0	F4	F3	F2	F1	0

FIGURE 1. JRMK Instruction Example

The JRMK instruction contains four (4) fields that control its operation—a source register field, a rotate field, a mask field, and the opcode itself. The source register may be any register in the BCP that is always available or is currently bank switched in. The source register is not modified by the operation of the JRMK instruction. Even in the case of the {RTR} register, the receiver FIFO is not changed and the same byte remains at the top of the FIFO after executing JRMK. The rotation field directs the BCP to rotate the source register to the right by 0–7 bits. The mask field indicates how many bits to mask from the source register starting at the MSB after the rotation is complete. Up to 7 bits may be masked off in addition to the LSB. If the mask field equals zero (0), only the LSB will be masked. If the mask field equals one (1), the MSB will be masked as well as the LSB. Similarly, if the mask field equals two (2), bits 7,6 and the LSB will be masked. *Figure 2* shows the construction of the JRMK instruction opcode.

Opcode

1	0	0	0	0	m	m	m	b	b	b		Rs		

m—Mask Field
b—Bit Places to Rotate
Rs—Source Register

FIGURE 2. JRMK Opcode Construction

JRMK can be set up to provide more than two instruction words per table entry, if the source register data format is known. If the rotation causes a zero bit to always appear in bit 1 of the rotated register, then each table entry will have four instruction words.

The JRMK instruction executes in 4 T-states if there are no instruction wait states. If the BCP's CPU clock is running at a speed of 20 MHz, a T-state is 50 ns in duration. In this case, each JRMK instruction will complete in 200 ns.

AN EXAMPLE

A good example of how to use the JRMK instruction is found in the Multi-Protocol Adaptor (MPA). The MPA is a design/evaluation kit available from National Semiconductor. It provides complete link level source code, hardware, and development notes for creating a 3270 or 5250 PC terminal emulator card.

This example comes from actual MPA code in the Data Available interrupt handler for 3270 terminal emulation. All overhead such as bank switching, register saving, and index register setting have been previously executed, and the 3270 command is at the top of the receiver FIFO. The actual implementation of executing each 3270 instruction, as well as the decode tables for devices other than the base, is not shown. Additionally, the code for handling data is not presented. These are all included with the MPA source code.

When a 3270 message is available in the receiver FIFO, a determination is made whether that message is a command or data at the *rxcx_fast* label as shown in *Figure 3*. If the receiver contains data, the BCP vectors to a location held in the index register equated to DATA_VECTOR. If the message is a command, the BCP will jump to the label *cx_comm* to check for common commands. The Network Control Flag (NCF) register contains bits for hardware decoded commands, POLL, POLL/ACK, and TT/AR. POLL and POLL/ACK will jump to their respective command handlers. Since a TT/AR shold not be received by a terminal, its decode will jump to the *cx_perr* error handler. A no-operation, NOOP, is inserted after the first jump because the JRMK instruction is set in this case to jump to every other address. The NOOP takes up an instruction location to ensure that the table conforms to this specification. A NOOP is a macro that stands for MOVE ACC,ACC. If the command is not one of these three, then the address of the command must be checked.

At the label *addr__dec*, the BCP will vector to different command handlers based on the feature address of the received command. All unimplemented features jump to the *cx__dec__err* error handler. The JRMK instruction is used to look at bits 4–7 of {RTR} which point to the 3270 feature that the command is for. Based on these bits, the different feature command decoders will be jumped to as shown in *Figure 4*.

```
;
; setup code here
;
        .
        .
        .
rxcx_fast:
        ljmp            TSR,1,S,cx_comm         ; command or data?
                                                ; jump if command
        ljmp            [DATA_VECTOR]           ; data, jump to appropriate
                                                ; handler


;
; check for quick command decodes
;
cx_comm:
        jrmk            NCF,7,4                 ; jump on immediate decode prior to
                                                ; advancing FIFO
cx_immed:
        jmp             addr_dec                ; not an immediate decode command
        NOOP
        ljmp            cx_poll                 ; poll command decoded
        ljmp            cx_pack                 ; pack
        ljmp            cx_perr                 ; should not get here (TT/AR)
```
FIGURE 3. JRMK Fast Command Determination

```
;
; find out which feature that the command is addressed to
;
addr_dec:
        jrmk            RTR,3,3         ; jump based on 4 bit address field

; address parse table

cx_addr:
        jmp             base_dec        ; 0 decode base/keyboard command
        NOOP
        jmp             base_dec        ; 1 decode base/keyboard
        NOOP
        ljmp            cx_dec_err      ; 2 light pen
        ljmp            cx_dec_err      ; 3 reserved
        ljmp            cx_dec_err      ; 4 magnetic stripe reader
        ljmp            cx_dec_err      ; 5 PC adapter
        ljmp            cx_dec_err      ; 6 3180 advanced
        ljmp            eab_dec         ; 7 EAB
        ljmp            cx_dec_err      ; 8 reserved
        ljmp            cx_dec_err      ; 9 reserved
        ljmp            cx_dec_err      ; A reserved
        ljmp            cx_dec_err      ; B convergence
        ljmp            cx_dec_err      ; C reserved
        ljmp            cx_dec_err      ; D reserved
        ljmp            cx_dec_err      ; E reserved
        ljmp            cx_dec_err      ; F reserved
```
FIGURE 4. JRMK Feature Determination

At the base feature decoder *base_dec*, the actual com-mand is decoded and jumps are taken to the different ad-dresses to handle each one. *Figure 5* details this operation.

```
;
; base command parse table
;
base_dec:
        jrmk            RTR,7,2             ; decode base command
cx_base:
        ljmp            cx_ignore           ; 00 should not get here
        ljmp            cx_poll             ; 01 poll command
        ljmp            cx_reset            ; 02 reset device
        ljmp            cx_readata          ; 03 read data
        ljmp            cx_lach             ; 04 load address counter high
        ljmp            cx_rach             ; 05 read address counter high
        ljmp            cx_clear            ; 06 clear
        ljmp            cx_rdex             ; 07 read extended terminal ID
        ljmp            cx_start            ; 08 start operation
        ljmp            cx_rdid             ; 09 read terminal ID
        ljmp            cx_lcont            ; 0A load control register
        ljmp            cx_rdmul            ; 0B read multiple
        ljmp            cx_write            ; 0C write data
        ljmp            cx_rdstat           ; 0D read status
        ljmp            cx_insert           ; 0E insert byte
        ljmp            cx_ignore           ; 0F reserved
        ljmp            cx_sforward         ; 10 search forward
        ljmp            cx_pack             ; 11 poll with acknowledge set
        ljmp            cx_sback            ; 12 search backward
        ljmp            cx_ignore           ; 13 reserved
        ljmp            cx_lacl             ; 14 load address counter low
        ljmp            cx_racl             ; 15 read address counter low
        ljmp            cx_mask             ; 16 load mask
        ljmp            cx_ignore           ; 17 reserved
        ljmp            cx_ignore           ; 18 reserved
        ljmp            cx_ignore           ; 19 reserved
        ljmp            cx_lscont           ; 1A load secondary control
        ljmp            cx_ignore           ; 1B reserved
        ljmp            cx_diagreset        ; 1C diagnostic reset
        ljmp            cx_ignore           ; 1D reserved
        ljmp            cx_ignore           ; 1E reserved
        ljmp            cx_ignore           ; 1F reserved
```

FIGURE 5. JRMK Decoding of 3270 Instructions

If our command was a Load Control Register command (00001010), the JRMK instruction at label *cx__comm* would send us to a jump to *addr__dec* to decode which feature the command is directed to. At that label, JRMK would send us to the jump to *base__dec* since our address is "0000". Since the command is "01010", the JRMK relative jump will move to the instruction *ljmp cx__lcont* which jumps to the appropriate code to handle that instruction.

From *rxcx__fast* to the proper command to the base feature, there are 24 T-states of time used. At 20 MHz with no wait states, this translates to 1.2 μs. With a maximum interrupt latency of 225 ns, this leaves at least 4.075 μs to handle all other aspects of each command to the base. Commands to other features will probably take 1 T-state longer for the long jump to the command decode table (also using JRMK) for that feature, whereas the base feature used a relative jump.

The JRMK instruction is one example of how the BCP is optimized for high speed communications.

2

DP8344 Remote Processor Interfacing

This application note is provided to help the reader understand the information give in Table 24: Remote Rest Time of the DP8344AV 4.1 datasheet.*

For the BCP to operate properly, remote accesses to the BCP must be separated by a minimum amount of time. This minimum amount of time has been termed 'rest time'.

To give the reader a better understanding of rest time, the following items will be discussed in this application note:

1. The causes of remote rest time.

2. The way to interpret Table 24 and the worst case rest time.

3. The desirable features of a rest time circuit.

4. A design example of a rest time circuit for the CT-104 board.

Before proceeding any further, it must be stated that the design of DP8344AV did not introduce remote rest time. Remote rest time exists on all versions of the BCP. New tests have recently provided the remote rest time specification. Now we are releasing these specifications to assist our customers in their designs.

*All specifications used in this application note are from the DP8344AV 4.1 datasheet. Please refer to the latest datasheet available for the most current specifications.

CAUSES OF REMOTE REST TIME

There are two causes for remote rest time. The first cause is implied in the state diagrams for remote accesses and can be explained as follows:

At the beginning of every T-state the validity of a remote access is sampled for that T-state. To guarantee that the BCP recognizes the end of a remote cycle, the time between remote accesses must be a minimum of one T-state plus setup and hold times. This worst case rest time for the DP8344AV is:

$$\text{rest time} = 1T + t \text{ (setup time)} + t \text{ (hold time)}$$
$$= 1T + 23 \text{ ns} + 10 \text{ ns}$$
$$= 1T + 33 \text{ ns}$$

In the case of Latched Read and Fast Buffered Write, the validity of a remote access is not sampled on the first rising edge of the CPU-CLK following XACK rising. However, on all subsequent rising edges of the CPU-CLK, the validity of the remote access is sampled. As a result, if the remote processor can terminate its remote access quickly after XACK rises (within a T-state), up to a T-state may be added to the above equation for Latched Read and Fast Buffered Write modes. On the other hand, if the remote processor does not terminate its remote access within a T-state of XACK rising, the above equation remains valid for Latched Read and Fast Buffered Write modes.

If this specification is not adhered to, the BCP may sample the very end of one valid remote access and one T-state later sample the very beginning of a second valid remote access. Thus, the BCP will treat the second access as a continuation of the first remote access and will not perform the second read/write. The second access will be ignored. (Reference *Figure 1* for timing diagrams which demonstrate how two remote accesses can be mistaken as one.)

The second source of remote rest time is due to the manner in which the BCP samples the CMD signal. (Please note that when CMD is high all remote accesses are to the Remote Interface Control register {RIC}. When CMD is low all remote accesses are to where RIC's Memory Select Bits point.) CMD is sampled once at the beginning of each remote access. Due to the manner in which CMD is sampled, CMD will not be sampled again if a second remote access begins within 1.5(T-states) plus a hold time, after the BCP recognizes the end of the first remote access. If this happens, the BCP will use the value of CMD from the previous remote access during the second remote access. If the value of CMD is the same for both accesses, the second access will proceed as intended. However, if the value of CMD is different for the two remote accesses, the second remote access would read/write the wrong location.

The reader should note that the timing of the second source of rest time begins at the same time that the BCP first samples the end of the previous remote access. Thus, when the first source of rest time ends, the second source of rest time begins. (Reference *Figure 2* for timing diagrams for rest time in all modes except latched write.)

LATCHED WRITE MODE

Latched write mode is a special case of rest time and needs to be discussed separately from the other modes. The first cause of rest time affects every mode including latched write. In regards to the second source of rest time, latched write mode was designed to allow a second remote access to start while a write is still pending (i.e., $\overline{\text{WR-PEND}} = 0$). Thus, when $\overline{\text{WR-PEND}}$ rises (signaling the end of the previous write) the value of CMD is sampled for the seccond remote access. This will result in sampling the correct value of CMD for the second access. This allows latched write to avoid the second cause of rest time mentioned above.

However, if a remote access begins within half a T-state after $\overline{\text{WR-PEND}}$ rises, CMD will not be sampled again. For this case, if the value of CMD changed just after $\overline{\text{WR-PEND}}$ rose and at the same time the remote access began, the BCP would read/write the wrong location. At this time there is no specification for this rest time. Nonetheless, for a very conservative rule of thumb, if a remote access cannot setup for the rising edge of $\overline{\text{WR-PEND}}$, then the access needs to be delayed for one T-state after $\overline{\text{WR-PEND}}$ rises. (Reference *Figure 3* for timing diagrams of rest time for latched write mode.)

HOW TO INTERPRET TABLE 24 AND WORST CASE REST TIMES

At this time it is desirable to review how to interpret Table 24 and to review what the actual worst case rest time is. To interpret the specifications in Table 24, the reader must understand the differences between running the BCP at full speed (i.e., [CCS] = 0) and half speed (i.e., [CCS] = 1). At full speed both the CPU-CLK and CLK-OUT operate at the same frequency as OCLK. When the BCP runs at half speed, CLK-OUT remains at the same frequency as OCLK, but the CPU-CLK operates at half the frequency of OCLK. In the data sheet, one T-state is defined as one CPU-CLK cy-
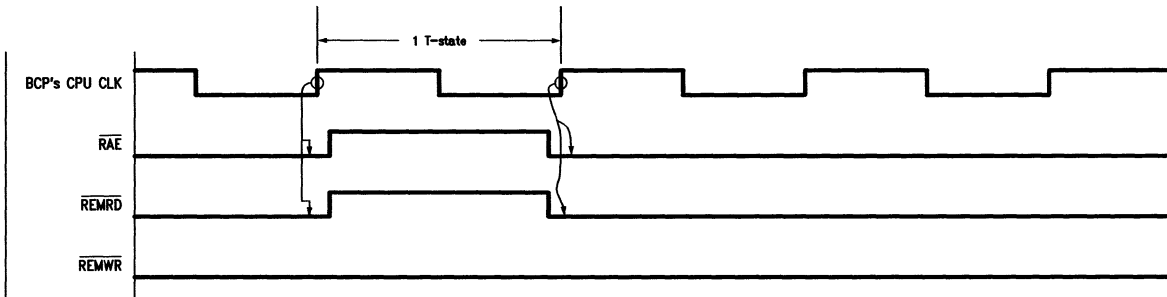
TL/F/10451–1

(a) This timing diagram shows two remote accesses within one T-state. The first set of arrows shows the BCP sampling a valid remote read. The next time the BCP samples the validity of the remote access is shown by the second set of arrows (1 T-state later). In this case, it will sample the second remote access and mistake it as a continuation of the first remote access.
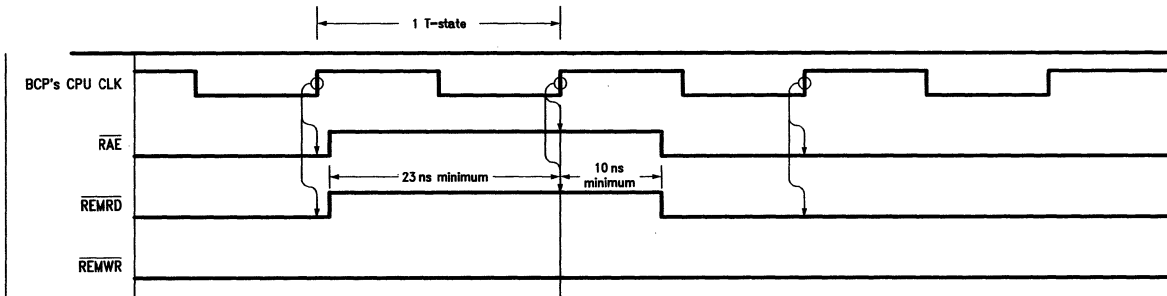
TL/F/10451–2

(b) This timing diagram shows the timing necessary for the BCP to recognize both accesses as separate accesses. The first set of arrows shows the BCP sampling a valid remote read. One T-state later at the second set of arrows, the BCP will sample the end of the first remote access. Another T-state later at the third set of arrows, the BCP will sample the beginning of the second remote access.

FIGURE 1. Mistaking Two Remote Accesses as Only One

TL/F/10451-3

(a) This timing diagram shows the second remote access violating rest time. The first set of arrows shows the BCP sampling a valid remote write. The second set of arrows (1 T-state later), shows the BCP sampling the end of the first remote access. If a second remote access starts before the position of the third set of arrows (another 1.5 T-states later), the value of CMD will not be sampled. The value of CMD has changed from the first remote access, so the BCP will write to the wrong location during the second access.

TL/F/10451-4
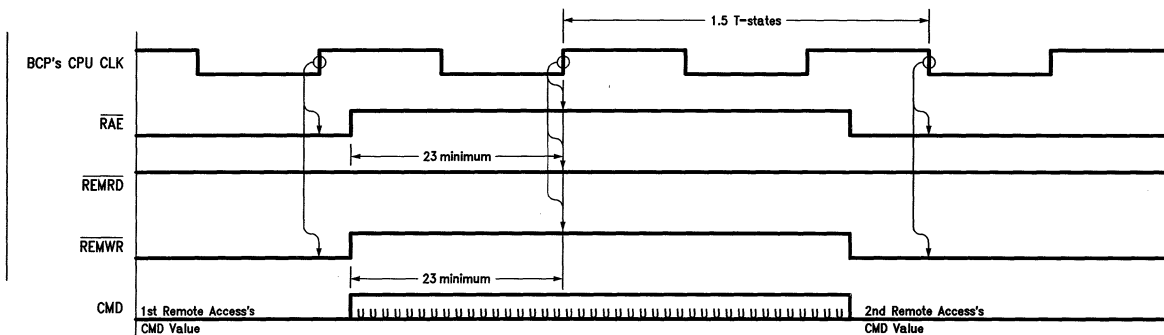
(b) This timing diagram shows the second remote access violating rest time. The first set of arrows shows the BCP sampling a valid remote write. The second set of arrows (1 T-state later), shows the BCP sampling the end of the first remote access. If a second remote access starts before the position of the third set of arrows (another 1.5 T-states later), the value of CMD will not be sampled. The value of CMD does not change from the first remote access, so the BCP will write to the intended location during the second remote access.

FIGURE 2. Remote Rest Time for All Modes except Latched Write

TL/F/10451–5

(c) This timing diagram shows the timing needed to avoid rest time for all modes except latched write. The first set of arrows shows the BCP sampling the end of the first remote access. The second set of arrows (1.5 T-states later), shows the BCP recognizing no remote access has started and the value of CMD will be sampled for the next remote access. The third set of arrows shows the BCP sampling the correct value of CMD for the second remote access.

FIGURE 2. Remote Rest Time for All Modes except Latched Write (Continued)
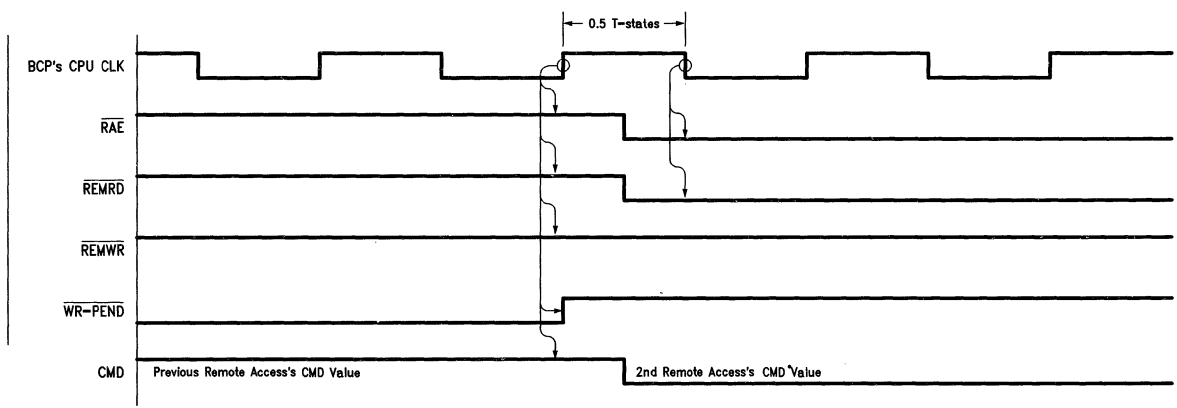
TL/F/10451–6

(a) This timing diagram shows a remote access violating remote rest time. The first set of arrows shows the BCP sampling the value of CMD when WR-PEND rises. If a remote access begins after WR-PEND rises and before the position of the second set of arrows (0.5 T-states later), the value of CMD will not be sampled again. The value of CMD has changed since WR-PEND rose, so the BCP will read the wrong location.

FIGURE 3. Rest Time for Latched Write Mode

2
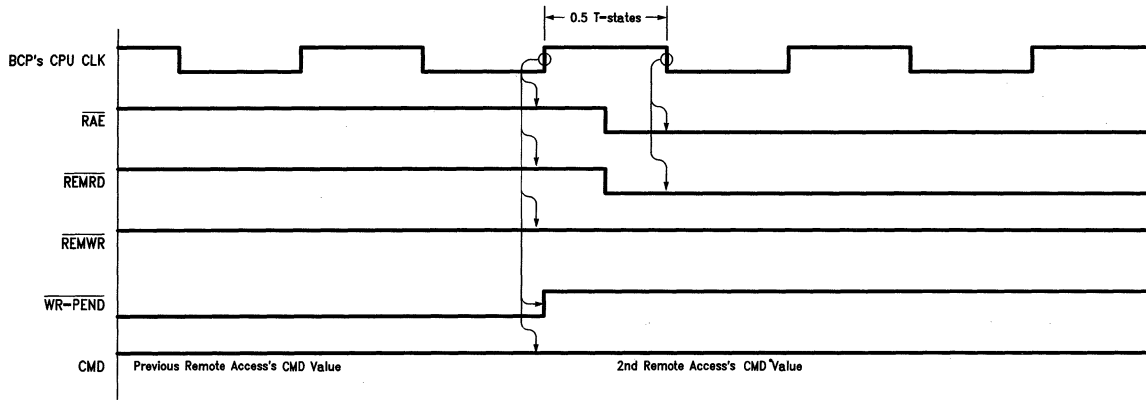
TL/F/10451–7

(b) This timing diagram shows a remote access violating remote rest time. The first set of arrows shows the BCP sampling the value of CMD when $\overline{\text{WR-PEND}}$ rises. If a remote access begins after $\overline{\text{WR-PEND}}$ rises and before the position of the second set of arrows (0.5 T-states later), the value of CMD will not be sampled again. The value of CMD has not changed since $\overline{\text{WR-PEND}}$ rose, so the BCP will read the intended location.
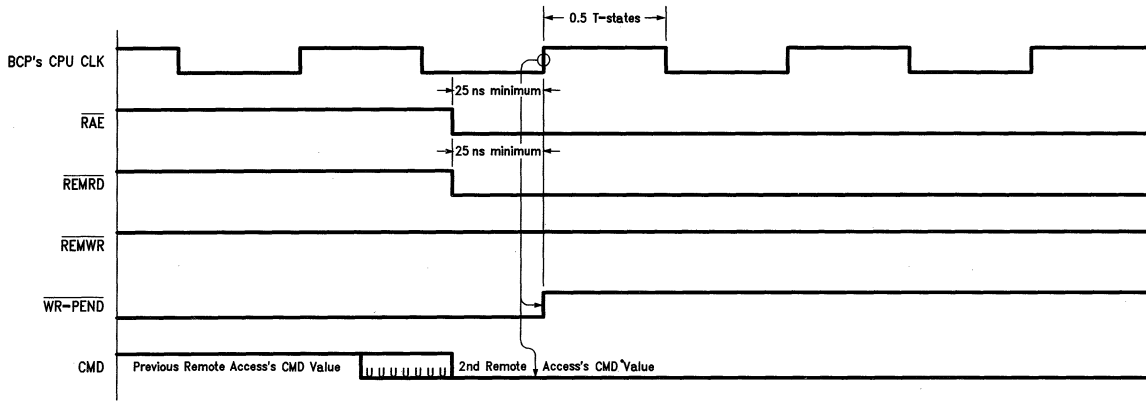
TL/F/10451–8

(c) This timing diagram shows a remote access setting up in time for $\overline{\text{WR-PEND}}$ rising to latch in the proper value of CMD. The only set of arrows shows the BCP sampling the second remote access's CMD value when $\overline{\text{WR-PEND}}$ rises. The value of CMD will not be sampled again. The BCP will carry out the second remote access as it was intended.

FIGURE 3. Rest Time for Latched Write Mode (Continued)
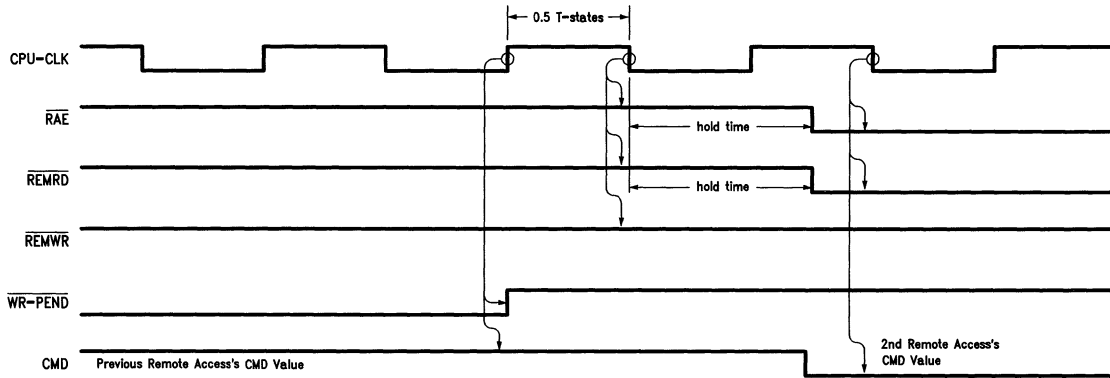
TL/F/10451-13

(d) This timing diagram shows a remote access starting after a half T-state plus a hold time since $\overline{\text{WR-PEND}}$ rose. The first set of arrows shows the BCP sampling the value of CMD when $\overline{\text{WR-PEND}}$ rises. The second set of arrows shows the BCP recognizing that no remote access has started and the value of CMD will be sampled for the next remote access. The third set of arrows shows the BCP sampling the correct value of CMD for the second remote access. The BCP will carry out the second remote access as it was intended.

FIGURE 3. Rest Time for Latched Write Mode (Continued)
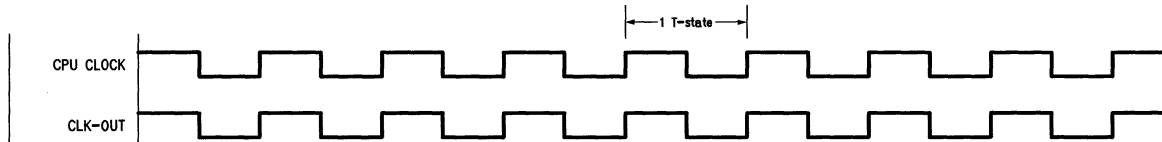
TL/F/10451-9

(a) BCP Running at Full Speed
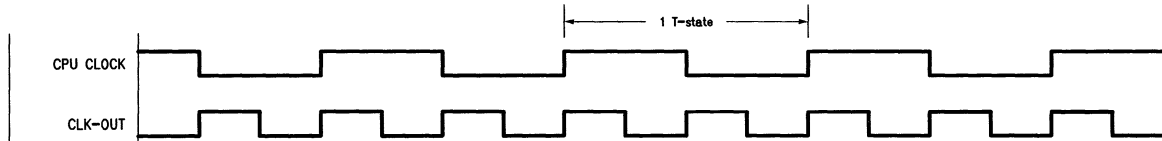
TL/F/10451-10

(b) BCP Running at Half Speed

FIGURE 4. Relationship between the BCP's CPU-Clock and CLK-OUT

cle. As a result, at full speed one T-state equals one CLK-OUT cycle, but at half speed one T-state equals two CLK-OUT cycles. (Reference *Figure 4* to see the relationship between the BCP's CPU-CLK and CLK-OUT at full speed and half speed.) The specifications in Table 24 are all measured with the BCP running at full speed. All of the rest time specifications are dependent on the CPU-CLK and not on CLK-OUT. At full speed, the CPU-CLK and CLK-OUT are the same, and this fact allows specifications to CLK-OUT in place of the CPU-CLK. On the other hand, at half speed the specifications to CLK-OUT are no longer valid because one cannot tell if a rising edge of CLK-OUT is a rising or falling edge of the CPU-CLK.

Earlier the worst case rest time for the BCP mistaking two fast back to back accesses as only one was given as:

rest time = 1T + t (setup time) + t (hold time)
(mistaking two accesses as one)

The real time worst case for the BCP mistaking two accesses as one, happens when the BCP runs at half speed. So for the BCP running at half speed and OCLK = 18.8696 MHz, the worst case rest time for mistaking two accesses as one is:

rest time = 2(CLK-OUT cycles) + t_{SU} + t_h
(mistaking two accesses as one)

rest time = 2(53 ns) + 23 ns + 10 ns
(mistaking two accesses as one)

rest time = 136 ns
(mistaking two accesses as one)

Up to a full T-state (or two CLK-OUT cycles) may be added to the above equation if one is using Latched Read or Fast Buffered Write modes. As explained in the CAUSES of Remote Rest Time section, this extra T-state is only added if the remote processor can terminate the remote access quickly after XACK rises (within a T-state). Otherwise, the above equation remains valid as written. The reader should note that this extra T-state is not mentioned or included in the following calculations because it takes place coincidentally with that cause of rest time.

As mentioned previously, the absolute worst case rest time for all modes, except latched write mode, may be calculated by adding the above case of rest time to the second source of rest time caused by fast back to back accesses with different values for CMD. This rest time can be calculated as follows:

rest time = first source + second source
(CMD changes)

rest time   = [1T + t (setup time) + t (hold time)]
(CMD changes)   + [1.5T + t (hold time)]

Note: The first hold time is during the second source's 1.5 T-states, so in the following formula it disappears.

rest time = 2.5T + t (setup time) + t (hold time)
(CMD changes)

For the BCP running at half speed and OCLK = 18.8696 MHz, the absolute worst case rest time is:

rest time = 5(CLK-OUT cycles) + t_{SU} + t_h
(CMD changes)

rest time = 5(53 ns) + 23 ns + 10 ns
(CMD changes)

rest time = 298 ns
(CMD changes)

For latched write mode the remote rest time starts when $\overline{\text{WR-PEND}}$ rises. The rest time for this case can be calculated as follows:

rest time = 0.5T + t (hold time)
(CMD changes)

The real time worst case for rest time in latched write mode is with the BCP running at half speed. The following is a calculation of this rest time with the BCP running at half speed and OCLK = 18.8196 MHz.

rest time = 1(CLK-OUT cycle) + t (hold time)
(CMD changes)

rest time = 53 ns + t_h
(CMD changes)

This rest time has not been specified in the DP8344AV 4.1 specifications, but an extremely conservative rule of thumb is to use one T-state as this rest time. This will assume the hold time is one half T-state long. At half speed that will be two CLK-OUT cycles and if OCLK = 18.8696 MHz, then this rule of thumb rest time equals 106 ns (following the rise of $\overline{\text{WR-PEND}}$).

Please refer to the latest datasheet for more information and the most current specifications.

DESIRABLE FEATURES OF A REST TIME CIRCUIT

In regards to designing with the rest time specifications, the first suggestion is to determine if rest time is an issue in one's design(s). If one's present or future design(s) is for systems which can never violate the rest time specification, the whole issue of rest time is a moot point.

On the other hand, designs such as terminal emulation boards, which may be placed in faster and faster PC buses, must address rest time. In slower PCs one's product may never violate rest time, but in faster PCs rest time may become an issue.

All remote accesses are susceptible to having two fast back to back accesses recognized as only one. The worst case rest time for this was determined earlier as:

rest time = 136 ns
(mistaking two accesses as one)
(where OCLK = 18.8696 MHz and the BCP runs at half speed, [CCS] = 1)

All designs with the BCP must guarantee this minimum amount of time between every access.

The second issue of remote rest time involves fast back to back accesses that have different values for CMD. The worst case for this was also calculated earlier as:

rest time = 298 ns
(CMD changes)
(where OCLK = 18.8696 MHz and [CCS] = 1)

Two ways to handle this rest time issue are:

1. Prevent all remote accesses to the BCP for at least 298 ns after the end of every remote access.

2. Hold off remote accesses that change the value of CMD for a minimum of 298 ns after the last remote access. However, allow remote accesses that do not change the value of CMD to occur a minimum of 136 ns after the last access. When the value of CMD does not change from one access to the next, this will allow accesses up to 162 ns sooner than option 1).

When designing with rest time one must decide if the increase in speed of option 2) is worth the extra logic. Howev-

er, as is demonstrated by the design example for the CT-104 (Next section), the increase in logic between option 1) and option 2) may be minimal.

Again, latched write mode is addressed separately. Unlike the other modes, latched write's rest time starts when $\overline{\text{WR-PEND}}$ rises. Two possible design options are:

1. Hold off all remote accesses for at least 106 ns (If OCLK = 18.8696 MHz) after $\overline{\text{WR-PEND}}$ rises. However, doing this will result in slowing every remote access to the BCP. Furthermore, it should be noted that $\overline{\text{WR-PEND}}$ will not rise until a minimum of three T-states after the previous access has ended. If no accesses are allowed until after $\overline{\text{WR-PEND}}$ rises, then the second access will never be mistaken as a continuation of the previous access.

2. Similar to the previous options, allow accesses after 136 ns if CMD has not changed between accesses. Then hold off access for at least 106 ns after $\overline{\text{WR-PEND}}$ rises when CMD changes between accesses.

The last design issue that must be addressed is how to wait the host processor while preventing remote accesses to the BCP. Normally the wait signal of a remote processor is driven by the XACK signal out of the BCP. (Please note that the XACK signal can be active low only when a remote access to the BCP is in progress.) During rest time, the rest time circuit prevents remote accesses to the BCP, so the XACK signal will not wait the remote processor. PC buses specify the maximum amount of time before the bus must be waited (if it is going to be waited). It is possible that not allowing remote accesses to the BCP (during rest time) may delay the XACK signal long enough to violate this bus specification. To prevent this, designs which wait a PC bus, must use logic to wait the bus whenever a remote access begins during rest time. Furthermore, the logic that starts waiting the bus before remote access is allowed to the BCP, must continue to wait the bus until XACK takes over waiting the bus.

DESIGN EXAMPLE FOR THE CT-104

The four major goals in designing a rest time circuit for the CT-104 were:

1. Keep the component count to a minimum.

2. Keep the impact to the original CT-104 design to a minimum.

3. Allow the CT-104 to operate in every mode.

4. Take advantage of the faster accesses allowed when CMD does not change from one access to the next.

The rest time circuit is implemented on one PAL16R4B and one 74ALS74. Only a single signal (REM_enable) is fed back into the original CT-104 design. In addition, the XACK signal from the BCP is now fed into the rest time PAL16R4B and the IO_CHRDY signal to the PC bus is controlled by this PAL®. This rest time circuit implements all modes and takes advantage of the increase in speed possible when CMD does not change from one access to the next.

First, how the REM_enable signal controls remote accesses will be discussed. Then, the functions implemented by the two positive-edge-triggered D flip-flops in the 74ALS74 will be discussed. Finally, a description of the operation of the rest time state machine, in the PAL16R4B, will be given. *Figure 5* is the schematic for the CT-104's rest time circuit.

The REM_enable *(Figure 5)* signal is produced in the rest time PAL16R4B and is low during rest time. After rest time is over the REM_enable signal goes high until the end of the next access, when it once again goes low during rest time.

The signal REM_enable is fed back into U22 (a PAL16L8) on the CT-104. (Note that this PAL had one unused pin so the design of this PAL was only slightly altered.)

On the original CT-104, the $\overline{\text{REMRD}}$ and $\overline{\text{REMWR}}$ outputs of U22 were buffered signals of $\overline{\text{MEMR}}$ and $\overline{\text{MEMW}}$ respectively. With the new rest time circuit both $\overline{\text{REMRD}}$ and $\overline{\text{REMWR}}$ are held high when REM_enable = 0. This prevents all remote accesses during rest time. When rest time is over REM_enable = 1 and once again, $\overline{\text{MEMR}}$ and $\overline{\text{MEMW}}$ control $\overline{\text{REMRD}}$ and $\overline{\text{REMWR}}$ respectively.

One of the D flip-flops in the 74ALS74 stores the value of the previous access's CMD (L_CMD). This value (L_CMD) was latched at the beginning of the previous valid remote access. With this value stored in a flip-flop, the rest time state machine can determine if the present value of CMD has changed since the last remote access.

The other D flip-flop acts as a part of the rest time circuit's state machine. When $\overline{\text{RAE}}$ rises (signaling the end of that access) a one (1) is latched into this flip-flop. This signal (WAIT_START) forces the state machine to move through the next three states in sequence. If this latch is not used, the rest time state machine may also miss the ending of an access if back to back accesses occur within one CLK-OUT cycle plus the setup time for a PAL16R4B's register input. If OCLK = 18.8696 MHz this time will be:

time = 1(CLK-OUT cycle) + t (setup time for PAL16R4B)

time = 53 ns + 20 ns

= 73 ns

This in effect, trades a rest time of 136 ns for one of 73 ns. However, while the output of this latch (WAIT_START, *Figure 5*) equals one, REM_enable will be low and the state machine will be forced to start the rest time states. In the third rest time state the WAIT_START latch is cleared by the $\overline{\text{CLR\_START}}$ *(Figure 5)* signal going low. $\overline{\text{CLR\_START}}$ is produced in the rest time PAL16R4B and $\overline{\text{CLR\_START}}$ equals zero (0) only when in the third rest time state. In this way the WAIT_START signal guarantees the minimal rest time of 136 ns by keeping REM_enable equal to zero through at least three CLK-OUT cycles (i.e., 3[53 ns] = 159 ns if OCLK = 18.8696 MHz).

To describe the operation of the state machine, a state by state description follows. When reading through the states one should remember that the state machine can only change states on the rising edge of CLK-OUT. A flow chart of this state machine is provided as *Figure 6*. *Figure 7* is a PAL program (written in the ABEL program language) for the PAL16R4, rest time PAL. *Figure 8* shows the reduced equations that result for the PAL program given in *Figure 7*.

STATE: IDLE

This state is entered when a system reset occurs. In this state REM_enable = 1, CMD_clk = 0, and XACK controls the state of IO_CHRDY.

The state machine will stay in this state until a valid remote access starts (i.e., $\overline{\text{RAE}}$ = 0). Then the state machine moves to CYCLE_START.

Note: On the CT-104, the signal $\overline{\text{RAE}}$ is a full decode of a valid access. This means that it decodes a valid address and a valid $\overline{\text{MEMR}}$ or $\overline{\text{MEMW}}$. If $\overline{\text{RAE}}$ is only an address decode, it alone would not indicate that a valid access had started.

STATE: CYCLE_START

In this state REM_enable = 1, CMD_clk = 1 as long as $\overline{\text{RAE}}$ = 0, $\overline{\text{CLR\_START}}$ = 1, and XACK controls the state of IO_CHRDY. Note, when CMD_clk rises it latches

2
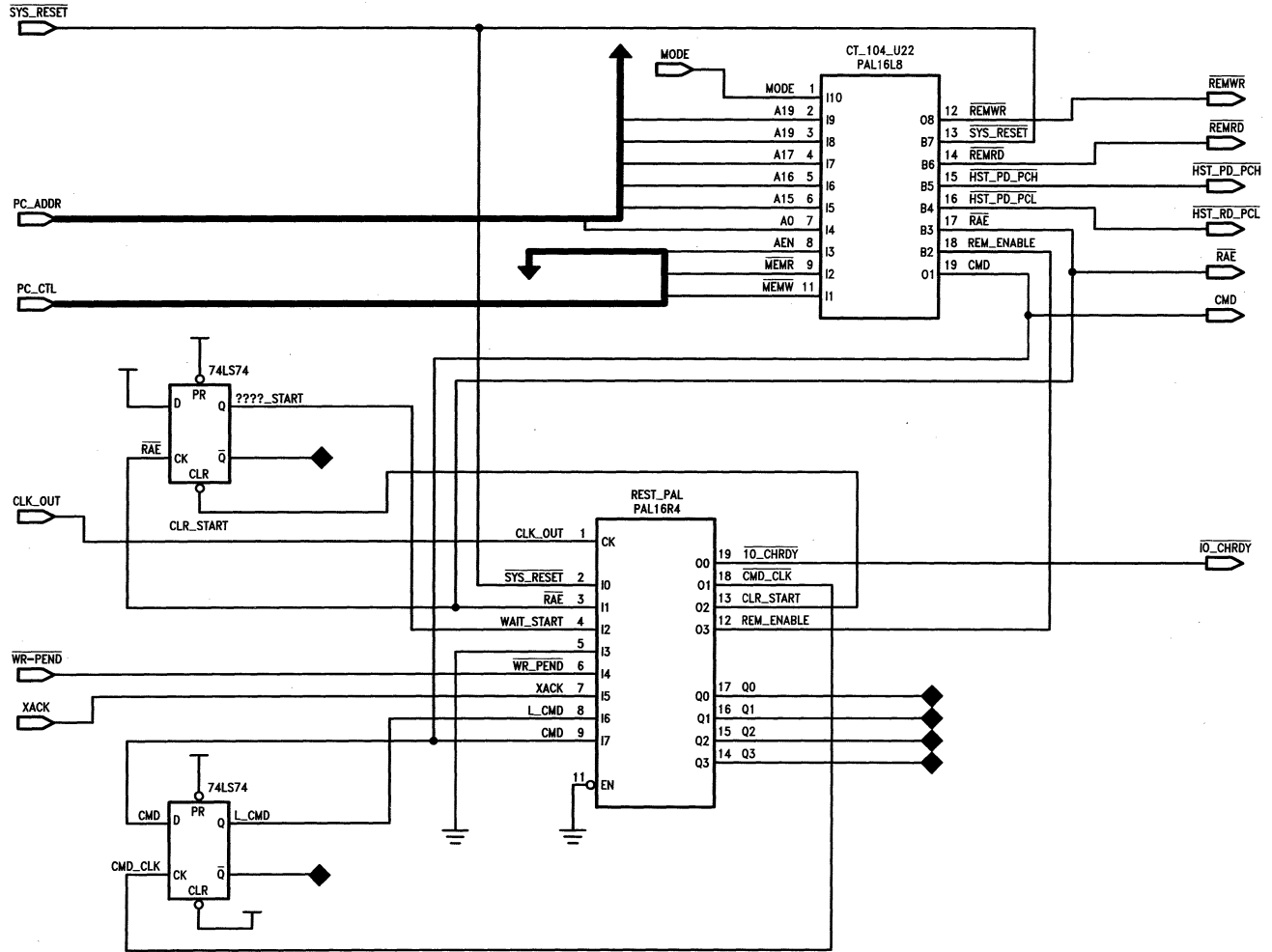
FIGURE 5
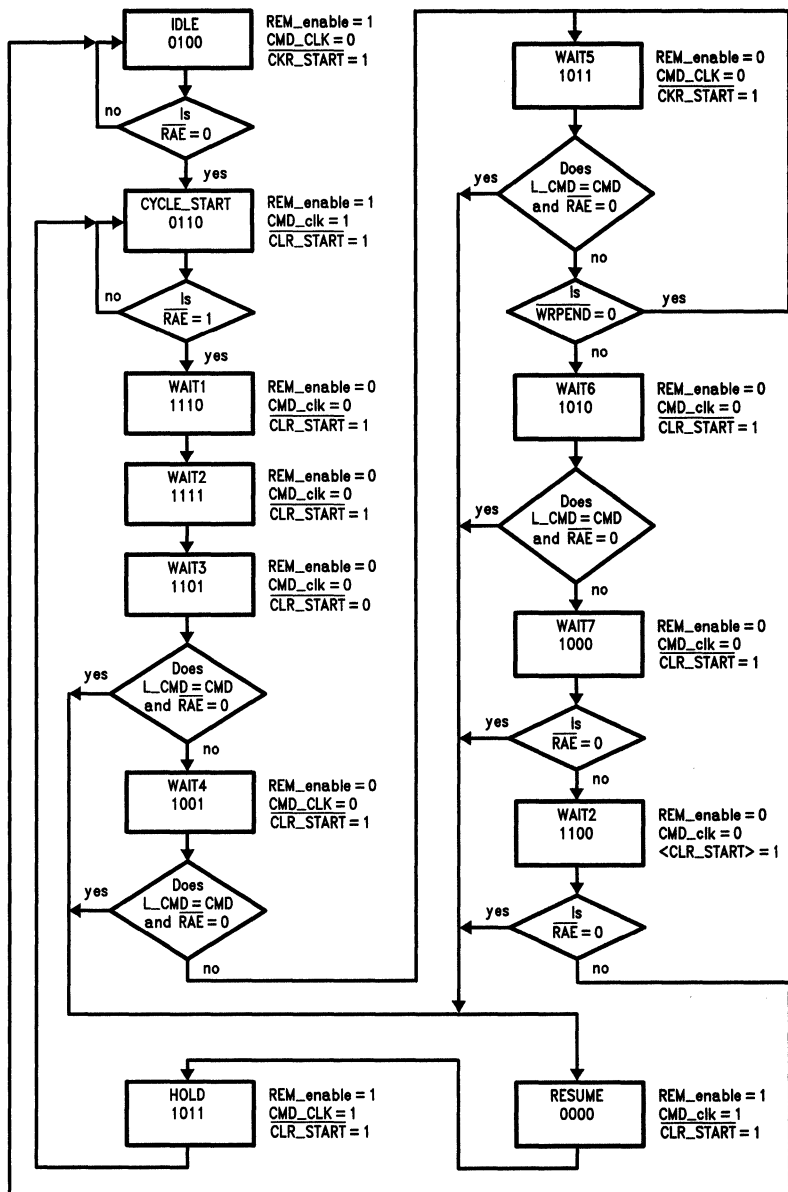
TL/F/10451–11

FIGURE 6. State Diagram of Rest Time Circuit

TL/F/10451–12

in the present value of CMD. The state machine will stay in this state until the remote access ends, indicated by either $\overline{RAE} = 1$ or WAIT__START = 1. Then the state machine moves to WAIT1.

STATE: WAIT1

In this state REM__enable = 0, CMD__clk = 0, $\overline{CLR\_}$ $\overline{START} = 1$, and if a remote access starts, IO__CHRDY is driven low whenever $\overline{RAE} = 0$. While in this state WAIT__ START remains equal to one because it has not been cleared yet. Thus, after one CLK__OUT cycle the state machine moves to WAIT2.

STATE: WAIT2

In this state REM__enable = 0, CMD__clk = 0, $\overline{CLR\_}$ $\overline{START} = 1$, and IO__CHRDY is driven low whenever $\overline{RAE} = 0$. Again WAIT__START = 1 and after another CLK-OUT cycle the state machine moves to WAIT3.

STATE: WAIT3

In this state REM__enable = 0, CMD__clk = 0, $\overline{CLR\_}$ $\overline{START} = 0$ which clears WAIT__START, and IO__CHRDY is driven low whenever $\overline{RAE} = 0$. Since WAIT__START is cleared, on the next rising edge of CLK-OUT the state machine will make a decision:

IF L__CMD equals CMD (indicating no change in the value of CMD between cycles) and a valid remote access has started (i.e., $\overline{RAE}$ = 0), then the state machine will move to the RESUME state. (The RESUME state is covered after the WAIT8 state.) However, if those conditions are not met then the state machine moves to WAIT4.

STATE: WAIT4

In this state REM__enable = 0, CMD__clk = 0, $\overline{CLR\_\_START}$ = 1, and IO__CHRDY is driven low whenever $\overline{RAE}$ = 0. If L__CMD equals CMD and $\overline{RAE}$ = 0, then on the next rising edge of CLK-OUT the state machine will move to the RESUME state. Otherwise the state machine moves to state WAIT5.

STATE: WAIT5

In this state REM__enable = 0, CMD__clk = 0, $\overline{CLR\_\_START}$ = 1, and IO__CHRDY is driven low whenever $\overline{RAE}$ = 0. IF L__CMD equals CMD and $\overline{RAE}$ = 0 then the next state will be RESUME.

As long as the above condition is not met and $\overline{WR\text{-}PEND}$ = 0, the state machine will remain in this state. $\overline{WR\text{-}PEND}$ = 0 indicates that the previous access was a write with the BCP in latched write mode. Holding the state machine at WAIT5 prevents remote accesses, that changes the value of CMD, for the required latched write rest time.

If both of the above conditions are false then the next state will be WAIT6.

STATE: WAIT6

In this state REM__enable = 0, CMD__clk = 0, $\overline{CLR\_\_START}$ = 1, and IO__CHRDY is driven low whenever $\overline{RAE}$ = 0. If L__CMD equals CMD and $\overline{RAE}$ = 0, then on the next rising edge of CLK-OUT the state machine will move to the RESUME state. Otherwise the state machine moves to state WAIT7.

STATE: WAIT7

In this state REM__enable = 0, CMD__clk = 0, $\overline{CLR\_\_START}$ = 1, and IO__CHRDY is driven low whenever $\overline{RAE}$ = 0. Any remote access that has changed the value of CMD will be prevented until the end of this state. That would be a minimum of seven CLK-OUT cycles between accesses or 371 ns if OCLK = 18.8696 MHz.

Also, all remote accesses which follow a latched write and change the value of CMD have been prevented at least two CLK-OUT cycles or 106 ns, if OCLK = 18.8696 MHz. Thus after one CLK-OUT cycle, if $\overline{RAE}$ = 0 the next state will be RESUME. Otherwise, it will be WAIT8.

STATE: WAIT8

In this state REM__enable = 1, (allows accesses), CMD__clk = 0, $\overline{CLEAR\_\_START}$ = 1, and IO__CHRDY is driven low

```
module rest_pal        flag '-r3'
title 'REST-TIME Compliance State Machine';

        REST_PAL      device        'p16r4';


"inputs:
        clock,enab     pin 1,11;
        !sys_reset     pin 2;
        !rae           pin 3;
        wait_start     pin 4;
        !wr_pend       pin 6;
        xack           pin 7;
        L_cmd          pin 8;
        cmd            pin 9;

"outputs:
        rem_enable     pin 12;
        clr_start      pin 13;
        q3,q2,q1,q0    pin 14,15,16,17;
        cmd_clk        pin 18;
        IO_chrdy       pin 19;

        sreg          = [q3,q2,q1,q0];
        outputs       = rem_enable;

"definitions:
        ck,x,z,L,H    = .C., .X., .Z.,0,1;
        access        = rae;
        st            = [q3,q2,q1,q0];

"State Values...

        idle      = ^b0100;      " 4h
        start     = ^b0110;      " 6h
        wait1     = ^b1110;      " Eh
        wait2     = ^b1111;      " Fh
        wait3     = ^b1101;      " Dh
        wait4     = ^b1001;      " 9h
        wait5     = ^b1011;      " Bh
        wait6     = ^b1010;      " Ah
        wait7     = ^b1000;      " 8h
        wait8     = ^b1100;      " Ch
        resume    = ^b0000;      " 0h
        hold      = ^b0010;      " 2h
        notused1  = ^b0111;      " 7h
        notused2  = ^b0011;      " 3h
        notused3  = ^b0101;      " 5h
        notused4  = ^b0001;      " 1h
```

TL/F/10451–14

**FIGURE 7. PAL Program File
(Written in the ABEL Program Language)**

equations

    enable outputs = 1;

    enable IO_chrdy = access;

    !IO_chrdy   = (q3 * access) # (!q2 * access) # (q0 * access)
            # (!xack) # (wait_start * access);

    !clr_start = ((q3) * (q2) * (!q1) * (q0))
            # (sys_reset);

    cmd_clk   = (access * !q3 * !q2 * !q0 * !wait_start)
            #(access * !q3 * q1 * !q0 * !wait_start)
            # (access * cmd_clk * !wait_start);

    !rem_enable = (!q2 * q3) # q0 # (q1 * q3) # wait_start;

state_diagram sreg;

        State idle: " Remain in idle while sys_reset is active.
          IF (sys_reset) THEN idle;
           ELSE IF (access) THEN start;
          ELSE idle;

        State start: " Begin normal access.
          IF (sys_reset) THEN idle;
          ELSE IF (!access # wait_start) THEN wait1;
          ELSE start;

        State wait1:   " First wait cycle.
          IF (sys_reset) THEN idle;
          ELSE IF (access & L_cmd & cmd & !wait_start) THEN resume;
          ELSE IF (access & !L_cmd & !cmd & !wait_start) THEN resume;
          ELSE wait2;

        State wait2:
          IF (sys_reset) THEN idle;
          ELSE IF (access & L_cmd & cmd & !wait_start) THEN resume;
          ELSE IF (access & !L_cmd & !cmd & !wait_start) THEN resume;
          ELSE wait3;

        State wait3:
          IF (sys_reset) THEN idle;
          ELSE IF (access & L_cmd & cmd & !wait_start) THEN resume;
          ELSE IF (access & !L_cmd & !cmd & !wait_start) THEN resume;
          ELSE wait4;

TL/F/10451-15

FIGURE 7. PAL Program File (Written in the ABEL Program Language) (Continued)

whenever $\overline{RAE}$ = 0. This state was included in the state machine to reduce the state machine's logic. Otherwise it would have been logical to return to the IDLE state from WAIT7 if $\overline{RAE}$ = 1 (no access in progress). If $\overline{RAE}$ = 0, then the next state will be RESUME. Otherwise the state machine returns to IDLE.

STATE: RESUME

In this state REM_enable = 1, CMD_clk = 1 (rising edge of CMD_clk latches in the present value of CMD), $\overline{CLR\_START}$ = 1, and IO_CHRDY is driven low while $\overline{RAE}$ = 0. When the state machine moves to this state, it means that a remote access took place quickly after the previous access. The state machine has allowed the remote access

to proceed. However, the state machine must have waited the PC-bus for some period of time before entering this state. As a result, the PC-bus should be waited until the XACK signal can take over control of driving IO_CHRDY. For the design of the CT-104, it was determined that once REM_enable = 1, the XACK signal would take over control within two CLK-OUT cycles. So the state machine will wait the PC-bus through this state and the next. On the next rising edge of CLK-OUT the state machine will move to the HOLD state.

STATE: HOLD

In this state REM_enable = 1, CMD_clk = 1, $\overline{CLR\_START}$ = 1, and IO_CHRDY is driven low while $\overline{RAE}$ = 0. Again, this state is provided to wait the PC-bus for a second CLK-OUT cycle while still allowing remote access. The next state is CYCLE_START. In CYCLE_START, XACK will take over control of IO_CHRDY.

```
State wait4:
    IF (sys_reset) THEN idle;
    ELSE IF (access & L_cmd & cmd & !wait_start) THEN resume;
    ELSE IF (access & !L_cmd & !cmd & !wait_start) THEN resume;
     ELSE wait5;

State wait5:
    IF (sys_reset) THEN idle;
    ELSE IF (access & L_cmd & cmd & !wait_start) THEN resume;
    ELSE IF (access & !L_cmd & !cmd & !wait_start) THEN resume;
    ELSE IF (wr_pend) THEN wait5;
    ELSE wait6;

State wait6:
    IF (sys_reset) THEN idle;
    ELSE IF (access & L_cmd & cmd & !wait_start) THEN resume;
    ELSE IF (access & !L_cmd & !cmd & !wait_start) THEN resume;
    ELSE wait7;

State wait7:
    IF (sys_reset) THEN idle;
    ELSE IF (access) THEN resume;
    ELSE wait8;

State wait8:
    IF (sys_reset) THEN idle;
    ELSE IF (access) THEN resume;
    ELSE idle;

State resume:
    IF (sys_reset) THEN idle;
    ELSE hold;

State hold:
    IF (sys_reset) THEN idle;
    ELSE start;

State notused1:
    IF (sys_reset) THEN idle;
    ELSE wait2;

State notused2:
    IF (sys_reset) THEN idle;
    ELSE wait2;

State notused3:
    IF (sys_reset) THEN idle;
    ELSE wait2;

State notused4:
    IF (sys_reset) THEN idle;
    ELSE wait2;

end
```

TL/F/10451–16

FIGURE 7. PAL Program File (Written in the ABEL Program Language) (Continued)

REST-TIME Compliance State Machine
Equations for Module rest_pal

Device REST_PAL

Reduced Equations:

enable rem_enable = (1);

enable IO_chrdy = (!~rae);

!IO_chrdy = (!~rae & wait_start
        # !xack
        # q0 & !~rae
        # !q2 & !~rae
        # q3 & !~rae);

!clr_start = (!~sys_reset # q0 & !q1 & !q2 & q3);

!cmd_clk = (wait_start
        # !cmd_clk & q0
        # !cmd_clk & !q1 & q2
        # !cmd_clk & q3
        # ~rae);

!rem_enable = (wait_start # q1 & q3 # q0 # !q2 & q3);

!q3 := (!q0 & !q2 & !q3
    # !q0 & !q1 & !~rae
    # !L_cmd & !cmd & q3 & !~rae & !wait_start
    # L_cmd & cmd & q3 & !~rae & !wait_start
    # !q0 & !q3 & !~rae & !wait_start
    # !~sys_reset
    # !q0 & !q1 & q2);

!q2 := (!q0 & !q1 & !q2 & !q3 & ~sys_reset
    # !q1 & q3 & !~rae & ~sys_reset
    # q1 & !q2 & q3 & ~sys_reset
    # q0 & !q1 & q3 & ~sys_reset
    # !L_cmd & !cmd & q3 & !~rae & ~sys_reset & !wait_start
    # L_cmd & cmd & q3 & !~rae & ~sys_reset & !wait_start);

!q1 := (!q0 & !q1 & q3
    # !q0 & !q2 & q3
    # q0 & q2 & q3
    # !L_cmd & !cmd & q3 & !~rae & !wait_start
    # L_cmd & cmd & q3 & !~rae & !wait_start
    # !q0 & !q1 & q2 & ~rae
    # !~sys_reset);

REST-TIME Compliance State Machine
Equations for Module rest_pal

Device REST_PAL

!q0 := (!q0 & !q1
    # !q0 & !q2
    # q1 & !q2 & q3 & ~wr_pend
    # !L_cmd & !cmd & q3 & !~rae & !wait_start
    # L_cmd & cmd & q3 & !~rae & !wait_start
    # !~sys_reset
    # !q0 & !q3);

TL/F/10451-18

TL/F/10451-17

FIGURE 8. Reduced Equations for Rest Time State Machine PAL

DP8344 Timer Application

National Semiconductor
Application Note 626
William V. Miller

INTRODUCTION

The DP8344 is a communications processor which handles IBM 3270, 3299 and 5250 protocols along with NSC general 8-bit protocol. In order to reduce the impact on the DP8344's CPU the timer was designed to stand-alone and count independently of the CPU.

The timer's circuitry includes a unique holding register. This holding register can be loaded with a sixteen-bit countdown-value, which will remain unchanged until a new value is loaded or the DP8344 is reset.

When the timer counts to zero it takes two actions: 1) it sets both the timer interrupt and the Time Out flag [TO], and 2) the timer reloads the sixteen-bit countdown-value stored in the holding register and continues the countdown cycle. This demonstrates a significant advantage of the DP8344's timer; the timer continues keeping accurate time while notifying the CPU that the timer has completed a cycle. The timer does not wait for the CPU to service it, instead the timer notifies the CPU of the completion of a cycle and allows the CPU to take the desired action when it has the time.

With the use of the holding register, a multiple number of timer cycles of the exact same duration can be performed consecutively. The other major advantage of the holding register is that it allows the interleaving of any number of countdown-values. Loading the holding register with a new countdown-value does not affect the countdown-value presently in the timer's countdown circuitry. In this way a countdown-value (call it A) can be counting down and the holding register can be loaded with a new countdown-value (call it B). When the value A reaches zero, both the timer interrupt and Time Out flag [TO] are set, and the value B is loaded into the countdown circuitry and starts its countdown. Then the value A can be loaded back into the holding register when the CPU has the time. This demonstrates how countdown-values with different durations can be interleaved and once again how the timer does not have to wait to be serviced by the CPU, making both the timer and CPU more efficient.

The CPU can load the upper and lower bytes of the holding register by writing the desired value to the CPU registers {TRH} and {TRL} respectively.

Control of the timer's countdown circuitry is maintained via three bits in the Auxiliary Control Register {ACR}.

Timer STart [TST] (bit 7 of {ACR}) is the start/stop control bit for the timer. Writing a one to [TST] starts the timer counting down from the present value in the countdown circuitry. When [TST] is zero the timer stops and the timer interrupt is cleared.

The second control bit is Timer LoaD [TLD] (bit 6 of {ARC}). This bit allows the CPU to immediately load the timer's countdown circuitry with the value in the timer's holding register. This capability is required after the DP8344 is reset; the value in the timer's countdown circuitry will be the reset value and not the desired value. CPU controlled loading can also be used to load higher priority countdown-values before a lower priority countdown is completed. The 5250 Protocol application implements the timer in this manner.

Writing a one to [TLD] will load the timers countdown circuitry with the value in the timer's holding register and initializes the timer clock in preparation to start counting down. Upon completing the load operation [TLD] is cleared by internal hardware.

When the timer is loaded by writing a one to [TLD], the timer is re-initialized to prevent the timer's circuitry from decrementing the newly loaded countdown-value prematurely. By initializing the countdown circuitry after a CPU load, the newly loaded countdown-value's duration will be accurately measured. The reader should note that there is no way to precisely measure the total elapse time of two or more countdown-values if the CPU loads them (using [TLD]) into the countdown circuitry. However, the error due to CPU loading will be a maximum of one period of the timer for each CPU load and can often be ignored if the countdown values are large.

EXAMPLE: countdown-value   = 1000
            maximum count error = 1
            maximum error   = 0.1%

The last control bit is TiMer Clock select (bit 5 of {ACR}). This bit determines the rate at which the countdown-value will be decremented. When [TMC] is low, the timer decrements the countdown-value at one-sixteenth the CPU's clock frequency. When [TMC] is high the rate is one-half the CPU's clock frequency. The reader should note that the timer's decrement rate is based on the CPU's clock frequency, which is controlled by CPU Clock Select [CCS] (bit 7 of {DCR}). When [CCS] is low the CPU's clock frequency equals the oscillator's clock frequency, and when [CCS] is high the CPU's clock frequency equals one-half the oscillator's clock frequency.

The last portion of the timer's circuitry is a sixteen-bit output register. This output register is loaded with the present value of the countdown-value in the countdown circuitry, at the end of every execution cycle. This register is loaded even if the timer is stopped.

The CPU can read the upper and lower bytes of this output register by reading the CPU registers {TRH} and {TRL} respectively.

The reader should note that when the CPU reads and writes to the registers {TRH} and {TRL} the timer's circuitry accesses different registers. All writes will load the timer's holding register and all reads will read the timer's output register.

The count status of the timer can be monitored by reading {TRL} and/or {TRH}. When the registers are read, the value in the timer's output register is presented to the CPU and not the value in the input holding register. To read back what was written to {TRL} and {TRH}, the timer must be loaded first, followed by a one instruction delay before reading {TRL} and {TRH} to allow the output register to be updated after the load operation. *Figure 1* is a block diagram of the Timer-CPU interface.
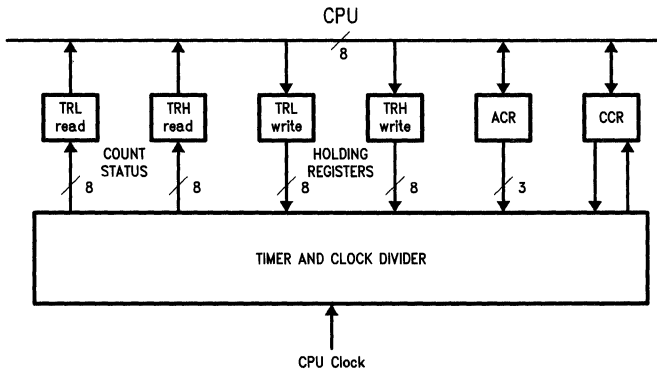
FIGURE 1. Block Diagram of Timer-CPU Interface

TL/F/10450-1

TIMER OPERATION

This section of the application note reviews the general operation of the timer. Constraints and suggestions for software are included as well as a short review of the timing equations.

After the desired sixteen bit time-out value is written into the timer's holding register via {TRL} and {TRH}, the start, load and clock selection can be achieved in one write to {ACR}. A glitch, which will cause a loss of timer accuracy, may occur if the timer's clock frequency is changed while the timer is running. To prevent this, a restriction exists on changing the timer's clock frequency in that [TMC] should not be changed while the timer is running (i.e., [TST] is high). After the write to {ACR}, the timer starts counting down at the selected frequency starting with the loaded value from the timer's holding register. Upon reaching a count of zero the timer reloads the current word in its holding register and recycles through the count.

The timing waveforms shown in *Figure 2* show a write to {ACR} that loads, starts and selects the divide by two of the CPU clock rate. The timer interrupt has also been selected. Prior to the write to {ACR}, the holding register in the timer was loaded with 0002 (Hex) by writing 02 (Hex) and 00 (Hex) to {TRL} and {TRH} respectively. The timer interrupt has also been selected.

The timer can be selected as an interrupt source by unmasking it in the Interrupt Control Register {ICR}. This is achieved by writing a zero to bit 4 of {ICR} and asserting the Global Interrupt Enable [GIE] (bit 0 of {ARC}). The timer interrupt is the lowest priority interrupt and is latched and maintained until it is cleared in software. If the timer times out prior to T2 of an instruction, the call to the interrupt service routine will occur in the next instruction. When the time out occurs in T2, the call will occur in the instruction after the next instruction.

The timer may also be used in a polled configuration. This is achieved by masking the timer interrupt bit (i.e., writing a one to bit 4 of {ICR}) and writing software which will poll the [TO] flag (bit 7 of {CCR}). Both [TO] and the timer interrupt are set high when the timer counts to zero.

Then the timer reloads the current word in its holding register and recycles through the count. This means that the timer continues to keep track of time while leaving the task of handling the timer interrupt and/or the [TO] poll to be performed by the CPU. To operate correctly in the polled configuration, software must be written that will guarantee that [TO] is polled and cleared at a rate that prevents [TO] being set twice before it is polled again.

The interface between the CPU and the timer allows only one byte of information to be transferred at a time. This
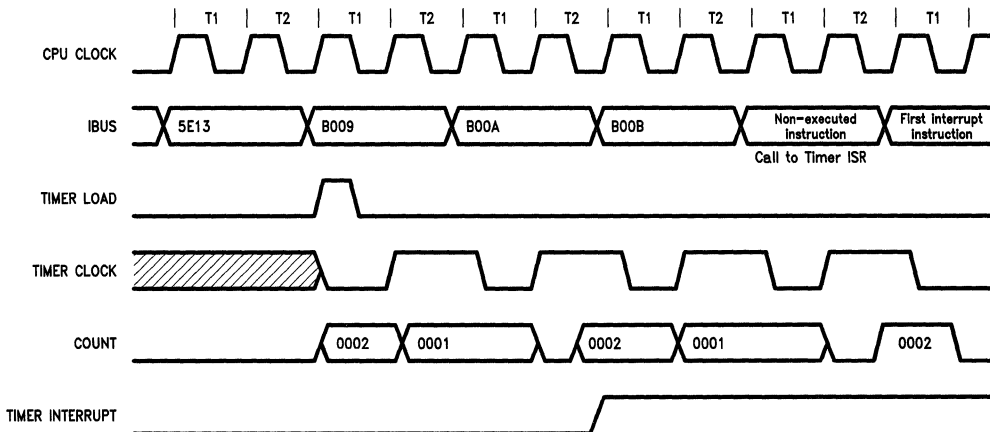
FIGURE 2. Timing Waveforms of Timer Operation

TL/F/10450-2

prevents the CPU from accessing both {TRL} and {TRH} in the same instruction. Since the timer's output register is updated after every instruction cycle, two consecutive reads of {TRL} and {TRH} will not correspond to the same count status in the timer. This error will be slight except when one of the output register values rolls over or when the count in the timer reaches zero and the timer reloads between instructions.

The suggested software for this situation is to read {TRH}, then read {TRL} and then read {TRH} again. If the values for both reads of {TRH} are the same, then the output register values did not roll over and the timer did not reload. This eliminates the error due to rolling over or reloading, but increases the amount of software.

The reader must be aware that stopping the timer (i.e., writing a zero to [TST]) will clear both the [TO] flag and the timer interrupt. For the case where the timer counts down to zero just prior to or during a stop timer instruction, the [TO] flag and timer interrupt will be cleared before the software can take the desired action. Thus, the information that the timer counted to zero will also be lost. The software to handle this situation should check the [TO] flag one instruction before the stop instruction and then check the value in the timer's output register one instruction after the stop instruction. Checking the [TO] flag before the stop instruction will insure that any previous count to zero will be verified. On the other hand, if the [TO] flag was low and the value in the output register is the same as the value stored in the holding register, then the timer counted to zero and reloaded just prior to or during the stop instruction.

For any value except 0000 (Hex) loaded into the timer's holding register, the following equations can be used to determine the time out delay for that value:

With the CPU running with a 18.8 MHz crystal, the maximum single loop time out attainable would be 55.6 ms ([TMC] = 0). The minimum time out with the same constraints is 106 ns ([TMC] = 1). For accumulating time out intervals, the total time out is simply the number of loops accumulated multiplied by the calculated Timeout. The equations above do not account for any overhead for processing the timer interrupt and for precision timing this may need to be included.

INTRODUCTION TO APPLICATIONS

In a communications environment a timer may be needed to determine the appropriate response time, the polling rate of a device or the length of a signal.

The first two applications discussed are for the communications environment.

In the first application the response time for the BCP operating in the 5250 protocol mode is controlled by the timer.

In the second application, the serial input from a keyboard is connected to the DP8344's BIRQ pin and the timer determines at what rate the input is sampled to read in the valid keystroke serial data.

To further demonstrate the timer's versatility the last two applications discuss how to implement basic timer uses not restricted to the communications environment; namely blinking the terminal's cursor and a real time clock.

All four applications implement the timer as an interrupt source, none poll the [TO] flag. Using the interrupt reduces the amount of software needed and it also results in the fastest responses to a time out. However, the reader should note that the [TO] flag may be read even during other interrupt routines while the timer's interrupt is masked off. This may be important if the other interrupt routines are long and could delay the service of the timer interrupt longer than the

$$\text{Timeout} = (\text{value in the holding register}) \times \text{Tcpu} \times \begin{cases} 2 \; ; & [\text{TMC}] = 1 \\ 16; & [\text{TMC}] = 0 \end{cases}$$

where:

$\text{Tcpu} =$ The period of the CPU clock
CPU clock = oscillator clock rate ; [CCS] = 0
CPU clock = ½ oscillator clock rate ; [CCS] = 1

$\text{Timeout} =$ The amount of time after the end of the instruction that asserts [TST]

When the value of 0000 (Hex) is loaded into the timer, the maximum time out is obtained and is calculated as follows:

$$\text{Timeout} = 65536 \times \text{Tcpu} \times \begin{cases} 2 \; ; & [\text{TMC}] = 1 \\ 16; & [\text{TMC}] = 0 \end{cases}$$

time out length. Thus the [TO] flag can be used to guarantee the timer is serviced even during anothers interrupts service routine.

5250 PROTOCOL

Introduction

The DP8344 Biphase Communication Processor (BCP) is capable of responding to received data within 5.5 μs. This is a stringent requirement for the IBM 3270 protocol. However, the IBM 5250 protocol requires a response time of 45 μs $\pm$ 15 μs. Obviously the powerful BCP will respond too rapidly if it is not programmed to wait at least 30 μs before responding.

Also while operating in the IBM 5250 protocol mode, the BCP often expects to be polled at some minimum rate. In this discussion the BCP expects to be polled within every two seconds. If the BCP is not polled within this time it is assumed that a problem exists and the BCP is programmed to reset.

In this application the timer and some DP8344 software are used to guarantee the proper response time, and to determine how long it has been since the last poll.

General Description

For the majority of time, the timer will be used to keep track of the real-time which has transpired since the BCP was last polled. However, once a receiver interrupt is set, the timer loads and counts down a 45 μs delay value. This count down is used to delay the BCP's response so that it will lie between 30 μs and 60 μs. After the 45 μs delay value is handled, the timer returns to keeping track of the two seconds of real-time.

Resetting the BCP, after two seconds have passed since it was last polled, is not a stringent requirement. Thus the 45 μs delays are not included in the two seconds. In effect a receiver interrupt 1) stops the timer, 2) records the present value of the two second count down value, 3) loads and counts down the 45 μs delay value, and 4) reloads and continues the two second count down from the value recorded after the receiver interrupt stopped the timer.

Detailed Description

After a reset the timer must be programmed to operate in the desired configuration before the BCP can start its operation in the 5250 protocol mode. For this application the timer is pre-configured to divide the CPU clock by sixteen (CPU clock = $\frac{1}{2}$ oscillator clock, OCLK = 18.8696 MHz). All interrupts are unmasked and enabled. The time out value 60BE (Hex) is then loaded into the timer's holding register via {TRL} and {TRH}.

After the timer is programmed properly the timer is loaded and started with one write to {ACR}. The reader will note that the count down value of 60BE (Hex) corresponds to 21 ms not two seconds. As shown earlier in this application note (OPERATION section), the maximum single loop time out attainable for this mode is 55.6 ms. Since it is impossible to load the timer with a countdown-value of two seconds,

software is written to record the number of times the 21 ms count down value reaches zero. Still more software is responsible for resetting the BCP if one hundred time outs occur before the BCP is polled again. The use of 21 ms time outs instead of 20 ms time outs will guarantee that a minimum of two seconds has passed, even if there are small timing errors by either the controller or the BCP's oscillator clock.

The timer will continue to count down the 21 ms time outs until a receiver interrupt is set. The BCP's software then calls a receiver interrupt service routine. (Refer to *Figure 3* for the BCP code for this service routine.) The timer is stopped. The present value of the 21 ms count down is stored in a temporary memory location. The time out value 0011 (Hex) which corresponds to 28 μs is loaded into the timer's holding register via {TRL} and {TRH}. (Adding the delay due to setting up and responding to the timer together with the 28 μs time out, results in a total elapse time delay which guarantees the response between 30 μs and 60 μs.) The timer is loaded and restarted. Then the partially completed 21 ms countdown-value stored in a temporary memory location is loaded into the timer's holding register via {TRL} and {TRH}. Once the 28 μs time out counts to zero, the partially completed 21 ms time out is resumed as the timer is loaded with the value in the holding register and continues the 21 ms count down.

Every time the timer counts down to zero, it sets the timer interrupt and the DP8344 is programmed to call a timer interrupt service routine. In order to operate correctly the service routine must first determine if a 21 ms or a 28 μs time out has occured. If a 28 μs time out has taken place, the timer is stopped. The value in the timer's holding register will not be the 21 ms count down value; it will be the value which was in the timer when the receiver interrupt stopped the timer. So the 21 ms countdown-value 60BE (Hex) is loaded into the timer's holding register via {TRL} and {TRH}. Then the timer is started. If a 21 ms timer interrupt is pending it will be serviced, otherwise the software will return with all interrupts unmasked and enabled.

In the case of a 21 ms timer interrupt, the number of 21 ms time outs is recorded for all seven sessions in data memory. For every 21 ms timer interrupt a one is added to the value stored in data memory for each session. An exception is made when FF (Hex) is the value stored in data memory. Adding a one would result in the value 00 (Hex) replacing FF (Hex) in memory. This would falsely indicate that less than 21 ms has passed since the BCP was last polled. As a result if FF (Hex) is the number in memory nothing is added to it. As before the software returns with all interrupts unmasked and enabled.

The software which 1) clears the number of 21 ms time outs recorded when the BCP is polled and 2) resets the BCP after two seconds have passed without the BCP being polled, is not discussed in this application note because it does not effect the normal operation of the timer.

This application describes the use of the timer in the 5250 protocol mode as it is implemented in the Multi-Protocol Adapter (MPA).

2

```
;---------------------------------------------------------------------
;       name: tw_timer_int
;
;       description: The timer interrupt service routine is responsible
;               for:
;                   1) Maintains a real time clock counter for each session:
;                       - Increments a real time clock counter which controls
;                         System Available flag, auto reset and reset
;                       complete;
;                       - Prevents counter roll over by keeping a max count
;                       of FFh;
;                   2) Provides 45us time out signal for poll response
;                       - If interrupt is due to 45us poll response, unmasks
;                       Tx int to
;                       allow for response.
;               note : The timer interrupt service routine lock out host
;                       access and
;                       other interrupts except TFE interrupt.
;       scope: global
;
;       entry: timer interrupt hits, ie. timer reaches a count of zero.
;               the timer is pre-configured to use 1/16 cpu clock with a
;               count value of 305Fh which corresponds to 21ms.
;       inputs: 1) tw_sysa_por_cnt(0-6)
;                   real time clock counters, reset to 0 by receiver when
;                   Poll received, and
;                   by session task when going to do a POR.
;               2) tw_sysa_resp_flag (in RSTATE)
;               - TW_TIMER_RESP
;                   timer response flag, set by receiver for 45us poll ]
;                   response.
;               - TW_TO_PEND
;                   timer interrupt pending flag, set by receiver if it sees
;                   a pending
;                   timer interrupt.
;
;       exit:
;       outputs: 1) tw_sysa_por_cnt(0-6)
;                   for all sessions, counters are incremented by 1.
;                   Counters will remain in 'FF' without roll over.
;               2) tw_sysa_resp_flag (in RSTATE)
;                   - TW_TIMER_RESP
;                       reset if interrupt is due to 45us poll response.
;                   - TW_TO_PEND
;                       reset if there is a pending timer interrupt.
;
```

TL/F/10450–3

FIGURE 3

```
;
;       COPYRIGHT NATIONAL SEMICONDUCTOR, INC. 1987,1988
;--------------------------------------------------------------------
;
;              pseudo code
;
;tw_rel_time_clk_timer()
;
;
;{
;       lock out remote access
;         reset time-out flag by setting [TO] to 1;
;         if (time-out of 45us)
;      {
;              stop timer;
;              reload timer input register;
;              start timer;
;                  allow poll response by unmasking transmitter interrupt;
;               reset poll response flag;
;              if NOT (timer interrupt pending)
;              {
;                     call check birq;
;              enable interrupt;
;                     return with flags and reg banks restored;
;              }
;         enable interrupt;
;      }
;       push regs being used in following section;
;        for all 7 sessions do
;      {
;                 if tw_sysa_por_cnt = 5sec next session
;            else
;            {
;                      increment tw_sysa_por_cnt;
;                  next session;
;            }
;      }
;        restore registers;
;          set interrupt mask to enable all interrupts;
;        call check birq;
;          return with flags and reg banks restored;
;}
;
;
   ;--------------------------------------------------------------------
```

TL/F/10450–4

FIGURE 3 (Continued)

2

```
;
        .GLOBAL dca_fast_birq, tw_tx_tm_entry

TW_TIMER.BCP:        .SECT  X


  tw_timer_int::

            exx    MA,AB                 ; switch reg bank
            PUSHP  IZ                     ; save IZ
            or     CCR_TO,CCR             ; clear time out of timer
            LJMPBP  RSTATE,TW_TIMER_RESP,NS,tm_relti_clk
                                          ;jump to real time clock counter

; timer poll/activate read response timeout and offline response
; timing

            and  ~ACR_TST,ACR            ; stop timer
            move        TM_21MS_HI,ACC    ; prepare timer input
                                             upper byte
            move        ACC,TRH           ; move to TRH
            move        TM_21MS_LO,ACC    ; prepare timer input
                                             lower byte
            move        ACC,TRL           ; move to TRL
            or     ACR_TST,ACR            ; start timer
            LJMPBP      RSTATE,RX_RESPONSE_WAIT,S,tm_skip_tfe
                                          ; if offline response
                                            timeout, skip tfe call
            and    ~ICR_TX,ICR            ; unmask Tx interrupt
                                            since interrupt expects
                                            to be unmasked
            lcall  tw_tx_tm_entry         ; go handle response
                                            via TFE interrupt
tm_skip_tfe:
            LJMPBP      RSTATE,TW_TO_PEND,S,tm_relti_clk_1
                                          ; jump to real time clock
                                            if interrupt pending
            and    ~(TW_TIMER_RESP|RX_RESPONSE_WAIT),RSTATE;
                                          ; reset poll response flag
            jmp  tm_check_birq            ; go check birq, do birq
                                            if needed [V0.5]



; real timer clock counter

tm_relti_clk_1:
   and ~(TW_TIMER_RESP|RX_RESPONSE_WAIT|
        TW_TO_PEND),RSTATE
                                          ; reset poll response, response
                                            wait, and int pending FLAGS
tm_relti_clk:
            move  DCPHI,IZHI              ; setup IZHI
            move  LOW(tw_sysa_por_cnt0-1),ACC    ; setup IZLO
            move  ACC,IZLO                ;
            move  1,ACC                   ; set a '1' for
                                            later use
            move  [+IZ],GP7               ; get counter
            cmp   GP7,TM_5SEC             ; equal to 5.4sec?
            jz    tm_next_1               ; yes, goto next session
                                            without counter +1
            adda  GP7,[IZ]                ; increment counter
```

TL/F/10450–5

FIGURE 3 (Continued)

```
tm_next_1:
        move    [+IZ],GP7           ; get counter
        cmp     GP7,TM_5SEC          ; equal to 5.4sec?
        jz      tm_next_2           ; yes, goto next session
                                        without counter +1
        adda    GP7,[IZ]            ; increment counter
tm_next_2:
        move    [+IZ],GP7           ; get counter
        cmp     GP7,TM_5SEC         ; equal to 5.4sec?
        jz      tm_next_3           ; yes, goto next session
                                        without counter +1
        adda    GP7,[IZ]            ; increment counter
tm_next_3:
        move    [+IZ],GP7           ; get counter
        cmp     GP7,TM_5SEC         ; equal to 5.4sec?
        jz      tm_next_4           ; yes, goto next session
                                        without counter +1
        adda    GP7,[IZ]            ; increment counter
tm_next_4:
        move    [+IZ],GP7           ; get counter
        cmp     GP7,TM_5SEC         ; equal to 5.4sec?
        jz      tm_next_5           ; yes, goto next session
                                        without counter +1
        adda    GP7,[IZ]            ; increment counter
tm_next_5:
        move    [+IZ],GP7           ; get counter
        cmp     GP7,TM_5SEC  .      ; equal to 5.4sec?
        jz      tm_next_6           ; yes, goto next session
                                        without counter +1
        adda    GP7,[IZ]            ; increment counter
tm_next_6:
        move    [+IZ],GP7           ; get counter
        cmp     GP7,TM_5SEC          ; equal to 5.4sec?
        jz      tm_next_end         ; yes, goto next session
                                        without counter +1
        adda    GP7,[IZ]            ; increment counter

tm_next_end:
tm_check_birq:                      ; following codes added
                                      in [V0.5]
        ljmp CCR,BIRQ,S,tm_no_birq  ; check pending
                                      birq
        lcall dca_fast_birq         ; yes, go do it
tm_no_birq:
        POPP        IZ              ; restore Z


        UNLOCK                      ; unlock remote
        ret    RI,RFB              ; return with GIE, ALU flags
                                       and reg bank restored


        .end
```

TL/F/10450-6

FIGURE 3 (Continued)

DP8344 AS A SERIAL INPUT FROM A KEYBOARD

Introduction

To keep the cost of terminals low, the 3270 protocol was designed to place all of the intelligence of the system in the cluster controller while all of the memory remained in the terminal. In this protocol the terminal is responsible for recording all keystrokes until the cluster controller can poll the terminal and process the keystroke data.

With that in mind this application uses the timer along with the BIRQ interrupt pin as a serial port to read in keystrokes from a serial keyboard. The timer in this application is used to read the serial keystroke dataword at the proper baud rate. (Refer to *Figure 4* for the actual BCP code used for this application.)

Description

The specifications for the serial dataword produced by the serial keyboard and read by the DP8344 are as follows: it is 1) asynchronous, 2) ten bits long (1 startbit, 8 databit with the most significant bit first, and 1 stopbit), and 3) transmitted at 1200 baud. When no serial data is being transmitted

the serial data line will be held high. The start bit will be a zero to indicate the beginning of the serial bit string.

As mentioned in the introduction, the serial data line from the keyboard is connected to the BIRQ interrupt pin (Pin 53 on the DP8344). The BIRQ interrupt pin acts as the serial port through which the serial keystroke dataword is read into the DP8344.

First, BCP software programs the timer to determine the baud rate at which the DP8344 will read the serial dataword presented at the BIRQ pin. The timer is pre-configured to divide the CPU clock by two (i.e., [TMC] = 1) with the CPU clock set equal to the oscillator clock at 18.8696 MHz (i.e., [CCS] = 0). The time out value of 03D7 (Hex) is loaded into the timer's holding register via {TRL} and {TRH}. The time out value of 03D7 (Hex) corresponds to 0.104188748 ms or approximately one eighth of 0.833333 . . . ms, which is the period of one bit at 1200 baud. After the holding register is loaded, the timer is loaded but not started. Both the timer and BIRQ interrupts are unmasked and enabled. Now the DP8344 is ready to read an asynchronous, ten bit long serial keystroke dataword at 1200 baud via its BIRQ interrupt pin.

After the timer is configured the DP8344's software can perform other operations until a zero on the serial data line activates the BIRQ interrupt (Note: the BIRQ interrupt is active low). The software then jumps to a BIRQ interrupt service routine. The service routine will mask off the BIRQ interrupt and start the timer and then return to perform other operations. After four consecutive timer interrupts the middle of the startbit should be present at the BIRQ pin. To ensure that a glitch or noise did not produce a zero momentarily and that the zero is actually a startbit, the BIRQ interrupt is unmasked. If the value at the BIRQ pin is a one instead of a startbit zero, the timer is stopped and reloaded with the countdown-value 03D7 (Hex). The BIRQ interrupt will remain unmasked waiting for the next zero. However, if the value at the BIRQ pin is a zero (indicating a valid startbit), the software jumps to the BIRQ interrupt service routine. The BIRQ interrupt is masked off and the timer continues to run and the software returns to perform other operations.

For the case of a true startbit the DP8344 needs to read in the value of the serial keystroke dataword. The value of each data bit must be read one at a time. After each value is read, it is added to a temporary value stored in a register.

Once the value of the keystroke dataword has been calculated it is transferred to data memory and the temporary register is cleared so that the next keystroke value may be calculated there. The following is a more detailed description of this process, starting in the middle of the start bit.

After eight more timer interrupts the middle of the first and most significant data bit is present at the BIRQ pin. So the software unmasks the BIRQ interrupt. If a zero is present at the BIRQ pin, the software calls the BIRQ interrupt service routine. The BIRQ interrupt is masked off and nothing is added to the value in the temporary register. After masking off the BIRQ interrupt the software returns to perform other operations. On the other hand, if the value at the BIRQ pin is a one, the BIRQ interrupt is masked off and the value 80 (Hex) (Most Significant Bit) is added to the value (initially 00 (Hex)) in the temporary register.

Likewise, after eight more timer interrupts the middle of the second serial databit is present at the BIRQ pin. So the BIRQ interrupt is unmasked and goes through the same procedure as above to decide if 40 (Hex) should be added to the value in the temporary register. Similarly this method will continue for the next six data bits. After the least significant bit has been evaluated, the value in the temporary register is moved to data memory. The reader should realize that this value can also be stored in another register if desired. Then the temporary register is cleared so that the next keystroke value can be recorded there.

The software continues to mask off the BIRQ interrupt until the end of the stopbit. At the end of the stopbit the timer is stopped and the time out value 03D7 (Hex) is loaded into the timer. The BIRQ interrupt is unmasked to wait for the next start bit zero.

This application has demonstrated how the timer and the BIRQ input/output pin can be used as a serial input. However there should be a note of warning that a production program should sample the serial input signal more than once every bit-time to guarantee valid data at a given baud rate. Furthermore, the software for the DP8344 must guarantee that the timer and/or BIRQ interrupts are not masked off by higher priority interrupts for too great a time; this could delay the sampling of the serial input signal for more than a bit-time, resulting in invalid data being read.

```
;------------------------------------------------------------------------
;:
;      This program will receive serial data using the BIRQ pin as
;   a serial input pin.
;      The timer will be used to determine when the middle of a bit
;   is present at the BIRQ pin.  Then the value of the bit is sampled
;   with the use of the BIRQ interrupt along with software to decide
;   if the bit is a one or a zero.  Then the software takes the
;   appropriate action for each case.
;      In this program all keystoke values are stored in consecutive
;   memory locations.
;
;   ******   National Semiconductor  Copyright 1988   *****
;------------------------------------------------------------------------
        .input          "stdequ.hdr"

CODE:           .sect  x
initialization:
            exx    AA,AB,DI
            move   5Fh,DCR              ;set CPU-CLK equal to OCLK
            move   02h,IBR              ;set up interrupts
            exx    MA,MB,DI
            move   0E7h,ICR             ;unmask timer and BIR interrupts
            move   10,IWLO              ;clear IW
            move   00,IWHI
            move   0,IX                 ;clear IX
            move   0,GP0                ;clear temporary registers
            move   0,GP1
            move   0,GP2
            move   0,GP3
            move   0,GP4
            move   0,GP5
            move   GP5,IZLO             ;load IZ with
            move   1,GP6                ;base address in data memory
            move   GP5,IZHI             ;for bit constant values
            move   80h,GP7              ;storing constants for
            move   GP7,[IZ+2]           ;  the most significant bit
            move   40h,GP7
            move   GP7,[IZ+3]           ; bit 6
            move   20h,GP7
            move   GP7,[IZ+4]           ; bit 5
            move   10h,GP7
            move   GP7,[IZ+5]           ; bit 4
            move   08h,GP7
            move   GP7,[IZ+6]           ; bit 3
            move   04h,GP7
            move   GP7,[IZ+7]           ; bit 2
            move   02h,GP7
            move   GP7,[IZ+8]           ; bit 1
            move   01h,GP7
            move   GP7,[IZ+9]           ; the least significant bit
            move   0D7h,GP7
            move   GP7,TRL              ;load timer's holding register
            move   03h,GP7              ;with count down value
```

TL/F/10450-7

FIGURE 4

```
                  move    GP7,TRH
                  move    61h,ACR              ;load timer
                          ;END of initialization
back:
                  cmp     GP0,0
                  jz      back                 ;waiting for BIRQ interrupt
                  cmp     GP0,0
                  jz      back                 ;protection
                  cmp     GP1,0                ;Is this a true start bit?
                  jnz     next_1
                  move    0,GP2                ;NO, then clear GP2
                  jmp     next_2
next_1:
                  move    0EFh,ICR             ;mask off the BIRQ interrupt
                  move    GP1,GP4
                  move    [IZ+A],GP4           ;load value of present bit
                  adda    GP5,GP5              ;add value of present bit
                  cmp     GP1,9                ;Is this bit 0?
                  jnz     next_2
                  move    GP5,[IW+]            ;YES, store byte of imformation
                  move    0,GP5                ;clear temporary registers
                  move    0,GP6
next_2:
                  move    0,GP0
                  ljmp    back


          ;Timer Interrupt Service Routine
          ;****************************
          ;
tm:
                  or      80h,CCR              ;clear [TO] flag
                  cmp     GP6,0                ;Is GP6 = 0?
                  jnz     next_10
                  add     1,GP3                ;loop until the stop
                  cmp     GP3,14h              ;bit has passed
                  jnz     next_11
                  move    60h,ACR              ;stop and load timer
                  move    0,GP1                ;clear GP1
                  move    0,GP2                ;clear GP2
                  move    0,GP3                ;clear GP3
                  move    1,GP6                ;set GP6 = 1
                  move    0E7h,ICR             ;unmask the BIRQ interrupt
next_11:
                  ret     RI
next_10:
                  cmp     GP1,0                ;Is this the start bit?
                  jnz     next_12
                  add     1,GP3                ;YES, add one to GP3
                  cmp     GP3,4                ;Is it the middle of the
                  jnz     next_13              ;start bit?
                  move    1,GP0                ;YES, set GP0 = 1
                  move    0,GP3                ;clear GP3
                  move    0E7h,ICR             ;unmask BIRQ interrupt
next_13:
                  ret     RI
```

TL/F/10450-8

FIGURE 4 (Continued)

```
next_12:
              add     1,GP3
              cmp     GP3,08h          ;Is it the middle of a
              jnz     next_14          ;data bit?
              move    1,GP0            ;YES, set GP0 = 1
              add     1,GP1            ;update which bit it is
              move    0,GP3            ;clear GP3
              move    0E7h,ICR         ;unmask the BIRQ interrupt
next_14:
              ret     RI


              ;BIRQ Interrupt Service Routine
              ;****************************
              ;
bq:
              move    0EFh,ICR         ;mask off the BIRQ interrupt
              move    0,GP0            ;clear GP0
              cmp     GP1,0            ;Is this the start bit?
              jnz     next_20
              cmp     GP2,0            ;YES, is this the first
              jnz     next_21          ;indication?
              move    0A0h,ACR         ;YES, start the timer
              move    1,GP2            ;set GP2 =1
              ret     RI
next_21:
              move    1,GP1            ;set GP1 = 1
next_20:
              cmp     GP1,9
              jnz     next_22
              move    GP5,[IW+]
              move    0,GP5
              move    0,GP6
next_22:
              ret     RI

CODE:
              .sect   ax
              .org    210h
              ljmp    bq
              .org    214h
              ljmp    tm
       .END
```

TL/F/10450-9

FIGURE 4 (Continued)

2
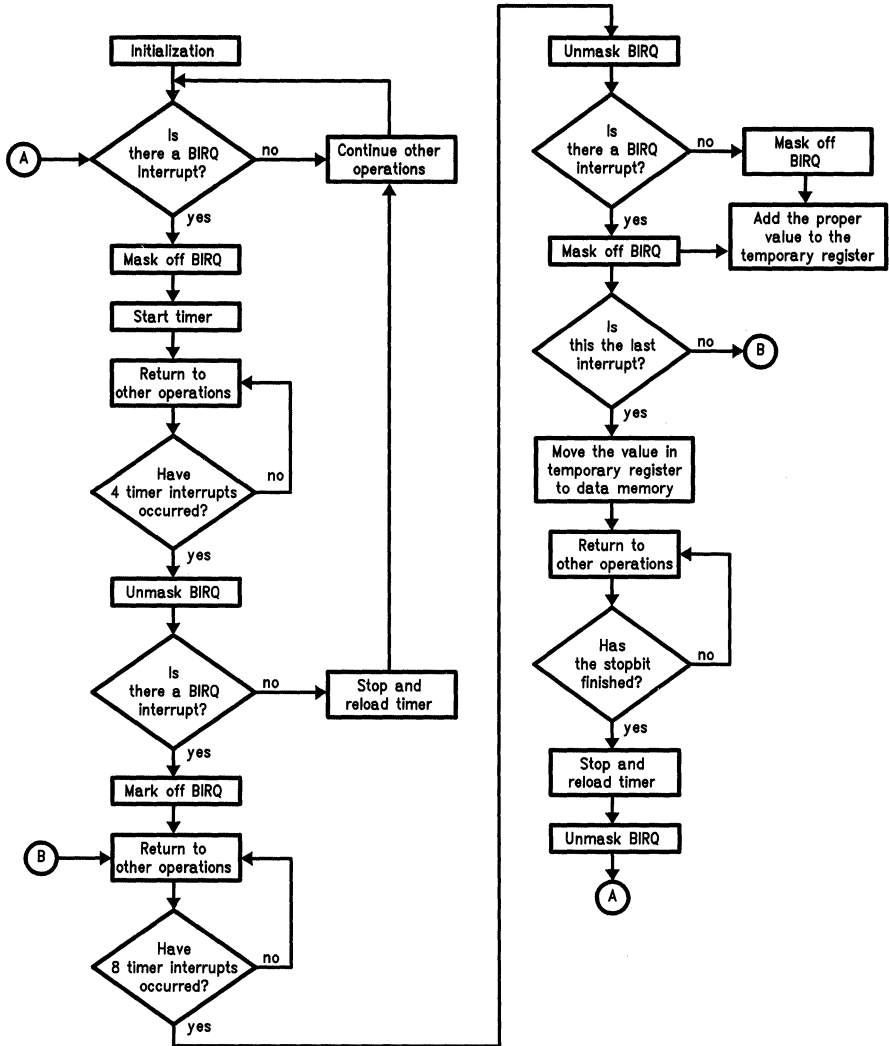
FIGURE 5. A Flow Chart of the Basic Application of the DP8344 as a Serial Input Keyboard

TL/F/10450–10

```
;-----------------------------------------------------------------------
;
;       This is the very simple cursor program for the BCP.
;       The timer along with the software toggles the state of the
;   cursor every 200msec.  The state of the cursor is stored in data
;   memory.
;
;
;-----------------------------------------------------------------------
        .input      "stdequ.hdr"

CODE:               .sect   x
initialization:
                exx         AA,AB,DI
                move    5Fh,DCR             ;set CPU-CLK equal to OCLK
                move    01,IBR              ;set up interrupt
                exx         MA,MB,DI
                move    0EFh,ICR            ;unmasktimer interrupt
                move    0,GP0               ;clear temperary register
                move    0,GP1               ;clear temperary register
                move    0,GP2               ;clear temperary register
                move    GP2,IZLO            ;clear IZ
                move    GP2,IZHI
                move    GP2,[IZ+0]          ;clear data memory location
                move    5Ch,GP2
                move    GP2,TRH             ;load high bit of the timer
                move    23h,GP2
                move    GP2,TRL             ;load low bit of the timer
                move    0C1h,ACR            ;load and start timer
                                ;END of initialization
loop:
                jmp             loop                ;wait for timer interrupt

        ;Timer Interrupt Service Routine
        ;*****************************
dest:
                or      80h,CCR             ;clear [TO] flag
                add     1,GP0
                cmp     GP0,0Ah             ;Has 200msec passed?
                rnz     R                       ;NO, return with interrupts on
                move    0,GP0               ;YES,
                cmp     GP1,0               ;Toggle
                jnz     next                ;   the
                move    1,GP1               ;       value
                jmp     send                ;           of
next
                move    0,GP1               ;               the
send:
                move    GP1,[IZ+0]          ;                   cursor
                ret     RI                  ;return with interrupts enabled

CODE:               .sect       ax
                    .org        114h
                    ljmp        dest
            .END
```

TL/F/10450–11

FIGURE 6

BLINK THE CURSOR

Introduction

Blinking the cursor is performed on virtually all computers. With its powerful CPU and programmable timer, the DP8344 can easily implement this basic function without any additional components.

In this application the timer along with a small amount of software will turn the cursor on and off at a periodic rate. The following is a description of the way the timer is programmed and the DP8344's software used to implement this function. (Refer to *Figure 6* for the actual BCP code for this application.)

Description

The following is one of many ways to perform the blinking cursor operation.

First, timer must be programmed to operate in the desired configuration. For this application the timer is pre-configured to divide the CPU clock by sixteen with the CPU clock set equal to the oscillator clock at 18.8696 MHz (i.e., [CCS] = 0). The timer interrupt is unmasked and enabled. The time out value of 5C23 (Hex) is loaded into the timer's holding register via {TRL} and {TRH}.

After the timer is programmed properly, the timer is loaded and started by writing to {ACR}. Once the timer is loaded

and started, it will continually cycle through the time out value of 5C23 (Hex), which corresponds to 20 ms. When the timer counts down to zero, it will set the timer interrupt and the BCP's software is programmed to call a timer interrupt service routine. The timer interrupt service routine updates and stores the state of the cursor in data memory.

State of Cursor: 0 $\rightarrow$ Cursor is OFF
                  1 $\rightarrow$ Cursor is ON

After ten timer interrupts, the service routine will toggle the state of the cursor. Thus, the cursor will blink 2.5 times a second.

Blinking the cursor every 200 ms is not a stringent requirement. As a result the 5250 protocol's 45 μs delays may be interleaved with the 20 ms time outs. This once again demonstrates that the flexibility of the timer can enhance the performance of two functions at the same time.

REAL-TIME CLOCK

Introduction

An added feature on most personal computers is a real-time clock. The clock is used to provide the time, the day, the month and the year. With the BCP's programmable timer and powerful CPU, this clock function can be performed by the DP8344 without any additional components.

In this application the timer along with a small amount of DP8344 software keeps track of the time. Software also allows the user to set the initial time, then the DP8344's timer and software takes over and accurately keeps track of the time. The following is a description of the way the timer is programmed and the DP8344's software used to implement a real-time clock. (Refer to *Figure 7* for the actual BCP code for this application.)

Description

The following describes one basic way to implement a real-time clock using the BCP.

First, the timer must be programmed to operate in the desired configuration. For this application the timer is pre-configured to divide the CPU clock by sixteen with the CPU clock set equal to the oscillator clock at 18.8696 MHz (i.e., [CCS] = 0). The timer interrupt is unmasked and enabled. The countdown-value of 5C23 (Hex) is loaded into the timer's holding register via {TRL} and {TRH}.

After the timer is programmed properly the timer is loaded and started by writing to {ACR}. Once the timer is loaded and started, it should remain on and continually cycle through the time out value of 5C23 (Hex), which corresponds to 20 ms exactly. When the timer counts down to

zero, it sets the timer interrupt and the CPU is programmed to call a timer interrupt service routine.

The timer interrupt service routine is very basic. The number of seconds, minutes, hours, days and years are all recorded in separate data memory locations. The service routine will add one to the seconds value after fifty timer interrupts. After sixty seconds, the seconds value is reset to zero and a one is added to the minutes value. After sixty minutes, the minutes value is reset to zero and a one is added to the hours value. Likewise the number of hours, days and years are recorded in a similar manner.

As mentioned in the introduction, in order to set the present date and time after powering up requires software which allows the user to define the present time and date. This software would be remote processor software, not DP8344 software. This remote processor's software should allow the user to enter the present time and date, then this software must transform the entered time and data into data which can be transferred to the real-time clock's data memory locations. This starts the clock at the entered time, and the timer and DP8344 software will be responsible for updating the clock accurately.

The reader may desire a clock which records time in increments as small as a hundredth of a second. In this case the timer should be programmed to count down 10 ms time outs, and another data memory location must be used to record the number of these hundredths of a second.

For the application of a real-time clock, the timer cannot interleave two timing values as in the 5250 Protocol application. The timer must be pre-configured and allowed to run without interruption. Otherwise, timing errors will occur and the clock will not record time accurately.

However, the reader may notice that the BLINK THE CURSOR application uses the same time out value (i.e., 5C23 (Hex)) as the real-time clock. This demonstrates how the BCP can be programmed to use one countdown-value to implement two desirable functions without effecting the performance of either operation.

A final warning to the reader. The oscillator clock must be extremely accurate for this application. For the program provided, and error of 0.0002 MHz in the oscillator clock (OCLK = 18.8696 MHz) will result in an error of 0.916 seconds a day or 5 minutes 34 seconds per year. The best way to prevent timing problems is to accurately measure the oscillator clock frequency first, then calculate and implement all time out values based on that measurement.

```
;------------------------------------------------------------------
;
;       This is the third version of the real-time clock.
;       This version like the second uses the timer interrupt to make
;   service calls, instead of polling the TO flag (bit 7 of CCR)
;   to see when the timer has counted to zero.  The timer is pre-
;   configured to use CPU-CLK/16 and the CPU-CLK is set equal to
;   OCLK (oscillation clock, in this case 18.8696 MHz).  The countdown
;   value is 5C23 Hex, which corresponds to 20 ms.
;       The IW register is incremented every 20 ms interrupt until
;   it contains 32(Hex) or 50(Dec), which corresponds to every second
;   exactly.  Then the IX register is incremented.
;       Unlike version 2 this version uses data memory to store and
;   record the time that has elapsed.  The following table gives
;   the memory locations of the stored time values.
;
;           value           |   memory location (HEX)
;           -----------------|------------------------
;           seconds          |       00 40
;           minutes          |       00 30
;           hours            |       00 20
;           days             |       00 10
;           years            |       00 00   (not implimented in program)
;------------------------------------------------------------------
        .input          "stdequ.hdr"

CODE:                   .sect   x
initialization:
        exx     AA,AB,DI
        move    5Fh,DCR                 ;set CPU-CLK equal to OCLK
        move    01,IBR                  ;set up interrupt
        exx     MA,MB,DI
        move    0EFh,ICR                ;unmask timer interrrupt
        move    0,IW                    ;clear IW
        move    0,IX                    ;clear IX
        move    0,GP5
        move    GP5,IYLO                ;clear IY
        move    GP5,IYHI
        move    GP5,IZLO                ;clear IZ
        move    GP5,IZHI
        move    GP5,[IZ+0]              ;clear year
        move    GP5,[IZ+10h]            ;clear days
        move    GP5,[IZ+20h]            ;clear hours
        move    GP5,[IZ+30h]            ;clear minutes
        move    GP5,[IZ+40h]            ;clear seconds
        move    1,GP6
        move    5Ch,GP5
        move    GP5,TRH                 ;load high byte of the time out value
        move    21h,GP5
        move    GP5,TRL                 ;load low byte of the time out value
        move    40h,ACR                 ;load timer from the holding reg.
        move    81h,ACR                 ;start timer
                ;END of initialization
loop:
```

2

TL/F/10450–12

FIGURE 7

```
            jmp     loop

;Timer interrupt service routine
;****************************
;
dest:
            or      80h,CCR         ;clear [TO] flag
            add     1,IWLO          ;increment IW
            cmp     IWLO,32h        ;does IW = 50 decimal
            rnz     RI              ;NO, then return with interrupt on
            move    0,IWLO          ;clear IW
            move    [IZ+40h],IXLO
            add     1,IXLO          ;YES, then increment IX
            cmp     IXLO,3Ch        ;does seconds = 60
            jnz     next_1
            move    0,IXLO          ;YES, then clear seconds
next_1:     move    IXLO,[IZ+40h]   ;move seconds to data memory
            rnz     RI
            move    [IZ+30h],IXLO
            add     1,IXLO          ;increment minutes
            cmp     IXLO,3Ch        ;does minutes = 60
            jnz     next_2
            move    0,IXLO          ;YES, then clear minutes
next_2:     move    IXLO,[IZ+30h]   ;move minutes to data memory
            move    0,IXLO
            rnz     RI
            move    [IZ+20h],IXLO
            add     1,IXLO          ;increment hours
            cmp     IXLO,18h        ;does hours = 24
            jnz     next_3
            move    0,IXLO          ;YES, then clear hours
next_3:     move    IXLO,[IZ+20h]   ;move hours to data memory
            move    0,IXLO
            rnz     RI
            move    [IZ+10h],IXLO
            add     1,IXLO          ;increment days
            move    IXLO,[IZ+10h]
            move    0,IXLO
            ret     RI

CODE:       .sect   ax
            .org    114h
            ljmp    dest
      .END
```

TL/F/10450–13

FIGURE 7 (Continued)

MPA — A Multi-Protocol Terminal Emulation Adapter Using the DP8344

Thomas Norcross
Paul J. Patchen
Thomas J. Quigley
Tim Short
Debra Worsley

Table of Contents

SECTION ONE — INTRODUCTION

About This Application Note and Technical Reference

The purpose of this application note is to provide a complete description of the Multi-Protocol Adapter (MPA™), a hardware and software design solution emulating basic 3270 and 5250 terminal emulation products in an IBM® PC® environment. This document discusses the system support hardware and complete link level firmware to achieve 3270 CUT, DFT, 3287, 3299, and 5250 emulation with the BCP™. The document is divided into six sections and three sets of appendices.

Section 1. Introduction provides a summary of each section and Appendices along with a checklist of items included in the MPA Design/Evaluation Kit. This section also describes the Multi-Protocol Adapter, DP8344 Biphase Communications Processor, and Advanced Peripherals Products.

Section 2. System Overview describes the 3270 environment, 5250 environment, and terminal emulation.

This section also describes the DCA® and IBM system architectures and discusses the MPA system organization.

Section 3. Hardware Architecture discusses the MPA architecture including a description of the BCP core, PC interface, Front-end interface, and miscellaneous support circuitry.

Section 4. Software Architecture discusses the Kernel, coax task, twinax task, and interrupt structure. Included in this section is an in depth discussion of the IRMA™, IBM, and Smart Alec™ interfaces. This section also provides a description of the Loader and Selftest facilities.

Section 5. Operation describes the system requirements, installation instructions, and steps for running the 3270 and 5250 emulation.

Section 6. Development Environment describes the tools used in developing the MPA including abel™, the CT-104 Demonstration/Development kit, and logic analyzer application.

Appendix A. Hardware Reference provides the complete MPA schematic, assembly drawing, and PAL® equations.

Appendix B. Timing Analysis discusses the timing of the MPA system, PC-AT/XT, and bus contention.

Appendix C. References is a list of reference materials.

Multi-Protocol Adapter

The Multi-Protocol Adapter (MPA) is a complete design solution for IBM 3270, 3299, and 5250 connectivity products. The MPA is intended to be a design example for customers to use in developing their own products using the Biphase

2

Communications Processor (BCP). The BCP is a "system on a chip" designed by National Semiconductor's Integrated Systems Group to specifically address the IBM connectivity market place. Built on the tradition of the DP8340/41 3270 receiver/transmitter pair, the BCP takes the state of the art in IBM communications a step further. The MPA provides the system support hardware and complete link level firmware to achieve 3270 CUT, DFT, 3287, 3299 and 5250 emulation with the BCP. Now National provides a total IBM communications solution; the MPA is a blueprint for terminal emulation designs and the basis for many BCP applications.

DP8344 Biphase Communications Processor (BCP)

The DP8344 BCP is a communications processor designed to efficiently process IBM 3270, 3299 and 5250 communications protocols. A general purpose 8-bit protocol is also supported.

The BCP integrates a 20 MHz, 8-bit, Harvard architecture, RISC processor and a flexible, software-configurable transceiver on the same low power microCMOS chip. The transceiver is capable of operating without significant processor interaction, releasing processor power for other tasks. Fast, flexible interrupt and subroutine capabilities with on-chip stacks make the power readily available.

The transceiver is mapped into the processor's register space, communicating with the processor via an asynchronous interface which enables both sections of the chip to run from different clock sources. The transmitter and receiver run at the same basic clock frequency although the receiver extracts a clock from the incoming data stream to ensure timing accuracy.

The BCP is designed to stand alone and is capable of implementing a complete communications interface, using the processor's spare power to control the complete system. Alternatively, the BCP can be interfaced to another processor with an on-chip interface controller arbitrating access to data memory. Access to program memory is also possible, providing the ability to softload BCP code.

A simple line interface connects the BCP to the communications media. The receiver includes an on-chip analog comparator suitable for use in transformer-coupled environments, and a TTL-level serial input for applications where an external comparator is preferred.

A typical system is shown in *Figure 1-1*. Both coax and twin-ax line interfaces are shown, as well as an example of the (optional) remote processor interface.

For a detailed discussion on the BCP refer to the DP8344 Biphase Communications Processor data sheet. For more information on the BCP, contact Integrated Systems Group Marketing, M/S 16-197, 2900 Semiconductor Drive, P.O. Box 58090, Santa Clara, CA 95052-8090.
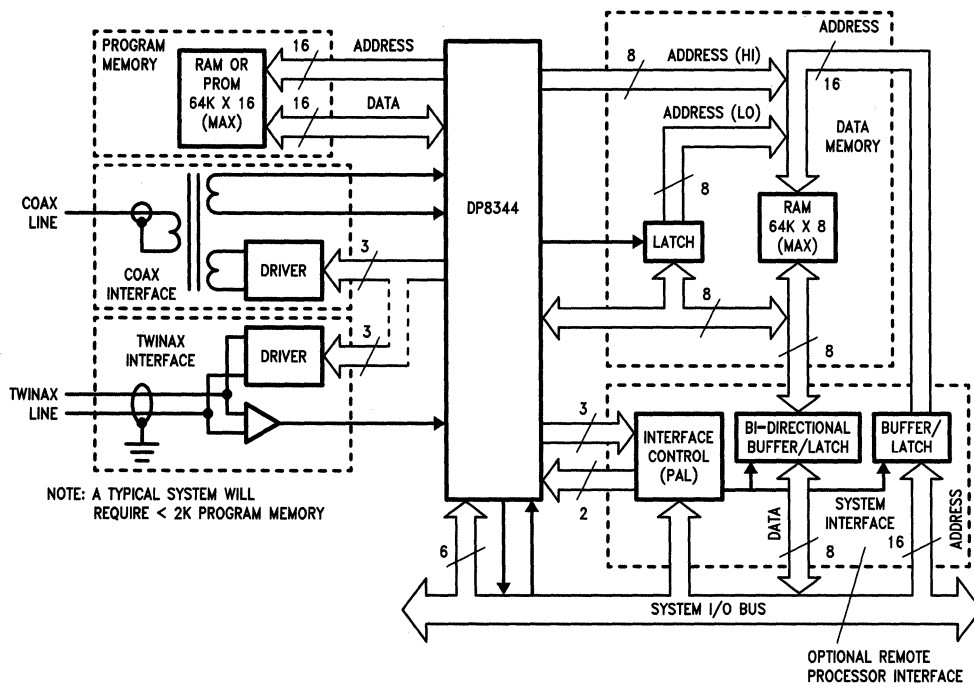
FIGURE 1-1. Block Diagram of Typical BCP System

TL/F/10488–1

SECTION TWO — SYSTEM OVERVIEW

The MPA addresses a systems market that is driven by the large installed base of IBM systems throughout the world. The IBM plug compatible peripheral and terminal emulation markets are growing along with the success of IBM in the overall computer market place. The originally proprietary architecture of IBM peripherals and the subsequent vague and confusing Product Attachment Interface documents (PAIs) have kept the attachment technology elusive. The IBM communications system in general is not well understood. The desire of customers and systems vendors to achieve more attachment options, however, is significant.

IBM 3270 and 5250 Environments

The study of IBM communications fills many volumes. The intent of this discussion is not to describe it fully, but to highlight the areas of IBM communications that the BCP and MPA address. Specifically, these areas are the controller/peripheral links that use the 3270/3299 and 5250 data streams. These links are found in 370 class mainframe networks and the smaller, mid-range System/3x and AS400 lines.

The 3270 communications sub-system was developed for 370 class mainframes as demand for terminal support began to outstrip batch job entry modes. These systems had large scale networking needs and often needed to support thousands of terminals and printers. The original systems were linked together through dedicated telephony lines using Binary Synchronous Communications (BSC) serial protocol. The 5250 communications system was developed originally for the Series 3 and became widely used on the System/34. The System/34 was a small, office environment processor with limited networking and terminal support capabilities. Typical System/34 installations supported up to 16 terminals and printers. The System/36 replaced the System/34 in 1984. The System/38 is a mid-range processor that can rival the 4300 series (small 370 class) mainframes in processing power. The System/36 and 38 machines now have greatly enhanced networking facilities, and can support up to 256 local terminals. The 370 class and System/3x machines have grown closer together through the advent of SNA (Systems Network Architecture). SNA allows both systems to function together in an integrated network. The AS/400 combines the power of the large system /38 with the ease of use of the smaller system /36 machines.

The 3270 and 5250 communications systems evolved at a time when hardware design constraints were very different than today. Microprocessors and 1 Mb DRAMs were not available. Memory in general was very expensive. Telecommunications channel sharing between multiple peripherals was imperative. Even so, fast screen updates and keystroke handling were necessary. The 3270 and 5250 data stream architectures were developed to address specific design goals within IBM's overall network communications system. The controller sub-system where they were implemented has proved adaptable to new directions in SNA and the migration of processor power out into workstations.

The 3270 and 5250 controller sub-systems split the peripheral support tasks into two sections: screen with keyboard, and host communications interface (see *Figure 2-1*). The controller architectures can be thought of as having integral screen buffers and keyboards for each of their associated terminals with the caveat that screens and keyboards must be accessed through a secondary, high speed serial link. Since the controller views the terminal's screen buffer as its own, the controller does not maintain a copy of the information on that screen. The processing capability of some terminals is severely limited; the early terminals were state machines designed to handle the specific data stream. With the advent of SNA and APPC, (Advanced Peer to Peer Communications) the intelligence in some peripherals has become significant. The data streams have essentially remained the same, with hierarchically structured protocols built upon them. SNA and these higher protocols will be discussed later.

Separating the screen buffer and keyboard from the intelligence to handle the terminal addressed several design goals. Since the terminal needs screen memory to regenerate its CRT, the "regen" buffer logically resides in the terminal. The controller need not duplicate expensive memory by maintaining another screen copy. The data stream architectures implemented with high speed serial links between the controller and terminal allow fast keystroke echoing. It also allows fast, single screen updates, giving the appearance of good system performance. The terminal screen maintenance philosophy developed with these architectures lends itself well to the batch processing mode that traditionally was IBM's strong suit. The terminal system is optimized for single screen presentation with highly structured field oriented screens. Data entry applications common in business computing are well suited to this. Essentially, the architecture places field attribute and rudimentary error checking in the controller, so that most keystrokes can pass from the terminal to the controller and back to the screen very quickly without host CPU intervention. Only when particular keystrokes are sent (AID keys) does the controller read the contents of the screen fields and present the host with the screen data.

3270 Data Stream Architecture

The 3270 communications system, as discussed above, is a single logical function separated into two physical pieces of hardware connected by a protocol implemented on a high speed serial link. The terminal hardware has the screen buffers and keyboard, magnetic slot reader, light pen, etc. (i.e., all the user interface mechanisms). The controller has a communications link to the host CPU or network and the processing power to administrate the terminal functions. Controllers typically support multiple terminals and essentially concentrate the terminal traffic onto the host communications channel. The controller has a secondary communications system that implements the 3270 data stream protocol over coaxial cable at 2.3587 Mb/s. Each peripheral connected to the controller has its own coax port. The coax lengths may be up to 5000 feet. The protocol is controller initiated, poll/response type.

The serial protocol organizes data into discrete groups of 12 bits or frames. Biphase (Manchester II) encoding is used to impress the data frames onto the transmission medium. Biphase data have embedded clock information denoted as mid-bit transitions. Frames may be concatenated to form packets of commands and/or data. All transmissions begin with a line quiesce sequence of five biphase one bits followed by a three-bit time line violation. The first bit of all frames is called the sync bit and is always a logic one. The sync bit follows the line violation and precedes all successive frames. Each frame includes a parity bit that establishes even parity over the 12-bit frame. Each transmission from the controller elicits a response of data or status from the device. The response time requirements are such that a device must begin its response within 5.5 μs after reception

2
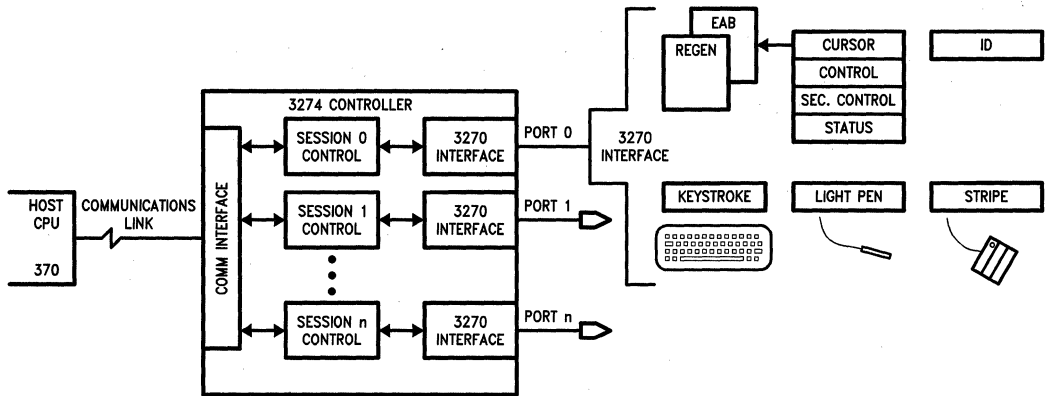
TL/F/10488–2

FIGURE 2-1. 3270 Communications Systems

of the controller transmission. Simple reception of a correct packet is acknowledged by transmission of "TTAR", or transmission turn around/auto response. The controller initiated, poll/response format protocol addresses multiple logical devices inside the peripheral through a three- or four-bit command modifier. The different logical devices decode the remaining bits as their command sets. Commands to the base or keyboard are decoded as shown in Table 2-1.

The 3299 variant on the 3270 data stream uses an additional eight-bit address field to address up to 8 more 3270 devices with the same coax. Since coax installations are point-to-point between controller and peripheral, cabling costs motivated the introduction of 3299 multiplexer/demultiplexers. Using the extended address field, eight devices can be connected via one coax cable between the controller and the multiplexer.

Basic terminals have a structure as shown in *Figure 2-2*. The EAB (Extended Attribute Buffer) is a shadow of the regen buffer; each location in the regen has a corresponding location in the EAB. The EAB is a separately addressable device with an address modifier of 7h. The EAB bytes are used to provide extra screen control information. In the 3270 world, the screen and field attributes that the controller uses to format and restrict access to fields on the screen take up space in the screen. The attribute characters appear as blanks and cannot be used for displayable characters at the same time. Since the number of permutations of the 8-bit character byte is limited to 256, the number of attributes is limited by the size of the displayable character set. The EAB provides a method to enhance screen control, with color for instance, without losing character space. The EAB contains both character attributes, that correspond to characters in the regen buffer, and field attributes that correspond to attributes in the regen.

Status developed in the terminal, such as keystrokes or errors, are reported in the poll/response mechanism. A POLL command to the base device with keyboard status elicits the keystroke in 5.5 μs. The controller then sends a POLL/ACK command to clear the key status. The terminal should respond with "clean" status, i.e., TTAR. Controllers poll frequently to assure that status updates are quick. Outstanding status is reported in the poll response and in some cases is handled directly by POLL command modifiers in the command. Keystrokes are the most common status and hence are acknowledged by the POLL/ACK command. Status reported in the status register must be read and acknowledged independently of the polling mechanism.

The SEARCH FORWARD, SEARCH BACKWARD, INSERT and CLEAR commands require the terminal to process in the foreground while responding with "BUSY" status to the controller. Processing these commands requires substantially more time than the others, and hence are allowed to proceed without real-time response restrictions.

An interesting feature found in terminals and printers is the START OP command. Originally, this command was used only by controllers and printers to begin print jobs. Printers have specific areas within their buffers that are reserved for higher level commands from the controller. These higher level protocols started as formatting commands and extra printer feature control. With the advent of SNA and Distributed Function Devices, this concept is used in terminals to pass SNA command blocks to multiple NAUs (Network Addressable Units) within the terminal. These NAUs are complete terminals, or peers, not just simple user interface devices.

TABLE 2-1. 3270 Data Stream Command Set

Command	Value	Description
READ TYPE: TO BASE—device address 0 or 1		
POLL	01h	respond with status
READ DATA	03h	respond with data at address counter
READ ADDRESS COUNTER HI	05h	respond with address counter high byte
READ ADDRESS COUNTER LO	15h	respond with address counter low byte
READ TERMINAL ID	09h	respond with terminal type
POLL/ACK	11h	special status acknowledgment poll
READ STATUS	0Dh	respond with special status
READ MULTIPLE	0Bh	respond with up to 4 or 32 bytes
READ EXTENDED ID	07h	respond with 4 byte ID
**WRITE TYPE: TO BASE—device address 0 or 1		
RESET	02h	POR device
*CLEAR	06h	clear regen buffer to nulls
WRITE DATA	0Ch	load regen buffer with data
LOAD ADDRESS COUNTER HI	04h	load address counter high byte
LOAD ADDRESS COUNTER LO	14h	load address counter low byte
START OPERATION	08h	begin execution of higher level command
LOAD SECONDARY CONTROL	1Ah	load additional control byte
*INSERT BYTE	0Eh	insert byte at address counter
*SEARCH FORWARD	10h	search forward in buffer until match
*SEARCH BACKWARD	12h	search back in buffer until match
LOAD MASK	16h	load mask used in Searches, CLEAR

*denotes foregound task

**ALL WRITE type commands elicit TTAR upon clean reception.

2
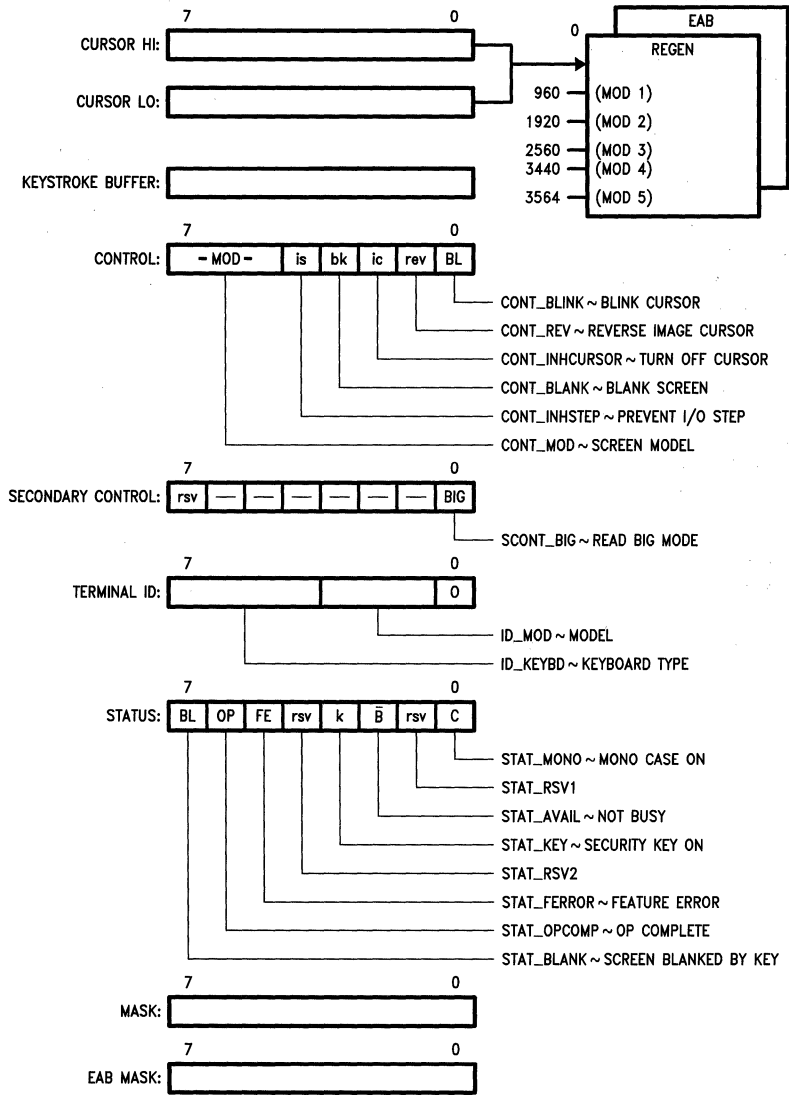
FIGURE 2-2. 3278/79 Interface Terminal Architecture

TL/F/10488-3

As large mainframes proliferated, so did the need to off-load terminal support from the emerging 370 class mainframe. The need to "network" both remotely and locally was becoming apparent. In addition, the need to separate display and printer interface tasks from applications was sorely felt. The system developed by IBM eventually became Systems Network Architecture (SNA). The 370 class machines use secondary processors, or "front-ends" to handle the networking aspect of large scale systems and they in turn use terminal and printer controllers to interface locally with the user interface devices. The controllers handle the device specific tasks associated with interfacing to different printers and displays. The front-ends handle connecting the routes from terminals or printers to applications on the mainframe. A session is a logical entity split into two halves; the application half and the terminal half, and connected by a virtual circuit. Virtual circuits can be set up and torn down by the system between applications and terminals easily, and the location of the specific terminal or printer is not important. NAUs are merely devices that can be addressed directly within the global network. Setting up multiple NAUs within a terminal allows all sorts of gateway opportunities, multi-display workstations, combination terminal/printers, and other things.

DFD devices can support up to five separate NAUs using a basic 3270 port. Using 3299 addressing allows eight sessions for each DFD device, or 40 possible NAUs per coax. By layering protocols over the basic 3270/3299 data stream, the controller can distribute more of the SNA processing to intelligent devices that replace terminals. APPC will allow more and more functions to be shared by NAUs that act as "peers" in the network.

5250 Data Stream Architecture

The 5250 data stream architecture has many similarities to 3270, although it is different in many ways. The primary difference is the multi-drop nature of 5250. Up to seven devices may be "daisy chained" together on the same twinax cable. Twinax is a very bulky, shielded, twisted pair as opposed to the RG/62U coax used in 3270.

The 5250 bit stream used between the host control units and stations on the twinax line consists of three separate parts; a bit synchronization pattern, a frame synchronization pattern, and one or more command or data frames. The bit sync pattern is typically five one bit cells. This pattern serves to charge the distributed capacitance of the transmission line in preparation for data transmission and to synchronize receivers on the line to the bit stream. Following the bit sync or line quiesce pattern is the frame sync or line violation. This is a violation of the biphase, NRZI data mid bit transition rule. A positive going half bit, 1.5 times normal duration, followed by a negative going signal, again 1.5 times normal width, allows the receiving circuitry to establish frame sync.

Frames are 16 bits in length and begin with a sync or start bit that is always a 1. The next 8 bits comprise the command or data frame, followed by the station address field of three bits, a parity bit establishing even parity over the start, data and address fields, and ending with a minimum of three fill bits (fill bits are always zero). A message consists of a bit sync, frame sync, and some number of frames up to 256 in total. A variable amount of inter-frame fill bits may be used to control the pacing of the data flow. The SET MODE command from the host controller sets the number of bytes of zero fill sent by attached devices between data frames.

Message routing is accomplished through use of the three-bit address field and some basic protocol rules. There is a maximum of eight devices on a given twinax line. One device is designated the controller or host, the remaining seven are slave devices. All communication on the twinax line is host initiated and half duplex. Each of the seven devices is assigned a unique station address from zero to six; address seven is used for an End Of Message delimiter, or EOM. The first or only frame of a message from controller to device must contain the address of the device. Succeeding frames do not have to contain the same address for the original device to remain selected. The last frame must contain the EOM delimiter. For responses from the device to the controller, the responding device places its own address in the address field in frames 1 to (n−1), where n ≤256, and places the EOM delimiter in the address field of frame n. However, if the response to the controller is only one frame, the EOM delimiter is used. The controller assumes that the responding device was the one addressed in the initiating command.

Responses to the host must begin in 60 μs ±20 μs, although some specifications state a 45 μs ±15 μs response time. In practice, controllers do not change their time out values per device type so that anywhere from 30 μs to 60 μs response times are appropriate.

The 5250 terminal organization is set up such that there are multiple logical devices within the terminal as in 3270. These devices are addressed through a command modifier field in the command frame. The command set for the base logical devices is shown in Table 2-2. Note that except for POLL and ACTIVATE commands, all commands are executed in foreground by the terminals. Commands are loaded on a queue for passing to the foreground while the terminal responds with "busy" status to the host. See *Figure 2-3* for the 5251 terminal architecture.

Personal computers are often used to emulate 3270 and 5250 terminals, and in fact, have hastened the arrival of APPC functions in both the 3270 and 5250 arenas. Basic CUT (Control Unit Terminal), i.e., non-DFD emulation is often accomplished by splitting the terminal functions into real-time chores and presentation services. The presentation services, such as video refresh and keyboard functions are handled by the PC, and real-time response generation, etc., by an adapter card. This is a somewhat expensive alternative to a "dumb" terminal. However, since PCs are becoming more and more powerful, their use as peers in SNA networks, as multiple NAUs, or multiple display sessions in 5250 is very promising. Although primitive in many ways, the 3270 and 5250 communications system's fast response times, unique serial protocols and processing overhead requirements have traditionally limited the confidence of third party developers in designing attachments. In addition, the high cost of many early solutions discouraged many would-be developers.

National Semiconductor opened the 3270 attachment market place to many third parties in 1980 with the release of the DP8340/41 protocol translation chip set. The chip set removed one of the major stumbling blocks to attachment designs, although formidable design challenges remained. Bit-slice or esoteric microcontrollers were still required to meet the fast response times specified by IBM. The difficulties and costs in designing interface circuitry for these solutions remained a problem. Other major communications protocols in use by IBM were not addressed by these or any

2

TABLE 2-2. 5250 Command Set			
READS	**WRITES**	**CONTROL**	**OPERATORS**
QUEUEABLE COMMAND			
READ DATA [1, 3] READ DEVICE [1, 3] READ IMMEDIATE [2, 3] READ LIMITS [1, 3] READ REGISTERS [1, 3] READ IBM [1, 3]	WRITE CONTROL DATA WRITE DATA AND LOAD CURSOR WRITE IMMEDIATE [2, 3] WRITE DATA [1, 3]	EOQ LOAD ADDR COUNTER LOAD CURSOR REG. LOAD REF. COUNTER RESET SET MODE	CLEAR INSERT CHAR. MOVE DATA SEARCH

RESPONDERS	**ACCEPTORS**
NON-QUEUEABLE COMMANDS	
POLL ACTIVATE READ	ACTIVATE WRITE

Note 1: Must be last command loaded onto queue. EOQ may follow.

Note 2: EOQ must follow.

Note 3: An ACTIVATE command required after this command.

Note 4: READ IBM is not a documented command in the IBM PAI. Respond with 4 zero bytes.

Note 5: WRITE DATA is documented as a printer command.
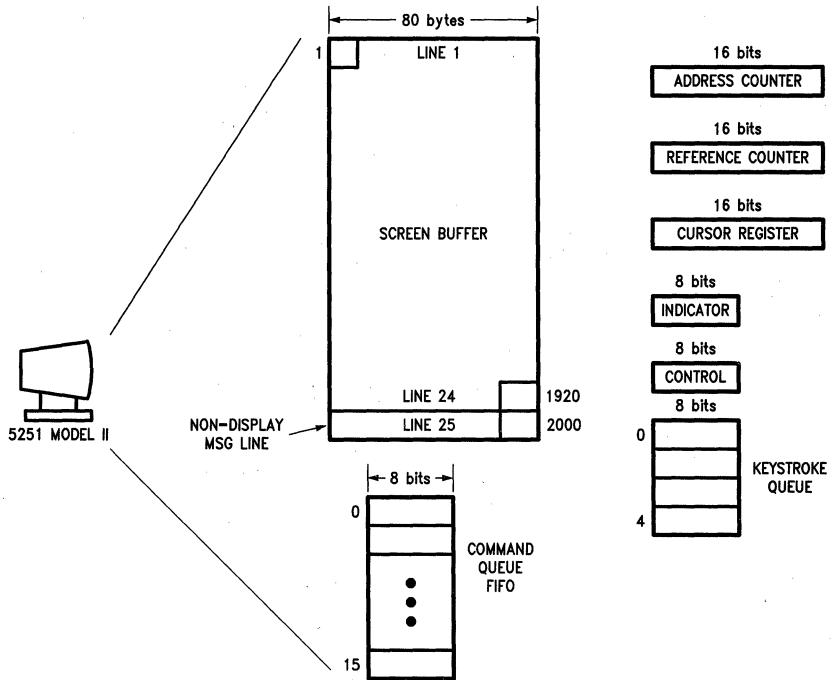
FIGURE 2-3. 5251 Terminal Architecture

TL/F/10488-4

other chips on the market. The BCP removed the major hardware design problems associated with connecting to IBM 3270, 5250, and 3299 communications systems. The BCP's combination of a powerful, highly optimized CPU coupled tightly with protocol translation and interface peripherals make it the preferred solution for a wide range of design goals.

Terminal Emulation

The terminal emulation market opened with Technical Analysis Corporation's IRMA product in 1982. The 3278/79 terminal emulator quickly became the industry standard, even as IBM and many others entered the market place. Technical Analysis Corporation merged with Digital Communications Associates in 1983. The 3270 emulation market is now dominated by DCA and IBM. IBM produced the first 5250 terminal emulator in 1984, although it was a severely limited product. The market opened up in 1985 with the release of products by AST Research, IDEAssociates, and DCA. DCA's Smart Alec was the first product to provide seven session support, address bidding, and a documented open architecture for third party interfacing. DCA's IRMA was released with a technical reference detailing their Decision Support Interface (DSI). This document along with the source code to E78 (their PC emulator software) allowed many companies to design micro to mainframe products using the DSI as the mainframe interface. IBM provides a technical reference for their 3278 Entry Level Emulator as well (see Appendix C for a complete list of references).

The proliferation of the IBM and DCA interfaces coupled with the availability of detailed technical information about them made those interfaces good choices for the MPA. The MPA system was designed to do two major functions: one is emulation of the DCA and IBM emulation products; the second is to provide a powerful, multi-protocol interface that will afford greater utilization of the DP8344. Specifically, the MPA emulates the hardware/firmware resident in PC add-in boards for 3270 and 5250 emulation products from DCA and IBM. To do this, we have constructed hardware and firmware that mimics the corresponding system components of the other emulators. The MPA system appears in every sense to be the board it is emulating, once it has been loaded and configured.

The DCA and IBM system organizations are similar. Each system is divided into two major functional groups: presentation services, and terminal emulation. The terminal emulation function resides entirely on the adapter hardware and maintains the screen buffers that belong to the host control unit. The terminal emulation function includes all real time responses and status generation necessary to appear as a true 5250 or 3270 device to the host controller. Presentation services carried out by the PC processor include fetching screen data from the adapter, translating it into displayable form, and providing the data to the PC's display adapter. In addition, the PC side presentation services collect keystrokes from the keyboard and present them to the adapter. The communication between the PC presentation handler and adapter emulation function consists mainly of status updates, keystrokes, and screen data.

DCA

The DCA products use an I/O mapped, 4 byte mailbox to pass information encoded in a <command>, [<argument>], [<argument>], [<argument>], and <status>, [response], [response], [response] format. Information flow is controlled through a Command/Attention

semaphore implemented in hardware. Both the Smart Alec (5250) and IRMA (3270) interfaces have command sets that include reading and writing the screen buffers maintained on the adapter boards, sending keystrokes, and passing display information such as cursor position and general screen modes. The interfaces are both used in a polled manner, although both are capable of generating interrupts to the PC processor.

Both Smart Alec and IRMA have Signetics 8x305 processors that run the terminal emulation functions and interface to the PC presentation services. The PC function initiates commands and status requests by writing the appropriate value into the mailbox and setting the Command semaphore. The semaphore is then polled for a change in state that signals completion of the command and valid response data is in the mailbox. The PC will poll for a specific amount of time before assuming a hardware malfunction has occurred. The 8x305 processors have no interrupt capabilities and handle all terminal emulation sub-tasks in a polled manner. The PC interface tasks are the lowest priority of all. The 8x305 may initiate information transfer to the PC by posting the Attention semaphore, and/or setting a PC interrupt, although this is not generally done. Both interfaces are implemented with 74LS670 dual-ported register files so that reads and writes from each processor are directed into separate register files.

Both of the DCA interfaces were designed for compatibility at the expense of interface through-put. The small I/O requirements and the fact that interrupts to the PC are not necessary allow the interfaces to install easily in most environments. The IRMA Decision Support Interface (DSI) utilizes eight I/O locations at 220h–227h. Smart Alec resides in locations 228h–22Fh. All screen data and status information must pass through these mailboxes with the semaphore mechanism. This makes repainting the entire screen very slow. Both IRMA and Smart Alec utilize different schemes to reduce the necessity of reading entire screen buffers often. IRMA maintains a screen image in PC memory that is used in conjunction with a complex algorithm to determine which lines of the screen to update. Smart Alec maintains a 16 entry FIFO queue that contains screen modification information encoded in start/end addresses. This information is processed to decide which screen locations should be updated.

IBM

The IBM system organization is, in general, very similar to the DCA systems. The major differences lie in the interface implementations. The IBM system utilizes RAM dual-ported between the PC processor and the adapter board processor to transfer screen data from the adapter. In addition, IBM does not use an interpreted command/response I/O interface. The IBM interface uses 12 I/O locations with individual bits defined in each register for direct status availability. The status bits consumed by the PC presentation services are cleared through a "write under mask" mechanism. Consumable bits are read by the PC and when written to, corresponding status bits are cleared by one bits in the value written. Reading a register of consumable bits and writing the value back out clears the bits set in the register. The interface can operate in a polled manner, although typically it is operated via interrupts. One register in the interface is dedicated to interrupt status (ISR—Interrupt Status Register, 2d0h) and when the PC is interrupted, the particular status change event is indicated in that register. Buffer

2

modifications are indicated through a status change in the I/O interface which also provides an indication of the block modified. The actual screen data is in 8k of dual-port RAM and may be read by the PC when a "Buffer-Being-Modified" flag is cleared. This type of interface affords the IBM products great speed advantages, although limits compatibility with other add-in PC boards.

Both the IBM and DCA systems present EBCDIC data to the PC presentation services for display. The presentation software must translate the EBCDIC codes into ASCII for PC display adapters. In addition, the screen attribute schemes for PCs and mainframe terminals differ greatly. The presentation services must provide the necessary display interface to emulate the "look" of the terminal that is being emulated. The PC keyboard scan codes are incompatible with mainframe scan codes, and must be translated for the keyboard type of the terminal being emulated. Both systems provide advanced PC functions such as residency, keyboard remapping, and multiple display support.

MPA

The MPA implements both emulation of the DCA and IBM interfaces. In order to achieve these two goals, an overall architecture similar to the DCA and IBM systems is employed (see *Figure 2-4*). The logical split in functionality between the PC and the adapter board processors is roughly analogous; the PC provides presentations services and the adapter hardware/firmware handles the host terminal emulation tasks. The BCP on the adapter board is soft-loaded by the PC and configured to operate in one of the protocol and interface modes. The adapter board then assumes the hardware emulation tasks of the physical interfaces of the DCA or IBM products.

The MPA hardware consists of a DP8344V, or Biphase Communications Processor, running with 8k • 16 bits zero wait state instruction memory at 9.45 MHz, 32k • 8 bits zero wait state data RAM, a dual coax/twinax front end, and a chameleon-like interface that enables the MPA to appear to have multiple interfaces. The BCP Remote Interface Configuration register (RIC) is located in PC I/O space at 2DFh (see *Figure 2-5*). This register facilitates downloading of instructions and data memory from the PC, starting and stopping the processor, and configuring the low level interface mode. The MPA utilizes the low level fast buffered write/latched read interface mode.

For debugging purposes, the BCP's program counter is available at locations 2DDh and 2DEh (low byte, high byte). The MPA Configuration register (see *Figure 2-6*) is located at 2DCh and controls which type of high level interface the MPA board is operating in (i.e., IRMA, Smart Alec, IBM coax, etc.). Changing the value of this register while the MPA is operating will cause the interface mode to change and reset the emulation session in progress.

When either of the DCA modes are enabled, the I/O block 220h–22Fh is decoded, split into read and write banks, and mapped into BCP memory. For the IBM mode, the I/O block from 2D0h–2DAh is decoded and the Write-Under-Mask function is enabled. In addition, the 8k of dual-port RAM is defined according to the IBM interface mode. For CUT and 3287 printer emulation, only the lower 4k of the dual-port RAM is used. For DFD mode, the entire 8k block may be utilized. Neither DCA mode utilizes dual-port memory, so the MPA firmware maps control and screen information into the dual-port area for debugging purposes.

The MPA interface mimics the DCA and IBM interfaces by interrupting the BCP when accesses occur to the I/O space

of interest, and then holding off further PC accesses until the RAM area the PC reads can be updated to the proper format. In all interface modes, the BCP monitors I/O accesses through the use of the "Access" register. This register captures the exact location and type of access the PC has made. The BCP responds in its interrupt service routine by locking out PC accesses before any further PC I/O cycles can complete. The extreme speed of interrupt processing by the BCP makes this feasible. Accesses of the dual-port RAM by the PC are regulated by the interface only in assuring that simultaneous accesses do not occur. The location of the dual-port RAM in the PC memory map is determined by a value written into the 2D7h I/O location. This "Segment" register is the upper 7 bits of the PC address field and is compared with the address presented during memory cycles in the I/O channel for decoding. Writing different values to this register moves the decoded memory block anywhere within the PC memory space to avoid conflicts. The pacing of dual-port accesses is handled by provisions in the emulated interface definition.

The MPA can utilize the DCA or IBM presentation services for display and keyboard functions, or presentation software designed to take full advantage of the MPA interface. The DP8344-EB kit provides software for the IBM PC to load and configure the MPA adapter board.

The PC I/O map for the MPA adapter board in DCA and IBM modes is as follows:

TABLE 2-3. I/O Map

220h	IRMA command/status register
221h	IRMA argument/response
222h	IRMA argument/response
223h	IRMA argument/response
224h	decoded, unused
225h	decoded, unused
226h	IRMA command/attention semaphore control
227h	IRMA command/attention semaphore
228h	Smart Alec command/status register
229h	Smart Alec argument/response register
22Ah	Smart Alec argument/response register
22Bh	Smart Alec argument/response register
22Ch	decoded, unused
22Dh	Smart Alec control register
22Eh	Smart Alec control register, command/attention semaphore
22Fh	Smart Alec strobe
2D0h	IBM Interrupt Status Register
2D1h	IBM Visual/Sound
2D2h	IBM cursor address low
2D3h	IBM cursor address high
2D4h	IBM adapter control
2D5h	IBM scan code
2D6h	IBM terminal ID
2D7h	IBM/MPA dual-port segment location register
2D8h	IBM page change low
2D9h	IBM page change high
2DAh	IBM 87E status
2DCh	MPA configuration register
2DDh	MPA program counter low
2DEh	MPA program counter high
2DFh	MPA RIC register
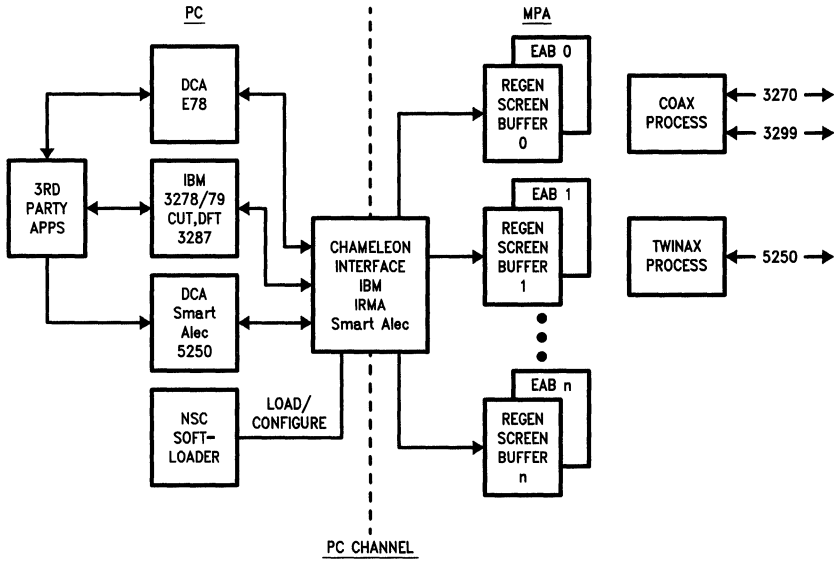
TL/F/10488–5

FIGURE 2-4. MPA System Architecture

7   0

MS0: MEMORY SELECT: DATA, INSTRUCTION
MS1: MEMORY SELECT
STRT: BCP START/STOP
LW: LATCHED WRITE INTERFACE MODE
LR: LATCHED READ MODE
RW: FAST WRITE MODE
SS: SINGLE STEP THE BCP
BIS: BIRQ STATUS
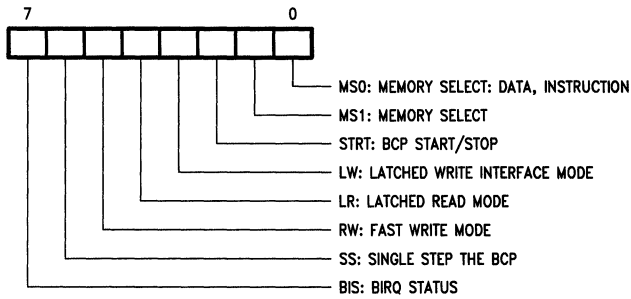
TL/F/10488–6

FIGURE 2-5. BCP Remote Configuration Register

7   0

POR INTERFACE (SELF CLEAR)
POR SESSION (SELF CLEAR)
3299 MODE
COAX EAB INSTALLED
NOT USED
IBM INTERFACE MODE
DCA INTERFACE MODE
5250/3270
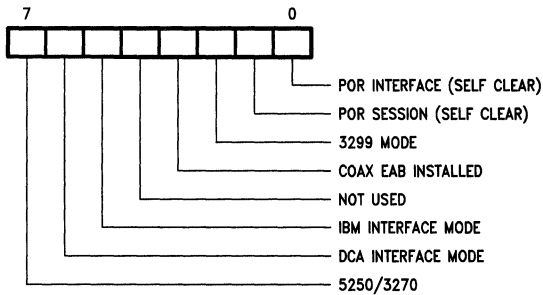
TL/F/10488–7

FIGURE 2-6. MPA Configuration Register

2

MPA Firmware Organization

The BCP firmware provides true 5250, 3270, and 3299 emulation support, as well as providing the intelligence behind the PC interface. To do this, a software architecture radically different than the DCA or IBM systems was developed. The real power of the BCP lies in its rich instruction set and full featured CPU. Taking advantage of that power, the BCP firmware is interrupt driven and task oriented. It is not truly multi-tasking, although the firmware logically handles multiple tasks at once. The firmware basically consists of a round robin, prioritized task scheduler with real-time interrupt handlers to drive the system. Events that happen in real-time, such as accesses by the PC or host commands, schedule tasks to complete in foreground processing. Real-time status and responses are developed and presented in real-time.

The BCP firmware uses a number of memory constructs known as templates to handle its data structures. The primary construct is the DCP, or Device Control Page. The DCP is a 256 byte block that contains all global system variables. The DCP contains a map of which SCPs, or Session Control Pages are active. Each SCP is 256 bytes and contains all variable storage for a particular session; 3270, 5250, or 3299. Each SCP has a corresponding screen buffer, and optionally an EAB buffer (there is no EAB in 5250 terminals). The video buffer, when used, is 4k in length and corresponds to a PC monochrome display driver buffer. The different SCPs are controlled by re-entrant firmware that is run by a central tasker, the Kernel. The Kernel controls which SCP is given control, and control always passes back to Kernel when that SCP has been updated. Also, the Kernel runs non-SCP related tasks such as the interface tasks.

The dual-port RAM is used for different purposes depending on the particular mode of the MPA system. In the DCA emulation modes, the dual-port RAM is used for debugging purposes only. In the IBM emulation modes, the dual-port is utilized according to which type of device the IBM software is emulating. In CUT mode emulation, only 4k is used; for 3287 emulation, 4k is used; for DFT mode, the entire 8k is used.

SECTION THREE — HARDWARE ARCHITECTURE

This section focuses on the hardware design of the MPA. Designed to support both the coax (3270/3299) and twinax (5250) protocols, the hardware also allows emulation of the PC interfaces outlined in Section 2. By taking advantage of the BCP's power and integrating the extra logic requirements into programmable logic devices, this level of functionality was provided on a single half-length PC XT/AT card. In an effort to convey the reasons behind specific decisions made in the hardware design, the design methodology is presented from a "top-down" perspective.

Architectural Overview

The MPA hardware should be viewed as three conceptual modules (see *Figure 3-1*), including:

1. BCP minimum system core, consisting of the BCP, instruction memory, data memory, clock, and reset logic.

2. PC interface, including the PC and BCP memory decode and interrupts.

3. Coax and twinax front-end logic and connectors.

These module divisions are denoted by the dotted lines seen in *Figure 3-1*. The minimum system core is required, with some modifications, for any design using the BCP. The type of bus (PC, PS/2® Micro Channel™, VME, etc.) and transfer rate requirements dictate the interface logic, which, for the MPA design, is optimized for the PC XT/AT I/O channel. The front-end logic meets the physical-layer requirements of the 3270 and 5250 protocols, and are adaptions of the line interface designs presented in the BCP coax and twinax application notes (see References, Appendix C).

Since much of the logic external to the BCP is implemented in programmable logic devices (PALs), these conceptual partitions overlap at the device level. Although the design can be implemented in discrete logic, we chose to use programmable logic devices to shorten development time, decrease board real-estate requirements, and maintain maximum future adaptability. The schematic and the listings describing the logic embodied in the PALs are in the Hardware Reference in Appendix A.

BCP Minimum System Core

The BCP offers a high level of integration and many functions are provided on-chip; there is, however, a minimum amount of external logic required. This core is comprised of the BCP and the external logic required to support the clock requirements, reset control, Harvard memory architecture, and multiplexed AD bus (see *Figure 3-2*).

Clock Source

The coax and twinax protocols operate at substantially different clock frequencies (2.3587 MHz and 1 MHz, respectively), therefore two clock sources are required. The BCP has the software-programmable flexibility to drive both the CPU and transceiver from the same clock, the clock independently divided down to either or both sections, or by two separate asynchronous clocks (utilizing the external transceiver clock input, XTCLK). To provide sufficient waveform resolution, the transceiver must be clocked at a frequency equal to eight times the required serial bit rate. This means that an 18.8696 MHz (8 x 2.3587 MHz) clock source is required when operating in the 3270 coax environment and an 8 MHz clock (8 x 1 MHz) is needed for the 5250 twinax protocol. The 18.8696 MHz clock is also a good choice for the BCP's CPU section; hence, the two sections share this clock source in the coax mode. To maximize available CPU bandwidth in the twinax mode, this same clock source drives the CPU while a TTL clock is used to drive the BCP's external transceiver clock input. Therefore, in the twinax mode the BCP's CPU and transceiver sections operate completely asynchronously.
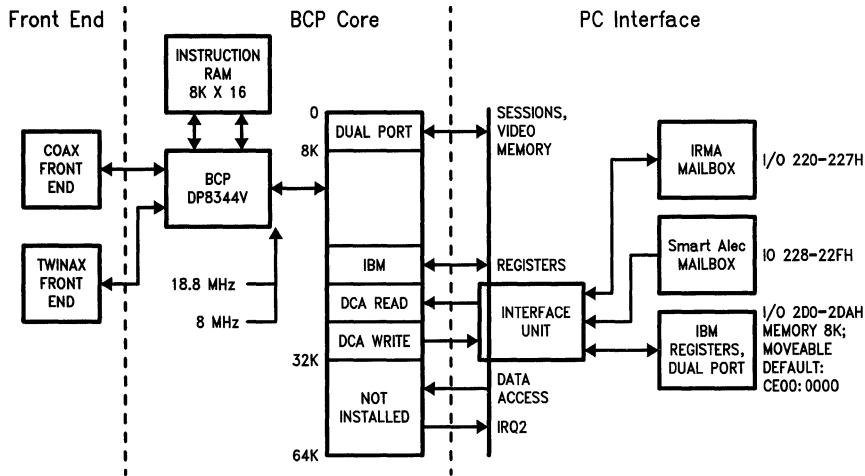
FIGURE 3-1. MPA Hardware Architecture

TL/F/10488–8

The 18.8696 MHz clock is provided by the BCP's on-chip clock circuitry and an external oscillator. This circuit, in conjunction with external series load capacitors, forms a "Pierce" parallel resonance crystal oscillator design. The oscillator is physically located as close as possible to the X1 and X2 pins of the BCP to minimize the effects of trace inductances. The traces (0.05″) are wider than normal. NEL Industries makes a crystal specifically cut for the 18.8696 MHz frequency and is the recommended source for these devices. This crystal requires a 20 pF load capacitance which can be implemented as 40 pF on each lead to ground minus the BCP/socket capacitance and the trace capacitance. A typical value for the BCP/socket combination capacitance is 12 pF. The wide short traces contribute very little additional capacitance. We chose a standard value of 27 pF for the discrete ceramic capacitors C19 and C20, placing them as close as possible to the crystal. The twinax clock is provided by a standard, 8 MHz, TTL, monolithic clock oscillator attached to the BCP's external clock input, XTCLK.

The MPA runs the BCP at half speed, 9.45 MHz (CCS[7] = 0), with zero instruction (niw) and zero data (ndw) wait states resulting in a T-state of 106 ns. For a system running the BCP at full speed, 18.8696 MHz (CCS[7] = 1), the T-state would be 53 ns. The T-state period can be calculated using the following equation:

$$\text{T-state} = 1/(\text{CPU clock frequency})$$

Reset Control

Power-up reset for the BCP consists of providing a debounced, active low, minimum pulse width of ten T-states. Since the BCP powers up in the slowest configuration, a T-state is the period of the oscillator divided by two, or 106 ns. The external logic must therefore provide a minimum 1.06 μs reset pulse to the BCP. In addition to the minimum power-up reset requirement for the BCP, the MPA design incorporates several reset sources including: the PC I/O channel reset control signal (active high), a manual switch (for debug purposes), and an automatic reset if the digital supply voltage drops by more than 10%.

We chose the Texas Instruments TL7705A supply voltage supervisor to monitor V_{CC} and provide the minimum pulse width requirement. This device will reset the system if the digital 5V supply drops by more than 0.5V, and keep the reset asserted until the voltage returns to an acceptable level or a minimum time delay is met. The time delay is set by an external capacitor and an internal current source. Since this time delay is not guaranteed in the data sheet and because we have a non-debounced manual switch, we chose a 0.2 μF ceramic capacitor resulting in a typical 1.3 ms reset pulse width. A 0.1 μF ceramic capacitor is connected to the REF input of the chip to reduce the influence of fast transients in the supply voltage. The active high PC reset signal is inverted and logically ANDed with the manual switch output in the AUX_CTL (AUXiliary ConTroL) PAL. The active low output of the bipolar TL7705A is the MPA system reset and is pulled up by a 10k resistor for greater noise immunity.

Memory Architecture

The BCP utilizes separate instruction and data memory banks to overcome the single bus bandwidth bottleneck often associated with more conventional architectures. Instruction memory is owned exclusively by the BCP (remote processor accesses to this memory occur through the BCP, and only when the BCP is idle); therefore, the entire instruction memory/bus bandwidth is available to the BCP. This architecture allows the BCP to simultaneously fetch instructions and access data memory, thus load/store operations can be very quick. It is important to note, however, that the instruction bus bandwidth does have some dependency on data bus activity. If a remote processor, for instance, is currently the data bus master, execution of an instruction accessing data memory will be waited, degrading BCP CPU performance.

The speed of both instruction and data memory accesses is limited by memory access time. Since the BCP features programmable memory wait states, the designer has the flexibility of choosing memories strictly on a cost/performance trade-off. No external hardware is required to slow the BCP

2
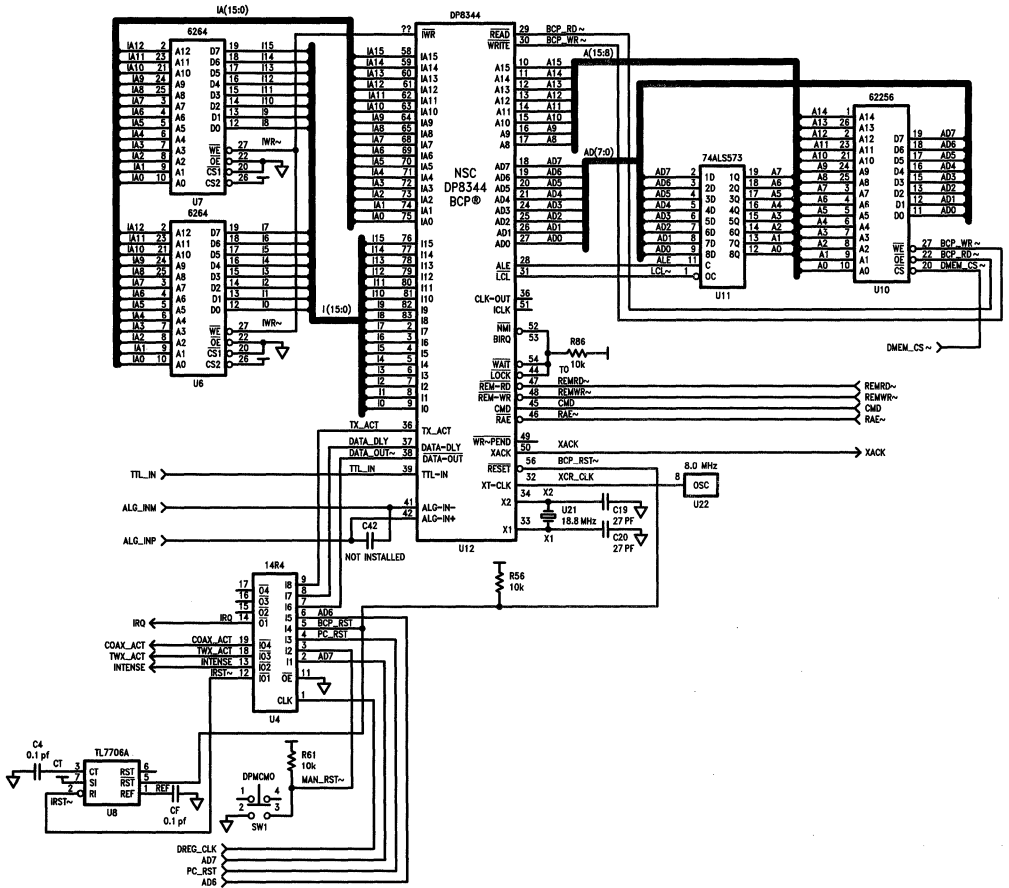
FIGURE 3-2. BCP Core

TL/F/10488-9

memory accesses down (unless the maximum number of programmable wait states is insufficient, in which case the WAIT input of the BCP can be utilized). Instruction memory access time has the biggest impact on system performance since every instruction executed involves an access of this memory. Each added instruction wait state degrades zero wait state performance by approximately 40%. Load/store operations occur less frequently in normal code execution, therefore relatively slower data memory can often be utilized. Each additional data memory wait state degrades the performance of a zero wait-state data access by about 33%.

Instruction Memory

A design goal for the MPA project dictated our choice of static RAM for instruction memory, since the ability to soft-load code from the PC was necessary. Furthermore, to maximize CPU bandwidth we chose zero wait-state instruction memory operation. When the hardware was designed, instruction memory requirements were estimated at 4k to 8k words, therefore two 8k x 8 bit static RAMs were employed.

Instruction memory access time requirements can be calculated by plugging the MPA T-state value, 106 ns, into the formulas presented in Table 3 of the Electrical Specifications section of the DP8344 data sheet. Using Table 3, the instruction memory access time requirement is calculated by using the following formula (parameter 4, the instruction memory read time):

$$(niw + 1.5) \times T + (-24) \text{ ns}$$

Where: niw, the number of instruction wait-states, and T = 106 ns. The maximum access time is $(1.5 \times 106) - 24 = 135.5$ ns. For a system running the BCP at full speed (T-state = 53 ns), the maximum access time is $(1.5 \times 53) - 24 = 55.5$ ns. Comparing both the half and full speed maximum instruction memory access time requirements, it is apparent 55 ns RAMs are appropriate. A complete instruction memory timing analysis is provided in Appendix B.

Reads of instruction memory by the remote system occur through the BCP and look identical in timing to the local (BCP) reads on the instruction bus.

Soft-Load Operation

The BCP cannot modify instruction memory itself. Memory is only written through the BCP (while the BCP is idle) from the remote system (PC), and is referred to as the "soft-load" operation. Since the BCP has an 8-bit data path and a 16-bit instruction bus (see *Figure 3-2*), instructions are read or written by the PC in two access cycles; the first transferring the low byte, the second accessing the high byte of the instruction and automatically incrementing the Program Counter after the instruction has been accessed. See the Remote Interface section of the DP8344 data sheet for a complete description.

The critical parameter for instruction writes is the minimum write strobe pulse width of the RAM, which is about 40 ns for most 8k x 8, 55 ns static RAMs (55 ns RAM specifications are compared to the BCP minimum requirements since it represents the worst case). The IWR (BCP Instruction WRite output, active low) minimum pulse width is calculated from the formula in Table 20 of the Electrical Specifications section of the DP8344 data sheet, (niw + 1) T − 12 ns. For soft-loads that occur after reset, the CPU clock is in the POR half-speed state, therefore a T-state is 106 ns; thus, IWR tpw is (niw + 1) 106 − 12 = 94 ns. Soft-loads that occur after the BCP Device Control Register has been initialized to full speed operation represent the worst case tim-

ing of (niw + 1) 53 − 12 = 41 ns, which is still greater than the 55 ns RAM requirement.

Other parameters that must be considered are data setup and hold times for the RAM. The BCP must provide valid data on the Instruction bus before the minimum setup time of the RAM and hold the valid data on the bus at least as long as the minimum hold time. For the RAMs we considered, these times were 25 ns and 0 ns, respectively. Again, looking at Table 20 we see that if valid data for the high byte of the instruction is present on the AD bus in time, the BCP is guaranteed to present valid data on the Instruction bus a minimum of (niw + 1) 106 + (−26) ns = 26 ns before the rising edge of IWR, and will hold that data on the bus for a minimum of 33 ns afterward, thus ensuring successful operation. See the MPA timing analysis in Appendix B and the PC interface section in this chapter for a discussion of AD bus timing.

Data Memory

A considerable amount of data memory was required for the MPA design since the system supports multiple sessions (see Chapter Four, MPA Software Architecture, for more information). For this reason we specified 32k of 8-bit data memory.

Data Memory Timing

Data RAM can be accessed by both the BCP and the remote system, part of the RAM appears to the remote system as dual-ported RAM via the Remote Interface logic of the BCP (see *Figure 3-1*). This memory can be both read from and written to during BCP code execution. Designing in the data RAM is therefore a more complicated procedure than selecting instruction memory. The timing parameters and formulas associated with BCP accesses of data memory (referred to as local accesses) are found in Tables 1 and 2 of the Electrical Specifications of the DP8344 data sheet. Using 106 ns for the MPA T-state and zero for ndw (number of data wait-states) as defined earlier, we can verify the critical memory parameters by comparing the results of the calculations against the RAM requirements. The 32k x 8, 100 ns static CMOS RAM minimum requirements for the critical parameters are compared against the BCP's minimum specifications and are listed in Table 3-1.

TABLE 3-1. Data Memory

Parameter	RAM	BCP**
Address Setup	0	129
Chip Select to Write End	90	208*
Access Time	100	131
Write Strobe Width	60	96
Data Setup	40	94
Data Hold	0	62

All units are in nanoseconds

*This number is derived from parameter 5 in Table 2 minus the maximum prop delay through a 20L10A PAL plus the minimum write pulse width low.

**106 ns T-state.

Again, the numbers reveal the validity of the hardware design for local (BCP) accesses of data memory. Please see the PC interface section for timing related to remote accesses. Also, an MPA timing analysis of both 106 ns and 53 ns T-states is provided in Appendix B.

Multiplexed AD Bus

The BCP's 8-bit data bus is multiplexed with the lower 8 bits of the data memory address bus to lower pin count. This necessitates de-multiplexing the Address/Data bus exter-

nally. The timing of the ALE (Address Latch Enable) control signal relative to the AD bus is optimized for use with a standard octal latch, therefore a 74ALS573 is employed to provide separate Address and Data buses for the system. The TRI-STATE® buffers of the latch are enabled by the BCP output LCL (active low) such that if a remote access occurs this device will TRI-STATE.

PC Interface

As mentioned earlier, the MPA supports the industry-standard interfaces associated with coax and twinax terminal emulation. These include:

coax —   IBM 3270/3278 Emulation Adapter interface
          DCA Decision Support interface (IRMA)

twinax — DCA Smart Alec interface

These interfaces share some common elements, but have many differences as well. The IBM adapter employs an interrupt-driven interface, IRMA's PC interface is a polled implementation, and Smart Alec, while operating in a polled environment, has the capability of interrupting the PC as well. The IBM Emulation Adapter's control registers are mapped into the PC's I/O space; the screen buffer is mapped into the PC's memory space and is relocatable (see Table 3-2). The two DCA interfaces occupy a contiguous block of PC I/O space only; their screen buffer(s) are not directly visible to the PC. These architectures are explored in much greater detail elsewhere in this manual.

TABLE 3-2. PC Mapping of the MPA Board

Description	Address	
	I/O	Memory
IBM Interface		
Remote Interface Control (RIC)	02DF*	—
Program Counter High Byte	02DE*	—
Program Counter Low Byte	02DD*	—
MPA Configuration	02DC*	—
IBM Control Registers	02D0–02DA	—
IBM Screen Buffer		CE000 (relocatable)
DCA DSI Interface		
IRMA (coax)	0220–0227	
Smart Alec (twinax)	0228–022F	

*reserved IBM register spaces

The MPA design had to encompass all of these implementations. This was accomplished by taking advantage of the underlying similarity of the interfaces and the flexibility of the BCP. We minimized chip count and board space requirements through judicious partitioning of the PC address decode while emulating the interface registers in data RAM.

The PC address decoding is partitioned into sections that check for the relocatable memory block and the I/O register address of the different interfaces to translate these addresses into the proper area of the BCP data memory. The BCP data memory map is divided in half, the lower 32k contained in the single 32k x 8 RAM described earlier and the upper half decoded for several functions including two chip selects that are available for further product enhancement

(see Table 3-3). The decoding sections feed into a control section that makes the final decision on whether or not the current PC bus cycle is an access of one of the emulated systems. It should be noted that the type of emulation is not selectable; the MPA board will respond to accesses of all of the PC addresses detailed in Table 3-2. The MPA will not run concurrently with any of the boards it emulates, or any other board that overlaps with these same addresses.

The BCP's RIC (Remote Interface Control) register and the program counter read registers are mapped into the PC's I/O space. The PC can always find these three registers by reading I/O hex addresses 2DD, 2DE, and 2DF for PC LO, HI, and RIC, respectively. The DCA interfaces (both IRMA and Smart Alec) occupy PC I/O addresses 220–22F only. The IBM interface occupies PC I/O addresses 2D0–2DF for register space, and a relocatable 8k block of memory for the screen buffer. Note that RIC and the PC HI and LO read registers occupy the upper three addresses of the space dedicated to the IBM registers.

PC Address Decode

The relocatable screen buffer of the IBM interface is decoded in discrete hardware consisting of three components: U18, a 16L8B PAL that buffers the high byte of the PC address and asserts the output IO__MAYBE if these PC address lines and the PC AEN (Address ENable) line are all low; U19, a 74ALS521 magnitude comparator that compares the buffered PC address against the stored value of the address; and the Segment Register U20, a 74ALS574 that contains the stored address used to identify the screen buffer block. The relocatable block of data memory defaults to base address CE000 on the IBM adapter. On the MPA board, this value must be loaded into the segment register (PC I/O address 2D7h) before the PC can access the MPA screen buffer area. This segment register is not accessible to the BCP. A PC read of this address accesses the corresponding RAM location only.

PC address lines A12–4 are brought into the PAD__DEC (Pc ADdress DECode) PAL for decode and translation. This PAL outputs IOD1 and IOD0 that indicate which, if any, emulated interface is being accessed. These signals are used in conjunction with MMATCH, IO__MAYBE, and the read and write strobes of the PC in the REG__DEC PAL to make the final determination on the validity of the access. If it is an emulated interface I/O register access, IO__ACCESS will be asserted back to the PAD__DEC PAL. This PAL will in turn translate the access to the top of the BCP data RAM where the I/O register page is located (see Table 3-2). Note the differentiation between PC reads and writes for the DCA translation. This is required to emulate the dual-ported register files used on the DCA boards.

If the PC access is to the IBM screen buffer, IO__ACCESS will not be asserted out of the REG__DEC PAL and the PAD__DEC PAL will, when LCL goes high on the remote access, force A15–13 low and pass the buffered A12–8 onto the data RAM. PC address lines A7–0 are buffered by a 74ALS541 and passed onto the BCP data memory address lines AD7–0 when LCL switches high for the remote access. The data memory RAM's chip select, DMEM__CS, is asserted on any remote access. If the BCP's LCL output goes high, DMEM__CS will be asserted low; on a local ac-

cess, this signal will be asserted if the BCP's A15 signal is low (RAM occupies the lower half of the BCP's memory map).

This scenario for remote accesses works because RAM is the only element external to the BCP that is visible to the PC. If the PC is accessing the BCP (RIC, the Program Counters, or instruction memory), the BCP's READ/WRITE strobes will not be asserted to the data RAM. On a PC access of the BCP's RIC register, for example, data RAM will be selected and the CMD (CoMmanD) output of the REG__DEC PAL will be asserted to the BCP, selecting the BCP's RIC. No bus collision will occur on a read nor data inadvertently destroyed on a write because the BCP will not assert the external strobes on an internal register access.

The REG__DEC PAL also combines the memory and I/O read/write strobes to form the REMRD/REMWR strobes to the rest of the MPA system. Since PC bus cycles can only be validated by the assertion of one of these strobes, this PAL makes the final decision on the validity of the bus cycle. If the PC cycle is a valid access of the BCP system, this PAL will assert RAE (Remote Access Enable), the BCP's chip select, RIC, the output CMD, and the BCP's READ/WRITE strobes will determine which part of the system receives or provides data. The REG__DEC PAL also decodes the external BCP Program Counter low and high latches U1 and U2 and provides the TRI-STATE enables for these latches. These are not part of the BCP system and RAE will not be asserted to the BCP on reads of these latches (see the Miscellaneous section page 3-13 for more information).

The PC interrupt for the IBM interface is set and cleared by the BCP through U4, the AUX__CTL PAL. The interrupt is set from the BCP by pointing to address range A000–BFFF and writing to this register with AD7 set high; it is likewise cleared by writing AD7 low to this register. The interrupt powers up low (de-asserted) and can be assigned to PC interrupts IRQ2, 3, or 4 by setting the appropriate jumper (JP7–9).

Remote accesses of the BCP are arbitrated and handled by the Remote Interface control logic. The arbiter is a sequential machine internal to the BCP that shares the same clock with the CPU but otherwise operates autonomously. This unit is very flexible and offers a number of configurations for different external interfaces (see the Remote Interface chapter of the BCP data book). We chose to use the Fast Buffered Write/Latched Read interface configuration to maximize the possible data transfer rate and minimize the BCP performance degradation by the slower PC bus cycles. Data are buffered between the PC and BCP data buses with a 74ALS646, giving us a monolithic, bi-directional transceiver with latches for PC reads and buffering PC writes.

As mentioned earlier, the BCP CPU and Remote Interface units operate autonomously. The CPU can poll the Condition Code register to see if a remote access of data memory has occurred, but internally it cannot identify which location was addressed. Since the I/O registers are mapped into the BCP's data RAM and the CPU has to know which register was written to by the PC, external logic is provided that latches the low four bits of the address bus during remote accesses. The BCP can read this external register to identify which emulated register has been modified and take the appropriate action.

The BCP's bi-directional interrupt, BIRQ, is configured as an interrupt into the BCP, and is set on the trailing edge of a PC write of the BCP I/O register space (excluding RIC and the PC hi and lo addresses). The BCP can identify which register was accessed by reading the Access register (U5 in *Figure 3-3*), which is a 16R6 registered PAL, mapped directly above the dual ported RAM in the BCP's data memory map (see Table 3-3). A BCP read of this register will clear BIRQ and the Access Valid bit (AD5). The next most significant bit, AD4, will always be low. The low nibble of the register, AD0–3, reveals which of 16 registers was accessed. Timing for the clock and TRI-STATE enable of this PAL is provided by the CTL__TIM PAL (U9). The PAL is clocked only on remote writes to the I/O register page (denoted by IO__ACCESS being asserted from the REG__DEC PAL) and local (BCP) reads of the ACC__REG. The Access Valid bit is merely the value of LCL when the PAL was clocked, therefore it will be high for a remote write and low for a local read. This bit gives the software the flexibility to operate in either an interrupt-driven mode using BIRQ, or a polled mode by reading this register.

Front-End Interface

The line interface is divided into coax and twinax sections, each section being comprised of an interface connector, receiver, and driver logic (see *Figure 3-4*). These sections are independent but are never operated concurrently. The coax medium requires a transformer-coupled interface while the multi-drop twinax medium is directly coupled to each device (see NSC Application Notes, IBM PAIs listed in the References, Appendix C).

The transmitter interface on the DP8344 is sufficiently general to allow use in 3270, 5250, and 8-bit transmission systems. Because of this generality, some external hardware is needed to adapt the outputs to form the signals necessary to drive the twinax line. The chip provides three signals: DATA-OUT/, DATA-DLY, and TX-ACT. DATA-OUT/ is bi-phase serial data (inverted). DATA-DLY is the biphase serial data output (non-inverted) delayed one-quarter bit-time. TX-ACT, or transmitter active, signals that serial data is being transmitted when asserted. DATA-OUT/ and DATA-DLY can be used to form the A and B phase signals with their three levels by the circuits shown in *Figure 3-4*. TX-ACT functions as an external transmitter enable. The BCP can invert the sense of the DATA-OUT/ and DATA-DLY signals by asserting TIN {TMR[3]}. This feature allows both 3270 and 5250 type biphase data to be generated, and/or utilization of inverting or non-inverting transmitter stages.

The module line drivers are software selectable from the BCP via logic embedded in the AUX__CTL and CTL__TIM PALs. The Auxiliary Control Register is mapped into the A000–BFFF area of the BCP memory map. The coax module is selected by pointing to this address area and writing a '0' out on the AD6 data line. The twinax is selected by writing a '1' on this signal. The coax section is selected on power-up. The voltage supervisor described earlier in the Reset Control section also plays a role here, deactivating the line drivers of both modules if the +5V supply drops more than 10% at any time. The receivers are selected on-board the BCP by the SLR (Select Line Receiver) control in the Transceiver Control Register. Setting {TCR[5]} to a '1' selects the on-chip comparator and thus the coax module input; a '0' on this control selects the TTL-IN receiver input for the twinax module.
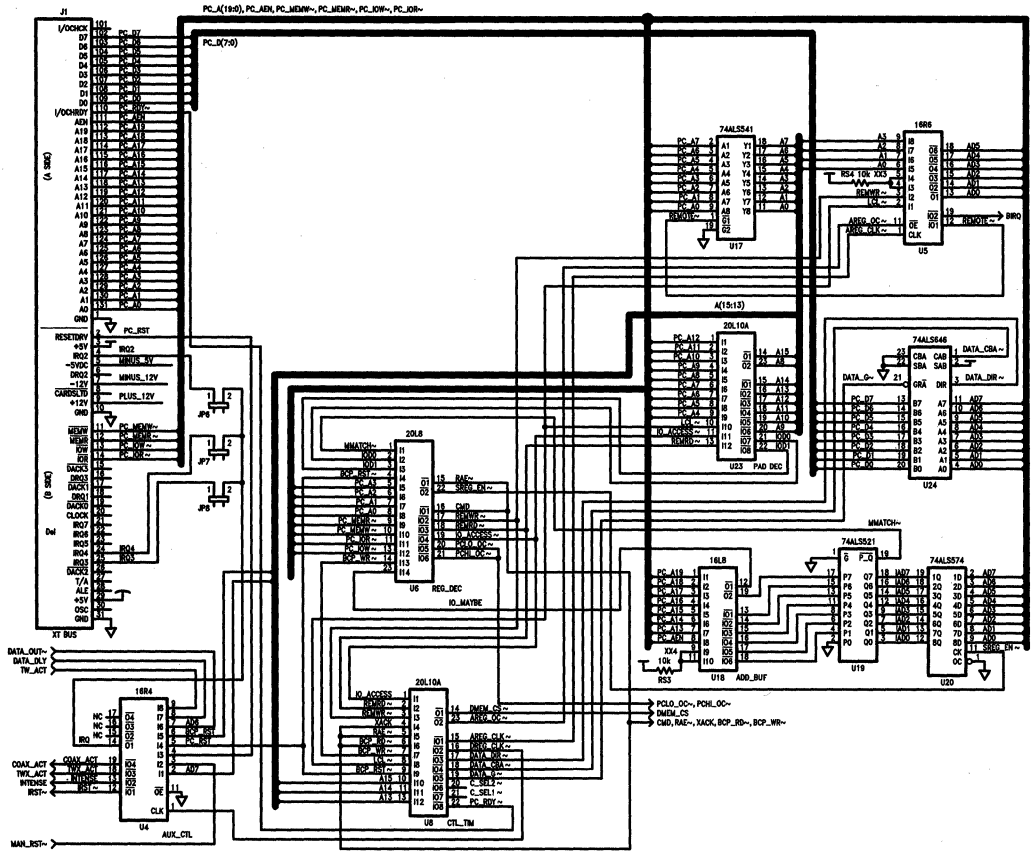
FIGURE 3-3. PC Interface/Memory Decode

TL/F/10488–10

TABLE 3-3. BCP Data Memory Map

Description	BCP Address (A15–0)	PC Address
Chip Select 2 (not used)	E000–FFFF	—
Chip Select 1 (not used)	C000–DFFF	—
Auxiliary Control Registers	A000–BFFF	—
PC Access Registers	8000–9FFF	—
*IBM API Registers	7FD0–7FDF	2D0–2DF
*DCA API (IRMA and Smart Alec)		220–22F
PC Writes	7F20–7F2F	↓
PC Reads	7E20–7E2F	
BCP-Owned Memory Area	2000–7E1F	—
*Screen Buffer Area	0000–1FFF	Relocatable
*—Dual-Ported RAM (Visible to Both BCP and PC)		
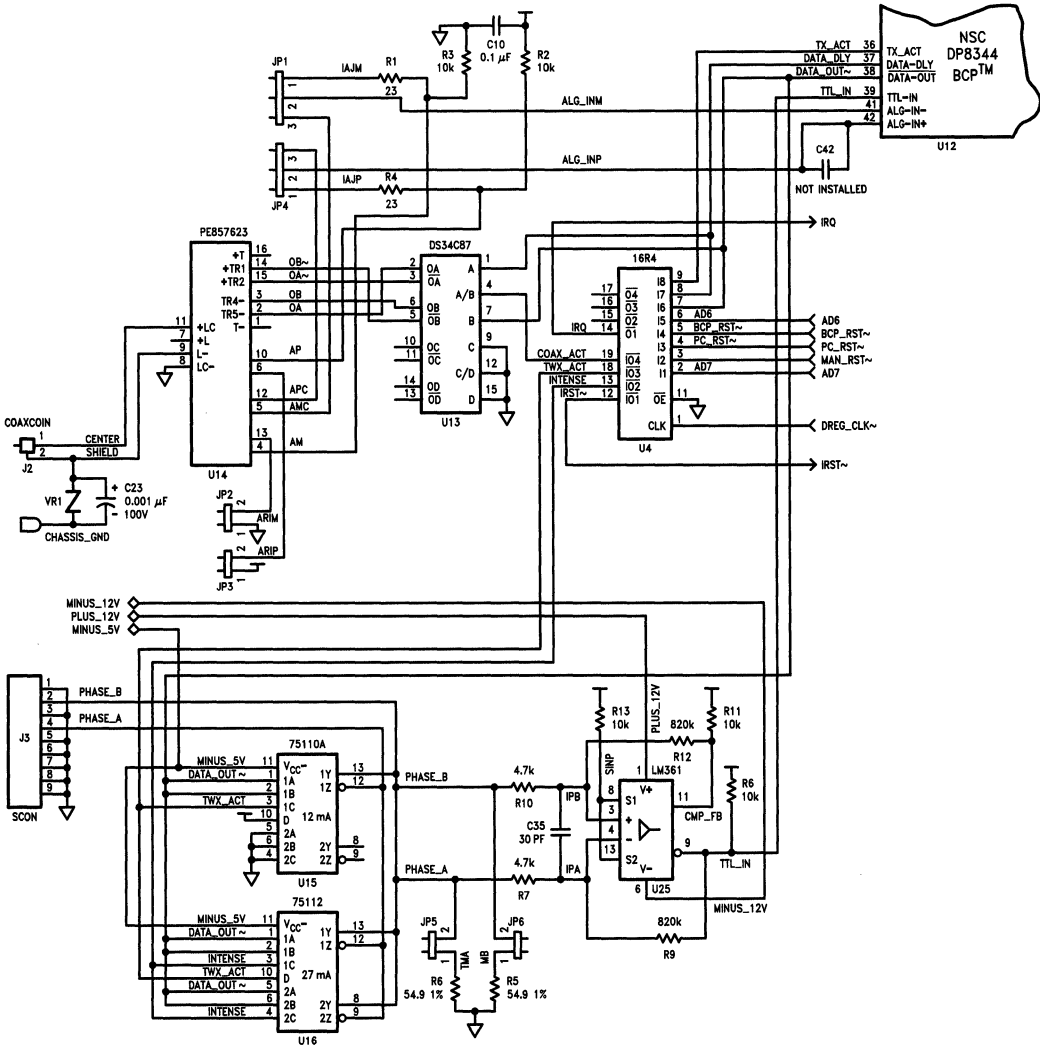
FIGURE 3-4. Front End

TL/F/10488–11

Coax Module

This module attaches to the coax medium through a BNC connector. An over-voltage protection circuit is provided to protect the board. The over-voltage protection circuit is a common configuration of a varistor and high-voltage capacitor in parallel, tied between the shield of the coaxial connector and chassis ground. The transformer isolation required in this transmission protocol is provided by a hybrid designed specifically for the BCP by Pulse Engineering. In addition to the transformer this circuit contains all of the discrete components needed to generate the proper voltage levels for the transmitter and to provide the proper bias for the BCP comparator. When the hardware was designed, the hybrid package did not have the proper biasing circuit, therefore the jumpers and resistors seen in *Figure 3-4* are provided to allow the design to run in either configuration.

The coax line driver is a National Semiconductor LSI device, the DS34C87. This is the new CMOS equivalent of our standard bipolar 3487, and either part will work well in this circuit. These designs are optimized for minimum skew between the two outputs, and the voltage levels driven onto the coax are fixed by a resistor network embedded in the hybrid transformer package. The on-chip comparator of the BCP is optimized for the specifications of the coax medium, and requires only the external DC biasing resistors for correct operation.

Twinax (5250) Module

The 5250 transmission system is implemented in a balanced current mode; every receiver/transmitter pair is directly coupled to the twinax at all times. Data is impressed on the transmission line by unbalancing the line voltage with the driver current. The system requires passive termination at both ends of the transmission line. The termination resistance value is given by:

Rt = Zo/2; where
Rt : termination resistance
Zo : characteristic impedance

In practice, termination is accomplished by connecting both conductors to the shield via 54.9Ω, 1% resistors; hence the characteristic impedance of the twinax cable of 107Ω ± 5% at 1.0 MHz. Intermediate stations must not terminate the line; each is configured for "pass-through" instead of "terminate" mode. Stations do not have to be powered on to pass twinax signals on to other stations; all of the receiver/transmitter pairs are DC coupled. Consequently, devices must never output any signals on the twinax line during power-up or down that could be construed as data, or interfere with valid data transmission between other devices. The MPA board is factory set to "terminate" mode. To effect "pass-through" mode, jumpers JP5 and JP6 should be removed.

Twinax Waveforms

The bit rate utilized in the 5250 protocol is 1 MHz ±2% for most terminals, printers and controllers. The IBM 3196 display station has a bit rate of 1.0368 MHz ±0.01%. The data are encoded in biphase, NRZI (non-return to zero inverted) manner; a "1" bit is represented by a positive to negative transition, a "0" is a negative to positive transition in the center of a bit cell. This is opposite from the somewhat more familiar 3270 coax method. The biphase NRZI data is encoded in a pseudo-differential manner; i.e., the signal on the 'A' conductor is subtracted from the signal on 'B' to form the waveform shown in *Figure 3-5*. Signals A and B are not differentially driven; one phase lags the other in time by 180°. *Figures 3-6* and *3-7* show actual signals taken at the driver and receiver after 5000 ft. of twinax, respectively.

The signal on either the A or B phase is a negative going pulse with an amplitude of −0.32V ±20% and duration of 500 ±20ns. During the first 250 ±20 ns, a pre-distortion or pre-emphasis pulse is added to the waveform yielding an amplitude of −1.6V ±20%. When a signal on the A phase is considered together with its B phase counterpart, the resultant waveform represents a bit cell or bit time, comprised of two half-bit times. A bit cell is 1 μs ±20 ns in duration and must have a mid bit transition. The mid bit transition is the synchronizing element of the waveform and is key to maintaining transmission integrity throughout the system. The maximum length of a twinax line is 5000 ft. and the maximum number of splices in the line is eleven. Devices count as splices, so that with eight devices on line, there can be four other splices. The signal 5000 ft. and eleven splices from the controller has a minimum amplitude of 100 mV and a slower edge rate. The bit cell transitions have a period of 1 μs ±30 ns.

The current mode drive method used by native twinax devices has both distinct advantages and disadvantages. Current mode drivers require less power to drive properly terminated, low-impedance lines than voltage mode drivers. Large output current surges associated with pulse transition are also avoided. Unwanted current surges can contribute to both crosstalk and radiated emission problems. When data rate is increased, the surge time (representing the energy required to charge the distributed capacitance of the transmission line) represents a larger percentage of the driver's duty cycle and results in increased total power dissipation and performance degradation.

A disadvantage of current mode drive is that DC coupling is required. This implies that system grounds are tied together from station to station. Ground potential differences result in ground currents that can be significant. AC coupling removes the DC component and allows stations to float with respect to the host ground potential. AC coupling can also be more expensive to implement.

Twinax signals can be viewed as consisting of two distinct phases, phase A and phase B, each with three levels: off, high, and low. The off level corresponds with 0 mA current being driven, the high level is nominally 62.5 mA, +20% −30%, and the low level is nominally 12.5 mA, +20% −30%. When these currents are applied to a properly terminated transmission line the resultant voltages impressed at the driver are: off level is 0V, low level is 0.32V ±20%, high level is 1.6V ±20%. The interface must provide for switching of the A and B phases and the three levels. A bimodal constant current source for each phase can be built that has a TTL level interface for the BCP.

The MPA's twinax line drivers are current mode driver parts available from National Semiconductor and Texas Instruments. The 75110A and 75112 can be combined to provide both the A and B phases and the bi-modal current drive required as in *Figure 3-4*. The AUX_CTL PAL shown in *Figure 3-4* adapts the coax oriented BCP outputs to the twinax interface circuit and prevents spurious transmissions during power-up or down. The serial NRZ data is inverted prior to being output by the BCP by setting TIN, {TMR[3]}.
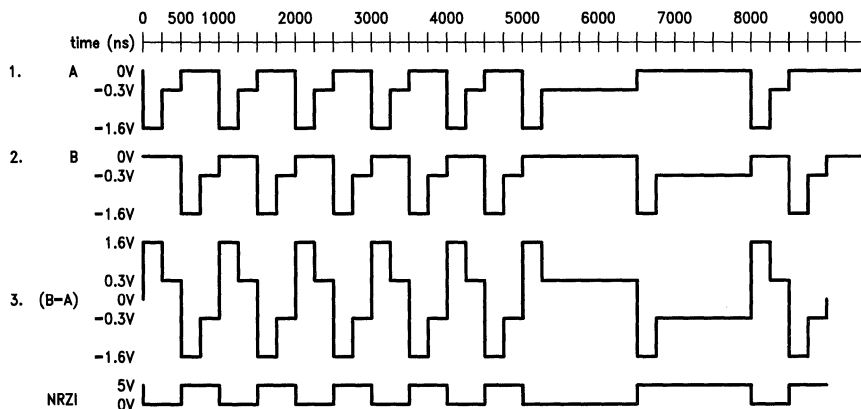
TL/F/10488-12

1. The signal on phase A is shown at the initiation of the line quiesce/line violation sequence

2. Phase B is shown for that sequence, delayed in time by 500 ns

3. The NRZI data recovered from the transmission

FIGURE 3-5. Twinax Waveforms
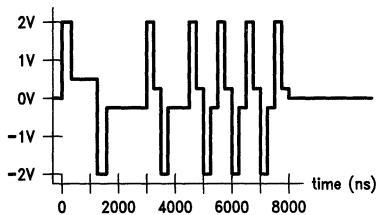
TL/F/10488-13

The signal shown was taken with channel 1 of an oscilloscope connected to phase B, channel 2 connected to A, and then channel 2 inverted and added to channel 1.

FIGURE 3-6. Signal at the Driver
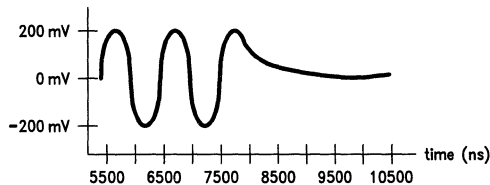
TL/F/10488-14

The signal shown was viewed in the same manner as *Figure 3-6*. The severe attenuation is due to the filtering affect of 5000 ft. of twinax cable.

FIGURE 3-7. Signal at the Receiver

2

Receiver Circuit

The pseudo-differential mode of the twinax signals make receiver design requirements somewhat different than those of the coax circuit. Hence, the analog receiver on the BCP is not used. The BCP provides both analog inputs to an on-board comparator circuit as well as a TTL level serial data input, TTL-IN. The sense of this serial data can be inverted in software by asserting RIN, {TMR[4]}.

The external receiver circuit must be designed with care to assure reliable decoding of the bit-stream in the worst environments. Signals as small as 100 mV must be detected. In order to receive the worst case signals, the input level switching threshold or hysteresis for the receiver should be nominally 29 mV ±20%. This value allows the steady state, worst case signal level of 100 mV, 66% of its amplitude before transitioning.

To achieve this, the National Semiconductor LM361 was chosen. The LM361 is a differential comparator with complementary outputs. The complementary outputs are useful in setting the hysteresis or switching threshold to the appropriate levels. The LM361 also provides excellent common mode noise rejection and a low input offset voltage. Low input leakage current allows the design of an extremely sensitive receiver without loading the transmission line excessively.

In addition to good analog design techniques, a passive, single-pole, low pass filter with a roll-off of approximately 1 MHz was applied to both the A and B phases. This filter essentially conducts high frequency noise to the opposite phase, effectively making the noise common mode and easily rejectable.

Design equations for the LM361 in a 5250 application are shown here for example. The hysteresis voltage, V_H, can be expressed the following way:

$$V_H = V_{RIO} + ((R_{IN}/(R_{IN} + R_F) * V_{OH}) - (R_{IN}/(R_{IN} + R_F) * V_{OL}))$$

where

V_H — hysteresis voltage (V)

R_{IN} — series input resistance (Ω)

R_F — feedback resistance (Ω)

C_{IN} — input capacitance (F)

V_{RIO} — receiver input offset voltage (V)

V_{OH} — output voltage high (V)

V_{OL} — output voltage low (V)

The input filter values can be found through this relationship:

$$V_{CIN} = V_{IN1} - V_{IN2/1} + j\omega C_{IN} (R_{IN1} + R_{IN2})$$

where

$R_{IN1} = R_{IN2} = R_{IN}$

$F_{RO} = \omega/2C$

$F_{RO} = 1/(2C * R_{IN} * C_{IN})$

$C_{IN} = 1/(2C * R_{IN} * F_{RO})$

where

V_{IN1}, V_{IN2} — phase A and B signal voltages (V)

V_{CIN} — voltage across C_{IN}, or the output of the filter (V)

R_{IN1}, R_{IN2} — input resistor values, $R_{IN1} = R_{IN2}$ (Ω)

F_{RO} — roll-off frequency (Hz)

ω — frequency (radians)

The roll-off frequency, F_{RO}, should be set nominally to 1 MHz to allow for transitions at the transmission bit rate. The transition rate when both phases are taken together is 2 MHz, but then both R_{IN1} and R_{IN2} must be considered, so:

$$F_{RO2} = 1/(2C * (R_{IN1} + R_{IN2}) * C_{IN})$$

or,

$$F_{RO2} = 1/(2C * 2* R_{IN} * C_{IN})$$

where $F_{RO2} = 2 * F_{RO}$, yielding the same results.

Table 3-4 shows the range of values expected:

$$F_{RO2} = V_{CIN}$$

TABLE 3-4. Twinax Receiver Design Values

Value	Maximum	Minimum	Nominal	Units	Tolerance
R_{IN}	4.935E+03	4.465E+03	4.700E+03	Ω	0.05
R_F	8.295E+05	7.505E+05	7.900E+05	Ω	0.05
C_{IN}	4.4556E−11	2.6875E−11	3.3863E−11	F	
V_{OH}	5.250E+00	4.750E+00	5.000E+00	V	
V_{OL}	4.000E−01	2.000E−01	3.000E−01	V	
V_{IN}	+1.920E+00	1.000E−01		V	
V_{IN}	−1.920E+00	1.000E−01		V	
V_{RIO}	5.000E−03	0.000E+00	1.000E−03	V	
R	6.533E−03	5.354E−03	5.914E−03	Ω	
F_{RO}	1.200E+06	8.000E+05	1.000E+06	Hz	0.2
V_H	3.368E−02	2.691E−02	2.880E−02	V	
X_C	7.4025E+03	2.9767E+03	4.7000E+03	Ω	

Advanced Features of the BCP

The BCP has a number of advanced features that give designers flexibility to adapt products to a wide range of IBM environments. Besides the basic multi-protocol capability of the BCP, the designer may select the inbound and outbound serial data polarity, the number of received and transmitted line quiesces, and in 5250 modes, a programmable extension of the TX-ACT signal after transmission.

The polarity selection on the serial data stream is useful in building single products that handle both 3270 and 5250 protocols. The 3270 biphase data is inverted with respect to 5250.

Selecting the number of line quiesces on the inbound serial data changes the number of line quiesce bits that the receiver requires before a line violation to form a valid start sequence. This flexibility allows the BCP to operate in extremely noisy environments, allowing more time for the transmission line to charge at the beginning of a transmission. The selection of the transmitted line quiesce pattern is not generally used in the 5250 arena, but has applications in 3270. Changing the number of line quiesces at the start of a line quiesce pattern may be used by some equipment to implement additional repeater functions, or for certain inflexible receivers to sync up.

The most important advanced feature of the BCP for 5250 applications is the programmable TX-ACT extension. This feature allows the designer to vary the length of time that the TX-ACT signal from the BCP is active after the end of a transmission. This can be used to drive one phase of the twinax line in the low state for up to 15.5 μs. Holding the line low is useful in certain environments where ringing and reflections are a problem, such as twisted pair applications. Driving the line after transmitting assures that receivers see no transitions on the twinax line for the specified duration. The transmitter circuit shown in *Figure 3-4* can be used to hold either the A or B phase by using the serial inversion capability of the BCP in addition to swapping the A and B phases. Choosing which phase to hold active is up to the designer, 5250 devices use both. Some products hold the A phase, which means that another transition is added after the last half bit time including the high and low states, with the low state held for the duration. Alternatively, some products hold the B phase. Holding the B phase does not require an extra transition and hence is inherently quieter.

To set the TX-ACT hold feature, the upper five bits of the Auxiliary Transceiver Register, {ATR[3–7]}, are loaded with one of thirty-two possible values. The values loaded select a TX-ACT hold time between 0 μs and 15.5 μs in 500 ns increments.

The connectors called out in the IBM specifications for the twinax medium are too bulky to mount directly to a PC board, therefore a 9-pin D subminiature connector is provided. This connector is then attached to a cable assembly consisting of a 1 foot section of twin-axial cable with the opposite gender 9-pin on one end and a twinax 'T' connector on the other. This is then spliced into the twinax multi-drop trunk.

Miscellaneous Support

Remaining components of the MPA will be covered in the following section, including the board itself, decoupling capacitors, and the Program Counter latches (see *Figure 3-8*).

The MPA is implemented on a four-layer pcb, using minimum 8 mil trace widths and spacing for all signals except the analog traces in the front-end. Here we specified minimal trace lengths and 100 mil trace widths. We additionally requested that parallel analog traces be constrained to similar lengths to minimize phase skew. The analog logic and traces are grouped together near the line interface connectors to minimize interaction with the digital logic. These fairly common analog layout techniques are justified due to the complexity and power level of the analog waveform present in the line interface.

Each device has one 0.1 μF decoupling capacitor located as close as possible to the chip. These are chip capacitors (0.3 spacing, DIP configuration) to minimize lead length inductance and facilitate placement. The $+5$V supply line has two 22 μF electrolytic capacitors, one at each end of the board. The other three supply lines (-5V, $+12$V, -12V) drive only the twinax analog circuitry, and are bypassed with 10 μF electrolytics where they come on to the board and 0.1 μF chip caps at the device(s). The BCP requires additional decoupling due to the large number of outputs, high frequency operation, and CMOS switching characteristics. We used a capacitor on each side of the BCP using 0.33 μF and 0.01 μF values to provide different transient response characteristics. These decoupling capacitors, together with the ground and power planes of the multi-layer board, provide effective supply isolation from the switching noise of the circuitry.

The two Program Counter latches, U1 and U2 (see *Figure 3-8*) are provided for debug purposes only and are not present on the production board. These are 74ALS573s when present and provide a non-intrusive way to monitor the BCP's activity. The Program Counter can always be read by stopping the BCP and accessing the low and high bytes through the Remote Interface (see the DP8344 data sheet), but this affects BCP operation and makes break point capability difficult to implement, particularly given the BCP's interrupt capability. These latches are not considered part of the BCP system and accessing them does not involve the Remote Interface Arbitration System. As shown in Tables 3-1 and 3-2, these latches are mapped into the PC I/O space in the reserved register space of the IBM interface. The REG_DEC PAL, U8, (see *Figure 3-8*) provides the TRI-STATE control for these latches and the PC I/O read strobe controls their latch enables asynchronously with respect to their data inputs. This does in fact present a metastability issue but was deemed an acceptable implementation in order to minimize the control logic overhead associated with this seldom used function.

SECTION FOUR — MPA SOFTWARE ARCHITECTURE

The MPA software is designed to be flexible and powerful. The primary goal of the design was to accommodate multiple interfaces and protocol modes within a single, integrated structure (see *Figure 4-1*). By exploiting the powerful CPU of the BCP, the advanced software tools available for it, and the wealth of development options, the MPA software is much more advanced than commonly found for other microcontrollers. The MPA software is integrated; the system loads all the code for 3270, 3299, 5250, and all the interfaces at once. The system is configured at run time for the different options, or may be reconfigured "on the fly". New tasks may be added to the MPA system easily. The modular organization of the system allows for simple maintenance and enhancement.

PC D(7:0), PC_IOR~

IA(15:0)

74HCT573

PC D7	19	1Q	1D	2	IA15
PC D6	18	2Q	2D	3	IA14
PC D5	17	3Q	3D	4	IA13
PC D4	16	4Q	4D	5	IA12
PC D3	15	5Q	5D	6	IA11
PC D2	14	6Q	6D	7	IA10
PC D1	13	7Q	7D	8	IA9
PC D0	12	8Q	8D	9	IA8
			C	11	PC IOR ~
			OC	1	PCHI OC ~

U2

PC_IOR ~

74HCT573

PC D7	19	1Q	1D	2	IA7
PC D6	18	2Q	2D	3	IA6
PC D5	17	3Q	3D	4	IA5
PC D4	16	4Q	4D	5	IA4
PC D3	15	5Q	5D	6	IA3
PC D2	14	6Q	6D	7	IA2
PC D1	13	7Q	7D	8	IA1
PC D0	12	8Q	8D	9	IA0
			C	11	PC IOR ~
			OC	1	PCHI OC ~

U1

PCLO_OC ~

TL/F/10488–15

MINUS 5V   MINUS 12V   PLUS 12V

| C27 | C1 | C34 | C26 | C28 | C29 | C12 | C11 | C30 | C13 | C25 | C6 | C17 | C16 | C21 | C9 | C15 | C14 | C9 | C7 | C5 | C2 | C24 | C18 | C33 | C22 | C38 | C32 | C31 | C41 | C36 | C40 | C37 | C39 |
| 0.1 μF | 22 μF | 22 μF | 0.1 μF | 0.1 μF | 0.1 μF | 0.1 μF | 0.1 μF | 0.1 μF | 0.1 μF | 0.1 μF | 0.1 μF | 0.1 μF | 0.1 μF | 0.33 μF | 0.1 μF | 0.1 μF | 0.01 μF | 0.01 μF | 0.1 μF | 0.1 μF | 1 μF | 1 μF | 1 μF | 1 μF | 1 μF | 10 μF | 10 μF | 1 μF | 10 μF |

TL/F/10488–16

FIGURE 3-8. Miscellaneous Support

The basic concepts employed in the software design are: modularity, comprehensive data structures, and round-robin task scheduling. The system was designed to allow modules to be written and integrated into the system by different groups. In the case of the National team developing the MPA, different teams developed the 3270 and 5250 software modules. Some modules were set up in advance of any protocol development and have been the basis of the software development. The Kernel.bcp module contains the task switching and scheduling routines. The header files MPA.HDR and DATARAM.HDR contain the basic global symbolic equates and data structures. DATARAM.HDR is organized such that the BCP's data RAM may be viewed through a number of templates, or maps. In other words, except for specific hardware devices mapped into memory, there are no hard coded RAM addresses. The 8k dual-port block is fixed at the top of RAM, and the PC I/O space is mapped into the upper page of installed RAM, but the locations of screen buffers and variable storage are all determined through the set of templates used. The templates serve only to cause the assembler to produce relative offsets. The software developer chooses which base physical address to reference the offset to in order to address RAM. Usually, a pointer to RAM is set up in the IZ register pair, and the data are referenced by the assembler mnemonics:

```
MOVE [IZ+n],rd
```
or
```
MOVE rs,[IZ+n]
```

where:

n is the symbolic template offset.
rs,rd are source and destination registers 0–15, respectively.

This scheme allows the actual locations of data structures to move based on the system mode. This allows the use of the dual-port RAM to change with the interface mode or protocol mode.

The MPA.HDR module is included (via the .INPUT assembler directive) in every module for use in the MPA system, regardless of protocol or interface mode. MPA.HDR defines specific hardware related constants such as RAM size, hardware I/O locations, etc ... MPA.HDR in turn includes MACRO.HDR, which contains commonly used macros, BCP.HDR, which defines specific bits and bit fields for BCP registers, STD__EQU.HDR, which contains the recommended BCP and assembler specific declarations, and DATA RAM.HDR, which contains the RAM templates. Equate files for specific functions such as twinax, coax, and the different interfaces are included where needed. The Kernel module contains the basic software structures which support all system activities. System initialization, scheduling tasks, reconfiguration and halting the system all fall under its jurisdiction. All tasks are called from the Kernel and return to it. The global interrupt enable [GIE] is controlled by the Kernel to allow interrupt handlers to start and end operation. The Kernel must be understood in order to understand the software architecture of the MPA.
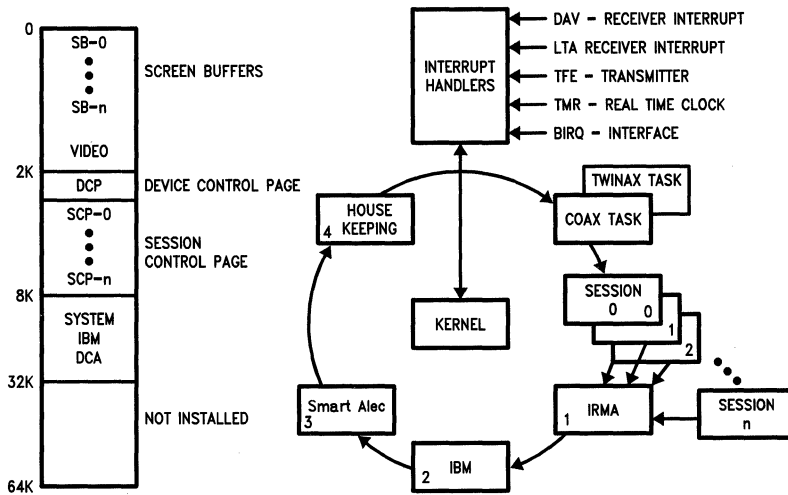
TL/F/10488–17

FIGURE 4-1. MPA Software Architecture

A number of rules have been adhered to during the MPA software development. These can best be discussed by referring to the BCP register allocation shown in *Figure 4-2*. The Main and Alternate banks are used separately for foreground and background functions, respectively. The interrupt handlers are all considered background tasks. All 3270 foreground processing, 5250 command processing, and system functions are foreground tasks. The IZ pointer is not reserved, but is generally used as the SCP pointer by all tasks and interrupts. All tasks are restricted to six levels of nesting to prevent the address stack from overflowing. Interrupt handlers are limited to three levels. Interrupts are generally not interruptable. Some special cases exist, but they are detailed later in this document.

The R20 and R21 registers are permanently reserved for the system. R20 is used as the R_CONFIG storage, or the current configuration as set by the Loader. R21 is the R_TASK register as defined by the Kernel. The Kernel uses this register as its task list, with scheduled tasks signified by their corresponding bits set and unscheduled tasks' bits cleared. Registers R24 and R25 are used to store the PCHI and PCLO address values for fast access by the BIRQ interrupt. At half speed, the BIRQ interrupt handlers need the extra time for processing, and this is a way to decrease overhead.

Kernel

The major part of the Kernel module is a global routine called tasker. Tasker is a round robin, prioritized task scheduler. Each major functional group in the MPA system has a corresponding task that is invoked in this way. All tasks run to completion, meaning that once a task is given control, the task must return to the tasker in order to relinquish control. Interrupt handlers are initialized and masked on and off by their corresponding tasks, although the tasker maintains [GIE], giving it ultimate control over interrupt activity.

The Kernel consists of tasker, schedule_task, and deschedule_task routines. These three combine to allow tasks to be added or removed from the active task list, providing orderly execution and re-prioritization of tasks on the list. All tasks are scheduled by calling schedule_task with the task's identification byte in the selected accumulator. Schedule_task then adds the task to the active task list if it was not already there. If the task is already scheduled, the priority of the task will be bumped if the currently scheduled "next_task" has a lower priority. Priority is implied by the value of the task ID byte. If the next task has a higher ID (i.e., lower ID value) than the requesting task, the task remains scheduled where it is in the task list. The task list is implemented in R21 as discussed above. The highest priority task corresponds to bit 0 of the R_TASK register, the lowest corresponds to bit 7. The list of tasks and their priority in the MPA system is shown in Table 4-1.

TABLE 4-1. MPA Tasks

Task ID	Task Name	Description	Priority
0	cx_task	coax session processor	highest
1	tw_task	twinax session processor	↑
2	ibm_task	IBM interface emulation	|
3	irma_task	IRMA interface emulation	|
4	sa_task	Smart Alex interface emulation	|
6	video_task	video buffer maintenance	↓
7	house_task	system control	lowest

System Initialization

MPA.BCX is the file the Loader loads for MPA system operation. The MPA microcode is started at the sys_main entry point. Just prior to starting the BCP, the MPA_CONFIG register is set for the particular mode passed in from the Loader command line. The program proceeds by initializing the CPU system registers. On the Alternate A bank, the Device Control Register {DCR} is loaded with DCR_CCS, a constant defined in BCP.HDR that establishes the CPU clock at half the oscillator frequency, or OCLK/2; the transceiver sets OCLK. The Interrupt Base Register {IBR} is loaded with INT_BASE, a constant that places the interrupt vector table at 0000. The Fill Bit Register {FBR} is initialized to 0FFh, or zero fill bits. The Auxiliary Transceiver Register {ATR} is cleared to zero for no hold and address zero. {ATR} and {FBR} are initialized here to default known values for completeness; they will be set appropriately by the individual initializers for the different protocols. On the Main A bank, the Interrupt Control Register, {ICR} is set to mask all interrupts off and select the receiver interrupt source to Data Available or Error. Next {ACR}, or the Auxiliary Control Register, sets [BIRQ], the Bi-directional Interrupt Request pin to input mode, clears [GIE], the Global Interrupt Enable bit, assures that [LOR], or the remote host Lock is de-asserted, and configures the timer for CLK/2 operation.

Once the BCP has been configured, the external MPA hardware must be set up. MPA_DATA is cleared to assure no spurious PC host interrupts are generated. The MPA_ACCESS register is read to clear any pending BIRQ interrupt. The entire RAM array is initialized to zero. Note that if a RAM pattern is loaded it will be lost unless this initialization procedure is changed. The MPA_CONFIG register is saved away and restored to the RAM array; all other locations are zeroed.

The tasker gains control at this point with only the HOUSEKEEP task scheduled. When HOUSEKEEP runs, the MPA_CONFIG register is written into R20, the R_CONFIG register, and then is used to call the appropriate task initialization routines. These routines set up any variables needed for the task, initialize interrupt handlers associated with them, and schedule their tasks. For instance, if the MPA_CONFIG register was loaded with 48h, the routine would call cx_init to initialize the 3270 coax task, set up the appropriate interrupt handlers, and schedule itself. Then the irma_init routine would be called; which sets up the interface registers, the BIRQ interrupt, etc . . . When HOUSEKEEP passes control back to the tasker, all applicable tasks are scheduled and interrupts have been unmasked. HOUSEKEEP remains scheduled so that subsequent executions the RAM value for MPA_CONFIG can be compared with R_CONFIG. If a difference is found, the initialization process takes place. If no difference is detected, the task returns directly to the tasker.

Coax Task

Basic 3270 emulation is handled by the cx_task and its associated routines independent of the interface mode configured. The coax routines are set up to exploit the extremely quick interrupt latency of the BCP. Even so, at 9.45 MHz the coax routines are fairly time critical. The basic structure used is divided into two distinct parts: the interrupt handler executes all real time tasks in the background and the cx_task routine handles the four foreground commands of the

ALTERNATE | MAIN

DEVICE DCR	CCS	TCS 1 0	IW 1 0	DW 2 1 0			
BASE IBR	IV15-8						
ADDRESS ATR	HOLD		2 1 0				
FILL FBR	FILL						

DEVICE DCR — CCS | TCS 1 / 0 | IW 1 / 0 | DW 2 1 0
BASE IBR — IV15-8
ADDRESS ATR — HOLD | 2 1 0
FILL FBR — FILL

CCR | RR RW BIRQ N V C Z | CONDITION (FLAG) | R0
NCF | TFE RFF LA LTA DEMI RAR ACK POLL | NETWORK FLAGS | R1
ICR | RISI RISO RSV 4 3 IM2 1 0 | INT | R2
ACR | TST TLD TMC BIC RSV COD LOR GIE | AUX | R3

DATA RTR — DATA | GP0 | R4
STATUS TSR — TFF TA ERR RA DA 10 RF9 8 | GP1 | R5
COMMAND TCR — RLQ SEC SLR ATA OWP 10 TF9 8 | GP2 | R6
MODE TMR — TRES LOOP RPEN RIN TIN 2 PS1 0 | GP3 | R7
ACC' | GP4 | R8
GP5' | GP5 | R9
GP6' — COAX STATE | GP6 | R10
GP7' | GP7 | R11

BACKGROUND ← (HIGH) (LOW) → FOREGROUND

W | LTA VECTOR | R13, 12
X | DATA VECTOR | R15, 14
Y | | R17, 16
Z | SCP POINTER | R19, 18

GP8 | R_CONFIG | R20
GP9 | R_TASK | R21
GP10 | | R22 } SYSTEM
GP11 | | R23
GP12 | PCIO_HI | R24
GP13 | PCIO_LO | R25
GP14 | | R26
GP15 | | R27

(TR HIGH) (TR LOW)
TIMER | R29 | R28 | R29, 28

ISP | ASD 3 2 1 0 DSP 3 2 1 0 | R30
DS | | R31

TL/F/10488-18

FIGURE 4-2. DP8344 MPA Register Map©

2

3270 protocol. The vast majority of decisions and command executions must be carried out "on the fly", or under the auspices of the interrupt handlers. Primarily, the RA/DAV and LTA handlers do the bulk of command execution. See Table 2-1 for the 3270 commands supported.

The SCP_coax template is a reference to the RAM array that locates all the terminal variables, including relative pointers into the screen buffers. Both a Regen buffer and EAB is supported if the MPA_CONFIG register is set for EAB. The refresh_stack is contained on the SCP and is used to store the beginning and ending addresses of buffer modify operations. All WRITE type commands produce buffer modifications, and the compacted information is consumed by the interface tasks.

The cx_task module contains the task initialization routine as well as the task itself. Cx_init sets up the RA/DAV and LTA interrupts and initializes all SCP_coax variables and inter-task communications. The task functions are processing inter-task mail, updating poll status, and processing foreground commands. These foreground commands include SEARCH foreward, SEARCH backward, INSERT, and CLEAR.

The SCP for coax defines registers for each of the 3278 terminal registers, as well as additional ones for control of internal functions. Refer to *Figure 2-2* in Section 2 for the internal structure of a 3278/79 terminal. Initially, the host primary and secondary control registers are cleared, [STAT_AVAIL] is loaded into status_reg, and the poll response is set to POR (Power On Reset). GP6' is dedicated as the coax_state register. It is used to provide fast access to protocol state information such as 3299 address, cursor change, and write in progress status.

The MPA system uses a number of variables to maintain the coax session, including:

coax_stat — system status

mpa_mainstat — main interface control bits, such as clicker and alarm status

mpa_auxstat — auxiliary interface control, such as loopback and on-line/off-line control

mpa_control — status state control, such as POR, key pending, FERR, Operation Complete

old_control — state memory for control, last control value

mpa_auxcontrol — extra status, light pen, slot reader

The initial state of the mpa_auxstat register sets up flags to signal that a new cursor position is available and that the key buffer is empty. Mpa_control is set up with POR state and the status_pending flag set. Status_pending signals other routines which figure new status that the POR state is available. In addition to flags and registers, there are two mailboxes that are used: the sub-task mailbox, and sync_mailbox. Sub-task communicates which foreground coax command to execute from the receiver interrupt handler. Initially this is cleared. The sync-mailbox is the interface communication mechanism and reveals which bits of mpa_mainstat, mpa_auxstat or mpa_control should be cleared by the coax task. This is cleared initially.

The interrupt handlers require two special register pairs for operation that are initialized here. These are DATA_VECTOR and LTA_VECTOR. These vectors are used by the interrupt handlers to keep track of the protocol state that the handler is in; i.e., how to interpret incoming data or what response is necessary. The DATA_VECTOR is initialized to dv_idle, a routine that will throw away any data that it receives, and LTA_VECTOR is set to respond with TTAR if the line drops. The transceiver is reset and initialized for operation with the on-chip comparator, three line quiesces, and 3270 protocol mode. This is accomplished by loading {TCR} with the sum of the equates TCR_RLQ and TCR_SLR, and {TMR} with TMR_3270. The interrupt handlers will be discussed in greater detail in the next section. The screen buffer pointers are set up for MOD 2 size screen and EAB. The refresh stack is set up through a call to rfrsh_init. The cx_task is scheduled, RA interrupt unmasked, and the initialization routine returns to the tasker.

In normal operation, the cx_task routine remains scheduled and the normal execution proceeds in the manner suggested in *Figure 4-1*. The update_poll response uses the value in mpa_control to determine if the session should adjust its status level to the controller. New_status maintains the sync-mailbox and therefore communication with the various interface tasks. If there is mail, new_status reads and executes the interfaces' commands. The state of key_buffer_empty is checked here. It is the mechanism through which keystrokes may be passed from the interfaces to the poll response for transmission to the host controller. A low key_buffer_empty signals that the interface may supply a keystroke. If key_buffer_empty is high, the interface must wait. Key_buffer_empty is cleared by the task when it infers from mpa_control that the previous key has been acknowledged.

The sub-task communication mailbox is checked by cx_task next. If the receiver interrupt handler has decoded a foreground command request from the host controller, the mailbox will be non-zero. The value in the mailbox indicates that either a forward or backward SEARCH, an INSERT, a CLEAR command and its associated parameters are ready for execution. The appropriate foreground routine is then run to completion. The host status_reg is now updated, since completion of a foreground requires an Operation Complete to the host. The poll response is updated again, if necessary, and then the routine relinquishes control to the tasker.

Coax Interrupt Handlers

The coax mode uses four interrupts: RA/DAV, for handling receiver data; TFE, for re-filling the transmit FIFO; LTA, for initiating responses; and BIRQ for host interface notifications. The transceiver interrupt handlers utilize the IW and IX register pairs exclusively. These registers may not be used except by the interrupt handlers themselves. The two pointers are used as vectors in the receive protocol state machine and are called DATA_VECTOR and LTA_VECTOR, respectively. When multi-byte transmissions require more transmit FIFO space than is available, IW is used as TX_VECTOR. The interrupt handlers are all treated as background tasks by the Kernel and must obey all the rules set up by the Kernel for co-existing with foreground tasks. These rules include saving and restoring any registers used except those on the alternate B bank, IW, and IX. Alternate

B is reserved for interrupt handlers. Interrupts must not nest subroutine calls deeper than three levels. Interrupts must lock out remote host access by asserting [LOR] in {ACR} and unlock when finished. No interrupt handler may lock out the remote host for longer than eight microseconds.

The interrupt vector table for 3270 is defined by the INT__ BASE constant in MPA.HDR and is loaded into {IBR} by the sys__init routine. The interrupt table is defined in module INT__PAGE.HDR and is located by the linker. Vector 0 is used as a LJUMP to sys__main for startup.

The RA/DAV interrupt handler entry point in 3270 mode is rxcx__entry, located in the RXCX__INT.BCP module. A flow chart of the basic logic followed can be found in *Figure 4-3*. DAV occurs when the receiver has loaded a valid frame into the receive FIFO or a frame error has occurred. The two main tasks RA/DAV must handle are processing commands and handling data frames. The 3270 data stream and frame format issues bit 11 of the received frame to signify a command if asserted or data if not. The decision to branch to command decoding or data handling is based on this bit. If the received frame is a command, the three immediate decode bits the BCP provides are checked. These bits are hardware decodes of three common 3270 commands: POLL, POLL/ACK, and TTAR. If one of these bits is sent, the command decode thread branches to the appropriate POLL or POLL/ACK command code. If not, the command frame is further decoded by logical device. The 3270 data stream defines different logical devices in a physical terminal such as light pens, magnetic slot readers, etc . . . and addresses them through the upper four bits in the command frame. POLL commands are always assumed to be for the base and use the upper four modifiers for other specific uses, however. The command decodes proceed in separate tables for each of the logical devices. For instance, the base command decode table is reached when the base is addressed. If the EAB is addressed, the EAB command decode table is jumped to. Individual commands are decoded

and the appropriate routines to handle them are jumped to. All the routines vectored to in this way return either to a common return point, rxint__ret, or obey the same rules as rxint__ret in restoring the foreground environment.

The LTA interrupt is asserted when the end of the current receive frame is detected (i.e., when the "line drops"). The LTA interrupt handler is located in the module lta__int.bcp and is entered through lta__int. The LTA__VECTOR is used to branch to the appropriate response routine dictated by the protocol. The host controller expects a response of either status or data within 5.5 μs of the line drop condition. Transmission Turn Around/Auto response is used to acknowledge correct response of the last frame sent, unless the last frame was a command that elicits some data or other status from the terminal. The command processor on the DAV interrupt handler sets up the appropriate vector in LTA__VECTOR for the routine stub to respond with the appropriate data. All stubs return to the lta__ret label for clean up and exit. Refer to *Figure 4-4* for the logic flow diagram of the LTA interrupt handler.

The BIRQ interrupt handler is invoked whenever the remote host writes to the I/O registers mapped into the upper page of the BCP's 32k data memory. The interrupt is handled by the specific routine that is currently configured. The interface discussions will go into detail on this.

The TFE interrupt is un-masked only when a READ__MULTIPLE command from the host has been issued, the BIG__READ mode is selected, and the number of available bytes to send is greater than four. The limitation of four bytes is imposed by the physical transmit FIFO length in the BCP transceiver. The FIFO length is actually three, although when empty, the first byte loaded goes directly into the transmitter allowing three more bytes to be queued. The READ__MULTIPLE command will be covered in detail later. Refer to *Figure 4-5* for the logic flow diagram of the TFE interrupt handler.
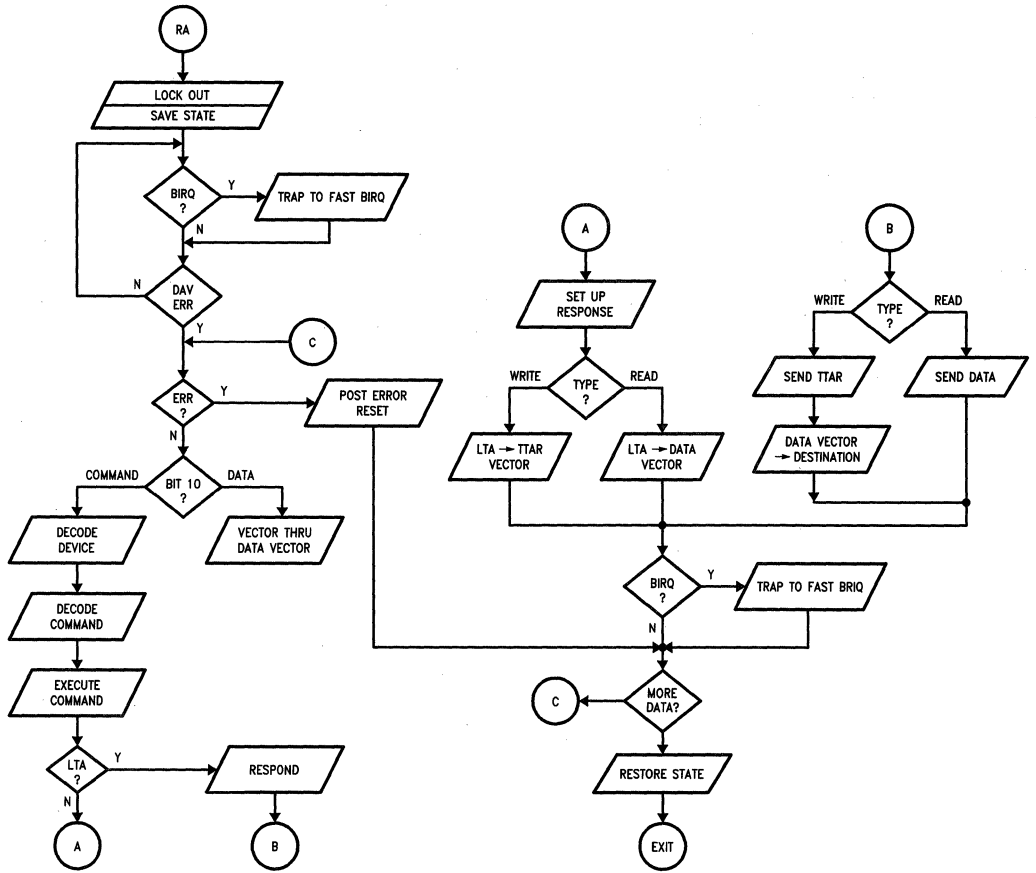
FIGURE 4-3. Coax DAV Handler

TL/F/10488–19
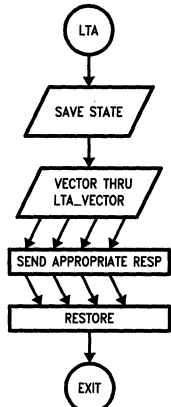
FIGURE 4-4. Coax LTA Vector

TL/F/10488–20
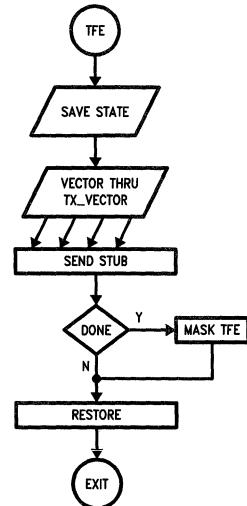
FIGURE 4-5. Coax TFE Interrupt

TL/F/10488–21

TABLE 4-2. Interrupt Handlers

Interrupt	Source	Description	Initializing Task
RA/DAV	Transceiver	Receiver Active/Data Available in receiver fifo	cx_task
TFE	Transceiver	Transmit FIFO Empty	cx_task
LTA	Transceiver	Line Turn Around, line drop	
TMR	Timer	Timer—Real Time Clock	twinax (coax—not used)
BIRQ	EXT Hardware	Remote Host write access	Interface

The basic operation of a session occurs in this way: when the controller sends a command to the emulated session, it is received by the BCP and the RA interrupt fires. The PC host is locked out in the vector table, then rxcx_int is vectored to, where: the alternate bank is selected, and IZ is pushed on the stack. BIRQ is checked, and if set, is branched, DAV is then monitored until it is set. If the interrupt is not because of a receiver error condition, the command/data bit in the 3270 command frame is queried to determine if the incoming frame is command or data. If the frame is a data frame, the IW pointer contains the DATA_VECTOR, i.e., the address of the code expecting the data. The routine is vectored to in order to handle the incoming data. If the frame is a command, it is interpreted and the appropriate action is taken. For commands that expect data, the DATA_VECTOR is set to point to a routine to process that data. All commands require a response of some sort, and this is handled in two ways. When any frame is received, inside the various handlers the LTA condition is checked. If the line has dropped, TTAR or other response is sent immediately. If the line has not dropped, the LTA vector is set to respond with appropriate status/data when it does.

Commands that modify the regen or EAB communicate the change information to consumer tasks (the interface tasks) through a structure called the refresh stack. The refresh stack, which is located on the SCP, is loaded with the beginning and ending addresses of the modified buffer area. This information is used by the interface tasks to determine which locations in the buffer have been modified, without having to read the entire buffer. Refresh stack entries are primarily produced by the DAV interrupt handler, but may also be produced in the foreground by the INSERT and CLEAR commands. When DAV handles a write type command, the beginning address of the write (the address counter) is stored in a temporary storage location called wrt_start. Data associated with the write command are written into the appropriate buffer locations one byte per interrupt (unless interrupt latency has caused the FIFO to fill, then it is emptied before exiting). Since the end of a write command is determined only by the transmission of the next command frame (usually a POLL), the RA/DAV handler remembers that a write has occurred and terminates the refresh stack entry when the next command is decoded.

Poll/Response Mechanism

The POLL and POLL/ACK commands are handled in the cx_basrd.bcp module in routines cx_poll and cx_pack, respectively. The basic functions of the cx_poll routine are to decide if TTAR or special status should be returned to the host and to handle the POLL modifiers in the upper bits of

the POLL command. These modifiers include the terminal alarm and key click control. The determination of which status to send is made after checking mpa_control for the MPA_STAT_PEND bit. If MPA_STAT_PEND is asserted, the poll response variables have new status to send. If no status is pending, TTAR is sent. If the line has dropped (LTA has fired) the response is sent immediately. If not, the LTA_VECTOR is loaded with the appropriate routine stub address and the routine continues on. In either case, the POLL command modifiers are applied to the alarm and clicker status bits in mpa_mainstat.

The POLL/ACK routine immediately checks for LTA and responds with TTAR if LTA is not asserted. The poll response bytes, pollresp_lo and pollresp_hi, are cleared next regardless of other pending status. The mpa_control variable is then checked for the MPA_STAT_PEND asserted condition. If status is pending, the variable old_control is used to clear out the freshly acknowledged status. The LTA_VECTOR is set to lv_ttar whether TTAR was sent initially or not. This serves to keep the response vector set to TTAR just in case the software becomes confused. Update_poll in the CX_TASK.BCP module handles updating mpa_control to reflect new status conditions. This routine updates the pollresp_lo and hi bytes based on the priority of the status in mpa_control. Feature error is the highest priority condition and outstanding status from the light pen or magnetic slot reader is the lowest.

Read Commands

All read type commands to the base are found in the CX_BASRD.BCP module. Each read type command is decoded by the RA/DAV interrupt handler and vectored to the appropriate routine. Each read type command has a corresponding stub to handle LTA interrupt vectors. The most basic read type command is cx_readata. This is invoked upon decoding the READ DATA data stream command. The character pointed to by the address counter is either sent immediately or is set to be sent when the LTA interrupt fires through the lv_readata routine. The addr counter variable is incremented after the character is sent. Both cx_readata and lv_readata use the IW register, which is normally reserved for the LTA_VECTOR, for speed.

Cx_readmul is also found in the CD_BASRD.BCP module and is vectored to when a READ MULTIPLE command is decoded. READ MULTIPLE expects multiple bytes of screen data to be sent within 5.5 μs. The response is initiated inside cx_rdmul if the line has dropped, or lv_rdmul if the LTA vector is executed. The routine has two modes: 4 byte and 32 byte. The default mode is 4 byte and is determined by the state of the LSB in the host secondary control

register. Both modes use the variable addrcounter on the SCP to determine both where to find the data to send and how many bytes to send, up to the 4 or 32 byte limit. In other words, 4 and 32 bytes are the maximum that will be sent to the host. The addrcounter is incremented after sending each byte and terminates the response when the two or five low order bits roll to zero. For "small" reads, i.e., 4 byte mode, the transmitter will accept the maximum amount of data at one time. The transmit FIFO on the BCP will hold up to three bytes after one has been loaded into the transmitter. In "Big Read" mode, if more than 4 bytes are expected, the TX_VECTOR must be initialized. This requires loading the address of the tx_rdmul stub into the TX_VECTOR register and unmasking the TFE interrupt. The cx_rdmul or lv_rdmul routines return after loading the FIFO with 4 bytes. TFE will fire when the FIFO is about to empty giving the tx_rdmul routine only 3.5 μs to reload before the transmission stops. Tx_rdmul can be found in the TXCX_INT.BCP module. The remaining read type commands are all handled similarly. Cx_rach and cx_racl respond with the high and low bytes of the addrcounter variable, respectively. Cx_rdid responds with the terminal ID byte. Cx_rxid responds with TTAR, since it is not implemented. Cx_rdstat responds with the status variable. All these commands check for LTA prior to responding. If LTA has occurred, the responses are sent immediately. If not, the LTA_VECTOR is set up to vector to the lv_stubs. The cx_rdid routine does additional processing, however. The status conditions OPERATION COMPLETE and FEATURE ERROR are cleared by reception of the READ ID command. All these routines are found in the CX_BASRD.BCP module.

Write Commands

All write type commands to the base are found in the CX_BASWR.BCP module. Commands are decoded by the RA/DAV interrupt handler and vectored to this module at the cx_addresses. Each write command has an associated dv_stub for handling incoming data. The routines load the DATA_VECTOR with the appropriate stub before exiting.

Cx_write and its data vector stub dv_write are responsible for writing data into the screen buffer, starting refresh stack entries, and setting the BUFFER_BEING_MODIFIED semaphore. The semaphore is used to lock out the IBM emulator from reading the dual-port screen buffer. This semaphore would have been more logical to include in the interface code itself, but the "real time" nature of BUFFER_BEING_MODIFIED means that any time lag is too much. When the next command is decoded, the refresh stack entry is terminated, and the BUFFER_BEING_MODIFIED bit is cleared. The dv_write stub is very critical in that very large blocks of data may be sent to the device through the routine and cumulative interrupt latency effects may become significant. To address this, the dv_write routine always empties the receive FIFO. Since the time spent on the interrupt may be longer than usual, the BIRQ interrupt flag is checked for activity. A pending BIRQ might hold the PC host off long enough to undermine its memory refresh needs. If the PC's memory is not refreshed periodically, the memory will develop parity errors and halt. If BIRQ is pending, a special "fast" BIRQ routine is called and then the routine exits.

Other write type commands found in the CW_BASWR.BCP module include the initial stubs for the foreground commands; SEARCH FORWARD, SEARCH BACK, INSERT,

and CLEAR. All these commands are initially decoded and vectored here in real-time. When their associated parameters are received, the foreground commands are scheduled through the sub-task communication mailbox. All the foreground commands cause the terminal to set NOT_AVAIL status (busy) in the status register. All four respond with TTAR to acknowledge reception of the command and parameters cleanly.

All the other write commands load variables on the SCP corresponding to registers in the emulated terminal, or cause some controlling action in the terminal. These include the low and high bytes of the address counter, the mask value for CLEARs and INSERTs, the control registers and resetting the terminal. Cx_reset calls the host_reset routine that re-initializes the SCP variables to the POR state. The screen buffers are not cleared. The START OPERATION command causes a vector to the cx_start routine and returns TTAR.

Foreground Commands

The foreground routines are all executed by cx_task when the sub-task communication mailbox is filled with the appropriate value. These are tk_insert, tk_clear, tk_sforward and tk_sback. The routines are found in the CX_COM.BCP module along with other local support routines.

EAB Commands

The EAB commands are found in the CX_EAB.BCP module. Read and write type commands addressed to the EAB feature are included here. The number of commands for the EAB feature are small enough that they are logically grouped together in one module, as opposed to the base commands. Some of the more complex commands from a performance standpoint are addressed to the EAB feature. WRITE ALTERNATE, WRITE UNDER MASK, and READ MULTIPLE EAB require the most real time bandwidth of any coax function.

The READ MULTIPLE EAB command is the same as its base counterpart except for two features: it functions with the EAB exclusively and, if the Inhibit Feature I/O step bit in the Control register is set, the cursor will not be updated after the read. WRITE ALTERNATE receives a variable length stream of data that is written in the base and EAB alternately. The WRITE UNDER MASK command uses data associated with the command, the EAB byte pointed to by the cursor register, and the EAB mask to modify the contents of the EAB. The algorithm is quite strange and is best described by the code. Please refer to eab_wum and dv_wum for specifics on the command implementation.

IRMA Interface Overview

IRMA is a member of a family of micro-to-mainframe links produced by Digital Communications Associates. It provides the IBM PC, PC XT, or PC AT with a direct link to IBM 3270 networks via a coaxial cable connection to an IBM 3174, 3274, or integral terminal controllers with type "A" adapters. The IRMA product includes a printed circuit board that fits into any available slot in IBM PCs and a software package that consists of a 3278/79 Terminal Emulator program, called E78, and two file transfer utilities for TSO and CMS environments. Also included in the software are BASICA subroutines useful in developing other application programs for automatic data transfer.

The 3278/79 Terminal Emulator provides the user with all the features of a 3278 monochrome or 3279 color terminal.

The IRMA file transfer program provides all the functions required for the successful transfer of files under the TSO or CMS IBM mainframe software packages. Also included in the IRMA software package are many other features such as program customization, keyboard reconfiguration, independent and concurrent operation, ASYNC Character Support, and PC clone support.

As discussed in the introduction, the IRMA product was a forerunner in the 3270 emulation marketplace and quickly gained wide acceptance. DCA made a considerable effort in documenting the interface between IRMA and its PC host. As a result this interface has become one of the industry standards used today. So it is only natural that this interface be used on the DP8344 Multi-Protocol Adapter to highlight the power and versatility of the DP8344 Biphase Communications Processor. The MPA hardware with the MPA soft-loadable DP8344 microcode is equivalent in function to the DCA IRMA board with its associated microcode. Both directly interface with the IRMA software that runs on the PC (E78, file transfer utilities, etc.) providing all functions and features of the IRMA product. The following sections describe the hardware interface and the BCP software in the Multi-protocol Adapter Design/Evaluation kit that is used to implement the IRMA interface. All of the following information corresponds to Rev. 1.42 of the IRMA Application software.

Hardware Considerations

The IRMA printed circuit board plugs into any normal expansion slot in the IBM PC System Unit. It provides a back-panel BNC connector for attachment by coaxial cable to a 3174, 3274, or integral controller. IRMA operates in a stand-alone mode, using an on-board microprocessor (the Signetics 8x305) to handle the 3270 protocol and screen buffer. Because of the timing requirements of the 3270 protocol, the on-board 8x305 operates independently of the PC microprocessor. The 8x305 provides the intelligence required for decoding the 3270 protocol, managing the coax interface, maintaining the screen buffer, and handling the data transfer and handshaking to the System Unit (PC microprocessor).

COAX

DP8340   DP8341

8X305   RAM

DUAL
REGISTER
FILE

PC BUS
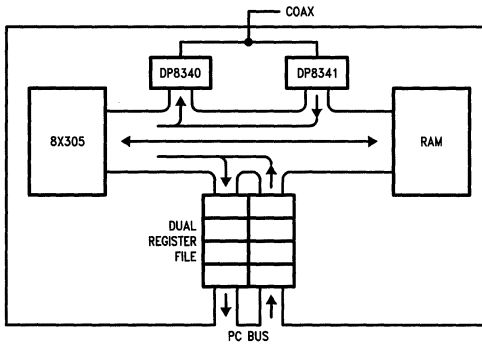
TL/F/10488-22

FIGURE 4-6. IRMA Hardware Block Diagram

The IRMA card uses National Semiconductor's DP8340 and DP8341 3270 coax transmitter and receiver (respectively) to interface the 8x305 to the coaxial cable. The DP8340 takes data in a parallel format and converts it to a serial form while adding all the necessary 3270 protocol information. It then transmits the converted data over the coax in a biphase encoded format. The DP8341 receives the biphase transmissions from the control unit via the coaxial cable. It extracts the 3270 protocol specific information and converts the serial data to a parallel format for the 8x305 to read.

The card contains 8k of RAM memory for the screen buffers and temporary storage. The screen and extended attribute buffers use approximately 6k of this memory. The remaining memory space is used by the 8x305 for local storage. A block diagram of the IRMA hardware is shown in *Figure 4-6*.

The hardware used in enabling the 8x305 to communicate with the PC's 8088 processor is a dual four-byte register array. The 8x305 writes data into one of the four byte register arrays which is read by the 8088. The 8088 writes data into the other four byte register array which is in turn read by the 8x305. The dual register array is mapped into the PC's I/O space at locations (addresses) 220h–223h.

A handshaking process is used between the two processors when transferring data. After the 8088 writes data into the array for the 8x305, it sets the "Command Request" flag by writing to I/O location 226h. The write to this location is decoded in hardware and sets a flip-flop whose output is read as bit 6 at location 227h. When the 8x305 has read the registers and responded with appropriate data for the 8088, it clears this flag by resetting the flip-flop. A similar function is provided in the same manner for transfers initiated by the 8x305. Here the flag is called the "Attention Request" flag and can be read as bit 7 at location 227h. This flag is cleared when the 8088 writes to I/O location 227h.

The Multi-Protocol Adapter printed circuit board also plugs into any expansion slot in the IBM PC System Unit. Like the IRMA card, it provides a back panel BNC connector for attachment by coaxial cable to a 3174, 3274, or integral controller. The MPA operates in a stand-alone mode, using the DP8344 Biphase Communications Processor to handle the 3270 protocol and screen buffer. Again, because of the timing requirements of the 3270 protocol, the BCP operates independently of the 8088 microprocessor of the System Unit. As with the 8x305, the BCP provides the intelligence required for decoding the 3270 protocol, managing the coax interface, maintaining the screen buffer, and handling the data transfer and handshaking to the System Unit. However, with the BCP's higher level of integration, it also directly interfaces with the coaxial cable. The BCP has an internal biphase transmitter and receiver that provides all the functions of the DP8340 and DP8341. However, unlike the 8x305, the DP8344's CPU can handle the 3270 communications interface very efficiently. To illustrate this fact, the CPU clock frequency on the MPA card is set at 9.45 MHz instead of its top speed of 20 MHz. This allows slower, less expensive RAM to be used on the board, if desired.

The MPA card contains a single 32k x 8 RAM memory device for the screen buffers and temporary storage. This memory size was chosen for the 5250 environment, where the BCP can handle up to seven sessions. In the IRMA mode, only 8k of memory is accessed. The MPA hardware block is shown in *Figure 4-7*.

2
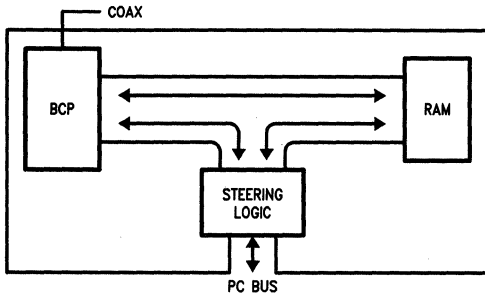
TL/F/10488–23

FIGURE 4-7. MPA Hardware Block Diagram
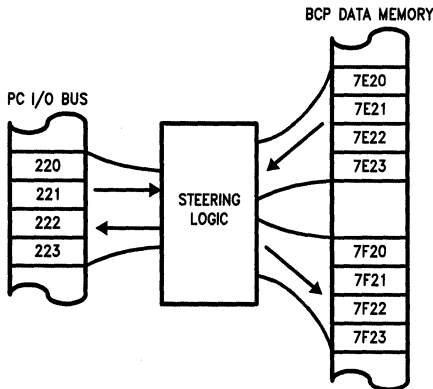
TL/F/10488–24

FIGURE 4-8. MPA Register Array Implementation

The hardware used to enable the BCP to communicate with the PC's 8088 processor is steering logic (contained in PALs) and the BCP's data memory. In a typical application, the BCP communicates with a remote processor by sharing its data memory. This is true with the MPA, but because the MPA must run with the IRMA software, steering logic was used to direct the 8088's I/O reads and writes of the IRMA dual register array locations (220h–22Fh) into the data memory on the MPA card. By using data memory instead of a discrete register file the component count was reduced. The IRMA software requires that a "dual" register file be used (or in this case emulated). Therefore the writes from the 8088 are directed to memory locations 7F20h–7F23h and the reads from the 8088 are sourced from memory locations 7E20h–7E23h. The MPA Register Array Implementation is shown in *Figure 4-8*.

The handshaking process is still used when the BCP and the 8088 are transferring data. When the 8088 goes to set the command flag by writing to I/O location 226h, it actually does a write to 7F26h in the MPA's data memory via the steering logic. The steering logic interrupts the BCP telling it an access has been made to the IRMA I/O space. The BCP then determines if it was a write to the PC I/O location 226h by reading a byte of data from the steering logic. If a write occurs to I/O location 226h, the BCP sets bit 6 in the MPA memory location that the PC's 8088 will read as its

I/O location 227h. In the case of the "Attention Request" flag, the BCP will set this flag by simply setting bit 7 in the memory location which the 8088 reads as I/O 227h. The clearing of this flag by the 8088 is done in a similar fashion as the setting of the "Command Request" flag. Note that each time the 8088 writes to an I/O location between 220h and 22Fh the BCP is interrupted. However specific action is taken only on writes to 226h or 227h. With all other locations the BCP simply returns from the interrupt service routine once it has determined the 8088 did not write to I/O 226h or 227h. This approach to the hardware was chosen to minimize the discrete logic on the MPA card by taking advantage of the power of the 8344's CPU to handle some tasks that were typically done with hardware in the past.

IRMA Microcode

The IRMA application software written for the personal computer (E78, file transfers, etc.) is designed around a defined interface between IRMA and the System Unit (the 8088 and its peripheral devices). The hardware portion of this interface is discussed above. The software portion of this interface is the microcode written for the 8x305 processor. When the software and hardware are viewed as one function, it is referred to as the Decision Support Interface (DSI). All of the IRMA application software was written around this interface. When configured in the IRMA mode the MPA becomes the DSI. The method of communication between the DSI and the System Unit will be discussed briefly in the next section. A more exhaustive discussion on this interface is given in the IRMA Technical Reference.

The DSI and the System unit communicate through the dual four-byte register array. The System Unit issues commands to the DSI by writing to this array. This register array is viewed by the System Unit as four I/O locations (220h–223h). Each I/O location corresponds to one eight-bit word. When the System Unit issues a command the first byte, word 0, is defined as the command number. The next three bytes, word 1 through word 3, are defined as arguments for the command. The number of arguments associated with an individual command varies from zero to three. Sixteen commands have been defined for the DSI. These commands allow the System Unit program to read and write bytes in the screen buffer, send keystrokes, and access special features available on the DSI. To begin a command the System Unit program sets byte 0 equal to the command number and provides any necessary arguments in byte 1 through byte 3. It then sets the Command request flag. The Command Request flag is continually polled by the 8x305 processor when it is not busy managing the higher priority 3270 communications interface. When it detects the setting of this flag by the System Unit, it reads the data from the register array and executes the command. Once the command has been executed, the 8x305 will place the resulting data into the other side of the register array and clear the Command Request Flag (see *Figure 4-9*). The System Unit program has been continually polling this flag and after seeing it cleared reads the result from the register array. The Command Request flag can only be set by the System Unit. This is done by a write to I/O location 226h. The Command Request Flag can only be cleared by the DSI's 8x305.
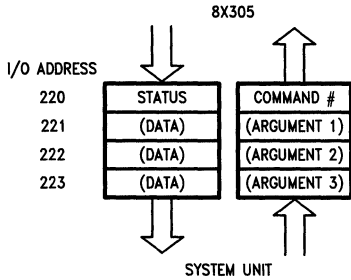
FIGURE 4-9. Command and Response Locations in the IRMA Register Array

The DSI can not issue commands to the System Unit but it can inform the System Unit of a status change. If a status change occurs in a status bit location when the corresponding attention mask bit is set, the 8x305 will set the Attention Request flag. This flag can be polled by the System Unit and is viewed as bit 7 in the I/O register at address 227h. The System Unit can clear this flag by executing a write to I/O location 227h. As is the case with both flags, the action of writing to the specific I/O location clears or sets the flags, the data written during the write have no affect. In typical operation the Attention Request flag is not used; however, it is implemented on the MPA. The current status of both flags can be read by both processors. The System Unit does this by reading I/O location 227h. The resulting eight bit number has the Attention flag as bit 7, the MSB, and the Command flag as bit 6. The other bits are not used.

MPA Implementation

The IRMA interface on the MPA board operates essentially in the same manner as described above. The System Unit I/O accesses to the IRMA register array space are transferred to two areas in the BCP's data memory (see *Figure 4-10*). One location is for System Unit reads of the array (7E20h–7E23h), the other is for System Unit writes to the array (7F20h–7F23h). Different BCP memory locations are used because the register array on the IRMA card actually contains eight byte wide registers (four for System Unit reads and four for System Unit writes) in hardware. E78 was written to make the best use of this hardware design and in doing so it may write a new command and/or arguments be-

fore it reads the results of the old command. Therefore if just four memory locations were used, E78 would read back part of a new command it had just written and interpret this as data from the DSI from the previous command.

The Command Request and Attention Request flags are implemented using LS74's on the IRMA card, hence the setting and clearing by writing to 226h and 227h (this clocks or clears the associated flip-flop). This function is implemented on the MPA using an external PAL and the bi-directional interrupt pin, BIRQ. If there is a write to the IRMA/IO space 220h–22Fh, the ACC_REG PAL issues an interrupt to the BCP via the BIRQ input. The BCP reads the other outputs of that PAL to determine which of the sixteen locations has been written to. If it is 226h or 227h then the appropriate bits are set or cleared in the "IRMA read location" (7E27h) in the BCP data memory. The BIRQ interrupt is generated only on System Unit I/O writes to 220h–22Fh but this also includes writes to the dual register array. If a write to 220h–223h occurred, the BCP irma birq interrupt routine simply clears the interrupt and takes no further action.

The commands from the System Unit are executed in the irma task routine. This routine is a foreground, scheduled task in the MPA Kernel. The irma task routine first updates both the main and auxiliary status registers as defined by the DSI (see *Figure 4-11*). It then looks at the state of the command request flag in memory to determine if there is a command pending from the System Unit. If so, it reads the command number and the arguments from the BCP's data memory and executes the command. The task then places the results back in the data memory in the appropriate location (7E20h–7E23h). After this is complete the task clears the command request flag and returns program control to the Kernel.

There are three separate code modules used to allow the MPA to emulate the DSI.

1. Power-up initialization routine
2. BIRQ interrupt routine
3. irma task routine

These three routines will be discussed in the following section. For clarity, the term "irma" is capitalized when referring to DCA products and lower case when referring to the MPA software that was written to emulate the IRMA DSI. *Figure 4-12* gives a graphical representation of where these routines fit into the software architecture of the MPA.
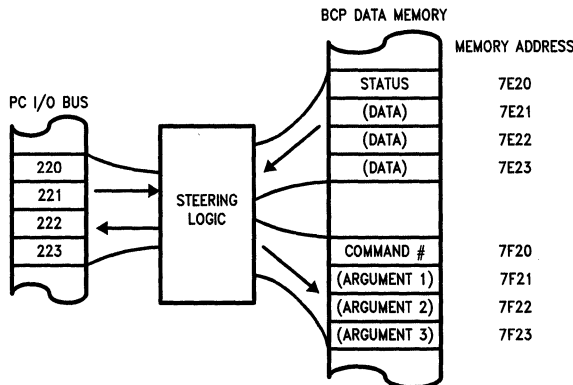
FIGURE 4-10. Command and Response Locations in the MPA Register Array

Main Status Byte		Auxiliary Status Byte	
Bit	**Meaning**	**Bit**	**Meaning**
7 (MSB)	Aux Status Change Has Occurred(*)	7 (MSB)	Unused
6	Trigger Occurred(*)	6	Unit Polled Since Last Status Read
5	Key Buffer Empty	5	Sound Alarm
4	Fatal IRMA Hardware Error(+)	4	Display Inhibited
3	Unit Reset by Controller	3	Cursor Inhibited
2	Command Interrupt Request(+)	2	Reverse Cursor Enabled
1	Buffer Modified(*)	1	Cursor Blink Enabled
0	Cursor Position Set, or Search Backward(*)	0	Keyboard Click Enabled

(*) Bits which must be cleared by user program

(+) Bits which will never be set in MPA implementation

FIGURE 4-11. IRMA Main and Auxiliary Status Byte Definition
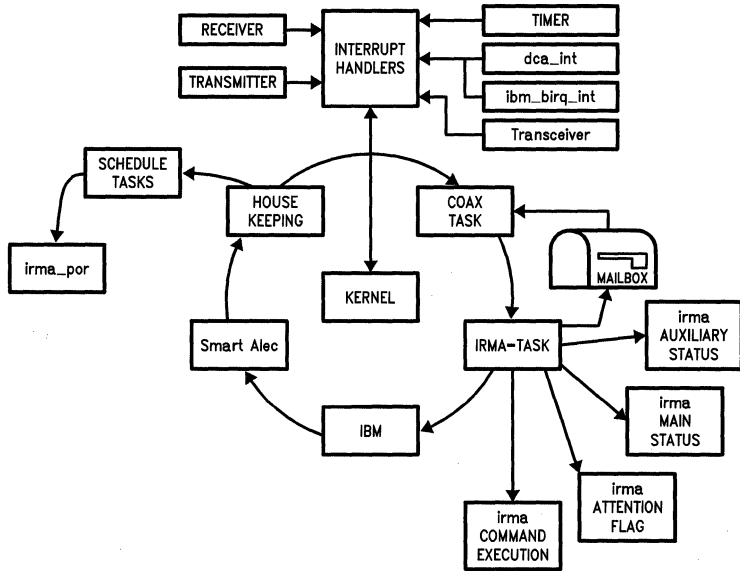
TL/F/10488–27

FIGURE 4-12. MPA Software Block Diagram in IRMA DSI Emulation Mode

MPA irma Power-Up Initialization Routine

The irma power up initialization routine is called by the housekeeping task if it detects that the irma bit was set in the MPA configuration register. The irma initialization routine is titled irma_por in the MPA source code. This routine initializes the memory locations and DP8344 internal registers that are used by the irma emulation code. It also unmasks the BIRQ interrupt and schedules the irma_task in the MPA Kernel. The first memory location initialized is the Command Request and Attention Request flag byte, which is location 7E27h in the BCP's data memory. The data at location 7E27h is passed to the System Unit by the steering logic when the System Unit reads I/O location 227h. This byte is set to zero by the irma_por routine even though only bits 6 and 7, the command and attention request flags respectively, are used. The irma_por routine also initializes the memory locations that the irma_task routine uses to store the trigger variables and the attention mask. These are initialized so that the trigger bit or Attention Request flag will not be set.

The irma_por routine also initializes internal BCP registers. It does this because other routines, such as the dca_int interrupt routine, must access certain stored values very quickly to keep execution time short. The execution time in these routines is decreased if data needed in the routine are kept in internal registers rather than in data memory. For example, the value of the high byte of the address page of the "IRMA read registers" is stored in register GP14. In the BIRQ interrupt routine, the Z index register needs to point to that address page. This is done in the routine with a single 2 T-state instruction which moves the contents of GP14 to the high byte of the Z index register. If the value of the high byte of the address page was in memory, it would take a 3 T-state move to an immediate addressable register followed by a 2 T-state move to the Z index register. The irma_por routine initializes the registers GP14 and GP13 with the "IRMA_read register" page memory address and GP15 with the status flag register with the command request flag always set. The final function of the irma_por routine is to schedule the irma_task routine. This is done by loading the task number into the accumulator and calling the schedule_task routine. After this, program control is returned to the tasker.

DCA_INT Interrupt Routine

The second code module required to emulate the IRMA DSI is the dca_int routine. On the IRMA card, the Command Request and Attention Request flags are implemented in hardware. This implementation requires a number of discrete components to decode the System Unit I/O addresses 226h and 227h and to provide the set and clear function of these flags. The MPA board, on the other hand, uses extra CPU bandwidth to reduce the discrete components needed to provide the Command Request and Attention Request flag function. It does this by letting the CPU decode part of the System Unit I/O access address and provide the set and clear function of these flags. The BCP code necessary for this is the BIRQ interrupt routine whose source module is labeled dca_int. The BIRQ interrupt is generated when the System Unit writes to any I/O locations from 220h to 22Fh. It would have been more expedient in this case to only have interrupts generated on writes to I/O locations

226h and 227h. However, the MPA hardware also supports the IBM emulation programs. The MPA implementation for the IBM interface requires interrupts to be generated from more System Unit I/O access locations, so to reduce external hardware, interrupts are generated for a sixteen byte I/O block. This flexibility of hardware design further illustrates usefulness of the extra CPU bandwidth of the DP8344.

When the BCP detects the BIRQ interrupt, it transfers program control to the dca_int routine. The function of this routine is to set the Command Request flag if the System Unit wrote to I/O location 226h or clear the Attention Request flag if the system unit wrote to I/O location 227h. This must be done as quickly as possible because the System Unit will begin to poll the Command Request flag very soon after setting it. In fact, to make sure the System Unit cannot poll the flag before the BCP has set it, the interrupt routines lock out System Unit accesses until BIRQ has set the flag. The 3270 protocol timing requirements place another time constraint on this routine. Because this is an interrupt service routine, all other BCP interrupts are disabled upon entering. This means the coax interrupts will not be acknowledged until they are re-enabled by the program. To meet this critical timing constraint, the dca_int routine execution time must be as short as possible. The routine reads the ACC_REG PAL to acquire the information needed to determine which register the System Unit actually wrote to. Reading this PAL clears the external BIRQ interrupt signal. It then determines which I/O locations the System Unit wrote to by using the JRMK instruction and a jump table. If the write was to 226h then the Command Request flag is set. Now the routine only has to restore the environment (registers used in interrupt routines are pushed on the data stack and must be restored before leaving the interrupt service routine) and return to the foreground program. If the write was to I/O location 227h, the routine clears the Attention Request flag. It then restores the environment and returns program control to the foreground program. Finally, if the write was to any other of the sixteen locations, the environment is restored, and program control is returned to the foreground task.

There is a section of code in the dca_int routine that does the same function as that described above, but is called from the coax interrupt routine and not by the external birq interrupt. To increase performance, the transceiver interrupt handlers check the BIRQ flag in the CCR register before they return to the foreground task. If the flag is set, they call the dca_fast_birq section of the dca_int routine. Here the same operations as described earlier are performed except for the saving and restoring of the environment. The dca_fast_birq routine does not have to provide this function because the coax interrupt routine does it. This substantially decreases the number of instructions executed and therefore improves the overall performance.

MPA irma Task Routine

The majority of the DSI emulation takes place in the irma_task routine. This routine is run in the foreground as a scheduled task. Therefore the decision to execute this routine is dependent only on the MPA's task scheduler and is not impacted by the System Unit. In reality the task is run many times between System Unit accesses because the code execution speed of the BCP is greater than that of the

8088. Therefore the most current information and status is always available to the System Unit. The irma task routine, appropriately labeled in the source code as "irma_task" contains four sections. These sections are the auxiliary status, main status, Attention Request flag, and command execution routines.

The auxiliary status routine, called irma_aux_status in the source code, gathers and formats the information required to produce the auxiliary status byte as defined by the DSI. This routine is implemented in the irma_task routine as a subroutine. It gets the necessary status information from two pre-defined memory locations which contain general coax information placed there by the coax routine. These memory locations are labeled MPA_MAINSTAT and CONT_REG in the source code. The auxiliary status routine first moves the MPA_MAINSTAT byte from data memory into an internal register. It masks off the unwanted bits and combines the register with the contents of the CONT_REG memory location, which is also loaded into an internal register from data memory. The routine then loads the previous value of the auxiliary status byte from data memory. This value was saved from the previous time the task was executed and is required when determining the main status byte. The routine then stores the new value of the auxiliary status register in that same data memory location. The new auxiliary status byte is maintained in register GP6 for the remainder of the irma task.

The main status routine, called irma_main_status in the source code, gathers and formats the information required to produce the main status byte as defined by the DSI. This routine is also implemented in the irma task as a subroutine. The information required to determine the main status is gained partly from the pre-defined MPA_MAINSTAT byte, however, three of the status bits must be generated by this routine. These are the "Aux (auxiliary) Status change has occurred" bit, the "trigger occurred" bit, and the "buffer modified" bit. The "Aux Status change has occurred" bit is generated by comparing the old and new auxiliary status bytes from the auxiliary status subroutine. If the values are different the bit is set. If the values are identical, the bit is left in its previous state. It is not cleared because this bit can only be cleared by a DSI command from the System Unit. The "trigger occurred" bit is set if a trigger data match occurs. The System Unit program can define an address location in the screen buffer and a corresponding data byte. If the data byte is found at that location in the actual screen buffer, the trigger occurs. The System Unit program can look for any number of bits in the data byte to match by applying a mask value. It can look for a change of state in the data byte by specifying a mask value of all zeros. The trigger mask, address location and data byte values are stored in the BCP's data memory and are set by two of the defined DSI commands. The main status routine gets these values from memory and checks the screen buffer to see if the trigger bit should be set. Actually, this function is rarely used in the IRMA System Unit software. The "buffer modified" bit is generated by checking the MPA's action stack pointer. If the pointer is non zero the main status routine resets the stack and sets the "buffer modified" bit. As with the "Aux status change has occurred" bit, the "key buffer empty", "Unit reset by controller", and "buffer modified" bits in the main status register must be reset by the System Unit program. Therefore the main status subroutine logically "ORs" these bits with their previous value. Two bits defined

by the DSI in the main status register are always left cleared by the main status routine. These are the Fatal IRMA hardware error and the command interrupt request bits. After the main status byte has been generated, it is kept in register GP5 for the remainder of the irma task. The main status routine also loads the previous value of the main status from data memory and stores the new value in that same location.

The Attention Request flag routine, called "irma_atten_flag" in the source code, determines if the Attention Request flag should be set as defined by the DSI. This routine is also implemented in the irma task routine as a subroutine. This routine compares the old main status value with the new main status value. If it detects that a bit in the old register was a zero and the corresponding bit in the new main status register is a one, it will compare this bit position to the attention mask. If the attention mask also has a "1" in that bit position the Attention Request flag will be set in the appropriate location in data memory. The attention mask is loaded from the BCP's data memory and its value is set by one of the sixteen defined DSI commands. The routine also saves this flag information in the internal copy of the IRMA status flag register in GP15. Once this is complete, the routine returns to the main body of the irma task.

The final section of the irma task is the command execution routine which is called "irma_command_decode" in the source code. This routine, like the others, is implemented in the irma task routine as a subroutine. However unlike the other routines, it is not executed every time the irma task is run. The System Unit program must have requested that a command be executed or the irma task will skip the command execution routine and return program control to the task scheduler. The irma task determines this by checking the Command Request flag in the IRMA status flag register at memory address 7E27h. If this bit is set the irma task calls the command execution routine.

The command execution routine begins by determining which of the sixteen commands is to be executed. This is done by moving the command number data byte at memory address 7F20h into an internal register. It then uses the JRMK instruction and a jump table to transfer program control to the specific routine that corresponds to that command number. The individual command routine then loads any required command arguments from data memory locations 7F21h–7F23h and executes the command. The resulting data is placed in the data memory locations 7E20h–7E23h with the IRMA main status byte always in the first location (7E20h). The command execution routine then clears the Command Request flag in the data memory. After this it returns to the main body of the irma task routine.

The sixteen commands defined by the DSI are thoroughly documented in the IRMA Technical Reference. The implementation of each command in the command execution routine is well documented in the corresponding section of source code. For reference, the commands and the associated source code routine labels are given in Table 4-3.

As mentioned earlier, the MPA software uses a synchronous method of passing some status information between tasks. This is necessary because the status information can be updated on both foreground and interrupt routines. In this case the updating of such status information must be synchronized between the routines or the data could be cor-

TABLE 4-3. IRMA DSI Commands and the Corresponding MPA Source Code Labels

IRMA DSI Commands		MPA IRMA Command Source Labels
Code	Command Definition	Source Code Label
0	Read Buffer Data	irma__com__read__buffer
1	Write Buffer Data	irma__com__write__buffer
2	Read Status/Cursor Position	irma__com__status__cursor
3	Clear Main Status Bits	irma__com__clr__mstatus
4	Send Keystroke	irma__com__send__keystroke
5	Light Pen Transmit	irma__com__lpen__transmit
6	Execute Power-On-Reset	irma__com__por
7	Load Trigger Data and Mask	irma__com__trig__data__mask
8	Load Trigger Address	irma__com__trig__addr
9	Load Attention Mask	irma__com__attn__mask
10	Set Terminal Type	irma__com__set__term
11	Enable Auxiliary Relay	irma__com__aux__relay
12	Read Terminal Information	irma__com__read__term
13	Noop	irma__com__noop
14	Return Revision ID and OEM Number	irma__com__rev__oem
15	Reserved—Do Not Use	irma__com__reserved

rupted. The synchronizing method is a "mailbox" in memory where the location of the status information and the change required is placed. The irma task uses the sync__mailbox to tell the coax task when to reset the "cursor change", "key buffer empty", "unit polled since last status read", and "unit reset by controller" status bits. The irma task also uses the mailbox to tell the coax routine that the System Unit has instructed the MPA to execute a Power On Reset sequence on the coax. The irma task accumulates the status change information in register GP2 throughout the routine (more specifically the cursor change reset from the main status routine and the others from the command execution routine). It then loads the mailbox just before returning to the task scheduler.

IBM Interface Overview

The IBM Personal Computer 3278/79 Emulation Adapter uses sixteen I/O mapped locations, PC interrupt level 2, and 8k of re-mappable shared RAM to provide the necessary hooks to do 3278/79 terminal emulation, 3287 printer, and DFT emulation. The PC emulation software reads and writes to the I/O locations to determine session status and reads the screen buffer maintained in the shared RAM when screen updates are made by the host. The shared RAM concept and use of a PC interrupt make the speed of the terminal emulator very fast and efficient.

The IBM Adapter card uses a gate array, PALs and various logic chips to manage the interface and coax sessions. A block diagram of the IBM adapter hardware is shown in *Figure 4-13*. The sixteen I/O locations reserved for the interface appear to be registers physically resident in the gate array located on the IBM Emulation Adapter card. The addresses of the sixteen I/O locations are 2D0h–2DFh. PC register addresses along with their corresponding read and write capabilities are defined in Table 4-4. The PC accesses the registers in four different modes of operation which are: 1) read only, 2) write only, 3) read/write, and 4) read/write with reset mask. The first three modes are self explanatory. The read/write with reset mask mode means that the PC reads the value of the register as a normal I/O read to acquire the addressed byte of information. After reading the byte, the PC will write a mask with ones in bit positions that

the PC wishes to clear. This "write with reset mask" is usually used as an acknowledgment that the byte was read by an earlier read. The resulting contents of the register will be cleared in bit positions that were written with corresponding ones. A brief description of each register and its function follows. For a detailed discussion on each register, refer to the *IBM Technical Reference for the Advanced Adapter* (see References in Appendix C).

PC Adapter Interrupt Status

The Interrupt Status register contains six interrupt flags and two status bits. The interrupts are set based on events occurring on the coax. If the interrupts are enabled in the adapter control register (2D4h), the PC interrupt level 2 is set when one of the five interrupt conditions occur. The buffer-being modified status flag is set when the screen buffer is being modified by a WRITE DATA, a CLEAR, or INSERT command. The interrupt status flag is set whenever any interrupt has been set. The register is read/write with reset mask by the PC as defined above. Acknowledging that an interrupt has been read as set, the PC will write back to the register with a one in the corresponding bit location that was read. The PC write of a one clears the interrupt. The write with reset mask scheme provides a clean handshake between the two asynchronous systems.

Visual/Sound Register

The Visual/Sound register contains control settings for the terminals that are affected by the load control register command, clicker status, and alarm status. This register is a PC read/write with reset mask with a different twist. Any value written to this register results in the clearing of the alarm bit only. Other bits are not affected by the PC write. This arrangement is interesting in that the host and/or the PC emulation program can clear the alarm status of the terminal being emulated.

Cursor Address Low and High

The Cursor Address registers contain the sixteen bit cursor value owned by the host. This register is read only by the PC and provides the location of the current cursor position.
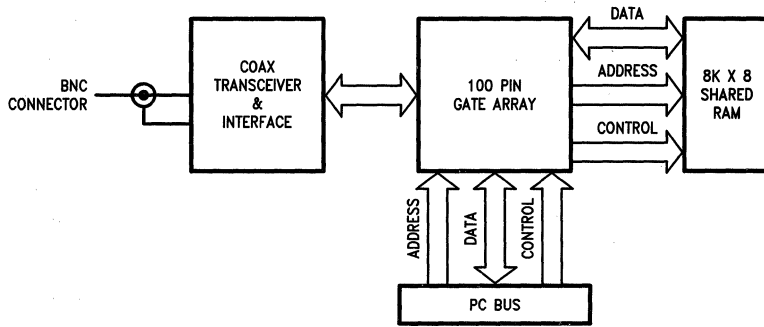
FIGURE 4-13. IBM Hardware Implementation

TL/F/10488-28

TABLE 4-4. PC Register Address Locations and Read/Write Functionality

Address	PC Register	PC Read	PC Write
02D0	PC Adapter Interrupt Status	Data	Reset Mask
02D1	Visual Sound	Data	Reset Alarm
02D2	Cursor Address Lo	Data	—
02D3	Cursor Address Hi	Data	—
02D4	PC-Adaptor Control	Data	Data
02D5	Scan Code	—	Data
02D6	Terminal ID	—	Data
02D7	Segment	—	Data
02D8	Page Change Lo	Data	Reset Mask
02D9	Page Change Hi	Data	Reset Mask
02DA	87E Status	Data	Reset Mask
02DB–02DF	Reserved		

PC Adapter Control Register

The Adapter Control register determines the mode of operation of the adapter (i.e., 3278 terminal, 3287 printer, or DFT emulation), controls keystroke passing with a bit used as a handshake, and controls the masking of interrupts. The remaining bits control various operation situations (i.e., enabling/disabling the coax session, keystroke wrap testing etc.). This register is read/write by both the PC and the adapter software. This function makes synchronization of reads and writes critical to ensure no data is lost.

Scan Code Register

The Scan Code register, as the name implies, is where scan codes are written by the PC corresponding to the keystrokes struck on the keyboard. This register is PC write only and the byte written is the one's complement of the scan code to be sent to the host.

Terminal Id Register

The Terminal Id register is write only by the PC and should not be changed once the terminal has gone on line. The value written is the one's complement of the keyboard Id and model number of the terminal that will be requested by the host when initializing the session.

Segment Register

The Segment register is used for relocation of the memory segment at which the adapter recognizes a memory read or write from the PC. The default value is CE. This register is write only by the PC.

Page Change Low and High Registers

The Page Change registers are used to communicate the change in the screen buffer. Each bit corresponds to a 256 byte block of the 4k screen buffer and is set by the adapter hardware when any screen modification occurs. The register is read/write with reset mask by the PC as described earlier.

87E Status Register

The 87E status register contains status flags relevant to 3287 printer emulation. Included is a flag for the alarm and operation condition of the printer. The register is read/write with reset mask by the PC as described earlier.

The Multi-Protocol Adapter Solution

The Multi-Protocol Adapter (MPA) card has the ability to emulate the IBM Personal Computer 3278/79 Emulation Adapter allowing the PC emulation to be run using the MPA hardware in place of the adapter card while maintaining close to the same functionality. To emulate the adapter, the MPA utilizes the power of the DP8344 BCP to handle the coax session and interface maintenance in software. *Figure 4-14* gives a block diagram of the MPA hardware.

The I/O registers described above are maintained in a shared RAM located on the MPA board and the BCP software must "fake out" the PC software when any register update is made leaving the correct value in the RAM for the next access. To emulate the function of the I/O registers, the MPA hardware sets the bi-directional interrupt pin
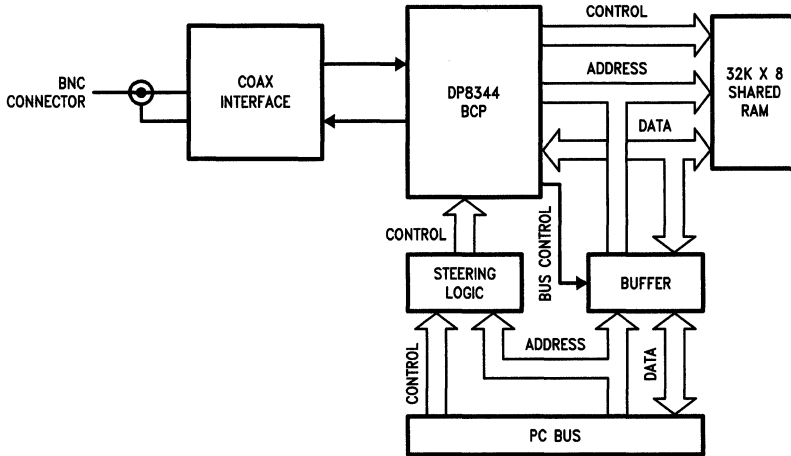
FIGURE 4-14. MPA Implementation

TL/F/10488–29

TABLE 4-5. I/O Registers

PC I/O Address		BCP CODE Variable Address
		Absolute RAM address = PCIO value + offset value
		offset values:
02D0	Interrupt Status	ibm–isr
	Local Copy	ibm–Lisr
02D1	Visual/Sound	ibm–vsr
	Local Copy	ibm–Lvsr
02D2	Cursor Address Low	ibm–cursorlo
02D3	Cursor Address Hi	ibm–cursorhi
02D4	Adaptor Control	ibm–control
02D5	Scan Code	ibm–scan
02D6	Terminal Id	ibm–id
02D7	Segment	ibm–segment
02D8	Page Change Low	ibm–pagelo
	Local Copy	ibm–Lpagelo
02D9	Page Change High	ibm–pagehi
	Local Copy	ibm–Lpagehi
02DA	87E Status	ibm–status
	Local Copy	ibm–Lstatus

TL/F/10488–30

2

(BIRQ) low on any PC write to the reserved I/O locations. The write to the I/O location is routed into reserved locations in the shared RAM. The mapping of the I/O registers in the shared RAM is shown in Table 4-5. The BCP Code Variable Address column in Table 4-5 shows the variables used in the MPA source code to form the absolute RAM address of the I/O register contents. The PCIO value is a sixteen-bit value and is the base pointer into the page of memory where the I/O registers reside. The variables listed are added to the PCIO base to form the absolute address pointer to the specified register in data memory. All registers that are cleared by the write under mask scheme have duplicate copies that are maintained solely under BCP control to allow software implementation of the write under mask handshake.

The BCP software to handle the interface and coax routine contains interrupt driven routines as well as foreground routines. A block diagram showing the code arrangement used to handle the IBM interface and coax session is shown in *Figure 4-15*. Four blocks run as tasks while the interrupt sources are used where immediate attention is required (i.e., the communication with the host [DAV interrupt] and the PC interface maintenance [BIRQ interrupt]). The three sections of code that will be discussed below are responsible for initializing the I/O registers at power up, maintaining the I/O registers, and setting/clearing the PC level 2 interrupt. Each routine is described in the paragraphs that follow.

IBM—Initialization

The ibm—init routine initializes the I/O registers to the expected state at power up and clears the 4k screen buffer in preparation for a new session. After clearing the screen buffer, the program schedules the ibm—task routine as a task to the Kernel routine and unmasks the BIRQ interrupt to enable the ibm—birq—int routine to run when the PC writes to the reserved I/O registers. This code is only executed when the card initially runs at power on time or when

forcing an interface reset. The Power On Reset values of the I/O registers are defined in the IBM.HDR file and are set upon assembling the routines. Upon completion of this and other initialization routines, the PC emulation software can be started to bring the emulator resident. If there is no activity on the PC interface or coax, the BCP runs the kernel, the ibm—task routine, and the coax routine in sequence. All PC interface writes and coax activity are handled by interrupts.

IBM—BIRQ Interrupt Routine

The BIRQ routine is unmasked by the ibm—init routine as mentioned above. The BIRQ input goes low (asserted) when the PC writes to the reserved I/O locations. BIRQ is unaffected by PC reads of the I/O locations since no action is required by the MPA board. BIRQ is set high when the BCP acknowledges the interrupt by reading from the mpa— access section of data memory (locations 8000h to 9FFFh).

The BIRQ routine is responsible for performing the write with reset mask function for updating the I/O registers defined in the interface. To perform the write with reset mask operation, local copies of the five write with reset mask registers are maintained. When the PC writes out the mask to a particular register, the contents of the register are destroyed and written over by the mask in the RAM. The BIRQ routine first determines which register was written to by reading the access register in data memory. The contents of the access register is decoded to determine which register was written to. If the register written uses the write with reset mask operation, the mask is loaded into a general purpose BCP register. Then, the local (old) copy of the register being modified is pulled in, modified, and placed back in the appropriate I/O location in correct form for future PC accesses. Bit locations that have a one in the mask value just written are cleared. The updated value is then written by the BCP back to the register and the local copy of the register is updated as well.
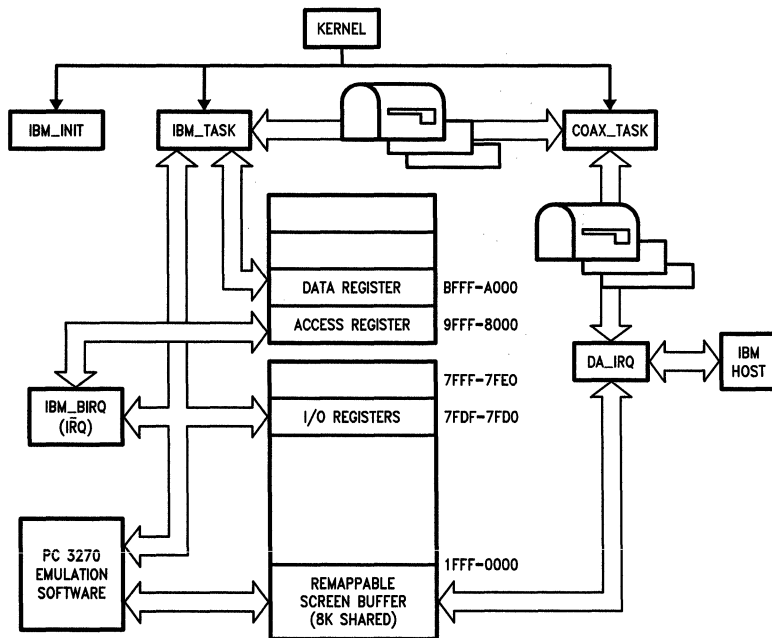
FIGURE 4-15. IBM Interface Code Block Diagram

TL/F/10488–31

For PC writes to non-write with reset mask registers, the BIRQ routine updates any variables for the coax routines that are affected and then exits. An example of this would be a write of the terminal id into the terminal id register. The BIRQ routine complements the value written and places it in a variable that the coax routine will use to respond to read terminal id commands.

The BIRQ routines have to be fast to ensure that the modified I/O register is in the correct state before the next PC read or write of the modified register and also to free up time for the transceiver interrupt routines that are of utmost priority to maintain the coax communication. For speed considerations, any write to the interrupt status register results in the clearing of the PC level 2 interrupt in the BIRQ routine. Modifications of the PC level 2 interrupt are done by writing to the mpa__data section of data memory (locations A000h to BFFFh). Writing with the AD7 pin low clears the PC interrupt, while writing with AD7 high sets the PC interrupt. The PC interrupt is set in the ibm__task routine (IBM__TASK.BCP) if interrupts are pending and not disabled.

The BIRQ routine, upon being called, asserts the lock out remote bit in the Auxiliary Control Register so that subsequent PC accesses are waited until the current update has been processed. This technique ensures that the I/O updates are synchronized, eliminating the possibility of destroying data.

IBM__TASK Foreground Routine

The ibm__task routine runs in the foreground and is called as a task by the Kernel. The ibm__task is enabled to run by the ibm__init routine. Once it has been scheduled by the initialization routine, the ibm__task runs any time it is called by the kernel.

The primary purpose of the ibm__task routine is to keep the I/O registers current as to the state of the session so the PC software can update the screen in a timely manner. The ibm__task routine maintains communication with the coax task routine via a two byte mailbox in data memory. The ibm__task routine monitors activity on the coax through bit settings in the mailbox variables (mpa__mainstat and mpa__auxstat) and updates the I/O interrupt status register, visual sound register, PC adapter control register, and PC interrupt level accordingly. The task is non-interrupt driven and uses both main banks of the CPU for processing.

The ibm__task first handles the refresh stack which contains the beginning and ending addresses of screen buffer modifications. The refresh stack is implemented in data memory and is modified by calling the rfrsh__push and the rfrsh__pop subroutines. Values are pushed onto the refresh stack in the coax routine as modifications are made to the screen buffer. To determine if there have been modifications, a variable called rfrsh__cntr is maintained and its contents tell how many 4 byte deep entries are currently on the stack. The stack itself is 80 bytes deep thus allowing 20 entries before overflowing. In the event of an overflow, rfrsh__cntr is set to FFh and a call to refresh__pop will reinitialize the stack. The ibm__task routine uses the information on the stack for determining what bits to set in the I/O page change registers. Each bit in the page change registers corresponds to a page of the screen buffer and is set when that particular page is modified. The page registers get updated each time the ibm__task runs to communicate modifications back to the PC. Bits in the page change registers are cleared by a write with reset mask as described in the BIRQ routine.

After the page change registers have been updated, the ibm__task routine reads in the current value of the visual/sound register and the PC adapter control register for mode determination. The routine branches to one of three different sections depending on the MPA mode of operation (3278 terminal, 3287 printer, or DFT emulation). At this point, mpa__mainstat and mpa__auxstat variables are loaded and as mentioned before, bits in the variables determine what updates to make in the I/O registers. Bits in mpa__mainstat determine whether to do a power on reset, tell when a POLL command is received, communicate the state of the alarm and clicker, provide a handshake bit for keystroke passing, communicate when the cursor has moved due to a load address counter command decode, communicate when a RESET command is decoded, and communicate when the control register has been loaded by the host. Bits in aux__stat communicate when the card is enabled for operation, when the adapter card is in test mode for keystroke wrap testing, when the reset cursor bit is set in the PC adapter control register for printer emulation, and when either a READ ID, START OPERATION, or DIAGNOSTIC RESET command is decoded. After scanning bits that affect the interface, variables are passed back to the coax routine by placing a value in the mailbox (sync__mailbox location) and placing toggle under mask variables (sync__data1 and/or sync__data2) back to the coax routine to alter or acknowledge bit settings. The value put in the mailbox tells the coax routine which of the status variables need to be altered as a result of the ibm__task routine's execution. The toggle under mask variables are exclusive ORed with the mask variables and passed by the ibm__task routine when the coax routine runs. This approach ensures a clean handshake between routines. All bits are under the control of the coax task routine so the ibm__task routine will write a zero back to the read only bit positions and write a one back to read/write bit locations only when the state of the bit is requested to toggle.

Upon returning control to the kernel, the I/O registers have been updated to reflect the current state of the emulation. Also, the PC interrupt level 2 is set if the interrupts are pending in the interrupt status register.

Twinax Task

The twinax task tw__task (located in module TW__TASK.BCP) is responsible for directing twinax terminal emulation. It monitors all seven internal twinax sessions for current polling status, for 2 second Auto-POR time-outs, and for 5 second POR OFFLINE time-outs. In addition, tw__task invokes the twinax command processor, tw__session (located in module TW__SESS.BCP), for each twinax session that requires attention.

When the MPA__CONFIG register is set (or changed) to select twinax emulation, the task housekeep calls tw__init (located in module TW__TASK.BCP) to initialize the twinax routines, and then calls tw__sa__init (located in module SA__INIT.BCP) to initialize the Smart Alec interface routines. The routine tw__init initializes the hardware interface

2

for twinax, initializes and unmasks the twinax receiver interrupt, initializes and unmasks the transmitter interrupt, initializes and unmasks the timer interrupt, initializes the twinax dependent Device Control Page (DCP) variables, and initializes all seven Session Control Pages (SCPs) for twinax emulation. The initialization of everything except the SCPs is straight forward; the appropriate bits and bytes are simply set to their required values. The initialization of the SCPs are a bit more complicated, however, with the following steps performed for each SCP. First, the SCP is filled with "55" hex (as a debugging aid). Second, tw_por (located in module TW_CNTL.BCP) is called, which initializes the twinax dependent SCP variables, except for those set by the Smart Alec interface routines (i.e., Model ID, Keyboard ID, etc . . .). Third, tw_init takes each session out of POR since a true POR has not been requested yet. (A true POR can only be performed on an active session.) After the SCPs are initialized, tw_init schedules the twinax task tw_task to run under the Kernel. It is tw_task's job to direct twinax emulation in the foreground. tw_init then returns control to housekeep, which in turn calls tw_sa_init. The tw_sa_init routine initializes the memory locations and internal registers that are used by the Smart Alec emulation code. This is discussed in detail in the Smart Alec Interface Overview section later in this chapter. Housekeep then enables interrupts and returns control to the Kernel's tasker with the twinax emulation and interface tasks now scheduled to execute.

The monitoring functions performed by tw_task break down into two groups: ONLINE sessions, those sessions which are configured by the Smart Alec emulator (attached) and seen by the host 3x system; and OFFLINE sessions, whose sessions not configured by the Smart Alec emulator (unattached) and therefore not seen by the host 3x system. ONLINE (configured) sessions are monitored for current polling status, Auto-POR time-outs, and POR complete time-outs. Current polling status simply indicates whether the physical address for a session is being polled at least once every 2 seconds. When this is false, tw_task clears the line active indicator for that session. (The System Available indicator status is monitored by the Smart Alec interface task.) An Auto-POR time-out occurs when tw_task determines that 2 seconds have elapsed since the last poll to a physical address. The task tw_task requests that the session attached to that physical address perform a POR. It then schedules the session in question so that the request will be processed. (Scheduling sessions is discussed in the following paragraph.) POR complete time-outs occur when tw_task determines that 5 seconds have elapsed since a given session initiated a POR. It is tw_task's responsibility to bring the session ONLINE by signaling the receiver interrupt handler to start responding to and accepting commands from the host 3x system. OFFLINE (non-configured) sessions are only monitored for current polling status.

After every internal session has been checked by the monitor, tw_task invokes the twinax session command processor, tw_session for each scheduled session. (This action is similar to the Kernel's tasker.) Both background and foreground tasks schedule sessions when they require a session to perform some sort of action. For example, a session is scheduled when a new command is placed onto the internal command queue, or when another task, such as the Smart Alec interface task, requires a session to POR. The task tw_task calls the twinax command processor, tw_session, and passes a pointer to the SCP of the scheduled session. The command processor then performs the requested action and/or executes the command(s) in the internal command queue.

When all the sessions have been checked and all the scheduled sessions have been processed by the command processor once, tw_task returns control to the Kernel's tasker.

Twinax Interrupt Handlers

The twinax mode uses four interrupts: DAV for handling receiver data; TFE for all responses; TIMER for handling response window timing and as a real time clock for 5250 protocol requirements; and BIRQ for host interface accesses. All interrupts except BIRQ are unmasked in the tw_init routine after initialization requirements for each have been executed. The BIRQ interrupt is unmasked in the sa_init routine. As with the coax interrupt routines, the twinax interrupt routines can use the alternate B bank registers without having to save and restore them. The twinax DAV and TFE interrupt routines are set up as state machines whose current state is stored in the "DATA_VECTOR" and "TX_VECTOR" memory locations. IW and IX are reserved for the TX_VECTOR and DATA_VECTOR addresses that point to the appropriate state in the TFE interrupt and DAV interrupt routines, respectively. The TFE routine always expects TX_VECTOR to be set appropriately upon entry. DAV loads the DATA_VECTOR from memory upon reception of the first frame of a message and uses IX directly for frames 2−n. Also, GP5 on alternate B bank has been reserved for DAV, TFE, and TIMER interrupt routine usage. The name of this register is "R_STATE" since it is used primarily by the receiver for station address information and protocol control.

Twinax Receiver Interrupts Routine

The DAV interrupt routine is responsible for decoding the commands sent by the controller, loading commands on the internal processing queue, stuffing data in to the regen buffer, "OFFLINE" address activity determination, maintaining protocol related real time status bits, and supporting all seven station addresses if necessary. A flow diagram of the DAV interrupt routine is shown in *Figure 4-16*.

Initialization requirements of the DAV interrupt are:

1. R_STATE (GP5 on alternate B) set to
   TW_RSTATE_INIT;

2. tw_level_cnt set to TW_LEVEL_INIT;

3. tw_busy_cnt set to TW_BUSY_MAX.

The host is locked out upon entry, i.e., [LOR] is set; this is done on the interrupt page prior to the branch to the twinax receiver interrupt handler. The main A and the alternate B bank of registers are then selected and IZ is saved so that it can be restored upon exiting the interrupt. Since the DAV interrupt source is an "OR" of both the reception of a valid data frame and the flagging of an error by the receiver, a check for an error is done first to make this distinction. (Error handling will be discussed later in this section.)
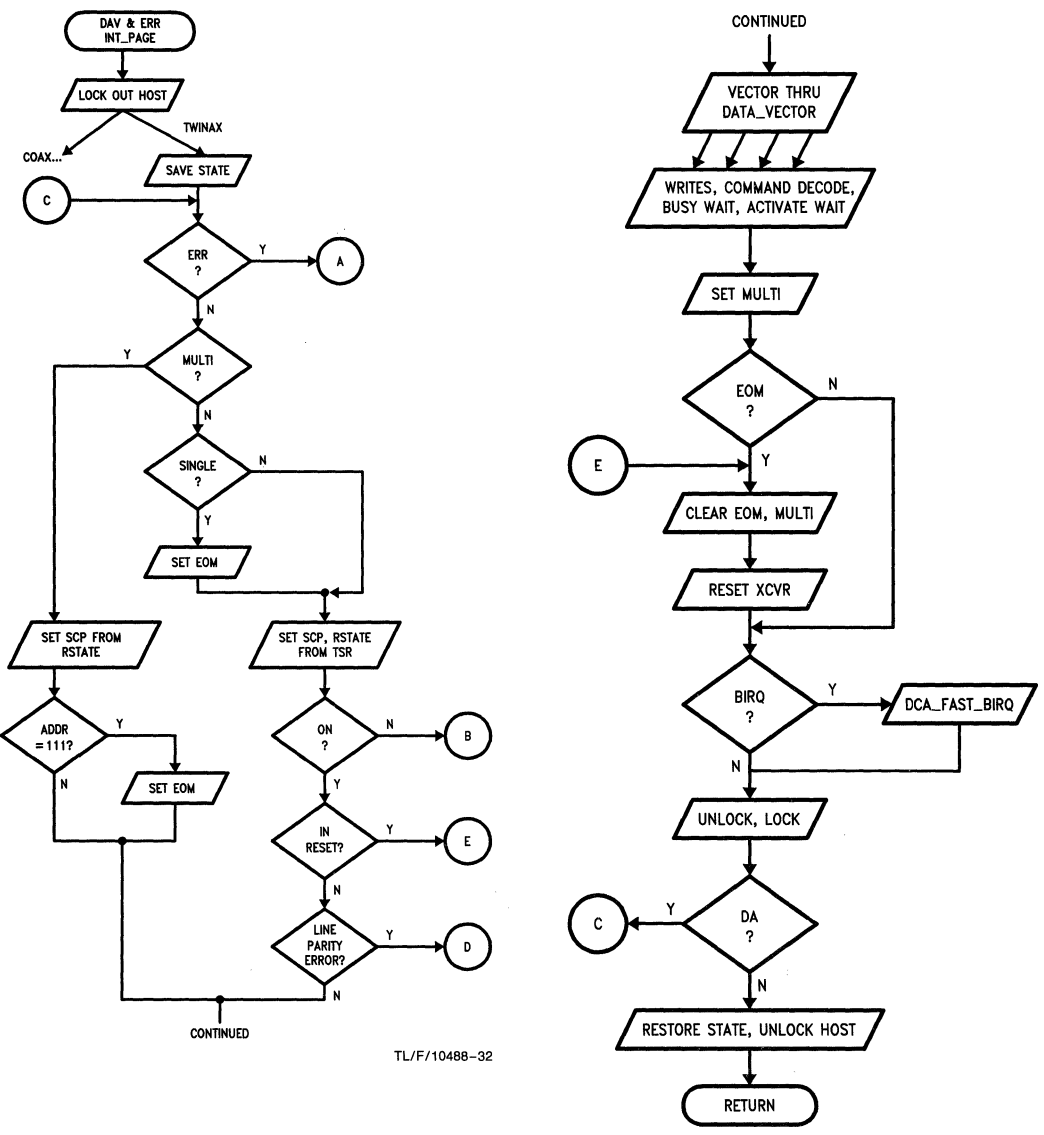
TL/F/10488–32

TL/F/10488–33

FIGURE 4-16. Twinax DAV Handler

2
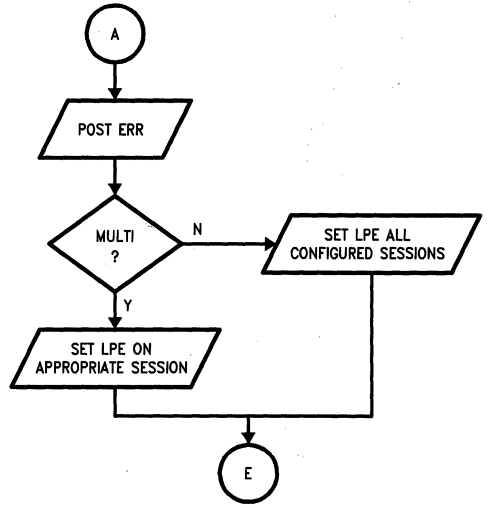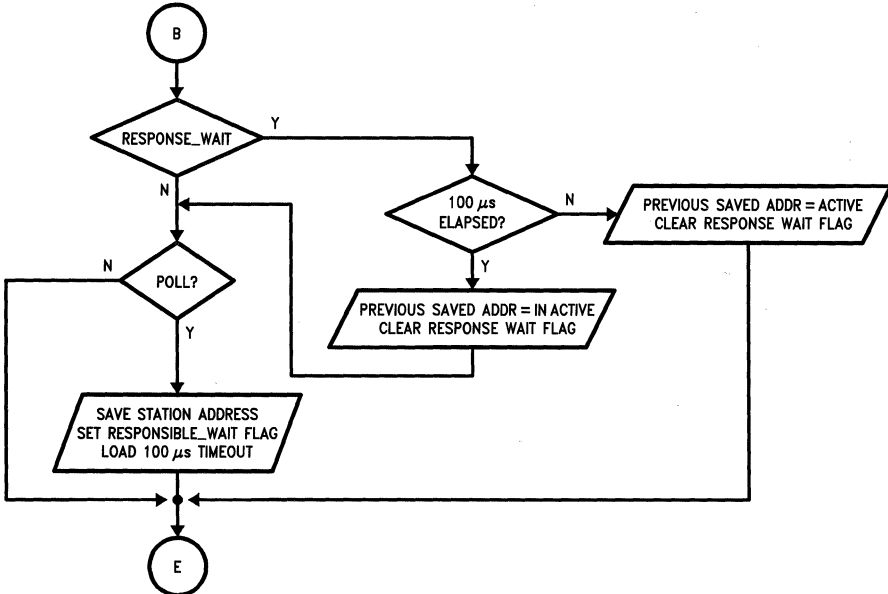
FIGURE 4-16. Twinax DAV Handler (Continued)

TL/F/10488–34

TL/F/10488–36

TL/F/10488–35

A pivotal point in the routine is controlled by a flag set in R__STATE called RX__MULTI which is set after processing the first frame of a multiframe message. The purpose of RX__MULTI is to ensure that the received station address is only sampled on the first frame of each message from the controller and causes the DAV interrupt routine to search for the "111" end of message delimiter on all subsequent frames received. The station address saved in R__STATE[2–0] will be used by the receiver for setting the SCP pointer on all subsequent frames of the multiframe message. When the end of message is detected, the flag RX__EOM is set in R__STATE. If RX__EOM is set at exit time, then RX__MULTI and RX__EOM will be reset along with the transceiver to ensure that any errors flagged by the receiver logic of the BCP resulting from a noisy line after the transmission of the fill bits will be ignored. If RX__MULTI is not set, the data received is either the first frame of a multiframe message or a single frame command. In this condition, the station address is placed in R__STATE[2–0] and IZ is set to point to the SCP page of memory corresponding to the station address. RX__EOM will get set here only if the data is a single frame command, which is determined by the state of RTR[0] (bit 14, see 5250 PAI). The station address received is the "physical station address" and should not be confused with the "logical station address" which is used solely by Smart Alec for aesthetics. The physical station address is loaded into bits 8–10 of the sixteen-bit SCP pointer. This scheme provides 256 bytes of data memory for emulating each station address.

Once the SCP pointer has been established, the receiver interrupt must know if the station address of the data received is currently being emulated ("ONLINE") or is not being emulated ("OFFLINE"). Addresses that are offline have to be monitored for activity to inform Smart Alec whether or not the address can be attached as an online session in the future (see OFFLINE section for line activity determination).

When the session is ONLINE, checks are made upon reception of the first frame of the message to see if the session is currently in a reset state or if a line parity error is pending. For subsequent frames of the message, no checks are made for reset or pending line parity errors, although each frame is still parity checked. The reset state is determined by the RX__RESET flag stored in tw__rxtx__status on each SCP page. When the reset flag is set, all data is ignored. The line parity error state is needed since once a line parity is detected, only POLL commands are processed by the terminal until the error condition is cleared. The error is cleared when a POLL is received with the Reset Line Parity Error bit set in conjunction with the terminal being in the nonbusy state. (See POLL discussion in 5250 PAI.)

If the terminal is not in a reset condition and no line parity error is pending, the DATA__VECTOR is loaded to determine what state to branch to. The DATA__VECTOR must be stored on the SCP page due to the multi-session nature of twinax. When the first frame of a message is received, the IX index register is loaded from the SCP tw__data__vectorhi and tw__data__vectorlo locations prior to the indexed jump to the appropriate processing state. For frames 2–n of a message, IX is used in its current state for processing speed since it is reserved for the interrupt and is already set accordingly.

Command/Data Processing Routines

There are basically four states used in the DAV interrupt routine: 1) command decode, 2) writes, 3) busy__wait, and 4) activate wait. Each state is vectored to via an indexed jump using the DATA__VECTOR as discussed above. However, when exceptions are detected by the foreground command processing routines, the DATA__VECTOR is modified.

The command decode state, as the name implies, is where the received byte is decoded and pushed onto the 16-byte internal processing queue as specified in the 5250 protocol. Commands are decoded first by checking to see if the command is a POLL. Next, two jump tables are used to further decode the command. One table is used for commands addressed to features (i.e., RTR[7] = 1) and only the lower four bits of the command are decoded. The other jump table processes all commands in base format so the lower five bits of the command are decoded. No distinction is made as to what device is addressed since this is done by the foreground tw__session routine when the command is unloaded from the queue. The only commands that can have duplicate meanings in this scenario are the END OF QUEUE and RESET BASE since they are identical in the lower five bits of the commands. They are further processed before being loaded onto the queue to handle this overlap.

Once the command is decoded, it is loaded onto the queue by the QUE__LOADER routine which will be discussed later. Since commands may or may not have associated operands with them, the DAV interrupt modifies DATA__VECTOR appropriately for the command just decoded. Single frame commands do not change the DATA__VECTOR from command decode since there are no operands associated with them. This is not true for the end of queue command as it results in the DAV routine moving into the busy__wait state which will be discussed later. Commands that have associated operands with them, for example LOAD ADDRESS COUNTER, set the DATA__VECTOR to the rx__operands routine and a frame count value is maintained on the SCP (tw__frame__cnt) to control how many additional frames to stay in the rx__operands state for processing the entire command packet. Some commands require special routines to process them. The READ and WRITE IMMEDIATE commands set DATA__VECTOR to rx__imm__operands so that it will be set to activate__wait upon completion of the commands operands. WRITE CONTROL DATA requires a special stub since it can be a +2 operand command or +3 for the 3180 emulation (see 5250 PAI). WRITE DATA AND LOAD CURSOR also requires a special routine since the number of associated operands expected is embedded in the first operand of the command.

After a complete command packet (i.e., the command plus any associated operands) has been loaded into the queue, the DAV interrupt schedules the twinax command processor, tw__session, to process the command. The appropriate session task is scheduled by moving TW__SESS__SCHED into tw__sess__state on the SCP corresponding to this command's physical address. This scheme provides the communication to the foreground task to tell it which of the seven sessions to process.

2

The QUE_LOADER routine is called upon reception of all commands and operands that are queueable and handles stuffing the command in the queue with some exception detection. (Commands that are not queueable are POLLS and ACTIVATES.) The QUE_LOADER maintains the position of commands on the queue and status of the queue with a byte on the SCP called tw_que_ptr. The lower five bits of the byte form a pointer to the next available position to stuff a byte on the queue. Each time a byte is loaded, the pointer is incremented, making bit 5 correspond to the queue being full (TW_QUE_FULL) since it will be set upon loading the sixteenth entry into the queue. Another flag, TW_QUE_NOT_RDY, in tw_que_ptr is used to tell tw_session if a complete command packet (i.e., a command and associated operands) is ready for processing. This flag uses tw_frame_cnt to determine packet boundaries and allows tw_session to process packets as soon as they are available, instead of waiting for a complete queue load before processing the queue. If QUE_LOADER detects that the queue is full, flag TW_QUE_COMPLETE in tw_que_ptr is set and DATA_VECTOR is set to busy_wait for handling busy. TW_QUE_COMPLETE is used as a handshake between the background DAV interrupt and foreground command processor to communicate when the terminal can go unbusy. Exceptions that are set by QUE_LOADER are invalid command and queue overrun exceptions. When an exception is detected, it will not be set if there is already a pending exception. Also, when the exception is detected, the DATA_VECTOR is set to busy_wait to ensure that the terminal will go unbusy to allow the controller to handle the posted exception. The invalid command exception is posted by the queue loader and the tw_session command processor. QUE_LOADER will post an invalid command when a command with associated operands is loaded in the last queue position but operands are still expected. The queue overrun exception is posted when the sixteenth frame received completes a queue load but the RX_EOM flag is not set, meaning more frames are still being received.

The busy_wait state of the DAV interrupt has a number of functions. The DATA_VECTOR is set to busy_wait when exceptions are detected in both foreground and background routines. Also, DATA_VECTOR is set to busy_wait upon receiving a complete queue load of sixteen frames or the reception of an End Of Queue command. The major role of the busy_wait state is to handle the transition of busy (i.e., having commands on the queue) to unbusy (queue empty waiting for more commands). To go unbusy the foreground command processor must have finished processing all the commands from the prior queue load. Once the last command of the queue load is received, TW_QUE_COMPLETE is set by DAV in tw_que_ptr to mark the completion of the queue load. Then, in busy_wait, the DAV routine uses the clearing of TW_QUE_COMPLETE as an indication to clear the POLL response busy bit. In conjunction with TW_QUE_COMPLETE, the DAV interrupt maintains a POLL counter called tw_busy_cnt to provide maximum flexibility in going unbusy. It has been observed that some IBM controllers require that after a complete queue load is received, the terminal must be busy for some finite amount of time before being unbusy. To accomplish this task, the value of tw_busy_cnt is decremented with each POLL received while in the busy_wait state. Upon reaching a count

of zero with TW_QUE_COMPLETE low, busy will go low in tw_presp_stat and tw_busy_cnt will be reinitialized to TW_BUSY_MAX in preparation for the next queue load. The TW_BUSY_MAX equate is set up in TWINAX.HDR and should be set to accordingly. We recommend that TW_BUSY_MAX be set to one since older versions of the 5294 remote controller require at least one "busy" POLL response after a queue load. If a command other than a POLL is received prior to signaling unbusy, the DAV will process the command and set DATA_VECTOR to command decode if TW_QUE_COMPLETE is low. In this case, the tw_busy_cnt value is ignored to ensure that commands are not discarded.

When a PREACTIVATE READ or WRITE command packet is completely received, the DATA_VECTOR is set to the activate_wait state. The role of activate_wait is to handle the transition of busy to unbusy (as with busy_wait), flag an invalid ACTIVATE exception if the controller sends the ACTIVATE before the terminal is unbusy, set up the write_both state for reception of ACTIVATE WRITEs, and schedule the response for an ACTIVATE READ reception. As with busy_wait, TW_QUE_COMPLETE has been set high before entering this state and the interrupt routine uses both seeing TW_QUE_COMPLETE low and tw_busy_cnt equal to zero as criteria for going unbusy. Once the terminal is unbusy, a flag stored in tw_rx_act_flags called RX_PREAC_WR determines whether or not to look for an ACTIVATE WRITE or an ACTIVATE READ command. When an ACTIVATE WRITE is received and expected, the busy flag is set in tw_presp_stat to ensure that the terminal is busy upon completion of the write and the DATA_VECTOR is set to write_both since the WRITE IMMEDIATE command and WRITE DATA command are similar enough to be handled by one state. When an ACTIVATE READ is received or expected, a response is scheduled by loading a timeout into the timer and setting TW_TIMER_RESP in R_STATE. Also, busy is set so that at the end of the read the terminal is busy, and DATA_VECTOR is set to command decode in preparation for the next queue load. Commands other than ACTIVATEs are simply discarded in this state. An invalid ACTIVATE exception is posted if the expected ACTIVATE arrives before the terminal is unbusy. TW_QUE_COMPLETE is set in conjunction with TW_QUE_CORRUPT to tell tw_session to flush the queue. DATA_VECTOR is set to busy wait to handle going unbusy. As with QUE_LOADER, the exception is only posted if there is no pending exception.

As mentioned above, DATA_VECTOR is set to the write_both state to handle stuffing data in the regen buffer following reception of the ACTIVATE WRITE command. The data is always concatenated with the ACTIVATE WRITE command. The write_both state is responsible for detecting the storage overrun exception when the controller attempts to send data beyond the size of the regen buffer. The only difference between the WRITE IMMEDIATE and WRITE DATA commands is that the address counter remains unchanged with the WRITE DATA command while the address counter is set to one greater than the address of the last byte stuffed in the WRITE IMMEDIATE command. To determine whether a WRITE IMMEDIATE or WRITE DATA command is being processed, a flag in tw_rx_act_flags called RX_WR_DATA is set upon reception of the WRITE DATA command. To minimize time on the DAV interrupt, the

WRITE DATA or WRITE IMMEDIATE command routines set up the starting location of the write in tw__act__beginhi/lo on the appropriate SCP. tw__act__beginhi/lo are then used as a pseudo address counter as each byte is received, incrementing upon stuffing the byte in the regen buffer. Upon completion of the write, which is determined by reception of an end of message indicator (RX__EOM set), the pseudo address counter is placed into tw__act__endhi and lo locations with the most significant bit of tw__act__endhi set to inform tw__session that the write is complete. tw__session can then make an action stack entry for Smart Alec screen updates.

POLL

POLL commands are processed completely by the background interrupt routines. The POLL command is decoded in several states since polls play a part in all states mentioned above. The key decisions that are made in the DAV interrupt when a POLL is received and the associated station address is configured by Smart Alec are what is the state of level and what "type" of POLL response to make. The 5250 PAI states that after a Power On Reset, the 5251-11 will respond with a single frame POLL response that is simply a status byte. After the SET MODE command is received, the next reception of a POLL/ACK command causes the terminal to respond with a two frame poll response; the first frame being the former mentioned status byte and the second a keystroke. Also, the PAI states that the first two frame response after receiving the SET MODE will be from level 1. To function in this manner, a flag called TW__PACK__SM is maintained by the DAV interrupt in location tw__level__cnt on the SCP. This bit is set when TX__SET__MODE__RCVD (a SET MODE command has been processed) located in tw__rxtx__status is set and a POLL/ACK is received. Level is used to indicate to the controller that new status is available from the terminal and toggles each time a new keystroke is presented. The reception of a POLL/ACK after the terminal has been put in the two byte response mode results in the POLL response with level toggled from its prior state. Each toggle of level also contains a new keystroke if available. The section of code in the DAV routine that handles level transitions is rx__level__hndlr.

POLLs to nonconfigured station addresses do not result in a response but are used in monitoring activity on station addresses for Smart Alec address bidding purposes. When a frame to an OFFLINE address (i.e., not configured by Smart Alec) is received, the OFFLINE activity monitoring routine is responsible for setting or clearing bits corresponding to each OFFLINE address in tw__line__act on the DCP. Each bit in this location corresponds to a physical address on the network (therefore bit7 is unused), and is set when another terminal or printer is active on that particular address. If the address is available for attachment, the corresponding bit is cleared. Smart Alec monitors this status regularly to communicate to the user whether or not he can attach to addresses via seven locations on the screen. To determine if the address is active, the DAV interrupt looks for POLLs on all OFFLINE addresses. Once a POLL is received, RX__RESPONSE__WAIT and TW__TIMER__RESP flags are set in RSTATE in conjunction with loading a 100 μs response count into the timer to set a time limit for a response to be received. Also, R__STATE is saved at tw__off__save__addr on the DCP to store the address and response flag.

The next time the DAV interrupt hits with a frame to this address, tw__off__save__addr is fetched to see whether we are waiting for a response or not. If we are waiting for a response, RX__RESPONSE__WAIT is checked. If the timer interrupt routine has already run, RX__RESPONSE__WAIT will be cleared which means that a response was not received and the saved address is marked inactive. If RX__RESPONSE__WAIT is still set, this means that the frame just received was a response and the saved address marked active. When an address is marked active, the saved address and response flag are cleared in preparation for the next OFFLINE reception. When an address is marked inactive, the saved address and response flag are cleared only if the frame received is not a POLL. A reception of a POLL results in the new address being saved with a timeout scheduled just as before mentioned.

Errors detected by the receiver are handled on the DAV interrupt and can result in two different actions. All error types flagged by the receiver are treated as equal importance and are logged by maintaining error counters on the DCP for each error type. The appropriate error counter is fetched and incremented upon reception of an error. Once the error is handled, a check to see if the error occurred in the first frame of a message or frames 2−n is checked. First frame errors result in the setting of the line parity error detected bit, TW__LP, and TW__BUSY in tw__presp__stat on each of the current ONLINE sessions. Also, the TW__QUE__COMPLETE flag is set in tw__que__ptr marking the End Of Queue load to ensure we can eventually go unbusy. The 5250 PAI states that all active addresses will report line errors on the first frame since the error could have occurred in the address portion of the frame. If the error is encountered in frames 2−n of a message, the station address is known so only that station sets TW__LP in tw__presp__stat. Also, TW__QUE__COMPLETE and TW__QUE__CORRUPT are set since the validity of the queue load is in question. The task tw__session will flush the queue in this case, allowing the terminal to go unbusy. This allows the controller to handle the line error.

All receiver states exit through a common exit point. Upon exit, if RX__EOM has not been set, RX__MULTI is set to indicate that a multi-frame is in progress. If RX__EOM is set, this means that no more frames are expected and results in the transceiver being reset with RX__EOM and RX__MULTI being cleared. Many subroutines in the DAV interrupt branch directly to rx__eom__rcvd which results in the reset just mentioned. Using the transceiver reset capability of the BCP avoids spending unnecessary time on the DAV interrupt processing information of no concern. For example, the OFFLINE activity monitoring routine only looks for POLLs and flushes any other frames. What this means is that the DAV interrupt has to process the first frame of each message but by issuing a reset, subsequent frames of a multi-frame can be entirely ignored for they will not be recognized by the BCP. After the reset, the receiver hardware looks for a starting sequence and will not extract data until seeing it. Therefore, the remainder of the frame is ignored and the next message will be recognized. Before returning, the state of BIRQ is checked to see if a host access needs service. If BIRQ is low, a call to dca__fast__birq handles the access

2

and returns control back to the DAV interrupt routine. At this point, a check to see if more data is ready for processing is done to avoid unnecessary overhead of exiting the DAV interrupt only to be interrupted again. If no more data is available, IZ, banks, and flags are restored on the return back to the foreground.

Twinax Transmitter Interrupt Routine

The TFE interrupt routine is responsible for loading the transmit FIFO and making the correct response to the controller. The TFE interrupt is normally masked and is unmasked by the timer interrupt when a response timeout count is encountered. A flow diagram of the TFE interrupt routine is shown in *Figure 4-17*.

Initialization requirements for the TFE interrupt are 1) the station address bits, [TCR3–0], must be set to 111; 2) keystroke holding locations tw_presp_key_newxand tw_presp_key_crnt must be cleared; 3) all tw_mode locations on the SCPs should be set to 00h to ensure we come on line with maximum fill between frames.
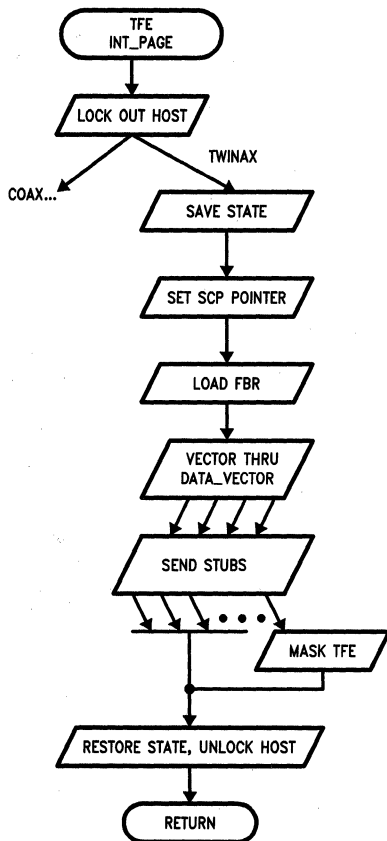
TL/F/10488–37

FIGURE 4-17. Twinax TFE Interrupt

Upon entering the TFE interrupt, the host has been locked on the interrupt page from making accesses for speed considerations. After saving the contents of the IZ pointer, the pointer is loaded with the appropriate SCP address. The appropriate SCP address corresponds to the physical address of the session that is responding to the controller. The address is stored in R_STATE bits 2–0 and these bits are loaded into IZHI bits 2–0 with IZLO cleared, forming the pointer to the first location on the appropriate SCP. Finally, {FBR} is loaded with the value at the tw_mode offset on the SCP to determine the number of fill bits to insert between frames.

Commands that require a response back to the controller are POLLs and ACTIVATE READs. All PREACTIVATE READ commands are processed in the various command processing routines branched to from tw_session. The various routines do exception checking and are responsible for setting up TX_VECTOR to the correct address corresponding to the command type decoded. Most of the READ type commands result in the transmission of an ID type, data for the magnetic stripe reader which we chose not to install, and the three sixteen-bit registers used for processing the regen buffer. READS of the regen buffer are accomplished with the READ IMMEDIATE and READ LIMITS commands. When the ACTIVATE READ is received in the DAV interrupt, a response is scheduled by setting the TW_TIMER_RESP flag in R_STATE and loading an appropriate timeout into the timer. When the TIMER interrupt hits, TW_TIMER_RESP is cleared and the TFE interrupt routine is called to make the response. POLL commands are handled entirely on the background interrupts due to the real time nature of the status response associated with the command. The DAV interrupt schedules the response just as described above for ACTIVATE READs and sets TX_VECTOR to one of three addresses to cover the various POLL responses that can be made. The first frame of all responses must be sent to the controller in a 45 ±15 μs window as defined in the 5250 product attachment information. The response timing is controlled by loading a timeout value of 10 μs (TW_RESPONSE_CNT) into the timer when reception of a POLL or PREACTIVATE READ command is processed in the DAV interrupt routine. For responses that are less than or equal to four bytes, only one entry into the TFE interrupt is required to send the entire frame back to the controller. To load the fourth byte successfully, a short delay loop is put in the code prior to loading the fourth byte to ensure the first byte has propagated through the transmit FIFO and is being transmitted out the serial shift register. If a user wishes to run the BCP at full speed, the delay loop will need to be modified to accommodate the speed increase of the CPU relative to the transmitter. When responses are greater than four bytes in length, the TX_VECTOR is modified prior to exiting so that the next time TFE hits, the correct state will be branched to continue or complete the remainder of the message. Upon determining that the last frame of the response is ready for load, [TCR2–0] are set to 111 for the end of message delimiter as required by the protocol.

In 5250 protocol, keystroke passing is different than in 3270. After a POR, 5250 terminals respond with a single status response. For the 5251-11, a SET MODE followed by a POLL/ACK, causes the terminal to go into a two-byte poll

response mode where the second byte is a keystroke. If no keystroke is pending, the keystroke value is a null (00h). New keystrokes can only be presented following a POLL/ACK from the controller. When a new keystroke is made available to the controller, the LEVEL bit in the first frame status byte of the response toggles from the prior value to inform the controller that new status is now available. The DAV routine controls the poll responses by setting the TX_VECTOR to one of three possible locations for POLL or POLL/ACK responses. For single frame status responses to polls, TX_VECTOR is set to tx_presp_one. As soon as the criteria to go into two frame poll response mode is met, the DAV interrupt sets TX_VECTOR to either tx_presp_crnt or tx_presp_new. In tx_pres_crnt, the keystroke sent back to the controller is the value stored in tw_presp_key_crnt and LEVEL remains unchanged in the first frame stored in tw_presp_key_new, LEVEL is toggled in the first frame status byte response, and tw_presp_key_new is cleared after moving its value to tw_presp_key_crnt. With this approach, keystroke passing with the terminal emulation is simple since by simply checking to see if tw_presp_key_new = 00h determines whether a new keystroke can be loaded for passing back to the controller. In other words, if tw_presp_key_new is nonzero, a keystroke is pending and the emulation program must wait before loading a new keystroke into tw_presp_key_new.

All TFE "states" exit through a common exit point that handles masking the TFE interrupt if no more frames are to be sent, checking to see if a pending BIRQ interrupt is present, restoring foreground registers, unlocking remote accesses, and restoring banks and flags upon returning. If a BIRQ interrupt is pending, DCA_FAST_BIRQ is called to handle the remote access (see Smart Alec Interface discussion). When more frames need to be sent, all of the above occur except masking the TFE interrupt. Also, TX_VECTOR may be modified to ensure that the correct state is entered upon re-entering TFE when it hits again.

TW_TIMER

The timer of the BCP serves dual purposes in the twinax emulation program: as a real timer clock counter and as an interval timer.

A 5251 terminal will turn off the System Available flag if no POLL is received for more than 200 ms. It will initiate automatic power on reset if no POLL is received for more than 2 seconds. Furthermore, the terminal will return to ONLINE from reset mode in approximately 5 seconds. The emulation program uses seven 8-bit counters (tw_sysa_por_cntX, $0 < X < 7$) to keep track of these real time events (one for each session). These counters are incremented by one every 21 ms. This 21 ms clock tick is generated by the TIMER interrupt. The value of 21 ms gives a maximum counting time (around 5.4 seconds) and a reasonable counting resolution ($\pm 10\%$ for a count of 200 ms). The timer of the BCP is configured to use $\frac{1}{16}$ CPU clock as input clock.

On the other hand, the emulation program utilizes the timer to provide a 45 μs time-out signal. When the receiver routine receives a POLL or ACTIVATE READ command and decides to respond to the host, as per IBM's requirement, it has to do it 45 μs ± 15 μs after the reception of the command. The program will setup the timer to generate a 45 μs time-out signal which in turn activates the transmitter routine. The program first stops the 21 ms counting of the timer, it saves the current counting value, it loads the timer to a count of 45 μs (minus some offset to compensate for program execution time), it then starts the timer and reloads the previous counting value to the timer registers. When a time-out occurs, the previous counting value will be loaded into the timer automatically to resume the 21 ms counting. In addition, the program will set a flag to indicate that the timer has counted 45 μs. In this way, the timer is occasionally interrupted from the normal 21 ms counting and "borrowed" to provide 45 μs time-out. As 45 μs is much shorter than 21 ms and the interruption is not too frequent, the error introduced is negligible.

When either 21 ms or 45 μs time-out occurs, program execution will be transferred to the timer interrupt service routine (tw_timer_int). At the very beginning of the routine, it locks out remote accesses to prevent losing back-to-back PC accesses. The program then checks the source of the interrupt. If it is due to 45 μs time-out, the program simply reloads the 21 ms count value into the timer registers and unmasks the TFE interrupt. Once the TFE interrupt is enabled, the transmitter interrupt routine will be invoked to respond to the host. If the interrupt is due to 21 ms time-out, the program increments all real time clock counters by one unless the counter has already reached 'FF'. It is necessary to keep these counters from overflow because the foreground program has no way to distinguish counter overflow. In order to keep the execution time of the interrupt service routine as short as possible, the timer routine does not perform other checking to these counters. However, the routine still has to check pending host accesses and call dca_fast_birq if needed. The foreground program (tw_session) is responsible for checking these counters and invoking real time events at the right moment.

The Command Stubs

The twinax part of the MPA program emulates the IBM's 5251 model 11 display terminal. The following discussion will be based on the commands for 5251 model 11. The command set of 5251 model 11 is shown in Table 2-2 located in Section 2. The commands are divided into two main groups: the queueable commands and non-queueable commands. The three non-queueable commands: POLL, ACTIVATE READ, and ACTIVATE WRITE are not handled by the foreground programs as they are not queueable. Instead they are handled in real time by the background interrupt service routines. For details of the processing of these three commands, refer to the twinax receiver interrupt and transmitter interrupt sections.

All other commands are queueable, namely, they are pushed into the command queue when received by the receiver interrupt routine. They are processed by the foreground task, tw_session, when it is invoked by the Kernel. In order to divide the program into properly grouped modules and make documentation easier, the queueable commands are further divided into four groups according to their functionalities: Reads, Writes, Control and Operators. This grouping is not a definition by IBM's PAI document. The commands shall be discussed according to this grouping.

2

One may observe that in addition to the 5251 model 11 command set documented in the IBM's PAI, there are two extra commands in Table 2-2. These are READ IBM and WRITE DATA. READ IBM is an undocumented read command that reads to the end of line. To allow the MPA board to work with IBM's System Units properly, the emulation program must be able to handle this command. Response to this command will be discussed under the READS section. WRITE DATA is documented only under the Printer Interface of the IBM's PAI, but actually the IBM's 5251 terminals are able to handle this command properly. Therefore we also implemented this command in the emulation program.

Commands to the display terminal can be addressed to different logical devices and feature devices. It is specified in the modifier/device address field of the command. The device address or feature address should not be confused with the station address. Station address appears in another field and is handled by the receiver and transmitter interrupt routines. In the MPA twinax emulation program, Base and regeneration buffer, Keyboard, Indicators, and Model id are implemented. The Magnetic Stripe Reader feature is not implemented and commands to this feature will return a "not installed" response.

As described earlier, tw__session is responsible for decoding the commands and directing the execution of the program to the proper command processing routines. There are some common practices or "rules" in coding the command processing routine so that they can be interfaced with the session task properly. On entering a command routine, GP0 contains the command word and IZ contains the current SCP pointer, Main Bank A & B are selected. On leaving a command routine, IZ, GP7 and register bank selection must be restored if changed. The common point of exit is to LJMP to tw__cmd__ret (twinax command return). For most commands, all 8 bits of the device address and command fields have been fully decoded upon entry and, therefore, require no additional decode in the command routine. However, for the RESET, READ DEVICE ID and READ DATA commands, the device/feature address field must be decoded in the command routines. This is because these three commands can be addressed to a number of device/features or can be addressed to uninstalled device/features. A number of commands are associated with one or more data frames. Therefore the command routines must pop those frames off the command queue with LCALL(s) to tw__que__popper. The command routines should check the queue empty flag to prevent catastrophic errors. In normal operation, the queue will never be empty when it is popped by the command routines. Should the empty flag be true after a call to the tw__que__popper, it suggests that a programming error has been encountered. At this time a LCALL to tw__bugs is performed followed by a graceful error recovery (the tw__bugs routine is discussed in the Software Debugging Aid section). Most commands require the emulation program to check for the validity of the operands which are held by the address counter, reference counter or cursor register prior to, or in the course of the operation of the command. If any invalid operand is detected, it must be reported back to the system unit through the exception status. The command processing routines should set the exception

type, LCALL to tw__post__exception and then pass control back to tw__session via tw__cmd__ret if an exception is detected. The tw__clear__exception routine should be called if a command is going to clear exception status. In addition, command routines should never flush the command queue directly.

The 5250-11 regeneration buffer size is 2000 bytes. The valid values of the address counter, reference counter and cursor register ranges from 0 to 1999. However, within the twinax emulation program, these counters contain an offset which corresponds to their starting address within the BCP's data memory. For example, if the address counter sent by the system unit is 20h and the regen buffer of that session starts at BCP's data memory address of 2048h, then the address counter value stored in the SCP is 2068h. We refer to the original values of the counters as relative addresses and the stored values as absolute addresses. The reason for storing these counters in absolute address form is that the command processing routines can use them directly as data pointers without adding an offset value. This can speed up the time-critical interrupt service routines. However, whenever these counter values are passed to or from the System Unit via the Smart Alec interface, a conversion procedure is needed. Furthermore, as these values no longer start from zero, one has to check whether they are less than the lower boundary of the regen buffer address when performing the validity check. Another point is that for some commands, the final values of the counters may be rolled to 2000 if the last affected location is 1999 (in forward operation) or 65535 if the last affected location is 0 (in backward operation). Exception status should not be reported in these cases.

As mentioned in Section 2, Smart Alec utilizes a 31 entry FIFO queue that contains screen modification information. The FIFO queue contains starting and ending addresses of the screen area that has been modified. In the Smart Alec documentation this queue is referred to as the action stack. In order to emulate the Smart Alec interface an action stack was implemented on the MPA. Every command processing routine that will modify the screen is therefore responsible for loading the action stack with the proper address values. In the tw__util module, there is an action stack loader, tw__act__ldr, and an action stack popper, tw__act__popper, dedicated to maintaining the action stack. The action stack is actually a circular FIFO queue with a length of 124 bytes located in the SCP of every session. It can hold up to 31 entries as defined by the Smart Alec document. To load the action stack, the command processing routines must first load the appropriate memory locations and registers with the starting and ending address of the modified buffer area. Second, they must determine the type of modification as defined by the Smart Alec interface. Finally, the routines should call the action stack loader.

READ Commands

All read type commands are grouped in the TW__ READ.BCP module. The entry names of the command routines are shown in Table 4-6. The read command routines are in general, quite straightforward. This is because the actual response of all read commands is controlled by the transmitter interrupt routine. The foreground read command routines are only responsible for setting up the proper re-

sponse routine addresses in IW for the transmitter interrupt and for performing some regen buffer address checking if needed.

The tw_read_regs_cmd command routine sets up the READ REGISTERS routine tx_read_registers for the transmitter and then jumps back to tw_cmd_ret. The transmitter will in turn respond to the system unit with six bytes containing the values of the address counter, cursor register, and reference counter.

The tw_read_ibm_cmd command routine sets up the READ IBM routine, tx_read_ibm, for the transmitter and then jumps back to tw_cmd_ret. The transmitter will in turn respond to the System Unit with four bytes of zero. This is how the IBM's 5251-11 terminals respond to this undocumented command.

The tw_read_dev_id_cmd command routine first decodes the device/feature address by comparing the field to all defined logical devices and feature addresses. If there is a match, it will jump to the appropriate command routines to set up routines to respond to the device or feature id. Otherwise it will jump to the tw_read_fid_not_install routine which will direct the transmitter to respond with a zero data.

The tw_read_data_cmd command routine sets up the READ DATA with address msr routine, tx_read_data, for the transmitter and then jumps back to tw_cmd_ret. The transmitter will in turn respond to the System Unit with sixteen bytes of zero data.

The tw_read_limits_cmd transfers a display field of data to the controller. The area of transfer is delimited by the address counter and reference counter; therefore, tw_read_limits_cmd first checks whether they lie within valid range and whether the reference counter is greater than or equal to the address counter. If any one of these tests fail, the program will post an invalid register value exception and return to the session task. Otherwise, it will pass the address counter and the byte count (reference minus address) to the transmitter interrupt through four memory storage locations: tw_act_beginlo, tw_act_beginhi, tw_act_endlo and tw_act_endhi, and then set up the READ LIMITS routine. The transmitter will then fetch the data from the regen buffer and send it to the System Unit. Before returning to session task, this command routine will update the address counter to the value of reference counter plus one.

TABLE 4-6. Entry Names of Module tw_read

Command Name	Command Routine Entry Name
READ REGISTER	tw_read_regs_cmd
READ IBM	tw_read_ibm_cmd
READ DEVICE ID—base	tw_read_dev_id_cmd
READ DEVICE ID—keyboard	tw_read_dev_id_cmd
READ DEVICE ID—msr	tw_read_dev_id_cmd
READ DATA	tw_read_data_cmd
READ LIMITS	tw_read_limits_cmd
READ IMMEDIATE DATA	tw_read_imm_cmd

The tw_read_imm_cmd command pops out the starting address from the command queue and determines whether it is valid. If it is valid, it will be converted into an absolute address, as we discussed in the introduction, and passed to the transmitter. The tw_read_imm stub will be set up. The ending point of operation is unknown at this moment. It will be determined by the transmitter during its reading of the regen buffer.

WRITE Commands

All write type commands are grouped in the TW_WRITE.BCP module. The entry names of the command routines are shown in Table 4-7. The PREACTIVATE WRITE command routines, tw_write_imm_cmd and tw_write_data_cmd, are relatively simple. They just set the beginning address of the operation to tw_act_beginhi and tw_act_beginlo. When the receiver interrupt gets an ACTIVATE WRITE command, the receiver interrupt will put the data into the regen buffer and determine the end of operation. Processing of other write commands is done completely in the foreground. We shall discuss each command in more detail.

The tw_write_cntl_cmd command pops the data byte following the command from the queue and puts it into the control register location (tw_ctrl1) in the SCP. It also checks the Reset Exception Status bit (bit 12) of the data word. If the bit is set, the tw_clear_exception subroutine is called. On the 3180-2 model terminal, the command may have a second data byte. This routine checks bit 8 of the first data byte, if it is set, one more byte will be popped out and saved into tw_ctrl2 in the SCP.

TABLE 4-7. Entry Names of Module tw_write

Command Name	Command Routine Entry Name
WRITE CONTROL DATA	tw_write_cntl_cmd
WRITE DATA and LOAD CURSOR—base	tw_write_data_id_cur_cmd
WRITE DATA and LOAD CURSOR—indicat	tw_write_data_to_ind_cmd
WRITE IMMEDIATE DATA	tw_write_imm_cmd
WRITE DATA	tw_write_data_cmd

2

The tw__write__data__ld__cur__cmd command may also have one or more data bytes associated with it. This routine checks the first data byte to determine if it is in the range of 1 to Eh. If the data byte is not in this range, it is the only data byte associated with the command and the routine just writes it to the location pointed to by the address counter. If the data byte is in this range, the routine will take the first byte as the byte count and will pop that number of data bytes from the queue and write them into the regen buffer. During the write operation, the address counter will be incremented and checked for overflow. Storage exception status will be posted if an overflow occurs. At the end of the operation, the program updates the cursor register to the value of the address counter and loads up the action stack by calling the tw__act__ldr routine.

The tw__write__data__to__ind__cmd command routine handles the WRITE DATA AND LOAD CURSOR command addressed to the indicators. It simply pops out the data byte following the command and saves it in the memory location tw__idctr__data in the appropriate SCP. It also notes the transition direction of certain indicators and saves this information in the memory location tw__sa__trans__ident for Smart Alec.

The tw__write__imm__cmd routine first pops the starting address from the queue, then checks to see if it is valid. If it is valid, it will be converted into absolute form and passed to the receiver interrupt. The starting address entry of the action stack is also set up. The receiver will then pick up the rest of the operation when the ACTIVATE WRITE command is received.

The tw__write__data__cmd routine checks the address counter and passes it to the receiver interrupt as the starting address of the operation. The subsequent operation is identical to the WRITE IMMEDIATE command.

Operators

The module TW__OPER.BCP contains command routines for all operator commands. Entry names of these routines are shown in Table 4-8.

The CLEAR command routine is actually a subroutine that returns to its caller. Therefore, the command routine tw__clear__cmd simply calls the actual clear routine, tw__clear__routine, and upon return from that routine, tw__clear__cmd LJMP's back to tw__session as required by all command routine. The subroutine tw__clear__routine checks the address and reference counters to see if they point at valid screen addresses and that the address counter is less than or equal to the reference counter. If any of these are false an invalid register exception is posted and no clearing takes place. Otherwise, the bytes starting with the byte pointed to by the address counter are zeroed up to and including the byte pointed to by the reference counter. Then an action stack entry is made to notify the Smart Alec interface of the screen update. The address counter and reference counter's contents are not modified.

TABLE 4-8. Entry Names of Module tw__oper

Command Name	Command Routine Entry Name
INSERT CHARACTER	tw__insert__cmd
CLEAR	tw__clear__cmd
MOVE DATA	tw__move__cmd
SEARCH NEXT ATTRIBUTE	tw__search__attr__cmd
SEARCH NEXT NULL	tw__search__null__cmd

The tw__insert__cmd command routine first examines the regen buffer location pointed to by the reference counter. If it is not a null, a null or attribute error exception will be posted and operation terminates. If it is a null, the program proceeds to check the address counter and reference counter to see whether they are valid. If the counter values are valid, the insert operation will be carried out. At the end of the operation, the address counter and cursor register will be updated and the action stack will be loaded by calling the tw__act__ldr routine.

Although the operation of the tw__move__cmd command is quite complex, the IBM PAI gives a fairly clear description of it. This routine checks the address counter, reference counter and cursor register to determine whether the move is forward or backward. The program then carries out the move operation as per the description of PAI. The action stack load for the move command consists of two entries or four values. The first entry is the starting address and ending address of the destination area of the move. The second entry is the starting addresss of the source area and the direction of operation. Details of these entries can be found in the Smart Alec user manual.

The tw__search__attr__cmd command routine first checks the address counter to make sure it is within the valid range. Next, starting from the current address counter value, the routine searches the regen buffer to find an attribute. If an attribute is located, the reference counter will be set to the address of the attribute minus one. The routine will post a null or attribute error exception if no attribute is found when the end of buffer is reached.

At the beginning of the tw__search__null__cmd routine, it checks both the address counter and reference counter to make sure they are within valid range and that the reference counter is equal to or greater than the address counter. If the checks are successful, the program proceeds to search for a null character starting from the current value of the address counter. If a null is found, the reference counter will be set to the address of the null minus one. Otherwise the operation will terminate when the reference counter is reached and a null or attribute error exception will be posted.

TABLE 4-9. Entry Names of Module tw__cntl

Command Name	Command Routine Entry Name
LOAD ADDRESS COUNTER	tw__load__addr__cmd
LOAD CURSOR REGISTER	tw__load__cursor__cmd
LOAD REFERENCE COUNTER	tw__load__ref__cmd
SET MODE	tw__set__mode__cmd
RESET	tw__reset__cmd
EOQ	(none)

Control

The module TW__CNTL.BCP contains all the routines that handle the control commands. The entry names of all routines are shown in Table 4-9.

The tw__load__addr__cmd command routine pops the address counter value from the command queue and saves it on the SCP in absolute form. However, as per IBM's PAI, there is no need to check the validity of the value before loading. As a remark to clarify the ambiguity of the PAI, the address counter value consists of two bytes, the high order byte is the first data byte following the command while the low order byte is in the second byte.

The tw__load__cursor__cmd command routine loads the cursor register in the SCP with a new value. The operation is similar to tw__load__addr__cmd routine.

The tw__load__ref__cmd command routine loads the reference counter in the SCP with a new value. The operation is similar to tw__load__addr__cmd routine.

The tw__set__mode__cmd routine pops the fill bit count from the command queue, converts it to the BCP's Fill Bit Register format, and saves it on the SCP. Next, the set mode received bit is set in the SCP. This signals the background receiver task that it may start responding to polls using the two byte response format, (after a PACK is received). Finally, if the current exception state indicates POR then the exception state is cleared.

Like the tw__clear__cmd routine, tw__reset__cmd actually calls the subroutine tw__por (which performs a POR on the current session). The routine tw__por first places the current session OFFLINE by signaling to the background receiver task (via the RX__RESET bit) that it is not to respond to the host until further notice. Once this is done, the tw__por routine can begin changing memory locations normally updated by the background receiver task without disabling interrupts because the RX__RESET bit effectively disables the receiver task when working with this physical session. Next, the exception status is changed, notifying other tasks that this session is in POR. The time count for this session is cleared and a bit is set (in the tw__por__waited__session byte on the DCP) informing the other tasks that the 5 second POR time-out has commenced. The tw__task routine will use this time count and this session's por wait bit to determine when to bring the session back on line. Other tasks use the POR wait bit when interpreting the meaning of the time count for the current session. The action stack is cleared next, along

with the Smart Alec task handshake bits. Then, the screen buffer for this session is cleared via a call to tw__clear__routine, which issues an action stack entry reflecting the cleared screen. (This allows the PC to accurately reflect the POR state.) Finally, the remaining SCP variables are set to their appropriate values, except for the variables controlled by the smart alec task, (i.e., Model ID, Keyboard ID, etc . . . ,) which are left unchanged.

The End Of Queue command does not actually have a command routine, for at this point in the command decoding process of the MPA it does not provide any additional information. As far as the command processor is concerned, the queue load complete flag, set by the background receiver task, indicates the actual end of queue. So the act of popping the EOQ command off the queue completed the command's execution, no call to a command routine is required.

The Twinax Session Command Processor

The twinax session command processor, tw__session, is located in module TW__SESS.BCP. Its job is to perform all non time-critical functions related to sustaining an active twinax session. This includes processing the internal command queue, error recovery, and performing a POR. In addition, tw__session and its subordinate routines are responsible for communicating important events (like screen updates) to the emulation interface routine (i.e., the smart alec task), which operates asynchronously to twinax session activity. (tw—session's subroutines include all the command routines previously discussed.)

The command processor, tw__session, and its subordinate routines are written with "reusable" code. That is, all the information regarding a given twinax session's state is kept in the SCP (the data memory Session Control Page) attached to that physical session. There is no dependency between tw__session and an active session's state from one call to the next. At any time, any SCP may be passed to tw__session. In other words, the current state of a given physical twinax session exists only in its SCP, not in the command processor. This gives one set of routines (tw__session and its subordinates) the ability to process all the active twinax sessions concurrently. The twinax task tw__task simply passes the pointer of the scheduled session's SCP (via the IZ register) to tw__session and tw__session then determines the current state of that session and what action(s) need to be performed.

The program flow of tw__session proceeds as follows. First, tw__session checks for the ACTIVATE WRITE command for the current session completed in the background. If one has occurred, tw__session performs an action stack push, which notifies the Smart Alec interface of the screen update. Next, the command processor checks for actions requested by other tasks. Currently, two actions are defined: the "forced" POR and the "requested" POR. The "forced" POR is usually issued by the smart alec interface task and it forces a POR regardless of the current session status. After the POR is initiated control returns to the calling routine (tw__task). The "requested" POR is usually issued by tw__task when an Auto-POR is desired. A POR is only performed if the current session is not already in the POR exception state or if an error condition does not exist. Otherwise, this request is ignored. In this way, the twinax session will not unnecessarily POR. Again, after a POR is initiated control returns to the calling routine.

2

Once all the requested actions from other tasks have been handled, the command processor attempts to process the internal command queue of the current session. Rather then holding off the command processor from processing commands on the queue until a queue load is complete, we opted to exploit the power of the BCP by using a parallel processing approach where both the background receiver task and the foreground command processor have access to the command queue simultaneously. This enables the command processor to execute commands even while the queue is still being loaded by the host. To avoid conflicts, the command processor tw__session takes a "snap shot" of the current internal command queue and current exception status (in the poll response byte). The command processor then works from the "snap shot" while the background receiver task updates in real time.

The "snap shot" involves the following steps. Interrupts are disabled to prevent background tasks from updating the command queue. The command queue is then checked to see if another task has marked it as "corrupt". When a background task determines that the command queue may contain invalid data (for example, due to a line parity error or the detection of an exception) it marks the queue as corrupt and schedules that session. The tw__session routine then flushes the queue when it gets control. Flushing the command queue resets all the queue pointers and flags. This marks the command queue as empty. It also signals the background tasks that tw__session has acknowledged the error and cleaned up the command queue. This handshake is required since background tasks are only allowed to push onto the internal command queue, never flush it. (At the next poll to this session, the background receiver task will indicate "not busy" to signal the host that this device has completed error recovery.) After the command queue is flushed, tw__session will deschedule this twinax session and return to the calling routine (tw__task). If the internal command queue is not corrupt, tw__session checks to see if it is "ready" for processing. The command queue is marked as "not ready" while the background receiver task is in the middle of pushing a multi-byte command (for example the LOAD ADDRESS COUNTER command) onto the queue. While the queue is marked as "not ready" tw__session will not attempt to process any commands on the queue. Instead, tw__session leaves this session scheduled and returns to tw__task. This keeps the command processor and its subordinate routines from attempting to pop incomplete commands off the internal command queue. On the next Kernel cycle tw__session will once again be called upon (by tw__task) to process this session's command queue. If the internal command queue is marked "ready" for processing then tw__session copies the current queue pointer, the current exception status (located in the poll response byte), and then deschedules this session. This completes the "snap shot". Interrupts are enabled so that other tasks may continue to update the command queue.

Now that the "snap shot" of the command queue has been taken, tw__session can begin popping commands off the queue and decoding them. The command queue is processed based on tw__session's current version of the exception status, initially recorded during the "snap shot". This excep-tion status is checked before the decode of each command to determine the current exception state of this session, since command decode depends on this state and previous command execution may change the state. (Note that this copy of the poll response's exception status may not match the actual exception status after the "snap shot" has been taken. This is simply a consequence of background/foreground parallel processing and is not a problem. The next time a queue "snap shot" is taken the tasks are brought back into sync.) While in POR exception state, only the SET MODE and RESET commands are considered valid. While in any other exception state, only the SET MODE, RESET and WRITE CONTROL DATA commands are considered valid. In normal mode (no exception state) all commands are considered valid. If an invalid command for the current exception state is decoded, the command queue is flushed and tw__session will attempt to post an exception. A valid command decode causes tw__session to pass control to that command's routine (called a command stub) for processing. Most of the commands have been fully decoded by tw__session before their command routine is executed, but a few commands require the command routines to further decode the feature address field. Each command routine is responsible for popping its associated data off the command queue. Each command stub is responsible for carry-ing out complete command execution, including posting exceptions, making action stack entries, etc ... (Many of these tasks are actually carried out by calls to support sub-routines.) All command routines return to the same entry point in tw__session. (See the comments in tw__session, at the command decode section, for a complete set of rules regarding command stub coding.)

When all the commands have been popped off the current command queue snap shot, the queue load complete flag (TW__QUE__COMPLETE) is checked. This flag is set by the background receiver task when an EOQ designator has been received. (An EOQ designator can be an EOQ com-mand, a PRE-ACTIVATE command, or a full command queue.) If the queue load complete flag is set then tw__ses-sion flushes the command queue, clearing this flag and re-setting the queue load complete flag by tw__session signals the receiv-er task that it may clear the poll response busy status flag at its discretion. This in turn signals the host that the queue load has been completely processed and a new queue load may be initiated.

Finally, tw__session returns control to the calling routine, tw__task, not to be called again for the current session until another task schedules this session to perform additional work.

Handling Exceptions

Exceptions are posted by the subroutine tw__post__excep-tion (located in module TW__UTIL.BCP). This is the only reliable way for foreground tasks to post exceptions since both foreground and background tasks must be made aware of the exception. The tw__post__exception routine first disables interrupts to hold off background processing. It then updates tw__session's exception status. Next, it up-dates the poll response exception status, but only when no exception is currently pending. The tw__post__exception routine then places the background receiver task into its busy wait state. This prepares the receiver task to respond

"not busy" on subsequent polls from the host. Following that, tw_post_exception flushes the command queue per the PAI. Finally, after a quick check of BIRQ to avoid missing PC accesses, interrupts are enabled and tw_post_exception returns to the calling command stub.

Exception status is cleared by tw_clear_exception, located in module TW_UTIL.BCP, for the same reason as stated above. This routine sets both tw_session's exception status and the poll response exception status to zero while interrupts are disabled. Again, BIRQ is checked before interrupts are enabled to avoid missing PC accesses and then control returns to the calling command routine.

Software Debugging Aids

The subroutine tw_bugs, located in the module TW_TASK.BCP, is used for a debugging aid. Routines call tw_bugs when they detect invalid states; for example, the Smart Alec read command addressed to physical session 7 (the seven physical sessions are numbered 0–6). During initial debug, the SCPs and DCP are usually relocated into dual port memory by trading them with screen buffer 3 (sbp 3). The tw_bugs routine is then set to disable interrupts, unlock the PC, and jump to itself so that when called, the current state of the MPA is frozen and can then be viewed using the Capstone Technology debugger. After initial debug is complete, tw_bugs is set to simply log the occurrence of a bug by incrementing a counter in the DCP and return to the caller. The caller should then attempt a graceful recovery. A check of the tw_bugs counter will reveal if routines are detecting unexpected conditions when in the field.

Smart Alec Interface Overview

Smart Alec is a micro-to-System 3x link produced by Digital Communications Associates. It provides the IBM PC, PC XT, or PC AT with a direct link to IBM System 34, System 36, or System 38 midrange computers. The Smart Alec product includes a printed circuit board that installs in any full length slot in the PC, and a software package that consists of a 5250 terminal emulation program, called EMU, and a bi-directional file transfer utility. A splice box to facilitate connection to the twinaxial cable is also included.

The terminal emulation program provides the user with all the features of 5251 model 2, 5291, or 5292 model 1 terminal. It also allows a PC printer to emulate the IBM 5256, 5219, 5224, 5225, and 4214 system printers. The file transfer utility provides bi-directional data transfer between the PC and the System 3x. Additional features include the ability to support up to seven host sessions, the capability to bid for unused addresses, compatibility with software written to comply with the IBM Application Program Interface, "hot key" access, and 3270 pass through support.

As mentioned earlier, IBM was the first to enter the marketplace with a 5250 terminal emulator. This was soon followed by the release of similar products including DCA's Smart Alec. Smart Alec was however, the first product to offer seven session support, address bidding, and a documented architecture for third party interfacing. As with IRMA, Smart Alec and its associated interface gained acceptance in its respective marketplace. As a result of this the Smart Alec interface was chosen for the DP8344 Multi-Protocol Adapter to further show the power and versatility of the DP8344 Biphase Communications Processor. The MPA hardware with the MPA soft-loadable microcode is equiva-

lent in function to the DCA Smart Alec board and its associated microcode with respect to terminal emulation and file transfer capabilities (the printer emulation and non-vol RAM configuration storage were not implemented on this version of the MPA). Both directly interface with the Smart Alec terminal emulation software that runs on the PC (EMU, file transfer utilities, etc . . .) providing the same terminal emulation functions and features of the Smart Alec product. The following sections describe the hardware interface and the BCP software in the Multi-Protocol Adapter Design and Evaluation kit that is used to implement the Smart Alec interface. All of the following information corresponds to Rev 1.51 of the Smart Alec product.

Hardware Considerations

The Smart Alec printed circuit board plugs into any full size expansion slot in the IBM PC System Unit. It provides a cable and splice box that allows the bulky twinaxial cable from the System 3x to be connected to the back panel of the Smart Alec board. The splice box also contains termination resistors that can be switched in to terminate the line if it is the last device. Smart Alec operates in a stand-alone mode, using an on-board microprocessor (the Signetics 8X305) to handle the 5250 protocol and multiple session screen buffers. Because of the timing requirements of the 5250 protocol, the on-board 8X305 operates independently of the 8088 or the System Unit. The 8X305 provides the intelligence required for decoding the 5250 protocol, maintaining the multiple screen buffers, and handling the data transfer and handshaking to the System Unit.

The Smart Alec card uses a custom integrated circuit to interface the 8X305 to the twinaxial cable. This custom device is essentially a transmitter and receiver built for the 5250 environment. It can take parallel data from the 8X305 and convert it to a serial format while adding the necessary 5250 protocol information and transmit this to the twinaxial cable through additional interface circuitry. It also accepts a serial TTL level signal in the 5250 word format and extracts the 5250 protocol specific information and converts it to a parallel format for the 8X305 to read.

The card contains 16k of data memory for the screen buffers and temporary storage. Each session can require up to 2k of data memory for its associated screen buffer, accounting for a total of 14k. The remaining memory space is used by the 8X305 for local storage.

The hardware used in enabling the 8X305 to communicate with the PC's 8088 processor is a dual four byte register array. The 8X305 writes into one side of the four byte dual register array which is read by the 8088. The 8088 writes into the other side of the dual array which is in turn read by the 8X305. The dual register array is mapped into the PC's I/O space at locations (addresses) 228h–22Bh. This interface is identical to that found on the IRMA board except for the I/O addresses.

A handshaking process is used between the two processors when transferring data. After the 8088 writes data into the array for the 8X305, it sets the "Command" flag by toggling bit 0 (writing a "1" then writing a "0") in I/O location 22Bh. This is decoded in hardware and sets a flip-flop whose output is read as bit 7 (the msb) at location 22Eh, When the 8X305 has read the registers and responded with appropriate data for the 8088, it clears this flag by resetting the flip-flop. A similar function is provided in like manner for transfers initiated by the 8X305. Here the flag is called the "Attention" flag and can be read as bit 6 at location 22Eh. This

flag is cleared when the 8088 toggles an active low bit in bit position 0 at location 22Dh. Even though the attention flag function is documented, it is not used on this revision of Smart Alec.

Two additional features not found on Rev. 1.42 of the IRMA card were implemented on the Smart Alec board. These are the ability to softload the 8X305's instruction memory and the ability to save configuration information in a non-volatile RAM on the board. The control signals needed for these tasks are transferred to the Smart Alec board from the 8088 in bits 1–5 at location 22Dh and 22Eh, and in bits 6 and 7 at I/O location 22Fh, When the terminal emulation program, EMU, is invoked for the first time after each power up the 8X305 microcode is downloaded into RAM on the Smart Alec board. Information generated through the configuration program EMUCON is loaded into a 9306 serial non-vol RAM on the Smart Alec board. This is accessed at power up thus eliminating the need for the user to configure the board every time the PC is turned on. A block diagram of the Smart Alec hardware is shown in *Figure 4-18*.
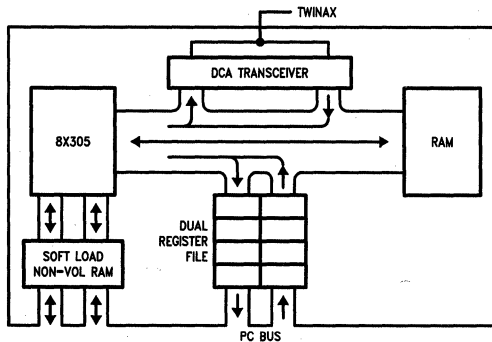
TL/F/10488–38

FIGURE 4-18. Smart Alec Hardware Block Diagram

The Multi Protocol Adapter printed circuit board also plugs into any expansion slot in the IBM PC System Unit. Like Smart Alec, it provides an adapter to allow the bulky twinaxial cable from the System 3x to be connected to the back panel of the card. The MPA board contains the termination resistors on the PC card and not in a splice box. These resistors can be "switched in" via two jumpers. The MPA operates in a stand-alone mode, using the DP8344 Biphase Communications Processor to handle the 5250 protocol and multiple screen buffers. Again, because of the timing requirements of the 5250 protocol, the BCP operates independently of the 8088 microprocessor of the System Unit. As with the 8X305, the BCP provides the intelligence required for decoding the 5250 protocol, maintaining the multiple screen buffers, and handling the data transfer and handshaking to the System Unit. However, with the BCP's higher level of integration, it also interfaces with the twinaxial cable. The BCP has an internal biphase transmitter and receiv-

er that provides functions similar to the custom transceiver on the Smart Alec board. As is the case in 3270, the DP8344's CPU can handle the 5250 communications interface very efficiently. It also has the extra bandwidth to allow the MPA to easily handle the multiple sessions. To illustrate this fact, the CPU clock frequency on the MPA card is set at 9.5 MHz instead of its top speed of 20 MHz. This also allows slower, less expensive RAM to be used on the board if desired.

The MPA card contains a single 32k x 8 RAM memory device for the screen buffers and temporary storage. This memory size was chosen to handle all seven sessions in a single RAM.

The hardware used to enable the MPA's BCP to communicate with the PC's 8088 processor is steering logic (contained in PALs) and the data RAM. In a typical application, the BCP communicates with a remote processor by sharing its data memory. This is true with the MPA, but because the MPA must run with the Smart Alec software, steering logic was used to direct the 8088's I/O reads and writes done by the Smart Alec software into data memory locations on the MPA card. The I/O accesses performed by the Smart Alec software can be fit into three groups; accesses to the dual register array, accesses to the handshaking flags, and accesses to configure the card. All of these are directed in the BCP's data memory, however each are handled differently by the MPA. By using data memory and the extra processing power of the BCP's CPU instead of discrete components the number of integrated circuits on the board was reduced.

The Smart Alec dual register array is implemented on the MPA card in the same fashion as was the IRMA dual register array. The I/O accesses from the System Unit are "steered" to two different BCP data memory locations depending on if they are reads or writes. The writes from the 8088 are directed to memory locations 7F28h–7F2Bh, and the reads from the 8088 are sourced from memory locations 7E28h–7E2Bh. The MPA Register Array Implementation is shown in *Figure 4-19*.

The handshaking process on the Smart Alec card differs from the IRMA implementation. To set the command flag, bit 0 in the register at I/O location 22Eh must be toggled (a write of a "1", followed by a write of a "0". In the IRMA interface, just writing to an I/O location would set the command flag. This is not the case with Smart Alec because the additional softload and configuration capabilities of the Smart Alec card are required that each of the bits in these registers have different functions. The MPA hardware used to emulate the handshaking function for Smart Alec is similar to its IRMA implementation. When the 8088 goes to set the command flag by toggling bit 0 at I/O location 22Eh, it actually does a write to 7F2Eh in the MPA's data memory via the steering logic. The steering logic also interrupts the BCP telling it an access has been made to the Smart Alec I/O space. The BCP then determines if it was a write to the PC I/O location 22Eh by reading a byte of data from the steering logic. If a write occurs to I/O location 22Eh, the BCP reads the memory location 7E2Eh and determines if

the "set command flag" bit was toggled. It does this by checking to see if bit 0 and bit 4 (the non-vol RAM enable bits) are low. If this is the case, it then knows that the Smart Alec software intended to set the command flag. The attention flag is not implemented on this version of Smart Alec and is therefore not implemented on the MPA. However, if one chooses to do so it can easily be done in the same manner.

The System Unit accesses done to configure the Smart Alec board consist of a method to softload the 8X305 and to read and write set-up information into a non-vol RAM. Because the MPA uses the DP8344, there is no need to emulate the 8X305 softload function. The DP8344 is itself softloaded using the MPA loader before the Smart Alec software is invoked. The reading and writing of the non-vol RAM is done using additional bits in the control and strobe registers at I/O locations 22Dh, 22Eh, and 22Fh. In the Smart Alec implementation the System Unit must provide all the control, data and clock signals to the non-vol RAM via the above mentioned I/O locations. The non-vol RAM is not implemented on the MPA card but because the Smart Alec emulator, EMU, reads this information on power-up the MPA does emulate the non-vol RAM when it is being read. This is done in the same manner as the handshaking flags and further illustrates the flexibility a designer is given with the additional bandwidth of the DP8344's CPU.
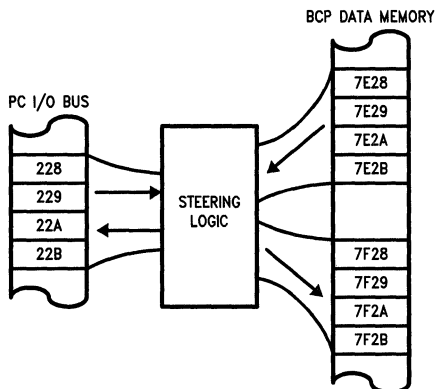
TL/F/10488–39

FIGURE 4-19. MPA Register Array Implementation

Smart Alec Microcode

The Smart Alec application software written for the personal computer (EMU, file transfers, etc . . .) is architected around a defined interface between Smart Alec and the System Unit (the 8088 and its peripheral devices). The hardware portion of this interface was discussed in a previous section. The software portion of this interface is the microcode written for the 8X305 processor. For the following discussion, the software and hardware are viewed as a single interface function. All of the Smart Alec application software was written around this interface. When configured in the Smart Alec mode the MPA becomes this interface. The method of communication between Smart Alec and the System Unit will be discussed briefly in the next section. A more exhaustive discussion on this interface is given in the Smart Alec manual.

Smart Alec and the System unit communicate through the dual four byte register array. The System Unit issues commands to Smart Alec by writing to this array. This register array is viewed by the System Unit as four I/O locations (228h–22Bh). Each I/O location corresponds to one eight bit word. When the system Unit issues a command the first byte, word 0, is defined as the command number and logical device. The next three bytes, word 1 through word 3, are defined as arguments for the command. The number of arguments associated with an individual command varies from zero to three. Twenty-three commands are used in the communication between the System Unit and Smart Alec. The upper three bits of each command specify the logical device to be referenced by the command. To begin a command the System Unit program sets word 0 equal to the logical device and the command number. It also provides any necessary arguments in word 1 through word 3, and sets the command flag. The command flag is continually being polled by the 8X305 processor when it is not busy managing the higher priority 5250 communications interface. When it detects the setting of this flag by the System Unit, it will read the data from the register array and execute the command. Once the command has been executed, the 8X305 will place the resulting data into the other side of the register array and clear the command flag. The System Unit program has been continually polling this flag and after seeing it cleared reads the result from the register array. The command flag can only be set by the System Unit. This is done by toggling bit 0 at I/O location 22Eh. The command flag can only be cleared by the Smart Alec's 8X305.

The Smart Alec board was designed at DCA after the IRMA product. It is obvious from the additional commands that steps were taken to improve the performance of the interface with the System Unit. An action stack was generated to hold address pairs that denoted where the screen buffer had been modified and with what type of modification. Also read multiple commands were added to speed up data transfer through the interface. While this did improve the performance of the interface it still contains the inherent limitations of not dual porting data memory.

MPA Implementation

The smart alec interface on the MPA board operates essentially in the same manner as described above. The System Unit I/O accesses to the Smart Alec register array space are transferred to two locations in the BCP's data memory. One location is for System Unit reads of the array (7E28h–7E2Bh), the other is for System Unit writes to the array (7F28h–7F2Bh). Different BCP memory locations were used because the register array on the Smart Alec card actually contains eight byte wide registers (four for System Unit reads and four for System Unit writes) in hardware.

The command flag is implemented using a 74LS74 on the Smart Alec board, hence the setting and clearing by toggling a bit in the control register at 22Eh (this clocks the flip-flop). This function was implemented on the MPA using an external PAL and the bi-directional interrupt pin, BIRQ. Also the MPA takes advantage of the fact that the Smart Alec software accesses the I/O locations in exactly the same

fashion for each command. This is done because the Smart Alec emulation program, EMU, was written in the C programming language. It accesses the Smart Alec I/O registers by calling an assembly language subroutine to perform the command/data and handshaking flag communications. This assembly routine writes to the I/O locations 228h through 22Bh, toggles the command flag, and then reads the state of the command flag (bit 7 at location 22Eh) until it returns low. If there is a write to the Smart Alec I/O space 228h–22Fh, then a PAL issues an interrupt to the BCP via the BIRQ input. The BCP then reads other outputs of that PAL to determine to which of those sixteen locations the write occured. If it is to 228h–22Bh then the MPA will assert the bit which tells the System Unit that the command flag is set. The MPA then waits for a System Unit write to I/O location 22Eh with the set command flag bit (bit 0 at 22Eh) low. The MPA then sets an internal command flag. It is this internal command flag that tells the MPA's smart alec task routine that an actual command has been issued by the System Unit. This was necessary to allow the MPA to emulate the function of a flip flop using only its data RAM.

The commands from the System Unit are executed in the smart alec task routine. This routine is a foreground scheduled task in the MPA Kernel. The smart alec task routine first checks to see if the non-vol RAM is being read. If so it supplies the necessary data in bit position 6 at I/O location 22Fh. If the non-vol RAM is not being read, the smart alec task routine then determines if a command is present. If so the command is decoded and executed by the appropriate command routine. In most cases, the appropriate physical device will have to be determined in order to access the correct session control page, or SCP, and the correct screen buffer for each active session. The SCP contains status and control information for each of the seven possible sessions. During the command execution the required status is calculated by calling the status update subroutine. The command's result and the calculated status are then placed in the appropriate memory locations (7E28h–7E2Bh). After this is complete, the task clears the command flags and returns program control to the Kernel.

There are three separate code modules used to allow the MPA to emulate the Smart Alec Interface.

1. power-up initialization routine
2. BIRQ interrupt routine
3. smart alec task routine

These three routines will be discussed in the following section. For clarity, the term "smart alec" is capitalized when referring to DCA products and lower case when referring to the MPA software that was written to emulate the interface. *Figure 4-20* gives a graphical representation of where these routines fit into the software architecture of the MPA.

MPA smart alec Power-Up Initialization Routine

The smart alec power-up initialization routine is called by the housekeeping task if it detects that the smart alec bit was set in the MPA configuration register. The smart alec initialization routine is titled sa_init in the MPA source code. This routine initializes the memory locations and DP8344 internal registers that are used by the smart alec emulation code. It also unmasks the BIRQ interrupt and schedules the smart_alec_task in the MPA Kernel. The memory locations that are initialized in this routine are the blocks of mem-
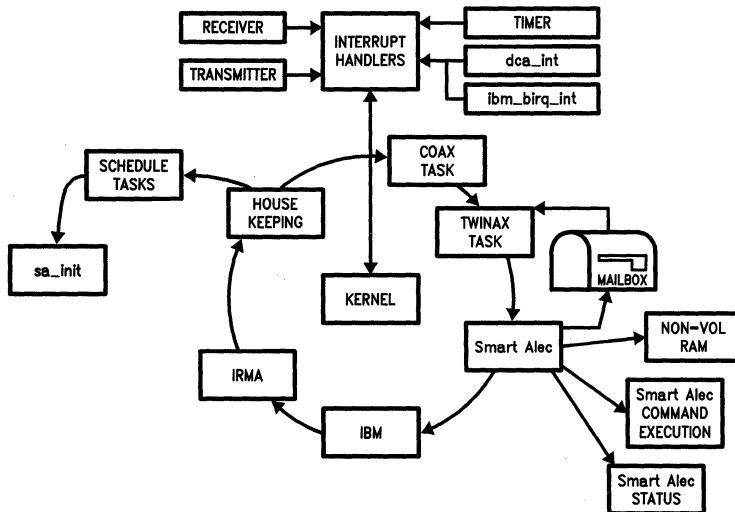
TL/F/10488–40

FIGURE 4-20. MPA Software Block Diagram in Smart Alec Mode

ory that corresponds to the contents of the emulated non-vol RAM, the memory locations used to emulate the dual register array and the flag registers, the locations on the seven session control pages that EMU controls, and the device control page memory locations that control the logical to physical mapping for the multiple sessions.

The sa_init routine also initializes some internal BCP registers. It does this because other routines, such as the dca BIRQ interrupt routine, must access certain stored values very quickly to keep their expedient time under 2 micro-seconds. The execution time in these routines is decreased if data needed in the routine is kept in internal registers rather than in data memory. The final function of the sa_init routine is to schedule the sa_task routine. This is done by loading the task number into the accumulator and calling the schedule_task routine. After this, program control is returned to the Kernel.

MPA dca Interrupt Routine

The second code module required to emulate the Smart Alec Interface is the dca birq interrupt routine. The MPA board uses the extra CPU bandwidth of the BCP to reduce the discrete components needed to provide the command and flag function. It does this by letting the CPU decode part of the System Unit I/O access address and by letting it provide the set function of this flag. The BCP code necessary for this is the BIRQ interrupt routine whose source module is DCA_INT.BCP. The BIRQ interrupt is generated when the System Unit writes to any I/O locations from 220h to 22Fh. It would have been more expedient in this case to only have interrupts generated on writes to I/O location 22Eh. However, the MPA hardware also supports the IBM emulation programs. The MPA implementation for the IBM interface requires interrupts to be generated from more System Unit I/O access locations, so to reduce external hardware, interrupts are generated for a sixteen byte I/O block. This flexibility of hardware design further illustrates the usefulness of the extra CPU bandwidth of the DP8344.

When the BCP detects the BIRQ interrupt, it transfers program control to the dca_int routine. The function of this routine is to set the command flag or provide the appropriate serial non-vol RAM data. There is a section of code in the dca_int routine that does the same function as that described above, but is called from the other routines and not by the external BIRQ interrupt. To increase performance, the interrupt routines check the BIRQ flag in the CCR register before they return. If the flag is set, it calls the dca_fast_birq section of the dca_int routine. Here the same operations as described earlier are performed except for the saving and restoring of the environment. The dca_fast_birq routine does not have to provide this function because the other routines are doing it. This substantially decreases the number of instructions executed and therefore improves the overall performance.

MPA smart alec Task Routine

The majority of the Smart Alec emulation takes place in the smart alec task routine. This routine is run in the foreground as a scheduled task. Therefore the decision to execute this routine is dependent only on the MPA's task scheduler and is not impacted by the System Unit. In reality the task is run many times between System Unit accesses because the code execution speed of the BCP is much greater than that of the 8088. The smart alec task routine, appropriately labeled in the source code as "sa_task", contains four major sections. These sections are non-vol RAM routine, the command execution routines, the physical session determination routine, and the status update routine.

When the smart alec task is called, it first checks to see if EMU has tried to read the non-vol RAM. If so, it determines how many times it was read (the non-vol RAM is read serially) so it can adjust the serial bit stream it is providing to EMU. If no accesses have been made to the non-vol RAM, the smart alec task checks to see if a command is present. If there is no command present (the internal command flag is not set), the task returns to the Kernel. If a command is present, the lower five bits of the command word is decoded to determine which of the twenty three commands has been issued by the System Unit. Program control is then transferred to the specific routine that executes the command. In most cases, the first thing done in the specific command routine is to determine which session the command was issued to. This is done by decoding the top three bits in the command word to determine what logical session the command was issued for. After that, the corresponding physical session is determined and pointers are set up in internal registers to point to the appropriate session control page and screen buffer. Both of these functions are performed in the tw_sa_spset subroutine. Using this information the command is executed and the required status is calculated. The status is calculated in the tw_sa_all_status routine if full status is required. If only common status is required, the tw_sa_common_status routine is called instead. After this, the resulting data is placed in the section of memory that is accessed by the System Unit when it reads the I/O locations 228h–22Bh. The smart alec task then clears both the internal and Smart Alec command flags and returns program control to the Kernel.

Loader

The Loader is a PC program developed to load, configure and run the MPA hardware. It is written in Microsoft "C" 5.0 and the source code is included on the distribution diskettes.

The MPA system uses soft-loaded instruction memory. Upon system reset, the BCP is idle, RIC [Remote Interface Control register] is set to point to instruction memory, and the program counter is pointing to zero. Thje MPA_LDR program or some other PC program must be used to configure, load, and run the MPA system at this point.

The first step by the Loader is to set RIC to 32h, setting fast buffered write, latched read mode. The memory segment register is then loaded, and unless otherwise specified, the loader will default to CEh. This value locates the memory array at CE000h. RIC is then set to point to data memory, and the 8k array is functionally tested. Next, RIC is set to point to instruction memory and a selftest program is loaded. The BCP is then run by asserting the start bit in RIC. Finally, the result of the selftest program is read from location 2DCh. If the result was good, the loader then loads the user program.

The loader operates with RIC in a straight forward manner; .BCX files or .FMT files are read in from the PC disk and are interpreted for address or instruction records. Address records tell the loader to force the BCP's program counter to that instruction location. Instruction records are 16-bit values that are loaded into the MPA board 8 bits at a time. The BCP handles concatenating the bytes into 16-bit instructions and writing them into instruction memory. The entire memory array can be loaded and verified in this manner. Any data RAM image needed may also be loaded, although the PC has access only to the 8k dual-port provided by the MPA hardware.

Selftest

The Selftest function of the Loader serves two basic functions. First, the hardware verification is useful in manufacturing and assuring a working product reaches our customers. Secondly, the example of the selftest code provides an example of testing not just the MPA but the BCP itself. Selftest returns a number of codes based on the specific result of the testing operation.

SECTION FIVE — OPERATION

System Requirements

Since the MPA system emulates three emulation systems, the system requirements are the same as the sum of the systems that are emulated. In DCA modes, the system will not use interrupts, but if IBM is operated, the PC IRQ level 2 is required. IRQ2 is selected by jumper JP9. IRQ4 and IRQ3 can be selected with JP7 and JP8, respectively. In all modes, an 8k block of dual-port RAM must be located somewhere within the PC's memory space. The MPA Loader program (LD) will initialize this block at whatever location is desired. In all modes, the I/O space requirements are the total of the IBM and DCA requirements. This means I/O locations 220h–22Fh and 2D0h–2DFh are required.

For execution space, the LD requirements are minimal (less than 64k). The amount of free RAM available for an emulator depends obviously on the particular emulation package. The MPA system does not use any resident software of its own accord.

Installation

The first step in using the MPA is installing the circuit board in an IBM PC or close compatible. The MPA installs in the usual way: please be sure that the power is OFF and the system unit is unplugged.

- Remove the cover by following the directions supplied by the manufacturer.
- Remove the end plate from the system unit in the slot desired for the MPA.
- Remove the MPA from its anti-static bag, and holding it by the edges, install it in the open, normal slot.

- If the MPA will be used for Twinax operation, determine if the MPA should be the terminating device in the multi-drop environment. If it is NOT the terminator, remove jumpers JP5 and JP6. The factory default is terminate.
- Replace the screw from the end plate removal to hold the MPA firmly in place.
- Install any 3270 coax type "A" port cable to the rear BNC connector.
- For twinax operation, install the Twinax Adapter cable to the MPA by inserting the 9-Pin D-Sub-miniature connector onto the mating connector on the rear panel, and connect the twinax cable(s) to the Tee connector.
- Close the system unit and replace all screws, etc . . . according to the manufacturers instructions.

Invoking LD

The LD program loads .FMT and .BCX files into the MPA board, configures and runs the system. It is invoked by:

    LD filename[.ext] −m[=]mode   [options]

Invoking the program with no arguments produces a comprehensive help screen. Selecting −m=irma, alec, or ibm will automatically configure the hardware for use with DCA's E78 or E78 plus, DCA's EMU, or IBM's entry level or Version 3.0 products.

Batch files nscirma.bat and nscibm.bat will initialize the MPA system for operation as the IRMA DSI or the IBM 3278/79 emulation board, respectively. These packages may be invoked any time after the loader is finished with no special considerations. Follow their respective manuals for directions.

SECTION SIX — DEVELOPMENT ENVIRONMENT

The environment used for development of the MPA consists of a few readily available, relatively inexpensive tools. The hardware was first proto-typed with the Capstone Technology CT-104 BCP Demonstration/Development card. The software was developed with the National Semiconductor BCP Assembler, and tested with Capstone's ICC (Integrated Coax Controller) and NAM (Network Analysis Monitor). In addition, Azure Technologies' coax and Twinax scope products were used. Debugging was accomplished with ISID, Capstone's debugger/monitor which we modified for use with the MPA software model. (For more information concerning ISID contact Capstone Technology.) For particularly difficult interrupt problems a Hewlett Packard model 1630 logic analyzer was used to monitor instruction execution and host accesses.

The CT-104 board was modified through the wire-wrap area to approximate the hardware design. This wire-wrap card allowed us to get a working version of the hardware design very quickly, since most of the circuitry was already there. In some development projects, it is often faster to go directly to pcbs as a proto-type run. This process has advantages in speed when the device is large and complex, but often debugging is quite messy with multi-layer pcbs, not to mention expensive. Since the CT-104 has the major functional blocks already and the wire wrap area is large, the wire-wrap time was minimal, thus allowed us to easily debug the hardware.

A majority of the logic for the DCA and IBM interfaces is implemented in Programmable Array Logic. We used the abel program from DATA I/O to prepare the JEDEC files for programming the devices.

Software development was done on IBM PCs with the National Semiconductor DP8344 Assembler. The assembler allows relocatable code, equate files, macros, and many other "large CPU" features that make using it a pleasure. The modularity of the software design allowed us to use multiple coders and a single "system integrator" who linked the modules and handled system debugging. The assem-bler adapts well to large projects like this because of its relocation capability. The Microsoft MAKE utility was used to provide the system integrator with an automated way of keeping up with source modules' dependencies and changes. The BRIEF™ text editor from UnderWare™ was used for editing. This editor allowed us to invoke the National Semiconductor DP8344 Assembler from within the editor and to locate and correct bugs quickly. Finally, an ethernet LAN allowed the software development team to share files and update each other quickly and efficiently. These tools are not all necessary, but are common enough to be useful in illustrating a typical environment.

The BCP's sophistication and advanced development tools made the MPA development project proceed at a much greater rate than is possible with other comparable solutions.

The debugger allows a developer to load and run code on the target system, set breakpoints, examine and modify instruction or data memory. Early configurations were accomplished using the standard DOS DEBUG tool, but once the MPA Loader program (LD) was operational, configuration and loading was accomplished through it. Testing the coax code was done with the ICC and NAM tools available also from Capstone. The ICC can send command strings via coax to the target system to verify protocol compliance without jeopardizing a live controller port. The NAM can be used to monitor coax activity and produce reports on the line activity.

The HP logic analyzer was attached to the target system to monitor the instruction accesses and data bus activity on the target card. This information is helpful in finding interrupt problems that the debugger cannot. Using ICLK from the BCP to sample the BCP instruction address and data busses allows one to monitor instruction execution. Symbolic disassembly can be done with the HP logic analyzers if one has the time to enter the BCP instruction set by hand. Even with only basic instruction groups decoded the information was very helpful.

Section 3
ISDN Components

Section 3 Contents

Introduction To National Semiconductor
Basic Access I. C. Set

In developing the architecture of this ISDN chip set, National's major objective has been to create a flexible set of building blocks which provide elegant and cost-effective solutions for a wide range of applications. With just a few highly integrated devices, a broad spectrum of ISDN equipment can be designed, ranging from Central Office and PBX line cards to X.25 and ISDN Terminals and telephones, PC and Terminal Adapters, packet-mode statistical multiplexers, NT-1's and other ISDN equipment.

One of the keys to this flexibility is the concept that device functions in the chip set should be specifically aligned with the first 3 layers of the ISO 7 layer Protocol Reference Model. Thus, National's chip set has a distinct partitioning of functions into several transceivers which provide the bit-level transport for Layer 1, (the Physical Layer), while the functions of Layer 2, (the Data Link Layer), and Layer 3, (the Network Layer), are supported entirely by a single micro-processor. All devices in the chip set, together with other standard components such as COMBOs, can be interconnected via a common serial interface without the need for any "glue" components. The result is a very elegant architecture offering many advantages including the following:

- A high degree of modularity with minimal component count
- The same transceiver at both ends of a loop
- No interrupts for D-Channel flow control
- Powerful Packet buffer management

Other chip set architectures, which divide a layer into some functions in one device and the rest in other devices, are unable to offer all these advantages.

ISDN Chip Set Partitioning

ISO Layer	National	Others	
4–7	NS32322		
3	HPC16400	Chip C	Chip B
2		Chip B	
1	TP3401 DASL or TP3410 EC or TP3420 SID	Chip A	Chip A

NSC Solutions for Layer 1

National's solution for Layer 1 consists of 3 CMOS transceivers, which cover a wide variety of twisted-pair applications for ISDN Basic Access. Each transceiver is capable of transmitting and receiving 2 'B' channels plus 1 'D' channel, and has mode selections to enable it to operate at either end of the loop.

Transceiver Number 1

The TP3400 Digital Adapter for Subscriber Loops (DASL) is a low-cost burst-mode transceiver for 2 wire PBX and private network loops up to 6 kft in range. Scrambled Alternate Mark Inversion coding is used, together with adaptive equalization and timing-recovery, to ensure low bit error rates on a wide variety of cable types. All activation and loop timing control circuitry is also included.

Transceiver Number 2

The TP3410 Echo-canceller Family is a set of 2-wire transceivers designed to meet the rigorous requirements of the 'U' interface. Derived from a common basic architecture, these devices will be compatible with the line-code and framing structure specifications of various PTT administrations and with the U.S. standard.

Transceiver Number 3

The TP3420 'S' Interface Device (SID), is a 4-wire transceiver which includes all the Layer 1 functions specified in CCITT Recommendation I.430. In addition, the TP3420 includes noise filtering and adaptive equalization, as well as a high resolution digital phase-locked loop, to provide transmission performance far in excess of that specified in I.430. All Activation and 'D' channel access sequences are handled automatically without the need to invoke any action from a microprocessor.

Digital Chip to Chip Interfaces

To retain the flexibility of interfacing components from this chip set with a variety of other products, two digital interfaces are provided on each device. One is for the synchronous transfer of 'B' and 'D' channel information in any of several popular multiplexed serial formats. This means that National's chip to chip interface is all encompassing of proprietary frame structures such as the IOM, IDL, ST-BUS and more.

A second interface, for device mode control, e.g. power up/down, setting loopbacks etc., uses the popular MICRO-WIRE/PLUS™. MICROWIRE/PLUS is a synchronous serial data transfer between a microcontroller and one or more peripheral devices. National's HPC and COPS™ microcontroller families, together with a broad range of peripheral devices, support this interface, which is also easy to emulate with any microprocessor.

NSC ISDN Transceiver Chip Set

TL/X/0008–1

**Popular Frame Structures
Addressed by NSC ISDN Devices**

TL/X/0008–2
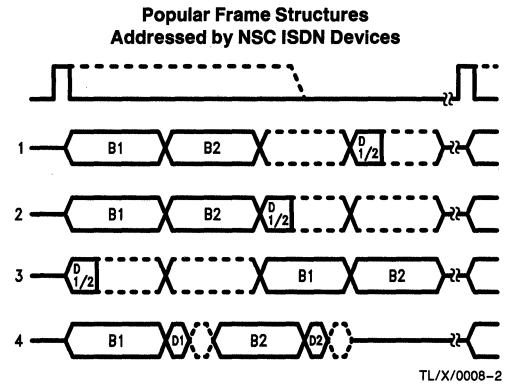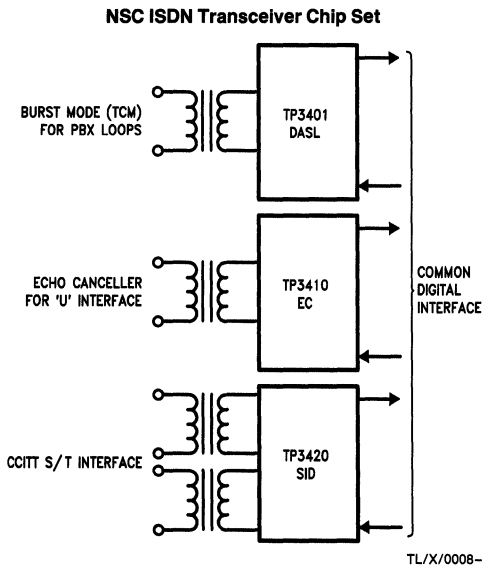
NSC Solutions for Layers 2 and 3

National has developed an extremely powerful solution for implementing various protocols for both Layer 2 (Data Link Layer) and Layer 3 (Network Layer), including X.25 LAPB and LAPD (Q.921 and Q.931), together with the capability of several packet-mode Terminal Adaption schemes*. A single device incorporates all the processing for these functions: the HPC16400. One of National's growing family of 16-bit single chip CMOS microcontrollers, the HPC16400 is based on a high-speed (17 MHz) 16-bit CPU "core". To this core has been added 2 full HDLC formatters supported by DMA to external memory, and a UART.

This set of features makes the HPC16400 an ideal processor for running all the functions of an ISDN Terminal Adapter, TE or telephone, or the communications port of a high-end terminal. In a typical application, one of the HDLC channels may be dedicated to running the LAPD protocol in the 'D' channel, while the other provides packet-mode access to one of the 'B' channels. The UART would serve as an RS232 interface running at any of the standard synchronous or asynchronous rates up to 128 kbaud. A serial interface decoder allows either or both HDLC controllers to be directly interfaced to any of the 3 Layer 1 transceivers or to a variety of backplanes, line-card controllers and other devices using time-division multiplexed serial interfaces.

Because of the large ROM and RAM requirements for Layer 3 and the Control Field procedures of Layer 2 in LAPB and LAPD protocols, the HPC16400 has 256 bytes of RAM and no internal ROM for storage of user variables. Packet storage RAM and all user ROM is off-chip, this is by far the most

*For example, as per DMI Modes 2 and 3.

cost-effective and flexible combination. A multiplexed bus to external memory provides direct addressing for up to 64 kbytes of memory, and on-chip I/O allows for expanded addressing for up to 544 kbytes of memory.

The HDLC controllers on the HPC16400 allow continuous HDLC data rates up to 4.1 Mb/s to be used. In addition to handling all Layer 2 framing, the HDLC circuitry includes automatic multiple address recognition to support, for example, multiple TEI's in LAPD. Furthermore, the DMA controller provides several register sets for packet RAM management with minimal CPU intervention, including "chaining" of successive packets. This integrated design achieves a high throughput of packet data without the need for costly FIFO's and external interrupts, thereby minimizing the impact of packet handling on CPU time.

In many applications a number of other peripheral functions must also be provided, such as sensing switches or scanning a small keyboard, interfacing to a display controller etc. A number of extra I/O ports and a MICROWIRE/PLUS serial data expansion interface are available on the HPC16400 to service these functions. In addition, 4 user configurable 16 bit timer-counters simplify the many time-outs required to manage such a system, including the default timers specified in the various protocol specifications.

Terminal adaption consistent with the CCITT V.110 method, which is based on a synchronous 80 bit frame, is readily implemented with another member of the HPC family, the HPC16040. Around the standard core CPU, the 16040 has on board I/O and 4 additional PWM timers, a UART, 4k of ROM and 256 bytes of RAM.

HPC16400 Simplified Block Diagram
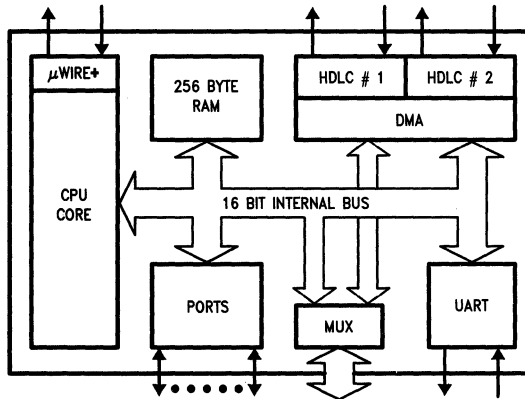
TL/X/0008-3

NSC Solutions: Systems Level

Building an ISDN TE or TA

Shown below is a typical application of the chip set in a Basic Access TE, which offers one voice channel and an RS232 interface to support an external terminal. The TP3420 'S' Interface Device ensures that the system is compatible with any 'S' or 'T' standard jack socket and provides the multiplexing for the other devices operating in the 'B' and 'D' channels. All timing for the TE is derived by the TP3420 from the received line signal. In a typical application, LAPD signalling in the 'D' channel is provided via

HDLC #1 on the HPC16400. HDLC #2 is working in conjunction with the UART to provide an X.25 or LAPD packet-mode in a 'B' channel at 64 kb/s. Terminal Adaption of both the data and the terminal handshaking signals is performed by the HPC16400 via the UART and HDLC controller #2, which can use either of the 'B' channels. DMI modes 2 and 3 (for a single channel) can be supported using this method, with the necessary data buffers set up in internal RAM. The other 'B' channel is occupied by the TP3054/7 PCM COMBO providing the digitized voice channel.
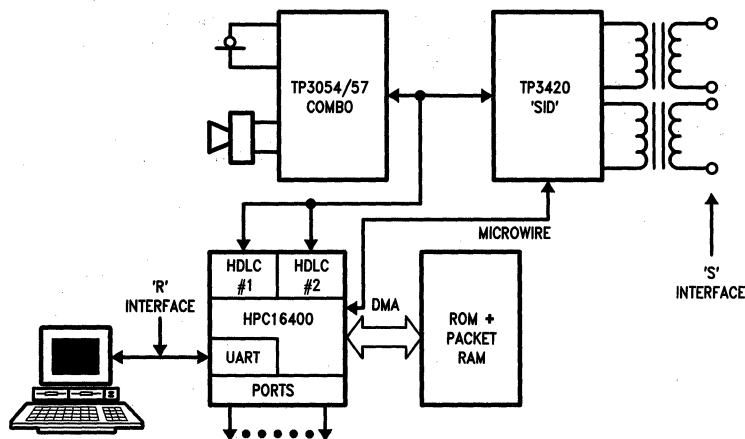
TL/X/0008-4

PBX 2 Wire Terminals

The following example shows how simple it is to convert an 'S' Interface terminal, which requires 2 twisted pairs, to a terminal using only a single pair by replacing the

TP3420 SID with a TP3401 DASL. The clean partitioning of device functions makes this possible with no other changes to the design.
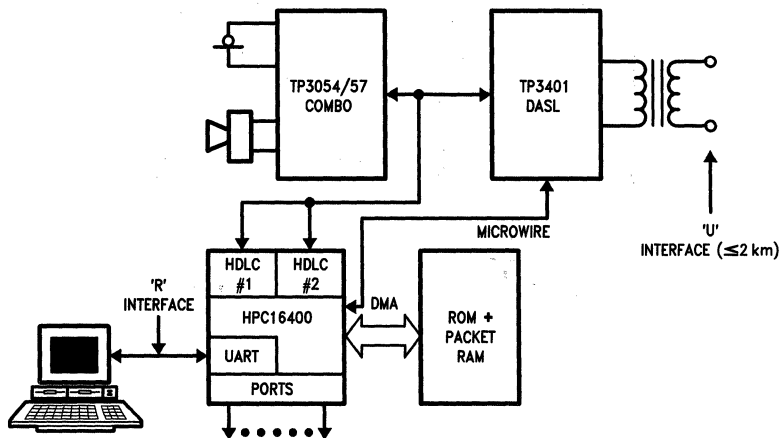
TL/X/0008-5

NSC Solutions: Systems Level

Basic Access Line Cards

For operation on a line card in a C.O., PABX or NT-2, each of the 3 transceiver devices can be set to operate as the timing master for the loop, being synchronized to the system clock and controlling all loop frame timing. If programmable time-slot assignment is required, the TP3155 TSAC provides 8 individually programmable frame sync pulse outputs locked to a common frame marker. 'B' channels can be interfaced to standard backplane interfaces, while 'D' chan-nels can be either multiplexed on and off the card for processing or can undergo Layer 2 processing on the card itself.

For the latter method, one HPC16400 handles Layer 2 framing for 2 basic access lines. In this manner, packets are first identified as data or signalling type by analysis of the SAPI field, with data packets being routed separately to a packet switch access node. If required, signalling packets can undergo protocol conversion in the HPC to an existing internal switch control protocol.
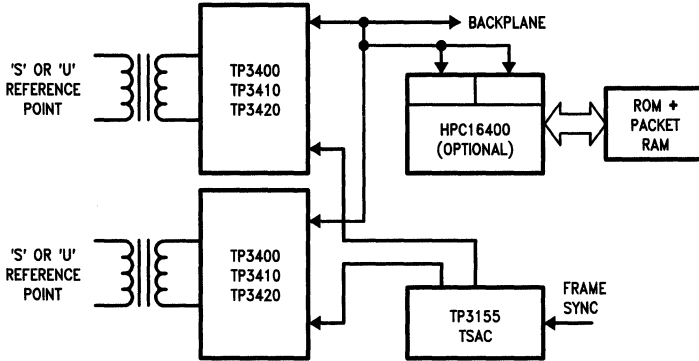
TL/X/0008–6

Building an NT-1

An NT-1 Network Termination is defined as a Layer 1 device only, which converts the 2-wire long-haul 'U' interface to the limited distance 4-wire 'S' interface. It has no capability for intercepting higher layers of the 'D' channel protocol. As such, it is built simply by connecting a TP3420 SID, configured in NT (or Master) mode, to a TP3410 Echo-canceller operating in Slave mode. Sharing a common 15.36 MHz crystal, these devices pass 'B' and 'D' channel information across the standard 4-wire interface. Layer 1 maintenance protocols across both the 'U' and the 'S/T' interfaces, which are as of yet not definitively specified by most adminstrations, may be handled by a low cost 4-Bit COPS™ Microcontroller via its Microwire Interface.

TL/X/0008–7

3

![National Semiconductor logo] **National Semiconductor**
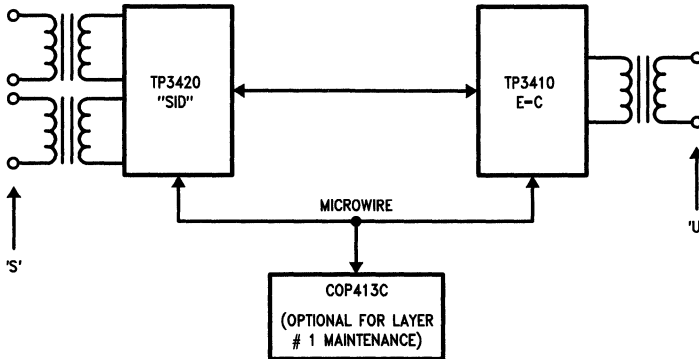
TP3401, TP3402
DASL Digital Adapter for Subscriber Loops

General Description

The TP3401 and TP3402 are complete monolithic transceivers for data transmission on twisted pair subscriber loops. They are built on National's double poly microCMOS process, and require only a single $+5$ Volt supply. Alternate Mark Inversion (AMI) line coding, in which binary '1's are alternately transmitted as a positive pulse then a negative pulse, is used to ensure low error rates in the presence of noise with lower emi radiation than other codes such as Bi-phase (Manchester).

Full-duplex transmission at 144 kb/s is achieved on a single twisted wire pair using a burst-mode technique (Time Compression Multiplexed). Thus the device operates as an ISDN 'U' Interface for short loop applications, typically in a PBX environment, providing transmission for 2 B channels and 1 D channel. On #24 cable, the range is at least 1.8 km (6k ft).

System timing is based on a Master/Slave configuration, with the line card end being the Master which controls loop timing and synchronisation. All timing sequences necessary for loop activation and de-activation are generated on-chip.

Selection of Master and Slave mode operation is programmed via the Microwire Control Interface.

A 2.048 MHz clock, which may be synchronized to the system clock, controls all transmission-related timing functions.

For the TP3401, this clock must be provided from an external source; the TP3402 includes an oscillator circuit requiring an external crystal.

Features

Complete ISDN PBX 2-Wire Data Transceiver including:
- 2 B plus D channel interface for PBX U' Interface
- 144 kb/s full-duplex on 1 twisted pair using Burst Mode
- Loop range up to 6 kft (#24AWG)
- Alternate Mark Inversion coding with transmit filter and scrambler for low emi radiation
- Adaptive line equalizer
- On-chip timing recovery, no external components
- Standard TDM interface for B channels
- Separate interface for D channel
- 2.048 MHz master clock
- Driver for line transformer
- 4 loop-back test modes
- Single $+5V$ supply
- MICROWIRE™ compatible serial control interface
- 20-pin package
- Applications in:
  PBX Line Cards
  Terminals
  Regenerators

Block Diagram
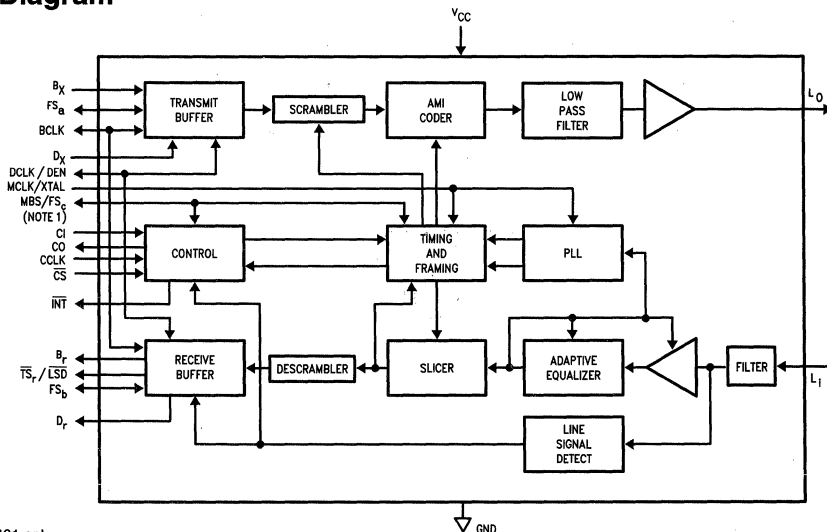
Note 1: TP3401 only.

TL/H/9264-1

![National Semiconductor logo] **National Semiconductor**

TP3410 ISDN Basic Access Echo-Cancelling 2B1Q U Transceiver

General Description

The TP3410 is a complete monolithic transceiver for ISDN Basic Access data transmission at either end of the U interface. Fully compatible with ANSI specification T1.601-1988, it is built on National's advanced 1.5 micron double-metal CMOS process, and requires only a single +5V power supply. A total of 160 kbps full-duplex transmission on a single twisted-pair is provided, with user-accessible channels including 2 'B' channels, each at 64 kbps, 1 'D' channel at 16 kbps, and an additional 4 kbps for loop maintenance. 12 kbps of bandwidth is reserved for framing. 2B1Q Line coding is used, in which pairs of binary bits are coded into 1 of 4 quantum levels for transmission at 80k symbols/sec (hence 2 Binary/1 Quaternary). To meet the very demanding specifications for <1 in 10e7 Bit Error Rate even on long loops with crosstalk, the device includes 2 Adaptive Digital Signal Processors, 2 Digital Phase-locked Loops and a controller for automatic activation.

The digital interface on the device can be programmed for compatibility with either of two types of control interface for chip control and access to all spare bits. In one mode a Microwire serial control interface is used together with a 2B+D digital interface which is compatible with the Time-division Multiplexed format of PCM Combo devices and backplanes. This mode allows independent time-slot assignment for the 2 B channels and the D channel. Alternatively, the GCI (General Circuit Interface) may be selected, in which the 2B+D data is multiplexed together with control, spare bits and loop maintenance data on 4 pins.

Features

- 2 'B' + 'D' channel 160 kbps transceiver for LT and NT
- Meets ANSI T1.601-1988 U.S. Standard

- 2B1Q line coding with scrambler/descrambler
- Range exceeds 18 kft bridge taps
- >70 dB adaptive echo-cancellation and equalization
- On-chip timing recovery, no precision external components
- Direct connection to small line transformer (27 mH)
- Automatic activation controller
- Selectable digital interface formats:
  - TDM with time-slot assigner plus MICROWIRE™ control
  - GCI (General Circuit Interface), or
  - IDL (Inter-chip Digital Link)
- Backplane clock DPLL allows free-running XTAL
- Elastic data buffers meet Q.502 wander/jitter for Slave-slave mode on PBX Trunk Cards
- EOC and spare bits access with automatic validation
- 2 block error counters
- 6 loopback test modes
- Single +5V supply, 300 mW active power
- 5 mW idle mode with line signal detector

Applications

- LT, NT-1, NT-2 Trunks, U-TE's Regenerators etc.
- Easy Interface to:
  - Line Card Backplanes
  - "S" Interface Device TP3420
  - Codec/Filter Combos .... TP3054/7 and TP3075/6
  - LAPD Processor HPC16400
  - HDLC Controller TP3451

Block Diagram
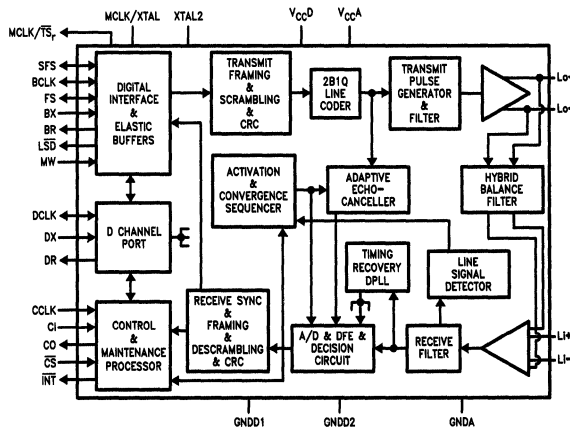
Note: Pin names show Microwire mode.

TL/H/9151–1

3

National Semiconductor

TP3420 ISDN Transceiver S/T Interface Device

General Description

The TP3420 S Interface Device (SID™) is a complete monolithic transceiver for data transmission on twisted pair subscriber loops. It is built on National's advanced 1.5 micron double-metal CMOS process, and requires only a single +5V supply. All functions specified in CCITT recommendation I.430 for ISDN basic access at the 'S' and 'T' interfaces are provided, and the device can be configured to operate either in a TE (Terminal Equipment), in an NT-1 or NT-2 (Network Termination) or as a PABX line-card or trunk-card device.

As specified in I.430, full-duplex transmission at 192 kb/s is provided on separate transmit and receive twisted wire pairs using inverted Alternate Mark Inversion (AMI) line coding. Various channels are combined to form the 192 kb/s aggregate rate, including 2 'B' channels, each of 64 kb/s, and 1 'D' channel at 16 kb/s. In addition, the TP3420 provides the 800 b/s "S1" & "Q"multiframe channels for Layer 1 maintenance.

All I.430 wiring configurations are supported by the TP3420 SID, including the "passive bus" for up to 8 TE's distributed within 200 meters of low capacitance cable, and point-to-point and point-to-star connections up to at least 1500 meters (24AWG). Adaptive receive signal processing enables the device to operate with low bit error rates on any of the standard types of cable pairs commonly found in premise wiring installations when tested with the noise sources specified in I.430.

Features

- Single chip 4 wire 192 kb/s transceiver
- Provides all CCITT I.430 layer 1 functions
- Exceeds I.430 range: 1.5 km point-to-point
- Adaptive and fixed timing options for NT-1
- Clock resynchronizer and elastic buffers for NT-2/LT
- Slave-slave mode for NT-2 trunks
- Multiframe channel for layer 1 maintenance
- Selectable system interface formats
- MICROWIRE™ compatible serial control interface
- Single +5V supply
- 20-pin package

Applications

- Same Device for NT, TE and PBX Line Card
- Point-to-Point Range Extended to 1.5 km
- Point-to-Multipoint for all I.430 Configurations
- Easy Interface to:

LAPD Processor	HPC16400
Terminal Adapter	HPC16400
Codec/Filter COMBO™	TP3054/7 and TP3075/6
"U" Interface Device	TP3410
Line Card Backplanes—No External PLL Needed	

- Line Monitor Mode for Test Equipment

Block Diagram

TL/H/9143–1

PRELIMINARY

![National Semiconductor logo] **National Semiconductor**
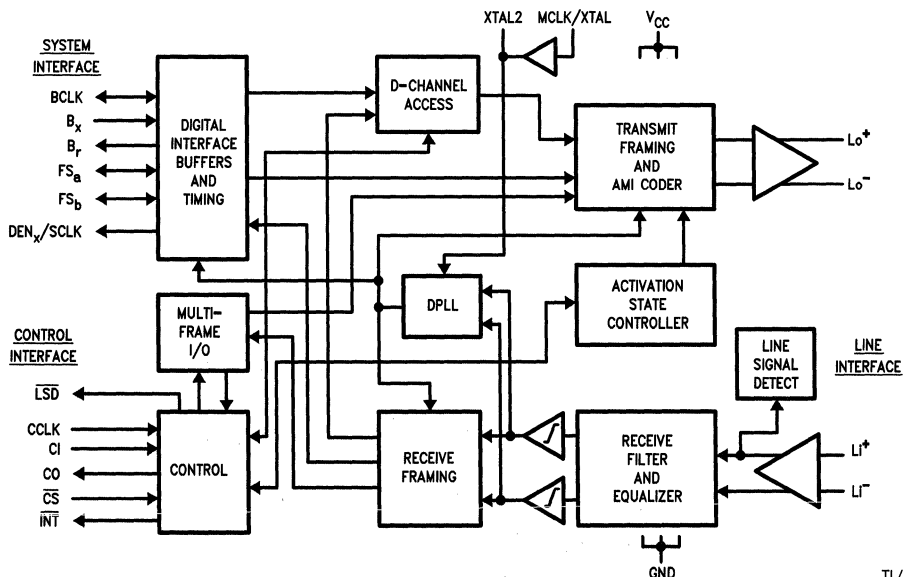
HPC16083/HPC26083/HPC36083/HPC46083/ HPC16003/HPC26003/HPC36003/HPC46003 High-Performance microControllers

General Description

The HPC16083 and HPC16003 are members of the HPC™ family of High Performance microControllers. Each member of the family has the same core CPU with a unique memory and I/O configuration to suit specific applications. The HPC16083 has 8k bytes of on-chip ROM. The HPC16003 has no on-chip ROM and is intended for use with external direct memory. Each part is fabricated in National's advanced microCMOS technology. This process combined with an advanced architecture provides fast, flexible I/O control, efficient data manipulation, and high speed computation.

The HPC devices are complete microcomputers on a single chip. All system timing, internal logic, ROM, RAM, and I/O are provided on the chip to produce a cost effective solution for high performance applications. On-chip functions such as UART, up to eight 16-bit timers with 4 input capture registers, vectored interrupts, WATCHDOG™ logic and MICRO-WIRE/PLUS™ provide a high level of system integration. The ability to address up to 64k bytes of external memory enables the HPC to be used in powerful applications typically performed by microprocessors and expensive peripheral chips. The term "HPC16083" is used throughout this datasheet to refer to the HPC16083 and HPC16003 devices unless otherwise specified.

The microCMOS process results in very low current drain and enables the user to select the optimum speed/power product for his system. The IDLE and HALT modes provide further current savings. The HPC is available in 68-pin PLCC, LCC, LDCC, PGA and 84-Pin TapePak® packages.

Features

- HPC family—core features:
  - 16-bit architecture, both byte and word
  - 16-bit data bus, ALU, and registers
  - 64k bytes of external direct memory addressing
  - FAST—200 ns for fastest instruction when using 20.0 MHz clock, 134 ns at 30 MHz
  - High code efficiency—most instructions are single byte
  - 16 x 16 multiply and 32 x 16 divide
  - Eight vectored interrupt sources
  - Four 16-bit timer/counters with 4 synchronous outputs and WATCHDOG logic
  - MICROWIRE/PLUS serial I/O interface
  - CMOS—very low power with two power save modes: IDLE and HALT
- UART—full duplex, programmable baud rate
- Four additional 16-bit timer/counters with pulse width modulated outputs
- Four input capture registers
- 52 general purpose I/O lines (memory mapped)
- 8k bytes of ROM, 256 bytes of RAM on chip
- ROMless version available (HPC16003)
- Commercial (0°C to +70°C), industrial (−40°C to +85°C), automotive (−40°C to +105°C) and military (−55°C to +125°C) temperature ranges

Block Diagram (HPC16083 with 8k ROM shown)

TL/DD/8801–1

3

National Semiconductor
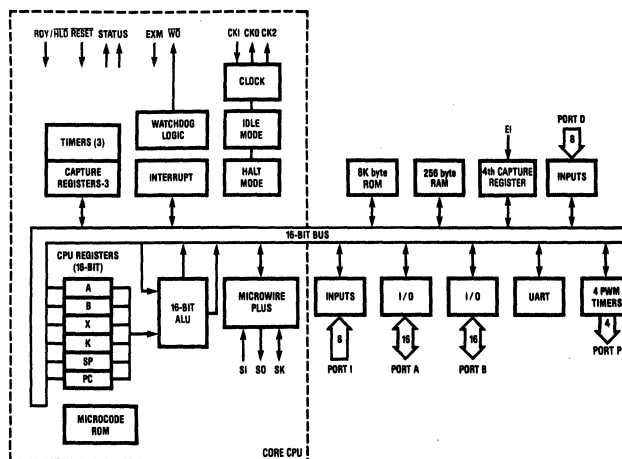
HPC16400/HPC36400/HPC46400 High-Performance Communications microController

General Description

The HPC16400 is a member of the HPC™ family of High Performance microControllers. Each member of the family has the same identical core CPU with a unique memory and I/O configuration to suit specific applications. Each part is fabricated in National's advanced microCMOS technology. This process combined with an advanced architecture provides fast, flexible I/O control, efficient data manipulation, and high speed computation.

The HPC16400 has 4 functional blocks to support a wide range of communication application-2 HDLC channels, 4 channel DMA controller to facilitate data flow for the HDLC channels, programmable serial interface and UART.

The serial interface decoder allows the 2 HDLC channels to be used with devices using interchip serial link for point-to-point & multipoint data exchanges. The decoder generates enable signals for the HDLC channels allowing multiplexed D and B channel data to be accessed.

The HDLC channels manage the link by providing sequencing using the HDLC framing along with error control based upon a cyclic redundancy check (CRC). Multiple address recognition modes, and both bit and byte modes of operation are supported.

The HPC16400 is available in 68-pin PLCC, LCC, LDCC and 84-pin TapePak® packages.

Features

■ HPC family—core features:
 — 16-bit data bus, ALU, and registers
 — 64 kbytes of external direct memory addressing
 — FAST!—20.0 MHz system clock
 — High code efficiency
 — 16 x 16 multiply and 32 x 16 divide
 — Eight vectored interrupt sources
 — Four 16-bit timer/counters with WATCHDOG logic
 — MICROWIRE/PLUS serial I/O interface
 — CMOS—low power with two power save modes
■ Two full duplex HDLC channels
 — Optimized for X.25 and LAPD applications
 — Programmable frame address recognition
 — Up to 4.65 Mbps serial data rate
 — Built in diagnostics
 — Synchronous bypass mode
■ Programmable interchip serial data decoder
■ Four channel DMA controller
■ UART—full duplex, programmable baud rate (up to 208.3 kBaud)
■ 544 kbytes of extended addressing
■ Easy interface to National's DASL, 'U' and 'S' transceivers—TP3400, TP3410 and TP3420
■ Commercial (0°C to 70°C) Industrial (−40°C to +85°C) and military (−55°C to +125°C) temperature ranges

Block Diagram
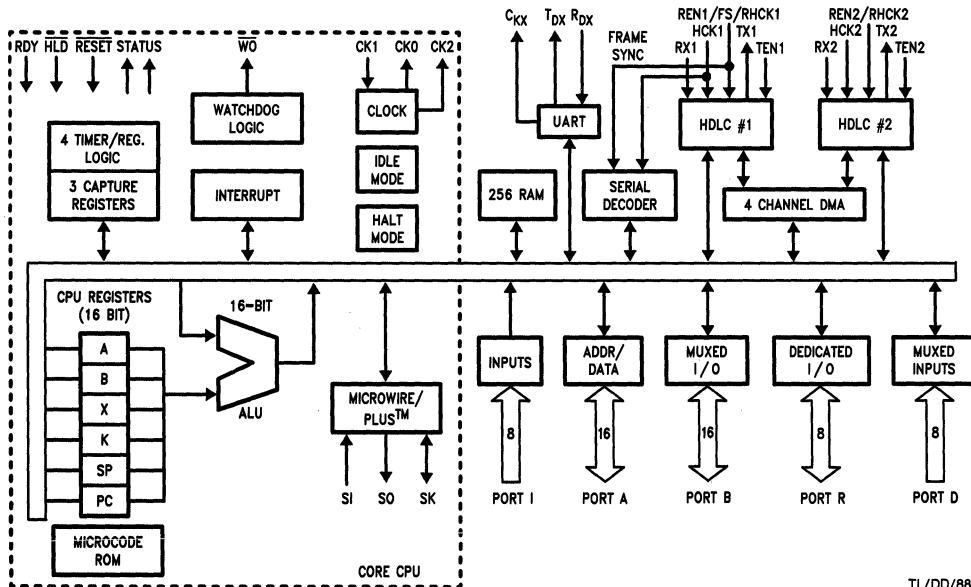
TL/DD/8802-1

"B" Channel, or DS0 Channel

A "B" (for Basic) channel is a 64 kb/s full-duplex transparent data channel. It is octet (=byte) oriented, that is it can be considered as a channel bearing 8k octets/sec. "B" channels are synchronized to the network and are generally circuit-switched (not packet switched). The 64 kb/s rate is also known as a DS0 interface.

"D" Channel

The "D" channel is a packet-mode message-oriented channel on which the data-link layer (layer 2) protocol is carried in HDLC frames. At a basic access point the "D" channel runs at 16 kb/s, while at a primary access point it runs at 64 kb/s. (There is no reason why a "D" channel could not be defined to run at even higher speeds, e.g., 1.544 or 2.048 Mb/s, though that does not seem to be a part of current standardization work.)

Three types of data may be handled by a "D" channel:

1. Type "s" (signaling) using layer 3 of the LAPD protocol.
2. Type "p" (packet) user's packet-oriented data.
3. Type "t" (telemetry) data, typically alarms and energy monitoring functions operating at a low scan rate.

   The data type is identified by the SAPI (Service Access Point Identifier) in the HDLC extended address field.

Basic Access to the ISDN

Two independent "B" channels (B1 and B2) together with a "D" channel operating at 16 kb/s form the basic access structure. A minimum transmission rate of 144 kb/s full duplex is therefore required for basic access transport, although in some applications additional bits are used for localized functions.

Figure 1 shows the names of the functional blocks and interfaces as defined in CCITT specifications.

The 'U' interface is the single twisted pair loop between a customer's premises and the local central office. To transmit 144 kb/s or more full-duplex over this link, which may be several miles long and have over 40 dB of attenuation of the data signal, requires a complex transceiver. Adaptive echo-cancellation techniques are necessary and, although the transmission format is not yet specified by CCITT, considerable work is in progress in the U.S. T1D1.3 ISDN Study Group to establish a standard for North America. 160 kb/s is the likely transmission rate, while the line code will be 2B1Q.

The 'S' interface passes the same 2 'B' channels and the 'D' channel on to the terminals, together with some additional bits used for synchronization, contention control in the 'D' channel, and other housekeeping functions. CCITT specification I.430 defines the physical layer of this interface. A transceiver is required for transmission at the 192 kb/s bit rate, over separate transmit and receive twisted pairs (which already exist in both office and residential telephone wiring within the premises in many countries). Alternate Mark Inversion coding is used.

2 additional pairs are specified as an option, 1 for power and 1 for spare, making this an 8 wire interface. A plug and jack have been standardized so that the 'S' interface can be a "universal portability point" for ISDN terminals from any manufacturer in the world.

Primary Access to the ISDN

Primary access is provided at a DS1 interface, consisting of either:

1. Twenty-three "B" channels plus one 64 kb/s "D" channel at 1.544 Mb/s (North America), or :
2. Thirty "B" channels plus one 64 kb/s "D" channel at 2.048 Mb/s (Europe and Rest of World).

CCITT specification I.431 defines the multiplexing and control schemes for primary access.

TE—Terminal Equipment

Two sub-groups of terminals are defined:

1. TE-1 is a full ISDN terminal which is synchronized to the network channels (not just the far-end terminal) and uses LAPD signaling. It connects to the ISDN at the "S" reference point, which is intended to be the point in the network at which any type of **basic** access terminal can be connected, i.e., the "portability" point.
2. TE-2 is a non-ISDN terminal, generally one of today's asynchronous or synchronous terminals operating at rates < 64 kb/s. This includes terminals which have RS232C, RS449, V.21, V.24, V.35, X.21 or X.25 packet-mode interfaces. Each type of interface must be adapted from the "R" reference point to the "S" reference point by means of a Terminal Adapter (TA).

TA—Terminal Adapter

A terminal adapter converts either asynchronous or synchronous data from non-ISDN terminals into data which is synchronized with ISDN B or D channels. The data rate must be adapted by means of stuffing extra bits in a prescribed pattern into the bit stream to adapt the data rate to 64 kb/s.

Terminal adaption also requires the conversion of modem handshaking signals to ISDN compatible signaling, and currently there are 2 competing schemes: either using LAPD in the D channel (i.e. out-of-band signaling) or applying LAPD-type messages but passing them end-to-end via the B channel (i.e. in-band). There are strong arguments for both methods, mostly concerned with how signaling is converted at the boundary between an ISDN and today's network ("interworking"), and it remains to be seen which will win as a standard.

NT—Network Termination

The NT terminates the network at the user's end of the 2 wire loop at the customer's premises. It converts the "U" interface to the "S" and "T" interface (see *Figure 1*) and acts as the "master" end of the user's passive bus. B and D channels must pass transparently through the NT, and there is no capability for intercepting LAPD messages in the NT.

Thus a typical NT for **basic** access will consist of an 'S' interface transceiver and a 'U' interface transceiver connected back-to-back with appropriate power supplies and fault monitoring capability.

An NT can also be an intelligent controller such as a PABX, LAN access node, or a terminal cluster controller.

3

LT—Line Termination

Typically, the LT consists of the "U" interface tranceiver and power feeding functions on the ISDN line card. These functions must interface to the switch at the "V" reference point, which is not currently being standardized by CCITT. It could be a proprietary backplane interface or a nationally specified interface which would allow the LT to be physically and electrically separated from the switch.

ISO Layered Protocol Model

The ISO (International Standards Organization) has defined a 7 layer model structure which describes convenient break points between various parts of the hardware and software in any data communications system.

Layer 1: Physical layer, that is the hardware which transports bits across interfaces. This includes ISDN transceivers, modems etc., power supplies, methods of activating and de-activating a transmission link, and also the transmission medium itself, such as wire, fiber, plugs and sockets, etc.

Layer 2: Data Link layer, which describes a basic framing structure and bit assignments to enable higher layer messages to be passed across a physical link. HDLC framing, addressing and error control are the major elements of this layer in ISDN.

Layer 3: Network layer, that is those parts of a message associated with setting-up, controlling and tearing-down a call through the network. These are all software control functions, and generally this is the highest layer in the ISO protocol model which is considered in chip development.

The top 4 layers relate to the structure of the actual application programs;

Layer 4: Transport layer, concerned with defining sources and destinations within an operating system for the transfer of application programs.

Layer 5: Session layer.

Layer 6: Presentation layer.

Layer 7: Application layer.

These layers are generally running on a high level machine, and discussion regarding this machine is outside the scope of this document.

LAPD

Link Access Protocol in the "D" channel is the name given to the packet-mode signaling protocol defined in CCITT specs Q920 and Q921 for the data link layer (layer 2) and Q930 and Q931 for the network layer (layer 3 in the ISO 7 layer reference model). At layer 2, LAPD uses the HDLC framing format. This protocol defines the bits, bytes and sequence of states necessary between the user and the network to establish, control and terminate calls using any of the 100 or more types of services which may be available via an ISDN. If the users at both ends of the call are connected to the ISDN and there is a through path for the D channel then end-to-end call control is available.

Because of this extensive range of services, implementation of full LAPD requires considerable memory and processing power. Standards work has recently focused on definition of a minimal subset of LAPD to cover the basic requirements of call control.

Activation/De-activation

Activation is the process of powering up the 'S' and 'U' interfaces from their standby (i.e. de-activated) states and sending specific signals across the interfaces to get the whole loop synchronized to the network. A small state machine in each TE and the NT controls this sequence of events, and uses timers to ensure that, if the activation attempt should fail for any reason, the user or network is alerted. At the end of a call an orderly exit from the network is effected by sending de-activation sequences before any equipment can power-down.

FIGURE 1. The ISDN Interfaces
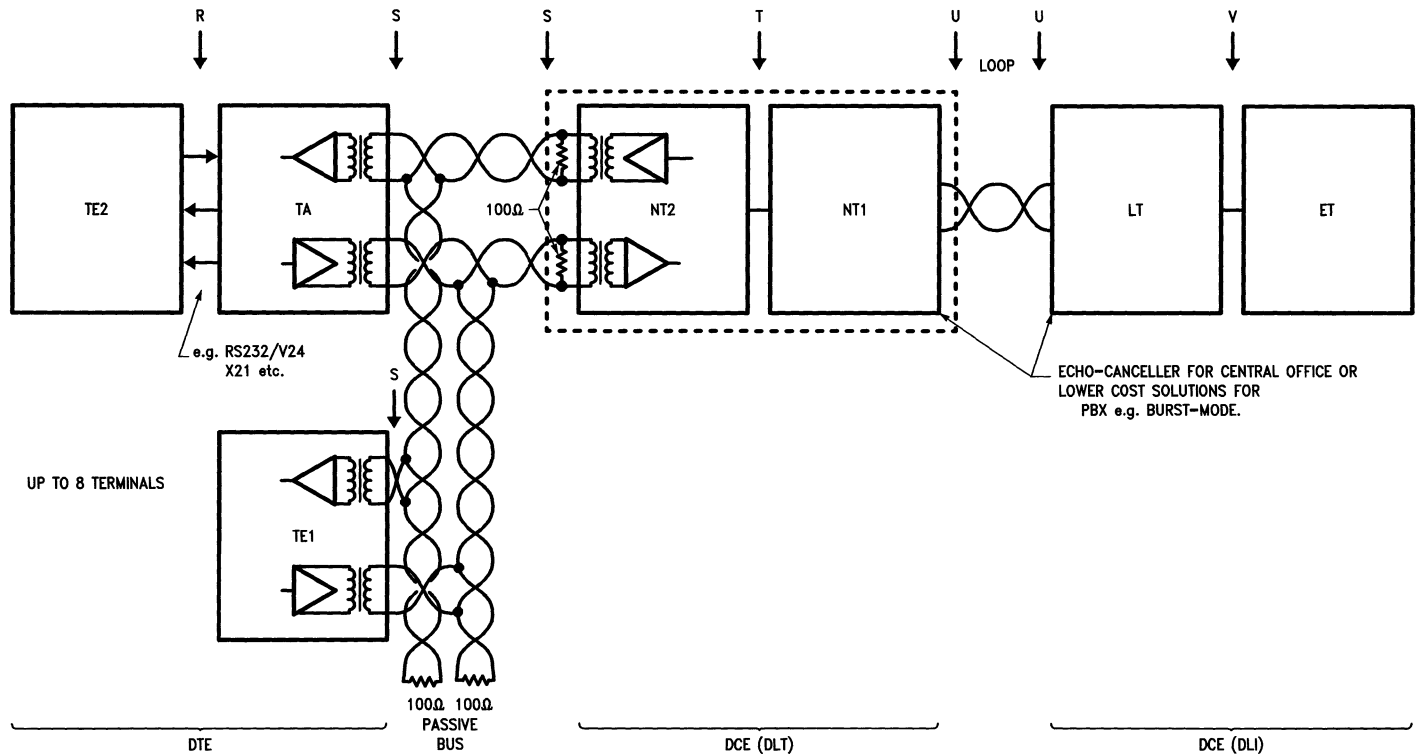
TE: Terminal Equipment
TA: Terminal Adaptor (Protocol Conversion & Rate Adaption for Non-ISDN Terminals)
NT2: Network Termination 2 (Protocol for Link Control, MUX/DEMVX etc)
NT1: Network Termination 1 (Loop Transceiver, Power Extraction)
LT: Line Termination (Loop Transceiver, Power Feed)
ET: Exchange Termination (Protocol Handling, MUX/DEMUX, Switching)

3

Section 4

UARTs

4

Section 4 Contents

National Semiconductor

INS8250, INS8250-B
Universal Asynchronous Receiver/Transmitter

General Description

Each of these parts function as a serial data input/output interface in a microcomputer system. The system software determines the functional configuration of the UART via a TRI-STATE® 8-bit bidirectional data bus.

The UART performs serial-to-parallel conversion on data characters received from a peripheral device or a MODEM, and parallel-to-serial conversion on data characters received from the CPU. The CPU can read the complete status of the UART. Status information reported includes the type and condition of the transfer operations being performed by the UART, as well as any error conditions (parity, overrun, framing, or break interrupt).

The UART includes a programmable baud rate generator that is capable of dividing the timing reference clock input by divisors of 1 to ($2^{16}-1$), and producing a 16 × clock for driving the internal transmitter logic. Provisions are also included to use this 16 × clock to drive the receiver logic. The UART includes a complete MODEM-control capability and a processor-interrupt system. Interrupts can be programmed to the user's requirements minimizing the computing required to handle the communications link.

National's INS8250 universal asynchronous receiver transmitter (UART) is the unanimous choice of almost every PC and add-on manufacturer in the world. The INS8250 is a programmable communications chip available in a standard 40-pin dual-in-line and a 44-pin PCC package. The chip is fabricated using N-channel silicon gate technology.

Features

- Easily interfaces to most popular microprocessors.
- Adds or deletes standard asynchronous communication bits (start, stop, and parity) to or from serial data stream.
- Holding and shift registers eliminate the need for precise synchronization between the CPU and the serial data.
- Independently controlled transmit, receive, line status, and data set interrupts.
- Programmable baud generator allows division of any input clock by 1 to ($2^{16} - 1$) and generates the internal 16 × clock.
- Independent receiver clock input.
- MODEM control functions (CTS, RTS, DSR, DTR, RI, and DCD).
- Fully programmable serial-interface characteristics:
  — 5-, 6-, 7-, or 8-bit characters
  — Even, odd, or no-parity bit generation and detection
  — 1-, 1½-, or 2-stop bit generation
  — Baud generation (DC to 56k baud).
- False start bit detection.
- Complete status reporting capabilities.
- TRI-STATE TTL drive capabilities for bidirectional data bus and control bus.
- Line break generation and detection.
- Internal diagnostic capabilities:
  — Loopback controls for communications link fault isolation
  — Break, parity, overrun, framing error simulation.
- Fully prioritized interrupt system controls.

Connection Diagram

TL/C/9329-1

Table of Contents

1.0 Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Temperature Under Bias	0°C to +70°C
Storage Temperature	−65°C to +150°C

All Input or Output Voltages
with Respect to V_{SS} −0.5V to +7.0V

Power Dissipation 400 mW

Note: *Maximum ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended and should be limited to those conditions specified under DC electrical characteristics.*

2.0 DC Electrical Characteristics

T_A = 0°C to +70°C, V_{CC} = +5V ±5%, V_{SS} = 0V, unless otherwise specified.

Symbol	Parameter	Conditions	INS8250		INS8250-B		Units
			Min	Max	Min	Max	
V_{ILX}	Clock Input Low Voltage		−0.5	0.8	−0.5	0.8	V
V_{IHX}	Clock Input High Voltage		2.0	V_{CC}	2.0	V_{CC}	V
V_{IL}	Input Low Voltage		−0.5	0.8	−0.5	0.8	V
V_{IH}	Input High Voltage		2.0	V_{CC}	2.0	V_{CC}	V
V_{OL}	Output Low Voltage	I_{OL} = 1.6 mA on all (Note 1)		0.4		0.4	V
V_{OH}	Output High Voltage	I_{OH} = −1.0 mA (Note 1)	2.4		2.4		V
$I_{CC}(AV)$	Avg. Power Supply Current (V_{CC})	V_{CC} = 5.25V, T_A = 25°C No Loads on output SIN, DSR, DCD, CTS, RI = 2.4V All other inputs = 0.4V		80		80	mA
I_{IL}	Input Leakage	V_{CC} = 5.25V, V_{SS} = 0V		±10		±10	μA
I_{CL}	Clock Leakage	All other pins floating. V_{IN} = 0V, 5.25V		±10		±10	μA
I_{OZ}	TRI-STATE Leakage	V_{CC} = 5.25V, V_{SS} = 0V V_{OUT} = 0V, 5.25V 1) Chip deselected 2) WRITE mode, chip selected		±20		±20	μA

Capacitance T_A = 25°C, V_{CC} = V_{SS} = 0V

Symbol	Parameter	Conditions	Min	Typ	Max	Units
C_{XIN}	Clock Input Capacitance			15	20	pF
C_{XOUT}	Clock Output Capacitance	f_c = 1 MHz		20	30	pF
C_{IN}	Input Capacitance	Unmeasured pins		6	10	pF
C_{OUT}	Output Capacitance	returned to V_{SS}		10	20	pF

Note 1: Does not apply to XOUT.

4

3.0 AC Electrical Characteristics $T_A = 0°C$ to $+70°C$, $V_{CC} = +5V \pm 5\%$

Symbol	Parameter	Conditions	INS8250		INS8250-B		Units
			Min	Max	Min	Max	
t_{ADS}	Address Strobe Width		90		120		ns
t_{AH}	Address Hold Time		0		60		ns
t_{AR}	$\overline{RD}$/RD Delay from Address	(Note 1)	110		110		ns
t_{AS}	Address Setup Time		110		110		ns
t_{AW}	$\overline{WR}$/$\overline{WR}$ Delay from Address	(Note 1)	160		160		ns
t_{CH}	Chip Select Hold Time		0		60		ns
t_{CS}	Chip Select Setup Time		110		110		ns
t_{CSC}	Chip Select Output Delay from Select	@100 pF loading (Note 1)		200		200	ns
t_{CSR}	$\overline{RD}$/RD Delay from Chip Select	(Note 1)	110		110		ns
t_{CSS}	Chip Select Output Delay from Strobe		0	150	0	150	ns
t_{CSW}	$\overline{WR}$/WR Delay from Select	(Note 1)	160		160		ns
t_{DH}	Data Hold Time		60		100		ns
t_{DS}	Data Setup Time		175		350		ns
t_{HZ}	$\overline{RD}$/RD to Floating Data Delay	@100 pF loading (Note 3)	0	150	0	150	ns
t_{MR}	Master Reset Pulse Width		10		10		μs
t_{RA}	Address Hold Time from $\overline{RD}$/RD	(Note 1)	50		50		ns
t_{RC}	Read Cycle Delay		1735		1735		ns
t_{RCS}	Chip Select Hold Time from $\overline{RD}$/RD	(Note 1)	50		50		ns
t_{RD}	$\overline{RD}$/RD Strobe Width		175		350		ns
t_{RDA}	Read Strobe Delay		0		0		ns
t_{RDD}	$\overline{RD}$/RD to Driver Disable Delay	@100 pF loading (Note 3)		150		250	ns
t_{RVD}	Delay from $\overline{RD}$/RD to Data	@100 pF loading		250		300	ns
t_{WA}	Address Hold Time from $\overline{WR}$/WR	(Note 1)	50		50		ns
t_{WC}	Write Cycle Delay		1785		1785		ns
t_{WCS}	Chip Select Hold Time from $\overline{WR}$/WR	(Note 1)	50		50		ns
t_{WDA}	Write Strobe Delay		50		50		ns
t_{WR}	$\overline{WR}$/WR Strobe Width		175		350		ns
t_{XH}	Duration of Clock High Pulse	External Clock (3.1 MHz Max.)	140		140		ns
t_{XL}	Duration of Clock Low Pulse	External Clock (3.1 MHz Max.)	140		140		ns
RC	Read Cycle = $t_{AR} + t_{DIW} + t_{RC}$		2000		2205		ns
WC	Write Cycle = $t_{DDA} + t_{DOW} + t_{WC}$		2100		2305		ns
Baud Generator							
N	Baud Divisor		1	$2^{16}-1$	1	$2^{16}-1$	
t_{BHD}	Baud Output Positive Edge Delay	100 pF Load		250		250	ns
t_{BLD}	Baud Output Negative Edge Delay	100 pF Load		250		250	ns
t_{HW}	Baud Output Up Time	$f_X = 3$ MHz, $\div 3$, 100 pF Load	330		330		uns
t_{LW}	Baud Output Down Time	$f_X = 2$ MHz, $\div 2$, 100 pF Load	425		425		ns
Receiver							
t_{RINT}	Delay from $\overline{RD}$/RD (RD RBR or RD LSR) to Reset Interrupt	100 pF Load		1000		1000	ns
t_{SCD}	Delay from RCLK to Sample Time			2000		2000	ns
t_{SINT}	Delay from Stop to Set Interrupt			2000		2000	ns

Note 1: Applicable only when $\overline{ADS}$ is tied low.

Note 2: Charge and discharge time is determined by V_{OL}, V_{OH} and the external loading.

3.0 AC Electrical Characteristics T_A = 0°C to +70°C, V_{CC} = +5V ±5% (Continued)

Symbol	Parameter	Conditions	INS8250		INS82C50-B		Units
			Min	Max	Min	Max	
Transmitter							
t_{HR}	Delay from $\overline{WR}$/WR (WR THR) to Reset Interrupt	100 pF Load		1000		1000	ns
t_{IR}	Delay from $\overline{RD}$/RD (RD IIR) to Reset Interrupt (THRE)	100 pF Load		1000		1000	ns
t_{IRS}	Delay from Initial INTR Reset to Transmit Start			16		16	BAUDOUT Cycles
t_{SI}	Delay from Initial Write to Interrupt			50		50	BAUDOUT Cycles
t_{SS}	Delay from Stop to Next Start			1000		1000	ns
t_{STI}	Delay from Stop to Interrupt (THRE)			8		8	BAUDOUT Cycles
Modem Control							
t_{MDO}	Delay from $\overline{WR}$/WR (WR MCR) to Output	100 pF Load		1000		1000	ns
t_{RIM}	Delay to Reset Interrupt from $\overline{RD}$/RD (RD MSR)	100 pF Load		1000		1000	ns
t_{SIM}	Delay to Set Interrupt from MODEM Input	100 pF Load		1000		1000	ns

4.0 Timing Waveforms (All timings are referenced to valid 0 and valid 1)

External Clock Input (3.1 MHz Max.)
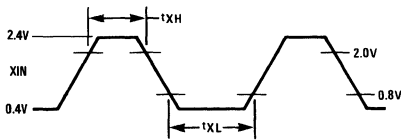
AC Test Points

TL/C/9329-2
TL/C/9329-3

Note 1: The 2.4V and 0.4V levels are the voltages that the inputs are driven to during AC testing.

Note 2: The 2.0V and 0.8V levels are the voltages at which the timing tests are made.
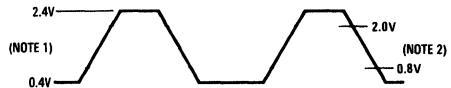
$\overline{BAUDOUT}$ Timing
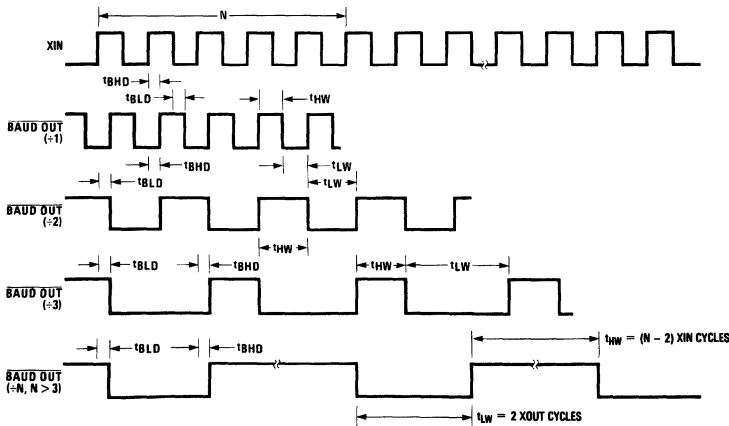
TL/C/9329-4

4.0 Timing Waveforms (Continued)
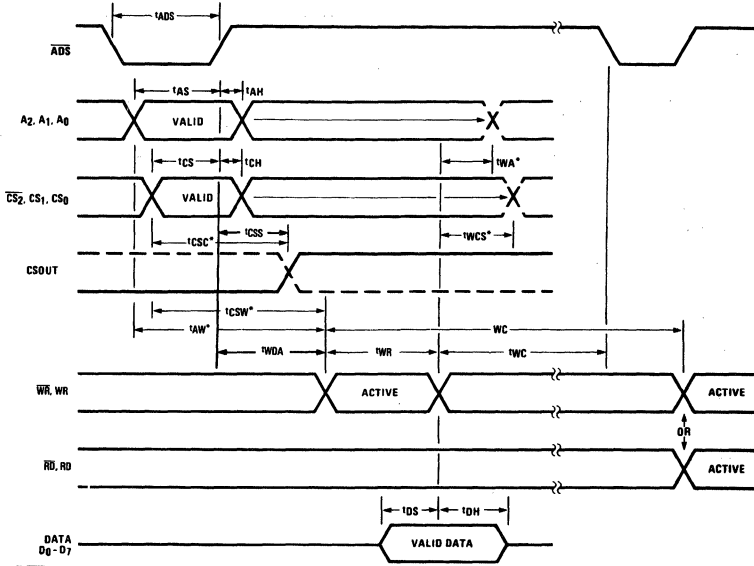
Write Cycle

*Applicable Only When $\overline{ADS}$ is Tied Low.

TL/C/9329–5

Read Cycle
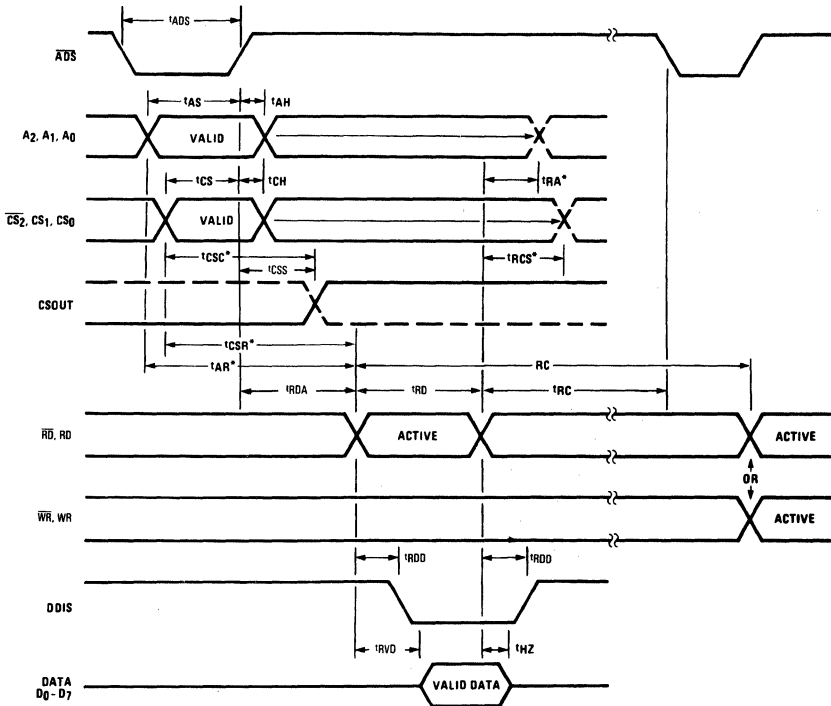
*Applicable Only When $\overline{ADS}$ is Tied Low.

TL/C/9329–6

4.0 Timing Waveforms (Continued)

Receiver Timing
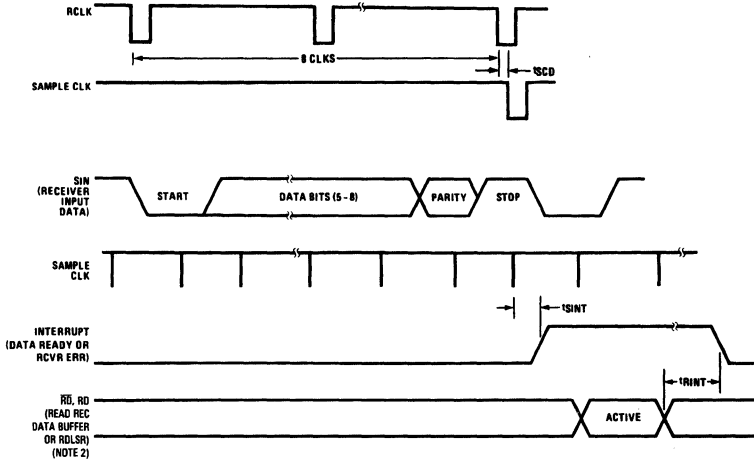
TL/C/9329-7

Transmitter Timing
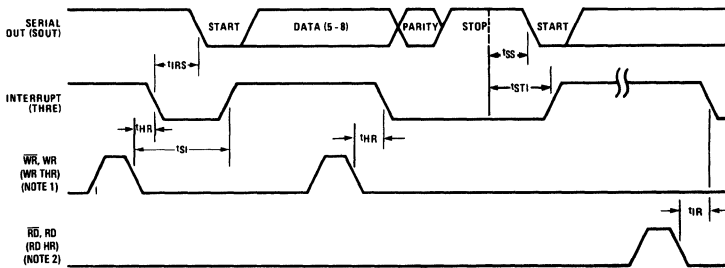
TL/C/9329-8

MODEM Controls Timing
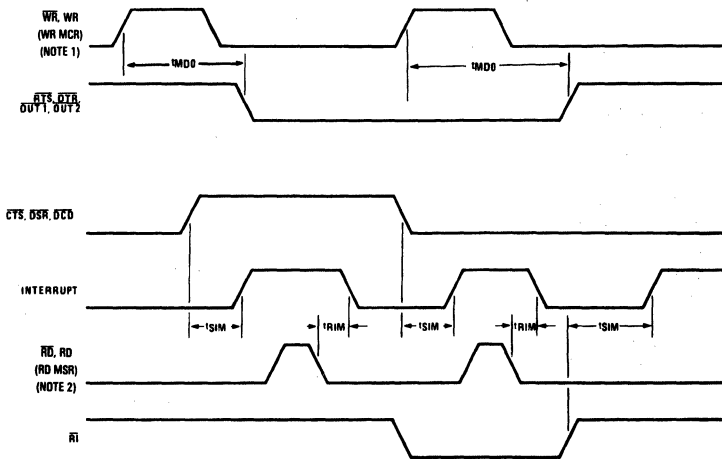
TL/C/9329-9

Note 1: See Write Cycle Timing

Note 2: See Read Cycle Timing

5.0 Block Diagram
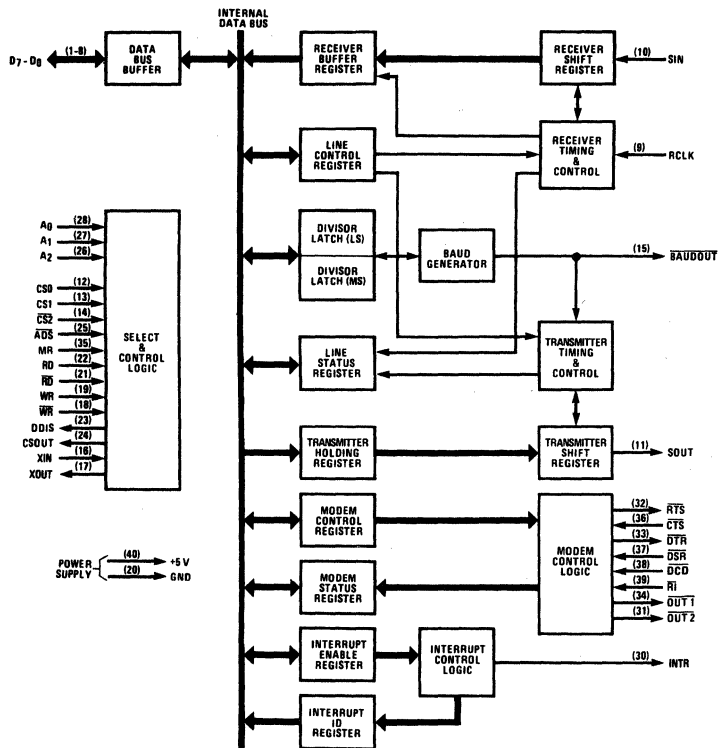
TL/C/9329-10

Note: Applicable pinout numbers are included within parenthesis.

6.0 Pin Descriptions

The following describes the function of all UART, pins. Some of these descriptions reference internal circuits.

In the following descriptions, a low represents a logic 0 (0V nominal) and a high represents a logic 1 (+2.4V nominal).

6.1 INPUT SIGNALS

Chip Select (CS0, CS1, $\overline{CS2}$), Pins 12-14: When CS0 and CS1 are high and $\overline{CS2}$ is low, the chip is selected. This enables communication between the UART and the CPU. The positive edge of an active Address Strobe signal latches the decoded chip select signals, completing chip selection. If $\overline{ADS}$ is always low valid chip selects should stabilize according to the t_{CSW} parameter.

Read (RD, $\overline{RD}$), Pins 22 and 21: When RD is high or $\overline{RD}$ is low while the chip is selected, the CPU can read status information or data from the selected UART register.

Note: Only an active RD or $\overline{RD}$ input is required to transfer data from the UART during a read operation. Therefore, tie either the RD input permanently low or the $\overline{RD}$ input permanently high, when it is not used.

Write (WR, $\overline{WR}$), Pins 19 and 18: When WR is high or $\overline{WR}$ is low while the chip is selected, the CPU can write control words or data into the selected UART register.

Note: Only an active WR or $\overline{WR}$ input is required to transfer data to the UART during a write operation. Therefore, tie either the WR input permanently low or the $\overline{WR}$ input permanently high, when it is not used.

Address Strobe ($\overline{ADS}$), Pin 25: The positive edge of an active Address Strobe ($\overline{ADS}$) signal latches the Register Select (A0, A1, A2) and Chip Select (CS0, CS1, CS2) signals.

Note: An active $\overline{ADS}$ input is required when the Register Select (A0, A1, A2) signals are not stable for the duration of a read or write operation. If not required, tie the $\overline{ADS}$ input permanently low.

Register Select (A0, A1, A2), Pins 26-28: Address signals connected to these 3 inputs select a UART register for the CPU to read from or write to during data transfer. A table of registers and their addresses is shown below. Note that the state of the Divisor Latch Access Bit (DLAB), which is the most significant bit of the Line Control Register, affects the selection of certain UART registers. The DLAB must be set high by the system software to access the Baud Generator Divisor Latches.

6.0 Pin Descriptions (Continued)

DLAB	A₂	A₁	A₀	Register
0	0	0	0	Receiver Buffer (read), Transmitter Holding Register (write)
0	0	0	1	Interrupt Enable
X	0	1	0	Interrupt Identification (read only)
X	0	1	1	Line Control
X	1	0	0	MODEM Control
X	1	0	1	Line Status
X	1	1	0	MODEM Status
1	0	0	0	Divisor Latch (least significant byte)
1	0	0	1	Divisor Latch (most significant byte)

Register Addresses

Master Reset (MR), Pin 35: When this input is high, it clears all the registers (except the Receiver Buffer, Transmitter Holding, and Divisor Latches), and the control logic of the UART. The states of various output signals (SOUT, INTR, $\overline{OUT\ 1}$, $\overline{OUT\ 2}$, $\overline{RTS}$, $\overline{DTR}$) are affected by an active MR input. (Refer to Table I.).

Receiver Clock (RCLK), Pin 9: This input is the 16 × baud rate clock for the receiver section of the chip.

Serial Input (SIN), Pin 10: Serial data input from the communications link (peripheral device, MODEM, or data set).

Clear to Send ($\overline{CTS}$), Pin 36: When low, this indicates that the MODEM or data set is ready to exchange data. The $\overline{CTS}$ signal is a MODEM status input whose conditions can be tested by the CPU reading bit 4 (CTS) of the MODEM Status Register. Bit 4 is the complement of the $\overline{CTS}$ signal. Bit 0 (DCTS) of the MODEM Status Register indicates whether the $\overline{CTS}$ input has changed state since the previous reading of the MODEM Status Register. $\overline{CTS}$ has no effect on the Transmitter.

Note: Whenever the CTS bit of the MODEM Status Register changes state, an interrupt is generated if the MODEM Status Interrupt is enabled.

Data Set Ready ($\overline{DSR}$), Pin 37: When low, this indicates that the MODEM or data set is ready to establish the communications link with the UART. The $\overline{DSR}$ signal is a MODEM status input whose condition can be tested by the CPU reading bit 5 (DSR) of the MODEM Status Register. Bit 5 is the complement of the $\overline{DSR}$ signal. Bit 1 (DDSR) of the MODEM Status Register indicates whether the $\overline{DSR}$ input has changed state since the previous reading of the MODEM Status Register.

Note: Whenever the DSR bit of the MODEM Status Register changes state, an interrupt is generated if the MODEM Status Interrupt is enabled.

Data Carrier Detect ($\overline{DCD}$), Pin 38: When low, indicates that the data carrier has been detected by the MODEM or data set. The $\overline{DCD}$ signal is a MODEM status input whose condition can be tested by the CPU reading bit 7 (DCD) of the MODEM Status Register. Bit 7 is the complement of the $\overline{DCD}$ signal. Bit 3 (DDCD) of the MODEM Status Register indicates whether the $\overline{DCD}$ input has changed state

since the previous reading of the MODEM Status Register. $\overline{DCD}$ has no effect on the receiver.

Note: Whenever the DCD bit of the MODEM Status Register changes state, an interrupt is generated if the MODEM Status Interrupt is enabled.

Ring Indicator ($\overline{RI}$), Pin 39: When low, this indicates that a telephone ringing signal has been received by the MODEM or data set. The $\overline{RI}$ signal is a MODEM status input whose condition can be tested by the CPU reading bit 6 (RI) of the MODEM Status Register. Bit 6 is the complement of the $\overline{RI}$ signal. Bit 2 (TERI) of the MODEM Status Register indicates whether the $\overline{RI}$ input **signal** has changed from a low to a high state since the previous reading of the MODEM Status Register.

Note: Whenever the RI **bit** of the MODEM Status Register changes from a high to a low state, an interrupt is generated if the MODEM Status Interrupt is enabled.

V_{CC}, Pin 40: +5V supply.

V_{SS}, Pin 20: Ground (0V) reference.

6.2 OUTPUT SIGNALS

Data Terminal Ready ($\overline{DTR}$), Pin 33: When low, this informs the MODEM or data set that the UART is ready to establish a communications link. The $\overline{DTR}$ output signal can be set to an active low by programming bit 0 (DTR) of the MODEM Control Register to a high level. A Master Reset operation sets this signal to its inactive (high) state.

Request to Send ($\overline{RTS}$), Pin 32: When low, this informs the MODEM or data set that the UART is ready to exchange data. The $\overline{RTS}$ output signal can be set to an active low by programming bit 1 (RTS) of the MODEM Control Register. A Master Reset operation sets this signal to its inactive (high) state.

Output 1 ($\overline{OUT\ 1}$), Pin 34: This user-designated output can be set to an active low by programming bit 2 (OUT 1) of the MODEM Control Register to a high level. A Master Reset operation sets this signal to its inactive (high) state. In the XMOS parts this will achieve TTL levels.

Output 2 ($\overline{OUT\ 2}$), Pin 31: This user-designated output can be set to an active low by programming bit 3 (OUT 2) of the MODEM Control Register to a high level. A Master Reset operation sets this signal to its inactive (high) state. In the XMOS parts this will achieve TTL levels.

Chip Select Out (CSOUT), Pin 24: When high, it indicates that the chip has been selected by active, CS0, CS1, and $\overline{CS2}$ inputs. No data transfer can be initiated until the CSOUT signal is a logic 1. CSOUT goes low when the UART is deselected.

Driver Disable (DDIS), Pin 23: This goes low whenever the CPU is reading data from the UART. It can disable or control the direction of a data bus transceiver between the CPU and the UART (see Typical Interface for a High Capacity Data Bus).

Baud Out ($\overline{BAUDOUT}$), Pin 15: This is the 16 × clock signal from the transmitter section of the UART. The clock rate is equal to the main reference oscillator frequency divided by the specified divisor in the Baud Generator Divisor Latches. The $\overline{BAUDOUT}$ may also be used for the receiver section by tying this output to the RCLK input of the chip.

4

6.0 Pin Descriptions (Continued)

Interrupt (INTR), Pin 30: This goes high whenever any one of the following interrupt types has an active high condition and is enabled via the IER: Receiver Line Status; Received Data Available; Transmitter Holding Register Empty; and MODEM Status. The INTR signal is reset low upon the appropriate interrupt service or a Master Reset operation.

Serial Output (SOUT), Pin 11: This is the composite serial data output to the communications link (peripheral, MODEM or data set). The SOUT signal is set to the Marking (logic 1) state upon a Master Reset operation or when the transmitter is idle.

6.3 INPUT/OUTPUT SIGNALS

Data (D_7-D_0) Bus, Pins 1-8: This bus is comprised of eight TRI-STATE input/output lines. The bus provides bidirectional communications between the UART and the CPU. Data, control words, and status information are transferred via the D_7-D_0 Data Bus.

External Clock Input/Output (X_{IN}, X_{OUT}) Pins 16 and 17: These two pins connect the main timing reference (crystal or signal clock) to the UART. When a crystal oscillator or a clock signal is provided, it drives the UART via XIN (see typical oscillator network illustration).

7.0 Connection Diagrams

Dual-In-Line Package
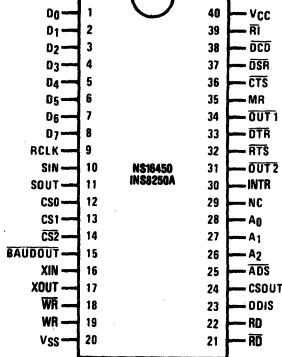
TL/C/9329-11

Top View

Order Number INS8250N, INS8250N-B or INS8250N/A+
See NS Package Number N40A

PCC Package
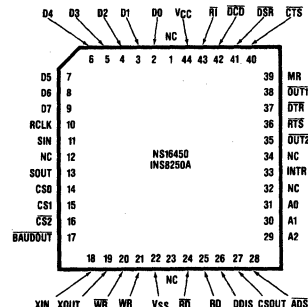
TL/C/9329-18

Top View

Order Number INS8250V-B
See NS Package Number V44A

TABLE I. UART Reset Functions

Register/Signal	Reset Control	Reset State
Interrupt Enable Register	Master Reset	**0000** 0000 (Note 1)
Interrupt Identification Register	Master Reset	**00000** 001
Line Control Register	Master Reset	0000 0000
MODEM Control Register	Master Reset	**0000** 0000
Line Status Register	Master Reset	**0**110 0000
MODEM Status Register	Master Reset	XXXX 0000 (Note 2)
SOUT	Master Reset	High
INTR (RCVR Errs)	Read LSR/MR	Low
INTR (RCVR Data Ready)	Read RBR/MR	Low
INTR (THRE)	Read IIR/Write THR/MR	Low
INTR (Modem Status Changes)	Read MSR/MR	Low
$\overline{OUT\,2}$	Master Reset	High
$\overline{RTS}$	Master Reset	High
$\overline{DTR}$	Master Reset	High
$\overline{OUT\,1}$	Master Reset	High

Note 1: Underlined bits are permanently low.

Note 2: Bits 7-4 are driven by the input signals.

8.0 Registers

The system programmer may access any of the UART registers summarized in Table II via the CPU. These registers control UART operations including transmission and reception of data. Each register bit in Table II has its name and reset state shown.

8.1 LINE CONTROL REGISTER

The system programmer specifies the format of the asynchronous data communications exchange and sets the Divisor Latch Access bit via the Line Control Register (LCR). The programmer can also read the contents of the Line Control Register. The read capability simplifies system programming and eliminates the need for separate storage in system memory of the line characteristics. Table II shows the contents of the LCR. Details on each bit follow:

Bits 0 and 1: These two bits specify the number of bits in each transmitted or received serial character. The encoding of bits 0 and 1 is as follows:

Bit 1	Bit 0	Character Length
0	0	5 Bits
0	1	6 Bits
1	0	7 Bits
1	1	8 Bits

Bit 2: This bit specifies the number of Stop bits transmitted and received in each serial character. If bit 2 is a logic 0, one Stop bit is generated or checked in the serial data. If bit 2 is a logic 1 when a 5-bit word length is selected via bits 0

TABLE II. Summary of Registers

Bit No.	0 DLAB=0 Receiver Buffer Register (Read Only) RBR	0 DLAB=0 Transmitter Holding Register (Write Only) THR	1 DLAB=0 Interrupt Enable Register IER	2 Interrupt Ident. Register (Read Only) IIR	3 Line Control Register LCR	4 MODEM Control Register MCR	5 Line Status Register LSR	6 MODEM Status Register MSR	0 DLAB=1 Divisor Latch (LS) DLL	1 DLAB=1 Division Latch (MS) DLM
0	Data Bit 0 (Note 1)	Data Bit 0	Received Data Available	"0" if Interrupt Pending	Word Length Select Bit 0 (WLS0)	Data Terminal Ready (DTR)	Data Ready (DR)	Delta 0 Clear to Send (DCTS)	Bit 0	Bit 8
1	Data Bit 1	Data Bit 1	Transmitter Holding Register Empty	Interrupt ID Bit (0)	Word Length Select Bit 1 (WLS1)	Request to Send (RTS)	Overrun Error (OE)	Delta Data Set Ready (DDSR)	Bit 1	Bit 9
2	Data Bit 2	Data Bit 2	Receiver Line Status	Interrupt ID Bit (1)	Number of Stop Bits (STB)	Out 1	Parity Error (PE)	Trailing Edge Ring Indicator (TERI)	Bit 2	Bit 10
3	Data Bit 3	Data Bit 3	MODEM Status	0	Parity Enable (PEN)	Out 2	Framing Error (FE)	Delta Data Carrier Detect (DDCD)	Bit 3	Bit 11
4	Data Bit 4	Data Bit 4	0	0	Even Parity Select (EPS)	Loop	Break Interrupt (BI)	Clear to Send (CTS)	Bit 4	Bit 12
5	Data Bit 5	Data Bit 5	0	0	Stick Parity	0	Transmitter Holding Register (THRE)	Data Set Ready (DSR)	Bit 5	Bit 13
6	Data Bit 6	Data Bit 6	0	0	Set Break	0	Transmitter Shift Register Empty (TSRE)	Ring Indicator (RI)	Bit 6	Bit 14
7	Data Bit 7	Data Bit 7	0	0	Divisor Latch Access Bit (DLAB)	0	0	Data Carrier Detect (DCD)	Bit 7	Bit 15

Note 1: Bit 0 is the least significant bit. It is the first bit serially transmitted or received.

4

8.0 Registers (Continued)

and 1, one and a half Stop bits are generated. If bit 2 is a logic 1 when either a 6-, 7-, or 8-bit word length is selected, two Stop bits are generated. The Receiver checks the first Stop bit only, regardless of the number of Stop bits selected.

Bit 3: This bit is the Parity Enable bit. When bit 3 is a logic 1, a Parity bit is generated (transmit data) or checked (receive data) between the last data word bit and Stop bit of the serial data. (The Parity bit is used to produce an even or odd number of 1s when the data word bits and the Parity bit are summed.)

Bit 4: This bit is the Even Parity Select bit. When bit 3 is a logic 1 and bit 4 is a logic 0, an odd number of logic 1s is transmitted or checked in the data word bits and Parity bit. When bit 3 is a logic 1 and bit 4 is a logic 1, an even number of logic 1s is transmitted or checked.

Bit 5: This bit is the Stick Parity bit. When bits 3, 4 and 5 are logic 1 the Parity bit is transmitted and checked as a logic 0. If bits 3 and 5 are 1 and bit 4 is a logic 0 then the Parity bit is transmitted and checked as a logic 1. If bit 5 is a logic 0 Stick Parity is disabled.

Bit 6: This bit is the Break Control bit. It causes a break condition to be transmitted by the UART. When it is set to a logic 1, the serial output (SOUT) is forced to the Spacing (logic 0) state. The break is disabled by clearing bit 6 to a logic 0. The Break Control bit acts only on SOUT and has no effect on the transmitter logic.

Note: This feature enables the CPU to alert a terminal in a computer communications system. If the following sequence is used no erroneous or extraneous characters will be transmitted because of the break.

1. Load an all 0s, pad character, in response to THRE.

2. Set break after the next THRE.

3. Wait for the transmitter to be idle, (TSRE = 1), and clear break when normal transmission has to be restored.

During the break, the Transmitter can be used as a character timer to accurately establish the break duration.

Bit 7: This bit is the Divisor Latch Access Bit (DLAB). It must be set high (logic 1) to access the Divisor Latches of the Baud Generator during a Read or Write operation. It must be set low (logic 0) to access the Receiver Buffer, the Transmitter Holding Register, or the Interrupt Enable Register.

TABLE III. Baud Rates Using 1.8432 MHz Crystal

Desired Baud Rate	Decimal Divisor Used to Generate 16 x Clock	Percent Error Difference Between Desired and Actual
50	2304	—
75	1536	—
110	1047	0.026
134.5	857	0.058
150	768	—
300	384	—
600	192	—
1200	96	—
1800	64	—
2000	58	0.69
2400	48	—
3600	32	—
4800	24	—
7200	16	—
9600	12	—
19200	6	—
38400	3	—
56000	2	2.86

8.2 Typical Clock Circuits
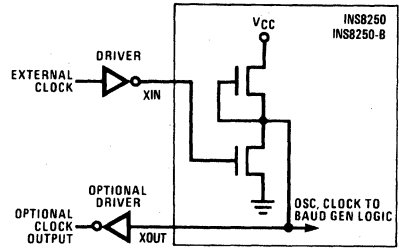
TL/C/9329–12
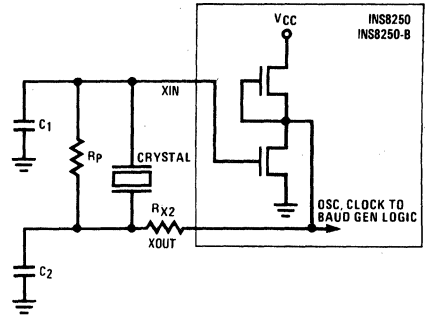
TL/C/9329–13

Typical Oscillator Networks (Note)

Crystal	R_P	R_{X2}	C_1	C_2
1.8–3.1 MHz	1 MΩ	1.5k	10–30 pF	40–60 pF

Note: These R and C values are approximate and may vary 2x depending on the crystal characteristics. All crystal circuits should be designed specifically for the system.

TABLE IV. Baud Rates Using 3.072 MHz Crystal

Desired Baud Rate	Decimal Divisor Used to Generate 16 x Clock	Percent Error Difference Between Desired and Actual
50	3840	—
75	2560	—
110	1745	0.026
134.5	1428	0.034
150	1280	—
300	640	—
600	320	—
1200	160	—
1800	107	0.312
2000	96	—
2400	80	—
3600	53	0.628
4800	40	—
7200	27	1.23
9600	20	—
19200	10	—
38400	5	—

8.0 Registers (Continued)

8.3 PROGRAMMABLE BAUD GENERATOR

The UART contains a programmable Baud Generator that is capable of taking any clock input from DC to 3.1 MHz and dividing it by any divisor from 1 to $2^{16}-1$. The output frequency of the Baud Generator is 16 $\times$ the Baud [divisor # = (frequency input) $\div$ (baud rate $\times$ 16)]. Two 8-bit latches store the divisor in a 16-bit binary format. These Divisor Latches must be loaded during initialization in order to ensure proper operation of the Baud Generator. Upon loading either of the Divisor Latches, a 16-bit Baud counter is immediately loaded.

Tables III and IV provide decimal divisors to use with crystal frequencies of 1.8432 MHz and 3.072 MHz, respectively, for common baud rates. For baud rates of 38400 and below, the error obtained is minimal. The accuracy of the desired baud rate is dependent on the crystal frequency chosen. Using a division of 0 is **not** recommended.

Note: The maximum operating frequency of the Baud Generator is 3.1 MHz. However, when using divisors of 3 and below, the maximum frequency is equal to the divisor in MHz. For example, if the divisor is 1, then the maximum frequency is 1 MHz. In no case should the data rate be greater than 56k Baud.

8.4 LINE STATUS REGISTER

This 8-bit register provides status information to the CPU concerning the data transfer. Table II shows the contents of the Line Status Register. Details on each bit follow:

Bit 0: This bit is the receiver Data Ready (DR) indicator. Bit 0 is set to a logic 1 whenever a complete incoming character has been received and transferred into the Receiver Buffer Register. Bit 0 is reset to a logic 0 by reading the data in the Receiver Buffer Register.

Bit 1: This bit is the Overrun Error (OE) indicator. Bit 1 indicates that data in the Receiver Buffer Register was not read by the CPU before the next character was transferred into the Receiver Buffer Register, thereby destroying the previous character. The OE indicator is set to a logic 1 upon detection of an overrun condition and reset whenever the CPU reads the contents of the Line Status Register.

Bit 2: This bit is the Parity Error (PE) indicator. Bit 2 indicates that the received data character does not have the

correct even or odd parity, as selected by the even-parity-select bit. The PE bit is set to a logic 1 upon detection of a parity error and is reset to a logic 0 whenever the CPU reads the contents of the Line Status Register.

Bit 3: This bit is the Framing Error (FE) indicator. Bit 3 indicates that the received character did not have a valid Stop bit. Bit 3 is set to a logic 1 whenever the Stop bit following the last data bit or parity bit is a logic 0 (Spacing level). The FE indicator is reset whenever the CPU reads the contents of the Line Status Register. The UART will try to resynchronize after a framing error. To do this it assumes that the framing error was due to the next start bit, so it samples this "start" bit twice and then takes in the "data".

Bit 4: This bit is the Break Interrupt (BI) indicator. Bit 4 is set to a logic 1 whenever the received data input is held in the Spacing (logic 0) state for longer than a full word transmission time (that is, the total time of Start bit + data bits + Parity + Stop bits). The BI indicator is reset whenever the CPU reads the contents of the Line Status Register. Restarting after a break is received, requires the SIN pin to be logical 1 for at least ½ bit time.

Note: Bits 1 through 4 are the error conditions that produce a Receiver Line Status interrupt whenever any of the corresponding conditions are detected and the interrupt is enabled.

Bit 5: This bit is the Transmitter Holding Register Empty (THRE) indicator. Bit 5 indicates that the UART is ready to accept a new character for transmission. In addition, this bit causes the UART to issue an interrupt to the CPU when the Transmit Holding Register Empty Interrupt enable is set high. The THRE bit is set to a logic 1 when a character is transferred from the Transmitter Holding Register into the Transmitter Shift Register. The bit is reset to logic 0 whenever the CPU loads the Transmitter Holding Register.

Bit 6: This bit is the Transmitter Shift Register Empty (TSRE) indicator. Bit 6 is set to a logic 1 whenever the Transmitter Shift Register (TSR) is empty. It is reset to a logic 0 whenever a data character is transferred to the TSR.

Bit 7: This bit is permanently set to logic 0.

Note: The Line Status Register is intended for read operations only. Writing to this register is not recommended as this operation is only used for factory testing.

TABLE V. Interrupt Control Functions

Interrupt Identification Register				Interrupt Set and Reset Functions		
Bit 2	Bit 1	Bit 0	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Control
0	0	1	—	None	None	—
1	1	0	Highest	Receiver Line Status	Overrun Error or Parity Error or Framing Error or Break Interrupt	Reading the Line Status Register
1	0	0	Second	Received Data Available	Receiver Data Available	Reading the Receiver Buffer Register
0	1	0	Third	Transmitter Holding Register Empty	Transmitter Holding Register Empty	Reading the IIR Register (if source of interrupt) or Writing into the Transmitter Holding Register
0	0	0	Fourth	MODEM Status	Clear to Send or Data Set Ready or Ring Indicator or Data Carrier Detect	Reading the MODEM Status Register

4

8.0 Registers (Continued)

8.5 INTERRUPT IDENTIFICATION REGISTER

In order to provide minimum software overhead during data character transfers, the UART prioritizes interrupts into four levels and records these in the Interrupt Identification Register. The four levels of interrupt conditions in order of priority are Receiver Line Status; Received Data Ready; Transmitter Holding Register Empty; and MODEM Status.

When the CPU accesses the IIR, the UART freezes all interrupts and indicates the highest priority pending interrupt to the CPU. While this CPU access is occurring, the UART records new interrupts, but does not change its current indication until the access is complete. Table II shows the contents of the IIR. Details on each bit follow:

Bit 0: This bit can be used in an interrupt environment to indicate whether an interrupt condition is pending. When bit 0 is a logic 0, an interrupt is pending and the IIR contents may be used as a pointer to the appropriate interrupt service routine. When bit 0 is a logic 1, no interrupt is pending.

Bits 1 and 2: These two bits of the IIR are used to identify the highest priority interrupt pending as indicated in Table V.

Bits 3 through 7: These five bits of the IIR are always logic 0.

8.6 INTERRUPT ENABLE REGISTER

This register enables the four types of UART interrupts. Each interrupt can individually activate the interrupt (INTR) output signal. It is possible to totally disable the interrupt system by resetting bits 0 through 3 of the Interrupt Enable Register (IER). Similarly, setting bits of this register to a logic 1, enables the selected interrupt(s). Disabling an interrupt prevents it from being indicated as active in the IIR and from activating the INTR output signal. All other system functions operate in their normal manner, including the setting of the Line Status and MODEM Status Registers. Table II shows the contents of the IER. Details on each bit follow.

Bit 0: This bit enables the Received Data Available Interrupt when set to logic 1.

Bit 1: This bit enables the Transmitter Holding Register Empty Interrupt when set to logic 1.

Bit 2: This bit enables the Receiver Line Status Interrupt when set to logic 1.

Bit 3: This bit enables the MODEM Status Interrupt when set to logic 1.

Bits 4 through 7: These four bits are always logic 0.

8.7 MODEM CONTROL REGISTER

This register controls the interface with the MODEM or data set (or a peripheral device emulating a MODEM). The contents of the MODEM Control Register (MCR) are indicated in Table II and are described below. Table II shows the contents of the MCR. Details on each bit follow.

Bit 0: This bit controls the Data Terminal Ready ($\overline{DTR}$) output. When bit 0 is set to a logic 1, the $\overline{DTR}$ output is forced to a logic 0. When bit 0 is reset to a logic 0, the $\overline{DTR}$ output is forced to a logic 1.

Note: The $\overline{DTR}$ output of the UART may be applied to an EIA inverting line driver (such as the DS1488) to obtain the proper polarity input at the succeeding MODEM or data set.

Bit 1: This bit controls the Request to Send ($\overline{RTS}$) output. Bit 1 affects the $\overline{RTS}$ output in a manner identical to that described above for bit 0.

Bit 2: This bit controls the Output 1 ($\overline{OUT\ 1}$) signal, which is an auxiliary user-designated output. Bit 2 affects the $\overline{OUT\ 1}$ output in a manner identical to that described above for bit 0.

Bit 3: This bit controls the Output 2 ($\overline{OUT\ 2}$) signal, which is an auxiliary user-designated output. Bit 3 affects the $\overline{OUT\ 2}$ output in a manner identical to that described above for bit 0.

Bit 4: This bit provides a local loopback feature for diagnostic testing of the UART. When bit 4 is set to logic 1, the following occur: the transmitter Serial Output (SOUT) is set to the Marking (logic 1) state; the receiver Serial Input (SIN) is disconnected; the output of the Transmitter Shift Register is "looped back" into the Receiver Shift Register input; the four MODEM Control inputs ($\overline{DSR}$, $\overline{CTS}$, $\overline{RI}$, and $\overline{DCD}$) are disconnected. In loopback mode the modem control outputs $\overline{RTS}$, $\overline{DTR}$, $\overline{OUT\ 1}$, and $\overline{OUT\ 2}$ remain connected to the associated Modem Control Register bits. In the diagnostic mode, data that is transmitted is immediately received. This feature allows the processor to verify the transmit-and received-data paths of the UART.

In the diagnostic mode, the receiver and transmitter interrupts are fully operational. The MODEM Control Interrupts are also operational, but the interrupts' sources are now the lower four bits of the MODEM Control Register instead of the four MODEM Control inputs. The interrupts are still controlled by the Interrupt Enable Register.

Bits 5 through 7: These bits are permanently set to logic 0.

8.8 MODEM STATUS REGISTER

This register provides the current state of the control lines from the MODEM (or peripheral device) to the CPU. In addition to this current-state information, four bits of the MODEM Status Register provide change information. These bits are set to a logic 1 whenever a control input from the MODEM changes state. They are reset to logic 0 whenever the CPU reads the MODEM Status Register.

Table II shows the contents of the MSR. Details on each bit follow:

Bit 0: This bit is the Delta Clear to Send (DCTS) indicator. Bit 0 indicates that the $\overline{CTS}$ input to the chip has changed state since the last time it was read by the CPU.

Bit 1: This bit is the Delta Data Set Ready (DDSR) indicator. Bit 1 indicates that the $\overline{DSR}$ input to the chip has changed state since the last time it was read by the CPU.

Bit 2: This bit is the Trailing Edge of Ring Indicator (TERI) detector. Bit 2 indicates that the $\overline{RI}$ input to the chip has changed from a low to a high state.

Bit 3: This bit is the Delta Data Carrier Detect (DDCD) indicator. Bit 3 indicates that the $\overline{DCD}$ input to the chip has changed state.

Note: Whenever bit 0, 1, 2, or 3 is set to logic 1, a MODEM Status Interrupt is generated.

Bit 4: This bit is the complement of the Clear to Send ($\overline{CTS}$) input. If bit 4 (loop) of the MCR is set to a 1, this bit is equivalent to RTS in the MCR.

Bit 5: This bit is the complement of the Data Set Ready ($\overline{DSR}$) input. If bit 4 of the MCR is set to a 1, this bit is equivalent to DTR in the MCR.

Bit 6: This bit is the complement of the Ring Indicator ($\overline{RI}$) input. If bit 4 of the MCR is set to a 1, this bit is equivalent to OUT 1 in the MCR.

Bit 7: This bit is the complement of the Data Carrier Detect ($\overline{DCD}$) input. If bit 4 of the MCR is set to a 1, this bit is equivalent to OUT 2 in the MCR.

Basic Connections of an INS8250 to an 8088 CPU
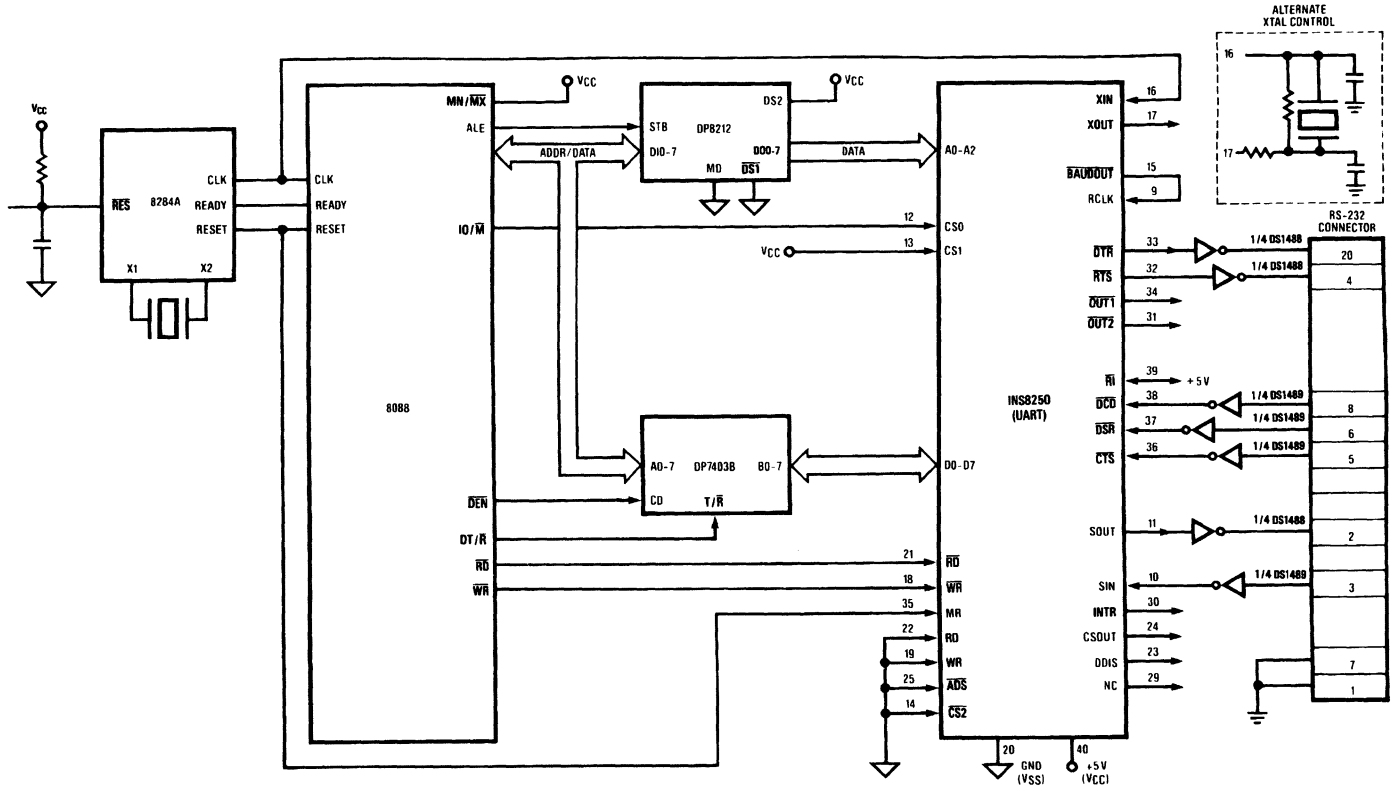
TL/C/9329-15

9.0 Typical Applications (Continued)

**Typical Interface for a
High-Capacity Data Bus**

TL/C/9329–16

**Typical Supply Current vs
Temperature, Normalized**
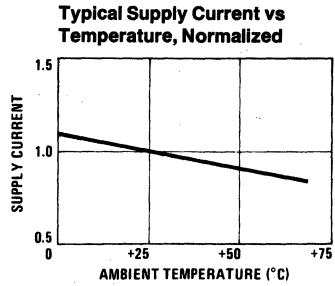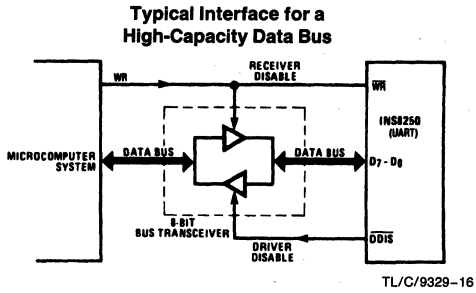
TL/C/9329–17

10.0 Ordering Information

INS8250XX

/A⁺ = A⁺ RELIABILITY SCREENING

N = PLASTIC PACKAGE

TL/C/9329–19

INS8250XX

–B

N = PLASTIC PACKAGE
V = PLASTIC LEADED CHIP CARRIER PACKAGE

TL/C/9329–20

![National Semiconductor logo]

NS16450, INS8250A, NS16C450, INS82C50A
Universal Asynchronous Receiver/Transmitter

General Description

Each of these parts function as a serial data input/output interface in a microcomputer system. The system software determines the functional configuration of the UART via a TRI-STATE® 8-bit bidirectional data bus.

The UART performs serial-to-parallel conversion on data characters received from a peripheral device or a MODEM, and parallel-to-serial conversion on data characters received from the CPU. The CPU can read the complete status of the UART at any time during the functional operation. Status information reported includes the type and condition of the transfer operations being performed by the UART, as well as any error conditions (parity, overrun, framing, or break interrupt).

The UART includes a programmable baud rate generator that is capable of dividing the timing reference clock input by divisors of 1 to $(2^{16}-1)$, and producing a 16 × clock for driving the internal transmitter logic. Provisions are also included to use this 16 × clock to drive the receiver logic. The UART includes a complete MODEM-control capability and a processor-interrupt system. Interrupts can be programmed to the user's requirements, minimizing the computing required to handle the communications link.

The NS16450 is an improved specification version of the INS8250A Universal Asynchronous Receiver/Transmitter (UART). Functionally, the NS16450 is equivalent to the INS8250A. The UART is fabricated using National Semiconductor's advanced scaled N-channel silicon-gate MOS process, XMOS.

The NS16C450 and INS82C50A are functionally equivalent to their XMOS counterparts, except that they are CMOS parts.

Features

■ Easily interfaces to most popular microprocessors.
■ Adds or deletes standard asynchronous communication bits (start, stop, and parity) to or from serial data stream.
■ Holding and shift registers eliminate the need for precise synchronization between the CPU and the serial data.
■ Independently controlled transmit, receive, line status, and data set interrupts.
■ Programmable baud generator allows division of any input clock by 1 to $(2^{16} - 1)$ and generates the internal 16 × clock.
■ Independent receiver clock input.
■ MODEM control functions (CTS, RTS, DSR, DTR, RI, and DCD).
■ Fully programmable serial-interface characteristics:
   — 5-, 6-, 7-, or 8-bit characters
   — Even, odd, or no-parity bit generation and detection
   — 1-, 1½-, or 2-stop bit generation
   — Baud generation (DC to 56k baud).
■ False start bit detection.
■ Complete status reporting capabilities.
■ TRI-STATE TTL drive capabilities for bidirectional data bus and control bus.
■ Line break generation and detection.
■ Internal diagnostic capabilities:
   — Loopback controls for communications link fault isolation
   — Break, parity, overrun, framing error simulation.
■ Fully prioritized interrupt system controls.

Connection Diagram

TL/C/8401–1

4

Table of Contents

1.0 Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Temperature Under Bias	0°C to +70°C
Storage Temperature	−65°C to +150°C

All Input or Output Voltages with Respect to V_{SS} −0.5V to +7.0V

Power Dissipation   700 mW

Note: *Maximum ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended and should be limited to those conditions specified under DC electrical characteristics.*

2.0 DC Electrical Characteristics

T_A = 0°C to +70°C, V_{CC} = +5V ±5%, V_{SS} = 0V, unless otherwise specified.

Symbol	Parameter	Conditions	NS16450 NS16C450 (Note 1)		INS8250A INS82C50A (Note 1)		Units
			Min	Max	Min	Max	
V_{ILX}	Clock Input Low Voltage		−0.5	0.8	−0.5	0.8	V
V_{IHX}	Clock Input High Voltage		2.0	V_{CC}	2.0	V_{CC}	V
V_{IL}	Input Low Voltage		−0.5	0.8	−0.5	0.8	V
V_{IH}	Input High Voltage		2.0	V_{CC}	2.0	V_{CC}	V
V_{OL}	Output Low Voltage	I_{OL} = 1.6 mA on all (Note 2)		0.4		0.4	V
V_{OH}	Output High Voltage	I_{OH} = −1.0 mA (Note 2)	2.4		2.4		V
$I_{CC}(AV)$	Avg. Power Supply Current (V_{CC}) **XMOS Parts Only**	V_{CC} = 5.25V, T_A = 25°C No Loads on output SIN, DSR, DCD, CTS, RI = 2.4V All other inputs = 0.4V		120		95	mA
$I_{CC}(AV)$	Avg. Power Supply Current (V_{CC}) **CMOS Parts Only**	V_{CC} = 5.25V, T_A = 25°C No Loads on output SIN, DSR, DCD, CTS, RI = 2.4V All other inputs = 0.4V Baud Rate Generator is 4 MHz Baud Rate is 50k		10		10	mA
I_{IL}	Input Leakage	V_{CC} = 5.25V, V_{SS} = 0V All other pins floating. V_{IN} = 0V, 5.25V		±10		±10	μA
I_{CL}	Clock Leakage			±10		±10	μA
I_{OZ}	TRI-STATE Leakage	V_{CC} = 5.25V, V_{SS} = 0V V_{OUT} = 0V, 5.25V 1) Chip deselected 2) WRITE mode, chip selected		±20		±20	μA
V_{ILMR}	MR Schmitt V_{IL}			0.8		0.8	V
V_{IHMR}	MR Schmitt V_{IH}		2.0		2.0		V

Capacitance T_A = 25°C, V_{CC} = V_{SS} = 0V

Symbol	Parameter	Conditions	Min	Typ	Max	Units
C_{XIN}	Clock Input Capacitance			15	20	pF
C_{XOUT}	Clock Output Capacitance	f_c = 1 MHz		20	30	pF
C_{IN}	Input Capacitance	Unmeasured pins		6	10	pF
C_{OUT}	Output Capacitance	returned to V_{SS}		10	20	pF

Note 1: Inputs on the CMOS parts are TTL compatible; outputs on the CMOS parts drive to GND and V_{CC}.

Note 2: Does not apply to XOUT.

3.0 AC Electrical Characteristics $T_A = 0°C$ to $+70°C$, $V_{CC} = +5V \pm 5\%$

Symbol	Parameter	Conditions	NS16450 NS16C450		INS8250A INS82C50A		Units
			Min	Max	Min	Max	
t_{ADS}	Address Strobe Width		60		90		ns
t_{AH}	Address Hold Time		0		0		ns
t_{AR}	RD, $\overline{RD}$ Delay from Address	(Note 1)	60		80		ns
t_{AS}	Address Setup Time		60		90		ns
t_{AW}	WR, $\overline{WR}$ Delay from Address	(Note 1)	60		80		ns
t_{CH}	Chip Select Hold Time		0		0		ns
t_{CS}	Chip Select Setup Time		60		90		ns
t_{CSC}	Chip Select Output Delay from Select	@100 pF loading (Note 1)		100		125	ns
t_{CSR}	RD, $\overline{RD}$ Delay from Chip Select	(Note 1)	50		80		ns
t_{CSW}	WR, $\overline{WR}$ Delay from Select	(Note 1)	50		80		ns
t_{DH}	Data Hold Time		40		60		ns
t_{DS}	Data Setup Time		40		90		ns
t_{HZ}	RD, $\overline{RD}$ to Floating Data Delay	@100 pF loading (Note 3)	0	100	0	100	ns
t_{MR}	Master Reset Pulse Width		5		10		μs
t_{RA}	Address Hold Time from RD, $\overline{RD}$	(Note 1)	20		20		ns
t_{RC}	Read Cycle Delay		175		500		ns
t_{RCS}	Chip Select Hold Time from RD, $\overline{RD}$	(Note 1)	20		20		ns
t_{RD}	RD, $\overline{RD}$ Strobe Width		125		175		ns
t_{RDD}	RD, $\overline{RD}$ to Driver Disable Delay	@100 pF loading (Note 3)		60		75	ns
t_{RVD}	Delay from RD, $\overline{RD}$ to Data	@100 pF loading		125		175	ns
t_{WA}	Address Hold Time from WR, $\overline{WR}$	(Note 1)	20		20		ns
t_{WC}	Write Cycle Delay		200		500		ns
t_{WCS}	Chip Select Hold Time from WR, $\overline{WR}$	(Note 1)	20		20		ns
t_{WR}	WR, $\overline{WR}$ Strobe Width		100		175		ns
t_{XH}	Duration of Clock High Pulse	External Clock (3.1 MHz Max.)	140		140		ns
t_{XL}	Duration of Clock Low Pulse	External Clock (3.1 MHz Max.)	140		140		ns
RC	Read Cycle = $t_{AR} + t_{RD} + t_{RC}$		360		755		ns
WC	Write Cycle = $t_{AW} + t_{WR} + t_{WC}$		360		755		ns
Baud Generator							
N	Baud Divisor		1	$2^{16}-1$	1	$2^{16}-1$	
t_{BHD}	Baud Output Positive Edge Delay	100 pF Load		175		250	ns
t_{BLD}	Baud Output Negative Edge Delay	100 pF Load		175		250	ns
t_{HW}	Baud Output Up Time	$f_X = 3$ MHz, $\div 3$, 100 pF Load	250		250		ns
t_{LW}	Baud Output Down Time	$f_X = 2$ MHz, $\div 2$, 100 pF Load	425		425		ns
Receiver							
t_{RINT}	Delay from RD, $\overline{RD}$ (RD RBR or RD LSR) to Reset Interrupt	100 pF Load		1		1	μs
t_{SCD}	Delay from RCLK to Sample Time			2		2	μs
t_{SINT}	Delay from Stop to Set Interrupt			1		1	RCLK Cycles (Note 2)

Note 1: Applicable only when $\overline{ADS}$ is tied low.

Note 2: RCLK is equal to t_{XH} and t_{XL}.

Note 3: Charge and discharge time is determined by V_{OL}, V_{OH} and the external loading.

3.0 AC Electrical Characteristics $T_A = 0°C$ to $+70°C$, $V_{CC} = +5V \pm 5\%$ (Continued)

Symbol	Parameter	Conditions	NS16450 NS16C450		INS8250A INS82C50A		Units
			Min	Max	Min	Max	
Transmitter							
t_{HR}	Delay from WR, $\overline{WR}$ (WR THR) to Reset Interrupt	100 pF Load		175		1000	ns
t_{IR}	Delay from RD, $\overline{RD}$ (RD IIR) to Reset Interrupt (THRE)	100 pF Load		250		1000	ns
t_{IRS}	Delay from Initial INTR Reset to Transmit Start		24	40	24	40	BAUDOUT Cycles
t_{SI}	Delay from Initial Write to Interrupt	(Note 1)	16	24	16	24	BAUDOUT Cycles
t_{STI}	Delay from Stop to Interrupt (THRE)		8	8	8	8	BAUDOUT Cycles
Modem Control							
t_{MDO}	Delay from WR, $\overline{WR}$ (WR MCR) to Output	100 pF Load		200		1000	ns
t_{RIM}	Delay to Reset Interrupt from RD, $\overline{RD}$ (RD MSR)	100 pF Load		250		1000	ns
t_{SIM}	Delay to Set Interrupt from MODEM Input	100 pF Load		250		1000	ns

Note 1: For both the NS16C450 and INS82C50A, t_{SI} is a minimum of 16 and a maximum of 48 BAUDOUT cycles.

4.0 Timing Waveforms (All timings are referenced to valid 0 and valid 1)

External Clock Input (3.1 MHz Max.)

TL/C/8401-2

AC Test Points
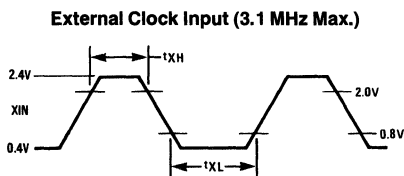
TL/C/8401-3

Note 3: The 2.4V and 0.4V levels are the voltages that the inputs are driven to during AC testing.

Note 4: The 2.0V and 0.8V levels are the voltages at which the timing tests are made.

$\overline{BAUDOUT}$ Timing

$t_{HW} = (N - 2)$ XIN CYCLES

$t_{LW} = 2$ XOUT CYCLES
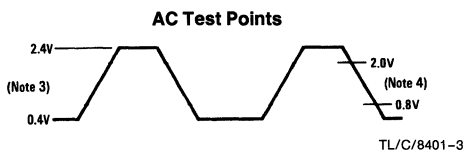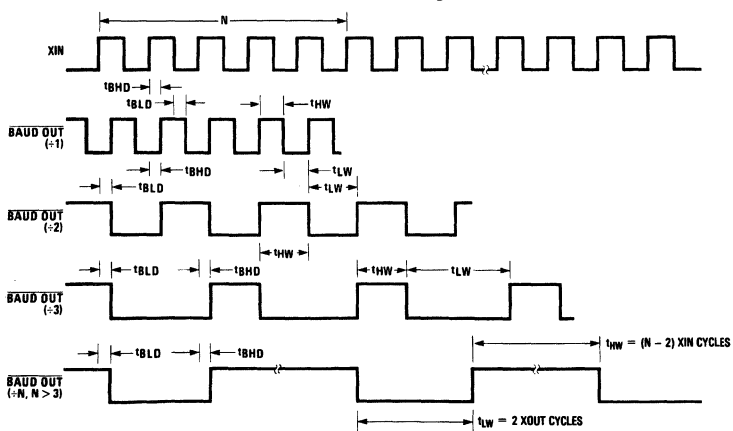
TL/C/8401-4

4.0 Timing Waveforms (Continued)

Write Cycle

t_{ADS}

$\overline{ADS}$

t_{AS} t_{AH}

A_2, A_1, A_0 VALID

t_{CS} t_{CH} t_{WA}^*

$\overline{CS_2}, CS_1, CS_0$ VALID

t_{CSC}^* t_{WCS}^*

CSOUT

t_{CSW}^* WC

t_{AW}^*

t_{WR} t_{WC}

$\overline{WR}, WR$ ACTIVE ACTIVE

OR

$\overline{RD}, RD$ ACTIVE

t_{DS} t_{DH}

DATA $D_0 - D_7$ VALID DATA
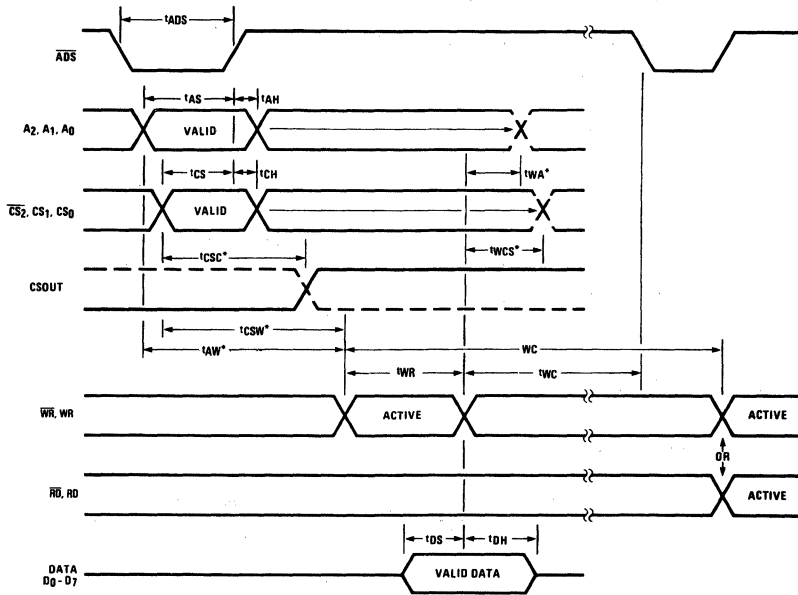
*Applicable Only When $\overline{ADS}$ is Tied Low.

TL/C/8401–5

Read Cycle

t_{ADS}

$\overline{ADS}$

t_{AS} t_{AH}

A_2, A_1, A_0 VALID

t_{CS} t_{CH} t_{RA}^*

$\overline{CS_2}, CS_1, CS_0$ VALID

t_{CSC}^* t_{RCS}^*

CSOUT

t_{CSR}^* RC

t_{AR}^*

t_{RD} t_{RC}

$\overline{RD}, RD$ ACTIVE ACTIVE

OR

$\overline{WR}, WR$ ACTIVE

t_{RDD} t_{RDD}

DDIS

t_{RVD} t_{HZ}

DATA $D_0 - D_7$ VALID DATA
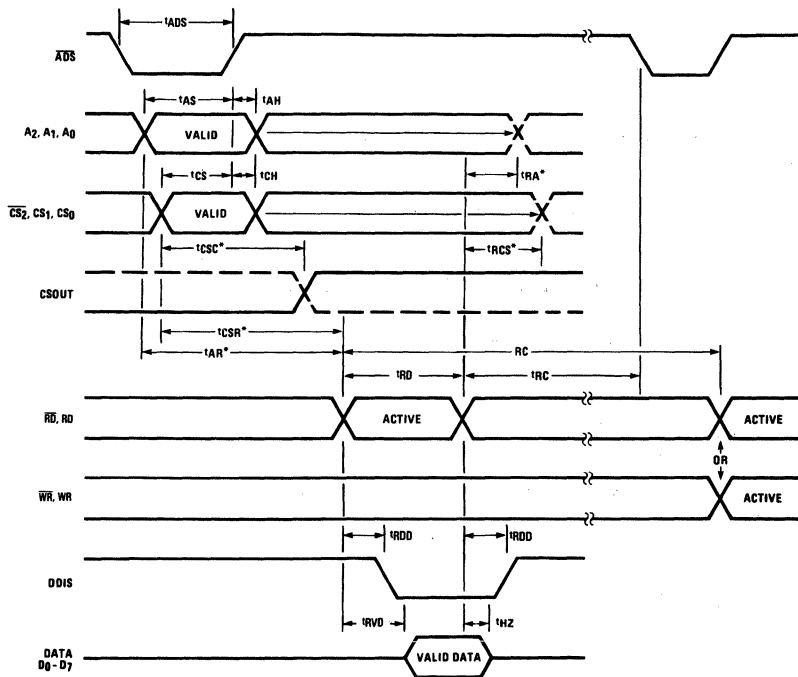
*Applicable Only When $\overline{ADS}$ is Tied Low.

TL/C/8401–6

4.0 Timing Waveforms (Continued)

Receiver Timing
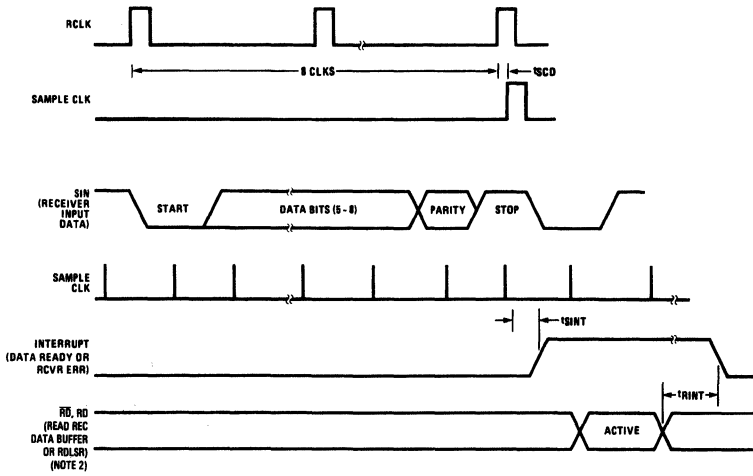
TL/C/8401-7

Transmitter Timing
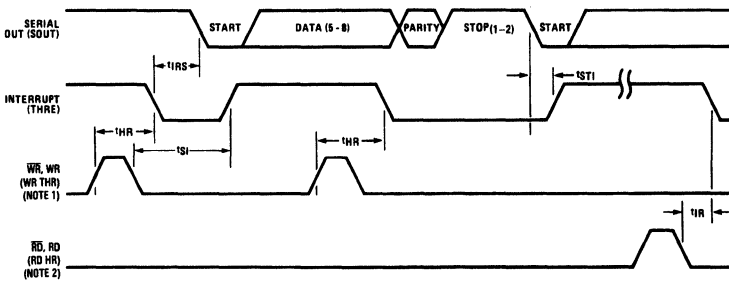
TL/C/8401-8

MODEM Controls Timing
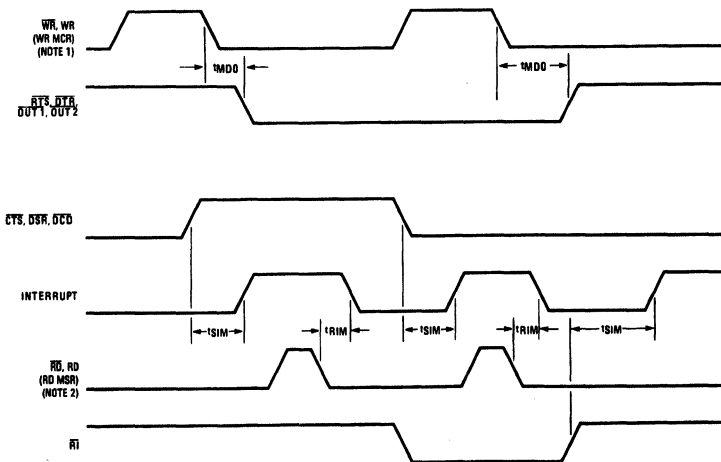
TL/C/8401-9

Note 1: See Write Cycle Timing

Note 2: See Read Cycle Timing

4

5.0 Block Diagram
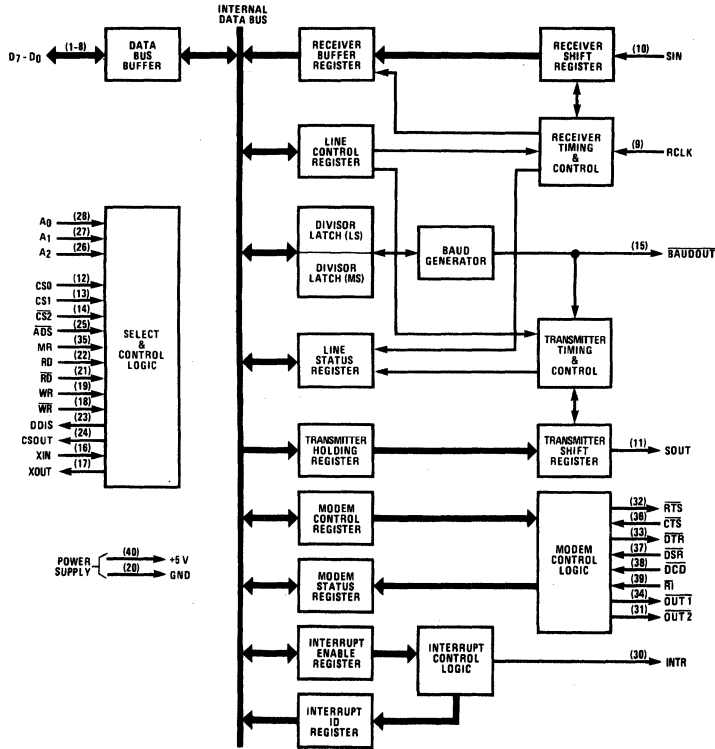
TL/C/8401–10

Note: Applicable pinout numbers are included within parenthesis.

6.0 Pin Descriptions

The following describes the function of all UART pins. Some of these descriptions reference internal circuits.

In the following descriptions, a low represents a logic 0 (0V nominal) and a high represents a logic 1 (+2.4V nominal).

A0, A1, A2: Register Select Pins 26–28: Address signals connected to these 3 inputs select a UART register for the CPU to read from or write to during data transfer. The Register Addresses table associates these address inputs with the register they select. Note that the state of the Divisor Latch Access Bit (DLAB), which is the most significant bit of the Line Control Register, affects the selection of certain UART registers. The DLAB must be set high by the system software to access the Baud Generator Divisor Latches.

ADS: Address Strobe Pin 25: The positive edge of an active Address Strobe (ADS) signal latches the Register Select (A0, A1, A2) and Chip Select (CS0, CS1, CS2) signals.

Note: An active ADS input is required when the Register Select (A0, A1, A2) signals are not stable for the duration of a read or write operation. If not required, tie the ADS input permanently low.

BAUDOUT: Baud Out Pin 15: This is the 16 × clock signal from the transmitter section of the UART. The clock rate is equal to the main reference oscillator frequency divided by the specified divisor in the Baud Generator Divisor Latches. The BAUDOUT may also be used for the receiver section by tying this output to the RCLK input of the chip.

Register Addresses

DLAB	A_2	A_1	A_0	Register
0	0	0	0	Receiver Buffer (read), Transmitter Holding Register (write)
0	0	0	1	Interrupt Enable
X	0	1	0	Interrupt Identification (read only)
X	0	1	1	Line Control
X	1	0	0	MODEM Control
X	1	0	1	Line Status
X	1	1	0	MODEM Status
X	1	1	1	Scratch
1	0	0	0	Divisor Latch (least significant byte)
1	0	0	1	Divisor Latch (most significant byte)

6.0 Pin Descriptions (Continued)

CS0, CS1, $\overline{CS2}$: Chip Select Pins 12–14: When CS0 and CS1 are high and $\overline{CS2}$ is low, the chip is selected. This enables communication between the UART and the CPU. The positive edge of an active Address Strobe signal latches the decoded chip select signals, completing chip selection. If $\overline{ADS}$ is always low, valid chip selects should stabilize according to the t_{CSW} parameter.

CSOUT: Chip Select Out Pin 24: When high, it indicates that the chip has been selected by active, CS0, CS1, and $\overline{CS2}$ inputs. No data transfer can be initiated until the CSOUT signal is a logic 1. CSOUT goes low when the UART is deselected.

$\overline{CTS}$: Clear to Send Pin 36: When low, this indicates that the MODEM or data set is ready to exchange data. The $\overline{CTS}$ signal is a MODEM status input whose conditions can be tested by the CPU reading bit 4 (CTS) of the MODEM Status Register. Bit 4 is the complement of the $\overline{CTS}$ signal. Bit 0 (DCTS) of the MODEM Status Register indicates whether the $\overline{CTS}$ input has changed state since the previous reading of the MODEM Status Register. $\overline{CTS}$ has no effect on the Transmitter.

Note: Whenever the CTS bit of the MODEM Status Register changes state, an interrupt is generated if the MODEM Status Interrupt is enabled.

D_7–D_0: Data Bus, Pins 1–8: This bus is comprised of eight TRI-STATE input/output lines. The bus provides bidirectional communications between the UART and the CPU. Data, control words, and status information are transferred via the D_7–D_0 Data Bus.

$\overline{DCD}$: Data Carrier Detect Pin 38: When low, indicates that the data carrier has been detected by the MODEM or data set. The $\overline{DCD}$ signal is a MODEM status input whose condition can be tested by the CPU reading bit 7 (DCD) of the MODEM Status Register. Bit 7 is the complement of the $\overline{DCD}$ signal. Bit 3 (DDCD) of the MODEM Status Register indicates whether the $\overline{DCD}$ input has changed state since the previous reading of the MODEM Status Register. $\overline{DCD}$ has no effect on the receiver.

Note: Whenever the DCD bit of the MODEM Status Register changes state, an interrupt is generated if the MODEM Status Interrupt is enabled.

DDIS: Driver Disable Pin 23: This goes low whenever the CPU is reading data from the UART. It can disable or control the direction of a data bus transceiver between the CPU and the UART (see Typical Interface for a High Capacity Data Bus).

$\overline{DSR}$: Data Set Ready Pin 37: When low, this indicates that the MODEM or data set is ready to establish the communications link with the UART. The $\overline{DSR}$ signal is a MODEM status input whose condition can be tested by the CPU reading bit 5 (DSR) of the MODEM Status Register. Bit 5 is the complement of the $\overline{DSR}$ signal. Bit 1 (DDSR) of the MODEM Status Register indicates whether the $\overline{DSR}$ input has changed state since the previous reading of the MODEM Status Register.

Note: Whenever the DSR bit of the MODEM Status Register changes state, an interrupt is generated if the MODEM Status Interrupt is enabled.

$\overline{DTR}$: Data Terminal Ready Pin 33: When low, this informs the MODEM or data set that the UART is ready to establish a communications link. The $\overline{DTR}$ output signal can be set to an active low by programming bit 0 (DTR) of the MODEM Control Register to a high level. A Master Reset operation sets this signal to its inactive (high) state. Loop mode operation holds this signal in its inactive state.

INTR: Interrupt Pin 30: This goes high whenever any one of the following interrupt types has an active high condition and is enabled via the IER: Receiver Line Status; Received Data Available; Transmitter Holding Register Empty; and MODEM Status. The INTR signal is reset low upon the appropriate interrupt service or a Master Reset operation.

MR: Master Reset Pin 35: When this input is high, it clears all the registers (except the Receiver Buffer, Transmitter Holding, and Divisor Latches), and the control logic of the UART. The states of various output signals (SOUT, INTR, $\overline{OUT\ 1}$, $\overline{OUT\ 2}$, $\overline{RTS}$, $\overline{DTR}$) are affected by an active MR input. (Refer to Table I.) This input is buffered with a TTL-compatible Schmitt Trigger with 0.5V typical hysteresis.

$\overline{OUT\ 1}$: Output 1 Pin 34: This user-designated output can be set to an active low by programming bit 2 (OUT 1) of the MODEM Control Register to a high level. A Master Reset operation sets this signal to its inactive (high) state. Loop mode operation holds this signal in its inactive state. In the XMOS parts this will achieve TTL levels.

$\overline{OUT\ 2}$: Output 2 Pin 31: This user-designated output can be set to an active low, by programming bit 3 (OUT 2) of the MODEM Control Register to a high level. A Master Reset operation sets this signal to its inactive (high) state. Loop mode operation holds this signal in its inactive state. In the XMOS parts this will achieve TTL levels.

RCLK: Receiver Clock Pin 9: This input is the 16 × baud rate clock for the receiver section of the chip.

RD, $\overline{RD}$, Read Pins 22 and 21: When RD is high or $\overline{RD}$ is low while the chip is selected, the CPU can read status information or data from the selected UART register.

Note: Only an active RD or $\overline{RD}$ input is required to transfer data from the UART during a read operation. Therefore, tie either the RD input permanently low or the $\overline{RD}$ input permanently high, when it is not used.

$\overline{RI}$: Ring Indicator Pin 39: When low, this indicates that a telephone ringing signal has been received by the MODEM or data set. The $\overline{RI}$ signal is a MODEM status input whose condition can be tested by the CPU reading bit 6 (RI) of the MODEM Status Register. Bit 6 is the complement of the $\overline{RI}$ signal. Bit 2 (TERI) of the MODEM Status Register indicates whether the $\overline{RI}$ input signal has changed from a low to a high state since the previous reading of the MODEM Status Register.

Note: Whenever the RI bit of the MODEM Status Register changes from a high to a low state, an interrupt is generated if the MODEM Status interrupt is enabled.

$\overline{RTS}$: Request to Send Pin 32: When low, this informs the MODEM or data set that the UART is ready to exchange data. The $\overline{RTS}$ output signal can be set to an active low by programming bit 1 (RTS) of the MODEM Control Register to a high level. A Master Reset operation sets this signal to its inactive (high) state. Loop mode operation holds this signal in its inactive state.

SIN: Serial Input Pin 10: Serial data input from the communications link (peripheral device, MODEM, or data set).

SOUT: Serial Output Pin 11: This is the composite serial data output to the communications link (peripheral, MODEM or data set). The SOUT signal is set to the Marking (logic 1) state upon a Master Reset operation or when the transmitter is idle.

V_{CC}, Pin 40: +5V supply.

V_{SS}, Pin 20: Ground (0V) reference.

4

6.0 Pin Descriptions (Continued)

WR, WR̄: Write Pins 19 and 18: When WR is high or WR̄ is low while the chip is selected, the CPU can write control words or data into the selected UART register.

Note: Only an active WR or WR̄ input is required to transfer data to the UART during a write operation. Therefore, tie either the WR input permanently low or the WR̄ input permanently high, when it is not used.

XIN: (External Crystal Input), Pin 16: This signal input is used in conjunction with XOUT to form a feedback circuit for the baud rate generator's oscillator. If a clock signal will be generated off-chip, then it should drive the baud rate generator through this pin.

XOUT: (External Crystal Output), Pin 17: This signal output is used in conjunction with XIN to form a feedback circuit for the baud rate generator's oscillator. If the clock signal will be generated off-chip, then this pin is unused.

7.0 Connection Diagrams

Dual-In-Line Package

TL/C/8401–11

Top View

Order Number NS16450N, NS-16450N,
INS8250AN, NS16C450N or INS82C50AN
See NS Package Number N40A

PCC Package
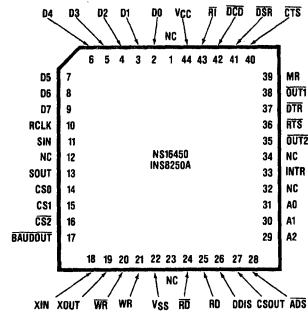
TL/C/8401–18

Top View

Order Number NS16450V, NS-16450V,
INS8250AV, NS16C450V or INS82C50AV
See NS Package Number V44A

TABLE I. UART Reset Functions

Register/Signal	Reset Control	Reset State
Interrupt Enable Register	Master Reset	**0000** 0000 (Note 1)
Interrupt Identification Register	Master Reset	**0000 0**001
Line Control Register	Master Reset	0000 0000
MODEM Control Register	Master Reset	**0000** 0000
Line Status Register	Master Reset	**0**110 0000
MODEM Status Register	Master Reset	XXXX 0000 (Note 2)
SOUT	Master Reset	High
INTR (RCVR Errs)	Read LSR/MR	Low
INTR (RCVR Data Ready)	Read RBR/MR	Low
INTR (THRE)	Read IIR/Write THR/MR	Low
INTR (Modem Status Changes)	Read MSR/MR	Low
OUT 2	Master Reset	High
RTS	Master Reset	High
DTR	Master Reset	High
OUT 1	Master Reset	High

Note 1: Boldface bits are permanently low.

Note 2: Bits 7–4 are driven by the input signals.

8.0 Registers

The system programmer may access any of the UART registers summarized in Table II via the CPU. These registers control UART operations including transmission and reception of data. Each register bit in Table II has its name and reset state shown.

8.1 LINE CONTROL REGISTER

The system programmer specifies the format of the asynchronous data communications exchange and sets the Divisor Latch Access bit via the Line Control Register (LCR). The programmer can also read the contents of the Line Control Register. The read capability simplifies system programming and eliminates the need for separate storage in system memory of the line characteristics. Table II shows the contents of the LCR. Details on each bit follow:

Bits 0 and 1: These two bits specify the number of bits in each transmitted or received serial character. The encoding of bits 0 and 1 is as follows:

Bit 1	Bit 0	Character Length
0	0	5 Bits
0	1	6 Bits
1	0	7 Bits
1	1	8 Bits

Bit 2: This bit specifies the number of Stop bits transmitted and received in each serial character. If bit 2 is a logic 0, one Stop bit is generated in the transmitted data. If bit 2 is a logic 1 when a 5-bit word length is selected via bits 0 and 1, one and a half Stop bits are generated. If

TABLE II. Summary of Registers

Bit No.	0 DLAB = 0 Receiver Buffer Register (Read Only)	0 DLAB = 0 Transmitter Holding Register (Write Only)	1 DLAB = 0 Interrupt Enable Register	2 Interrupt Ident. Register (Read Only)	3 Line Control Register	4 MODEM Control Register	5 Line Status Register	6 MODEM Status Register	7 Scratch Register	0 DLAB = 1 Divisor Latch (LS)	1 DLAB = 1 Divisor Latch (MS)
	RBR	THR	IER	IIR	LCR	MCR	LSR	MSR	SCR	DLL	DLM
0	Data Bit 0 (Note 1)	Data Bit 0	Received Data Available	"0" if Interrupt Pending	Word Length Select Bit 0 (WLS0)	Data Terminal Ready (DTR)	Data Ready (DR)	Delta Clear to Send (DCTS)	Bit 0	Bit 0	Bit 8
1	Data Bit 1	Data Bit 1	Transmitter Holding Register Empty	Interrupt ID Bit (0)	Word Length Select Bit 1 (WLS1)	Request to Send (RTS)	Overrun Error (OE)	Delta Data Set Ready (DDSR)	Bit 1	Bit 1	Bit 9
2	Data Bit 2	Data Bit 2	Receiver Line Status	Interrupt ID Bit (1)	Number of Stop Bits (STB)	Out 1	Parity Error (PE)	Trailing Edge Ring Indicator (TERI)	Bit 2	Bit 2	Bit 10
3	Data Bit 3	Data Bit 3	MODEM Status	0	Parity Enable (PEN)	Out 2	Framing Error (FE)	Delta Data Carrier Detect (DDCD)	Bit 3	Bit 3	Bit 11
4	Data Bit 4	Data Bit 4	0	0	Even Parity Select (EPS)	Loop	Break Interrupt (BI)	Clear to Send (CTS)	Bit 4	Bit 4	Bit 12
5	Data Bit 5	Data Bit 5	0	0	Stick Parity	0	Transmitter Holding Register (THRE)	Data Set Ready (DSR)	Bit 5	Bit 5	Bit 13
6	Data Bit 6	Data Bit 6	0	0	Set Break	0	Transmitter Empty (TEMT)	Ring Indicator (RI)	Bit 6	Bit 6	Bit 14
7	Data Bit 7	Data Bit 7	0	0	Divisor Latch Access Bit (DLAB)	0	0	Data Carrier Detect (DCD)	Bit 7	Bit 7	Bit 15

Note 1: Bit 0 is the least significant bit. It is the first bit serially transmitted or received.

4

8.0 Registers (Continued)

bit 2 is a logic 1 when either a 6-, 7-, or 8-bit word length is selected, two Stop bits are generated. The Receiver checks the first Stop-bit only, regardless of the number of Stop bits selected.

Bit 3: This bit is the Parity Enable bit. When bit 3 is a logic 1, a Parity bit is generated (transmit data) or checked (receive data) between the last data word bit and Stop bit of the serial data. (The Parity bit is used to produce an even or odd number of 1s when the data word bits and the Parity bit are summed.)

Bit 4: This bit is the Even Parity Select bit. When bit 3 is a logic 1 and bit 4 is a logic 0, an odd number of logic 1s is transmitted or checked in the data word bits and Parity bit. When bit 3 is a logic 1 and bit 4 is a logic 1, an even number of logic 1s is transmitted or checked.

Bit 5: This bit is the Stick Parity bit. When bits 3, 4 and 5 are logic 1 the Parity bit is transmitted and checked as a logic 0. If bits 3 and 5 are 1 and bit 4 is a logic 0 then the Parity bit is transmitted and checked as a logic 1. If bit 5 is a logic 0 Stick Parity is disabled.

Bit 6: This bit is the Break Control bit. It causes a break condition to be transmitted by the UART. When it is set to a logic 1, the serial output (SOUT) is forced to the Spacing (logic 0) state. The break is disabled by clearing bit 6 to a logic 0. The Break Control bit acts only on SOUT and has no effect on the transmitter logic.

Note: This feature enables the CPU to alert a terminal in a computer communications system. If the following sequence is used, no erroneous or extraneous characters will be transmitted because of the break.

1. Load all 0s, pad character, in response to THRE.

2. Set break after the next THRE.

3. Wait for the transmitter to be idle, (TEMT = 1), and clear break when normal transmission has to be restored.

During the break, the Transmitter can be used as a character timer to accurately establish the break duration.

Bit 7: This bit is the Divisor Latch Access Bit (DLAB). It must be set high (logic 1) to access the Divisor Latches of the Baud Generator during a Read or Write operation. It must be set low (logic 0) to access the Receiver Buffer, the Transmitter Holding Register, or the Interrupt Enable Register.

8.2 TYPICAL CLOCK CIRCUITS

TL/C/8401–12

TL/C/8401–13

Typical Oscillator Networks

Crystal	R_P	R_{X2}	C_1	C_2
1.8–3.1 MHz	1 MΩ	1.5k	10–30 pF	40–60 pF

Note: These R and C values are approximate and may vary 2X depending on the crystal characteristics. All crystal circuits should be designed specifically for the system.

TABLE III. Baud Rates Using 1.8432 MHz Crystal

Desired Baud Rate	Decimal Divisor Used to Generate 16 x Clock	Percent Error Difference Between Desired and Actual
50	2304	—
75	1536	—
110	1047	0.026
134.5	857	0.058
150	768	—
300	384	—
600	192	—
1200	96	—
1800	64	—
2000	58	0.69
2400	48	—
3600	32	—
4800	24	—
7200	16	—
9600	12	—
19200	6	—
38400	3	—
56000	2	2.86

TABLE IV. Baud Rates Using 3.072 MHz Crystal

Desired Baud Rate	Decimal Divisor Used to Generate 16 x Clock	Percent Error Difference Between Desired and Actual
50	3840	—
75	2560	—
110	1745	0.026
134.5	1428	0.034
150	1280	—
300	640	—
600	320	—
1200	160	—
1800	107	0.312
2000	96	—
2400	80	—
3600	53	0.628
4800	40	—
7200	27	1.23
9600	20	—
19200	10	—
38400	5	—

8.0 Registers (Continued)

8.3 PROGRAMMABLE BAUD GENERATOR

The UART contains a programmable Baud Generator that is capable of taking any clock input from DC to 3.1 MHz and dividing it by any divisor from 1 to $2^{16}-1$. The output frequency of the Baud Generator is 16 $\times$ the Baud [divisor # = (frequency input) ÷ (baud rate $\times$ 16)]. Two 8-bit latches store the divisor in a 16-bit binary format. These Divisor Latches must be loaded during initialization in order to ensure proper operation of the Baud Generator. Upon loading either of the Divisor Latches, a 16-bit Baud counter is immediately loaded.

Tables III and IV provide decimal divisors to use with crystal frequencies of 1.8432 MHz and 3.072 MHz respectively for common baud rates. For baud rates of 38400 and below, the error obtained is minimal. The accuracy of the desired baud rate is dependent on the crystal frequency chosen. Using a division of 0 is **not** recommended.

Note: The maximum operating frequency of the Baud Generator is 3.1 MHz. However, when using divisors of 3 and below, the maximum frequency is equal to the divisor in MHz. For example, if the divisor is 1, then the maximum frequency is 1 MHz. In no case should the data rate be greater than 56k Baud.

8.4 LINE STATUS REGISTER

This 8-bit register provides status information to the CPU concerning the data transfer. Table II shows the contents of the Line Status Register. Details on each bit follow:

Bit 0: This bit is the receiver Data Ready (DR) indicator. Bit 0 is set to a logic 1 whenever a complete incoming character has been received and transferred into the Receiver Buffer Register. Bit 0 is reset to a logic 0 by reading the data in the Receiver Buffer Register.

Bit 1: This bit is the Overrun Error (OE) indicator. Bit 1 indicates that data in the Receiver Buffer Register was not read by the CPU before the next character was transferred into the Receiver Buffer Register, thereby destroying the previous character. The OE indicator is set to a logic 1 upon detection of an overrun condition and reset whenever the CPU reads the contents of the Line Status Register.

Bit 2: This bit is the Parity Error (PE) indicator. Bit 2 indicates that the received data character does not have the correct even or odd parity, as selected by the even-parity-select bit. The PE bit is set to a logic 1 upon detection of a parity error and is reset to a logic 0 whenever the CPU reads the contents of the Line Status Register.

Bit 3: This bit is the Framing Error (FE) indicator. Bit 3 indicates that the received character did not have a valid Stop bit. Bit 3 is set to a logic 1 whenever the Stop bit following the last data bit or parity bit is a logic 0 (Spacing level). The FE indicator is reset whenever the CPU reads the contents of the Line Status Register. The UART will try to resynchronize after a framing error. To do this it assumes that the framing error was due to the next start bit, so it samples this "start" bit twice and then takes in the "data".

Bit 4: This bit is the Break Interrupt (BI) indicator. Bit 4 is set to a logic 1 whenever the received data input is held in the Spacing (logic 0) state for longer than a full word transmission time (that is, the total time of Start bit + data bits + Parity + Stop bits). The BI indicator is reset whenever the CPU reads the contents of the Line Status Register. Restarting after a break is received, requires the SIN pin to be logical 1 for at least ½ bit time.

Note: Bits 1 through 4 are the error conditions that produce a Receiver Line Status interrupt whenever any of the corresponding conditions are detected and the interrupt is enabled.

Bit 5: This bit is the Transmitter Holding Register Empty (THRE) indicator. Bit 5 indicates that the UART is ready to accept a new character for transmission. In addition, this bit causes the UART to issue an interrupt to the CPU when the Transmit Holding Register Empty Interrupt enable is set high. The THRE bit is set to a logic 1 when a character is transferred from the Transmitter Holding Register into the Transmitter Shift Register. The bit is reset to logic 0 whenever the CPU loads the Transmitter Holding Register.

Bit 6: This bit is the Transmitter Empty (TEMT) indicator. Bit 6 is set to a logic 1 whenever the Transmitter Holding Register (THR) and the Transmitter Shift Register (TSR) are both empty. It is reset to a logic 0 whenever either the THR or TSR contains a data character.

Bit 7: This bit is permanently set to logic 0.

Note: The Line Status Register is intended for read operations only. Writing to this register is not recommended as this operation is only used for factory testing.

TABLE V. Interrupt Control Functions

Interrupt Identification Register				Interrupt Set and Reset Functions		
Bit 2	Bit 1	Bit 0	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Control
0	0	1	—	None	None	—
1	1	0	Highest	Receiver Line Status	Overrun Error or Parity Error or Framing Error or Break Interrupt	Reading the Line Status Register
1	0	0	Second	Received Data Available	Receiver Data Available	Reading the Receiver Buffer Register
0	1	0	Third	Transmitter Holding Register Empty	Transmitter Holding Register Empty	Reading the IIR Register (if source of interrupt) or Writing into the Transmitter Holding Register
0	0	0	Fourth	MODEM Status	Clear to Send or Data Set Ready or Ring Indicator or Data Carrier Detect	Reading the MODEM Status Register

4

8.0 Registers (Continued)

8.5 INTERRUPT IDENTIFICATION REGISTER

In order to provide minimum software overhead during data character transfers, the UART prioritizes interrupts into four levels and records these in the Interrupt Identification Register. The four levels of interrupt conditions in order of priority are Receiver Line Status; Received Data Ready; Transmitter Holding Register Empty; and MODEM Status.

When the CPU accesses the IIR, the UART freezes all interrupts and indicates the highest priority pending interrupt to the CPU. While this CPU access is occurring, the UART records new interrupts, but does not change its current indication until the access is complete. Table II shows the contents of the IIR. Details on each bit follow:

Bit 0: This bit can be used in an interrupt environment to indicate whether an interrupt condition is pending. When bit 0 is a logic 0, an interrupt is pending and the IIR contents may be used as a pointer to the appropriate interrupt service routine. When bit 0 is a logic 1, no interrupt is pending.

Bits 1 and 2: These two bits of the IIR are used to identify the highest priority interrupt pending as indicated in Table V.

Bits 3 through 7: These five bits of the IIR are always logic 0.

8.6 INTERRUPT ENABLE REGISTER

This register enables the four types of UART interrupts. Each interrupt can individually activate the interrupt (INTR) output signal. It is possible to totally disable the interrupt system by resetting bits 0 through 3 of the Interrupt Enable Register (IER). Similarly, setting bits of this register to a logic 1, enables the selected interrupt(s). Disabling an interrupt prevents it from being indicated as active in the IIR and from activating the INTR output signal. All other system functions operate in their normal manner, including the setting of the Line Status and MODEM Status Registers. Table II shows the contents of the IER. Details on each bit follow.

Bit 0: This bit enables the Received Data Available Interrupt when set to logic 1.

Bit 1: This bit enables the Transmitter Holding Register Empty Interrupt when set to logic 1.

Bit 2: This bit enables the Receiver Line Status Interrupt when set to logic 1.

Bit 3: This bit enables the MODEM Status Interrupt when set to logic 1.

Bits 4 through 7: These four bits are always logic 0.

8.7 MODEM CONTROL REGISTER

This register controls the interface with the MODEM or data set (or a peripheral device emulating a MODEM). The contents of the MODEM Control Register (MCR) are indicated in Table II and are described below. Table II shows the contents of the MCR. Details on each bit follow.

Bit 0: This bit controls the Data Terminal Ready ($\overline{\text{DTR}}$) output. When bit 0 is set to a logic 1, the $\overline{\text{DTR}}$ output is forced to a logic 0. When bit 0 is reset to a logic 0, the $\overline{\text{DTR}}$ output is forced to a logic 1.

Note: The $\overline{\text{DTR}}$ output of the UART may be applied to an EIA inverting line driver (such as the DS1488) to obtain the proper polarity input at the succeeding MODEM or data set.

Bit 1: This bit controls the Request to Send ($\overline{\text{RTS}}$) output. Bit 1 affects the $\overline{\text{RTS}}$ output in a manner identical to that described above for bit 0.

Bit 2: This bit controls the Output 1 ($\overline{\text{OUT 1}}$) signal, which is an auxiliary user-designated output. Bit 2 affects the $\overline{\text{OUT 1}}$ output in a manner identical to that described above for bit 0.

Bit 3: This bit controls the Output 2 ($\overline{\text{OUT 2}}$) signal, which is an auxiliary user-designated output. Bit 3 affects the $\overline{\text{OUT 2}}$ output in a manner identical to that described above for bit 0.

Bit 4: This bit provides a local loopback feature for diagnostic testing of the UART. When bit 4 is set to logic 1, the following occur: the transmitter Serial Output (SOUT) is set to the Marking (logic 1) state; the receiver Serial Input (SIN) is disconnected; the output of the Transmitter Shift Register is "looped back" into the Receiver Shift Register input; the four MODEM Control inputs ($\overline{\text{DSR}}$, $\overline{\text{CTS}}$, $\overline{\text{RI}}$, and $\overline{\text{DCD}}$) are disconnected; and the four MODEM Control outputs ($\overline{\text{DTR}}$, $\overline{\text{RTS}}$, $\overline{\text{OUT 1}}$, and $\overline{\text{OUT 2}}$) are internally connected to the four MODEM Control inputs. The MODEM Control output pins are forced to their inactive state (high). In the diagnostic mode, data that is transmitted is immediately received. This feature allows the processor to verify the transmit-and received-data paths of the UART.

In the diagnostic mode, the receiver and transmitter interrupts are fully operational. The MODEM Control Interrupts are also operational, but the interrupts' sources are now the lower four bits of the MODEM Control Register instead of the four MODEM Control inputs. The interrupts are still controlled by the Interrupt Enable Register.

Bits 5 through 7: These bits are permanently set to logic 0.

8.8 MODEM STATUS REGISTER

This register provides the current state of the control lines from the MODEM (or peripheral device) to the CPU. In addition to this current-state information, four bits of the MODEM Status Register provide change information. These bits are set to a logic 1 whenever a control input from the MODEM changes state. They are reset to logic 0 whenever the CPU reads the MODEM Status Register.

8.0 Registers (Continued)

Table II shows the contents of the MSR. Details on each bit follow.

Bit 0: This bit is the Delta Clear to Send (DCTS) indicator. Bit 0 indicates that the $\overline{CTS}$ input to the chip has changed state since the last time it was read by the CPU.

Bit 1: This bit is the Delta Data Set Ready (DDSR) indicator. Bit 1 indicates that the $\overline{DSR}$ input to the chip has changed state since the last time it was read by the CPU.

Bit 2: This bit is the Trailing Edge of Ring Indicator (TERI) detector. Bit 2 indicates that the $\overline{RI}$ input to the chip has changed from a low to a high state.

Bit 3: This bit is the Delta Data Carrier Detect (DDCD) indicator. Bit 3 indicates that the $\overline{DCD}$ input to the chip has changed state.

Note: Whenever bit 0, 1, 2, or 3 is set to logic 1, a MODEM Status Interrupt is generated.

Bit 4: This bit is the complement of the Clear to Send ($\overline{CTS}$) input. If bit 4 (loop) of the MCR is set to a 1, this bit is equivalent to RTS in the MCR.

Bit 5: This bit is the complement of the Data Set Ready ($\overline{DSR}$) input. If bit 4 of the MCR is set to a 1, this bit is equivalent to DTR in the MCR.

Bit 6: This bit is the complement of the Ring Indicator ($\overline{RI}$) input. If bit 4 of the MCR is set to a 1, this bit is equivalent to OUT 1 in the MCR.

Bit 7: This bit is the complement of the Data Carrier Detect ($\overline{DCD}$) input. If bit 4 of the MCR is set to a 1, this bit is equivalent to OUT 2 in the MCR.

8.9 SCRATCHPAD REGISTER

This 8-bit Read/Write Register does not control the UART in any way. It is intended as a scratchpad register to be used by the programmer to hold data temporarily.

4

Typical shows the basic connections of an INS8250A to an 8088 CPU
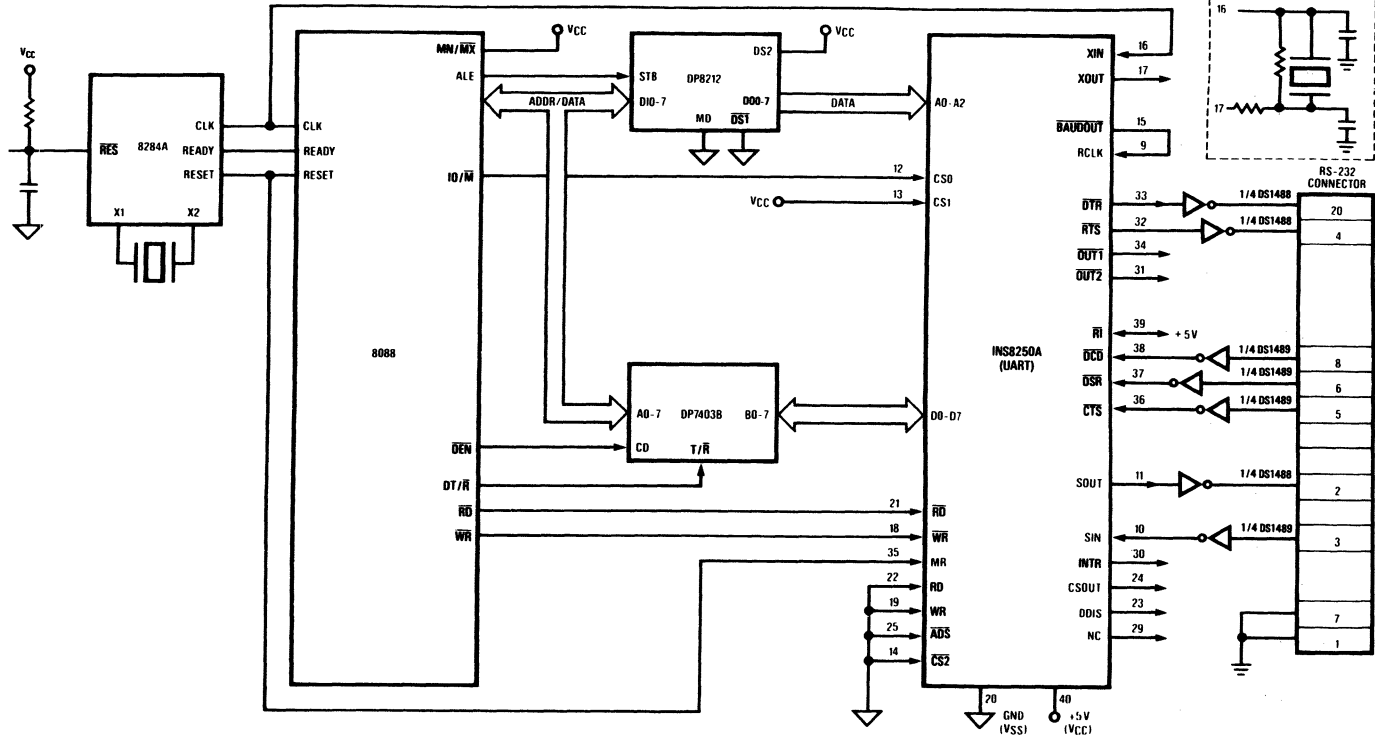
TL/C/8401-15

9.0 Typical Applications (Continued)

**Typical Interface for a
High-Capacity Data Bus**

TL/C/8401-16

**Typical Supply Current vs
Temperature, Normalized**

TL/C/8401-17
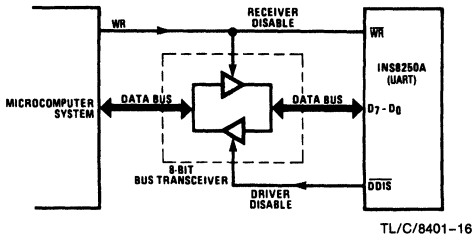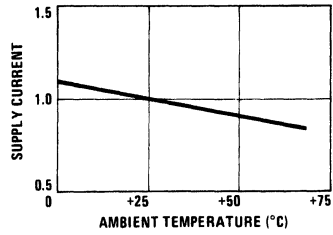
10.0 Ordering Information

Order Number	Description
Plastic Dip Package	
NS16450N	
or	high speed part
NS-16450N	
INS8250AN	$V_{CC} = 5V \pm 5\%$
NS16C450N	CMOS high speed part
INS82C50AN	CMOS $V_{CC} = 5V \pm 5\%$
Plastic Chip Carrier Package	
NS16450V	
or	high speed part
NS-16450V	
INS8250A	$V_{CC} = 5V \pm 5\%$
NS16C450V	CMOS high speed part
INS82C50AV	CMOS $V_{CC} = 5V \pm 5\%$

4

National Semiconductor

NS16550AF Universal Asynchronous Receiver/Transmitter with FIFOs†

General Description

The NS16550AF is an improved version of the NS16450 Universal Asynchronous Receiver/Transmitter (UART). Functionally identical to the NS16450 on powerup (CHARACTER mode)* the NS16550AF can be put into an alternate mode (FIFO mode) to relieve the CPU of excessive software overhead.

In this mode internal FIFOs are activated allowing 16 bytes (plus 3 bits of error data per byte in the RCVR FIFO) to be stored in both receive and transmit modes. All the logic is on chip to minimize system overhead and maximize system efficiency. Two pin functions have been changed to allow signalling of DMA transfers.

The UART performs serial-to-parallel conversion on data characters received from a peripheral device or a MODEM, and parallel-to-serial conversion on data characters received from the CPU. The CPU can read the complete status of the UART at any time during the functional operation. Status information reported includes the type and condition of the transfer operations being performed by the UART, as well as any error conditions (parity, overrun, framing, or break interrupt).

The UART includes a programmable baud rate generator that is capable of dividing the timing reference clock input by divisors of 1 to ($2^{16}-1$), and producing a 16 × clock for driving the internal transmitter logic. Provisions are also included to use this 16 × clock to drive the receiver logic. The UART has complete MODEM-control capability, and a processor-interrupt system. Interrupts can be programmed to the user's requirements, minimizing the computing required to handle the communications link.

The UART is fabricated using National Semiconductor's advanced scaled N-channel silicon-gate MOS process, XMOS.

*Can also be reset to NS16450 Mode under software control.

†Note: This part is patented.

Features

■ Capable of running all existing 16450 software.

■ Pin for pin compatible with the existing 16450 except for CSOUT (24) and NC (29). The former CSOUT and NC pins are $\overline{TXRDY}$ and $\overline{RXRDY}$, respectively.

■ After reset, all registers are identical to the 16450 register set.

■ In the FIFO mode transmitter and receiver are each buffered with 16 byte FIFO's to reduce the number of interrrupts presented to the CPU.

■ Adds or deletes standard asynchronous communication bits (start, stop, and parity) to or from the serial data.

■ Holding and shift registers in the NS16450 Mode eliminate the need for precise synchronization between the CPU and serial data.

■ Independently controlled transmit, receive, line status, and data set interrupts.

■ Programmable baud generator divides any input clock by 1 to (2^{16} − 1) and generates the 16 × clock.

■ Independent receiver clock input.

■ MODEM control functions (CTS, RTS, DSR, DTR, RI, and DCD).

■ Fully programmable serial-interface characteristics:
  — 5-, 6-, 7-, or 8-bit characters
  — Even, odd, or no-parity bit generation and detection
  — 1-, 1½-, or 2-stop bit generation
  — Baud generation (DC to 256k baud).

■ False start bit detection.

■ Complete status reporting capabilities.

■ TRI-STATE® TTL drive for the data and control buses.

■ Line break generation and detection.

■ Internal diagnostic capabilities:
  — Loopback controls for communications link fault isolation
  — Break, parity, overrun, framing error simulation.

■ Full prioritized interrupt system controls.

Basic Configuration

TL/C/8652–1

Table of Contents

4

1.0 Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Note: *Maximum ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended and should be limited to those conditions specified under DC electrical characteristics.*

Temperature Under Bias	0°C to +70°C
Storage Temperature	−65°C to +150°C
All Input or Output Voltages with Respect to V_{SS}	−0.5V to +7.0V
Power Dissipation	1W

2.0 DC Electrical Characteristics

T_A = 0°C to +70°C, V_{CC} = +5V ±5%, V_{SS} = 0V, unless otherwise specified.

Symbol	Parameter	Conditions	Min	Max	Units
V_{ILX}	Clock Input Low Voltage		−0.5	0.8	V
V_{IHX}	Clock Input High Voltage		2.0	V_{CC}	V
V_{IL}	Input Low Voltage		−0.5	0.8	V
V_{IH}	Input High Voltage		2.0	V_{CC}	V
V_{OL}	Output Low Voltage	I_{OL} = 1.6 mA on all (Note 1)		0.4	V
V_{OH}	Output High Voltage	I_{OH} = −1.0 mA (Note 1)	2.4		V
I_{CC}(AV)	Avg. Power Supply Current (V_{CC})	V_{CC} = 5.25V No Loads on output SIN, DSR, DCD, CTS, RI = 2.0V All other inputs = 0.8V		160 (Note 2) 140 (Note 3)	mA mA
I_{IL}	Input Leakage	V_{CC} = 5.25V, V_{SS} = 0V All other pins floating. V_{IN} = 0V, 5.25V		±10	μA
I_{CL}	Clock Leakage			±10	μA
I_{OZ}	TRI-STATE Leakage	V_{CC} = 5.25V, V_{SS} = 0V V_{OUT} = 0V, 5.25V 1) Chip deselected 2) WRITE mode, chip selected		±20	μA
V_{ILMR}	MR Schmitt V_{IL}			0.8	V
V_{IHMR}	MR Schmitt V_{IH}		2.0		V

Note 1: Does not apply to XOUT

Note 2: T_A = 25°C

Note 3: T_A = 70°C

Capacitance T_A = 25°C, V_{CC} = V_{SS} = 0V

Symbol	Parameter	Conditions	Min	Typ	Max	Units
C_{XIN}	Clock Input Capacitance	f_c = 1 MHz Unmeasured pins returned to V_{SS}		15	20	pF
C_{XOUT}	Clock Output Capacitance			20	30	pF
C_{IN}	Input Capacitance			6	10	pF
C_{OUT}	Output Capacitance			10	20	pF

3.0 AC Electrical Characteristics $T_A = 0°C$ to $+70°C$, $V_{CC} = +5V \pm 5\%$

Symbol	Parameter	Conditions	Min	Max	Units
t_{ADS}	Address Strobe Width		60		ns
t_{AH}	Address Hold Time		0		ns
t_{AR}	$\overline{RD}$, RD Delay from Address	(Note 1)	30		ns
t_{AS}	Address Setup Time		60		ns
t_{AW}	$\overline{WR}$, WR Delay from Address	(Note 1)	30		ns
t_{CH}	Chip Select Hold Time		0		ns
t_{CS}	Chip Select Setup Time		60		ns
t_{CSR}	$\overline{RD}$, RD Delay from Chip Select	(Note 1)	30		ns
t_{CSW}	$\overline{WR}$, WR Delay from Select	(Note 1)	30		ns
t_{DH}	Data Hold Time		30		ns
t_{DS}	Data Setup Time		30		ns
t_{HZ}	$\overline{RD}$, RD to Floating Data Delay	@100 pF loading (Note 3)	0	100	ns
t_{MR}	Master Reset Pulse Width		5		μs
t_{RA}	Address Hold Time from $\overline{RD}$, RD	(Note 1)	20		ns
t_{RC}	Read Cycle Delay		125		ns
t_{RCS}	Chip Select Hold Time from $\overline{RD}$, RD	(Note 1)	20		ns
t_{RD}	$\overline{RD}$, RD Strobe Width		125		ns
t_{RDD}	$\overline{RD}$, RD to Driver Enable/Disable	@100 pF loading (Note 3)		60	ns
t_{RVD}	Delay from $\overline{RD}$, RD to Data	@100 pF loading		125	ns
t_{WA}	Address Hold Time from $\overline{WR}$, WR	(Note 1)	20		ns
t_{WC}	Write Cycle Delay		150		ns
t_{WCS}	Chip Select Hold Time from $\overline{WR}$, WR	(Note 1)	20		ns
t_{WR}	$\overline{WR}$, WR Strobe Width		100		ns
t_{XH}	Duration of Clock High Pulse	External Clock (8.0 MHz Max.)	55		ns
t_{XL}	Duration of Clock Low Pulse	External Clock (8.0 MHz Max.)	55		ns
RC	Read Cycle = $t_{AR} + t_{RD} + t_{RC}$		280		ns
WC	Write Cycle = $t_{AW} + t_{WR} + t_{WC}$		280		ns
Baud Generator					
N	Baud Divisor		1	$2^{16} - 1$	
t_{BHD}	Baud Output Positive Edge Delay	100 pF Load		175	ns
t_{BLD}	Baud Output Negative Edge Delay	100 pF Load		175	ns
t_{HW}	Baud Output Up Time	$f_X = 8.0$ MHz, $\div 2$, 100 pF Load	75		ns
t_{LW}	Baud Output Down Time	$f_X = 8.0$ MHz, $\div 2$, 100 pF Load	100		ns
Receiver					
t_{RINT}	Delay from $\overline{RD}$, RD (RD RBR or RD LSR) to Reset Interrupt	100 pF Load		1	μs
t_{RXI}	Delay from $\overline{RD}$ RBR to $\overline{RXRDY}$ Inactive			290	ns
t_{SCD}	Delay from RCLK to Sample Time			2	μs
t_{SINT}	Delay from Stop to Set Interrupt	(Note 2)		1	RCLK Cycles

Note 1: Applicable only when $\overline{ADS}$ is tied low.

Note 2: In the FIFO mode (FCR0 = 1) the trigger level interrupts, the receiver data available indication, the active RXRDY indication and the overrun error indication will be delayed 3 RCLKs. Status indicators (PE, FE, BI) will be delayed 3 RCLKs after the first byte has been received. For subsequently received bytes these indicators will be updated immediately after RDRBR goes inactive. Timeout interrupt is delayed 8 RCLKs.

Note 3: Charge and discharge time is determined by V_{OL}, V_{OH} and the external loading.

4

3.0 AC Electrical Characteristics (Continued)

Symbol	Parameter	Conditions	Min	Max	Units
Transmitter					
t_{HR}	Delay from $\overline{WR}$, WR (WR THR) to Reset Interrupt	100 pF Load		175	ns
t_{IR}	Delay from $\overline{RD}$, RD (RD IIR) to Reset Interrupt (THRE)	100 pF Load		250	ns
t_{IRS}	Delay from Initial INTR Reset to Transmit Start		8	24	BAUDOUT Cycles
t_{SI}	Delay from Initial Write to Interrupt	(Note 1)	16	24	BAUDOUT Cycles
t_{STI}	Delay from Stop to Interrupt (THRE)	(Note 1)	8	8	BAUDOUT Cycles
t_{SXA}	Delay from Start to TXRDY active	100 pF Load		8	BAUDOUT Cycles
t_{WXI}	Delay from Write to TXRDY inactive	100 pF Load		195	ns
Modem Control					
t_{MDO}	Delay from $\overline{WR}$, WR (WR MCR) to Output	100 pF Load		200	ns
t_{RIM}	Delay from $\overline{RD}$, RD to Reset Interrupt (RD MSR)	100 pF Load		250	ns
t_{SIM}	Delay from MODEM Input to Set Interrupt	100 pF Load		250	ns

Note 1: This delay will be lengthened by 1 character time, minus the last stop bit time if the transmitter interrupt delay circuit is active. (See FIFO Interrupt Mode Operation).

4.0 Timing Waveforms (All timings are referenced to valid 0 and valid 1)

External Clock Input (8.0 MHz Max.)

TL/C/8652–2

AC Test Points
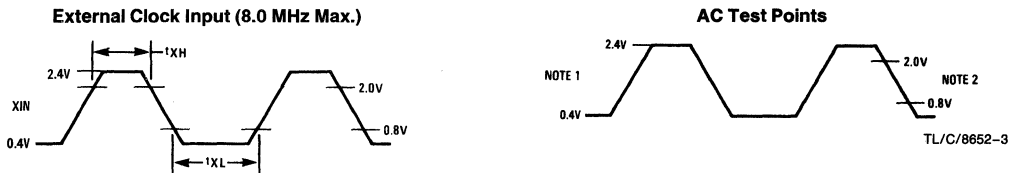
TL/C/8652–3

Note 1: The 2.4V and 0.4V levels are the voltages that the inputs are driven to during AC testing.

Note 2: The 2.0V and 0.8V levels are the voltages at which the timing tests are made.

$\overline{BAUDOUT}$ Timing
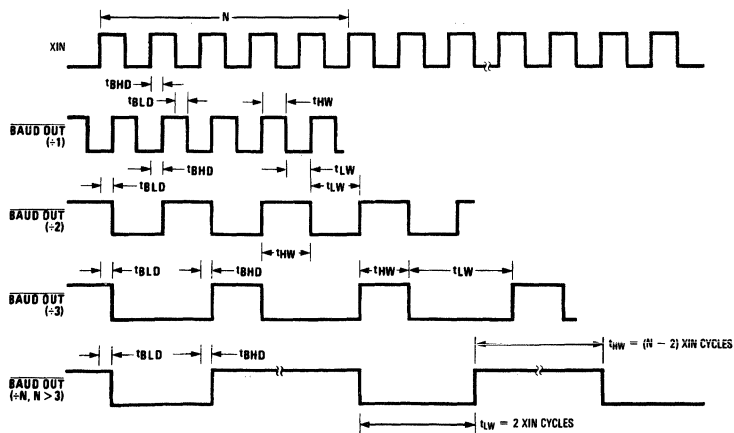
TL/C/8652–4

4.0 Timing Waveforms (Continued)
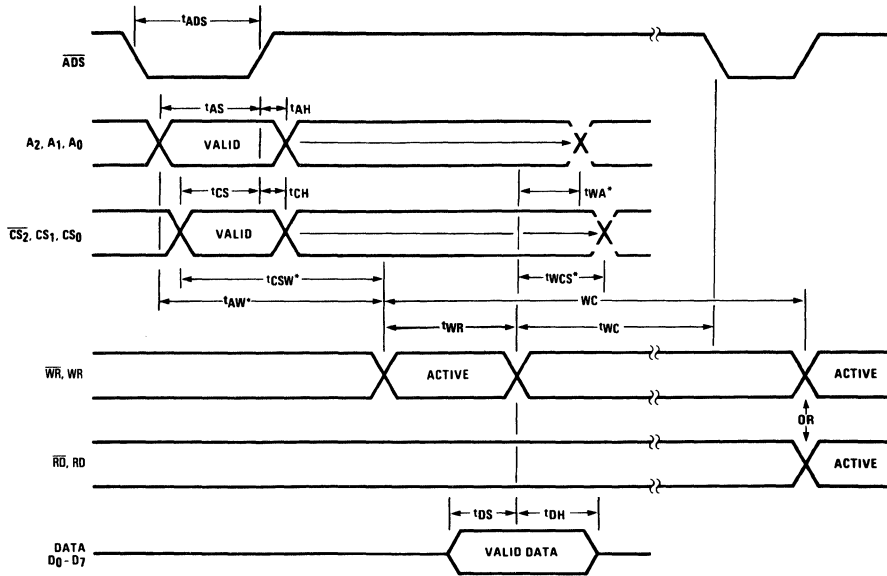
Write Cycle

TL/C/8652–5

*Applicable Only When $\overline{ADS}$ is Tied Low.

Read Cycle
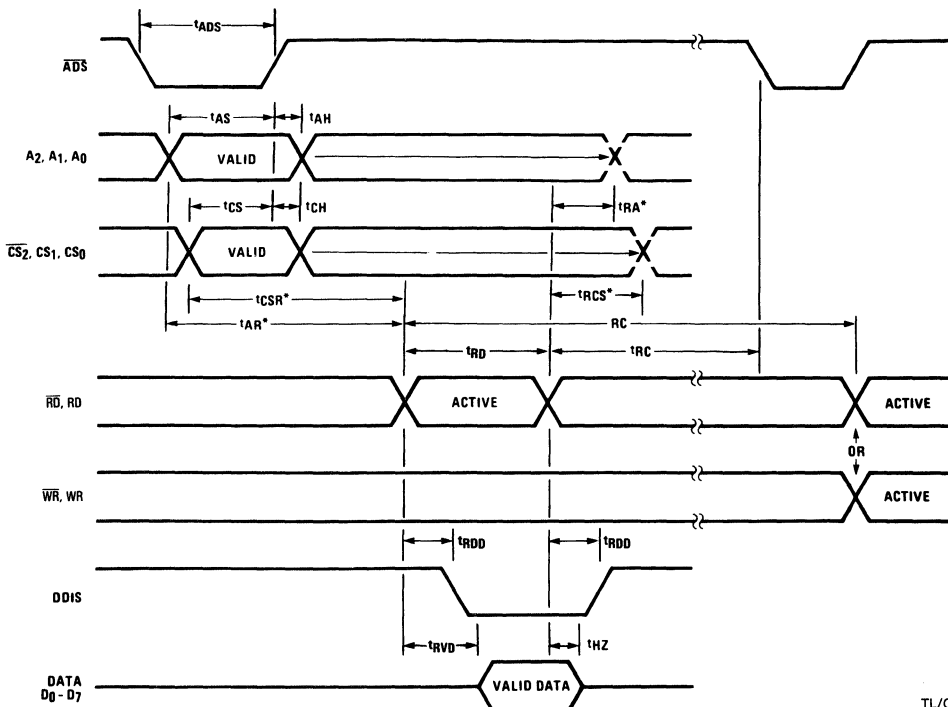
TL/C/8652–6

*Applicable Only When $\overline{ADS}$ is Tied Low.

4

4.0 Timing Waveforms (Continued)

Receiver Timing
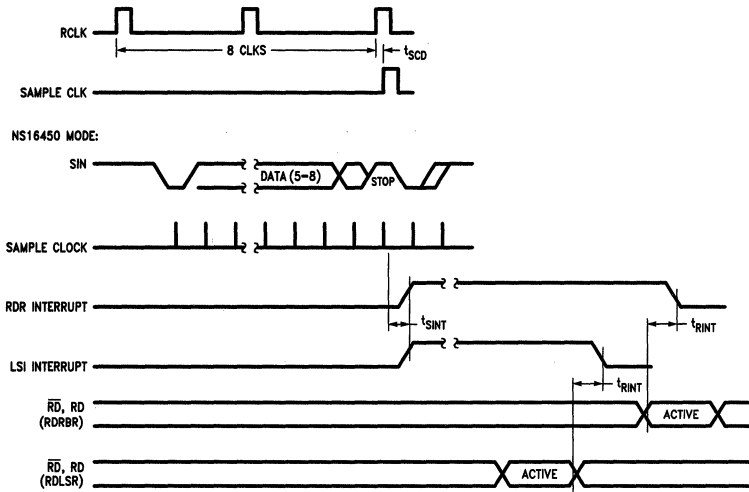
TL/C/8652–7

Transmitter Timing
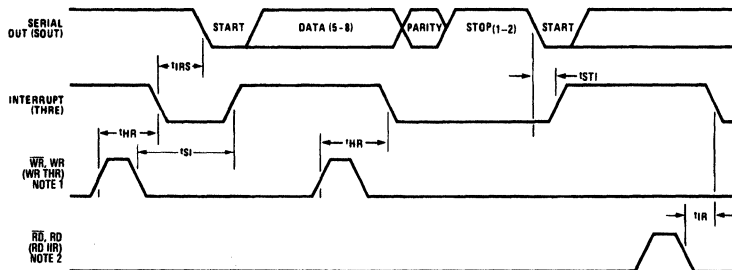
TL/C/8652–8

MODEM Control Timing

TL/C/8652–9

Note 1: See Write Cycle Timing

Note 2: See Read Cycle Timing

4.0 Timing Waveforms (Continued)

RCVR FIFO First Byte (This Sets RDR)

SIN

DATA (5-8)   STOP

SAMPLE CLOCK

TRIGGER LEVEL
INTERRUPT
(FCR6, 7=0, 0)

NOTE 2

t_{SINT}

(FIFO AT OR
ABOVE TRIGGER LEVEL)

(FIFO BELOW
TRIGGER LEVEL)

t_{RINT}

LSI INTERRUPT

t_{RINT}

$\overline{RD}$, RD
(RDLSR)   ACTIVE

$\overline{RD}$, RD
(RDRBR)   ACTIVE
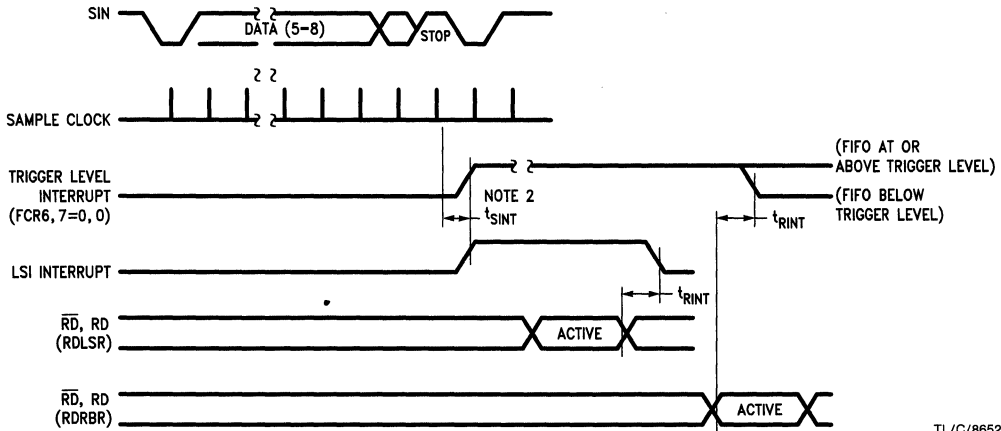
TL/C/8652-10

RCVR FIFO Bytes Other Than the First Byte (RDR Is Already Set)

SIN   STOP

SAMPLE CLOCK

TIMEOUT OR
TRIGGER LEVEL
INTERRUPT

NOTE 2
t_{SINT}

(FIFO AT OR
ABOVE TRIGGER LEVEL)

(FIFO BELOW
TRIGGER LEVEL)

t_{RINT}

LSI INTERRUPT   TOP BYTE OF FIFO

t_{SINT} t_{RINT}

$\overline{RD}$, RD
(RDLSR)   ACTIVE

$\overline{RD}$, RD
(RDRBR)   ACTIVE   ACTIVE

PREVIOUS BYTE
READ FROM FIFO
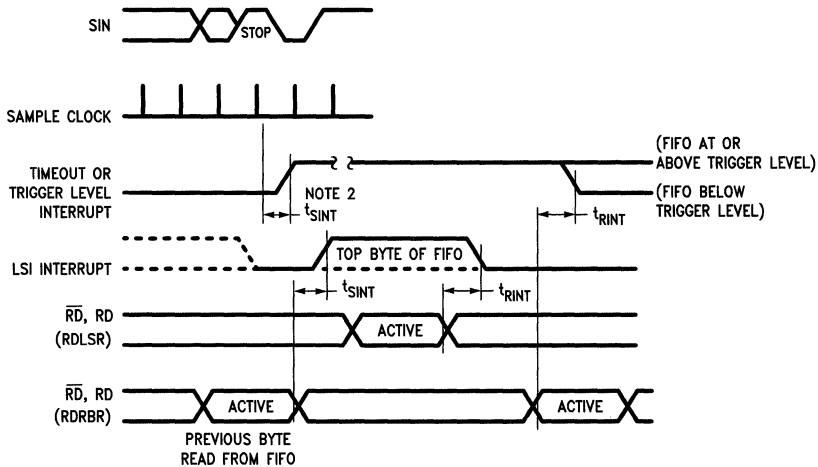
TL/C/8652-11

Receiver Ready (Pin 29) FCR0 = 0 or FCR0 = 1 and FCR3 = 0 (Mode 0)

$\overline{RD}$, RD
(RDRBR)   ACTIVE

NOTE 1

SIN
(FIRST BYTE)   STOP

SAMPLE CLK

$\overline{RXRDY}$

t_{SINT}
NOTE 2

t_{RXI}
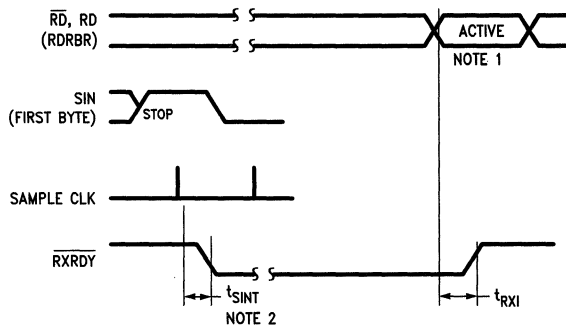
TL/C/8652-12

Note 1: This is the reading of the last byte in the FIFO.

Note 2: If FCR0 = 1, then t_{SINT} = 3 RCLKs. For a timeout interrupt, t_{SINT} = 8 RCLKs.

4

4.0 Timing Waveforms (Continued)

Receiver Ready (Pin 29) FCR0 = 1 and FCR3 = 1 (Mode 1)

RD, RD
(RDRBR)

ACTIVE
NOTE 1

SIN
(FIRST BYTE THAT
REACHES THE
TRIGGER LEVEL)

STOP

SAMPLE CLK

RXRDY

t_{SINT}
NOTE 2

t_{RXI}
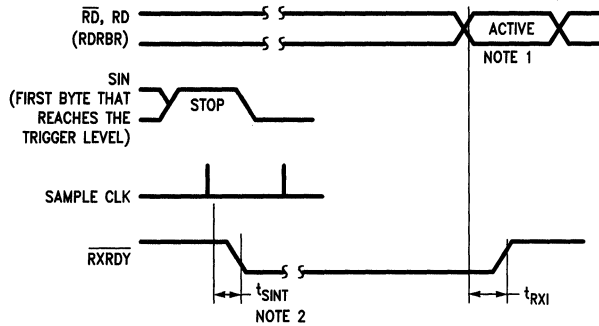
TL/C/8652–13

Note 1: This is the reading of the last byte in the FIFO.

Note 2: If FCR0 = 1, t_{SINT} = 3 RCLKs.

Transmitter Ready (Pin 24) FCR0 = 0 or FCR0 = 1 and FCR3 = 0 (Mode 0)

WR, WR
(WRTHR)

BYTE #1

SOUT

DATA   PARITY   STOP   START

TXRDY

t_{WXI} t_{SXA}

TL/C/8652–14

Transmitter Ready (Pin 24) FCR0 = 1 and FCR3 = 1 (Mode 1)

WR, WR
(WRTHR)

BYTE #16

SOUT

DATA   PARITY   STOP   START   START OF BYTE #16

TXRDY

t_{WXI} FIFO FULL t_{SXA}
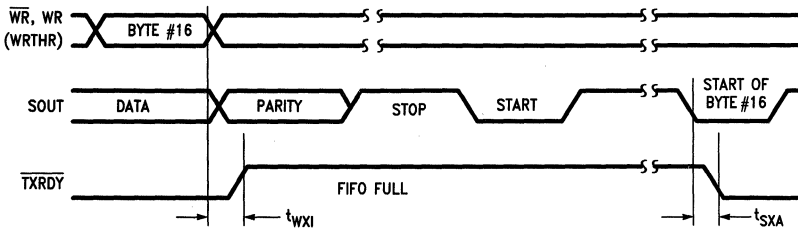
TL/C/8652–15

5.0 Block Diagram

TL/C/8652–16

Note: Applicable pinout numbers are included within parenthesis.

6.0 Pin Descriptions

The following describes the function of all UART pins. Some of these descriptions reference internal circuits.

In the following descriptions, a low represents a logic 0 (0V nominal) and a high represents a logic 1 (+2.4V nominal).

A0, A1, A2, Register Select, Pins 26–28: Address signals connected to these 3 inputs select a UART register for the CPU to read from or write to during data transfer. A table of registers and their addresses is shown below. Note that the state of the Divisor Latch Access Bit (DLAB), which is the most significant bit of the Line Control Register, affects the selection of certain UART registers. The DLAB must be set high by the system software to access the Baud Generator Divisor Latches.

Register Addresses

DLAB	A$_2$	A$_1$	A$_0$	Register
0	0	0	0	Receiver Buffer (read), Transmitter Holding Register (write)
0	0	0	1	Interrupt Enable
X	0	1	0	Interrupt Identification (read)
X	0	1	0	FIFO Control (write)
X	0	1	1	Line Control
X	1	0	0	MODEM Control
X	1	0	1	Line Status
X	1	1	0	MODEM Status
X	1	1	1	Scratch
1	0	0	0	Divisor Latch (least significant byte)
1	0	0	1	Divisor Latch (most significant byte)

ADS, Address Strobe, Pin 25: The positive edge of an active Address Strobe (ADS) signal latches the Register Select (A0, A1, A2) and Chip Select (CS0, CS1, CS2) signals.

Note: An active ADS input is required when the Register Select (A0, A1, A2) and Chip Select (CS0, CS1, CS2) signals are not stable for the duration of a read or write operation. If not required, tie the ADS input permanently low.

BAUDOUT, Baud Out, Pin 15: This is the 16 × clock signal from the transmitter section of the UART. The clock rate is equal to the main reference oscillator frequency divided by the specified divisor in the Baud Generator Divisor Latches. The BAUDOUT may also be used for the receiver section by tying this output to the RCLK input of the chip.

CS0, CS1, CS2, Chip Select, Pins 12–14: When CS0 and CS1 are high and CS2 is low, the chip is selected. This enables communication between the UART and the CPU. The positive edge of an active Address Strobe signal latches the decoded chip select signals, completing chip selection. If ADS is always low, valid chip selects should stabilize according to the t$_{CSW}$ parameter.

CTS, Clear to Send, Pin 36: When low, this indicates that the MODEM or data set is ready to exchange data. The CTS signal is a MODEM status input whose conditions can be tested by the CPU reading bit 4 (CTS) of the MODEM Status Register. Bit 4 is the complement of the CTS signal. Bit 0 (DCTS) of the MODEM Status Register indicates whether the CTS input has changed state since the previous reading of the MODEM Status Register. CTS has no effect on the Transmitter.

Note: Whenever the CTS bit of the MODEM Status Register changes state, an interrupt is generated if the MODEM Status Interrupt is enabled.

D$_7$–D$_0$, Data Bus, Pins 1–8: This bus comprises eight TRI-STATE input/output lines. The bus provides bidirectional communications between the UART and the CPU. Data, control words, and status information are transferred via the D$_7$–D$_0$ Data Bus.

DCD, Data Carrier Detect, Pin 38: When low, indicates that the data carrier has been detected by the MODEM or data set. The DCD signal is a MODEM status input whose condition can be tested by the CPU reading bit 7 (DCD) of the MODEM Status Register. Bit 7 is the complement of the DCD signal. Bit 3 (DDCD) of the MODEM Status Register indicates whether the DCD input has changed state since the previous reading of the MODEM Status Register. DCD has no effect on the receiver.

Note: Whenever the DCD bit of the MODEM Status Register changes state, an interrupt is generated if the MODEM Status Interrupt is enabled.

DDIS, Driver Disable, Pin 23: This goes low whenever the CPU is reading data from the UART. It can disable or control the direction of a data bus transceiver between the CPU and the UART.

DSR, Data Set Ready, Pin 37: When low, this indicates that the MODEM or data set is ready to establish the communications link with the UART. The DSR signal is a MODEM status input whose condition can be tested by the CPU reading bit 5 (DSR) of the MODEM Status Register. Bit 5 is the complement of the DSR signal. Bit 1 (DDSR) of the MODEM Status Register indicates whether the DSR input has changed state since the previous reading of the MODEM Status Register.

Note: Whenever the DSR bit of the MODEM Status Register changes state, an interrupt is generated if the MODEM Status Interrupt is enabled.

DTR, Data Terminal Ready, Pin 33: When low, this informs the MODEM or data set that the UART is ready to establish a communications link. The DTR output signal can be set to an active low by programming bit 0 (DTR) of the MODEM Control Register to a high level. A Master Reset operation sets this signal to its inactive (high) state. Loop mode operation holds this signal in its inactive state.

INTR, Interrupt, Pin 30: This pin goes high whenever any one of the following interrupt types has an active high condition and is enabled via the IER: Receiver Error Flag; Received Data Available: timeout (FIFO Mode only); Transmitter Holding Register Empty; and MODEM Status. The INTR signal is reset low upon the appropriate interrupt service or a Master Reset operation.

MR, Master Reset, Pin 35: When this input is high, it clears all the registers (except the Receiver Buffer, Transmitter Holding, and Divisor Latches), and the control logic of the UART. The states of various output signals (SOUT, INTR, OUT 1, OUT 2, RTS, DTR) are affected by an active MR input (Refer to Table I.) This input is buffered with a TTL-compatible Schmitt Trigger with 0.5V typical hysteresis.

OUT 1, Output 1, Pin 34: This user-designated output can be set to an active low by programming bit 2 (OUT 1) of the MODEM Control Register to a high level. A Master Reset operation sets this signal to its inactive (high) state. Loop mode operation holds this signal in its inactive state. In the XMOS parts this will achieve TTL levels.

OUT 2, Output 2, Pin 31: This user-designated output that can be set to an active low by programming bit 3 (OUT 2) of the MODEM Control Register to a high level. A Master Reset operation sets this signal to its inactive (high) state. Loop mode operation holds this signal in its inactive state. In the XMOS parts this will achieve TTL levels.

6.0 Pin Descriptions (Continued)

RCLK, Receiver Clock, Pin 9: This input is the 16 × baud rate clock for the receiver section of the chip.

RD, $\overline{RD}$, Read, Pins 22 and 21: When RD is high or $\overline{RD}$ is low while the chip is selected, the CPU can read status information or data from the selected UART register.

Note: Only an active RD or $\overline{RD}$ input is required to transfer data from the UART during a read operation. Therefore, tie either the RD input permanently low or the $\overline{RD}$ input permanently high, when it is not used.

$\overline{RI}$, Ring Indicator, Pin 39: When low, this indicates that a telephone ringing signal has been received by the MODEM or data set. The $\overline{RI}$ signal is a MODEM status input whose condition can be tested by the CPU reading bit 6 (RI) of the MODEM Status Register. Bit 6 is the complement of the $\overline{RI}$ signal. Bit 2 (TERI) of the MODEM Status Register indicates whether the $\overline{RI}$ input **signal** has changed from a low to a high state since the previous reading of the MODEM Status Register.

Note: Whenever the RI **bit** of the MODEM Status Register changes from a high to a low state, an interrupt is generated if the MODEM Status Interrupt is enabled.

$\overline{RTS}$, Request to Send, Pin 32: When low, this informs the MODEM or data set that the UART is ready to exchange data. The $\overline{RTS}$ output signal can be set to an active low by programming bit 1 (RTS) of the MODEM Control Register. A Master Reset operation sets this signal to its inactive (high) state. Loop mode operation holds this signal in its inactive state.

SIN, Serial Input, Pin 10: Serial data input from the communications link (peripheral device, MODEM, or data set).

SOUT, Serial Output, Pin 11: Composite serial data output to the communications link (peripheral, MODEM or data set). The SOUT signal is set to the Marking (logic 1) state upon a Master Reset operation.

$\overline{TXRDY}$, $\overline{RXRDY}$, Pins 24, 29: Transmitter and Receiver DMA signalling is available through two pins (24 and 29). When operating in the FIFO mode, one of two types of DMA signalling per pin can be selected via FCR3. When operating as in the NS16450 Mode, only DMA mode 0 is allowed. Mode 0 supports single transfer DMA where a transfer is made between CPU bus cycles. Mode 1 supports multi-transfer DMA where multiple transfers are made continuously until the RCVR FIFO has been emptied or the XMIT FIFO has been filled.

RXRDY, Mode 0: When in the NS16450 Mode (FCR0 = 0) or in the FIFO Mode (FCR0 = 1, FCR3 = 0) and there is at least 1 character in the RCVR FIFO or RCVR holding register, the RXRDY pin (29) will be low active. Once it is activated the RXRDY pin will go inactive when there are no more characters in the FIFO or holding register.

RXRDY, Mode 1: In the FIFO Mode (FCR0 = 1) when the FCR3 = 1 and the trigger level or the timeout has been reached, the RXRDY pin will go low active. Once it is activated it will go inactive when there are no more characters in the FIFO or holding register.

TXRDY, Mode 0: In the NS16450 Mode (FCR0 = 0) or in the FIFO Mode (FCR0 = 1, FCR3 = 0) and there are no characters in the XMIT FIFO or XMIT holding register, the TXRDY pin (24) will be low active. Once it is activated the TXRDY pin will go inactive after the first character is loaded into the XMIT FIFO or holding register.

TXRDY, Mode 1: In the FIFO Mode (FCR0 = 1) when FCR3 = 1 and there are no characters in the XMIT FIFO, the $\overline{TXRDY}$ pin will go low active. This pin will become inactive when the XMIT FIFO is completely full.

V$_{CC}$, Pin 40: +5V supply.

V$_{SS}$, Pin 20: Ground (0V) reference.

WR, $\overline{WR}$, Write, Pins 19 and 18: When WR is high or $\overline{WR}$ is low while the chip is selected, the CPU can write control words or data into the selected UART register.

Note: Only an active WR or $\overline{WR}$ input is required to transfer data to the UART during a write operation. Therefore, tie either the WR input permanently low or the $\overline{WR}$ input permanently high, when it is not used.

XIN (External Crystal Input), Pin 16: This signal input is used in conjunction with XOUT to form a feedback circuit for the baud rate generator's oscillator. If a clock signal will be generated off-chip, then it should drive the baud rate generator through this pin.

XOUT (External Crystal Output), Pin 17: This signal output is used in conjunction with XIN to form a feedback circuit for the baud rate generator's oscillator. If the clock signal will be generated off-chip, then this pin is unused.

4

7.0 Connection Diagrams

Dual-In-Line Package

```
D0    1        40   VCC
D1    2        39   RI
D2    3        38   DCD
D3    4        37   DSR
D4    5        36   CTS
D5    6        35   MR
D6    7        34   OUT 1
D7    8        33   DTR
RCLK  9        32   RTS
SIN   10       31   OUT 2
SOUT  11       30   INTR
CS0   12       29   RXRDY
CS1   13       28   A0
CS2   14       27   A1
BAUDOUT 15     26   A2
XIN   16       25   ADS
XOUT  17       24   TXRDY
WR    18       23   DDIS
WR    19       22   RD
VSS   20       21   RD
```

TL/C/8652–17

Top View

Order Number NS16550AFN
See NS Package Number N40A

Chip Carrier Package

```
        D4 D3 D2 D1 D0 VCC RI DCD DSR CTS
                        NC
         6  5  4  3  2  1 44 43 42 41 40
D5    7                              39  MR
D6    8                              38  OUT1
D7    9                              37  DTR
RCLK  10                             36  RTS
SIN   11                             35  OUT2
NC    12                             34  NC
SOUT  13                             33  INTR
CS0   14                             32  RXRDY
CS1   15                             31  A0
CS2   16                             30  A1
BAUDOUT 17                           29  A2
        18 19 20 21 22 23 24 25 26 27 28
                        NC
        XIN XOUT WR WR VSS RD RD DDIS TXRDY ADS
```

TL/C/8652–18

Top View

Order Number NS16550AFV
See NS Package Number V44A

TABLE I. UART Reset Configuration

Register/Signal	Reset Control	Reset State
Interrupt Enable Register	Master Reset	**0000** 0000 (Note 1)
Interrupt Identification Register	Master Reset	00**0**0 0001
FIFO Control	Master Reset	00**0**0 0000
Line Control Register	Master Reset	0000 0000
MODEM Control Register	Master Reset	**0000** 0000
Line Status Register	Master Reset	0110 0000
MODEM Status Register	Master Reset	XXXX 0000 (Note 2)
SOUT	Master Reset	High
INTR (RCVR Errs)	Read LSR/MR	Low
INTR (RCVR Data Ready)	Read RBR/MR	Low
INTR (THRE)	Read IIR/Write THR/MR	Low
INTR (Modem Status Changes)	Read MSR/MR	Low
OUT 2	Master Reset	High
RTS	Master Reset	High
DTR	Master Reset	High
OUT 1	Master Reset	High
RCVR FIFO	MR/FCR1•FCR0/ΔFCR0	All Bits Low
XMIT FIFO	MR/FCR1•FCR0/ΔFCR0	All Bits Low

Note 1: Boldface bits are permanently low.

Note 2: Bits 7–4 are driven by the input signals.

					Register Address							
Bit No.	0 DLAB=0	0 DLAB=0	1 DLAB=0	2	2	3	4	5	6	7	0 DLAB=1	1 DLAB=1
	Receiver Buffer Register (Read Only)	Transmitter Holding Register (Write Only)	Interrupt Enable Register	Interrupt Ident. Register (Read Only)	FIFO Control Register (Write Only)	Line Control Register	MODEM Control Register	Line Status Register	MODEM Status Register	Scratch Register	Divisor Latch (LS)	Divisor Latch (MS)
	RBR	THR	IER	IIR	FCR	LCR	MCR	LSR	MSR	SCR	DLL	DLM
0	Data Bit 0 (Note 1)	Data Bit 0	Enable Received Data Available Interrupt (ERBFI)	"0" if Interrupt Pending	FIFO Enable	Word Length Select Bit 0 (WLS0)	Data Terminal Ready (DTR)	Data Ready (DR)	Delta Clear to Send (DCTS)	Bit 0	Bit 0	Bit 8
1	Data Bit 1	Data Bit 1	Enable Transmitter Holding Register Empty Interrupt (ETBEI)	Interrupt ID Bit (0)	RCVR FIFO Reset	Word Length Select Bit 1 (WLS1)	Request to Send (RTS)	Overrun Error (OE)	Delta Data Set Ready (DDSR)	Bit 1	Bit 1	Bit 9
2	Data Bit 2	Data Bit 2	Enable Receiver Line Status Interrupt (ELSI)	Interrupt ID Bit (1)	XMIT FIFO Reset	Number of Stop Bits (STB)	Out 1	Parity Error (PE)	Trailing Edge Ring Indicator (TERI)	Bit 2	Bit 2	Bit 10
3	Data Bit 3	Data Bit 3	Enable MODEM Status Interrupt (EDSSI)	Interrupt ID Bit (2) (Note 2)	DMA Mode Select	Parity Enable (PEN)	Out 2	Framing Error (FE)	Delta Data Carrier Detect (DDCD)	Bit 3	Bit 3	Bit 11
4	Data Bit 4	Data Bit 4	0	0	Reserved	Even Parity Select (EPS)	Loop	Break Interrupt (BI)	Clear to Send (CTS)	Bit 4	Bit 4	Bit 12
5	Data Bit 5	Data Bit 5	0	0	Reserved	Stick Parity	0	Transmitter Holding Register (THRE)	Data Set Ready (DSR)	Bit 5	Bit 5	Bit 13
6	Data Bit 6	Data Bit 6	0	FIFOs Enabled (Note 2)	RCVR Trigger (LSB)	Set Break	0	Transmitter Empty (TEMT)	Ring Indicator (RI)	Bit 6	Bit 6	Bit 14
7	Data Bit 7	Data Bit 7	0	FIFOs Enabled (Note 2)	RCVR Trigger (MSB)	Divisor Latch Access Bit (DLAB)	0	Error in RCVR FIFO (Note 2)	Data Carrier Detect (DCD)	Bit 7	Bit 7	Bit 15

TABLE II. Summary of Registers

Note 1: Bit 0 is the least significant bit. It is the first bit serially transmitted or received.

Note 2: These bits are always 0 in the NS16450 Mode.

8.0 Registers

The system programmer may access any of the UART registers summarized in Table II via the CPU. These registers control UART operations including transmission and reception of data. Each register bit in Table II has its name and reset state shown.

8.1 LINE CONTROL REGISTER

The system programmer specifies the format of the asynchronous data communications exchange and set the Divisor Latch Access bit via the Line Control Register (LCR). The programmer can also read the contents of the Line Control Register. The read capability simplifies system programming and eliminates the need for separate storage in system memory of the line characteristics. Table II shows the contents of the LCR. Details on each bit follow:

Bits 0 and 1: These two bits specify the number of bits in each transmitted or received serial character. The encoding of bits 0 and 1 is as follows:

Bit 1	Bit 0	Character Length
0	0	5 Bits
0	1	6 Bits
1	0	7 Bits
1	1	8 Bits

Bit 2: This bit specifies the number of Stop bits transmitted and received in each serial character. If bit 2 is a logic 0, one Stop bit is generated in the transmitted data. If bit 2 is a logic 1 when a 5-bit word length is selected via bits 0 and 1, one and a half Stop bits are generated. If bit 2 is a logic 1 when either a 6-, 7-, or 8-bit word length is selected, two Stop bits are generated. The Receiver checks the first Stop-bit only, regardless of the number of Stop bits selected.

Bit 3: This bit is the Parity Enable bit. When bit 3 is a logic 1, a Parity bit is generated (transmit data) or checked (receive data) between the last data word bit and Stop bit of the serial data. (The Parity bit is used to produce an even or odd number of 1s when the data word bits and the Parity bit are summed.)

Bit 4: This bit is the Even Parity Select bit. When bit 3 is a logic 1 and bit 4 is a logic 0, an odd number of logic 1s is transmitted or checked in the data word bits and Parity bit. When bit 3 is a logic 1 and bit 4 is a logic 1, an even number of logic 1s is transmitted or checked.

Bit 5: This bit is the Stick Parity bit. When bits 3, 4 and 5 are logic 1 the Parity bit is transmitted and checked as a logic 0. If bits 3 and 5 are 1 and bit 4 is a logic 0 then the Parity bit is transmitted and checked as a logic 1. If bit 5 is a logic 0 Stick Parity is disabled.

Bit 6: This bit is the Break Control bit. It causes a break condition to be transmitted to the receiving UART. When it is set to a logic 1, the serial output (SOUT) is forced to the Spacing (logic 0) state. The break is disabled by setting bit 6 to a logic 0. The Break Control bit acts only on SOUT and has no effect on the transmitter logic.

Note: This feature enables the CPU to alert a terminal in a computer communications system. If the following sequence is followed, no erroneous or extraneous characters will be transmitted because of the break.

1. Load an all 0s, pad character, in response to THRE.

2. Set break after the next THRE.

3. Wait for the transmitter to be idle, (TEMT = 1), and clear break when normal transmission has to be restored.

During the break, the Transmitter can be used as a character timer to accurately establish the break duration.

Bit 7: This bit is the Divisor Latch Access Bit (DLAB). It must be set high (logic 1) to access the Divisor Latches of the Baud Generator during a Read or Write operation. It must be set low (logic 0) to access the Receiver Buffer, the Transmitter Holding Register, or the Interrupt Enable Register.

8.2 TYPICAL CLOCK CIRCUITS

TL/C/8652-19

TL/C/8652-20

Typical Crystal Oscillator Network (Note)

CRYSTAL	R_P	R_{X2}	C_1	C_2
3.1 MHz	1 MΩ	1.5k	10-30 pF	40-60 pF
1.8 MHz	1 MΩ	1.5k	10-30 pF	40-60 pF

Note: These R and C values are approximate and may vary 2x depending on the crystal characteristics. All crystal circuits should be designed specifically for the system.

TABLE III. Baud Rates Using 1.8432 MHz Crystal

Desired Baud Rate	Decimal Divisor Used to Generate 16 x Clock	Percent Error Difference Between Desired and Actual
50	2304	—
75	1536	—
110	1047	0.026
134.5	857	0.058
150	768	—
300	384	—
600	192	—
1200	96	—
1800	64	—
2000	58	0.69
2400	48	—
3600	32	—
4800	24	—
7200	16	—
9600	12	—
19200	6	—
38400	3	—
56000	2	2.86

8.0 Registers (Continued)

8.3 PROGRAMMABLE BAUD GENERATOR

The UART contains a programmable Baud Generator that is capable of taking any clock input from DC to 8.0 MHz and dividing it by any divisor from 2 to $2^{16}-1$. 4 MHz is the highest input clock frequency recommended when the divisor = 1. The output frequency of the Baud Generator is 16 × the Baud [divisor # = (frequency input) ÷ (baud rate × 16)]. Two 8-bit latches store the divisor in a 16-bit binary format. These Divisor Latches must be loaded during initialization to ensure proper operation of the Baud Generator. Upon loading either of the Divisor Latches, a 16-bit Baud counter is immediately loaded.

Tables III, IV and V provide decimal divisors to use with crystal frequencies of 1.8432 MHz, 3.072 MHz and 8 MHz, respectively. For baud rates of 38400 and below, the error obtained is minimal. The accuracy of the desired baud rate is dependent on the crystal frequency chosen. Using a divisor of zero is **not** recommended.

8.4 LINE STATUS REGISTER

This register provides status information to the CPU concerning the data transfer. Table II shows the contents of the Line Status Register. Details on each bit follow.

Bit 0: This bit is the receiver Data Ready (DR) indicator. Bit 0 is set to a logic 1 whenever a complete incoming character has been received and transferred into the Receiver Buffer Register or the FIFO. Bit 0 is reset to a logic 0 by reading all of the data in the Receiver Buffer Register or the FIFO.

Bit 1: This bit is the Overrun Error (OE) indicator. Bit 1 indicates that data in the Receiver Buffer Register was not read by the CPU before the next character was transferred into the Receiver Buffer Register, thereby destroying the previous character. The OE indicator is set to a logic 1 upon detection of an overrun condition and reset whenever the CPU reads the contents of the Line Status Register. If the FIFO mode data continues to fill the FIFO beyond the trigger level, an overrun error will occur only after the FIFO is full and the next character has been completely received in the shift register. OE is indicated to the CPU as soon as it happens. The character in the shift register is overwritten, but it is not transferred to the FIFO.

Bit 2: This bit is the Parity Error (PE) indicator. Bit 2 indicates that the received data character does not have the correct even or odd parity, as selected by the even-parity-select bit. The PE bit is set to a logic 1 upon detection of a parity error and is reset to a logic 0 whenever the CPU reads the contents of the Line Status Register. In the FIFO mode this error is associated with the particular character in the FIFO it applies to. This error is revealed to the CPU when its associated character is at the top of the FIFO.

Bit 3: This bit is the Framing Error (FE) indicator. Bit 3 indicates that the received character did not have a valid Stop bit. Bit 3 is set to a logic 1 whenever the Stop bit following the last data bit or parity bit is detected as a logic 0 bit (Spacing level). The FE indicator is reset whenever the CPU reads the contents of the Line Status Register. In the FIFO mode this error is associated with the particular character in the FIFO it applies to. This error is revealed to the CPU when its associated character is at the top of the FIFO. The UART will try to resynchronize after a framing error. To do this it assumes that the framing error was due to the next start bit, so it samples this "start" bit twice and then takes in the "data".

Bit 4: This bit is the Break Interrupt (BI) indicator. Bit 4 is set to a logic 1 whenever the received data input is held in the Spacing (logic 0) state for longer than a full word transmission time (that is, the total time of Start bit + data bits + Parity + Stop bits). The BI indicator is reset whenever the CPU reads the contents of the Line Status Register. In the FIFO mode this error is associated with the particular character in the FIFO it applies to. This error is revealed to the CPU when its associated character is at the top of the FIFO. When break occurs only one zero character is loaded into the FIFO. The next character transfer is enabled after SIN goes to the marking state and receives the next valid start bit.

Note: Bits 1 through 4 are the error conditions that produce a Receiver Line Status interrupt whenever any of the corresponding conditions are detected and the interrupt is enabled.

TABLE IV. Baud Rates Using 3.072 MHz Crystal

Desired Baud Rate	Decimal Divisor Used to Generate 16 x Clock	Percent Error Difference Between Desired and Actual
50	3840	—
75	2560	—
110	1745	0.026
134.5	1428	0.034
150	1280	—
300	640	—
600	320	—
1200	160	—
1800	107	0.312
2000	96	—
2400	80	—
3600	53	0.628
4800	40	—
7200	27	1.23
9600	20	—
19200	10	—
38400	5	—

TABLE V. Baud Rates Using 8 MHz Crystal

Desired Baud Rate	Decimal Divisor Used to Generate 16 x Clock	Percent Error Difference Between Desired and Actual
50	10000	—
75	6667	0.005
110	4545	0.010
134.5	3717	0.013
150	3333	0.010
300	1667	0.020
600	833	0.040
1200	417	0.080
1800	277	0.080
2000	250	—
2400	208	0.160
3600	139	0.080
4800	104	0.160
7200	69	0.644
9600	52	0.160
19200	26	0.160
38400	13	0.160
56000	9	0.790
128000	4	2.344
256000	2	2.344

4

8.0 Registers (Continued)

TABLE VI. Interrupt Control Functions

FIFO Mode Only	Interrupt Identification Register			Interrupt Set and Reset Functions			
Bit 3	Bit 2	Bit 1	Bit 0	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Control
0	0	0	1	—	None	None	—
0	1	1	0	Highest	Receiver Line Status	Overrun Error or Parity Error or Framing Error or Break Interrupt	Reading the Line Status Register
0	1	0	0	Second	Received Data Available	Receiver Data Available or Trigger Level Reached	Reading the Receiver Buffer Register or the FIFO Drops Below the Trigger Level
1	1	0	0	Second	Character Timeout Indication	No Characters Have Been Removed From or Input to the RCVR FIFO During the Last 4 Char. Times and There Is at Least 1 Char. in It During This Time	Reading the Receiver Buffer Register
0	0	1	0	Third	Transmitter Holding Register Empty	Transmitter Holding Register Empty	Reading the IIR Register (if source of interrupt) or Writing into the Transmitter Holding Register
0	0	0	0	Fourth	MODEM Status	Clear to Send or Data Set Ready or Ring Indicator or Data Carrier Detect	Reading the MODEM Status Register

Bit 5: This bit is the Transmitter Holding Register Empty (THRE) indicator. Bit 5 indicates that the UART is ready to accept a new character for transmission. In addition, this bit causes the UART to issue an interrupt to the CPU when the Transmit Holding Register Empty Interrupt enable is set high. The THRE bit is set to a logic 1 when a character is transferred from the Transmitter Holding Register into the Transmitter Shift Register. The bit is reset to logic 0 concurrently with the loading of the Transmitter Holding Register by the CPU. In the FIFO mode this bit is set when the XMIT FIFO is empty; it is cleared when at least 1 byte is written to the XMIT FIFO.

Bit 6: This bit is the Transmitter Empty (TEMT) indicator. Bit 6 is set to a logic 1 whenever the Transmitter Holding Register (THR) and the Transmitter Shift Register (TSR) are both empty. It is reset to a logic 0 whenever either the THR or TSR contains a data character. In the FIFO mode this bit is set to one whenever the transmitter FIFO and shift register are both empty.

Bit 7: In the NS16450 Mode this is a 0. In the FIFO mode LSR7 is set when there is at least one parity error, framing error or break indication in the FIFO. LSR7 is cleared when the CPU reads the LSR, if there are no subsequent errors in the FIFO.

Note: The Line Status Register is intended for read operations only. Writing to this register is not recommended as this operation is only used for factory testing. In the FIFO mode the software must load a data byte in the Rx FIFO via Loopback Mode in order to write to LSR2–LSR4. LSR0 and LSR7 can't be written to in FIFO mode.

8.5 FIFO CONTROL REGISTER

This is a write only register at the same location as the IIR (the IIR is a read only register). This register is used to enable the FIFOs, clear the FIFOs, set the RCVR FIFO trigger level, and select the type of DMA signalling.

Bit 0: Writing a 1 to FCR0 enables both the XMIT and RCVR FIFOs. Resetting FCR0 will clear all bytes in both FIFOs.

When changing from FIFO Mode to NS16450 Mode and vice versa, data is automatically cleared from the FIFOs. This bit must be a 1 when other FCR bits are written to or they will not be programmed.

Bit 1: Writing a 1 to FCR1 clears all bytes in the RCVR FIFO and resets its counter logic to 0. The shift register is not cleared. The 1 that is written to this bit position is self-clearing.

Bit 2: Writing a 1 to FCR2 clears all bytes in the XMIT FIFO and resets its counter logic to 0. The shift register is not cleared. The 1 that is written to this bit position is self-clearing.

Bit 3: Setting FCR3 to a 1 will cause the RXRDY and TXRDY pins to change from mode 0 to mode 1 if FCR0 = 1 (see description of RXRDY and TXRDY pins).

Bit 4, 5: FCR4 to FCR5 are reserved for future use.

Bit 6, 7: FCR6 and FCR7 are used to set the trigger level for the RCVR FIFO interrupt.

7	6	RCVR FIFO Trigger Level (Bytes)
0	0	01
0	1	04
1	0	08
1	1	14

8.6 INTERRUPT IDENTIFICATION REGISTER

In order to provide minimum software overhead during data character transfers, the UART prioritizes interrupts into four levels and records these in the interrupt Identification Register. The four levels of interrupt conditions in order of priority are Receiver Line Status; Received Data Ready; Transmitter Holding Register Empty; and MODEM Status.

8.0 Registers (Continued)

When the CPU accesses the IIR, the UART freezes all interrupts and indicates the highest priority pending interrupt to the CPU. While this CPU access is occurring, the UART records new interrupts, but does not change its current indication until the access is complete. Table II shows the contents of the IIR. Details on each bit follow:

Bit 0: This bit can be used in a prioritized interrupt environment to indicate whether an interrupt is pending. When bit 0 is a logic 0, an interrupt is pending and the IIR contents may be used as a pointer to the appropriate interrupt service routine. When bit 0 is a logic 1, no interrupt is pending.

Bits 1 and 2: These two bits of the IIR are used to identify the highest priority interrupt pending as indicated in Table VI.

Bit 3: In the NS16450 Mode this bit is 0. In the FIFO mode this bit is set along with bit 2 when a timeout interrupt is pending.

Bits 4 and 5: These two bits of the IIR are always logic 0.

Bits 6 and 7: These two bits are set when FCR0 = 1.

8.7 INTERRUPT ENABLE REGISTER

This register enables the five types of UART interrupts. Each interrupt can individually activate the interrupt (INTR) output signal. It is possible to totally disable the interrupt system by resetting bits 0 through 3 of the Interrupt Enable Register (IER). Similarly, setting bits of the IER register to a logic 1, enables the selected interrupt(s). Disabling an interrupt prevents it from being indicated as active in the IIR and from activating the INTR output signal. All other system functions operate in their normal manner, including the setting of the Line Status and MODEM Status Registers. Table II shows the contents of the IER. Details on each bit follow.

Bit 0: This bit enables the Received Data Available Interrupt (and timeout interrupts in the FIFO mode) when set to logic 1.

Bit 1: This bit enables the Transmitter Holding Register Empty Interrupt when set to logic 1.

Bit 2: This bit enables the Receiver Line Status Interrupt when set to logic 1.

Bit 3: This bit enables the MODEM Status Interrupt when set to logic 1.

Bits 4 through 7: These four bits are always logic 0.

8.8 MODEM CONTROL REGISTER

This register controls the interface with the MODEM or data set (or a peripheral device emulating a MODEM). The contents of the MODEM Control Register are indicated in Table II and are described below.

Bit 0: This bit controls the Data Terminal Ready ($\overline{\text{DTR}}$) output. When bit 0 is set to a logic 1, the $\overline{\text{DTR}}$ output is forced to a logic 0. When bit 0 is reset to a logic 0, the $\overline{\text{DTR}}$ output is forced to a logic 1.

Note: The $\overline{\text{DTR}}$ output of the UART may be applied to an EIA inverting line driver (such as the DS1488) to obtain the proper polarity input at the succeeding MODEM or data set.

Bit 1: This bit controls the Request to Send ($\overline{\text{RTS}}$) output. Bit 1 affects the $\overline{\text{RTS}}$ output in a manner identical to that described above for bit 0.

Bit 2: This bit controls the Output 1 ($\overline{\text{OUT 1}}$) signal, which is an auxiliary user-designated output. Bit 2 affects the $\overline{\text{OUT 1}}$ output in a manner identical to that described above for bit 0.

Bit 3: This bit controls the Output 2 ($\overline{\text{OUT 2}}$) signal, which is an auxiliary user-designated output. Bit 3 affects the $\overline{\text{OUT 2}}$ output in a manner identical to that described above for bit 0.

Bit 4: This bit provides a local loopback feature for diagnostic testing of the UART. When bit 4 is set to logic 1, the following occur: the transmitter Serial Output (SOUT) is set to the Marking (logic 1) state; the receiver Serial Input (SIN) is disconnected; the output of the Transmitter Shift Register is "looped back" into the Receiver Shift Register input; the four MODEM Control inputs ($\overline{\text{DSR}}$, $\overline{\text{CTS}}$, $\overline{\text{RI}}$, and $\overline{\text{DCD}}$) are disconnected; and the four MODEM Control outputs ($\overline{\text{DTR}}$, $\overline{\text{RTS}}$, $\overline{\text{OUT 1}}$, and $\overline{\text{OUT 2}}$) are internally connected to the four MODEM Control inputs, and the MODEM Control output pins are forced to their inactive state (high). In the diagnostic mode, data that is transmitted is immediately received. This feature allows the processor to verify the transmit-and received-data paths of the UART.

In the diagnostic mode, the receiver and transmitter interrupts are fully operational. Their sources are external to the part. The MODEM Control Interrupts are also operational, but the interrupts' sources are now the lower four bits of the MODEM Control Register instead of the four MODEM Control inputs. The interrupts are still controlled by the Interrupt Enable Register.

Bits 5 through 7: These bits are permanently set to logic 0.

8.9 MODEM STATUS REGISTER

This register provides the current state of the control lines from the MODEM (or peripheral device) to the CPU. In addition to this current-state information, four bits of the MODEM Status Register provide change information. These bits are set to a logic 1 whenever a control input from the MODEM changes state. They are reset to logic 0 whenever the CPU reads the MODEM Status Register.

The contents of the MODEM Status Register are indicated in Table II and described below.

Bit 0: This bit is the Delta Clear to Send (DCTS) indicator. Bit 0 indicates that the $\overline{\text{CTS}}$ input to the chip has changed state since the last time it was read by the CPU.

Bit 1: This bit is the Delta Data Set Ready (DDSR) indicator. Bit 1 indicates that the $\overline{\text{DSR}}$ input to the chip has changed state since the last time it was read by the CPU.

Bit 2: This bit is the Trailing Edge of Ring Indicator (TERI) detector. Bit 2 indicates that the $\overline{\text{RI}}$ input to the chip has changed from a low to a high state.

Bit 3: This bit is the Delta Data Carrier Detect (DDCD) indicator. Bit 3 indicates that the $\overline{\text{DCD}}$ input to the chip has changed state.

Note: Whenever bit 0, 1, 2, or 3 is set to logic 1, a MODEM Status Interrupt is generated.

Bit 4: This bit is the complement of the Clear to Send ($\overline{\text{CTS}}$) input. If bit 4 (loop) of the MCR is set to a 1, this bit is equivalent to RTS in the MCR.

Bit 5: This bit is the complement of the Data Set Ready ($\overline{\text{DSR}}$) input. If bit 4 of the MCR is set to a 1, this bit is equivalent to DTR in the MCR.

4

8.0 Registers (Continued)

Bit 6: This bit is the complement of the Ring Indicator ($\overline{RI}$) input. If bit 4 of the MCR is set to a 1, this bit is equivalent to OUT 1 in the MCR.

Bit 7: This bit is the complement of the Data Carrier Detect ($\overline{DCD}$) input. If bit 4 of the MCR is set to a 1, this bit is equivalent to OUT 2 in the MCR.

8.10 SCRATCHPAD REGISTER

This 8-bit Read/Write Register does not control the UART in anyway. It is intended as a scratchpad register to be used by the programmer to hold data temporarily.

8.11 FIFO INTERRUPT MODE OPERATION

When the RCVR FIFO and receiver interrupts are enabled (FCR0 = 1, IER0 = 1) RCVR interrupts will occur as follows:

A. The receive data available interrupt will be issued to the CPU when the FIFO has reached its programmed trigger level; it will be cleared as soon as the FIFO drops below its programmed trigger level.

B. The IIR receive data available indication also occurs when the FIFO trigger level is reached, and like the interrupt it is cleared when the FIFO drops below the trigger level.

C. The receiver line status interrupt (IIR = 06), as before, has higher priority than the received data available (IIR = 04) interrupt.

D. The data ready bit (LSR0) is set as soon as a character is transferred from the shift register to the RCVR FIFO. It is reset when the FIFO is empty.

When RCVR FIFO and receiver interrupts are enabled, RCVR FIFO timeout interrupts will occur as follows:

A. A FIFO timeout interrupt will occur, if the following conditions exist:

— at least one character is in the FIFO

— the most recent serial character received was longer than 4 continuous character times ago (if 2 stop bits are programmed the second one is included in this time delay).

— the most recent CPU read of the FIFO was longer than 4 continuous character times ago.

The maximum time between a received character and a timeout interrupt will be 160 ms at 300 baud with a 12-bit receive character (i.e., 1 Start, 8 Data, 1 Parity and 2 Stop Bits).

B. Character times are calculated by using the RCLK input for a clock signal (this makes the delay proportional to the baudrate).

C. When a timeout interrupt has occurred it is cleared and the timer reset when the CPU reads one character from the RCVR FIFO.

D. When a timeout interrupt has not occurred the timeout timer is reset after a new character is received or after the CPU reads the RCVR FIFO.

When the XMIT FIFO and transmitter interrupts are enabled (FCR0 = 1, IER1 = 1), XMIT interrupts will occur as follows:

A. The transmitter holding register interrupt (02) occurs when the XMIT FIFO is empty; it is cleared as soon as the transmitter holding register is written to (1 to 16 characters may be written to the XMIT FIFO while servicing this interrupt) or the IIR is read.

B. The transmitter FIFO empty indications will be delayed 1 character time minus the last stop bit time whenever the following occurs: THRE = 1 and there have not been at least two bytes at the same time in the transmit FIFO, since the last THRE = 1. The first transmitter interrupt after changing FCR0 will be immediate, if it is enabled.

Character timeout and RCVR FIFO trigger level interrupts have the same priority as the current received data available interrupt; XMIT FIFO empty has the same priority as the current transmitter holding register empty interrupt.

8.12 FIFO POLLED MODE OPERATION

With FCR0 = 1 resetting IER0, IER1, IER2, IER3 or all to zero puts the UART in the FIFO Polled Mode of operation. Since the RCVR and XMITTER are controlled separately either one or both can be in the polled mode of operation.

In this mode the user's program will check RCVR and XMITTER status via the LSR. As stated previously:

LSR0 will be set as long as there is one byte in the RCVR FIFO.

LSR1 to LSR4 will specify which error(s) has occurred. Character error status is handled the same way as when in the interrupt mode, the IIR is not affected since IER2 = 0.

LSR5 will indicate when the XMIT FIFO is empty.

LSR6 will indicate that both the XMIT FIFO and shift register are empty.

LSR7 will indicate whether there are any errors in the RCVR FIFO.

There is no trigger level reached or timeout condition indicated in the FIFO Polled Mode, however, the RCVR and XMIT FIFOs are still fully capable of holding characters.

This shows the basic connections of an NS16550AF to an 8088 CPU
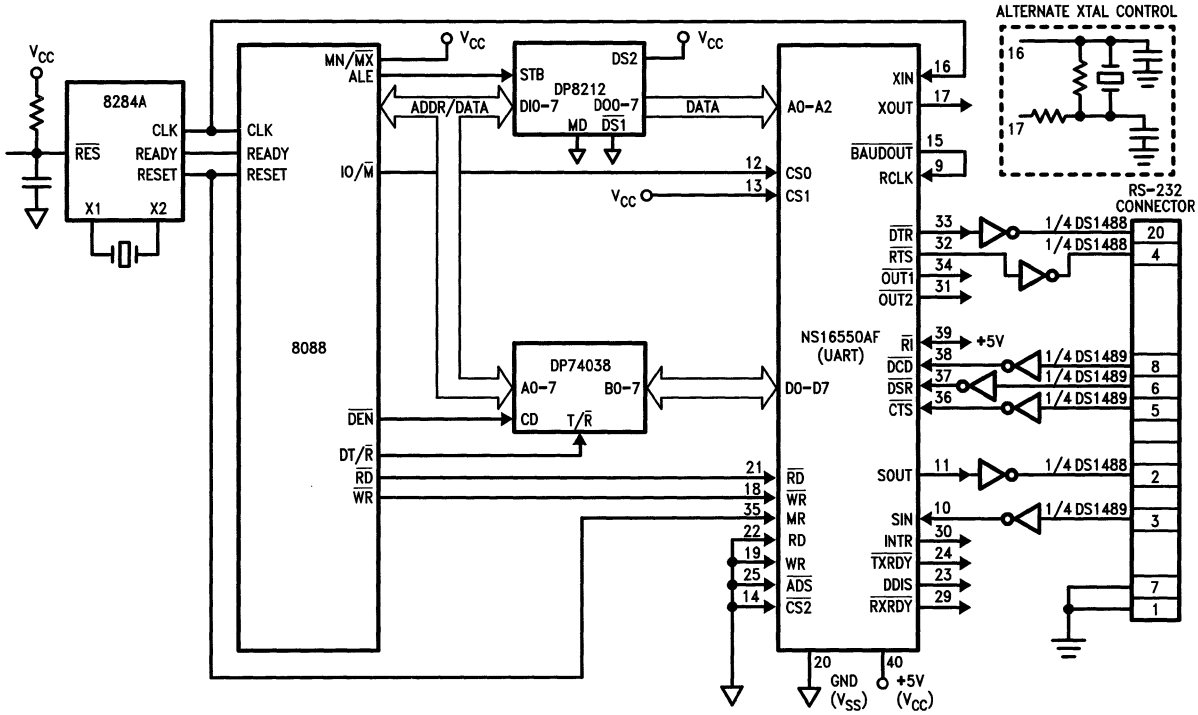
TL/C/8652–22

9.0 Typical Applications (Continued)

**Typical Interface for a
High-Capacity Data Bus**

TL/C/8652–23

**Typical Supply Current vs.
Temperature, Normalized**
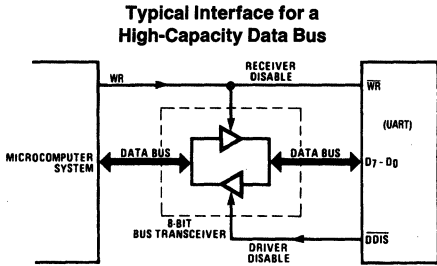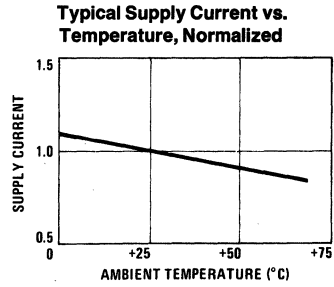
TL/C/8652–24

10.0 Ordering Information

NS16550AFXX

/A⁺ = A⁺ RELIABILITY SCREENING

N = PLASTIC PACKAGE
V = PLASTIC LEADED CHIP CARRIER (PCC)
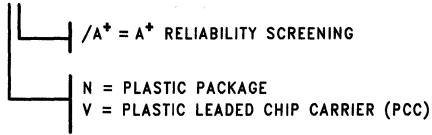
TL/C/8652–25

The NS16550A: UART Design and Application Considerations

BACKGROUND

UARTs like other system components have evolved for many years to become faster, more integrated and less expensive. The rise in popularity of the personal computer with its focus and competition primarily centered on an architecture introduced by IBM®, has driven both UART performance and software compatibility issues. As transmission rates have increased, the amount of time the CPU has for other tasks while handling an active serial channel has been sharply reduced. One byte of data received at 1200 baud (8.3 ms) is received in $\frac{1}{16}$th the time at 19.2 kbaud (520 μs). Software compatibility among the PC-based UARTs is critical due to the thousands of existing programs which use the serial channel and the new programs continually being offered.

Higher baud rates and compatibility requirements influence new UART designs. These two constraints result in UARTs that are capable of higher data rates, increasingly independent of CPU intervention and providing more autonomous features, while maintaining software compatibility. These development paths have been brought together in a new UART from National Semiconductor designated the NS16550A.

The NS16550A has all of the registers of its two predecessor parts (INS8250 and NS16450), so it can run all existing IBM PC, XT, AT, RT and compatible serial port software. In addition, it has a programmable mode which incorporates new high-performance features. Of course, all of these advanced features are useful in any asynchronous serial communications application regardless of the host architecture.

The reader is assumed to be familiar with the standard features of the NS16450, so this paper will concentrate mainly on the new features of the NS16550A. If the reader is unfamiliar with these UARTs it is advisable to start by reading their data sheets.

The first section reviews some of the design considerations and the operation of the NS16550A advanced features. The second section shows an NS16550A initialization routine written in 80286 assembly code with an explanation of the routine. The third section gives a detailed example of communications drivers written to interface two NS16550As on individual boards. These drivers are written for use with National Semiconductor's DB32032 evaluation boards, but can be ported to any NS32032-based system containing an NS32202 (ICU).

1.0 Design Considerations and Operation of the New UART Features

In order to optimize CPU/UART data transactions, the UART design takes into consideration the following constraints:

1. The CPU is usually much faster than the UART at transferring data. A high speed CPU could transfer a byte of data to/from the UART in a minimum of 280 ns. The UART would take over 1800 times longer to transmit/receive this data serially if it were operating at 19.2 kbaud.

2. There is a finite amount of wasted CPU time due to software overhead when stopping its current task to service the UART (context switching overhead).

3. The CPU may be required to complete a certain portion of its current task in a multitasking system before servicing the UART. This delay is the CPU latency time associated with servicing the interrupt. The amount of time that the receiver can accept continuous data after it requests service from the CPU constrains CPU latency time.

The design constraints listed above are met by adding two FIFOs and specialized transmitter/receiver support circuitry to the existing NS16450 design. The FIFOs are 16 bytes deep—one holds data for the transmitter, the other for the receiver (see *Figure 1*). Similarity between the FIFOs stops with their size, as each has been customized for special
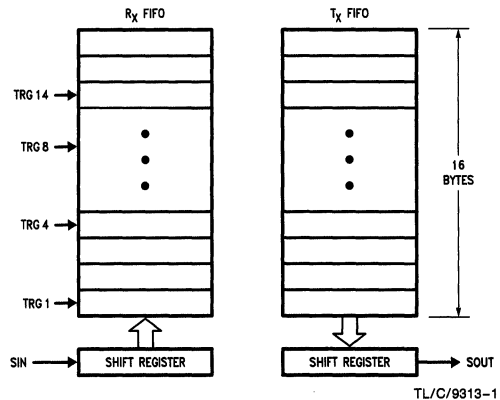
FIGURE 1. Rx and Tx FIFOs

TL/C/9313–1

transmitter or receiver functions. Each has support circuitry to minimize software overhead when handling interrupts. The NS16550A **receiver** optimizes the CPU/UART data transaction via the following features:

1. The depth of the Receiver (Rx) FIFO ensures that as many as 16 characters will be ready to transfer when the CPU services the Rx interrupt. Therefore, the CPU transfer rate is effectively buffered from the serial data rate.

2. The program can select the number of bytes required in the Rx FIFO (1, 4, 8 or 14) before the UART issues an interrupt. This allows the software to modify the interrupt trigger levels depending on its current task or loading. It also ensures that the CPU doesn't continually waste time switching context for only a few characters.

4

3. The Rx FIFO will hold 16 bytes regardless of which trigger level the CPU selects. This makes allowances for a variety of CPU latency times, as the FIFO continues to fill after the interrupt is issued.

The NS16550A **transmitter** optimizes the CPU/UART data transaction via the following features:

1. The depth of the Transmitter (Tx) FIFO ensures that as many as 16 characters can be transferred when the CPU services the Tx interrupt. Once again, this effectively buffers the CPU transfer rate from the serial data rate.

2. The Transmitter (Tx) FIFO is similar in structure to FIFOs the user may have previously set up in RAM. The Tx depth allows the CPU to load 16 characters each time it switches context to the service routine. This reduces the impact of the CPU time lost in context switching.

3. Since a time lag in servicing an asynchronous transmitter usually has no penalty, CPU latency time is of no concern to transmitter operation.

TX AND RX FIFO OPERATION

The Tx portion of the UART transmits data through SOUT as soon as the CPU loads a byte into the Tx FIFO. The UART will prevent loads to the Tx FIFO if it **currently** holds 16 characters. Loading to the Tx FIFO will again be enabled as soon as the next character is transferred to the Tx shift register. These capabilities account for the largely autonomous operation of the Tx.

The UART starts the above operations typically with a Tx interrupt. The NS16550A issues a Tx interrupt whenever the Tx FIFO is empty and the Tx interrupt is enabled, except in the following instance. Assume that the Tx FIFO is empty and the CPU starts to load it. When the first byte enters the FIFO, the Tx FIFO empty interrupt will transition from active to inactive. Depending on the execution speed of the service routine software, the UART may be able to transfer this byte from the FIFO to the shift register before the CPU loads another byte. If this happens, the Tx FIFO will be empty again and typically the UART's interrupt line would transition to the active state. This could cause a system with an interrupt control unit to record a Tx FIFO empty condition, even though the CPU is currently servicing that interrupt. Therefore, after the first byte has been loaded into the FIFO the UART will wait one serial character transmission time before issuing a new Tx FIFO empty interrupt.

This one character Tx interrupt delay will remain active until at least two bytes have been loaded into the FIFO, concurrently. When the Tx FIFO empties after this condition, the Tx interrupt will be activated without a one character delay.

Rx support functions and operation are quite different from those described for the transmitter. The Rx FIFO receives data until the number of bytes in the FIFO equals the selected interrupt trigger level. At that time if Rx interrupts are enabled, the UART will issue an interrupt to the CPU. The Rx FIFO will continue to store bytes until it holds 16 of them. It will not accept any more data when it is full. Any more

data entering the Rx shift register will set the Overrun Error flag. Normally, the FIFO depth and the programmable trigger levels will give the CPU ample time to empty the Rx FIFO before an overrun occurs.

One side-effect of having a Rx FIFO is that the selected interrupt trigger level may be above the data level in the FIFO. This could occur when data at the end of the block contains fewer bytes than the trigger level. No interrupt would be issued to the CPU and the data would remain in the UART. To prevent the software from having to check for this situation the NS16550A incorporates a timeout interrupt.

The timeout interrupt is activated when there is at least one byte in the Rx FIFO, and neither the CPU nor the Rx shift register has accessed the Rx FIFO within 4 character times of the last byte. The timeout interrupt is cleared or reset when the CPU reads the Rx FIFO or another character enters it.

These FIFO related features allow optimization of CPU/UART transactions and are especially useful given the higher baud rate capability (256 kbaud). However, in order to eliminate most CPU interactions, the UART provides DMA request signals. Two DMA modes are supported: single-transfer and multi-transfer. These modes allow the UART to interface to higher performance DMA units, which can interleave their transfers between CPU cycles or execute multiple byte transfers.

In single-transfer mode the receiver DMA request signal (Rx RDY) goes active whenever there is at least one character in the Rx FIFO. It goes inactive when the Rx FIFO is empty. The transmitter DMA request signal (Tx RDY) goes active when there are no characters in the Tx FIFO. It goes inactive when there is at least one character in the Tx FIFO. Therefore, in single-transfer mode active and inactive DMA signals are issued on a one byte basis.

In multi-transfer mode Rx RDY goes active whenever the trigger level or the timeout has been reached. It goes inactive when the Rx FIFO is empty. Tx RDY goes active when there is at least one unfilled position in the Tx FIFO. It goes inactive when the Tx FIFO is completely full. Therefore in multi-transfer mode active and inactive DMA signals are issued as the FIFO fills and empties. With 2 DMA channels (one for each Rx and Tx) assigned to it, the NS16550A could run somewhat independently of the CPU when the DMA unit transfers data composed of blocks with checksums.

SYSTEM OPERATION: THE NS16550A VS THE NS16450

Consider the typical system interface block diagram in *Figure 2*. This is a simple diagram, but it includes all of the components that typically interact with a UART. The advantages of the NS16550A over the NS16450 can be illustrated by comparing some of the system constraints when each UART is substituted into this basic system.

Both RS-232C and RS-422A interfaces can be used with either UART, however, the NS16550A can drive these interfaces up to 256 kbaud. Regarding the RS-422A specifica-
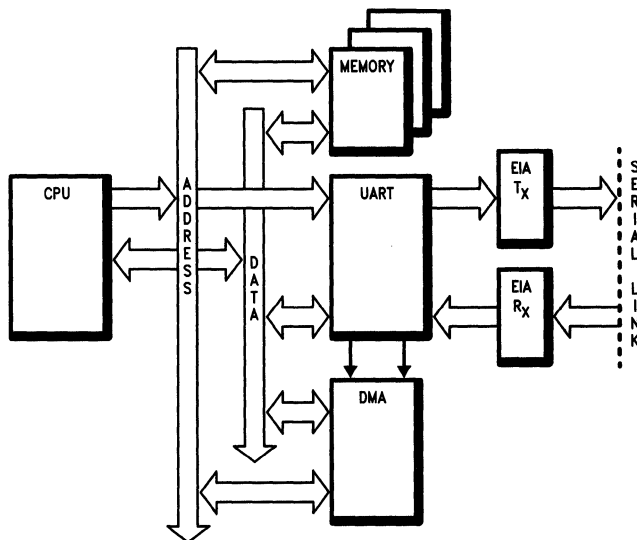
TL/C/9313-2

FIGURE 2. Typical System Interface

tion (max. 10 Mbaud) this is significantly faster than the NS16450 (max. 56 kbaud).

The NS16450 has no DMA request signals, so the DMA unit would not interact with the NS16450. The NS16550A, however, has DMA request signals and two modes of data transfer, as previously described, to interface with a variety of DMA units.

The greatest advantages of the NS16550A over the NS16450 are seen when considering the CPU/UART interface. Some characteristics of the transactions occurring between the CPU and the UART were previously cited. However, optimizing these transactions involves two issues:

1. Decreasing the amount of time the CPU interacts with the UART.

2. Increasing the amount of data transferred between the CPU and UART during their interaction time.

These optimization criteria are directly opposed to each other, but various features on the NS16550A have improved both.

One of the more obvious ways to decrease the CPU/UART interaction time is to decrease the time it takes for the transaction to occur. The NS16550A has an access cycle time that is almost 25% shorter than the NS16450. In addition, other timing parameters were made faster to simplify high speed CPU interactions.

The actual software required to transfer the data between the CPU and the UART is a small percentage of that required to support this transfer. However, each time a transfer occurs in the NS16450, this support software (overhead) must also be executed. With the NS16550A each time the UART needs service the CPU can theoretically transfer 16 bytes while only running through its overhead once. Tests have shown that this will increase the performance by a factor of 5 at the system level over the NS16450.

Another time savings for the CPU is a new feature of the UART interrupt structure. Unlike most other UARTs with Rx

FIFOs, the NS16550A will issue an interrupt when there are characters below the interrupt trigger level after a preset time delay. This saves the extra time spent by the CPU to check for bytes that are at the end of a block, but won't reach the interrupt level.

Since the NS16550A register set is identical to the NS16450 on power-up, all existing NS16450 software will run on it. The FIFOs are only enabled under program control.

All of this added performance is not without some trade-offs. Two of the NS16450 pins, no connect (NC) and chip select out (CSOUT) have been replaced by the RxRDY and TxRDY pins. Most serial cards that currently use the NS16450 don't use these pins, so in those situations the NS16550A could be used as a plug-in upgrade. The software drivers for the NS16550A operating in FIFO mode need to be a little more sophisticated than for the NS16450. This will not cause a great penalty in CPU operating time as there is only one additional UART register to program and one to check during the initialization. One additional service routine is required to handle Rx timeout interrupts. This routine does not execute, except during intermittent transmissions or as described above.

All of these speed improvements and allowances for software constraints will make the NS16550A an optimal UART for both multi-tasking systems and multiport systems. Multi-tasking systems benefit from the increased time and flexibility offered to the CPU during context switching. Multiport systems, such as terminal concentrators, benefit from the on-board FIFOs and relatively autonomous functions of the UART.

SYSTEM INTERRUPT GENERATION

As a prelude to the topic of the next section (80286™-based system initialization) a review of a typical PC hardware interrupt path is given. This concerns only the interrupt path between the UART and the CPU (see *Figure 3*).
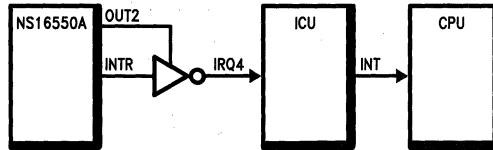
TL/C/9313-3

FIGURE 3. Typical PC Interrupt System Hardware

In order to enable interrupts from the UART to the CPU each hardware device must be correctly initialized. While initializing the hardware path, CPU interrupts are turned off to avoid false interrupts from this path. This initialization should be as short as possible to avoid other devices "stacking up" interrupts during this time.

After the NS16550A is initialized the bits 0–3 in the Interrupt Enable Register (IER) are set enabling all UART interrupts. Also, bit 3 in the Modem Control Register (MCR) is set to enable the buffer between the UART and the ICU.

The ICU has bit 4 of its Interrupt Mask Register (IMR) cleared, allowing interrupts occuring on IRQ4 to be transferred to the CPU via the group interrupt (INT). Finally, CPU interrupts are enabled again via the STI instruction.

The programmer should be aware that the ICU will be initialized for edge-triggered interrupts and that the UART always produces level active interrupts. This allows the system to get into a situation where the UART has multiple interrupts pending (signaled via a constantly high INTR), but the ICU fails to respond because it expects an edge for each pending interrupt. To avoid this situation, the programmer should disable all UART interrupts via the IER when entering each UART interrupt service routine and then reenable all UART interrupts that are to be used just before exiting each interrupt service routine.

SUMMARY

Up to this point the features of the NS16550A have been described, some of the design goals that resulted in these features have been reviewed, and a comparison has been given between it and the NS16450. Increases in bus speed and specialized functions make this part both faster from the hardware point of view and more efficient from the software point of view.

2.0 NS16550A Initialization

This initialization can be used on any 80286-based system; it enables both FIFOs and all interrupts on the UART. Additional procedures would have to be written to actually transfer data and service interrupts. These procedures would be similar in form to the 32000-based example in the next section, but the code would be different. The general flow of the initialization is shown in *Figure 4* and described below.

DETAILED SOFTWARE DESCRIPTION

The first block in the initialization establishes abbreviations for the NS16550A registers and assigns addresses to them. The next three blocks establish code and data segments for the 80286. After jumping to the code start, the program disables CPU interrupts (CLI) until it has finished the initialization routine. Other interrupts may be active while CPU inter-

rupts are masked, so the section of code following CLI should be as short as possible. The next block replaces the existing COM1 interrupt vector with the address of NS16550A interrupt handler (INTH in this case).

Initialization of the NS16550A is similar to the NS16450, except that there is one additional register to program which controls the FIFOs (Refer to the datasheet for a complete description). The sequence shown here sets bit 7 (DLAB) of the line control register (LCR), which enables access to the baud rate generator divisor. The divisor programmed is 0006 (19.2 kbaud) in this example. Programming the LCR again resets bit 7 (allowing access to the operational registers) and programs each frame for 7 data bits, one stop bit and even parity. The additional register that needs to be programmed in the NS16550A is the FIFO control register (FCR). The FCR data is 1100 0001. Bits 6 and 7 set the Rx FIFO interrupt trigger level at 14 characters. Bits 5 and 4 are reserved. Bit 3 keeps the DMA signal lines in mode 0. Setting bits 2 and 1 clear the Tx and Rx FIFOs, but this is done automatically when the FIFOs are first enabled by setting bit 0. Bit 0 of the FCR should ALWAYS BE SET whenever changes are to be made to the other bits of the FCR and the UART is to remain in FIFO Mode. When the FIFOs on the NS16550A are enabled bits 6 and 7 in the Interrupt Identification Register are set. Thus the program can distinguish between an NS16450 and an NS16550A, taking advantage of the FIFOs.

Sending a 0F to the Interrupt Enable Register enables all UART interrupts. The next two register accesses, reading the Line Status Register and the Modem Status Register, are optional. They are conservatively included in this initialization in order to defeat false interrupt indications in these registers caused by noise on the external lines.

The next block of code enables the interrupt signal to go beyond the UART through the system hardware. In many popular 80286-based personal computers, an interrupt control unit (ICU) has its mask register at I/O address 21H. To enable interrupts through this ICU for COM1 without disturbing other interrupts, the Interrupt Mask Register (IMR) is read. This data is combined with 1110 1111 via an AND instruction to unmask the COM1 interrupt and then loaded it back to the IMR. On these personal computers there is also a buffer on the interrupt line between the UART and ICU. This buffer is enabled by setting the OUT2 bit of the MODEM Control Register in the UART.

Before enabling CPU interrupts (STI) pointer registers to the data buffers of each service routine are loaded. After enabling CPU interrupts this program jumps to a holding loop to wait for an interrupt, whereas most programs would continue initializing other devices or jump to the system loop.

ESTABISH REGISTER
AND ADDRESS EQUATES

ESTABLISH CODE AND
DATA SEGMENTS;
JUMP TO CODE START

ESTABLISH DATA BUFFERS,
RAM REGISTERS

CODE START

DISABLE CPU
INTERRUPTS

LOAD COM1 RAM
POINTER WITH
SERVICE ROUTINE
ADDRESS

INITIALIZE NS16550A

ENABLE COM1 INTERRUPTS
AT THE ICU

SET OUT2 TO
ENABLE COM1
INTERRUPTS

ESTABLISH RUN TIME BUFFER
POINTERS IN CPU REGISTERS

ENABLE CPU INTERRUPTS

INITIALIZATION

ACTIVE
COM1
INTERRUPT
?   N   Y

READ IIR AND CALL
APPROPRIATE SERVICE ROUTINE

| LINE STATUS INTTERUPT | RECEIVER INTERRUPT | RECEIVER TIMEOUT INTERRUPT | TRANSMITTER EMPTY INTERRUPT | MODEM STATUS INTERRUPT | FALSE INTERRUPT |

TL/C/9313–4

FIGURE 4. NS16550A Initialization and Driver Flowchart

4

```
          TITLE    550APP.ASM - NS16550A INITIALIZATION
;
;ESTABLISH NS16550A REGISTER ADDRESS/DATA EQUATES
;
;************ UART REGISTERS ************************
;
rxd       EQU 3F8H          ;RECEIVE DATA REG
txd       EQU 3F8H          ;TRANSMITT DATA REG
ier       EQU 3F9H          ;INTERRUPT ENABLE REG
dll       EQU 3F8H          ;DIVISOR LATCH LOW
dlh       EQU 3F9H          ;DIVISOR LATCH HIGH
iir       EQU 3FAH          ;INTERRUPT IDENTIFICATION REG
fcr       EQU 3FAH          ;FIFO CONTROL REG
lcr       EQU 3FBH          ;LINE CONTROL REG
mcr       EQU 3FCH          ;MODEM CONTROL REG
lsr       EQU 3FDH          ;LINE STATUS REG
msr       EQU 3FEH          ;MODEM STATUS REG
scr       EQU 3FFH          ;SCRATCH PAD REG
;
;*************** DATA EQUATES *****************
;
bufsize EQU 7CFH            ;TX AND RX BUFFER SIZE
dosrout EQU 25H             ;DOS ROUTINE SPECIFICATION
intnum  EQU 0CH             ;INTERRUPT NUMBER (0CH = COM1)
icumask EQU 0EFH            ;ICU INTERRUPT ENABLE MASK
divacc  EQU 80H             ;DIVISOR LATCH ACCESS CODE
lowdiv  EQU 06H             ;LOWER DIVISOR
uppdiv  EQU 00H             ;UPPER DIVISOR
dataspc EQU 1AH             ;DLAB = 0, 7 BITS, 1 STOP, EVEN
fifospc EQU 0C1H            ;FIFOS ENABLED, TRIG = 14, DMA MODE = 0
setout2 EQU 08H             ;SETTING OUT2 ENABLES INTRs TO THE ICU
intmask EQU 0FH             ;UART INTERRUPT ENABLE MASK
;
;*********** ESTABLISH CODE AND DATA SEGMENTS ******************
;
cseg    SEGMENT PARA PUBLIC "code"
        ORG     100H
        ASSUME  CS:cseg,DS:cseg
INIT:
        PUSH    CS
        POP     DS
        JMP     START
;
;********* ESTABLISH DATA BUFFERS AND RAM REGISTERS ********
;
msflag  DB      0
txflag  DB      0
sbuf    DB      bufsize DUP ("S")     ; STRING BUFFER
rbuf    DB      bufsize DUP ("R")     ; RECEIVE BUFFER
sbufe   EQU     sbuf + bufsize        ; END OF STRING BUFFER
rbufe   EQU     rbuf + bufsize        ; END OF RECEIVE BUFFER
;
START:
        CLI                     ;>>> DISABLE CPU INTERRUPTS <<<
```

```
;
;****** LOAD NEW INTERRUPT SERVICE ROUTINE POINTER FOR COM1 ***
;
        PUSH    DS              ;SAVE EXISTING DATA SEG
        MOV     AH,dosrout      ;DESIGNATE FUNCTION NUMBER
        MOV     AL,intnum       ;DESIGNATE INTERRUPT
        PUSH    CS              ;ALIGN CODE SEG
        POP     DS              ;WITH DATA SEG
        MOV     DX,OFFSET INTH  ;SPECIFY SERVICE ROUTINE OFFSET
        INT     21H             ;REPLACE EXISTING INTR VECTOR
        POP     DS              ;RESTORE CURRENT DATA SEG
;
;***************  INITIALIZE NS16550A  **********************
;
;This enables both FIFOs for data transfers at 19.2 kbaud using
;7 bit data, 1 stop bit and even parity. The Rx FIFO interrupt
;trigger level is set at 14 bytes.
        MOV     AL,divacc       ;SET-UP ACCESS TO DIVISOR LATCH
        MOV     DX,lcr
        OUT     DX,AL
        MOV     AL,lowdiv       ;LOWER DIVISOR LATCH, 19.2 kbaud
        MOV     DX,dll
        OUT     DX,AL
        MOV     AL,uppdiv       ;UPPER DIVISOR LATCH
        MOV     DX,dlh
        OUT     DX,AL
        MOV     AL,dataspc      ;DLAB = 0, 7 BITS, 1 STOP, EVEN
        MOV     DX,lcr
        OUT     DX,AL
        MOV     AL,fifospc      ;FIFOS ENABLED, TRIGGER = 14,
        MOV     DX,fcr          ;DMA MODE = 0
        OUT     DX,AL
        MOV     AL,intmask      ;ENABLE ALL UART INTERRUPTS
        MOV     DX,ier
        OUT     DX,AL
        MOV     DX,lsr          ;READ THE LSR TO CLEAR ANY FALSE
        IN      AL,DX           ;STATUS INTERRUPTS
        MOV     DX,msr          ;READ THE MSR TO CLEAR ANY FALSE
        IN      AL,DX           ;MODEM INTERRUPTS
;
;*************** ENABLE COM1 INTERRUPTS **********************
;
        IN      AL,21H          ;CHECK IMR
        AND     AL,icumask      ;ENABLE ALL EXISTING AND COM1
        OUT     21H,AL
        MOV     AL,setout2      ;SET OUT2 TO ENABLE INTR
        MOV     DX,mcr
        OUT     DX,AL
;
;********* ESTABLISH RUN TIME BUFFER POINTERS IN REGISTERS ***
;
        MOV     SI,OFFSET sbuf
        MOV     DI,OFFSET rbuf
        MOV     BX,OFFSET sbuf
        MOV     BP,OFFSET rbuf
        STI     ;>>> ENABLE CPU INTERRUPTS <<<
```

4

3.0 Board to Board Communications with the NS16550A

The following section describes the hardware and software for a fully asynchronous two board application. The two boards communicate simultaneously with each other via the NS16550As. Predetermined data is exchanged between the NS16550As and checked by the software for accuracy. Any data mismatches are flagged and stop the programs. Any data errors (i.e. overrun, parity, framing or break) will also stop the program. The NS16550A interface schematic, software flow chart and software are provided.

HARDWARE REQUIREMENTS

Running this application requires two NS32032-based boards. Each board must have one CPU, one ICU (NS32202), 256k of RAM (000000–03FFFF), the capability of running a monitor program (MON 32) and the capability of interfacing with a terminal. If MON 32 is not available, the display monitor service calls (SVC) must be altered to interface properly to the available terminal driver routines. In addition to these requirements, the NS16550A is enabled starting at address 0d00000.

The system described above was implemented on two DB32032 boards and used as an alpha site to test the NS16550A during its development. An NS16550A and appropriate decode logic were wirewrapped to each board (see *Figure 5*). As shown, an 8 MHz crystal is used to drive the baud rate generator, but for baud rates at or below 56 kbaud a 1.8432 MHz crystal can be substituted with changes to the divisor. Once this hardware is on both boards 5 connections between the NS16550As must be made—SIN to SOUT, SOUT to SIN, $\overline{CTS}$ to $\overline{RTS}$, $\overline{RTS}$ to $\overline{CTS}$, and GND to GND. Each DB32032 board has a port for attaching a terminal and a port available for downloading code. The applications software for these boards is downloaded from a VAX™ running the GNX™ debugger (V1.02). Once the downloads are complete to both boards the program D1APPS.EXE is started, then D2APPS.EXE is started.

If a VAX or the GNX debugger is not available the code can be loaded into PROMs and run directly.
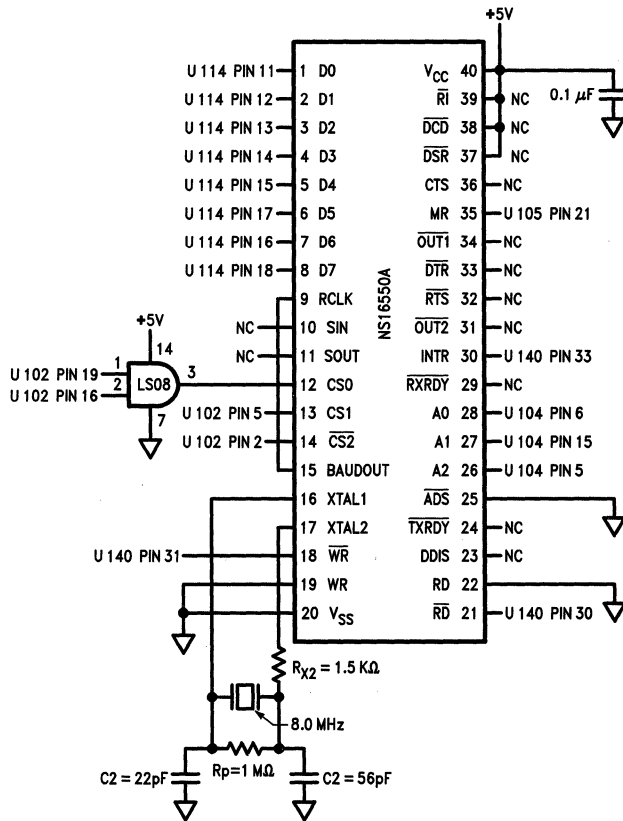
FIGURE 5. NS16550A and DB32032 Board Interconnections

TL/C/9313–5

SOFTWARE OVERVIEW

The programs shown at the end of this application note are the assembly listings for D1APPS.ASM and D2APPS.ASM. These can be assembled, linked and loaded to form the executable (.EXE) files. The flowchart shown before them illustrates both programs.

Both programs are interrupt driven. D1APPS.EXE has its transmitter empty interrupt disabled until it receives its first 16 bytes from D2APPS.EXE. This allows the two programs to be started at different times. Data flow is controlled between the programs via $\overline{RTS}$ and $\overline{CTS}$ handshakes. D1APPS.EXE is started first and it loops until the first data from D2APPS.EXE arrives. As D1APPS.EXE exits its receiver interrupt routine, it enables its transmitter interrupt and begins to send bytes to D2APPS.EXE.

Transmission of a block of 16 bytes occurs when the Tx FIFO of the NS16550A is empty, the Tx interrupt is enabled and the receiver activates its clear to send ($\overline{CTS}$) signal. Each transmitter sends the next sequential block of data from a 256 byte buffer. When the bottom of the buffer is reached, the transmitter starts at the top of the buffer, again. The data transmitted from D1APPS.EXE to D2APPS.EXE is 00 to FF and from D2APPS.EXE to D1APPS.EXE is FF to 00. Since these are bench test programs for the NS16550A, the receiver subroutines compare the data they receive with the data they expect. This is done on a block-by-block basis and any mismatches result in both a message sent to the terminal and the program stopping.

DETAILED SOFTWARE DESCRIPTION

Initialization begins by equating NS16550A and ICU (NS32202) registers to the addresses in memory. The equates finish with a list of offsets associated with the static base register. These offsets give the starting locations for the RAM areas assigned to be data buffers. These include the UART interrupt entry offset (irl_mod); the string (sbuf), receive (rbuf), compare (cbuf) buffers and the interrupt table offset (intable).

At the code start (START::) the processor is put in the supervisor mode so that the interrupt dispatch table can be transferred from ROM to RAM. This transfer is essential in order to change the starting address of the UART interrupt service routine. To do this the interrupt service routine offset from the code start is calculated (isr-start). Combining this with the module table address (set-up by the linker, i.e., 9020) results in the interrupt table descriptor entry for UART interrupt service routine (isrent).

The next two sections of code load the data to be transmitted and compared into the RAM buffers sbuf and cbuf, respectively. The two programs differ at this point— D1APPS.EXE transmits 00 to FF and compares FF to 00 sequentially. D2APPS.EXE transmits FF to 00 and compares 00 to FF sequentially.

The NS16550A initialization starts with setting the divisor latch access bit, so the divisor can be loaded. It then determines the serial data format and disables all UART interrupts. The NS16550A initialization finishes by enabling and resetting the FIFOs and programming the receiver interrupt level for 14 bytes.

Next the ICU interrupt registers are set-up and interrupts are enabled. In program D1APPS.ASM the initialization finishes by enabling the receive data and line status interrupts. Since the transmitter FIFO empty interrupt is disabled D1APPS.EXE will stay in its hold loop until it receives data from D2APPS.EXE. D2APPS.EXE has its transmitter FIFO empty interrupt enabled at the end of its initialization, so it will send one block of 16 characters to D1APPS.EXE immediately.

When there are no interrupts pending and no service routines being executed, the programs run in a holding loop until the next interrupt.

Whenever the CPU enters the service routine (isr:) it checks the interrupts identification register (IIR) for the type of interrupt pending and branches to the appropriate subroutine. If the IIR value doesn't match a known interrupt condition, an invalid interrupt message is sent to the terminal and the program stops. Out of the five possible interrupts, two (line status and receiver timeout) have simple routines that only send a message to the terminal and then branch to the receiver data available routine. Modem status interrupts send a message to the CRT and then stop the program. Two robust interrupt service routines exist—one for the receiver and one for the transmitter.

The receiver interrupt service routine (rdai:) does the following:

1. Disables the $\overline{RTS}$ signal which stops the transmitter on the other board from sending more data.
2. Transfers all data from the UART Rx FIFO to the RAM receiver buffer (rbuf).
3. Branches to the compare subroutine when all data is transferred from the Rx FIFO.
4. Enables Tx interrupts in D1APPS.EXE.
5. Enables the $\overline{RTS}$ signal which allows the transmitter on the other board to send another block of data.

The compare interrupt service routine (compare:) does the following:

1. Aligns the receive buffer pointer to the last character taken in to the receive buffer (rbuf).
2. Compares each new byte in rbuf with the expected value (data stored in cbuf).
3. Sends a data mismatch message to the terminal and stops the program if the rbuf data fails to match the cbuf data.
4. Returns to rdai: when all of the new data in rbuf has compared successfully.

The transmitter interrupt service routine (threi:) does the following:

1. Decides whether to send 16 or 15 bytes in a block of data. **Note:** This decision is for testing purposes.
2. Sends one byte of data.
3. Checks for an active $\overline{CTS}$ condition. If it is active then it sends another byte of data. It continues to check and send a byte of data until all 15 or 16 bytes are sent.

4

DIAPPS.ASM Flow Chart

ESTABLISH REGISTER AND
ADDRESS EQUATES FOR THE NS16550A
AND THE NS32202 (ICU)

INITIALIZATION

ESTABLISH STATIC BASE
STARTING LOCATIONS

START::

SET UP INTERRUPT DISPATCH
TABLE FOR THE 32032

LOAD RAM STRING BUFFER
FF TO 00 (NOTE)

R1 = 00
?

NO

YES

LOAD RAM COMPARISON
BUFFER 00 TO FF

R1 = FF
?

NO

YES

SET UP INTERRUPT SERVICE
ROUTINE PARAMETER

INITIALIZE NS16550A
(NOTE)

INITIALIZE NS32202

INITIALIZE TRANSMITTER
BUFFER OFFSET

ENABLE CPU INTERRUPTS, ENABLE
/RTS AND NS16550A INTERRUPTS

HOLDLOOP::

NO

INTR
ACTIVE
?

YES

ISR:

TL/C/9313–7

Note: This part of the software differs slightly in D2APPS.ASM

SAVE CPU
GENERAL PURPOSE
REGISTERS

INTERRUPT
SERVICE
ROUTINE

ISR:

READ UART INTERRUPT
STATUS REGISTER

IS IT A
LINE STATUS
INTERRUPT
? — YES → JUMP TO LSINT:

NO

IS IT A RECEIVER
DATA AVAILABLE
INTERRUPT ? — YES → JUMP TO RDAI:

NO

IS IT A
RECEIVER TIMEOUT
INTERRUPT ? — YES → JUMP TO RTMOUT:

NO

IS IT A TRANSMITTER
HOLDING REGISTER EMPTY
INTERRUPT ? — YES → JUMP TO THREI:

NO

IS IT A
MODEM STATUS
INTERRUPT → JUMP TO MSINT:

PRINT INVALID INTERRUPT → STOP:

TL/C/9313-8

LSINT: SEND MESSAGE "LINE STATUS INTERRUPT"
SAVE RECEIVER STATUS REGISTER → JUMP TO RDAI:

RTMOUT: SEND MESSAGE "LINE STATUS INTERRUPT" → JUMP TO RDAI:

MSINT: SEND MESSAGE "MODEM STATUS INTERRUPT" → JUMP TO POPALL:

POPALL: RESTORE CPU GENERAL PURPOSE REGISTERS

RETURN FROM INTERRUPT

TL/C/9313-9

4

RDAI: → DISABLE /RTS, SET-UP RECEIVER POINTER BASE ADDRESS AND OFFSET

RDRBR: → STORE RECEIVER BYTE INCREMENT RECEIVER OFFSET

LAST POSITION IN RECEIVER BUFFER ? — YES → REINITIALIZE RECEIVER POINTER OFFSET

NO

CONTINUE: → READ RECEIVER STATUS REGISTER IN UART

IS THERE ANOTHER BYTE IN THE RECEIVER ? — YES

NO

SAVE RX POINTER OFFSET

BRANCH TO COMPARE

ENABLE /RTS ENABLE TX INTERRUPTS (NOTE)

JUMP TO POPALL

TL/C/9313–10

Note: This part of the software differs slightly in D2APPS.ASM

4-68

THREI:

SET UP TRANSMITTER
POINTER BASE ADDRESS
AND POINTER OFFSET

16
BLOCKS
SENT
?

YES → INITIALIZE BYTE COUNTER TO
SEND 1 BLK OF 15 BYTES

NO

LOAD TRANSMITTER
WITH 1 BYTE OF DATA

INCREMENT
OFFSET R1

END
OF TX BUFFER
?

YES → REINITIALIZE
POINTER OFFSET

IS
RECEIVER
/CTS INACTIVE
?

YES → SAVE
POINTER OFFSET

NO

DECREMENT
BYTE COUNTER

ALL
BYTES SENT
?

SAVE TX
POINTER OFFSET

16
BLOCKS SENT
?

YES → RELOAD BLOCK
COUNTER WITH H'10

NO

NO

DECREMENT
BYTE COUNTER

JUMP
TO
POPALL

TL/C/9313–11

4

4-69

COMPARE:

SET UP COMPARE
BUFFER POINTER
BASE ADDRESS

IS RECEIVED OFFSET AT TOP OF RECEIVE BUFFER ?

YES → LOAD LAST BYTE POSITION OFFSET INTO RECEIVER BYTE POINTER

NO → DECREMENT RECEIVE OFFSET

LOAD COMPARE BUFFER OFFSET

COMPARE DATA SENT WITH DATA RECEIVER

DOES DATA MATCH ?

NO → SET ERROR STROBE SIGNAL → CALL SERVICE ROUTINE TO DISPLAY "DATA MISMATCH MESSAGE" → STOP

YES

IS THIS THE END OF THE COMPARE BUFFER ?

YES → INCREMENT TRANSMITTER BUFFER COUNT → RESET COMPARE BUFFER OFFSET

NO → INCREMENT COMPARE BUFFER OFFSET

IS THE COMPARE BUFFER OFFSET = TO THE RECEIVE BUFFER OFFSET ?

YES → RETURN FROM BRANCH

NO

TL/C/9313–12

4-70

```
#3/30/87.....D1APPS.ASM.........ADAPTED ORIGINALLY FROM D1RON56K.ASM
#
#THIS PROGRAM RUNS USING 2 DB32000 BOARDS WITH 16550As ENABLED AT ADDRESS 0d00000
#WIRE-WRAPPED ON THE BOARDS. THIS SOFTWARE TRANSMITS THE DATA 00 THROUGH FF
#REPEATEDLY TO THE REMOTE UART AND EXPECTS TO REPEATEDLY RECEIVE THE DATA FF
#THROUGH 00 FROM THE REMOTE UART. IT SHOULD BE RUN IN CONJUNCTION WITH THE
#PROGRAM D2APPSC.ASM RUNNING ON THE OTHER DB32000 BOARD. THE TX PIN OF
#THIS 16550A SHOULD CONNECT TO THE RX PIN OF THE 16550A ON THE OTHER BOARD AND
#VICE VERSA. ALSO, THE CTS PIN OF THIS 16550A SHOULD BE CONNECTED TO THE RTS PIN
#OF THE 16550A ON THE OTHER BOARD AND VICE VERSA. THIS WILL ENABLE THE
# APPROPRIATE HANDSHAKES TO OCCUR.
#
#TO RUN THIS PROGRAM YOU MUST:
#
#        1. CONNECT THE RX & TX OF THE 2 16550As ON THE 2 DB32000 BOARDS
#        2. CONNECT THE CTS & RTS OF THE 2 16550As ON THE 2 DB32000 BOARDS
#        3. DOWNLOAD D1APPS.EXE TO THIS BOARD VIA THE GNX DEBUGGER [REV 1.02]
#        4. DOWNLOAD D2APPS.EXE TO OTHER BOARD VIA THE GNX DEBUGGER [REV 1.02]
#        5. START D1APPS.EXE RUNNING ON THIS DB32000 BOARD
#        6. START D2APPS.EXE RUNNING ON THE OTHER DB32000 BOARD
#
#PROGRAM DETAILS:
#
#
# ISR contains the TX SERVICE ROUTINE
#
# TX OVERWRITES are PREVENTED by the ICU
#
# TX FIFO is CLEARED before a transmission
#
# DATA SENT  00 ------ FF
#
# DATA RECEIVED  and COMPARED FF ------ 00
#
# BAUDRATE 128k WITH A 8.0 MHZ XTAL INPUT TO THE 16550A
#
#********************** ESTABLISH 16550A REGISTER ADDRESSES ********************
#                                     #
        .globl        isr            #
            .set rxd,       0x0d00000  #Equate registers to their addresses
            .set txd,       0x0d00000  #
            .set ier,       0x0d00004  #
            .set iir,       0x0d00008  #
            .set fcr,       0x0d00008  #
            .set lcr,       0x0d0000c  #
            .set mcr,       0x0d00010  #
            .set lsr,       0x0d00014  #
            .set msreg,     0x0d00018  #
            .set scr,       0x0d0001c  #
                                       #
#****************** ESTABLISH ADDRESSES FOR THE 32202 (ICU) ******************
#                                     #
            .set a0,4                  #Establish address alignment
                                       #between CPU and ICU
            .set icu_hvct,0            #ICU register addresses
            .set icu_svct,1 *a0        #
            .set icu_elgt,2 *a0        #
            .set icu_tpl,4  *a0        #
            .set icu_ipnd,6 *a0        #
            .set icu_isrv,8 *a0        #
```

TL/C/9313–13

4

```
            .set  icu_imsk,10 *a0          #
            .set  icu_csrc,12 *a0          #
            .set  icu_fprt,14 *a0          #
            .set  icu_mctl,16 *a0          #
            .set  icu_ciptr,18 *a0         #
            .set  icu_pdat,19  *a0         #
            .set  icu_ips,20   *a0         #
            .set  icu_pdir,21  *a0         #
            .set  icu_cctl,22  *a0         #
            .set  icu_cictl,23 *a0         #
                                           #
                                           #First ICU register address
                                           #
            .set  icu_addr,0xfffe00         #
                                           #
#************************* STATIC BASE STARTING LOCATIONS ***********************
                                           #
            .set  irl_mod, 17*4            #
            .set  irl_off, 17*4+2          #
            .set  start2, 0x0              #The following are static base variables
            .set  start1, 0x0a             #used as base pointers. Start1/2 = flags
            .set  txflag, 0x14             #txflaf = flag, sbuf = area used to
            .set  sbuf, 0x1e               #store data to be transmitted, rbuf =
            .set  rbuf, 0x41e              #area used to store received data,
            .set  cbuf, 0x61e              #cbuf = area used to store compare
            .set  intable, 0x81e           #buffer, intable = base pointer to the
                                           #interrupt table
                                           #
#********************** SET UP DISPATCH TABLE FOR THE 32032 ********************
                                           #
start::     bicpsrw $(0x100)              #Clear intr's
            movd $0x0c,r0                 #Set for monitor svc to move intbase
            movd $0x055555555,r1          #from ROM to ram because you have
            addr intable(sb),r2           #to change the address for the
            movd $0x0c,r3                 #interrupt service routine.
            svc                           #Actual svc for move
            sprd intbase,r2               #Put base addr of intbase in r2
            movd isrent,irl_mod(r2)       #Put offset of isr into 1st location
                                          #of dispatch table
                                          #
#********************** LOAD TRANSMITTER BUFFER (00 to FF) ********************
                                          #
senddat:    addr  sbuf(sb),r0             #R0 contains string buffer ptr.
            movd  $0,r1                    #R1 contains offset
            movb  $0,r2                    #Init data reg.
sbufloop:   movb  r2,0(r0)[r1:b]          #Load char. to string buffer
            addqw 1,r1                     #Increment offset ptr.
            addqw 1,r2                     #Increment data
            cmpw  r1,$256                  #Check for 256 chars. loaded
            bne   sbufloop                 #Jump back if not done
                                          #
#*********************** LOAD COMPARISON BUFFER (FF TO 00)*********************
                                          #
compdat:    addr  cbuf(sb),r0             #R0 contains pointer
            movd  $0,r1                    #R1 contains offset
            movb  $0x0ff,r2                #Init data reg.
cbufloop:   movb  r2,0(r0)[r1:b]          #Load char. to compare buffer
            addqw 1,r1                     #Increment ptr. offset
            subb  $1,r2                    #Decrement data
            cmpw  r1,$256                  #Check for 256 chars. loaded
```

TL/C/9313–14

```
                bne    cbufloop          #Jump back if not done
                                          #
#*************** SET UP INTERRUPT SERVICE ROUTINE PARAMETERS ******************
                                          #
                movd $0x0ff,start2(sb)    #Initialize compare
                movd $0x0ff,startl(sb)    #Initialize receiver data intr
                movd $16,blk16cnt         #Initialize 16 byte block counter
                movd $0,sbufcnt           #Initialize string bufffer transmitted
                                          #count
                                          #
#************************** 16550A INITIALIZATION *****************************
                                          #
                movb  $0x080,lcr          #Set dlab = 1 for divisor latch access
                movb  $4,txd              #Low divisor latch 128k w/8.0 MHz xtal
                movb  $0,ier              #Upper divisor latch
                movb  $0x003,lcr          #Dlab = 0, 8 bits, no parity, 1 stop
                movb  $0,ier              #Disable UART interrupts
                movb  $0x0c7,fcr          #Fifo=> trigger = 14, reset & enable
                                          #
#*************************** INITIALIZE 32202 (ICU) ***************************
                                          #
        movd    $icu_addr,r0              #R0 = icu address
        movb    $0xca,icu_mctl(r0)        #Set mode : 8 bit bus mode,
                                          #           freeze counters,
                                          #           disable interrupts,
                                          #           fixed priority.
        movqb 0,icu_cctl(r0)              #Halt the counters
        movqb -1,icu_ips(r0)              #Set all pins to interrupt source
        movqb 0,icu_csrc(r0)              #No cascaded interrupts (low reg)
        movqb 0,icu_csrc+a0(r0)           # (high reg)
        movb $0x10,icu_svct(r0)           #Set interrupt base vector
        movqb -1,icu_elgt(r0)             #Set level triggering mode (low reg)
        movqb -1,icu_elgt+a0(r0)          #(high reg)
        movqb $2,icu_tpl(r0)              #Set level triggering mode (low reg)
        movqb 0,icu_tpl+a0(r0)            #(high reg)
        movqb 0,icu_fprt(r0)              #Set highest priority to 0 (low reg)
        movqb 0,icu_fprt+a0               #(high reg)
        movqb 0,icu_isrv(r0)              #Clear intr in-service regs (low reg)
        movqb 0,icu_isrv+a0(r0)           #(high reg)
        movqb -1,icu_imsk(r0)             #Mask all intr (low reg)
        movqb -1,icu_imsk+a0(r0)          #(high reg)H
        setcfg [i]                        #Enable vectored intrp (I=1)
        movd $icu_addr,r0                 #
        movb $0x02,icu_mctl(r0)           #Fixed mode, 8 bit bus mode
        movb $0x010,icu_cctl(r0)          #Set to internal sampling
        movb $0xfd,icu_imsk(r0)           #Enable irl
        movb $0xff,icu_imsk+a0(r0)        #Mask all other interrupts
        bispsrw $(0x800)                  #Enable cpu intr's
                                          #
        movd $0,r1                        #Initialize transmitter buffer offset
                                          #
#*************************** ENABLE 16550A INTERRUPTS *************************
                                          #
                movb $2,mcr               #Clear outl, out2 and enable rts
endinit:        movb $0x05,ier            #Enable all but modem status interrupts
                                          #and the THRE so the boards can be
                                          #started.
                                          #
#********************** ENDLESS LOOP WAITING FOR INTERRUPTS ******************
                                          #
```

TL/C/9313–15

4

```
holdloop:         nop                    #
                  br holdloop            #
                                         #
#************************** INTERRUPT HANDLER ***************************
                                         #
isr:              save [r0,r1,r2,r3,r4,r5,r6,r7]
                  movb iir,r0            #R0- contains iir
                  cmpb r0,$0x0c6         #
                  beq lsint             #Line status interrupt
                  cmpb r0,$0x0c4         #
                  beq rdai              #Receiver interrupt
                  cmpb r0,$0x0cc         #
                  beq rtmout            #Rec timeout interrupt
                  cmpb r0,$0x0c2         #
                  beq threi            #THRE interrupt
                  cmpb r0,$0x0c0         #
                  beq msint            #Modem status interrupt
                                         #
#************************ INVALID INTERRUPT ROUTINE **********************
                                         #
                  save [r0,r1,r2,r3]     #
                  movd $4,r0             #
                  addr message2,r1       #
                  movd $21,r2            #
                  movd $0,r3             #
                  svc                    #
                  restore [r0,r1,r2,r3]  #
                                         #
                                         #
                  jump stop             #Restore all registers
                                         #
                                         #
#********************* RECEIVER TIMEOUT INTERRUPT ROUTINE *********************
                                         #
rtmout:           jump rdai
                                         #
#*********************** RECEIVER INTERRUPT ROUTINE ***********************
                                         #
#This portion of the program is reached by a positive test for the received data
#available interrupt. Once in this routine each byte is removed from the FIFO,
#placed in a designated static base memory location and the LSR is tested to see
#if the data ready (DR) bit is still set. Data is removed from the FIFO and
#placed in memory until the DR bit is no longer set. The data sent will be
#compared to known data, located in another designated static base location, by
#calling the compare subroutine.          #
rdai:             movb $0,mcr           #Disable RTS; stop transmission
                  addr rbuf(sb),r4      #r4 contains rbuf base address
                  movd rbufoff,r6       #Put rbuf offset runner into r6
rdrbr:            movb rxd,0(r4)[r6:b]  #Store a byte in the receiver buffer
                  cmpb $0x00,0(r4)[r6:b] #Is it the last character
                  addqw 1,r6            #Increment offset ptr.
                  addqw 1,rbufoff       #Track r6
                  bne continue          #
                  movw $0,r6            #Reset pointer offset
                  movw $0,rbufoff       #Reset rbufoff
continue:         movb lsr,r3          #Read lsr
                  andb $01,r3          #Mask all but bit 0
                  cmpb $01,r3          #
```

TL/C/9313–16

```
                beq rdrbr               #Read rbr again if set
                movd r6,rbufoff         #Put result of r6 back into rbufoff
                bsr compare             #
                movb $7,ier             #Turn on transmitter interrupts
                movb $2,mcr             #Enable rts
                jump popall             #
                                        #
#***************************** TRANSMIT ROUTINE *******************************
                                        #
#Before the transmitter sends data, the data has been loaded into static base
#memory for transmission. The transmitter routine is called to send data. (ie
#THREI is set) Data is sent as 16 blocks of 16 bytes and 1 block of 15 bytes
#continuously. NOTE: Before transmission occurs /CTS is checked to ensure that
#the receiver is ready.
                                        #
threi:          addr sbuf(sb),r0        #R0 contains base pointer
                movw xmitoff,r1         #setup xmit ptr offset
                cmpd $0,blk16cnt        #Check to see if it is the 16th block *
                beq send15              #Yes, send only 15 bytes instead of 16 *
                movd $0x10,r7           #No, send 16 bytes *
                jump sendnext           #Jump around 15 byte load *
send15:         movd $0x0f,r7           #Load counter for 15 byte load *
sendnext:       movb 0(r0)[r1:b],txd    #Load a byte into the transmitter
                addqw 1,r1              #
                cmpw r1,$256            #Are we one address past end of table
                beq reload              #Yes, reload ptr
finish:         save [r7]               #
                movb msreg,r7           #Read modem status reg
                andb $0x10,r7           #Mask all bits except CTS (MSR4)
                cmpb $0,r7              #Check for disabled CTS
                restore [r7]            #
                beq abort               #Wait for active CTS (MSR4=1)
                subb $1,r7              #No, decrement counter and continue
                cmpb $0,r7              #Is byte counter 0?
                bne sendnext            #No, send next byte
abort:          movw r1,xmitoff         #save xmit ptr offset in ram
                cmpd $0,blk16cnt        #Check to see if it is 16th block *
                beq setsnd16            #Yes, reload block counter *
                subb $1,blk16cnt        #Decrement block counter *
                jump popall             #Finished sending 16 bytes
setsnd16:       movd $16,blk16cnt       #Reload block counter *
                jump popall             #Finished sending 15 bytes *
reload:         movd $0,r1              #Reset offset
                jump finish             #Go back and finish
                                        #
#*********************** LINE STATUS INTERRUPT ROUTINE ***********************
                                        #
lsint:          save [r0,r1,r2,r3]      #
                movd $4,r0              #
                addr message6,r1        #
                movd $25,r2             #
                movd $0,r3              #
                svc                     #
                restore [r0,r1,r2,r3]   #
                movb lsr,r3             #Read lsr
                                        #
                                        #
                jump rdai               #
                                        #
#*********************** MODEM STATUS INTERRUPT ROUTINE **********************
```

TL/C/9313-17

4

```
msint:          save [r0,r1,r2,r3]      #
                movd $4,r0              #
                addr message7,r1        #
                movd $26,r2             #
                movd $0,r3              #
                svc                     #
                movb 0x0d00018,r0       #
                restore [r0,r1,r2,r3]   #
                jump popall             #
#**************************** COMPARE DATA ROUTINE ****************************
                                        #
#This subroutine is called by the receiver interrupt routine which has set the
#receiver offset (rbufoff) to point at the last byte received. This subroutine
#uses the compare offset (compoff) pointer as the pointer for both receive
#buffer data and compare buffer data. Each location is compared to ensure data
#sent is identical to data received. This is done until compoff equals rbufoff
#stopping the process and returning from the interrupt. NOTE: Data being
#received is known data and an exact copy is loaded into memory prior to any
#transmission.
                                        #
compare:        addr cbuf(sb),r1        #R1- base address of cbuf base
                cmpd $0,r6              #Check for potential invalid subtraction
                beq zeror6              #Jump around subtraction
                subd $1,r6             #
                jump compbyte           #Jump around subtraction fix
zeror6:         movd $0xff,r6           #
compbyte:       movd compoff,r5         #
                cmpb 0(r1)[r5:b],0(r4)[r5:b] #Compare data sent to data received
                bne wrong               #Branch and set out1 if wrong
                                        #
                cmpb $0x00,0(r4)[r5:b]  #Check for end of buffer
                bne notend              #Branch and increment pointers
                jump reload1            #Test for having compared all bytes
                                        #
notend:         addd $1,compoff         #Increment pointer
notend1:        cmpd r5,r6              #
                beq bye                 #
                jump compbyte           #
                                        #
reload1:        addd $1,sbufcnt         #Increment transmiter cnt
                movd $0,compoff         #Reload offset of pointer
                jump notend1            #
                                        #
wrong:          nop                     #
                movb $0x0c,mcr          #Set out 2, for error strobe
                                        #
#*********************** DATA MISMATCH MESSAGE *****************************
                                        #
                save [r0,r1,r2,r3]      #Save register for supervisor call
                movd $4,r0              #Value required by svc call
                addr message8,r1        #Mover address of message into r1
                movd $17,r2             #Number of characters into r2
                movd $0,r3              #Value required by svc call
                svc                     #Actual call
                restore [r0,r1,r2,r3]   #Restore registers
                                        #
stop:           nop                     #
                jump stop               #Test point
                                        #
```

TL/C/9313-18

```
bye:            ret 0                     #
                                          #
#*************************** RETURN FROM INTERRUPT ****************************
                                          #
popall:         restore [r0,r1,r2,r3,r4,r5,r6,r7]
                reti                      #
                                          #
#********************************* Messages ***********************************
                                          #
                message1: .byte 13,10,"Compare Complete",13,10
                message2: .byte 13,10,"Invalid Interrupt",13,10
                message3: .byte 13,10,"Receiver Timeout",13,10
                message4: .byte 13,10,"Receive data available Interrupt",13,10
                message5: .byte 13,10,"THRE Interrupt",13,10
                message6: .byte 13,10,"Line Status Interrupt",13,10
                message7: .byte 13,10,"Modem Status Interrupt",13,10
                message8: .byte 13,10,"Data Mismatch",13,10
                xmitoff: .double 0
                compoff: .double 0
                blk16cnt: .double 0
                sbufcnt: .double 0
                rbufoff: .double 0
isrent: .word   0x9020                    #Mod table
        .word   isr-start                 #Offset of service routine for
                                          #Dispatch table.
```

                                          TL/C/9313-19

4

```
#3/30/87.....D2APPS.ASM.........ADAPTED ORIGINALLY FROM D1RON56K.ASM
#
#THIS PROGRAM RUNS USING 2 DB32000 BOARDS WITH 16550As ENABLED AT ADDRESS
#0d00000 WIRE-WRAPPED ON THE BOARDS. THIS SOFTWARE TRANSMITS THE DATA FF
#THROUGH 00 REPEATEDLY TO THE REMOTE UART AND EXPECTS TO REPEATEDLY RECEIVE
#THE DATA 00 THROUGH FF FROM THE REMOTE UART. IT SHOULD BE RUN IN CONJUNCTION
#WITH THE PROGRAM D1APPS.ASM RUNNING ON THE OTHER DB32000 BOARD. THE TX PIN OF
#THIS 16550A SHOULD CONNECT TO THE RX PIN OF THE 16550A ON THE OTHER BOARD AND
#VICE VERSA. ALSO, THE CTS PIN OF THIS 16550A SHOULD BE CONNECTED TO THE RTS PIN
#OF THE 16550A ON THE OTHER BOARD AND VICE VERSA. THIS WILL ENABLE THE
# APPROPRIATE HANDSHAKES TO OCCUR.
#
#TO RUN THIS PROGRAM YOU MUST:
#
#        1. CONNECT THE RX & TX OF THE 2 16550As ON THE 2 DB32000 BOARDS
#        2. CONNECT THE CTS & RTS OF THE 2 16550As ON THE 2 DB32000 BOARDS
#        3. DOWNLOAD D2APPS.EXE TO THIS BOARD VIA THE GNX DEBUGGER [REV 1.02]
#        4. DOWNLOAD D1APPS.EXE TO OTHER BOARD VIA THE GNX DEBUGGER [REV 1.02]
#        5. START D1APPS.EXE RUNNING ON THE OTHER DB32000 BOARD
#        6. START D2APPS.EXE RUNNING ON THIS DB32000 BOARD
#
#PROGRAM DETAILS:
#
#
# ISR contains the TX SERVICE ROUTINE
#
# TX FIFO is CLEARED before a transmission
#
# DATA SENT  FF ------ 00
#
# DATA RECEIVED  and COMPARED 00 ------ FF
#
# BAUDRATE 128k WITH A 8.0 MHZ XTAL INPUT TO THE 16550A
#                                          #
#********************** ESTABLISH 16550A REGISTER ADDRESSES ********************
#                                          #
        .globl          isr               #
            .set rxd,       0x0d00000     #Equate registers to their addresses
            .set txd,       0x0d00000     #
            .set ier,       0x0d00004     #
            .set iir,       0x0d00008     #
            .set fcr,       0x0d00008     #
            .set lcr,       0x0d0000c     #
            .set mcr,       0x0d00010     #
            .set lsr,       0x0d00014     #
            .set msreg,     0x0d00018     #
            .set scr,       0x0d0001c     #
                                          #
#***************** ESTABLISH ADDRESSES FOR THE 32202 (ICU) ******************
#                                          #
            .set a0,4                     #Establish address alignment
                                          #between CPU and ICU
            .set icu_hvct,0               #ICU register addresses
            .set icu_svct,1 *a0           #
            .set icu_elgt,2 *a0           #
            .set icu_tpl,4  *a0           #
            .set icu_ipnd,6 *a0           #
            .set icu_isrv,8 *a0           #
            .set icu_imsk,10 *a0          #
            .set icu_csrc,12 *a0          #
```

TL/C/9313–20

```
            .set icu_fprt,14 *a0        #
            .set icu_mctl,16 *a0        #
            .set icu_ciptr,18 *a0       #
            .set icu_pdat,19  *a0       #
            .set icu_ips,20   *a0       #
            .set icu_pdir,21  *a0       #
            .set icu_cctl,22  *a0       #
            .set icu_cictl,23 *a0       #
                                        #
                                        #First ICU register address
                                        #
            .set icu_addr,0xfffe00      #
                                        #
#*********************** STATIC BASE STARTING LOCATIONS **********************
                                        #
            .set irl_mod, 17*4          #Dispatch table offset for IRl entry
            .set sbuf, 0x1e             #sbuf = area used to
            .set rbuf, 0x41e            #store data to be transmitted, rbuf =
            .set cbuf, 0x61e            #area used to store received data,
            .set intable, 0x81e         #cbuf = area used to store compare
                                        #buffer, intable = base pointer to the
                                        #interrupt table

#********************** SET UP DISPATCH TABLE FOR THE 32032 ******************
                                        #
start::         bicpsrw $(0x100)           #Clear intr's
                movd $0x0c,r0              #Set for monitor svc to move intbase
                movd $0x055555555,r1      #from ROM to ram because you have
                addr intable(sb),r2       #to change the address for the
                movd $0x0c,r3             #interrupt service routine.
                svc                       #Actual svc for move
                sprd intbase,r2           #Put base addr of intbase in r2
                movd isrent,irl_mod(r2)   #Put offset of isr into 1st location
                                          #of dispatch table
                                          #
#********************* LOAD TRANSMITTER BUFFER (FF to 00) ********************
                                          #
senddat:        addr  sbuf(sb),r0         #R0 contains string buffer ptr.
                movd  $0,r1               #R1 contains offset
                movb  $0x0ff,r2           #Init data reg.
sbufloop:       movb  r2,0(r0)[r1:b]      #Load char. to string buffer
                addqw 1,r1                #Increment offset ptr.
                subb  $1,r2               #Increment data
                cmpw  r1,$256             #Check for 256 chars. loaded
                bne   sbufloop            #Jump back if not done
                                          #
#********************** LOAD COMPARISON BUFFER (00 TO FF) *******************
                                          #
compdat:        addr  cbuf(sb),r0         #R0 contains pointer
                movd  $0,r1               #R1 contains offset
                movb  $0,r2               #Init data reg.
cbufloop:       movb  r2,0(r0)[r1:b]      #Load char. to compare buffer
                addqw 1,r1                #Increment ptr. offset
                addqw 1,r2                #Decrement data
                cmpw  r1,$256             #Check for 256 chars. loaded
                bne   cbufloop            #Jump back if not done
                                          #
#*************** SET UP INTERRUPT SERVICE ROUTINE PARAMETERS ****************
                                          #
                movd $16,blk16cnt         #Initialize 16 byte block counter
```

TL/C/9313-21

4

```
                                          #
#************************* 16550A INITIALIZATION *****************************
                                          #
            movb   $0x080,lcr            #Set dlab = 1 for divisor latch access
            movb   $4,txd               #Low divisor latch 56k w/8.0 xtal
            movb   $0,ier               #Upper divisor latch
            movb   $0x003,lcr           #Dlab = 0, 8 bits, no parity, 1 stop
            movb   $0,ier               #Disable UART interrupts
            movb   $0x0c7,fcr           #Fifo=> trigger = 14, reset & enable
                                          #
#************************* INITIALIZE 32202 (ICU) ***************************
                                          #
                                          #
          movd   $icu_addr,r0           #R0 = icu address
          movb   $0xca,icu_mctl(r0)     #Set mode : 8 bit bus mode,
                                          #              freeze counters,
                                          #              disable interrupts,
                                          #              fixed priority.
          movqb 0,icu_cctl(r0)          #Halt the counters
          movqb -1,icu_ips(r0)          #Set all pins to interrupt source
          movqb 0,icu_csrc(r0)          #No cascaded interrupts (low reg)
          movqb 0,icu_csrc+a0(r0)       # (high reg)
          movb $0x10,icu_svct(r0)       #Set interrupt base vector
          movqb -1,icu_elgt(r0)         #Set level triggering (low reg)
          movqb -1,icu_elgt+a0(r0)      #(high reg)
          movqb $2,icu_tpl(r0)          #Set high polarity mode (low reg)
          movqb 0,icu_tpl+a0(r0)        #(high reg)
          movqb 0,icu_fprt(r0)          #Set highest priority to 0 (low reg)
          movqb 0,icu_fprt+a0           #(high reg)
          movqb 0,icu_isrv(r0)          #Clear intr in-service regs (low reg)
          movqb 0,icu_isrv+a0(r0)       #(high reg)
          movqb -1,icu_imsk(r0)         #Mask all intr (low reg)
          movqb -1,icu_imsk+a0(r0)      #(high reg)H
          setcfg [i]                    #Enable vectored intrp (I=1)
          movd $icu_addr,r0             #
          movb $0x02,icu_mctl(r0)       #Fixed mode, 8 bit bus mode
          movb $0x010,icu_cctl(r0)      #Set to internal sampling
          movb $0xfd,icu_imsk(r0)       #Enable irl
          movb $0xff,icu_imsk+a0(r0)    #Mask all other interrupts
          bispsrw $(0x800)              #Enable cpu intr's
                                          #
#************************* ENABLE 16550A INTERRUPTS *************************
                                          #
            movb $2,mcr                 #Clear out1, out2 and enable rts
endinit:    movb $0x07,ier              #Enable all but modem status interrupts
                                          #
#********************* ENDLESS LOOP WAITING FOR INTERRUPTS *****************
                                          #
holdloop:      nop                      #
               br holdloop              #
                                          #
#************************* INTERRUPT HANDLER ******************************
                                          #
isr:           save [r0,r1,r2,r3,r4,r5,r6,r7]
               movb iir,r0              #R0- contains iir
               cmpb r0,$0x0c6           #
               beq lsint                #Line status interrupt
               cmpb r0,$0x0c4           #
               beq rdai                 #Receiver interrupt
               cmpb r0,$0x0cc           #
```

TL/C/9313-22

```
                beq rtmout              #Rec timeout interrupt
                cmpb r0,$0x0c2          #
                beq threi              #THRE interrupt
                cmpb r0,$0x0c0          #
                beq msint              #Modem status interrupt
                                        #
                                        #
#************************** INVALID INTERRUPT ROUTINE **************************
                                        #
                save [r0,r1,r2,r3]      #
                movd $4,r0              #
                addr message2,r1        #
                movd $21,r2             #
                movd $0,r3              #
                svc                     #
                restore [r0,r1,r2,r3]   #
                                        #
                                        #
                jump stop               #Restore all registers
                                        #
                                        #
#********************* RECEIVER TIMEOUT INTERRUPT ROUTINE *********************
                                        #
rtmout:         jump rdai
                                        #
#*************************** RECEIVER INTERRUPT ROUTINE ***********************
                                        #
#This portion of the program is reached when the received data available
#interrupt is active. Once in this routine each byte removed from the FIFO
#is placed in the designated static base memory location (labelled rbuf).
#The data ready bit (DR) in the LSR is checked before each byte is removed
#from the FIFO. Data sent will be compared to known data in another designated
#static base area (labelled cbuf) by calling the compare subroutine.
                                        #
rdai:           movb $0,mcr             #Disable RTS; stop transmission
                addr rbuf(sb),r4        #r4 contains rbuf base address
                movd rbufoff,r6         #Put rbuf offset runner into r6
rdrbr:          movb rxd,0(r4)[r6:b]    #Store a byte in the receive buffer
                cmpb $0xff,0(r4)[r6:b]  #Is it the last character
                addqw 1,r6              #Increment offset ptr.
                addqw 1,rbufoff         #Track r6
                bne continue            #
                movw $0,r6              #Reset pointer offset
                movw $0,rbufoff         #Reset rbufoff
continue:       movb lsr,r3             #Read lsr
                andb $01,r3             #Mask all but bit 0
                cmpb $01,r3             #
                beq rdrbr               #Read rbr again if set
                movd r6,rbufoff         #Put result of r6 back into rbufoff
                bsr compare             #
                movb $2,mcr             #Enable rts
                jump popall             #
                                        #
#***************************** TRANSMIT ROUTINE ******************************
                                        #
#The transmitter sends data previously loaded into the static base memory area
#labelled sbuf. Thids routine sends data as 16 blocks of 16 bytes and 1 block
#of 15 bytes, continuously. NOTE: Before each block transmission occurs /CTS
#is checked to ensure that the receiver ready.
                                        #
```

TL/C/9313-23

4

```
threi:          addr abuf(sb),r0        #R0 contains base pointer
                movw xmitoff,r1         #setup xmit ptr offset
                cmpd $0,blk16cnt        #Check to see if it is the 16th block
                beq send15             #Yes, send only 15 bytes instead of 16
                movd $0x10,r7           #No, send 16 bytes
                jump sendnext          #Jump around 15 byte load
send15:         movd $0x0f,r7           #Load counter for 15 byte load
sendnext:       movb 0(r0)[r1:b],txd   #Load a byte into the transmitter
                addqw 1,r1             #
                cmpw r1,$256            #Are we one address past end of table
                beq reload             #Yes, reload ptr
finish:         save [r7]
                movb msreg,r7           #Read modem status reg
                andb $0x10,r7           #Mask all bits except CTS (MSR4)
                cmpb $0,r7              #Check for disabled CTS
                restore [r7]
                beq abort              #Leave on inactive CTS (MSR4=0)
                subb $1,r7             #No, decrement counter and continue
                cmpb $0,r7              #Is byte counter 0?
                bne sendnext           #No, send next byte
abort:          movw r1,xmitoff         #save xmit ptr offset in ram
                cmpd $0,blk16cnt        #Check to see if it is 16th block
                beq setsnd16           #Yes, reload block counter
                subb $1,blk16cnt        #Decrement block counter
                jump popall            #Finished sending 16 bytes
setsnd16:       movd $16,blk16cnt       #Reload block counter
                jump popall            #Finished sending 15 bytes
reload:         movd $0,r1             #Reset offset
                jump finish            #Go back and finish
                                        #
#*********************** LINE STATUS INTERRUPT ROUTINE ***********************
                                        #
lsint:          save [r0,r1,r2,r3]      #
                movd $4,r0             #
                addr message6,r1        #
                movd $25,r2             #
                movd $0,r3             #
                svc                    #
                restore [r0,r1,r2,r3]   #
                movb lsr,r3            #Read lsr
                jump rdai              #
                                        #
#*********************** MODEM STATUS INTERRUPT ROUTINE ***********************
                                        #
msint:          save [r0,r1,r2,r3]      #
                movd $4,r0             #
                addr message7,r1        #
                movd $26,r2             #
                movd $0,r3             #
                svc                    #
                movb 0x0d00018,r0       #
                restore [r0,r1,r2,r3]   #
                jump popall            #
#*********************** COMPARE DATA ROUTINE ***********************
                                        #
#The receiver subroutine branches to this subroutine after it has removed all of
#the data from the Rx FIFO. The receive offset (rbufoff) is changed to point to
#the last byte received in rbuf. The compare offset (compoff) points to each
#byte in the receive buffer and its associated byte in the compare register.
#Compoff is incremented after each successful comparison and the comparisons
```

TL/C/9313-24

```
#end when compoff equals rbufoff. NOTE: Data being received by this test program
#is known data and a copy of it is loaded into cbuf before transmissions begin.
                                          #
compare:        addr cbuf(sb),r1          #R1- base address of cbuf base
                cmpd $0,r6                 #Check for potential invalid subtraction
                beq zeror6                 #Jump around subtraction
                subd $1,r6                 #
                jump compbyte              #Jump around subtraction fix
zeror6:         movd $0xff,r6             #
compbyte:       movd compoff,r5           #
                cmpb 0(r1)[r5:b],0(r4)[r5:b] #Compare data sent to data received
                bne wrong                 #Branch and set out1 if wrong
                                          #
                                          #
                cmpb $0xff,0(r4)[r5:b]    #Check for end of buffer
                bne notend                #Branch and increment pointers
                jump reload1              #Test for having compared all bytes
                                          #
                                          #
notend:         addd $1,compoff           #Increment pointer
notend1:        cmpd r5,r6                #
                beq bye                   #
                jump compbyte             #
                                          #
reload1:        addd $1,sbufcnt           #Increment transmiter cnt
                movd $0,compoff           #Reload offset of pointer
                jump notend1              #
                                          #
wrong:          movb $0x0c,mcr            #Set out 2, for error strobe
                                          #
#*********************** DATA MISMATCH MESSAGE ****************************
                                          #
                save [r0,r1,r2,r3]        #Save register for supervisor call
                movd $4,r0                #Value required by svc call
                addr message8,r1          #Mover address of message into r1
                movd $17,r2               #Number of characters into r2
                movd $0,r3                #Value required by svc call
                svc                       #Actual call
                restore [r0,r1,r2,r3]     #Restore registers
                                          #
stop:           nop                       #
                jump stop                 #Test point
                                          #
bye:            ret 0                     #
                                          #
#*********************** RETURN FROM INTERRUPT ***************************
                                          #
popall:         restore [r0,r1,r2,r3,r4,r5,r6,r7]
                reti                      #
                                          #
#********************************* Messages ******************************
                                          #
                message1: .byte 13,10,"Compare Complete",13,10
                message2: .byte 13,10,"Invalid Interrupt",13,10
                message3: .byte 13,10,"Receiver Timeout",13,10
                message4: .byte 13,10,"Receive data available Interrupt",13,10
                message5: .byte 13,10,"THRE Interrupt",13,10
                message6: .byte 13,10,"Line Status Interrupt",13,10
                message7: .byte 13,10,"Modem Status Interrupt",13,10
                message8: .byte 13,10,"Data Mismatch",13,10
```

TL/C/9313–25

4

A Comparison of the INS8250, NS16450 and NS16550AF Series of UARTs

National currently produces seven versions of the INS8250 UART. Functionally, these parts appear to be the same, however, there are differences that the designer and purchaser need to understand. For each version, this document provides a brief overview of their distinct characteristics, a detailed function and timing section, a discussion of software compatibility issues and the AC timing parameters.

1.0 Part Summary

The seven versions currently produced are designated INS8250, INS8250-B, INS8250A, NS16450, INS82C50A, NS16C450, and NS16550AF. These devices are grouped below by process type.

XMOS DEVICES

1. INS8250: This is the original version produced by National. It is the same part as the INS8250-B, but with faster CPU bus timings.

2. INS8250-B: This is the slower speed (CPU bus timing) version of the INS8250. It is used by many popular 8088-based microcomputers.

3. INS8250A: This is a revision of the INS8250 using the more advanced XMOS process. The INS8250A is better than the aforementioned parts due to the redesign (compare section 2.0 to 3.0) and the following process characteristics—closer threshold voltage control, more reliably implemented process topography and finer control over the active area critical dimensions. XMOS and CMOS parts should be used for all new designs. This part is used in many popular 8086-based microcomputers.

4. NS16450: This is the faster speed (CPU bus timing) version of the INS8250A. It is used by many popular 80286-based microcomputers.

5. NS16550AF: This is the newest member of the UART family. It powers-up in the NS16450 mode and is completely compatible with all software written for the NS16450. It has advanced features such as on-board FIFOs, a DMA interface, faster CPU bus timings and a much higher maximum baud rate than the NS16450. The NS16550AF should be used for all new non-CMOS designs, including those that were originally done with the NS16550. It is used in recent versions of popular 80286-based, 80386-based and ROMP-based microcomputers. Software written for the NS16550 is completely compatible with the NS16550AF. Section 5.0 describes how the software can distinguish between the NS16550 and the NS16550AF.

6. NS16550: This part powers-up in the NS16450 mode and is completely compatible with all software written for the NS16450. It has advanced features, such as a DMA interface. The on-board FIFOs are essentially non-functional. This part was issued on a limited basis. Any user that wants this part should order the NS16550AF. Section 5.0 describes the differences between the NS16550 and the NS16550AF in detail.

CMOS DEVICES

1. INS82C50A: This is a CMOS version of the INS8250A. It functions identically and for most AC parameters has the same timing specification as the INS8250A (see Section 4.0). It draws approximately 1/10 (10 mA) of the maximum operating current of the INS8250A.

2. NS16C450: This is a CMOS version of the NS16450. It functions identically and for most AC parameters has the same timing specification as the NS16450 (see Section 4.0). It draws approximately 1/12 (10 mA) of the maximum operating current of the NS16450.

Note: The XMOS and CMOS UARTs are not plug-in replacements for the INS8250/INS8250-B when used with ICUs that are in the popular edge-triggered configuration. However, there are easily implemented adjustments to the driving software or associated hardware that will allow these parts to be a plug-in replacement (see Section 6.0).
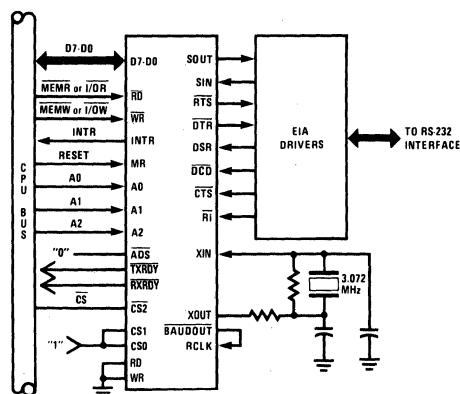
TL/C/9320–1

FIGURE 1. Connection Diagram

2.0 INS8250 and INS8250-B Functional Considerations

Designers using these parts should be aware of the following considerations.

1. When multiple interrupts are pending, the interrupt line (INTR) pulses low after each interrupt instead of remaining high continuously.

   Recommendation: This will not cause problems in normal operation, however, it is a condition necessary for compatibility in some popular 8086- and 80286-based microcomputers that use an edge-triggered ICU (see Section 6.0).

2. Bit No. 6 (TSRE) of the line status register is set as soon as the transmitter shift register empties whether or not the transmitter holding register contains a character. Bit No. 6 is then reset when the transmitter shift register is reloaded.

Recommendation: This will not cause problems in normal operation. However, it is a function tested on some popular 8088-based microcomputer systems diagnostic programs.

3. In loopback mode the modem control outputs $\overline{RTS}$, $\overline{DTR}$, $\overline{OUT1}$ and $\overline{OUT2}$ remain connected to the associated Modem Control Register bits.

3.0 INS8250A and NS16450 Function and Timing Considerations

1. The loopback diagnostic function sets the modem control outputs $\overline{RTS}$, $\overline{DTR}$, $\overline{OUT1}$ and $\overline{OUT2}$ to their inactive state (logic "1"), so they will send no spurious signals.

2. A one byte scratch pad register is included at location 111. This register is not on the INS8250 or -B.

3. When multiple interrupts are pending the interrupt line remains high rather than pulsing low after each interrupt is serviced. The INS8250A and NS16450 have level sensitive interrupts as opposed to edge-triggered interrupts. This requires a change in the UART driver software or associated hardware if the INS8250A, NS16450 is used with some popular microcomputers, and their edge-triggered ICUs (see Section 6.0).

4. Bit 6 of the line status register is set to 1 when both the transmitter holding and shift register are empty. This causes the INS8250A and NS16450 to be incompatible with some INS8250 software utilizing this bit.

3.1 TIMING CONSIDERATIONS

1. A start bit will be sent typically 16 clocks (1 bit time) after the WRTHR signal goes active.

2. The leading edge of WRTHR resets THRE and TEMT.

3. All of the line status errors and the received data flag (DR, data ready) are set during the time of the first stop bit.

4. TEMT is set 2 RCLK clock periods after the stop bit(s) are sent.

5. The modem control register updates the modem outputs on the trailing edge of WRMCR.

4.0 INS82C50A and NS16C450 Function and Timing Considerations

All of the information presented in Sections 3.0 through 3.2 is applicable to the CMOS parts. In addition, the following items specify differences between XMOS and CMOS parts. They are applicable to the CMOS parts only:

1. Anytime a reset pulse is issued to the INS82C50A or NS16C450 the divisor latches must be rewritten with the appropriate divisors in order to start the baud rate generator.

2. t_{SI} is from 16 to 48 RCLK cycles in length

5.0 NS16550AF and NS16550 Function and Timing Considerations

All of the information present in Sections 3.0 and 3.1 is applicable to the NS16550AF and NS16550.

The primary difference between these two parts is in the operation of the FIFOs. The NS16550 will sometimes transfer extra characters when the CPU reads the RX FIFO. Due to the asynchronous nature of this failure there is no workaround and the NS16550 should NOT be used in the FIFO mode. The NS16550AF has no problems operating in the FIFO mode and should be used on all new designs.

The programmer should note the difference in the function of bit 6 in the Interrupt Identification Register (IIR6). This bit is permanently at logical 0 in the NS16550. In the NS16550AF this bit will be set to a 1 when the FIFOs are enabled. In both parts bit 7 of the IIR is set to a 1 when the FIFOs are enabled. Therefore, the program can distinguish when the FIFOs are enabled and whether the part is an NS16550AF or an NS16550 by checking these two bits. In order to enable the FIFO mode and set IIR6 and IIR7 bit 0 of the FIFO Control Register (FCR0) should be set. Remember unless both bits IIR6 and IIR7 are set, the program should not transfer data via the FIFOs.

The following are improvements in the AC timings for the NS16550AF over the NS16450:

1. t_{AR} changes from 60 ns to 30 ns.

2. t_{CSW} changes from 50 ns to 30 ns.

3. t_{CSR} changes from 50 ns to 30 ns.

4. RC changes from 360 ns to 280 ns.

5. t_{RC} changes from 175 ns to 125 ns.

6. t_{DS} changes from 40 ns to 30 ns.

7. t_{DH} changes from 40 ns to 30 ns.

8. t_{AW} changes from 60 ns to 30 ns.

9. t_{WC} changes from 200 ns to 150 ns.

10. WC changes from 360 ns to 280 ns.

11. Timing parameters specified by t_{SINT} will change in some cases when the FIFOs are enabled. Refer to the data sheet for specific changes.
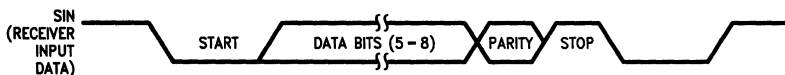
FIGURE 2. Serial Data Timing

TL/C/9320–2

6.0 Software Compatibility

Two of the conditions present in the INS8250-B are required in many of these personal computers (see Items 1 and 2 in Section 2.0). These two detect multiple pending interrupts from the INS8250-B and test the baud rate. These two conditions were eliminated in the revision part and all parts thereafter. Thus, the more recent UARTs require that one of the following recommendations or a similar change is made to the target system. Changing the software or hardware allows the more recent UARTs to replace the INS8250-B. If the target system services the UART via polling rather than interrupts, then all of the more recent parts will be plug-in replacements for the INS8250-B.

Note: The NS16550AF has two pins with new functions (see the data sheet for specifics).

6.1 USING THE INS8250A, NS16450, INS82C50A, NS16C450 AND NS16550AF WITH EDGED-TRIGGERED ICUs

Using these UARTs with an edge-triggered ICU as in some of the popular microcomputers requires a signal edge on the INTR pin for each pending UART interrupt. Otherwise, when multiple interrupts are pending the interrupt line will be constantly high active and the edge-triggered ICU will not request additional service for the UART.

6.2 CREATING AN INTERRUPT EDGE VIA SOFTWARE

This is done by disabling and then re-enabling UART interrupts via the Interrupt Enable Register (IER) before a specific UART interrupt handling routine (line status errors, received data available, transmitter holding register empty or modem status) is exited. To disable interrupts write H'00 to the IER. To re-enable interrupts write a byte containing ones to the IER bit positions whose interrupts are supposed to be enabled.

6.3 CREATING AN INTERRUPT EDGE IN HARDWARE

This is done externally to the UART. One approach is to connect the INTR pin of the UART to the input of an AND gate. The other input of this AND gate is connected to a signal that will always go low active when the UART is accessed (see *Figure 3*). The output of the AND gate is used as the interrupt to the ICU.

Note: This simple hardware recommendation will result in one invalid interrupt being generated, so the software routine should be able to handle this. The example shown below was tested using a modified asynchronous card in a few 8088-based microcomputer systems.

7.0 Acknowledgements

The editor expresses his gratitude to all of the applications, design and field applications engineers whose laboratory and field research have discovered most of the technical information used in this document.

(FROM INS8250A, NS16450 OR NS16550AF) INTR

(FROM 8288) $\overline{IOR}$
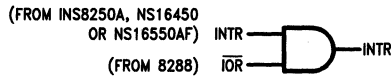
INTR

TL/C/9320–4

FIGURE 3. Creating an INTR Edge in Hardware

AC Electrical Characteristics $T_A = 0°C$ to $+70°C$, $V_{CC} = 5V \pm 5\%$

Symbol	Parameter	Conditions	NS16550AF Min	NS16550AF Max	NS16450 NS16C450 Min	NS16450 NS16C450 Max	INS8250A INS82C50A Min	INS8250A INS82C50A Max	INS8250 Min	INS8250 Max	INS8250-B Min	INS8250-B Max	Units
t_{ADS}	Address Strobe Width		60		60		90		90		120		ns
t_{AH}	Address Hold Time		0		0		0		0		60		ns
t_{AR}	RD/$\overline{RD}$ Delay from Address	(Note 1)	30		60		80		110		110		ns
t_{AS}	Address Setup Time		60		60		90		110		110		ns
t_{AW}	WR/$\overline{WR}$ Delay from Address	(Note 1)	30		60		80		160		160		ns
t_{CH}	Chip Select Hold Time		0		0		0		0		60		ns
t_{CS}	Chip Select Setup Time		60		60		90		110		110		ns
t_{CSC}	Chip Select Output Delay from Select	(Notes 1, 8)		NA		100		125		200		200	ns
t_{CSR}	RD/$\overline{RD}$ Delay from Chip Select	(Note 1)	30		50		80		110		110		ns
t_{CSS}	Chip Select Output Delay from Strobe			NA		NA		NA	0	150	0	150	ns
t_{CSW}	WR/$\overline{WR}$ Delay from Select	(Note 1)	30		50		80		160		160		ns
t_{DH}	Data Hold Time		30		40		60		60		100		ns
t_{DS}	Data Setup Time		30		40		90		175		350		ns
t_{HZ}	RD/$\overline{RD}$ to Floating Data Delay	(Notes 3, 8)	0	100	0	100	0	100	0	150	0	150	ns
t_{MR}	Master Reset Pulse Width		5		5		10		10		10		μs
t_{RA}	Address Hold Time from RD/$\overline{RD}$	(Note 1)	20		20		20		50		50		ns
t_{RC}	Read Cycle Delay		125		175		500		1735		1735		ns
t_{RCS}	Chip Select Hold Time from RD/$\overline{RD}$	(Note 1)	20		20		20		50		50		ns
t_{RD}	RD/$\overline{RD}$ Strobe Width		125		125		175		175		350		ns
t_{RDA}	RD/$\overline{RD}$ Strobe Delay from $\overline{ADS}$		NA		NA		NA		0		0		ns
t_{RDD}	RD/$\overline{RD}$ Driver Enable/Disable	(Notes 3, 8)		60		60		75		150		250	ns
t_{RVD}	Delay from RD/$\overline{RD}$ to Data	(Note 8)		125		125		175		250		300	ns
t_{WA}	Address Hold Time from WR/$\overline{WR}$	(Note 1)	20		20		20		50		50		ns
t_{WC}	Write Cycle Delay		150		200		500		1785		1785		ns

Note 1: Applicable only when $\overline{ADS}$ is tied low.

Note 3: Charge and discharge time is determined by V_{OL}, V_{OH} and the external timing.

Note 8: Loading of 100 pF.

NA = Not Applicable.

AC Electrical Characteristics T_A = 0°C to +70°C, V_{CC} = 5V ±5% (Continued)

Symbol	Parameter	Conditions	NS16550AF		NS16450 NS16C450		INS8250A INS82C50A		INS8250		INS8250-B		Units
			Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	
t_{WCS}	Chip Select Hold Time from WR/$\overline{WR}$	(Note 1)	20		20		20		50		50		ns
t_{WDA}	WR/$\overline{WR}$ Delay from Address		NA		NA		NA		50		50		ns
t_{WR}	WR/$\overline{WR}$ Strobe Width		100		100		175		175		350		ns
t_{XH}	Duration of Clock High Pulse	(Note 4)	55		140		140		140		140		ns
t_{XL}	Duration of Clock Low Pulse	(Note 4)	55		140		140		140		140		ns
RC	Read Cycle = $t_{AR} + t_{DIW} + t_{RC}$		280		360		755		2000		2205		ns
WC	Write Cycle = $t_{DDA} + t_{DOW} + t_{WC}$		280		360		755		2100		2305		ns
BAUD GENERATOR													
N	Baud Divisor		1	$2^{16}-1$	1	$2^{16}-1$	1	$2^{16}-1$	1	$2^{16}-1$	1	$2^{16}-1$	ns
t_{BHD}	Baud Output Positive Edge Delay	(Note 8)		175		175		250		250		250	ns
t_{BLD}	Baud Output Negative Edge Delay	(Note 8)		175		175		250		250		250	ns
t_{HW}	Baud Output Up Time	(Note 5)	75		250		250		330		330		ns
t_{LW}	Baud Output Down Time	(Note 6)	100		425		425		425		425		ns
RECEIVER (Note 2)													
t_{RINT}	Delay from RD/$\overline{RD}$ (RD RBR/RDLSR) to Reset Interrupt	(Note 8)		1000		1000		1000		1000		1000	ns
t_{RXI}	Delay from Read to $\overline{RXRDY}$ Inactive			290		NA		NA		NA		NA	ns
t_{SCD}	Delay from RCLK to Sample Time			2000		2000		2000		2000		2000	ns
t_{SINT}	Delay from Stop to Set Interrupt			1 RCLK		1 RCLK		1 RCLK		2000		2000	ns

Note 1: Applicable only when $\overline{ADS}$ is tied low.

Note 2: For the NS16550AF in the FIFO Mode (FCR0 = 1) the trigger level and timeout interrupts, the receiver data available indication, the active RXRDY indication and the overrun error indication will be delayed 3 RCLKs. Status indicators (PE, FE, BI) will be delayed 3 RCLKs after the first byte has been received. For subsequently received bytes these indicators will be updated immediately after RDRBR goes inactive.

Note 4: The maximum external clock for the NS16550AF is 8 MHz, NS16450 and INS8250A is 3.1 MHz and INS8250 and INS8250-B is 3.1 MHz. 100 pF load.

Note 5: The maximum external clock for the NS16550AF is 8 MHz, NS16450 and INS8250A is 3.1 MHz and INS8250 and INS8250-B is 3.1 MHz. 100 pF load. This parameter is tested on the NS16550AF and guaranteed by design on all other parts.

Note 6: The maximum external clock for the NS16550AF is 8 MHz, NS16450 and INS8250A is 2.1 MHz and INS8250 and INS8250-B is 3.1 MHz. 100 pF load. This parameter is tested on the NS16550AF and guaranteed by design on all other parts.

Note 8: Loading of 100 pF.

NA = Not Applicable.

AC Electrical Characteristics $T_A = 0°C$ to $+70°C$, $V_{CC} = 5V \pm 5\%$ (Continued)

Symbol	Parameter	Conditions	NS16550AF Min	NS16550AF Max	NS16450 NS16C450 Min	NS16450 NS16C450 Max	INS8250A INS82C50A Min	INS8250A INS82C50A Max	INS8250 Min	INS8250 Max	INS8250-B Min	INS8250-B Max	Units
TRANSMITTER													
t_{HR}	Delay from WR/$\overline{WR}$ (WR THR) to Reset Interrupt	(Note 8)		175		175		1000		1000		1000	ns
t_{IR}	Delay from RD/$\overline{RD}$ (RD IIR) to Reset Interrupt (THRE)	(Note 8)		250		250		1000		1000		1000	ns
t_{IRS}	Delay from Initial INTR Reset to Transmit Start	(Note 10)	8	24	24	40	24	40		16		16	Baudout Cycles
t_{SI}	Delay from Initial Write to Interrupt	(Notes 7, 9)	16	24	16	24	16	24		50		50	Baudout Cycles
t_{SS}	Delay from Stop to Next Start			NA		NA		NA		1000		1000	ns
t_{STI}	Delay from Stop to Interrupt (THRE)	(Note 7)	8	8	8	8	8	8		8		8	Baudout Cycles
t_{SXA}	Delay from Start to TXRDY Active	(Note 8)		8		NA		NA		NA		NA	Baudout Cycles
t_{WXI}	Delay from Write to TXRDY Inactive	(Note 8)		195		NA		NA		NA		NA	ns
MODEM CONTROL													
t_{MDO}	Delay from WR/$\overline{WR}$ (WR MCR) to Output	(Note 8)		200		200		1000		1000		1000	ns
t_{RIM}	Delay to Reset Interrupt from RD/$\overline{RD}$ (RD MSR)	(Note 8)		250		250		1000		1000		1000	ns
t_{SIM}	Delay to Set Interrupt from MODEM Input	(Note 8)		250		250		1000		1000		1000	ns

Note 7: This delay will be lengthened by 1 character time, minus the last stop bit time if the transmitter interrupt delay circuit is active.

Note 8: Loading of 100 pF.

Note 9: For both the NS16C450 and INS82C50A the value of t_{SI} will range from 16 to 48 baudout cycles.

Note 10: For both the NS16C450 and the INS82C50A the value of t_{IRS} will range from 24 to 40 baudout cycles.

NA = Not Applicable.

National Semiconductor

NS16C451
Universal Asynchronous Receiver/Transmitter
with Parallel Interface†

General Description

The NS16C451 integrates a CMOS version of the NS16450 UART with a bidirectional parallel interface into a single IC. The serial port is fully compatible with all existing software written for the INS8250A, INS82C50A, NS16450, and NS16C450. The parallel port is fully compatible with all existing software written for the IBM® PC, XT, AT, PS/2 and Centronics parallel ports.

The serial port includes one programmable baud rate generator capable of dividing the clock input by divisor of 1 to $(2^{16} - 1)$, and producing a 16 × clock for driving the internal logic of both the receiver and transmitter sections. The serial port has MODEM-control capability and a processor interrupt system which supports 4 types of interrupts.

The parallel port has three registers—data, status, and control registers and is bidirectional. All of the signals required by PC and Centronics printers to transfer data and monitor printer status are provided.

Features

- Serial port capable of running existing software written for INS8250A and NS16450 series of products used in the IBM PC, XT, AT and PS/2
- Parallel port capable of running existing software written for the standard parallel port on the IBM PC, XT, AT and Centronics printers

- National's 1.25µ CMOS technology provides faster AC timing
- Maximum operating frequency 24 MHz
- Separate interrupt request lines for the parallel and serial ports
- Separate Chip Select signals for the parallel and serial ports
- Bus Direction control output helps avoid bus conflict when using an external data bus latch
- Adds or deletes standard asynchronous communication bits (start, parity, and stop) to or from the serial data
- Independently controlled transmit, receive, line status, and data set interrupts
- Programmable baud generator divides any input clock by 1 to $(2^{16} - 1)$ and generate the 16 × clock
- MODEM control functions (CTS, RTS, DSR, DTR, RI and DCD)
- Fully programmable serial-interface characteristics:
  - 5, 6, 7, or 8 bit characters
  - Even, odd, or no parity generation and detection
  - 1, 1½, or 2 stop bit generation
- High current drive capability for the parallel port

†**Note:** This part is patented.

Basic Configuration

TL/C/10428-1

National Semiconductor
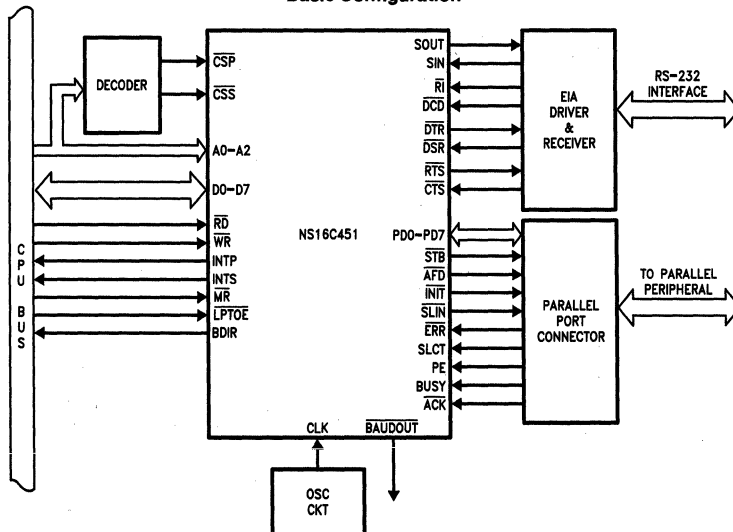
NS16C551 Universal Asynchronous Receiver/Transmitter with FIFOs, Parallel Interface and Decode Logic†

General Description

The NS16C551 integrates a CMOS version of the NS16550AF UART with a bidirectional parallel interface and an on-chip address decoder into a single IC. The UART is compatible with all existing software written for the INS8250A, NS16450, INS82C50A, NS16C450 and NS16550AF. The parallel port is compatible with all existing software written for the IBM® PC®, XT®, AT®, PS/2® and Centronics parallel ports. Chip selection can be done through an on-chip decoder to reduce the external hardware required when intefacing the NS16C551 with an IBM AT or compatible I/O map. The improved AC timings ensure compatibility with state-of-the-art CPUs.

The UART can operate with on-chip transmitter and receiver FIFOs (FIFO mode) to relieve the CPU of excessive software overhead. In FIFO mode each channel is capable of buffering 16 bytes (plus 3 bits of error data per byte in the RCVR FIFO) of data in both the transmitter and receiver. All the FIFO control logic is on-chip to minimize system overhead and maximize system efficiency.

Signalling for DMA transfers is done through two pins per channel ($\overline{\text{TXRDY}}$ and $\overline{\text{RXRDY}}$). The $\overline{\text{RXRDY}}$ function is multiplexed on one pin with the $\overline{\text{OUT 2}}$ and BAUDOUT functions. The CPU can select these functions through a new UART register (Alternate Function Register).

The UART includes one programmable baud rate generator capable of dividing the clock input by divisors of 1 to (2^{16} − 1), and producing a 16 × clock for driving the internal logic of both the receiver and transmitter sections. The UART has complete MODEM-control capability, and a processor-interrupt system.

The parallel port has three registers, two of which provide status and control for the data register. The CPU can transfer data through this register in both directions by control of the $\overline{\text{POS}}$ Mode Pin, a bit in the Control register and the $\overline{\text{RD}}$ $\overline{\text{WR}}$ signals. All of the signals required by PC and Centronics printers to transfer data and monitor printer status are provided. On-Chip buffers meet or exceed drive current requirements of the PS/2 systems.

The on-chip decode logic can be used as an alternate to the chip select and channel select pins. When the NS16C551 is mapped to the same addresses as COM1, COM2, LPT1, LPT2 and LPT3 on the AT bus, the decode logic will sense these addresses and enable the appropriate serial or parallel port.

The NS16C551 is fabricated using National Semiconductor's advanced M2CMOS™.

Features

- UART capable of interfacing with existing INS8250A, NS16450, INS82C50A, NS16C450 and NS16550AF software
- Capable of interfacing with all PC, PS/2 and Centronics parallel port software
- High current drivers that meet or exceed all Micro Channel and PS/2 parallel port drive current requirements
- Provides all control and status pins for a complete PC, AT, PS/2, Micro Channel, and Centronics parallel port interface
- Monitors all signals necessary to decode standard COM1, COM2, LPT1, LPT2 and LPT3 addresses on the PC AT bus
- Read and Write cycle times of 84 ns
- After reset, all UART registers are identical to the 16450 register set
- In the FIFO mode transmitter and receiver are each buffered with 16-byte FIFOs to reduce the number of interrupts presented to the CPU
- Adds or deletes standard asynchronous communication bits (start, stop, and parity) to or from the serial data
- Independently controlled transmit, receive, line status, and data set interrupts
- Programmable baud generator divide any input clock by 1 to (2^{16} − 1) and generate the 16 × clock
- MODEM control functions (CTS, RTS, DSR, DTR, RI, and DCD)
- Fully programmable serial-interface characteristics:
  - 5-, 6-, 7-, or 8-bit characters
  - Even, odd, or no-parity bit generation and detection
  - 1-, 1½-, or 2-stop bit generation
  - Baud generation (DC to 1.5M baud) with 16 × clock
- False start bit detection
- Line break generation and detection
- Loopback controls for communications link fault isolation
- Break, parity, overrun, framing error simulation
- Full prioritized interrupt system controls

†Note: This part is patented.

4

![National Semiconductor logo]

NS16C552 Dual Universal Asynchronous Receiver/Transmitter with FIFOs†

General Description

The NS16C552 is a dual version of the NS16550AF Universal Asynchronous Receiver/Transmitter (UART). The two serial channels are completely independent except for a common CPU interface and crystal input. On power-up both channels are functionally identical to the NS16450*. Each channel can operate with on-chip transmitter and receiver FIFOs (FIFO mode) to relieve the CPU of excessive software overhead. In FIFO mode each channel is capable of buffering 16 bytes (plus 3 bits of error data per byte in the RCVR FIFO) of data in both the transmitter and receiver. All the FIFO control logic is on-chip to minimize system overhead and maximize system efficiency.

Signalling for DMA transfers is done through two pins per channel ($\overline{\text{TXRDY}}$ and $\overline{\text{RXRDY}}$). The $\overline{\text{RXRDY}}$ function is multiplexed on one pin with the $\overline{\text{OUT 2}}$ and BAUDOUT functions. The CPU can select these functions through a new register (Alternate Function Register).

Each channel performs serial-to-parallel conversion on data characters received from a peripheral device or a MODEM, and parallel-to-serial conversion on data characters received from the CPU. The CPU can read the complete status of each channel at any time. Status information reported includes the type and condition of the transfer operations being performed by the DUART, as well as any error conditions (parity, overrun, framing, or break interrupt).

The DUART includes one programmable baud rate generator for each channel. Each is capable of dividing the clock input by divisors of 1 to ($2^{16} - 1$), and producing a 16 × clock for driving the internal transmitter logic. Provisions are also included to use this 16 × clock to drive the receiver logic. The DUART has complete MODEM-control capability, and a processor-interrupt system. Interrupts can be programmed to the user's requirements, minimizing the computing required to handle the communications link.

The DUART is fabricated using National Semiconductor's advanced M2CMOS™.

Features

■ Dual independent UARTs
■ Capable of running all existing NS16450 and NS16550AF software
■ After reset, all registers are identical to the 16450 register set
■ Read and write cycle times of 84 ns
■ In the FIFO mode transmitter and receiver are each buffered with 16-byte FIFOs to reduce the number of interrupts presented to the CPU
■ Holding and shift registers in the NS16450 Mode eliminate the need for precise synchronization between the CPU and serial data
■ Adds or deletes standard asynchronous communication bits (start, stop, and parity) to or from the serial data
■ Independently controlled transmit, receive, line status, and data set interrupts
■ Programmable baud generators divide any input clock by 1 to ($2^{16} - 1$) and generate the 16 × clock
■ MODEM control functions (CTS, RTS, DSR, DTR, RI, and DCD)
■ Fully programmable serial-interface characteristics:
  — 5-, 6-, 7-, or 8-bit characters
  — Even, odd, or no-parity bit generation and detection
  — 1-, 1½-, or 2-stop bit generation
  — Baud generation (DC to 1.5M baud) with 16 × clock
■ False start bit detection
■ Complete status reporting capabilities
■ TRI-STATE® TTL drive for the data and control buses
■ Line break generation and detection
■ Internal diagnostic capabilities:
  — Loopback controls for communications link fault isolation
  — Break, parity, overrun, framing error simulation
■ Full prioritized interrupt system controls

*Can also be reset to NS16450 Mode under software control.
†Note: This part is patented.

Table of Contents

Basic Configuration

TL/C/9426–1

1.0 Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Temperature under Bias	0°C to +70°C
Storage Temperature	−65°C to +150°C
All Input or Output Voltages with Respect to V_{SS}	−0.5V to +7.0V
Power Dissipation	1W

Note: *Maximum ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended and should be limited to those conditions specified under DC electrical characteristics.*

2.0 DC Electrical Characteristics

T_A = 0°C to +70°C, V_{DD} = +5V ±10%, V_{SS} = 0V, unless otherwise specified

Symbol	Parameter	Conditions	Min	Max	Units
V_{ILX}	Clock Input Low Voltage		−0.5	0.8	V
V_{IHX}	Clock Input High Voltage		2	V_{CC}	V
V_{IL}	Input Low Voltage		−0.5	0.8	V
V_{IH}	Input High Voltage		2	V_{CC}	V
V_{OL}	Output Low Voltage	I_{OL} = 1.6 mA on all (Note 1)		0.4	V
V_{OH}	Output High Voltage	I_{OH} = −1 mA (Note 1)	2.4		V
$I_{CC}(AV)$	Average Power Supply Current	V_{DD} = 5.5V No Loads on Output; CS, RD, WR, SIN, DSR, DCD, CTS, RI = 2V All Other Inputs = 0.8V XIN = 24 MHz Divisor = EFFF		30	mA
I_{IL}	Input Leakage	V_{DD} = 5.5V, V_{SS} = 0V V_{IN} = 0V, 5.5V		±10	μA
I_{CL}	Clock Leakage			±10	μA
I_{OZ}	TRI-STATE Leakage	V_{DD} = 5.5V, V_{SS} = 0V V_{OUT} = 0V, 5.5V 1) Chip Deselected 2) WRITE Mode, Chip Selected		±20	μA
V_{ILMR}	MR Schmitt V_{IL}			0.8	V
V_{IHMR}	MR Schmitt V_{IH}		2		V

Note 1: Does not apply to XOUT

Note 2: T_A = 25°C

Capacitance T_A = 25°C, V_{DD} = V_{SS} = 0V

Symbol	Parameter	Conditions	Min	Typ	Max	Units
C_{XIN}	Clock Input Capacitance	f_c = 1 MHz Unmeasured Pins Returned to V_{SS}		7	9	pF
C_{XOUT}	Clock Output Capacitance			7	9	pF
C_{IN}	Input Capacitance			5	7	pF
C_{OUT}	Output Capacitance			6	8	pF
$C_{I/O}$	Input/Output Capacitance			10	12	pF

3.0 AC Electrical Characteristics $T_A = 0°C$ to $+70°C$, $V_{DD} = +5V \pm 10\%$

Symbol	Parameter	Conditions	Min	Max	Units
t_{AR}	$\overline{RD}$ Delay from Address		15		ns
t_{AW}	$\overline{WR}$ Delay from Address		15		ns
t_{DH}	Data Hold Time		5		ns
t_{DS}	Data Setup Time		15		ns
t_{HZ}	$\overline{RD}$ to Floating Data Delay	(Note 2)	10	20	ns
t_{MR}	Master Reset Pulse Width		500		ns
t_{RA}	Address Hold Time from $\overline{RD}$		0		ns
t_{RC}	Read Cycle Update		29		ns
t_{RD}	$\overline{RD}$ Strobe Width		40		ns
t_{RVD}	Delay from $\overline{RD}$ to Data			25	ns
t_{WA}	Address Hold Time from $\overline{WR}$		0		ns
t_{WC}	Write Cycle Update		29		ns
t_{WR}	$\overline{WR}$ Strobe Width		40		ns
t_{XH}	Duration of Clock High Pulse	External Clock (24 MHz Max)	17		ns
t_{XL}	Duration of Clock Low Pulse	External Clock (24 MHz Max)	17		ns
RC	Read Cycle = $t_{AR} + t_{RD} + t_{RC}$		84		ns
WC	Write Cycle = $t_{AW} + t_{WR} + t_{WC}$		84		ns
BAUD GENERATOR					
N	Baud Divisor		1	$2^{16} - 1$	
t_{BHD}	Baud Output Positive Edge Delay	$f_X = 24$ MHz, $\div 2$		45	ns
t_{BLD}	Baud Output Negative Edge Delay	$f_X = 24$ MHz, $\div 2$		45	ns
RECEIVER					
t_{RAI}	Delay from Active Edge of $\overline{RD}$ to Reset Interrupt			78	ns
t_{RINT}	Delay from Inactive Edge of $\overline{RD}$ (RD LSR) to Reset Interrupt			40	ns
t_{RXI}	Delay from READ to RXRDY Inactive			78	ns
t_{SCD}	Delay from RCLK to Sample Time			33	ns
t_{SINT}	Delay from Stop to Set Interrupt	(Note 1)		2	BAUDOUT Cycles

Note 1: In the FIFO mode (FCR0 = 1) the trigger level interrupts, the receiver data available indication, the active RXRDY indication and the overrun error indication will be delayed 3 RCLKs. Status indicators (PE, FE, BI) will be delayed 3 RCLKs after the first byte has been received. For subsequently received bytes these indicators will be updated immediately after RDRBR goes inactive. Timeout interrupt is delayed 8 RCLKs.

Note 2: Charge and discharge time is determined by V_{OL}, V_{OH} and the external loading.

Note 3: All AC timings can be met with current loads that don't exceed 3.2 mA or -80 μA at 100 pF capacitive loading.

Note 4: For capacitive loads that exceed 100 pF the following typical derating factors should be used:

100 pF < C_L ≤ 150 pF t = (0.1 ns/pF)(C_L − 100 pF)

150 pF < C_L ≤ 200 pF t = (0.08 ns/pF)(C_L − 100 pF)

I_{SINK} t = (0.5 ns/mA)(I_{SINK} mA)

I_{SOURCE} t = (0.5 ns/mA)(I_{SOURCE} mA)

Limits: I_{SOURCE} is negative, I_{SINK} ≤ 4.8 mA, I_{SOURCE} ≤ -120 μA, C_L ≤ 250 pF

AC Testing Load Circuit

DEVICE UNDER TEST — 100 pF

TL/C/9426–22

4
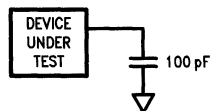
3.0 AC Electrical Characteristics $T_A = 0°C$ to $+70°C$, $V_{DD} = +5V \pm 10\%$ (Continued)

Symbol	Parameter	Conditions	Min	Max	Units
TRANSMITTER					
t_{HR}	Delay from $\overline{WR}$ ($\overline{WR}$ THR) to Reset Interrupt			40	ns
t_{IR}	Delay from $\overline{RD}$ ($\overline{RD}$ IIR) to Reset Interrupt (THRE)			40	ns
t_{IRS}	Delay from Initial INTR Reset to Transmit Start		8	24	BAUDOUT Cycles
t_{SI}	Delay from Initial Write to Interrupt	(Note 1)	16	24	BAUDOUT Cycles
t_{STI}	Delay from Start to Interrupt (THRE)	(Note 1)		8	BAUDOUT Cycles
t_{SXA}	Delay from Start to TXRDY Active			8	BAUDOUT Cycles
t_{WXI}	Delay from Write to TXRDY Inactive			60	ns
MODEM CONTROL					
t_{MDO}	Delay from $\overline{WR}$ ($\overline{WR}$ MCR) to Output			40	ns
t_{RIM}	Delay to Reset Interrupt from $\overline{RD}$ ($\overline{RD}$ MSR)			78	ns
t_{SIM}	Delay to Set Interrupt from MODEM Input			40	ns

Note 1: This delay will be lengthened by 1 character time, minus the last stop bit time if the transmitter interrupt delay circuit is active. (See FIFO Interrupt Mode Operation).

4.0 Timing Waveforms All timings are referenced to valid 0 and valid 1

External Clock Input (24 MHz Max)

TL/C/9426–2

AC Test Points
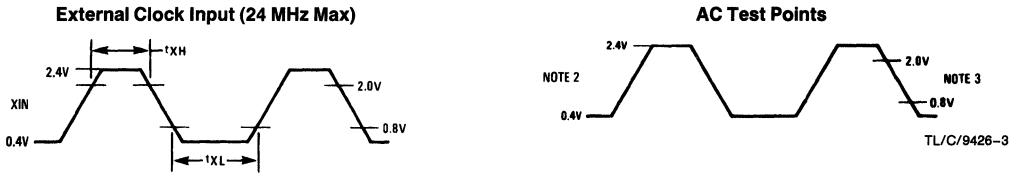
TL/C/9426–3

Note 2: The 2.4V and 0.4V levels are the voltages that the inputs are driven to during AC testing.

Note 3: The 2.0V and 0.8V levels are the voltages at which the timing tests are made.

$\overline{\text{BAUDOUT}}$ Timing
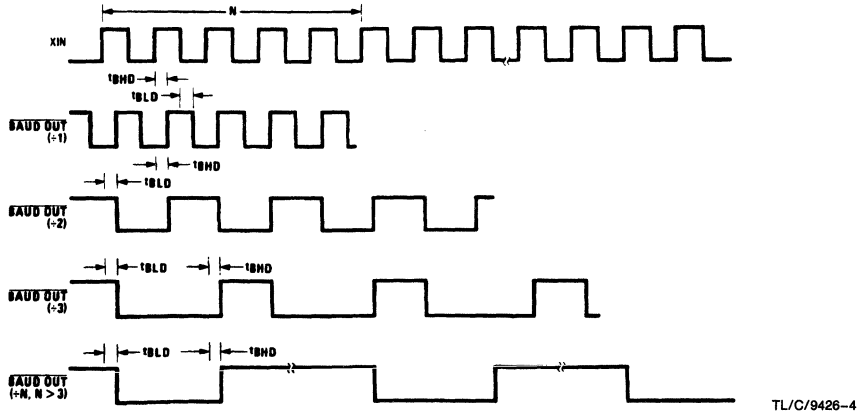
TL/C/9426–4

4.0 Timing Waveforms All timings are referenced to valid 0 and valid 1 (Continued)

Read Cycle
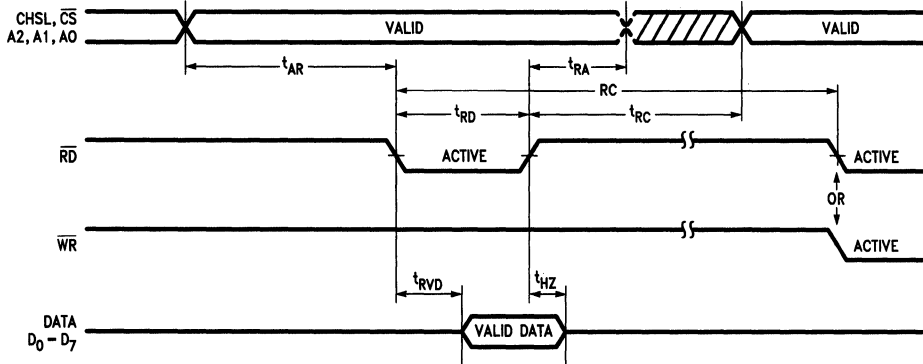
TL/C/9426-6

Write Cycle
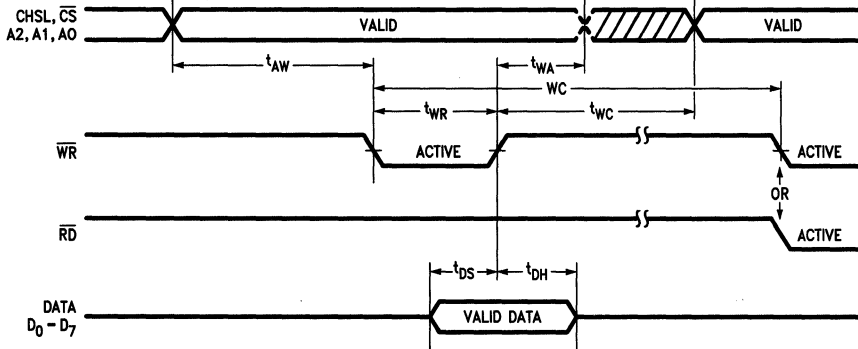
TL/C/9426-5

Transmitter Timing
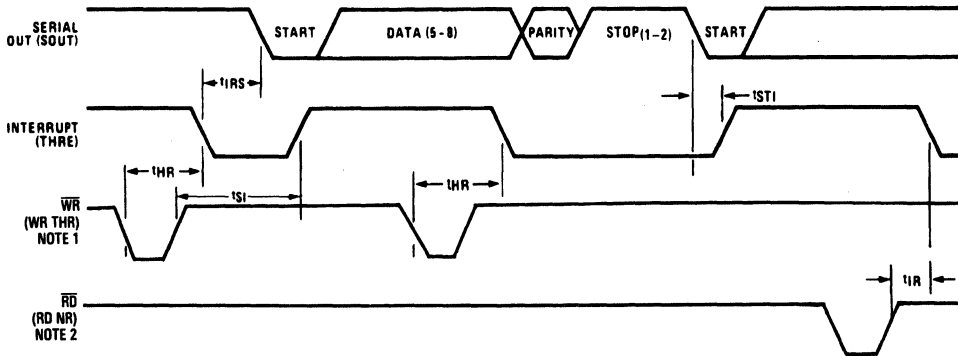
TL/C/9426-8

Note 1: See Write Cycle Timing.

Note 2: See Read Cycle Timing.

4.0 Timing Waveforms All timings are referenced to valid 0 and valid 1 (Continued)

Receiver Timing

RCLK

SAMPLE CLK

8 CLKS

t_{SCD}

NS16450 MODE:

SIN — DATA (5-8) — STOP

SAMPLE CLOCK

RDR INTERRUPT

t_{SINT}

t_{RAI}

LSI INTERRUPT

t_{RINT}

$\overline{RD}$ (RDRBR)

ACTIVE

$\overline{RD}$ (RDLSR)

ACTIVE

TL/C/9426-7

MODEM Control Timing

$\overline{WR}$ (WR MCR) NOTE 1

t_{MD0}

t_{MD0}

$\overline{RTS}, \overline{DTR},$ $\overline{OUT\ 2}$

$\overline{CTS}, \overline{DSR}, \overline{DCD}$

INTERRUPT

t_{SIM}

t_{RIM}

t_{SIM}

t_{RIM}

t_{SIM}

$\overline{RD}$ (RD MSR) NOTE 2

$\overline{RI}$
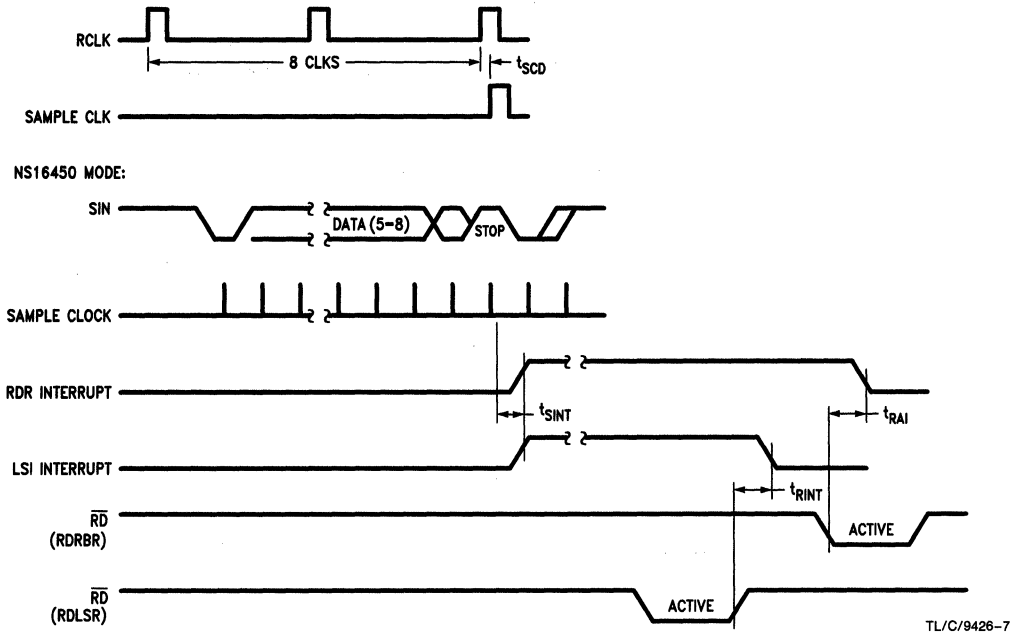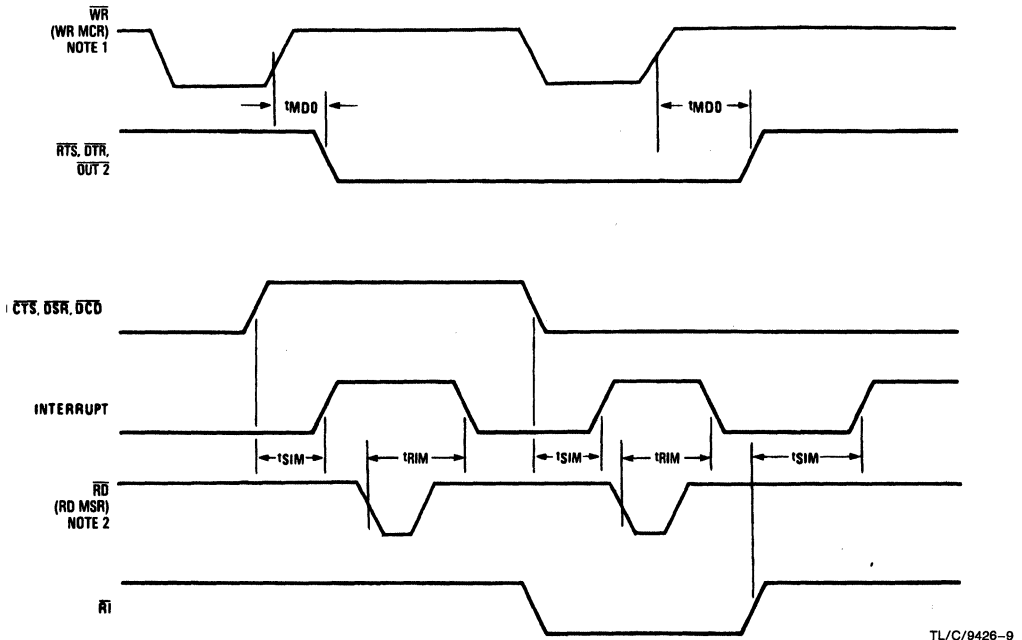
TL/C/9426-9

Note 1: See Write Cycle Timing.

Note 2: See Read Cycle Timing.

4.0 Timing Waveforms All timings are referenced to valid 0 and valid 1 (Continued)

RCVR FIFO First Byte (This Sets RDR)
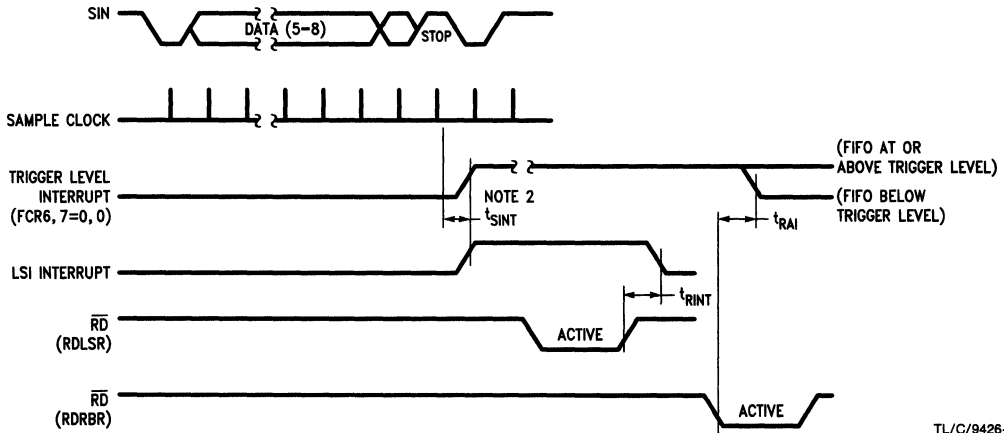
TL/C/9426–10

RCVR FIFO Bytes Other Than the First Byte (RDR Is Already Set)
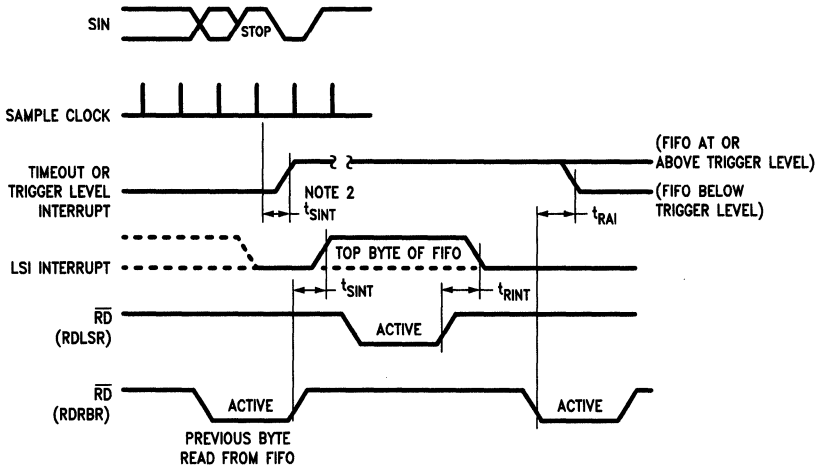
TL/C/9426–11

Receiver Ready FCR0 = 0 or FCR0 = 1 and FCR3 = 0 (Mode 0)
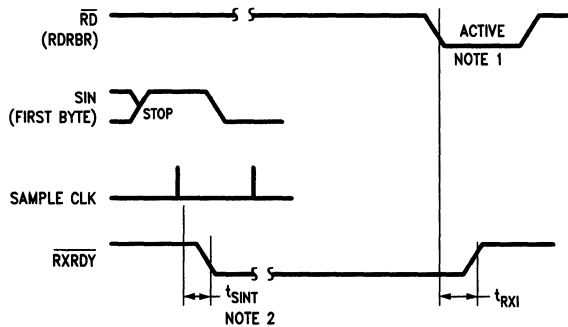
TL/C/9426–12

Note 1: This is the reading of the last byte in the FIFO.

Note 2: If FCR0 = 1, then t_{SINT} = 3 RCLKs. For a timeout interrupt, t_{SINT} = 8 RCLKs.

4.0 Timing Waveforms All timings are referenced to valid 0 and valid 1 (Continued)

Receiver Ready FCR0 = 1 and FCR3 = 1 (Mode 1)

$\overline{RD}$
(RDRBR)

ACTIVE
NOTE 1

SIN
(FIRST BYTE THAT
REACHES THE
TRIGGER LEVEL)

STOP

SAMPLE CLK

$\overline{RXRDY}$

t_{SINT}
NOTE 2

t_{RXI}
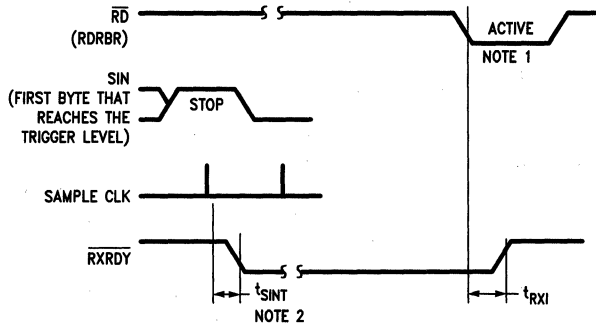
TL/C/9426–13

Note 1: This is the reading of the last byte in the FIFO.

Note 2: If FCR0 = 1, t_{SINT} = 3 RCLKs.

Transmitter Ready FCR0 = 0 or FCR0 = 1 and FCR3 = 0 (Mode 0)

$\overline{WR}$
(WRTHR)

BYTE #1

SOUT   DATA   PARITY   STOP   START

$\overline{TXRDY}$

t_{WXI}

t_{SXA}
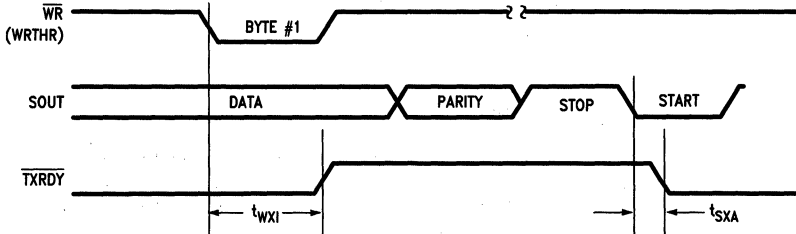
TL/C/9426–14

Transmitter Ready FCR0 = 1 and FCR3 = 1 (Mode 1)

$\overline{WR}$
(WRTHR)

BYTE #16

START OF
BYTE
#16

SOUT   DATA   PARITY   STOP   START

$\overline{TXRDY}$

FIFO FULL

t_{WXI}

t_{SXA}
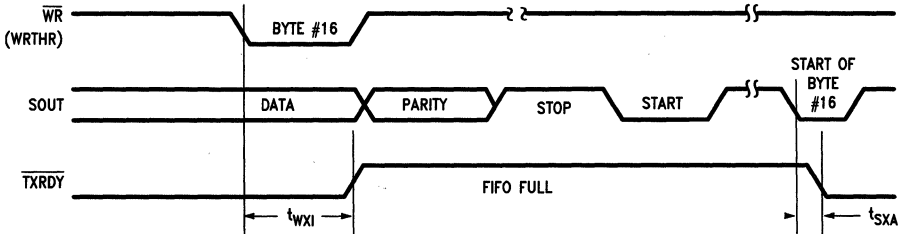
TL/C/9426–15

5.0 Block Diagram of a Single Channel

TL/C/9426–16

4
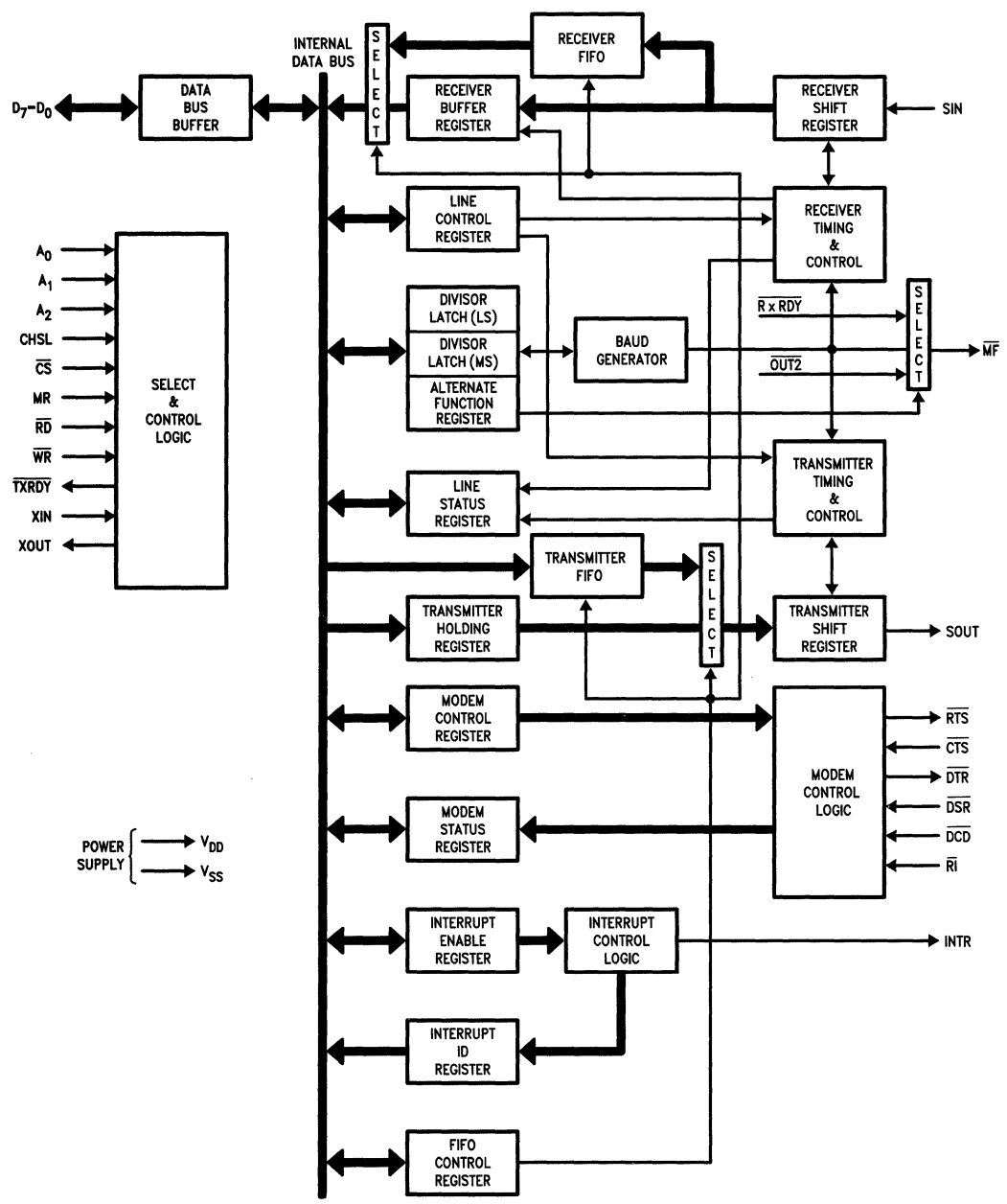
6.0 Pin Descriptions

The following describes the function of all DUART pins. Some of these descriptions reference internal circuits.

In the following descriptions, a low represents a logic 0 (0V nominal) and a high represents a logic 1 (+2.4V nominal).

Serial channels are designated by a numerical suffix (1 or 2) after each pin name. If a numerical suffix is **not** associated with the pin name, then the information applies to both channels.

A0, A1, A2 (Register Select), pins 10, 14, 15: Address signals connected to these 3 inputs select a DUART register for the CPU to read from or write to during data transfer. Table I shows the registers and their addresses. Note that the state of the Divisor Latch Access Bit (DLAB), which is the most significant bit of the Line Control Register, affects the selection of certain DUART registers. The DLAB must be set high by the system software to access the Baud Generator Divisor Latches and the Alternate Function Register.

CHSL (Channel Select), pin 16: This directs the address and data information to the selected serial channel. When CHSL is high, channel 1 is selected. When CHSL is low channel 2 is selected.

$\overline{CS}$ (Chip Select), pin 18: When $\overline{CS}$ is low, the chip is selected. This enables communication between the DUART and the CPU. Valid chip selects should stabilize according to the t_{AW} parameter.

$\overline{CTS1}$, $\overline{CTS2}$ (Clear to Send), pins 40, 28: When low, this indicates that the MODEM or data set is ready to exchange data. The $\overline{CTS}$ signal is a MODEM status input whose condition the CPU can test by reading bit 4 (CTS) of the MODEM Status Register for the appropriate channel. Bit 4 is the complement of the $\overline{CTS}$ signal. Bit 0 (DCTS) of the MODEM Status Register indicates whether the $\overline{CTS}$ input has changed state since the previous reading of the MODEM Status Register. $\overline{CTS}$ has no effect on the Transmitter.

Note: Whenever the CTS bit of the MODEM Status Register changes state, an interrupt is generated if the MODEM Status Interrupt is enabled.

D_7–D_0 (Data Bus), pins 2–9: This bus comprises eight TRI-STATE input/output lines. The bus provides bidirectional communications between the UART and the CPU. Data, control words, and status information are transferred via the D_7–D_0 Data Bus.

$\overline{DCD1}$, $\overline{DCD2}$ (Data Carrier Detect), pins 42, 30: When low, indicates that the data carrier has been detected by the MODEM or data set. The $\overline{DCD}$ signal is a MODEM status input whose condition the CPU can test by reading bit 7 (DCD) of the MODEM Status Register for the appropriate channel. Bit 7 is the complement of the $\overline{DCD}$ signal. Bit 3 (DDCD) of the MODEM Status Register indicates whether the $\overline{DCD}$ input has changed state since the previous reading of the MODEM Status Register. $\overline{DCD}$ has no effect on the receiver.

Note: Whenever the DCD bit of the MODEM Status Register changes state, an interrupt is generated if the MODEM Status Interrupt is enabled.

$\overline{DSR1}$, $\overline{DSR2}$ (Data Set Ready), pins 41, 29: When low, this indicates that the MODEM or data set is ready to establish the communications link with the DUART. The $\overline{DSR}$ signal is a MODEM status input whose condition the CPU can test by reading bit 5 (DSR) of the MODEM Status Register for the

appropriate channel. Bit 5 is the complement of the $\overline{DSR}$ signal. Bit 1 (DDSR) of the MODEM Status Register indicates whether the $\overline{DSR}$ input has changed state since the previous reading of the MODEM Status Register.

Note: Whenever the DSR bit of the MODEM Status Register changes state, an interrupt is generated if the MODEM Status Interrupt is enabled.

$\overline{DTR1}$, $\overline{DTR2}$ (Data Terminal Ready), pins 37, 27: When low, this informs the MODEM or data set that the DUART is ready to establish a communications link. The $\overline{DTR}$ output signal can be set to an active low by programming bit 0 (DTR) of the MODEM Control Register to a high level. A Master Reset operation sets this signal to its inactive (high) state. Loop mode operation holds this signal in its inactive state.

INTR1, INTR2 (Interrupt), pins 34, 17: This goes high whenever any one of the following interrupt types has an active high condition and is enabled via the IER: Receiver Error Flag; Received Data Available: timeout (FIFO Mode only); Transmitter Holding Register Empty; and MODEM Status. The INTR signal is reset low upon the appropriate interrupt service or a Master Reset operation.

$\overline{MF1}$, $\overline{MF2}$ (Multi-Function), pins 35, 19: This can be programmed for any one of three signal functions $\overline{OUT\ 2}$, $\overline{BAUDOUT}$ or $\overline{RXRDY}$. Bits 2 and 1 of the Alternate Function Register select which output signal will be present on this pin. $\overline{OUT\ 2}$ is the default signal and it is selected immediately after master reset or power-up.

The $\overline{OUT\ 2}$ signal can be set active low by programming bit 3 (OUT 2) of the associated channel's MODEM Control Register to a 1. A Master Reset operation sets this signal to its inactive (high) state. Loop Mode holds this signal in its inactive state.

The $\overline{BAUDOUT}$ signal is the 16 × clock output that drives the transmitter and receiver logic of the associated serial channel. This signal is the result of the XIN clock divided by the value in the Division Latch Registers. The $\overline{BAUDOUT}$ signal for each channel is internally connected to provide the receiver clock (formerly RCLK on the NS16550AF).

The $\overline{RXRDY}$ signal can be used to request a DMA transfer of data from the RCVR FIFO. Details regarding the active and inactive states of this signal are given in Section 8.5, Bit 3.

MR (Master Reset), pin 21: When this input is high, it clears all the registers (except the Receiver Buffer, Transmitter Holding, and Divisor Latches), and the control logic of the DUART. The states of various output signals (SOUT, INTR, $\overline{OUT\ 2}$, $\overline{RTS}$, $\overline{DTR}$) are affected by an active MR input (Refer to Table III.) This input is buffered with a TTL-compatible Schmitt Trigger.

$\overline{RD}$ (Read), pin 24: When $\overline{RD}$ is low while the chip is selected, the CPU can read status information or data from the selected DUART register.

$\overline{RTS1}$, $\overline{RTS2}$ (Request to Send), pins 36, 23: When low, this informs the MODEM or data set that the UART is ready to exchange data. The $\overline{RTS}$ output signal can be set to an active low by programming bit 1 (RTS) of the MODEM Control Register. A Master Reset operation sets this signal to its inactive (high) state. Loop mode operation holds this signal in its inactive state.

6.0 Pin Descriptions (Continued)

$\overline{RI1}$, $\overline{RI2}$ (Ring Indicator), pins 43, 31: When low, this indicates that a telephone ringing signal has been received by the MODEM or data set. The $\overline{RI}$ signal is a MODEM status input whose condition the CPU can test by reading bit 6 (RI) of the MODEM Status Register for the appropriate channel. Bit 6 is the complement of the $\overline{RI}$ signal. Bit 2 (TERI) of the MODEM Status Register indicates whether the $\overline{RI}$ input **signal** has changed from a low to a high state since the previous reading of the MODEM Status Register.

Note: Whenever the RI **bit** of the MODEM Status Register changes from a high to a low state, an interrupt is generated if the MODEM Status Interrupt is enabled.

SIN1, SIN2 (Serial Input), pins 39, 25 : Serial data input from the communications link (peripheral device, MODEM, or data set).

SOUT1, SOUT2 (Serial Output), pins 38, 26: Composite serial data output to the communications link (peripheral, MODEM or data set). The SOUT signal is set to the Marking (logic 1) state upon a Master Reset operation.

$\overline{TXRDY1}$, $\overline{TXRDY2}$ (Transmitter Ready), pins 1, 32: Transmitter DMA signalling is available through two pins. When operating in the FIFO mode, the CPU selects one of two types of DMA transfer via FCR3. When operating as in the NS16450 Mode, only DMA mode 0 is allowed. Mode 0 supports single transfer DMA where a transfer is made between CPU bus cycles. Mode 1 supports multi-transfer DMA where multiple transfers are made continuously until the XMIT FIFO has been filled. Details regarding the active and inactive states of this signal are given in Section 8.5, Bit 3.

V_{DD} (Power), pins 33, 44: +5V Supply

V_{SS} (Ground), pins 12, 22: 0V Reference

$\overline{WR}$ (Write), pin 20: When $\overline{WR}$ is low while the chip is selected, the CPU can write control words or data into the selected DUART register.

XIN (External Crystal Input), pin 11: This signal input is used in conjunction with XOUT to form a feedback circuit for the baud rate generator's oscillator. If a clock signal will be generated off-chip, then it should drive the baud rate generator through this pin.

XOUT (External Crystal Output), pin 13: This signal output is used in conjunction with XIN to form a feedback circuit for the baud rate generator's oscillator. If the clock signal will be generated off-chip, then this pin is unused.

7.0 Connection Diagram

Chip Carrier Package
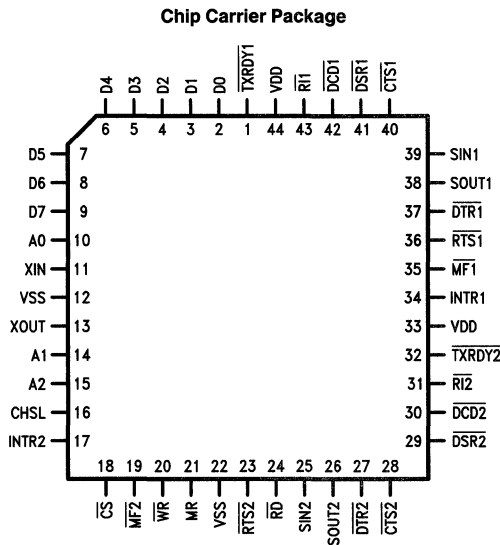
TL/C/9426–17

Top View

Order Number NS16C552V
See NS Package Number V44A

4

8.0 Registers

TABLE I: Register Addresses

DLAB1	CHSL	A₂	A₁	A₀	Register	
0	1	0	0	0	Receiver Buffer (Read), Transmitter Holding Register (Write)	
0	1	0	0	1	Interrupt Enable	C
0	1	0	1	0	Interrupt Identification (Read)	H
0	1	0	1	0	FIFO Control (Write)	A
X	1	0	1	1	Line Control	N
X	1	1	0	0	MODEM Control	N
X	1	1	0	1	Line Status	E
X	1	1	1	0	MODEM Status	L
X	1	1	1	1	Scratch	
1	1	0	0	0	Divisor Latch (Least Significant Byte)	1
1	1	0	0	1	Divisor Latch (Most Significant Byte)	
1	1	0	1	0	Alternate Function	

DLAB2	CHSL	A₂	A₁	A₀	Register	
0	0	0	0	0	Receiver Buffer (Read), Transmitter Holding Register (Write)	
0	0	0	0	1	Interrupt Enable	C
0	0	0	1	0	Interrupt Identification (Read)	H
0	0	0	1	0	FIFO Control (Write)	A
X	0	0	1	1	Line Control	N
X	0	1	0	0	MODEM Control	N
X	0	1	0	1	Line Status	E
X	0	1	1	0	MODEM Status	L
X	0	1	1	1	Scratch	
1	0	0	0	0	Divisor Latch (Least Significant Byte)	2
1	0	0	0	1	Divisor Latch (Most Significant Byte)	
1	0	0	1	0	Alternate Function	

TABLE II. Register Summary for an Individual Channel													
Register Address													
	0 DLAB=0	0 DLAB=0	1 DLAB=0	2	2	3	4	5	6	7	0 DLAB=1	1 DLAB=1	2 DLAB=1
Bit No.	Receiver Buffer Register (Read Only)	Transmitter Holding Register (Write Only)	Interrupt Enable Register	Interrupt Ident. Register (Read Only)	FIFO Control Register (Write Only)	Line Control Register	MODEM Control Register	Line Status Register	MODEM Status Register	Scratch Register	Divisor Latch (LS)	Divisor Latch (MS)	Alternate Function Register
	RBR	THR	IER	IIR	FCR	LCR	MCR	LSR	MSR	SCR	DLL	DLM	AFR
0	Data Bit 0 (Note 1)	Data Bit 0	Enable Received Data Available Interrupt (ERDAI)	"0" if Interrupt Pending	FIFO Enable	Word Length Select Bit 0 (WLS0)	Data Terminal Ready (DTR)	Data Ready (DR)	Delta Clear to Send (DCTS)	Bit 0	Bit 0	Bit 8	Concurrent Write
1	Data Bit 1	Data Bit 1	Enable Transmitter Holding Register Empty Interrupt (ETHREI)	Interrupt ID Bit	RCVR FIFO Reset	Word Length Select Bit 1 (WLS1)	Request to Send (RTS)	Overrun Error (OE)	Delta Data Set Ready (DDSR)	Bit 1	Bit 1	Bit 9	BAUDOUT Select
2	Data Bit 2	Data Bit 2	Enable Receiver Line Status Interrupt (ELSI)	Interrupt ID Bit	XMIT FIFO Reset	Number of Stop Bits (STB)	Out 1 (Note 3)	Parity Error (PE)	Trailing Edge Ring Indicator (TERI)	Bit 2	Bit 2	Bit 10	RXRDY Select
3	Data Bit 3	Data Bit 3	Enable MODEM Status Interrupt (EMSI)	Interrupt ID Bit (Note 2)	DMA Mode Select	Parity Enable (PEN)	Out 2	Framing Error (FE)	Delta Data Carrier Detect (DDCD)	Bit 3	Bit 3	Bit 11	0
4	Data Bit 4	Data Bit 4	0	0	Reserved	Even Parity Select (EPS)	Loop	Break Interrupt (BI)	Clear to Send (CTS)	Bit 4	Bit 4	Bit 12	0
5	Data Bit 5	Data Bit 5	0	0	Reserved	Stick Parity	0	Transmitter Holding Register (THRE)	Data Set Ready (DSR)	Bit 5	Bit 5	Bit 13	0
6	Data Bit 6	Data Bit 6	0	FIFOs Enabled (Note 2)	RCVR Trigger (LSB)	Set Break	0	Transmitter Empty (TEMT)	Ring Indicator (RI)	Bit 6	Bit 6	Bit 14	0
7	Data Bit 7	Data Bit 7	0	FIFOs Enabled (Note 2)	RCVR Trigger (MSB)	Divisor Latch Access Bit (DLAB)	0	Error in RCVR FIFO (Note 2)	Data Carrier Detect (DCD)	Bit 7	Bit 7	Bit 15	0

Note 1: Bit 0 is the least significant bit. It is the first bit serially transmitted or received.

Note 2: These bits are always 0 in the NS16450 Mode.

Note 3: This bit no longer has a pin associated with it.

NS16C552

8.0 Registers (Continued)

Two identical register sets, one for each channel, are in the DUART. All register descriptions in this section apply to the register sets in both channels.

8.1 LINE CONTROL REGISTER

The system programmer specifies the format of the asynchronous data communications exchange and sets the Divisor Latch Access bit via the Line Control Register (LCR). This is a read and write register. Table II shows the contents of the LCR. Details on each bit follow:

Bits 0 and 1: These two bits specify the number of data bits in each transmitted or received serial character. The encoding of bits 0 and 1 is as follows:

Bit 1	Bit 0	Data Length
0	0	5 Bits
0	1	6 Bits
1	0	7 Bits
1	1	8 Bits

Bit 2: This bit specifies the number of Stop bits transmitted with each serial character. If bit 2 is a logic 0, one Stop bit is generated in the transmitted data. If bit 2 is a logic 1 when a 5-bit data length is selected, one and a half Stop bits are generated. If bit 2 is a logic 1 when either a 6-, 7-, or 8-bit word length is selected, two Stop bits are generated. The receiver checks the first Stop bit only, regardless of the number of Stop bits selected.

Bit 3: This bit is the Parity Enable bit. When bit 3 is a logic 1, a Parity bit is generated (transmit data) or checked (receive data) between the last data bit and Stop bit of the serial data. (The Parity bit is used to produce an even or odd number of 1s when the data bits and the Parity bit are summed.)

Bit 4: This bit is the Even Parity Select bit. When parity is enabled and bit 4 is a logic 0, an odd number of logic 1s is transmitted or checked in the data word bits and Parity bit. When parity is enabled and bit 4 is a logic 1, an even number of logic 1s is transmitted or checked.

Bit 5: This bit is the Stick Parity bit. When parity is enabled it is used in conjunction with bit 4 to select Mark or Space Parity. When bits 3, 4 and 5 are logic 1 the Parity bit is transmitted and checked as a logic 0 (Space Parity). If bits 3 and 5 are 1 and bit 4 is a logic 0 then the Parity bit is transmitted and checked as a logic 1 (Mark Parity). If bit 5 is a logic 0 Stick Parity is disabled.

Bit 6: This bit is the Break Control bit. It causes a break condition to be transmitted to the receiving UART. When it is set to a logic 1, the serial output (SOUT) is forced to the Spacing state (logic 0). The break is disabled by setting bit 6 to a logic 0. The Break Control bit acts only on SOUT and has no effect on the transmitter logic.

Note: This feature enables the CPU to alert a terminal in a computer communications system. If the following sequence is followed, no erroneous or extraneous characters will be transmitted because of the break.

1. Load an all 0s, pad character, in response to THRE.
2. Set break after the next THRE.
3. Wait for the transmitter to be idle, (TEMT = 1), and clear break when normal transmission has to be restored.

During the break, the Transmitter can be used as a character timer to accurately establish the break duration.

Bit 7: This bit is the Divisor Latch Access Bit (DLAB). It must be set high (logic 1) to access the Divisor Latches of the Baud Generator or the Alternate Function Register during a Read or Write operation. It must be set low (logic 0) to access any other register.

8.2 TYPICAL CLOCK CIRCUITS
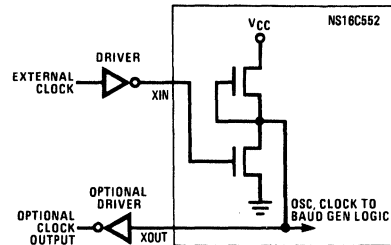
TL/C/9426–18
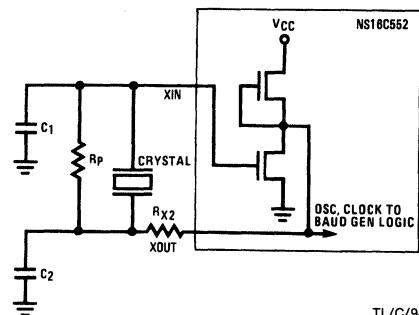
TL/C/9426–19

Typical Crystal Oscillator Network (Note)

Crystal	R_P	R_{X2}	C_1	C_2
3.1 MHz	1 MΩ	1.5k	10–30 pF	40–60 pF
1.8 MHz	1 MΩ	1.5k	10–30 pF	40–60 pF

Note: These R and C values are approximate and may vary 2 x depending on the crystal characteristics. All crystal circuits should be designed specifically for the system.

Composite Serial Data

TL/C/9426–21

8.0 Registers (Continued)

TABLE III. DUART Reset Configuration

Register/Signal	Reset Control	Reset State
Interrupt Enable Register	Master Reset	**0000** 0000 (Note 1)
Interrupt Identification Register	Master Reset	00**00** 0001
FIFO Control	Master Reset	00**00** 0000
Line Control Register	Master Reset	0000 0000
MODEM Control Register	Master Reset	**0000** 0000
Line Status Register	Master Reset	0110 0000
MODEM Status Register	Master Reset	XXXX 0000 (Note 2)
Alternate Function Register	Master Reset	000**0 0000**
SOUT	Master Reset	High
INTR (RCVR Errs)	Read LSR/MR	Low
INTR (RCVR Data Ready)	Read RBR/MR	Low
INTR (THRE)	Read IIR/Write THR/MR	Low
INTR (Modem Status Changes)	Read MSR/MR	Low
$\overline{OUT\ 2}$	Master Reset	High
$\overline{RTS}$	Master Reset	High
$\overline{DTR}$	Master Reset	High
RCVR FIFO	MR/FCR1•FCR0/ΔFCR0	All Bits Low
XMIT FIFO	MR/FCR1•FCR0/ΔFCR0	All Bits Low

Note 1: Boldface bits are permanently low.

Note 2: Bits 7–4 are driven by the input signals.

8.3 PROGRAMMABLE BAUD GENERATOR

The DUART contains two independently programmable Baud Generators. Each is capable of taking a common clock input from DC to 24.0 MHz and dividing it by any divisor from 1 to $2^{16} - 1$. The highest input clock frequency recommended with a divisor = 1 is 24 MHz. The output frequency of the Baud Generator is 16 × the baud rate, [divisor # = (frequency input) ÷ (baud rate × 16)]. The output of each Baud Generator drives the transmitter and receiver sections of the associated serial channel. Two 8-bit latches per channel store the divisor in a 16-bit binary format. These Divisor Latches must be loaded during initialization to ensure proper operation of the Baud Generator. Upon loading either of the Divisor Latches, a 16-bit Baud Counter is loaded.

Table IV provides decimal divisors to use with crystal frequencies of 1.8432 MHz, 3.072 MHz and 18.432 MHz. For baud rates of 38400 and below, the error obtained is minimal. The accuracy of the desired baud rate is dependent on the crystal frequency chosen. Using a divisor of zero is **not** recommended.

8.4 LINE STATUS REGISTER

This register provides status information to the CPU concerning the data transfer. Table II shows the contents of the Line Status Register. Details on each bit follow:

Bit 0: This bit is the receiver Data Ready (DR) indicator. Bit 0 is set to a logic 1 whenever a complete incoming character has been received and transferred into the Receiver Buffer Register or the FIFO. Bit 0 is reset to a logic 0 by reading all of the data in the Receiver Buffer Register or the FIFO.

Bit 1: This bit is the Overrun Error (OE) indicator. Bit 1 indicates that the next character received was transferred into the Receiver Buffer Register before the CPU could read the previously received character. This transfer destroys the previous character. The OE indicator is set to a logic 1 during the character stop bit time when the overrun condition exists. It is reset whenever the CPU reads the contents of the Line Status Register. If the FIFO mode data continues to fill the FIFO beyond the trigger level, an overrun error will occur only after the FIFO is full and the next character has been completely received in the shift register. OE is indicated to the CPU as soon as it happens. The character in the shift register can be overwritten, but it is not transferred to the FIFO.

Bit 2: This bit is the Parity Error (PE) indicator. Bit 2 indicates that the received data character does not have the correct even or odd parity, as selected by the even-parity-select bit. The PE bit is set to a logic 1 during the character Stop bit time when the character has a parity error. It is reset to a logic 0 whenever the CPU reads the contents of the Line Status Register or when the next character is loaded into the Receiver Buffer Register. In the FIFO mode this error is associated with the particular character in the FIFO it applies to. This error is revealed to the CPU when its associated character is at the top of the FIFO.

Bit 3: This bit is the Framing Error (FE) indicator. Bit 3 indicates that the received character did not have a valid Stop bit. The FE bit is set to a logic 1 when the serial channel detects a logic 0 during the first Stop bit time. The FE indicator is reset whenever the CPU reads the contents of the Line Status Register or when the next character is loaded into the Receiver Buffer Register. In the FIFO Mode this error is associated with the particular character in the FIFO it applies to. This error is revealed to the CPU when its associated character is at the top of the FIFO. The serial channel will try to resynchronize after a framing error. To do this it assumes that the framing error was due to the next start bit, so it samples this "start" bit twice and then takes in the "data".

4

8.0 Registers (Continued)

Bit 4: This bit is the Break Interrupt (BI) indicator. Bit 4 is set to a logic 1 whenever the received data input is held in the Spacing (logic 0) state for longer than a full word transmission time (that is, the total time of Start bit + data bits + Parity + Stop bits). The BI indicator is reset whenever the CPU reads the contents of the Line Status Register or when the next valid character is loaded into the Receiver Buffer Register. In the FIFO Mode this condition is associated with the particular character in the FIFO it applies to. It is revealed to the CPU when its associated character is at the top of the FIFO. When break occurs only one zero character is loaded into the FIFO. The next character transfer is enabled after SIN goes to the marking state and receives the next valid start bit.

Note: Bits 1 through 4 are the error conditions that produce a Receiver Line Status interrupt whenever any of the corresponding conditions are detected and the interrupt is enabled.

Bit 5: This bit is the Transmitter Holding Register Empty (THRE) indicator. In the 16450 mode bit 5 indicates that the associated serial channel is ready to accept a new character for transmission. In addition, this bit causes the DUART to issue an interrupt to the CPU when the Transmit Holding Register Empty Interrupt enable is set high. The THRE bit is set to a logic 1 when a character is transferred from the Transmitter Holding Register into the Transmitter Shift Register. The bit is reset to logic 0 concurrently with the loading of the Transmitter Holding Register by the CPU. In the FIFO mode this bit is set when the XMIT FIFO is empty; it is cleared when at least 1 byte is written to the XMIT FIFO.

Bit 6: This bit is the Transmitter Empty (TEMT) indicator. Bit 6 is set to a logic 1 whenever the Transmitter Holding Register (THR) and the Transmitter Shift Register (TSR) are both empty. It is reset to a logic 0 whenever either the THR or TSR contains a data character. In the FIFO mode this bit is set to one whenever the transmitter FIFO and shift register are both empty.

Bit 7: In the NS16450 Mode this is a 0. In the FIFO Mode LSR7 is set when there is at least one parity error, framing error or break indication in the FIFO. LSR7 is cleared when the CPU reads the LSR, if there are no subsequent errors in the FIFO.

Note: The Line Status Register is intended for read operations only. Writing to this register is not recommended as this operation is only used for factory testing. In the FIFO mode the user must load a data byte into the Rx FIFO in order to write to LSR2–4. LSR0 and LSR7 cannot be written to in the FIFO mode.

8.5 FIFO CONTROL REGISTER

This is a write only register at the same location as the IIR (the IIR is a read only register). This register is used to enable the FIFOs, clear the FIFOs, set the RCVR FIFO trigger level, and select the type of DMA signalling.

Bit 0: Writing a 1 to FCR0 enables both the XMIT and RCVR FIFOs. Resetting FCR0 will clear all bytes in both FIFOs. When changing from FIFO Mode to NS16450 Mode and vice versa, data is automatically cleared from the FIFOs. This bit must be a 1 when other FCR bits are written to or they will not be programmed.

Bit 1: Writing a 1 to FCR1 clears all bytes in the RCVR FIFO and resets its counter logic to 0. The shift register is not cleared. The 1 that is written to this bit position is self-clearing.

Bit 2: Writing a 1 to FCR2 clears all bytes in the XMIT FIFO and resets its counter logic to 0. The shift register is not cleared. The 1 that is written to this bit position is self-clearing.

Bit 3: Writing a 1 to FCR3 causes $\overline{RXRDY}$ and $\overline{TXRDY}$ operations to change from mode 0 to mode 1 if FCR0 = 1.

RXRDY Mode 0: When in the NS16450 Mode (FCR0 = 0) or in the FIFO Mode (FCR0 = 1, FCR3 = 0) and there is at least 1 character in the RCVR FIFO or RCVR Buffer Register, the $\overline{RXRDY}$ pin will go low active. Once active the $\overline{RXRDY}$ pin will go inactive when there are no more characters in the FIFO or Buffer Register.

TABLE IV. Baud Rates, Divisors and Crystals

Baud Rate	1.8432 MHz Crystal		3.072 MHz Crystal		18.432 MHz Crystal	
	Decimal Divisor for 16 × Clock	Percent Error	Decimal Divisor for 16 × Clock	Percent Error	Decimal Divisor for 16 × Clock	Percent Error
50	2304	—	3840	—	23040	—
75	1536	—	2560	—	15360	—
110	1047	0.026	1745	0.026	10473	—
134.5	857	0.058	1428	0.034	8565	—
150	768	—	1280	—	7680	—
300	384	—	640	—	3840	—
600	192	—	320	—	1920	—
1200	96	—	160	—	920	—
1800	64	—	107	0.312	640	—
2000	58	0.69	96	—	576	—
2400	48	—	80	—	480	—
3600	32	—	53	0.628	320	—
4800	24	—	40	—	240	—
7200	16	—	27	1.23	160	—
9600	12	—	20	—	120	—
19200	6	—	10	—	60	—
38400	3	—	5	—	30	—
56000	2	2.86	—	—	21	2.04
128000	—	—	—	—	9	—

Note: For baud rates of 250k, 300k, 375k, 500k, 750k and 1.5M using a 24 MHz crystal causes minimal error.

8.0 Registers (Continued)

RXRDY Mode 1: In the FIFO Mode (FCR0 = 1) when the FCR3 = 1 and the trigger level or the timeout has been reached, the $\overline{\text{RXRDY}}$ pin will go low active. Once it is activated it will go inactive when there are no more characters in the FIFO.

TXRDY Mode 0: In the NS16450 Mode (FCR0 = 0) or in the FIFO Mode (FCR0 = 1, FCR3 = 0) when there are no characters in the XMIT FIFO or XMIT Holding Register, the $\overline{\text{TXRDY}}$ pin will go low active. Once active the $\overline{\text{TXRDY}}$ pin will go inactive after the first character is loaded into the XMIT FIFO or Holding Register.

TXRDY Mode 1: In the FIFO Mode (FCR0 = 1, FCR3 = 1) and when there are no characters in the XMIT FIFO, the $\overline{\text{TXRDY}}$ pin will go low active. This pin will become inactive when the XMIT FIFO is completely full.

Bit 4, 5: FCR4 to FCR5 are reserved for future use.

Bit 6, 7: FCR6 and FCR7 are used to designate the interrupt trigger level. When the number of bytes in the RCVR FIFO equals the designated interrupt trigger level, a Received Data Available Interrupt is activated. This interrupt must be enabled by setting IER0.

FCR Bits		RCVR FIFO
7	**6**	**Trigger Level (Bytes)**
0	0	01
0	1	04
1	0	08
1	1	14

8.6 INTERRUPT IDENTIFICATION REGISTER

In order to provide minimum software overhead during data character transfers, each serial channel of the DUART prioritizes interrupts into four levels and records these in the Interrupt Identification Register. The four levels of interrupt conditions in order of priority are Receiver Line Status; Received Data Ready; Transmitter Holding Register Empty; and MODEM Status.

When the CPU reads the IIR, the associated DUART serial channel freezes all interrupts and indicates the highest priority pending interrupt to the CPU. While this CPU access is occurring, the associated DUART serial channel records new interrupts, but does not change its current indication until the access is complete. Table II shows the contents of the IIR. Details on each bit follow:

Bit 0: This bit can be used in a prioritized interrupt environment to indicate whether an interrupt is pending. When bit 0 is a logic 0, an interrupt is pending and the IIR contents may be used as a pointer to the appropriate interrupt service routine. When bit 0 is a logic 1, no interrupt is pending.

Bits 1 and 2: These two bits of the IIR identify the highest priority interrupt pending from those shown in Table V.

Bit 3: In the NS16450 Mode this bit is 0. In the FIFO Mode this bit is set along with bit 2 when a timeout interrupt is pending.

Bits 4 and 5: These two bits of the IIR are always logic 0.

Bits 6 and 7: These two bits are set when FCR0 = 1. (FIFO Mode enabled.)

TABLE V. Interrupt Control Functions

FIFO Mode Only	Interrupt Identification Register			Interrupt Set and Reset Functions			
Bit 3	**Bit 2**	**Bit 1**	**Bit 0**	**Priority Level**	**Interrupt Type**	**Interrupt Source**	**Interrupt Reset Control**
0	0	0	1	—	None	None	—
0	1	1	0	Highest	Receiver Line Status	Overrun Error or Parity Error or Framing Error or Break Interrupt	Reading the Line Status Register
0	1	0	0	Second	Received Data Available	Receiver Data Available or Trigger Level Reached	Reading the Receiver Buffer Register or the FIFO Drops below the Trigger Level
1	1	0	0	Second	Character Timeout Indication	No Characters Have Been Removed from or Input to the RCVR FIFO During the Last 4 Char. Times and There is at Least 1 Char. in it During This Time	Reading the Receiver Buffer Register
0	0	1	0	Third	Transmitter Holding Register Empty	Transmitter Holding Register Empty	Reading the IIR Register (if Source of Interrupt) or Writing into the Transmitter Holding Register
0	0	0	0	Fourth	MODEM Status	Clear to Send or Data Set Ready or Ring Indicator or Data Carrier Detect	Reading the MODEM Status Register

4

8.0 Registers (Continued)

8.7 INTERRUPT ENABLE REGISTER

This register enables five types of interrupts for the associated serial channel. Each interrupt can individually activate the interrupt (INTR) output signal. It is possible to totally disable the interrupt system by resetting bits 0 through 3 of the Interrupt Enable Register (IER). Similarly, setting bits of the IER register to a logic 1, enables the selected interrupt(s). Disabling an interrupt prevents it from being indicated as active in the IIR and from activating the INTR output signal. All other system functions operate in their normal manner, including the setting of the Line Status and MODEM Status Registers. Table II shows the contents of the IER. Details on each bit follow:

Bit 0: When set to logic 1 this bit enables the Received Data Available Interrupt and Timeout Interrupt in the FIFO Mode.

Bit 1: When set to logic 1 this bit enables the Transmitter Holding Register Empty Interrupt.

Bit 2: When set to logic 1 this bit enables the Receiver Line Status Interrupt.

Bit 3: When set to logic 1 this bit enables the MODEM Status Interrupt.

Bits 4 through 7: These four bits are always logic 0.

8.8 MODEM CONTROL REGISTER

This register controls the interface with the MODEM or data set (or a peripheral device emulating a MODEM). The contents of the MODEM Control Register are indicated in Table II and are described below:

Bit 0: This bit controls the Data Terminal Ready ($\overline{\text{DTR}}$) output. When bit 0 is set to a logic 1, the $\overline{\text{DTR}}$ output is forced to a logic 0. When bit 0 is reset to a logic 0, the $\overline{\text{DTR}}$ output is forced to a logic 1.

Bit 1: This bit controls the Request to Send ($\overline{\text{RTS}}$) output. Bit 1 affects the $\overline{\text{RTS}}$ output in a manner identical to that described above for bit 0.

Bit 2: This bit is the $\overline{\text{OUT 1}}$ bit. It does **not** have an output pin associated with it. It can be written to and read by the CPU. In Local Loopback Mode this bit controls bit 2 of the Modem Status Register.

Bit 3: This bit controls the Output 2 ($\overline{\text{OUT 2}}$) signal, which is an auxiliary user-designated output. Bit 3 affects the $\overline{\text{OUT 2}}$ pin in a manner identical to that described above for bit 0.

The function of this bit is multiplexed on a single output pin with two other functions: $\overline{\text{BAUDOUT}}$ and $\overline{\text{RXRDY}}$. The $\overline{\text{OUT 2}}$ function is the default function of the pin after a master reset. See Section 8.10 for more information about selecting one of these 3 pin functions.

Bit 4: This bit provides a local loopback feature for diagnostic testing of the associated serial channel. When bit 4 is set to logic 1, the following occur: the transmitter Serial Output (SOUT) is set to the Marking (logic 1) state; the receiver Serial Input (SIN) is disconnected; the output of the Transmitter Shift Register is "looped back" into the Receiver Shift Register input; the four MODEM Control inputs ($\overline{\text{DSR}}$, $\overline{\text{CTS}}$, $\overline{\text{RI}}$, and $\overline{\text{DCD}}$) are disconnected; the four MODEM Control outputs ($\overline{\text{DTR}}$, $\overline{\text{RTS}}$, $\overline{\text{OUT 1}}$, and $\overline{\text{OUT 2}}$) are internally connected to the four MODEM Control inputs; and the MODEM Control output pins are forced to their inactive state (high). In this diagnostic mode, data that is transmitted is immediately received. This feature allows the processor to verify transmit and receive data paths of the DUART.

In this diagnostic mode, the receiver and transmitter interrupts are fully operational. Their sources are external to the part. The MODEM Control Interrupts are also operational, but the interrupts' sources are now the lower four bits of the MODEM Control Register instead of the four MODEM Control inputs. The interrupts are still controlled by the Interrupt Enable Register.

Bits 5 through 7: These bits are permanently set to logic 0.

8.9 MODEM STATUS REGISTER

This register provides the current state of the control lines from the MODEM (or peripheral device) to the CPU. In addition to this current-state information, four bits of the MODEM Status Register provide change information. The latter bits are set to a logic 1 whenever a control input from the MODEM changes state. They are reset to logic 0 whenever the CPU reads the MODEM Status Register.

The contents of the MODEM Status Register are indicated in Table II and described below.

Bit 0: This bit is the Delta Clear to Send (DCTS) indicator. Bit 0 indicates that the $\overline{\text{CTS}}$ input to the chip has changed state since the last time it was read by the CPU.

Bit 1: This bit is the Delta Data Set Ready (DDSR) indicator. Bit 1 indicates that the $\overline{\text{DSR}}$ input to the chip has changed state since the last time it was read by the CPU.

Bit 2: This bit is the Trailing Edge of Ring Indicator (TERI) detector. Bit 2 indicates that the $\overline{\text{RI}}$ input to the chip has changed from a low to a high state.

Bit 3: This bit is the Delta Data Carrier Detect (DDCD) indicator. Bit 3 indicates that the $\overline{\text{DCD}}$ input to the chip has changed state.

Note: Whenever bit 0, 1, 2, or 3 is set to logic 1, a MODEM Status Interrupt is generated.

Bit 4: This bit is the complement of the Clear to Send ($\overline{\text{CTS}}$) input. If bit 4 (loop) of the MCR is set to a 1, this bit is equivalent to RTS in the MCR.

Bit 5: This bit is the complement of the Data Set Ready ($\overline{\text{DSR}}$) input. If bit 4 of the MCR is set to a 1, this bit is equivalent to DTR in the MCR.

Bit 6: This bit is the complement of the Ring Indicator ($\overline{\text{RI}}$) input. If bit 4 of the MCR is set to a 1, this bit is equivalent to OUT 1 in the MCR.

Bit 7: This bit is the complement of the Data Carrier Detect ($\overline{\text{DCD}}$) input. If bit 4 of the MCR is set to a 1, this bit is equivalent to OUT 2 in the MCR.

8.10 ALTERNATE FUNCTION REGISTER

This is a read/write register used to select specific modes of operation. It is located at address 010 when the DLAB bit is set.

Bit 0: When this bit is set the CPU can write concurrently to the same register in both register sets. This function is intended to reduce the DUART initialization time. It can be used by a CPU when both channels are initialized to the same state. The CPU can set or clear this bit by accessing either register set. When this bit is set the channel select pin still selects the channel to be accessed during read operations. Setting or clearing this bit has **no** effect on read operations.

The user should ensure that the DLAB bit (LCR7) of both channels are in the same state before executing a concurrent write to register addresses 0, 1 and 2.

8.0 Registers (Continued)

Bits 1 and 2: These select the output signal that will be present on the multi-function pin, $\overline{MF}$. These bits are individually programmable for each channel, so that different signals can be selected on each channel. Table VI associates the signal present at the multi-function pin with the bit code.

TABLE VI

AFR Bit Code		Multi-Function Pin Signal
Bit 2	Bit 1	
0	0 (Note 1)	$\overline{OUT\,2}$
0	1	$\overline{BAUDOUT}$
1	0	$\overline{RXRDY}$
1	1	Reserved (Note 2)

Note 1: This is the state after power-up or master reset.

Note 2: Output is forced high.

Bits 3 through 7: These bits are permanently set to a logic 0.

8.11 SCRATCHPAD REGISTER

This 8-bit Read/Write Register does not control the serial channel in any way. It is intended as a Scratchpad Register to be used by the programmer to hold data temporarily.

9.0 FIFO Mode Operation

Each serial channel has two 16-byte FIFOs associated with it. The operational description that follows is applicable to the FIFOs of both channels.

9.1 FIFO INTERRUPT OPERATION

When the RCVR FIFO and receiver interrupt are enabled (FCR0 = 1, IER0 = 1) Receive Data Available Interrupts will occur as follows:

A. The Receive Data Available Interrupt will be issued to the CPU when the number of bytes in the RCVR FIFO equals the programmed trigger level; it will be cleared as soon as the number of bytes in the RCVR FIFO drops below its programmed trigger level.

B. The IIR Receive Data Available Indication also occurs when the FIFO trigger level is reached, and like the interrupt it is cleared when the FIFO drops below the trigger level.

C. The Receiver Line Status Interrupt (IIR = 06), as before, has higher priority than the Received Data Available (IIR = 04) Interrupt.

D. The data ready bit (LSR0) is set as soon as a character is transferred from the shift register to the RCVR FIFO. It is reset when the RCVR FIFO is empty.

When RCVR FIFO and receiver interrupts are enabled, RCVR FIFO timeout interrupts will occur as follows:

A. A RCVR FIFO Timeout Interrupt will occur, if the following conditions exist:

— at least one character is in the RCVR FIFO

— the most recent serial character received was longer than 4 continuous character times ago (if 2 stop bits are programmed the second one is included in this time delay).

— the most recent CPU read of the RCVR FIFO was longer than 4 continuous character times ago.

The maximum time between a received character and a timeout interrupt will be 160 ms at 300 baud with a 12-bit receive character (i.e. 1 START, 8 DATA, 1 PARITY and 2 STOP BITS).

B. Character times are calculated by using the $\overline{BAUDOUT}$ signal as a clock signal (this makes the delay proportional to the baud rate).

C. When a timeout interrupt has occurred it is cleared and the timer reset when the CPU reads one character from the RCVR FIFO.

D. When the timeout interrupt indication is inactive the timeout indication timer is reset after a new character is received or after the CPU reads the RCVR FIFO.

When the XMIT FIFO interrupts are enabled (FCR0 = 1, IER1 = 1), XMIT interrupts will occur as follows:

A. The Transmitter Holding Register Empty Interrupt occurs when the XMIT FIFO is empty. It is cleared as soon as the Transmitter Holding Register is written to (1 to 16 characters may be written to the XMIT FIFO while servicing this interrupt) or the IIR is read.

B. The transmitter FIFO empty indications will be delayed 1 character time minus the last Stop bit time whenever the following occurs: THRE = 1 and there have not been at least two bytes at the same time in the transmit FIFO, since the last THRE = 1. The first Transmitter Holding Register Empty Interrupt after changing FCR0 will be immediate, if it is enabled.

This delay prevents the DUART from issuing a second Transmitter Holding Register Empty Interrupt as soon as it transfers the first character into the Transmitter Shift Register.

Character timeout and RCVR FIFO trigger level interrupts have the same priority as the current received data available interrupt; XMIT FIFO Empty has the same priority as the current Transmitter Holding Register Empty Interrupt.

9.2 FIFO POLLED OPERATION

With FCR0 = 1 resetting IER0, IER1, IER2, IER3 or all to zero puts the associated serial channel in the FIFO Polled Mode of operation. Since the receiver and transmitter are controlled separately either one or both can be in the polled mode of operation.

In this mode the user's program will check receiver and transmitter status via the LSR. As stated in Section 8.4:

LSR0 will be set as long as there is one byte in the RCVR FIFO.

LSR1 to LSR4 will specify which error(s) has occurred. Character error status is handled the same way as in the interrupt mode.

LSR5 will indicate when the XMIT FIFO is empty.

LSR6 will indicate that both the XMIT FIFO and shift register are empty.

LSR7 will indicate whether there are any errors in the RCVR FIFO.

There is no trigger level reached or timeout condition indicated in the FIFO Polled Mode, however, the RCVR and XMIT FIFOs are otherwise functional.

4

10.0 Ordering Information

NS16C552XX

/A⁺ = A⁺ RELIABILITY SCREENING

V = PLASTIC LEADED CHIP CARRIER (PLCC)

TL/C/9426–20

Accessing the NS16550A UART in the PS/2 Model 50, 60, 70, and 80

INTRODUCTION

This paper reviews fundamental concepts of the Micro Channel Architecture and their relation to the NS16550A UART. All 4 of the PS/2 personal computers reviewed, use the NS16550A for asynchronous serial communication.

The first part is an overview of the PS/2 system board and Micro Channel Architecture (MCA) in the Models 50, 60, 70, and 80 personal computers. The next part explains the basic configuration and system initialization procedures for the UART that occur after power-up. The last part describes the overall interrupt procedure and the advantages of using the on-chip FIFOs of the NS16550A. These explanations describe the CPU accesses to the UART via MCA. Timing diagrams in the appendix show these accesses to the UART.

OVERVIEW OF THE PS/2 MODEL 50, 60, 70, AND 80 SYSTEM ARCHITECTURE

The block diagram indicates a number of identical functions that all system boards have *(Figure 1)*. Each system CPU has an 8 channel DMA Controller and an optional math co-processor associated with it via the local bus. The DMA Controller emulates the dual 8237 DMA controllers found on the IBM AT. Additionally, this DMA Controller provides Extended and Virtual Mode operation. These modes allow it to interface with various DMA slave devices and the CPU to dynamically select the arbitration level for 2 of the DMA channels. A central arbitration point allows certain adapter cards and system peripherals to compete for DMA transfers. These adapter cards must have the appropriate arbitration and DMA logic.

Buffers condition the bus signals from the system CPU and send them directly to the Micro Channel Interface. These signals, after further buffering, reach the system memory and the system peripherals. The system ROM on the Models 50 and 60 also interfaces via these buffers to the 80286 CPU. In the Models 70 and 80 the 128 kbyte ROM interfaces via the local bus to the 80386 CPU.

The dynamic RAM is expandable on the system board or on adapter cards. DMA controller addressing capability limits the total DRAM available on any of these systems to 16 Mbytes. The maximum DRAM available on the various system boards is:

1. Model 50; Type 1 = 1 Mbyte, Type 2 = 2 Mbytes
2. Model 60; = 1 Mbyte
3. Model 70; Type 1 or Type 2 = 6 Mbytes
4. Model 80; Type 1 = 2 Mbytes, Type 2 = 4 Mbytes

Beyond the memory, coprocessor, and DMA there are a number of major peripheral functions resident on the system board. These are:

1. serial port (NS16550A)
2. video graphics controller
3. diskette controller
4. parallel port
5. keyboard and pointing device controller
6. CMOS clock and configuration RAM
7. dual interrupt controllers (16 channels)
8. timer (3 channels)

The configuration software for the serial port on the system board restricts the addressing of the NS16550A to COM1 and COM2 on the Models 50, 60, 70, and 80. Adapter card serial ports, however, may be assigned any 1 of the 8 base I/O addresses.

Adding adapter cards extends the PS/2 functionality beyond the system board. These cards plug into the Micro Channel Bus connectors and conform to the MCA protocols.

OVERVIEW OF THE MICRO CHANNEL ARCHITECTURE

MCA functionality increases as the data bus width increases from 16 to 32 bits. Both bus widths support certain fundamental features regardless of the data bus width. One of these is a centralized arbitration controller that allows up to 16 devices to contend for the 8 available DMA channels. These devices compete based on an assigned priority level for the DMA resource. A "fairness" option allows lower priority devices to compete successfully for a DMA channel, even though, higher priority devices may require a transfer. If the fairness option is enabled, each device that has received DMA service must wait until all other devices requesting the DMA have been serviced before they are allowed to compete for the DMA resource, again. MCA fixes the priority levels of the DMA channels, except for channels 0 and 4, which the CPU can program to any priority level. The DMA channels are capable of both 8- and 16-bit transfers.

MCA also features level sensitive interrupts, provides for interrupt sharing and brings 11 of the 16 hardware interrupts out to the Micro Channel Bus for the adapters to use. The last section of this paper describes interrupt handling in more detail.

Previous PC architectures used jumpers and switches to configure the adapter cards. MCA uses programmable configuration registers on each adapter card, instead. This adds to system flexibility by allowing automatic card configuration via software.

The Models 50 and 60 support 8- or 16-bit transfers over a 64 kbyte range of I/O addresses and over a 16 Mbyte range of memory addresses. The Models 70 and 80 have all of these capabilities and can execute 32-bit transfers over the 64 kbyte I/O address range or the 4 Gbyte memory address range.

4

MODELS 50, 60, 70, 80

MODEL 70 and 80 only

MODELS 50, 60 only

MODEL 70 only

OPTIONAL DRAM

STANDARD DRAM

128k ROM

MEMORY CONTROL

BUFFER

MATH COPROCESSOR

8 CHANNEL DMA

128k ROM

SYSTEM CPU

CLK

DIRECT BUS DISK ATTACHMENT

DISKETTE CONTROL

VIDEO GRAPHICS CONTROL

SERIAL PORT (NS16550A)

PARALLEL PORT

16 CHANNEL ICU

3 CHANNEL TIMER

KBRD. AND PT. DEVICE

CMOS CLK AND CONFIG. RAM

BUFFER

BUFFER

MULTI-DEVICE INTRF.

BUS CONTROLLER

CENTRAL ARBITRATION POINT

MICRO CHANNEL BUS

TL/C/10456-1

FIGURE 1. PS/2 Block Diagram

16-bit bus or 32-bit bus

16-bit bus or 32-bit bus

32-bit bus

V10
V1

AUXILIARY VIDEO EXTENSION

M4
M1

MATCHED MEMORY EXTENSION

01

01

01

8-BIT CORE BUS

45

45

45

46, 47 (KEY)

48
58

48
58

48
58

16-BIT EXTENSION

59

89

32-BIT EXTENSION

TL/C/10456-2

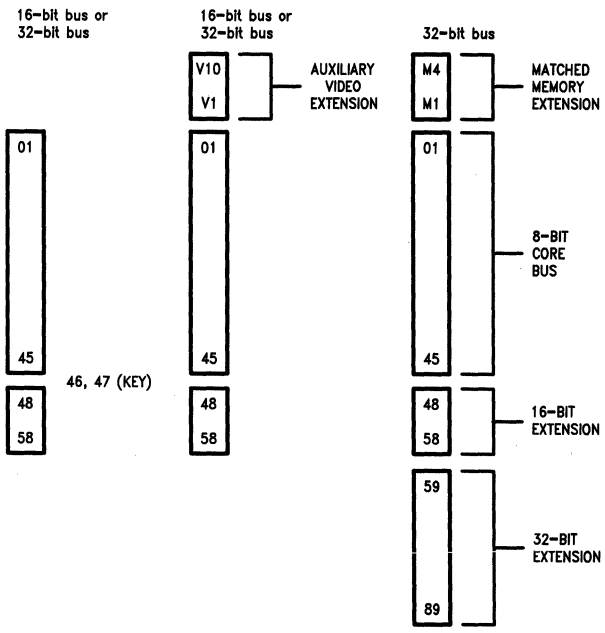**FIGURE 2. Micro Channel Architecture**

SUBDIVIDING THE MICRO CHANNEL ARCHITECTURE

MCA has an 8-bit core set of bus signals *(Figure 2)* and 4 types of extensions. If the system CPU is an 80286, a 16-bit extension and an auxiliary video extension are present. If the system CPU is an 80386, a 32-bit extension and a Matched Memory extension are present in addition to the 16-bit and auxiliary video extensions.

The **8-bit core** set of 90 signals are composed of 9 groups. Most of these are the typical bus signals associated with any CPU:

24 address	14 ground
8 data	8 DMA/arbitration
14 control	2 audio
6 interrupts	a 14.3 MHz clock signal.
9 power	

IBM reserved four signals in this set.

The 14 control signals can be grouped by function. A typical data transfer uses 7. The bus master uses 4 to sense card and channel status. Reset and channel configuration require 2, and a DRAM refresh cycle activates 1 signal.

The centralized arbitration controller uses 8 DMA/arbitration signals to support its features plus the facility for single or burst mode transfers. One signal notifies the DMA slave when the DMA channel it is using reaches its terminal count.

An audio summation input is available with a separate audio ground, so that all cards can drive the speaker.

Power and ground pin spacing keeps all bus signals within 0.1 inch of an AC ground potential, thus minimizing EMI.

This core set of signals provides the fundamental structure of MCA. Four extensions contain the remaining MCA Bus signals.

The **16-bit Extension** widens the data bus and adds more interrupts. It adds to the core 8 more data lines, 5 more interrupt lines, and a high byte enable line. This extension also provides a status signal driven by the adapter card to indicate its 16-bit wide data bus capability.

The **Auxiliary Video Extension** provides signals from the system board for use by video adapter cards. These are all of standard video sync signals (i.e., vsync, hsync, and blank) along with enable lines for the Video DAC clock and data lines. Through these enable lines the adapter controls whether it will source or receive the video DAC clock and data. Both of these extensions have the appropriate number of power and ground lines to maintain the reduced EMI specification.

Two additional bus extensions found only in the 32-bit MCA are the 32-bit extension and the Matched Memory Extension. The **32-bit Extension** widens the data bus by adding:

    16 data signals
    8 address signals
    4 byte enable signals
    3 32-bit data transfer status signals
    15 power and ground signals to maintain reduced EMI

The **Matched Memory Extension** provides signal lines to the adapter, so that the CPU can access RAM on the adapter cards as fast as it accesses system RAM. During Matched Memory cycles to the adapter card, the accesses will take 3 CPU clock cycles instead of 4. Matched Memory cycles use three signal lines. Two of these allow the CPU to indicate when it will provide a Matched Memory Cycle and when valid data is on the bus. The adapter card uses the third signal to request a Matched Memory Cycle.

In summary, the MCA provides all signals necessary for CPU or DMA data transfer to additional memory and peripherals and for monitoring card or bus status. It also supports some miscellaneous functions (e.g., audio, more interrupts, a clock oscillator, etc.). The four extensions provide for data bus expansion to 32 bits, fast memory access and auxiliary video control.

OVERVIEW OF THE PROGRAMMABLE OPTION SELECT (POS)

The Models 50, 60, 70, and 80 use software to configure the system peripherals and adapter cards, rather than providing switches and jumpers. This software is called the Configuration Utilities. These utilities match the system peripherals and adapter cards to their appropriate initialization files. The initialization files are known as Adapter Description Files (ADFs). These files each have an I.D. number that associates each ADF with the matching adapter card I.D. number. The ADF contains specific information used by the operator during system configuration such as the adapter card name, the system resources the adapter can use, and help information. It indicates to the system the minimum resources required for the adapter card to run. It also contains the codes used to record the specific options chosen by the operator (e.g., one of the options for an asynchronous communication card is the assignment of the COM port number to each UART). The ADF will specify the interrupt level, the arbitration level, the I/O register addresses, and the memory range addresses of the adapter card. The Configuration Utilities use this information to configure the adapter card and provide selectable options to the operator. ADFs are ASCII files.

When invoked, the Configuration Utilities begin reading the adapter card I.D. numbers and then read the ADF I.D. numbers *(Figure 3)*. They disable any cards that don't have an ADF and indicate any conflict of resources (e.g., the same COM port address assigned to two different UARTs) to the operator. When there are no conflicts or when the system is automatically configured, the utilities generate the system configuration data. The 64-byte CMOS RAM (all models) and the 2 kbyte CMOS RAM extension (Model 60, 70, and 80) store the configuration data. Once stored the system is ready for normal power-up operation.

During normal power-up *(Figure 4)* the Power On Self-Test (POST) software tests the hardware and compares the adapter I.D. numbers to those in the configuration RAM. If the numbers match, it initializes each adapter card. If they don't match, it requests that the Configuration Utilities be run to resolve the conflicts.

UART ACCESSES—SOFTWARE

During normal operation UART accesses are done by the applications software via DOS routines, BIOS routines, or by direct access to the UARTs designated addresses. The addresses of each UART in the system are assigned during system configuration.

Using the Configuration Utilities the operator assigns 1 of 8 available base addresses to the asynchronous communications ports on the adapter cards. Table 1 lists all possible addresses for the asynchronous communications ports; they include the original "COM1" and "COM2" addresses of the IBM AT in order to maintain software compatibility.

4

```
        ┌─────────────┐
        │   INVOKE    │
        │CONFIGURATION│
        │  UTILITIES  │
        └──────┬──────┘
               │
        ┌──────▼──────┐
        │ READ ADAPTER│
        │ CARD ID #'s │
        └──────┬──────┘
               │
        ┌──────▼──────┐
        │ READ ADAPTER│
        │ DESCRIPTION │
        │    FILE     │
        └──────┬──────┘
               │
            ◇──▼──◇        Y  ┌────────────┐
           ◇ CONFLICTS ◇─────▶│    USER    │
            ◇   ?   ◇         │  RESOLVES  │
            ◇──┬──◇           │  CONFLICTS │
               │ N            └─────┬──────┘
               │◀───────────────────┘
        ┌──────▼──────┐
        │  GENERATE   │
        │CONFIGURATION│
        │    DATA     │
        └──────┬──────┘
               │
        ┌──────▼──────┐
        │  CONFIGURE  │
        │   SYSTEM    │
        └──────┬──────┘
               │
        ┌──────▼──────┐    ┌──────┐
        │STORE ID #'s │───▶│ EXIT │
        │and CONFIG-  │    └──────┘
        │URATION DATA │
        │   IN RAM    │
        └─────────────┘
```
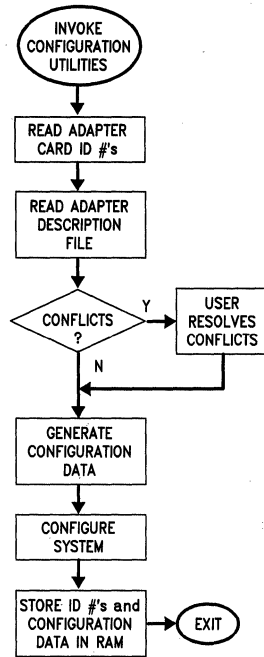
TL/C/10456–3

FIGURE 3. Configuration Utilities

```
        ┌─────────────┐
        │    POST     │
        │   STARTS    │
        └──────┬──────┘
               │
        ┌──────▼──────┐
        │ READ ADAPTER│
        │ CARD ID #'s │
        └──────┬──────┘
               │
        ┌──────▼──────┐
        │COMPARE ID #'s│
        │ WITH THOSE IN│
        │CONFIGURATION │
        │     RAM      │
        └──────┬──────┘
               │
            ◇──▼──◇        Y  ┌─────────────┐
           ◇ CONFLICTS ◇─────▶│   REQUEST   │
            ◇   ?   ◇         │CONFIGURATION│
            ◇──┬──◇           │  UTILITIES  │
               │ N            └─────────────┘
        ┌──────▼──────┐
        │  INTIALIZE  │
        │ADAPTER CARDS│
        └──────┬──────┘
               │
        ┌──────▼──────┐
        │  CONTINUE   │
        │    POST     │
        └──────┬──────┘
               │
            ┌──▼──┐
            │EXIT │
            └─────┘
```
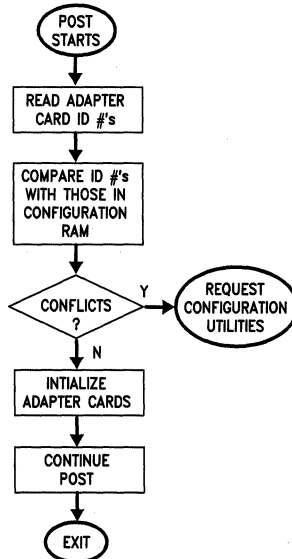
TL/C/10456–4

FIGURE 4. POST and Configuration Utilities

TABLE 1. Asynchronous Communication Port Addresses

Name	I/O Addr.	Interrupt	Comment
Serial 1	03f8–03ff	IRQ 4	COM1
Serial 2	02f8–02ff	IRQ 3	COM2
Serial 3	3220–3227	IRQ 3	
Serial 4	3228–322f	IRQ 3	
Serial 5	4220–4227	IRQ 3	
Serial 6	4228–422f	IRQ 3	
Serial 7	5220–5227	IRQ 3	
Serial 8	5228–522f	IRQ 3	

The system board description file restricts the addresses of the system board NS16550A to either COM1 or COM2. Adapter card ADFs determine which addresses are available for the UARTs on the adapter cards. Each adapter card contains a set of registers in the adapter I/O address space from 0100 to 0107 hex and also at 94 and 96 hex. Since these registers are at identical locations on all adapter cards, the system decodes a unique signal for each card when it needs to address these registers. This unique signal is called Card Setup [−CD SETUP (n)]. These ten registers in the adapter I/O address space:

1. Enable either the adapter card or the system board.

2. Store the adapter card I.D. number.

3. Record the selected card options.

4. Store card initialization data contained in the ADF.

5. Provide error status or a pointer to error status.

The NMI error handler uses this error status when the adapter card signals (via channel check, CHCK) a serious error. The error must be one that threatens the continued operation of the system (i.e., a parity error). One of the selectable options that the POS registers store is the address assigned to each UART on the adapter card.

The UART and these POS registers may be accessed at any time through the DOS Debug port I/O utility. This is done by enabling the setup of a particular adapter card through POS registers 0094, 0102 hex and then transferring the data to the assigned UART addresses. The Debug utility allows the adapter card configuration to be changed without running the Configuration Utilities. By not using the Configuration Utilities the user can easily cause configuration conflicts and should be cautious. IBM does not recommend this method of access, but is useful when testing new hardware.

UART ACCESSES—HARDWARE

Appendix A contains timing diagrams of accesses to the NS16550A addressed as COM2 on an IBM Dual Asynchronous Card. Signals from the Micro Channel Bus access this card. The timing of these signals should be the same for all PS/2 systems with MCA, however the measurements in Appendix A were done on only the Model 50 and Model 80. The read pulse width is approximately 390 ns. The write pulse width is approximately 220 ns and the chip select setup time is approximately 290 ns. The first two diagrams in Appendix A illustrate the basic read and write accesses to the UART. The middle two diagrams illustrate "back-to-back" write and read accesses to the Scratch Pad Register in the UART. The last two diagrams illustrate the initialization steps done during POST.

INTERRUPT HANDLING AND USE OF THE FIFOS

Interrupts on the PS/2 Models 50, 60, 70, and 80 are level-sensitive low active. This differs from the positive edge-sensitive interrupts on the IBM PCs, XTs, and ATs. There are several reasons for this change. One of the main reasons is interrupt sharing. It is apparent from the serial port addresses listed in Table 1, that 7 of the 8 serial ports will activate IRQ 3 for interrupt service. In an edge-sensitive system the Interrupt Control Unit (ICU) will not record any interrupt edges arriving after the first edge, but before the first interrupt is cleared. This makes interrupt sharing in an edge-sensitive system impossible, unless each device sharing the interrupt is sophisticated enough to only issue an interrupt when another device hasn't.

In a level-sensitive system multiple open-collector devices can hold the same interrupt line low until the CPU services each one. The only additional hardware required is an interrupt pending latch on each adapter card so that the CPU, can identify the card with an active interrupt. The CPU then executes the appropriate service routine for that card. This normally clears the device interrupt. The software then sends an End of Interrupt (EOI) to the ICU. If the same interrupt is still pending (interrupt sharing) the CPU checks the next card that could activate the interrupt signal for an active interrupt pending bit. If the bit is active the CPU services the interrupt as described above. It continues trying to clear this interrupt by checking and servicing the cards left in the chain until the interrupt clears or a higher priority event occurs.

Devices sharing the same interrupt level may have a lengthy wait before the CPU can service their interrupts. This can have a significant impact on the performance of a serial channel receiver. Since the receiver has no immediate control over the arrival of the incoming data without CPU intervention, it must be able to store the data for a period longer than the interrupt latency time of the CPU. The NS16550A has a provision for long interrupt latencies. Both the receiver and transmitter have 16-byte FIFOs. The CPU enables these FIFOs by writing an x1 hex to the third register (FCR) of the UART. All UARTs that have this FIFO capability will set two indicator bits in their Interrupt Identification Register (IIR).

Therefore, the software must read a Cx hex from the third register (IIR) of the UART before relying on the FIFOs. The receiver FIFO buffers incoming data. As this receiver FIFO fills, it will activate an interrupt. The CPU programs the level of receiver FIFO "fullness" needed to trigger this interrupt. The CPU selects one of these interrupt trigger levels during the UART initialization. This programmable trigger level allows the receiver FIFO to accommodate varying system interrupt latency times. When the receiver FIFO fills to this trigger level the UART issues an interrupt.

Another advantage of having the FIFOs on both the transmitter and receiver is the reduction in the number of interrupts the CPU must process. The NS16550A with enabled FIFOs can issue $1/16$th the number of transmitter interrupts to the CPU as compared to one with disabled FIFOs. The reduction in receiver interrupts is proportional to the number of bytes stored in the FIFO before the programmed trigger level is reached. Handling fewer interrupts reduces the amount of overhead the CPU needs to execute. Using the transmitter FIFO allows the CPU to send the same amount of data while executing $1/16$th the overhead.

4

The PS/2 architecture is well structured. It has many of the data and control pathways one expects in a more advanced system and some unexpected ones. This structure ensures that the software accesses to the NS16550A are identical, whether the UART is on the system board or an MCA adapter card. Allow the programmer needs to know is the serial port address of the target UART and its register set. The timing of signals accessing the UART on the MCA Bus is identical regardless of the system CPU. The timing of these signals is more than adequate for reliable operation of the NS16550A. As more multitasking and multiuser applications are written for the serial channels, the benefits of on-chip FIFOs will be apparent.

APPENDIX A

Dual ASYNC Card Read and Write

MARKER		ADDR	DATA		TIME	DISTANCE	ADDR
START MAG	S	−00044	0010 0011 1111 1111		S − T +000.44 US		S' +01360
CURSOR	C	+00038	0110 1111 0000 0011		C − T +000.38 US		C' +01792
TRIGGER	T	+00000	0110 1101 1111 1111		C − S +000.02 US		
	−0253	S TC					+1791

Roll to desired selection
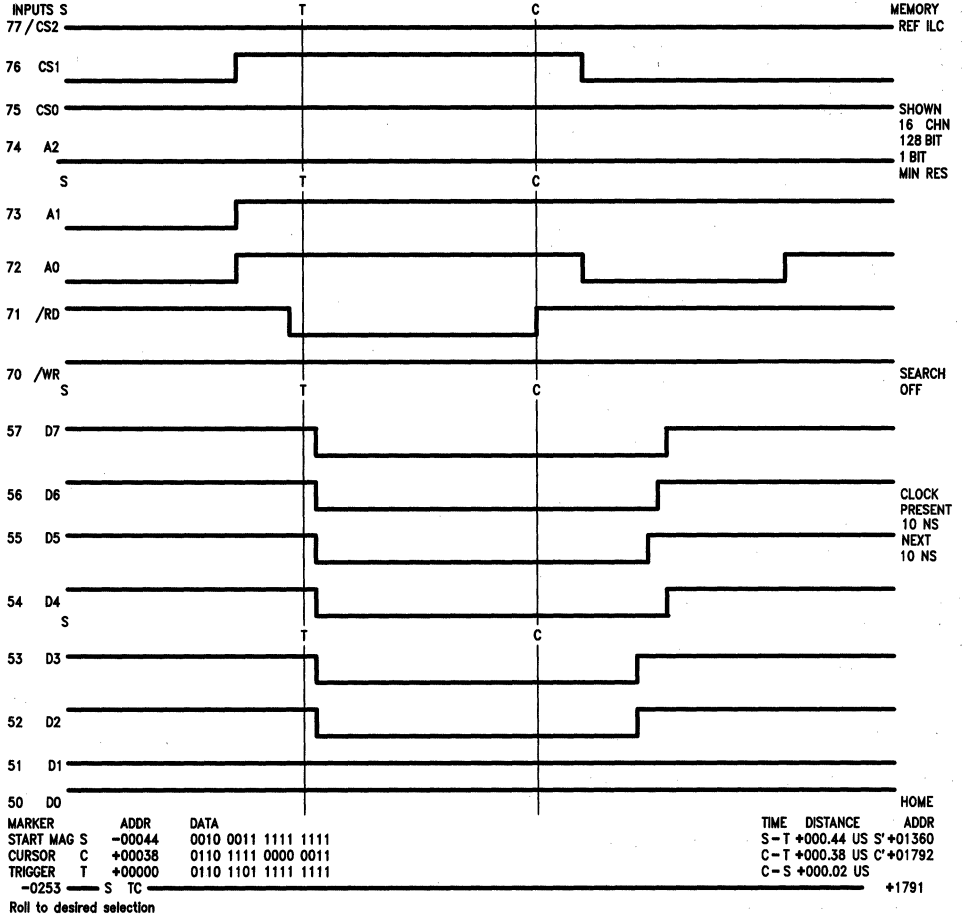
TL/C/10456–5

PS/2 Model 50 & Model 80 read from Dual ASYNC card (03 from LCR) read width 390 ns

Data file M50 IORD-Ref, Kontron PS/2 info Disk

INPUTS S   T   C   MEMORY

77 / CS2   REF ILC

76 CS1

75 CS0   SHOWN
     16 CHN

74 A2   128 BIT
     1 BIT

S   T   C   MIN RES

73 A1

72 A0

71 /RD

70 /WR   SEARCH
S   OFF

57 D7   T   C

56 D6   CLOCK
     PRESENT
     10 NS

55 D5   NEXT
     10 NS

54 D4

S   T   C

53 D3

52 D2

51 D1

50 D0   HOME

MARKER		ADDR	DATA		TIME DISTANCE	ADDR
START MAG	S	−00043	0011 0011 1111 1111		S − T +000.43 US	S' +01360
CURSOR	C	+00021	0110 1111 0000 0011		C − T +000.21 US	C' +01792
TRIGGER	T	+00000	0110 1110 0000 0011		C − S +000.64 US	
−0253 — S T						+1791

Roll to desired selection

TL/C/10456−6

PS/2 Model 50 & 80 write to Dual ASYNC card (03 to LCR) write width 220 ns

Data file M50 IOWR-Ref, Kontron PS/2 info Disk

4

4-119

Back-to-Back Write to SCR and Read from SCR

INPUTS S				
77 /CS2				MEMORY REF ILC
76 CS1				
75 CS0				SHOWN 16 CHN
74 A2				512 BIT 4 BIT MIN RES
73 A1				
72 A0				
71 /RD				
70 /WR				SEARCH OFF
57 D7				
56 D6				CLOCK PRESENT 10 NS
55 D5				NEXT 10 NS
54 D4				
53 D3				
52 D2				
51 D1				
50 D0				HOME

MARKER	ADDR	DATA		TIME DISTANCE	ADDR
START MAG S	−00387	0011 0011 1111 1111		S−T +003.87 US	S'+01360
CURSOR C	+00036	0111 1111 1010 1010		C−T +000.36 US	C'+01792
TRIGGER T	+00000	0111 1101 1111 1111		C−S +004.23 US	
−2301		S TC			+1791

Roll to desired selection

TL/C/10456–7

Back to back write to SCR and read from SCR during power-up

Write width 220 ns
Read width 340 ns
time /wr to /rd model 50 279 ns, model 80 192 ns

```
INPUTS S          T              C                                    MEMORY
77 /CS2                                                               REF ILC

76  CS1

75  CS0                                                               SHOWN
                                                                      16  CHN
74  A2                                                                128 BIT
    S             T              C                                    1 BIT
                                                                      MIN RES
73  A1

72  A0

71  /RD

70  /WR                                                               SEARCH
    S             T              C                                    OFF

57  D7

56  D6                                                                CLOCK
                                                                      PRESENT
                                                                      10 NS
55  D5                                                                NEXT
54  D4                                                                10 NS

    S             T              C

53  D3

52  D2

51  D1

50  D0                                                                HOME
```

MARKER		ADDR	DATA		TIME	DISTANCE	ADDR
START MAG	S	−00020	0111 1111 1111 1111		S − T +000.20 US		S'+01360
CURSOR	C	+00036	0111 1111 1010 1010		C − T +000.36 US		C'+01792
TRIGGER	T	+00000	0111 1101 1111 1111		C − S +000.56 US		
−2301				TC			+1791

Roll to desired selection

TL/C/10456–8

Detail of read cycle shown above

File M50 bk-bk ref, Kontron PS/2 info Disk

4

COM2 Initialization

INPUTS S

77 / CS2

76 CS1

75 CS0

74 A2

73 A1

72 A0

71 /RD

70 /WR

57 D7

56 D6

55 D5

54 D4

53 D3

52 D2

51 D1

50 D0

S   C   T

MEMORY
REF ILC

SHOWN
16 CHN
2048 BIT
16 BIT
MIN RES

SEARCH
OFF

CLOCK
PRESENT
10 NS
NEXT
10 NS

HOME

MARKER		ADDR	DATA		TIME	DISTANCE	ADDR
START MAG	S	−00861	0011 0011 1111 1111		S − T +008.61 US		S' +01360
CURSOR	C	+00573	0110 1111 1000 0000		C − T +005.73 US		C' +01536
TRIGGER	T	+00000	0110 0110 0000 0000		C − S +002.88 US		

−2557 ————————————— S   C   T   ——— +1535

Roll to desired selection
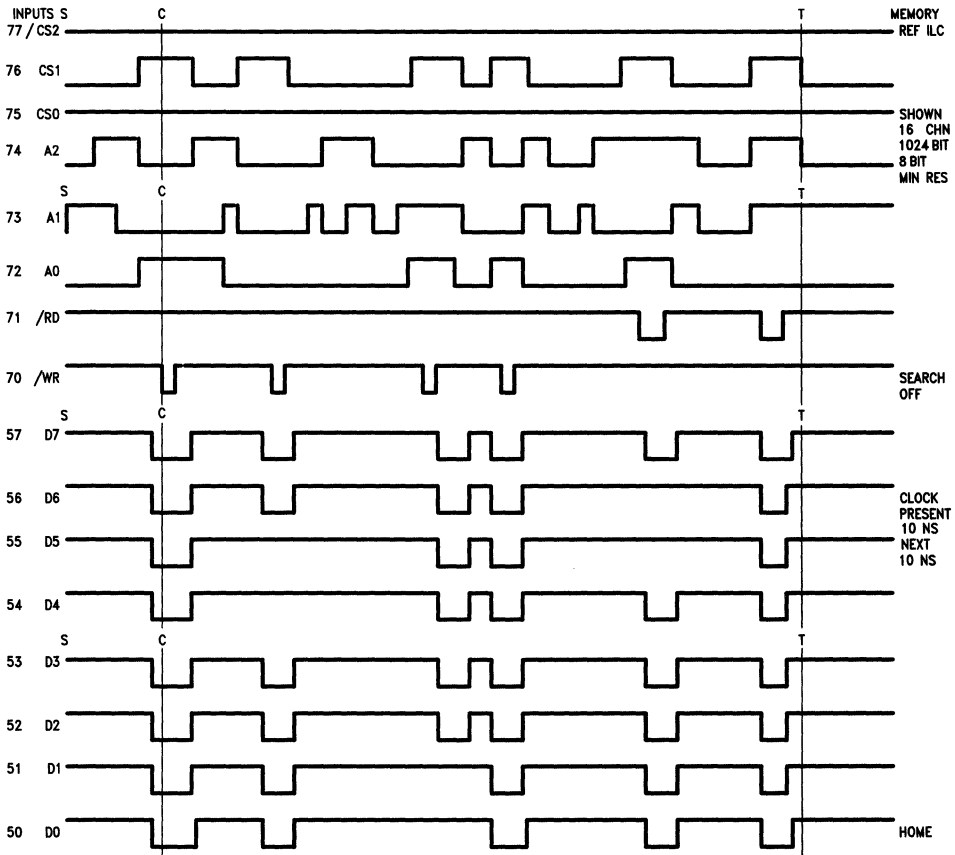
TL/C/10456−9

PS/2 Model 50 and 80 Dual ASYNC card initialization for COM2. The cursor shows the second write to the UART. The first write is shown on the "Back-TO-Back write to SCR Read from SCR" Timing Diagram

Detail of Last Part of Initialization

INPUTS				MEMORY
77 / CS2				REF ILC
76	CS1			
75	CS0			SHOWN 16 CHN
74	A2			1024 BIT 8 BIT MIN RES
73	A1			
72	A0			
71	/RD			
70	/WR			SEARCH OFF
57	D7			
56	D6			CLOCK PRESENT 10 NS
55	D5			NEXT 10 NS
54	D4			
53	D3			
52	D2			
51	D1			
50	D0			HOME

MARKER		ADDR	DATA		TIME	DISTANCE	ADDR
START MAG	S	−00109	0010 0011 1111 1111		S − T +001.09 US	S' +01360	
CURSOR	C	+00835	0110 1011 1111 1111		C − T +006.35 US	C' +01536	
TRIGGER	T	+00000	0110 0110 0000 0000		C − S +009.44 US		
−2557				S T	C	+1535	

Roll to desired selection

TL/C/10456−10

Initialization Sequence
(Including SCR)

WR	SCR	AAH	Check for 8250-B Part
RD	SCR	AAH	
WR	CCR	80H	Set Dlab
WR	DLH	00H	Set Baud to 2400
WR	DLL	30H	
WR	LCR	03H	8 Data, 1 Stop, no Parity
WR	IER	00H	Disable Intrs
RD	LSR	60H	Clear Status
RD	MSR	00H	Clear Status

4

Section 5
Modems

5

Section 5 Contents

National Semiconductor

MM74HC942 300 Baud Modem

General Description

The MM74HC942 is a full duplex low speed modem. It provides a 300 baud bidirectional serial interface for data communication over telephone lines and other narrow bandwidth channels. It is Bell 103 compatible.

The MM74HC942 utilizes advanced silicon-gate CMOS technology. Switched capacitor techniques are used to perform analog signal processing.

MODULATOR SECTION

The modulator contains a frequency synthesizer and a sine wave synthesizer. It produces a phase coherent frequency shift keyed (FSK) output.

LINE DRIVER AND HYBRID SECTION

The line driver and hybrid are designed to facilitate connection to a 600Ω phone line. They can perform two-to-four-wire conversion and drive the line at a maximum of 0 dBm.

DEMODULATOR SECTION

The demodulator incorporates anti-aliasing filters, a receive filter, limiter, discriminator, and carrier detect circuit. The nine pole receive filter provides 60 dB of transmitted tone rejection. The discriminator is fully balanced for stable operation.

Features

- Drives 600Ω at 0 dBm
- All filters on chip
- Transmit level adjustment compatible with universal service order code
- TTL and CMOS compatible logic
- All inputs protected against static damage
- ±5V supplies
- Low power consumption
- Full duplex answer or originate operation
- Analog loopback for self test
- Power down mode

Applications

- Built-in low speed modems
- Remote data collection
- Radio telemetry
- Credit verification
- Stand-alone modems
- Point-of-sale terminals
- Tone signalling systems
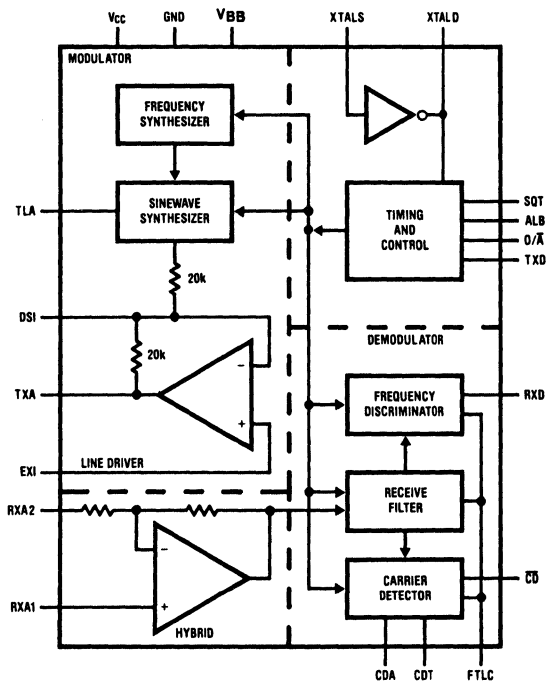- Remote process control

Connection and Block Diagrams

Dual-In-Line Package

Top View

TL/F/5348-1
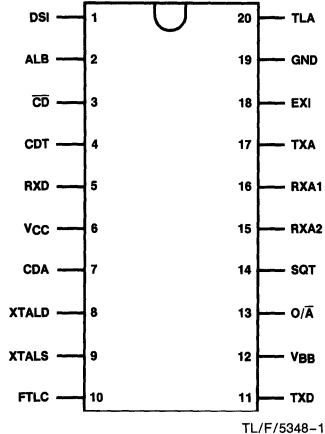
Order Number MM54HC942* or MM74HC942*

*Please look into Section 8, Appendix D for availability of various package types.

TL/F/5348-2

5

Absolute Maximum Ratings (Notes 1 & 2)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	-0.5 to $+7.0$V
Supply Voltage (V_{BB})	$+0.5$ to -7.0V
DC Input Voltage (V_{IN})	$V_{BB}-1.5$ to $V_{CC}+1.5$V
DC Output Voltage (V_{OUT})	$V_{BB}-0.5$ to $V_{CC}+0.5$V
Clamp Diode Current (I_{IK}, I_{OK})	±20 mA
DC Output Current, per pin (I_{OUT})	±25 mA
DC V_{CC} or GND Current, per pin (I_{CC})	±50 mA
Storage Temperature Range (T_{STG})	$-65°C$ to $+150°C$
Power Dissipation (P_D)	
(Note 3)	600 mW
S.O. Package only	500 mW
Lead Temp. (T_L)	
(Soldering 10 seconds)	260°C

Operating Conditions

	Min	Max	Units
Supply Voltage (V_{CC})	4.5	5.5	V
Supply Voltage (V_{BB})	-4.5	-5.5	V
DC Input or Output Voltage (V_{IN}, V_{OUT})	0	V_{CC}	V
Operating Temp. Range (T_A) MM74HC	-40	$+85$	°C
Input Rise or Fall Times (t_r, t_f)		500	ns
Crystal frequency		3.579	MHz

DC Electrical Characteristics

Symbol	Parameter	Conditions	T = 25°C	74HC T = −40 to 85°C	Units
			Typ	Guaranteed Limits	
V_{IH}	Minimum High Level Input Voltage		3.15	3.15	V
V_{IL}	Maximum Low Level Input Voltage		1.1	1.1	V
V_{OH}	Minimum High Level Output Voltage	$V_{IN}=V_{IH}$ or V_{IL} $\|I_{OUT}\|=20\ \mu A$ $\|I_{OUT}\|=4.0$ mA, $V_{CC}=4.5V$	V_{CC}	$V_{CC}-0.1$ 3.98	$V_{CC}-0.1$ 3.7	V V
V_{OL}	Maximum Low Level Voltage	$V_{IN}=V_{IH}$ or V_{IL} $\|I_{OUT}\|=20\ \mu A$ $\|I_{OUT}\|=4.0$ mA, $V_{CC}=4.5V$		0.1 0.26	0.1 0.4	V V
I_{IN}	Maximum Input Current	$V_{IN}=V_{CC}$ or GND		±0.1	±1.0	μA
I_{OZ}	Output TRI-STATE® Leakage Current RXD and $\overline{CD}$ Outputs	ALB = SQT = V_{CC}			±5	μA
I_{CC}, I_{BB}	Maximum Quiescent Supply Current	$V_{IH}=V_{CC}$, $V_{IL}=$ GND ALB or SQT = GND Transmit Level = -9 dBm	8.0	12.0	12.0	mA
I_{CC}, I_{BB}	Power Down Supply Current	ALB = SQT = V_{CC} $V_{IH}=V_{CC}$, $V_{IL}=$ GND			300	μA

Note 1: Absolute Maximum Ratings are those values beyond which damage to the device may occur.

Note 2: Unless otherwise specified all voltages are referenced to ground.

Note 3: Power Dissipation temperature derating — plastic "N" package: -12 mW/°C from 65°C to 85°C; ceramic "J" package: -12 mW/°C from 100°C to 125°C.
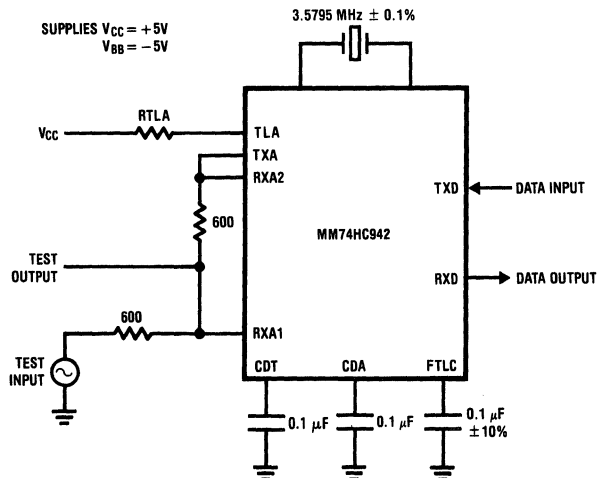
*The demodulator specifications apply to the MM74HC942 operating with a modulator having frequency accuracy, phase jitter and harmonic content equal to or better than the MM74HC942 modulator.

AC Electrical Characteristics

Unless otherwise specified, all specifications apply to the MM74HC942 over the range $-40°C$ to $+85°C$ using a $V_{CC} = +5V$ $\pm 10\%$, a $V_{BB} = -5V \pm 10\%$ and a 3.579MHz $\pm 0.1\%$ crystal.*

Symbol	Parameter	Conditions		Min	Typ	Max	Units
TRANSMITTER							
F_{CE}	Carrier Frequency Error					4	Hz
	Power Output	$V_{CC} = 5.0V$	$R_{TLA} = 0\Omega$	-3	-1.5	0	dBm
		$R_L = 1.2\,k\Omega$	$R_{TLA} = 5.49\,k\Omega$	-12	-10.5	-9	dBm
	2nd Harmonic Energy	$R_{TLA} = 0\Omega$			-62	-56	dBm
RECEIVE FILTER AND HYBRID							
	Hybrid Input Impedance (Pins 15 and 16)			50			$k\Omega$
	FTLC Output Impedance			5	10	50	$k\Omega$
	Adjacent Channel Rejection	RXA2 = GND TXA = GND or V_{CC} Input to RXA1		60			dB
DEMODULATOR (INCORPORATING HYBRID, RECEIVE FILTER AND DISCRIMINATOR)							
	Carrier Amplitude			-48		-9	dBm
	Bit Jitter	SNR = 30 dB Input = -38 dBm Baud Rate = 300 Baud			100	200	μS
	Bit Bias	Alternating 1-0 Pattern			5	10	%
	Carrier Detect Trip Points	CDA = 1.2V	Off to On	-45	-42	-40	dBm
		$V_{CC} = 5.0V$	On to Off	-47	-45	-42	dBm
	Carrier Detect Hysteresis	$V_{CC} = 5V$		2	3	4	dB

AC Specification Circuit

TL/F/5348–3

Description of Pin Functions

Pin No.	Name	Function
1	DSI	Driver Summing Input: This may be used to transmit externally generated tones such as dual tone multifrequency (DTMF) dialing signals.
2	ALB	Analog Loop Back: A logic high on this pin causes the modulator output to be connected to the demodulator input so that data is looped back through the entire chip. This is used as a chip self test. If ALB and SQT are simultaneously held high the chip powers down.
3	$\overline{CD}$	Carrier Detect: This pin goes to a logic low when carrier is sensed by the carrier detect circuit.
4	CDT	Carrier Detect Timing: A capacitor on this pin sets the time interval that the carrier must be present before the $\overline{CD}$ goes low.
5	RXD	Received Data: This is the data output pin.
6	V_{CC}	Positive Supply Pin: A +5V supply is recommended.
7	CDA	Carrier Detect Adjust: This is used for adjustment of the carrier detect threshold. Carrier detect hysteresis is set at 3 dB.
8	XTALD	Crystal Drive: XTALD and XTALS connect to a 3.5795 MHz crystal to generate a crystal locked clock for the chip. If an external circuit requires this clock XTALD should be sensed. If a suitable clock is already available in the system, XTALD can be driven.
9	XTALS	Crystal Sense: Refer to Pin 8 for details.
10	FTLC	Filter Test/Limiter Capacitor: This is connected to a high impedance output of the receive filter. It may thus be used to evalu-

Pin No.	Name	Function
		ate filter performance. This pin may also be driven to evaluate the demodulator. RXA1 and RXA2 must be grounded during this test.
		For normal modem operation FTLC is AC grounded via a 0.1 µF bypass capacitor.
11	TXD	Transmitted Data: This is the data input.
12	V_{BB}	Negative Supply: The recommended supply is −5V.
13	O/$\overline{A}$	Originate/$\overline{\text{Answer}}$ mode select: When logic high this pin selects the originate mode of operation.
14	SQT	Squelch Transmitter: This disables the modulator when held high. The EXI input remains active. If SQT and ALB are simultaneously held high the chip powers down.
15	RXA2	Receive Analog #2: RXA2 and RXA1 are analog inputs. When connected as recommended they produce a 600Ω hybrid.
16	RXA1	Receive Analog #1: See RXA2 for details.
17	TXA	Transmit Analog: This is the output of the line driver.
18	EXI	External Input: This is a high impedance input to the line driver. This input may be used to transmit externally generated tones. When not used for this purpose it should be grounded.
19	GND	Ground: This defines the chip 0V.
20	TLA	Transmit Level Adjust: A resistor from this pin to V_{CC} sets the transmit level.

Functional Description

INTRODUCTION

A modem is a device for transmitting and receiving serial data over a narrow bandwidth communication channel. The MM74HC942 uses frequency shift keying (FSK) of an audio frequency tone. The tone may be transmitted over the switched telephone network and other voice grade channels. The MM74HC942 is also capable of demodulating FSK signals. By suitable tone allocation and considerable signal processing the MM74HC942 is capable of transmitting and receiving data simultaneously.

The tone allocation by the MM74HC942 and other Bell 103 compatible modems is shown in Table I. The terms "originate" and "answer" which define the frequency allocation come from use with telephones. The modem on the end of the line which initiates the call is called the originate modem. The other modem is the answer modem.

TABLE I. BELL 103 Allocation

Data	Originate Modem		Answer Modem	
	Transmit	Receive	Transmit	Receive
Space	1070Hz	2025Hz	2025Hz	1070Hz
Mark	1270Hz	2225Hz	2225Hz	1270Hz

THE LINE INTERFACE

The line interface section performs two to four wire conversion and provides impedance matching between the modem and the phone line.

THE LINE DRIVER

The line driver is a power amplifier for driving the line. If the modem is operating as an originate modem, the second harmonics of the transmitted tones fall close to the frequencies of the received tones and degrade the received signal to noise ratio (SNR). The line driver must thus produce low second harmonic distortion.

THE HYBRID

The voltage on the telephone line is the sum of the transmitted and received signals. The hybrid subtracts the transmitted voltage from the voltage on the telephone line. If the telephone line was matched to the hybrid impedance, the output of the hybrid would be only the received signal. This rarely happens because telephone line characteristic impedances vary considerably. The hybrid output is thus a mixture of transmitted and received signals.

Functional Description (Continued)

THE DEMODULATOR SECTION

The Receive Filter

The demodulator recovers the data from the received signals. The signal from the hybrid is a mixture of transmitted signal, received signals and noise. The first stage of the receive filter is an anti-alias filter which attenuates high frequency noise before sampling occurs. The signal then goes to the second stage of the receive filter where the transmitted tones and other noise are filtered from the received signal. This is a switched capacitor nine-pole filter providing at least 60 dB of transmitted tone rejection. This also provides high attenuation at 60 Hz, a common noise component.

The Discriminator

The first stage of the discriminator is a hard limiter. The hard limiter removes from the received signal any amplitude modulation which may bias the demodulator toward a mark or a space. It compares the output of the receive filter to the voltage on the 0.1 μF capacitor on the FTLC pin.

The hard limiter output connects to two parallel bandpass filters in the discriminator. One filter is tuned to the mark frequency and the other to the space frequency. The outputs of these filters are rectified, filtered and compared. If the output of the mark path exceeds the output of the space path the RXD output goes high. The opposite case sends RXD low.

The demodulator is implemented using precision switched capacitor techniques. The highly critical comparators in the limiter and discriminator are auto-zeroed for low offset.

Carrier Detector

The output of the discriminator is meaningful only if there is sufficient carrier being received. This is established in the carrier detection circuit which measures the signal on the line. If this exceeds a certain level for a preset period (adjustable by the CDT pin) the $\overline{CD}$ output goes low indicating that carrier is present. Then the carrier detect threshold is lowered by 3 dB. This provides hysteresis ensuring the $\overline{CD}$ output remains stable. If carrier is lost $\overline{CD}$ goes high after the preset delay and the threshold is increased by 3 dB.

MODULATOR SECTION

The modulator consists of a frequency synthesizer and a sine wave synthesizer. The frequency produces one of four tones depending on the O/$\overline{A}$ and TXD pins. The frequencies are synthesized to high precision using a crystal oscillator and variable dual modulus counter. The counters used respond quickly to data changes, introducing negligible bit jitter while maintaining phase coherence.

The sine wave synthesizer uses switched capacitors to "look up" the voltages of the sine wave. This sampled signal is then further processed by switched capacitor and continuous filters to ensure the high spectral purity required by FCC regulations.

Applications Information

TRANSMIT LEVEL ADJUSTMENT

The transmitted power levels of Table II refer to the power delivered to a 600Ω load from the external 600Ω source impedance. The voltage on the load is half the TXA voltage. This should be kept in mind when designing interface circuits which do not match the load and source impedances.

The transmit level is programmable by placing a resistor from TLA to VCC. With a 5.5k resistor the line driver transmits a maximum of -9 dBm. Since most lines from a phone installation to the exchange provide 3 dB of attenuation the maximum level reaching the exchange will be -12 dBm. This is the maximum level permitted by most telephone companies. Thus with this programming the MM74HC942 will interface to most telephones. This arrangement is called the "permissive arrangement." The disadvantage with the permissive arrangement is that when the loss from a phone to the exchange exceeds 3 dB, no compensation is made and SNR may be unnecessarily degraded.

SNR can be maximized by adjusting the transmit level until the level at the exchange reaches -12 dBm. This must be done with the cooperation of the telephone company. The programming resistor used is specific for a given installation and is often included in the telephone jack at the installation. The modem is thus programmable and can be used with any jack correctly wired. This arrangement is called the universal registered jack arrangement and is possible with the MM74HC942. The values of resistors required to program the MM74HC942 follow the most common code in use; the universal service order code. The required resistors are given in Table II.

TABLE II. Universal Service Order Code Resistor Values

Line Loss (dB)	Transmit Level (dBm)	Programming Resistor (R_{TLA}) (Ohms)
0	-12	Open
1	-11	19,800
2	-10	9,200
3	-9	5,490
4	-8	3,610
5	-7	2,520
6	-6	1,780
7	-5	1,240
8	-4	866
9	-3	562
10	-2	336
11	-1	150
12	0	0

CARRIER DETECT THRESHOLD ADJUSTMENT

The carrier detect threshold is directly proportional to the voltage on CDA. This pin is connected internally to a high impedance source. This source has a nominal Thevenin equivalent voltage of 1.2V and output impedance of 100 kΩ.

By forcing the voltage on CDA the carrier detect threshold may be adjusted. To find the voltage required for a given threshold the following equation may be used;

$$V_{CDA} = 244 \times V_{ON}$$
$$V_{CDA} = 345 \times V_{OFF}$$

CARRIER DETECT TIMING ADJUSTMENT

CDT: A capacitor on Pin 4 sets the time interval that the carrier must be present before $\overline{CD}$ goes low. It also sets the time interval that carrier must be removed before $\overline{CD}$ returns high. The relevant timing equations are:

$$T_{\overline{CD}L} \cong 6.4 \times C_{CDT} \quad \text{for } \overline{CD} \text{ going low}$$
$$T_{\overline{CD}H} \cong 0.54 \times C_{CDT} \quad \text{for } \overline{CD} \text{ going high}$$

Where $T_{\overline{CD}L}$ & $T_{\overline{CD}H}$ are in seconds, and C_{CDT} is in μF.

5

Applications Information (Continued)

DESIGN PRECAUTIONS

Power supplies to digital systems may contain high amplitude spikes and other noise. To optimize performance of the MM74HC942 operating in close proximity to digital systems, supply and ground noise should be minimized. This involves attention to power supply design and circuit board layout.

Power supply decoupling close to the device is recommended. Ground loops should be avoided. For further discussion of these subjects see the Audio/Radio Handbook published by National Semiconductor Corporation.

Interface Circuits for MM74HC942 300 Baud Modem
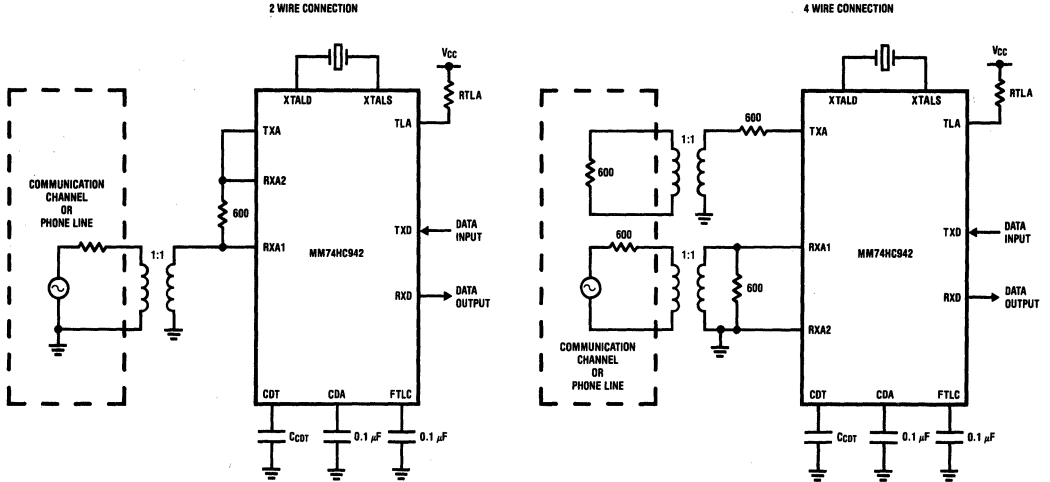
C$_{CDT}$ and R$_{TLA}$ should be chosen to suit the application. See the Applications Information for more details.

TL/F/5348–4

Complete Acoustically Coupled 300 Baud Modem
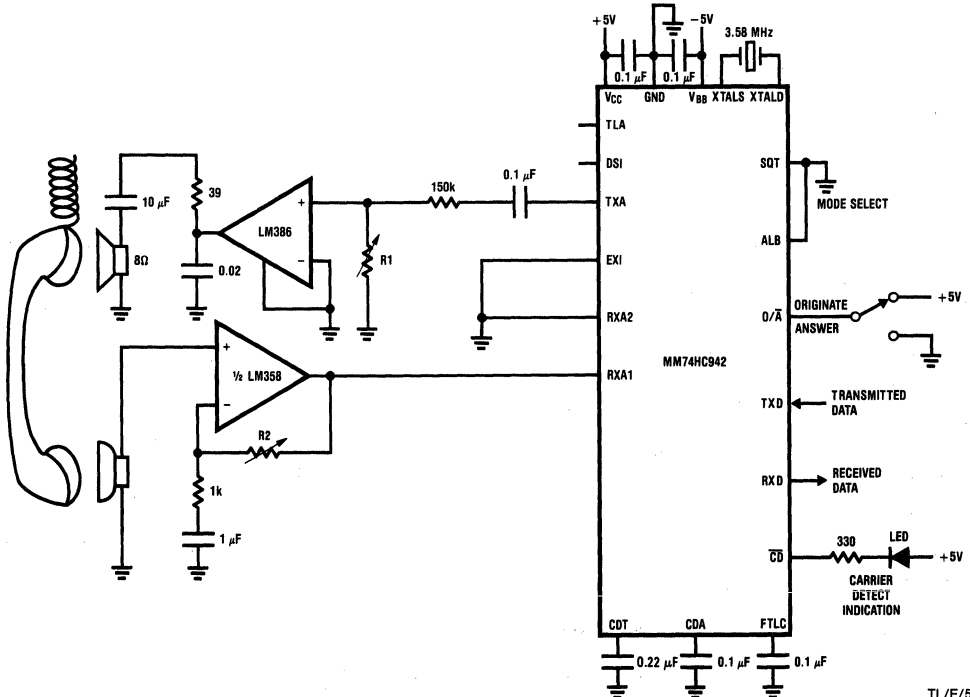
TL/F/5348–5

Note: The efficiency of the acoustic coupling will set the values of R1 and R2.

![National Semiconductor logo] **National Semiconductor**

MM74HC943 300 Baud Modem

General Description

The MM74HC943 is a full duplex low speed modem. It provides a 300 baud bidirectional serial interface for data communication over telephone lines and other narrow bandwidth channels. It is Bell 103 compatible.

The MM74HC943 utilizes advanced silicon-gate CMOS technology. Switched capacitor techniques are used to peform analog signal processing.

MODULATOR SECTION

The modulator contains a frequency synthesizer and a sine wave synthesizer. It produces a phase coherent frequency shift keyed (FSK) output.

LINE DRIVER AND HYBRID SECTION

The line driver and hybrid are designed to facilitate connection to a 600Ω phone line. They can perform two to four wire conversion and drive the line at a maximum of −9 dBm.

DEMODULATOR SECTION

The demodulator incorporates anti-aliasing filters, a receive filter, limiter, discriminator, and carrier detect circuit. The nine-pole receive filter provides 60 dB of transmitted tone rejection. The discriminator is fully balanced for stable operation.
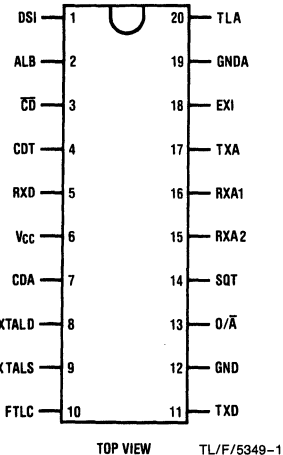
Features

- 5V supply
- Drives 600Ω at −9 dBm
- All filters on chip
- Transmit level adjustment compatible with universal service order code
- TTL and CMOS compatible logic
- All inputs protected against static damage
- Low power consumption
- Full duplex answer or originate operation
- Analog loopback for self test
- Power down mode

Applications

- Built-in low speed modems
- Remote data collection
- Radio telemetry
- Credit verification
- Stand-alone modems
- Point-of-sale terminals
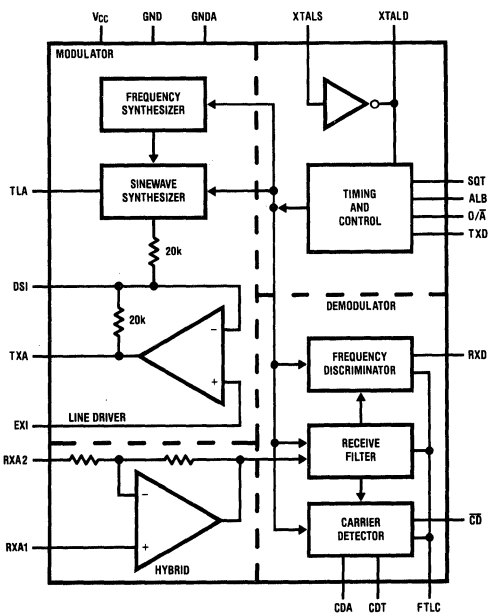- Tone signaling systems
- Remote process control

Connection and Block Diagrams

Dual-In-Line Package

Pin		Pin	
DSI	1	20	TLA
ALB	2	19	GNDA
CD	3	18	EXI
CDT	4	17	TXA
RXD	5	16	RXA1
Vcc	6	15	RXA2
CDA	7	14	SQT
XTALD	8	13	O/A
XTALS	9	12	GND
FTLC	10	11	TXD

TOP VIEW   TL/F/5349−1

Order Number MM74HC943*

*Please look into Section 8, Appendix D for availability of various package types.

TL/F/5349−2

5

Absolute Maximum Ratings (Notes 1 & 2)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	-0.5 to $+7.0$V
DC Input Voltage (V_{IN})	-1.5 to $V_{CC}+1.5$V
DC Output Voltage (V_{OUT})	-0.5 to $V_{CC}+0.5$V
Clamp Diode Current (I_{IK}, I_{OK})	± 20 mA
DC Output Current, per pin (I_{OUT})	± 25 mA
DC V_{CC} or GND Current, per pin (I_{CC})	± 50 mA
Storage Temperature Range (T_{STG})	$-65°$C to $+150°$C
Power Dissipation (P_D)	
(Note 3)	600 mW
S.O. Package only	500 mW
Lead Temp. (T_L) (Soldering 10 seconds)	260°C

Operating Conditions

	Min	Max	Units
Supply Voltage (V_{CC})	4.5	5.5	V
DC Input or Output Voltage (V_{IN}, V_{OUT})	0	V_{CC}	V
Operating Temp. Range (T_A)			
MM74HC	-40	$+85$	°C
Input Rise or Fall Times (t_r, t_f)		500	ns
Crystal frequency		3.579	MHz

DC Electrical Characteristics $V_{CC} = 5$V $\pm 10\%$ (unless otherwise specified)

Symbol	Parameter	Conditions	$T_A = 25°$C	74HC $T_A = -40$ to $85°$C	Units
			Typ	Guaranteed Limits	
V_{IH}	Minimum High Level Input Voltage			3.15 3.15	V
V_{IL}	Maximum Low Level Input Voltage			1.1 1.1	V
V_{OH}	Minimum High Level Output Voltage	$V_{IN} = V_{IH}$ or V_{IL} $\|I_{OUT}\| = 20\ \mu$A $\|I_{OUT}\| = 4.0$ mA, $V_{CC} = 4.5$V	$V_{CC}-0.05$	$V_{CC}-0.1$ $V_{CC}-0.1$ 3.84 3.7	V V
V_{OL}	Maximum Low Level Voltage	$V_{IN} = V_{IH}$ or V_{IL} $\|I_{OUT}\| = 20\ \mu$A $\|I_{OUT}\| = 4.0$ mA, $V_{CC} = 4.5$V		0.1 0.1 0.33 0.4	V V
I_{IN}	Maximum Input Current	$V_{IN} = V_{CC}$ or GND		± 0.1 ± 1.0	μA
I_{OZ}	Output TRI-STATE® Leakage Current, RXD and $\overline{CD}$ Outputs	ALB $=$ SQT $= V_{CC}$		± 5	μA
I_{CC}	Maximum Quiescent Supply Current	$V_{IH} = V_{CC}$, $V_{IL} =$ GND ALB or SQT $=$ GND Transmit Level $= -9$ dBm	8.0	10.0 10.0	mA
I_{GNDA}	Analog Ground Current		1.0	2.0 2.0	mA
I_{CC}	Power Down Supply Current	ALB $=$ SQT $= V_{CC}$ $V_{IH} = V_{CC}$, $V_{IL} =$ GND		300	μA

Note 1: Absolute Maximum Ratings are those values beyond which damage to the device may occur.

Note 2: Unless otherwise specified all voltages are referenced to ground.

Note 3: Power Dissipation temperature derating — plastic "N" package: -12 mW/°C from 65°C to 85°C; ceramic "J" package: -12 mW/°C from 100°C to 125°C.
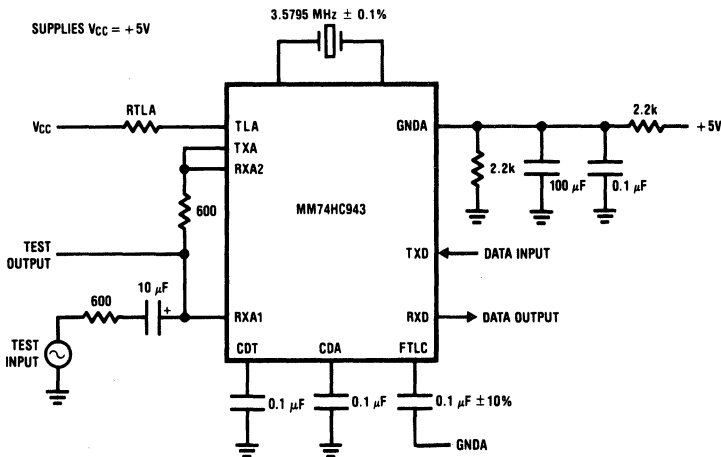
*The demodulator specifications apply to the MM74HC943 operating with a modulator having frequency accuracy, phase jitter and harmonic content equal to or better than the MM74HC943 modulator.

AC Electrical Characteristics

Unless otherwise specified, all specifications apply to the MM74HC943 over the range −40°C to +85°C using a V_{CC} of +5V ±10%, and a 3.579 MHz ±0.1% crystal.*

Symbol	Parameter	Conditions		Min	Typ	Max	Units
TRANSMITTER							
F_{CE}	Carrier Frequency Error					4	Hz
	Power Output	$V_{CC} = 5.0V$ $R_L = 1.2\,k\Omega$	$R_{TLA} = 5490\Omega$	−12	−10.5	−9	dBm
	2nd Harmonic Energy		$R_{TLA} = 5490\Omega$		−62	−56	dBm
RECEIVE FILTER AND HYBRID							
	Hybrid Input Impedance (Pins 15 and 16)			50			kΩ
	FTLC Output Impedance			5	10	50	kΩ
	Adjacent Channel Rejection	RXA2 = GNDA, TXD = GND or V_{CC} Input to RXA1		60			dB
DEMODULATOR (INCORPORATING HYBRID, RECEIVE FILTER AND DISCRIMINATOR)							
	Carrier Amplitude			−48		−12	dBm
	Bit Jitter	SNR = 30 dB Input = −38 dBm Baud Rate = 300 Baud			100	200	μS
	Bit Bias	Alternating 1−0 Pattern			5	10	%
	Carrier Detect Trip Points	CDA = 1.2V	Off to On	−45	−42	−40	dBm
		$V_{CC} = 5.0V$	On to Off	−47	−45	−42	dBm
	Carrier Detect Hystereisis	$V_{CC} = 5.0V$		2	3	4	dB

AC Specification Circuit

TL/F/5349−3

5

Description of Pin Functions

Pin No.	Name	Function
1	DSI	Driver Summing Input: This input may be used to transmit externally generated tones such as dual tone multifrequency (DTMF) dialing signals.
2	ALB	Analog Loop Back: A logic high on this pin causes the modulator output to be connected to the demodulator input so that data is looped back through the entire chip. This is used as a chip self test. If ALB and SQT are simultaneously held high the chip powers down.
3	$\overline{\text{CD}}$	Carrier Detect: This pin goes to a logic low when carrier is sensed by the carrier detect circuit.
4	CDT	Carrier Detect Timing: A capacitor on this pin sets the time interval that the carrier must be present before the $\overline{\text{CD}}$ goes low.
5	RXD	Received Data: This is the data output pin.
6	V_{CC}	Positive Supply Pin: A +5V supply is recommended.
7	CDA	Carrier Detect Adjust: This is used for adjustment of the carrier detect threshold. Carrier detect hysteresis is set at 3 dB.
8	XTALD	Crystal Drive: XTALD and XTALS connect to a 3.5795 MHz crystal to generate a crystal locked clock for the chip. If an external circuit requires this clock XTALD should be sensed. If a suitable clock is already available in the system. XTALD can be driven.
9	XTALS	Crystal Sense: Refer to pin 8 for details.
10	FTLC	Filter Test/Limiter Capacitor: This is connected to a high impedance output of the receiver filter. It may thus be used to evalu-

Pin No.	Name	Function
		ate filter performance. This pin may also be driven to evaluate the demodulator. RXA1 and RXA2 must be grounded during this test. For normal modem operation FTLC is AC grounded via a 0.1 μF bypass capacitor.
11	TXD	Transmitted Data: This is the data input.
12	GND	Ground: This defines the chip 0V.
13	O/$\overline{\text{A}}$	Originate/Answer mode select: When logic high this pin selects the originate mode of operation.
14	SQT	Squelch Transmitter: This disables the modulator when held high. The EXI input remains active. If SQT and ALB are simultaneously held high the chip powers down.
15	RXA2	Receive Analog #2: RXA2 and RXA1 are analog inputs. When connected as recommended they produce a 600Ω hybrid.
16	RXA1	Receive Analog #1: See RXA2 for details.
17	TXA	Transmit Analog: This is the output of the line driver.
18	EXI	External Input: This is a high impedance input to the line driver. This input may be used to transmit externally generated tones. When not used for this purpose it should be grounded to GNDA.
19	GNDA	Analog Ground: Analog signals within the chip are referred to this pin.
20	TLA	Transmit Level Adjust: A resistor from this pin to V_{CC} sets the transmit level.

Functional Description

INTRODUCTION

A modem is a device for transmitting and receiving serial data over a narrow bandwidth communication channel. The MM74HC943 uses frequency shift keying (FSK) of audio frequency tone. The tone may be transmitted over the switched telephone network and other voice grade channels. The MM74HC943 is also capable of demodulating FSK signals. By suitable tone allocation and considerable signal processing the MM74HC943 is capable of transmitting and receiving data simultaneously.

The tone allocation used by the MM74HC943 and other Bell 103 compatible modems is shown in Table I. The terms "originate" and "answer" which define the frequency allocation come from use with telephones. The modem on the end of the line which initiates the call is called the originate modem. The other modem is the answer modem.

TABLE I. Bell 103 Tone Allocation

Data	Originate Modem		Answer Modem	
	Transmit	Receive	Transmit	Receive
Space	1070Hz	2025Hz	2025Hz	1070Hz
Mark	1270Hz	2225Hz	2225Hz	1270Hz

THE LINE INTERFACE

The line interface section performs two to four wire conversion and provides impedance matching between the modem and the phone line.

THE LINE DRIVER

The line driver is a power amplifier for driving the line. If the modem is operating as an originate modem, the second harmonics of the transmitted tones fall close to the frequencies of the received tones and degrade the received signal to noise ratio (SNR). The line driver must thus produce low second harmonic distortion.

THE HYBRID

The voltage on the telephone line is the sum of the transmitted and received signals. The hybrid subtracts the transmitted voltage from the voltage on the telephone line. If the telephone line was matched to the hybrid impedance, the output of the hybrid would be only the received signal. This rarely happens because telephone line characteristic impedances vary considerably. The hybrid output is thus a mixture of transmitted and received signals.

Functional Description (Continued)

THE DEMODULATOR SECTION

The Receive Filter

The demodulator recovers the data from the received signals. The signal from the hybrid is a mixture of transmitted signal, received signals and noise. The first stage of the receive filter is an anti-alias filter which attenuates high frequency noise before sampling occurs. The signal then goes to the second stage of the receive filter where the transmitted tones and other noise are filtered from the received signal. This is a switch capacitor nine pole filter providing at least 60 dB of transmitted tone rejection. This also provides high attenuation at 60Hz, a common noise component.

The Discriminator

The first stage of the discriminator is a hard limiter. The hard limiter removes from the received signal any amplitude modulation which may bias the demodulator toward a mark or a space. It compares the output of the receive filter to the voltage on the 0.1 μF capacitor on the FTLC pin.

The hard limiter output connects to two parallel bandpass filters in the discriminator. One filter is tuned to the mark frequency and the other to the space frequency. The outputs of these filters are rectified, filtered and compared. If the output of the mark path exceeds the output of the space path the RXD output goes high. The opposite case sends RXD low.

The demodulator is implemented using precision switched capacitor techniques The highly critical comparators in the limiter and discriminator are auto-zeroed for low offset.

Carrier Detector

The output of the discriminator is meaningful only if there is sufficient carrier being received. This is established in the carrier detection circuit which measures the signal on the line. If this exceeds a certain level for a preset period (adjustable by the CDT pin) the $\overline{CD}$ output goes low indicating that carrier is present. Then the carrier detect threshold is lowered by 3 dB. This provides hysteresis ensuring the $\overline{CD}$ output remains stable. If carrier is lost $\overline{CD}$ goes high after the preset delay and the threshold is increased by 3 dB.

MODULATOR SECTION

The modulator consists of a frequency synthesizer and a sine wave synthesizer. The frequency synthesizer produces one of four tones depending on the O/A and TXD pins. The frequencies are synthesized to high precision using a crystal oscillator and variable dual modulus counter.

The counters used respond quickly to data changes, introducing negligible bit jitter while maintaining phase coherence.

The sine wave synthesizer uses switched capacitors to "look up" the voltages of the sine wave. This sampled signal is then further processed by switched capacitor and continuous filters to ensure the high spectral purity required by FCC regulations.

Applications Information

TRANSMIT LEVEL ADJUSTMENT

The transmitted power levels of Table II refer to the power delivered to a 600Ω load from the external 600Ω source impedance. The voltage on the load is half the TXA voltage. This should be kept in mind when designing interface circuits which do not match the load and source inpedances.

The transmit level is programmable by placing a resistor from TLA to V_{CC}. With a 5.5k resistor the line driver transmits a maximum of -9 dBm. Since most lines from a phone installation to the exchange provide 3 dB of attenuation the maximum level reaching the exchange will be -12 dBm. This is the maximum level permitted by most telephone companies. Thus with this programming the MM74HC943 will interface to most telephones. This arrangement is called the "permissive arrangement." The disadvantage with the permissive arrangement is that when the loss from a phone to the exchange exceeds 3 dB, no compensation is made and SNR may be unnecessarily degraded.

TABLE II. Universal Service Order Code Resistor Values

Line Loss (dB)	Transmit Level (dBm)	Programming Resistor (R_{TLA}) (Ω)
0	-12	Open
1	-11	19,800
2	-10	9,200
3	-9	5,490

CARRIER DETECT THRESHOLD ADJUSTMENT

The carrier detect threshold is directly proportional to the voltage on CDA. This pin is connected internally to a high impedance source. This source has a nominal Thevenin equivalent voltage of 1.2V and output impedance of 100 kΩ.

By forcing the voltage on CDA the carrier detect threshold may be adjusted. To find the voltage required for a given threshold the following equation may be used:

$$V_{CDA} = 244 \times V_{ON}$$
$$V_{CDA} = 345 \times V_{OFF}$$

CARRIER DETECT TIMING ADJUSTMENT

CDT: A capacitor on Pin 4 sets the time interval that the carrier must be present before $\overline{CD}$ goes low. It also sets the time interval that carrier must be removed before $\overline{CD}$ returns high. The relevant timing equations are:

$$T_{\overline{CDL}} \cong 6.4 \times C_{CDT} \quad \text{for } \overline{CD} \text{ going low}$$
$$T_{\overline{CDH}} \cong 0.54 \times C_{CDT} \quad \text{for } \overline{CD} \text{ going high}$$

Where $T_{\overline{CDL}}$ & $T_{\overline{CDH}}$ are in seconds, and C_{CDT} is in μF.

DESIGN PRECAUTIONS

Power supplies to digital systems may contain high amplitude spikes and other noise. To optimize performance of the MM74HC943 operating in close proximity to digital systems, supply and ground noise should be minimized. This involves attention to power supply design and circuit board layout. Power supply decoupling close to the device is recommended. Ground loops should be avoided. For further discussion of these subjects see the Audio/Radio Handbook published by National Semiconductor Corporation.
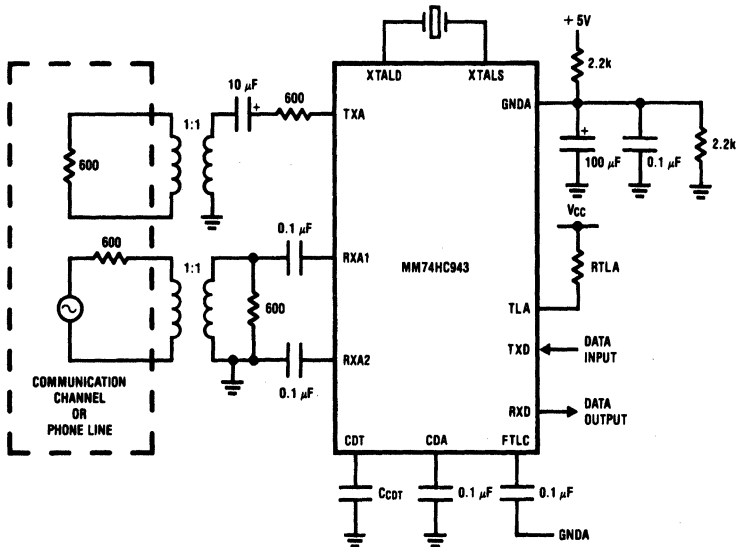
Applications Information (Continued)

Interface Circuits for MM74HC943 300 Baud Modem
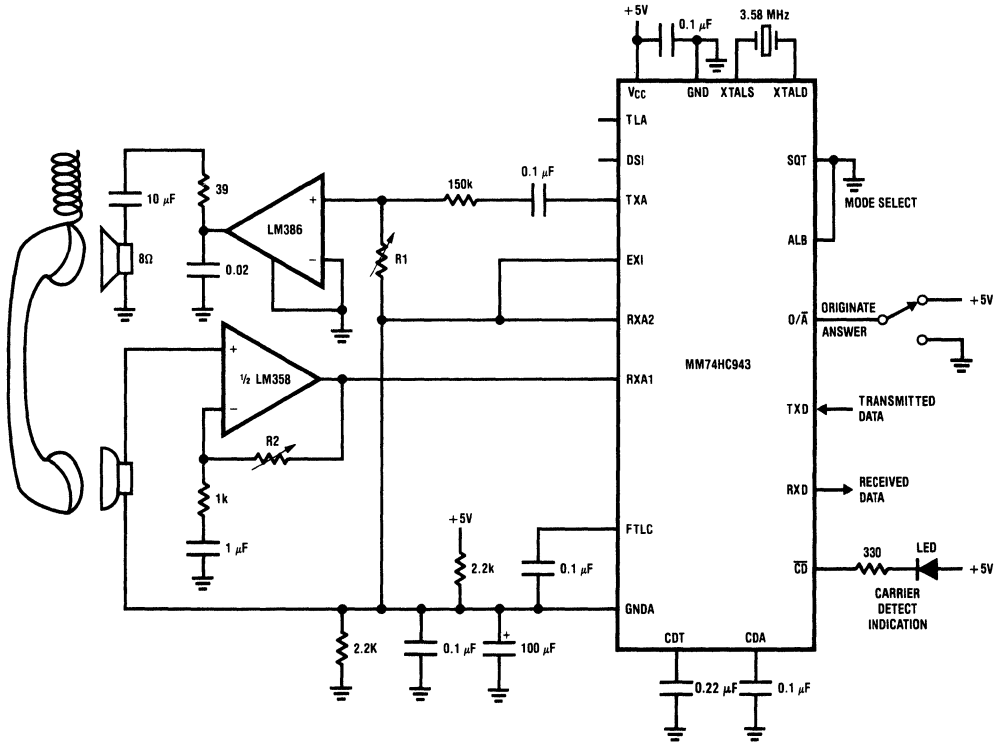
2 Wire Connection
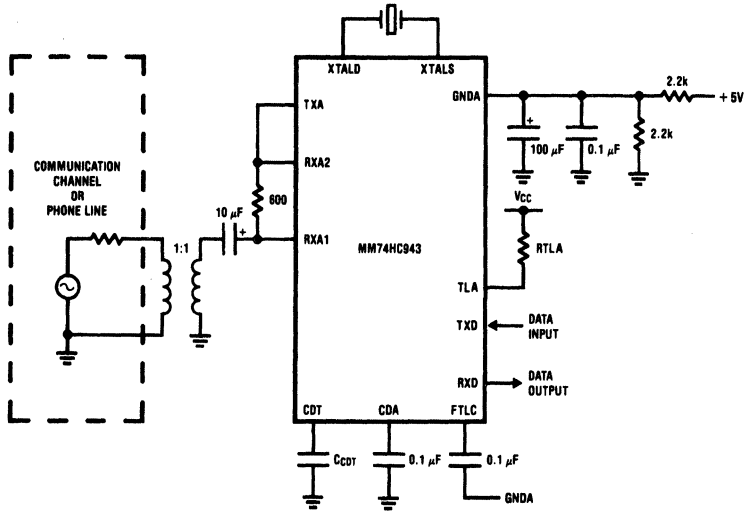
TL/F/5349–4

4 Wire Connection

TL/F/5349–5

C_{CDT} and R_{TLA} should be chosen to suit the application. See the Applications Information for more details.

Applications Information (Continued)

Complete Acoustically Coupled 300 Baud Modem

TL/F/5349–6

Note: The efficiency of the acoustic coupling will set the values of R1 and R2.

Section 6
Transmission Line
Drivers & Receivers

NOTE: For complete specifications on datasheets in this section please see the Interface databook.

Section 6 Contents

National Semiconductor

Transmission Line Drivers/Receivers

The common purpose of transmission line drivers and receivers is to transmit data quickly and reliably through a variety of environments over electrically long distances. This task is complicated by the fact that externally introduced noise and ground shifts can severely degrade the data.

The connection between two elements in a system should be considered a transmission line if the transmitted signal takes longer than twice its rise or fall time to travel from the driver to the receiver.

Single-Ended Data Transmission

In data processing systems today there are two basic means of communicating between components. One method is single-ended, which uses only one signal line for data transmission, and the other is differential, which uses two signal lines.

The Electronics Industry Association (EIA) has developed several standards to simplify the interface in data communications systems.

RS-232

The first of these, RS-232, was introduced in 1962 and has been widely used throughout the industry. RS-232 was developed for single-ended data transmission at relatively slow data rates (20 kBaud) over short distances (up to 50 ft.).

RS-423

With the need to transmit data faster and over longer distances, RS-423, a newer standard for single-ended applications, was established. RS-423 extends the maximum data rate to 100 kBaud (up to 30 ft.) and the maximum distance to 4000 feet (up to 1 kBaud). RS-423 also requires high impedance driver outputs with power off so as not to load the transmission line.

Differential Data Transmission

When transmitting at very high data rates, over long distances and through noisy environments, single-ended transmission is often inadequate. In these applications, differential data transmission offers superior performance. Differential transmission nullifies the effects of ground shifts and noise signals which appear as common mode voltages on the transmission line.

RS-422

RS-422 was defined by the EIA for this purpose and allows data rates up to 10 MBaud (up to 40 ft.) and line lengths up to 4000 feet (up to 100 kBaud).

Drivers designed to meet this standard are well suited for party-line type applications where only one driver is connected to, and transmits on, a bus and up to 10 receivers can receive the data. While a party-line type of application has many uses, RS-422 devices cannot be used to construct a truly multipoint bus. A multipoint bus consists of multiple drivers and receivers connected to a single bus, and any one of them can transmit or receive data.

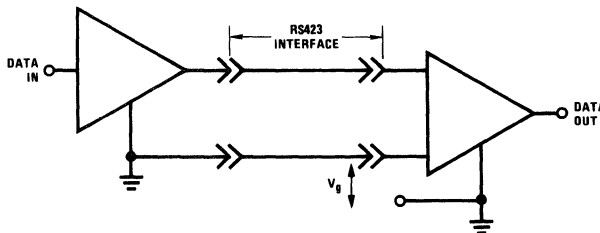### RS-485

To meet the need for truly multipoint communications, the EIA established RS-485 in 1983. RS-485 meets all the requirements of RS-422, but in addition, this new standard allows up to 32 drivers and 32 receivers to be connected to a single bus—thus allowing a truly multipoint bus to be constructed.
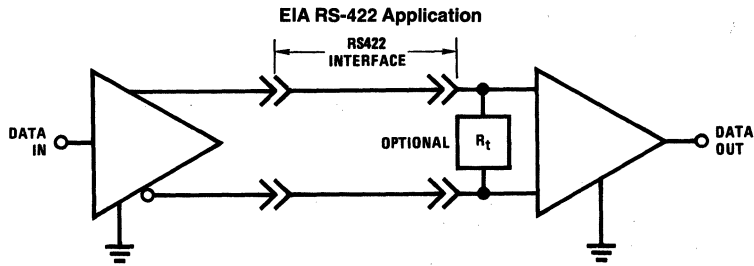
RS-232C Application
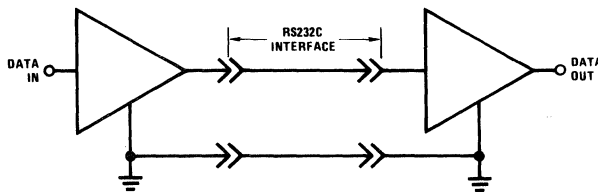
TL/00/2901–1

EIA RS-423 Application

TL/00/2901–2

6

Differential Data Transmission (Continued)

### EIA RS-422 Application

TL/00/2901–3

The key features of RS-485:
■ Implements a truly multipoint bus consisting of up to 32 drivers and 32 receivers
■ An extended common-mode range for both drivers and receivers in TRI-STATE and with power off (−7V to +12V)

■ Drivers can withstand bus contention and bus faults

National Semiconductor produces a variety of drivers, receivers, and transceivers for these four very popular transmission standards and numerous other data transmission requirements.

Shown below is a table that highlights key aspects of the EIA Standards. More detailed comparisons can be found in the various application notes in Section 1.

RS-485 Application
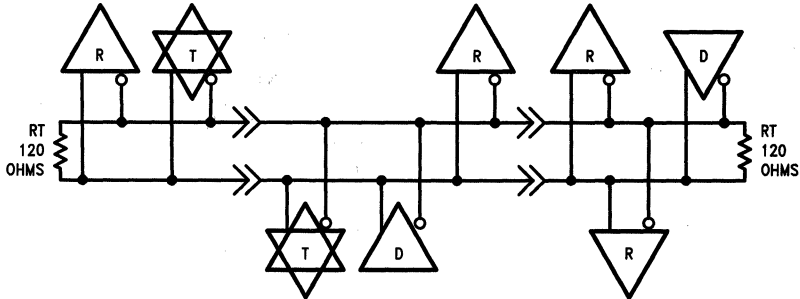
D — Driver
R — Receiver
T — Transceiver

TL/00/2901–4

Specification		RS-232C	RS-423	RS-422	RS-485
Mode of Operation		Single-Ended	Single-Ended	Differential	Differential
Number of Drivers and Receivers Allowed on One Line		1 Driver, 1 Receiver	1 Driver, 10 Receivers	1 Driver, 10 Receivers	32 Drivers, 32 Receivers
Maximum Cable Length		50 feet	4000 feet	4000 feet	4000 feet
Maximum Data Rate		20 kb/s	100 kb/s	10 Mb/s	10 Mb/s
Driver Output Maximum Voltage		±25V	±6V	−0.25V to +6V	−7V to +12V
Driver Output Signal Level	Loaded	±5V	±3.6V	±2V	±1.5V
	Unloaded	±15V	±6V	±5V	±5V
Driver Load Impedance		3 kΩ to 7 kΩ	450Ω min	100Ω	54Ω
Maximum Driver Output Current (High Impedance State)	Power On	— —	— —	— —	±100 μA
	Power Off	V_{MAX}/300Ω	±100 μA	±100 μA	±100 μA
Slew Rate		30 V/μs max	Controls Provided	— —	— —
Receiver Input Voltage Range		±15V	±12V	−7V to +7V	−7V to +12V
Receiver Input Sensitivity		±3V	±200 mV	±200 mV	±200 mV
Receiver Input Resistance		3 kΩ to 7 kΩ	4 kΩ min	4 kΩ min	12 kΩ min

![National Semiconductor logo]

DS1488 Quad Line Driver

General Description

The DS1488 is a quad line driver which converts standard TTL input logic levels through one stage of inversion to output levels which meet EIA Standard No. RS-232C and CCITT Recommendation V.24.

Features

- Current limited output   ±10 mA typ
- Power-off source impedance   300Ω min
- Simple slew rate control with external capacitor
- Flexible operating supply range
- Inputs are TTL/LS compatible

Schematic and Connection Diagrams

1/4 Circuit

TL/F/5776–1

Dual-In-Line Package

TL/F/5776–2

Top View
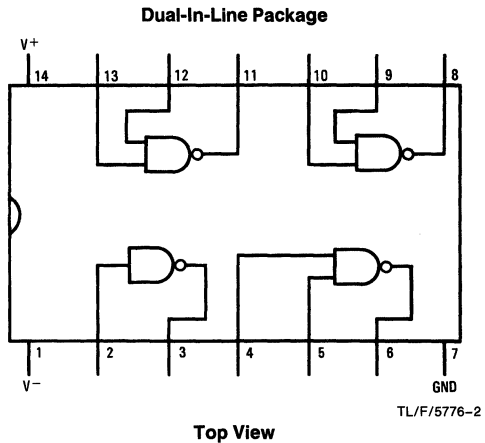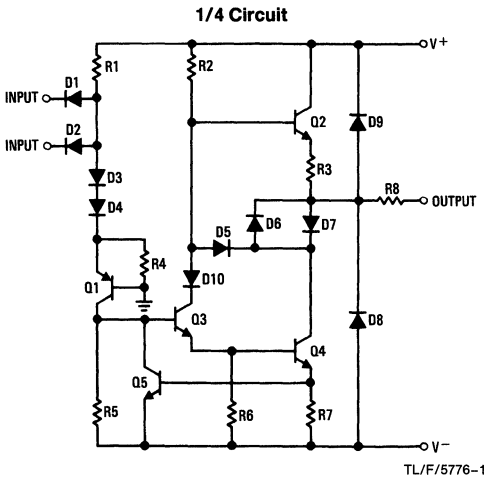
Order Number DS1488J, DS1488M or DS1488N
See NS Package Number J14A, M14A or N14A

Typical Applications

RS-232C Data Transmission

TL/F/5776–3

*Optional for noise filtering

National Semiconductor

DS14C88/DS14C89A Quad CMOS Line Driver/Receiver

General Description

The DS14C88 and DS14C89A, pin-for-pin replacements for the DS1488/MC1488 and the DS1489/MC1489, are line drivers/receivers designed to interface data terminal equipment (DTE) with data communications equipment (DCE). These devices translate standard TTL or CMOS logic levels to/from levels conforming to RS-232-C and CCITT V.24 standards.

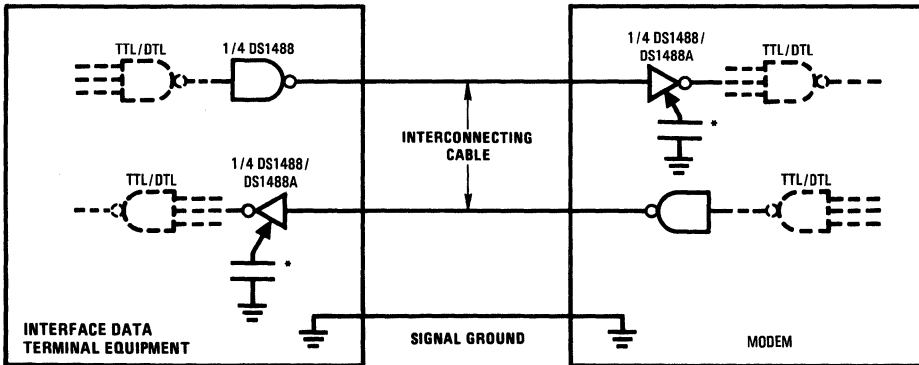Both devices are fabricated in low threshold CMOS metal gate technology. They provide very low power consumption in comparison to their bipolar equivalents; 900 μA versus 26 mA for the receiver and 500 μA versus 25 mA for the driver.

The DS14C88/DS14C89A simplify designs by eliminating the need for external capacitors. For the DS14C88, slew rate control in accordance with RS-232-C is provided on chip, eliminating the output capacitors. For the DS14C89A, noise pulse rejection circuitry eliminates the need for response control filter capacitors. When replacing the DS1489 with DS14C89A, the response control filter pins can be tied high, low or not connected.

Features

■ Meets EIA RS-232-C and CCITT V.24 standard
■ Low power consumption
■ Pin-for-pin equivalent to DS1488/MC1488 and DS1489/MC1489
■ Low Delay Slew
■ DS14C88 Driver
  — Power-off source impedance 300Ω min.
  — Wide operating voltage range: 4.5V–12.6V
  — TTL/LSTTL compatible
■ DS14C89A
  — Internal noise filter
  — Inputs withstand ±30V
  — Fail-safe operating mode
  — Internal input threshold with hysteresis

Connection Diagrams

DS14C88 Dual-In-Line Package

TL/F/8508–9
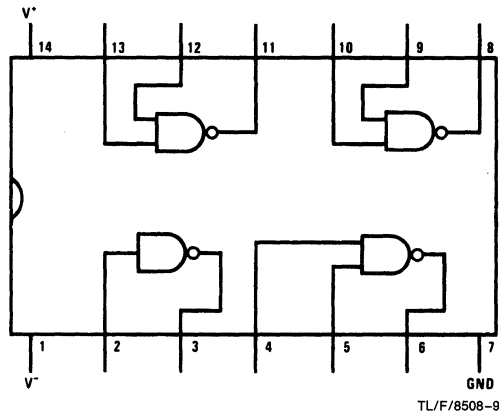
Order Number DS14C88J, DS14C88N and DS14C88M
See NS Package Number J14A, M14A or N14A

DS14C89A Dual-In-Line Package

TL/F/8508–2

Order Number DS14C89AJ, DS14C89AM or DS14C89AN
See NS Package Number J14A, M14A or N14A

![National Semiconductor]

DS1489/DS1489A Quad Line Receiver

General Description

The DS1489/DS1489A are quad line receivers designed to interface data terminal equipment with data communications equipment. They are constructed on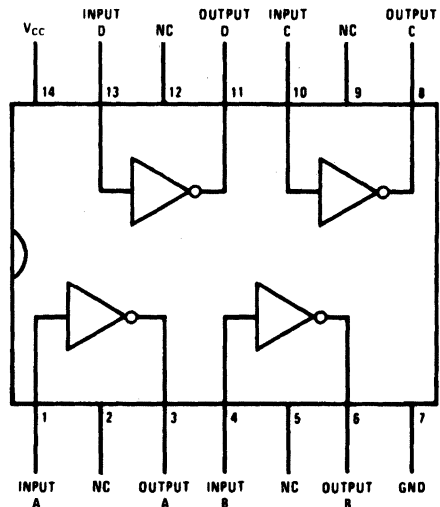 a single monolithic silicon chip. These devices satisfy the specifications of EIA Standard RS-232C. The DS1489/DS1489A meet and exceed the specifications of MC1489/MC1489A and are pin-for-pin replacements.

Features

■ Four totally separate receivers per package
■ Programmable threshold
■ Built-in input threshold hysteresis
■ "Fail safe" operating mode
■ Inputs withstand ±30V

Schematic and Connection Diagrams

(¼ of unit shown)

9k   5k   2k

R_F

RESPONSE CONTROL

V_{CC}

OUTPUT

INPUT   4k

10k

GND

DS1489: R_F = 10k
DS1489A: R_F = 2k

TL/F/5777–1

Dual-In-Line Package

V_{CC} INPUT D   RESPONSE CONTROL D   OUTPUT D   INPUT C   RESPONSE CONTROL C   OUTPUT C

14   13   12   11   10   9   8

1   2   3   4   5   6   7

INPUT A   RESPONSE CONTROL A   OUTPUT A   INPUT B   RESPONSE CONTROL B   OUTPUT B   GND

TL/F/5777–2
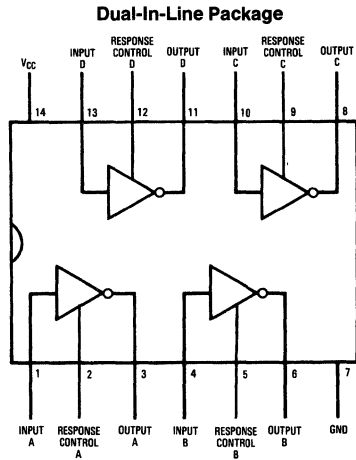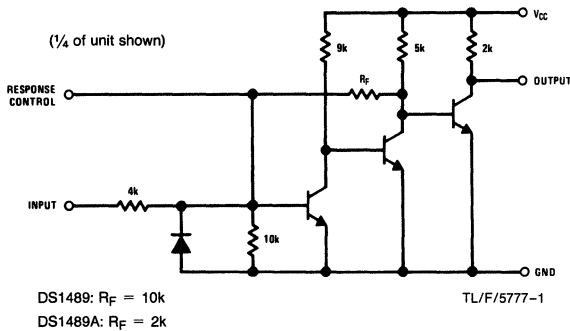
Top View
Order Number DS1489J, DS1489AJ,
DS1489M, DS1489AM, DS1489N or DS1489AN
See NS Package Number J14A, M14A or N14A

AC Test Circuit and Voltage Waveforms

RESPONSE CONTROL = OPEN   OUTPUT V_{CC}

R_L

PULSE GENERATOR

15 pF INCLUDING JIG AND PROBE

TL/F/5777–3

INPUT   3V   50%   0V

OUTPUT   90%   1.5V   10%

t_r t_r

t_{pd0} t_{pd1}

TL/F/5777–4

6

National Semiconductor
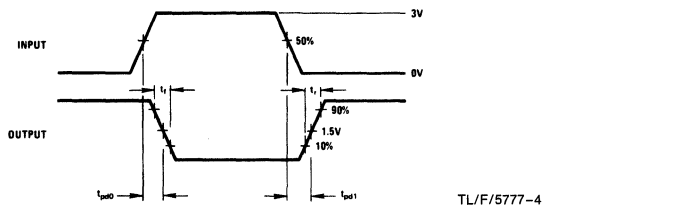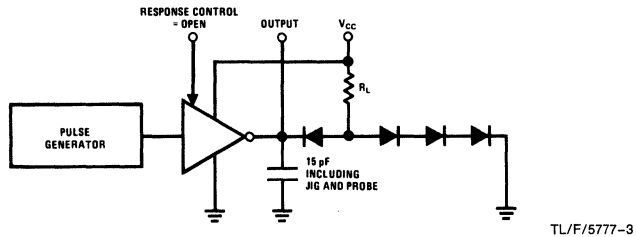
DS26LS31C/DS26LS31M Quad High Speed Differential Line Driver

General Description

The DS26LS31 is a quad differential line driver designed for digital data transmission over balanced lines. The DS26LS31 meets all the requirements of EIA Standard RS-422 and Federal Standard 1020. It is designed to provide unipolar differential drive to twisted-pair or parallel-wire transmission lines.

The circuit provides an enable and disable function common to all four drivers. The DS26LS31 features TRI-STATE® outputs and logically ANDed complementary outputs. The inputs are all LS compatible and are all one unit load.

The DS26LS31 features a power up/down protection circuit which keeps the output in a high impedance state (TRI-STATE) during power up or down preventing erroneous glitches on the transmission lines.

Features

- Output skew—2.0 ns typical
- Input to output delay—10 ns
- Operation from single 5V supply
- 16-pin hermetic and molded DIP package
- Outputs won't load line when $V_{CC} = 0V$
- Four line drivers in one package for maximum package density
- Output short-circuit protection
- Complementary outputs
- Meets the requirements of EIA Standard RS-422
- Pin compatible with AM26LS31
- Available in military and commercial temperature range
- Glitch free power up/down

Logic and Connection Diagrams
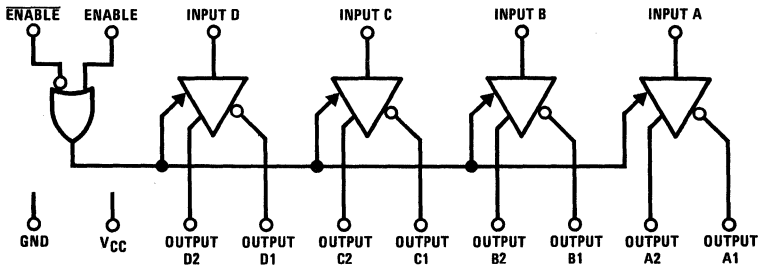
TL/F/5778-1

Dual-In-Line Package
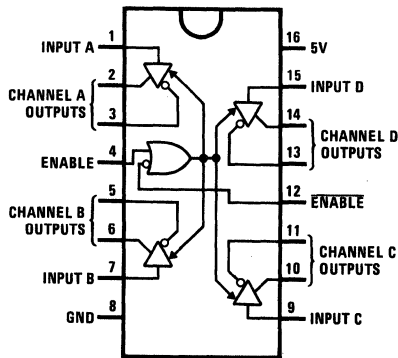
TL/F/5778-2

Top View
Order Number DS26LS31CJ, DS26LS31CM,
DS26LS31CN or DS26LS31MJ
See NS Package Number J16A, M16A or N16A

National Semiconductor

DS26C31C CMOS Quad TRI-STATE® Differential Line Driver

General Description

The DS26C31 is a quad differential line driver designed for digital data transmission over balanced lines. The DS26C31 meets all the requirements of EIA standard RS-422 while retaining the low power characteristics of CMOS. This enables the construction of serial and terminal interfaces while maintaining minimal power consumption.

The DS26C31 accepts TTL or CMOS input levels and translates these to RS-422 output levels. This part uses special output circuitry that enables the individual drivers to power down without loading down the bus. The DS26C31 also includes special power up and down circuitry which will TRI-STATE the outputs during power up or down, preventing spurious glitches on its outputs. This device has enable and disable circuitry common to all four drivers. The DS26C31 is pin compatible to the AM26LS31 and the DS26LS31.

All inputs are protected against damage due to electrostatic discharge by diodes to V_{CC} and ground.

Features

■ TTL input compatible
■ Typical propagation delays: 6 ns
■ Typical output skew: 0.5 ns
■ Outputs won't load line when $V_{CC} = 0V$
■ Meets the requirements of EIA standard RS-422
■ Operation from single 5V supply
■ TRI-STATE outputs for connection to system buses
■ Low quiescent current
■ Available in surface mount

Connection Diagram

Dual-In-Line Package
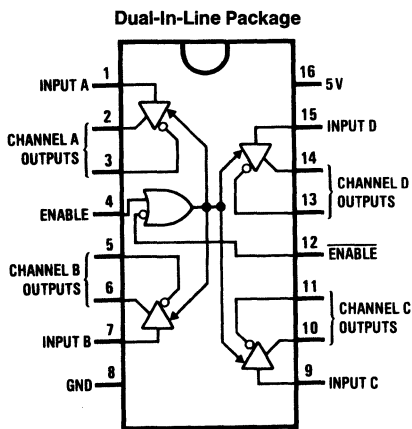
TL/F/8574–1

Top View

**Order Number DS26C31CJ, DS26C31CM or DS26C31CN
See NS Package Number J16A, M16A or N16A**

Truth Table

Active High Enable	Active Low Enable	Input	Non-Inverting Output	Inverting Output
L	H	X	Z	Z
All other combinations of enable inputs		L	L	H
		H	H	L

L = Low logic state
H = High logic state
X = Irrelevant
Z = TRI-STATE (high impedance)

6

National Semiconductor

DS26LS32C/DS26LS32M/DS26LS32AC/DS26LS33C/DS26LS33M/DS26LS33AC Quad Differential Line Receivers

General Description

The DS26LS32 and DS26LS32A are quad differential line receivers designed to meet the RS-422, RS-423 and Federal Standards 1020 and 1030 for balanced and unbalanced digital data transmission.

The DS26LS32 and DS26LS32A have an input sensitivity of 200 mV over the input voltage range of ±7V and the DS26LS33 and DS26LS33A have an input sensitivity of 500 mV over the input voltage range of ±15%V.

Both the DS26LS32A and DS26LS33A differ in function from the popular DS26LS32 and DS26LS33 in that input fail-safe circuitry is provided for each receiver, which causes the outputs to go to a logic "1" state when the inputs are open.

Each version provides an enable and disable function common to all four receivers and features TRI-STATE® outputs with 8 mA sink capability. Constructed using low power Schottky processing, these devices are available over the full military and commerical operating temperature ranges.

Features

■ High differential or common-mode input voltage ranges of ±7V on the DS26LS32 and DS26LS32A and ±15V on the DS26LS33 and DS26LS33A

■ ±0.2V sensitivity over the input voltage range on the DS26LS32 and DS26LS32A, ±0.5V sensitivity on the DS26LS33 and DS26LS33A

■ Input fail-safe circuitry on the DS26LS32A and DS26LS33A

■ DS26LS32 and DS26LS32A meet all requirements of RS-422 and RS-423

■ 6k minimum input impedance

■ 100 mV input hysteresis on the DS26LS32 and DS26LS32A, 200 mV on the DS26LS33 and DS26LS33A

■ Operation from a single 5V supply

■ TRI-STATE drive, with choice of complementary output enables for receiving directly onto a data bus

■ Pin replacement for Advanced Micro Devices AM26LS32

Logic Diagram
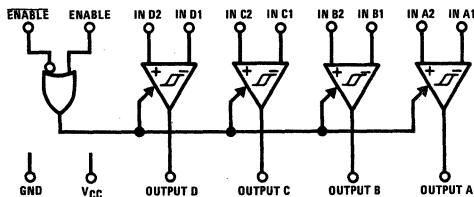
TL/F/5255–1

Connection Diagram

Dual-In-Line Package

TL/F/5255–2

Top View
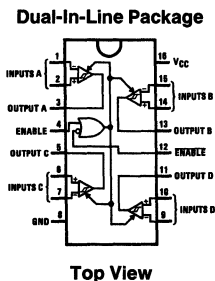
Truth Table

ENABLE	$\overline{\text{ENABLE}}$	Input	Output
1	1	X	Hi-Z
See Note Below		$V_{ID} \geq V_{TH}$ (Max)	1
		$V_{ID} \leq V_{TH}$ (Min)	0
		Open	1*

HI-Z = TRI-STATE

*DS26LS32A and DS26LS33A only

Note: Input conditions may be any combination not defined for ENABLE and $\overline{\text{ENABLE}}$.

Order Number DS26LS32MJ, DS26LS32CJ, DS26LS32CM, DS26LS32CN, DS26LS32ACJ, DS26LS32ACN, DS26LS32ACM, DS26LS33MJ, DS26LS33CJ, DS26LS33CN, DS26LS33ACJ or DS26LS33ACN
See NS Package Number J16A, M16A or N16A

National Semiconductor

DS26C32AC Quad Differential Line Receiver

General Description

The DS26C32A is a quad differential line receiver designed to meet the RS-422, RS-423, and Federal Standards 1020 and 1030 for balanced and unbalanced digital data transmission, while retaining the low power characteristics of CMOS.

The DS26C32A has an input sensitivity of 200 mV over the common mode input voltage range of ±7V. Each receiver is also equipped with input fail-safe circuitry, which causes the output to go to a logic "1" state when the inputs are open.

The DS26C32A provides an enable and disable function common to all four receivers, and features TRI-STATE® outputs with 6 mA source and sink capability. This product is pin compatible with the DS26LS32A and the AM26LS32.

Features

■ Low power CMOS design
■ ±0.2V sensitivity over the entire common mode range
■ Typical propagation delays: 19 ns
■ Typical input hysteresis: 60 mV
■ Input fail-safe circuitry
■ Inputs won't load line when $V_{CC} = 0V$
■ Meets the requirements of EIA standard RS-422
■ TRI-STATE outputs for connection to system buses
■ Available in Surface Mount

Logic Diagram
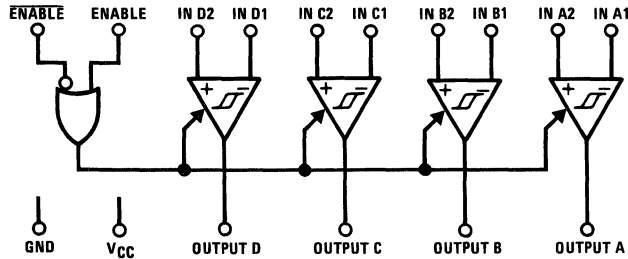
TL/F/8764–1

Connection Diagram

Dual-In-Line Package
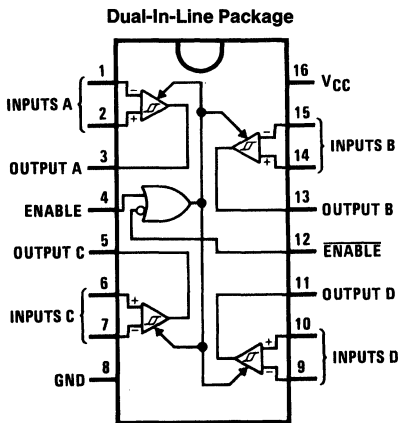
TL/F/8764–2

Top View

Order Number DS26C32ACJ,
DS26C32ACM or DS26C32ACN
See NS Package J16A, M16A or N16A

Truth Table

ENABLE	ENABLE	Input	Output
0	1	X	Hi-Z
See Note Below		$V_{ID} \geq V_{TH}$ (Max)	1
		$V_{ID} \leq V_{TH}$ (Min)	0
		Open	1

Hi-Z = TRI-STATE

Note: Input conditions may be any combination not defined for ENABLE and ENABLE.

National Semiconductor

DS3486 Quad RS-422, RS-423 Line Receiver

General Description

National's quad RS-422, RS-423 receiver features four independent receiver chains which comply with EIA Standards for the electrical characteristics of balanced/unbalanced voltage digital interface circuits. Receiver outputs are 74LS compatible, TRI-STATE® structures which are forced to a high impedance state when the appropriate output control pin reaches a logic zero condition. A PNP device buffers each output control pin to assure minimum loading for either logic one or logic zero inputs. In addition, each receiver chain has internal hysteresis circuitry to improve noise margin and discourage output instability for slowly changing input waveforms.

Features

■ Four independent receiver chains
■ TRI-STATE outputs
■ High impedance output control inputs (PIA compatible)
■ Internal hysteresis −140 mV (typ)
■ Fast propagation times −18 ns (typ)
■ TTL compatible
■ Single 5V supply voltage
■ Pin compatible and interchangeable with MC3486

Block and Connection Diagrams
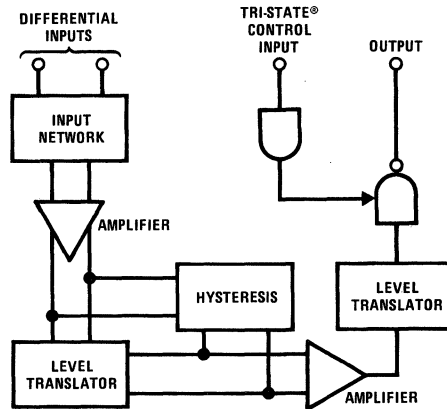
TL/F/5779–1

Dual-In-Line Package
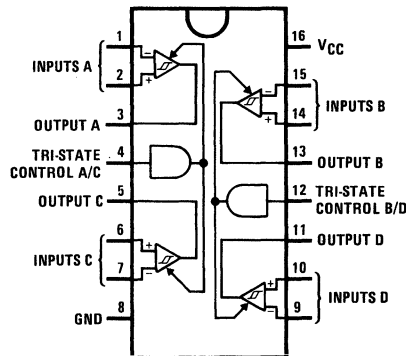
TL/F/5779–2

Top View
Order Number DS3486J, DS3486M or DS3486N
See NS Package Number J16A, M16A or N16A

![National Semiconductor logo] **National Semiconductor**

DS34C86
Quad CMOS Differential Line Receiver

General Description

The DS34C86 is a quad differential line receiver designed to meet the RS-422, RS-423, and Federal Standards 1020 and 1030 for balanced and unbalanced digital data transmission, while retaining the low power characteristics of CMOS.

The DS34C86 has an input sensitivity of 200 mV over the common mode input voltage range of ± 7V. Hysteresis is provided to improve noise margin and discourage output instability for slowly changing input waveforms.

Separate enable pins allow independent control of receiver pairs. The TRI-STATE® outputs have 6 mA source and sink capability. The DS34C86 is pin compatible with the DS3486.

Features

■ Low power CMOS design
■ ± 0.2V sensitivity over the entire common mode range
■ Typical propagation delays: 19 ns
■ Typical input hysteresis: 60 mV
■ Inputs won't load line when $V_{CC} = 0$V
■ Meets the requirements of EIA standard RS-422
■ TRI-STATE outputs for connection to system buses
■ Available in surface mount

Logic Diagram
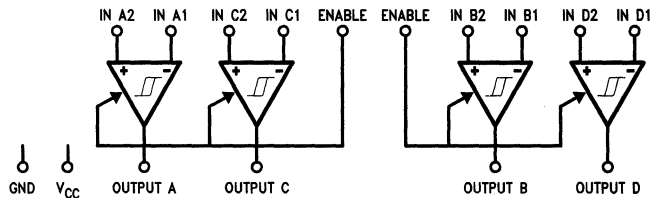
TL/F/8699–1

Connection Diagram

Dual-In-Line Package

TL/F/8699–2

Top View
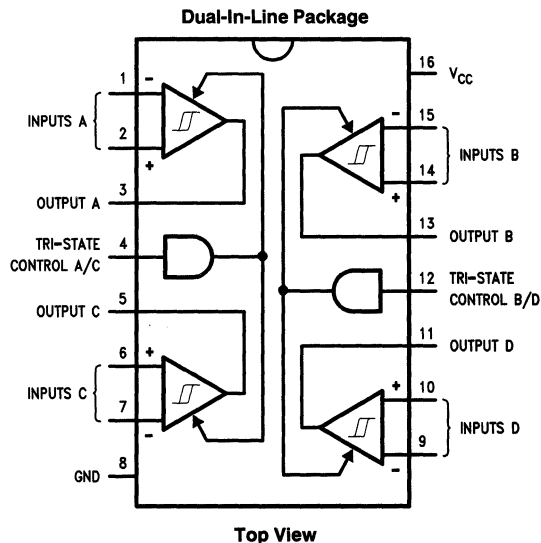
Order Number DS34C86J, DS34C86M, and DS34C86N
See NS Package Number J16A, M16A and N16A

6

National Semiconductor

DS3587/DS3487 Quad TRI-STATE® Line Driver

General Description

National's quad RS-422 driver features four independent driver chains which comply with EIA Standards for the electrical characteristics of balanced voltage digital interface circuits. The outputs are TRI-STATE structures which are forced to a high impedance state when the appropriate output control pin reaches a logic zero condition. All input pins are PNP buffered to minimize input loading for either logic one or logic zero inputs. In addition, internal circuitry assures a high impedance output state during the transition between power up and power down.

Features

■ Four independent driver chains
■ TRI-STATE outputs
■ PNP high impedance inputs (PIA compatible)
■ Power up/down protection
■ Fast propagation times (typ 10 ns)
■ TTL compatible
■ Single 5V supply voltage
■ Output rise and fall times less than 20 ns (typ 10 ns)
■ Pin compatible with DS8924 and MC3487
■ Output skew—2 ns typ

Block and Connection Diagrams

TL/F/5780–1

Dual-In-Line Package
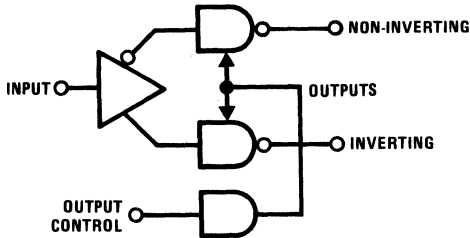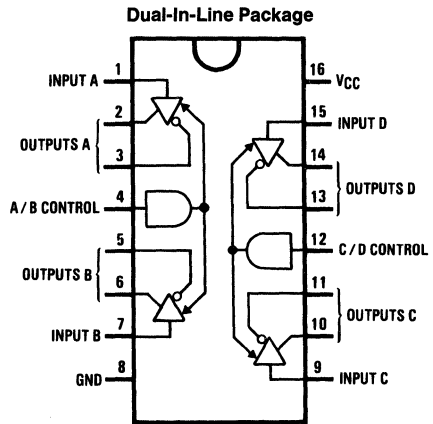
TL/F/5780–2

Top View

Order Number DS3587J, DS3487J,
DS3487M or DS3487N
See NS Package Number J16A, M16A or N16A

Truth Table

Input	Control Input	Non-Inverter Output	Inverter Output
H	H	H	L
L	H	L	H
X	L	Z	Z

L = Low logic state
H = High logic state
X = Irrelevant
Z = TRI-STATE (high impedance)

National Semiconductor

DS34C87 CMOS Quad TRI-STATE®
Differential Line Driver

General Description

The DS34C87 is a quad differential line driver designed for digital data transmission over balanced lines. The DS34C87 meets all the requirements of EIA standard RS-422 while retaining the low power characteristics of CMOS. This enables the construction of serial and terminal interfaces while maintaining minimal power consumption.

The DS34C87 accepts TTL or CMOS input levels and translates these to RS-422 output levels. This part uses special output circuitry that enables the individual drivers to power down without loading down the bus. The DS34C87 also includes special power up and down circuitry which will TRI-STATE the outputs during power up or down, preventing spurious glitches on its outputs. This device has separate enable circuitry for each pair of the four drivers. The DS34C87 is pin compatible to the DS3487.

All inputs are protected against damage due to electrostatic discharge by diodes to V_{CC} and ground.

Features

- TTL input compatible
- Typical propagation delays: 6 ns
- Typical output skew: 0.5 ns
- Outputs won't load line when $V_{CC} = 0V$
- Meets the requirements of EIA standard RS-422
- Operation from single 5V supply
- TRI-STATE outputs for connection to system buses
- Low quiescent current
- Available in surface mount

Connection and Logic Diagrams

Dual-In-Line Package

TL/F/8576–1
Top View

Order Number DS34C87J, DS34C87M or DS34C87N
See NS Package Number J16A, M16A or N16A

TL/F/8576–2
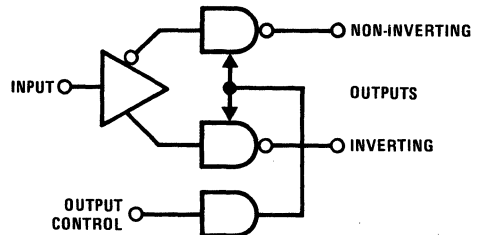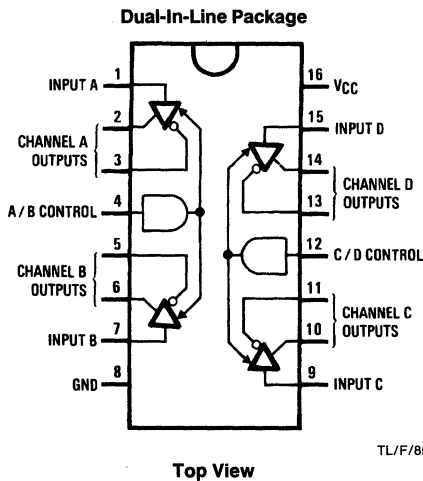
Truth Table

Input	Control Input	Non-Inverting Output	Inverting Output
H	H	H	L
L	H	L	H
X	L	Z	Z

L = Low logic state X = Irrelevant
H = High logic state Z = TRI-STATE (high impedance)

6

National Semiconductor

DS1691A/DS3691 (RS-422/RS-423) Line Drivers with TRI-STATE® Outputs

General Description

The DS1691A/DS3691 are low power Schottky TTL line drivers designed to meet the requirements of EIA standards RS-422 and RS-423. They feature 4 buffered outputs with high source and sink current capability with internal short circuit protection. A mode control input provides a choice of operation either as 4 independent line drivers or 2 differential line drivers. A rise time control pin allows the use of an external capacitor to reduce rise time for suppression of near end crosstalk to other receivers in the cable.

With the mode select pin low, the DS1691A/DS3691 are dual-differential line drivers with TRI-STATE outputs. They feature ±10V output common-mode range in TRI-STATE mode and 0V output unbalance when operated with ±5V supply.

Features

■ Dual RS-422 line driver with mode pin low, or quad RS-423 line driver with mode pin high
■ TRI-STATE control for individual outputs
■ Short circuit protection for both source and sink outputs
■ Outputs will not clamp line with power off or in TRI-STATE
■ Individual rise mode time control for each output
■ 100Ω transmission line drive capability
■ Low I_{CC} and I_{EE} power consumption
  RS-422   35 mW/driver typ
  RS-423   26 mW/driver typ
■ Low current PNP inputs compatible with TTL, MOS and CMOS
■ Pin compatible with AM26LS30

Connection Diagram

With Mode Select LOW (RS-422 Connection)

Pin	Signal		Pin	Signal
1	V_{CC}		16	RISE TIME CONTROL A
2	INPUT A		15	OUTPUT A
3	INPUT B/DISABLE		14	OUTPUT B
4	MODE SELECT		13	RISE TIME CONTROL B
5	GND		12	RISE TIME CONTROL C
6	INPUT C/DISABLE		11	OUTPUT C
7	INPUT D		10	OUTPUT D
8	V_{EE}		9	RISE TIME CONTROL D

TL/F/5783–1

Top View

With Mode Select HIGH (RS-423 Connection)

Pin	Signal		Pin	Signal
1	V_{CC}		16	RISE TIME CONTROL A
2	INPUT A		15	OUTPUT A
3	INPUT B/DISABLE		14	OUTPUT B
4	MODE SELECT		13	RISE TIME CONTROL B
5	GND		12	RISE TIME CONTROL C
6	INPUT C/DISABLE		11	OUTPUT C
7	INPUT D		10	OUTPUT D
8	V_{EE}		9	RISE TIME CONTROL D

TL/F/5783–2

Top View

Truth Table

Operation	Inputs			Outputs	
	Mode	A (D)	B (C)	A (D)	B (C)
RS-422	0	0	0	0	1
	0	0	1	TRI-STATE	TRI-STATE
	0	1	0	1	0
	0	1	1	TRI-STATE	TRI-STATE
RS-423	1	0	0	0	0
	1	0	1	0	1
	1	1	0	1	0
	1	1	1	1	1

Order Number DS1691AJ, DS3691J, DS3691M or DS3691N
See NS Package Number J16A, M16A or N16A

National Semiconductor

DS1692/DS3692 TRI-STATE® Differential Line Drivers

General Description

The DS1692/DS3692 are low power Schottky TTL line drivers electrically similar to the DS1691A/DS3691 but tested to meet the requirements of MIL-STD-188-114 (see Application Note AN-216). They feature 4 buffered outputs with high source and sink current capability with internal short circuit protection. A mode control input provides a choice of operation either as 4 independent line drivers or 2 differential line drivers. A rise time control pin allows the use of an external capacitor to reduce rise time for suppression of near end cross-talk to other receivers in the cable.

With the mode select pin low, the DS1692/DS3692 are dual differential line drivers with TRI-STATE outputs. They feature ±10V output common-mode range in TRI-STATE and 0V output unbalance when operated with ±5V supply.
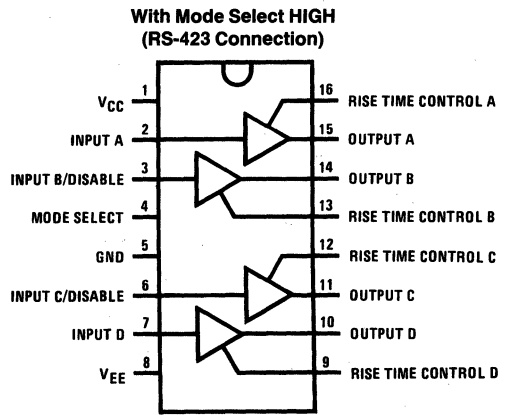
Features

- Dual differential line driver or quad single-ended line driver
- TRI-STATE differential drivers meet MIL-STD-188-114
- Short circuit protection for both source and sink outputs
- Individual rise time control for each output
- 100Ω transmission line drive capability
- Low I_{CC} and I_{EE} power consumption
  Differential mode   35 mW/driver typ
  Single-ended mode   26 mW/driver typ
- Low current PNP inputs compatible with TTL, MOS and CMOS

Logic Diagram (½ Circuit Shown)
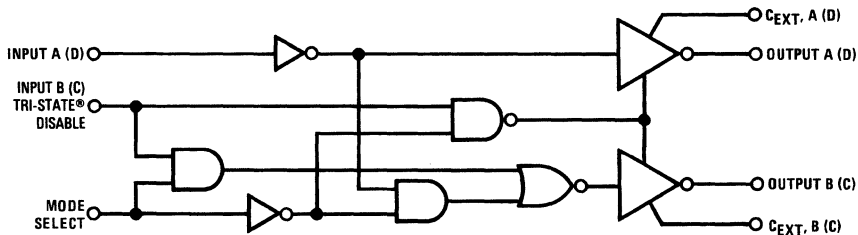
TL/F/5784–1

Connection Diagram

TL/F/5784–2

Top View

Order Number DS1692J, DS3692J or DS3692N
See NS Package Number J16A or N16A
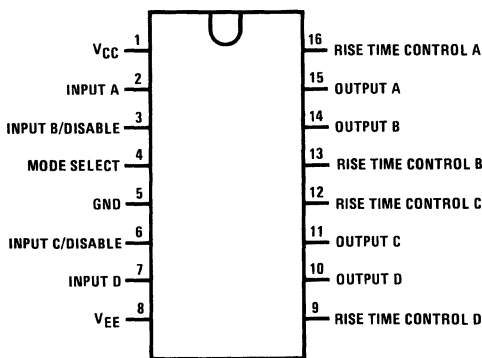
Truth Table

Inputs			Outputs	
Mode	A (D)	B (C)	A (D)	B (C)
0	0	0	0	1
0	0	1	TRI-STATE	TRI-STATE
0	1	0	1	0
0	1	1	TRI-STATE	TRI-STATE
1	0	0	0	0
1	0	1	0	1
1	1	0	1	0
1	1	1	1	1

6

National Semiconductor

DS3695A/DS3695AT
Multipoint RS485/RS422 Transceiver

General Description

The DS3695A is a high speed differential TRI-STATE® bus/line transceiver designed to meet the requirements of EIA standard RS485 with extended common mode range (+12V to −7V), for multipoint data transmission. In addition, it is compatible with the requirements of RS422.

The driver and receiver outputs feature TRI-STATE capability. The driver outputs remain in TRI-STATE over the entire common mode range of +12V to −7V. Bus faults that cause excessive power dissipation within the device trigger a thermal shutdown circuit, which forces the driver outputs into the high impedance state.

The receiver incorporates a fail safe feature which guarantees a high output state when the inputs are left open.

Both AC and DC specifications are guaranteed over the 0 to 70°C temperature and 4.75V to 5.25V supply voltage range.

Features

■ Meets EIA standard RS485 for multipoint bus transmission and is compatible with RS-422
■ 15 ns driver propagation delays with 2 ns skew (typical)
■ Single +5V supply
■ −7V to +12V bus common mode range permits ±7V ground difference between devices on the bus
■ Thermal shutdown protection
■ Power-up/down glitch-free driver outputs permit live insertion or removal of transceivers
■ High impedance to bus with driver in TRI-STATE or with power off, over the entire common mode range allows the unused devices on the bus to be powered down
■ Combined impedance of a driver output and receiver input is less than one RS485 unit load, allowing up to 32 transceivers on the bus
■ 70 mV typical receiver hysteresis

Connection and Logic Diagram

Molded Package, Small Outline (M)
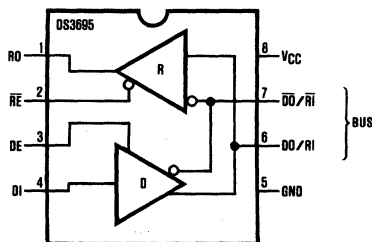
TL/F/5272–1

Top View

Order Number DS3695AM DS3695ATM
See NS Package Number M08A

![National Semiconductor logo]

DS75150 Dual Line Driver

General Description

The DS75150 is a dual monolithic line driver designed to satisfy the requirements of the standard interface between data terminal equipment and data communication equipment as defined by EIA Standard RS-232-C. A rate of 20,000 bits per second can be transmitted with a full 2500 pF load. Other applications are in data-transmission systems using relatively short single lines, in level translators, and for driving MOS devices. The logic input is compatible with most TTL and LS families. Operation is from −12V and +12V power supplies.

Features

- Withstands sustained output short-circuit to any low impedance voltage between −25V and +25V
- 2 μs max transition time through the −3V to +3V transition region under full 2500 pF load
- Inputs compatible with most TTL and LS families
- Common strobe input
- Inverting output
- Slew rate can be controlled with an external capacitor at the output
- Standard supply voltages   ±12V

Schematic and Connection Diagrams

Component values shown are nominal.
1/2 of circuit shown

TL/F/5794–1

Dual-In-Line Package

TL/F/5794–2

Top View

Positive Logic C = $\overline{AS}$

Order Number DS75150J-8, DS75150M or DS75150N See NS Package Number J08A, M08A or N08E

National Semiconductor

DS75154 Quad Line Receiver

General Description

The DS75154 is a quad monolithic line receiver designed to satisfy the requirements of the standard interface between data terminal equipment and data communication equipment as defined by EIA Standard RS-232C. Other applications are in relatively short, single-line, point-to-point data transmission systems and for level translators. Operation is normally from a single 5V supply; however, a built-in option allows operation from a 12V supply without the use of additional components. The output is compatible with most TTL and LS circuits when either supply voltage is used.

In normal operation, the threshold-control terminals are connected to the V_{CC1} terminal, pin 15, even if power is being supplied via the alternate V_{CC2} terminal, pin 16. This provides a wide hysteresis loop which is the difference between the positive-going and negative-going threshold voltages. In this mode, if the input voltage goes to zero, the output voltage will remain at the low or high level as determined by the previous input.

For fail-safe operation, the threshold-control terminals are open. This reduces the hysteresis loop by causing the nega-

tive-going threshold voltage to be above zero. The positive-going threshold voltage remains above zero as it is unaffected by the disposition of the threshold terminals. In the fail-safe mode, if the input voltage goes to zero or an open-circuit condition, the output will go to the high level regardless of the previous input condition.

Features

- Input resistance, 3 kΩ to 7 kΩ over full RS-232C voltage range
- Input threshold adjustable to meet "fail-safe" requirements without using external components
- Inverting output compatible with TTL or LS
- Built-in hysteresis for increased noise immunity
- Output with active pull-up for symmetrical switching speeds
- Standard supply voltage—5V or 12V

Schematic Diagram
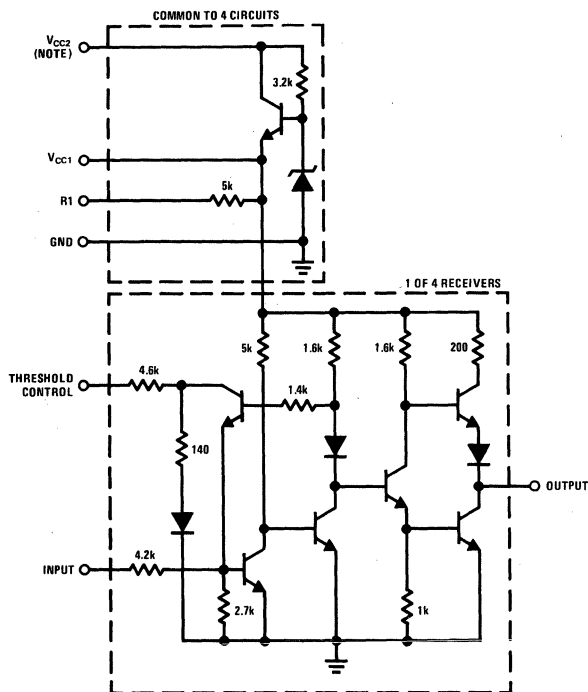
TL/F/5795–1

Note: When using V_{CC1} (pin 15), V_{CC2} (pin 16) may be left open or shorted to V_{CC1}. When using V_{CC2}, V_{CC1} must be left open or connected to the threshold control pins.

National Semiconductor

DS75176B/DS75176BT
Multipoint RS-485/RS-422 Transceivers

General Description

The DS75176B is a high speed differential TRI-STATE® bus/line transceiver designed to meet the requirements of EIA standard RS485 with extended common mode range ($+12V$ to $-7V$), for multipoint data transmission. In addition, it is compatible with RS-422.

The driver and receiver outputs feature TRI-STATE capability, for the driver outputs over the entire common mode range of $+12V$ to $-7V$. Bus contention or fault situations that cause excessive power dissipation within the device are handled by a thermal shutdown circuit, which forces the driver outputs into the high impedance state.

The receiver incorporates a fail safe feature which guarantees a high output state when the inputs are left open.

DC specifications are guaranteed over the 0 to 70°C temperature and 4.75V to 5.25V supply voltage range.

Features

■ Meets EIA standard RS485 for multipoint bus transmission and is compatible with RS-422.
■ Small Outline (SO) Package option available for minimum board space.
■ 22 ns driver propagation delays.
■ Single channel per package isolates faulty channels (from shutting down good channels).
■ Single $+5V$ supply.
■ $-7V$ to $+12V$ bus common mode range permits $\pm7V$ ground difference between devices on the bus.
■ Thermal shutdown protection.
■ Power-up down glitch-free driver outputs permit live insertion or removal of transceivers.
■ High impedance to bus with driver in TRI-STATE or with power off, over the entire common mode range allows the unused devices on the bus to be powered down.
■ Pin out compatible with DS3695 and SN75176A/B.
■ Combined impedance of a driver output and receiver input is less than one RS485 unit load, allowing up to 32 transceivers on the bus.
■ 70 mV typical receiver hysteresis.

Connection and Logic Diagram
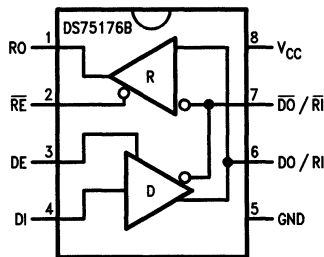
TL/F/8759–1

Top View

Order Number DS75176BN, DS75176BM, DS75176BTM
DS75176BJ, DS75176BTN or DS75176BTJ
See NS Package Number N08E, M08A or J08A

6

National Semiconductor

DS78C120/DS88C120 Dual CMOS Compatible Differential Line Receiver

General Description

The DS78C120 and DS88C120 are high performance, dual differential, CMOS compatible line receivers for both balanced and unbalanced digital data transmission. The inputs are compatible with EIA, Federal and MIL standards.

Input specifications meet or exceed those of the popular DS7820/DS8820 line receiver.

The line receiver will discriminate a ±200 mV input signal over a common-mode range of ±10V and a ±300 mV signal over a range of ±15V.

Circuit features include hysteresis and response control for applications where controlled rise and fall times and/or high frequency noise rejection are desirable. Threshold offset control is provided for fail-safe detection, should the input be open or short. Each receiver includes a 180Ω terminating resistor and the output gate contains a logic strobe for time discrimination. The DS78C120 is specified over a −55°C to +125°C temperature range and the DS88C120 from 0°C to +70°C.

Features

- Full compatibility with EIA Standards RS232-C, RS422 and RS423, Federal Standards 1020, 1030 and MIL-188-114
- Input voltage range of ±15V (differential or common-mode)
- Separate strobe input for each receiver
- 1/2 V_{CC} strobe threshold for CMOS compatibility
- 5k typical input impedance
- 50 mV input hysteresis
- 200 mV input threshold
- Operation voltage range = 4.5V to 15V
- Separate fail-safe mode

Connection Diagram

Dual-In-Line Package

Top View

TL/F/5801–1

Order Number DS78C120J, DS88C120J or DS88C120N
See NS Package Number J16A or N16A

National Semiconductor

DS78LS120/DS88LS120 Dual Differential Line Receiver (Noise Filtering and Fail-Safe)

General Description

The DS78LS120 and DS88LS120 are high performance, dual differential, TTL compatible line receivers for both balanced and unbalanced digital data transmission. The inputs are compatible with EIA, Federal and MIL standards.
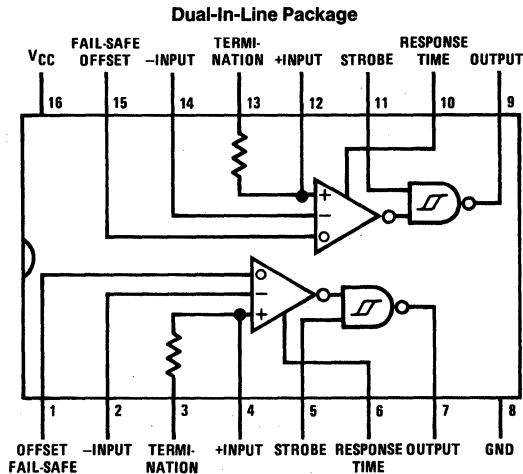
The line receiver will discriminate a ±200 mV input signal over a common-mode range of ±10V and a ±300 mV signal over a range of ±15V.

Circuit features include hysteresis and response control for applications where controlled rise and fall times and/or high frequency noise rejection are desirable. Threshold offset control is provided for fail-safe detection, should the input be open or short. Each receiver includes an optional 180Ω terminating resistor and the output gate contains a logic strobe for time discrimination. The DS78LS120 is specified over a −55°C to +125°C temperature range and the DS88LS120 from 0°C to +70°C.

Input specifications meet or exceed those of the popular DS7820/DS8820 line receiver.

Features

- Meets EIA standards RS232-C, RS422 and RS423, Federal Standards 1020, 1030 and MIL-188-114
- Input voltage range of ±15V (differential or common-mode)
- Separate strobe input for each receiver
- 5k typical input impedance
- Optional 180Ω termination resistor
- 50 mV input hysteresis
- 200 mV input threshold
- Separate fail-safe mode

Connection Diagram

Dual-In-Line Package
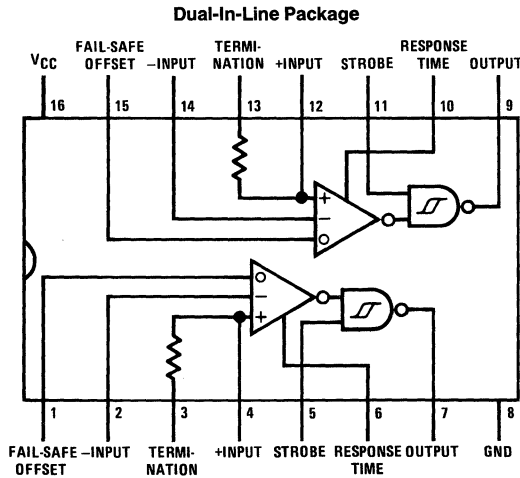
Top View

TL/F/7499–1

Order Number DS78LS120J, DS88LS120J or DS88LS120N
See NS Package Number J16A or N16A

6

National Semiconductor

DS8921/DS8921A Differential Line Driver and Receiver Pair

General Description

The DS8921, DS8921A are Differential Line Driver and Receiver pairs designed specifically for applications meeting the ST506, ST412 and ESDI Disk Drive Standards. In addition, these devices meet the requirements of the EIA Standard RS-422.

The DS8921A receiver offers an input sensitivity of 200 mV over a $\pm 7V$ common mode operating range. Hysteresis is incorporated (typically 70 mV) to improve noise margin for slowly changing input waveforms. An input fail-safe circuit is provided such that if the receiver inputs are open the output assumes the logical one state.

The DS8921A driver is designed to provide unipolar differential drive to twisted pair or parallel wire transmission lines. Complementary outputs are logically ANDed and provide an output skew of 0.5 ns (typ.) with propagation delays of 12 ns.

Power up/down circuitry is featured which will TRI-STATE® the outputs and prevent erroneous glitches on the trans-mission lines during system power up or power down operation.

The DS8921A is designed to be compatible with TTL and CMOS.

Features

- 12 ns typical propagation delay
- Output skew - 0.5 ns typical
- Meet the requirements of EIA Standard RS-422
- Complementary Driver Outputs
- High differential or common-mode input voltage ranges of $\pm 7V$
- $\pm 0.2V$ receiver sensitivity over the input voltage range
- Receiver input fail-safe circuitry
- Receiver input hysteresis-70 mV typical
- Glitch free power up/down

Connection Diagram
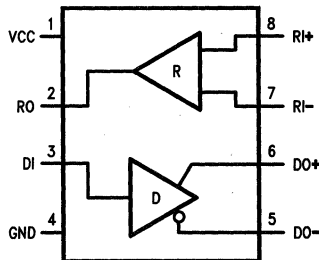
TL/F/8512-1

Order Number DS8921M, DS8921N, DS8921AM, DS8921AN, DS8921J or DS8921AJ
See NS Package Number J08A, M08A or N08E

Truth Table

Receiver		Driver		
Input	V_{OUT}	Input	V_{OUT}	$\overline{V_{OUT}}$
$V_{ID} \geq V_{TH}$ (MAX)	1	1	1	0
$V_{ID} \leq V_{TH}$ (MIN)	0	0	0	1

For complete specifications see the Interface Databook.

National Semiconductor

DS8922/22A/DS8923/23A TRI-STATE® RS-422 Dual Differential Line Driver and Receiver Pairs

General Description

The DS8922/22A and DS8923/23A are Dual Differential Line Driver and Receiver pairs. These devices are designed specifically for applications meeting the ST506, ST412 and ESDI Disk Drive Standards. In addition, the devices meet the requirements of the EIA Standard RS-422.

These devices offer an input sensitivity of 200 mV over a ±7V common mode operating range. Hysteresis is incorporated (typically 70 mV) to improve noise margin for slowly changing input waveforms. An input fail-safe circuit is provided such that if the receiver inputs are open the output assumes the logical one state.

The DS8922A and DS8923A drivers are designed to provide unipolar differential drive to twisted pair or parallel wire transmission lines. Complementary outputs are logically ANDed and provide an output skew of 0.5 ns (typ.) with propagation delays of 12 ns.

Both devices feature TRI-STATE outputs. The DS8922/22A have independent control functions common to a driver and receiver pair. The DS8923/23A have separate driver and receiver control functions.

Power up/down circuitry is featured which will TRI-STATE the outputs and prevent erroneous glitches on the transmission lines during system power up or power down operation.

The DS8922/22A and DS8923/23A are designed to be compatible with TTL and CMOS.

Features

- 12 ns typical propagation delay
- Output skew—±0.5 ns typical
- Meets the requirements of EIA Standard RS-422
- Complementary Driver Outputs
- High differential or common-mode input voltage ranges of ±7V
- ±0.2V receiver sensitivity over the input voltage range
- Receiver input fail-safe circuitry
- Receiver input hysteresis—±70 mV typical
- Glitch free power up/down
- TRI-STATE outputs

Connection Diagrams

DS8922A Dual-In-Line
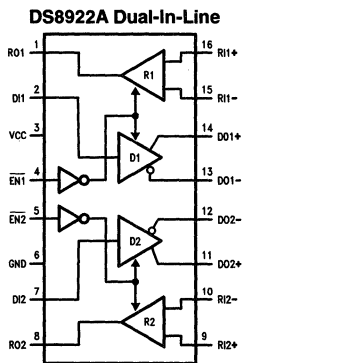
TL/F/8511–1

Order Number DS8922N, J, M,
DS8922AN, AJ, AM
See NS Package Number N16A, J16A or M16A

DS8923A Dual-In-Line
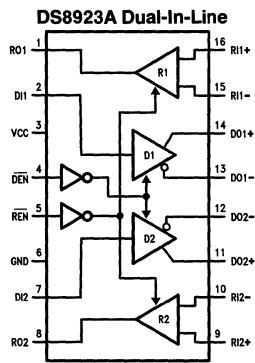
TL/F/8511–2

Order Number DS8923N, J, M
DS8923AN, AJ, AM
See NS Package Number N16A, J16A or M16A

Truth Tables

DS8922/22A

EN1	EN2	RO1	RO2	DO1	DO2
0	0	ACTIVE	ACTIVE	ACTIVE	ACTIVE
1	0	HI-Z	ACTIVE	HI-Z	ACTIVE
0	1	ACTIVE	HI-Z	ACTIVE	HI-Z
1	1	HI-Z	HI-Z	HI-Z	HI-Z

DS8923/23A

DEN	REN	RO1	RO2	DO1	DO2
0	0	ACTIVE	ACTIVE	ACTIVE	ACTIVE
1	0	ACTIVE	ACTIVE	HI-Z	HI-Z
0	1	HI-Z	HI-Z	ACTIVE	ACTIVE
1	1	HI-Z	HI-Z	HI-Z	HI-Z

For complete specifications see the Interface Databook.

6

National Semiconductor

DS8924 Quad TRI-STATE® Differential Line Driver

General Description

The DS8924 is a quad differential line driver designed for digital data transmission over balanced lines. The outputs are TRI-STATE® structures which are forced to a high impedance state when the appropriate output control pin reaches a logic zero condition. All input pins are PNP buffered to minimize input loading for either logic one or logic zero inputs. In addition, internal circuitry assures a high impedance output state during the transition between power up and power down.

The DS8924 is pin and functionally compatible with DS3487. It features improved performance over 3487-type circuit as outputs can source and sink 48 mA. In addition, outputs are not significantly affected by negative line reflections that can occur when the transmission line is unterminated at the receiver end.

Features

- Four independent driver chains
- TRI-STATE outputs
- PNP high impedance inputs
- Power up/down protection
- Fast propagation times (typ 12 ns)
- TTL compatible
- Single 5V supply voltage
- Output rise and fall times less than 20 ns (typ 10 ns)
- Pin compatible with DS3487 and MC3487
- Output skew—2 ns typ

Block and Connection Diagrams

TL/F/8507–1

Dual-In-Line Package
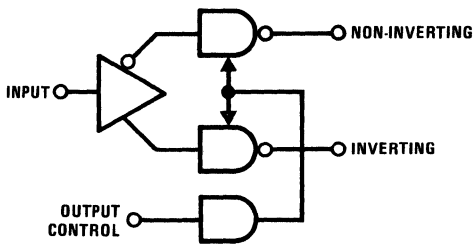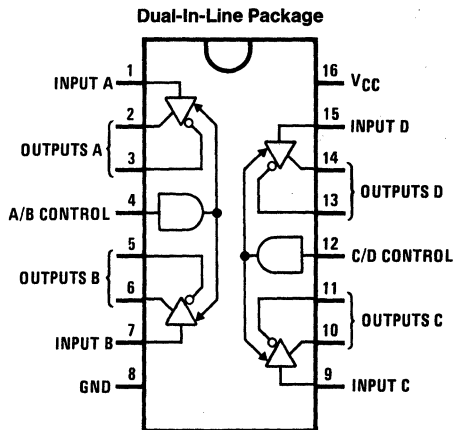
TL/F/8507–2

Top View

Order Number DS8924J or N
See NS Package J16A or N16A

Truth Table

Input	Control Input	Non-Inverter Output	Inverter Output
H	H	H	L
L	H	L	H
X	L	Z	Z

L = Low logic state
H = High logic state
X = Irrelevant
Z = TRI-STATE (high impedance)

![National Semiconductor logo] National Semiconductor

DS96172/µA96172/DS96174/µA96174
RS-485/RS-422 Quad Differential Line Drivers

General Description

The DS96172/µA96172 and DS96174/µA96174 are high speed quad differential line drivers designed to meet EIA Standard RS-485. The devices have TRI-STATE® outputs and are optimized for balanced multipoint data bus transmission at rates up to 10 Mbps. The drivers have wide positive and negative common mode range for multipoint applications in noisy environments. Positive and negative current-limiting is provided which protects the drivers from line fault conditions over a +12V to −7.0V common mode range. A thermal shutdown feature is also provided and occurs at junction temperature of approximately 160°C. The DS96172/µA96172 features an active high and active low Enable, common to all four drivers. The DS96174/µA96174 features separate active high Enables for each driver pair. Compatible RS-485 receivers, transceivers, and repeaters are also offered to provide optimum bus performance. The respective device types are DS96173/µA96173, DS96175/µA96175, DS96176/µA96176, DS96177/µA96177 and DS96178/µA96178.

Features

- Meets EIA Standard RS-485 and RS-422A
- Monotonic differential output switching
- Transmission rate to 10 Mbs
- TRI-STATE outputs
- Designed for multipoint bus transmission
- Common mode output voltage range: −7V to +12V
- Operates from single +5V supply
- Thermal shutdown protection
- DS96172/µA96172/DS96174/µA96174 are lead and function compatible with the SN75172/75174 or the AM26LS31/MC3487 respectively

Connection Diagrams

16-Lead DIP
DS96172/µA96172

TL/F/9626−1

Top View

16-Lead DIP
DS96174/µA96174
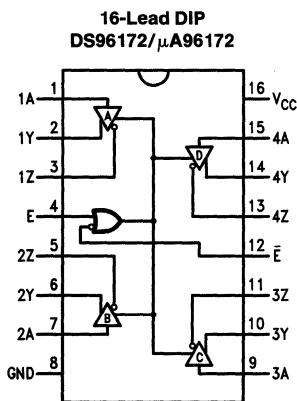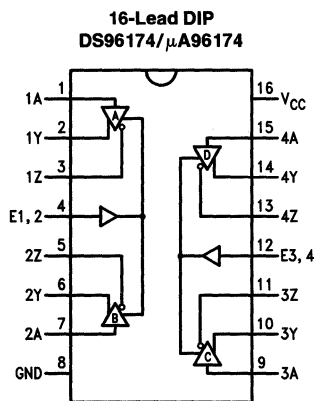
TL/F/9626−2

Top View

Order Number DS96172J, µA96172DC or DS96174J, µA96174DC
See NS Package Number J16A

Order Number DS96172N, µA96172PC or DS96174N, µA96174PC
See NS Package Number N16A

6

National Semiconductor

DS96173/µA96173/DS96175/µA96175
RS-485/RS-422 Quad Differential Line Receivers

General Description

The DS96173/µA96173 and DS96175/µA96175 are high speed quad differential line receivers designed to meet EIA Standard RS-485. The devices have TRI-STATE® outputs and are optimized for balanced multipoint data bus transmission at rates up to 10 Mbps. The receivers feature high input impedance, input hysteresis for increased noise immunity, and input sensitivity of 200 mV over a common mode input voltage range of −12V to +12V. The receivers are therefore suitable for multipoint applications in noisy environments. The DS96173/µA96173 features an active high and active low Enable, common to all four receivers. The DS96175/µA96175 features separate active high Enables for each receiver pair. Compatible RS-485 drivers, transceivers, and repeaters are also offered to provide optimum bus performance. The respective device types are DS96172/µA96172, DS96174/µA96174, DS96176/µA96176, DS96177/µA96177 and DS96178/µA96178.

Features

■ Meets EIA Standard RS-485, RS-422A, RS-423A
■ Designed for multipoint bus applications
■ TRI-STATE Outputs
■ Common mode input voltage range: −7V to +12V
■ Operates from single +5V supply
■ Input sensitivity of ±200 mV over common mode range
■ Input hysteresis of 50 mV typical
■ High input impedance
■ Fail-safe input/output features drive output HIGH when input is open
■ DS96173/µA96173/DS96175/µA96175 are lead and function compatible with SN75173/75175 or the AM26LS32/MC3486 respectively.

Connection Diagrams

16-Lead DIP
DS96173/µA96173

TL/F/9628–1

16-Lead DIP
DS96175/µA96175
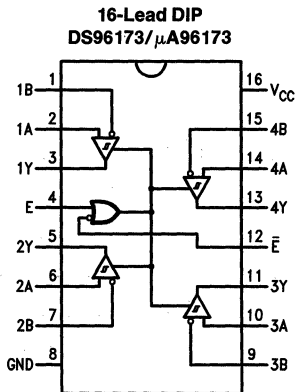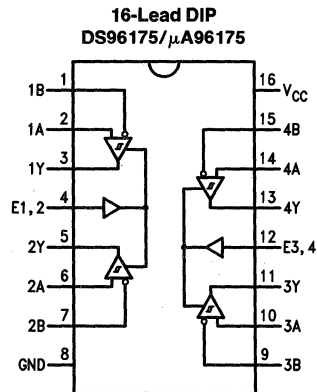
TL/F/9628–2

Order Number DS96173J, µA96173DC, DS96175J, µA96175DC
See NS Package Number J16A*

Order Number DS96173N, µA96173PC, DS96175N, µA96175PC
See NS Package Number N16A

*For most current package information, contact product marketing.

![National Semiconductor logo]

DS9636A/μA9636A
RS-423 Dual Programmable Slew Rate Line Driver

General Description

The DS9636A/μA9636A is a TTL/CMOS compatible, dual, single ended line driver which has been specifically designed to satisfy the requirements of EIA Standard RS-423.

The DS9636A/μA9636A is suitable for use in digital data transmission systems where signal wave shaping is desired. The output slew rates are jointly controlled by a single external resistor connected between the wave shaping control lead (WS) and ground. This eliminates any need for external filtering of the output signals. Output voltage levels and slew rates are independent of power supply variations. Current-limiting is provided in both output states. The DS9636A/μA9636A is designed for nominal power supplies of ±12V.

Inputs are TTL compatible with input current loading low enough (1/10 UL) to be also compatible with CMOS logic. Clamp diodes are provided on the inputs to limit transients below ground.

Features

- Programmable slew rate limiting
- Meets EIA Standard RS-423
- Commercial or extended temperature range
- Output short circuit protection
- TTL and CMOS compatible inputs

Connection Diagram

8-Lead DIP

WAVESHAPE CONTROL — 1, 2
IN A — 3
IN B — 4
GND — (4)

8 — V+
7 — OUT A
6 — OUT B
5 — V−

A, B
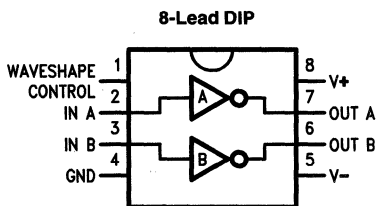
TL/F/9620–1

Top View

Order Number DS9636ACJ, μA9636ARC,
DS9636AMJ, μA9636ARM or DS9636ACN, μA9636ATC
See NS Package Number J08A or N08E

*For most current package information, contact product marketing.

National Semiconductor

DS9637A/μA9637A
Dual Differential Line Receiver

General Description

The DS9637A/μA9637A is a Schottky dual differential line receiver which has been specifically designed to satisfy the requirements of EIA Standards RS-422 and RS-423. In addition, the DS9637A/μA9637A satisfies the requirements of MIL-STD 188-114 and is compatible with the International Standard CCITT recommendations. The DS9637A/μA9637A is suitable for use as a line receiver in digital data systems, using either single ended or differential, unipolar or bipolar transmission. It requires a single 5V power supply and has Schottky TTL compatible outputs. The DS9637A/μA9637A has an operational input common mode range of ±7V either differentially or to ground.

Features

■ Dual channels
■ Single 5V supply
■ Satisfies EIA standards RS-422 and RS423
■ Built-in ±35 mV hysteresis
■ High common mode range
■ High input impedance
■ TTL compatible output
■ Schottky technology
■ Extended temperature range

Connection Diagram

8-Lead DIP and SO-8 Package

TL/F/9621–1

Top View
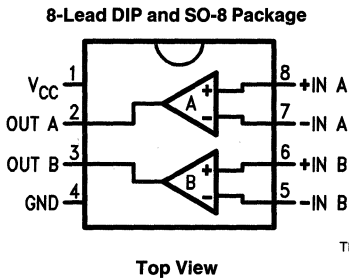
Order Number DS9637ACJ, μA9637ARC,
DS9637AMJ, μA9637ARM
See NS Package Number J08A*

Order Number DS9637ACM, μA9637ASC
See NS Package Number M08A

Order Number DS9637ACN, μA9637ATC
See NS Package Number N08E

*For most current package information, contact product marketing.

National Semiconductor

DS9638/µA9638
RS-422 Dual High Speed Differential Line Driver

General Description

The DS9638/µA9638 is a Schottky, TTL compatible, dual differential line driver designed specifically to meet the EIA Standard RS-422 specifications. It is designed to provide unipolar differential drive to twisted pair or parallel wire transmission lines. The inputs are TTL compatible. The outputs are similar to totem pole TTL outputs, with active pull-up and pull-down. The device features a short circuit protected active pull-up with low output impedance and is specified to drive 50Ω transmission lines at high speed. The mini-DIP provides high package density.

Features

- Single 5V supply
- Schottky technology
- TTL and CMOS compatible inputs
- Output short circuit protection
- Input clamp diodes
- Complementary outputs
- Minimum outupt skew (<1.0 ns typical)
- 50 mA output drive capability for 50Ω transmission lines
- Meets EIA RS-422 specifications
- Propagation delay of less than 10 ns
- "Glitchless" differential output
- Delay time stable with V_{CC} and temperature variations (<2.0 ns typical) *(Figure 3)*
- Extended temperature range

Connection Diagram

8-Lead DIP and SO-8 Package

```
        1            8
Vcc ---|      ___    |--- OUT A
        2    | A >   |    7
IN A ---|    |__>o   |--- OUT Ā
        3            6
IN B ---|    | B >   |--- OUT B
        4    |__>o   |    5
GND ---|            |--- OUT B̄
```
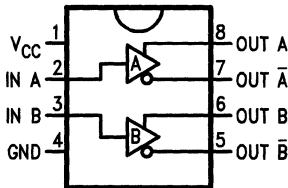
TL/F/9622-1

Top View

Order Number DS9638MJ, µA9638RM,
DS9638CJ or µA9638RC
See NS Package Number J08A*

Order Number DS9638CM or µA9638SC
See NS Package Number M08A

Order Number DS9638CN or µA9638TC
See NS Package Number N08E

*For most current package information: contact product marketing.

6

National Semiconductor

DS9639A/μA9639A
Dual Differential Line Receiver

General Description

The DS9639A/μA9639A is a Schottky dual differential line receiver which has been specifically designed to satisfy the requirements of EIA Standards RS-422, RS-423 and RS-232C. In addition, the DS9639A/μA9639A satisfies the requirements of MIL-STD 188-114 and is compatible with the International Standard CCITT recommendations. The DS9639A/μA9639A is suitable for use as a line receiver in digital data systems, using either single ended or differential, unipolar or bipolar transmission. It requires a single 5.0V power supply and has Schottky TTL compatible outputs. The DS9639A/μA9639A has an operational input common mode range of ±7.0V either differentially or to ground.

Features

- Dual channels
- Single 5.0V supply
- Satisfies EIA Standards RS-422, RS-423 and RS-232C
- Built-in ±35 mV hysteresis
- High common mode range
- High input impedance
- TTL compatible output
- Schottky technology

Connection Diagram

8-Lead DIP

TL/F/9623–1

Top View
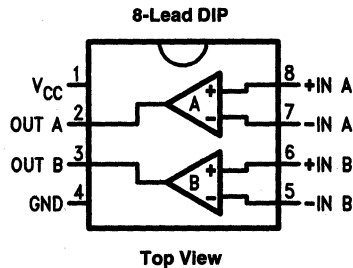
Order Number DS9639ACN/μA9639ATC
See NS Package Number N08E

National Semiconductor

DS26F31C/DS26F31M
Quad High Speed Differential Line Driver

General Description

The DS26F31 is a quad differential line driver designed for digital data transmission over balanced lines. The DS26F31 meets all the requirements of EIA Standard RS-422 and Federal Standard 1020. It is designed to provide unipolar differential drive to twisted-pair or parallel-wire transmission lines.

The DS26F31 offers improved performance due to the use of state-of-the-art L-FAST bipolar technology. The L-FAST technology allows for higher speeds and lower currents by utilizing extremely short gate delay times. Thus, the DS26F31 features lower power, extended temperature range, and improved specifications.

The circuit provides an enable and disable function common to all four drivers. The DS26F31C/DS26F31M features TRI-STATE® outputs and logical OR-ed complementary enable inputs. The inputs are all LS compatible and are all one unit load.

The DS26F31C/DS26F31M offers optimum performance when used with the DS26F32 Quad Differential Line Receiver.

Features

■ Military temperature range
■ Low power version
■ Output skew—2.0 ns typical
■ Input to output delay—12 ns
■ Operation from single +5.0V supply
■ 16-lead ceramic DIP Package
■ Outputs won't load line when $V_{CC} = 0V$
■ Output short circuit protection
■ Complementary outputs
■ Meets the requirements of EIA standard RS-422
■ High output drive capability for 100Ω terminated transmission lines

Connection and Logic Diagrams

16-Lead Dual-In-Line Package

Top View

TL/F/9614–1

Order Number DS26F31CJ or DS26F31MJ
See NS Package Number J16A

TL/F/9614–2
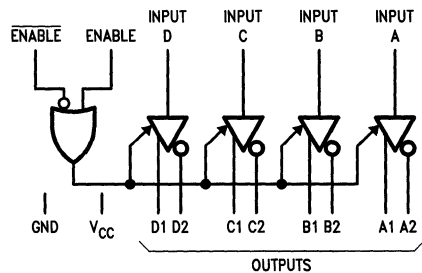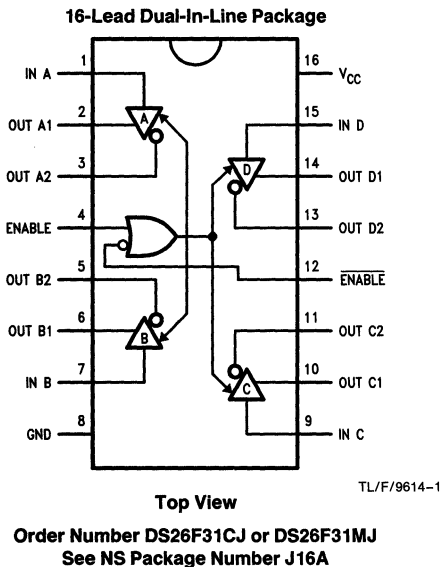
FIGURE 1. Logic Symbol

Function Table (Each Driver)

Input	Enable	Outputs	
		Y	Z
H	H	H	L
L	H	L	H
X	L	Z	Z

H = High Level   X = Immaterial
L = Low Level   Z = High Impedance (Off)

6

National Semiconductor

DS26F32C/DS26F32M
Quad Differential Line Receiver

General Description

The DS26F32 is a quad differential line receiver designed to meet the requirements of EIA Standards RS-422 and RS-423, and Federal Standards 1020 and 1030 for balanced and unbalanced digital data transmission.

The DS26F32 offers improved performance due to the use of state-of-the-art L-FAST bipolar technology. The L-FAST technology allows for higher speeds and lower currents by utilizing extremely short gate delay times. Thus, the DS26F32 features lower power, extended temperature range, and improved specifications.

The device features an input sensitivity of 200 mV over the input range of ±7.0V. The DS26F32 provides an enable function common to all four receivers and TRI-STATE® outputs with 8.0 mA sink capability. Also, a fail-safe input/output relationship keeps the outputs high when the inputs are open.

The DS26F32 offers optimum performance when used with the DS26F31 Quad Differential Line Driver.

Features

■ Military temperature range
■ Low power version
■ Input voltage range of ±7.0V (differential or common mode) ±0.2V sensitivity over the input voltage range
■ Meets all the requirements of EIA standards RS-422 and RS-423
■ Input impedance (18k typical)
■ 30 mV input hysteresis
■ Operation from single +5.0V supply
■ Fail-safe input/output relationship. Output always high when inputs are open
■ TRI-STATE drive, with choice of complementary output enables, for receiving directly onto a data bus
■ Propagation delay 15 ns typical
■ Advanced low power Schottky processing

Connection Diagram

16-Lead DIP
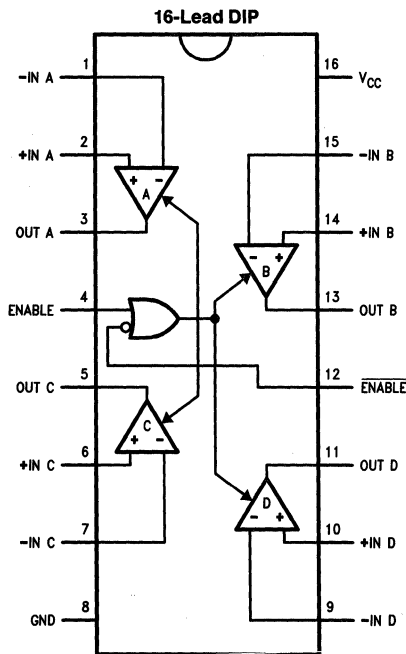
TL/F/9615–1

Top View

Function Table (Each Receiver)

Differential Inputs	Enables		Outputs
A−B	E	Ē	V
$V_{ID} \geq 0.2V$	H	X	H
	X	L	H
$V_{ID} \leq -0.2V$	H	X	L
	X	L	L
X	L	H	Z

H = High Level
L = Low Level
X = Immaterial

Order Number DS26F32CJ or DS26F32MJ
See NS Package Number J16A

![National Semiconductor logo]

DS35F86/DS34F86
RS-422/RS-423 Quad Line Receiver
with TRI-STATE® Outputs

General Description

The DS34F86/DS35F86 RS-422/3 Quad Receiver features four independent receivers, which comply with EIA Standards for the electrical characteristics of balanced/unbalanced voltage digital interface circuits. Receiver outputs are 74LS compatible TRI-STATE structures which are forced to a high impedance state when the appropriate output control lead reaches a logic zero condition. A PNP device buffers each output control lead to assure minimum loading for either logic one or logic zero inputs. In addition each receiver chain has internal hysteresis circuitry to improve noise margin and discourage output instability for slowly changing input waveforms.

The DS34F86/DS35F86 offers improved performance due to the use of state-of-the-art L-FAST bipolar technology. The L-FAST technology allows for higher speeds and lower currents by utilizing extremely short gate delay times. Thus, the DS34F86/DS35F86 features lower power, extended temperature range, and improved specifications.

The DS34F86/DS35F86 offers optimum performance when used with the DS34F87/DS35F87 Quad Line Driver.

Features

- Military temperature range
- Low power version
- Four independent receiver chains
- TRI-STATE outputs
- High impedance output control inputs
- Fast propagation times 15 ns typical
- TTL compatible
- Single 5.0V supply voltage
- Output rise and fall times less than 20 ns
- Lead compatible and interchangeable with MC3486 and DS3486

Connection Diagram

16-Lead DIP
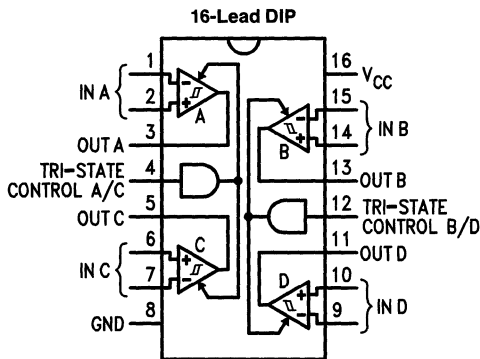
TL/F/9616–1

Top View

Order Number DS34F86J or DS35F86J
See NS Package Number J16A
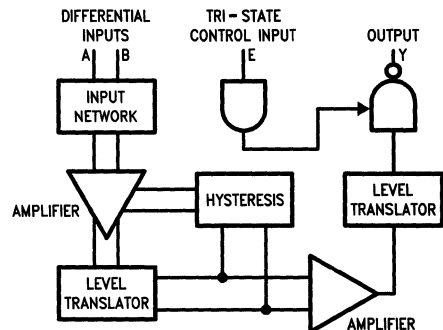
TL/F/9616–2

FIGURE 1. Block Diagram

Function Table (Each Receiver)

Differential Inputs AB	Enable E	Output Y
$V_{ID} \leq 0.2V$	H	H
$V_{ID} \leq -0.2V$	H	L
X	L	Z

H = High Level
L = Low Level
Z = High Impedance (off)

6

National Semiconductor

DS35F87/DS34F87
RS-422 Quad Line Driver with TRI-STATE® Outputs

General Description

The DS34F87/DS35F87 RS-422 Quad Line Driver features four independent driver chains which comply with EIA Standards for the electrical characteristics of balanced voltages digital interface circuits. The outputs are TRI-STATE structures which are forced to a high impedance state when the appropriate output control lead reaches a logic zero condition. All input leads are PNP buffered to minimize input loading for either logic one or logic zero inputs. In addition, internal circuitry assures a high impedance output state during the transition between power-up and power-down.

The DS34F87/DS35F87 offers improved performance due to the use of state-of-the-art L-FAST bipolar technology. The L-FAST technology allows for higher speeds and lower currents by utilizing extremely short gate delay times. Thus, the DS34F87/DS35F87 features lower power, extended temperature range, and improved specifications.

The DS34F87/DS35F87 offers optimum performance when used with the DS34F86/DS35F86 Quad Line Receiver.

Features

- Military temperature range
- Four independent driver chains
- TRI-STATE outputs
- PNP high impedance inputs
- Fast propagation time
- TTL compatible
- Single 5.0V supply voltage
- Output rise and falls times less than 20 ns
- Lead compatible and interchangeable with MC3487 and DS3487

Block and Connection Diagrams

FIGURE 1

TL/F/9618-2

16-Lead DIP

Top View

TL/F/9618-1

Order Number DS34F87J or DS35F87J
See NS Package Number J16A
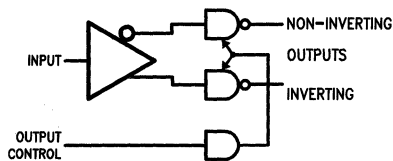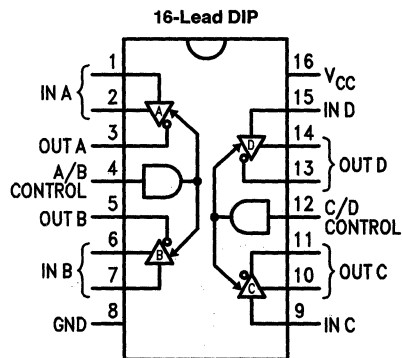
Function Table (Each Driver)

Input	Enable	Output	
		Y	Z
H	H	H	L
L	H	L	H
X	L	Z	Z

H = High Level
L = Low Level
X = Immaterial
Z = High Impedance (off)

National Semiconductor

DS1691A/DS3691 (RS-422/RS-423) Line Drivers with TRI-STATE® Outputs

General Description

The DS1691A/DS3691 are low power Schottky TTL line drivers designed to meet the requirements of EIA standards RS-422 and RS-423. They feature 4 buffered outputs with high source and sink current capability with internal short circuit protection. A mode control input provides a choice of operation either as 4 independent line drivers or 2 differential line drivers. A rise time control pin allows the use of an external capacitor to reduce rise time for suppression of near end crosstalk to other receivers in the cable.

With the mode select pin low, the DS1691A/DS3691 are dual-differential line drivers with TRI-STATE outputs. They feature ±10V output common-mode range in TRI-STATE mode and 0V output unbalance when operated with ±5V supply.

Features

■ Dual RS-422 line driver with mode pin low, or quad RS-423 line driver with mode pin high
■ TRI-STATE control for individual outputs
■ Short circuit protection for both source and sink outputs
■ Outputs will not clamp line with power off or in TRI-STATE
■ Individual rise mode time control for each output
■ 100Ω transmission line drive capability
■ Low I_{CC} and I_{EE} power consumption
  RS-422   35 mW/driver typ
  RS-423   26 mW/driver typ
■ Low current PNP inputs compatible with TTL, MOS and CMOS
■ Pin compatible with AM26LS30

Connection Diagram

With Mode Select LOW
(RS-422 Connection)

TL/F/5783–1

Top View

With Mode Select HIGH
(RS-423 Connection)

TL/F/5783–2

Top View

Truth Table

Operation	Inputs			Outputs	
	Mode	A (D)	B (C)	A (D)	B (C)
RS-422	0	0	0	0	1
	0	0	1	TRI-STATE	TRI-STATE
	0	1	0	1	0
	0	1	1	TRI-STATE	TRI-STATE
RS-423	1	0	0	0	0
	1	0	1	0	1
	1	1	0	1	0
	1	1	1	1	1

Order Number DS1691AJ, DS3691J, DS3691M or DS3691N
See NS Package Number J16A, M16A or N16A

6

National Semiconductor

DS16F95/DS36F95
RS-485/RS-422 Differential Bus Transceiver

General Description

The DS16F95/DS36F95 Differential Bus Transceiver is a monolithic integrated circuit designed for bidirectional data communication on balanced multipoint bus transmission lines. The transceiver meets EIA Standard RS-485 as well as RS-422A.

The DS16F95/DS36F95 offers improved performance due to the use of state-of-the art L-FAST bipolar technology. The L-FAST technology allows for higher speeds and lower currents by utilizing extremely short gate delay times. Thus, the DS16F95/DS36F95 features lower power, extended temperature range, and improved specifications.

The DS16F95/DS36F95 combines a TRI-STATE® differential line driver and a differential input line receiver, both of which operate from a single 5.0V power supply. The driver and receiver have an active Enable that can be externally connected to function as a direction control. The driver differential outputs and the receiver differential inputs are internally connected to form differential input/output (I/O) ports that are designed to offer minimum loading to the bus whenever the driver is disabled or when $V_{CC} = 0V$. These ports feature wide positive and negative common mode voltage ranges, making the device suitable for multipoint applications in noisy environments.

The driver is designed to handle loads up to 60 mA of sink or source current. The driver features positive and negative current-limiting and thermal shutdown for protection from line fault conditions.

The DS16F95/DS36F95 can be used in transmission line applications employing the DS96F172 and the DS96F174 quad differential line drivers and the DS96F173 and DS96F175 quad differential line receivers.

Features

- Military temperature range
- Bidirectional transceiver
- Meets EIA Standard RS-422A and RS-485
- Meets SCSI specifications
- Designed for multipoint transmission
- TRI-STATE driver and receiver enables
- Individual driver and receiver enables
- Wide positive and negative input/output bus voltage ranges
- Driver output capability +60 mA maximum
- Thermal shutdown protection
- Driver positive and negative current-limiting
- High impedance receiver input
- Receiver input sensitivity of ±200 mV
- Receiver input hysteresis of 50 mV typical
- Operates from single 5.0V supply
- Low power version
- Pin compatible with DS3695 and SN75176A

Connection Diagram

8-Lead Dual-In-Line Package
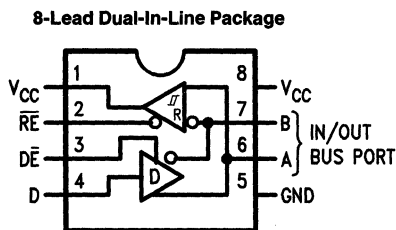
TL/F/9629–1

Top View

Order Number DS16F95J, DS36F95J
See NS Package Number J08A

Function Tables

Driver

Differential Inputs	Enable	Outputs	
D	DE	A	B
H	H	H	L
L	H	L	H
X	L	Z	Z

Receiver

Differential Inputs	Enable	Output
A–B	$\overline{RE}$	R
$V_{ID} \geq 0.2V$	L	H
$V_{ID} \leq -0.2V$	L	L
X	H	Z

H = High Level
L = Low Level
X = Immaterial
Z = High impedance (Off)

National Semiconductor

DS96F172/DS96F174
RS-485/RS-422 Quad Differential Drivers

General Description

The DS96F172 and the DS96F174 are high speed quad differential line drivers designed to meet EIA Standard RS-485. The DS96F172 and the DS96F174 offer improved performance due to the use of new, state-of-the-art L-FAST bipolar technology. The L-FAST technology allows for higher speeds and lower currents by utilizing extremely short gate delay times. Thus, the DS96F172 and the DS96F174 feature lower power, extended temperature range, improved RS-485 specifications, and meet SCSI specifications.

The DS96F172 and the DS96F174 have TRI-STATE® outputs and are optimized for balanced multipoint data bus transmission at rates up to 15 Mbps. The drivers have wide positive and negative common mode range for multipoint applications in noisy environments. Positive and negative current-limiting is provided which protects the drivers from line fault conditions over a +12V to −7.0V common mode range. A thermal shutdown feature is also provided. The DS96F172 features an active high and active low Enable, common to all four drivers. The DS96F174 features separate active high Enables for each driver pair.

Compatible RS-485 receivers, transceivers, and repeaters are also offered to provide optimum bus performance. The respective device types are DS96F173, DS96F175 and DS36F95.

Features

- Military temperature range available
- Meets EIA Standard RS-485 and RS-422A
- Meets SCSI specifications
- Monotonic differential output switching
- Transmission rate to 10 Mbps
- TRI-STATE outputs
- Designed for multipoint bus transmission
- Common mode output voltage range: −7.0V to +12V
- Operates from single +5.0V supply
- Lower power version
- Thermal shutdown protection
- DS96F172 and DS96F174 are lead and function compatible with the SN75172/174 or the AM26LS31/MC3487

Connection Diagrams

Dual-In-Line Package

DS96F172

Top View

TL/F/9625–1

DS96F174

Top View
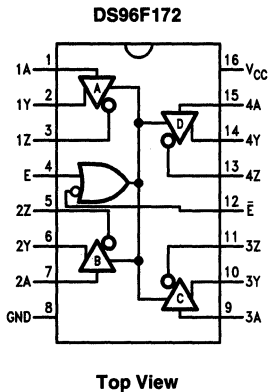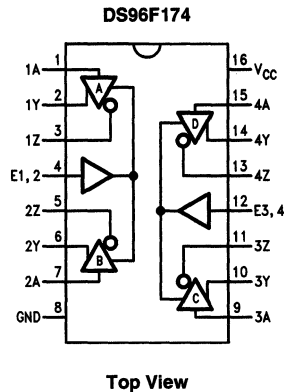
TL/F/9625–2

Order Number DS96F172CJ, DS96F172MJ, DS96F174CJ or DS96F174MJ
See NS Package Number J16A

6

DS96F173/DS96F175
RS-485/RS-422 Quad Differential Receivers

General Description

The DS96F173 and the DS96F175 are high speed quad differential line receivers designed to meet EIA Standard RS-485. The DS96F173 and the DS96F175 offer improved performance due to the use of state-of-the-art L-FAST bipolar technology. The L-FAST technology allows for higher speeds and lower currents by utilizing extremely short gate delay times. Thus, the DS96F173 and the DS96F175 feature lower power, extended temperature range, improved RS-485 specifications, and meet SCSI specifications.

The DS96F173 and the DS96F175 have TRI-STATE® outputs and are optimized for balanced multipoint data bus transmission at rates up to 15 Mbps. The receivers feature high input impedance, input hysteresis for increased noise immunity, and input sensitivity of 200 mV over a common mode input volage range of −12V to +12V. The receivers are therefore suitable for multipoint applications in noisy environments. The DS96F173 features an active high and active low Enable, common to all four receivers. The DS96F175 features separate active high Enables for each receiver pair.

Compatible RS-485 drivers, transceivers, and repeaters are also offered to provide optimum bus performance. The re-spective device types are DS96F172, DS96F174 and DS36F95.

Features

■ Military temperature range available
■ Meets EIA Standard RS-485, RS-422A, RS-423A
■ Meets SCSI specifications
■ Designed for multipoint bus applications
■ TRI-STATE outputs
■ Common mode input voltage range: −12V to +12V
■ Operates from single +5.0V supply
■ Lower power version
■ Input sensitivity of ±200 mV over common mode range
■ Input hysteresis of 50 mV typical
■ High input impedance
■ Fail-safe input/output features drive output HIGH when input is open
■ DS96F173 and DS96F175 are lead and function compatible with SN75173/175 or the AM26LS32/MC3486

Connection Diagrams

16-Lead Ceramic Dual-In-Line Package

DS96F173

TL/F/9627–1

Top View

DS96F175

TL/F/9627–2

Top View
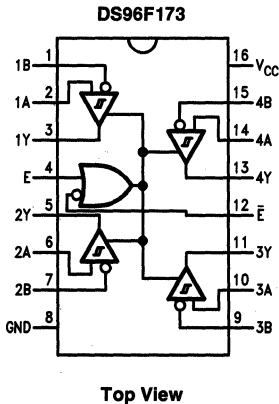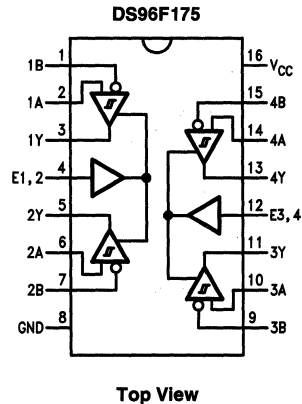
Order Number DS96F173CJ, DS96F173MJ,
DS96F175CJ or DS96F175MJ
See NS Package Number J16A

![National Semiconductor logo]

DS96176/μA96176
RS-485/RS-422 Differential Bus Transceiver

General Description

The DS96176/μA96176 Differential Bus Transceiver is a monolithic integrated circuit designed for bidirectional data communication on balanced multipoint bus transmission lines. The transceiver meets EIA Standard RS-485 as well as RS-422A.

The DS96176/μA96176 combines a TRI-STATE® differential line driver and a differential input line receiver, both of which operate from a single 5.0V power supply. The driver and receiver have an active Enable that can be externally connected to function as a direction control. The driver differential outputs and the receiver differential inputs are internally connected to form differential input/output (I/O) bus ports that are designed to offer minimum loading to the bus whenever the driver is disabled or when $V_{CC} = 0V$. These ports feature wide positive and negative common mode voltage ranges, making the device suitable for multipoint applications in noisy environments.

The driver is designed to handle loads up to 60 mA of sink or source current. The driver features positive and negative current-limiting and thermal shutdown for protection from line fault conditions. Thermal shutdown is designed to occur at junction temperature of approximately 160°C. The receiver features a typical input impedance of 15 kΩ, an input sensitivity of ±200 mV, and a typical input hysteresis of 50 mV.

The DS96176/μA96176 can be used in transmission line applications employing the DS96172/μA96172 and the DS96174/μA96174 quad differential line drivers and the DS96173/μA96173 and DS96175/μA96175 quad differential line receivers.

Features

- Bidirectional transceiver
- Meets EIA Standard RS-422A and RS-485
- Designed for multipoint transmission
- TRI-STATE driver and receiver enables
- Individual driver and receiver enables
- Wide positive and negative input/output bus voltage ranges
- Driver output capability ±60 mA Maximum
- Thermal shutdown protection
- Driver positive and Negative current-limiting
- High impedance receiver input
- Receiver input sensitivity of ±200 mV
- Receiver input hysteresis of 50 mV typical
- Operates from single 5.0V supply
- Low power requirements

Connection Diagram

8-Lead DIP
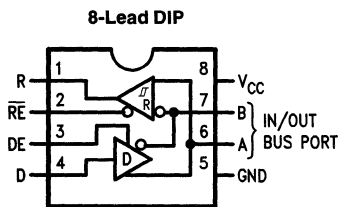
TL/F/9630-1

Top View

Order Number DS96176J, μA96176RC
See NS Package Number J08A*

Order Number DS96176N, μA96176TC
See NS Package Number N08E

Function Table

Driver

Differential Inputs	Enable	Outputs	
D	DE	A	B
H	H	H	L
L	H	L	H
X	L	Z	Z

Receiver

Differential Inputs	Enable	Output
A-B	$\overline{RE}$	R
$V_{ID} \geq 0.2V$	L	H
$V_{ID} \leq -0.2V$	L	L
X	H	Z

H = High Level
L = Low Level
X = Immaterial
Z = High Impedance (off)

6

*For most current package information, contact product marketing.

Section 7
Physical Dimensions

Section 7 Contents

![National Semiconductor logo]

20 Lead Ceramic Dual-In-Line Package (J)
NS Package Number J20A

0.985
(25.019)
MAX

0.025
(0.635)
RAD

0.220 − 0.310
(5.588 − 7.874)

0.005 − 0.020
(0.127 − 0.508)
RAD TYP

0.037 ± 0.005
(0.940 ± 0.127)

0.180
(4.572)
MAX

0.290 − 0.320
(7.366 − 8.128)

GLASS SEALANT

0.005
(0.127)
MIN

0.055 ± 0.005
(1.397 ± 0.127)

0.020 − 0.060
(0.508 − 1.524)

0.200
(5.080)
MAX

0.150
(3.810)
MIN

0.125 − 0.200
(3.175 − 5.080)

95° ± 5°

0.008 − 0.012
(0.203 − 0.305)

86° 94°

0.310 − 0.410
(7.874 − 10.41)

0.060
(1.524)
MAX
BOTH ENDS

0.018 ± 0.003
(0.457 ± 0.076)

0.100 ± 0.010
(2.540 ± 0.254)

J20A (REV M)

24 Lead Ceramic Dual-In-Line Package (J)
NS Package Number J24A

1.290
(32.766)
MAX

0.600
(15.240)
MAX
GLASS

0.025
(0.635)
RAD

0.514 − 0.526
(13.06 − 13.36)

0.030 − 0.055
(0.762 − 1.397)
RAD TYP

0.180
(4.572)
MAX

0.590 − 0.620
(14.986 − 15.748)

0.005
(0.127)
MIN

GLASS
SEALANT

0.055 ± 0.005
(1.397 ± 0.127)

0.020 − 0.070
(0.508 − 1.778)

95° ± 5°

0.008 − 0.012
(0.203 − 0.305)

0.685 +0.025
      −0.060

(17.40 +0.635
       −1.524)

0.098
(2.489)
MAX

0.100 ± 0.010
(2.540 ± 0.254)

0.018 ± 0.003
(0.457 ± 0.076)

86°
94°
TYP

0.125 − 0.200
(3.175 − 5.080)
MIN

0.150
(3.810)
MIN

J24A (REV H)

7

16 Lead Molded Dual-In-Line Package (N)
NS Package Number N16A

N16A (REV E)

20 Lead Molded Dual-In-Line Package (N)
NS Package Number N20A
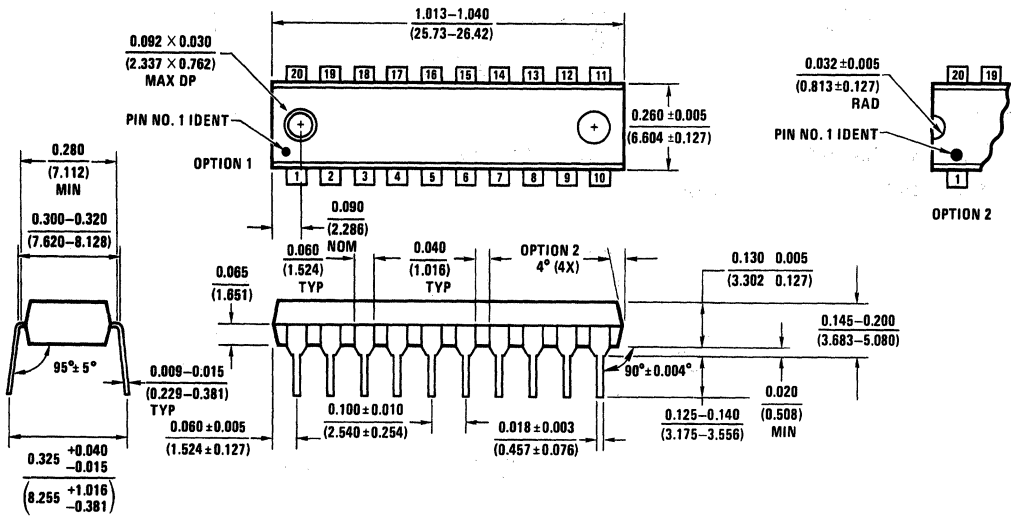
N20A (REV G)

24 Lead Molded Dual-In-Line Package (N)
NS Package Number N24A

1.243−1.270
(31.57−32.26)

0.062
(1.575)
RAD

PIN NO. 1 IDENT

0.540 ±0.005
(13.716 ±0.127)

DOTTED OUTLINES
REFLECT ALTERNATE
MOLDED BODY CONFIGURATION

0.580
(14.73)
MIN

0.030
(0.762)
MAX

0.075
(1.905)

0.600−0.620
(15.24−15.748)

0.060
(1.524)

0.040
(1.016)
TYP

0.160 ±0.005
(4.064 ±0.127)

0.170−0.210
(4.318−5.334)

95°±5°

0.625 +0.025
−0.015

0.009−0.015
(0.229−0.381)

0.075 ±0.015
(1.905 ±0.381)

86°94°
TYP

0.018 ±0.003
(0.457 ±0.076)

0.015
(0.381)

0.125−0.140
(3.175−3.556)
MIN

$\left(15.875\substack{+0.635 \\ -0.381}\right)$

0.100 ±0.010
(2.540 ±0.254)

N24A (REV E)

24 Lead Skinny Dual-In-Line Package (0.300″ Centers Molded) (N)
NS Package Number N24C

1.243−1.270
(31.57−32.26)
MAX

0.092
(2.337)
(2 PLS)

PIN NO. 1
IDENT

0.032
(0.813)
RAD

OPTION 2

0.260 ±0.005
(6.604 ±0.127)

OPTION 2

0.062
(1.575)
RAD

EJECTOR PINS
OPTIONAL

0.300−0.320
(7.62−8.128)

0.040
(1.016)
TYP

0.130 ±0.005
(3.302 ±0.127)

0.009−0.015
(0.229−0.381)

95° ±5°

0.020
(0.508)
MIN

0.145−0.200
(3.683−5.080)

0.280
(7.112)
MIN

0.065
(1.651)

0.325 +0.040
−0.015

0.075 ±0.015
(1.905 ±0.381)

0.018 ±0.003
(0.457 ±0.076)
TYP

0.125−0.145
(3.175−3.556)
MIN

$\left(8.255\substack{+1.016 \\ -0.381}\right)$

0.100 ±0.010
(2.54 ±0.254)
TYP

90° ±4° TYP

N24C (REV F)

7

40 Lead Molded Dual-In-Line Package (N)
NS Package Number N40A

2.043–2.070
(51.89–52.58)

0.062
(1.575)
RAD

PIN NO. 1 IDENT

0.550 ±0.005
(13.970 ±0.127)

0.580
(14.73)
MIN

0.600–0.620
(15.240–15.748)

0.030
(0.762)
MAX

0.050
(1.270)
TYP

0.125–0.165
(3.175–4.191)

0.145–0.210
(3.683–5.334)

95°±5°

0.625 +0.025 −0.015
(15.875 +0.635 −0.381)

0.009–0.015
(0.229–0.381)

0.075 ±0.015
(1.905 ±0.381)

0.100 ±0.010
(2.540 ±0.254)

0.018 ±0.003
(0.457 ±0.076)

86°–94°
TYP

0.020
(0.508)
MIN

0.125–0.140
(3.175–3.556)

N40A (REV E)

48 Lead Molded Dual-In-Line Package (N)
NS Package Number N48A

2.393–2.440
(60.78–61.98)

0.550 ±0.005
(13.97 ±0.127)

0.062
(1.574)
RAD

PIN NO. 1
IDENT

0.030
(0.762)
MAX

0.600–0.620
(15.24–15.74)

0.125–0.165
(3.175–4.191)

0.145–0.200
(3.683–5.080)

95° ±5°

0.580
(14.73)
MIN

0.020
(0.508)
MIN

0.050 ±0.015
(1.27 ±0.381)

0.100 ±0.010
(2.54 ±0.254)

0.018 ±0.003
(0.457 ±0.076)

0.050
(1.27)
TYP

86°–94°
TYP

0.009–0.015
(0.229–0.381)

0.625 +0.025 −0.015
(15.88 +0.635 −0.381)

0.125–0.145
(3.175–3.683)
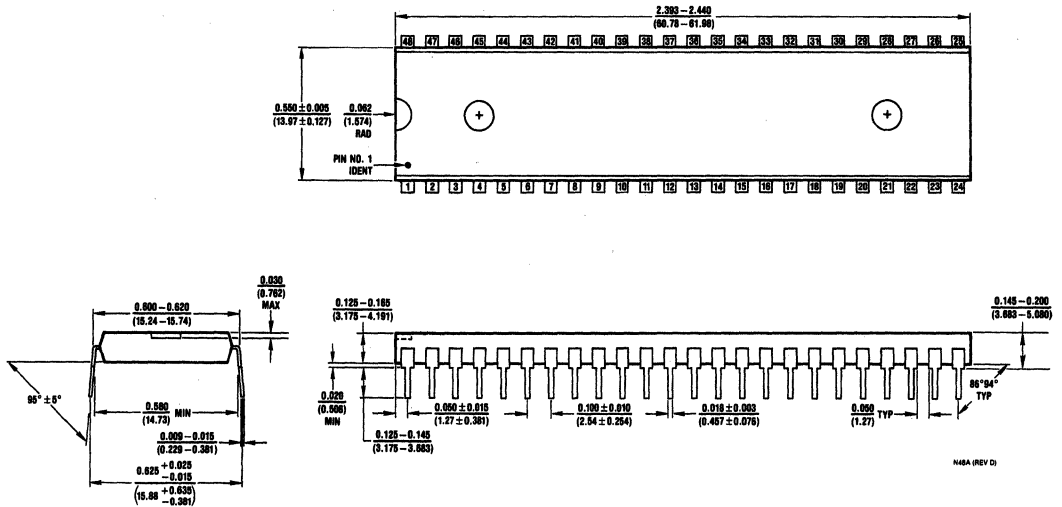
N48A (REV D)

28 Lead Plastic Chip Carrier (V)
NS Package Number V28A
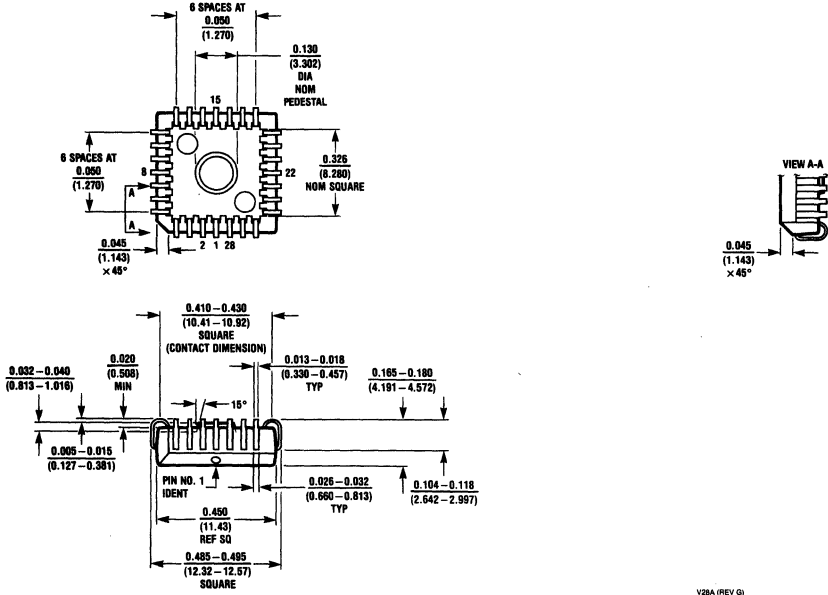
V28A (REV G)

44 Lead Plastic Chip Carrier (V)
NS Package Number V44A

V44A (REV H)

7

68 Lead Plastic Chip Carrier (V)
NS Package Number V68A

0.050 = 0.800
(1.270 = 20.32)
16 SPACES AT

0.050
(1.270)
REF

61   68 1   9

60°   10

0.826
(20.98)
NOM

0.050 = 0.800
(1.270 = 20.32)
16 SPACES AT

0.050
(1.270)
REF

0.330
(8.382)
DIA NOM
PEDESTAL

44   26

43   27

0.826
(20.98)
NOM

0.020
(0.508)
MIN

0.045
(1.143)

0.104 – 0.118
(2.642 – 2.997)

0.045
(1.143)

0.013 – 0.018
(0.330 – 0.457)
TYP

0.910 – 0.930
(23.11 – 23.62)
SQUARE
CONTACT
DIMENSION

0.950
(24.13)
REF SQ

0.985 – 0.995
(25.02 – 25.27)
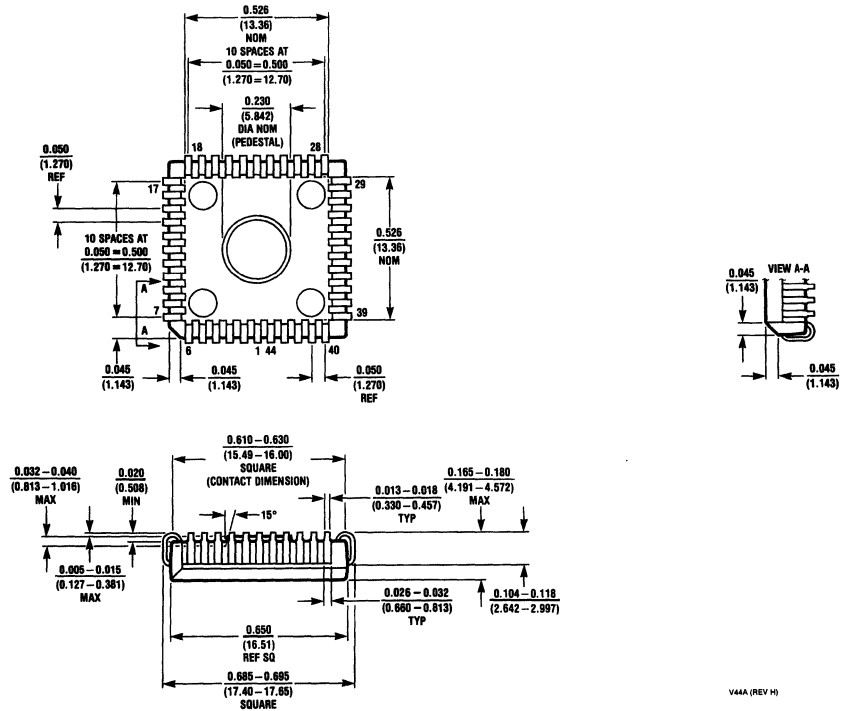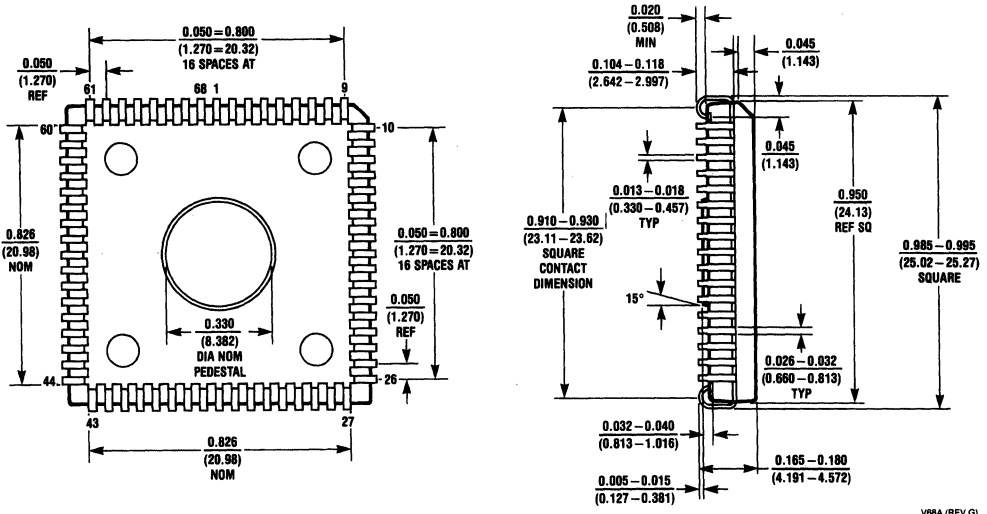SQUARE

15°

0.026 – 0.032
(0.660 – 0.813)
TYP

0.032 – 0.040
(0.813 – 1.016)

0.165 – 0.180
(4.191 – 4.572)

0.005 – 0.015
(0.127 – 0.381)

V68A (REV G)

84 Lead Plastic Chip Carrier (V)
NS Package Number V84A

1.190 ± 0.005
(30.23 ± 0.127)
SQ

0.100
(2.540)
NOM
DIA
PEDESTAL

54   74

53   75

83
84
1
2
3

1.120 ± 0.010
(28.45 ± 0.254)
SQ
CONTACT
DIMENSION

0.042 – 0.048
(1.067 – 1.219)

33   11

32   12

0.047 – 0.053
(1.193 – 1.346)
REF

20 SPACES AT
0.047 – 0.053
(1.193 – 1.346)
SQ

A   A

0.060
(1.524)
MIN
TYP

0.104 – 0.118
(2.642 – 2.997)

0.020
(0.508)
MIN

0.013 – 0.018
(0.330 – 0.457)
TYP

0.026 – 0.032
(0.660 – 0.813)
TYP

1.150
(29.21)
REF SQ

0.030 – 0.045
(0.762 – 1.143)
RAD
TYP

0.165 – 0.180
(4.191 – 4.572)

0.004
(0.102)
TOTAL

0.005 – 0.015
(0.127 – 0.381)

0.065
(2.160)
NOM

LEAD NO. 1 IDENT
0.060
(1.524)
DIA
NOM

0.042 – 0.048
(1.067 – 1.219)

0.165 – 0.180
(4.191 – 4.572)

0.042 – 0.048
(1.067 – 1.219)

VIEW A–A

V84A (REV C)

NOTES

National Semiconductor

Bookshelf of Technical Support Information

National Semiconductor Corporation recognizes the need to keep you informed about the availability of current technical literature.

This bookshelf is a compilation of books that are currently available. The listing that follows shows the publication year and section contents for each book.

Please contact your local National sales office for possible complimentary copies. A listing of sales offices follows this bookshelf.

We are interested in your comments on our technical literature and your suggestions for improvement.

Please send them to:

Technical Communications Dept. M/S 16300
2900 Semiconductor Drive
P.O. Box 58090
Santa Clara, CA 95052-8090

ALS/AS LOGIC DATABOOK—1987

Introduction to Bipolar Logic • Advanced Low Power Schottky • Advanced Schottky

ASIC DESIGN MANUAL/GATE ARRAYS & STANDARD CELLS—1987

SSI/MSI Functions • Peripheral Functions • LSI/VLSI Functions • Design Guidelines • Packaging

CMOS LOGIC DATABOOK—1988

CMOS AC Switching Test Circuits and Timing Waveforms • CMOS Application Notes • MM54HC/MM74HC
MM54HCT/MM74HCT • CD4XXX • MM54CXXX/MM74CXXX • Surface Mount

DATA ACQUISITION LINEAR DEVICES—1989

Active Filters • Analog Switches/Multiplexers • Analog-to-Digital Converters • Digital-to-Analog Converters
Sample and Hold • Temperature Sensors • Voltage Regulators • Surface Mount

DATA COMMUNICATION/LAN/UART DATABOOK—1990

LAN IEEE 802.3 • High Speed Serial/IBM Data Communications • ISDN Components • UARTs
Modems • Transmission Line Drivers/Receivers

DISCRETE SEMICONDUCTOR PRODUCTS DATABOOK—1989

Selection Guide and Cross Reference Guides • Diodes • Bipolar NPN Transistors
Bipolar PNP Transistors • JFET Transistors • Surface Mount Products • Pro-Electron Series
Consumer Series • Power Components • Transistor Datasheets • Process Characteristics

DRAM MANAGEMENT HANDBOOK—1989

Dynamic Memory Control • Error Detection and Correction • Microprocessor Applications for the
DP8408A/09A/17/18/19/28/29 • Microprocessor Applications for the DP8420A/21A/22A
Microprocessor Applications for the NS32CG821

EMBEDDED SYSTEM PROCESSOR DATABOOK—1989

Embedded System Processor Overview • Central Processing Units • Slave Processors • Peripherals
Development Systems and Software Tools

F100K DATABOOK—1989

Family Overview • F100K Datasheets • 11C Datasheets • 10K and 100K Memory Datasheets
Design Guide • Circuit Basics • Logic Design • Transmission Line Concepts • System Considerations
Power Distribution and Thermal Considerations • Testing Techniques • Quality Assurance and Reliability

FACT™ ADVANCED CMOS LOGIC DATABOOK—1989
Description and Family Characteristics • Ratings, Specifications and Waveforms
Design Considerations • 54AC/74ACXXX • 54ACT/74ACTXXX

FAST® ADVANCED SCHOTTKY TTL LOGIC DATABOOK—Rev. 1—1988
Circuit Characteristics • Ratings, Specifications and Waveforms • Design Considerations • 54F/74FXXX

FAST® APPLICATIONS HANDBOOK—REPRINT
Reprint of 1987 Fairchild FAST Applications Handbook
Contains application information on the FAST family: Introduction • Multiplexers • Decoders • Encoders
Operators • FIFOs • Counters • TTL Small Scale Integration • Line Driving and System Design
FAST Characteristics and Testing • Packaging Characteristics • Index

GENERAL PURPOSE LINEAR DEVICES DATABOOK—1989
Continuous Voltage Regulators • Switching Voltage Regulators • Operational Amplifiers • Buffers • Voltage Comparators
Instrumentation Amplifiers • Surface Mount

GRAPHICS HANDBOOK—1989
Advanced Graphics Chipset • DP8500 Development Tools • Application Notes

INTERFACE DATABOOK—1988
Transmission Line Drivers/Receivers • Bus Transceivers • Peripheral Power Drivers • Display Drivers
Memory Support • Microprocessor Support • Level Translators and Buffers • Frequency Synthesis • Hi-Rel Interface

LINEAR APPLICATIONS HANDBOOK—1986
The purpose of this handbook is to provide a fully indexed and cross-referenced collection of linear integrated circuit
applications using both monolithic and hybrid circuits from National Semiconductor.

Individual application notes are normally written to explain the operation and use of one particular device or to detail various
methods of accomplishing a given function. The organization of this handbook takes advantage of this innate coherence by
keeping each application note intact, arranging them in numerical order, and providing a detailed Subject Index.

LS/S/TTL DATABOOK—1989
Contains former Fairchild Products
Introduction to Bipolar Logic • Low Power Schottky • Schottky • TTL • TTL—Low Power

MASS STORAGE HANDBOOK—1989
Rigid Disk Pulse Detectors • Rigid Disk Data Separators/Synchronizers and ENDECs
Rigid Disk Data Controller • SCSI Bus Interface Circuits • Floppy Disk Controllers • Disk Drive Interface Circuits
Rigid Disk Preamplifiers and Servo Control Circuits • Rigid Disk Microcontroller Circuits • Disk Interface Design Guide

MEMORY DATABOOK—1988
PROMs, EPROMs, EEPROMs • Flash EPROMs and EEPROMs • TTL I/O SRAMs
ECL I/O SRAMs • ECL I/O Memory Modules

MICROCONTROLLER DATABOOK—1989
COP400 Family • COP800 Family • COPS Applications • HPC Family • HPC Applications
MICROWIRE and MICROWIRE/PLUS Peripherals • Microcontroller Development Tools

MICROPROCESSOR DATABOOK—1989
Series 32000 Overview • Central Processing Units • Slave Processors • Peripherals
Development Systems and Software Tools • Application Notes • NSC800 Family

PROGRAMMABLE LOGIC DATABOOK & DESIGN MANUAL—1989
Product Line Overview • Datasheets • Designing with PLDs • PLD Design Methodology • PLD Design Development Tools
Fabrication of Programmable Logic • Application Examples

REAL TIME CLOCK HANDBOOK—1989

Real Time Clocks and Timer Clock Peripherals • Application Notes

RELIABILITY HANDBOOK—1986

Reliability and the Die • Internal Construction • Finished Package • MIL-STD-883 • MIL-M-38510
The Specification Development Process • Reliability and the Hybrid Device • VLSI/VHSIC Devices
Radiation Environment • Electrostatic Discharge • Discrete Device • Standardization
Quality Assurance and Reliability Engineering • Reliability and Documentation • Commercial Grade Device
European Reliability Programs • Reliability and the Cost of Semiconductor Ownership
Reliability Testing at National Semiconductor • The Total Military/Aerospace Standardization Program
883B/RETS™ Products • MILS/RETS™ Products • 883/RETS™ Hybrids • MIL-M-38510 Class B Products
Radiation Hardened Technology • Wafer Fabrication • Semiconductor Assembly and Packaging
Semiconductor Packages • Glossary of Terms • Key Government Agencies • AN/ Numbers and Acronyms
Bibliography • MIL-M-38510 and DESC Drawing Cross Listing

SPECIAL PURPOSE LINEAR DEVICES DATABOOK—1989

Audio Circuits • Radio Circuits • Video Circuits • Motion Control Circuits • Special Function Circuits
Surface Mount

TELECOMMUNICATIONS—1987

Line Card Components • Integrated Services Digital Network Components • Modems
Analog Telephone Components • Application Notes

National Semiconductor

National Semiconductor Corporation
2900 Semiconductor Drive
P.O. Box 58090
Santa Clara, CA 95052-8090
Tel: (408) 721-5000
TWX: (910) 339-9240

SALES OFFICES (Continued)

INTERNATIONAL OFFICES

Electronica NSC de Mexico SA
Juventino Rosas No. 118-2
Col Guadalupe Inn
Mexico, 01020 D.F. Mexico
Tel: 52-5-524-9402

National Semicondutores Do Brasil Ltda.
Av. Brig. Faria Lima, 1383
6.0 Andor-Conj. 62
01451 Sao Paulo, SP, Brasil
Tel: (55/11) 212-5066
Fax: (55/11) 211-1181 NSBR BR

National Semiconductor GmbH
Industriestrasse 10
D-8080 Furstenfeldbruck
West Germany
Tel: (0-81-41) 103-0
Telex: 527-649
Fax: (08141) 103554

National Semiconductor (UK) Ltd.
The Maple, Kembrey Park
Swindon, Wiltshire SN2 6UT
United Kingdom
Tel: (07-93) 61-41-41
Telex: 444-674
Fax: (07-93) 69-75-22

National Semiconductor Benelux
Vorstlaan 100
B-1170 Brussels
Belgium
Tel: (02) 6-61-06-80
Telex: 61007
Fax: (02) 6-60-23-95

National Semiconductor (UK) Ltd.
Ringager 4A, 3
DK-2605 Brondby
Denmark
Tel: (02) 43-32-11
Telex: 15-179
Fax: (02) 43-31-11

National Semiconductor S.A.
Centre d'Affaires-La Boursidiere
Bâtiment Champagne, B.P. 90
Route Nationale 186
F-92357 Le Plessis Robinson
France
Tel: (1) 40-94-88-88
Telex: 631065
Fax: (1) 40-94-88-11

National Semiconductor (UK) Ltd.
Unit 2A
Clonskeagh Square
Clonskeagh Road
Dublin 14
Tel: (01) 69-55-89
Telex: 91047
Fax: (01) 69-55-89

National Semiconductor S.p.A.
Strada 7, Palazzo R/3
I-20089 Rozzano
Milanofiori
Italy
Tel: (02) 8242046/7/8/9
Twx: 352647
Fax: (02) 8254758

National Semiconductor S.p.A.
Via del Cararaggio, 107
00147 Rome
Italy
Tel: (06) 5-13-48-80
Fax: (06) 5-13-79-47

National Semiconductor (UK) Ltd.
P.O. Box 29
N-1321 Stabekk
Norway
Tel: (2) 12-53-70
Fax: (2) 12-53-75

National Semiconductor AB
Box 1009
Grosshandlarvagen 7
S-12123 Johanneshov
Sweden
Tel: (08) 7228050
Fax: (08) 7229095

National Semiconductor GmbH
Calle Agustin de Foxa, 27 (9°D)
28036 Madrid
Spain
Tel: (01) 733-2958
Telex: 46133
Fax: (01) 733-8018

National Semiconductor Switzerland
Alte Winterthurerstrasse 53
Postfach 567
Ch-8304 Wallisellen-Zurich
Switzerland
Tel: (01) 830-2727
Telex: 828-444
Fax: (01) 830-1900

National Semiconductor
Kauppakartanonkatu 7 A22
SF-00930 Helsinki
Finland
Tel: (90) 33-80-33
Telex: 126116
Fax: (90) 33-81-30

National Semiconductor
Postbus 90
1380 AB Weesp
The Netherlands
Tel: (0-29-40) 3-04-48
Telex: 10-956
Fax: (0-29-40) 3-04-30

National Semiconductor Japan Ltd.
Sanseido Bldg. 5F
4-15 Nishi Shinjuku
Shinjuku-ku
Tokyo 160 Japan
Tel: 3-299-7001
Fax: 3-299-7000

National Semiconductor Hong Kong Ltd.
Suite 513, 5th Floor,
Chinachem Golden Plaza,
77 Mody Road, Tsimshatsui East,
Kowloon, Hong Kong
Tel: 3-7231290
Telex: 52996 NSSEA HX
Fax: 3-3112536

National Semiconductor (Australia) PTY, Ltd.
1st Floor, 441 St. Kilda Rd.
Melbourne, 3004
Victory, Australia
Tel: (03) 267-5000
Fax: 61-3-2677458

National Semiconductor (PTE), Ltd.
200 Cantonment Road 13-01
Southpoint
Singapore 0208
Tel: 2252226
Telex: RS 33877

National Semiconductor (Far East) Ltd.
Taiwan Branch
P.O. Box 68-332 Taipei
7th Floor, Nan Shan Life Bldg.
302 Min Chuan East Road,
Taipei, Taiwan R.O.C.
Tel: (86) 02-501-7227
Telex: 22837 NSTW
Cable: NSTW TAIPEI

National Semiconductor (Far East) Ltd.
Korea Branch
13th Floor, Dai Han Life Insurance
63 Building,
60, Yoido-dong, Youngdeungpo-ku,
Seoul, Korea 150-763
Tel: (02) 784-8051/3, 785-0696/8
Telex: 24942 NSPKLO
Fax: (02) 784-8054