1394 Design Schematic (TSBKPRPHRL)

The enclosed information is a copy of the documentation supplied to Texas Instruments by TechnoBox as documentation for a test board designed and built by TechnoBox for Texas Instruments. Texas Instruments does not guarantee the accuracy of this information. There are no known errors in this document.

TEXAS INSTRUMENTS

1394 Solutions Leader

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain applications using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

Errata for Peripheral EVM board - 07/06/96

On schematic page 6:
The connections to the power programming pins on the TSB11C01 are all tied low. Per the 1394-1995 standard table for the Self-ID Packet this is incorrect. This node is self-powered and supplies power to the cable. The amount of power will depend on the power supply used with the external power connector shown on schematic page 2.

Also note this board correctly pulls the TSB11C01 Configuration Manager Contender (CMC) pin low to indicate that this board is not a contender to be the bus manager or isochronous resource manager. However what is missing for this board to be eligible for those roles is the necessary SW. If the user creates SW to support these roles this pin would need to be pulled high.

The files included on this disk are compressed using the utility PKZIP
and may be uncompressed using the PKUNZIP utility.

This design does not incorporate galvanic isolation.

These are the files included on this disk + the AAREADME.TXT (this file).
STARTER ZIP
STRTCODE ZIP
M210PRIO ZIP
M445STRT ZIP

The file STRTCODE.ZIP contains:
the TMS320C52 "C" and assembly code source,
the executable files, and
and hex files for use with the peripheral EVM.

The file STARTER.ZIP contain:
the softcopy of the design introduction,
the softcopy of the programmer's guide,
softcopies of various manufacturing documentation files,
the ORCAD schematic data bases, and
the PC-CARDs layout database

The file M210PRIO.ZIP contains:
the programming for the MACH210 PLD

The file M445STRT.ZIP contains:
the programming for the MACH445 PLD

The files that will be uncompressed have the following extensions:

.asm - TMS320C52 assembler source code
.bat - DOS batch files
*.c - TMS320C52 C source code
.cfg - PCAD configuration files
.cmd - Command files for the DOS based TMS320C52 linker, emulator, & ROM HEX

code generator
.doc - Documentation files in either Microsoft Word format or text format.
.HEX - ROM code
.HXM - Hex map created by ROM HEX code generator
.LIB - Library file
.LST - Listing file
.map - Link map
.obj - TMS320C52 intermediate object code
.out - TMS320C52 executable code
.pcb - PCAD PWB database
.sch - ORCAD format schematic pages
.src - Source code for the programming inside the FPGA

For information on the PC/104 interface and available PC/104 compatible cards, please see http://www.pc104.com.

For the latest information about this or any other 1394 EVM kit offered by TI, please see http://www.ti.com/sc/1394.

Acknowledgement:

This 12C01/11C01 Starter Kit was designed for Texas Instruments by Technobox, Inc.

Technobox, Inc.
12-B The Ellipse
Suite 300
Mt. Laurel, NJ 08054

V: 609-778-5512
F: 609-727-5346
E: techno@novalink.com
W: www.technobox.com

12C01/11C01 IEEE 1394 Starter Kit

Introduction

The IEEE 1394 “starter kit” design provides a Texas Instruments 12C01/11C01 IEEE 1394 interface, an RS232 interface, PC/104 expansion interface, and a Voice Band TLC32046 A:D converter. All these units are programmed by a TMS320C52 DSP with 64Kword of 16-bit FLASH memory and 64K words of SRAM holding the DSP program and data spaces.

A single MACH445 complex PLD provides the hardware control for the design. It implements a single channel DMA controller which transfers data between the 12C01 interface and the SRAM.

A 2-byte to 4-byte transceiver bridges the 16-bit DSP bus with the 32-bit 12C01/SRAM bus. Data Space accesses from the DSP are mapped into the SRAM, so the DSP has direct access to the SRAM for variable storage, as well as IEEE 1394 buffer pools.

The PC104 expansion interface is essentially a 16-bit “ISA” bus found in Personal Computers. The “PC104” term applies for the unique mechanical packaging of the ISA bus in a “stacked” fashion, thereby allowing off-the-shelf I/O functions, such as SCSI, Video, Data Acquisition and Prototype boards to be used with the DSP starter kit. The “PC104” specification is controlled by the PC104 Consortium, and over 100 companies are currently producing PC104 modules.

The Starter Kit design supports Bus Mastering of the PC104 bus; i.e., PC104 connected to the Starter Kit are slaves on the bus. The Starter Kit does not support slave access by other bus members on the PC104 bus (i.e., “Master*” signal is not supported). Note that the vast majority of PC104 boards do not support Master*.

The design supports DRQ/DACK interface on the ISA bus for handling DMA transfers from an I/O device to memory. The DRQ becomes an interrupt request to the DSP, and because of the fast interrupt context switching feature of the TMS320C52, adequate performance for many PC104 applications can be expected.

Since the performance of the ISA bus is slow compared to the DSP bus, the Starter Kit design features a programmable timer located in the MACH445 to control timing of the read/write strobes on the ISA bus. When this timer is set to a minimal value, the ISA bus is accessed at DSP bus speeds. This is useful for custom designs which are not bottlenecked by the ISA bus timing.

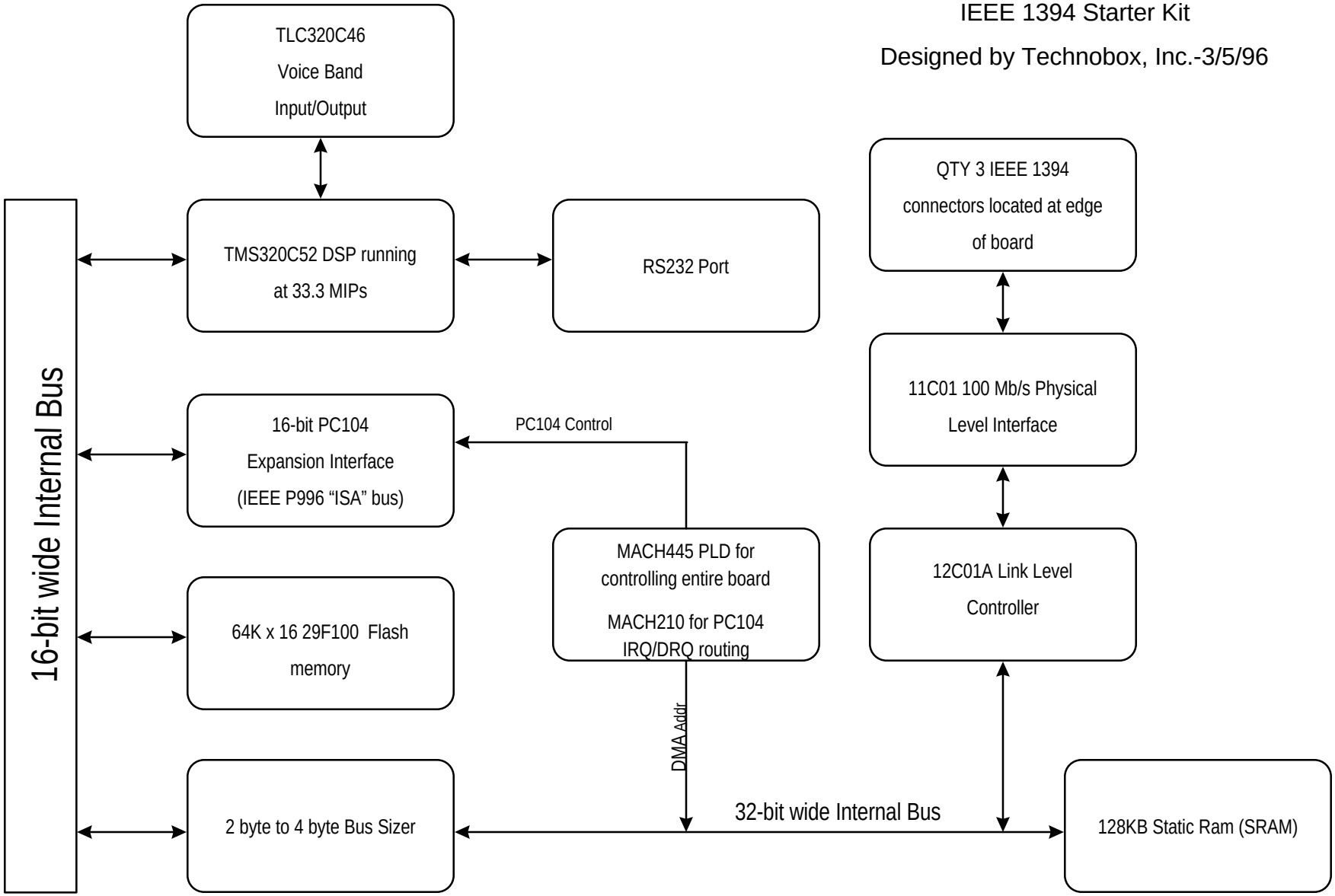
DRQ and IRQ lines are routed to their respective INT2, INT3, and INT4 inputs to the DSP via a MACH210 complex PLD. Thus, both ISA interrupts and DRQ requests are serviced in DSP firmware.

The TLC32046 Voice Band interface provides an audio-bandwidth D:A and A:D channel, and these channels are connected to external amplifiers via standard “RCA” style jacks.

Power is provided by a either a Personal Computer 4-pin Hard Disk connector (+12, +5VDC), or by a standard PC104 power module. Power for the 11C01 is from the Cable power regulated down by a linear regulator to 5 VDC. The board also sources Cable Power via a schottky diode as found in most IEEE 1394 implementations.

IEEE 1394 Starter Kit

Designed by Technobox, Inc.-3/5/96



Programmer's guide to the IEEE 1394 Starter Kit

Introduction

The IEEE 1394 Starter Kit provides a variety of interfaces all controlled by a TMS320C52 Digital Signal Processor (DSP). The design provides a three-port IEEE 1394 100 Mb/s interfaced based on Texas Instruments 11C01/12C01 chipset, a Voice Band Audio interface (both analog in and out) based on the TLS32046 AIC from TI, and an expansion interface to a 16-bit industry standard ISA bus in a PC/104 form factor.

64Kwords of FLASH memory based program space visible to the TMS320C52, as well as 64K words of Static Random Access Memory (SRAM). There are also regions of the TMS320C52 data space which is used to access on board registers and off board components on the PC/104 bus. The TMS320C52 I/O space is not used in this design.

This section details registers accessible from the TMS320C52. In the case of more complex functions, such as the 11C01/12C01 IEEE 1394 chipset, the user is referred to the relevant documentation provided by the chipset manufacture.

Although the registers presented herein are directly accessible through TMS320C52 assembly-level language, it is highly recommended that the C-language equivalents be used to implement programs for the Starter Kit. The optimizing C compilers, available from Texas Instruments, do a fine job in creating efficient machine code which can only slightly be improved upon by hand assembly coding.

As another foundation for programming the Starter Kit, it's highly recommended that the user examine the RS232 Monitor Program source code which is provided with the Starter Kit software package. In many cases the code used by the monitor can be duplicated or directly used by the Starter Kit programmer to accomplish the requisite tasks. For example, there are routines which perform DMA transfers between the 12C01 and SRAM which would otherwise be fairly difficult to implement without a thorough understanding of the Starter Kit hardware.

Overall Memory Map

The TMS320C52 defines three disjoint memory spaces, each 64 Kwords in size. Note that the TMS320C52 is a 16-bit machine, and a 'word' in this context is 16-bits.

First, the 'program space' is the area in which TMS320C52 machine instructions are located. Programs execute out of the program space. The TMS320C52 also has some special instructions to access data within program space, but this is normally transparent to the programmer at the C-language level. In the Starter Kit design, the program space is implemented with a 64Kword x 16-bit FLASH memory. The initial code in the FLASH memory is the RS232-based Monitor Program, which provides a simple menu-driven human interface to interrogate, access, and initialize Starter Kit hardware components. The monitor provides a mechanism to upload new monitor code and re-burn the FLASH memory in-circuit accordingly.

As part of the TMS320C52 architecture, some of the program space is located in SRAM within the TMS320C52 DSP chip. This region is from 0xFE00 to 0xFFFF in program space. When the DSP accesses programs in this address range, the instructions are actually coming from within the chip as opposed to the external FLASH memory. Execution is significantly faster within this on-chip space; a full 33 MIPs can be achieved in the Starter Kit case. Execution from the FLASH is done with 2 DSP wait states, since the FLASH used in the design is 90-ns access. Program execution out of external FLASH effectively cuts the performance to 11 MIPS, which is still respectable for many applications.

Note that the On-Chip program space is automatically initialized with selected high-speed programs used in the RS232 Monitor Program. This code is found in the 'RAMCODE.ASM' program, and is mapped using the 'ramcode' segment in the TMS320C5X linker.

A second distinctly addressable region is 'data space.' Here, the DSP instruction set provides a rich assortment of address modes, both direct and indirect with autoincrement options. In the Starter Kit design, a 64-Kword SRAM is visible from the DSP data space. The Starter Kit hardware registers are also visible from Data Space, including access to the PC/104 expansion bus.

A third addressable region is 'I/O space.' The DSP provides a limited number of address modes and instructions to access I/O space (specifically 'IN' and 'OUT'). The Starter Kit does not use I/O space. Note that the C-Language does not provided constructs to easily access I/O space, but it's very easy to access Data Space, which is why the Starter Kit implements the hardware registers in DSP Data Space.

There's yet another DSP addressable space, called 'global memory,' which is normally used in multi-processing shared memory architectures. The Starter Kit does not use global memory.

To summarize, the 'overall' memory map for the IEEE 1394 Starter Kit is as follows:

DSP space	Start	End	Description
Program	0x0000	0xFDFF	Holds RS232 Monitor Program
Program	0xFE00	0xFFFF	On chip Program Space
Data	0x0000	0x02FF	DSP internal registers
Data	0x0300	0x04FF	On-chip general purpose SRAM
Data	0x0500	0xEFFF	Starter Kit SRAM
Data	0xF000	0xFFFF	Starter Kit hardware registers
I/O	0x0000	0xFFFF	Not used in Starter Kit design

Most of the spaces as described above should be familiar to a programmer who wishes to use the Starter Kit. However, the Data Space registers in the range 0xF000 to 0xFFFF are external hardware registers peculiar to the Starter Kit design, and these are described later in this programming guide.

Starter Kit hardware Register Summary

The specific addresses for the Starter Kit registers visible within the TMS320C52 data space are as follows:

Mnemonic	Data Space Addr	Description
la_spareF0	0xF000h	Spare decode (not used)
la_spareF1	0xF100h	Spare decode (not used)
la_11c01	0xF200h	Load 11C01 chip address latch for 12C01
la_pc104adr	0xF300h	Write PC104 address register
la_trig	0xF400h	Logic Analyzer Trigger
la_ctrlstat	0xF500h	Write control register/Read Status reg.
la_rdytime	0xF600h	PC104 sample ready delay
la_xfercntr	0xF700h	Read/write transfer counter
la_dmago	0xF800h	Write here kicks off DMA transfer
la_addrhi	0xF900h	Load address counter hi
la_addrlo	0xFA00h	Load address counter low
la_pc104dat	0xFB00h	Read/Write PC104 space
la_dmactrl	0xFC00h	DMA control register (Contains GRRDYEN)
la_wrprio	0xFD00h	Write PC/104 IRQ/DRQ select mux
la_ldpg	0xFE00h	Load PC/104 Page Register
la_lddack	0xFF00h	Load PC/104 Address Register

In this table the 'Mnemonic' is the same character string used in the MACH445 design, which contains the address recognition for the DSP access to hardware registers, and also used in the C-language code for the RS232 Monitor Program which is burned into the Starter Kit FLASH memory.

Accessing Starter Kit Registers from C-Language

Since most programmers will be using C-language to manipulate the Starter Kit hardware registers, this section will discuss what is involved in using C-language in this regard.

First, the register addresses need to be defined using the C-language #DEFINE statement. #DEFINE's for the Starter Kit hardware are found in the RS232 Monitor Program code (STRT.C), and is represented below:

```
#define la_spareF0    (*(volatile int *)0xF000)  /* Spare address                    */
#define la_spareF1    (*(volatile int *)0xF100)  /* Spare address                    */
#define la_11c01      (*(volatile int *)0xF200)  /* Load 11C01 chip address latch    */
#define la_pc104adr           (*(volatile int *)0xF300)  /* PC/104 address register  */
#define la_trig       (*(volatile int *)0xF400)  /* Logic Analyzer Trigger           */
#define la_ctrlstat   (*(volatile int *)0xF500)  /* Write control register/Read Status reg  */
#define la_rdytime    (*(volatile int *)0xF600)  /* PC/104 sample ready delay        */
#define la_xfercntr   (*(volatile int *)0xF700)  /* Read/write transfer counter      */
#define la_dmago      (*(volatile int *)0xF800)  /* Write here kicks off DMA transfer  */
#define la_addrhi     (*(volatile int *)0xF900)  /* Load address counter hi          */
#define la_addrlo     (*(volatile int *)0xFA00)  /* Load address counter low         */
#define la_pc104dat   (*(volatile int *)0xFB00)  /* PC/104 data bus read/write       */
#define la_dmactrl    (*(volatile int *)0xFC00)  /* DMA control register             */
#define la_wrprio2    (*(volatile int *)0xFD00)  /* Write MACH210 DRQ/IRQ mux control INT2  */
#define la_wrprio3    (*(volatile int *)0xFD01)  /* Write MACH210 DRQ/IRQ mux control INT3  */
#define la_wrprio4    (*(volatile int *)0xFD02)  /* Write MACH210 DRQ/IRQ mux control INT4  */
#define la_ldpg       (*(volatile int *)0xFE00)  /* Load PC/104 page register                */
#define la_lddack     (*(volatile int *)0xFF00)  /* Load PC/104 DACK register                */
```

Note that each address is first cast as a 'pointer' type (using "(volatile int *)") and then the C-language 'indirect' operator ("*") is used to access the register. Thus, a programmer can access the 'contents' of the hardware register simply by using the la_xxxxx reference.

For example, to write a constant to the PC/104 bus, assuming the bus control registers have been set up by some prior code, one can write:

```
la_pc104dat = 0x1234;          /* Write PC104 bus with constant 0x1234 */
```

Similarly to read the PC/104 bus, one writes:

```
k = la_pc104dat;               /* Read PC/104 bus into variable “k” */
```

Note that the use of the ‘volatile’ modifier in the pointer cast is required so the C compiler does not optimize out the reference. If this were not used, then the global optimizer (may) observe the reference was not changed in prior code and consequently remove the code from the program!

How to access the TLS32046 AIC

The Starter Kit has a TLS32046 based Analog Interface Circuit (AIC) which digitizes voice-band analog input and generates analog output. The analog in and analog out is accessed via ‘RCA’ phone jacks at the edge of the board. Note that the ‘out’ jack is closest to the 4-pin Power connector.

The AIC uses a synchronous serial connection to the TMS320C52 DSP. The DSP converts this serial data to a 16-bit parallel form and the data can be read and written by accessing internal DSP registers.

When the Starter Kit is powered up or reset, the RS232 Monitor Program sets up the AIC to sample at 16khz. It also sets up the TMS320C52 to operate in an interrupt mode to send and receive analog data samples. Also, for each transmit and receive interrupt a 16-bit global location in low DSP data space is incremented; by observing when these locations change, the user can synchronize the supply of analog data to the AIC sample rate.

The Interrupt service routines for the AIC are shown below. These are located in RAMCODE.ASM

```
;
; TLS32046 AIC RECEIVER INTERRUPT SERVICE ROUTINE
;

receive:    lamm  DRR           ;Read new sample value
            samm  _adcdata      ;Save it in sample location
            lamm  _adccount     ;
            add   #1h           ;Increment sample counter
            samm  _adccount
            rete

;
; TLS32046 transmit interrupt handler
;

transmit:   lamm  _adcout
            samm  dxr
            lamm  _adcoutcnt
            add   #1h
            samm  _adcoutcnt
            RETE                ;Return from interrupt w/enable interrupts
```

Note the use of the 'lamm' and 'samm' instructions to access the AIC related variables. Also note the incrementing of the _adccount and _adcoutcnt locations for each receive and transmit interrupt. The '_' prior to the label is a C-language to Assembly language interface convention.

The corresponding 16-bit locations used to access the AIC from C-language are as follows:

```
volatile extern int adcdata;            /* Current AIC sample value          */
volatile extern unsigned int adccount;          /* Counts ADC samples            */
volatile extern int adcout;             /* Output D:A convertor value        */
volatile extern unsigned int adcoutcnt;         /* Counter ... D:A convertor     */
```

As an example of how to interface to the AIC from C-language, the RS232 Monitor Program contains a loopback program which is used during the manufacturing test of the Starter Kit, and that test is duplicated here:

```
/* Loopback audio output to audio input     */

void analogloop()
{
 int i,j,k;
 unsigned int maxdiff;
 unsigned int diff;

 maxdiff = 0;
 for (i = 0; i < 5000; i++)
 {
  adcout = randu() & 0xfffc;
  for (j = 0; j < 30; j++)
  {
   while (k == adcoutcnt) {};
   k = adcoutcnt;
  };
  diff = abs(adcout - (adcdata * 2));
  if (diff > maxdiff) maxdiff = diff;
  if (diff > 3000)
  {
   j = adcout;
```

```
    k = adcdata;
    write("Error: Analog IN and OUT data not equal.");
    write("  Out = ");
    wordascii(j);
    write("  In = ");
    wordascii(k);
    writeln("");
   };
  }
 write("Maximum input/output difference during testing: ");
 wordascii(maxdiff);
 writeln("");
}
```

How to access the IEEE 1394 interface

The key thing to understand regarding access to the 11C01/12C01 chipset is that all transfers between the 12C01 are accomplished using DMA transfers to the 64Kword Starter Kit SRAM. Even reading and writing 12C01 control registers are done through the DMA mechanism.

The procedure for performing a DMA transfer is summarized below+

1. Set up the 12C01 address latch to point to the 12C01 register you are interested in accessing. This also sets up the TRW input to the 12C01 for the subsequent DMA transfer. You do this by reading/writing la_11C01 with address of register you will be accessing.
2. Set up the DMA hardware with a starting address in SRAM for the transfer.
3. Set up the DMA hardware with a 'transfer count' indicating the number of quadlets to transfer.
4. Initiate the DMA operation by writing to la_dmago.
5. Wait for DMA operation completion by polling a hardware status bit.

The RS232 Monitor Program contains two C-language procedures to set up the DMA controller for either a read or write operation:

```
/* Set up DMA controller for read transfer and start DMA transfer */

void dmaread(int srcadr, int destadr, int cnt)

{
 int i,j;

 la_xfercntr = cnt;                          /* Load transfer counter */
 la_addrhi = destadr >> 8;
 la_addrlo = destadr;                        /* Load addr cntr hi/lo */
 j = *(volatile unsigned int *)(la_11c01_addr + srcadr);   /* Set up source address */
 la_dmago = 0;                               /* Start DMA transfer  */
}

/* Set up DMA controller for write transfer and start DMA transfer */

void dmawrite(int srcadr, int destadr, int cnt)

{
 int i,j;

 la_xfercntr = cnt;                          /* Load transfer counter */
 la_addrhi = srcadr >> 8;
 la_addrlo = srcadr;                         /* Load addr cntr hi/lo */
 j = 0;
 *(volatile unsigned int *)(la_11c01_addr + destadr) = j;   /* Set up source address */
 la_dmago = 0;                               /* Start DMA transfer  */
}
```

Note the line just before the “la_dmago = 0” is used to write the hardware address latch for identifying the address of the 12C01 register as well as set up the TRW control line going into the 12C01.

Also observe that these routines do not poll the hardware status register looking for DMA completion. This is because the TMS320C52 external bus accesses as suspended soon after the la_dmago=0 take effect (i.e., the TMS320C52 is placed in a bus ‘hold’ state), and in many cases the DMA is completed before the execution of instructions which access the DMA data.

How to access the PC/104 interface

The PC/104 interface is accessed as follows:

1. Set up the PC/104 la_wripriox (where x=2,3,4) to multiplex the DRQ/IRQ lines from the PC104 bus to the desired INTx (x=2,3,4) interrupt lines going to the TMS320C52. You only need to do this if you are using interrupt service routines to handle PC/104 DMA and IRQ requests. Note that an RS232 Monitor Program function is available to more conveniently do this operation.
2. Set up the PC/104 RD/WR strobe timing by writing appropriate values to la_rdytime. An RS232 Monitor Program function is available for this operation as well.
3. Set up the PC/104 control register to indicate IO, MEMORY, or SYSTEM MEMORY access by writing to the la_dmactrl register. Note that PC/104 has a separate set of RD/WR strobes for each of these address spaces, and the user needs to tell the Starter Kit hardware which RD/WR strobe set to use in subsequent PC/104 bus cycles.
4. Initialize the AEN, REFRESH, TC and SBHE control signals in the la_ctrlstat register appropriately for the PC/104 cycle you wish to do.
5. Identify the specific PC/104 address you want to access by writing to la_pc104adr and la_ldpg registers. This creates a 24-bit address emitted onto the PC104 bus.
6. Access the PC/104 data by reading/writing the la_pc104dat register.

Many of the above steps can be eliminated if the user finds the defaults set up by the RS232 Monitor Program is adequate for the application. The Monitor Program defaults the PC/104 bus to IO access (i.e., use IOR and IOW strobes on the PC/104 bus), 330/270 nanosecond read/write strobe width, and AEN, REFRESH, TC and SBHE all low. This is fine for 8-bit accesses to I/O ports on the PC/104 bus, such as to RS232 interface UARTS.

As an example of how to access the PC/104 bus, consider the following program fragment which echoes RS232 characters on a COM1 port attached to the PC/104 bus:

```
/* Set PC/104 modes for System memory, memory, and I/O accesses */

void setpc104mode(int i)
{
 la_dmactrl = ((la_dmactrl & pc104mode_mask_n) | (i << 1));
}

/* Set up PC/104 address bus */

void pc104addr(unsigned long addr)
{
 la_pc104adr = addr;                    /* Main address bus                    */
 la_ldpg = (addr >> 16);                /* Load PC/104 addr[23..16]    */
}

/* Echo COM1 characters of PCM-3410 PC/104 board */

void echocom1()
{
 unsigned long i;
 unsigned int k,j;

 setpc104mode(pc104mode_io);                    /* Access PC/104 bus I/O space    */
 pc104addr(0x3fb);                              /* Address Line Control Reg       */
 la_pc104dat = 0x83;                            /* Access DLL and DLH             */
 pc104addr(0x3f8);
 la_pc104dat = 6;                               /* Write DLL for 19.2 KB                  */
 pc104addr(0x3f9);
 la_pc104dat = 0;                               /* Write DLH                      */
 pc104addr(0x3fb);
 la_pc104dat = 0x03;
 writeln("Now echoing characters on PCM-3410 COM1 port at 19.2Kbaud");
 writeln(" Hit reset button to terminate");
```

```
 for(;;)
 {
  pc104addr(0x3fd);                    /* Address ready flags          */
  while ((la_pc104dat & 0x01) == 0) {}; /* Wait for received char       */
  pc104addr(0x3f8);
  la_pc104dat = la_pc104dat;           /* Echo back character          */
 };
}
```

12C01/11C01 IEEE 1394 Starter Kit register summary

Register Address	Register Mnemonic	Short Description
0xF000	la_mach_rev	Returns MACH 445 revision code
0xF200	la_11c01	Load chip address latch for 12C01
0xF300	la_pc104adr	PC/104 address register LSBs
0xF400	la_trig	Logic Analyzer Trigger
0xF500(rd)	la_ctrlstat	Hardware Control Register
0xF500(wr)	la_ctrlstat	Hardware Status Register
0xF600	la_rdytime	PC/104 Sample Ready Delay
0xF700	la_xfercntr	DMA Transfer Counter
0xF800	la_dmago	Kick off DMA transfers
0xF900	la_addrhi	Load DMA address counter high byte
0xFA00	la_addrlo	Load DMA address counter low byte
0xFB00	la_pc104dat	Access PC/104 space
0xFC00	la_dmactrl	DMA control/status register + misc
0xFD00	la_wrprio2	Write PC/104 IRQ/DRQ select multiplexer for INT2 DSP input
0xFD01	la_wrprio3	Write PC/104 IRQ/DRQ select multiplexer for INT3 DSP input
0xFD02	la_wrprio4	Write PC/104 IRQ/DRQ select multiplexer for INT4 DSP input
0xFE00	la_ldpg	Load PC/104 address register bits [23..16]
0xFF00	la_lddack	Load PC/104 DACK register

12C01/11C01 IEEE 1394 Starter Kit Register detail

Register Address: 0xF000
Register Mnemonic: la_mach_rev
Short Description: Returns MACH 445 revision code

Register Description: Returns an 8-bit value as read from the MACH 445 PLD on the board, indicating the revision level of the MACH 445 code. Used by the Monitor program to determine if the MACH 445 revision is appropriate for the code revision.

Bit Number	Mnemonic	Read/Write	Description
0	D0	Rd	Least Significant Bit
1	D1	Rd	
2	D2	Rd	
3	D3	Rd	
4	D4	Rd	
5	D5	Rd	
6	D6	Rd	
7	D7	Rd	Most Significant Bit
8			Not Used
9			Not Used
10			Not Used
11			Not Used
12			Not Used
13			Not Used
14			Not Used
15			Not Used

Register Address: 0xF200
Register Mnemonic: la_11c01
Short Description: Load chip address latch for 12C01

Register Description: The Starter Kit hardware maintains a register which holds the 12C01 register address which will be used in subsequent DMA transfers between the 12C01 and SRAM. A read or write to la_11c01, with the address pointing to the register to be accessed, will condition the hardware for a DMA read or write.

Bit Number	Mnemonic	Read/Write	Description
0	D0	Rd/Wr	Not Used
1	D1	Rd/Wr	Not Used
2	D2	Rd/Wr	Not Used
3	D3	Rd/Wr	Not Used
4	D4	Rd/Wr	Not Used
5	D5	Rd/Wr	Not Used
6	D6	Rd/Wr	Not Used
7	D7	Rd/Wr	Not Used
8	D8	Rd/Wr	Not Used
9	D9	Rd/Wr	Not Used
10	D10	Rd/Wr	Not Used
11	D11	Rd/Wr	Not Used
12	D12	Rd/Wr	Not Used
13	D13	Rd/Wr	Not Used
14	D14	Rd/Wr	Not Used
15	D15	Rd/Wr	Not Used

12C01/11C01 IEEE 1394 Starter Kit Register detail

Register Address: 0xF300
Register Mnemonic: la_pc104adr
Short Description: PC/104 address register LSBs

Register Description: Register holding the least significant 16 bits of the 24-bit ISA address as emitted on the PC/104 bus. Note that this is a write-only register.

Bit Number	Mnemonic	Read/Write	Description
0	PA0	Wr	PC/104 Address bit [0]
1	PA1	Wr	PC/104 Address bit [1]
2	PA2	Wr	PC/104 Address bit [2]
3	PA3	Wr	PC/104 Address bit [3]
4	PA4	Wr	PC/104 Address bit [4]
5	PA5	Wr	PC/104 Address bit [5]
6	PA6	Wr	PC/104 Address bit [6]
7	PA7	Wr	PC/104 Address bit [7]
8	PA8	Wr	PC/104 Address bit [8]
9	PA9	Wr	PC/104 Address bit [9]
10	PA10	Wr	PC/104 Address bit [10]
11	PA11	Wr	PC/104 Address bit [11]
12	PA12	Wr	PC/104 Address bit [12]
13	PA13	Wr	PC/104 Address bit [13]
14	PA14	Wr	PC/104 Address bit [14]
15	PA15	Wr	PC/104 Address bit [15]

Register Address: 0xF400
Register Mnemonic: la_trig
Short Description: Logic Analyzer Trigger

Register Description: A write to this address causes the "Trigger" signal on the logic analyzer headers to be asserted. The purpose of this is to provide a means for DSP firmware to trigger a logic analyzer synchronously with execution of the program.

Bit Number	Mnemonic	Read/Write	Description
0		Wr	Not Used
1		Wr	Not Used
2		Wr	Not Used
3		Wr	Not Used
4		Wr	Not Used
5		Wr	Not Used
6		Wr	Not Used
7		Wr	Not Used
8		Wr	Not Used
9		Wr	Not Used
10		Wr	Not Used
11		Wr	Not Used
12		Wr	Not Used
13		Wr	Not Used
14		Wr	Not Used
15		Wr	Not Used

12C01/11C01 IEEE 1394 Starter Kit Register
detail

Register Address: 0xF500(rd)
Register Mnemonic: la_ctrlstat
Short Description: Hardware Control Register

Register Description: A 16-bit status register is maintained in the Starter Kit hardware. This read-only register provides current hardware for a variety of signals as defined below.

Bit Number	Mnemonic	Read/Write	Description
0	stat_cystart	Rd	Cycle Start from 12C01 (Pin #50)
1	stat_cydone	Rd	Cycle Done from 12C01 (Pin #49)
2	stat_grfemp	Rd	GRF Empty signal from 12C01 (Pin #48)
3	stat_cyclout	Rd	Cycle Out signal from 12C01 (Pin #44)
4	N/A	Rd	Reads as constant "0"
5	stat_mem16	Rd	"MEM16" input from PC/104 interface
6	stat_io16	Rd	"IO16" input from PC/104 interface
7	stat_cts	Rd	Clear to Send input from RS232 interface
8			Not Used
9			Not Used
10			Not Used
11			Not Used
12			Not Used
13			Not Used
14			Not Used
15			Not Used

Register Address: 0xF500(wr)
Register Mnemonic: la_ctrlstat
Short Description: Hardware Status Register

Register Description: A 16-bit register in the Starter Kit hardware is used for control. Since it is a write-only register, an image of the contents of the register is maintained in the variable "ctrl_image" by the Monitor Firmware. This register is cleared by hardware reset.

Bit Number	Mnemonic	Read/Write	Description
0	ctrl_led0	Wr	When 0, turns "D4" LED on
1	ctrl_led1	Wr	When 0, turns "D5" LED on
2	ctrl_led2	Wr	When 0, turns "D6" LED on
3	ctrl_rts	Wr	When 1, asserts "Request to Send" on RS232 Inf.
4	ctrl_ra	Wr	When 0, applies hard-reset to 12C01/11C01
5	ctrl_aen	Wr	When 1, asserts acitve-high "AEN" on PC/104 inf.
6	ctrl_refresh	Wr	When 1, asserts "REFRESH" on PC/104 Interface
7	ctrl_tc	Wr	When 1, asserts "TC" (terminal count" on PC/104
8	ctrl_sbhe	Wr	When 1, asserts "SBHE" on PC/104 interface
9			Not Used
10			Not Used
11			Not Used
12			Not Used
13			Not Used
14			Not Used
15			Not Used

12C01/11C01 IEEE 1394 Starter Kit Register detail

Register Address: 0xF600
Register Mnemonic: la_rdytime
Short Description: PC/104 Sample Ready Delay

Register Description: A 4-bit hardware field whcih defines the number of clock cycles to delay sampling of the "ready" signal on the PC/104 interface. When 0, the PC/104 cycles will be minimal. Current value can be viewed and changed using a Monitor Menu command

Bit Number	Mnemonic	Read/Write	Description
0	TM0	Wr	Least Significant Timer bit
1	TM1	Wr	
2	TM2	Wr	
3	TM3	Wr	Most Significant Timer bit
4			Not Used
5			Not Used
6			Not Used
7			Not Used
8			Not Used
9			Not Used
10			Not Used
11			Not Used
12			Not Used
13			Not Used
14			Not Used
15			Not Used

Register Address: 0xF700
Register Mnemonic: la_xfercntr
Short Description: DMA Transfer Counter

Register Description: This 8-bit quantity is loaded with the number of Quadlets (ie, 4-byte data values) transfered between SRAM and the 12C01. It counts down to zero during a DMA transfer. Note that the register is readable as well as writable

Bit Number	Mnemonic	Read/Write	Description
0	XFR0	Rd/Wr	Transfer counter least significant bit
1	XFR1	Rd/Wr	
2	XFR2	Rd/Wr	
3	XFR3	Rd/Wr	
4	XFR4	Rd/Wr	
5	XFR5	Rd/Wr	
6	XFR6	Rd/Wr	
7	XFR7	Rd/Wr	Transfer counter most significant bit
8			Not Used
9			Not Used
10			Not Used
11			Not Used
12			Not Used
13			Not Used
14			Not Used
15			Not Used

12C01/11C01 IEEE 1394 Starter Kit Register
detail

Register Address: 0xF800
Register Mnemonic: la_dmago
Short Description: Kick off DMA transfers

Register Description: A write to the register will start the DMA transfers between SRAM and the 12C01. Note that the DMA direction and 12C01 register address was set up by a prior read or write to la_11c01. The data associated with this write is meaningless.

Bit Number	Mnemonic	Read/Write	Description
0		Wr	Not Used
1		Wr	Not Used
2		Wr	Not Used
3		Wr	Not Used
4		Wr	Not Used
5		Wr	Not Used
6		Wr	Not Used
7		Wr	Not Used
8		Wr	Not Used
9		Wr	Not Used
10		Wr	Not Used
11		Wr	Not Used
12		Wr	Not Used
13		Wr	Not Used
14		Wr	Not Used
15		Wr	Not Used

Register Address: 0xF900
Register Mnemonic: la_addrhi
Short Description: Load DMA address counter high byte

Register Description: The 16-bit DMA address counter is loaded into the Starter Kit hardware via two 8-bit writes. This particular write will load DMA address counter bits [15..8].

Bit Number	Mnemonic	Read/Write	Description
0	ADDR8	Wr	DMA transfer address bit [8]
1	ADDR9	Wr	DMA transfer address bit [9]
2	ADDR10	Wr	DMA transfer address bit [10]
3	ADDR11	Wr	DMA transfer address bit [11]
4	ADDR12	Wr	DMA transfer address bit [12]
5	ADDR13	Wr	DMA transfer address bit [13]
6	ADDR14	Wr	DMA transfer address bit [14]
7	ADDR15	Wr	DMA transfer address bit [15]
8			Not Used
9			Not Used
10			Not Used
11			Not Used
12			Not Used
13			Not Used
14			Not Used
15			Not Used

12C01/11C01 IEEE 1394 Starter Kit Register
detail

Register Address: 0xFA00
Register Mnemonic: la_addr0
Short Description: Load DMA address counter low byte

Register Description: The 16-bit DMA address counter is loaded into the Starter Kit hardware via two 8-bit writes. This particular write will load DMA address counter bits [7..0].

Bit Number	Mnemonic	Read/Write	Description
0	ADDR0	Wr	DMA transfer address bit [0]
1	ADDR1	Wr	DMA transfer address bit [1]
2	ADDR2	Wr	DMA transfer address bit [2]
3	ADDR3	Wr	DMA transfer address bit [3]
4	ADDR4	Wr	DMA transfer address bit [4]
5	ADDR5	Wr	DMA transfer address bit [5]
6	ADDR6	Wr	DMA transfer address bit [6]
7	ADDR7	Wr	DMA transfer address bit [7]
8			Not Used
9			Not Used
10			Not Used
11			Not Used
12			Not Used
13			Not Used
14			Not Used
15			Not Used

Register Address: 0xFB00
Register Mnemonic: la_pc104dat
Short Description: Access PC/104 space

Register Description: A read or write to this DSP Data Space address will perform an ISA bus cycle on the PC/104 interface. The address of the PC/104 access is set up by prior writes to the la_pc104adr and la_idpg registers.

Bit Number	Mnemonic	Read/Write	Description
0	PC104D0	Rd/Wr	Least Significant PC/104 data bus bit
1	PC104D1	Rd/Wr	
2	PC104D2	Rd/Wr	
3	PC104D3	Rd/Wr	
4	PC104D4	Rd/Wr	
5	PC104D5	Rd/Wr	
6	PC104D6	Rd/Wr	
7	PC104D7	Rd/Wr	
8	PC104D8	Rd/Wr	
9	PC104D9	Rd/Wr	
10	PC104D10	Rd/Wr	
11	PC104D11	Rd/Wr	
12	PC104D12	Rd/Wr	
13	PC104D13	Rd/Wr	
14	PC104D14	Rd/Wr	
15	PC104D15	Rd/Wr	Most Significant PC/104 data bus bit

12C01/11C01 IEEE 1394 Starter Kit Register detail

Register Address: 0xFC00
Register Mnemonic: la_dmactrl
Short Description: DMA control/status register + misc

Register Description: The register, located in the MACH445 PLD, has control for the DMA hardware. It also contains some other bits related to the PC/104 interface and 12LV31 handshake/pulse mode. Operates both as a control and status register

Bit Number	Mnemonic	Read/Write	Description
0	dmactrl_grlrdyen	Rd/Wr	When set, makes DMA hardware observe "GRFEMP"
1	dmactrl_pc104md0	Rd/Wr	Establishes which space on PC/104 is accessed
2	dmactrl_pc104md1	Rd/Wr	Establishes which space on PC/104 is accessed
3	0	Rd	Zero
4	0	Rd	Zero
5	dmastat_busy	Rd	When set, DMA transfer is not yet completed (busy)
6	dmastat_timeout	Rd	When set, a timeout detected during DMA w/12C01
7	dmastat_trw	Rd/Wr	When set, DMA direction is 12C01 to SRAM
8			Not Used
9			Not Used
10			Not Used
11			Not Used
12			Not Used
13			Not Used
14			Not Used
15			Not Used

Register Address: 0xFD00
Register Mnemonic: la_wrprio2
Short Description: Write PC/104 IRQ/DRQ select multiplexer for INT2 DSP input

Register Description: The Starter Kit hardware will map, under program control, the PC/104 DRQ and IRQ lines to three interrupt request lines going into the C52 DSP -- INT2, INT3, and INT 4. The Monitor program has a menu option to set up the mapping. Please use the Monitor menu option for more understanding.

Bit Number	Mnemonic	Read/Write	Description
0	I2SEL0	Wr	INT2 input select least significant bit
1	I2SEL1	Wr	
2	I2SEL2	Wr	
3	I2SEL3	Wr	INT2 input select most significant bit
4			Not Used
5			Not Used
6			Not Used
7			Not Used
8			Not Used
9			Not Used
10			Not Used
11			Not Used
12			Not Used
13			Not Used
14			Not Used
15			Not Used

12C01/11C01 IEEE 1394 Starter Kit Register
detail

Register Address: 0xFD01
Register Mnemonic: la_wrprio3
Short Description: Write PC/104 IRQ/DRQ select multiplexer for INT3 DSP input

Register Description: The Starter Kit hardware will map, under program control, the PC/104 DRQ and IRQ lines to three interrupt request lines going into the C52 DSP -- INT2, INT3, and INT 4. The Monitor program has a menu option to set up the mapping. Please use the Monitor menu option for more understanding.

Bit Number	Mnemonic	Read/Write	Description
0	I3SEL0	Wr	INT3 input select least significant bit
1	I3SEL1	Wr	
2	I3SEL2	Wr	
3	I3SEL3	Wr	INT3 input select most significant bit
4			Not Used
5			Not Used
6			Not Used
7			Not Used
8			Not Used
9			Not Used
10			Not Used
11			Not Used
12			Not Used
13			Not Used
14			Not Used
15			Not Used

Register Address: 0xFD02
Register Mnemonic: la_wrprio4
Short Description: Write PC/104 IRQ/DRQ select multiplexer for INT4 DSP input

Register Description: The Starter Kit hardware will map, under program control, the PC/104 DRQ and IRQ lines to three interrupt request lines going into the C52 DSP -- INT2, INT3, and INT 4. The Monitor program has a menu option to set up the mapping. Please use the Monitor menu option for more understanding.

Bit Number	Mnemonic	Read/Write	Description
0	I4SEL0	Wr	INT4 input select least significant bit
1	I4SEL1	Wr	
2	I4SEL2	Wr	
3	I4SEL3	Wr	INT4 input select most significant bit
4			Not Used
5			Not Used
6			Not Used
7			Not Used
8			Not Used
9			Not Used
10			Not Used
11			Not Used
12			Not Used
13			Not Used
14			Not Used
15			Not Used

12C01/11C01 IEEE 1394 Starter Kit Register detail

Register Address: 0xFE00
Register Mnemonic: la_ldpg
Short Description: Load PC/104 address register bits [23..16]

Register Description: The 24-bit PC/104 address is driven by two registers in the Starter Kit Hardware. A write to this address will load bits [23..16] into the PC/104 address register.

Bit Number	Mnemonic	Read/Write	Description
0	PA16	Wr	PC/104 Address bit [16]
1	PA17	Wr	PC/104 Address bit [17]
2	PA18	Wr	PC/104 Address bit [18]
3	PA19	Wr	PC/104 Address bit [19]
4	PA20	Wr	PC/104 Address bit [20]
5	PA21	Wr	PC/104 Address bit [21]
6	PA22	Wr	PC/104 Address bit [22]
7	PA23	Wr	PC/104 Address bit [23]
8			Not Used
9			Not Used
10			Not Used
11			Not Used
12			Not Used
13			Not Used
14			Not Used
15			Not Used

Register Address: 0xFF00
Register Mnemonic: la_lddack
Short Description: Load PC/104 DACK register

Register Description: The ISA function on the PC/104 bus includes a multi-channel DMA transfer capability which is managed by radially connected DRQ (DMA request) and DACK (DMA acknowledge) signals to DMA members on the ISA bus. This register drives the DACK lines. The corresponding DRQ lines is handled by la_wrprioX (X = 2,3,4)

Bit Number	Mnemonic	Read/Write	Description
0	DACK0	Wr	PC/104 DMA acknowledge 0
1	DACK1	Wr	PC/104 DMA acknowledge 1
2	DACK2	Wr	PC/104 DMA acknowledge 2
3	DACK3	Wr	PC/104 DMA acknowledge 3
4	DACK4	Wr	PC/104 DMA acknowledge 4
5	DACK5	Wr	PC/104 DMA acknowledge 5
6	DACK6	Wr	PC/104 DMA acknowledge 6
7	DACK7	Wr	PC/104 DMA acknowledge 7
8			Not Used
9			Not Used
10			Not Used
11			Not Used
12			Not Used
13			Not Used
14			Not Used
15			Not Used

12C01/11C01 Starter Kit Revised: September 9, 1996
Revision: 1

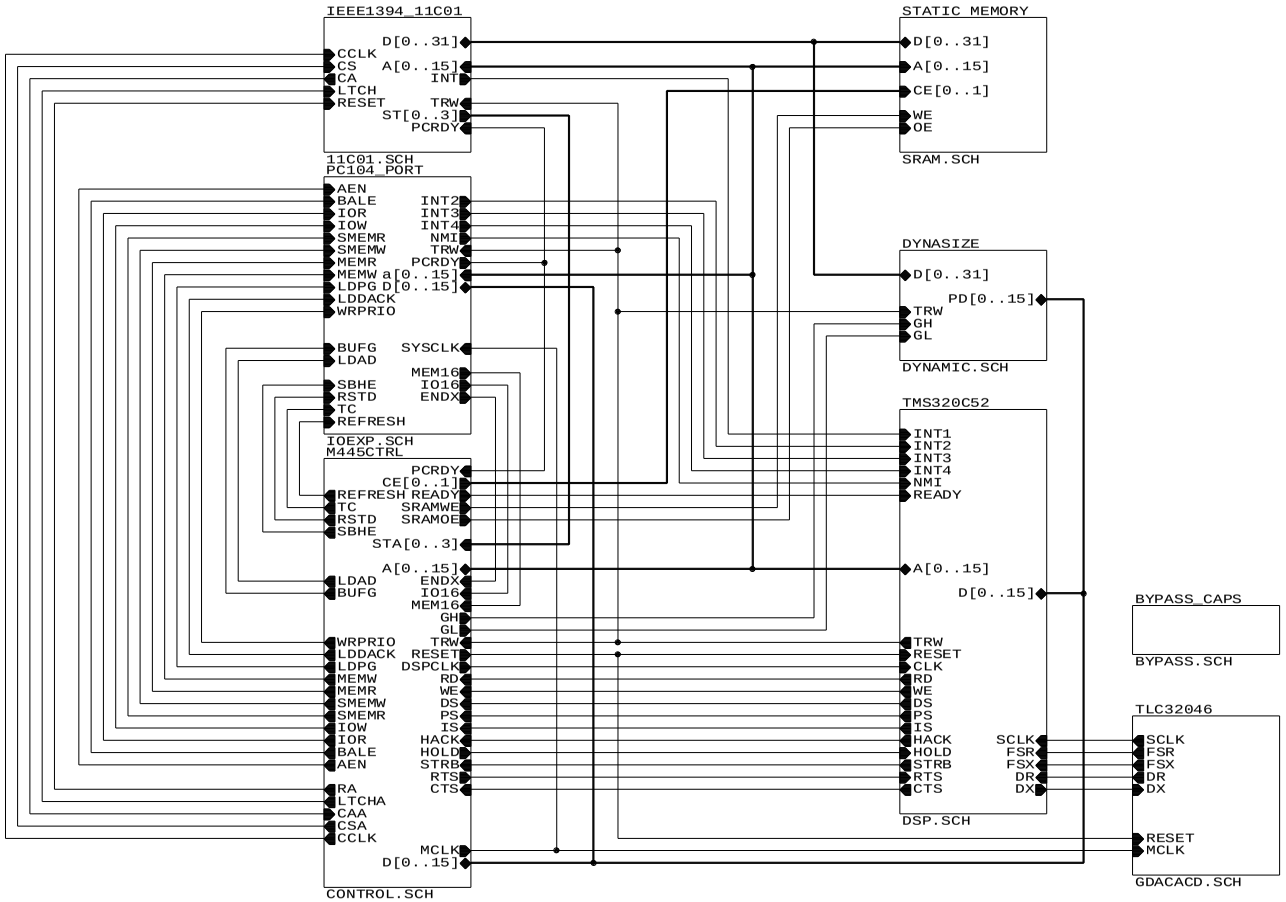
Texas Instruments

Designed ByTechnobox, Inc.
(609)-778-5512

Bill Of Materials September 9, 1996 11:07:42 Page 1

Item	Quantity	Reference	Part
1	3	C100,C102,C105	10uFD 6032 1021
2	1	C101	100uFD,16V ELECE 1541
3	1	C103	2.2uFD35V 6032 1629
4	1	C104	1.0uFD CC3216 1007
5	52	C200,C201,C202,C203,C204, C205,C206,C207,C208,C209, C210,C213,C214,C215,C217, C218,C219,C220,C221,C223, C224,C225,C227,C228,C231, C232,C233,C234,C235,C236, C237,C238,C239,C240,C241, C242,C243,C244,C245,C246, C247,C248,C249,C250,C251, C252,C253,C254,C255,C256, C257,C258	0.1uFD RC1206 1004
6	3	C211,C212,C216	220pFD RC1206 1039
7	1	C222	100pFD RC1206 1012
8	1	C226	0.01uFD RC1206 1003
9	2	C229,C230	22pFD RC1206 1047
10	2	D1,D2	1N5819 MELF 1566
11	3	D3,D4,D5	YLED SOT23 1471
12	8	FD1,FD2,FD3,FD4,FD5,FD6, FD7,FD8	FADUC60 FADUC60 1619
13	1	JPC1	PC104A PC104A 1721
14	1	JPC2	PC104B PC104B 1722
15	1	JP1	DISKPWR DISKPWR 1613
16	1	JP2	HEADER 12 HDR12X1 1611
17	2	JP12,JP3	HEADER 10X2 HDR10X2N 1347
18	3	JP4,JP5,JP9	MTG1394 1394HTHR 1716
19	3	JP6,JP7,JP8	CONN1394 1394HSMT 1708
20	1	JP10	HEADER 7X2M HDR7X2ME 1557
21	1	JP11	HEADER 5X2M HDR5X2ME 1498
22	2	J1,J2	RCA JACK RCAJACK 1715
23	1	L1	47uHY PM54 1547
24	4	P1,P2,P3,P4	TESTPT 1HOLE 1244
25	1	P5	CONNECTOR DB9 DB9FRAS 1724
26	1	Q1	LM317H T220BCER 1714
27	1	R100	330x8 SOIC16 1464
28	3	R200,R204,R205	220OHM RC1206 1038
29	1	R201	6.36K RC1206 1083

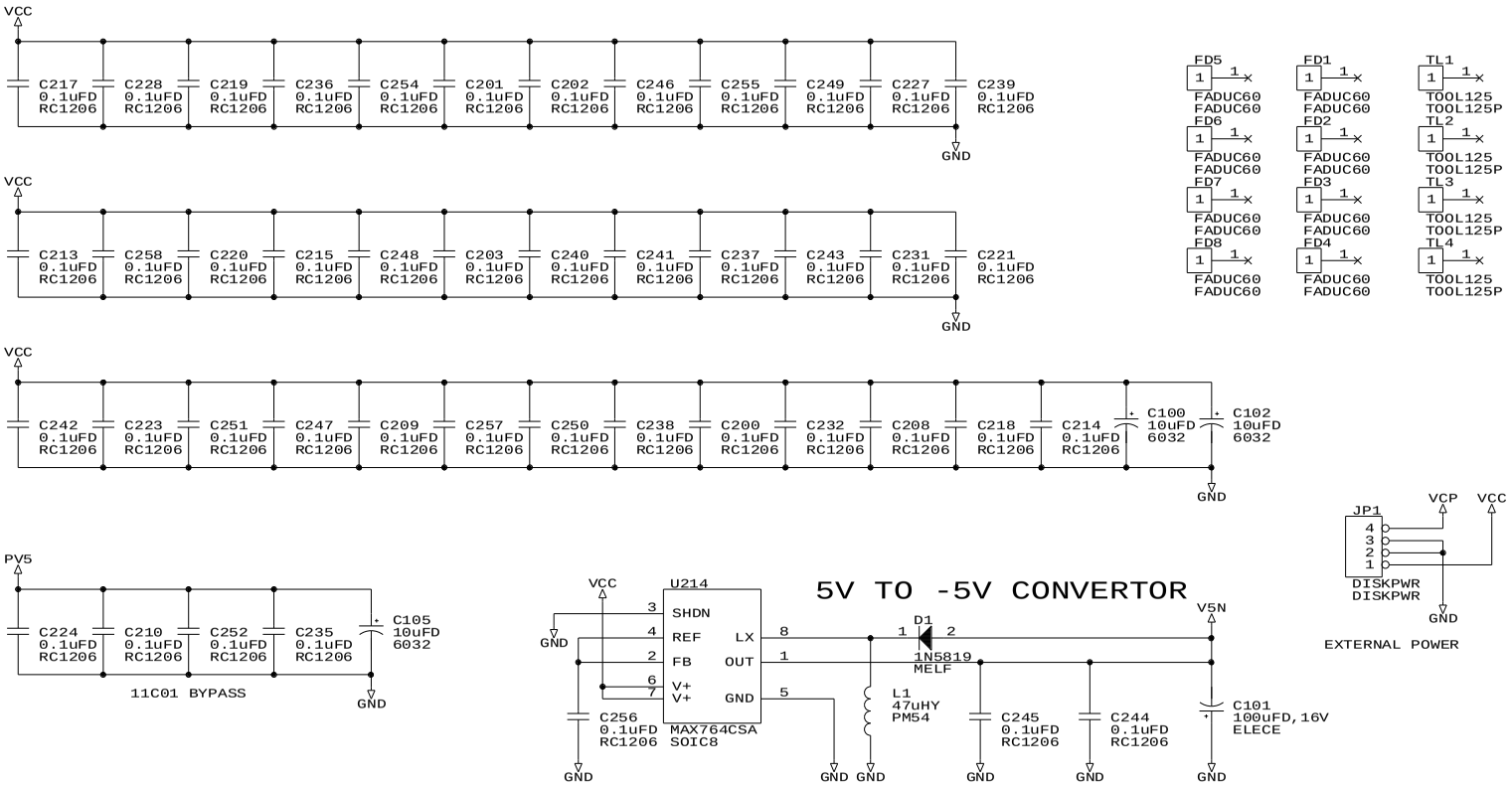
30	1	R202	5.1Kx8 SOIC16 1266
31	2	R203,R211	RP2.2K SOIC16 1034
32	1	R206	560OHM RC1206 1081
33	2	R213,R207	10K RC1206 1015
34	2	R208,R212	56x8 SOIC16 1267
35	1	R209	1M RC1206 1006
36	1	R210	96OHM RC1206 1129
37	3	R214,R215,R216	1K RC1206 1029
38	1	R217	330K RC1206 1061
39	1	R218	82.5_OHM RC1206 1626
40	1	R219	243_OHM RC1206 1627
41	1	S1	SW-SPST-TINYPB SWSPSTBP 1230
42	4	TL1,TL2,TL3,TL4	TOOL125 TOOL125P 1630
43	1	U100	33.3333MHz 8DIPOSC 1067
44	1	U101	M445STRT PQFP100 1535
45	1	U102	TLC32046 PLCC28 1536
46	6	U103,U106,U112,U201,U202, U203	74FCT2374 QSOP20 1719
47	1	U104	TMS320C52-PQFP PQFP100 1534
48	4	U105,U107,U111,U114	MT5C2568 SOJ28 1209
49	1	U108	TSB11C01 SSOP56 1254
50	6	U109,U204,U206,U207,U209, U213	74FCT2245 QSOP20 1718
51	1	U110	TSB12C01A TQFP100 1550
52	1	U113	14.31818MHz 8DIPOSC 1717
53	1	U200	MAX232 SOIC16 1197
54	1	U205	M210PRIO TQFP44M 1532
55	1	U208	4.7Kx15 SOIC16 1723
56	1	U210	74FCT2823 QSOP24 1720
57	1	U211	AM29F100B-PSOP PSOP44 1607
58	1	U212	MAX705 SOIC8 1304
59	1	U214	MAX764CSA SOIC8 1549
60	1	Y1	24.576MHz SMTXTL 1564



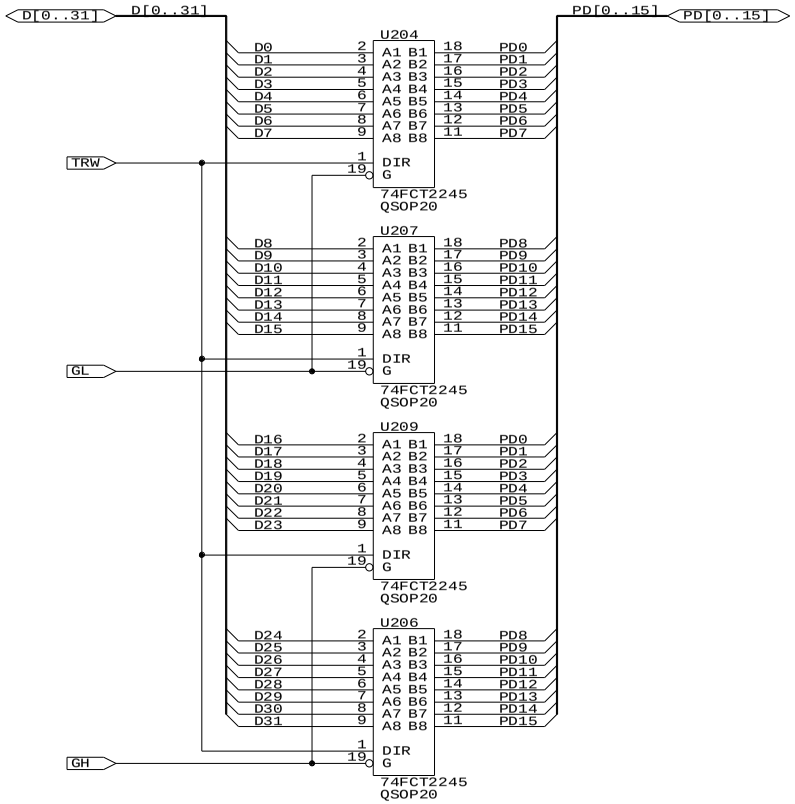
Texas Instruments

Designed By Technobox, Inc.
(609)-778-5512

Title		
12C01/11C01 Starter Kit		
Size: B	Document Number	REV: 1
Date: September 9, 1996	Sheet 1 of 9	



TEXAS INSTRUMENTS, INC DESIGNED BY TECHNOBOX, INC. (609)-778-5512		
Title IEEE 1394 Starter Kit - Bypass Caps		
Size B	Document Number STRT1394	REV 1
Date: July 31, 1996	Sheet 2 of 9	

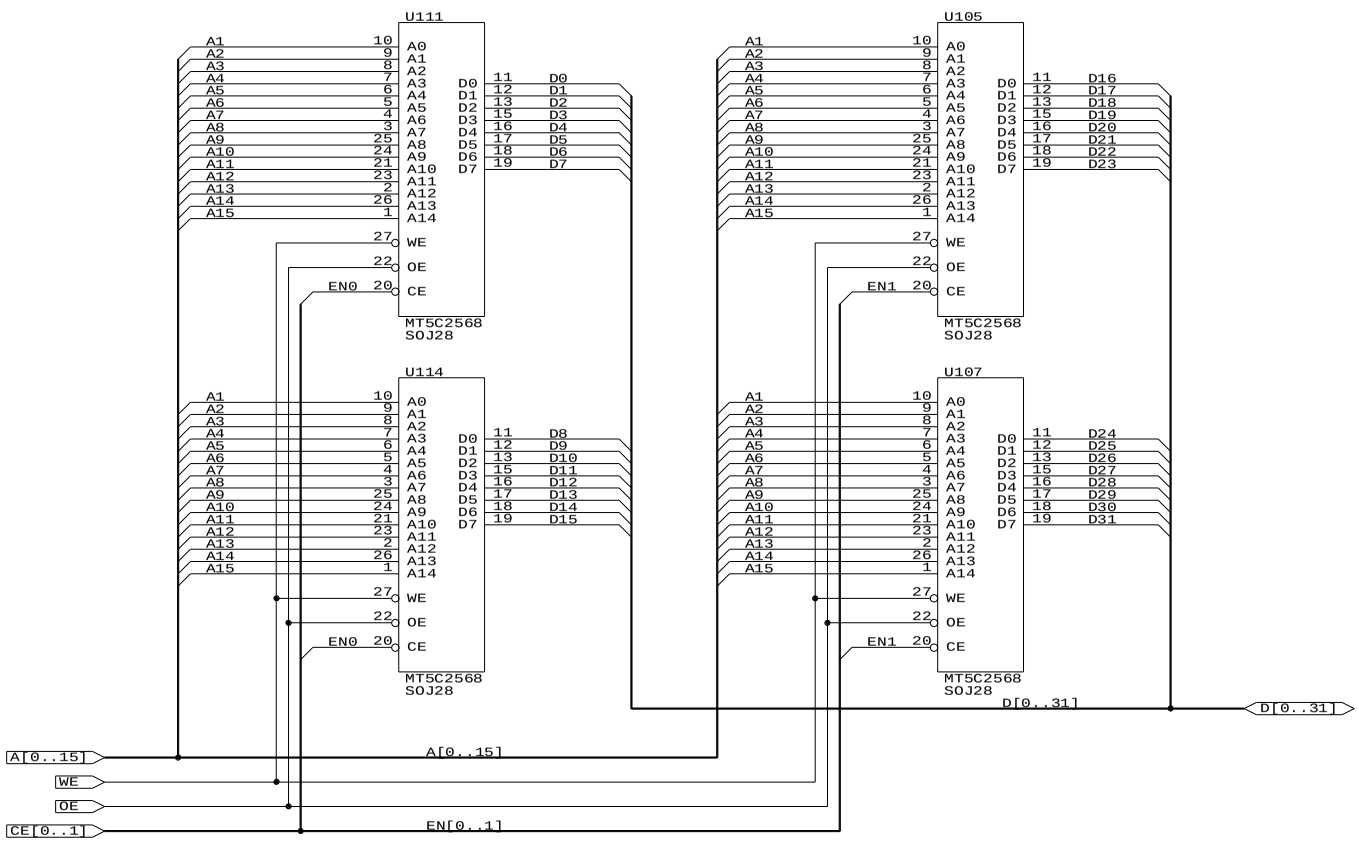


DYNAMIC BUS SIZER

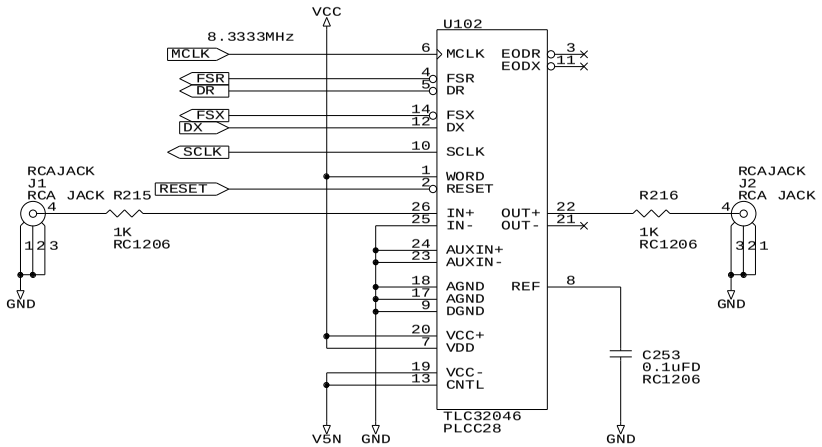
TEXAS INSTRUMENTS, INC

DESIGNED BY TECHNOBOX, INC.
(609)-778-5512

Title: IEEE 1394 Starter Kit - 2:4 Byte Mux		
Size: B	Document Number: STRT1394	REV: 1
Date: May 4, 1996	Sheet 3 of 9	

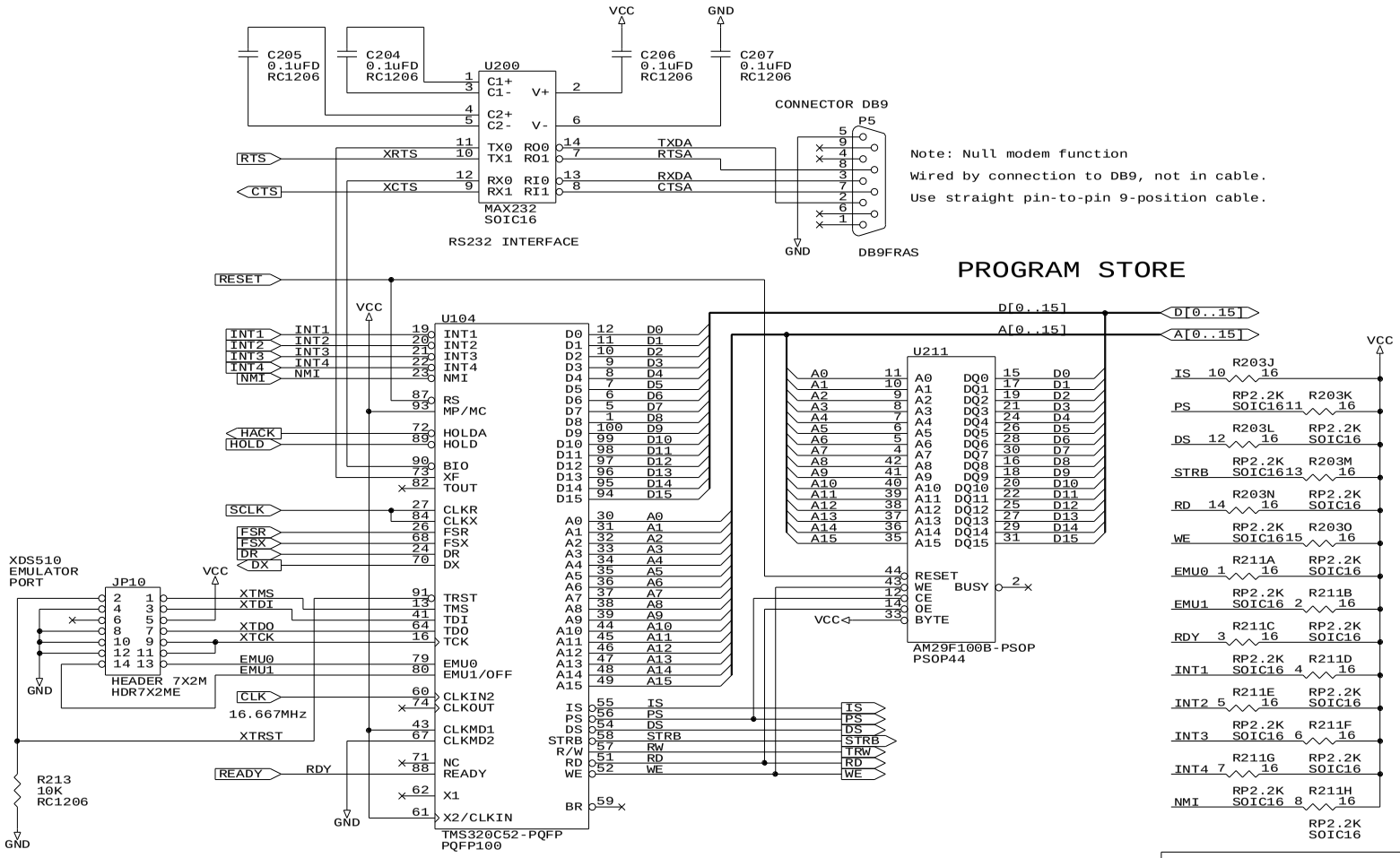


TEXAS INSTRUMENTS, INC DESIGNED BY TECHNOBOX, INC. (609)-778-5512		
Title IEEE 1394 Starter Kit - SRAM		
Size B	Document Number STRT1394	REV 1
Date: May 4, 1996	Sheet 4 of 9	

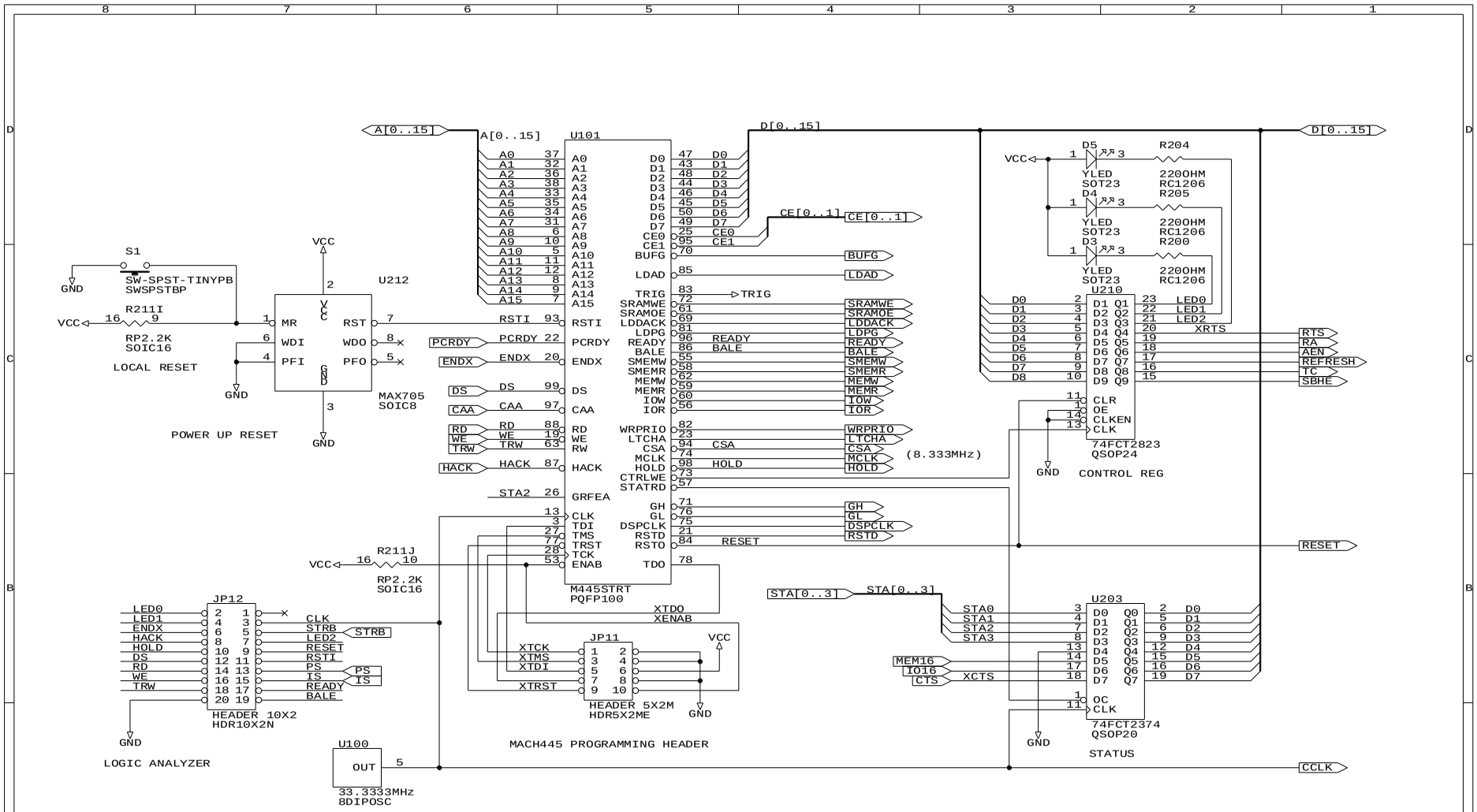


VOICE BAND ADC

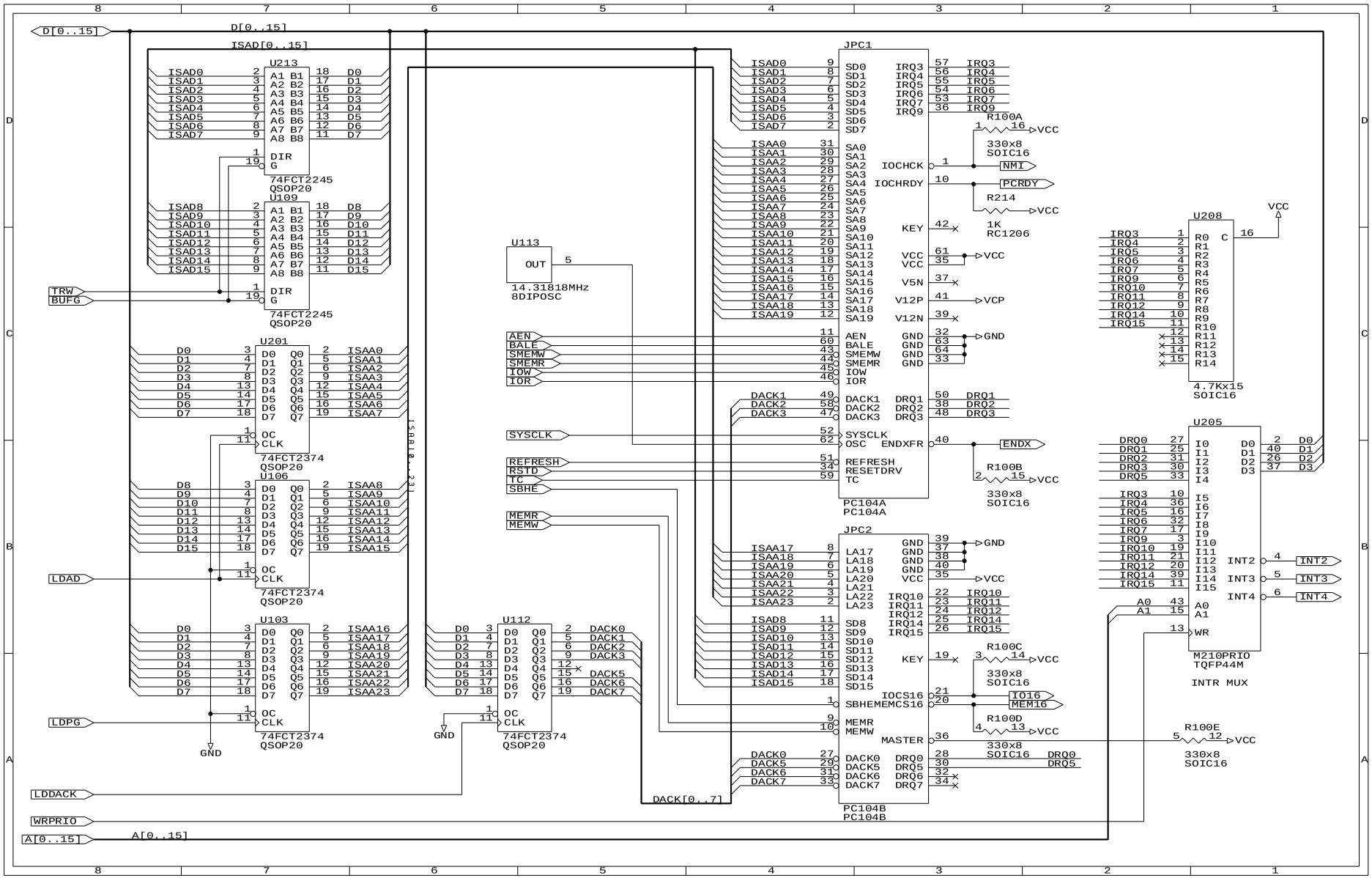
TEXAS INSTRUMENTS, INC DESIGNED BY TECHNOBOX, INC. (609)-778-5512		
Title IEEE 1394 Starter Kit - AIC		
Size B	Document Number STRT1394	REV 1
Date: May 4, 1996	Sheet 5 of 9	

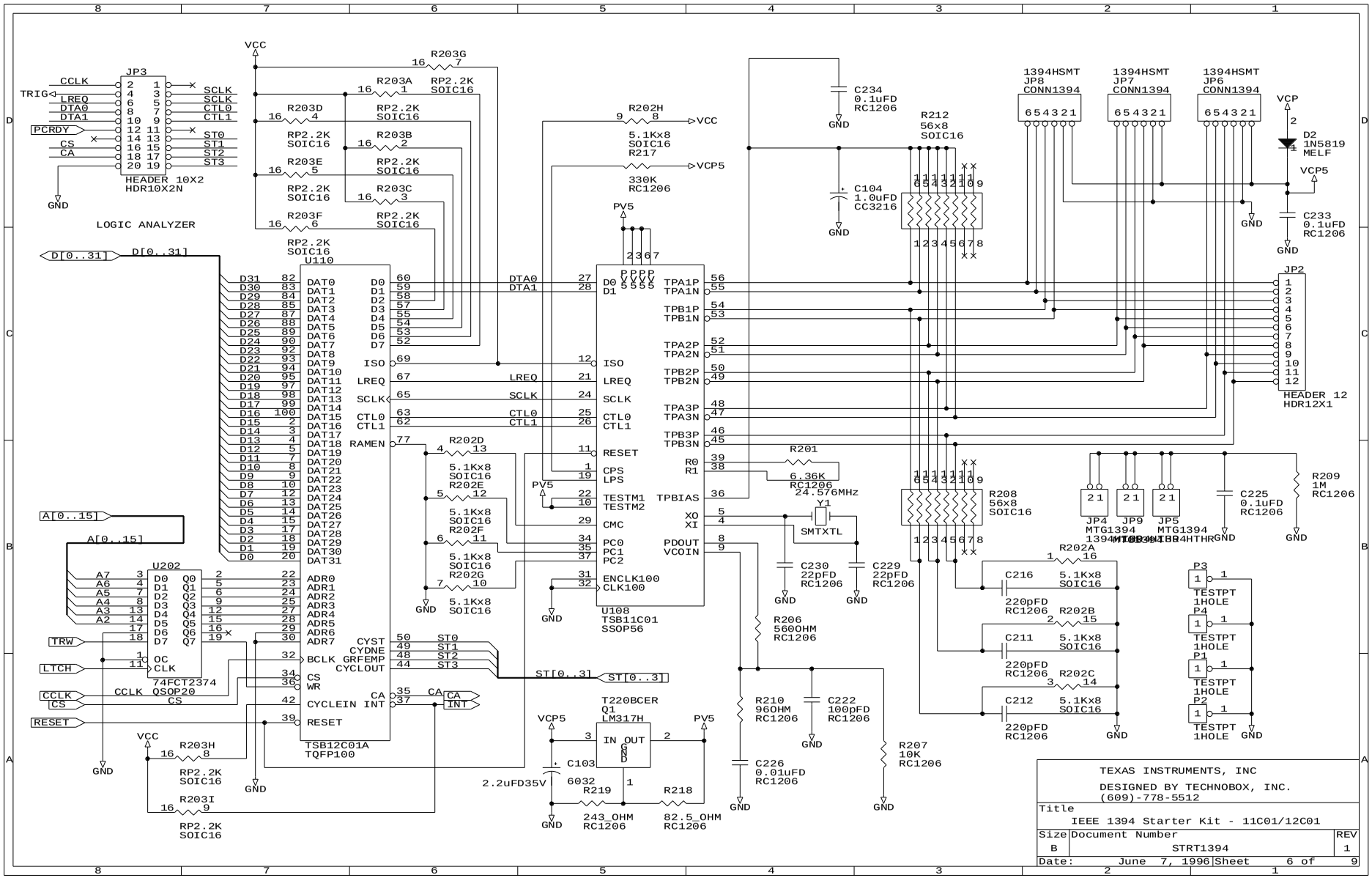


TEXAS INSTRUMENTS, INC DESIGNED BY TECHNOBOX, INC. (609)-778-5512		
Title IEEE 1394 Starter Kit - DSP/FLASH		
Size B	Document Number STRT1394	REV 1
Date: May 4, 1996	Sheet 7 of 9	



TEXAS INSTRUMENTS, INC DESIGNED BY TECHNOBOX, INC. (609)-778-5512		
Title IEEE 1394 Starter Kit - MACH 445		
Size B	Document Number STRT1394	REV 1
Date: July 8, 1996	Sheet 8 of 9	


D[0..15]
ISAD[0..15]
U213
74FCT2245
QSOP20
U109
74FCT2245
QSOP20
TRW
BUFG
U201
74FCT2374
QSOP20
U106
74FCT2374
QSOP20
LDAD
U103
74FCT2374
QSOP20
LDPG
GND
U113
OUT
14.31818MHz
8DIPOSC
AEN
BALE
SMEMW
SMEMR
IOW
IOR
SYSCLK
REFRESH
RSTD
TC
SBHE
MEMR
MEMW
U112
74FCT2374
QSOP20
GND
DACK[0..7]
LDDACK
WRPRIO
A[0..15]
JPC1
PC104A
PC104A
JPC2
PC104B
PC104B
R100A
330x8
SOIC16
NMI
PCRDY
R214
1K
RC1206
VCC
R100B
330x8
SOIC16
ENDX
R100C
330x8
SOIC16
IO16
MEM16
R100D
330x8
SOIC16
MASTER
R100E
330x8
SOIC16
DRQ0
DRQ5
U208
4.7Kx15
SOIC16
U205
M210PRIO
TQFP44M
INTR MUX
INT2
INT3
INT4


JP3
HEADER 10X2
HDR10X2N
LOGIC ANALYZER
U110
TSB12C01A
TQFP100
U202
74FCT2374
U108
TSB11C01
SSOP56
JP2
HEADER 12
HDR12X1
1394HSMT
JP8
CONN1394
JP7
JP6
JP4
MTG1394
JP9
JP5
Y1
24.576MHz
SMTXTL
Q1
LM317H
P3
P4
P1
P2
TESTPT
1HOLE
TEXAS INSTRUMENTS, INC
DESIGNED BY TECHNOBOX, INC.
(609)-778-5512
Title
IEEE 1394 Starter Kit - 11C01/12C01
Size
B
Document Number
STRT1394
REV
1
Date: June 7, 1996
Sheet 6 of 9

```
"
" 1394 Starter Kit MACH 445 interface design
"

#TITLE '1394 Starter Kit control PLD';
#ENGINEER 'JOE NORRIS';
#REVISION '1.0';
#COMPANY 'TECHNOBOX, INC.';
#PROJECT 'STRT1394';

"
" Revision history:
"
"       3/5/96: Initial coding as cloned fromBackplane PHY design
"       3/8/96: Added PC/104 ENDXFR* signal for early bus cycle termination
"       4/28/96: MCLK was too fast....corrected to 8.3333MHz
"       5/2/96: Changed SYNCWE generation to solve temperature related problem
"       9/7/96: Added support for MACH revision control
"
" Change mach revision code here
"

MACRO         mach_rev     01h;              "MACH revision code

"
" Define misc control
"

INPUT         clk;                           "clock INPUT

OUTPUT        dspclk CLOCKED_BYclk DEFAULT_TO 0;

              dspclk = /dspclk;               "DSP clock = 16.666 MHz

OUTPUT        mclk CLOCKED_BYclk DEFAULT_TO LAST_VALUE;

              IF dspclk THEN
                mclk = /mclk;                 "ISA/ADC = 8.333 MHz
              END IF;

LOW_TRUE INPUT reset_pin;                     "BOARD reset
NODE          reset CLOCKED_BYclk;
              reset = reset_pin;              "PROVIDE SYNC reset

LOW_TRUE OUTPUTrsto CLOCKED_BYclk DEFAULT_TO 0;          "Reset output
              rsto = reset;                   "Reset output follow reset input

OUTPUT        rstd CLOCKED_BYclk DEFAULT_TO 0;           "Inverted Reset (Active True)
              rstd = /rsto;

LOW_TRUE INPUT endx_pin;
NODE          endx CLOCKED_BYclk;             "PC/104 early termination signal
              endx = endx_pin;
```

```
LOW_TRUE INPUT  hack;                                   "DSPack's hold
LOW_TRUE OUTPUT hold CLOCKED_BYclk RESET_BY reset
                DEFAULT_TO LAST_VALUE;                  "Hold DSP bus

NODE            synchack CLOCKED_BYclk RESET_BY reset
                  DEFAULT_TO 0;
            synchack = hack;                            "Sync'd version of HACK from DSP

"
" DSP control signals
"

LOW_TRUE INPUT ds;
LOW_TRUE INPUT  rd;
LOW_TRUE INPUT  we;
INPUT           trw;

"
" Generate synchronized version of DSP we
"

NODE            syncwr1 CLOCKED_BY clk RESET_BY reset
                  DEFAULT_TO 0;
            syncwr1 = we AND ds;                        "Sync'd version of DSP we

NODE            syncwr2 CLOCKED_BY clk RESET_BY reset   "Modified 5/2/96 to invert clk
                  DEFAULT_TO 0;
            syncwr2 = syncwr1;

NODE            syncwe CLOCKED_BYclk RESET_BY reset
                  DEFAULT_TO 0;
            syncwe = syncwr1 AND /syncwr2;              "Generate 1 clock wide write

"
" Define DMA flags
"

NODE            flg_trw  CLOCKED_BYclk RESET_BY reset
                  DEFAULT_TO LAST_VALUE;        "DMA read/write mode
NODE            flg_timeout CLOCKED_BYclk RESET_BY reset
                  DEFAULT_TO LAST_VALUE;        "Timeout detected during DMA transfer
NODE            flg_busy CLOCKED_BYclk RESET_BY reset
                  DEFAULT_TO 0;                 "DMA is busy

"
" Define DSP data lines
"

PHYSICAL NODE   doe DEFAULT_TO 0;                   "OE for Data Bus

OUTPUT          d_pin[7..0] ENABLED_BY doe;         "DSP data bus
NODE            d[7..0] DEFAULT_TOmach_rev;
            d_pin = d;
```

```
"
" Define transfer counter
"

NODE          xfercntr[7..0] CLOCKED_BY clk RESET_BY reset
                    DEFAULT_TO LAST_VALUE;

PHYSICAL NODE  ldxfer DEFAULT_TO 0;
PHYSICAL NODE  decxfer DEFAULT_TO 0;

          IF ldxfer THEN
            xfercntr = d_pin;                    "Load transfer counter
          ELSIF decxfer THEN
            xfercntr = xfercntr .-. 1;           "Decrement xfer counter
          END IF;

PHYSICAL NODE  xferzero DEFAULT_TO 0;                    "Detect whenxfer cntr is zero

          IF xfercntr = 0 THEN
            xferzero = 1;
          END IF;

"
" Define DMA control register
"

NODE          dmactrl[3..0] CLOCKED_BY clk RESET_BY reset
                    DEFAULT_TO LAST_VALUE;

PHYSICAL NODE  lddmactrl DEFAULT_TO 0;

          IF lddmactrl THEN
            dmactrl[3..0] = d_pin[3..0];
          END IF;

NODE          grfrdyen DEFAULT_TO 0;          "Define GRF ready enable control bit
          grfrdyen = dmactrl[0];

NODE          pc104mode[1..0] DEFAULT_TO 0;
          pc104mode = dmactrl[2..1];          "PC104 I/O access when set

"
" Define PC104 operational modes
"

MACRO   pc104mode_smem   0;          "PC104 System Memory Access
MACRO   pc104mode_mem    1;          "Memory access
MACRO   pc104mode_io     2;          "I/O access

"
" Define delay before sampling IOCHRDY assertion
"

NODE          rdytimer[3..0] CLOCKED_BY clk RESET_BY reset
```

```
            DEFAULT_TO LAST_VALUE;

PHYSICAL NODE  ldrdytime DEFAULT_TO 0;

            IF ldrdytime AND syncwe THEN
              rdytimer[3..0] = d_pin[3..0];
            END IF;

"
" Define source of DSP data bus drive
"

PHYSICAL NODE  selxfer DEFAULT_TO 0;
PHYSICAL NODE  seldmactrl DEFAULT_TO 0;

            IF selxfer THEN
              d = xfercntr;                    "Select read xfer cntr
            ELSIF seldmactrl THEN
              d[3..0] = dmactrl[3..0];          "Read back DMA control register
              d[5] = flg_busy;                  "DMA is busy
              d[6] = flg_timeout;               "Timeout detected
              d[7] = flg_trw;                   "Transfer read/write flag
            END IF;

"
" Define timeout counter
"

NODE          timer[7..0] CLOCKED_BY clk RESET_BY reset
                  DEFAULT_TO LAST_VALUE;

PHYSICAL NODE  timer_rst DEFAULT_TO 0;
PHYSICAL NODE  timer_max DEFAULT_TO 0;
PHYSICAL NODE  timer_loadrdy DEFAULT_TO 0;

            IF timer = 0ffh THEN               "Detect when we havemax timer value
              timer_max = 1;
            END IF;

            IF timer_rst THEN
              timer = 0;                       "Clear timer
            ELSIF timer_loadrdy THEN
              timer[7..4] = 0fh;                "Load timer w/PC104 ready count value
              timer[3..0] = /rdytimer;
            ELSIF /timer_max THEN
              timer = timer .+. 1;              "Update timer if not max'd out
            END IF;

"
" Define DSP address lines
"

OUTPUT        a_pin[15..0] ENABLED_BY hack;          "DSP address
NODE          a[15..0] CLOCKED_BY clk RESET_BY reset      "Address counter
```

```
                DEFAULT_TO LAST_VALUE;
            a_pin = a;

PHYSICAL NODE  ldaddrhi DEFAULT_TO 0;                  "Reg load controls
PHYSICAL NODE  ldaddrlo DEFAULT_TO 0;
PHYSICAL NODE  incaddr DEFAULT_TO 0;                   "Increment address counter
PHYSICAL NODE  acy DEFAULT_TO 0;                       "Carry from address counter lsb's

            IF a[7..1] = 1111111b THEN
              acy = 1;
            END IF;

            IF ldaddrhi THEN
              a[15..8] = d_pin;                        "Load address high
            ELSIF incaddr AND acy THEN
              a[15..8] = a[15..8] .+. 1;               "Increment address counter
            END IF;

            IF ldaddrlo THEN
              a[7..0] = d_pin;                         "Load address low
            ELSIF incaddr THEN
              a[7..1] = a[7..1] .+. 1;                 "Increment address counter
            END IF;

"
" Define control outputs
"

LOW_TRUE OUTPUT trig DEFAULT_TO 0;                     "Logic analyzer trigger

LOW_TRUE OUTPUTstatrd DEFAULT_TO 0;                    "Read external status register
LOW_TRUE OUTPUTctrlwe DEFAULT_TO 0;                    "Write LED/Control register

LOW_TRUE OUTPUTsramwe DEFAULT_TO 0;                    "Write SRAM
LOW_TRUE OUTPUTsramoe DEFAULT_TO 0;                    "Read SRAM

NODE           sramwe_dsp DEFAULT_TO 0;
NODE           sramoe_dsp DEFAULT_TO 0;                "DSP access SRAM

PHYSICAL NODE  sramwe_dma CLOCKED_BY clk RESET_BY reset
                  DEFAULT_TO LAST_VALUE;
PHYSICAL NODE  sramoe_dma CLOCKED_BY clk RESET_BY reset
                  DEFAULT_TO LAST_VALUE;           "DMA access SRAM

            sramwe = sramwe_dma OR sramwe_dsp;
            sramoe = sramoe_dma OR sramoe_dsp;

LOW_TRUE OUTPUTce[1..0] DEFAULT_TO 0;                  "CE's to SRAM low/high word

PHYSICAL NODE  ce_dsp[1..0] DEFAULT_TO 0;

NODE           ce_dma[1..0] CLOCKED_BY clk RESET_BY reset
                  DEFAULT_TO LAST_VALUE;
```

```
        ce = ce_dsp OR ce_dma;

LOW_TRUE OUTPUT gh DEFAULT_TO 0;                    "Enable '245 dyna sizer hi word
LOW_TRUE OUTPUT gl DEFAULT_TO 0;                    "Enable '245 dyna sizer lo word

OUTPUT ltcha DEFAULT_TO 0;                          "Load 11C01 address latch

"
" Define synchronous handshakes with 12C01 link level controllers
"

LOW_TRUE INPUT caa;                                 "Transfer ACK from 11C01 link controller

LOW_TRUE OUTPUT csa CLOCKED_BY clk DEFAULT_TO 0;    "Transfer start to 11C01 link
controller

INPUT       grfea;                                  "11C01 GRF is empty

"
" Define control signals related to PC104 access
"

LOW_TRUE OUTPUT bufg  DEFAULT_TO 0;                 "'245 buffer control

LOW_TRUE OUTPUT smemw CLOCKED_BY clk RESET_BY /bufg
            DEFAULT_TO LAST_VALUE;                  "ISA bus SMEMW
LOW_TRUE OUTPUT smemr CLOCKED_BY clk RESET_BY /bufg
            DEFAULT_TO LAST_VALUE;                  "ISA bus SMEMR
LOW_TRUE OUTPUT memw CLOCKED_BY clk RESET_BY /bufg
            DEFAULT_TO LAST_VALUE;                  "ISA bus MEMW
LOW_TRUE OUTPUT memr CLOCKED_BY clk RESET_BY /bufg
            DEFAULT_TO LAST_VALUE;                  "ISA bus MEMR
LOW_TRUE OUTPUT iow  CLOCKED_BY clk RESET_BY /bufg
            DEFAULT_TO LAST_VALUE;                  "ISA bus IOW
LOW_TRUE OUTPUT ior  CLOCKED_BY clk RESET_BY /bufg
            DEFAULT_TO LAST_VALUE;                  "ISA bus IOR

"
" Ready control signalsassociatd with DSP and PC104
"

INPUT       pcrdy_pin;                              "PC104 Ready line
NODE        pcrdy CLOCKED_BY clk;
        pcrdy = pcrdy_pin;                          "PROVIDE SYNC ready

PHYSICAL NODE  rdyclk DEFAULT_TO 0;
PHYSICAL NODE  rdyset CLOCKED_BY clk RESET_BY reset
            DEFAULT_TO 0;
NODE        rdysetx DEFAULT_TO 0;
        rdysetx = reset OR rdyset;

OUTPUT      ready CLOCKED_BY rdyclk PRESET_BY rdysetx
        DEFAULT_TO 0;
```

```
"
" I/O register control strobes related to PC104 interface
"

OUTPUT              ldpg DEFAULT_TO 0;                  "Load address pagereg
LOW_TRUE OUTPUT     lddack DEFAULT_TO 0;                "Load DACK register
LOW_TRUE OUTPUT     wrprio DEFAULT_TO 0;                "Load MACH210 IRQ/DRQmux
LOW_TRUE OUTPUT     ldad DEFAULT_TO 0;                  "Load PC104 address hi
OUTPUT              bale  CLOCKED_BYclk RESET_BY reset
                    DEFAULT_TO 0;                       "ISA bus BALE

"
" DSP address map
"

MACRO   la_mach_rev     000h;           "Read Mach Revision Code
MACRO   la_11c01        002h;           "Load 11C01 chip address latch
MACRO   la_pc104adr     003h;           "Write PC104 address register
MACRO   la_trig         004h;           "Logic Analyzer Trigger
MACRO   la_ctrlstat     005h;           "Write control register/Read Statusreg
MACRO   la_rdytime      006h;           "PC104 sample ready delay
MACRO   la_xfercntr     007h;           "Read/write transfer counter
MACRO   la_dmago        008h;           "Write here kicks off DMA transfer
MACRO   la_addrhi       009h;           "Load address counter hi
MACRO   la_addrlo       00Ah;           "Load address counter low
MACRO   la_pc104dat     00Bh;           "Read/Write PC104 space
MACRO   la_dmactrl      00Ch;           "DMA control register (Contains GRFRDYEN)

MACRO   la_wrprio       00Dh;           "Write PC/104 IRQ/DRQ selectmux
MACRO   la_ldpg         00Eh;           "Load PC/104 Page Register
MACRO   la_lddack       00Fh;           "Load PC/104 Address Register

MACRO   la_spareF0      000h;           "Spare decodes
MACRO   la_spareF1      001h;

"
" Mirror 'trw' signal captured in 12C01 address latch
"

PHYSICAL NODE   trwclk DEFAULT_TO 0;

NODE            trw_11c01 CLOCKED_BYtrwclk RESET_BY reset
                  DEFAULT_TO LAST_VALUE;
            trw_11c01 = trw;

"
" I/O decode
"

PHYSICAL NODEdlyrst DEFAULT_TO 0;
      dlyrst = rd OR we;

NODE    dlystrobe CLOCKED_BYclk RESET_BY /dlyrst DEFAULT_TO LAST_VALUE;
      dlystrobe = dlyrst;
```

```
NODE    startdma CLOCKED_BY clk RESET_BY reset "Start DMA transfers
          DEFAULT_TO 0;

NODE    startpc104 CLOCKED_BY clk RESET_BY reset
            DEFAULT_TO 0;           "Start PC104 transfer

PHYSICAL NODE pagefx DEFAULT_TO 0;

 IF (a_pin[15..12] = 0fh) THEN
  pagefx = 1;
 END IF;

  IF pagefx THEN
   IF (ds AND rd) THEN
    CASE a_pin[11..8]                   "Decode address
     WHEN la_mach_rev =>
      doe = 1;                      "Read Mach Revision Code
     WHEN la_11c01 =>
      ltcha = dlystrobe;              "Load 11C01 address latch
      trwclk = dlystrobe;
     WHEN la_ctrlstat =>
      statrd = 1;                   "Read status register
     WHEN la_xfercntr =>
      doe = 1;                      "Drive data bus
      selxfer = 1;                   "Select transfer counter for read
     WHEN la_dmactrl =>                 "DMA control register access
      seldmactrl = 1;                 "Select DMA control reg
      doe = 1;                      "Drive data bus
     WHEN la_pc104dat =>
      bufg = 1;                     "Enable PC104 '245 buffer
      rdyclk = 1;                    "Put DSP in wait state;
      startpc104 = 1;                 "Kick off PC104 transfer state machine
    END CASE;
   ELSIF (ds AND we) THEN
    CASE a_pin[11..8]                   "Decode address
     WHEN la_11c01 =>
      ltcha = dlystrobe;              "Load 11C01 address latch
      trwclk = dlystrobe;
     WHEN la_trig => trig = 1;          "Logic analyzer trigger
     WHEN la_ctrlstat =>
      ctrlwe = 1;                   "Write control register
     WHEN la_xfercntr =>
      IF syncwe THEN
       ldxfer = 1;                  "Load transfer counter
      END IF;
     WHEN la_addrhi =>
      IF syncwe THEN
       ldaddrhi = 1;                "Load address counter hi
      END IF;
     WHEN la_addrlo =>
      IF syncwe THEN
       ldaddrlo = 1;                "Load address counter low
      END IF;
```

```
        WHEN la_dmago =>
          IF syncwe THEN
            startdma = 1;                 "Kick off DMA transfer
          END IF;
        WHEN la_dmactrl =>                "DMA control register access
          IF syncwe THEN
            lddmactrl = 1;                "Load DMA control register
          END IF;
        WHEN la_ldpg =>
          IF syncwe THEN
            ldpg = 1;                     "Load PC104 address page register
          END IF;
        WHEN la_lddack =>
          lddack = 1;                     "Load PC104 DACK register
        WHEN la_wrprio =>
          wrprio = 1;                     "Load MACH210 IRQ/DRQ mux select register
        WHEN la_pc104adr =>
          ldad = 1;                       "Load PC104 addr reg w/address stepping
        WHEN la_rdytime =>
          IF syncwe THEN
            ldrdytime = 1;                "Load PC104 ready time
          END IF;
        WHEN la_pc104dat =>
          bufg = 1;                       "Enable PC104 '245 buffer
          rdyclk = 1;                     "Put DSP in wait state;
          startpc104 = 1;                 "Kick off PC104 transfer state machine
      END CASE;
    END IF;
  ELSIF (ds AND (rd OR we)) THEN
    IF trw THEN
      ce_dsp[1..0] = 11b;                 "Enable read from upper and lower SRAM words
      sramoe_dsp = 1;
      IF a_pin[0] THEN
        gh = 1;                           "Hi word read
      ELSE
        gl = 1;                           "Lo word read
      END IF;
    ELSE
      gh = 1;
      gl = 1;                             "If writing, enable both hi and low words
      sramwe_dsp = 1;                     "Write SRAM
      IF a_pin[0] THEN                    "Select upper or lower word to write
        ce_dsp[1] = 1;
      ELSE
        ce_dsp[0] = 1;
      END IF;
    END IF;
  END IF;

"
" State machine which handles DMA transfers with 12C01 link chips
"

NODE      dmastate[3..0] CLOCKED_BY clk;
```

```
STATE_MACHINE VXXXISTA CLOCKED_BY clk RESET_BY reset STATE_VALUES
GRAY_CODE STATE_BITS dmastate;

"
" Wait for start DMA indication from DSP
"

  STATE x_idle:
    IF startpc104 THEN
      IF trw THEN
        CASE pc104mode                    "PC104 Reads
          WHEN pc104mode_smem => smemr = 1;
          WHEN pc104mode_mem => memr = 1;
          WHEN pc104mode_io => ior = 1;
        END CASE;
      ELSE
        CASE pc104mode                    "PC104 writes
          WHEN pc104mode_smem => smemw =1;
          WHEN pc104mode_mem => memw = 1;
          WHEN pc104mode_io => iow = 1;
        END CASE;
      END IF;
      timer_loadrdy = 1;                  "Setup ready timeout counter
      GOTO x_pc104;                       "Handle PC104 cycle
    ELSIF /startdma THEN
      GOTO x_idle;                        "Stay here if no start yet
    ELSE
      flg_busy = 1;                       "Indicate DMA channel busy
      flg_timeout = 0;                    "Clear timeout error flag
      hold = 1;                           "Initiate bus request to DSP
      GOTO x_wait_hack;
    END IF;

  STATE x_wait_hack:                      "Wait for HACK from DSP
    IF /synchack THEN
      GOTO x_wait_hack;                   "Keep waiting if no hack yet
    ELSE
      GOTO x_dma;                         "Else, go on to perform DMA transfer
    END IF;

"
" DSP relinquished bus...perform transfer between selected LINK controller and SRAM
"

  STATE x_dma:
    ce_dma[1..0] = 11b;                   "Enable SRAM
    flg_trw = trw_11c01;
    GOTO x_dma0;

  STATE x_dma0:
    timer_rst = 1;                        "Reset timeout timer
    IF flg_trw THEN
      GOTO x_dmard;                       "Handle LINK -> SRAM transfers
```

```
   ELSE
    sramoe_dma = 1;                    "Turn on SRAM output enable
    GOTO x_dmawr;                      "Handle SRAM -> LINK transfers
   END IF;

"
" Write LINK data to SRAM
"

  STATE x_dmard:                       "Wait for appropriate GRF empty flag no longer empty
   IF xferzero THEN
    GOTO x_dma_done;
   ELSIF timer_max THEN
    flg_timeout = 1;                   "Indicate timeout detected
    GOTO x_dma_done;
   ELSE
    sramwe_dma = 1;                    "Turn on write to SRAM
    IF grfea AND grfrdyen THEN
     GOTO x_dmard;
    ELSE
     csa = 1;                          "Start transfer with 11C01
     GOTO x_dmard0;
    END IF;
   END IF;

  STATE x_dmard0:
   IF /caa THEN
    GOTO x_dmard0;                     "Stay here if no ack yet
   ELSE
    sramwe_dma = 0;                    "De-assert write to complete transfer
    GOTO x_dmard1;
   END IF;

  STATE x_dmard1:
   decxfer = 1;
   incaddr = 1;                        "Update transfer and address counters
   timer_rst = 1;
   GOTO x_dmard;                       "Loop back for next transfer

"
" Write SRAM data to link controller
"

  STATE x_dmawr:                       "Wait for appropriate GRF empty flag no longer empty
   IF xferzero THEN
    GOTO x_dma_done;
   ELSE
    csa = 1;                           "Start transfer with 11C01
    GOTO x_dmawr0;
   END IF;

  STATE x_dmawr0:
   IF /caa THEN                        "Do we have ack from 12C01?
    IF timer_max THEN
```

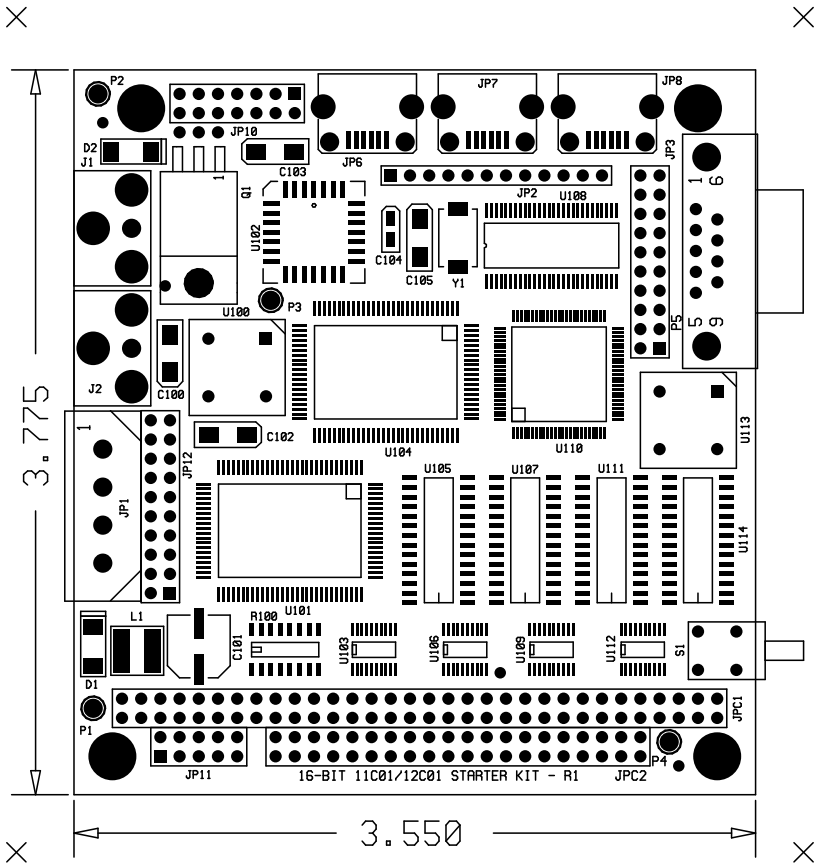
```
      flg_timeout = 1;
      GOTO x_dma_done;                    "Timeout waiting for ack
    ELSE
      GOTO x_dmawr0;                      "Stay here if no ack yet
    END IF;
  ELSE
    decxfer = 1;
    incaddr = 1;                          "Update transfer and address counters
    timer_rst = 1;
    GOTO x_dmawr;                         "Loop back for next transfer
  END IF;

"
" Completed DMA transfer....de-assert HOLD and wait for hack de-assertion
"

 STATE x_dma_done:
  ce_dma = 0;                             "Turn off SRAM access
  sramoe_dma = 0;
  sramwe_dma = 0;
  hold = 0;                               "Deassert HOLD to DSP
  flg_busy = 0;                           "DMA no longer busy
  GOTO x_dma_done0;

 STATE x_dma_done0:
  IF synchack THEN
    GOTO x_dma_done0;                     "Stay here if still hack'd
  ELSE
    GOTO x_idle;                          "No more hack....go back to idle state
  END IF;

"
" States to handle PC104 access...
"

 STATE x_pc104:
  IF timer_max OR (endx AND pcrdy) THEN       "Ready timer max'd out or ENDXFR?
    smemw = 0; memw = 0; iow = 0;             "In case of write to PC104
    rdyset = 1;
    GOTO x_pc104cont;
  ELSIF /pcrdy THEN                           "PC104 in wait condition?
    GOTO x_pc104wait;                         "Go wait if so
  ELSE
    GOTO x_pc104;                             "Otherwise, stay here
  END IF;

"
" PC104 is requesting wait states
"

 STATE x_pc104wait:
  IF pcrdy THEN
    smemw = 0; memw = 0; iow = 0;             "In case of write to PC104
    rdyset = 1;
```

```
    GOTO x_pc104cont;
  ELSE
    GOTO x_pc104wait;
  END IF;

"
" Finish off PC104 bus cycle
"

 STATE x_pc104cont:
  IF startpc104 THEN
    GOTO x_pc104cont;                  "Stay here until DSP cycle done
  ELSE
    GOTO x_idle;                       "Else, back to idle state
  END IF;

"
" END OF STATE MACHINE
"

END VXXXISTA;

"
" Minor state machine to handle assertion of BALE
"
STATE_MACHINE balemach CLOCKED_BY clk RESET_BY reset;

 STATE x_idle:
  IF ldpg THEN              "Look for page register load
    bale = 1;
    GOTO x_bale0;
  ELSE
    GOTO x_idle;
  END IF;

 STATE x_bale0:              "Keep BALE asserted for 2 clocks
  bale = 1;
  GOTO x_bale1;

 STATE x_bale1:
  bale = 0;
  GOTO x_idle;

END balemach;

"
" END OF FILE
"
```

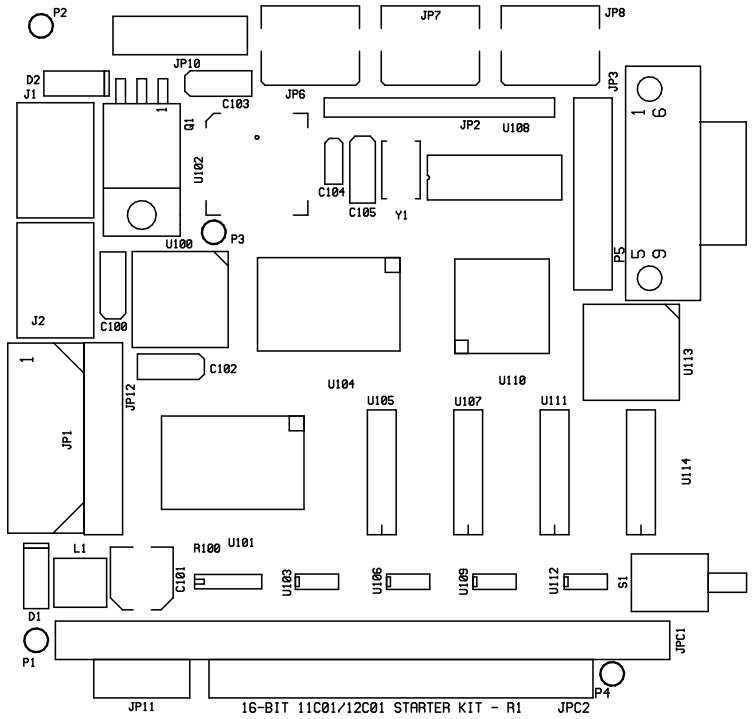
```
#TITLE 'PC104 Interface DMA/IRQ to DSP INTmux';
#ENGINEER 'JOE NORRIS';
#REVISION '1.0';
#COMPANY 'TECHNOBOX INC.';
#PROJECT 'M210PRIO';
#COMMENT 'Containsmux for IRQ, DRQ';

"
" Revision History
"

"
" Define multiplexer function
"

FUNCTION mux(in[7..0],sel[2..0])[1];

  CASE sel
    WHEN 0 => RETURN in[0];
    WHEN 1 => RETURN in[1];
    WHEN 2 => RETURN in[2];
    WHEN 3 => RETURN in[3];
    WHEN 4 => RETURN in[4];
    WHEN 5 => RETURN in[5];
    WHEN 6 => RETURN in[6];
    WHEN 7 => RETURN in[7];
  END CASE;
END mux;

"
" Define I/O
"

INPUT          wr;                          "DSP writes Priority Enc

INPUT          irqdrq[15..0];                   "irq/drq inputs
```

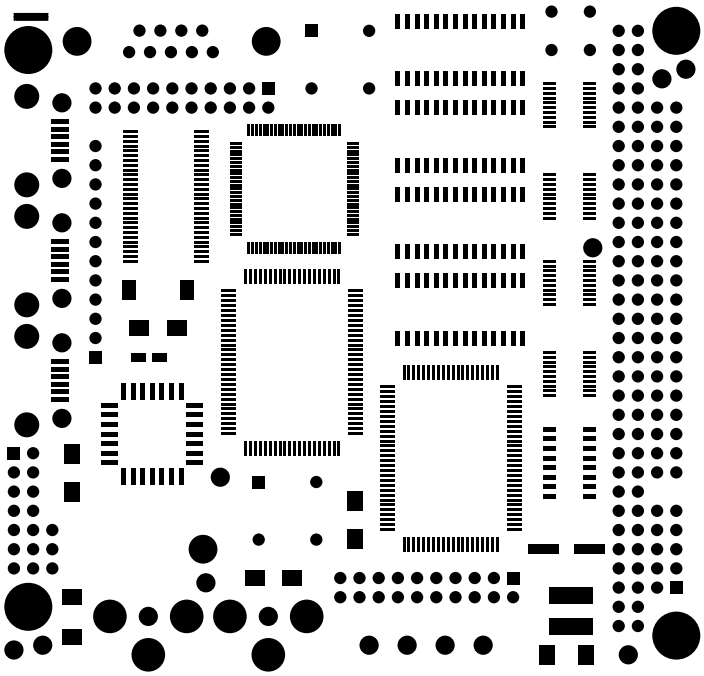
```
INPUT          d[3..0];                    "DSP data lines

INPUT          a[1..0];                    "DSP address lines

LOW_TRUE OUTPUT int2 DEFAULT_TO 0;
LOW_TRUE OUTPUT int3 DEFAULT_TO 0;
LOW_TRUE OUTPUT int4 DEFAULT_TO 0;              "Int requests to DSP

"
" Define mux select registers
"

NODE int2sel[3..0] CLOCKED_BYwr DEFAULT_TO LAST_VALUE;
NODE int3sel[3..0] CLOCKED_BYwr DEFAULT_TO LAST_VALUE;
NODE int4sel[3..0] CLOCKED_BYwr DEFAULT_TO LAST_VALUE;

  CASE a
    WHEN 0 => int2sel = d;
    WHEN 1 => int3sel = d;
    WHEN 2 => int4sel = d;
  END CASE;

"
" Create interrupt select multiplexers
"

PHYSICAL NODE int2a DEFAULT_TO 0;
PHYSICAL NODE int2b DEFAULT_TO 0;
PHYSICAL NODE int3a DEFAULT_TO 0;
PHYSICAL NODE int3b DEFAULT_TO 0;
PHYSICAL NODE int4a DEFAULT_TO 0;
PHYSICAL NODE int4b DEFAULT_TO 0;

  int2a = mux(irqdrq[7..0],int2sel[2..0]);
  int3a = mux(irqdrq[7..0],int3sel[2..0]);
```

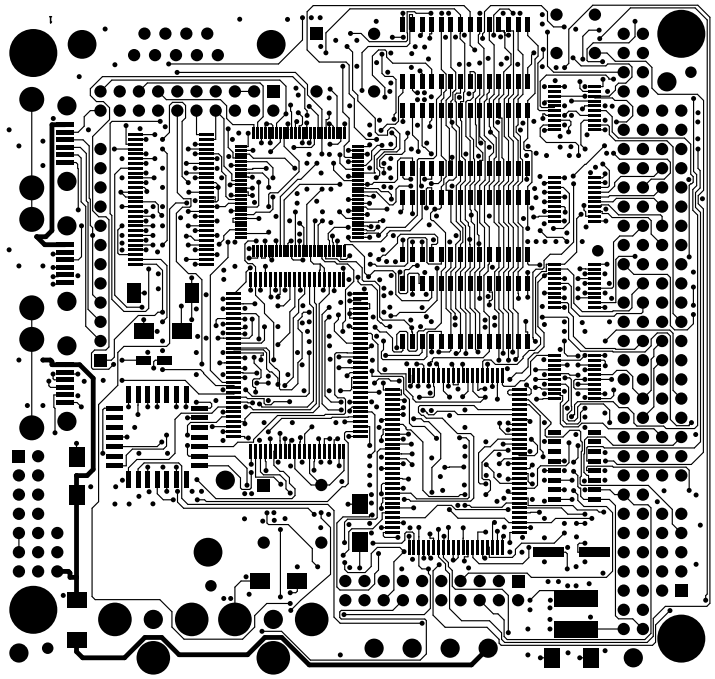
```
  int4a = mux(irqdrq[7..0],int4sel[2..0]);

  int2b = mux(irqdrq[15..8],int2sel[2..0]);
  int3b = mux(irqdrq[15..8],int3sel[2..0]);
  int4b = mux(irqdrq[15..8],int4sel[2..0]);

  IF /int2sel[3]
    THEN int2 = int2a;
    ELSE int2 = int2b;
  END IF;

  IF /int3sel[3]
    THEN int3 = int3a;
    ELSE int3 = int3b;
  END IF;

  IF /int4sel[3]
    THEN int4 = int4a;
    ELSE int4 = int4b;
  END IF;

"
" End of file
"
```

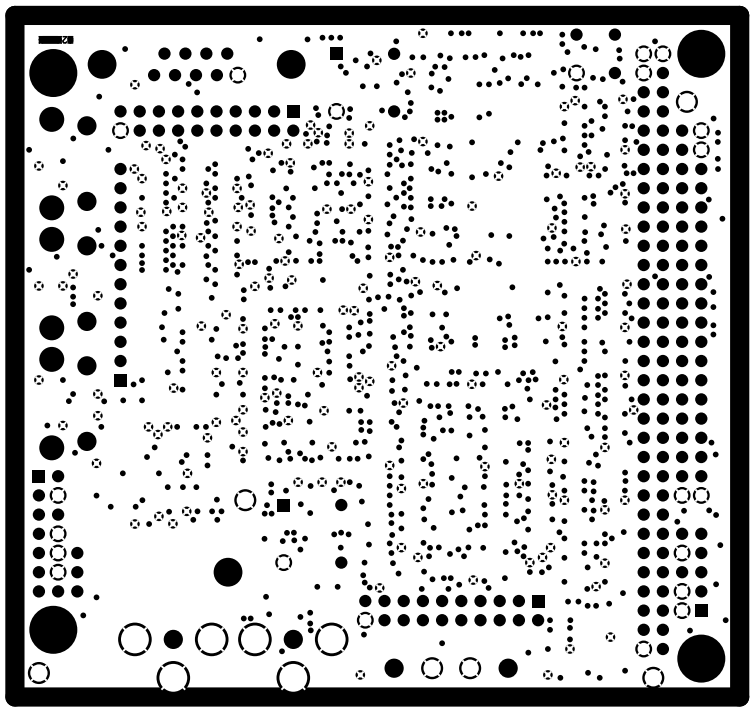

3.775
3.550
P2
JP7
JP8
JP10
D2
J1
JP6
JP3
C103
Q1
JP2
U108
U102
C104
C105
Y1
P3
U100
P5
J2
C100
C102
U104
U110
U113
JP12
JP1
U105
U107
U111
U114
L1
R100
U101
C101
U103
U106
U109
U112
S1
D1
JPC1
P1
P4
JP11
16-BIT 11C01/12C01 STARTER KIT - R1
JPC2

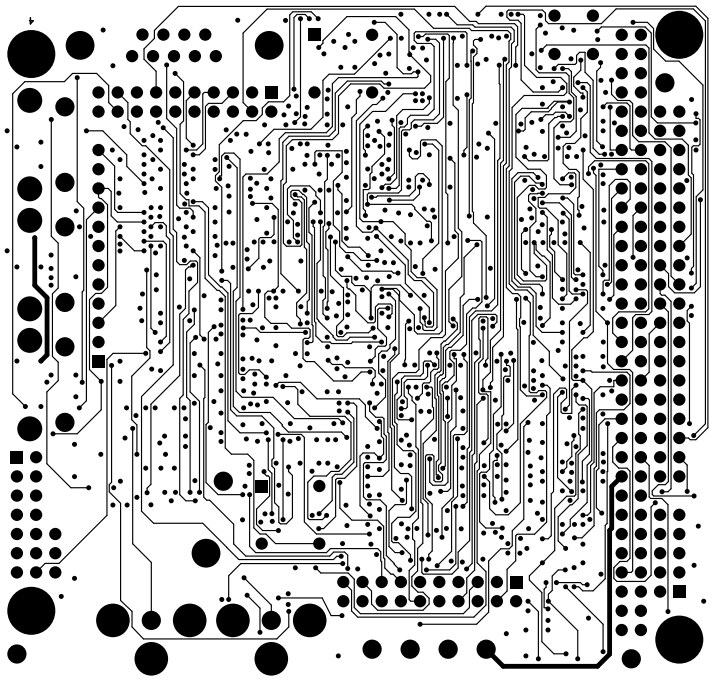
3.775
3.550

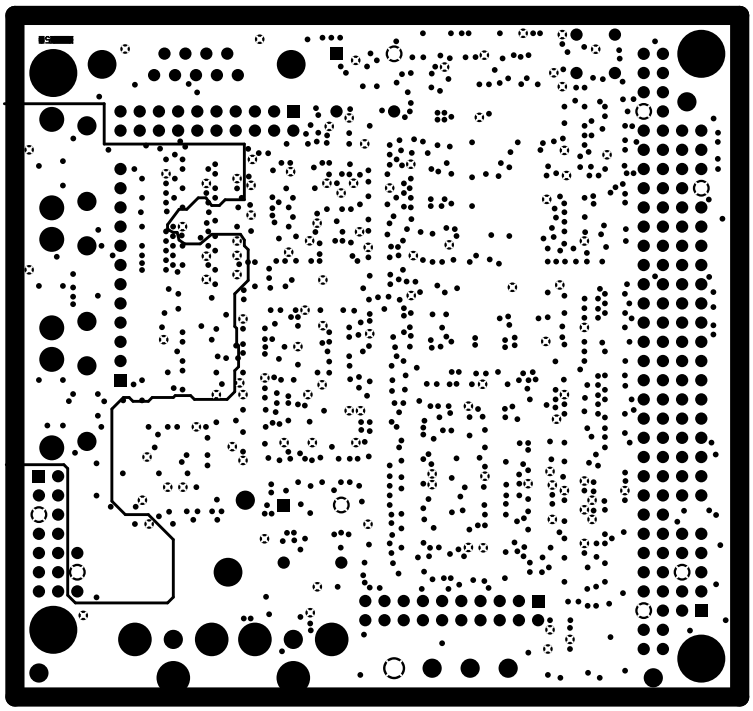

P2
JP10
JP7
JP8
D2
J1
C103
JP6
JP3
Q1
JP2
U108
U102
C104
C105
Y1
P3
U100
P5
J2
C100
C102
U104
U110
U113
JP12
JP1
U105
U107
U111
U114
L1
R100
U101
C101
U103
U106
U109
U112
S1
D1
JPC1
P1
P4
JP11
16-BIT 11C01/12C01 STARTER KIT - R1
JPC2

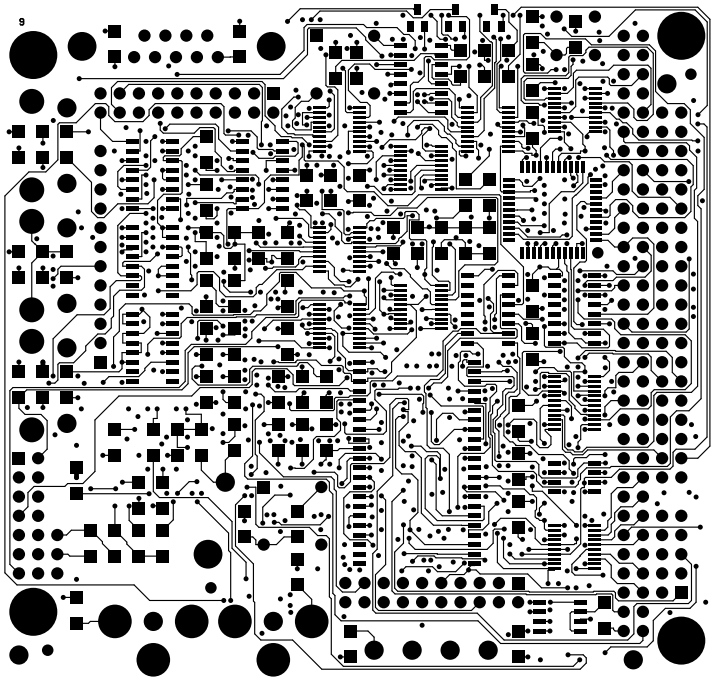


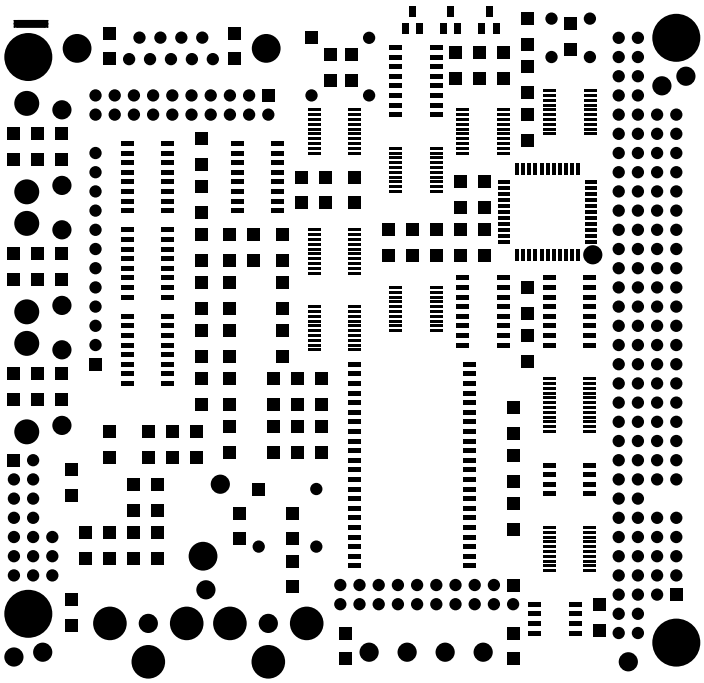









9



CS01
CS02
CS03
CS04
CS05
CS06
CS07
CS08
CS09
CS10
CS11
CS12
CS13
CS14
CS15
CS16
CS17
CS18
CS19
CS20
CS21
CS22
CS23
CS24
CS25
CS26
CS27
CS28
CS29
CS30
CS31
CS32
CS33
CS34
CS35
CS36
CS37
CS38
CS39
CS40
CS41
CS42
CS43
CS44
CS45
CS46
CS47
CS48
D1
D2
D3
D4
R01
R02
R03
R04
R05
R06
R07
R08
R09
R10
R11
R12
R13
R14
R15
R16
R17
R18
R19
R20
R21
R22
R23
R24
R25
R26
R27
R28
R29
R30
R31
R32
R33
R34
R35
R36
R37
R38
R39
R40

Printed Circuit Specification - "Starter Kit" board Rev 1 (6/14/96)

1. Manufacture QTY 85 with 4 week turn
2. Board outline: one rectangular board, 3.775" x 3.550"
3. Material: FR4,Tg 140 to 160 degrees Centigrade
4. 6-Layer board (4 signal, 2 planes) in order: S/P/S/S/P/S
5. Drill file: Excellon format.
6. Finished plated hole size tolerance: +/- 0.003". Non-plated: +/-0.002"
7. Copper weight: per vendor's standard 6-layer board process.
8. Overall board thickness: 0.062" nom. Layer thickness to be specified by Technobox.
9. Soldermask: ***Via holes are to be plugged with solder mask per vendor's recommended process.*** Then, LPI mask on primary and secondary sides is to be applied. ***Exposure of any copper at via pads on either side of the board is cause for rejection***. A separate 'via-only' gerber file ("viaonly.lgx") is supplied to locate via holes for this purpose.
10. Soldermask: both sides, ***to cover via pads. Any copper exposure of Via pad will be cause for rejection.*** Maximum elevation of solder mask over via holes is 4 mils.
11. Silkscreen: Two Sides
12. Copper Geometry: 6 mil line, 6 mil space
13. Via pad size: 26 mil dia. Via Hole Size: Per vendor recommendation.
14. Electrical Test: YES, complete for both sides. Reuse REV 0 test fixturing.
15. Technobox to provide all artwork in RS-274X format.
16. Solder Mask Artwork supplied same size as copper features. Vendor to enlarge these solder mask openings in accordance with Vendor's solder mask process.
17. Vendor to add manufacturing logo, date code. Any vendor added test coupons will be outside the perimeter of the board.
18. 125 mil dia tooling holes are located at four corners of this board. Also serves as standoff.
19. Drill sizes supplied are nominal FINISHED hole sizes, to be adjusted by vendor according to vendor's standard plating processes.
20. Board to be trimmed to line center. Trim line to be presented in a separate gerber file, registered by targets across all layers.
21. No gold.
22. Vendor may optionally remove outer surface pads on non-plated holes.

Layer/File stackup as follows:

Gerber File Name	Layer
startera.lgx	Silk Screen top side
starterb.lgx	Solder Mask top side
starterc.lgx	Copper signal top side
starterd.lgx	GND plane
startere.lgx	Interior signal layer
starterf.lgx	Interior signal layer
starterg.lgx	VCC plane (split)
starterh.lgx	Copper signal bottom side
starteri.lgx	Solder mask bottom side
starterj.lgx	Silk Screen Legend - bottom side
trimline.lgx	Trim line w/fabrication information
viaonly.lgx	Layer containing via pads for plugging via holes

Drill specifications follow (note plated vs non-plated holes):

TOOL	COUNT	SIZE	PATH LENGTH	
1	1004	0.013	102708	
2	204	0.038	34182	
4	6	0.064	8000	
5	7	0.125	9850	(NON-PLATED)
6	6	0.110	3150	
23	4	0.070	1800	
26	6	0.098	5293	
27	6	0.065	5043	(NON-PLATED)
TOTAL		170027		

END OF FILE

Company: **Technobox, Inc.**
Board Name: STARTER
BBS File Name: STARTER.ZIP

Layer Count: Six (4 signal, 2 planes)

Barriers:

BARVIA: keep out vias
BARALL: Keep out everything, all layers
BARTOP: Keep out traces, COMP layer
BARBOT: Keep out traces, SOLDER layer

Pin Type Definitions for Vias when using 30 mil dia via pad:

0	*	c30d10.PS	% Vias	30 dia.
51	*	g30d10.PS	% Ground thermal	30 dia.
52	*	v30d10.PS	% Power thermal (VCC, PV5)	30 dia.

Pin Type definitions for Vias when using 26 mil dia via pad:

60	*	c26d10.ps	% Vias	26 dia.
61	*	g26d10.ps	% Ground thermal	26 dia.
62	*	v26d10.ps	% Power Thermal (VCC, PV5)	26 dia.

(Note VCC and PV5 signals are routed to same 'split' power plane in this design and therefore use same pin time for this design only).

Layers:

COMP: Component side signal
GND: ground plane (Route using 'GND' net)
INT1: Internal signal, component side
INT2: Internal signal, solder side
VCC: power.... Route 'VCC' and 'PV5' nets on this layer. I'll split plane manually.
SOLDER: Solder side signal

Special trace widths: 'VCP' and 'VCP5' route with 25 mil wide trace.

Conservative Routing Rules (use this for First Attempt):

8 mil line, 8 mil space
30 mil dia via (Use pin types 0, 51, 52)

Routing rules (more aggressive, if required):

6 mil line, 6 mil space
26 mil diag vias (Use pin types 60, 61, 62)