Z-29 Terminal

ROM Source Listing

This binder contains the source listing of the code in your Z-29 Terminal ROM.

595-3053

First Printing
Printed in the
United States of America

ZENITH | data systems

```
                        ;;;     Z-29 Terminal Firmware Version 1.03
                        ;
                        ;
                        ;       Written by R. Nick K. Boland    February 10, 1983
                        ;
                        ;       Copyright (C) 1983
                        ;       Zenith Data Systems
                        ;       Saint Joseph, Michigan
                        ;
                        ;
                        ;       Initial consultation by Tom Blog
                        ;       Initial hardware by Michael Gorbutt
                        ;       Hardware finished by Anthony M. Olson
                        ;
                        ;
                        ;       Many things have been used to compact the program.  Look
                        ;       for these situations.  The most common are where subroutines
                        ;       use parts of other subroutines.  Other instances are where
                        ;       a single LJMP could be used and many AJMP's used to reference
                        ;       the long jump.
                        ;
                        ;
                                ENABL   LC
                                NLIST   TOC
                                NAME    Z29ROM
                                SBTTL   *** Z-29 Computer Terminal Firmware Version 1.03 ***
```

```
                                ;               ASCII character equivalences
                                ;
0000                    NULL    EQU     00000000B       ; Null
0007                    BELL    EQU     00000111B       ; Bell
0008                    BS      EQU     00001000B       ; Backspace
0009                    HT      EQU     00001001B       ; Horizontal tab
000A                    LF      EQU     00001010B       ; Line feed
000D                    CR      EQU     00001101B       ; Carriage return
0011                    XON     EQU     00010001B       ; DC1 (XON)
0013                    XOFF    EQU     00010011B       ; DC3 (XOFF)
0018                    CAN     EQU     00011000B       ; Cancel
001B                    ESC     EQU     00011011B       ; Escape
007F                    RUBOUT  EQU     01111111B       ; Rubout / delete


                                ;               Constants
                                ;
0018                    MAXLINE EQU     24              ; Max number of lines is 0..24
0019                    NUMLINES EQU    25              ; Number of lines is 1..25
004F                    MAXCHAR EQU     79              ; Max number of chars/line 0..79
0050                    NUMCHARS EQU    80              ; Number of characters is 1..80


                                ;;              8051 processor constants
                                ;
0000                    BANK0   EQU     00000000B       ; Select register bank 0
0008                    BANK1   EQU     00001000B       ; Bank 1
0023                    ITMOD   EQU     00100011B       ; Init value for -TMOD-
005A                    ISCON   EQU     01011010B       ; Init value for -SCON-
0003                    IPINIT  EQU     00000011B       ; Init value for interrupt priorities
0095                    IEINIT  EQU     10010101B       ; Init value for interrupts
0082                    IEDMA   EQU     10000010B       ; Init value for interrupts during DMA
```

```
                        ;;              Keyboard data
                        ;
                        ;               Data from the keyboard other than ASCII data is in the format
                        ;               of iscXXXXXB where s = shift status, c = control status and
                        ;               the rest of the word defines the function.  Excertions to this
                        ;               are the access function, power up function and ID number.
                        ;
                        ;                               0aaaaaaaB       ASCII data to be transmitted
                        ;
0040                    KB_SHFT         EQU     01000000B       ; Keyboard shift mask
0020                    KB_CNTR         EQU     00100000B       ; Keyboard control mask


0080                    KB_UP           EQU     10000000B       ; Up arrow key
0081                    KB_DOWN         EQU     10000001B       ; Down arrow key
0082                    KB_LEFT         EQU     10000010B       ; Left arrow key
0083                    KB_RIGHT        EQU     10000011B       ; Right arrow key
0084                    KB_HOME         EQU     10000100B       ; Home key
0085                    KB_ERASE        EQU     10000101B       ; Erase key
0086                    KB_HELP         EQU     10000110B       ; Help key
0087                    KB_SCROLL       EQU     10000111B       ; Scroll key
0088                    KB_SETUP        EQU     10001000B       ; Setup key
0089                    KB_BREAK        EQU     10001001B       ; Break key
008A                    KB_CAPS         EQU     10001010B       ; Caps lock key
008B                    KB_TAB          EQU     10001011B       ; Tab key
008C                    KB_SPACE        EQU     10001100B       ; Space bar

008F                    KB_POWER        EQU     10001111B       ; Keyboard just powered up

0090                    KB_0            EQU     10010000B       ; Numeric keypad 0 key
0091                    KB_1            EQU     10010001B       ; Numeric keypad 1 key
0092                    KB_2            EQU     10010010B       ; Numeric keypad 2 key
0093                    KB_3            EQU     10010011B       ; Numeric keypad 3 key
0094                    KB_4            EQU     10010100B       ; Numeric keypad 4 key
0095                    KB_5            EQU     10010101B       ; Numeric keypad 5 key
0096                    KB_6            EQU     10010110B       ; Numeric keypad 6 key
0097                    KB_7            EQU     10010111B       ; Numeric keypad 7 key
0098                    KB_8            EQU     10011000B       ; Numeric keypad 8 key
0099                    KB_9            EQU     10011001B       ; Numeric keypad 9 key
009A                    KB_DEC          EQU     10011010B       ; Numeric keypad decimal point key
009B                    KB_ENT          EQU     10011011B       ; Numeric keypad enter key
009C                    KB_DASH         EQU     10011100B       ; Numeric keypad dash key
009D                    KB_COMMA        EQU     10011101B       ; Numeric keypad comma key
```

```
                        ;       Access code and keys that follow
                        ;
                        ;       The access code from the keyboard is very much like an escape
                        ;       character.  It indicates that the second character from the
                        ;       keyboard will be the data with the following definitions.
                        ;
009F                    KB_ACC  EQU     10011111B       ; Access code

0080                    KB_F1   EQU     10000000B       ; Function key #1
0081                    KB_F2   EQU     10000001B       ; Function key #2
0082                    KB_F3   EQU     10000010B       ; Function key #3
0083                    KB_F4   EQU     10000011B       ; Function key #4
0084                    KB_F5   EQU     10000100B       ; Function key #5
0085                    KB_F6   EQU     10000101B       ; Function key #6
0086                    KB_F7   EQU     10000110B       ; Function key #7
0087                    KB_F8   EQU     10000111B       ; Function key #8
0088                    KB_F9   EQU     10001000B       ; Function key #9

0098                    KB_ID   EQU     10011000B       ; ID number  10011xxxB



                        ;;      Keyboard commands
                        ;
                        ;       These are commands that are sent from the terminal to the
                        ;       keyboard.  They tell the keyboard to enter and exit different
                        ;       modes and to perform different functions.
                        ;
0080                    KBC_ID  EQU     10000000B       ; Send keyboard ID number
0081                    KBC_BELL EQU    10000001B       ; Ring bell
0082                    KBC_EC  EQU     10000010B       ; Enable key click
0083                    KBC_DC  EQU     10000011B       ; Disable key click
0084                    KBC_EL  EQU     10000100B       ; Enter caps locked
0085                    KBC_XL  EQU     10000101B       ; Exit caps locked
0086                    KBC_ER  EQU     10000110B       ; Enable auto repeat mode
0087                    KBC_DR  EQU     10000111B       ; Disable auto repeat mode
0088                    KBC_DK  EQU     10001000B       ; Disable keyboard
0089                    KBC_EK  EQU     10001001B       ; Enable keyboard
008A                    KBC_OFFLN EQU   10001010B       ; Turn on offline LED
008B                    KBC_ONLN EQU    10001011B       ; Turn off offline LED
```

```
                                ;;              CRT video controler equivalences
                                ;
0000                    CRESET  EQU     00000000B       ; Reset 8275
004F                    SCRN1   EQU     01001111B       ; Normal rows, 80 chars/row
0098                    SCRN2A  EQU     10011000B       ; 3 VRTC, 25 rows/screen
00DC                    SCRN2B  EQU     11011100B       ; 4 VRTC, 29 rows/screen
0099                    SCRN3   EQU     10011001B       ; line 10 underline, 10 scan lines
004E                    SCRN4A  EQU     01001110B       ; attributes, cursor, 30 HRTC
004F                    SCRN4B  EQU     01001111B       ; attributes, cursor, 32 HRTC

0020                    CSTRT   EQU     00100000B       ; Start display, 0 delay, 1/burst
0040                    CSTOP   EQU     01000000B       ; Stop display
0080                    CCRSR   EQU     10000000B       ; Load cursor position
00A0                    CINT    EQU     10100000B       ; Enable end of frame interrupt
00E0                    CSET    EQU     11100000B       ; Preset counters
```

```
                        ;;              External memory address map
                        ;
2000                    DDATREG         EQU     8*1024          ; 8275 data register
2001                    DCMDREG         EQU     8*1024 + 1      ; 8275 command register
2001                    DSTAREG         EQU     8*1024 + 1      ; 8275 status register

3000                    MLATCH          EQU     12*1024         ; Memory latch

4000                    ATRIBMEM        EQU     16*1024         ; Attribute memory

5000                    CHARMEM         EQU     20*1024         ; Character memory
5780                    L25MEM          EQU     20*1024 + 1920  ; Line 25 memory

57D0                    IBUF            EQU     20*1024 + 2000  ; Input buffer 32 bytes
0009                    ILO             EQU     9               ; Input queue lo water mark
0015                    IHI             EQU     21              ; Input queue hi water mark
0020                    ILEN            EQU     32              ; Input queue length

57F0                    OBUF            EQU     20*1024 + 2032  ; Output buffer 8 bytes
57F7                    OLEN            EQU     OBUF + 7        ; Output queue length

57F8                    PMBUF           EQU     20*1024 + 2040  ; ANSI parameter buffer 8 bytes

6000                    DMAMEM          EQU     24*1024         ; DMA memory

7000                    EAROM           EQU     28*1024         ; EAROM memory


                        ;               External memory constants
                        ;
00EF                    ATRANL          EQU     11101111B       ; Mask -CHARMEM- to -ATRIBMEM-
00CF                    DMAANL          EQU     11001111B       ; Mask -DMAMEM- to -ATRIBMEM-
0010                    CHRORL          EQU     00010000B       ; Mask -ATRIBMEM- to -CHARMEM-
0020                    DMAORL          EQU     00100000B       ; Mask -ATRIBMEM- to -DMAMEM-
```

```
                                ;;              Internal RAM memory setup
                                ;
                                ;               Register bank #0 used in background jobs
                                ;
REG             PTR1            EQU     R0      ; Work pointer #1               (R0)
0000            PTR1A           EQU     00H     ; Address of -PTR1-
REG             PTR2            EQU     R1      ; Work pointer #2               (R1)
0001            PTR2A           EQU     01H     ; Address of -PTR2-
REG             INDX1           EQU     R2      ; Index register 1              (R2)
0002            INDX1A          EQU     02H     ; Address of -INDX1-
REG             INDX2           EQU     R3      ; Index register 2              (R3)
0003            INDX2A          EQU     03H     ; Address of -INDX2-
REG             WORK            EQU     R4      ; Work register                 (R4)
0004            WORKA           EQU     04H     ; Address of -WORK-
REG             TEMP            EQU     R5      ; Temporary holding register    (R5)
0005            TEMPA           EQU     05H     ; Address of -TEMP-
REG             WORK1           EQU     R6      ; Work register 1               (R6)
0006            WORK1A          EQU     06H     ; Address of -WORK1-
REG             WORK2           EQU     R7      ; Work register 2               (R7)
0007            WORK2A          EQU     07H     ; Address of -WORK2-

                                ;               Register bank #1 used in DMA routine
                                ;
REG             DMPTR           EQU     R0      ; Pointer into line order       (R0)
0008            DMPTRA          EQU     08H     ; Address of -DMPTR-
REG             DMLN            EQU     R1      ; Current DMA line number       (R1)
0009            DMLNA           EQU     09H     ; Address of -DMLN-
```

```
   1                            ;               General use variables from 0AH to 1FH
   2                            ;
   3        000A                DMADRH          EQU     0AH             ; Next address of DMA high byte
   4        000B                DMADRL          EQU     0BH             ; Next address of DMA low byte
   5
   6        000C                ISTORE          EQU     0CH             ; Input store pointer
   7        000D                IFETCH          EQU     0DH             ; Input fetch pointer
   8        000E                ICOUNT          EQU     0EH             ; Input number count
   9
  10        000F                OSTORE          EQU     0FH             ; Output store pointer
  11        0010                OFETCH          EQU     10H             ; Output fetch pointer
  12
  13        0011                XPOS            EQU     11H             ; Actual X-position
  14        0012                YPOS            EQU     12H             ; Actual Y-position
  15
  16        0013                LINADH          EQU     13H             ; Address of next character to plot
  17        0014                LINADL          EQU     14H             ; Address of next character to plot
  18
  19        0015                DSADRH          EQU     15H             ; Dispatch address high
  20        0016                DSADRL          EQU     16H             ; Dispatch address low
  21
  22        0017                TSCROLL         EQU     17H             ; Top line of scrolling region
  23        0018                BFIX            EQU     18H             ; Top line of the bottom fixed region
  24
  25        0019                PMADRH          EQU     19H             ; ANSI parameter address high
  26        001A                PMADRL          EQU     1AH             ; ANSI parameter address low
  27        001B                PMNUM           EQU     1BH             ; ANSI number of parameters
  28        001C                PMVALUE         EQU     1CH             ; ANSI current value of parameter
  29
  30        001D                KBIDNUM         EQU     1DH             ; Keyboard ID number
  31        001E                HSLINE          EQU     1EH             ; Hold screen mode line count
  32        001F                GSET            EQU     1FH             ; Graphic set G1 and G0 select
```

```
                                ;               Flags that define byte 20H
                                ;
0020                    XLATCH          EQU     20H             ; External latch byte

0000                    EAROMSAVE       BIT     00H             ; 0 = save EAROM        1 = normal
0001                    BLINKF          BIT     01H             ; 0 = blink chars on    1 = blanked
0002                    PORTF           BIT     02H             ; 0 = main port         1 = auxiliary
0003                    EAROMRCALL      BIT     03H             ; 0 = recall            1 = normal


                                ;               Flags that define byte 21H
                                ;
0021                    ATTRIBUTES      EQU     21H             ; Attribute byte

0008                    RVVA            BIT     08H             ; Reverse video attribute
0009                    UNDLNA          BIT     09H             ; Underline attribute
000A                    BLINKA          BIT     0AH             ; Blink attribute
000B                    HALFIA          BIT     0BH             ; Half intensity attribute
000C                    ALTCHARA        BIT     0CH             ; Alternate character set attribute
000D                    GRPH            BIT     0DH             ; 0 = no graphics       1 = graphics
000E                    PROTECT         BIT     0EH             ; 0 = no protected      1 = protected
000F                    ALTGRPH         BIT     0FH             ; 0 = no alt graphics   1 = alt graphic set


                                ;               Flags that define byte 22H
                                ;
0010                    XOFFSENT        BIT     10H             ; 0 = no terminal XOFF  1 = term XOFF
0011                    UXOFFSENT       BIT     11H             ; 0 = no user XOFF      1 = user XOFF
0012                    GSEL            BIT     12H             ; 0 = G0 selected       1 = G1 selected
0013                    OKTRANS         BIT     13H             ; 0 = do not transmit   1 = ok to trans
0014                    DX              BIT     14H             ; 0 = no X-pos change   1 = changes
0015                    L25EN           BIT     15H             ; 0 = line 25 disabled  1 = 25 enabled
0016                    XOFFRCVED       BIT     16H             ; 0 = no XOFF received  1 = received


                                ;               Flags that define byte 23H
                                ;
0018                    PSDF            BIT     18H             ; Indicates if *PSD* has been done once
0019                    GNPF            BIT     19H             ; Indicates if *GNP* has been done once
001A                    ENBLCUR         BIT     1AH             ; 0 = cursor disabled   1 = enabled
001B                    ERM             BIT     1BH             ; 0 = erase all data    1 = only unprotected
001C                    SHIFTF          BIT     1CH             ; shift data from keyboard
001D                    CONTLF          BIT     1DH             ; control data from keyboard
001E                    ACCESSF         BIT     1EH             ; 0 = no access from kb 1 = in progress
001F                    KBDISF          BIT     1FH             ; 0 = keyboard enabled  1 = disabled


                                ;               Flags that define byte 24H
                                ;
0020                    CLRFLG          BIT     20H             ; Used in screen DMA must allays be 0!
0021                    ICMODEF         BIT     21H             ; 0 = not insert char   1 = IC mode on
0022                    HSSTOPF         BIT     22H             ; 0 = dont hold screen  1 = stop screen
0023                    PRNTF           BIT     23H             ; 0 = not printing      1 = printing
0024                    VSPF            BIT     24H             ; 0 = suppress video    1 = normal
0025                    GATM            BIT     25H             ; 0 = trans all data    1 = only unprotected
```

```
                                ;       Flags that define byte 25H
                                ;
0028                    PRTYENA BIT     28H     ; 0 = no parity         1 = ok parity
0029                    PRTYSTK BIT     29H     ; 0 = normal parity     1 = stick
002A                    PRTYEVN BIT     2AH     ; 0 = odd parity        1 = even parity
002B                    AUTOWRAP BIT    2BH     ; 0 = no auto wrap      1 = wrap around
002C                    AUTOCR  BIT     2CH     ; 0 = no auto CR on LF  1 = auto CR
002D                    AUTOLF  BIT     2DH     ; 0 = no auto LF on CR  1 = auto LF
002E                    ONLINE  BIT     2EH     ; 0 = off line          1 = on line
002F                    FULLDPLX BIT    2FH     ; 0 = half duplex       1 = full duplex

                                ;       Flags that define byte 26H
                                ;
0030                    KPADSHTF BIT    30H     ; 0 = keypad ordinary   1 = shifted
0031                    KPADALTF BIT    31H     ; 0 = keypad ordinary   1 = alternate
0032                    CRUL    BIT     32H     ; 0 = block cursor      1 = underline
0033                    CLKF    BIT     33H     ; 0 = no key click      1 = click
0034                    REPF    BIT     34H     ; 0 = no auto repeat    1 = auto repeat
0035                    CAPLOCKF BIT    35H     ; 0 = caps unlocked     1 = locked
0036                    HSMODEF BIT     36H     ; 0 = not hold screen   1 = hold screen
0037                    L25ON   BIT     37H     ; 0 = status line off   1 = status on

                                ;       Flags that define byte 27H
                                ;
0038                    CRNBK   BIT     38H     ; 0 = blinking cursor   1 = non blink
0039                    FREQ    BIT     39H     ; 0 = 50 Hz             1 = 60 Hz
003A                    MONITOR BIT     3AH     ; 0 = not monitor mode  1 = monitor
003B                    SCRNSAVE BIT    3BH     ; 0 = no screen saver   1 = saver on
003C                    PRTF2   BIT     3CH     ; 0 = main port         1 = auxilary
003D                    ACHR2   BIT     3DH     ; 0 = normal char set   1 = alternate
003E                    HNDSHK  BIT     3EH     ; 0 = software          1 = hardware

                                ;       Flags that define byte 28H
                                ;
0028                    MODE    EQU     28H             ; Mode terminal is in

0040                    ANSI    BIT     40H     ; 0 = other modes       1 = ANSI mode
0041                    ZDS     BIT     41H     ; 0 = other modes       1 = ZDS mode
0042                    ADM3    BIT     42H     ; 0 = other modes       1 = ADM 3 mode
0043                    HAZ     BIT     43H     ; 0 = other modes       1 = Haz 1500

                                ;       Variable tab table
                                ;
0029                    TABTAB  EQU     29H             ; Variable tab table    10 bytes to 32H
```

```
                                ;               General use variables from 33H to 46H
                                ;
0033                    BAUDRATE        EQU     33H             ; Serial port baud rate
0034                    NVRSUM          EQU     34H             ; Non-volatile RAM checksum

0035                    XXPOS           EQU     35H             ; Saved X-position
0036                    XYPOS           EQU     36H             ; Saved Y-position

0037                    BLNKRATE        EQU     37H             ; Blinking rate of screen
0038                    BLNKCNT         EQU     38H             ; Running count for blink rate

0039                    AUTOCNT         EQU     39H             ; Auto off timer count

003A                    TCNT            EQU     3AH             ; Running clock count
003B                    TSEC            EQU     3BH             ; Seconds counter
003C                    TMIN            EQU     3CH             ; Minute counter
003D                    THOUR           EQU     3DH             ; Hour counter

003E                    DISPSEC         EQU     3EH             ; Last count of seconds displayed

003F                    PFIELD          EQU     3FH             ; Protected fields byte
0040                    ERRORS          EQU     40H             ; Powerup diagnostic error flags


                                ;               Screen line table
                                ;
0047                    LORDER          EQU     47H             ; Line order list       25 bytes to 5FH


                                ;               Stack area
                                ;
005F                    STACK           EQU     5FH             ; Stack pointer address 32 bytes to 7FH

                                ;               8051 processor bit port defines
                                ;
0090                    KBOUT           BIT     P1.0            ; Keyboard output line
0091                    KBIN            BIT     P1.1            ; Keyboard input line
0092                    VSP             BIT     P1.2            ; 0 = normal video      1 = suppressed
0093                    SETLCK          BIT     P1.3            ; 0 = setup mode locked 1 = normal
0094                    DTR             BIT     P1.4            ; Data terminal ready
0095                    RTS             BIT     P1.5            ; Ready to send
                        ;               BIT     P1.6            ; Expansion socket
0097                    CLKRUN          BIT     P1.7            ; Clock run for CRT controler

                        ;TXD            BIT     P3.1            ; Predefined serial port transmit pin
00B4                    DMATYPE         BIT     P3.4            ; 0 = clear memory      1 = DMA CRTC
00B5                    CTS             BIT     P3.5            ; Clear to send
```

```
                                ;;              ROM #2 VECTORS
                                ;
                                ;               These locations are provided to facilitate future modifications
                                ;               without a change to the masked processor.  Each location is
                                ;               accessed with a "CALL", and control is gained back with a "RET".
                                ;
                                ;               Initialization of internal RAM and any other devices that
                                ;               may be added.
                                ;
1000                            R2_IN           EQU     1000H

                                ;               Same as R2_IN but does not destroy current settings
                                ;               such as CAPS LOCK, ON/OFF LINE, TIME on clock, etc.
                                ;
1003                            R2_IN2          EQU     1003H

                                ;               Start DMA hook.  This is jumped to at the beginning of each
                                ;               DMA request from the CRTC.
                                ;
1006                            R2_STARTDMA     EQU     1006H

                                ;               Stop DMA hook.  This is jumped to at the beginning of each
                                ;               timer 0 interrupt.
                                ;
1009                            R2_STOPDMA      EQU     1009H

                                ;               Fill screen hook.  This is jumped to start the fill screen
                                ;               DMA feature.  A check for protected fields is made.
                                ;
100C                            R2_FILL         EQU     100CH

                                ;               IRQ hook.  This is jumped to at the beginning of each IRQ.
                                ;
100F                            R2_IRQ          EQU     100FH

                                ;               Serial hook.  This is jumped to at the beginning of each
                                ;               serial interrupt.
                                ;
1012                            R2_SERIAL       EQU     1012H
```

```
                                ;               Transmit hook.  This performs the transmit character
                                ;               function of the terminal for the serial port.
                                ;
1015            R2_XMT          EQU     1015H

                                ;               Send an XON to the host.
                                ;
1018            R2_XON          EQU     1018H

                                ;               Initialize the cursor format.
                                ;
101B            R2_ICUR         EQU     101BH

                                ;               Initialize the keyboard
                                ;
101E            R2_IKB          EQU     101EH

                                ;               SETUP mode vector
                                ;
1021            R2_SETUP        EQU     1021H

                                ;               Background loop hook.  This is called once every time through
                                ;               the main background loop.
                                ;
1024            R2_BACK         EQU     1024H

                                ;               Keyboard input hook.  This is called once before a character
                                ;               is received from the keyboard.
                                ;
1027            R2_KY1          EQU     1027H

                                ;               Keyboard output hook.  This is called once at the beginning
                                ;               of each transfer to the keyboard.
                                ;
102A            R2_KY2          EQU     102AH

                                ;               Escape parsing hook.  This is called after an <ESC> is
                                ;               received.
                                ;
102D            R2_ESC          EQU     102DH
```

```
                                ;               Transmit character.  This performs the function of
                                ;               transmitting the character at the current position.
                                ;
1030                    R2_TC           EQU     1030H

                                ;               Transmit line.  This performs the function of transmitting
                                ;               the line at the current position.
                                ;
1033                    R2_TL           EQU     1033H

                                ;               Transmit page.  This performs the function of transmitting
                                ;               the page to the port that is not selected.
                                ;
1036                    R2_TP           EQU     1036H

                                ;               Transmit 25th line.  This performs the function of transmitting
                                ;               the 25th line to the port that is not selected.
                                ;
1039                    R2_T25L         EQU     1039H

                                ;               Print page.  This performs the function of transmitting the
                                ;               page to the opposite port in a form proper for printing (uses
                                ;               CR, LF at the end of each line).
                                ;
103C                    R2_PRNT         EQU     103CH

                                ;               Display clock.  This updates the time on the 25th line.
                                ;
103F                    R2_DCLK         EQU     103FH

                                ;               Display 25th status line.  This displays the system status
                                ;               on the 25th line if the option is enabled and the user does
                                ;               not have the 25th line enabled.
                                ;
1042                    R2_DISP         EQU     1042H
```

```
  1
  2                                     ;;              INTERRUPT VECTORS
  3                                     ;
  4
  5
  6                                     ;               RESET vector
  7                                     ;
  8             0000                                    ORG     0000H
  9 0000      80      2E                                SJMP    START                   ; Start of program
 10
 11
 12                                     ;               External interrupt request 0
 13                                     ;
 14             0003                                    ORG     0003H
 15 0003     30      AF      02                          JNB     EA, I1                  ; Double check
 16 0006     80      22                                  SJMP    IEXTIO                  ; Start DMA transfer
 17 0008     D2      89              I1:                 SETB    TCON.1
 18 000A     32                                          RETI
 19
 20
 21                                     ;               Timer 0 interrupt vector
 22                                     ;
 23             000B                                    ORG     000BH
 24 000B     30      AF      02                          JNB     EA, I2                  ; Double check
 25 000E     80      1D                                  SJMP    ITMR0                   ; stop DMA transfer
 26 0010     D2      8D              I2:                 SETB    TCON.5
 27 0012     32                                          RETI
 28
 29
 30                                     ;               External interrupt request 1
 31                                     ;
 32             0013                                    ORG     0013H
 33 0013     30      AF      03                          JNB     EA, I3                  ; Double check
 34 0016     02      100F                                LJMP    R2_IRQ                  ; Interrupt request
 35 0019     D2      8B              I3:                 SETB    TCON.3
 36 001B     32                                          RETI
 37
 38
 39                                     ;               Serial port interrupt vector
 40                                     ;
 41             0023                                    ORG     0023H
 42 0023     30      AF      03                          JNB     EA, I5                  ; Double check
 43 0026     02      1012                                LJMP    R2_SERIAL               ; Serial port
 44 0029     32                              I5:         RETI
 45
 46
 47                                     ;               Jumps to other locations
 48                                     ;
 49 002A     02      1006            IEXTIO:             LJMP    R2_STARTDMA             ; Start DMA jump
 50 002D     02      1009            ITMR0:              LJMP    R2_STOPDMA              ; Stop DMA jump
```

```
                                ;;              Initialize system
                                ;
                                ;               Set stack pointer and call routine to set up everything

                0030                            ORG     30H             ; Somewhat arbitrary

0030    75      81      5F      START:          MOV     SP, #STACK      ; Initialize stack pointer
0033    12      1000                            LCALL   R2_IN           ; Initialize everything else


                                ;;              MAIN - main control loop
                                ;
                                ;               *MAIN* is in charge of giving control to the proper routine
                                ;               whenever there is a character present in one of the FIFOs.

0036    12      1024            MAIN:           LCALL   R2_BACK         ; Background hook

0039    20      22      06                      JB      HSSTOPF, MN1    ; IF not hold screen THEN
003C    31      F6                              ACALL   FCIF            ;   fetch char from input FIFO
003E    40      02                              JC      MN1             ;   IF character THEN
0040    11      79                              ACALL   IFCP            ;     processes character

0042    30      2E      20      MN1:            JNB     ONLINE, MN2     ; IF online AND
0045    30      13      1D                      JNB     OKTRANS, MN2    ; IF output port free THEN
0048    20      02      03                      JB      PORTF, MN1A     ;   IF auxiliary port OR
004B    30      3E      03                      JNB     HNDSHK, MN1B    ;   IF hardware handshake THEN
004E    20      B5      14      MN1A:           JB      CTS, MN2        ;     IF not CTS THEN skip
0051    20      16      11      MN1B:           JB      XOFFRCVED, MN2  ;   IF not -XOFF- received THEN
0054    51      4C                              ACALL   FCOF            ;     fetch char from output FIFO
0056    40      0D                              JC      MN2             ;     IF character THEN
0058    B4      13      02                      CJNE    A, #XOFF, MN1C  ;       IF char = xoff THEN
005B    D2      11                              SETB    UXOFFSENT       ;         flag user xoff sent
005D    B4      11      02      MN1C:           CJNE    A, #XON, MN1D   ;       IF char = xon THEN
0060    C2      11                              CLR     UXOFFSENT       ;         clear user xoff sent
0062    12      1015            MN1D:           LCALL   R2_XMT          ;       transmit character

0065    20      1F      06      MN2:            JB      KBDISF, MN3     ; IF keyboard enabled THEN
0068    51      B6                              ACALL   FCKB            ;   fetch char from keyboard
006A    40      02                              JC      MN3             ;   IF keyboard data THEN
006C    51      EE                              ACALL   KBCP            ;     process keyboard data

006E    30      37      C5      MN3:            JNB     L25ON, MAIN     ; IF 25th status enabled AND
0071    20      15      C2                      JB      L25EN, MAIN     ; IF 25th line disabled THEN
0074    12      103F                            LCALL   R2_DCLK         ;   display clock every second

0077    80      BD                              SJMP    MAIN            ; Go forever
```

```
                                ;;              IFCP - input FIFO character processor
                                ;
                                ;               *IFCP* interprets characters which have been received (or
                                ;               struck on the keyboard if off-line or keyed as a local
                                ;               function).  If the character is part of an escape sequence
                                ;               then control is given directly to that routine by means of
                                ;               the -dispatch address-.  If it is a control character then it
                                ;               is looked up in *CTLTAB*.  If found the program jumps to the
                                ;               appropriate routine, otherwise the character is discarded.  If
                                ;               it is a displayable character it is placed on the screen
                                ;               after checking for graphics mode, attributes, etc.
                                ;
                                ;
                                ;               ENTRY   (A) = character from input FIFO
                                ;
                                ;               EXIT    none
                                ;                         .
                                ;               USES    all

0079  B4     18     06          IFCP:           CJNE    A, #CAN, IFPA           ; IF canceled AND
007C  20     43     03                          JB      HAZ, IFPA               ; IF not Haz mode THEN
007F  12     082F                               LCALL   SETNORM                 ;   Restore dispatch address

0082  C0     16                 IFPA:           PUSH    DSADRL                  ; Jump to the
0084  C0     15                                 PUSH    DSADRH                  ; -dispatch address-
0086  22                                       RET

0087  B4     0D     12          NORM:           CJNE    A, #CR, IFP1            ; IF carriage return THEN
008A  30     36     0F                          JNB     HSMODEF, IFP1           ;   IF hold screen mode THEN
008D  E5     12                                 MOV     A, YPOS
008F  04                                        INC     A                       ;     set Y-pos to check
0090  B5     18     00                          CJNE    A, BFIX, NRM0           ;     with bottom fixed region
0093  40     05                 NRM0:           JC      NRM1                    ;     IF YPOS >= bottom THEN
0095  D5     1E     02                          DJNZ    HSLINE, NRM1            ;       decrement line
0098  D2     22                                 SETB    HSSTOPF                 ;       IF zero THEN stop disp
009A  74     0D                 NRM1:           MOV     A, #CR                  ;         restore carriage return

                                ;               Check for monitor mode and control characters
                                ;
009C  30     3A     0D          IFP1:           JNB     MONITOR, IFP1B          ; IF monitor THEN
009F  C2     08                                 CLR     RVVA                    ;   start with normal display
00A1  B4     20     00                          CJNE    A, #' ', IFP1A
00A4  50     5E                 IFP1A:          JNC     IFP6                    ;   IF control character THEN
00A6  D2     08                                 SETB    RVVA                    ;     make it reverse video
00A8  44     40                                 ORL     A, #01000000B           ;     convert to ordinary char
00AA  80     58                                 SJMP    IFP6                    ;     skip ahead

00AC  B4     20     00          IFP1B:          CJNE    A, #' ', IFP1C          ; Normal place to jump to
00AF  40     03                 IFP1C:          JC      IFP2                    ; IF control character OR
00B1  B4     7F     21                          CJNE    A, #RUBOUT, IFP3        ; IF 'rubout' THEN

                                ;               Character is a control character
                                ;
00B4  75     F0     08          IFP2:           MOV     B, #CTAB1L              ; Get table info
00B7  90     0119                               MOV     DPTR, #CTAB1
00BA  20     40     12                          JB      ANSI, IFP2A             ; IF not ANSI mode OR
```

```
 1 00BD  20  41  0F          JB      ZDS, IFP2A          ; IF not ZDS mode THEN
 2 00C0  75  F0  0E          MOV     B, #CTAB2L          ;   set other table info
 3 00C3  90  0107            MOV     DPTR, #CTAB2
 4 00C6  20  42  06          JB      ADM3, IFP2A         ; IF not ADM 3 mode THEN
 5 00C9  75  F0  06          MOV     B, #CTAB3L          ;   set other table info
 6 00CC  90  0128            MOV     DPTR, #CTAB3
 7 00CF  31  3A          IFP2A:  ACALL   STAB            ; Search the table
 8
 9                       ;       Control character found in table, go to associated address.
10                       ;       Subroutine return will cause return to main idle loop
11                       ;
12 00D1  40  33                 JC      IFPRET          ; IF found THEN
13 00D3  E4                     CLR     A
14 00D4  73                     JMP     @A+DPTR         ;   Jump to address in (DPTR)
15
16                       ;       Character is a displayable character
17                       ;
18 00D5  30  43  06     IFP3:   JNB     HAZ, IFP3A      ; IF Hazeltine mode AND
19 00D8  B4  7E  03             CJNE    A, #'~', IFP3A  ; IF char = '~' THEN
20 00DB  02  067B               LJMP    ESCCODE         ;   do escape stuff
21 00DE  30  0D  13     IFP3A:  JNB     GRPH, IFP5      ; IF in graphics mode THEN
22 00E1  B4  5E  06             CJNE    A, #5EH, IFP4
23 00E4  20  0F  0D             JB      ALTGRPH, IFP5   ;   IF alternate THEN dont map
24 00E7  74  7F                 MOV     A, #7FH         ;   IF 5EH THEN map to 7FH
25 00E9  D3                     SETB    C
26 00EA  40  08         IFP4:   JC      IFP5            ;   IF in graphics range THEN
27 00EC  54  1F                 ANL     A, #00011111B   ;     mask to graphic character
28 00EE  30  0F  03             JNB     ALTGRPH, IFP5   ;   IF alt graphic set THEN
29 00F1  D3                     SETB    C               ;     set carry and
30 00F2  80  02                 SJMP    IFP5A           ;     place into alt mode
31
32
33                       ;       Check for insert character mode
34                       ;
35 00F4  A2  0C         IFP5:   MOV     C, ALTCHARA     ; Get alt char mode
36 00F6  92  E7         IFP5A:  MOV     ACC.7, C        ; Place into data byte
37
38 00F8  30  21  09             JNB     ICMODEF, IFP6   ; IF insert character mode THEN
39 00FB  C0  E0                 PUSH    ACC             ;   save character
40 00FD  31  A9                 ACALL   SPF             ;   skip protected fields
41 00FF  12  0DC1               CALL    PIC             ;   insert blank character
42 0102  D0  E0                 POP     ACC             ;   restore character
43
44 0104  31  57         IFP6:   ACALL   PUTCHR          ; Put character onto screen
45 0106  22             IFPRET: RET
```

```
                                ;;              CTAB1/2/3 - control character tables
                                ;
                                ;               *CTAB1* contains all control codes for the ANSI and ZDS
                                ;               emulation modes.
                                ;
                                ;               *CTAB2* contains all control codes for the ADM 3 emulation
                                ;               mode.
                                ;
                                ;               *CTAB3* contains all control codes for the Hazeltine 1500
                                ;               emulation mode.
                                ;
                                ;               Table entries are:  The control character followed by the
                                ;               address of the routine which performs the requested function.


        0107                    CTAB2           EQU     $

0107    0B                                      DB      0BH             ; Vertical tab
0108    0C8D                                    DW      CUP             ; Cursor up

010A    0C                                      DB      0CH             ; Form feed
010B    0C64                                    DW      CRT             ; Cursor right

010D    0E                                      DB      0EH             ; Shift out
010E    0FB3                                    DW      EKI             ; Enable keyboard input

0110    0F                                      DB      0FH             ; Shift in
0111    0FB9                                    DW      DKI             ; Disable keyboard input

0113    1A                                      DB      1AH             ; Substitute
0114    0BAC                                    DW      CLRS            ; Clear screen

0116    1E                                      DB      1EH             ; Record separator
0117    0C36                                    DW      SCH             ; Set cursor home

        0119                    CTAB1           EQU     $

0119    1B                                      DB      ESC             ; Escape
011A    067B                                    DW      ESCCODE         ; Escape code processor

011C    08                                      DB      BS              ; Backspace
011D    0C3D                                    DW      PBS             ; Perform backspace

011F    09                                      DB      HT              ; Horizontal tabulation
0120    05C1                                    DW      TAB             ; Tab to next column

0122    0E                                      DB      0EH             ; Shift out
0123    0F81                                    DW      G1PICK          ; Pick G1 character set

0125    0F                                      DB      0FH             ; Shift in
0126    0F7B                                    DW      G0PICK          ; Pick G0 character set

        0128                    CTAB3           EQU     $

0128    0A                                      DB      LF              ; Line feed
0129    0555                                    DW      PLFCR           ; Perform LF and/or CR
```

*** Z-29 COMPUTER TERMINAL FIRMWARE VERSION 1.03 ***

```
    1 012B   0D                              DB      CR              ; Carriage return
    2 012C   054F                            DW      PCRLF           ; Perform CR and/or LF
    3
    4 012E   07                              DB      BELL            ; Bell
    5 012F   066B                            DW      RING            ; Ring the bell
    6
    7        0008            CTAB1L          EQU     ($-CTAB1)/3     ; Table length
    8        000E            CTAB2L          EQU     ($-CTAB2)/3     ; Table length
    9
   10 0131   08                              DB      BS              ; Back space
   11 0132   0C51                            DW      HCLFT           ; Cursor left
   12
   13 0134   10                              DB      10H             ; Data link escape
   14 0135   0C79                            DW      HCRT            ; Cursor right
   15
   16 0137   09                              DB      HT              ; Horizontal tabulation
   17 0138   0636                            DW      HTAB            ; Special function
   18
   19        0006            CTAB3L          EQU     ($-CTAB3)/3     ; Table length
```

```
                                ;;              STAB - search table
                                ;
                                ;               *STAB* searches tables of specified length for a match
                                ;               with the first character in an entry.
                                ;
                                ;
                                ;               ENTRY   (A) = character to match
                                ;                       (B) = table length in number of entries
                                ;                       (DPTR) = table address
                                ;
                                ;               EXIT    (A) = original character
                                ;                       (B) = number of entries left to search + 1
                                ;                       (DPTR) = first address after character match
                                ;                       'C' = set if no match, clear otherwise
                                ;
                                ;               USES    A, B, DPTR, WORK

013A    FC                      STAB:           MOV     WORK, A                 ; Save character to match

013B    E4                      ST1:            CLR     A                       ; REPEAT
013C    93                                      MOVC    A, @A+DPTR              ;   get character from table
013D    A3                                      INC     DPTR                    ;   bump to address high byte
013E    B5      04      0E                      CJNE    A, WORKA, ST2           ;   compare char with work
0141    E4                                      CLR     A                       ;   IF character = work THEN
0142    93                                      MOVC    A, @A+DPTR              ;     get address high byte
0143    C0      E0                              PUSH    ACC                     ;     save on stack
0145    A3                                      INC     DPTR                    ;     bump to address low byte
0146    E4                                      CLR     A
0147    93                                      MOVC    A, @A+DPTR              ;     get address low byte
0148    F5      82                              MOV     DPL, A                  ;     save address low byte
014A    D0      83                              POP     DPH                     ;     pop address high byte
014C    EC                                      MOV     A, WORK                 ;     restore original character
014D    C3                                      CLR     C                       ;     clear 'C' for ok
014E    22                                      RET                             ;     return

014F    A3                      ST2:            INC     DPTR                    ;   bump to address low byte
0150    A3                                      INC     DPTR                    ;   bump to next entry if any
0151    D5      F0      E7                      DJNZ    B, ST1                  ; UNTIL #entries = 0

0154    EC                                      MOV     A, WORK                 ; Restore original character
0155    D3                                      SETB    C                       ; Flag 'C' for no match
0156    22                                      RET
```

```
                                ;;              PUTCHR - put character onto screen
                                ;
                                ;               *PUTCHR* puts the character onto the screen and checks for
                                ;               wrap around at the end of a line for auto CR or LF.
                                ;
                                ;
                                ;               ENTRY   (A) = character
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    A, B, DPTR, TEMP, PTR1


0157  D2    14          PUTCHR: SETB    DX                      ; Flag that cursor moved

0159  C0    E0                  PUSH    ACC                     ; Save character

015B  31    A9                  ACALL   SPF                     ; Skip any protected fields

015D  85    14    82            MOV     DPL, LINADL             ; Get line address
0160  85    13    83            MOV     DPH, LINADH
0163  53    83    EF            ANL     DPH, #ATRANL            ; Mask to attribute memory
0166  E5    21                  MOV     A, ATTRIBUTES
0168  F0                        MOVX    @DPTR, A                ; Output current attributes
0169  43    83    10            ORL     DPH, #CHRORL            ; Mask to character memory
016C  D0    E0                  POP     ACC                     ; Restore character

016E  F0                        MOVX    @DPTR, A                ; Store into screen memory

016F  E5    11                  MOV     A, XPOS
0171  B4    4F    09            CJNE    A, #MAXCHAR, PU1        ; IF cursor at end of line AND
0174  30    2B    0F            JNB     AUTOWRAP, PU2           ; IF auto wrap on THEN
0177  B1    5C                  ACALL   PCR                     ;   perform carriage return
0179  B1    61                  ACALL   PLF                     ;   perform line feed
017B  80    09                  SJMP    PU2                     ;   exit out

017D  05    11          PU1:    INC     XPOS                    ; Bump -XPOS-
017F  A3                        INC     DPTR                    ; Bump location in memory
0180  85    82    14            MOV     LINADL, DPL             ; Save new line address
0183  85    83    13            MOV     LINADH, DPH

0186  21    A9          PU2:    AJMP    SPF                     ; Skip any protected fields
```

```
                                ;;              CPF - check protected fields
                                ;
                                ;               *CPF* checks to see if the current position is protected.
                                ;               If so it returns a value in the carry flag.
                                ;
                                ;
                                ;               ENTER   XPOS and YPOS at position
                                ;
                                ;               EXIT    (C) = set if protected, clear otherwise
                                ;
                                ;               USES    A, B, DPTR, TEMP, WORK, PTR1

0188  E5      3F               CPF:            MOV     A, PFIELD               ; Get protected fields
018A  70      02                               JNZ     CPF1                    ; IF none protected THEN
018C  C3                       CPFR1:          CLR     C                       ;   clear carry
018D  22                                       RET                             ;   and return

018E  FC                       CPF1:           MOV     WORK, A                 ; Save value
018F  E5      11                               MOV     A, XPOS
0191  85      12      F0                       MOV     B, YPOS
0194  B1      03                               ACALL   CLA                     ; Calculate line address

0196  53      83      EF                       ANL     DPH, #ATRANL            ; Mask to attribute memory
0199  E0                                       MOVX    A, @DPTR                ; Get current attribute
019A  54      0F                               ANL     A, #00001111B           ; Mask to get only attributes

019C  70      05                               JNZ     CPF2                    ; IF no attributes set THEN
019E  BC      80      EB                       CJNE    WORK, #80H, CPFR1       ;   IF not protected THEN return
01A1  80      04                               SJMP    CPFR2                   ;   ELSE return protected
01A3  5C                       CPF2:           ANL     A, WORK                 ; ELSE mask with protected
01A4  B5      04      E5                       CJNE    A, WORKA, CPFR1         ;   IF no match THEN return

01A7  D3                       CPFR2:          SETB    C                       ; Set carry
01A8  22                                       RET                             ; Return
```

```
                                        ;;              SPF - skip over protected fields
                                        ;
                                        ;               *SPF* checks to see if the current position is protected.
                                        ;               If so then the cursor is moved to the next unprotected
                                        ;               character position on the screen.  If there are no unprotected
                                        ;               positions from the current cursor position to the end of the
                                        ;               screen then the cursor is moved to the HOME position.
                                        ;               The screen address is only updated if the cursor position
                                        ;               changes.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    A, B, DPTR, WORK, TEMP, PTR1

01A9  31      88                        SPF:            ACALL   CPF                     ; Check protected fields
01AB  40      01                                        JC      SP0                     ; IF not protected THEN
01AD  22                                                RET                             ;   return

01AE  E5      11                        SP0:            MOV     A, XPOS
01B0  B4      4F      11                                CJNE    A, #MAXCHAR, SP2        ; IF end of line THEN
01B3  75      11      00                                MOV     XPOS, #0                ;   beginning of line
01B6  E5      12                                        MOV     A, YPOS
01B8  B4      17      05                                CJNE    A, #MAXLINE-1, SP1      ;   IF end of screen THEN
01BB  75      12      00                                MOV     YPOS, #0                ;     beginning of screen
01BE  81      EA                                        AJMP    BLINAD                  ;     new address and return

01C0  05      12                        SP1:            INC     YPOS                    ;   ELSE bump line number
01C2  80      02                                        SJMP    SP3                     ;     calc address and go

01C4  05      11                        SP2:            INC     XPOS                    ; ELSE bump column number
01C6  91      EA                        SP3:            ACALL   BLINAD                  ;   calc new address
01C8  80      DF                                        SJMP    SPF                     ;   keep on going
```

```
                                ;;              PCIF - place character in input FIFO
                                ;
                                ;               *PCIF* places a single character into the input FIFO.
                                ;
                                ;
                                ;               ENTRY   (A) = character
                                ;
                                ;               EXIT    (A) = character
                                ;                       'C' = set if no room, clear otherwise
                                ;                       'F0' = set if XOFF is needed, clear otherwise
                                ;
                                ;               USES    A  B, DPTR

01CA  C2  AF            PCIF:   CLR     EA              ; Lock out entries
01CC  75  83  57                MOV     DPH, #HIGH (IBUF)
01CF  85  0C  82                MOV     DPL, ISTORE     ; Load input buffer pointer
01D2  F0                        MOVX    @DPTR, A        ; Write character into FIFO
01D3  F5  F0                    MOV     B, A            ; Save temporary

01D5  E5  0E                    MOV     A, ICOUNT
01D7  B4  20  03                CJNE    A, #ILEN, PI1   ; IF queue is full THEN
01DA  D3                        SETB    C               ;   set 'C' for no room
01DB  80  14                    SJMP    PIR             ;   and exit out

01DD  C2  D5            PI1:    CLR     F0              ; No -XOFF- until otherwise
01DF  B4  15  02                CJNE    A, #IHI, PI2    ; IF queue at XOFF point THEN
01E2  D2  D5                    SETB    F0              ;   flag that -XOFF- is needed

01E4  05  0E            PI2:    INC     ICOUNT          ; Bump count
01E6  05  0C                    INC     ISTORE          ; Bump store and check wrap
01E8  E5  0C                    MOV     A, ISTORE
01EA  B4  F0  03                CJNE    A, #LOW (IBUF+ILEN), PI3
01ED  75  0C  D0                MOV     ISTORE, #LOW (IBUF)

01F0  C3                PI3:    CLR     C               ; Flag everything OK
01F1  D2  AF            PIR:    SETB    EA              ; Enable entries
01F3  E5  F0                    MOV     A, B            ; Restore character
01F5  22                        RET
```

```
                                ;;              FCIF - fetch character from input FIFO
                                ;
                                ;               *FCIF* is used to fetch a single character from the input
                                ;               FIFO if any are available.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    (A) = character if available
                                ;                       'C' = set if no character, clear otherwise
                                ;
                                ;               USES    A, B, DPTR

01F6  E5  0E            FCIF:   MOV     A, ICOUNT
01F8  70  02                    JNZ     FI1             ; IF no data in queue THEN
01FA  D3                        SETB    C               ;   set carry and return
01FB  22                        RET

01FC  B4  09  03        FI1:    CJNE    A, #ILO, FI2    ; IF low water mark THEN
01FF  12  1018                  LCALL   R2_XON          ;   send -XON- now

0202  C2  AF            FI2:    CLR     EA              ; Lock out entries
0204  75  83  57                MOV     DPH, #HIGH (IBUF)
0207  85  0D  82                MOV     DPL, IFETCH     ; Load input buffer pointer
020A  E0                        MOVX    A, @DPTR        ; Read character from FIFO
020B  F5  F0                    MOV     B, A            ; Save temporary

020D  05  0D                    INC     IFETCH          ; Bump fetch and check wrap
020F  E5  0D                    MOV     A, IFETCH
0211  B4  F0  03                CJNE    A, #LOW (IBUF+ILEN), FI3
0214  75  0D  D0                MOV     IFETCH, #LOW (IBUF)     ; Reset value of fetch pointer

0217  15  0E            FI3:    DEC     ICOUNT          ; Decrement count
0219  21  F0                    AJMP    PI3             ; Return (saves ROM)
```

```
                                ;;              LCIF - look at next character on input FIFO
                                ;
                                ;               *LCIF* returns the next character from the input FIFO if there
                                ;               is one but does not take it off the FIFO.  This is needed for
                                ;               'look ahead' applications.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    (A) = character if available
                                ;                       'C' = set if no character, clear otherwise
                                ;
                                ;               USES    A, DPTR


021B   E5   0E          LCIF:   MOV     A, ICOUNT
021D   70   02                  JNZ     LI1                     ; IF no data in queue THEN
021F   D3                       SETB    C                       ;   set carry and return
0220   22                       RET

0221   75   83   57      LI1:    MOV     DPH, #HIGH (IBUF)
0224   85   0D   82              MOV     DPL, IFETCH             ; Load input buffer pointer
0227   E0                       MOVX    A, @DPTR                ; Read character from FIFO
0228   C3                       CLR     C                       ; Flag everything ok
0229   22                       RET
```

```
                                ;;              PCOF - place character in output FIFO
                                ;
                                ;               *PCOF* places a single character into the output FIFO and
                                ;               checks for online and half duplex.
                                ;
                                ;
                                ;               ENTRY   (A) = character
                                ;
                                ;               EXIT    (A) = character
                                ;                       'C' = set if no room in FIFO, clear otherwise
                                ;
                                ;               USES    A, B, DPTR

022A  20  2E  02  PCOF:         JB      ONLINE, PO1             ; IF offline THEN
022D  80  9B                    SJMP    PCIF                    ;   place on input and return

022F  20  2F  02  PO1:          JB      FULLDPLX, PO2           ; IF in half duplex mode THEN
0232  31  CA                    ACALL   PCIF                    ;   place on input also

0234  F5  F0      PO2:          MOV     B, A                    ; Save temporary
0236  E5  0F                    MOV     A, OSTORE
0238  B4  F7  04                CJNE    A, #LOW (OLEN), PO3     ; IF store = queue length THEN
023B  D3                        SETB    C                       ;   set 'C' for full
023C  E5  F0                    MOV     A, B                    ;   restore char and return
023E  22                        RET

023F  C3          PO3:          CLR     C                       ; ELSE clear 'C'
0240  75  83  57                MOV     DPH, #HIGH (OBUF)       ;   set up the output
0243  85  0F  82                MOV     DPL, OSTORE             ;   buffer pointer
0246  E5  F0                    MOV     A, B                    ;   restore character
0248  F0                        MOVX    @DPTR, A                ;   write character to buffer
0249  05  0F                    INC     OSTORE                  ;   bump store and return
024B  22                        RET
```

```
                                        ;;              FCOF - fetch character from output FIFO
                                        ;
                                        ;               *FCOF* fetches a character from the output FIFO if any
                                        ;               are available.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    (A) = character if available
                                        ;                       'C' = set if no characters, clear otherwise
                                        ;
                                        ;               USES    A, B, DPTR

024C E5     OF              FCOF:       MOV     A, OSTORE
024E B5     10      02                  CJNE    A, OFETCH, FOI          ; IF store = fetch THEN
0251 D3                                 SETB    C                       ;   set 'C' and return
0252 22                                 RET

0253 75     83      57      FOI:        MOV     DPH, #HIGH (OBUF)
0256 85     10      82                  MOV     DPL, OFETCH             ; Load output buffer pointer
0259 E0                                 MOVX    A, @DPTR                ; Get character from buffer
025A F5     F0                          MOV     B, A                    ; Save temporary
025C C2     AF                          CLR     EA                      ; Lock out entries
025E 05     10                          INC     OFETCH                  ; Bump fetch pointer
0260 E5     OF                          MOV     A, OSTORE

0262 B5     10      06                  CJNE    A, OFETCH, FOR          ; IF store = fetch THEN
0265 75     10      F0                  MOV     OFETCH, #LOW (OBUF)     ;   reset the values of
0268 75     OF      F0                  MOV     OSTORE, #LOW (OBUF)     ;   fetch and store

026B 21     F0              FOR:        AJMP    PI3                     ; Return (saves ROM)
```

```
                                ;;              PCKB - place character out to the keyboard
                                ;
                                ;               *PCKB* outputs bytes to the keyboard using a special handshake.
                                ;
                                ;
                                ;               ENTRY   (A) = byte to output
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    A, B, WORK

026D  12    102A        PCKB:   LCALL   R2_KY2          ; Keyboard hook

0270  30    91    0C            JNB     KBIN, PK1       ; IF data THEN
0273  C0    E0                  PUSH    ACC             ;   save byte
0275  51    B6                  ACALL   PCKB            ;   fetch char from keyboard
0277  40    02                  JC      PK0             ;   IF none now THEN exit
0279  51    EE                  ACALL   KBCP            ;   ELSE process character
027B  D0    E0          PK0:    POP     ACC
027D  80    EE                  SJMP    PCKB            ;     and try again

027F  C2    AF          PK1:    CLR     EA              ; Disable interrupts
0281  C2    90                  CLR     KBOUT           ; Request keyboard attention
0283  75    F0    08            MOV     B, #8           ; Initialize index for 8 bits

0286  30    91    06            JNB     KBIN, PK1A      ; IF contention THEN
0289  D2    AF                  SETB    EA              ;   enable interrupts
028B  D2    90                  SETB    KBOUT           ;   wipe out request
028D  80    DE                  SJMP    PCKB            ;   try again

028F  D2    AF          PK1A:   SETB    EA              ; Enable interrupts

0291  13                PKL:    RRC     A               ; REPEAT rotate next bit in
0292  7C    00                  MOV     WORK, #0        ;   init count for safety check
0294  20    91    0E    PK2:    JB      KBIN, PK3       ;   REPEAT wait for request
0297  51    B5                  ACALL   PKDLY           ;     delay some time
0299  51    B5                  ACALL   PKDLY
029B  51    B5                  ACALL   PKDLY
029D  51    B5                  ACALL   PKDLY
029F  51    B5                  ACALL   PKDLY
02A1  DC    F1                  DJNZ    WORK, PK2       ;   UNTIL check is done
02A3  80    0E                  SJMP    PKR

02A5  92    90          PK3:    MOV     KBOUT, C        ;   output data
02A7  7C    00                  MOV     WORK, #0        ;   init count for safety check
02A9  30    91    04    PK4:    JNB     KBIN, PK5       ;   REPEAT wait for signal
02AC  DC    FB                  DJNZ    WORK, PK4       ;   UNTIL safety check done
02AE  80    03                  SJMP    PKR

02B0  D5    F0    DE    PK5:    DJNZ    B, PKL          ; UNTIL all 8 bits are done

02B3  D2    90          PKR:    SETB    KBOUT           ; Restore status
02B5  22                PKDLY:  RET
```

```
                                        ;;              FCKB - fetch character from keyboard
                                        ;
                                        ;               *FCKB* takes characters from the keyboard using a special
                                        ;               handshaking.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    (A) = character if available
                                        ;                       'C' = set if no character, clear otherwise
                                        ;
                                        ;               USES    A, B, DPTR


02B6  12  1027            FCKB:         LCALL   R2_KY1                  ; Keyboard hook

02B9  20  91  02                        JB      KBIN, FK1               ; IF no data THEN
02BC  D3                                SETB    C                       ;   set carry and return
02BD  22                                RET

02BE  75  39  00          FK1:          MOV     AUTOCNT, #0
02C1  D2  24                            SETB    VSPF                    ; Toggle screen saver

02C3  E4                                CLR     A                       ; Init receiving byte
02C4  75  F0  08                        MOV     B, #8                   ; Init index counter

02C7  C2  90              FKL:          CLR     KBOUT                   ; REPEAT request data
02C9  51  E1                            ACALL   FKDLY                   ;   delay 200 usec
02CB  51  E1                            ACALL   FKDLY
02CD  A2  91                            MOV     C, KBIN                 ;   get data bit
02CF  D2  90                            SETB    KBOUT                   ;   data received
02D1  92  E7                            MOV     ACC.7, C                ;   place data into word
02D3  D5  F0  02                        DJNZ    B, FK2                  ;   IF all 8 bits THEN
02D6  80  05                            SJMP    FKR                     ;     exit out

02D8  03                  FK2:          RR      A                       ;   rotate to next bit
02D9  51  E1                            ACALL   FKDLY                   ;   delay 100 usec
02DB  80  EA                            SJMP    FKL                     ; UNTIL forever

02DD  51  E1              FKR:          ACALL   FKDLY                   ; Delay last time to let clear
02DF  C3                                CLR     C                       ; Clear 'C' for ok
02E0  22                                RET

                                        ;               Delay routine

02E1  75  83  1E          FKDLY:        MOV     DPH, #30                ; Delay of 100 usec
02E4  D5  83  FD          FKDLY1:       DJNZ    DPH, FKDLY1
02E7  22                                RET
```

```
 1                                ;;              LKB - look at keyboard
 2                                ;
 3                                ;               *LKB* looks at keyboard to see if it has any data it wants
 4                                ;               to send to the host
 5                                ;
 6                                ;
 7                                ;               ENTRY   none
 8                                ;
 9                                ;               EXIT    (C) = set if no character, clear otherwise
10                                ;
11                                ;               USES    none
12
13
14 02E8  C3                       LKB:            CLR     C
15 02E9  20 91 01                                 JB      KBIN, LKBR              ; IF no data THEN
16 02EC  D3                                       SETB    C                       ;   set carry and return
17 02ED  22                       LKBR:           RET
```

```
                                        ;;              KBCP - keyboard character processor
                                        ;
                                        ;               *KBCP* processes any characters received from the keyboard.
                                        ;
                                        ;
                                        ;               ENTRY   (A) = byte from keyboard
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    all

02EE  20      E7      03        KBCP:   JB      ACC.7, KPO              ; IF ASCII character THEN
02F1  41      2A                        AJMP    PCOF                    ;   output char and return
02F3  22                        KPR:    RET

02F4  A2      E6                KPO:    MOV     C, ACC.6
02F6  92      1C                        MOV     SHIFTF, C               ; Save shift value
02F8  A2      E5                        MOV     C, ACC.5
02FA  92      1D                        MOV     CONTLF, C               ; Save control value
02FC  54      9F                        ANL     A, #10011111B           ; Mask off shift and control
02FE  30      1E      02                JNB     ACCESSF, KP1            ; IF access in progress THEN
0301  61      CD                        AJMP    KPA                     ;   do a long jump

                                        ;               No access in progress
                                        ;
0303  75      F0      09        KP1:    MOV     B, #KBDTL               ; (B) = table length
0306  90      044B                      MOV     DPTR, #KBDT             ; (DPTR) = keyboard data table
0309  31      3A                        ACALL   STAB                    ; Search the table
030B  40      02                        JC      KP2                     ; IF found THEN

                                        ;               Command from keyboard
                                        ;
030D  E4                                CLR     A
030E  73                                JMP     @A+DPTR                 ; Jump to routine

                                        ;               Cursor keys
                                        ;
030F  FD                        KP2:    MOV     TEMP, A
0310  E5      0F                        MOV     A, OSTORE               ; IF output FIFO not empty THEN
0312  B5      10      DE                CJNE    A, OFETCH, KPR          ;   loose character and return
0315  ED                                MOV     A, TEMP

0316  75      F0      07                MOV     B, #KBCRTL              ; (B) = table length
0319  90      0466                      MOV     DPTR, #KBCRT            ; (DPTR) = kb cursor table
031C  31      3A                        ACALL   STAB                    ; Search the table
031E  40      2F                        JC      KP6                     ; IF found THEN
0320  20      42      D0                JB      ADM3, KPR               ;   IF ADM 3 mode THEN exit
0323  20      43      CD                JB      HAZ, KPR                ;   IF Haz mode THEN exit
0326  C0      82                        PUSH    DPL                     ;   save values
0328  C0      83                        PUSH    DPH
032A  74      1B                        MOV     A, #ESC
032C  91      14                        ACALL   KPCOF                   ;   output escape
032E  30      40      04                JNB     ANSI, KP3               ;   IF in ANSI mode THEN
```

```
  1 0331  74  5B                        MOV     A, #'['
  2 0333  91  14                        ACALL   KPCOF           ;     output '['
  3 0335  D0  E0              KP3:      POP     ACC
  4 0337  60  12                        JZ      KP4             ; IF erase key AND
  5 0339  30  1C  0F                    JNB     SHIFTF, KP4     ;  IF shift key down THEN
  6 033C  30  40  06                    JNB     ANSI, KP5       ;    IF in ANSI mode THEN
  7 033F  74  32                        MOV     A, #'2'
  8 0341  91  14                        ACALL   KPCOF           ;     output '2'
  9 0343  80  06                        SJMP    KP4
 10 0345  F5  82              KP5:      MOV     DPL, A          ; ELSE in ZDS mode
 11 0347  D0  E0                        POP     ACC             ;   replace char on stack
 12 0349  C0  82                        PUSH    DPL             ;   to be used below
 13 034B  D0  E0              KP4:      POP     ACC
 14 034D  81  14                        AJMP    KPCOF           ; Output next char and return
 15
 16
 17                           ;         Numeric keypad key
 18                           ;
 19 034F  A2  1C              KP6:      MOV     C, SHIFTF
 20 0351  30  30  01                    JNB     KPADSHTF, KP7   ; IF keypad is shifted THEN
 21 0354  B3                            CPL     C               ;   invert shift function
 22 0355  92  1C              KP7:      MOV     SHIFTF, C       ;   save final result
 23 0357  92  E6                        MOV     ACC.6, C        ;   place into data word also
 24 0359  75  F0  1C                    MOV     B, #KBPDTL      ;   (B) = table length
 25 035C  90  047B                      MOV     DPTR, #KBPDT    ;   (DPTR) = keypad table
 26 035F  31  3A                        ACALL   STAB            ;   search the table
 27 0361  50  02                        JNC     KP16            ;   IF not found THEN
 28 0363  80  8E                        SJMP    KPR             ;     exit out
 29
 30 0365  20  1C  22          KP16:     JB      SHIFTF, KP8     ; IF keypad unshifted THEN
 31
 32 0368  20  31  06                    JB      KPADALTF, KP9   ; IF keypad is ordinary THEN
 33 036B  E5  83              KP16A:    MOV     A, DPH
 34 036D  81  14                        AJMP    KPCOF           ;   output ordinary character
 35 036F  80  82                        SJMP    KPR
 36
 37 0371  20  42  F7          KP9:      JB      ADM3, KP16A     ; IF ADM 3 mode THEN ordinary
 38 0374  20  43  F4                    JB      HAZ, KP16A      ; IF Haz mode THEN ordinary
 39 0377  C0  82                        PUSH    DPL             ; IF keypad alternate THEN
 40 0379  74  1B                        MOV     A, #ESC         ;   save character
 41 037B  91  14                        ACALL   KPCOF           ;   output ESC
 42 037D  74  4F                        MOV     A, #'O'         ;   IF in ANSI mode THEN'
 43 037F  20  40  02                    JB      ANSI, KP11      ;     output 'O'
 44 0382  74  3F                        MOV     A, #'?'         ;   ELSE IF in ZDS mode THEN
 45 0384  91  14              KP11:     ACALL   KPCOF           ;     output '?'
 46 0386  D0  E0                        POP     ACC
 47 0388  81  14                        AJMP    KPCOF           ; Output final char and return
 48
 49
 50                           ;         Insert mode key
 51                           ;
 52 038A  20  42  DE          KP8:      JB      ADM3, KP16A     ; IF ADM 3 mode THEN ordinary
 53 038D  20  43  DB                    JB      HAZ, KP16A      ; IF Haz mode THEN ordinary
 54 0390  B4  D7  21                    CJNE    A, #KB_7 + KB_SHFT, KP12
 55 0393  74  1B                        MOV     A, #ESC
 56 0395  91  14                        ACALL   KPCOF           ; Output ESC
 57
```

```
    1 0397   20   40   09            JB    ANSI, KP13          ; IF in ZDS mode THEN
    2 039A   74   40                  MOV   A, #'@'
    3 039C   30   21   02             JNB   ICMODEF, KP14       ;   check if in IC mode
    4 039F   74   4F                  MOV   A, #'O'
    5 03A1   81   14         KP14:    AJMP  KPCOF               ;   output char and return
    6
    7 03A3   74   5B         KP13:    MOV   A, #'['             ; IF in ANSI mode THEN
    8 03A5   91   14                  ACALL KPCOF               ;   output '['
    9 03A7   74   34                  MOV   A, #'4'
   10 03A9   91   14                  ACALL KPCOF               ;   output '4'
   11 03AB   74   68                  MOV   A, #'h'
   12 03AD   30   21   02             JNB   ICMODEF, KP15       ;   IF in IC mode THEN
   13 03B0   74   6C                  MOV   A, #'l'
   14 03B2   81   14         KP15:    AJMP  KPCOF               ;     output char and return
   15
   16
   17                       ;        Numeric keypad shifted
   18                       ;
   19 03B4   FD             KP12:    MOV   TEMP, A
   20 03B5   AC   82                  MOV   WORK, DPL           ; Save second char
   21 03B7   E5   83                  MOV   A, DPH
   22 03B9   91   14                  ACALL KPCOF               ; Output first key
   23 03BB   EC                       MOV   A, WORK
   24 03BC   60   55                  JZ    KPR2                ; IF second char not null THEN
   25 03BE   30   40   0A             JNB   ANSI, KP10
   26 03C1   74   5B                  MOV   A, #'['
   27 03C3   91   14                  ACALL KPCOF               ; IF ANSI THEN output '['
   28 03C5   EC                       MOV   A, WORK
   29 03C6   B4   4E   02             CJNE  A, #'N', KP10       ; IF delete THEN
   30 03C9   74   50                  MOV   A, #'P'             ;   change it for ANSI mode
   31 03CB   81   14         KP10:    AJMP  KPCOF               ;   output second char & return
   32
   33
   34                       ;        Access code in progress
   35                       ;
   36 03CD   C2   1E         KPA:     CLR   ACCESSF             ; Take out of access mode
   37 03CF   FD                       MOV   TEMP, A             ; Save for a while
   38 03D0   54   F8                  ANL   A, #11111000B       ; Mask for keyboard ID number
   39 03D2   B4   98   06             CJNE  A, #KB_ID, KPB      ; IF ID number THEN
   40 03D5   ED                       MOV   A, TEMP             ;   get original data
   41 03D6   54   07                  ANL   A, #00000111B       ;   mask for ID number
   42 03D8   F5   1D                  MOV   KBIDNUM, A          ;   save ID number
   43 03DA   22                       RET
   44
   45
   46                       ;        Function key
   47                       ;
   48 03DB   20   42   35     KPB:    JB    ADM3, KPR2          ; IF ADM 3 mode OR
   49 03DE   20   43   32             JB    HAZ, KPR2           ; IF Haz mode THEN exit
   50
   51 03E1   ED                       MOV   A, TEMP             ; Get original character back
   52
   53 03E2   75   F0   09             MOV   B, #KBFNTL          ; (B) = table length
   54 03E5   90   04CF                MOV   DPTR, #KBFNT        ; (DPTR) = function key table
   55 03E8   31   3A                  ACALL STAB                ; Search the table
   56 03EA   40   27                  JC    KPR2                ; IF found THEN
   57 03EC   C0   82                  PUSH  DPL                 ;   save values returned
```

*** Z-29 COMPUTER TERMINAL FIRMWARE VERSION 1.03 ***

```
    1 03EE   C0   83                        PUSH   DPH
    2 03F0   74   1B                        MOV    A, #ESC
    3 03F2   91   14                        ACALL  KPCOF              ;   output an ESC
    4 03F4   30   40   0B                   JNB    ANSI, KPC          ;   IF in ANSI mode THEN
    5 03F7   74   4F                        MOV    A, #'O'
    6 03F9   91   14                        ACALL  KPCOF              ;     output 'O'
    7 03FB   D0   E0                        POP    ACC
    8 03FD   91   14                        ACALL  KPCOF              ;     output next character
    9 03FF   D0   E0                        POP    ACC
   10 0401   22                             RET
   11
   12 0402   D0   E0             KPC:      POP    ACC                ;   IF in ZDS mode THEN
   13 0404   D0   E0                       POP    ACC                ;     get proper char
   14 0406   30   E7   0B                   JNB    ACC.7, KPCOF       ;     IF high bit set THEN
   15 0409   C2   E7                        CLR    ACC.7              ;       clear high bit
   16 040B   FD                             MOV    TEMP, A
   17 040C   74   30                        MOV    A, #'0'
   18 040E   91   14                        ACALL  KPCOF              ;       output '0'
   19 0410   ED                             MOV    A, TEMP
   20 0411   81   14                        AJMP   KPCOF              ;     output next character
   21 0413   22                   KPR2:     RET
   22
   23
   24                             ;         Check if local function and execute proper routine
   25                             ;
   26 0414   20   1D   02         KPCOF:    JB     CONTLF, KPC1       ; IF control not pressed THEN
   27 0417   41   2A                       AJMP   PCOF               ;   output char and return
   28
   29 0419   21   CA             KPC1:     AJMP   PCIF               ; ELSE do local and return
```

```
                                        ;;              KBTAB - keyboard tab routine
                                        ;
                                        ;               *KBTAB* checks to see if the SHIFT key is down.  If it is
                                        ;               down then it checks for ZDS or ANSI mode.  If it is in one
                                        ;               of those modes then the sequence to for backtab is sent.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    all

041B  30      1C      0F        KBTAB:  JNB     SHIFTF, KT1             ; IF not SHIFT THEN just TAB
041E  90      0431                      MOV     DPTR, #KTM1             ;   set ANSI sequence
0421  20      40      06                JB      ANSI, KTO               ;   IF ANSI mode THEN do it
0424  30      41      06                JNB     ZDS, KT1                ;   IF not ZDS THEN just TAB
0427  90      0434                      MOV     DPTR, #KTM2             ;   ELSE set ZDS sequence
042A  02      0E6D              KTO:    LJMP    PSOF                    ;   output back tab sequence

042D  74      09                KT1:    MOV     A, #HT
042F  41      2A                        AJMP    PCOF                    ; ELSE output horizontal tab

0431  1B      5B      DA        KTM1:   DB      ESC, '[', ('Z'+80H)     ; ESC [ Z
0434  1B      AD                KTM2:   DB      ESC, ('-'+80H)          ; ESC -


                                        ;;              KBSPACE - keyboard space routine
                                        ;
                                        ;               *KBSPACE* outputs a space unless the control key is down
                                        ;               in which case it outputs a null character.
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    all

0436  74      20                KBSPACE: MOV    A, #' '                 ; Default to space
0438  30      1D      01                JNB     CONTLF, KBS1            ; IF control space THEN
043B  E4                                CLR     A                       ;   output a null
043C  41      2A                KBS1:   AJMP    PCOF                    ; Output character and return
```

```
                                        ;;              KBENT - keyboard enter routine
                                        ;
                                        ;               *KBENT* outputs a carriage return unless the shift key is
                                        ;               down in which case it does the print page function.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    all

043E  20      1D      04        KBENT:  JB      CONTLF, KBE1            ; IF no control THEN
0441  74      9B                        MOV     A, #KB_ENT              ;   get keypad enter key
0443  61      0F                        AJMP    KP2                     ;   enter back into code
0445  02      103C              KBE1:   JMP     R2_PRNT                 ; ELSE do print page function


                                        ;;              KBACC - keyboard access routine
                                        ;
                                        ;               *KBACC* is run whenever the access character is received
                                        ;               from the keyboard.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    none

0448  D2      1E                KBACC:  SETB    ACCESSF                 ; Flag access received
044A  22                                RET
```

```
                                ;;              KBDT - keyboard data table
                                ;
                                ;               *KBDT* contains the commands received from the keyboard
                                ;
                                ;               Table entries are:  Keyboard data word followed by the
                                ;               address of the routine which performs the requested function.

                        044B    KBDT            EQU     $

044B    8B                                      DB      KB_TAB                  ; Tab key
044C    041B                                    DW      KBTAB

044E    8C                                      DB      KB_SPACE                ; Space bar
044F    0436                                    DW      KBSPACE

0451    9F                                      DB      KB_ACC                  ; Access code
0452    0448                                    DW      KBACC

0454    8F                                      DB      KB_POWER                ; Keyboard just powered up
0455    101E                                    DW      R2_IKB

0457    89                                      DB      KB_BREAK                ; Break pressed
0458    0FCC                                    DW      KBBREAK

045A    8A                                      DB      KB_CAPS                 ; Caps locked mode
045B    0FBF                                    DW      KBCPLK

045D    87                                      DB      KB_SCROLL               ; Scroll key pressed
045E    0FE3                                    DW      KBSCROLL

0460    88                                      DB      KB_SETUP                ; Setup key pressed
0461    1021                                    DW      R2_SETUP

0463    9B                                      DB      KB_ENT                  ; Enter key pressed
0464    043E                                    DW      KBENT

                        0009    KBDTL           EQU     ($-KBDT)/3              ; Table length
```

```
                                ;;              KBCRT - keyboard cursor table
                                ;
                                ;               *KBCRT* contains the ASCII values to be transmitted by the
                                ;               terminal when received from the keyboard.
                                ;
                                ;               Table entries are:  Keyboard data word followed by the
                                ;               ASCII character to be transmitted if in shift mode followed by
                                ;               the ASCII character to be transmitted always.  The special case
                                ;               here is the erase and shift erase function key.

                        0466    KBCRT           EQU     $
0466 80                                         DB      KB_UP                   ; Up arrow key
0467 00         41                              DB      0, 'A'
0469 81                                         DB      KB_DOWN                 ; Down arrow key
046A 00         42                              DB      0, 'B'
046C 82                                         DB      KB_LEFT                 ; Left arrow key
046D 00         44                              DB      0, 'D'
046F 83                                         DB      KB_RIGHT                ; Right arrow key
0470 00         43                              DB      0, 'C'
0472 84                                         DB      KB_HOME                 ; Home key
0473 00         48                              DB      0, 'H'
0475 86                                         DB      KB_HELP                 ; Help key
0476 00         7E                              DB      0, '~'
0478 85                                         DB      KB_ERASE                ; Erase key
0479 45         4A                              DB      'E', 'J'
                        0007    KBCRTL          EQU     ($-KBCRT)/3             ; Table length
```

```
                                ;;              KBPDT - keyboard key pad table
                                ;
                                ;               *KBPDT* contains the ASCII values to be transmitted by the
                                ;               terminal when received from the keyboard.
                                ;
                                ;               Table entries are:  Keyboard data word followed by the
                                ;               ASCII character to be transmitted if in unshifted followed
                                ;               by the ASCII character to be transmitted if in any mode.
                                ;               The shifted key format uses keyboard data + shift function
                                ;               followed by teh escape sequence or ASCII character to be
                                ;               transmitted.  The exception to this is the insert character
                                ;               key.  In this case the first character is the character for
                                ;               entering IC mode and the second is for leaving IC mode.


                        047B    KBPDT           EQU     $

047B 90                                         DB      KB_0                    ; Keypad 0 key
047C 30         70                              DB      '0', 'p'

047E 91                                         DB      KB_1                    ; Keypad 1 key
047F 31         71                              DB      '1', 'q'

0481 92                                         DB      KB_2                    ; Keypad 2 key
0482 32         72                              DB      '2', 'r'

0484 93                                         DB      KB_3                    ; Keypad 3 key
0485 33         73                              DB      '3', 's'

0487 94                                         DB      KB_4                    ; Keypad 4 key
0488 34         74                              DB      '4', 't'

048A 95                                         DB      KB_5                    ; Keypad 5 key
048B 35         75                              DB      '5', 'u'

048D 96                                         DB      KB_6                    ; Keypad 6 key
048E 36         76                              DB      '6', 'v'

0490 97                                         DB      KB_7                    ; Keypad 7 key
0491 37         77                              DB      '7', 'w'

0493 98                                         DB      KB_8                    ; Keypad 8 key
0494 38         78                              DB      '8', 'x'

0496 99                                         DB      KB_9                    ; Keypad 9 key
0497 39         79                              DB      '9', 'y'

0499 9A                                         DB      KB_DEC                  ; Keypad decimal point key
049A 2E         6E                              DB      '.', 'n'

049C 9B                                         DB      KB_ENT                  ; Keypad enter key
049D 0D         4D                              DB      CR, 'M'

049F 9C                                         DB      KB_DASH                 ; Keypad dash (minus) key
04A0 2D         6D                              DB      '-', 'm'

04A2 9D                                         DB      KB_COMMA                ; Keypad comma key
```

```
04A3   2C    6C                     DB     ',', 'l'

04A5   D0                           DB     KB_0 + KB_SHFT        ; Keypad 0 key shifted
04A6   30    00                     DB     '0', NULL

04A8   D1                           DB     KB_1 + KB_SHFT        ; Keypad 1 key shifted
04A9   1B    4C                     DB     ESC, 'L'

04AB   D2                           DB     KB_2 + KB_SHFT        ; Keypad 2 key shifted
04AC   1B    42                     DB     ESC, 'B'

04AE   D3                           DB     KB_3 + KB_SHFT        ; Keypad 3 key shifted
04AF   1B    4D                     DB     ESC, 'M'

04B1   D4                           DB     KB_4 + KB_SHFT        ; Keypad 4 key shifted
04B2   1B    44                     DB     ESC, 'D'

04B4   D5                           DB     KB_5 + KB_SHFT        ; Keypad 5 key shifted
04B5   1B    48                     DB     ESC, 'H'

04B7   D6                           DB     KB_6 + KB_SHFT        ; Keypad 6 key shifted
04B8   1B    43                     DB     ESC, 'C'

04BA   D7                           DB     KB_7 + KB_SHFT        ; Keypad 7 key shifted
04BB   1B    40                     DB     ESC, '@'

04BD   D8                           DB     KB_8 + KB_SHFT        ; Keypad 8 key shifted
04BE   1B    41                     DB     ESC, 'A'

04C0   D9                           DB     KB_9 + KB_SHFT        ; Keypad 9 key shifted
04C1   1B    4E                     DB     ESC, 'N'

04C3   DA                           DB     KB_DEC + KB_SHFT      ; Keypad decimal key shifted
04C4   2E    00                     DB     '.', NULL

04C6   DB                           DB     KB_ENT + KB_SHFT      ; Keypad enter key shifted
04C7   0D    00                     DB     CR, NULL

04C9   DC                           DB     KB_DASH + KB_SHFT     ; Keypad dash key shifted
04CA   2D    00                     DB     '-', NULL

04CC   DD                           DB     KB_COMMA + KB_SHFT    ; Keypad comma key shifted
04CD   2C    00                     DB     ',', NULL

       001C           KBPDTL        EQU    ($-KBPDT)/3           ; Table length
```

```
                                ;;              KBFNT - keyboard function table
                                ;
                                ;               *KBFNT* contains the ASCII values to be transmitted by the
                                ;               terminal when received from the keyboard.
                                ;
                                ;               Table entries are:  Keyboard data word followed by the
                                ;               ASCII character to be transmitted if in ANSI mode followed
                                ;               by the ASCII character to be transmitted in any other mode.

          04CF                  KBFNT           EQU     $

 04CF     80                                    DB      KB_F1                   ; Function key #1
 04D0     53          53                        DB      'S', 'S'                ; ESC O S,  ESC S

 04D2     81                                    DB      KB_F2                   ; Function key #2
 04D3     54          54                        DB      'T', 'T'                ; ESC O T,  ESC T

 04D5     82                                    DB      KB_F3                   ; Function key #3
 04D6     55          55                        DB      'U', 'U'                ; ESC O U,  ESC U

 04D8     83                                    DB      KB_F4                   ; Function key #4
 04D9     56          56                        DB      'V', 'V'                ; ESC O V,  ESC V

 04DB     84                                    DB      KB_F5                   ; Function key #5
 04DC     57          57                        DB      'W', 'W'                ; ESC O W,  ESC W

 04DE     85                                    DB      KB_F6                   ; Function key #6 (blue)
 04DF     50          50                        DB      'P', 'P'                ; ESC O P,  ESC P

 04E1     86                                    DB      KB_F7                   ; Function key #7 (red)
 04E2     51          51                        DB      'Q', 'Q'                ; ESC O Q,  ESC Q

 04E4     87                                    DB      KB_F8                   ; Function key #8 (gray)
 04E5     52          52                        DB      'R', 'R'                ; ESC O R,  ESC R

 04E7     88                                    DB      KB_F9                   ; Function key #9
 04E8     58          C9                        DB      'X', ('I'+80H)          ; ESC O X,  ESC O I

          0009                  KBFNTL          EQU     ($-KBFNT)/3             ; Table length
```

```
                                        ;;              BLINAD - build line address
                                        ;
                                        ;               *BLINAD* calculates character memory addresses using -XPOS-
                                        ;               and -YPOS- and places the result into -LINADH/L-.
                                        ;
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    A, PTR1

04EA  C0  F0            BLINAD:         PUSH    B               ; Save register
04EC  74  47                            MOV     A, #LORDER      ; Get base of table
04EE  25  12                            ADD     A, YPOS         ; Add in offset
04F0  F8                                MOV     PTR1, A         ; Place into pointer
04F1  E6                                MOV     A, @PTR1        ; Fetch pointer
04F2  75  F0  50                        MOV     B, #NUMCHARS    ; Set up for multiply
04F5  A4                                MUL     AB              ; Do it
04F6  25  11                            ADD     A, XPOS         ; Add in XPOS offset
04F8  F5  14                            MOV     LINADL, A       ; Save low order byte
04FA  E5  F0                            MOV     A, B            ; Get high byte
04FC  34  50                            ADDC    A, #HIGH (CHARMEM)  ; Add carry and offset
04FE  F5  13                            MOV     LINADH, A       ; Save high order byte
0500  D0  F0                            POP     B               ; Restore
0502  22                                RET
```

```
                                ;;              CLA - calculate line address
                                ;
                                ;               *CLA* inputs line coordinates and calculates the current
                                ;               address of that coordinate to character memory.
                                ;
                                ;
                                ;               ENTRY   (A) = X-position
                                ;                       (B) = Y-position
                                ;
                                ;               EXIT    (DPTR) = current address
                                ;
                                ;               USES    A, B, DPTR, TEMP, PTR1

0503    FD                      CLA:            MOV     TEMP, A                 ; Save -XPOS-
0504    74      47                              MOV     A, #LORDER              ; Get base of table
0506    25      F0                              ADD     A, B                    ; Add in -YPOS- offset
0508    F8                                      MOV     PTR1, A                 ; Move into pointer
0509    E6                                      MOV     A, @PTR1                ; Get value from table
050A    75      F0      50                      MOV     B, #NUMCHARS            ; Set up for multiply
050D    A4                                      MUL     AB                      ; Do it
050E    2D                                      ADD     A, TEMP                 ; Add in -XPOS- offset
050F    F5      82                              MOV     DPL, A                  ; Save low byte
0511    E5      F0                              MOV     A, B                    ; Get high byte
0513    34      50                              ADDC    A, #HIGH (CHARMEM)      ; Add memory offset and carry
0515    F5      83                              MOV     DPH, A                  ; Save high byte
0517    22                                      RET
```

```
                                ;;              PRLF - perform a reverse line feed
                                ;
                                ;               *PRLF* moves the cursor up one line in the video memory.
                                ;               If the cursor was already on the top line of the display,
                                ;               the display is scrolled down and the new line is filled
                                ;               with spaces.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

0518  E5      12                PRLF:           MOV     A, YPOS                 ; Get Y-position
051A  B5      17      2B                        CJNE    A, TSCROLL, RL1         ; IF top of scroll region THEN
051D  85      18      F0                        MOV     B, BFIX                 ;   clear bottom line
0520  15      F0                                DEC     B                       ;   of the scrolling region
0522  12      0B96                              LCALL   CLRLINE
0525  74      47                                MOV     A, #LORDER              ; Get top of table
0527  25      18                                ADD     A, BFIX                 ; Add in offset
0529  14                                        DEC     A
052A  F8                                        MOV     PTR1, A
052B  F9                                        MOV     PTR2, A
052C  19                                        DEC     PTR2
052D  86      05                                MOV     TEMPA, @PTR1            ; Save bottom to put on top
052F  E5      18                                MOV     A, BFIX
0531  95      17                                SUBB    A, TSCROLL
0533  14                                        DEC     A                       ; Calc # times through loop
0534  FC                                        MOV     WORK, A
0535  C2      AF                                CLR     EA                      ; Disable for flicker
0537  E7                              RLL:      MOV     A, @PTR2                ; REPEAT
0538  F6                                        MOV     @PTR1, A                ;   exchange values
0539  18                                        DEC     PTR1
053A  19                                        DEC     PTR2                    ;   bump down
053B  DC      FA                                DJNZ    WORK, RLL               ; UNTIL work = 0

053D  A6      05                                MOV     @PTR1, TEMPA            ; Put last value in
053F  D2      AF                                SETB    EA                      ; Enable interrupts again
0541  E5      12                                MOV     A, YPOS                 ; Keep same position
0543  F5      12                RLR:            MOV     YPOS, A
0545  91      EA                                ACALL   BLINAD                  ; Build line address
0547  22                                        RET                             ; Return
0548  B4      00      01        RL1:            CJNE    A, #0, RL1A             ; IF top of screen THEN
054B  22                                        RET                             ;   return
054C  14                        RL1A:           DEC     A                       ; ELSE bump up
054D  80      F4                                SJMP    RLR                     ;   and return
```

```
                                        ;;              PCRLF - perform carriage return and/or line feed
                                        ;
                                        ;               *PCRLF* performs a carriage return and then performs
                                        ;               a line feed if the auto line feed function is selected.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    A, B, DPTR, TEMP, PTR1

 054F   B1       5C              PCRLF:  ACALL   PCR             ; Perform carriage return
 0551   20       2D      0D              JB      AUTOLF, PLF     ; IF auto LF THEN do *PLF*
 0554   22                              RET                     ; ELSE return


                                        ;;              PLFCR - perform line feed and/or carriage return
                                        ;
                                        ;               *PLFCR* performs a carriage return prior to performing a
                                        ;               line feed if the auto carriage return function is selected.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    A, B, DPTR, TEMP, PTR1

 0555   30       2C      09      PLFCR:  JNB     AUTOCR, PLF     ; IF no auto CR THEN do *PLF*
 0558   B1       5C                      ACALL   PCR             ; ELSE do carriage return
 055A   80       05                      SJMP    PLF             ;   line feed and return
```

```
                                ;;              PCR - perform carriage return
                                ;
                                ;               *PCR* moves the cursor to the beginning of the current line.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    A

055C    75      11      00      PCR:            MOV     XPOS, #0                ; Return to beginning of line
055F    81      EA                              AJMP    BLINAD                  ; Build line address and return
```

```
                                        ;;              PLF - perform line feed
                                        ;
                                        ;               *PLF* moves the cursor down one line in the display memory.
                                        ;               Cursor never moves out of the top fixed region.  The display
                                        ;               is scrolled up one line if cursor is just above the bottom
                                        ;               fixed region.  The cursor never moves into the 25th line.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    all

0561  E5  12          PLF:             MOV     A, YPOS                 ; Get Y-position
0563  B4  18  01                       CJNE    A, #MAXLINE, PLF1       ; IF on 25th line THEN
0566  22                              .RET                            ;   return
0567  04                 PLF1:        INC     A                       ; ELSE bump value
0568  B5  18  02                       CJNE    A, BFIX, PLF2           ; IF bottom scroll region THEN
056B  80  04                          SJMP    PLF3                    ;   scroll as usual
056D  B4  18  24       PLF2:          CJNE    A, #MAXLINE, PLFRET     ; IF not 24th line THEN bump
0570  22                              RET                             ; ELSE return
0571  85  17  F0       PLF3:          MOV     B, TSCROLL              ; Set up to clear
0574  12  0B96                        LCALL   CLRLINE                 ; Top line of scroll region
0577  74  47                          MOV     A, #LORDER              ; Calculate pointers needed
0579  25  17                          ADD     A, TSCROLL              ; For adjusting -LORDER-
057B  F8                              MOV     PTR1, A
057C  F9                              MOV     PTR2, A
057D  09                              INC     PTR2
057E  86  05                          MOV     TEMPA, @PTR1            ; Save top value for later
0580  E5  18                          MOV     A, BFIX
0582  95  17                          SUBB    A, TSCROLL              ; Calc number of times
0584  14                              DEC     A                       ; To go through loop
0585  FC                              MOV     WORK, A
0586  C2  AF                          CLR     EA                      ; Stops any blinking
0588  E7                 PLFLP:       MOV     A, @PTR2                ; REPEAT
0589  F6                              MOV     @PTR1, A                ;   switch entries
058A  08                              INC     PTR1
058B  09                              INC     PTR2                    ;   bump pointers
058C  DC  FA                          DJNZ    WORK, PLFLP             ; UNTIL work = 0
058E  A6  05                          MOV     @PTR1, TEMPA            ; Place last value
0590  D2  AF                          SETB    EA                      ; Enable interrupts
0592  E5  12                          MOV     A, YPOS                 ; Keep same position
0594  F5  12           PLFRET:       MOV     YPOS, A                 ; Update -YPOS-
0596  81  EA                          AJMP    BLINAD                  ; Build line address and return
```

```
  1                                ;;              CLRTABS - clear all tabs
  2                                ;
  3                                ;               *CLRTABS* clears all tab settings.
  4                                ;
  5                                ;
  6                                ;               ENTRY   none
  7                                ;
  8                                ;               EXIT    none
  9                                ;
 10                                ;               USES    A, PTR1, INDX1
 11
 12
 13 0598  78      29               CLRTABS:        MOV     PTR1, #TABTAB           ; Init pointer to tab table
 14 059A  E4                                      CLR     A                       ; Init value to place in table
 15 059B  7A      0A                              MOV     INDX1, #10              ; Ten bytes in the tab table
 16
 17 059D  F6                       CLRTB:         MOV     @PTR1, A                ; REPEAT zero byte
 18 059E  08                                      INC     PTR1                    ;   bump pointer
 19 059F  DA      FC                              DJNZ    INDX1, CLRTB            ; UNTIL index = 0
 20 05A1  22                                      RET
```

```
                                ;;              SETTAB - change tab settings
                                ;
                                ;               *SETTAB* sets or clears a tab position depending on the
                                ;               values passed to it.
                                ;
                                ;
                                ;               ENTRY   (A) = tab position from 0..79
                                ;                       'F0' = value to place into tab position
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    A, B, PTR1, WORK, INDX1

05A2  F5     F0                 SETTAB:         MOV     B, A                    ; Save position to set/clear
05A4  54     F8                                 ANL     A, #11111000B           ; Make it a multiple of 8
05A6  FC                                        MOV     WORK, A                 ; Save as a start for counting
05A7  03                                        RR      A                       ; Divide by 8
05A8  03                                        RR      A
05A9  03                                        RR      A
05AA  24     29                                 ADD     A, #TABTAB              ; Add offset to tab table
05AC  F8                                        MOV     PTR1, A                 ; Init pointer
05AD  E6                                        MOV     A, @PTR1                ; Get tab byte
05AE  7A     08                                 MOV     INDX1, #8               ; Loop through all 8 bits
05B0  C5     F0                 STB1:           XCH     A, B                    ; Get count and save byte
05B2  B5     04     04                          CJNE    A, WORKA, STB2          ; If position if found
05B5  A2     D5                                 MOV     C, F0                   ; Get value desired
05B7  92     F7                                 MOV     B.7, C                  ; Place into tab byte
05B9  C5     F0                 STB2:           XCH     A, B                    ; Change back
05BB  23                                        RL      A                       ; Rotate to next tab position
05BC  0C                                        INC     WORK                    ; Bump count
05BD  DA     F1                                 DJNZ    INDX1, STB1             ; Until all 8 bits done
05BF  F6                                        MOV     @PTR1, A                ; Replace the tab byte
05C0  22                                        RET
```

```
                                ;;              TAB - horizontal tab routine
                                ;
                                ;               *TAB* sets current position and then does a search for the
                                ;               next tab stop.  If none is found then the default is the last
                                ;               column position.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    A, B, PTR1, WORK, TEMP, INDX1/2

05C1  E5      11                TAB:            MOV     A, XPOS                 ; Get current position
05C3  FD                                        MOV     TEMP, A                 ; Save position to tab from
05C4  54      F8                                ANL     A, #11111000B           ; Make a multiple of 8
05C6  FC                                        MOV     WORK, A                 ; Save as a start for count
05C7  03                                        RR      A                       ; Divide by 8
05C8  03                                        RR      A
05C9  03                                        RR      A
05CA  F5      F0                                MOV     B, A                    ; Save temporary byte offset
05CC  24      29                                ADD     A, #TABTAB              ; Add offset to tab table
05CE  F8                                        MOV     PTR1, A                 ; Initialize pointer
05CF  74      0A                                MOV     A, #10
05D1  95      F0                                SUBB    A, B                    ; Calc # times needed in loop
05D3  FA                                        MOV     INDX1, A                ; Place into index
05D4  C2      D5                                CLR     F0                      ; F0 = hunting for current pos

05D6  E6                        TBL1:           MOV     A, @PTR1                ; REPEAT get tab byte
05D7  30      D5      02                        JNB     F0, TB6                 ;   IF not looking for position
05DA  60      17                                JZ      TB1                     ;     quick check to exit
05DC  7B      08                TB6:            MOV     INDX2, #8               ;     look through the 8 bits
                                                                                ;     REPEAT
05DE  20      D5      09        TBL2:           JB      F0, TB3                 ;       IF looking for current
05E1  CD                                        XCH     A, TEMP                 ;         get original position
05E2  B5      04      02                        CJNE    A, WORKA, TB4           ;         IF equal to original
05E5  D2      D5                                SETB    F0                      ;           flag it
05E7  CD                        TB4:            XCH     A, TEMP                 ;         get current position
05E8  80      03                                SJMP    TB2                     ;       ELSE looking for tab
05EA  20      E7      0F        TB3:            JB      ACC.7, TBR              ;         IF tab set THEN exit
05ED  23                        TB2:            RL      A                       ;       rotate to next
05EE  0C                                        INC     WORK                    ;       bump count
05EF  DB      ED                                DJNZ    INDX2, TBL2             ;     UNTIL bit count = 0

05F1  80      04                                SJMP    TB5                     ;   IF need to bump by eight
05F3  74      08                TB1:            MOV     A, #8
05F5  2C                                        ADD     A, WORK                 ;     bump count by eight
05F6  FC                                        MOV     WORK, A
05F7  08                        TB5:            INC     PTR1                    ;   bump pointer to next byte
05F8  DA      DC                                DJNZ    INDX1, TBL1             ; UNTIL table is done

05FA  7C      4F                                MOV     WORK, #MAXCHAR          ; Default value if not found

05FC  8C      11                TBR:            MOV     XPOS, WORK              ; Set next tab position
05FE  91      EA                                ACALL   BLINAD                  ; Build new address
0600  21      A9                                JMP     SPF                     ; Skip any protected fields
```

```
                                ;;              BTAB - horizontal back tab routine
                                ;
                                ;               *BTAB* sets current position and then does a search for the
                                ;               previous tab stop.  If none is found then the default is the
                                ;               first column position.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    A, B, PTR1, WORK, TEMP, INDX1/2

0602    E5      11              BTAB:           MOV     A, XPOS                 ; Position to compare with
0604    60      19                              JZ      BTR                     ; IF first column THEN exit
0606    14                                      DEC     A                       ; Decrement for compare
0607    75      11      00                      MOV     XPOS, #0                ; Start looking from first pos

060A    85      11      F0      BTL:            MOV     B, XPOS                 ; REPEAT save previous position
060D    C0      E0                              PUSH    ACC                     ;   save compare value
060F    C0      F0                              PUSH    B                       ;   save previous position
0611    B1      C1                              ACALL   TAB                     ;   tab to next stop
0613    D0      F0                              POP     B                       ;   restore previous position
0615    D0      E0                              POP     ACC                     ;   restore compare value
0617    B5      11      00                      CJNE    A, XPOS, BTL1
061A    50      EE              BTL1:           JNC     BTL                     ; UNTIL next >= current pos

061C    85      F0      11                      MOV     XPOS, B                 ; Get previous position
061F    81      EA              BTR:            JMP     BLINAD                  ; Calc new address and return
```

```
                                ;;              HTS - horizontal tab set
                                ;
                                ;               *HTS* sets tab at current position.
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

0621  D2    D5                  HTS:            SETB    F0                  ; Set tab
0623  E5    11                  HTS1:           MOV     A, XPOS             ; Get position
0625  A1    A2                                  AJMP    SETTAB              ; Execute and return


                                ;;              HTC - Horizontal tab clear
                                ;
                                ;               *HTC* Clears tab at current position.
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

0627  C2    D5                  HTC:            CLR     F0                  ; Clear tab
0629  80    F8                                  SJMP    HTS1                ; Execute and return


                                ;;              ATBC - ANSI tabulation clear
                                ;
                                ;               *ATBC* performs the specified tab controls.
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

062B  12    09D1                ATBC:           LCALL   GNP                 ; Get parameter
062E  60    F7                                  JZ      HTC                 ; IF 0 THEN hor tab clear
0630  B4    03    02                            CJNE    A, #3, AT2          ; ELSE IF 3 THEN
0633  A1    98                                  AJMP    CLRTABS             ;   clear all tabs
0635  22                        AT2:            RET                         ; ELSE return
```

```
                                ;;              HTAB - Hazeltine tab routine
                                ;
                                ;               *HTAB* performs the Hazeltine tab function.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

0636  C2    D5          HTAB:   CLR     F0              ; Flag = FALSE
0638  E5    11                  MOV     A, XPOS
063A  25    12                  ADD     A, YPOS         ; Check for home position

063C  70    02                  JNZ     HT0             ; IF home position THEN
063E  D2    D5                  SETB    F0              ;   flag = TRUE

0640  C0    11          HT0:    PUSH    XPOS            ; Save current position
0642  C0    12                  PUSH    YPOS
0644  75    3F  80              MOV     PFIELD, #10000000B  ; Make normal protected
0647  31    A9                  ACALL   SPF             ; Skip over foreground
0649  E5    11                  MOV     A, XPOS
064B  25    12                  ADD     A, YPOS         ; Check for home position

064D  70    03                  JNZ     HT1             ; IF home position AND
064F  30    D5  09              JNB     F0, HT2         ; IF that is old pos THEN
0652  75    3F  08      HT1:    MOV     PFIELD, #00001000B  ;   make half protected
0655  31    A9                  ACALL   SPF             ;   skip over background
0657  E5    11                  MOV     A, XPOS
0659  25    12                  ADD     A, YPOS         ;   check for home position

065B  75    3F  00      HT2:    MOV     PFIELD, #0      ; Clear protected fields

065E  60    05                  JZ      HT3             ; IF position not home THEN
0660  D0    E0                  POP     ACC             ;   remove old position
0662  D0    E0                  POP     ACC             ;   and return
0664  22                        RET

0665  D0    12          HT3:    POP     YPOS            ; ELSE
0667  D0    11                  POP     XPOS            ;   use previous position
0669  81    EA                  AJMP    BLINAD          ;   update and return
```

```
                                ;;              RING - ring bell
                                ;
                                ;               *RING* rings the bell on the keyboard.  If the input FIFO
                                ;               has a bell as the next character it is dropped.  This is
                                ;               so that the terminal can keep up with multiple bells and
                                ;               communicate to the keyboard.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    none

066B 74 81              RING:   MOV     A, #KBC_BELL
066D 51 6D                      ACALL   PCKB            ; Output command to ring bell

066F 51 1B              RGL:    ACALL   LCIF            ; REPEAT get next input char
0671 40 07                      JC      RGR             ;   IF no character THEN exit
0673 B4 07 04                   CJNE    A, #BELL, RGR   ;   IF character = bell THEN
0676 31 F6                      ACALL   FCIF            ;     fetch character and drop
0678 80 F5                      SJMP    RGL             ; UNTIL char <> bell
067A 22                 RGR:    RET
```

```
                                        ;;              ESCCODE - escape code received
                                        ;
                                        ;               *ESCCODE* determines which mode the terminal is in and
                                        ;               sets the dispatch address to the proper routine.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    none

067B  12      102D              ESCCODE:  LCALL   R2_ESC                  ; Escape hook

067E  75      16      97                  MOV     DSADRL, #LOW (ANSIESC)  ; ANSI dispatch address
0681  75      15      06                  MOV     DSADRH, #HIGH (ANSIESC)

                                          IFC     NE, HIGH(ANSIESC)-HIGH(ZDSESC)
                                          ERROR   ;'ZDSESC' IS NOT THE SAME AS 'ANSIESC'
                                          ENDC
                                          IFC     NE,HIGH(ANSIESC)-HIGH(ADM3ESC)
                                          ERROR   ;'ADM3ESC' IS NOT THE SAME AS 'ANSIESC'
                                          ENDC
                                          IFC     NE,HIGH(ANSIESC)-HIGH(HAZESC)
                                          ERROR   ;'HAZESC' IS NOT THE SAME AS 'ANSIESC'
                                          ENDC

0684  20      40      0F                  JB      ANSI, ESCRET            ; IF not ANSI mode THEN
0687  75      16      B8                  MOV     DSADRL, #LOW (ZDSESC)   ;   ZDS dispatch address

068A  20      41      09                  JB      ZDS, ESCRET             ; IF not ZDS mode THEN
068D  75      16      A7                  MOV     DSADRL, #LOW (ADM3ESC)  ;   ADM 3 dispatch address

0690  20      42      03                  JB      ADM3, ESCRET            ; IF not ADM 3 mode THEN
0693  75      16      B0                  MOV     DSADRL, #LOW (HAZESC)   ;   Hazeltine dispatch address
0696  22                        ESCRET:   RET                             ;       return
```

*** Z-29 COMPUTER TERMINAL FIRMWARE VERSION 1.03 ***

```
                                ;:      ANSIESC - ANSI escape parser
                                ;
                                ;       *ANSIESC* parses the first character following an ESC when
                                ;       in the ANSI mode.
                                ;
                                ;
                                ;       ENTRY   (A) = character to parse
                                ;
                                ;       EXIT    none
                                ;
                                ;       USES    all

0697  C2    18          ANSIESC:    CLR     PSDF                ; Flag first time through *PSD*
0699  C2    19                      CLR     GNPF                ; Flag first time through *GNP*
069B  75    F0    0C                MOV     B, #ANSITL          ; ANSI mode table parameters
069E  90    0798                    MOV     DPTR, #ANSIT

06A1  12    082F        AESC1:      LCALL   SETNORM             ; Restore dispatch address
06A4  02    0836                    LJMP    STJMP               ; Jump to any routine


                                ;:      ADM3ESC - ADM 3 escape parser
                                ;
                                ;       *ADM3ESC* parses the character received to do the proper
                                ;       routine, if any, for ADM 3 mode.
                                ;
                                ;
                                ;       ENTRY   (A) = first character after ESC sequence
                                ;
                                ;       EXIT    none
                                ;
                                ;       USES    all

06A7  12    082F        ADM3ESC:    LCALL   SETNORM             ; Restore dispatch address

06AA  B4    3D    31                CJNE    A, #'=', ZE2        ; IF not '=' THEN return
06AD  02    0CD6                    LJMP    CUA                 ; ELSE same as cursor address
```

```
                                        ;;              HAZESC - Hazeltine 1500 escape (tilde) parser
                                        ;
                                        ;               *HAZESC* parses the character received to do the proper
                                        ;               routine, if any, for Hazeltine 1500 mode.
                                        ;
                                        ;
                                        ;               ENTRY   (A) = first character after ESC (tilde) sequence
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    all

06B0    75      F0      10      HAZESC:         MOV     B, #HAZTL               ; Hazeltine table parameters
06B3    90      0768                            MOV     DPTR, #HAZT
06B6    80      E9                              SJMP    AESC1                   ; Look throush table
```

*** Z-29 COMPUTER TERMINAL FIRMWARE VERSION 1.03 ***

```
                                ;;              ZDSESC - ZDS escape sequence
                                ;
                                ;               *ZDSESC* parses the character received to do the proper ZDS
                                ;               routine if any.
                                ;
                                ;
                                ;               ENTRY   (A) = first character after ESC sequence
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

06B8  12      082F              ZDSESC:         LCALL   SETNORM                 ; Restore dispatch address

06BB  B4      23      03                        CJNE    A, #'#', ZE1A           ; IF transmit page THEN
06BE  02      1036                              JMP     R2_TP                   ;   do it
06C1  B4      2D      02        ZE1A:           CJNE    A, #'-', ZE1B           ; IF back tab THEN
06C4  C1      02                                AJMP    BTAB                    ;   do it
06C6  B4      2E      03        ZE1B:           CJNE    A, #'.', ZE1C           ; IF tab functions THEN
06C9  02      0ED3                              JMP     ZTAB                    ;   do it

06CC  C3                        ZE1C:           CLR     C                       ; Set up to
06CD  94      3C                                SUBB    A, #'<'                 ; Normalize into range
06CF  40      0D                                JC      ZE2                     ; IF out of range THEN return

06D1  23                                        RL      A                       ; ELSE mult by 2
06D2  FD                                        MOV     TEMP, A                 ;   save value
06D3  24      0A                                ADD     A, #10                  ;   add code offset to low
06D5  83                                        MOVC    A, @A+PC                ;   set low byte
06D6  C0      E0                                PUSH    ACC                     ;   put onto stack
06D8  ED                                        MOV     A, TEMP                 ;   set value back
06D9  24      03                                ADD     A, #3                   ;   add code offset to high
06DB  83                                        MOVC    A, @A+PC                ;   set high byte
06DC  C0      E0                                PUSH    ACC                     ;   put onto stack
06DE  22                        ZE2:            RET                             ;   jump to address

                                                IFC     NE,ZDST-$
                                                ERROR   ;TABLE 'ZDST' MUST FOLLOW
                                                ENDC
```

```
                                ;;              ZDST - ZDS escape table
                                ;
                                ;               *ZDST* contains the second character in the range of
                                ;               all escape sequences the Z-29 will respond to.
                                ;
                                ;               Table entries are:  Address of routine which performs the
                                ;               requested operation in order of their ASCII sequence.


06DF  0B03    ZDST:             DW      EAM             ; <  enter ANSI mode
06E1  0B4E                      DW      EKAM            ; =  enter alt keypad mode
06E3  0B51                      DW      XKAM            ; >  exit alt keypad mode
06E5  0767                      DW      UNDEF           ; ?  undefined
06E7  0B54                      DW      EICM            ; @  enter insert char mode
06E9  0C8D                      DW      CUP             ; A  cursor up
06EB  0CA8                      DW      CDN             ; B  cursor down
06ED  0C64                      DW      CRT             ; C  cursor right
06EF  0C3D                      DW      CLFT            ; D  cursor left
06F1  0BAC                      DW      CLRS            ; E  clear entire display
06F3  0F39                      DW      EGM             ; F  enter graphic mode
06F5  0F3C                      DW      XGM             ; G  exit graphic mode
06F7  0C36                      DW      SCH             ; H  set cursor home
06F9  0518                      DW      PRLF            ; I  reverse index
06FB  0C08                      DW      EED             ; J  erase cursor to EOS
06FD  0BE0                      DW      EOL             ; K  erase cursor to EOL
06FF  0D47                      DW      PIL             ; L  perform insert line
0701  0D88                      DW      PDL             ; M  perform delete line
0703  0DF4                      DW      PDC             ; N  delete char
0705  0B58                      DW      XICM            ; O  exit insert char mode
0707  0767                      DW      UNDEF           ; P  undefined (blue)
0709  0767                      DW      UNDEF           ; Q  undefined (red)
070B  0767                      DW      UNDEF           ; R  undefined (gray)
070D  0767                      DW      UNDEF           ; S  undefined (F1)
070F  0767                      DW      UNDEF           ; T  undefined (F2)
0711  0767                      DW      UNDEF           ; U  undefined (F3)
0713  0767                      DW      UNDEF           ; V  undefined (F4)
0715  0767                      DW      UNDEF           ; W  undefined (F5)
0717  0767                      DW      UNDEF           ; X  undefined
0719  0CD6                      DW      CUA             ; Y  VT52 cursor addr (param)
071B  0EAB                      DW      IDT             ; Z  VT52 identify
071D  0B3C                      DW      EHSM            ; [  Enter hold screen mode
071F  0B43                      DW      XHSM            ; \  Exit hold screen mode
0721  1039                      DW      R2_T25L         ; ]  xmit 25th line
0723  1033                      DW      R2_TL           ; ^  transmit line
0725  1030                      DW      R2_TC           ; _  transmit character
0727  103C                      DW      R2_PRNT         ; `  print page
0729  0767                      DW      UNDEF           ; a  undefined
072B  0BF1                      DW      EBD             ; b  erase display to cursor
072D  0767                      DW      UNDEF           ; c  undefined
072F  0767                      DW      UNDEF           ; d  undefined
0731  0767                      DW      UNDEF           ; e  undefined
0733  0767                      DW      UNDEF           ; f  undefined
0735  0767                      DW      UNDEF           ; g  undefined
0737  0767                      DW      UNDEF           ; h  undefined
0739  0EB3                      DW      IDTT            ; i  identify terminal type
073B  0EE8                      DW      SCP             ; j  save cursor position
073D  0EF4                      DW      USCP            ; k  restore cursor position
```

```
    1 073F   0BDB                          DW      EEL             ; l  erase entire line
    2 0741   0767                          DW      UNDEF           ; m  undefined
    3 0743   0E83                          DW      CPR             ; n  cursor position report
    4 0745   0BCB                          DW      EBL             ; o  erase line to cursor
    5 0747   0F33                          DW      ERVM            ; p  enter reverse video mode
    6 0749   0F36                          DW      XRVM            ; q  exit reverse video mode
    7 074B   0E3D                          DW      ZDSSBR          ; r  modify baud rate (param)
    8 074D   0F08                          DW      ZATR            ; s  set video attributes
    9 074F   0B48                          DW      EKSM            ; t  enter keypad shifted mode
   10 0751   0B4B                          DW      XKSM            ; u  exit keypad shifted mode
   11 0753   0AFD                          DW      WEOL            ; v  enter wrap around mode
   12 0755   0B00                          DW      DEOL            ; w  exit wrap around mode
   13 0757   08E7                          DW      SMS             ; x  set modes (param)
   14 0759   092C                          DW      RMS             ; y  reset modes (param)
   15 075B   0AFA                          DW      RAMP            ; z  reset to power up
   16 075D   0FB3                          DW      EKI             ; {  enable keyboard
   17 075F   0767                          DW      UNDEF           ; |  undefined
   18 0761   0FB9                          DW      DKI             ; }  disable keyboard
   19 0763   0767                          DW      UNDEF           ; ~  undefined
   20 0765   0767                          DW      UNDEF           ; DEL  undefined
   21
   22
   23
   24
   25                           ;;         UNDEF - undefined escape sequence
   26                           ;
   27                           ;          *UNDEF* is used in ZDS escape parsing for undefined entries
   28                           ;
   29                           ;
   30                           ;          ENTRY   none
   31                           ;
   32                           ;          EXIT    none
   33                           ;
   34                           ;          USES    none
   35
   36
   37 0767   22                 UNDEF:    RET                           ; Undefined ESC sequence
```

```
                                ;;              HAZT - Hazeltine 1500 escape (tilde) table
                                ;
                                ;               *HAZT* contains the second character of the Hazeltine
                                ;               escape (tilde) table.
                                ;
                                ;               Table entries are:  The second character of the Hazeltine
                                ;               escape (tilde) sequence followed by the address of the routine
                                ;               which performs the requested function.

                0768            HAZT            EQU     $

0768 12                                         DB      12H             ; ~ DC2
0769 0C36                                       DW      SCH             ; Set cursor home

076B 0C                                         DB      0CH             ; ~ FF
076C 0C8D                                       DW      CUP             ; Cursor up

076E 0B                                         DB      0BH             ; ~ VT
076F 0CA8                                       DW      CDN             ; Cursor down

0771 1C                                         DB      1CH             ; ~ FS
0772 0BAC                                       DW      CLRS            ; Clear screen and home

0774 1D                                         DB      1DH             ; ~ GS
0775 0BC2                                       DW      CLFO            ; Clear foreground

0777 0F                                         DB      0FH             ; ~ SI
0778 0BE0                                       DW      EOL             ; Erase to end of line

077A 18                                         DB      18H             ; ~ CAN
077B 0C08                                       DW      EED             ; Erase to end of display

077D 17                                         DB      17H             ; ~ ETB
077E 0C19                                       DW      HEED            ; Erase to eod (with bckgrnd)

0780 15                                         DB      15H             ; ~ NAK
0781 0FB9                                       DW      DKI             ; Disable keyboard input

0783 06                                         DB      06H             ; ~ ACK
0784 0FB3                                       DW      EKI             ; Enable keyboard input

0786 1A                                         DB      1AH             ; ~ SUB
0787 0D47                                       DW      PIL             ; Perform insert line

0789 13                                         DB      13H             ; ~ DC3
078A 0D88                                       DW      PDL             ; Perform delete line

078C 19                                         DB      19H             ; ~ EM
078D 0F2A                                       DW      EHALF           ; Enter half intensity

078F 1F                                         DB      1FH             ; ~ US
0790 0F26                                       DW      XATR            ; Exit attributes

0792 11                                         DB      11H             ; ~ DC1 <col> <row>
0793 0CD6                                       DW      CUA             ; Cursor address
```

```
1 0795   05                                DB      05H                 ; ~ ENQ
2 0796   0E83                              DW      CPR                 ; Cursor position report
3
4        0010                HAZTL         EQU     ($-HAZT)/3          ; Table length
```

```
                                ;;              ANSIT - ANSI escape table
                                ;
                                ;               *ANSIT* contains the second character of the ANSI escape table
                                ;
                                ;               Table entries are:  The second character of the ANSI escape
                                ;               sequence followed by the address of the routine which performs
                                ;               the requested function.


          0798                  ANSIT           EQU     $

0798 5B                                         DB      '['             ; ESC [
0799 07BC                                       DW      ELB             ; Escape left bracket

079B 28                                         DB      '('             ; ESC (
079C 0F3F                                       DW      GODES           ; G0 designator

079E 29                                         DB      ')'             ; ESC )
079F 0F47                                       DW      G1DES           ; G1 designator

07A1 37                                         DB      '7'             ; ESC 7
07A2 0EE8                                       DW      SCP             ; Save cursor position

07A4 38                                         DB      '8'             ; ESC 8
07A5 0EF4                                       DW      USCP            ; Unsave cursor position

07A7 3D                                         DB      '='             ; ESC =
07A8 0B4E                                       DW      EKAM            ; Enter keypad alternate mode

07AA 3E                                         DB      '>'             ; ESC >
07AB 0B51                                       DW      XKAM            ; Exit keypad alternate mode

07AD 4D                                         DB      'M'             ; ESC M
07AE 0518                                       DW      PRLF            ; Reverse index (rev line feed)

07B0 63                                         DB      'c'             ; ESC c
07B1 0AF6                                       DW      ARAMP           ; ANSI reset to power up

07B3 44                                         DB      'D'             ; ESC D
07B4 0561                                       DW      PLF             ; Perform line feed

07B6 48                                         DB      'H'             ; ESC H
07B7 0621                                       DW      HTS             ; Horizontal tab set

07B9 23                                         DB      '#'             ; ESC # 7
07BA 0ADF                                       DW      APRNT           ; Print page

          000C                  ANSITL          EQU     ($-ANSIT)/3     ; Table length
```

```
                                ;;              ELB - escape left bracket
                                ;
                                ;               #ELB# is a continuation of the escape sequence processing
                                ;               for ANSI escape sequences.  The final character of the
                                ;               sequence is decoded with or without a preceeding parameter
                                ;               string and control is passed on to the associated routine.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    to requested routine
                                ;
                                ;               USES    all
                                
                                
07BC  12    082A        ELB:    LCALL   SETDISP         ; Set dispatch and return
                                
07BF  12    0971                LCALL   PSD             ; Get parameter string
07C2  50    78                  JNC     STRET           ; IF not done THEN return
07C4  12    082F                LCALL   SETNORM         ; ELSE restore dispatch address
07C7  75    F0    1E            MOV     B, #ELBTL       ; (B) = table length
07CA  90    07D0                MOV     DPTR, #ELBT     ; (DPTR) = ESC [ table
07CD  02    0836                LJMP    STJMP           ; Jump to any routine
```

```
    1                                   ;;              ELBT - escape left bracket table
    2                                   ;
    3                                   ;               *ELBT* contains the third and/or final character of the
    4                                   ;               ANSI escape sequence for ESC [ Pn F
    5                                   ;
    6                                   ;               Table entries are:  Final character of ESC [ Pn F sequence
    7                                   ;               followed by address of the routine which performs the
    8                                   ;               requested function.
    9
   10
   11           07D0                   ELBT            EQU     $
   12
   13 07D0      3E                                     DB      '>'                     ; ESC [ > Ps F
   14 07D1      0898                                   DW      A2M                     ; ANSI set mode #2
   15
   16 07D3      3F                                     DB      '?'                     ; ESC [ ? Ps F
   17 07D4      0880                                   DW      A1M                     ; ANSI set mode #1
   18
   19 07D6      41                                     DB      'A'                     ; ESC [ Pn A
   20 07D7      0C9D                                   DW      ACUP                    ; ANSI cursor up
   21
   22 07D9      42                                     DB      'B'                     ; ESC [ Pn B
   23 07DA      0CBB                                   DW      ACDN                    ; ANSI cursor down
   24
   25 07DC      43                                     DB      'C'                     ; ESC [ Pn C
   26 07DD      0C6E                                   DW      ACRT                    ; ANSI cursor right
   27
   28 07DF      44                                     DB      'D'                     ; ESC [ Pn D
   29 07E0      0C46                                   DW      ACLFT                   ; ANSI cursor left
   30
   31 07E2      48                                     DB      'H'                     ; ESC [ Pn ; Pn H
   32 07E3      09FA                                   DW      APCA                    ; ANSI perform cursor address
   33
   34 07E5      4A                                     DB      'J'                     ; ESC [ Ps J
   35 07E6      0A1A                                   DW      EID                     ; Erase in display
   36
   37 07E8      4B                                     DB      'K'                     ; ESC [ Ps K
   38 07E9      0A2B                                   DW      EIL                     ; Erase in line
   39
   40 07EB      4C                                     DB      'L'                     ; ESC [ Pn L
   41 07EC      0D7D                                   DW      APIL                    ; ANSI perform insert line
   42
   43 07EE      4D                                     DB      'M'                     ; ESC [ Pn M
   44 07EF      0DB6                                   DW      APDL                    ; ANSI perform delete line
   45
   46 07F1      50                                     DB      'P'                     ; ESC [ Pn P
   47 07F2      0E32                                   DW      APDC                    ; ANSI perform delete char
   48
   49 07F4      5A                                     DB      'Z'                     ; ESC [ Pn Z
   50 07F5      0CC6                                   DW      APBT                    ; ANSI perform back tab
   51
   52 07F7      63                                     DB      'c'                     ; ESC [ Pn c
   53 07F8      0EC3                                   DW      ADA                     ; ANSI device attributes
   54
   55 07FA      66                                     DB      'f'                     ; ESC [ Pn ; Pn f
   56 07FB      09FA                                   DW      APCA                    ; ANSI perform cursor address
   57
```

```
  1 07FD  67                          DB      'g'             ; ESC [ Ps g
  2 07FE  062B                        DW      ATBC            ; ANSI tabulation clear
  3
  4 0800  68                          DB      'h'             ; ESC [ Ps h
  5 0801  083D                        DW      ASM             ; ANSI set mode
  6
  7 0803  6C                          DB      'l'             ; ESC [ Ps l
  8 0804  085F                        DW      ARM             ; ANSI reset mode
  9
 10 0806  6D                          DB      'm'             ; ESC [ Ps m
 11 0807  0A3B                        DW      ASGM            ; ANSI set graphics mode
 12
 13 0809  6E                          DB      'n'             ; ESC [ Pn n
 14 080A  0AAF                        DW      ACPR            ; ANSI cursor position report
 15
 16 080C  70                          DB      'p'             ; ESC [ p
 17 080D  1030                        DW      R2_TC           ; Transmit page
 18
 19 080F  72                          DB      'r'             ; ESC [ Pn ; Pn r
 20 0810  0D24                        DW      DSR             ; ANSI define scrolling region
 21
 22 0812  73                          DB      's'             ; ESC [ s
 23 0813  0EEF                        DW      ASCP            ; ANSI save cursor position
 24
 25 0815  75                          DB      'u'             ; ESC [ u
 26 0816  0F03                        DW      AUSCP           ; ANSI unsave cursor position
 27
 28 0818  76                          DB      'v'             ; ESC [ Pn v
 29 0819  0F96                        DW      SBLR            ; Set blink rate
 30
 31 081B  77                          DB      'w'             ; ESC [ Pn w
 32 081C  0AEA                        DW      ASBR            ; ANSI set baud rate
 33
 34 081E  7A                          DB      'z'             ; ESC [ Pn z
 35 081F  0AF6                        DW      ARAMP           ; ANSI reset to power up
 36
 37 0821  7B                          DB      '{'             ; ESC [ Ps {
 38 0822  0B7A                        DW      PCLK            ; Program clock
 39
 40 0824  7D                          DB      '}'             ; ESC [ Ps }
 41 0825  0A65                        DW      ASPF            ; Set protected fields
 42
 43 0827  40                          DB      '@'             ; ESC [ Pn @
 44 0828  0DE9                        DW      APIC            ; ANSI perform insert character
 45
 46       001E               ELBTL    EQU     ($-ELBT)/3      ; Table length
```

```
                                        ;;              SETDISP - set dispatch address
                                        ;
                                        ;               *SETDISP* sets the dispatch address to the address just
                                        ;               following the call to this routine.  This routine then
                                        ;               executes a -RET-.  Be carefull with this.  A call to this
                                        ;               routine does not return to back to the routine that called
                                        ;               this one.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    none

082A    D0      15              SETDISP:        POP     DSADRH                  ; Get high byte of data
082C    D0      16                              POP     DSADRL                  ; Get low byte of data
082E    22                              RET


                                        ;;              SETNORM - set dispatch normal
                                        ;
                                        ;               *SETNORM* sets the dispatch address for processing characters
                                        ;               back to the 'normal' address.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    none

082F    75      16      87      SETNORM:        MOV     DSADRL, #LOW (NORM)     ; Restore dispatch address
0832    75      15      00                      MOV     DSADRH, #HIGH (NORM)
0835    22                              RET
```

.Z-29.COMPUTER.TERMINAL.FIRMWARE.VERSION.1.03.

```
                        ;;      STJMP - search table and jump
                        ;       STCALL - jump to routine in (DPTR)
                        ;
                        ;       *STJMP* does a search table operation.  If the data was found
                        ;       it does a jump to the address in the (DPTR), otherwise it
                        ;       does a return.
                        ;
                        ;       *STCALL only does a jump to the address in the (DPTR).
                        ;
                        ;
                        ;       ENTRY   (DPTR) = address to jump to
                        ;
                        ;       EXIT    none
                        ;
                        ;       USES    A

0836  31      F7        STJMP:  ACALL   XSTAB           ; Search the table
0838  40      02                JC      STRET           ; IF carry is set THEN return

083A  E4                STCALL: CLR     A
083B  73                        JMP     @A+DPTR         ; ELSE jump to address

083C  22                STRET:  RET                     ; Return
```

```
                                ;;              ASM - ANSI set mode sequence
                                ;
                                ;               *ASM* set the specified mode(s).
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

083D  31    D1          ASM:    ACALL   GNP             ; Get next (or first) parameter
083F  40    0E                  JC      ASMR            ; IF parameter THEN
0841  75    F0    05            MOV     B, #ASMTL       ;   (B) = table length
0844  90    0850                MOV     DPTR, #ASMT     ;   (DPTR) = "ESC [ Pn h" table
0847  31    F7                  ACALL   XSTAB           ;   search table
0849  40    04                  JC      ASMR            ;   IF found THEN
084B  11    3A                  ACALL   STCALL          ;     jump to routine
084D  80    EE                  SJMP    ASM             ;     and keep going

084F  22                ASMR:   RET


                                ;;              ASMT - ANSI set mode sequence table
                                ;
                                ;               *ASMT* contains valid parameters for "ESC [ Pn h" sequence.


        0850                    ASMT    EQU     $

0850  01                        DB      1               ; ESC [ 1 h
0851  0B6E                      DW      EGATM           ; Enter guarded area transfer

0853  02                        DB      2               ; ESC [ 2 h
0854  0FB9                      DW      DKI             ; Disable keyboard input

0856  04                        DB      4               ; ESC [ 4 h
0857  0B54                      DW      EICM            ; Enter insert character mode

0859  06                        DB      6               ; ESC [ 6 h
085A  0B74                      DW      EERM            ; Enter erasure mode

085C  14                        DB      20              ; ESC [ 20 h
085D  0B62                      DW      EACR            ; Enable auto CR on LF

        0005            ASMTL   EQU     ($-ASMT)/3      ; Table length
```

```
                                ;;              ARM - ANSI reset mode sequence
                                ;
                                ;               *ARM* reset the specified mode(s).
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

085F  31      D1                ARM:            ACALL   GNP                     ; REPEAT get parameter
0861  40      EC                                JC      ASMR                    ;   IF no parameter THEN return
0863  75      F0      05                        MOV     B, #ARMTL               ;   (B) = table length
0866  90      0871                              MOV     DPTR, #ARMT             ;   (DPTR) = "ESC [ Pn ]" table
0869  31      F7                                ACALL   XSTAB                   ;   search table
086B  40      E2                                JC      ASMR                    ;   IF not found THEN return
086D  11      3A                                ACALL   STCALL                  ;   call routine
086F  80      EE                                SJMP    ARM                     ; UNTIL forever


                                ;;              ARMT - ANSI reset mode sequence table
                                ;
                                ;               *ARMT* contains valid parameters for "ESC [ Pn ]" sequence


                0871            ARMT            EQU     $

0871  01                                        DB      1                       ; ESC [ 1 ]
0872  0B71                                      DW      XGATM                   ; Exit guarded area transfer

0874  02                                        DB      2                       ; ESC [ 2 ]
0875  0FB3                                      DW      EKI                     ; Enable keyboard input

0877  04                                        DB      4                       ; ESC [ 4 ]
0878  0B58                                      DW      XICM                    ; Exit insert character mode

087A  06                                        DB      6                       ; ESC [ 6 ]
087B  0B77                                      DW      XERM                    ; Exit erasure mode

087D  14                                        DB      20                      ; ESC [ 20 ]
087E  0B65                                      DW      XACR                    ; Disable auto CR on LF

                0005            ARMTL           EQU     ($-ARMT)/3              ; Table length
```

```
                                ;;              A1M - ANSI mode #1
                                ;
                                ;               *A1M* inputs the parameter string and final character for
                                ;               the  ESC [ ? Ps F  sequence.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

0880    C2      18              A1M:            CLR     PSDF                    ; Clear *PSD* flag
0882    11      2A                              ACALL   SETDISP                 ; Set dispatch and return

0884    31      71                              ACALL   PSD                     ; Get parameter string
0886    50      C7                              JNC     ASMR                    ; IF not done THEN return
0888    11      2F                              ACALL   SETNORM                 ; ELSE restore dispatch
088A    75      F0      02                      MOV     B, #EQMTL               ; (B) = table length
088D    90      0892                            MOV     DPTR, #EQMT             ; (DPTR) = ESC [ ? table
0890    01      36                              AJMP    STJMP                   ; Jump to any routine


                                ;;              EQMT - escape [ ? Pn Ps table
                                ;
                                ;               *EQMT* contains the fourth and/or final character


        0892                    EQMT            EQU     $

0892    68                                      DB      'h'                     ; ESC [ ? Pn h
0893    08B0                                    DW      A1SM                    ; ANSI set mode #1

0895    6C                                      DB      'l'                     ; ESC [ ? Pn l
0896    08CC                                    DW      A1RM                    ; ANSI reset mode

        0002                    EQMTL           EQU     ($-EQMT)/3              ; Table length
```

```
                                ;;              A2M - ANSI mode #2
                                ;
                                ;               *A2M* inputs the parameter string and final character for
                                ;               the  ESC [ > Pn Ps  sequence.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all
                                
                                
0898  C2     18                 A2M:            CLR     PSDF                    ; Clear #PSD* flag
089A  11     2A                                 ACALL   SETDISP                 ; Set dispatch and return
                                
089C  31     71                                 ACALL   PSD                     ; Get parameter string
089E  50     22                                 JNC     A1SR                    ; IF not done THEN return
08A0  11     2F                                 ACALL   SETNORM                 ; ELSE restore dispatch
08A2  75     F0     02                          MOV     B, #EGTTL               ; (B) = table length
08A5  90     08AA                               MOV     DPTR, #EGTT             ; (DPTR) = ESC [ > table
08A8  01     36                                 AJMP    STJMP                   ; Jump to any routine
                                
                                
                                
                                ;;              EGTT - escape [ > Pn Ps table
                                ;
                                ;               *EGTT* contains the fourth and/or final character.
                                
                                
              08AA              EGTT            EQU     $
                                
08AA  68                                        DB      'h'                     ; ESC [ > Pn h
08AB  08FF                                      DW      A2SM                    ; ANSI set mode #1
                                
08AD  6C                                        DB      'l'                     ; ESC [ > Pn l
08AE  0944                                      DW      A2RM                    ; ANSI reset mode
                                
              0002              EGTTL           EQU     ($-EGTT)/3              ; Table length
```

```
                                ;;              A1SM - ANSI mode #1 set mode sequence
                                ;
                                ;               *A1SM* set the specified mode(s).
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    A, B, DPTR

08B0    31      D1              A1SM:           ACALL   GNP                     ; Get next (or first) parameter
08B2    40      0E                              JC      A1SR                    ; IF parameter THEN
08B4    75      F0      03                      MOV     B, #A1STL               ; (B) = table length
08B7    90      08C3                            MOV     DPTR, #A1ST             ; (DPTR) = ESC [ ? Pn h table
08BA    31      F7                              ACALL   XSTAB                   ; Search table
08BC    40      04                              JC      A1SR                    ; IF found THEN
08BE    11      3A                              ACALL   STCALL                  ;   jump to routine
08C0    80      EE                              SJMP    A1SM                    ;   and keep going

08C2    22                      A1SR:           RET


                                ;;              A1ST - ANSI mode #1 set mode table
                                ;
                                ;               *A1SMT* contains valid parameters for ESC [ ? Pn h sequence.

        08C3                    A1ST            EQU     $

08C3    02                                      DB      2                       ; ESC [ ? 2 h
08C4    0B07                                    DW      EZM                     ; Enter ZDS mode

08C6    03                                      DB      3                       ; ESC [ ? 3 h
08C7    0BAC                                    DW      CLRS                    ; Enter 132 column mode

08C9    07                                      DB      7                       ; ESC [ ? 7 h
08CA    0AFD                                    DW      WEOL                    ; Wrap at end of line

        0003                    A1STL           EQU     ($-A1ST)/3              ; Table length
```

```
                                ;;              A1RM - ANSI mode #1 reset mode sequence
                                ;
                                ;               *A1RM* resets the specified mode(s).
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    A, B, DPTR

08CC   31      D1               A1RM:           ACALL   GNP                     ; REPEAT get parameter
08CE   40      F2                              JC      A1SR                    ;   IF no parameter THEN exit
08D0   75      F0      03                      MOV     B, #A1RTL               ;   (B) = table length
08D3   90      08DE                            MOV     DPTR, #A1RT             ;   (DPTR) = ESC [ ? Pn l table
08D6   31      F7                              ACALL   XSTAB                   ;   Search table
08D8   40      E8                              JC      A1SR                    ;   IF found THEN
08DA   11      3A                              ACALL   STCALL                  ;     jump to routine
08DC   80      EE                              SJMP    A1RM                    ; UNTIL forever


                                ;;              A1RT - ANSI mode #1 reset mode table
                                ;
                                ;               *A1RMT* contains valid parameters for ESC [ ? Pn l sequence.


        08DE                    A1RT            EQU     $

08DE   02                                       DB      2                       ; ESC [ ? 2 l
08DF   0B07                                     DW      EZM                     ; Enter VT52 mode

08E1   03                                       DB      3                       ; ESC [ ? 3 l
08E2   0BAC                                     DW      CLRS                    ; Enter 80 column mode

08E4   07                                       DB      7                       ; ESC [ ? 7 l
08E5   0B00                                     DW      DEOL                    ; Discard past end of line

        0003                    A1RTL           EQU     ($-A1RT)/3              ; Table length
```

```
                                ;;              SMS - set mode sequence
                                ;
                                ;               *SMS* sets the mode specified by the last character in the
                                ;               sequence.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all


08E7   11      2A               SMS:            ACALL   SETDISP                 ; Set dispatch and return

08E9   11      2F                              ACALL   SETNORM                 ; Restore dispatch address
08EB   C3                                      CLR     C
08EC   94      31                              SUBB    A, #'1'                 ; Subtract offset
08EE   40      0E                              JC      SMSRET                  ; Check for lower bounds
08F0   B4      09      00                      CJNE    A, #('9'-'0'), SMS1
08F3   50      09              SMS1:           JNC     SMSRET                  ; Check for upper bounds
08F5   04                                      INC     A                       ; Bump up into range
08F6   75      F0      09                      MOV     B, #SMSTL               ; (B) = table length
08F9   90      0911                            MOV     DPTR, #SMST             ; (DPTR) = ESC x Pn table
08FC   01      36                              AJMP    STJMP                   ; Jump to any routine

08FE   22                      SMSRET:         RET
```

```
                                ;;              A2SM - ANSI mode #2 set mode sequence
                                ;
                                ;               *A2SM* set the specified mode(s).
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    A, B, DPTR

                                A2SM:           ACALL   GNP             ; Get next (or first) parameter
0901    40      FB                              JC      SMSRET          ; IF parameter THEN
0903    75      F0      09                      MOV     B, #SMSTL       ; (B) = table length
0906    90      0911                            MOV     DPTR, #SMST     ; (DPTR) = ESC [ > Pn h table
0909    31      F7                              ACALL   XSTAB           ; Search table
090B    40      F1                              JC      SMSRET          ; IF found THEN
090D    11      3A                              ACALL   STCALL          ;   jump to routine
090F    80      EE                              JMP     A2SM            ;   and keep going


                                ;;              SMST - set mode sequence table
                                ;
                                ;               *SMST* contains valid parameters for  ESC x Pn  sequence
                                ;               and  ESC [ > Pn h  sequence

                0911            SMST            EQU     $

0911    01                                      DB      1               ; ESC x 1   ESC [ > 1 h
0912    0B0B                                    DW      E25L            ; Enable 25th line

0914    02                                      DB      2               ; ESC x 2   ESC [ > 2 h
0915    0B2D                                    DW      NKC             ; No keyboard click

0917    03                                      DB      3               ; ESC x 3   ESC [ > 3 h
0918    0B3C                                    DW      EHSM            ; Enter hold screen mode

091A    04                                      DB      4               ; ESC x 4   ESC [ > 4 h
091B    0B33                                    DW      SBC             ; Set "block" cursor

091D    05                                      DB      5               ; ESC x 5   ESC [ > 5 h
091E    0B5F                                    DW      DC              ; Disable cursor

0920    06                                      DB      6               ; ESC x 6   ESC [ > 6 h
0921    0B48                                    DW      EKSM            ; Enter keypad shifted mode

0923    07                                      DB      7               ; ESC x 7   ESC [ > 7 h
0924    0B4E                                    DW      EKAM            ; Enter keypad alternate mode

0926    08                                      DB      8               ; ESC x 8   ESC [ > 8 h
0927    0B68                                    DW      EALF            ; Enable auto LF on CR

0929    09                                      DB      9               ; ESC x 9   ESC [ > 9 h
```

```
        1 092A    0B62                                DW      EACR                    ; Enable auto CR on LF
        2
        3         0009                    SMSTL       EQU     ($-SMST)/3              ; Table length
```

```
    1                           ;;          RMS - reset mode sequence
    2                           ;
    3                           ;           *RMS* resets the mode specified by the last character in the
    4                           ;           sequence.
    5                           ;
    6                           ;
    7                           ;           ENTRY   none
    8                           ;
    9                           ;           EXIT    none
   10                           ;
   11                           ;           USES    all
   12
   13
   14 092C   11      2A          RMS:        ACALL   SETDISP             ; Set dispatch and return
   15
   16 092E   11      2F                      ACALL   SETNORM             ; Restore dispatch address
   17 0930   C3                              CLR     C
   18 0931   94      31                      SUBB    A, #'1'             ; Subtract offset
   19 0933   40      0E                      JC      RMSRET              ; Check for lower bounds
   20 0935   B4      09      00              CJNE    A, #('9'-'0'), RMS1
   21 0938   50      09          RMS1:       JNC     RMSRET              ; Check for upper bounds
   22 093A   04                              INC     A                   ; Bump up into range
   23 093B   75      F0      09              MOV     B, #RMSTL           ; (B) = table length
   24 093E   90      0956                    MOV     DPTR, #RMST         ; (DPTR) = ESC Y Pn table
   25 0941   01      36                      AJMP    STJMP               ; Jump to any routine
   26
   27 0943   22                  RMSRET:     RET
```

```
                                ;;              A2RM - ANSI mode #2 reset mode sequence
                                ;
                                ;               *A2RM* reset the specified mode(s).
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    A, B, DPTR

0944  31  D1            A2RM:                 ACALL   GNP             ; Get next (or first) parameter
0946  40  FB                                   JC      RMSRET          ; IF parameter THEN
0948  75  F0  09                              MOV     B, #RMSTL       ; (B) = table length
094B  90  0956                                 MOV     DPTR, #RMST     ; (DPTR) = ESC [ > Pn l table
094E  31  F7                                   ACALL   XSTAB           ; Search table
0950  40  F1                                   JC      RMSRET          ; IF found THEN
0952  11  3A                                   ACALL   STCALL          ;   jump to routine
0954  80  EE                                   SJMP    A2RM            ;   and keep going


                                ;;              RMST - ANSI reset mode sequence table
                                ;
                                ;               *RMST* contains valid parameters for ESC [ > Pn l sequence.


          0956                  RMST            EQU     $

0956  01                                       DB      1               ; ESC Y 1   ESC [ > 1 l
0957  0B15                                     DW      D25L            ; Disable 25th line

0959  02                                       DB      2               ; ESC Y 2   ESC [ > 2 l
095A  0B26                                     DW      EKC             ; Enable keyboard click

095C  03                                       DB      3               ; ESC Y 3   ESC [ > 3 l
095D  0B43                                     DW      XHSM            ; Exit hold screen mode

095F  04                                       DB      4               ; ESC Y 4   ESC [ > 4 l
0960  0B37                                     DW      SUC             ; Set "underscore" cursor

0962  05                                       DB      5               ; ESC Y 5   ESC [ > 5 l
0963  0B5C                                     DW      EC              ; Enable cursor

0965  06                                       DB      6               ; ESC Y 6   ESC [ > 6 l
0966  0B4B                                     DW      XKSM            ; Exit keypad shifted mode

0968  07                                       DB      7               ; ESC Y 7   ESC [ > 7 l
0969  0B51                                     DW      XKAM            ; Exit keypad alternate mode

096B  08                                       DB      8               ; ESC Y 8   ESC [ > 8 l
096C  0B6B                                     DW      XALF            ; Disable auto LF on CR

096E  09                                       DB      9               ; ESC Y 9   ESC [ > 9 l
096F  0B65                                     DW      XACR            ; Disable auto CR on LF
```

```
                1
                2          0009                    RMSTL           EQU     ($-RMST)/3              ; Table length
```

```
                                ;;              PSD - parameter string decoder
                                ;
                                ;               *PSD* inputs a parameter string of decimal numbers separated
                                ;               by a semicolon (to a maximum of 8) until the final character
                                ;               of the escape sequence (non decimal and not a semicolon) is
                                ;               inputed.
                                ;
                                ;
                                ;               ENTRY   (A) = next parameter character
                                ;
                                ;               EXIT    (A) = last character inputed (Ps usually)
                                ;                       'C' = set if done decoding, clear otherwise
                                ;
                                ;               USES    A, B, DPTR, TEMP


0971  20  18  0E  PSD:          JB      PSDF, PSDB              ; IF first time through THEN
0974  D2  18                    SETB    PSDF                    ;   toggle flag
0976  75  1A  F8                MOV     PMADRL, #LOW (PMBUF)    ;   set parameter pointer
0979  75  19  57                MOV     PMADRH, #HIGH (PMBUF)
097C  75  1B  FF                MOV     PMNUM, #-1              ;   init number of parameters

097F  75  1C  00                MOV     PMVALUE, #0             ;   init parameter value

0982  B4  3A  00  PSDB:         CJNE    A, #'9'+1, PSDB1
0985  50  1B      PSDB1:        JNC     PSDE                    ; IF below upper range AND
0987  B4  30  00                CJNE    A, #'0', PSDB2
098A  40  16      PSDB2:        JC      PSDE                    ; IF above lower range THEN

098C  94  30                    SUBB    A, #'0'                 ;   remove ASCII offset
098E  C5  1C                    XCH     A, PMVALUE              ;   get previous value
0990  75  F0  0A                MOV     B, #10
0993  A4                        MUL     AB                      ;   value := value * 10
0994  25  1C                    ADD     A, PMVALUE              ;   value := value + newparm
0996  F5  1C                    MOV     PMVALUE, A              ;   save new parameter value
0998  E5  1B                    MOV     A, PMNUM
099A  B4  FF  03                CJNE    A, #-1, PSD1            ;   IF parmnum = -1 THEN
099D  75  1B  00                MOV     PMNUM, #0               ;     parmnum := 0
09A0  C3          PSD1:         CLR     C                       ;   flag = still parsing string
09A1  22                        RET

09A2  85  1A  82  PSDE:         MOV     DPL, PMADRL             ; ELSE set parameter address
09A5  85  19  83                MOV     DPH, PMADRH
09A8  FD                        MOV     TEMP, A                 ;   save character temporary
09A9  E5  1C                    MOV     A, PMVALUE              ;   get current parameter value
09AB  F0                        MOVX    @DPTR, A                ;   save in parameter buffer
09AC  A3                        INC     DPTR                    ;   bump parameter address
09AD  E5  82                    MOV     A, DPL                  ;   set low byte address to
09AF  B4  00  05                CJNE    A, #LOW(PMBUF+8), PSD2  ;   IF overflow THEN
09B2  75  1B  08                MOV     PMNUM, #8               ;     give max number of parameters
09B5  80  08                    SJMP    PSD3                    ;     and skip
09B7  85  82  1A  PSD2:         MOV     PMADRL, DPL             ;   ELSE replace
09BA  85  83  19                MOV     PMADRH, DPH             ;     parameter address
09BD  05  1B                    INC     PMNUM                   ;     bump number of parameters
09BF  ED          PSD3:         MOV     A, TEMP                 ;   restore original character
09C0  B4  3B  0C                CJNE    A, #';', PSDRET         ;   IF ';' THEN
09C3  75  1C  00                MOV     PMVALUE, #0             ;     reset parameter value
```

```
     1 09C6  AD      1B                              MOV     TEMP, PMNUM
     2 09C8  BD      00      02                      CJNE    TEMP, #0, PSD4          ;   check for ';' as first char
     3 09CB  05      1B                              INC     PMNUM                   ;   IF so THEN adjust number
     4 09CD  C3                      PSD4:           CLR     C                       ;   flag = still parsing string
     5 09CE  22                                      RET
     6 09CF  D3                      PSDRET:         SETB    C                       ; Flag = finished parsing
     7 09D0  22                                      RET
```

```
                                        ;;              GNP - set next parameter
                                        ;
                                        ;               *GNP* sets the next (or first) parameter from the parameter
                                        ;               buffer.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    (A) = next (or first) parameter, 0 if no more
                                        ;                       'C' = set if no parameters left, clear otherwise
                                        ;
                                        ;               USES    A, DPTR

09D1 20 19 08           GNP:            JB      GNPF, GNP1              ; IF first time since ESC THEN
09D4 D2 19                              SETB    GNPF                    ;   toggle flag
09D6 75 1A F8                           MOV     PMADRL, #LOW (PMBUF)    ;   set parameter address
09D9 75 19 57                           MOV     PMADRH, #HIGH (PMBUF)

09DC 85 1A 82           GNP1:           MOV     DPL, PMADRL             ; Get pointer to next parameter
09DF 85 19 83                           MOV     DPH, PMADRH
09E2 E5 1B                              MOV     A, PMNUM                ; Get number of parameters 0..7
09E4 70 02                              JNZ     GNP2                    ; IF no more or none at all
09E6 D3                                 SETB    C                       ;   'C' = no more left
09E7 22                                 RET
09E8 E0                 GNP2:           MOVX    A, @DPTR                ; ELSE get parameter
09E9 A3                                 INC     DPTR                    ;   bump pointer
09EA 85 82 1A                           MOV     PMADRL, DPL             ;   and save
09ED 85 83 19                           MOV     PMADRH, DPH
09F0 15 1B                              DEC     PMNUM                   ;   dec number of parameters
09F2 C3                                 CLR     C                       ;   'C' = parameter available
09F3 22                                 RET
```

```
                                ;;              XBLINAD - build line address
                                ;
                                ;               *XBLINAD* is a jump to the routine *BLINAD*.  It is used to
                                ;               save bytes so that routines may use a short jump to here
                                ;               instead of a long jump to the other routine.


09F4    02      04EA            XBLINAD:        LJMP    BLINAD


                                ;;              XSTAB - search table
                                ;
                                ;               *XSTAB* is a jump to the routine *STAB*.  It is used to
                                ;               save bytes so that routines may use a short jump to here
                                ;               instead of a long jump to the other routine.

09F7    02      013A            XSTAB:          LJMP    STAB
```

```
                                ;;              APCA - ANSI perform cursor addressing
                                ;
                                ;               *APCA* checks to see that the line and column values are legal
                                ;               and sets the cursor to the requested address.  If the values
                                ;               are out of range, the old line number remains and/or the cursor
                                ;               goes to the last column on the line.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all


09FA 31      D1                 APCA:           ACALL   GNP                     ; Get next parameter
09FC 60      0A                                  JZ      ACA2                    ; IF non-zero THEN
09FE 14                                          DEC     A                       ;   bring into range 0..24
09FF B4      18      04                          CJNE    A, #MAXLINE, ACA1       ;   check range
0A02 30      15      05                          JNB     L25EN, ACA3             ;   IF line 25 disbld THEN skip
0A05 D3                                          SETB    C                       ;   trick ckeck below
0A06 50      02                 ACA1:           JNC     ACA3                    ;   IF in range THEN
0A08 F5      12                 ACA2:           MOV     YPOS, A                 ;     set Y-position

0A0A 31      D1                 ACA3:           ACALL   GNP                     ; Get next parameter
0A0C 60      08                                  JZ      ACA5                    ; IF non-zero THEN
0A0E 14                                          DEC     A                       ;   bring into range 0..79
0A0F B4      50      00                          CJNE    A, #MAXCHAR+1, ACA3A    ;   check range
0A12 40      02                 ACA3A:          JC      ACA5                    ;   IF not in range THEN
0A14 74      4F                                  MOV     A, #MAXCHAR             ;     go to end of line
0A16 F5      11                 ACA5:           MOV     XPOS, A                 ; Set X-position
0A18 21      F4                                  AJMP    XBLINAD                 ; Update and return
```

*** Z-29 COMPUTER TERMINAL FIRMWARE VERSION 1.03 ***

```
                                ;;              EID - erase in display
                                ;
                                ;               *EID* erases the amount of the screen specified by Pn.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

0A1A  31    D1                  EID:            ACALL   GNP             ; Get parameter
0A1C  70    02                                  JNZ     EID1            ; IF zero THEN
0A1E  81    08                                  AJMP    EED             ;   erase to eos and return

0A20  14                        EID1:           DEC     A
0A21  70    02                                  JNZ     EID2            ; IF one THEN
0A23  61    F1                                  AJMP    EBD             ;   erase from beginning & ret

0A25  14                        EID2:           DEC     A
0A26  70    02                                  JNZ     EIR             ; IF two THEN
0A28  61    AC                                  AJMP    CLRS            ;   erase screen and return

0A2A  22                        EIR:            RET
```

```
                                ;;              EIL - erase in line
                                ;
                                ;               *EIL* erases the portion of the current line as specified
                                ;               by Pn.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all


0A2B   31      D1               EIL:            ACALL   GNP             ; Get parameter
0A2D   70      02                               JNZ     EIL1            ; IF zero THEN
0A2F   61      E0                               AJMP    EOL             ;   erase to eol and return

0A31   14                       EIL1:           DEC     A
0A32   70      02                               JNZ     EIL2            ; IF one THEN
0A34   61      CB                               AJMP    EBL             ;   erase from beginning & ret

0A36   14                       EIL2:           DEC     A
0A37   70      F1                               JNZ     EIR             ; IF two THEN
0A39   61      DB                               AJMP    EEL             ;   erase entire line and ret
```

```
                                ;;      ASGM - ANSI set graphics mode
                                ;
                                ;       *ASGM* sets or resets any or all of the attributes and
                                ;       graphics modes.
                                ;
                                ;
                                ;       ENTRY   none
                                ;
                                ;       EXIT    none
                                ;
                                ;       USES    all
                
0A3B  31      D1                ASGM:   ACALL   GNP                     ; Get first parameter

0A3D  75      F0      07        ASG1:   MOV     B, #ASGMTL              ; (B) = table length
0A40  90      0A50              ASG1:   MOV     DPTR, #ASGMT            ; (DPTR) = set mode table
0A43  31      F7                        ACALL   XSTAB                   ; Search table
0A45  40      02                        JC      ASGO                    ; IF found THEN
0A47  51      4E                        ACALL   ASGO                    ;   call the routine
0A49  31      D1                ASGO:   ACALL   GNP                     ; Get next parameter
0A4B  50      F0                        JNC     ASG1                    ; IF done THEN return
0A4D  22                                RET                             ; ELSE keep looping

0A4E  E4                        ASGO:   CLR     A
0A4F  73                                JMP     @A+DPTR                 ; Jump to routine
```

```
                                ;;              ASGMT - ANSI set graphics mode table
                                ;
                                ;               *ASGMT* contains the address of the routines which set the
                                ;               requested attribute or graphic mode.
                                ;
                                ;               Table entries are:  The third character of the ANSi escape
                                ;               sequence followed by the address of the routine which performs
                                ;               the requested function.

                0A50            ASGMT           EQU     $

0A50    00                                      DB      0                       ; ESC [ 0 m
0A51    0F26                                    DW      XATR                    ; Exit all attributes

0A53    02                                      DB      2                       ; ESC [ 2 m
0A54    0F2A                                    DW      EHALF                   ; Enter half intensity mode

0A56    04                                      DB      4                       ; ESC [ 4 m
0A57    0F2D                                    DW      EUNDL                   ; Enter underline mode

0A59    05                                      DB      5                       ; ESC [ 5 m
0A5A    0F30                                    DW      EBLNK                   ; Enter blink mode

0A5C    07                                      DB      7                       ; ESC [ 7 m
0A5D    0F33                                    DW      ERVM                    ; Enter reverse video mode

0A5F    0A                                      DB      10                      ; ESC [ 10 m
0A60    0F39                                    DW      EGM                     ; Enter graphics mode

0A62    0B                                      DB      11                      ; ESC [ 11 m
0A63    0F3C                                    DW      XGM                     ; Exit graphics mode

                0007            ASGMTL          EQU     ($-ASGMT)/3             ; Table length
```

```
                                        ;;              ASPF - ANSI set protected fields
                                        ;
                                        ;               *ASPF* selects the attributes which imply protection.  Multiple
                                        ;               attributes may be selected in a single sequence.  This sequence
                                        ;               does not change any attributes of the characters displayed on
                                        ;               the screen or of the characters received.  The sequence merely
                                        ;               changes the way the characters with the specified attributes
                                        ;               are interpreted by the different modes.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    all

0A65  75  F0  00      ASPF:             MOV     B, #0                   ; Initialize byte

0A68  31  D1          ASPF1:            ACALL   GNP                     ; Get next parameter
0A6A  40  22                            JC      ASPF2                   ; Exit when done
0A6C  B4  00  03                        CJNE    A, #0, ASPF1A
0A6F  75  F0  00                        MOV     B, #0                   ; None protected
0A72  B4  02  02      ASPF1A:           CJNE    A, #2, ASPF1B
0A75  D2  F3                            SETB    B.3                     ; Half intensity
0A77  B4  04  02      ASPF1B:           CJNE    A, #4, ASPF1C
0A7A  D2  F1                            SETB    B.1                     ; Under line
0A7C  B4  05  02      ASPF1C:           CJNE    A, #5, ASPF1D
0A7F  D2  F2                            SETB    B.2                     ; Blink
0A81  B4  07  02      ASPF1D:           CJNE    A, #7, ASPF1E
0A84  D2  F0                            SETB    B.0                     ; Reverse video
0A86  B4  FE  03      ASPF1E:           CJNE    A, #254, ASPF1F
0A89  75  F0  80                        MOV     B, #10000000B           ; No attributes are protected
0A8C  80  DA          ASPF1F:           SJMP    ASPF1

0A8E  85  F0  3F      ASPF2:            MOV     PFIELD, B               ; Final change
0A91  22                                RET
```

```
                                ;;              CBA - convert byte to ASCII
                                ;
                                ;               *CBA* converts a byte into its ASCII equivalent.
                                ;
                                ;
                                ;               ENTRY   (A) = byte to convert
                                ;
                                ;               EXIT    (B) = hundreds digit
                                ;                       (DPH) = tens digit
                                ;                       (DPL) = ones digit
                                ;
                                ;               USES    A, B, DPTR

0A92  75  F0  64        CBA:    MOV     B, #100
0A95  84                        DIV     AB
0A96  C0  E0                    PUSH    ACC                     ; Save hundreds amount
0A98  E5  F0                    MOV     A, B
0A9A  75  F0  0A                MOV     B, #10
0A9D  84                        DIV     AB                      ; (A) = tens  (B) = ones
0A9E  24  30                    ADD     A, #'0'                 ; Make tens digit ASCII
0AA0  F5  83                    MOV     DPH, A                  ; Save it
0AA2  E5  F0                    MOV     A, B
0AA4  24  30                    ADD     A, #'0'                 ; Make ones digit ASCII
0AA6  F5  82                    MOV     DPL, A                  ; Save it
0AA8  D0  E0                    POP     ACC
0AAA  24  30                    ADD     A, #'0'                 ; Make hundreds digit ASCII
0AAC  F5  F0                    MOV     B, A                    ; Save it
0AAE  22                        RET
```

```
                                ;;              ACPR - ANSI cursor position report
                                ;
                                ;               *ACPR* outputs the current cursor position in the ANSI form
                                ;               of  ESC [ Pl ; Pc R     Please note the tricky code to save a
                                ;               few bytes of code.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

0AAF  31      D1                ACPR:           ACALL   GNP                     ; Get parameter
0AB1  B4      06      35                        CJNE    A, #6, APR              ; IF not 6 THEN return
0AB4  74      1B                                MOV     A, #ESC
0AB6  51      CC                                ACALL   ACP3                    ; Output an ESC
0AB8  74      5B                                MOV     A, #'['
0ABA  51      CC                                ACALL   ACP3                    ; Output a '['
0ABC  E5      12                                MOV     A, YPOS
0ABE  04                                        INC     A                       ; Bump to normalize
0ABF  51      CF                                ACALL   ACP1                    ; Output Y-position
0AC1  74      3B                                MOV     A, #';'
0AC3  51      CC                                ACALL   ACP3                    ; Output seperator
0AC5  E5      11                                MOV     A, XPOS
0AC7  04                                        INC     A                       ; Bump to normalize
0AC8  51      CF                                ACALL   ACP1                    ; Output X-position
0ACA  74      52                                MOV     A, #'R'

0ACC  02      022A              ACP3:           LJMP    PCOF                    ; Output 'R' and return

0ACF  51      92                ACP1:           ACALL   CBA                     ; Convert position to ASCII
0AD1  AD      82                                MOV     TEMP, DPL               ; Save one's digit
0AD3  E5      83                                MOV     A, DPH                  ; Get ten's digit
0AD5  B4      31      00                        CJNE    A, #'1', ACP1A
0AD8  40      02                ACP1A:          JC      ACP2                    ; IF not zero THEN
0ADA  51      CC                                ACALL   ACP3                    ;   output ten's digit
0ADC  ED                        ACP2:           MOV     A, TEMP
0ADD  80      ED                                SJMP    ACP3                    ; Output one's digit and return
```

```
                                ;;              APRNT - ANSI print page
                                ;
                                ;               *APRNT* prints the page to the opposite port.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

0ADF    11      2A              APRNT:          ACALL   SETDISP                 ; Set dispatch and return

0AE1    11      2F                              ACALL   SETNORM                 ; Restore dispatch address
0AE3    B4      37      03                      CJNE    A, #'7', APR            ; IF ESC # 7 THEN
0AE6    02      103C                            JMP     R2_PRNT                 ;   print page
0AE9    22                      APR:            RET


                                ;;              ASBR - ANSI set baud rate
                                ;
                                ;               *ASBR* sets the baud rate to that specified by Pn.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

0AEA    31      D1              ASBR:           ACALL   GNP                     ; Get parameter
0AEC    50      01                              JNC     ASBR1                   ; IF no parameter THEN
0AEE    04                                      INC     A                       ;   default to 1
0AEF    B4      0E      00      ASBR1:          CJNE    A, #14, ASBR2
0AF2    50      F5              ASBR2:          JNC     APR                     ; IF in range (0..13) THEN
0AF4    C1      4B                              AJMP    SBR                     ;   set baud rate and return
```

```
                                ;;              ARAMP - ANSI reset all modes to power up configuration
                                ;
                                ;               RAMP - reset all modes to power up configuration
                                ;
                                ;
                                ;               *ARAMP* and *RAMP* resets all flags etc. to the configuration
                                ;               on power up.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

0AF6    31      D1              ARAMP:          ACALL   GNP             ; Get any parameter
0AF8    50      EF                              JNC     APR             ; IF no parameters THEN
0AFA    02      1003            RAMP:           JMP     R2_IN2          ;   reset and return
```

```
                                ;;              WEOL - wrap at end of line
                                ;
                                ;               *WEOL* sets the flag which causes the terminal to perform
                                ;               a carriage return and line feed immediatly after the 80th
                                ;               character before a line is displayed.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    none

0AFD    D2      2B              WEOL:           SETB    AUTOWRAP                ; Flag auto wrap
0AFF    22                                      RET


                                ;;              DEOL - discard at end of line
                                ;
                                ;               *DEOL* resets the wrap around at end of line flag.
                                ;               characters past column 79 are placed in column 79 until
                                ;               a carriage return is received.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    none

0B00    C2      2B              DEOL:           CLR     AUTOWRAP                ; Clear auto wrap
0B02    22                                      RET
```

```
                                        ;;              EAM - enter ANSI mode
                                        ;
                                        ;               *EAM* places the terminal in the ANSI mode for escape codes.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    none

0B03    75      28      01      EAM:    MOV     MODE, #00000001B        ; Place in ANSI mode
0B06    22                              RET


                                        ;;              EZM - enter ZDS mode
                                        ;
                                        ;               *EZM* takes the terminal out of ANSI mode and into ZDS mode.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    none

0B07    75      28      02      EZM:    MOV     MODE, #00000010B        ; Take out of ANSI mode
0B0A    22                      RT1:    RET
```

```
                                        ;;              E25L - enable 25th line
                                        ;
                                        ;               *E25L* enables the display of the 25th line only if it has
                                        ;               not been previously enabled.  The 25th line is cleared at
                                        ;               the time it is enabled.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    A, B, DPTR, TEMP, PTR1

 0B0B   20      15      FC      E25L:   JB      L25EN, RT1              ; IF enabled THEN return
 0B0E   D2      15                      SETB    L25EN                   ; ELSE enable 25th line
 0B10   75      F0      18              MOV     B, #MAXLINE
 0B13   61      96                      AJMP    CLRLINE                 ; Clear 25th line


                                        ;;              D25L - disable 25th line
                                        ;
                                        ;               *D25L* disables the display of the 25th line and clears it.
                                        ;               If the cursor is on the 25th line then it is sent to the
                                        ;               home position.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    A, B, DPTR, TEMP, PTR1

 0B15   C2      15              D25L:   CLR     L25EN                   ; Disable 25th line
 0B17   75      F0      18              MOV     B, #MAXLINE
 0B1A   71      96                      ACALL   CLRLINE                 ; Clear 25th line
 0B1C   E5      12                      MOV     A, YPOS
 0B1E   B4      18      02              CJNE    A, #MAXLINE, XR2_DISP   ; IF cursor on 25th line THEN
 0B21   91      36                      ACALL   SCH                     ;   set cursor home

 0B23   02      1042            XR2_DISP:       LJMP    R2_DISP         ; Display any status
```

*** Z-29 COMPUTER TERMINAL FIRMWARE VERSION 1.03 ***

```
                                ;;              EKC - enable keyboard click
                                ;
                                ;               *EKC* resets the flag that disables keyboard click and
                                ;               outputs the command to the keyboard.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

0B26    D2      33              EKC:            SETB    CLKF                    ; Turn on key click
0B28    74      82                              MOV     A, #KBC_EC
0B2A    02      026D            XPCKB:          JMP     PCKB                    ; Output command and return


                                ;;              NKC - no keyboard click
                                ;
                                ;               *NKC* sets the flag that disables keyboard click and
                                ;               outputs the command to the keyboard.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

0B2D    C2      33              NKC:            CLR     CLKF                    ; Turn off key click
0B2F    74      83                              MOV     A, #KBC_DC
0B31    80      F7                              SJMP    XPCKB                   ; Output command and return
```

```
                                ;;              SBC - set block cursor
                                ;
                                ;               *SBC* programs the CRTC to display a block cursor.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    A, B, DPTR


0B33    C2      32              SBC:            CLR     CRUL                    ; Clear underline flag
0B35    80      02                              SJMP    SUC1                    ; Initialize cursor format


                                ;;              SUC - set underscore cursor
                                ;
                                ;               *SUC* programs the CRTC to display an underline cursor.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    A, B, DPTR


0B37    D2      32              SUC:            SETB    CRUL                    ; Set underline flag

0B39    02      101B            SUC1:           JMP     R2_ICUR                 ; Initialize cursor format
```

*** Z-29 COMPUTER TERMINAL FIRMWARE VERSION 1.03 ***

```
                                ;;              EHSM - enter hold screen mode
                                ;
                                ;               *EHSM* sets the hold screen mode flag.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    none

0B3C    D2      36              EHSM:           SETB    HSMODEF                 ; Enter mode
0B3E    75      1E      01                      MOV     HSLINE, #1              ; Set number of lines
0B41    80      02                              SJMP    XHSM1                   ; Clear stop display and return


                                ;;              XHSM - exit hold screen mode
                                ;
                                ;               *XHSM* clear hold screen mode flag.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    none

0B43    C2      36              XHSM:           CLR     HSMODEF                 ; Exit mode

0B45    C2      22              XHSM1:          CLR     HSSTOPF                 ; Clear stop display flag
0B47    22                                      RET
```

```
                                ;;              EKSM - enter keypad shifted mode
                                ;
                                ;               *EKSM* places the numeric keypad into the shifted mode.
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    none

0B48    D2      30              EKSM:           SETB    KPADSHTF                ; Set keypad shifted
0B4A    22                                      RET


                                ;;              XKSM - exit keypad shifted mode
                                ;
                                ;               *XKSM* takes the numeric keypad out of the shifted mode.
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    none

0B4B    C2      30              XKSM:           CLR     KPADSHTF                ; Clear keypad shifted
0B4D    22                                      RET
```

```
                                ;;              EKAM - enter keypad alternate mode
                                ;
                                ;               *EKAM* places the numeric keypad into the alternate mode.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    none

0B4E   D2       31              EKAM:           SETB    KPADALTF                ; Set keypad alternate
0B50   22                                       RET


                                ;;              XKAM - exit keypad alternate mode
                                ;
                                ;               *XKAM* takes the numeric keypad out of the alternate mode.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    none

0B51   C2       31              XKAM:           CLR     KPADALTF                ; Clear keypad alternate
0B53   22                                       RET
```

```
                                        ;;              EICM - enter insert character mode
                                        ;
                                        ;               *EICM* places terminal into insert character mode.
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    all
                                        ;
0B54    D2      21              EICM:   SETB    ICMODEF                 ; Set insert mode flag
0B56    61      23                      AJMP    XR2_DISP                ; Display any change in status
                                        ;;              XICM - exit insert character mode
                                        ;
                                        ;               *XICM* takes terminal out of the insert character mode.
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    all
                                        ;
0B58    C2      21              XICM:   CLR     ICMODEF                 ; Clear insert mode flag
0B5A    61      23                      AJMP    XR2_DISP                ; Display any change in status
```

```
                                ;;              EC - enable cursor
                                ;
                                ;               *EC* enables the display of the cursor.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    none

OB5C    D2      1A              EC:             SETB    ENBLCUR                 ; Enable display of cursor
OB5E    22                                      RET


                                ;;              DC - disable cursor
                                ;
                                ;               *DC* disables the display of the cursor
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    none

OB5F    C2      1A              DC:             CLR     ENBLCUR                 ; Disable display of cursor
OB61    22                                      RET
```

```
                                ;;              EACR - enable auto carriage return
                                ;
                                ;               *EACR* enables the terminal to perform an automatic carriage
                                ;               return upon receipt of a line feed character.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    none

0B62    D2       2C             EACR:           SETB    AUTOCR                  ; Enter auto CR mode
0B64    22                                       RET

                                ;;              XACR - exit auto carriage return
                                ;
                                ;               *XACR* clears the auto carriage return flag.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    none

0B65    C2       2C             XACR:           CLR     AUTOCR                  ; Exit auto CR mode
0B67    22                                       RET
```

```
                                ;;              EALF - enable auto line feed
                                ;
                                ;               *EALF* enables the terminal to perform an automatic
                                ;               line feed upon receipt of a carriage return.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    none

0B68    D2      2D              EALF:           SETB    AUTOLF                  ; Enter auto LF mode
0B6A    22                                      RET


                                ;;              XALF - exit auto line feed mode
                                ;
                                ;               *XALF* clears the auto line feed mode flag.
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    none

0B6B    C2      2D              XALF:           CLR     AUTOLF                  ; Exit auto LF mode
0B6D    22                                      RET
```

```
                                ;;              EGATM - enter guarded area transfer mode
                                ;
                                ;               *EGATM* places the terminal in a mode such that all data
                                ;               is to be transmitted in a data stream such as transmit page.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    none


0B6E    C2      25              EGATM:          CLR     GATM                    ; Clear guarded area transfer
0B70    22                                      RET


                                ;;              XGATM - exit guarded area transfer mode
                                ;
                                ;               *XGATM* places the terminal in a mode such that only
                                ;               unprotected data is to be transmitted in a data stream
                                ;               such as transmit page.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    none


0B71    D2      25              XGATM:          SETB    GATM                    ; Set guarded area transfer
0B73    22                                      RET
```

```
                                ;;              EERM - enter erasure mode
                                ;
                                ;               *EERM* places the terminal in a mode such that all data is
                                ;               erased in functions such as erase screen.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    none

0B74    C2      1B              EERM:           CLR     ERM                     ; Clear erasure mode
0B76    22                                      RET


                                ;;              XERM - exit erasure mode
                                ;
                                ;               *ZERM* places the terminal in a mode such that only
                                ;               unprotected data is erased in functions such as erase
                                ;               screen.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    none

0B77    D2      1B              XERM:           SETB    ERM                     ; Set erasure mode
0B79    22                                      RET
```

```
                                        ;;              PCLK - program clock
                                        ;
                                        ;               *PCLK* sets the internal clock.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    A, DPTR

0B7A   31      D1                       PCLK:           ACALL   GNP                     ; Get hour parameter
0B7C   B4      18      00                               CJNE    A, #24, PCL1A
0B7F   50      14                       PCL1A:          JNC     PCL2                    ; IF bad parameter THEN exit
0B81   F5      3D                                       MOV     THOUR, A
0B83   31      D1                                       ACALL   GNP                     ; Get minutes parameter
0B85   B4      3C      00                               CJNE    A, #60, PCL1B
0B88   50      0B                       PCL1B:          JNC     PCL2                    ; IF bad parameter THEN exit
0B8A   F5      3C                                       MOV     TMIN, A
0B8C   31      D1                                       ACALL   GNP                     ; Get seconds parameter
0B8E   B4      3C      00                               CJNE    A, #60, PCL1C
0B91   50      02                       PCL1C:          JNC     PCL2                    ; IF bad parameter THEN exit
0B93   F5      3B                                       MOV     TSEC, A
0B95   22                               PCL2:           RET
```

```
                                ;;              CLRLINE - clear line completely
                                ;
                                ;               *CLRLINE* writes spaces into the entire line indicated and
                                ;               ignores any setting of protected fields.
                                ;
                                ;
                                ;               ENTRY   (B) = line number to erase
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    A, B, DPTR, TEMP, PTR1, WORK2, INDX2


0B96    C0      3F              CLRLINE:        PUSH    PFIELD                  ; Save protected fields
0B98    75      3F      00                      MOV     PFIELD, #0              ; Make everything unprotected
0B9B    71      A0                              ACALL   CLRL                    ; Clear line
0B9D    D0      3F                              POP     PFIELD                  ; Restore protected fields
0B9F    22                                      RET


                                ;;              CLRL - clear line
                                ;
                                ;               *CLRL* writes spaces into the entire line indicated.
                                ;
                                ;
                                ;               ENTRY   (B) = line number to erase
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    A, B, DPTR, TEMP, PTR1, WORK2, INDX2


0BA0    E4                      CLRL:           CLR     A
0BA1    12      0503                            LCALL   CLA                     ; Calculate address
0BA4    74      50                              MOV     A, #NUMCHARS            ; 80 characters
0BA6    75      F0      00                      MOV     B, #0                   ; No attributes
0BA9    02      100C                            LJMP    R2_FILL                 ; Fill with spaces and return
```

```
                                        ;;              CLRS - clear screen
                                        ;
                                        ;               *CLRS* clears entire display except the 25th line.  IF cursor
                                        ;               is on the 25th line then just the 25th line is cleared.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    all


0BAC    E5      12              CLRS:   MOV     A, YPOS
0BAE    B4      18      04              CJNE    A, #MAXLINE, CS1        ; IF ypos = 25th line THEN
0BB1    F5      F0                      MOV     B, A                    ;   setup for subroutine
0BB3    80      EB                      SJMP    CLRL                    ;   clear line and return

0BB5    7C      18              CS1:    MOV     WORK, #NUMLINES-1       ; Get number of lines
0BB7    7A      00                      MOV     INDX1, #0               ; Index for -LORDER-

0BB9    8A      F0              CS2:    MOV     B, INDX1                ; REPEAT setup for call
0BBB    71      A0                      ACALL   CLRL                    ;   to clear a line
0BBD    0A                              INC     INDX1                   ;   bump indx1
0BBE    DC      F9                      DJNZ    WORK, CS2               ; UNTIL work = 0

0BC0    81      36                      AJMP    SCH                     ; Set cursor home and return


                                        ;;              CLFO - clear foreground
                                        ;
                                        ;               *CLFO* clears entire display except for half intensity spaces.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    all


0BC2    75      3F      08      CLFO:   MOV     PFIELD, #00001000B      ; Make half intensity protected
0BC5    71      AC                      ACALL   CLRS                    ; Clear screen
0BC7    75      3F      00              MOV     PFIELD, #0              ; Make nothing protected
0BCA    22                              RET
```

```
                                ;;              EBL - erase from beginning of line
                                ;
                                ;               *EBL* erases from the beginning of the current line to the
                                ;               cursor.  Cursor position does not change.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    A, B, DPTR, TEMP, PTR1, WORK2, INDX2

0BCB  E4                        EBL:            CLR     A
0BCC  85      12      F0                        MOV     B, YPOS
0BCF  12      0503                              LCALL   CLA                     ; Calc address for (0,YPOS)
0BD2  E5      11                                MOV     A, XPOS                 ; Get number of characters
0BD4  04                                        INC     A                       ;   to erase
0BD5  75      F0      00                        MOV     B, #0                   ; Clear attributes
0BD8  02      100C                              LJMP    R2_FILL                 ; Clear and return


                                ;;              EEL - erase entire line
                                ;
                                ;               *EEL* clears the entire line where the cursor resides.
                                ;               Cursor position does not change.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    A, B, DPTR, TEMP, PTR1, WORK2, INDX2

0BDB  85      12      F0        EEL:            MOV     B, YPOS                 ; Get current position
0BDE  80      C0                                SJMP    CLRL                    ; Clear line and return
```

```
                                ;;              EOL - erase to end of line
                                ;
                                ;               *EOL* erases from the cursor to the end of the current line.
                                ;               Cursor position does not change.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    A, B, DPTR, TEMP, PTR1, WORK2, INDX2


0BE0    75      F0      00      EOL:            MOV     B, #0                   ; Clear attributes

0BE3    74      50              EOL1:           MOV     A, #NUMCHARS            ; Calculate number to erase
0BE5    C3                                      CLR     C
0BE6    95      11                              SUBB    A, XPOS
0BE8    85      14      82                      MOV     DPL, LINADL             ; Get line address
0BEB    85      13      83                      MOV     DPH, LINADH
0BEE    02      100C                            LJMP    R2_FILL                 ; Erase to end and return


                                ;;              EBD - erase from beginning of display
                                ;
                                ;               *EBD* clears the screen from 'HOME' position to the current
                                ;               cursor position.  The exception is the 25th line.  Cursor
                                ;               position does not change.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    A, B, DPTR, TEMP, PTR1, INDX1, WORK2, INDX2


0BF1    E5      12              EBD:            MOV     A, YPOS
0BF3    B4      18      02                      CJNE    A, #MAXLINE, EB1        ; IF 25th line THEN
0BF6    80      D3                              SJMP    EBL                     ;   erase just this line

0BF8    7A      00              EB1:            MOV     INDX1, #0               ; Start at top
0BFA    E5      12              EB2:            MOV     A, YPOS                 ; REPEAT
0BFC    B5      02      02                      CJNE    A, INDX1A, EB3          ;   IF index = current THEN
0BFF    80      CA                              SJMP    EBL                     ;     erase up to cursor

0C01    8A      F0              EB3:            MOV     B, INDX1                ;   ELSE set up and
0C03    71      A0                              ACALL   CLRL                    ;     clear entire line
0C05    0A                                      INC     INDX1                   ;     bump index counter
0C06    80      F2                              SJMP    EB2                     ; UNTIL forever
```

```
                                ;;              EED - erase to end of display
                                ;
                                ;               *EED* clears the screen from the current cursor position to
                                ;               the end of the screen.  Cursor position does not change.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    A, B, DPTR, TEMP, PTR1, INDX1, WORK2, INDX2


0C08    71      E0              EED:            ACALL   EOL                     ; Clear to end of line
0C0A    AA      12                              MOV     INDX1, YPOS             ; Set loop counter
0C0C    0A                      EE1:            INC     INDX1                   ; REPEAT bump to next line
0C0D    BA      18      00                      CJNE    INDX1, #MAXLINE, EE1A
0C10    50      06              EE1A:           JNC     EER                     ;   IF end of disp THEN exit
0C12    8A      F0                              MOV     B, INDX1                ;   get line to clear
0C14    71      A0                              ACALL   CLRL                    ;   clear line
0C16    80      F4                              SJMP    EE1                     ; UNTIL forever
0C18    22                      EER:            RET


                                ;;              HEED - Hazeltine erase to end of display (with background)
                                ;
                                ;               *HEED* clears the screen from the current cursor position
                                ;               to the end of the screen with half intensity spaces.  The
                                ;               cursor position does not change.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    A, B, DPTR, TEMP, PTR1, INDX1, WORK2, INDX2

0C19    75      F0      08      HEED:           MOV     B, #00001000B           ; Get half intensity
0C1C    71      E3                              ACALL   EOL1                    ; Clear to end of line

0C1E    AA      12                              MOV     INDX1, YPOS             ; Set loop counter
0C20    0A                      HEE1:           INC     INDX1                   ; REPEAT bump to next line
0C21    BA      18      00                      CJNE    INDX1, #MAXLINE, HEE1A
0C24    50      F2              HEE1A:          JNC     EER                     ;   IF end of disp THEN exit
0C26    E4                              CLR     A
0C27    8A      F0                              MOV     B, INDX1                ;   get line to clear
0C29    12      0503                            LCALL   CLA                     ;   calculate line address
0C2C    74      50                              MOV     A, #NUMCHARS            ;   complete line
0C2E    75      F0      08                      MOV     B, #00001000B           ;   set half intensity
0C31    12      100C                            LCALL   R2_FILL                 ;   fill with spaces
0C34    80      EA                              SJMP    HEE1                    ; UNTIL forever
```

```
                                ;;              SCH - set cursor home
                                ;
                                ;               *SCH* places the cursor at line zero, column zero.
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    A, DPTR, PTR1

0C36    E4                      SCH:            CLR     A
0C37    F5      11                              MOV     XPOS, A                 ; Zero X-position
0C39    F5      12                              MOV     YPOS, A                 ; Zero Y-position
0C3B    21      F4                              AJMP    XBLINAD                 ; Update address and return


                                ;;              PBS - perform back space
                                ;
                                ;               CLFT - cursor left
                                ;
                                ;               *PBS* - *CLFT* steps the cursor one position to the left, but
                                ;               does not wrap around to the previous line after reaching
                                ;               column zero.
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    A, DPTR, PTR1

        0C3D                    PBS             EQU     $
        0C3D                    CLFT            EQU     $
0C3D    E5      11                              MOV     A, XPOS                 ; Get X-position
0C3F    70      01                              JNZ     PBS1                    ; IF -XPOS- is zero THEN
0C41    22                                      RET                             ;   return

0C42    15      11              PBS1:           DEC     XPOS                    ; ELSE decrement -XPOS-
0C44    21      F4                              AJMP    XBLINAD                 ;   build address and return
```

```
  1                                     ;;              ACLFT - ANSI cursor left
  2                                     ;
  3                                     ;               *ACLFT* moves the cursor towards the beginning of a line
  4                                     ;               the specified Pn number of times.
  5                                     ;
  6                                     ;
  7                                     ;               ENTRY   none
  8                                     ;
  9                                     ;               EXIT    none
 10                                     ;
 11                                     ;               USES    A, DPTR, INDX1, PTR1
 12
 13
 14 0C46    31      D1                  ACLFT:          ACALL   GNP                     ; Get parameter
 15 0C48    70      01                                  JNZ     ACLO                    ; IF no parameters or 0 THEN
 16 0C4A    04                                          INC     A                       ;   bump to 1
 17
 18 0C4B    FA                          ACLO:           MOV     INDX1, A                ; Set up index
 19 0C4C    91      3D                  ACL:            ACALL   CLFT                    ; REPEAT move cursor left one
 20 0C4E    DA      FC                                  DJNZ    INDX1, ACL              ; UNTIL count = 0
 21 0C50    22                                          RET
 22
 23
 24
 25
 26                                     ;;              HCLFT - Hazeltine cursor left
 27                                     ;
 28                                     ;               *HCLFT* steps the cursor one position to the left and wraps
 29                                     ;               around to the previous line after reaching column zero.
 30                                     ;
 31                                     ;
 32                                     ;               ENTRY   none
 33                                     ;
 34                                     ;               EXIT    none
 35                                     ;
 36                                     ;               USES    A, DPTR, PTR1, TEMP
 37
 38
 39 0C51    E5      11                  HCLFT:          MOV     A, XPOS                 ; Get X-position
 40 0C53    70      E8                                  JNZ     CLFT                    ; IF not zero THEN use *CLFT*
 41 0C55    AD      12                                  MOV     TEMP, YPOS              ; Save Y-position
 42 0C57    91      8D                                  ACALL   CUP                     ; Move cursor up
 43 0C59    E5      12                                  MOV     A, YPOS
 44 0C5B    B5      05      01                          CJNE    A, TEMPA, HCLFT1        ; IF YPOS did not change THEN
 45 0C5E    22                                          RET                             ;   do nothing
 46
 47 0C5F    75      11      4F          HCLFT1:         MOV     XPOS, #MAXCHAR          ; ELSE move to end of line
 48 0C62    21      F4                                  AJMP    XBLINAD                 ;   update and return
```

```
                                ;;              CRT - cursor right
                                ;
                                ;               *CRT* moves the cursor right one column.  If the cursor is in
                                ;               the last column then the cursor will not be advanced.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    A, DPTR

0C64  E5      11                CRT:            MOV     A, XPOS                 ; Get X-position
0C66  B4      4F      01                        CJNE    A, #MAXCHAR, CRT1       ; IF end of line THEN
0C69  22                                        RET                             ;   return

0C6A  05      11                CRT1:           INC     XPOS                    ; ELSE bump position
0C6C  21      F4                                AJMP    XBLINAD                 ;   update and return


                                ;;              ACRT - ANSI cursor right
                                ;
                                ;               *ACRT* moves the cursor towards the end of the line
                                ;               the specified Pn number of times.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    A, DPTR, INDX1

0C6E  31      D1                ACRT:           ACALL   GNP                     ; Get parameter
0C70  70      01                                JNZ     ACR1                    ; IF no parameter or 0 THEN
0C72  04                                        INC     A                       ;   bump to 1

0C73  FA                        ACR1:           MOV     INDX1, A                ; Set up index
0C74  91      64                ACR:            ACALL   CRT                     ; REPEAT cursor right one
0C76  DA      FC                                DJNZ    INDX1, ACR              ; UNTIL count = 0
0C78  22                                        RET
```

```
***.Z-29.COMPUTER.TERMINAL.FIRMWARE.VERSION.1.03.***

                        ;;      HCRT - Hazeltine cursor right
                        ;
                        ;       *HCRT* moves the cursor right one column.  If the cursor is
                        ;       in the last column then the cursor moves to the beginning of
                        ;       the next line.
                        ;
                        ;
                        ;       ENTRY   none
                        ;
                        ;       EXIT    none
                        ;
                        ;       USES    A, DPTR, PTR1

0C79  E5  11            HCRT:   MOV     A, XPOS         ; Get X-position
0C7B  B4  4F  E6                CJNE    A, #MAXCHAR, CRT        ; IF not eol THEN use *CRT*
0C7E  AD  12                    MOV     TEMP, YPOS      ; Save Y-position
0C80  91  A8                    ACALL   CDN             ; Move cursor down
0C82  E5  12                    MOV     A, YPOS
0C84  B5  05  01                CJNE    A, TEMPA, HCRT1 ; IF xpos did not change THEN
0C87  22                        RET                     ;   do nothing

0C88  75  11  00        HCRT1:  MOV     XPOS, #0        ; ELSE move to beginning
0C8B  21  F4                    AJMP    XBLINAD         ;   update and return
```

```
                                        ;;              CUP - cursor up
                                        ;
                                        ;               *CUP* moves the cursor up one line on the display.  The
                                        ;               cursor may move out of but not into fixed regions and
                                        ;               never past line zero.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    A, DPTR, PTR1

0C8D  E5      12        CUP:            MOV     A, YPOS         ; Get Y-position
0C8F  60      03                        JZ      CUP1            ; IF top of screen THEN return
0C91  B5      17  01                    CJNE    A, TSCROLL, CUP1A ; IF top of scroll region THEN
0C94  22                CUP1:           RET                     ;   return

0C95  B4      18  01    CUP1A:          CJNE    A, #MAXLINE, CUP1B ; IF 25th line THEN
0C98  22                                RET                     ;   return

0C99  15      12        CUP1B:          DEC     YPOS            ; ELSE dec position
0C9B  21      F4                        AJMP    XBLINAD         ;   update and return


                                        ;;              ACUP - ANSI cursor up
                                        ;
                                        ;               *ACUP* moves the cursor toward the top of the display
                                        ;               the specified Pn number of times.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    A, DPTR, INDX1, PTR1

0C9D  31      D1        ACUP:           ACALL   GNP             ; Get parameter
0C9F  70      01                        JNZ     ACU1            ; IF no parameter or 0 THEN
0CA1  04                                INC     A               ;   bump to 1

0CA2  FA                ACU1:           MOV     INDX1, A        ; Set up index
0CA3  91      8D        ACU:            ACALL   CUP             ; REPEAT cursor up one line
0CA5  DA      FC                        DJNZ    INDX1, ACU      ; UNTIL count = 0
0CA7  22                                RET
```

```
                                ;;              CDN - cursor down
                                ;
                                ;               *CDN* moves the cursor down one line on the display.  The
                                ;               cursor may move out of but not into fixed regions and
                                ;               never past line 24.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    A, DPTR, PTR1

0CA8  E5      12        CDN:    MOV     A, YPOS                 ; Get Y-position
0CAA  B4      18  01            CJNE    A, #MAXLINE, CDN1       ; IF 25th line THEN return
0CAD  22                        RET
0CAE  B4      17  01    CDN1:   CJNE    A, #MAXLINE-1, CDN2     ; IF 24th line THEN return
0CB1  22                        RET
0CB2  04                CDN2:   INC     A                       ; Bump value
0CB3  B5      18  01            CJNE    A, BFIX, CDN3           ; IF scroll region THEN return
0CB6  22                        RET
0CB7  05      12        CDN3:   INC     YPOS                    ; ELSE bump position
0CB9  21      F4                AJMP    XBLINAD                 ; Update and return


                                ;;              ACDN - ANSI cursor down
                                ;
                                ;               *ACDN* moves the cursor toward the bottom of the display
                                ;               the specified Pn number of lines.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    A, DPTR, INDX1, PTR1

0CBB  31      D1        ACDN:   ACALL   GNP                     ; Get parameter
0CBD  70      01                JNZ     ACD1                    ; IF no parameters or 0 THEN
0CBF  04                        INC     A                       ;   bump to 1

0CC0  FA                ACD1:   MOV     INDX1, A                ; Set up index
0CC1  91      A8        ACD:    ACALL   CDN                     ; REPEAT cursor down one line
0CC3  DA      FC                DJNZ    INDX1, ACD              ; UNTIL count = 0
0CC5  22                        RET
```

```
                                ;;              APBT - ANSI perform back tab
                                ;
                                ;               *APBT* moves the cursor back the number of tab stops indicated.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

0CC6  31      D1                APBT:           ACALL   GNP                     ; Get parameter
0CC8  70      01                                JNZ     APBT1                   ; IF no parameter or 0 THEN
0CCA  04                                        INC     A                       ;   bump to 1

0CCB  C0      E0                APBT1:          PUSH    ACC                     ; REPEAT
0CCD  12      0602                              LCALL   BTAB                    ;   perform back tab
0CD0  D0      E0                                POP     ACC
0CD2  D5      E0      F6                        DJNZ    ACC, APBT1              ; UNTIL count = 0
0CD5  22                                        RET
```

```
                                ;;              CUA - cursor addressing routine
                                ;
                                ;               *CUA* does direct cursor addressing for a few emulation modes.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

0CD6  11    2A          CUA:    ACALL   SETDISP          ; Set dispatch and return

0CD8  75    16    04            MOV     DSADRL, #LOW (CUC)  ; Next dispatch address
0CDB  75    15    0D            MOV     DSADRH, #HIGH (CUC)
0CDE  30    43    11            JNB     HAZ, CUB0           ; IF Hazeltine THEN
0CE1  B4    60    00            CJNE    A, #96, CUA1
0CE4  40    02          CUA1:   JC      CUA2                ;   IF value >= 96 THEN
0CE6  94    60                  SUBB    A, #96              ;     subtract offset
0CE8  B4    50    00    CUA2:   CJNE    A, #NUMCHARS, CUA2A
0CEB  40    02          CUA2A:  JC      CUA3                ;   IF value >= numchars THEN
0CED  74    4F                  MOV     A, #MAXCHAR         ;     move to end of line
0CEF  80    2F          CUA3:   SJMP    CUR2                ;   update and return

0CF1  22                CUR:    RET

0CF2  C3                CUB0:   CLR     C
0CF3  94    20                  SUBB    A, #' '             ; ELSE subtract offset
0CF5  40    FA                  JC      CUR                 ;   check for lower bounds
0CF7  B4    18    04            CJNE    A, #MAXLINE, CUB1   ;   check for upper bounds
0CFA  30    15    F4            JNB     L25EN, CUR          ;   IF 25th disabled THEN ret
0CFD  D3                        SETB    C                   ;   ELSE fall through
0CFE  50    F1          CUB1:   JNC     CUR                 ;   IF upper bounds ok THEN
0D00  F5    12          CUR1:   MOV     YPOS, A             ;     replace Y-position
0D02  21    F4                  AJMP    XBLINAD             ;     build line address

0D04  11    2F          CUC:    ACALL   SETNORM             ; Next dispatch address
0D06  30    43    0B            JNB     HAZ, CUC0           ; IF Hazeltine THEN
0D09  54    1F                  ANL     A, #00011111B       ;   do mod function
0D0B  B4    18    00            CJNE    A, #MAXLINE, CUC1
0D0E  40    02          CUC1:   JC      CUC1A               ;   IF value >= 24 THEN
0D10  74    17                  MOV     A, #MAXLINE-1       ;     move to last line
0D12  80    EC          CUC1A:  SJMP    CUR1                ;   update and return

0D14  C3                CUC0:   CLR     C
0D15  94    20                  SUBB    A, #' '             ; Subtract offset
0D17  40    05                  JC      CUC2                ; Check for lower bounds
0D19  B4    50    00            CJNE    A, #NUMCHARS, CUC0A
0D1C  40    02          CUC0A:  JC      CUR2                ; Check for upper bounds
0D1E  74    4F          CUC2:   MOV     A, #MAXCHAR         ; IF bad range THEN move to end
0D20  F5    11          CUR2:   MOV     XPOS, A             ; Replace X-position
0D22  21    F4                  AJMP    XBLINAD             ; Build line address and return
```

```
                                ;;              DSR - define scrolling region
                                ;
                                ;               *DSR* defines the scrolling regions top and bottom line.
                                ;               The region must be at least two lines long and can not include
                                ;               the 25th line.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    A, DPTR

0D24  31      D1                DSR:            ACALL   GNP                     ; Get first parameter
0D26  60      01                                JZ      DSR0                    ; IF parameter and not 0 THEN
0D28  14                                        DEC     A                       ;   normalize to 0..24

0D29  B4      17      00        DSR0:           CJNE    A, #MAXLINE-1, DSR0A
0D2C  50      02                DSR0A:          JNC     DSR1                    ; IF top >= 22 THEN
0D2E  F5      17                                MOV     TSCROLL, A              ;   change value

0D30  31      D1                DSR1:           ACALL   GNP                     ; Get next parameter
0D32  60      01                                JZ      DSR1A                   ; IF parameter and not 0 THEN
0D34  14                                        DEC     A                       ;   normalize to 0..24

0D35  B5      17      01        DSR1A:          CJNE    A, TSCROLL, DSR2
0D38  D3                                        SETB    C                       ; IF bottom = top OR

0D39  40      05                DSR2:           JC      DSRERR                  ; IF bottom < top OR
0D3B  B4      18      00                        CJNE    A, #MAXLINE, DSR2A
0D3E  40      02                DSR2A:          JC      DSROK                   ; IF bottom >= 25 THEN error

0D40  74      17                DSRERR:         MOV     A, #MAXLINE-1           ; Default error value
0D42  04                        DSROK:          INC     A                       ; Bump for top of bottom fixed
0D43  F5      18                                MOV     BFIX, A                 ; Save value
0D45  81      36                                AJMP    SCH                     ; Set cursor home and return
```

```
                                        ;;              PIL - perform insert line
                                        ;
                                        ;               *PIL* inserts a blank line at the cursor position after
                                        ;               moving the remaining lines down one line.  Everything is
                                        ;               done relative to the region the cursor is in, top fixed,
                                        ;               bottom fixed, 25th line, or scrolling region.  No region
                                        ;               outside of the current region is effected.  The cursor is
                                        ;               moved to the beginning of the current line.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    all

0D47  C0   12            PIL:           PUSH    YPOS                    ; Save values
0D49  C0   17                           PUSH    TSCROLL
0D4B  C0   18                           PUSH    BFIX
0D4D  E5   12                           MOV     A, YPOS                 ; Get position
0D4F  B5   17   00                      CJNE    A, TSCROLL, PILA        ; IF upper fixed region THEN
0D52  40   21            PILA:          JC      PILRT1                  ;   just return
0D54  15   18                           DEC     BFIX
0D56  B5   18   07                      CJNE    A, BFIX, PIL1           ; IF last line scrolling region
0D59  85   12   F0                      MOV     B, YPOS
0D5C  71   96            PIL0:          ACALL   CLRLINE                 ;   just clear the line
0D5E  80   12                           SJMP    PILRET

0D60  40   08            PIL1:          JC      PIL2                    ; IF lower fixed THEN

0D62  B4   18   10                      CJNE    A, #MAXLINE, PILRT1     ;   IF not 25th line THEN return
0D65  75   F0   18                      MOV     B, #MAXLINE             ;   ELSE clear 25th line
0D68  80   F2                           SJMP    PIL0                    ;     and return

0D6A  05   18            PIL2:          INC     BFIX
0D6C  85   12   17                      MOV     TSCROLL, YPOS           ; Insert line uses
0D6F  12   0518                         CALL    PRLF                    ; Reverse line feed routine

0D72  75   11   00        PILRET:       MOV     XPOS, #0                ; Move to beginning of line
0D75  D0   18             PILRT1:       POP     BFIX                    ; Restore values
0D77  D0   17                           POP     TSCROLL
0D79  D0   12                           POP     YPOS
0D7B  21   F4                           AJMP    XBLINAD                 ; Build line address and return
```

```
                                ;;              APIL - ANSI perform insert line
                                ;
                                ;               *APIL* inserts the specified Pn number of lines at the
                                ;               current cursor position.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

0D7D   31       D1      APIL:   ACALL   GNP             ; Get next parameter
0D7F   70       01              JNZ     APILO           ; IF no parameter or 0 THEN
0D81   04                       INC     A               ;   bump to 1

0D82   FA               APILO:  MOV     INDX1, A        ; Set up index
0D83   B1       47      APIL1:  ACALL   PIL             ; REPEAT insert one line
0D85   DA       FC              DJNZ    INDX1, APIL1    ; UNTIL count = 0
0D87   22                       RET
```

```
                                ;;              PDL - perform delete line
                                ;
                                ;               *PDL* moves the remaining lines of the current region up
                                ;               one line, clears the last line of the region, and moves
                                ;               the cursor to the beginning of the current line.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

0D88  C0  12            PDL:    PUSH    YPOS            ; Save values
0D8A  C0  17                    PUSH    TSCROLL
0D8C  C0  18                    PUSH    BFIX
0D8E  E5  12                    MOV     A, YPOS         ; Get current position
0D90  F5  F0                    MOV     B, A            ; Save it in (B) also
0D92  B5  17  00                CJNE    A, TSCROLL, PDLA ; IF upper fixed THEN
0D95  40  DE            PDLA:   JC      PILRT1          ;   just return
0D97  04                        INC     A
0D98  B5  18  04                CJNE    A, BFIX, PDL1   ; IF last line of scroll THEN
0D9B  71  96            PDL0:   ACALL   CLRLINE         ;   clear current line
0D9D  80  D3                    SJMP    PILRET          ;   and return

0D9F  40  08            PDL1:   JC      PDL2            ; IF bottom fixed THEN
0DA1  B4  19  D1                CJNE    A, #NUMLINES, PILRT1 ;   IF 25th line THEN
0DA4  75  F0  18                MOV     B, #MAXLINE     ;     clear 25th line
0DA7  80  F2                    SJMP    PDL0            ;     and return

0DA9  85  18  12        PDL2:   MOV     YPOS, BFIX      ; Put cursor above fixed
0DAC  15  12                    DEC     YPOS
0DAE  85  F0  17                MOV     TSCROLL, B      ; Get saved current line
0DB1  12  0561                  LCALL   PLF             ; Line feed to simulate delete
0DB4  A1  72                    AJMP    PILRET          ; Return
```

```
                                ;;              APDL - ANSI perform delete line
                                ;
                                ;               *APDL* deletes the specified Pn number of lines at and
                                ;               following the cursor position.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    A, B, DPTR, INDX1

0DB6  31      D1                APDL:           ACALL   GNP                     ; Get parameter
0DB8  70      01                                JNZ     APDLO                   ; IF no parameter or 0 THEN
0DBA  04                                        INC     A                       ;   bump to 1

0DBB  FA                        APDLO:          MOV     INDX1, A                ; Set up index
0DBC  B1      88                APDL1:          ACALL   PDL                     ; REPEAT delete one line
0DBE  DA      FC                                DJNZ    INDX1, APDL1            ; UNTIL count = 0
0DC0  22                                        RET
```

```
                                ;;              PIC - perform insert character
                                ;
                                ;               *PIC* move all characters that are at and to the right of the
                                ;               cursor one column to the right.  Cursor position does not
                                ;               change.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all
                                
0DC1 74      50                 PIC:            MOV     A, #NUMCHARS            ; Calc number of moves to make
0DC3 C3                                         CLR     C
0DC4 95      11                                 SUBB    A, XPOS
0DC6 FA                                         MOV     INDX1, A
                                
0DC7 E5      11                                 MOV     A, XPOS                 ; Get current address
0DC9 85      12      F0                         MOV     B, YPOS
0DCC 12      0503                               LCALL   CLA
                                
0DCF 78      20                                 MOV     PTR1, #' '              ; Init value for 1st slot
0DD1 79      00                                 MOV     PTR2, #0
                                
0DD3 E0                         PIC1:           MOVX    A, @DPTR
0DD4 FC                                         MOV     WORK, A                 ; Save lower character
0DD5 53      83      EF                         ANL     DPH, #ATRANL
0DD8 E0                                         MOVX    A, @DPTR
0DD9 FD                                         MOV     TEMP, A                 ; Save lower attribute
                                
0DDA E9                                         MOV     A, PTR2
0DDB F0                                         MOVX    @DPTR, A                ; Write lower attribute
0DDC 43      83      10                         ORL     DPH, #CHRORL
0DDF E8                                         MOV     A, PTR1
0DE0 F0                                         MOVX    @DPTR, A                ; Write lower character
                                
0DE1 A8      04                                 MOV     PTR1, WORKA             ; Exchange character
0DE3 A9      05                                 MOV     PTR2, TEMPA             ; Exchange attribute
                                
0DE5 A3                                         INC     DPTR                    ; Bump up
0DE6 DA      EB                                 DJNZ    INDX1, PIC1             ; UNTIL all done
0DE8 22                                         RET
```

```
                                ;;              APIC - ANSI perform insert character
                                ;
                                ;               *APIC* inserts the specified number of characters into
                                ;               the line.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

0DE9  31      D1        APIC:   ACALL   GNP             ; Get parameter
0DEB  70      01                JNZ     APIC0           ; IF no parameter or 0 THEN
0DED  04                        INC     A               ;   bump to 1

0DEE  FB                APIC0:  MOV     INDX2, A        ; Setup index
0DEF  B1      C1        APIC1:  ACALL   PIC             ; REPEAT perform insert char
0DF1  DB      FC                DJNZ    INDX2, APIC1    ; UNTIL count = 0
0DF3  22                        RET
```

*** Z-29 COMPUTER TERMINAL FIRMWARE VERSION 1.03 ***

```
                                ;;              PDC - perform delete character
                                ;
                                ;               *PDC* deletes the character at the cursor position by moving
                                ;               the remaining characters on the line to the left one column
                                ;               and inserting a space in the last column on the line.  Cursor
                                ;               position does not change.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all


0DF4  74    4F          PDC:    MOV     A, #MAXCHAR             ; Calc number of moves
0DF6  C3                        CLR     C
0DF7  95    11                  SUBB    A, XPOS
0DF9  FA                        MOV     INDX1, A
0DFA  74    4F                  MOV     A, #MAXCHAR             ; Get address at end of line
0DFC  85    12      F0          MOV     B, YPOS
0DFF  12    0503                LCALL   CLA
0E02  E0                        MOVX    A, @DPTR
0E03  F8                        MOV     PTR1, A                 ; Save upper character
0E04  74    20                  MOV     A, #' '
0E06  F0                        MOVX    @DPTR, A                ; Clear last character
0E07  53    83      EF          ANL     DPH, #ATRANL
0E0A  E0                        MOVX    A, @DPTR
0E0B  F9                        MOV     PTR2, A                 ; Save upper attribute
0E0C  E4                        CLR     A
0E0D  F0                        MOVX    @DPTR, A                ; Clear last attribute
0E0E  EA                        MOV     A, INDX1
0E0F  60    20                  JZ      PDC2                    ; Exit out if last column

0E11  E5    82          PDC1:   MOV     A, DPL                  ; Decrement DPTR without
0E13  24    FF                  ADD     A, #-1                  ; Using a  -CLR C-
0E15  F5    82                  MOV     DPL, A                  ; Instruction to save
0E17  E5    83                  MOV     A, DPH                  ; Every bit of time
0E19  34    FF                  ADDC    A, #-1
0E1B  F5    83                  MOV     DPH, A

0E1D  E0                        MOVX    A, @DPTR
0E1E  FD                        MOV     TEMP, A                 ; Save lower attribute
0E1F  43    83      10          ORL     DPH, #CHRORL
0E22  E0                        MOVX    A, @DPTR
0E23  FC                        MOV     WORK, A                 ; Save lower character

0E24  E8                        MOV     A, PTR1
0E25  F0                        MOVX    @DPTR, A                ; Write character
0E26  53    83      EF          ANL     DPH, #ATRANL
0E29  E9                        MOV     A, PTR2
0E2A  F0                        MOVX    @DPTR, A                ; Write attribute

0E2B  A8    04                  MOV     PTR1, WORKA             ; Exchange character
0E2D  A9    05                  MOV     PTR2, TEMPA             ; Exchange attribute
0E2F  DA    E0                  DJNZ    INDX1, PDC1             ; UNTIL all done
0E31  22                PDC2:   RET
```

```
                                ;;              APDC - ANSI perform delete character
                                ;
                                ;               *APDC* deletes the specified number of characters from the
                                ;               line.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all


0E32    31      D1              APDC:           ACALL   GNP                     ; Get parameter
0E34    70      01                              JNZ     APDC0                   ; IF no parameter or 0 THEN
0E36    04                                      INC     A                       ;   bump to 1

0E37    FB                      APDC0:          MOV     INDX2, A                ; Setup index
0E38    B1      F4              APDC1:          ACALL   PDC                     ; REPEAT perform delete char
0E3A    DB      FC                              DJNZ    INDX2, APDC1            ; UNTIL count = 0
0E3C    22                                      RET
```

```
                                        ;;              ZDSSBR - ZDS set baud rate
                                        ;
                                        ;               *ZDSSBR* sets the baud rate of the UART built into the 8051.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    all

0E3D    11      2A              ZDSSBR: ACALL   SETDISP         ; Set dispatch and return
0E3F    11      2F                      ACALL   SETNORM         ; Restore dispatch address
0E41    C3                              CLR     C
0E42    94      40                      SUBB    A, #'@'         ; Subtract offset
0E44    40      18                      JC      SBRET           ; Check for lower bounds
0E46    B4      0E      00              CJNE    A, #('N'-'@'), ZDSO     ; Check for upper bounds
0E49    50      13              ZDSO:   JNC     SBRET

                                        IFC     NE,SBR-$
                                        ERROR   ;ROUTINE 'SBR' MUST FOLLOW
                                        ENDC


                                        ;;              SBR - set baud rate
                                        ;
                                        ;               *SBR* selects a baud rate given the value of which rate to
                                        ;               select.
                                        ;
                                        ;
                                        ;               ENTRY   (A) = which baud rate (0..13)
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    A, DPTR


0E4B    90      0E5F            SBR:    MOV     DPTR, #BRTAB    ; Point to baud rate table
0E4E    93                              MOVC    A, @A+DPTR      ; Get value from table
0E4F    60      0D                      JZ      SBRET           ; IF zero THEN exit
0E51    F5      33                      MOV     BAUDRATE, A     ; Save baud rate
0E53    F5      8D                      MOV     TH1, A          ; Change baud rate
0E55    C2      9F                      CLR     SM0             ; Place UART into mode 1
0E57    B4      53      00              CJNE    A,#83, SBR0
0E5A    50      02              SBR0:   JNC     SBRET           ; IF baud <= 300 THEN
0E5C    D2      9F                      SETB    SM0             ;   place UART into mode 3
0E5E    22                      SBRET:  RET
```

```
                                ;;              BRTAB - baud rate table
                                ;
                                ;               *BRTAB* contains the timer values for baud rates

0E5F            BRTAB           EQU     $
0E5F 01                         DB      1               ; 75 baud
0E60 52                         DB      82              ; 110 baud
0E61 80                         DB      128             ; 150 baud
0E62 C0                         DB      192             ; 300 baud
0E63 E0                         DB      224             ; 600 baud
0E64 F0                         DB      240             ; 1200 baud
0E65 F5                         DB      245             ; 1800 baud
0E66 00                         DB      0               ; 2000 baud (not available)
0E67 F8                         DB      248             ; 2400 baud
0E68 00                         DB      0               ; 3600 baud (not available)
0E69 FC                         DB      252             ; 4800 baud
0E6A 00                         DB      0               ; 7200 baud (not available)
0E6B FE                         DB      254             ; 9600 baud
0E6C FF                         DB      255             ; 19200 baud
```

*** Z-29 COMPUTER TERMINAL FIRMWARE VERSION 1.03 ***

```
                                ;;      PSOF - put string in output FIFO
                                ;
                                ;       *PSOF* places the character string pointed to by DPTR into
                                ;       the output FIFO.  Final character has bit seven set.
                                ;
                                ;
                                ;       ENTRY   (DPTR) = start of string
                                ;
                                ;       EXIT    none
                                ;
                                ;       USES    A, B, DPTR


0E6D E4                 PSOF:   CLR     A               ; REPEAT
0E6E 93                         MOVC    A, @A+DPTR      ;   set character
0E6F 20 E7 0D                   JB      ACC.7, SOR      ;   IF bit 7 set THEN exit
0E72 C0 82                      PUSH    DPL             ;   save DPTR
0E74 C0 83                      PUSH    DPH
0E76 D1 A6                      ACALL   CPR2            ;   place character on FIFO
0E78 D0 83                      POP     DPH             ;   restore DPTR
0E7A D0 82                      POP     DPL
0E7C A3                         INC     DPTR            ;   bump pointer
0E7D 80 EE                      SJMP    PSOF            ; UNTIL done

0E7F 54 7F              SOR:    ANL     A, #7FH         ; Mask off 7th bit
0E81 80 23                      SJMP    CPR2            ; Place last char on FIFO
```

```
                                        ;;              CPR - cursor position report
                                        ;
                                        ;               *CPR* outputs the cursor position in the format of the mode
                                        ;               currently set.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    all

0E83    30      43      11      CPR:    JNB     HAZ, CPR1       ; IF Hazeltine THEN
0E86    E5      11                      MOV     A, XPOS         ;   set -xpos-
0E88    B4      20      00              CJNE    A, #32, CPR0
0E8B    50      02              CPR0:   JNC     CPR0A           ;   IF value < 32 THEN
0E8D    24      60                      ADD     A, #96          ;     normalize
0E8F    D1      A6              CPR0A:  ACALL   CPR2            ;   output value
0E91    E5      12                      MOV     A, YPOS         ;   set -ypos-
0E93    24      60                      ADD     A, #96          ;   normalize
0E95    80      0F                      SJMP    CPR2            ;   output value and return

0E97    90      0EA9            CPR1:   MOV     DPTR, #CPRM     ; ELSE
0E9A    D1      6D                      ACALL   PSOF            ;   output  ESC Y
0E9C    E5      12                      MOV     A, YPOS         ;   set line number
0E9E    24      20                      ADD     A, #' '         ;   convert to ASCII
0EA0    D1      A6                      ACALL   CPR2            ;   output it
0EA2    E5      11                      MOV     A, XPOS         ;   set column number
0EA4    24      20                      ADD     A, #' '         ;   convert to ASCII
0EA6    02      022A            CPR2:   JMP     PCOF            ;   output it and return

0EA9    1B      D9              CPRM:   DB      ESC, ('Y'+80H)  ; ESC Y
```

```
                                        ;;              IDT - identify terminal
                                        ;
                                        ;               *IDT* identifies the terminal as a DEC VT52 so that existing
                                        ;               DEC software which interrogates the console type will operate
                                        ;               as it would with a VT52.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    all

0EAB  90      0EB0              IDT:    MOV     DPTR, #IDTM
0EAE  C1      6D                        AJMP    PSOF                    ; Output string and return

0EB0  1B      2F      CB        IDTM:   DB      ESC, '/', ('K'+80H)     ; ESC / K


                                        ;;              IDTT - identify terminal type
                                        ;
                                        ;               *IDTT* identifies the terminal as a Z-29 so that future
                                        ;               software can interrogate the console type and see what
                                        ;               version of the terminal is running.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    all

0EB3  11      2A                IDTT:   ACALL   SETDISP                 ; Set dispatch and return

0EB5  11      2F                        ACALL   SETNORM                 ; Restore dispatch address
0EB7  B4      30      6B                CJNE    A, #'0', ZAR            ; IF not '0' THEN return
0EBA  90      0EBF                      MOV     DPTR, #IDTTM
0EBD  C1      6D                        AJMP    PSOF                    ; Output string and return

0EBF  1B      69      42        IDTTM:  DB      ESC, 'iB', ('0'+80H)    ; ESC i B 0
0EC2  B0
```

```
                                ;;              ADA - ANSI device attributes
                                ;
                                ;               *ADA* identifies the terminal as a DEC VT100 so that existing
                                ;               DEC software which interrogates the console type will operate
                                ;               as it would with a VT100.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

0EC3  31      D1                ADA:            ACALL   GNP                     ; Get parameter
0EC5  70      5E                                JNZ     ZAR                     ; IF no parameter or 0 THEN
0EC7  90      0ECC                              MOV     DPTR, #ADAM
0ECA  C1      6D                                AJMP    PSOF                    ;   output string and return

0ECC  1B      5B      3F        ADAM:           DB      ESC, '[?1;2', ('c'+80H)
0ECF  31      3B      32
0ED2  E3
```

```
                                ;;              ZTAB - ZDS mode tab functions
                                ;
                                ;               *ZTAB* does the tab function requested.
                                ;
                                ;               ENTRY   none
                                ;               EXIT    none
                                ;               USES    all

0ED3  11  2A                    ZTAB:           ACALL   SETDISP          ;Get dispatch address and return

0ED5  11  2F                                    ACALL   SETNORM          ;Restore dispatch address
0ED7  75  F0  03                                MOV     B, #ZTABTL
0EDA  90  0EDF                                  MOV     DPTR, #ZTABT
0EDD  01  36                                    AJMP    STJMP            ; Jump to any routine


                                ;;              ZTABT - ZDS mode tab table
                                ;
                                ;               *ZTABT* contains valid parameters for the set/clear tab
                                ;               functions in ZDS mode.

          0EDF                  ZTABT           EQU     $

0EDF  30                                        DB      '0'              ; ESC , 0
0EE0  0627                                      DW      HTC              ; Horizontal tab clear

0EE2  33                                        DB      '3'              ; ESC , 3
0EE3  0598                                      DW      CLRTABS          ;Clear all tabs

0EE5  38                                        DB      '8'              ; ESC , 8
0EE6  0621                                      DW      HTS              ; Horizontal tab set

          0003                  ZTABTL          EQU     ($-ZTABT)/3      ; Table length
```

```
                                        ;;              SCP - save cursor position
                                        ;
                                        ;               *SCP* saves current cursor position.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    none

0EE8    85      11      35      SCP:                    MOV     XXPOS, XPOS             ; Save values
0EEB    85      12      36                              MOV     XYPOS, YPOS
0EEE    22                                              RET


                                        ;;              ASCP - ANSI save cursor position
                                        ;
                                        ;               *ASCP* saves current position and attributes.
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    all

0EEF    31      D1              ASCP:                   ACALL   GNP                     ; Get parameter if any
0EF1    40      F5                                      JC      SCP                     ; IF no parameter THEN ok
0EF3    22                                              RET
```

```
                                        ;;              USCP - unsave cursor position
                                        ;
                                        ;               *USCP* returns to position saved with the use of *SCP*.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    all

OEF4    85      35      11      USCP:           MOV     XPOS, XXPOS             ; Restore values
OEF7    E5      36                              MOV     A,XYPOS
OEF9    B4      18      03                      CJNE    A, #MAXLINE, USCP1      ; IF on 25th line THEN
OEFC    30      15      02                      JNB     L25EN, USCP2            ;   IF not enabled THEN exit
OEFF    F5      12              USCP1:          MOV     YPOS, A                 ; ELSE set position
OF01    21      F4              USCP2:          AJMP    XBLINAD                 ;   build address and return


                                        ;;              AUSCP - ANSI unsave cursor position
                                        ;
                                        ;               *AUSCP* returns to position saved with the use of *SPCP*.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    all

OF03    31      D1              AUSCP:          ACALL   GNP                     ; Get parameter if any
OF05    40      ED                              JC      USCP                    ; IF no parameters THEN ok
OF07    22                                      RET
```

```
                                ;;              ZATR - ZDS select attributes
                                ;
                                ;               *ZATR* selects which attributes are to be turned on or off
                                ;               using the configuration of the lower 5 bits of the character.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

0F08  11      2A                ZATR:           ACALL   SETDISP                 ; Set dispatch and return

0F0A  11      2F                                ACALL   SETNORM                 ; Restore dispatch address
0F0C  C3                                        CLR     C
0F0D  94      30                                SUBB    A, #'0'
0F0F  40      14                                JC      ZAR                     ; IF below range THEN exit
0F11  B4      20      00                        CJNE    A, #32, ZATR0
0F14  50      0F                ZATR0:          JNC     ZAR                     ; IF above range THEN exit
0F16  13                                        RRC     A                       ; Get reverse video
0F17  92      08                                MOV     RVVA, C
0F19  13                                        RRC     A                       ; Get blinking
0F1A  92      0A                                MOV     BLINKA, C
0F1C  13                                        RRC     A                       ; Get half intensity
0F1D  92      0B                                MOV     HALFIA, C
0F1F  13                                        RRC     A                       ; Get underline
0F20  92      09                                MOV     UNDLNA, C
0F22  13                                        RRC     A                       ; Get alternate char set
0F23  92      0C                                MOV     ALTCHARA, C
0F25  22                        ZAR:            RET


                                ;;              XATR - exit all attributes
                                ;
                                ;               *XATR* turns off all attributes so that none are active.
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    none

0F26  53      21      F0        XATR:           ANL     ATTRIBUTES, #0F0H       ; Mask off all attributes
0F29  22                                        RET
```

```
                                ;;              EHALF - enter half intensity mode
                                ;
                                ;               *EHALF* places the terminal into half intensity mode.
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    none

OF2A    D2      0B              EHALF:          SETB    HALFIA                  ; Set half intensity attribute
OF2C    22                                      RET


                                ;;              EUNDL - enter underline mode
                                ;
                                ;               *EUNDL* places the terminal into underline mode.
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    none

OF2D    D2      09              EUNDL:          SETB    UNDLNA                  ; Set underline attribute
OF2F    22                                      RET


                                ;;              EBLNK - enter blink mode
                                ;
                                ;               *EBLNK* places the terminal into blink mode
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    none

OF30    D2      0A              EBLNK:          SETB    BLINKA                  ; Set blink attribute
OF32    22                                      RET
```

```
                                ;;              ERVM - enter reverse video mode
                                ;
                                ;               *ERVM* places the terminal into reverse video mode.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    none

0F33    D2      08              ERVM:           SETB    RVVA                    ; Set reverse video attribute
0F35    22                                      RET


                                ;;              XRVM - exit reverse video mode
                                ;
                                ;               *XRVM* removes the terminal from reverse video mode.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    none

0F36    C2      08              XRVM:           CLR     RVVA                    ; Clear reverse video attribute
0F38    22                                      RET
```

```
                                ;;              EGM - enter graphics mode
                                ;
                                ;               *EGM* places terminal into graphics mode.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    none
                                
                                
0F39    D2      0D              EGM:            SETB    GRPH                    ; Set graphics mode
0F3B    22                                      RET


                                ;;              XGM - exit graphic mode
                                ;
                                ;               *XGM* exits the terminal out of graphic character mode.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    none

0F3C    C2      0D              XGM:            CLR     GRPH                    ; Clear graphic mode
0F3E    22                                      RET
```

```
                                ;;              GODES - GO designator
                                ;               G1DES - G1 designator
                                ;
                                ;               *GODES* and *G1DES* assign a graphic character set to be
                                ;               associated with itself.  If the character set is not found
                                ;               then no action is taken.  The values returned for the
                                ;               different character sets are as follows:
                                ;
                                ;               Character | Internal
                                ;               received  | value and meaning
                                ;               ------------------------------
                                ;                 A       |  0 = United States (should be United Kingdom)
                                ;                 B       |  0 = United States
                                ;                 0       |  1 = United States with alternate graphic characters
                                ;                 1       |  2 = Alternate character set
                                ;                 2       |  3 = Alternate with alternate graphic characters
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

0F3F  11  2A            GODES:  ACALL   SETDISP         ; Set dispatch and return

0F41  11  2F                    ACALL   SETNORM         ; Restore dispatch address
0F43  C2  D5                    CLR     F0              ; Flag for GODES
0F45  80  08                    SJMP    GDES            ; Do the stuff


0F47  11  2A            G1DES:  ACALL   SETDISP         ; Set dispatch and return

0F49  11  2F                    ACALL   SETNORM         ; Restore dispatch address
0F4B  D2  D5                    SETB    F0              ; Flag for G1DES
0F4D  80  00                    SJMP    GDES            ; Do the stuff


0F4F  B4  41  03        GDES:   CJNE    A, #'A', GDES1  ; IF char = 'A' THEN
0F52  E4                GDES0:  CLR     A               ;   value of zero
0F53  80  10                    SJMP    GDES3

0F55  B4  42  02        GDES1:  CJNE    A, #'B', GDES2  ; IF char = 'B' THEN
0F58  80  F8                    SJMP    GDES0           ;   value of zero

0F5A  C3                GDES2:  CLR     C
0F5B  94  30                    SUBB    A, #'0'         ; Subtract offset
0F5D  40  1B                    JC      GDESR           ; IF out of bounds THEN return
0F5F  B4  03  00                CJNE    A, #3, GDES2A
0F62  50  16            GDES2A: JNC     GDESR           ; IF out of bounds THEN return
0F64  04                        INC     A               ; Bump into range
```

*** Z-29 COMPUTER TERMINAL FIRMWARE VERSION 1.03 ***

```
    1 0F65   20   D5   09   GDES3:    JB    F0, GDES3A          ; IF for GO THEN
    2 0F68   53   1F   F0             ANL   GSET, #11110000B    ;   clear old value
    3 0F6B   42   1F                  ORL   GSET, A             ;   put in new value
    4 0F6D   30   12   0B             JNB   GSEL, GOPICK        ;   IF selected THEN pick it
    5 0F70   22                       RET                       ;   ELSE return
    6
    7 0F71   53   1F   0F   GDES3A:   ANL   GSET, #00001111B    ; ELSE clear old value
    8 0F74   C4                       SWAP  A                   ;   place into proper nibble
    9 0F75   42   1F                  ORL   GSET, A             ;   put in new value
   10 0F77   20   12   07             JB    GSEL, G1PICK        ;   IF selected THEN pick it
   11
   12 0F7A   22              GDESR:   RET                       ; Return
```

```
                                        ;;              GOPICK - pick GO graphic set
                                        ;               G1PICK - pick G1 graphic set
                                        ;
                                        ;               *GOPICK* and *G1PICK* place the terminal into the mode
                                        ;               indicated by what has been set for the particular graphic
                                        ;               set.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    all

0F7B  C2    12          GOPICK:         CLR     GSEL                    ; Clear select flag
0F7D  E5    1F                          MOV     A, GSET                 ; Get graphic set choosen
0F7F  80    05                          SJMP    GPICK

0F81  D2    12          G1PICK:         SETB    GSEL                    ; Set select flag
0F83  E5    1F                          MOV     A, GSET
0F85  C4                                SWAP    A                       ; Get graphic set choosen

0F86  53    21      4F  GPICK:          ANL     ATTRIBUTES, #01001111B  ; Clean different modes out

0F89  54    0F                          ANL     A, #00001111B           ; Mask to get number of set

0F8B  90    0F92                        MOV     DPTR, #GPICKT           ; Get address of table
0F8E  93                                MOVC    A, @A+DPTR              ; Get value from table
0F8F  42    21                          ORL     ATTRIBUTES, A           ; Set as indicated by table
0F91  22                                RET


                                        ;;              GPICKT - graphic pick table
                                        ;
                                        ;               *GPICKT* contains values that will be ORed into the
                                        ;               attribute byte to set the different character set modes
                                        ;               as indicated by the graphic mode defined above.

0F92                    GPICKT          EQU     $

0F92  00                                DB      00000000B               ; 0 - norm char

0F93  A0                                DB      10100000B               ; 1 - norm char and alt graphic set

0F94  10                                DB      00010000B               ; 2 - alt char set

0F95  B0                                DB      10110000B               ; 3 - alt char and graphic set
```

```
                                        ;;              SBLR - set blink rate
                                        ;
                                        ;               *SBLR* sets the blink rate in increments of 1/30 second.  The
                                        ;               default value is 32/30 second.  If a value of 255 is received
                                        ;               then all blinking data is suppressed.  If a value of 0
                                        ;               is received then all blinking data is continuously on.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    all

0F96  31  D1            SBLR:           ACALL   GNP
0F98  50  02                            JNC     SB1                     ; IF no parameter THEN
0F9A  74  20                            MOV     A, #32                  ;   set to default value

0F9C  F5  37            SB1:            MOV     BLNKRATE, A             ; Set blink rate
0F9E  60  06                            JZ      SB2                     ; IF not zero THEN
0FA0  B4  FF  0C                        CJNE    A, #255, SBRT           ;   IF max value THEN
0FA3  D3                                SETB    C                       ;     flag off and
0FA4  80  01                            SJMP    SB3                     ;     set output

0FA6  C3                SB2:            CLR     C                       ; ELSE IF zero THEN flag on

0FA7  92  01            SB3:            MOV     BLINKF, C               ; Set output of blink
0FA9  E5  20                            MOV     A, XLATCH
0FAB  90  3000                          MOV     DPTR, #MLATCH
0FAE  F0                                MOVX    @DPTR, A                ; Set latch

0FAF  75  38  00        SBRT:           MOV     BLNKCNT, #0             ; Zero count and return
0FB2  22                                RET
```

```
                                ;;              EKI - enable keyboard input
                                ;
                                ;               *EKI* resets the flag which disables input from the keyboard.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

0FB3    C2      1F              EKI:            CLR     KBDISF
0FB5    74      89                              MOV     A, #KBC_EK              ; Enable keyboard
0FB7    61      2A                              AJMP    XPCKB                   ; Output and return


                                ;;              DKI - disable keyboard input
                                ;
                                ;               *DKI* sets the flag which disables input from the keyboard.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

0FB9    D2      1F              DKI:            SETB    KBDISF
0FBB    74      88                              MOV     A, #KBC_DK              ; Disable keyboard
0FBD    61      2A                              AJMP    XPCKB                   ; Output and return
```

```
  1                             ;;      KBCPLK - keyboard caps lock routine
  2                             ;
  3                             ;       *KBCPLK* toggles the keyboard into and out of the caps
  4                             ;       locked mode.
  5                             ;
  6                             ;
  7                             ;       ENTRY   none
  8                             ;
  9                             ;       EXIT    none
 10                             ;
 11                             ;       USES    A, DPTR
 12
 13
 14 0FBF   B2      35            KBCPLK: CPL     CAPLOCKF                ; Toggle caps locked flag
 15 0FC1   74      84                    MOV     A, #KBC_EL
 16 0FC3   20      35      02            JB      CAPLOCKF, KC1
 17 0FC6   74      85                    MOV     A, #KBC_XL
 18 0FC8   71      2A            KC1:    ACALL   XPCKB                   ; Output command to keyboard
 19 0FCA   61      23                    AJMP    XR2_DISP                ; Display status line & return
```

```
                                ;;              KBBREAK - keyboard break key
                                ;
                                ;               *KBBREAK* indicates that the break key on the keyboard has
                                ;               been depressed.  The terminal should then hold the serial
                                ;               output line low for a length of time.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

0FCC  C2    B1          KBBREAK:    CLR     TXD                     ; Set break
0FCE  7A    00                      MOV     INDX1, #0
0FD0  7B    50                      MOV     INDX2, #80
0FD2  12    02B6        BR1:        LCALL   FCKB                    ; REPEAT look at keyboard
0FD5  40    05                      JC      BR2
0FD7  B4    89    02                CJNE    A, #KB_BREAK, BR2       ;   IF another break THEN
0FDA  80    F0                      SJMP    KBBREAK                 ;     start again
0FDC  DA    F4          BR2:        DJNZ    INDX1, BR1              ; UNTIL done
0FDE  DB    F2                      DJNZ    INDX2, BR1              ; UNTIL all done
0FE0  D2    B1                      SETB    TXD                     ; Back to normal
0FE2  22                            RET
```

```
                                        ;;      KBSCROLL = keyboard "NO SCROLL" key
                                        ;
                                        ;       *KBSCROLL* indicates that the "NO SCROLL" key on the
                                        ;       keyboard has been depressed.
                                        ;
                                        ;
                                        ;       ENTRY   none
                                        ;
                                        ;       EXIT    none
                                        ;
                                        ;       USES    all

0FE3  20  36  0A      KBSCROLL:         JB      HSMODEF, KBSCR2         ; IF not hold screen mode THEN
0FE6  74  13                            MOV     A, #XOFF                ;   start with XOFF
0FE8  30  11  02                        JNB     UXOFFSENT, KBSCR1       ;   IF user XOFF last sent THEN
0FEB  74  11                            MOV     A, #XON                 ;     send an XON now instead

0FED  02  022A        KBSCR1:          LJMP    PCOF                    ;   place in output FIFO and return

0FF0  C2  22          KBSCR2:          CLR     HSSTOPF                 ; ELSE allow screen to run
0FF2  05  1E                           INC     HSLINE                  ;   bump line count
0FF4  30  1C  07                       JNB     SHIFTF, KSR             ;   IF shift down THEN
0FF7  E5  18                           MOV     A, BFIX                 ;     calculate number
0FF9  C3                               CLR     C                       ;     between pages
0FFA  95  17                           SUBB    A, TSCROLL
0FFC  F5  1E                           MOV     HSLINE, A
0FFE  22              KSR:             RET
```

1

```
                                        ;;              ROM #2 vectors
                                        ;
                                        ;               These locations are provided to facilitate future modifications
                                        ;               without a change to the masked processor.  Each location is
                                        ;               accessed with a -CALL- and control is gained back with a -RET-.


                                        ;               Initialization of terminal.
                                        ;
                1000                                    ORG     R2_IN
1000    02      10AF                                    JMP     ALLINIT


                                        ;               Soft initialization of terminal.
                                        ;
                1003                                    ORG     R2_IN2
1003    02      10E4                                    JMP     SOFTINIT


                                        ;               Start DMA routine.
                                        ;
                1006                                    ORG     R2_STARTDMA
1006    02      11DC                                    LJMP    STARTDMA


                                        ;               Stop DMA routine.
                                        ;
                1009                                    ORG     R2_STOPDMA
1009    02      11FD                                    LJMP    STOPDMA


                                        ;               Fill screen routine.
                                        ;
                100C                                    ORG     R2_FILL
100C    02      1262                                    LJMP    STFILL


                                        ;               Interrupt request.
                                        ;
                100F                                    ORG     R2_IRQ
100F    02      1329                                    LJMP    IRQ


                                        ;               Serial interrupt.
                                        ;
                1012                                    ORG     R2_SERIAL
1012    02      12CB                                    LJMP    SERIAL


                                        ;               Transmit character from serial port.
                                        ;
                1015                                    ORG     R2_XMT
1015    02      139B                                    LJMP    TRANSMIT

                                        ;               Send XON to host
                                        ;
                1018                                    ORG     R2_XON
```

```
    1 1018    02      13D5                          LJMP    DOXON
    2
    3
    4                                 ;             Initialize the cursor format.
    5                                 ;
    6         101B                                  ORG     R2_ICUR
    7 101B    02      117B                          LJMP    SETCF
    8
    9
   10                                 ;             Initialize the keyboard
   11                                 ;
   12         101E                                  ORG     R2_IKB
   13 101E    02      119E                          LJMP    INITKB
   14
   15
   16                                 ;             Setup mode vector
   17                                 ;
   18         1021                                  ORG     R2_SETUP
   19 1021    02      1417                          LJMP    SETUP
   20
   21
   22                                 ;             Background loop hook.
   23                                 ;
   24         1024                                  ORG     R2_BACK
   25 1024    22                                    RET
   26
   27
   28                                 ;             Keyboard input hook.
   29                                 ;
   30         1027                                  ORG     R2_KY1
   31 1027    22                                    RET
   32
   33
   34                                 ;             Keyboard output hook.
   35                                 ;
   36         102A                                  ORG     R2_KY2
   37 102A    22                                    RET
   38
   39
   40                                 ;             Escape parsing hook.
   41                                 ;
   42         102D                                  ORG     R2_ESC
   43 102D    22                                    RET
   44
   45
   46                                 ;             Transmit character.
   47                                 ;
   48         1030                                  ORG     R2_TC
   49 1030    02      1D8B                          LJMP    SXMTC
   50
   51
   52                                 ;             Transmit line.
   53                                 ;
   54         1033                                  ORG     R2_TL
   55 1033    02      1DA8                          LJMP    SXMTL
   56
   57
```

```
                                        ;               Transmit page.
                                        ;
                1036                                    ORG     R2_TP
1036    02      1DAC                                    LJMP    XMTP

                                        ;               Transmit 25th line.
                                        ;
                1039                                    ORG     R2_T25L
1039    02      1DC6                                    LJMP    XMT25

                                        ;               Print page.
                                        ;
                103C                                    ORG     R2_PRNT
103C    02      1DD2                                    LJMP    PRINT

                                        ;               Display clock.
                                        ;
                103F                                    ORG     R2_DCLK
103F    02      1E1F                                    LJMP    DCLK

                                        ;               Display status.
                                        ;
                1042                                    ORG     R2_DISP
1042    02      1E69                                    LJMP    DISPSTAT
```

```
                                ;;              INIT - initialize terminal
                                ;
                                ;               *INIT* sets up internal RAM from the ROM and EAROM, reconfigures
                                ;               screen tables and initializes processor bytes.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

1045  75  D0  00    INIT:       MOV     PSW, #BANK0             ; Select bank 0

1048  78  0C                    MOV     PTR1, #ISTORE           ; Load constants for
104A  90  1FD1                  MOV     DPTR, #RAMCONST         ;  initialization from ROM
104D  7A  19                    MOV     INDX1, #25

104F  11  A7                    ACALL   INMOV                   ; Move data

1051  78  35                    MOV     PTR1, #XXPOS            ; Load constants for
1053  7A  0B                    MOV     INDX1, #11              ; Second half of ROM constants

1055  11  A7                    ACALL   INMOV                   ; Move data

                                ;               Recall from EAROM
                                ;
1057  C2  03                    CLR     EAROMRCALL
1059  12  1E18                  CALL    WLATCH                  ; Recall from EAROM
105C  D2  03                    SETB    EAROMRCALL
105E  12  1E18                  CALL    WLATCH                  ; By toggling bit

1061  78  25                    MOV     PTR1, #25H              ; Load constants for
1063  90  7000                  MOV     DPTR, #EAROM            ; Loading RAM from EAROM
1066  7A  10                    MOV     INDX1, #16

1068  E0              IN3:      MOVX    A, @DPTR                ; REPEAT get upper nibble
1069  54  0F                    ANL     A, #0FH                 ;   mask data
106B  C4                        SWAP    A
106C  FD                        MOV     TEMP, A                 ;   save upper nibble
106D  A3                        INC     DPTR                    ;   bump EAROM pointer
106E  E0                        MOVX    A, @DPTR                ;   get lower nibble
106F  54  0F                    ANL     A, #0FH                 ;   mask data
1071  4D                        ORL     A, TEMP                 ;   combine upper and lower
1072  F6                        MOV     @PTR1, A                ;   place into internal RAM
1073  A3                        INC     DPTR                    ;   bump EAROM pointer
1074  08                        INC     PTR1                    ;   bump internal RAM pointer
1075  DA  F1                    DJNZ    INDX1, IN3              ; UNTIL done

1077  C2  8C                    CLR     TR0                     ; Turn off timer 0
1079  75  89  23                MOV     TMOD, #ITMOD            ; Initialize values
107C  75  98  5A                MOV     SCON, #ISCON            ; Place UART into mode 1

107F  E5  33                    MOV     A, BAUDRATE             ; Get baud rate
1081  F5  8D                    MOV     TH1, A                  ; Initialize baud rate
1083  B4  53  00                CJNE    A,#83, IN3B
```

```
***  Z-29 COMPUTER TERMINAL FIRMWARE VERSION 1.03  ***

    1 1086  50  02          IN3B:   JNC   IN3A                ; IF baud <= 300 THEN
    2 1088  D2  9F                  SETB  SM0                 ;   place UART into mode 3
    3 108A  D2  8E          IN3A:   SETB  TR1                 ; Turn on baud rate timer
    4
    5 108C  A2  3C                  MOV   C, PRTF2
    6 108E  92  02                  MOV   PORTF, C            ; Initialize port direction
    7 1090  12  1E18                CALL  WLATCH
    8 1093  A2  3D                  MOV   C, ACHR2            ; Get alternate char set flag
    9 1095  92  0C                  MOV   ALTCHARA, C         ; Place in attributes
   10
   11 1097  30  43  02              JNB   HAZ, IN4            ; IF Hazeltine mode THEN
   12 109A  31  D4                  ACALL HAZINIT             ;   initialize for Hazeltine
   13
   14 109C  12  1F22        IN4:    CALL  NVRCHK              ; Checksum the NVRAM
   15 109F  B5  34  01              CJNE  A, NVRSUM, IN5      ; IF same THEN
   16 10A2  22                      RET                       ;   return
   17 10A3  43  40  10      IN5:    ORL   ERRORS, #00010000B  ; ELSE error #4
   18 10A6  22                      RET
   19
   20
   21                       ;
   22                       ;       Move from ROM to RAM using DPTR, PTR1, and INDX1
   23                       ;
   24
   25 10A7  E4              INMOV:  CLR   A                   ; REPEAT
   26 10A8  93                      MOVC  A, @A+DPTR          ;   load constant
   27 10A9  F6                      MOV   @PTR1, A            ;   place into internal RAM
   28 10AA  A3                      INC   DPTR                ;   bump ROM pointer
   29 10AB  08                      INC   PTR1                ;   bump RAM pointer
   30 10AC  DA  F9                  DJNZ  INDX1, INMOV        ; UNTIL done
   31 10AE  22                      RET
```

```
                                ;;              ALLINIT - initialize all of terminal
                                ;
                                ;               *ALLINIT* initializes everything including the CRTC.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

10AF   C2    AF                 ALLINIT:        CLR     EA                      ; Disable interrupts

10B1   75    90    CF                           MOV     P1, #11001111B          ; Initialize port #1

10B4   75    40    00                           MOV     ERRORS, #0              ; Initialize errors here
10B7   12    1EC4                               CALL    KBCHK                   ; Check keyboard first (timing)

10BA   11    45                                 ACALL   INIT                    ; Initialize everything

10BC   E4                                       CLR     A
10BD   78    47                                 MOV     PTR1, #LORDER           ; Setup to initialize
10BF   7A    19                                 MOV     INDX1, #NUMLINES        ;  the -LORDER- table entries

10C1   F6                       ALLI1:          MOV     @PTR1, A                ; REPEAT place entry into table
10C2   08                                       INC     PTR1                    ;   bump RAM pointer
10C3   04                                       INC     A                       ;   bump count
10C4   DA    FB                                 DJNZ    INDX1, ALLI1            ; UNTIL done

10C6   75    B8    03                           MOV     IP, #IPINIT             ; Interrupt priorities

10C9   31    1E                                 ACALL   INITCRT                 ; Start CRT controller
10CB   12    1ED9                               CALL    ROMCHK                  ; Check ROM
10CE   12    1EF1                               CALL    RAMCHK                  ; Check RAM

10D1   75    A8    95                           MOV     IE, #IEINIT             ; Enable interrupts

10D4   12    0BAC                               CALL    CLRS                    ; Clear screen
10D7   75    F0    18                           MOV     B, #MAXLINE
10DA   12    0B96                               CALL    CLRLINE                 ; Clear 25th line
10DD   D2    24                                 SETB    VSPF                    ; Flag screen to be turned on
10DF   12    1F2C                               CALL    ERROR                   ; Display any errors
10E2   21    9E                                 AJMP    INITKB                  ; Initialize keyboard & return
```

```
                                ;;              SOFTINIT - soft initialize terminal
                                ;
                                ;               *SOFTINIT* initializes the terminal but does not destroy the
                                ;               current settings such as caps lock, on/off line, and time on
                                ;               the clock.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

10E4  C0  3A            SOFTINIT:       PUSH    TCNT            ; Save clock information
10E6  C0  3B                            PUSH    TSEC
10E8  C0  3C                            PUSH    TMIN
10EA  C0  3D                            PUSH    THOUR
10EC  A2  2E                            MOV     C, ONLINE
10EE  92  E0                            MOV     ACC.0, C        ; Save online flag
10F0  A2  35                            MOV     C, CAPLOCKF
10F2  92  E1                            MOV     ACC.1, C        ; Save caps locked flag
10F4  C0  E0                            PUSH    ACC

10F6  75  90  CB                        MOV     P1, #11001011B  ; Initialize port #1

10F9  11  45                            ACALL   INIT            ; Initialize everything

10FB  D0  E0                            POP     ACC
10FD  A2  E1                            MOV     C, ACC.1
10FF  92  35                            MOV     CAPLOCKF, C     ; Restore caps locked flag
1101  A2  E0                            MOV     C, ACC.0
1103  92  2E                            MOV     ONLINE, C       ; Restore online flag
1105  D0  3D                            POP     THOUR           ; Restore clock information
1107  D0  3C                            POP     TMIN
1109  D0  3B                            POP     TSEC
110B  D0  3A                            POP     TCNT

110D  31  7B                            ACALL   SETCF           ; Set cursor format

110F  12  0BAC                          CALL    CLRS            ; Clear screen
1112  20  37  06                        JB      L25ON, SOFTI1   ; IF 25th line is off THEN
1115  75  F0  18                        MOV     B, #MAXLINE
1118  02  0B96                          JMP     CLRLINE         ;   clear 25th line

111B  02  1E69            SOFTI1:       JMP     DISPSTAT        ; ELSE display status info
```

```
                                ;;              INITCRT - initialize CRT controler
                                ;
                                ;               *INITCRT* does everything needed to set the CRT controler
                                ;               up and running properly.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    A, DPTR


111E  C0  A8            INITCRT:        PUSH    IE                      ; Save interrupt setup
1120  C2  AF                           CLR     EA                      ; Disable interrupts

1122  75  09  02                       MOV     DMLNA, #2               ; Start at first line to DMA
1125  75  0B  A0                       MOV     DMADRL, #LOW (DMAMEM+160)
1128  75  0A  60                       MOV     DMADRH, #HIGH (DMAMEM+160)

112B  D2  97                           SETB    CLKRUN                  ; Let the clock run

112D  31  52                           ACALL   SETCRT                  ; Set CRT controler

112F  D2  88                           SETB    IT0                     ; Int 0 transition activated
1131  53  88  D1                       ANL     TCON, #11010001B        ; Clear IT1, IE0, IE1, TF0

1134  90  2001                         MOV     DPTR, #DCMDREG
1137  74  E0                           MOV     A, #CSET
1139  F0                              MOVX    @DPTR, A                ; Preset counters

113A  C2  97                           CLR     CLKRUN                  ; Turn off clock

113C  74  20                           MOV     A, #CSTRT
113E  F0                              MOVX    @DPTR, A                ; Start CRT controler

113F  D2  97                           SETB    CLKRUN                  ; Start clock

1141  74  A0                           MOV     A, #CINT
1143  F0                              MOVX    @DPTR, A                ; Enable VRTC interrupts

                                        IFC     NE,DSTAREG-DCMDREG
                                        ERROR   ;'DSTAREG' and 'DCMDREG' are different
                                        ENDC

                                ;       MOV     DPTR, #DSTAREG

1144  E0                              MOVX    A, @DPTR                ; Get status resister
1145  44  B3                           ORL     A, #10110011B           ; Fill in dont cares
1147  64  08                           XRL     A, #00001000B           ; Toggle proper positions
1149  F4                              CPL     A
114A  60  03                           JZ      IC1                     ; IF not zero THEN
114C  43  40  04                       ORL     ERRORS, #00000100B      ;   flag error #2

114F  D0  A8            IC1:           POP     IE                      ; Enable old interrupt status
1151  22                              RET
```

```
                                ;;              SETCRT - set CRT controler (reset 8276)
                                ;
                                ;               *SETCRT* initializes the CRT controler screen parameter bytes.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    A, DPTR

1152  90      2001              SETCRT:         MOV     DPTR, #DSTAREG
1155  E0                                        MOVX    A, @DPTR                ; Clear status register

                                                IFC     NE,DCMDREG-DSTAREG
                                                ERROR   ;'DCMDREG' and 'DSTAREG' are different
                                                ENDC

                                ;               MOV     DPTR, #DCMDREG

1156  74      00                                MOV     A, #CRESET
1158  F0                                        MOVX    @DPTR, A                ; Reset and stop display

1159  90      2000                              MOV     DPTR, #DDATREG
115C  74      4F                                MOV     A, #SCRN1
115E  F0                                        MOVX    @DPTR, A                ; Output screen parameter bytes

115F  74      98                                MOV     A, #SCRN2A
1161  20      39      02                        JB      FREQ, SC1               ; Check for 50/60 Hz
1164  74      DC                                MOV     A, #SCRN2B
1166  F0                      SC1:              MOVX    @DPTR, A

1167  74      99                                MOV     A, #SCRN3
1169  F0                                        MOVX    @DPTR, A

116A  74      4E                                MOV     A, #SCRN4A
116C  20      39      02                        JB      FREQ, SC2               ; Check for 50/60 Hz
116F  74      4F                                MOV     A, #SCRN4B
1171  A2      38                SC2:            MOV     C, CRNBK                ; Get cursor no blink flag
1173  92      E5                                MOV     ACC.5, C
1175  A2      32                                MOV     C, CRUL                 ; Get cursor underline flag
1177  92      E4                                MOV     ACC.4, C
1179  F0                                        MOVX    @DPTR, A
117A  22                                        RET
```

```
                                ;;              SETCF - set cursor format
                                ;
                                ;               *SETCF* initializes the CRT controler for the cursor format
                                ;               indicated.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    A, DPTR

117B  31      52                SETCF:          ACALL   SETCRT                  ; Initialize screen resisters

117D  90      2001                              MOV     DPTR, #DCMDREG
1180  74      20                                MOV     A, #CSTRT
1182  F0                                       MOVX    @DPTR, A                ; Start CRT controler

1183  22                                        RET
```

```
***  Z-29 COMPUTER TERMINAL FIRMWARE VERSION 1.03  ***


                                        ;;              CURSOR - update cursor position
                                        ;
                                        ;               *CURSOR* sends the current cursor address to the CRTC and
                                        ;               disables cursor if data is > 600 baud or display of cursor
                                        ;               is disabled.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    A, DPTR

1184    90      2001            CURSOR: MOV     DPTR, #DCMDREG
1187    74      80                      MOV     A, #CCRSR
1189    F0                              MOVX    @DPTR, A                ; Load cursor position

118A    90      2000                    MOV     DPTR, #DDATREG
118D    E5      11                      MOV     A, XPOS
118F    F0                              MOVX    @DPTR, A                ; Load cursor X-position

1190    E5      12                      MOV     A, YPOS
1192    30      1A      03              JNB     ENBLCUR, CR1            ; IF cursor display disabled OR
1195    30      14      02              JNB     DX, CR2                 ; IF char rcvd since last time
1198    74      1A              CR1:    MOV     A, #NUMLINES + 1        ;   put in non-existing region
119A    F0                      CR2:    MOVX    @DPTR, A                ; ELSE put cursor Y-position

119B    C2      14                      CLR     DX                      ;   and clear -DX-
119D    22                              RET
```

```
                                ;;              INITKB - initialize keyboard
                                ;
                                ;               *INITKB* resets the keyboard to whatever settings the terminal
                                ;               has at the current moment.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all


119E                            INITKB:         IFC     NE,KBC_XL - KBC_EL - 1
                                                ERROR   ;'This code will not work'
                                                ENDC
119E    74      84                              MOV     A, #KBC_EL
11A0    20      35      01                      JB      CAPLOCKF, IKB1          ; IF not caps locked THEN
11A3    04                                      INC     A                       ;   output caps unlocked
11A4    31      C8              IKB1:           ACALL   IKB6                    ;   (save bytes this way)

                                                IFC     NE,KBC_ONLN - KBC_OFFLN - 1
                                                ERROR   ;'This code will not work'
                                                ENDC
11A6    74      8A                              MOV     A, #KBC_OFFLN
11A8    30      2E      01                      JNB     ONLINE, IKB2            ; IF on line THEN
11AB    04                                      INC     A                       ;   output on line command
11AC    31      C8              IKB2:           ACALL   IKB6

                                                IFC     NE,KBC_EK - KBC_DK - 1
                                                ERROR   ;'This code will not work'
                                                ENDC
11AE    74      88                              MOV     A, #KBC_DK
11B0    20      1F      01                      JB      KBDISF, IKB3            ; IF keyboard disabled THEN
11B3    04                                      INC     A                       ;   enable keyboard
11B4    31      C8              IKB3:           ACALL   IKB6

                                                IFC     NE,KBC_DC - KBC_EC - 1
                                                ERROR   ;'This code will not work'
                                                ENDC
11B6    74      82                              MOV     A, #KBC_EC
11B8    20      33      01                      JB      CLKF, IKB4              ; IF key click disabled THEN
11BB    04                                      INC     A                       ;   output no key click
11BC    31      C8              IKB4:           ACALL   IKB6

                                                IFC     NE,KBC_DR - KBC_ER - 1
                                                ERROR   ;'This code will not work'
                                                ENDC
11BE    74      86                              MOV     A, #KBC_ER
11C0    20      34      01                      JB      REPF, IKB5              ; IF auto repeat disabled THEN
11C3    04                                      INC     A                       ;   no auto repeat
11C4    31      C8              IKB5:           ACALL   IKB6

11C6    74      80                              MOV     A, #KBC_ID              ; Have it send it's ID number
11C8    02      026D            IKB6:           JMP     PCKB
```

```
Z29ROM; MICROBENCH 8051 CROSS ASSEMBLER (V1.7-218  24-FEB-83 15:30:08 PAGE 1
***.Z-29.COMPUTER.TERMINAL.FIRMWARE.VERSION.1.03.***

                                ;;              TERMINIT - terminal initialization
                                ;
                                ;               *TERMINIT* initializes the terminal for ZDS and ANSI
                                ;               emulation mode.  This places the terminal in whatever mode
                                ;               the terminal must be in to emulate those modes.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

                                TERMINIT:       ANL     ATTRIBUTES, #0F0H       ; Clear attributes
11CB  53      21      F0        TERMINIT:       ANL     ATTRIBUTES, #0F0H       ; Clear attributes
11CE  D2      1B                                SETB    ERM                     ; Set erasure mode
11D0  D2      25                                SETB    GATM                    ; Set guarded area xmt mode
11D2  80      04                                SJMP    HIR                     ; Clear protected and return


                                ;;              HAZINIT - Hazeltine initialization
                                ;
                                ;               *HAZINIT* initializes the terminal for the Hazeltine 1500
                                ;               emulation mode.  This places the terminal in whatever mode
                                ;               the terminal must be in to emulate the Hazeltine 1500.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

11D4  D2      0B                HAZINIT:        SETB    HALFIA                  ; Starts in background
11D6  D2      1B                                SETB    ERM                     ; Reset erasure mode
11D8  75      3F      00        HIR:            MOV     PFIELD, #0              ; No protected fields
11DB  22                                        RET
```

```
                                ;;              STARTDMA - start psuedo DMA transfer
                                ;
                                ;               *STARTDMA* can only be interrupted by the stop DMA routine.
                                ;               It is initiated by pushing the address to start DMA onto the
                                ;               stack and executing a -RET-.  The hardware takes over and
                                ;               executes an instruction that is hard wired to the data bus
                                ;               and initiates reads from the video memory to the CRT controler.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    none

11DC C0     D0          STARTDMA:   PUSH    PSW             ; Save processor status
11DE C0     E0                      PUSH    ACC
11E0 C0     F0                      PUSH    B

11E2 75     A8      82              MOV     IE, #IEDMA      ; Select interrupts for DMA
11E5 D2     B4                      SETB    DMATYPE         ; Set type for DMA transfer

11E7 E5     09                      MOV     A, DMLNA
11E9 70     05                      JNZ     SD1             ; IF DMA for line #0 THEN
11EB 30     24      02              JNB     VSPF, SD1       ;   IF no suppression THEN
11EE C2     92                      CLR     VSP             ;     always turn video on

11F0 C0     0B          SD1:        PUSH    DMADRL          ; Push DMA start address
11F2 C0     0A                      PUSH    DMADRH          ;  onto the stack
11F4 75     8C      FF              MOV     TH0, #0FFH      ; Initialize timer values
11F7 75     8A      D6              MOV     TL0, #0D6H      ; For 80 characters
11FA D2     8C                      SETB    TR0             ; Start timer running
11FC 32                             RETI                    ; Start psuedo DMA
```

*** Z-29 COMPUTER TERMINAL FIRMWARE VERSION 1.03 ***

```
   1                                ;;              STOPDMA - stop psuedo DMA transfer
   2                                ;
   3                                ;               *STOPDMA* is the other half of the DMA transfer routine.
   4                                ;               It stops the timer, cleans the stack, sets -DMATYPE- to it's
   5                                ;               normal value, and enables interrupts.  If it was filling memory
   6                                ;               then it finishes up that otherwise it continues and calculates
   7                                ;               the next line number, address, and indexes for DMA.
   8                                ;
   9                                ;
  10                                ;               ENTRY   none
  11                                ;
  12                                ;               EXIT    none
  13                                ;
  14                                ;               USES    none
  15
  16
  17 11FD  C2      8C               STOPDMA:        CLR     TR0                     ; Stop timer
  18 11FF  75      A8      95                       MOV     IE, #IEINIT             ; Reset interrupt enables
  19
  20 1202  20      B4      20                       JB      DMATYPE, DM1            ; IF filling memory THEN
  21 1205  D2      B4                               SETB    DMATYPE                 ;   set -DMATYPE- to normal
  22 1207  D0      83                               POP     DPH                     ;   get last address from
  23 1209  D0      82                               POP     DPL                     ;   the stack
  24 120B  AD      F0                               MOV     TEMP, B                 ;   get remainder
  25 120D  BD      00      01                       CJNE    TEMP, #0, FILL          ;   IF nothing left THEN
  26 1210  32                                       RETI                            ;     return
  27 1211  74      20               FILL:           MOV     A, #' '                 ;   ELSE set up and
  28 1213  53      83      CF                       ANL     DPH, #DMAANL
  29 1216  43      83      10                       ORL     DPH, #CHRORL            ;     mask into char memory
  30 1219  A3                       FL1:            INC     DPTR                    ;     REPEAT bump position
  31 121A  F0                                       MOVX    @DPTR, A                ;       fill with spaces
  32 121B  53      83      EF                       ANL     DPH, #ATRANL            ;       mask into attribute mem
  33 121E  F0                                       MOVX    @DPTR, A                ;       clear attributes
  34 121F  43      83      10                       ORL     DPH, #CHRORL            ;       mask into char memory
  35 1222  DD      F5                               DJNZ    TEMP, FL1               ;     UNTIL index1 = 0
  36 1224  32                                       RETI                            ;   Return
  37
  38
  39 1225  75      D0      08       DM1:            MOV     PSW, #BANK1             ; ELSE select bank #1
  40 1228  D0      E0                               POP     ACC                     ;   clean off stack
  41 122A  D0      E0                               POP     ACC
  42
  43 122C  09                                       INC     DMLN                    ;   bump current DMA line
  44
  45 122D  20      39      13                       JB      FREQ, DM1A              ;   IF 50 Hz THEN
  46 1230  B9      1D      02                       CJNE    DMLN, #29, DM1B         ;     IF line 29 THEN
  47 1233  80      13                               SJMP    DM2                     ;       wrap around
  48 1235  B9      19      00       DM1B:           CJNE    DMLN,#NUMLINES, DM1C    ;     ELSE IF line <= 25 THEN
  49 1238  40      10               DM1C:           JC      DM3                     ;       do usual stuff
  50 123A  B9      1A      00                       CJNE    DMLN,#NUMLINES+1, DM1D
  51 123D  40      1C               DM1D:           JC      DM4                     ;     ELSE IF line >= 26 THEN
  52 123F  D2      92                               SETB    VSP                     ;       begin blanking
  53 1241  80      18                               SJMP    DM4
  54
  55 1243  B9      19      00       DM1A:           CJNE    DMLN,#NUMLINES, DM1E    ; Check for wrap around
  56 1246  40      02               DM1E:           JC      DM3                     ; IF line > 25 THEN
  57 1248  79      00               DM2:            MOV     DMLN, #0                ;   wrap around
```

```
        1
        2 124A    74      47              DM3:            MOV     A, #LORDER              ; ELSE get base of table
        3 124C    29                                      ADD     A, DMLN                 ;   add in offset
        4 124D    F8                                      MOV     DMPTR, A                ;   move it into pointer
        5 124E    E6                                      MOV     A, @DMPTR               ;   get value from table
        6
        7 124F    75      F0      50                      MOV     B, #NUMCHARS            ;   set up for multiply
        8 1252    A4                                      MUL     AB                      ;   do it
        9 1253    F5      0B                              MOV     DMADRL, A               ;   save low order byte
       10 1255    E5      F0                              MOV     A, B                    ;   get high order byte
       11 1257    24      60                              ADD     A, #HIGH (DMAMEM)       ;   add offset
       12 1259    F5      0A                              MOV     DMADRH, A               ;   save high order byte
       13
       14 125B    D0      F0              DM4:            POP     B                       ; Restore registers and return
       15 125D    D0      E0                              POP     ACC
       16 125F    D0      D0                              POP     PSW
       17 1261    32                                      RETI
```

*** Z-29 COMPUTER TERMINAL FIRMWARE VERSION 1.03 ***

```
                                ;;      STFILL - start filling with spaces
                                ;
                                ;       *STFILL* takes the number of characters and location in
                                ;       screen memory and fills them with 20H and places the
                                ;       attribute given into attribute memory.  It takes into
                                ;       account the mode that ERM is in.
                                ;
                                ;
                                ;       ENTRY   (A) = number of characters less than 127
                                ;               (B) = attribute to write
                                ;               (DPTR) = character memory address to start from
                                ;
                                ;       EXIT    none
                                ;
                                ;       USES    A, B, DPTR, WORK2, TEMP, INDX2, TIMER 0


1262 FB                 STFILL: MOV     INDX2, A        ; Save count
1263 AF F0                      MOV     WORK2, B        ; Save attributes

1265 BF 00 3D                   CJNE    WORK2, #0, SFB  ; IF attributes THEN special

1268 30 1B 04                   JNB     ERM, SFA        ; IF not erasure mode THEN

126B E5 3F                      MOV     A, PFIELD       ;   IF protected fields THEN
126D 70 36                      JNZ     SFB             ;     special

126F 53 83 EF           SFA:    ANL     DPH, #ATRANL    ;   convert to DMA memory
1272 43 83 20                   ORL     DPH, #DMAORL
1275 EB                         MOV     A, INDX2        ;   get count back
1276 23                         RL      A               ;   multiply by 2
1277 75 F0 03                   MOV     B, #3           ;   set up for divide by 3
127A 84                         DIV     AB              ;   place remainder in (B)
127B C5 F0                      XCH     A, B            ;   switch to work with (B)
127D B4 02 06                   CJNE    A, #2, SF1      ;   IF remainder = 2 THEN
1280 05 F0                      INC     B               ;     bump clock value
1282 74 00                      MOV     A, #0           ;     remainder := 0
1284 80 07                      SJMP    SF2             ;   ELSE

1286 B4 00 04           SF1:    CJNE    A, #0, SF2      ;     IF remainder = 0 THEN
1289 74 02                      MOV     A, #2           ;       remainder := 2
128B 15 F0                      DEC     B               ;       decrement clock count
128D C5 F0              SF2:    XCH     A, B            ;   switch back
128F F4                         CPL     A               ;   take 2's complement of (A)
1290 04                         INC     A

1291 C2 AF                      CLR     EA              ;   disable interrupts
1293 00                         NOP                     ;   give time to clear
1294 75 A8 82                   MOV     IE, #IEDMA      ;   init for DMA
1297 C2 B4                      CLR     DMATYPE         ;   set type for clear memory
1299 C0 82                      PUSH    DPL             ;   push DMA start address
129B C0 83                      PUSH    DPH             ;    onto the stack
129D 75 8C FF                   MOV     TH0, #0FFH      ;   set timer 0
12A0 F5 8A                      MOV     TL0, A          ;   for number of characters
12A2 D2 8C                      SETB    TR0             ;   start timer running
12A4 22                         RET                     ;   start DMA
```

```
    1 12A5  E5      3F            SFB:    MOV     A, PFIELD           ; ELSE set protected fields
    2 12A7  B4      FE      01            CJNE    A, #254, SF3        ;   ELSE IF none protected THEN
    3 12AA  E4                            CLR     A                   ;     set for none
    4 12AB  FD                    SF3:    MOV     TEMP, A             ;   save value in -TEMP-
    5
    6 12AC  53      83      EF    SFL:    ANL     DPH, #ATRANL        ;   REPEAT mask to attribute
    7 12AF  E0                            MOVX    A, @DPTR            ;     get current attribute
    8 12B0  54      0F                    ANL     A, #00001111B       ;     mask data
    9 12B2  70      05                    JNZ     SF4                 ;     IF no attributes THEN
   10 12B4  BD      80      08            CJNE    TEMP, #80H, SF5     ;       IF protected THEN skip
   11 12B7  80      0E                    SJMP    SF6                 ;       ELSE
   12 12B9  5D                    SF4:    ANL     A, TEMP             ;         mask with protected
   13 12BA  B5      05      02            CJNE    A, TEMPA, SF5       ;         IF protected THEN
   14 12BD  80      08                    SJMP    SF6                 ;           skip
   15
   16 12BF  EF                    SF5:    MOV     A, WORK2            ;     set attribute to write
   17 12C0  F0                            MOVX    @DPTR, A            ;     write attribute out
   18 12C1  43      83      10            ORL     DPH, #CHRORL        ;     mask to character memory
   19 12C4  74      20                    MOV     A, #' '             ;     set space to write
   20 12C6  F0                            MOVX    @DPTR, A            ;     clear character
   21 12C7  A3                    SF6:    INC     DPTR                ;     bump pointer
   22 12C8  DB      E2                    DJNZ    INDX2, SFL          ;   UNTIL done
   23 12CA  22                            RET
```

```
                                ;;              SERIAL - serial port interrupt routine
                                ;
                                ;               *SERIAL* processes all serial port interrupts.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    none

12CB  C0  D0            SERIAL: PUSH    PSW             ; Save registers
12CD  C0  E0                    PUSH    ACC
12CF  C0  F0                    PUSH    B
12D1  C0  82                    PUSH    DPL
12D3  C0  83                    PUSH    DPH
12D5  20  98  05                JB      RI, SERIN       ; Check receiver interrupt
12D8  20  99  38                JB      TI, SEROUT      ; Check transmit interrupt
12DB  80  3A                    SJMP    SRT             ; This should never happen

12DD  75  39  00        SERIN:  MOV     AUTOCNT, #0
12E0  D2  24                    SETB    VSPF            ; Change auto off
12E2  C2  98                    CLR     RI              ; Clear receiver int flag
12E4  E5  99                    MOV     A, SBUF         ; Get character
12E6  54  7F                    ANL     A, #01111111B   ; Mask off parity
12E8  20  43  02                JB      HAZ, SIO        ; IF not Hazeltine THEN
12EB  60  2A                    JZ      SRT             ;   IF null THEN exit

12ED  20  23  03        SIO:    JB      PRNTF, SI1      ; IF not printing THEN
12F0  30  2E  24                JNB     ONLINE, SRT     ;   IF offline THEN exit
12F3  B4  13  0B        SI1:    CJNE    A, #XOFF, SI2   ; IF -XOFF- received THEN
12F6  20  23  06                JB      PRNTF, SI1A     ;   IF not printing THEN
12F9  20  42  05                JB      ADM3, SI2       ;     IF ADM 3A THEN skip
12FC  20  43  02                JB      HAZ, SI2        ;     IF Hazeltine THEN skip
12FF  D2  16            SI1A:   SETB    XOFFRCVED       ;   flag for XOFF
1301  B4  11  02        SI2:    CJNE    A, #XON, SI3    ; IF -XON- received THEN
1304  C2  16                    CLR     XOFFRCVED       ;   flag for no XOFF
1306  20  23  0E        SI3:    JB      PRNTF, SRT      ; IF printing THEN exit
1309  12  01CA                  LCALL   PCIF            ; Put character in input FIFO
130C  30  D5  08                JNB     F0, SRT         ; IF -XOFF- needed THEN
130F  71  C4                    ACALL   DOXOFF          ;   send an -XOFF- and exit
1311  80  04                    SJMP    SRT

1313  C2  99            SEROUT: CLR     TI              ; Clear transmit int flag
1315  D2  13                    SETB    OKTRANS         ; Set ok to transmit

1317  D0  83            SRT:    POP     DPH             ; Restore registers
1319  D0  82                    POP     DPL
131B  D0  F0                    POP     B
131D  D0  E0                    POP     ACC
131F  D0  D0                    POP     PSW
1321  71  26                    ACALL   SRT1            ; Double check
1323  D2  AF                    SETB    EA
1325  22                       RET
1326  C2  AF            SRT1:   CLR     EA
1328  32                       RETI                    ; And return
```

```
                                        ;;              IRQ - interrupt request
                                        ;
                                        ;               *IRQ* is executed when the CRTC starts to display the 25th
                                        ;               line.  At this point the DMA line number must have been set
                                        ;               to zero or some error has occured.  This routine is a safety
                                        ;               check for any such errors and will take at most one frome to
                                        ;               straighten out any such errors.  It also takes care of any
                                        ;               clock timing and blinking.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    none

1329  C0  D0              IRQ:          PUSH    PSW             ; Save registers
132B  C0  E0                          PUSH    ACC
132D  C0  F0                          PUSH    B
132F  C0  82                          PUSH    DPL
1331  C0  83                          PUSH    DPH

1333  75  09  00                      MOV     DMLNA, #0       ; Safety check

1336  31  84                          ACALL   CURSOR          ; Display cursor

1338  90  2001                        MOV     DPTR, #DSTAREG
133B  E0                              MOVX    A, @DPTR        ; Clear any CRTC flags

133C  05  38                          INC     BLNKCNT         ; Bump blink count
133E  E5  37                          MOV     A, BLNKRATE
1340  60  10                          JZ      IRQ2            ; IF not zero OR
1342  B4  FF  02                      CJNE    A, #0FFH, IRQ1  ; IF not full count THEN
1345  80  0B                          SJMP    IRQ2
1347  B5  38  08      IRQ1:           CJNE    A, BLNKCNT, IRQ2        ;   IF time to blink THEN
134A  75  38  00                      MOV     BLNKCNT, #0             ;     reset count
134D  B2  01                          CPL     BLINKF                  ;     toggle blink flag
134F  12  1E18                        CALL    WLATCH

1352  05  3A              IRQ2:       INC     TCNT            ; Bump time count
1354  E5  3A                          MOV     A, TCNT
1356  20  39  05                      JB      FREQ, IRQ3      ; IF 50 Hz AND 1 second OR
1359  B4  32  34                      CJNE    A, #50, IRQ5
135C  80  03                          SJMP    IRQ4            ; IF 60 Hz AND 1 second THEN
135E  B4  3C  2F      IRQ3:           CJNE    A, #60, IRQ5
1361  75  3A  00      IRQ4:           MOV     TCNT, #0        ;   initialize count
1364  05  3B                          INC     TSEC            ;   bump seconds
1366  E5  3B                          MOV     A, TSEC
1368  B4  3C  25                      CJNE    A, #60, IRQ5    ;   IF one minute THEN
136B  30  3B  0B                      JNB     SCRNSAVE, IRQ4A ;     IF screen saver THEN
136E  05  39                          INC     AUTOCNT         ;       bump auto off
1370  E5  39                          MOV     A, AUTOCNT
1372  B4  0F  04                      CJNE    A, #15, IRQ4A   ;       IF saver time up THEN
1375  C2  24                          CLR     VSPF            ;         flag it is off
1377  D2  92                          SETB    VSP             ;         and turn off display
```

```
    1 1379   75   3B   00     IRQ4A:   MOV   TSEC, #0          ;     initialize seconds
    2 137C   05   3C                   INC   TMIN              ;     bump minutes
    3 137E   E5   3C                   MOV   A, TMIN
    4 1380   B4   3C   0D              CJNE  A, #60, IRQ5      ;     IF one hour THEN
    5 1383   75   3C   00              MOV   TMIN, #0          ;       initialize minutes
    6 1386   05   3D                   INC   THOUR             ;       bump hours
    7 1388   E5   3D                   MOV   A, THOUR
    8 138A   B4   18   03              CJNE  A, #24, IRQ5      ;       IF 24 hours THEN
    9 138D   75   3D   00              MOV   THOUR, #0         ;         initialize hours
   10
   11 1390   D0   83           IRQ5:   POP   DPH               ; Restore registers and return
   12 1392   D0   82                   POP   DPL
   13 1394   D0   F0                   POP   B
   14 1396   D0   E0                   POP   ACC
   15 1398   D0   D0                   POP   PSW
   16 139A   32                       RETI
```

```
                                        ;;              TRANSMIT - transmit to host
                                        ;
                                        ;               *TRANSMIT* waits for an empty output holding register, sets
                                        ;               parity, and outputs the character to the serial port.
                                        ;
                                        ;
                                        ;               ENTRY   (A) = character to output
                                        ;
                                        ;               EXIT    (A) = character with correct parity
                                        ;
                                        ;               USES    A

139B  20  16  FD      TRANSMIT:         JB      XOFFRCVED, TRANSMIT     ; Wait until no -XOFF-

139E  C3                      TRANS:    CLR     C                       ; Start with no parity
139F  30  28  0A                        JNB     PRTYENA, TRA            ; IF parity enabled THEN
13A2  D3                                SETB    C                       ;   do set parity
13A3  20  29  06                        JB      PRTYSTK, TRA            ;   IF not stick parity THEN
13A6  A2  D0                            MOV     C, P                    ;     do even parity
13A8  20  2A  01                        JB      PRTYEVN, TRA            ;     IF not even parity THEN
13AB  B3                                CPL     C                       ;       do odd parity
13AC  92  E7                  TRA:      MOV     ACC.7, C                ; Put in parity bit

13AE  20  13  03      TRW:              JB      OKTRANS, TROK           ; IF ok to transmit OR
13B1  30  99  FA                        JNB     TI, TRW                 ; IF transmit buffer empty THEN
13B4  20  02  03      TROK:             JB      PORTF, TROK1            ;   IF auxiliary port OR
13B7  30  3E  03                        JNB     HNDSHK, TROK2           ;   IF hardware handshake THEN
13BA  20  B5  F1      TROK1:            JB      CTS, TRW                ;     IF not CTS THEN do again

13BD  C2  99          TROK2:            CLR     TI                      ;   clear buffer empty flag
13BF  C2  13                            CLR     OKTRANS                 ;   make not ok to transmit
13C1  F5  99                            MOV     SBUF, A                 ;   output data
13C3  22                                RET
```

```
                                        ;;              DOXOFF - do send XOFF to host
                                        ;
                                        ;               *DOXOFF* preempts outgoing data and outputs an XOFF to host
                                        ;               setting the proper flags.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    A

13C4  20      10      1A        DOXOFF: JB      XOFFSENT, SXR           ; IF XOFF already sent THEN exit
13C7  D2      10                        SETB    XOFFSENT                ; Flag that handshake was done

13C9  30      3E      05                JNB     HNDSHK, XMTXOFF         ; IF hardware handshake THEN
13CC  D2      95                        SETB    RTS                     ;   set handshake lines
13CE  D2      94                        SETB    DTR
13D0  22                                RET

13D1  74      13        XMTXOFF:        MOV     A, #XOFF                ; ELSE output an XOFF
13D3  61      9E                        AJMP    TRANS                   ;   transmit it and return


                                        ;;              DOXON - do send XON to host
                                        ;
                                        ;               *DOXON* preempts outgoing data and outputs an XON to host
                                        ;               checking that it does not override any user XOFF.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    A

13D5  30      10      09        DOXON:  JNB     XOFFSENT, SXR           ; IF no XOFF sent THEN exit
13D8  C2      10                        CLR     XOFFSENT                ; Clear flag on handshake

13DA  30      3E      05                JNB     HNDSHK, SXON1           ; IF hardware handshake THEN
13DD  C2      95                        CLR     RTS                     ;   clear handshake lines
13DF  C2      94                        CLR     DTR
13E1  22                        SXR:    RET

13E2  20      11      FC        SXON1:  JB      UXOFFSENT, SXR          ; IF no user XOFF THEN
13E5  74      11                XMTXON: MOV     A, #XON                 ;   send an XON
13E7  61      9E                        AJMP    TRANS                   ;   transmit it and return
```

```
                                ;;              LOOKUP - table lookup and jump
                                ;
                                ;               *LOOKUP* looks for the entry in the table passed to it.  If
                                ;               found the return address is removed and control given to the
                                ;               routine.  If not found then the main table is searched.  If
                                ;               found the return address is removed and control given to the
                                ;               routine.  If not found then control is given back to the
                                ;               calling routine.
                                ;
                                ;
                                ;               ENTRY   (B) = table length
                                ;                       (DPTR) = beginning of table
                                ;
                                ;               EXIT    return if not found in tables
                                ;
                                ;               USES    all

13E9  91  04            LOOKUP: ACALL   XSTABX          ; Search users table
13EB  50  0B                    JNC     LKUP1           ; IF not found THEN
13ED  75  F0  13                MOV     B, #LOTL
13F0  90  1432                  MOV     DPTR, #LOT
13F3  91  04                    ACALL   XSTABX          ;   search main table
13F5  50  01                    JNC     LKUP1           ;   IF not found THEN
13F7  22                        RET                     ;     return

13F8  D0  E0            LKUP1:  POP     ACC             ;   ELSE found THEN
13FA  D0  E0                    POP     ACC             ;     remove return address
13FC  E4                        CLR     A
13FD  73                        JMP     @A+DPTR         ;     jump to routine
```

```
                                        ;;      XSTX - transmit string to line 25 following CALL
                                        ;
                                        ;       *XSTX* is a jump to the routine *XST*.  It is used to save
                                        ;       bytes so that routines may use a short jump to here instead
                                        ;       of a long jump to the other routine.

 13FE   02      1B9D            XSTX:   LJMP    XST                     ; Jump to routine

                                        ;;      XSTATX - transmit string to line 25
                                        ;
                                        ;       *XSTATX* is a jump to the routine *XSTAT*.  It is used to
                                        ;       save bytes so that routines may use a short jump to here
                                        ;       instead of a long jump to the other routine.

 1401   02      1BA8            XSTATX: LJMP    XSTAT                   ; Jump to routine

                                        ;;      XSTABX - search table
                                        ;
                                        ;       *XSTABX* is a jump to the routine *STAB*.  It is used to
                                        ;       save bytes so that routines may use a short jump to here
                                        ;       instead of a long jump to the other routine.

 1404   02      013A            XSTABX: LJMP    STAB                    ; Jump to routine
```

```
                                        ;;              SFCKB - setup fetch character from keyboard
                                        ;
                                        ;               *SFCKB* waits for characters from the keyboard.  It does
                                        ;               not return until a character is returned.  All access codes
                                        ;               and the characters that follow are dropped.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    (A) = character from keyboard
                                        ;
                                        ;               USES    A, B, DPTR

1407    12      02B6            SFCKB:  CALL    FCKB                    ; Wait for key
140A    40      FB                      JC      SFCKB
140C    B4      9F      07              CJNE    A, #KB_ACC, SFCKB2      ; IF access code THEN
140F    12      02B6            SFCKB1: CALL    FCKB                    ;   drop next character
1412    40      FB                      JC      SFCKB1
1414    80      F1                      SJMP    SFCKB                   ;   and keep on going
1416    22                      SFCKB2: RET                             ; ELSE return
```

```
                                        ;;              SETUP1 - setup mode line #1
                                        ;
                                        ;               *SETUP1* runs the terminal setup mode program.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    all

1417    20      93      01      SETUP:          JB      SETLCK, SETUP0          ; Check for locked setup
141A    22                              RET

141B    12      1B2C            SETUP0:         CALL    SCLRLR                  ; Setup clear line reverse

141E    91      6E              SETUP1:         ACALL   L1                      ; Display first line
1420    91      07              STP1A:          ACALL   SFCKB                   ; Fetch character from keyboard
1422    75      F0      01                      MOV     B, #L1TL
1425    90      146B                            MOV     DPTR, #L1T
1428    71      E9                              ACALL   LOOKUP                  ; Lookup command
142A    80      F4                              SJMP    STP1A                   ; Keep on going


                                        ;               Toggle on/off line
                                        ;
142C    B2      2E              STP1B:          CPL     ONLINE                  ; Toggle on/off line
142E    31      9E                              CALL    INITKB                  ; Reconfigur keyboard
1430    80      EC                              SJMP    SETUP1


                                        ;;              LOT - every line table
                                        ;
                                        ;               *LOT* contains the jump table for every line.

        1432                    LOT             EQU     $

1432    0D                              DB      CR
1433    141E                            DW      SETUP1                  ; Main setup line

1435    61                              DB      'a'
1436    141E                            DW      SETUP1                  ; Main setup line

1438    41                              DB      'A'
1439    141E                            DW      SETUP1                  ; Main setup line

143B    74                              DB      't'
143C    14D7                            DW      SETUP2                  ; Set tabs

143E    54                              DB      'T'
143F    14D7                            DW      SETUP2                  ; Set tabs

1441    62                              DB      'b'
1442    1542                            DW      SETUP3                  ; Protocol #1
```

```
                1
1444  42                        DB      'B'
1445  1542                      DW      SETUP3                  ; Protocol #1
                4
1447  63                        DB      'c'
1448  16B0                      DW      SETUP4                  ; Protocol #2
                7
144A  43                        DB      'C'
144B  16B0                      DW      SETUP4                  ; Protocol #2
               10
144D  64                        DB      'd'
144E  1795                      DW      SETUP5                  ; Misc #1
               13
1450  44                        DB      'D'
1451  1795                      DW      SETUP5                  ; Misc #1
               16
1453  65                        DB      'e'
1454  1865                      DW      SETUP6                  ; Keyboard
               19
1456  45                        DB      'E'
1457  1865                      DW      SETUP6                  ; Keyboard
               22
1459  66                        DB      'f'
145A  18F4                      DW      SETUP7                  ; Misc #2
               25
145C  46                        DB      'F'
145D  18F4                      DW      SETUP7                  ; Misc #2
               28
145F  67                        DB      'g'
1460  19DD                      DW      SETUP8                  ; Misc #3
               31
1462  47                        DB      'G'
1463  19DD                      DW      SETUP9                  ; Misc #3
               34
1465  88                        DB      KB_SETUP                ; Setup key
1466  1AF6                      DW      SETUPRET                ; Return
               37
1468  C8                        DB      KB_SETUP + KB_SHFT      ; Shift setup key
1469  1B02                      DW      SETUPSAVE               ; Save and return
               40
0013                    LOTL    EQU     ($-LOT)/3               ; Table #0 length
               42
               43
               44
               45
                               ;;              L1T - line #1 table
                               ;
                               ;               *L1T* contains the jump table for line #1 in setup mode.
               49
146B                    L1T     EQU     $
               51
146B  31                        DB      '1'                     ; Option #1
146C  142C                      DW      STP1B                   ; Toggle on/off line
               54
0001                    L1TL    EQU     ($-L1T)/3               ; Table length
               56
               57
```

```
                                ;;              L1 - line #1
                                ;
                                ;               *L1* displays setup information for line #1.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

146E  E4                        L1:             CLR     A                       ; Position

146F  71      FE                                ACALL   XSTX
1471  2A      2A      20                        DB      '** SETUP MENU A **  (Ver 1.03)   1.', ('.'+80H)
1474  53      45      54
1477  55      50      20
147A  4D      45      4E
147D  55      20      41
1480  20      2A      2A
1483  20      20      28
1486  56      65      72
1489  20      31      2E
148C  30      33      29
148F  20      20      20
1492  31      2E      A0

1495  90      14C7                              MOV     DPTR, #LN1B1
1498  20      2E      03                        JB      ONLINE, L1A             ; Get proper message for
149B  90      14CF                              MOV     DPTR, #LN1B2            ;  on/off line
149E  91      01              L1A:              ACALL   XSTATX                  ; Next part

14A0  71      FE                                ACALL   XSTX
14A2  20      20      20                        DB      '    MENUS -A- to -G- or -T- for TAB', ('S'+80H)
14A5  20      4D      45
14A8  4E      55      53
14AB  20      2D      41
14AE  2D      20      74
14B1  6F      20      2D
14B4  47      2D      20
14B7  6F      72      20
14BA  2D      54      2D
14BD  20      66      6F
14C0  72      20      54
14C3  41      42      D3

14C6  22                                        RET

                                ;               Line #1 setup mode messages
                                ;
14C7  6F      6E      20        LN1B1:          DB      'on line', (' '+80H)
14CA  6C      69      6E
14CD  65      A0
14CF  6F      66      66        LN1B2:          DB      'off lin', ('e'+80H)
```

```
          14D2    20      6C      69
          14D5    6E      E5
```

*** Z-29 COMPUTER TERMINAL FIRMWARE VERSION 1.03 ***

```
                                ;;              SETUP2 - setup mode line #2
                                ;
                                ;               *SETUP2* is the setup mode line for set/clear tabs.
                                ;               -PTR2- is used as the cursor position.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

14D7  79    00          SETUP2:     MOV     PTR2, #0                ; -PTR2- is cursor position
14D9  B1    0D          STP2A:      ACALL   L2                      ; Display line #2
14DB  91    07          STP2B:      CALL    SFCKB                   ; Fetch character from keyboard
14DD  75    F0    04                MOV     B, #L2TL
14E0  90    1501                    MOV     DPTR, #L2T
14E3  71    E9                      ACALL   LOOKUP                  ; Look up command
14E5  80    F4                      SJMP    STP2B                   ; Keep on going

                                ;               Move left
                                ;
14E7  19                    STP2C:      DEC     PTR2                    ; Decrement
14E8  B9    FF    01                    CJNE    PTR2, #(0-1), STP2C1    ; IF wrap around THEN
14EB  09                              INC     PTR2                    ;   bump back up
14EC  80    EB              STP2C1:     SJMP    STP2A                   ;   keep going

                                ;               Move right
                                ;
14EE  09                    STP2D:      INC     PTR2                    ; Bump
14EF  B9    50    01                    CJNE    PTR2, #80, STP2D1       ; IF wrap around THEN
14F2  19                              DEC     PTR2                    ;   bump back down
14F3  80    E4              STP2D1:     SJMP    STP2A                   ;   keep going

                                ;               Set tab
                                ;
14F5  D2    D5              STP2E:      SETB    F0
14F7  80    02                      SJMP    STP2F1                  ; Set tab

                                ;               Clear tab
                                ;
14F9  C2    D5              STP2F:      CLR     F0
14FB  E9                    STP2F1:     MOV     A, PTR2                 ; Get position
14FC  12    05A2                    CALL    SETTAB                  ; Set/clear tab
14FF  80    D8                      SJMP    STP2A                   ; Keep going


                                ;;              L2T - line #2 table
                                ;
                                ;               *L2T* contains the jump table for line #2 in setup mode.
```

```
                1501                    L2T             EQU     $
                1501    82                              DB      KB_LEFT                 ; Cursor left
                1502    14E7                            DW      STP2C                   ; Move left
                1504    83                              DB      KB_RIGHT                ; Cursor right
                1505    14EE                            DW      STP2D                   ; Move right
                1507    80                              DB      KB_UP                   ; Cursor up
                1508    14F5                            DW      STP2E                   ; Set tab
                150A    81                              DB      KB_DOWN                 ; Cursor down
                150B    14F9                            DW      STP2F                   ; Clear tab
                0004                    L2TL            EQU     ($-L2T)/3               ; Table length
                                        ;;              L2 - line #2
                                        ;
                                        ;               *L2* displays setup information for line #2 set/clear tabs.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    all
                150D    90      5780            L2:     MOV     DPTR, #L25MEM           ; Work with 25th line
                1510    7D      00                      MOV     TEMP, #0                ; Counter variable
                1512    78      29                      MOV     PTR1, #TABTAB           ; Point to tab table
                1514    7A      0A                      MOV     INDX1, #10              ; 10 bytes
                1516    E6                      L2A:    MOV     A, @PTR1                ; REPEAT set byte
                1517    7B      08                      MOV     INDX2, #8               ;   8 bits
                1519    FE                              MOV     WORK1, A                ;   save value for later
                151A    ED                      L2B:    MOV     A, TEMP                 ;   REPEAT
                151B    04                              INC     A                       ;     set and adjust position
                151C    75      F0      0A              MOV     B, #10                  ;     calculate character
                151F    84                              DIV     AB
                1520    E5      F0                      MOV     A, B
                1522    24      30                      ADD     A, #'0'                 ;     convert to ASCII
                1524    FF                              MOV     WORK2, A                ;     and save it
                1525    ED                              MOV     A, TEMP
                1526    B5      01      02              CJNE    A, PTR2A, L2C           ;     IF cursor = position THEN
                1529    7F      09                      MOV     WORK2, #09H             ;       change to cursor char
                152B    EE                      L2C:    MOV     A, WORK1                ;     restore tab table value
                152C    33                              RLC     A
                152D    FE                              MOV     WORK1, A                ;     replace new value
                152E    50      08                      JNC     L2D                     ;     IF tab is set here THEN
                1530    7F      00                      MOV     WORK2, #00H             ;       change to tab char
                1532    ED                              MOV     A, TEMP
```

```
    1 1533   B5      01      02                      CJNE    A, PTR2A, L2D           ;     IF cursor = position THEN
    2 1536   7F      02                              MOV     WORK2, #02H             ;       change to both char
    3 1538   EF                              L2D:   MOV     A, WORK2                ;     set final character
    4 1539   F0                                      MOVX    @DPTR, A                ;     place into memory
    5 153A   A3                                      INC     DPTR                    ;     bump pointer
    6 153B   0D                                      INC     TEMP                    ;     bump count
    7 153C   DB      DC                              DJNZ    INDX2, L2B              ;   UNTIL all bits
    8 153E   08                                      INC     PTR1                    ;   bump tab table pointer
    9 153F   DA      D5                              DJNZ    INDX1, L2A              ; UNTIL all bytes
   10 1541   22                                      RET
```

```
                                ;;              SETUP3 - setup mode line #3
                                ;
                                ;               *SETUP3* runs the terminal setup mode to change baud rate,
                                ;               parity, duplex, and handshake.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

1542  B1      98        SETUP3: ACALL   L3              ; Display third line
1544  91      07        STP3A:  ACALL   SFCKB           ; Fetch character from keyboard
1546  75      F0  04            MOV     B, #L3TL
1549  90      158C              MOV     DPTR, #L3T
154C  71      E9                ACALL   LOOKUP          ; Look up command
154E  80      F4                SJMP    STP3A           ; Keep on going

                                ;               Change baud rate
                                ;
1550  E5      33        STP3B:  MOV     A, BAUDRATE     ; Search for current baud rate
1552  75      F0  0E            MOV     B, #L3ATL
1555  90      161E              MOV     DPTR, #L3AT
1558  91      04                ACALL   XSTABX
155A  40      06                JC      STP3B1          ; IF found THEN
155C  E5      F0                MOV     A, B            ;   set count + 1
155E  B4      0E  01            CJNE    A, #L3ATL, STP3B1 ;   IF time for wrap THEN
1561  E4                        CLR     A               ;     zero
1562  12      0E4B      STP3B1: CALL    SBR             ; Set baud rate
1565  80      DB                SJMP    SETUP3

                                ;               Change parity
                                ;
1567  20      28  08    STP3C:  JB      PRTYENA, STP3C1 ; IF disabled THEN
156A  D2      28                SETB    PRTYENA         ;   enable parity
156C  C2      29                CLR     PRTYSTK         ;   normal parity
156E  C2      2A                CLR     PRTYEVN         ;   odd parity
1570  80      D0                SJMP    SETUP3
1572  20      29  07    STP3C1: JB      PRTYSTK, STP3C2 ; IF normal parity AND
1575  20      2A  08            JB      PRTYEVN, STP3C3 ; IF odd parity THEN
1578  D2      2A                SETB    PRTYEVN         ;   make even parity
157A  80      C6                SJMP    SETUP3
157C  C2      28        STP3C2: CLR     PRTYENA         ; IF stick THEN disable parity
157E  80      C2                SJMP    SETUP3
1580  D2      29        STP3C3: SETB    PRTYSTK         ; IF even THEN stick parity
1582  80      BE                SJMP    SETUP3

                                ;               Toggle duplex
                                ;
1584  B2      2F        STP3D:  CPL     FULLDPLX        ; Toggle full/half duplex
1586  80      BA                SJMP    SETUP3
```

```
                                        ;               Toggle handshake
                                        ;
1588    B2      3E              STP3E:  CPL     HNDSHK                  ; Toggle soft/hard handshake
158A    80      B6                      SJMP    SETUP3

                                        ;;              L3T - line #3 table
                                        ;
                                        ;               *L3T* contains the jump table for line #3 in setup mode.

        158C                    L3T     EQU     $

158C    31                              DB      '1'                     ; Option #1
158D    1550                            DW      STP3B                   ; Change baud rate

158F    32                              DB      '2'                     ; Option #2
1590    1567                            DW      STP3C                   ; Change parity

1592    33                              DB      '3'                     ; Option #3
1593    1584                            DW      STP3D                   ; Toggle duplex

1595    34                              DB      '4'                     ; Option #4
1596    1588                            DW      STP3E                   ; Toggle handshake

        0004                    L3TL    EQU     ($-L3T)/3               ; Table length


                                        ;;              L3 - line #3
                                        ;
                                        ;               *L3* displays setup information for line #3.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    all

1598    E4                      L3:     CLR     A                       ; Position

1599    71      FE                      ACALL   XSTX
159B    2A      4D      45              DB      '*MENU B*   1. BAUD', (' '+80H)
159E    4E      55      20
15A1    42      2A      20
15A4    20      20      31
15A7    2E      20      42
15AA    41      55      44
15AD    A0

15AE    E5      33              MOV     A, BAUDRATE
15B0    75      F0      0E              MOV     B, #L3ATL
```

```
    1 15B3  90      161E                    MOV     DPTR, #L3AT
    2 15B6  91      04                      ACALL   XSTABX                 ; Search for correct message
    3 15B8  50      06                      JNC     L3A                    ; IF not found THEN
    4 15BA  75      33      00              MOV     BAUDRATE, #0           ;   clear baud rate
    5 15BD  90      167F                    MOV     DPTR, #LN3B14          ;   error message
    6 15C0  74      13              L3A:    MOV     A, #19                 ; Position
    7 15C2  91      01                      ACALL   XSTATX                 ; Baud rate message
    8
    9 15C4  71      FE                      ACALL   XSTX
   10 15C6  20      20      32              DB      ' 2. PARITY', (' '+80H)
      15C9  2E      20      50
      15CC  41      52      49
      15CF  54      59      A0
   11
   12 15D2  90      1684                    MOV     DPTR, #LN3C1           ; No parity
   13 15D5  30      28      0F              JNB     PRTYENA, L3B
   14 15D8  90      1689                    MOV     DPTR, #LN3C2           ; Stick parity
   15 15DB  20      29      09              JB      PRTYSTK, L3B
   16 15DE  90      168E                    MOV     DPTR, #LN3C3           ; Even parity
   17 15E1  20      2A      03              JB      PRTYEVN, L3B
   18 15E4  90      1693                    MOV     DPTR, #LN3C4           ; Odd parity
   19 15E7  91      01              L3B:    ACALL   XSTATX                 ; Parity message
   20
   21 15E9  71      FE                      ACALL   XSTX
   22 15EB  20      20      33              DB      ' 3. DUPLEX', (' '+80H)
      15EE  2E      20      44
      15F1  55      50      4C
      15F4  45      58      A0
   23
   24 15F7  90      1698                    MOV     DPTR, #LN3E1           ; Full duplex
   25 15FA  20      2F      03              JB      FULLDPLX, L3C
   26 15FD  90      169C                    MOV     DPTR, #LN3E2           ; Half duplex
   27 1600  91      01              L3C:    ACALL   XSTATX                 ; Duplex message
   28
   29 1602  71      FE                      ACALL   XSTX
   30 1604  20      20      34              DB      ' 4. HANDSHAKE', (' '+80H)
      1607  2E      20      48
      160A  41      4E      44
      160D  53      48      41
      1610  4B      45      A0
   31
   32 1613  90      16A0                    MOV     DPTR, #LN3G1           ; Hardware handshake
   33 1616  20      3E      03              JB      HNDSHK, L3D
   34 1619  90      16A8                    MOV     DPTR, #LN3G2           ; Software handshake
   35 161C  81      01              L3D:    AJMP    XSTATX                 ; Handshake message and return
   36
   37
   38
   39
   40                               ;;     L3AT - line #3 table
   41                               ;
   42                               ;      *L3AT* contains the addresses for the corresponding baud
   43                               ;      rates and the order is used to determine the next baud rate
   44                               ;      when changed.
   45
   46          161E                 L3AT   EQU     $
   47
```

```
                1 161E  FF                              DB      255             ; 19200 baud
                2 161F  1648                            DW      LN3B
                3
                4 1621  FE                              DB      254             ; 9600 baud
                5 1622  164D                            DW      LN3B1
                6
                7 1624  FC                              DB      252             ; 4800 baud
                8 1625  1652                            DW      LN3B3
                9
               10 1627  FF                              DB      255             ; skip 7200 baud
               11 1628  167F                            DW      LN3B14
               12
               13 162A  F8                              DB      248             ; 2400 baud
               14 162B  1657                            DW      LN3B5
               15
               16 162D  FF                              DB      255             ; skip 3600 baud
               17 162E  167F                            DW      LN3B14
               18
               19 1630  F5                              DB      245             ; 1800 baud
               20 1631  165C                            DW      LN3B7
               21
               22 1633  FF                              DB      255             ; skip 2000 baud
               23 1634  167F                            DW      LN3B14
               24
               25 1636  F0                              DB      240             ; 1200 baud
               26 1637  1661                            DW      LN3B8
               27
               28 1639  E0                              DB      224             ; 600 baud
               29 163A  1666                            DW      LN3B9
               30
               31 163C  C0                              DB      192             ; 300 baud
               32 163D  166B                            DW      LN3B10
               33
               34 163F  80                              DB      128             ; 150 baud
               35 1640  1670                            DW      LN3B11
               36
               37 1642  52                              DB      82              ; 110 baud
               38 1643  1675                            DW      LN3B12
               39
               40 1645  01                              DB      1               ; 75 baud
               41 1646  167A                            DW      LN3B13
               42
               43       000E            L3ATL           EQU     ($-L3AT)/3      ; Table length
               44
               45
               46
               47
               48                       ;;              Line #3 setup mode messages
               49                       ;
               50
               51 1648  31  39  32      LN3B:           DB      '1920', ('0'+80H)
                  164B  30  B0
               52 164D  39  36  30      LN3B1:          DB      '9600', (' '+80H)
                  1650  30  A0
               53 1652  34  38  30      LN3B3:          DB      '4800', (' '+80H)
                  1655  30  A0
               54 1657  32  34  30      LN3B5:          DB      '2400', (' '+80H)
```

```
165A    30      A0
```

```
     1 165C  31      38      30      LN3B7:          DB      '1800', ('0'+80H)
       165F  30      A0
     2 1661  31      32      30      LN3B8:          DB      '1200', ('0'+80H)
       1664  30      A0
     3 1666  36      30      30      LN3B9:          DB      '600 ', ('0'+80H)
       1669  20      A0
     4 166B  33      30      30      LN3B10:         DB      '300 ', ('0'+80H)
       166E  20      A0
     5 1670  31      35      30      LN3B11:         DB      '150 ', ('0'+80H)
       1673  20      A0
     6 1675  31      31      30      LN3B12:         DB      '110 ', ('0'+80H)
       1678  20      A0
     7 167A  37      35      20      LN3B13:         DB      '75  ', ('0'+80H)
       167D  20      A0
     8 167F  65      72      72      LN3B14:         DB      'erro', ('r'+80H)
       1682  6F      F2
     9 1684  73      70      61      LN3C1:          DB      'spac', ('e'+80H)
       1687  63      E5
    10 1689  6D      61      72      LN3C2:          DB      'mark', ('0'+80H)
       168C  6B      A0
    11 168E  65      76      65      LN3C3:          DB      'even', ('0'+80H)
       1691  6E      A0
    12 1693  6F      64      64      LN3C4:          DB      'odd ', ('0'+80H)
       1696  20      A0
    13 1698  66      75      6C      LN3E1:          DB      'ful', ('l'+80H)
       169B  EC
    14 169C  68      61      6C      LN3E2:          DB      'hal', ('f'+80H)
       169F  E6
    15 16A0  68      61      72      LN3G1:          DB      'hardwar', ('e'+80H)
       16A3  64      77      61
       16A6  72      E5
    16 16A8  73      6F      66      LN3G2:          DB      'softwar', ('e'+80H)
       16AB  74      77      61
       16AE  72      E5
```

```
                                ;;              SETUP4 - setup mode line #4
                                ;
                                ;               *SETUP4* runs the terminal setup mode to change port, mode
                                ;               hold screen mode, and monitor mode.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

16B0  D1  F8            SETUP4: ACALL   L4              ; Display fourth line
16B2  91  07            STP4A:  CALL    SFCKB           ; Fetch character from keyboard
16B4  75  F0  04                MOV     B, #L4TL
16B7  90  16EC                  MOV     DPTR, #L4T
16BA  71  E9                    CALL    LOOKUP          ; Look up command
16BC  80  F4                    SJMP    STP4A           ; Keep on going

                                ;               Toggle main port
                                ;
16BE  12  1E0F          STP4B:  CALL    CMPORT          ; Toggle port
16C1  80  ED                    SJMP    SETUP4

                                ;               Toggle ANSI/ZDS/ADM 3/Hazeltine mode
                                ;
16C3  E5  28            STP4C:  MOV     A, MODE         ; Get current mode
16C5  FD                        MOV     TEMP, A         ; Save it
16C6  53  05  F0                ANL     TEMPA, #11110000B ; Mask off mode portion
16C9  23                        RL      A               ; Rotate to next mode
16CA  54  0E                    ANL     A, #00001110B   ; Mask off extra stuff
16CC  70  01                    JNZ     STP4C1          ; IF zero THEN
16CE  04                        INC     A               ;   bump to one
16CF  4D                STP4C1: ORL     A, TEMP         ; Combine both halfs
16D0  F5  28                    MOV     MODE, A         ; Save new current mode
16D2  31  CB                    ACALL   TERMINIT        ; Initialize for any terminal
16D4  30  43  02                JNB     HAZ, STP4C2     ; IF Hazeltine THEN
16D7  31  D4                    ACALL   HAZINIT         ;   initialize for Hazeltine
16D9  80  D5            STP4C2: SJMP    SETUP4

                                ;               Toggle hold screen mode
                                ;
16DB  30  36  05        STP4D:  JNB     HSMODEF, STP4D1
16DE  12  0B43                  CALL    XHSM            ; IF hold screen THEN exit
16E1  80  CD                    SJMP    SETUP4
16E3  12  0B3C          STP4D1: CALL    EHSM            ; ELSE enter hold screen
16E6  80  C8                    SJMP    SETUP4

                                ;               Toggle monitor mode
                                ;
16E8  B2  3A            STP4E:  CPL     MONITOR         ; Toggle monitor mode
16EA  80  C4                    SJMP    SETUP4
```

```
                                ;;              L4T - Line #4 table
                                ;
                                ;               *L4T* contains the jump table for line #4 in setup mode.

                16EC            L4T             EQU     $

16EC  31                                        DB      '1'
16ED  16BE                                      DW      STP4B                   ; Toggle ports

16EF  32                                        DB      '2'
16F0  16C3                                      DW      STP4C                   ; Toggle mode

16F2  33                                        DB      '3'
16F3  16DB                                      DW      STP4D                   ; Toggle hold screen

16F5  34                                        DB      '4'
16F6  16E8                                      DW      STP4E                   ; Toggle monitor mode

                0004            L4TL            EQU     ($-L4T)/3               ; Table length



                                ;;              L4 - line #4
                                ;
                                ;               *L4* displays setup information for line #4.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all


16F8  E4                        L4:             CLR     A
16F9  71        FE                              ACALL   XSTX
16FB  2A        4D      45                      DB      '*MENU C*   1. PORT', (' '+80H)
16FE  4E        55      20
1701  43        2A      20
1704  20        20      31
1707  2E        20      50
170A  4F        52      54
170D  A0

170E  90        1769                            MOV     DPTR, #LN4B1            ; Normal
1711  30        02      03                      JNB     PORTF, L4A
1714  90        176F                            MOV     DPTR, #LN4B2            ; Auxiliary
1717  91        01              L4A:            ACALL   XSTATX                  ; Port message

1719  71        FE                              ACALL   XSTX
171B  20        20      20                      DB      '   2. MODE', (' '+80H)
171E  32        2E      20
```

```
1721    4D      4F      44
1724    45      A0
```

```
  1726   90    1775                    MOV     DPTR, #LN4D1          ; ANSI mode
  1729   20    40    0F                JB      ANSI, L4B
  172C   90    177D                    MOV     DPTR, #LN4D2          ; ZDS mode
  172F   20    41    09                JB      ZDS, L4B
  1732   90    1785                    MOV     DPTR, #LN4D3          ; ADM 3 mode
  1735   20    42    03                JB      ADM3, L4B
  1738   90    178D                    MOV     DPTR, #LN4D4          ; Hazeltine mode
  173B   91    01           L4B:       ACALL   XSTATX                ; ANSI/ZDS/ADM 3/Haz message

  173D   71    FE                      ACALL   XSTX
  173F   20    20    20                DB      '   3. HOLD SCRN', (' '+80H)
  1742   33    2E    20
  1745   48    4F    4C
  1748   44    20    53
  174B   43    52    4E
  174E   A0

  174F   A2    36                      MOV     C, HSMODEF
  1751   12    1BBF                    LCALL   XLONOFF               ; Hold on/off message

  1754   71    FE                      ACALL   XSTX
  1756   20    20    20                DB      '   4. MONITOR', (' '+80H)
  1759   34    2E    20
  175C   4D    4F    4E
  175F   49    54    4F
  1762   52    A0

  1764   A2    3A                      MOV     C, MONITOR
  1766   02    1BBF                    LJMP    XLONOFF               ; Monitor mode on/off and return




                            ;;         Line #4 setup mode messages
                            ;

  1769   6E    6F    72     LN4B1:     DB      'norma', ('l'+80H)
  176C   6D    61    EC
  176F   61    75    78     LN4B2:     DB      'auxil', (' '+80H)
  1772   69    6C    A0
  1775   61    6E    73     LN4D1:     DB      'ansi   ', (' '+80H)
  1778   69    20    20
  177B   20    A0
  177D   7A    65    6E     LN4D2:     DB      'zenith ', (' '+80H)
  1780   69    74    68
  1783   20    A0
  1785   61    64    6D     LN4D3:     DB      'adm 3a ', (' '+80H)
  1788   20    33    61
  178B   20    A0
  178D   68    61    7A     LN4D4:     DB      'haz 150', ('0'+80H)
  1790   20    31    35
  1793   30    B0
```

```
                                ;;              SETUP5 - setup mode line #5
                                ;
                                ;               *SETUP5* runs the terminal setup mode to change set clock,
                                ;               toggle status line, setup wrap around, and screen saver.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

1795  12    1808        SETUP5: CALL    L5                      ; Display fifth line
1798  91    07          STP5A:  CALL    SFCKB                   ; Fetch character from keyboard
179A  75    F0    04            MOV     B, #L5TL
179D  90    17FC                MOV     DPTR, #L5T
17A0  71    E9                  CALL    LOOKUP                  ; Look up command
17A2  80    F4                  SJMP    STP5A                   ; Keep on going

                                ;               Set clock
                                ;
17A4  12    1B2C        STP5B:  CALL    SCLRLR                  ; Setup clear line reverse
17A7  E4                        CLR     A

17A8  71    FE                  ACALL   XSTX
17AA  45    6E    74            DB      'Enter hours ', ('>'+80H)
17AD  65    72    20
17B0  68    6F    75
17B3  72    73    20
17B6  BE

17B7  74    0E                  MOV     A, #14
17B9  12    1B45                CALL    GETNUM                  ; Get number
17BC  BF    00    02            CJNE    WORK2, #0, STP5B1       ; IF zero THEN skip
17BF  80    D4                  SJMP    SETUP5
17C1  B4    18    00    STP5B1: CJNE    A, #24, STP5B2
17C4  50    CF          STP5B2: JNC     SETUP5                  ; IF number <= 24 THEN
17C6  F5    3D                  MOV     THOUR, A
17C8  12    1B2C                CALL    SCLRLR                  ;   Setup clear line reverse
17CB  E4                        CLR     A

17CC  71    FE                  ACALL   XSTX
17CE  45    6E    74            DB      'Enter minutes ', ('>'+80H)
17D1  65    72    20
17D4  6D    69    6E
17D7  75    74    65
17DA  73    20    BE

17DD  74    10                  MOV     A, #16
17DF  12    1B45                CALL    GETNUM                  ;   get number
17E2  B4    3C    00            CJNE    A, #60, STP5B3
17E5  50    AE          STP5B3: JNC     SETUP5                  ;   IF number <= 59 THEN
17E7  F5    3C                  MOV     TMIN, A

17E9  E4                        CLR     A
```

*** Z-29 COMPUTER TERMINAL FIRMWARE VERSION 1.03 ***

```
                17EA    F5      3B                      MOV     TSEC, A         ;   clear seconds
                17EC    F5      3A                      MOV     TCNT, A         ;   and count
                17EE    80      A5                      SJMP    SETUP5
                
                
                                                ;       Toggle 25th status line
                                                ;
                17F0    B2      37      STP5C:          CPL     L25ON           ; Toggle status line
                17F2    80      A1                      SJMP    SETUP5
                
                
                                                ;       Toggle wrap
                                                ;
                17F4    B2      2B      STP5D:          CPL     AUTOWRAP        ; Toggle auto wrap
                17F6    80      9D                      SJMP    SETUP5
                
                
                                                ;       Toggle screen saver mode
                                                ;
                17F8    B2      3B      STP5E:          CPL     SCRNSAVE        ; Toggle screen saver
                17FA    80      99                      SJMP    SETUP5
                
                
                
                                                ;;      L5T - line #5 table
                                                ;
                                                ;       *L5T* contains the jump table for line #5 in setup mode.
                
                        17FC                    L5T     EQU     $
                
                17FC    31                              DB      '1'
                17FD    17A4                            DW      STP5B           ; Set clock
                
                17FF    32                              DB      '2'
                1800    17F0                            DW      STP5C           ; Toggle status line
                
                1802    33                              DB      '3'
                1803    17F4                            DW      STP5D           ; Toggle wrap around
                
                1805    34                              DB      '4'
                1806    17F8                            DW      STP5E           ; Toggle screen saver
                
                        0004                    L5TL    EQU     ($-L5T)/3       ; Table length
                
                
                
                                                ;;      L5 - line #5
                                                ;
                                                ;       *L5* displays setup information for line #5.
                                                ;
                                                ;
                                                ;       ENTRY   none
                                                ;
                                                ;       EXIT    none
                                                ;
```

```
                                        ;               USES    all
                                        ;
1808    E4                      LS:     CLR     A

1809    71      9D                      ACALL   XST
180B    2A      4D      45              DB      '*MENU D*   1. SET CLOCK   2. STATUS LINE', ('' '+80H)
180E    4E      55      20
1811    44      2A      20
1814    20      20      31
1817    2E      20      53
181A    45      54      20
181D    43      4C      4F
1820    43      4B      20
1823    20      20      32
1826    2E      20      53
1829    54      41      54
182C    55      53      20
182F    4C      49      4E
1832    45      A0

1834    A2      37                      MOV     C, L25ON
1836    12      1BBF                    LCALL   XLONOFF                 ; Status line on/off

1839    71      9D                      ACALL   XST
183B    20      20      20              DB      '    3. WRAP', ('' '+80H)
183E    33      2E      20
1841    57      52      41
1844    50      A0

1846    A2      2B                      MOV     C, AUTOWRAP
1848    12      1BBF                    LCALL   XLONOFF                 ; Wrap around on/off

184B    71      9D                      ACALL   XST
184D    20      20      20              DB      '   4. SCREEN SAVER', ('' '+80H)
1850    34      2E      20
1853    53      43      52
1856    45      45      4E
1859    20      53      41
185C    56      45      52
185F    A0

1860    A2      3B                      MOV     C, SCRNSAVE
1862    02      1BBF                    LJMP    XLONOFF                 ; Screen saver on/off and return
```

```
                                ;;              SETUP6 - setup mode line #6
                                ;
                                ;               *SETUP6* runs the terminal setup mode to change keypad shift,
                                ;               keypad alternate, auto repeat, and key click.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

1865  11    97          SETUP6: ACALL   L6              ; Display sixth line
1867  12    1407        STP6A:  CALL    SFCKB           ; Fetch character from keyboard
186A  75    F0    04            MOV     B, #L6TL
186D  90    188B                MOV     DPTR, #L6T
1870  12    13E9                CALL    LOOKUP          ; Look up command
1873  80    F2                  SJMP    STP6A           ; Keep on going

                                ;               Toggle keypad shifted
                                ;
1875  B2    30          STP6B:  CPL     KPADSHTF        ; Toggle keypad shifted
1877  80    EC                  SJMP    SETUP6

                                ;               Toggle keypad alternate
                                ;
1879  B2    31          STP6C:  CPL     KPADALTF        ; Toggle keypad alternate
187B  80    E8                  SJMP    SETUP6

                                ;               Toggle auto repeat
                                ;
187D  B2    34          STP6D:  CPL     REPF            ; Toggle auto repeat
187F  12    119E                CALL    INITKB          ; Reconfisure keyboard
1882  80    E1                  SJMP    SETUP6

                                ;               Toggle key click
                                ;
1884  B2    33          STP6E:  CPL     CLKF            ; Toggle key click
1886  12    119E                CALL    INITKB          ; Reconfisur keyboard
1889  80    DA                  SJMP    SETUP6


                                ;;              L6T - line #6 table
                                ;
                                ;               *L6T* contains the jump table for line #6 in setup mode.

        188B                L6T     EQU     $
188B  31                            DB      '1'
188C  1875                          DW      STP6B           ; Toggle keypad shifted
```

```
    1
    2 188E    32                                DB      '2'
    3 188F    1879                              DW      STP6C                   ; Toggle keypad alternate
    4
    5 1891    33                                DB      '3'
    6 1892    187D                              DW      STP6D                   ; Toggle auto repeat
    7
    8 1894    34                                DB      '4'
    9 1895    1884                              DW      STP6E                   ; Toggle key click
   10
   11         0004                    L6TL      EQU     ($-L6T)/3               ; Table length
   12
   13
   14
   15
   16                                 ;;        L6 - line #6
   17                                 ;
   18                                 ;         *L6* displays setup information for line #6
   19                                 ;
   20                                 ;
   21                                 ;         ENTRY   none
   22                                 ;
   23                                 ;         EXIT    none
   24                                 ;
   25                                 ;         USES    all
   26
   27
   28 1897    E4                      L6:       CLR     A                       ; Position
   29
   30 1898    71      9D                        ACALL   XST
   31 189A    2A      4D      45                DB      '*MENU E*   1. KEYPAD SHFT', (' '+80H)
      189D    4E      55      20
      18A0    45      2A      20
      18A3    20      20      31
      18A6    2E      20      4B
      18A9    45      59      50
      18AC    41      44      20
      18AF    53      48      46
      18B2    54      A0
   32
   33 18B4    A2      30                        MOV     C, KPADSHTF
   34 18B6    71      BF                        ACALL   XLONOFF                 ; Keypad shifted on/off
   35
   36 18B8    71      9D                        ACALL   XST
   37 18BA    20      20      20                DB      '   2. KEYPAD ALT', (' '+80H)
      18BD    32      2E      20
      18C0    4B      45      59
      18C3    50      41      44
      18C6    20      41      4C
      18C9    54      A0
   38
   39 18CB    A2      31                        MOV     C, KPADALTF
   40 18CD    71      BF                        ACALL   XLONOFF                 ; Keypad alternate on/off
   41
   42 18CF    71      9D                        ACALL   XST
   43 18D1    20      20      20                DB      '   3. REPEAT', (' '+80H)
      18D4    33      2E      20
```

```
*** Z-29 COMPUTER TERMINAL FIRMWARE VERSION 1.03 ***

        18D7    52      45      50
        18DA    45      41      54
        18DD    A0
```

```
1
2 18DE    A2      34                          MOV     C, REPF
3 18E0    71      BF                          ACALL   XLONOFF                  ; Auto repeat on/off
4
5 18E2    71      9D                          ACALL   XST
6 18E4    20      20      20                  DB      '    4. CLICK', ('4'+80H)
  18E7    34      2E      20
  18EA    43      4C      49
  18ED    43      4B      A0
7
8 18F0    A2      33                          MOV     C, CLKF
9 18F2    61      BF                          AJMP    XLONOFF                  ; Key click on/off and return
```

```
                                ;;              SETUP7 - setup mode line #7
                                ;
                                ;               *SETUP7* runs the terminal setup mode to change auto carriage
                                ;               return, auto line feed, cursor, and frequency.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

18F4  31      3F                SETUP7:         ACALL   L7                      ; Display seventh line
18F6  12      1407              STP7A:          CALL    SFCKB                   ; Fetch character from keyboard
18F9  75      F0      04                      MOV     B, #L7TL
18FC  90      1933                            MOV     DPTR, #L7T
18FF  12      13E9                            CALL    LOOKUP                  ; Look up command
1902  80      F2                              SJMP    STP7A                   ; Keep on going

                                ;               Toggle auto carriage return
                                ;
1904  B2      2C                STP7B:          CPL     AUTOCR                  ; Toggle auto CR
1906  80      EC                              SJMP    SETUP7

                                ;               Toggle auto line feed
                                ;
1908  B2      2D                STP7C:          CPL     AUTOLF                  ; Toggle auto LF
190A  80      E8                              SJMP    SETUP7

                                ;               Toggle cursor
                                ;
190C  20      38      0D        STP7D:          JB      CRNBK, STP7D2           ; IF blinking AND
190F  20      32      04                      JB      CRUL, STP7D1            ; IF block THEN
1912  D2      32                              SETB    CRUL                    ;   set to blink line
1914  80      11                              SJMP    STP7D4

1916  C2      32                STP7D1:         CLR     CRUL                    ; IF blink line THEN
1918  D2      38                              SETB    CRNBK                   ;   set to block
191A  80      0B                              SJMP    STP7D4

191C  20      32      04        STP7D2:         JB      CRUL, STP7D3            ; IF block THEN
191F  D2      32                              SETB    CRUL                    ;   set to underline
1921  80      04                              SJMP    STP7D4

1923  C2      32                STP7D3:         CLR     CRUL                    ; IF underline THEN
1925  C2      38                              CLR     CRNBK                   ;   set to blink block

1927  12      117B              STP7D4:         CALL    SETCF                   ; Set cursor format
192A  80      C8                              SJMP    SETUP7

                                ;               Toggle line frequency
                                ;
```

```
192C   B2    39              STP7E:    CPL     FREQ                    ; Toggle frequency
192E   12    117B                      CALL    SETCF                   ; Set cursor format
1931   80    C1                        SJMP    SETUP7

                             ;;        L7T - line #7 table
                             ;
                             ;         *L7T* contains the jump table for line #7 in setup mode.

       1933                  L7T       EQU     $

1933   31                              DB      '1'
1934   1904                            DW      STP7B                   ; Toggle auto carriage return

1936   32                              DB      '2'
1937   1908                            DW      STP7C                   ; Toggle auto line feed

1939   33                              DB      '3'
193A   190C                            DW      STP7D                   ; Toggle cursor

193C   34                              DB      '4'
193D   192C                            DW      STP7E                   ; Toggle line freq 50/60 Hz

       0004                  L7TL      EQU     ($-L7T)/3               ; Table length

                             ;;        L7 - line #7
                             ;
                             ;         *L7* displays setup information for line #7
                             ;
                             ;
                             ;         ENTRY   none
                             ;
                             ;         EXIT    none
                             ;
                             ;         USES    all

193F   E4                    L7:       CLR     A                       ; Position

1940   71    9D                        ACALL   XST
1942   2A    4D    45                  DB      '*MENU F*   1. AUTO CR', ('  '+80H)
1945   4E    55    20
1948   46    2A    20
194B   20    20    31
194E   2E    20    41
1951   55    54    4F
1954   20    43    52
1957   A0

1958   A2    2C                        MOV     C, AUTOCR
195A   71    BF                        ACALL   XLONOFF                 ; Auto CR on/off
```

```
    1 195C  71  9D                          ACALL   XST
    2 195E  20  20  20                      DB      '  2. AUTO LF', ('F'+80H)
      1961  32  2E  20
      1964  41  55  54
      1967  4F  20  4C
      196A  46  A0
    3
    4 196C  A2  2D                          MOV     C, AUTOLF
    5 196E  71  BF                          ACALL   XLONOFF                 ; Auto LF on/off
    6
    7 1970  71  9D                          ACALL   XST
    8 1972  20  20  20                      DB      '  3. CURSOR', ('R'+80H)
      1975  33  2E  20
      1978  43  55  52
      197B  53  4F  52
      197E  A0
    9
   10 197F  20  38  0B                      JB      CRNBK, L7C1
   11 1982  90  19B7                        MOV     DPTR, #LN7D1            ; Blink block
   12 1985  30  32  0E                      JNB     CRUL, L7C2
   13 1988  90  19C0                        MOV     DPTR, #LN7D2            ; Blink underline
   14 198B  80  09                          SJMP    L7C2
   15 198D  90  19C9            L7C1:       MOV     DPTR, #LN7D3            ; Block steady
   16 1990  30  32  03                      JNB     CRUL, L7C2
   17 1993  90  19D2                        MOV     DPTR, #LN7D4            ; Underline steady
   18 1996  71  A8              L7C2:       ACALL   XSTAT                   ; Cursor message
   19
   20 1998  71  9D                          ACALL   XST
   21 199A  20  20  20                      DB      '  4. FREQ', ('Q'+80H)
      199D  34  2E  20
      19A0  46  52  45
      19A3  51  A0
   22
   23 19A5  90  19DB                        MOV     DPTR, #LN7F1            ; 50 Hz
   24 19A8  30  39  03                      JNB     FREQ, L7D
   25 19AB  90  19DC                        MOV     DPTR, #LN7F2            ; 60 Hz
   26 19AE  71  A8              L7D:        ACALL   XSTAT
   27
   28 19B0  71  9D                          ACALL   XST
   29 19B2  30  20  48                      DB      '0 H', ('z'+80H)
      19B5  FA
   30
   31 19B6  22                              RET
   32
   33
   34
   35
   36                           ;;          Line #7 setup mode messages
   37                           ;
   38
   39 19B7  62  6C  6E          LN7D1:      DB      'blnk blc', ('k'+80H)
      19BA  6B  20  62
      19BD  6C  63  EB
   40 19C0  62  6C  6E          LN7D2:      DB      'blnk lin', ('e'+80H)
      19C3  6B  20  6C
      19C6  69  6E  E5
   41 19C9  62  6C  6F          LN7D3:      DB      'block   ', (' '+80H)
```

```
19CC    63      6B      20
19CF    20      20      A0
```

```
      1 19D2    75      6E      64      LN7D4:          DB      'underlin', ('e'+80H)
        19D5    65      72      6C
        19D8    69      6E      E5
      2 19DB    B5                      LN7F1:          DB      ('5'+80H)
      3 19DC    B6                      LN7F2:          DB      ('6'+80H)
```

```
                                ;;              SETUP8 - setup mode line #8
                                ;
                                ;               *SETUP8* runs the terminal setup mode for simple terminal
                                ;               diagnostics.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all


19DD  51      8C                SETUP8:         ACALL   L8                      ; Display eighth line
19DF  12      1407              STP8A:          CALL    SFCKB                   ; Fetch character from keyboard
19E2  75      F0      04                        MOV     B, #L8TL
19E5  90      1A74                              MOV     DPTR, #L8T
19E8  12      13E9                              CALL    LOOKUP                  ; Look up command
19EB  80      F2                                SJMP    STP8A                   ; Keep on going


                                ;               Fill screen
                                ;
19ED  71      2C                STP8B:          ACALL   SCLRLR                  ; Setup clear line reverse
19EF  E4                                        CLR     A                       ; Position

19F0  71      9D                                ACALL   XST
19F2  50      72      65                        DB      'Press RETURN to exi', ('t'+80H)
19F5  73      73      20
19F8  52      45      54
19FB  55      52      4E
19FE  20      74      6F
1A01  20      65      78
1A04  69      F4


1A06  74      45                                MOV     A, #'E'                 ; Display 'E' first time
1A08  90      5000              STP8B1:         MOV     DPTR, #CHARMEM          ; Point to character memory
1A0B  71      21                                ACALL   SFILL                   ; Fill screen with char

1A0D  12      1407                              CALL    SFCKB                   ; Fetch character from keyboard
1A10  75      F0      04                        MOV     B, #L8TL2
1A13  90      1A80                              MOV     DPTR, #L8T2
1A16  12      013A                              CALL    STAB                    ; Search the table
1A19  40      ED                                JC      STP8B1                  ; IF not found THEN display it
1A1B  E4                                        CLR     A
1A1C  73                                        JMP     @A+DPTR                 ; Jump to routine

1A1D  74      20                STP8B3:         MOV     A, #' '                 ; Change funny space to real
1A1F  80      E7                                SJMP    STP8B1                  ; Fill screen


                                ;               Change attribute
                                ;
1A21  E5      21                STP8C:          MOV     A, ATTRIBUTES           ; Get current attributes
1A23  04                                        INC     A                       ; Bump to next attribute
1A24  54      0F                                ANL     A, #00001111B           ; Mask just attributes
1A26  53      21      F0                        ANL     ATTRIBUTES, #11110000B  ; Get attributes ready
```

*** Z-29 COMPUTER TERMINAL FIRMWARE VERSION 1.03 ***

```
                1A29    42      21                      ORL     ATTRIBUTES, A           ; Move new attributes in
                2 1A2B  90      4000                    MOV     DPTR, #ATRIBMEM         ; Point to attribute memory
                3 1A2E  71      21                      ACALL   SFILL                   ; Fill memory with attribute
                4 1A30  80      AB                      SJMP    SETUP8
                5
                6
                7                               ;               Toggle character set
                8                               ;
                9 1A32  B2      3D              STP8D:  CPL     ACHR2                   ; Toggle character set
               10 1A34  A2      3D                      MOV     C, ACHR2
               11 1A36  92      0C                      MOV     ALTCHARA, C             ; Place into attribute
               12 1A38  80      A3                      SJMP    SETUP8
               13
               14
               15                               ;               Continuous test
               16                               ;
               17 1A3A  75      40      00      STP8E:  MOV     ERRORS, #0              ; REPEAT start with no errors
               18 1A3D  D1      D9                      ACALL   ROMCHK                  ;   check ROM
               19 1A3F  D1      F1                      ACALL   RAMCHK                  ;   check RAM
               20 1A41  12      0BAC            STP8E0: CALL    CLRS                    ;   clear screen
               21 1A44  71      2C                      ACALL   SCLRLR                  ;   setup clear line reverse
               22 1A46  E4                              CLR     A                       ;   position
               23
               24 1A47  71      9D                      ACALL   XST
               25 1A49  43      6F      6E              DB      'Continuous tes', ('t'+80H)
                   1A4C  74      69      6E
                   1A4F  75      6F      75
                   1A52  73      20      74
                   1A55  65      73      F4
               26
               27 1A58  12      02B6                    CALL    FCKB                    ; Get any char from keyboard
               28 1A5B  50      80                      JNC     SETUP8                  ; IF any char THEN exit
               29 1A5D  12      066B                    CALL    RING                    ; Ring bell
               30
               31 1A60  E4                              CLR     A
               32 1A61  90      5000            STP8E1: MOV     DPTR, #CHARMEM          ;   REPEAT point to char memory
               33 1A64  71      21                      ACALL   SFILL                   ;     fill with char
               34 1A66  12      02E8                    CALL    LKB                     ;     look at keyboard
               35 1A69  50      D6                      JNC     STP8E0                  ;     IF any char THEN exit
               36 1A6B  04                              INC     A                       ;     bump to next char
               37 1A6C  70      F3                      JNZ     STP8E1                  ;   UNTIL all combinations
               38 1A6E  E5      40                      MOV     A, ERRORS
               39 1A70  60      C8                      JZ      STP8E                   ; UNTIL errors
               40 1A72  E1      2C                      AJMP    ERROR                   ; Jump to error routine
               41
               42
               43
               44
               45                               ;;              L8T - line #8 table
               46                               ;
               47                               ;               *L8T* contains the jump table for line #8 in setup mode.
               48
               49              1A74            L8T             EQU     $
               50
               51 1A74  31                              DB      '1'
               52 1A75  1A32                            DW      STP8D                   ; Toggle character set
               53
```

```
                                                DB      '2'
1A78    19ED                                    DW      STP8B                   ; Fill screen

1A7A    33                                      DB      '3'
1A7B    1A21                                    DW      STP8C                   ; Change attributes

1A7D    34                                      DB      '4'
1A7E    1A3A                                    DW      STP8E                   ; Continuous test
        0004            L8TL                    EQU     ($-L8T)/3               ; Table length



                        ;;                      L8T2 - line #8 table 2
                        ;
                        ;                       *L8T2* contains the jump table for line #8 in fill screen
                        ;                       option.

        1A80            L8T2                    EQU     $

1A80    8C                                      DB      KB_SPACE
1A81    1A1D                                    DW      STP8B3                  ; Access code (ignore)

1A83    0D                                      DB      CR
1A84    19DD                                    DW      SETUP8                  ; Carriage return

1A86    88                                      DB      KB_SETUP
1A87    1AF6                                    DW      SETUPRET                ; Exit setup mode

1A89    C8                                      DB      KB_SETUP + KB_SHFT
1A8A    1B02                                    DW      SETUPSAVE               ; Exit setup mode and save

        0004            L8TL2                   EQU     ($-L8T2)/3              ; Table length



                        ;;                      L8 - line #8
                        ;
                        ;                       *L8* displays setup information for line #8.
                        ;
                        ;                       ENTRY   none
                        ;
                        ;                       EXIT    none
                        ;
                        ;                       USES    all

1A8C    E4              L8:                     CLR     A                       ; Position

1A8D    71      9D                              ACALL   XST
1A8F    2A      4D      45                      DB      '*MENU G*    1. CHAR SET', (' '+80H)
        1A92    4E      55      20
        1A95    47      2A      20
        1A98    20      20      20
```

*** Z-29 COMPUTER TERMINAL FIRMWARE VERSION 1.03 ***

```
1A9B    31      2E      20
1A9E    43      48      41
1AA1    52      20      53
1AA4    45      54      A0
```

```
                1
                2 1AA7    90      1AE4                      MOV     DPTR, #LN8B1            ; Primary
                3 1AAA    30      3D      03                JNB     ACHR2, L8A
                4 1AAD    90      1AED                      MOV     DPTR, #LN8B2            ; Auxiliary
                5 1AB0    71      A8              L8A:      ACALL   XSTAT                   ; Character set
                6
                7 1AB2    71      9D                        ACALL   XST
                8 1AB4    20      20      20                DB      '    2. FILL SCREEN    3. ATTRIBUTES'
                  1AB7    20      32      2E
                  1ABA    20      46      49
                  1ABD    4C      4C      20
                  1AC0    53      43      52
                  1AC3    45      45      4E
                  1AC6    20      20      20
                  1AC9    20      33      2E
                  1ACC    20      41      54
                  1ACF    54      52      49
                  1AD2    42      55      54
                  1AD5    45      53
                9 1AD7    20      20      20                DB      '    4. TEST', (' '+80H)
                  1ADA    20      34      2E
                  1ADD    20      54      45
                  1AE0    53      54      A0
               10
               11 1AE3    22                                RET
               12
               13
               14
               15
               16                                ;;          Line #8 setup mode messages
               17                                ;
               18
               19 1AE4    6E      6F      72      LN8B1:    DB      'normal  ', (' '+80H)
                  1AE7    6D      61      6C
                  1AEA    20      20      A0
               20 1AED    61      6C      74      LN8B2:    DB      'alternat', ('e'+80H)
                  1AF0    65      72      6E
                  1AF3    61      74      E5
```

```
                                ;;              SETUPRET - return from setup mode
                                ;
                                ;               *SETUPRET* exits setup mode without permanently saving
                                ;               any values that have changed.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    none

1AF6  30  37  06  SETUPRET:     JNB     L25ON, SRET             ; IF 25th line is on AND
1AF9  20  15  03                JB      L25EN, SRET             ; IF not enabled for user THEN
1AFC  02  1E69                  JMP     DISPSTAT                ;   display stats and return
1AFF  E4                SRET:   CLR     A
1B00  80  2C                    SJMP    SCLRL                   ; ELSE clear line and return
```

```
                                ;;              SETUPSAVE - save setup configuration
                                ;
                                ;               *SETUPSAVE* returns all values back to the EAROM.
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

1B02  F1    22          SETUPSAVE:  ACALL   NVRCHK          ; Checksum the NVRAM
1B04  F5    34                      MOV     NVRSUM, A       ; Update checksum value

1B06  78    25                      MOV     PTR1, #25H      ; EAROM store
1B08  90    7000                    MOV     DPTR, #EAROM
1B0B  7A    10                      MOV     INDX1, #16      ; 16 bytes

1B0D  E6                    SAV2:   MOV     A, @PTR1        ; REPEAT
1B0E  C4                            SWAP    A
1B0F  F0                            MOVX    @DPTR, A        ;   store upper nibble
1B10  A3                            INC     DPTR
1B11  C4                            SWAP    A
1B12  F0                            MOVX    @DPTR, A        ;   store lower nibble
1B13  A3                            INC     DPTR
1B14  08                            INC     PTR1            ;   bump pointers
1B15  DA    F6                      DJNZ    INDX1, SAV2     ; UNTIL done

1B17  C2    00                      CLR     EAROMSAVE
1B19  D1    18                      ACALL   WLATCH          ; Save EAROM
1B1B  D2    00                      SETB    EAROMSAVE
1B1D  D1    18                      ACALL   WLATCH          ; By toggling flag

1B1F  41    F6                      AJMP    SETUPRET        ; Return from setup mode
```

*** Z-29 COMPUTER TERMINAL FIRMWARE VERSION 1.03 ***

```
                                ;;              SFILL - screen fill
                                ;
                                ;               *SFILL* fills the complete screen with the data given it
                                ;               into the memory location given it.  It does not touch the
                                ;               25th line however.
                                ;
                                ;
                                ;               ENTRY   (A) = character to write
                                ;                       (DPTR) = memory location to start at
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    DPTR, INDX1/2

1B21  7A      18        SFILL:  MOV     INDX1, #MAXLINE         ; Go through all lines
1B23  7B      50        SFIL1:  MOV     INDX2, #NUMCHARS        ; REPEAT
1B25  F0                SFIL2:  MOVX    @DPTR, A                ;   REPEAT write character
1B26  A3                        INC     DPTR                    ;     bump pointer
1B27  DB      FC                DJNZ    INDX2, SFIL2            ;   UNTIL line is done
1B29  DA      F8                DJNZ    INDX1, SFIL1            ; UNTIL screen is done
1B2B  22                        RET
```

```
                                        ;;              SCLRLR - setup clear line reverse video
                                        ;               SCLRL - setup clear line
                                        ;
                                        ;               *SCLRLR* clears 25th line with reverse video spaces.
                                        ;
                                        ;               *SCLRL* clears 25th line and places the attributes passed
                                        ;               to in on that line.
                                        ;
                                        ;
                                        ;               ENTRY   (A) = lower four bits used for attributes
                                        ;
                                        ;               EXIT    (A) = unchanged
                                        ;
                                        ;               USES    all

1B2C  74      01            SCLRLR:   MOV     A, #00000001B         ; Set for reverse video

1B2E  C0      E0            SCLRL:    PUSH    ACC                   ; Save attributes
1B30  75      F0      18              MOV     B, #MAXLINE           ; Clear line before display
1B33  12      0B96                    CALL    CLRLINE
1B36  D0      E0                      POP     ACC                   ; Restore attributes

1B38  90      5780                    MOV     DPTR, #L25MEM
1B3B  53      83      EF              ANL     DPH, #ATRANL          ; Point to attribute memory
1B3E  7A      50                      MOV     INDX1, #NUMCHARS
1B40  F0                      SLP:      MOVX    @DPTR, A              ; REPEAT write attribute
1B41  A3                                INC     DPTR                  ;   bump RAM pointer
1B42  DA      FC                        DJNZ    INDX1, SLP            ; UNTIL done
1B44  22                                RET
```

```
                                        ;;              GETNUM - set number string
                                        ;
                                        ;               *GETNUM* sets a number from 0..255 from the keyboard.  The
                                        ;               keys "RETURN" finishes the number and "BACKSPACE" or
                                        ;               "DELETE" clears the complete number entered.  The fourth
                                        ;               character received automatically makes the routine return.
                                        ;
                                        ;
                                        ;               ENTRY   (A) = position on 25th line to start
                                        ;
                                        ;               EXIT    (A) = number received (0..255)
                                        ;                       (WORK2) = number of characters picked up
                                        ;
                                        ;               USES    all


1B45    FE                      GETNUM: MOV     WORK1, A                ; Original position

1B46    7F      00              GNUM0:  MOV     WORK2, #0               ; Position count
1B48    7C      00                      MOV     WORK, #0                ; Running total

                                        ;               Get character
                                        ;
1B4A    12      1407            GNUM1:  CALL    SFCKB                   ; Get next character

                                        ;               Check for end of line
                                        ;
1B4D    B4      0D      02              CJNE    A, #CR, GNUM2           ; IF carriage return THEN
1B50    EC                      GNUMR:  MOV     A, WORK                 ;   get number and return
1B51    22                              RET

                                        ;               Check for backspace or delete
                                        ;
1B52    B4      08      02      GNUM2:  CJNE    A, #BS, GNUM2A          ; IF backspace OR
1B55    80      03                      SJMP    GNUM2B
1B57    B4      7F      0F      GNUM2A: CJNE    A, #RUBOUT, GNUM4       ; IF delete THEN

                                        ;               Clear current number
                                        ;
1B5A    EF                      GNUM2B: MOV     A, WORK2
1B5B    60      ED                      JZ      GNUM1                   ;   IF no chars THEN skip it
1B5D    1E                      GNUM2L: DEC     WORK1                   ;   REPEAT decrement position
1B5E    DF      FD                      DJNZ    WORK2, GNUM2L           ;   UNTIL count = 0

1B60    EE                              MOV     A, WORK1                ;   set position
1B61    71      9D                      ACALL   XST                     ;   wipe out old number
1B63    20      20      20              DB      ' ', (' '+80H)
1B66    A0
1B67    80      DD                      SJMP    GNUM0                   ;   start over again

                                        ;               Check number and position
                                        ;
1B69    BF      03      02      GNUM4:  CJNE    WORK2, #3, GNUM4A       ; IF all chars THEN
1B6C    80      E2                      SJMP    GNUMR                   ;   exit
1B6E    B4      3A      00      GNUM4A: CJNE    A, #'9'+1, GNUM4B
1B71    50      D7              GNUM4B: JNC     GNUM1                   ; Check upper bounds
```

```
1B73    B4      30      00                      CJNE    A, #'0', GNUM4C
1B76    40      D2              GNUM4C:         JC      GNUM1                           ; Check lower bounds

                                ;               Calculate next number and display
                                ;
1B78    94      30                              SUBB    A, #'0'                         ; Subtract ASCII offset
1B7A    C0      E0                              PUSH    ACC                             ; Save it
1B7C    CC                                      XCH     A, WORK
1B7D    75      F0      0A                      MOV     B, #10
1B80    A4                                      MUL     AB                              ; num := num * 10
1B81    2C                                      ADD     A, WORK
1B82    FC                                      MOV     WORK, A                         ; num := num + work
1B83    D0      E0                              POP     ACC                             ; Restore number
1B85    90      1B93                            MOV     DPTR, #GNUMS
1B88    25      82                              ADD     A,DPL
1B8A    F5      82                              MOV     DPL, A                          ; Bump address (caution)
1B8C    EE                                      MOV     A, WORK1                        ; Get position
1B8D    71      A8                              ACALL   XSTAT                           ; Display
1B8F    0E                                      INC     WORK1                           ; Bump position
1B90    0F                                      INC     WORK2                           ; Bump count
1B91    80      B7                              SJMP    GNUM1                           ; UNTIL done

                                ;               Must not cross any page boundaries
                                ;
1B93    B0      B1      B2      GNUMS:          DB      ('0'+80H), ('1'+80H), ('2'+80H), ('3'+80H), ('4'+80H)
1B96    B3      B4
1B98    B5      B6      B7                      DB      ('5'+80H), ('6'+80H), ('7'+80H), ('8'+80H), ('9'+80H)
1B9B    B8      B9

                                                IFC     NE,HIGH($)-HIGH(GNUMS)
                                                ERROR   ;'GNUMS' crosses a page boundary
                                                ENDC
```

```
                                ;;              XST - transmit string to 25th status line
                                ;
                                ;               *XST* places a string of displayable characters that
                                ;               follow the CALL onto the 25th status line of the screen.
                                ;
                                ;
                                ;               ENTRY   (A) = position on 25th line
                                ;                       String to be sent must immediately follow the call
                                ;                       to this routine.  Final character must have bit
                                ;                       seven set.
                                ;
                                ;               EXIT    (A) = next position on 25th line
                                ;
                                ;               USES    A, DPTR, PTR1

1B9D   D0       83      XST:    POP     DPH             ; Get return address
1B9F   D0       82              POP     DPL

1BA1   71       A8              ACALL   XSTAT           ; Output string

1BA3   C0       82              PUSH    DPL             ; Place return address
1BA5   C0       83              PUSH    DPH             ;   back onto the stack
1BA7   22                       RET
```

```
                                ;;              XSTAT - transmit string to 25th status line
                                ;
                                ;               *XSTAT* places a string of displayable characters onto
                                ;               the 25th status line of the screen.
                                ;
                                ;
                                ;               ENTRY   (A) = position on 25th line
                                ;                       (DPTR) = address of string, final character must have
                                ;                                bit seven set.
                                ;
                                ;               EXIT    (A) = next position on 25th line
                                ;                       (DPTR) = position following string
                                ;
                                ;               USES    A, DPTR, PTR1

1BA8  75      A0      57        XSTAT:          MOV     P2, #HIGH (L25MEM)      ; Set address of memory
1BAB  24      80                                ADD     A, #LOW (L25MEM)        ; Add in offset to low byte
1BAD  F8                                        MOV     PTR1, A                 ; Point to memory

1BAE  E4                        XSTL:           CLR     A
1BAF  93                                        MOVC    A, @A+DPTR              ; Get data byte
1BB0  A3                                        INC     DPTR                    ; Bump ROM pointer
1BB1  20      E7      04                        JB      ACC.7, XSTR             ; IF bit set THEN exit out
1BB4  F2                                        MOVX    @PTR1, A                ; Output data
1BB5  08                                        INC     PTR1                    ; Bump RAM pointer
1BB6  80      F6                                SJMP    XSTL                    ; Go forever

1BB8  54      7F                XSTR:           ANL     A, #7FH                 ; Mask off high bit
1BBA  F2                                        MOVX    @PTR1, A                ; Output last data
1BBB  E8                                        MOV     A, PTR1                 ; Get pointer
1BBC  24      81                                ADD     A, #-LOW (L25MEM) + 1   ; Subtract offset
1BBE  22                                        RET                             ; Return
```

```
                                ;;              XLONOFF - transmit string "ON" or "OFF" to 25th status line
                                ;
                                ;               *XLONOFF* displays "ON" or "OFF" as indicated by the carry
                                ;               flag.
                                ;
                                ;
                                ;               ENTRY   (A) = position on 25th line
                                ;                       (c) = 0 indicates "OFF"   1 indicates "ON"
                                ;
                                ;               EXIT    (A) = next position on 25th line
                                ;
                                ;               USES    A, DPTR, PTR1

1BBF    90      1BC9            XLONOFF:        MOV     DPTR, #LNOFF            ; Get the "OFF" message
1BC2    50      03                              JNC     XLON1                   ; IF carry set THEN
1BC4    90      1BCC                            MOV     DPTR, #LNON             ;   set the "ON" message
1BC7    61      A8              XLON1:          AJMP    XSTAT                   ; Display message and return


1BC9    6F      66      E6      LNOFF:          DB      'of', ('f'+80H)
1BCC    6F      6E      A0      LNON:           DB      'on', (' '+80H)
```

```
                                        ;;              XMTS - transmit string
                                        ;
                                        ;               *XMTS* transmits a string of characters out of the terminal
                                        ;               regardless of whether the terminal is on line or not.
                                        ;
                                        ;
                                        ;               ENTRY   String to be sent must immediately follow the call
                                        ;                       to this routine.  Final character must have bit
                                        ;                       seven set.
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    A, DPTR

1BCF   D0      83               XMTS:   POP     DPH             ; (DPTR) = first character
1BD1   D0      82                       POP     DPL

1BD3   E4                       XMTS1:  CLR     A
1BD4   93                               MOVC    A, @A+DPTR      ; REPEAT fetch character
1BD5   20      E7      06               JB      ACC.7, XMTS2    ;   IF bit 7 set THEN exit
1BD8   12      139B                     CALL    TRANSMIT        ;   transmit character
1BDB   A3                               INC     DPTR            ;   bump pointer
1BDC   80      F5                       SJMP    XMTS1           ; UNTIL forever

1BDE   12      139B             XMTS2:  CALL    TRANSMIT        ; Transmit last character
1BE1   74      01                       MOV     A, #1
1BE3   73                               JMP     @A+DPTR         ; Return to code after string
```

```
                                ;;              XMTC - transmit character
                                ;
                                ;               *XMTC* transmits one character at the current cursor position
                                ;               including the appropriate escape sequences for entering and
                                ;               exiting the graphics and attribute modes for the mode the
                                ;               terminal is in.  If the character is protected then no
                                ;               action is taken.
                                ;
                                ;
                                ;               ENTRY   (A) = current transmit mode
                                ;                       XPOS and YPOS point to screen coordinate
                                ;
                                ;               EXIT    (A) = current transmit mode (updated)
                                ;
                                ;               USES    all

1BE4  C0  21            XMTC:   PUSH    ATTRIBUTES      ; Save attribute byte
1BE6  C0  E0                    PUSH    ACC             ; Save current attributes

1BE8  30  25  0A                JNB     GATM, XLO       ; IF not GATM THEN
1BEB  12  0188                  CALL    CPF             ;   check for protected
1BEE  50  05                    JNC     XLO             ;   IF protected THEN
1BF0  D0  E0                    POP     ACC             ;     restore values
1BF2  D0  21                    POP     ATTRIBUTES
1BF4  22                        RET                     ;     and return

1BF5  E5  11            XLO:    MOV     A, XPOS
1BF7  85  12  F0                MOV     B, YPOS
1BFA  12  0503                  CALL    CLA             ; Calculate line address

1BFD  D0  F0                    POP     B               ; Get current attributes

1BFF  53  83  EF                ANL     DPH, #ATRANL    ; Mask to attribute memory
1C02  E0                        MOVX    A, @DPTR
1C03  F5  21                    MOV     ATTRIBUTES, A   ; Get current attributes
1C05  43  83  10                ORL     DPH, #CHRORL    ; Mask to character memory
1C08  E0                        MOVX    A, @DPTR
1C09  A2  E7                    MOV     C, ACC.7        ; Get alt char flag
1C0B  92  0C                    MOV     ALTCHARA, C     ; Place in proper location
1C0D  54  7F                    ANL     A, #7FH         ; Mask alternate char set
1C0F  C2  0D                    CLR     GRPH            ; Clear graphic flag

1C11  B4  20  00                CJNE    A, #' ', XL0A

1C14  40  07            XL0A:   JC      XL1             ; IF graphic char THEN jump
1C16  B4  7F  0F                CJNE    A, #RUBOUT, XL3 ; IF displayable THEN jump

                                ;               Character is graphic
                                ;
1C19  74  5E                    MOV     A, #'^'         ; Map 7Fh to 5Eh
1C1B  80  09                    SJMP    XL2

1C1D  B4  1F  04        XL1:    CJNE    A, #1FH, XL1A   ; Special case map
1C20  74  5F                    MOV     A, #'_'
1C22  80  02                    SJMP    XL2
1C24  44  60            XL1A:   ORL     A, #01100000B   ; Make graphic printable
```

```
1C26  D2  0D        XL2:      SETB   GRPH                     ; Set graphic flag

                    ;         Save character to output
                    ;
1C28  C0  E0        XL3:      PUSH   ACC

                    ;         See if character is graphic character
                    ;
1C2A  20  0D  30              JB     GRPH, XL4                ; IF not graphic AND

1C2D  30  F5  5D              JNB    B.5, XL7                 ; IF previous was graphic THEN
1C30  C2  F5                  CLR    B.5                      ;   clear graphic flag

1C32  30  40  22              JNB    ANSI, XL3C               ;   IF ANSI mode THEN
1C35  20  F4  13              JB     B.4, XL3B                ;     IF previous not alt THEN
1C38  20  0C  09              JB     ALTCHARA, XL3A           ;       IF this not alt THEN
1C3B  71  CF                  ACALL  XMTS                     ;         transmit "EXIT GRAPHIC MODE"
1C3D  1B  5B  31              DB     ESC, '[11', ('m'+80H)
1C40  31  ED
1C42  80  49                  SJMP   XL7

1C44  71  CF        XL3A:     ACALL  XMTS                     ;       ELSE this is alt THEN
1C46  1B  28  B1              DB     ESC, '(', ('1'+80H)      ;         transmit "ENTER ALT MODE"
1C49  80  2E                  SJMP   XL4SET                   ;         flag in alt now

1C4B  20  0C  F6    XL3B:     JB     ALTCHARA, XL3A           ;     ELSE IF this not alt THEN
1C4E  71  CF                  ACALL  XMTS                     ;       transmit "ENTER NORMAL"
1C50  1B  28  C2              DB     ESC, '(', ('B'+80H)
1C53  C2  F4                  CLR    B.4                      ;       flag in normal now
1C55  80  36                  SJMP   XL7

1C57  71  CF        XL3C:     ACALL  XMTS                     ;   ELSE transmit in ZDS mode
1C59  1B  C7                  DB     ESC, ('G'+80H)
1C5B  80  30                  SJMP   XL7

1C5D  20  F5  2D    XL4:      JB     B.5, XL7                 ; IF previous not graphic THEN
1C60  D2  F5                  SETB   B.5                      ;   flag changes

1C62  30  40  24              JNB    ANSI, XL4C               ;   IF ANSI mode THEN
1C65  20  F4  15              JB     B.4, XL4B                ;     IF previous normal THEN
1C68  20  0C  09              JB     ALTCHARA, XL4A           ;       IF not alt char THEN
1C6B  71  CF        XL4AA:    ACALL  XMTS                     ;         transmit "ENTER GRAPHIC"
1C6D  1B  5B  31              DB     ESC, '[10', ('m'+80H)
1C70  30  ED
1C72  80  19                  SJMP   XL7

1C74  71  CF        XL4A:     ACALL  XMTS                     ;       ELSE
1C76  1B  28  B0              DB     ESC, '(', ('0'+80H)      ;         transmit "ENTER ALT GRAPHIC"
1C79  D2  F4        XL4SET:   SETB   B.4                      ;         flag in alt now
1C7B  80  10                  SJMP   XL7

1C7D  20  0C  F4    XL4B:     JB     ALTCHARA, XL4A           ;     ELSE this is normal THEN
1C80  71  CF                  ACALL  XMTS                     ;       transmit "ENTER NORMAL"
1C82  1B  28  C2              DB     ESC, '(' ('B'+80H)
1C85  C2  F4                  CLR    B.4                      ;       flag in normal now
1C87  80  E2                  SJMP   XL4AA                    ;       and place into graphic

```

```
    1 1C89    71      CF              XL4C:       ACALL   XMTS                ;   ELSE transmit in ZDS mode
    2 1C8B    1B      C6                          DB      ESC, ('F'+80H)
    3
    4                                 ;           See if character is in reverse video
    5                                 ;
    6 1C8D    C2      D5              XL7:        CLR     F0                  ; Clear "CHANGE" flag
    7
    8 1C8F    20      08      0E                  JB      RVVA, XL8           ; IF current not rev video AND
    9 1C92    30      F0      1B                  JNB     B.0, XL9            ; IF previous was rev video THEN
   10 1C95    C2      F0                          CLR     B.0                 ;   clear rev video
   11
   12 1C97    20      40      14                  JB      ANSI, XL8A          ; IF ZDS mode THEN
   13 1C9A    71      CF                          ACALL   XMTS                ;   output "EXIT REV VIDEO"
   14 1C9C    1B      F1                          DB      ESC, ('q'+80H)
   15 1C9E    80      10                          SJMP    XL9
   16
   17 1CA0    20      F0      0D      XL8:        JB      B.0, XL9            ; IF previous not rev video THEN
   18 1CA3    D2      F0                          SETB    B.0                 ;   set rev video
   19
   20 1CA5    20      40      06                  JB      ANSI, XL8A          ; IF ZDS mode THEN
   21 1CA8    71      CF                          ACALL   XMTS                ;   output "ENTER REV VIDEO"
   22 1CAA    1B      F0                          DB      ESC, ('p'+80H)
   23 1CAC    80      02                          SJMP    XL9
   24
   25 1CAE    D2      D5              XL8A:       SETB    F0                  ; ELSE flag changes
   26
   27                                 ;           See if character is underline
   28                                 ;
   29 1CB0    20      09      07      XL9:        JB      UNDLNA, XL10        ; IF not underline AND
   30
   31 1CB3    30      F1      0B                  JNB     B.1, XL11           ; IF previous was underline THEN
   32 1CB6    C2      F1                          CLR     B.1                 ;   clear underline flag
   33 1CB8    80      05                          SJMP    XL10A               ;   flag changes
   34
   35 1CBA    20      F1      04      XL10:       JB      B.1, XL11           ; IF previous not underline
   36 1CBD    D2      F1                          SETB    B.1                 ;   flag changes
   37 1CBF    D2      D5              XL10A:      SETB    F0
   38
   39                                 ;           See if character is blinking
   40                                 ;
   41 1CC1    20      0A      07      XL11:       JB      BLINKA, XL12        ; IF not blinking AND
   42
   43 1CC4    30      F2      0B                  JNB     B.2, XL13           ; IF previous was blinking THEN
   44 1CC7    C2      F2                          CLR     B.2                 ;   clear blinking flag
   45 1CC9    80      05                          SJMP    XL12A               ;   flag changes
   46
   47 1CCB    20      F2      04      XL12:       JB      B.2, XL13           ; IF previous not blinking THEN
   48 1CCE    D2      F2                          SETB    B.2                 ;   flag changes
   49 1CD0    D2      D5              XL12A:      SETB    F0
   50
   51                                 ;           See if character is half intensity
   52                                 ;
   53 1CD2    20      0B      07      XL13:       JB      HALFIA, XL14        ; IF not half AND
   54
   55 1CD5    30      F3      0B                  JNB     B.3, XL15           ; IF previous was half THEN
   56 1CD8    C2      F3                          CLR     B.3                 ;   clear half flag
   57 1CDA    80      05                          SJMP    XL14A               ;   flag changes
```

```
                1
1CDC  20  F3  04      XL14:     JB    B.3, XL15          ; IF previous not half THEN
1CDF  D2  F3                    SETB  B.3                ;   flag changes
1CE1  D2  D5          XL14A:    SETB  F0
                2
                3               ;     See if character is alternate graphic
                4               ;
1CE3  20  0C  19      XL15:     JB    ALTCHARA, XL16     ; IF not alt AND
                5
1CE6  30  F4  31                JNB   B.4, XL17          ; IF previous was alt THEN
1CE9  C2  F4                    CLR   B.4                ;   clear alt flag
                6
1CEB  30  40  2A                JNB   ANSI, XL16B        ;   IF ANSI mode THEN
1CEE  71  CF                    ACALL XMTS               ;     transmit "EXIT EVERYTHING"
1CF0  1B  28  C2                DB    ESC, '(', ('B'+80H)
                7
1CF3  30  F5  24                JNB   B.5, XL17          ;     IF graphics mode THEN
1CF6  71  CF                    ACALL XMTS               ;       transmit "ENTER GRAPHIC MODE"
1CF8  1B  5B  31                DB    ESC, '[10', ('m'+80H)
1CFB  30  ED
1CFD  80  1B                    SJMP  XL17
                8
1CFF  20  F4  18      XL16:     JB    B.4, XL17          ; IF previous not alt THEN
1D02  D2  F4                    SETB  B.4                ;   flag changes
                9
1D04  30  40  11                JNB   ANSI, XL16B        ;   IF ANSI mode THEN
1D07  20  F5  07                JB    B.5, XL16A         ;     IF no graphics THEN
1D0A  71  CF                    ACALL XMTS               ;       transmit "ENTER ALT MODE"
1D0C  1B  28  B1                DB    ESC, '(', ('1'+80H)
1D0F  80  09                    SJMP  XL17
               10
1D11  71  CF          XL16A:    ACALL XMTS               ;     ELSE transmit "ENTER ALT GRAPHIC"
1D13  1B  28  B0                DB    ESC, '(', ('0'+80H)
1D16  80  02                    SJMP  XL17
               11
1D18  D2  D5          XL16B:    SETB  F0
               12
                               ;     Transmit any changes
               13              ;
1D1A  30  D5  46      XL17:     JNB   F0, XL20           ; IF no changes THEN skip
               14
1D1D  20  40  1F                JB    ANSI, XL18         ; Check for mode
               15
                               ;     ZDS mode changes
               16              ;
1D20  71  CF                    ACALL XMTS               ;     set attributes
1D22  1B  F3                    DB    ESC, ('s'+80H)
1D24  C0  F0                    PUSH  B                  ;     save attributes
1D26  E5  F0                    MOV   A, B               ;     reorder bits
1D28  13                        RRC   A                  ;     set reverse video
1D29  13                        RRC   A                  ;     set underline
1D2A  92  F3                    MOV   B.3, C             ;     put in position
1D2C  13                        RRC   A                  ;     set blink
1D2D  92  F1                    MOV   B.1, C             ;     put in position
1D2F  13                        RRC   A                  ;     set half intensity
1D30  92  F2                    MOV   B.2, C             ;     put in position
1D32  E5  F0                    MOV   A, B
```

```
    1 1D34   54      1F                              ANL     A, #00011111B           ;       mask proper bits
    2 1D36   24      30                              ADD     A, #'0'                 ;       add offset for sequence
    3 1D38   12      139B                            CALL    TRANSMIT                ;       output attribute data
    4 1D3B   D0      F0                              POP     B                       ;       restore attributes
    5 1D3D   80      24                              SJMP    XL20
    6
    7                                        ;      ANSI mode changes
    8                                        ;
    9 1D3F   71      CF              XL18:           ACALL   XMTS                    ;   clear any prev attributes
   10 1D41   1B      5B      B0                      DB      ESC, '[', ('0'+80H)
   11
   12 1D44   30      F3      04                      JNB     B.3, XL18B              ;   IF half intensity THEN
   13 1D47   71      CF                              ACALL   XMTS
   14 1D49   3B      B2                              DB      ';', ('2'+80H)
   15
   16 1D4B   30      F1      04      XL18B:          JNB     B.1, XL18C              ;   IF underline THEN
   17 1D4E   71      CF                              ACALL   XMTS
   18 1D50   3B      B4                              DB      ';', ('4'+80H)
   19
   20 1D52   30      F2      04      XL18C:          JNB     B.2, XL18D              ;   IF blinking THEN
   21 1D55   71      CF                              ACALL   XMTS
   22 1D57   3B      B5                              DB      ';', ('5'+80H)
   23
   24 1D59   30      F0      04      XL18D:          JNB     B.0, XL18E              ;   IF reverse video THEN
   25 1D5C   71      CF                              ACALL   XMTS
   26 1D5E   3B      B7                              DB      ';', ('7'+80H)
   27
   28 1D60   71      CF              XL18E:          ACALL   XMTS                    ;   output last character
   29 1D62   ED                                      DB      ('m'+80H)
   30
   31 1D63   D0      E0              XL20:           POP     ACC
   32 1D65   12      139B                            CALL    TRANSMIT                ; Output character
   33
   34 1D68   53      F0      3F                      ANL     B, #00111111B           ; Mask previous
   35 1D6B   E5      21                              MOV     A, ATTRIBUTES
   36 1D6D   54      3F                              ANL     A, #00111111B           ; Mask current
   37 1D6F   45      F0                              ORL     A, B                    ; Combine to make new previous
   38 1D71   D0      21                              POP     ATTRIBUTES              ; Restore true attributes
   39 1D73   22                                      RET
```

```
                                        ;;              XMTL - transmit line
                                        ;               XMTL1 - entry to transmit line
                                        ;
                                        ;               *XMTL* transmits the entire contents of the line as defined
                                        ;               by transmit character *XMTC*.
                                        ;
                                        ;               *XMTL1* needs a current transmit mode.
                                        ;
                                        ;
                                        ;               ENTRY   (A) = current transmit mode (*XMTL1*)
                                        ;                       YPOS points to screen line
                                        ;
                                        ;               EXIT    (A) = current transmit mode (updated)
                                        ;
                                        ;               USES    all
                                        XMTL:           CLR     A                       ; Start with no attributes
1D75  C0  11                            XMTL1:          PUSH    XPOS                    ; Save current position
1D77  75  11  00                                        MOV     XPOS, #0                ; Start at beginning of line
1D7A  71  E4                            XML2:           ACALL   XMTC                    ; REPEAT transmit character
1D7C  05  11                                            INC     XPOS                    ;   bump position
1D7E  AC  11                                            MOV     WORK, XPOS
1D80  BC  50  F7                                        CJNE    WORK, #NUMCHARS, XML2   ; UNTIL done
1D83  D0  11                                            POP     XPOS                    ; Restore position
1D85  FC                                                MOV     WORK, A                 ; Save (A)
1D86  12  04EA                                          CALL    BLINAD                  ; Update address
1D89  EC                                                MOV     A, WORK                 ; Restore (A)
1D8A  22                                                RET                             ; Return
```

```
                                        ;;              SXMTC - special transmit character
                                        ;
                                        ;               *SXMTC* transmits the character at the current cursor
                                        ;               position as defined by transmit character *XMTC*.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    all

1D8B  30      40      0F        SXMTC:  JNB     ANSI, SXC1      ; IF ANSI mode THEN
1D8E  12      09D1                      CALL    GNP             ;   set parameter
1D91  60      19                        JZ      XMTP            ;   IF zero THEN xmt page
1D93  14                                DEC     A
1D94  60      12                        JZ      SXMTL           ;   IF one THEN xmt line
1D96  14                                DEC     A
1D97  60      04                        JZ      SXC1            ;   IF two THEN xmt character
1D99  14                                DEC     A
1D9A  60      2A                        JZ      XMT25           ;   IF three THEN xmt 25th line
1D9C  22                                RET                     ;   ELSE return

1D9D  C0      3F              SXC1:     PUSH    PFIELD          ; Save protected fields
1D9F  75      3F      00                MOV     PFIELD, #0      ; Make none protected
1DA2  E4                                CLR     A               ; No attributes set
1DA3  71      E4                        ACALL   XMTC            ; Transmit character
1DA5  D0      3F                        POP     PFIELD          ; Restore protected fields
1DA7  22                                RET
```

```
                                ;;              SXMTL - special transmit line
                                ;
                                ;               *SXMTL* transmits the entire contents of the line as defined
                                ;               by transmit character *XMTC*.  The end of the line is signaled
                                ;               by a carriage return.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

1DA8    B1      74              SXMTL:          ACALL   XMTL                    ; Transmit line
1DAA    80      14                              SJMP    XMP3                    ; Send CR, bell, and return
```

```
                                ;;              XMTP - transmit page
                                ;
                                ;               *XMTP* transmits the entire contents of the page (excluding
                                ;               25th line) as defined by transmit character *XMTC*.  The end
                                ;               of the page is signaled by a carriage return.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

1DAC  E4                        XMTP:           CLR     A                       ; Start with no attributes set
1DAD  C0  12                                    PUSH    YPOS                    ; Save current position
1DAF  75  12  00                                MOV     YPOS, #0
1DB2  B1  75                    XMP1:           ACALL   XMTL1                   ; REPEAT transmit line
1DB4  05  12                                    INC     YPOS                    ;   bump position
1DB6  AC  12                                    MOV     WORK, YPOS
1DB8  BC  18  F7                                CJNE    WORK, #MAXLINE, XMP1    ; UNTIL done

1DBB  D0  12                    XMP2:           POP     YPOS                    ; Restore position
1DBD  12  04EA                                  CALL    BLINAD                  ; Update address

1DC0  71  CF                    XMP3:           ACALL   XMTS                    ; Send CR to terminate page
1DC2  8D                                        DB      (CR+80H)

1DC3  02  066B                                  JMP     RING                    ; Signal done to user
```

```
                                ;;              XMT25 - transmit 25th line
                                ;
                                ;               *XMT25* transmits the 25th line in the same manner as *XMTL*
                                ;               but only if the 25th line is enabled.  The end of the line
                                ;               is signaled by a carriage return.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

1DC6  30      15      F7        XMT25:          JNB     L25EN, XMP3             ; IF 25th line enabled THEN
1DC9  C0      12                                PUSH    YPOS                    ;   save current position
1DCB  75      12      18                        MOV     YPOS, #MAXLINE          ;   do 25th line
1DCE  B1      74                                ACALL   XMTL                    ;   transmit 25th line
1DD0  80      E9                                SJMP    XMP2                    ;   restore, update, return
```

```
                                ;;              PRINT - print page
                                ;
                                ;               *PRINT* acts exactly like *XMTP* (transmit page) except that
                                ;               the data goes out to the opposite port and every line ends with
                                ;               a carriage return and line feed.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

1DD2  D2      23          PRINT:    SETB    PRNTF                   ; Flag printing

1DD4  12      13C4                  LCALL   DOXOFF                  ; Output an XOFF

1DD7  D1      0F                    ACALL   CMPORT                  ; Toggle port

1DD9  C0      3F                    PUSH    PFIELD                  ; Save protected fields
1DDB  75      3F      00            MOV     PFIELD, #0              ; Make all unprotected

1DDE  E4                            CLR     A                       ; Start with no attributes set
1DDF  C0      12                    PUSH    YPOS                    ; Save current position
1DE1  75      12      00            MOV     YPOS, #0

1DE4  B1      75          PRNT1:    ACALL   XMTL1                   ; REPEAT transmit line
1DE6  C0      E0                    PUSH    ACC                     ;   save attributes
1DE8  71      CF                    ACALL   XMTS                    ;   transmit CR and LF
1DEA  0D      8A                    DB      CR, (LF+80H)
1DEC  D0      E0                    POP     ACC                     ;   restore attributes

1DEE  05      12                    INC     YPOS                    ;   bump position
1DF0  AC      12                    MOV     WORK, YPOS
1DF2  BC      18      EF            CJNE    WORK, #MAXLINE, PRNT1   ; UNTIL done

1DF5  D0      12                    POP     YPOS                    ; Restore position
1DF7  12      04EA                  LCALL   BLINAD                  ; Update address

1DFA  D0      3F                    POP     PFIELD                  ; Restore protected fields

1DFC  D1      0F                    ACALL   CMPORT                  ; Toggle port again

1DFE  20      22      03            JB      HSSTOPF, PRNT2          ; IF not hold screen THEN
1E01  12      13D5                  LCALL   DOXON                   ;   output an -XON-

1E04  C2      23          PRNT2:    CLR     PRNTF                   ; Flag done printing

1E06  12      02E8                  LCALL   LKB                     ; Look at keyboard
1E09  50      03                    JNC     PRNT3                   ; IF no keyboard data THEN

1E0B  02      066B                  LJMP    RING                    ;   signal done to user
1E0E  22                  PRNT3:    RET                             ; ELSE return
```

```
                                        ;;              CMPORT - complement port
                                        ;
                                        ;               *CMPORT* toggles direction of the I/O port.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    A, DPTR

1E0F    30      13      FD      CMPORT: JNB     OKTRANS, CMPORT         ; Wait until transmitter empty
1E12    B2      02                      CPL     PORTF                   ; Toggle port flag
1E14    B2      3C                      CPL     PRTF2                   ; Toggle EAROM flag
1E16    C2      16                      CLR     XOFFRCVED               ; Clear any received -XOFF-

                                        ;               SJMP    WLATCH                  ; Write and return


                                        ;;              WLATCH - write latch
                                        ;
                                        ;               -WLATCH- writes the current value of the byte *XLATCH* out
                                        ;               to the memory mapped latch.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    A, DPTR

1E18    E5      20              WLATCH: MOV     A, XLATCH               ; Get byte
1E1A    90      3000                    MOV     DPTR, #MLATCH           ; Get address
1E1D    F0                              MOVX    @DPTR, A                ; Output data to latch
1E1E    22                              RET
```

```
                                ;;              DCLK - Display clock
                                ;
                                ;               DCLKO - always display clock
                                ;
                                ;               *DCLK* displays the current setting of the clock on the 25th
                                ;               line once every second only.
                                ;
                                ;               *DCLKO* always displays the current setting of the clock on
                                ;               the 25th line.
                                ;
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    A, B, DPTR

1E1F  E5  3E            DCLK:   MOV     A, DISPSEC
1E21  B5  3B  01                CJNE    A, TSEC, DCLKO          ; IF no change in seconds THEN
1E24  22                        RET                             ;   exit

1E25  85  3B  3E        DCLKO:  MOV     DISPSEC, TSEC           ; Init *DISPSEC*
1E28  E5  3D                    MOV     A, THOUR
1E2A  12  0A92                  CALL    CBA                     ; Convert hours to ASCII
1E2D  C0  82                    PUSH    DPL                     ; Save ones digit
1E2F  E5  83                    MOV     A, DPH                  ; Get tens digit
1E31  B4  30  02                CJNE    A, #'0', DCLK1          ; IF tens digit = '0' THEN
1E34  74  20                    MOV     A, #' '                 ;   suppress leading digit
1E36  90  57C8          DCLK1:  MOV     DPTR, #L25MEM+72
1E39  F0                        MOVX    @DPTR, A                ; Place on screen
1E3A  D0  E0                    POP     ACC
1E3C  A3                        INC     DPTR
1E3D  F0                        MOVX    @DPTR, A                ; Place on screen
1E3E  74  3A                    MOV     A, #':'
1E40  A3                        INC     DPTR
1E41  F0                        MOVX    @DPTR, A                ; Place ':' on screen
1E42  E5  3C                    MOV     A, TMIN
1E44  12  0A92                  CALL    CBA                     ; Convert minutes to ASCII
1E47  C0  82                    PUSH    DPL                     ; Save ones digit
1E49  E5  83                    MOV     A, DPH
1E4B  90  57CB                  MOV     DPTR, #L25MEM+75
1E4E  F0                        MOVX    @DPTR, A                ; Place tens digit on screen
1E4F  D0  E0                    POP     ACC
1E51  A3                        INC     DPTR
1E52  F0                        MOVX    @DPTR, A                ; Place ones digit on screen
1E53  74  3A                    MOV     A, #':'
1E55  A3                        INC     DPTR
1E56  F0                        MOVX    @DPTR, A                ; Place ':' on screen
1E57  E5  3B                    MOV     A, TSEC
1E59  12  0A92                  CALL    CBA                     ; Convert seconds to ASCII
1E5C  C0  82                    PUSH    DPL                     ; Save ones digit
1E5E  E5  83                    MOV     A, DPH
1E60  90  57CE                  MOV     DPTR, #L25MEM+78
1E63  F0                        MOVX    @DPTR, A                ; Place tens digit on screen
1E64  D0  E0                    POP     ACC
```

```
      1 1E66    A3                                    INC     DPTR
      2 1E67    F0                                    MOVX    @DPTR, A                ; Place ones digit on screen
      3 1E68    22                                    RET
```

```
                                ;;              DISPSTAT - display status line
                                ;
                                ;               *DISPSTAT* displays everything that can be displayed onto
                                ;               the 25th line if the option is enabled.  This includes
                                ;               clock, caps locked, on/off line indicators, and insert
                                ;               character mode.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

1E69  30  37  03  DISPSTAT:     JNB     L25ON, DSPR             ; IF line 25 off OR
1E6C  30  15  01                JNB     L25EN, DSP0             ; IF line 25 enabled THEN
1E6F  22                DSPR:   RET                             ;   return

1E70  74  08            DSP0:   MOV     A, #00001000B
1E72  71  2E                    ACALL   SCLRL                   ; Clear with half intensity

1E74  D1  25                    ACALL   DCLK0                   ; Display clock

1E76  E4                        CLR     A
1E77  90  1E94                  MOV     DPTR, #MSGCL            ; Caps locked
1E7A  30  35  02                JNB     CAPLOCKF, DSP1
1E7D  71  A8                    ACALL   XSTAT

1E7F  74  10            DSP1:   MOV     A, #16
1E81  90  1EA5                  MOV     DPTR, #MSGOL            ; Offline
1E84  20  2E  02                JB      ONLINE, DSP2
1E87  71  A8                    ACALL   XSTAT

1E89  74  1D            DSP2:   MOV     A, #29
1E8B  90  1EB3                  MOV     DPTR, #MSGIC            ; Insert character
1E8E  30  21  02                JNB     ICMODEF, DSP3
1E91  71  A8                    ACALL   XSTAT                   ; Output

1E93  22                DSP3:   RET
```

```
                                        ;               Caps locked message
                                        ;
1E94    00      20      20      MSGCL:          DB      00H, '  CAPS LOCKED  ', (00H+80H)
1E97    43      41      50
1E9A    53      20      4C
1E9D    4F      43      4B
1EA0    45      44      20
1EA3    20      80
                                        ;
                                        ;               Off line message
                                        ;
1EA5    00      20      20      MSGOL:          DB      00H, '  OFF LINE  ', (00H+80H)
1EA8    4F      46      46
1EAB    20      4C      49
1EAE    4E      45      20
1EB1    20      80
                                        ;
                                        ;               Insert character message
                                        ;
1EB3    00      20      20      MSGIC:          DB      00H, '  INSERT MODE  ', (00H+80H)
1EB6    49      4E      53
1EB9    45      52      54
1EBC    20      4D      4F
1EBF    44      45      20
1EC2    20      80
```

```
                                ;;              KBCHK - keyboard check
                                ;
                                ;               *KBCHK* waits for the keyboard to send -KB_POWER- within
                                ;               75 ms.  If it is not found then an error condition exists.
                                ;               This routine must only be done after a hardware reset so that
                                ;               it is known that the keyboard is going to send the data.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    all

 1EC4  7A     0D        KBCHK:  MOV     INDX1, #13
 1EC6  7B     00        KBC0:   MOV     INDX2, #0               ; REPEAT init count
 1EC8  12     02B6      KBC1:   CALL    FCKB                    ;   REPEAT set any data
 1ECB  40     04              JC      KBC2                    ;     IF data AND
 1ECD  B4     8F       F8             CJNE    A, #KB_POWER, KBC1      ;       IF power up THEN
 1ED0  22                            RET                     ;         return
 1ED1  DB     F5        KBC2:   DJNZ    INDX2, KBC1             ;   UNTIL done
 1ED3  DA     F1              DJNZ    INDX1, KBC0             ; UNTIL all done

 1ED5  43     40       08             ORL     ERRORS, #00001000B      ; ELSE error #3
 1ED8  22                            RET
```

```
                                        ;;              ROMCHK - ROM checksum
                                        ;
                                        ;               -ROMCHK- does a checksum on the ROM and compares it with
                                        ;               the known value in ROM.
                                        ;
                                        ;
                                        ;               ENTRY   none
                                        ;
                                        ;               EXIT    none
                                        ;
                                        ;               USES    all

1ED9  7C      00            ROMCHK:     MOV     WORK, #0                ; Init checksum
1EDB  90      0000                      MOV     DPTR, #0                ; Init pointer
1EDE  7A      20                        MOV     INDX1, #20H             ; First index
1EE0  7B      00                        MOV     INDX2, #00H             ; Init count

1EE2  E4                      RMC1:       CLR     A                       ; REPEAT
1EE3  93                                  MOVC    A, @A+DPTR              ;   set ROM byte
1EE4  2C                                  ADD     A, WORK                 ;   add to previous amount
1EE5  FC                                  MOV     WORK, A                 ;   replace
1EE6  A3                                  INC     DPTR                    ;   bump pointer
1EE7  DB      F9                          DJNZ    INDX2, RMC1             ; UNTIL done
1EE9  DA      F7                          DJNZ    INDX1, RMC1             ; UNTIL all done

1EEB  60      03                          JZ      RMC2                    ; IF error THEN
1EED  43      40      01                  ORL     ERRORS, #00000001B      ;   error #0
1EF0  22                      RMC2:       RET
```

```
                                ;;              RAMCHK - RAM check
                                ;
                                ;               -RAMCHK- checks the RAM for any failures.  It writes two
                                ;               different values into character and attribute RAM and reads
                                ;               them back.  Any errors are flagged.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    none
                                ;
                                ;               USES    a'l

1EF1  90  5000          RAMCHK: MOV     DPTR, #CHARMEM          ; Point to character RAM
1EF4  7C  55                    MOV     WORK, #01010101B        ; First pattern

1EF6  7A  08            RAM0:   MOV     INDX1, #08H             ; First index
1EF8  7B  00                    MOV     INDX2, #00H             ; Second index

1EFA  EC                RAM1:   MOV     A, WORK                 ; REPEAT
1EFB  F0                        MOVX    @DPTR, A                ;   write pattern
1EFC  53  83  EF                ANL     DPH, #ATRANL            ;   move to attribute RAM
1EFF  F0                        MOVX    @DPTR, A                ;   write pattern
1F00  E0                        MOVX    A, @DPTR                ;   read pattern
1F01  54  0F                    ANL     A, #0FH                 ;   mask
1F03  FD                        MOV     TEMP, A                 ;   save
1F04  EC                        MOV     A, WORK                 ;   set orisinal pattern
1F05  54  0F                    ANL     A, #0FH                 ;   mask it also
1F07  B5  05  14                CJNE    A, TEMPA, RAMERR        ;   IF error THEN exit
1F0A  43  83  10                ORL     DPH, #CHRORL            ;   move to character RAM again
1F0D  E0                        MOVX    A, @DPTR                ;   read pattern
1F0E  B5  04  0D                CJNE    A, WORKA, RAMERR        ;   IF error THEN exit
1F11  A3                        INC     DPTR                    ;   bump pointer
1F12  DB  E6                    DJNZ    INDX2, RAM1             ; UNTIL done
1F14  DA  E4                    DJNZ    INDX1, RAM1             ; UNTIL all done

1F16  BC  55  04                CJNE    WORK, #01010101B, RAM2  ; IF pattern is orisinal THEN
1F19  7C  AA                    MOV     WORK, #10101010B        ;   change to second pattern
1F1B  80  D9                    SJMP    RAM0                    ;   and do it again

1F1D  22                RAM2:   RET                             ; IF no error THEN return

1F1E  43  40  02        RAMERR: ORL     ERRORS, #00000010B      ; ELSE flag error #1 and return
1F21  22                        RET
```

```
                                ;;              NVRCHK - non-volatile RAM checksum
                                ;
                                ;               *NVRCHK* does a checksum on the NVRAM and returns the value.
                                ;
                                ;
                                ;               ENTRY   none
                                ;
                                ;               EXIT    (A) = checksum value of NVRAM
                                ;
                                ;               USES    A, WORK, INDX1, PTR1

1F22  E4                        NVRCHK:         CLR     A                       ; Init for checksum
1F23  78  25                                    MOV     PTR1, #25H
1F25  7A  0F                                    MOV     INDX1, #15

1F27  26                        NVR1:           ADD     A, @PTR1                ; REPEAT add next value
1F28  08                                        INC     PTR1                    ;   bump RAM pointer
1F29  DA  FC                                    DJNZ    INDX1, NVR1             ; UNTIL done
1F2B  22                                        RET
```

```
                                        ;;      ERROR - diagnostic error
                                        ;
                                        ;       -ERROR- outputs the error messages for any errors that
                                        ;       may have been flagged in the -ERRORS- byte.  The status
                                        ;       line is displayed if there are no errors.  A bell is
                                        ;       given when finished and no errors occured.
                                        ;
                                        ;       Format of the error byte is as follows:
                                        ;
                                        ;       BIT     0 = ROM checksum error
                                        ;               1 = RAM error
                                        ;               2 = CRTC error
                                        ;               3 = Keyboard error
                                        ;               4 = Non-volatile RAM checksum error
                                        ;
                                        ;
                                        ;       ENTRY   none
                                        ;
                                        ;       EXIT    none
                                        ;
                                        ;       USES    all

1F2C    E5      40              ERROR:  MOV     A, ERRORS
1F2E    70      05                      JNZ     ERRO                    ; IF no errors THEN
1F30    D1      69                      ACALL   DISPSTAT                ;   display status
1F32    02      066B                    LJMP    RING                    ;   ring bell and return

1F35    E4                      ERRO:   CLR     A
1F36    71      2E                      ACALL   SCLRL                   ; Clear line full intensity
1F38    D1      25                      ACALL   DCLKO                   ; Display clock
1F3A    D1      0F                      ACALL   CMPORT                  ; Change ports

1F3C    85      40      F0              MOV     B, ERRORS               ; Get errors

1F3F    71      CF                      ACALL   XMTS                    ; Output message
1F41    45      52      52      ERM1:   DB      'ERROR -', (' '+80H)
        1F44    4F      52      20
        1F47    2D      A0

1F49    E4                              CLR     A
1F4A    90      1F41                    MOV     DPTR, #ERM1
1F4D    71      A8                      ACALL   XSTAT                   ; Output to 25th line

1F4F    30      F0      17              JNB     B.0, ERR1               ; IF ROM checksum error THEN
1F52    FF                              MOV     WORK2, A
1F53    71      CF                      ACALL   XMTS                    ;   output message
1F55    52      4F      4D      ERM2:   DB      'ROM checksum ', (' '+80H)
        1F58    20      63      68
        1F5B    65      63      6B
        1F5E    73      75      6D
        1F61    20      A0
1F63    EF                              MOV     A, WORK2
1F64    90      1F55                    MOV     DPTR, #ERM2
1F67    71      A8                      ACALL   XSTAT                   ;   output to 25th line

1F69    30      F1      14      ERR1:   JNB     B.1, ERR2               ; IF RAM error THEN
```

```
                1F6C    FF                                      MOV     WORK2, A
                1F6D    71      CF                              ACALL   XMTS                    ;   output message
                1F6F    52      41      4D      ERM3:           DB      'RAM fault ', ('' '+80H)
                1F72    20      66      61
                1F75    75      6C      74
                1F78    20      A0
                1F7A    EF                                      MOV     A, WORK2
                1F7B    90      1F6F                            MOV     DPTR, #ERM3
                1F7E    71      A8                              ACALL   XSTAT                   ;   output to 25th line
                
                1F80    30      F2      15      ERR2:           JNB     B.2, ERR3               ; IF CRTC error THEN
                1F83    FF                                      MOV     WORK2, A
                1F84    71      CF                              ACALL   XMTS                    ;   output message
                1F86    43      52      54      ERM4:           DB      'CRTC error ', ('' '+80H)
                1F89    43      20      65
                1F8C    72      72      6F
                1F8F    72      20      A0
                1F92    EF                                      MOV     A, WORK2
                1F93    90      1F86                            MOV     DPTR, #ERM4
                1F96    71      A8                              ACALL   XSTAT                   ;   output to 25th line
                
                1F98    30      F3      13      ERR3:           JNB     B.3, ERR4               ; IF keyboard error THEN
                1F9B    FF                                      MOV     WORK2, A
                1F9C    71      CF                              ACALL   XMTS                    ;   output message
                1F9E    4B      65      79      ERM5:           DB      'Keyboard ', ('' '+80H)
                1FA1    62      6F      61
                1FA4    72      64      20
                1FA7    A0
                1FA8    EF                                      MOV     A, WORK2
                1FA9    90      1F9E                            MOV     DPTR, #ERM5
                1FAC    71      A8                              ACALL   XSTAT                   ;   output to 25th line
                
                1FAE    30      F4      18      ERR4:           JNB     B.4, ERR5               ; IF NVRAM error THEN
                1FB1    FF                                      MOV     WORK2, A
                1FB2    71      CF                              ACALL   XMTS                    ;   output message
                1FB4    4E      56      52      ERM6:           DB      'NVRAM checksum', ('' '+80H)
                1FB7    41      4D      20
                1FBA    63      68      65
                1FBD    63      6B      73
                1FC0    75      6D      A0
                1FC3    EF                                      MOV     A, WORK2
                1FC4    90      1FB4                            MOV     DPTR, #ERM6
                1FC7    71      A8                              ACALL   XSTAT                   ;   output to 25th line
                
                1FC9    71      CF              ERR5:           ACALL   XMTS                    ; Output CR, LF's
                1FCB    0D      0A      0D                      DB      CR, LF, CR, (LF+80H)
                1FCE    8A
                
                1FCF    C1      0F                              AJMP    CMPORT                  ; Change ports back
```

*** Z-29 COMPUTER TERMINAL FIRMWARE VERSION 1.03 ***

```
                                ;;              RAMCONST - RAM constants
                                ;
                                ;               *RAMCONST* is the table with all of the RAM constants that
                                ;               the terminal will power up with.
                                ;
1FD1   D0                       RAMCONST:       DB      LOW (IBUF)              ; ISTORE
1FD2   D0                                       DB      LOW (IBUF)              ; IFETCH
1FD3   00                                       DB      0                       ; ICOUNT

1FD4   F0                                       DB      LOW (OBUF)              ; OSTORE
1FD5   F0                                       DB      LOW (OBUF)              ; OFETCH

1FD6   00                                       DB      0                       ; XPOS
1FD7   00                                       DB      0                       ; YPOS
1FD8   50                                       DB      HIGH (CHARMEM)          ; LINADH
1FD9   00                                       DB      LOW (CHARMEM)           ; LINADL

1FDA   00                                       DB      HIGH (NORM)             ; DSPADH
1FDB   87                                       DB      LOW (NORM)              ; DSPADL

1FDC   00                                       DB      0                       ; TSCROLL
1FDD   18                                       DB      MAXLINE                 ; BFIX

1FDE   00                                       DB      0                       ; PMADRH
1FDF   00                                       DB      0                       ; PMADRL
1FE0   00                                       DB      0                       ; PMNUM
1FE1   00                                       DB      0                       ; PMVALUE

1FE2   00                                       DB      0                       ; KBIDNUM
1FE3   01                                       DB      1                       ; HSLINE
1FE4   00                                       DB      0                       ; GSET

1FE5   39                                       DB      00111001B               ; XLATCH        bits #1
1FE6   00                                       DB      00000000B               ; ATTRIBUTES    bits #2
1FE7   08                                       DB      00001000B               ; Bits #3
1FE8   0C                                       DB      00001100B               ; Bits #4
1FE9   30                                       DB      00110000B               ; Bits #5

1FEA   00                                       DB      0                       ; XXPOS
1FEB   00                                       DB      0                       ; XYPOS

1FEC   20                                       DB      32                      ; BLNKRATE
1FED   00                                       DB      0                       ; BLNKCOUNT
1FEE   00                                       DB      0                       ; AUTOCNT

1FEF   00                                       DB      0                       ; TCNT
1FF0   00                                       DB      0                       ; TSEC
1FF1   00                                       DB      0                       ; TMIN
1FF2   00                                       DB      0                       ; THOUR

1FF3   FF                                       DB      255                     ; DISPSEC

1FF4   00                                       DB      0                       ; PFIELD
```

```
                                ;;              CHKS - check sum
                                ;
                                ;               -CHKS- is the checksum byte for the terminal ROM.  It is
                                ;               choosen so that the total sum of all the bytes gives a value
                                ;               of zero.

         1FFF                                   ORG     1FFFH

1FFF     D6                     CHKS:           DB      0D6H                    ; Checksum byte

                                                END
```

SYMBOL TABLE

AC	%00D6	01	ACA1	0A06	03	ACA2	0A08	03	ACA3	0A0A	03	ACA3A	0A12	03
ACA5	0A16	03	ACC	%00E0	00	ACCESS	001E	01	ACD	0CC1	03	ACDN	0CBB	03
ACD1	0CC0	03	ACHR2	003D	01	ACL	0C4C	03	ACLFT	0C46	03	ACLO	0C4B	03
ACPR	0AAF	03	ACP1	0ACF	03	ACP1A	0AD8	03	ACP2	0ADC	03	ACP3	0ACC	03
ACR	0C74	03	ACRT	0C6E	03	ACR1	0C73	03	ACU	0CA3	03	ACUP	0C9D	03
ACU1	0CA2	03	ADA	0EC3	03	ADAM	0ECC	03	ADM3	0042	01	ADM3ES	06A7	03
AESC1	06A1	03	ALLINI	10AF	03	ALLI1	10C1	03	ALTCHA	000C	01	ALTGRP	000F	01
ANSI	0040	01	ANSIES	0697	03	ANSIT =	0798	03	ANSITL=	000C	00	APBT	0CC6	03
APBT1	0CCB	03	APCA	09FA	03	APDC	0E32	03	APDCO	0E37	03	APDC1	0E38	03
APDL	0DB6	03	APDLO	0DBB	03	APDL1	0DBC	03	APIC	0DE9	03	APICO	0DEE	03
APIC1	0DEF	03	APIL	0D7D	03	APILO	0D82	03	APIL1	0D83	03	APR	0AE9	03
APRNT	0ADF	03	ARAMP	0AF6	03	ARM	085F	03	ARMT =	0871	03	ARMTL =	0005	00
ASBR	0AEA	03	ASBR1	0AEF	03	ASBR2	0AF2	03	ASCP	0EEF	03	ASGM	0A3B	03
ASGMT =	0A50	03	ASGMTL=	0007	00	ASGO	0A4E	03	ASGO	0A49	03	ASG1	0A3D	03
ASM	083D	03	ASMR	084F	03	ASMT =	0850	03	ASMTL =	0005	00	ASPF	0A65	03
ASPF1	0A68	03	ASPF1A	0A72	03	ASPF1B	0A77	03	ASPF1C	0A7C	03	ASPF1D	0A81	03
ASPF1E	0A86	03	ASPF1F	0A8C	03	ASPF2	0A8E	03	ATBC	062B	03	ATRANL=	00EF	
ATRIBM=	4000		ATTRIB=	0021		AT2	0635	03	AUSCP	0F03	03	AUTOCN=	0039	
AUTOCR	002C	01	AUTOLF	002D	01	AUTOWR	002B	01	A1M	0880	03	A1RM	08CC	03
A1RT =	08DE	03	A1RTL =	0003	00	A1SM	08B0	03	A1SR	08C2	03	A1ST =	08C3	03
A1STL =	0003	00	A2M	0898	03	A2RM	0944	03	A2SM	08FF	03	B	%00F0	00
BANK0 =	0000		BANK1 =	0008		BAUDRA=	0033		BELL =	0007		BFIX =	0018	
BLINAD	04EA	03	BLINKA	000A	01	BLINKF	0001	01	BLNKCN=	0038		BLNKRA=	0037	
BRTAB =	0E5F	03	BR1	0FD2	03	BR2	0FDC	03	BS =	0008		BTAB	0602	03
BTL	060A	03	BTL1	061A	03	BTR	061F	03	CAN =	0018		CAPLOC	0035	01
CBA	0A92	03	CCRSR =	0080		CDN	0CA8	03	CDN1	0CAE	03	CDN2	0CB2	03
CDN3	0CB7	03	CHARME=	5000		CHKS	1FFF	03	CHRORL=	0010		CINT =	00A0	
CLA	0503	03	CLFO	0BC2	03	CLFT =	0C3D	03	CLKF	0033	01	CLKRUN	0097	01
CLRFLG	0020	01	CLRL	0BA0	03	CLRLIN	0B96	03	CLRS	0BAC	03	CLRTAB	0598	03
CLRTB	059D	03	CMPORT	1E0F	03	CONTLF	001D	01	CPF	0188	03	CPFR1	018C	03
CPFR2	01A7	03	CPF1	018E	03	CPF2	01A3	03	CPR	0E83	03	CPRM	0EA9	03
CPRO	0E8B	03	CPROA	0E8F	03	CPR1	0E97	03	CPR2	0EA6	03	CR =	000D	
CRESET=	0000		CRNBK	0038	01	CRT	0C64	03	CRT1	0C6A	03	CRUL	0032	01
CR1	1198	03	CR2	119A	03	CSET =	00E0		CSTOP =	0040		CSTRT =	0020	
CS1	0BB5	03	CS2	0BB9	03	CTAB1 =	0119	03	CTAB1L=	0008	00	CTAB2 =	0107	03
CTAB2L=	000E	00	CTAB3 =	0128	03	CTAB3L=	0006	00	CTS	00B5	01	CUA	0CD6	03
CUA1	0CE4	03	CUA2	0CE8	03	CUA2A	0CEB	03	CUA3	0CEF	03	CUBO	0CF2	03
CUB1	0CFE	03	CUC	0D04	03	CUCO	0D14	03	CUCOA	0D1C	03	CUC1	0D0E	03
CUC1A	0D12	03	CUC2	0D1E	03	CUP	0C8D	03	CUP1	0C94	03	CUP1A	0C95	03
CUP1B	0C99	03	CUR	0CF1	03	CURSOR	1184	03	CUR1	0D00	03	CUR2	0D20	03
CY	%00D7	01	DC	0B5F	03	DCLK	1E1F	03	DCLKO	1E25	03	DCLK1	1E36	03
DCMDRE=	2001		DDATRE=	2000		DEOL	0B00	03	DISPSE=	003E		DISPST	1E69	03
DKI	0FB9	03	DMAANL=	00CF		DMADRH=	000A		DMADRL=	000B		DMAMEM=	6000	
DMAORL=	0020		DMATYP	00B4	01	DMLN	%0001		DMLNA =	0009		DMPTR	%0000	
DMPTRA=	0008		DM1	1225	03	DM1A	1243	03	DM1B	1235	03	DM1C	1238	03
DM1D	123D	03	DM1E	1246	03	DM2	1248	03	DM3	124A	03	DM4	125B	03
DOXOFF	13C4	03	DOXON	13D5	03	DPH	%0083	00	DPL	%0082	00	DSADRH=	0015	
DSADRL=	0016		DSPR	1E6F	03	DSP0	1E70	03	DSP1	1E7F	03	DSP2	1E89	03
DSP3	1E93	03	DSR	0D24	03	DSRERR	0D40	03	DSROK	0D42	03	DSRO	0D29	03
DSROA	0D2C	03	DSR1	0D30	03	DSR1A	0D35	03	DSR2	0D39	03	DSR2A	0D3E	03
DSTARE=	2001		DTR	0094	01	DX	0014	01	D25L	0B15	03	EA	%00AF	01
EACR	0B62	03	EALF	0B68	03	EAM	0B03	03	EAROM =	7000		EAROMR	0003	01
EAROMS	0000	01	EBD	0BF1	03	EBL	0BCB	03	EBLNK	0F30	03	EB1	0BF8	03
EB2	0BFA	03	EB3	0C01	03	EC	0B5C	03	EED	0C08	03	EEL	0BDB	03
EER	0C18	03	EERM	0B74	03	EE1	0C0C	03	EE1A	0C10	03	EGATM	0B6E	03
EGM	0F39	03	EGTT =	08AA	03	EGTTL =	0002	00	EHALF	0F2A	03	EHSM	0B3C	03
EICM	0B54	03	EID	0A1A	03	EID1	0A20	03	EID2	0A25	03	EIL	0A2B	03

SYMBOL TABLE

EIL1	0A31	03	EIL2	0A36	03	EIR	0A2A	03	EKAM	0B4E	03	EKC	0B26	03
EKI	0FB3	03	EKSM	0B48	03	ELB	07BC	03	ELBT =	07D0	03	ELBTL =	001E	00
ENBLCU	001A	01	EOL	0BE0	03	EOL1	0BE3	03	EQMT =	0892	03	EQMTL =	0002	00
ERM	001B	01	ERM1	1F41	03	ERM2	1F55	03	ERM3	1F6F	03	ERM4	1F86	03
ERM5	1F9E	03	ERM6	1FB4	03	ERROR	1F2C	03	ERRORS=	0040		ERRO	1F35	03
ERR1	1F69	03	ERR2	1F80	03	ERR3	1F98	03	ERR4	1FAE	03	ERR5	1FC9	03
ERVM	0F33	03	ES	%00AC	01	ESC =	001B		ESCCOD	067B	03	ESCRET	0696	03
ET0	%00A9	01	ET1	%00AB	01	EUNDL	0F2D	03	EX0	%00A8	01	EX1	%00AA	01
EZM	0B07	03	E25L	0B0B	03	FCIF	01F6	03	FCKB	02B6	03	FCOF	024C	03
FILL	1211	03	FI1	01FC	03	FI2	0202	03	FI3	0217	03	FKDLY	02E1	03
FKDLY1	02E4	03	FKL	02C7	03	FKR	02DD	03	FK1	02BE	03	FK2	02D8	03
FL1	1219	03	FOR	026B	03	FO1	0253	03	FREQ	0039	01	FULLDP	002F	01
FO	%00D5	01	GATM	0025	01	GDES	0F4F	03	GDESR	0F7A	03	GDES0	0F52	03
GDES1	0F55	03	GDES2	0F5A	03	GDES2A	0F62	03	GDES3	0F65	03	GDES3A	0F71	03
GETNUM	1B45	03	GNP	09D1	03	GNPF	0019	01	GNP1	09DC	03	GNP2	09E8	03
GNUMR	1B50	03	GNUMS	1B93	03	GNUM0	1B46	03	GNUM1	1B4A	03	GNUM2	1B52	03
GNUM2A	1B57	03	GNUM2B	1B5A	03	GNUM2L	1B5D	03	GNUM4	1B69	03	GNUM4A	1B6E	03
GNUM4B	1B71	03	GNUM4C	1B76	03	GPICK	0F86	03	GPICKT=	0F92	03	GRPH	000D	01
GSEL	0012	01	GSET =	001F		GODES	0F3F	03	GOPICK	0F7B	03	G1DES	0F47	03
G1PICK	0F81	03	HALFIA	000B	01	HAZ	0043	01	HAZESC	06B0	03	HAZINI	11D4	03
HAZT =	0768	03	HAZTL =	0010	00	HCLFT	0C51	03	HCLFT1	0C5F	03	HCRT	0C79	03
HCRT1	0C88	03	HEED	0C19	03	HEE1	0C20	03	HEE1A	0C24	03	HIR	11D8	03
HNDSHK	003E	01	HSLINE=	001E		HSMODE	0036	01	HSSTOP	0022	01	HT =	0009	
HTAB	0636	03	HTC	0627	03	HTS	0621	03	HTS1	0623	03	HTO	0640	03
HT1	0652	03	HT2	065B	03	HT3	0665	03	IBUF =	57D0		ICMODE	0021	01
ICOUNT=	000E		IC1	114F	03	IDT	0EAB	03	IDTM	0EB0	03	IDTT	0EB3	03
IDTTM	0EBF	03	IE	%00A8	00	IEDMA =	0082		IEINIT=	0095		IEXTIO	002A	03
IE0	%0089	01	IE1	%008B	01	IFCP	0079	03	IFETCH=	000D		IFPA	0082	03
IFPRET	0106	03	IFP1	009C	03	IFP1A	00A4	03	IFP1B	00AC	03	IFP1C	00AF	03
IFP2	00B4	03	IFP2A	00CF	03	IFP3	00D5	03	IFP3A	00DE	03	IFP4	00EA	03
IFP5	00F4	03	IFP5A	00F6	03	IFP6	0104	03	IHI =	0015		IKB1	11A4	03
IKB2	11AC	03	IKB3	11B4	03	IKB4	11BC	03	IKB5	11C4	03	IKB6	11C8	03
ILEN =	0020		ILO =	0009		INDX1	%0002		INDX1A=	0002		INDX2	%0003	
INDX2A=	0003		INIT	1045	03	INITCR	111E	03	INITKB	119E	03	INMOV	10A7	03
INT0	%00B2	01	INT1	%00B3	01	IN3	1068	03	IN3A	108A	03	IN3B	1086	03
IN4	109C	03	IN5	10A3	03	IP	%00B8	00	IPINIT=	0003		IRQ	1329	03
IRQ1	1347	03	IRQ2	1352	03	IRQ3	135E	03	IRQ4	1361	03	IRQ4A	1379	03
IRQ5	1390	03	ISCON =	005A		ISTORE=	000C		ITMOD =	0023		ITMRO	002D	03
IT0	%0088	01	IT1	%008A	01	I1	0008	03	I2	0010	03	I3	0019	03
I5	0029	03	KBACC	0448	03	KBBREA	0FCC	03	KBCHK	1EC4	03	KBCP	02EE	03
KBCPLK	0FBF	03	KBCRT =	0466	03	KBCRTL=	0007	00	KBC_BE=	0081		KBC_DC=	0083	
KBC_DK=	0088		KBC_DR=	0087		KBC_EC=	0082		KBC_EK=	0089		KBC_EL=	0084	
KBC_ER=	0086		KBC_ID=	0080		KBC_OF=	008A		KBC_ON=	008B		KBC_XL=	0085	
KBCO	1EC6	03	KBC1	1EC8	03	KBC2	1ED1	03	KBDISF	001F	01	KBDT =	044B	03
KBDTL =	0009	00	KBENT	043E	03	KBE1	0445	03	KBFNT =	04CF	03	KBFNTL=	0009	00
KBIDNU=	001D		KBIN	0091	01	KBOUT	0090	01	KBPDT =	047B	03	KBPDTL=	001C	00
KBSCR0	0FE3	03	KBSCR1	0FED	03	KBSCR2	0FF0	03	KBSPAC	0436	03	KBS1	043C	03
KBTAB	041B	03	KB_ACC=	009F		KB_BRE=	0089		KB_CAP=	008A		KB_CNT=	0020	
KB_COM=	009D		KB_DAS=	009C		KB_DEC=	009A		KB_DOW=	0081		KB_ENT=	009B	
KB_ERA=	0085		KB_F1 =	0080		KB_F2 =	0081		KB_F3 =	0082		KB_F4 =	0083	
KB_F5 =	0084		KB_F6 =	0085		KB_F7 =	0086		KB_F8 =	0087		KB_F9 =	0088	
KB_HEL=	0086		KB_HOM=	0084		KB_ID =	0098		KB_LEF=	0082		KB_POW=	008F	
KB_RIG=	0083		KB_SCR=	0087		KB_SET=	0088		KB_SHF=	0040		KB_SPA=	008C	
KB_TAB=	008B		KB_UP =	0080		KB_0 =	0090		KB_1 =	0091		KB_2 =	0092	
KB_3 =	0093		KB_4 =	0094		KB_5 =	0095		KB_6 =	0096		KB_7 =	0097	
KB_8 =	0098		KB_9 =	0099		KC1	0FC8	03	KPA	03CD	03	KPADAL	0031	01
KPADSH	0030	01	KPB	03DB	03	KPC	0402	03	KPCOF	0414	03	KPC1	0419	03

SYMBOL TABLE

KPR	02F3	03	KPR2	0413	03	KP0	02F4	03	KP1	0303	03	KP10	03CB	03
KP11	0384	03	KP12	03B4	03	KP13	03A3	03	KP14	03A1	03	KP15	03B2	03
KP16	0365	03	KP16A	036B	03	KP2	030F	03	KP3	0335	03	KP4	034B	03
KP5	0345	03	KP6	034F	03	KP7	0355	03	KP8	038A	03	KP9	0371	03
KSR	0FFE	03	KTM1	0431	03	KTM2	0434	03	KT0	042A	03	KT1	042D	03
LCIF	021B	03	LF =	000A		LINADH=	0013		LINADL=	0014		LI1	0221	03
LKB	02E8	03	LKBR	02ED	03	LKUP1	13F8	03	LNOFF	1BC9	03	LNON	1BCC	03
LN1B1	14C7	03	LN1B2	14CF	03	LN3B	1648	03	LN3B1	164D	03	LN3B10	166B	03
LN3B11	1670	03	LN3B12	1675	03	LN3B13	167A	03	LN3B14	167F	03	LN3B3	1652	03
LN3B5	1657	03	LN3B7	165C	03	LN3B8	1661	03	LN3B9	1666	03	LN3C1	1684	03
LN3C2	1689	03	LN3C3	168E	03	LN3C4	1693	03	LN3E1	1698	03	LN3E2	169C	03
LN3G1	16A0	03	LN3G2	16A8	03	LN4B1	1769	03	LN4B2	176F	03	LN4D1	1775	03
LN4D2	177D	03	LN4D3	1785	03	LN4D4	178D	03	LN7D1	19B7	03	LN7D2	19C0	03
LN7D3	19C9	03	LN7D4	19D2	03	LN7F1	19DB	03	LN7F2	19DC	03	LN8B1	1AE4	03
LN8B2	1AED	03	LOOKUP	13E9	03	LORDER=	0047		LOT =	1432	03	LOTL =	0013	00
L1	146E	03	L1A	149E	03	L1T =	146B	03	L1TL =	0001	00	L2	150D	03
L2A	1516	03	L2B	151A	03	L2C	152B	03	L2D	1538	03	L2T =	1501	03
L2TL =	0004	00	L25EN	0015	01	L25MEM=	5780		L25ON	0037	01	L3	1598	03
L3A	15C0	03	L3AT =	161E	03	L3ATL =	000E	00	L3B	15E7	03	L3C	1600	03
L3D	161C	03	L3T =	158C	03	L3TL =	0004	00	L4	16F8	03	L4A	1717	03
L4B	173B	03	L4T =	16EC	03	L4TL =	0004	00	L5	1808	03	L5T =	17FC	03
L5TL =	0004	00	L6	1897	03	L6T =	188B	03	L6TL =	0004	00	L7	193F	03
L7C1	198D	03	L7C2	1996	03	L7D	19AE	03	L7T =	1933	03	L7TL =	0004	00
L8	1A8C	03	L8A	1AB0	03	L8T =	1A74	03	L8TL =	0004	00	L8TL2 =	0004	00
L8T2 =	1A80	03	MAIN	0036	03	MAXCHA=	004F		MAXLIN=	0018		MLATCH=	3000	
MN1	0042	03	MN1A	004E	03	MN1B	0051	03	MN1C	005D	03	MN1D	0062	03
MN2	0065	03	MN3	006E	03	MODE =	0028		MONITO	003A	01	MSGCL	1E94	03
MSGIC	1EB3	03	MSGOL	1EA5	03	NKC	0B2D	03	NORM	0087	03	NRM0	0093	03
NRM1	009A	03	NULL =	0000		NUMCHA=	0050		NUMLIN=	0019		NVRCHK	1F22	03
NVRSUM=	0034		NVR1	1F27	03	OBUF =	57F0		OFETCH=	0010		OKTRAN	0013	01
OLEN =	57F7		ONLINE	002E	01	OSTORE=	000F		OV	%00D2	01	P	%00D0	01
PBS =	0C3D	03	PBS1	0C42	03	PCIF	01CA	03	PCKB	026D	03	PCLK	0B7A	03
PCL1A	0B7F	03	PCL1B	0B88	03	PCL1C	0B91	03	PCL2	0B95	03	PCOF	022A	03
PCR	055C	03	PCRLF	054F	03	PDC	0DF4	03	PDC1	0E11	03	PDC2	0E31	03
PDL	0D88	03	PDLA	0D95	03	PDLO	0D9B	03	PDL1	0D9F	03	PDL2	0DA9	03
PFIELD=	003F		PIC	0DC1	03	PIC1	0DD3	03	PIL	0D47	03	PILA	0D52	03
PILRET	0D72	03	PILRT1	0D75	03	PILO	0D5C	03	PIL1	0D60	03	PIL2	0D6A	03
PIR	01F1	03	PI1	01DD	03	PI2	01E4	03	PI3	01F0	03	PKDLY	02B5	03
PKL	0291	03	PKR	02B3	03	PKO	027B	03	PK1	027F	03	PK1A	028F	03
PK2	0294	03	PK3	02A5	03	PK4	02A9	03	PK5	02B0	03	PLF	0561	03
PLFCR	0555	03	PLFLP	0588	03	PLFRET	0594	03	PLF1	0567	03	PLF2	056D	03
PLF3	0571	03	PMADRH=	0019		PMADRL=	001A		PMBUF =	57F8		PMNUM =	001B	
PMVALU=	001C		PORTF	0002	01	PO1	022F	03	PO2	0234	03	PO3	023F	03
PRINT	1DD2	03	PRLF	0518	03	PRNTF	0023	01	PRNT1	1DE4	03	PRNT2	1E04	03
PRNT3	1E0E	03	PROTEC	000E	01	PRTF2	003C	01	PRTYEN	0028	01	PRTYEV	002A	01
PRTYST	0029	01	PS	%00BC	01	PSD	0971	03	PSDB	0982	03	PSDB1	0985	03
PSDB2	098A	03	PSDE	09A2	03	PSDF	0018	01	PSDRET	09CF	03	PSD1	09A0	03
PSD2	09B7	03	PSD3	09BF	03	PSD4	09CD	03	PSOF	0E6D	03	PSW	%00D0	00
PTR1	%0000		PTR1A =	0000		PTR2	%0001		PTR2A =	0001		PTO	%00B9	01
PT1	%00BB	01	PUTCHR	0157	03	PU1	017D	03	PU2	0186	03	PX0	%00B8	01
PX1	%00BA	01	P0	%0080	00	P1	%0090	00	P2	%00A0	00	P3	%00B0	00
RAMCHK	1EF1	03	RAMCON	1FD1	03	RAMERR	1F1E	03	RAMP	0AFA	03	RAM0	1EF6	03
RAM1	1EFA	03	RAM2	1F1D	03	RB8	%009A	01	RD	%00B7	01	REN	%009C	01
REPF	0034	01	RGL	066F	03	RGR	067A	03	RI	%0098	01	RING	066B	03
RLL	0537	03	RLR	0543	03	RL1	0548	03	RL1A	054C	03	RMC1	1EE2	03
RMC2	1EF0	03	RMS	092C	03	RMSRET	0943	03	RMST =	0956	03	RMSTL =	0009	00
RMS1	0938	03	ROMCHK	1ED9	03	RS0	%00D3	01	RS1	%00D4	01	RTS	0095	01

SYMBOL TABLE

RT1	0B0A	03	RUBOUT=	007F		RVVA	0008	01	RXD	%00B0	01	R2_BAC=	1024	
R2_DCL=	103F		R2_DIS=	1042		R2_ESC=	102D		R2_FIL=	100C		R2_ICU=	101B	
R2_IKB=	101E		R2_IN =	1000		R2_IN2=	1003		R2_IRQ=	100F		R2_KY1=	1027	
R2_KY2=	102A		R2_PRN=	103C		R2_SER=	1012		R2_SET=	1021		R2_STA=	1006	
R2_STO=	1009		R2_TC =	1030		R2_TL =	1033		R2_TP =	1036		R2_T25=	1039	
R2_XMT=	1015		R2_XON=	1018		SAV2	1B0D	03	SBC	0B33	03	SBLR	0F96	03
SBR	0E4B	03	SBRET	0E5E	03	SBRT	0FAF	03	SBR0	0E5A	03	SBUF	%0099	00
SB1	0F9C	03	SB2	0FA6	03	SB3	0FA7	03	SCH	0C36	03	SCLRL	1B2E	03
SCLRLR	1B2C	03	SCON	%0098	00	SCP	0EE8	03	SCRNSA	003B	01	SCRN1 =	004F	
SCRN2A=	0098		SCRN2B=	00DC		SCRN3 =	0099		SCRN4A=	004E		SCRN4B=	004F	
SC1	1166	03	SC2	1171	03	SD1	11F0	03	SERIAL	12CB	03	SERIN	12DD	03
SEROUT	1313	03	SETCF	117B	03	SETCRT	1152	03	SETDIS	082A	03	SETLCK	0093	01
SETNOR	082F	03	SETTAB	05A2	03	SETUP	1417	03	SETUPR	1AF6	03	SETUPS	1B02	03
SETUP0	141B	03	SETUP1	141E	03	SETUP2	14D7	03	SETUP3	1542	03	SETUP4	16B0	03
SETUP5	1795	03	SETUP6	1865	03	SETUP7	18F4	03	SETUP8	19DD	03	SFA	126F	03
SFB	12A5	03	SFCKB	1407	03	SFCKB1	140F	03	SFCKB2	1416	03	SFILL	1B21	03
SFIL1	1B23	03	SFIL2	1B25	03	SFL	12AC	03	SF1	1286	03	SF2	128D	03
SF3	12AB	03	SF4	12B9	03	SF5	12BF	03	SF6	12C7	03	SHIFTF	001C	01
SI0	12ED	03	SI1	12F3	03	SI1A	12FF	03	SI2	1301	03	SI3	1306	03
SLP	1B40	03	SMS	08E7	03	SMSRET	08FE	03	SMST	= 0911	03	SMSTL =	0009	00
SMS1	08F3	03	SM0	%009F	01	SM1	%009E	01	SM2	%009D	01	SOFTIN	10E4	03
SOFTI1	111B	03	SOR	0E7F	03	SP	%0081	00	SPF	01A9	03	SPO	01AE	03
SP1	01C0	03	SP2	01C4	03	SP3	01C6	03	SRET	1AFF	03	SRT	1317	03
SRT1	1326	03	STAB	013A	03	STACK =	005F		START	0030	03	STARTD	11DC	03
STB1	05B0	03	STB2	05B9	03	STCALL	083A	03	STFILL	1262	03	STJMP	0836	03
STOPDM	11FD	03	STP1A	1420	03	STP1B	142C	03	STP2A	14D9	03	STP2B	14DB	03
STP2C	14E7	03	STP2C1	14EC	03	STP2D	14EE	03	STP2D1	14F3	03	STP2E	14F5	03
STP2F	14F9	03	STP2F1	14FB	03	STP3A	1544	03	STP3B	1550	03	STP3B1	1562	03
STP3C	1567	03	STP3C1	1572	03	STP3C2	157C	03	STP3C3	1580	03	STP3D	1584	03
STP3E	1588	03	STP4A	16B2	03	STP4B	16BE	03	STP4C	16C3	03	STP4C1	16CF	03
STP4C2	16D9	03	STP4D	16DB	03	STP4D1	16E3	03	STP4E	16E8	03	STP5A	1798	03
STP5B	17A4	03	STP5B1	17C1	03	STP5B2	17C4	03	STP5B3	17E5	03	STP5C	17F0	03
STP5D	17F4	03	STP5E	17F8	03	STP6A	1867	03	STP6B	1875	03	STP6C	1879	03
STP6D	187D	03	STP6E	1884	03	STP7A	18F6	03	STP7B	1904	03	STP7C	1908	03
STP7D	190C	03	STP7D1	1916	03	STP7D2	191C	03	STP7D3	1923	03	STP7D4	1927	03
STP7E	192C	03	STP8A	19DF	03	STP8B	19ED	03	STP8B1	1A08	03	STP8B3	1A1D	03
STP8C	1A21	03	STP8D	1A32	03	STP8E	1A3A	03	STP8E0	1A41	03	STP8E1	1A61	03
STRET	083C	03	ST1	013B	03	ST2	014F	03	SUC	0B37	03	SUC1	0B39	03
SXC1	1D9D	03	SXMTC	1D8B	03	SXMTL	1DA8	03	SXON1	13E2	03	SXR	13E1	03
TAB	05C1	03	TABTAB=	0029		TBL1	05D6	03	TBL2	05DE	03	TBR	05FC	03
TB1	05F3	03	TB2	05ED	03	TB3	05EA	03	TB4	05E7	03	TB5	05F7	03
TB6	05DC	03	TB8	%009B	01	TCNT	= 003A		TCON	%0088	00	TEMP	%0005	
TEMPA =	0005		TERMIN	11CB	03	TF0	%008D	01	TF1	%008F	01	THOUR =	003D	
TH0	%008C	00	TH1	%008D	00	TI	%0099	01	TL0	%008A	00	TL1	%008B	00
TMIN =	003C		TMOD	%0089	00	TRA	13AC	03	TRANS	139E	03	TRANSM	139B	03
TROK	13B4	03	TROK1	13BA	03	TROK2	13BD	03	TRW	13AE	03	TR0	%008C	01
TR1	%008E	01	TSCROL=	0017		TSEC	= 003B		TXD	%00B1	01	T0	%00B4	01
T1	%00B5	01	UNDEF	0767	03	UNDLNA	0009	01	USCP	0EF4	03	USCP1	0EFF	03
USCP2	0F01	03	UXOFFS	0011	01	VSP	0092	01	VSPF	0024	01	WEOL	0AFD	03
WLATCH	1E18	03	WORK	%0004		WORKA	= 0004		WORK1	%0006		WORK1A=	0006	
WORK2	%0007		WORK2A=	0007		WR	%00B6	01	XACR	0B65	03	XALF	0B6B	03
XATR	0F26	03	XBLINA	09F4	03	XERM	0B77	03	XGATM	0B71	03	XGM	0F3C	03
XHSM	0B43	03	XHSM1	0B45	03	XICM	0B58	03	XKAM	0B51	03	XKSM	0B4B	03
XLATCH=	0020		XLONOF	1BBF	03	XLON1	1BC7	03	XLO	1BF5	03	XLOA	1C14	03
XL1	1C1D	03	XL1A	1C24	03	XL10	1CBA	03	XL10A	1CBF	03	XL11	1CC1	03
XL12	1CCB	03	XL12A	1CD0	03	XL13	1CD2	03	XL14	1CDC	03	XL14A	1CE1	03
XL15	1CE3	03	XL16	1CFF	03	XL16A	1D11	03	XL16B	1D18	03	XL17	1D1A	03

SYMBOL TABLE

XL18	1D3F	03	XL18B	1D4B	03	XL18C	1D52	03	XL18D	1D59	03	XL18E	1D60	03
XL2	1C26	03	XL20	1D63	03	XL3	1C28	03	XL3A	1C44	03	XL3B	1C4B	03
XL3C	1C57	03	XL4	1C5D	03	XL4A	1C74	03	XL4AA	1C6B	03	XL4B	1C7D	03
XL4C	1C89	03	XL4SET	1C79	03	XL7	1C8D	03	XL8	1CA0	03	XL8A	1CAE	03
XL9	1CB0	03	XML2	1D7A	03	XMP1	1DB2	03	XMP2	1DBB	03	XMP3	1DC0	03
XMTC	1BE4	03	XMTL	1D74	03	XMTL1	1D75	03	XMTP	1DAC	03	XMTS	1BCF	03
XMTS1	1BD3	03	XMTS2	1BDE	03	XMTXOF	13D1	03	XMTXON	13E5	03	XMT25	1DC6	03
XOFF =	0013		XOFFRC	0016	01	XOFFSE	0010	01	XON =	0011		XPCKB	0B2A	03
XPOS =	0011		XRVM	0F36	03	XR2_DI	0B23	03	XST	1B9D	03	XSTAB	09F7	03
XSTABX	1404	03	XSTAT	1BA8	03	XSTATX	1401	03	XSTL	1BAE	03	XSTR	1BB8	03
XSTX	13FE	03	XXPOS =	0035		XYPOS =	0036		YPOS =	0012		ZAR	0F25	03
ZATR	0F08	03	ZATR0	0F14	03	ZDS	0041	01	ZDSESC	06B8	03	ZDSSBR	0E3D	03
ZDST	06DF	03	ZDS0	0E49	03	ZE1A	06C1	03	ZE1B	06C6	03	ZE1C	06CC	03
ZE2	06DE	03	ZTAB	0ED3	03	ZTABT =	0EDF	03	ZTABTL=	0003	00			

$$$DSG	0000	00
$$$BSG	00B5	01
$$$XSG	0000	02
$$$CSG	2000	03

MODULE: Z29ROM
ERRORS DETECTED: 0

FREE CORE: 10574. WORDS
B:Z29,B:Z29=A:Z29/E:ABS/E:LC/C

CROSS REFERENCE TABLE (CREF V01-05)

CROSS REFERENCE TABLE (CREF.V01-05.)

CROSS REFERENCE TABLE (CREF V01-05)

CROSS REFERENCE TABLE (CREF V01-05)

CROSS REFERENCE TABLE (CREF V01-05)

```
DMAMEM   6-24#  163-18  163-19  171-11
DMAORL   6-34#  172-29
DMATYP  11-47#  169-22*  170-20  170-21*  172-50*
DMLN     7-29#  170-43  170-46  170-48  170-50  170-55  170-57  171-3
DMLNA    7-30#  163-17*  169-24  175-24*
DMPTR    7-27#  171-4  171-5
DMPTRA   7-28#
DOXOFF 174-42  178-14#  230-17
DOXON  157-1  178-41#  230-46
DSADRH   8-19#  17-26  57-17*  69-17*  69-38*  124-16*
DSADRL   8-20#  17-25  57-16*  57-30*  57-33*  57-36*  69-18*  69-37*  124-15*
DSPO   234-17  234-20#
DSP1   234-27  234-30#
DSP2   234-32  234-35#
DSP3   234-37  234-40#
DSPR   234-16  234-18#
DSR     68-20  125-15#
DSRO   125-16  125-19#
DSROA  125-19  125-20#
DSR1   125-20  125-23#
DSR1A  125-24  125-27#
DSR2   125-27  125-30#
DSR2A  125-31  125-32#
DSRERR 125-30  125-34#
DSROK  125-32  125-35#
DSTARE   6-5#  163-42  164-13  164-16  175-28
DTR     11-41#  178-19*  178-46*
DX       9-31#  22-14*  166-25  166-29*
E25L    78-34  99-15#
EACR    71-47  79-1  107-14#
EALF    78-55  108-14#
EAM     61-10  98-13#
EAROM    6-26#  159-35  211-17
EAROMR   9-8#  159-29*  159-31*
EAROMS   9-5#  211-30*  211-32*
EB1    115-41  115-44#
EB2    115-45#  115-52
EB3    115-46  115-49#
EBD     61-48  88-19  115-40#
EBL     62-4  89-20  114-14#  115-42  115-47
EBLNK   91-23  144-49#
EC      81-45  106-13#
EE1    116-16#  116-21
EE1A   116-17  116-18#
EED     61-24  63-32  88-15  116-14#
EEL     62-1  89-24  114-38#
EER    116-18  116-22#  116-47
EERM    71-44  110-14#
EGATM   71-35  109-14#
EGM     61-20  91-29  146-13#
EGTT    74-21  74-32#  74-40
EGTTL   74-20  74-40#
EHALF   63-50  91-17  144-13#
EHSM    61-41  78-40  102-13#  193-50
EICM    61-14  71-41  105-13#
EID     67-35  88-13#
EID1    88-14  88-17#
```

CROSS REFERENCE TABLE (CREF V01-05.)

```
EID2     88-18    88-21#
EIL      67-38    89-14#
EIL1     89-15    89-18#
EIL2     89-19    89-22#
EIR      88-22    88-25#   89-23
EKAM     61-11    65-28    78-52   104-13#
EKC      81-36   100-14#
EKI      19-25    62-16    63-41    72-36   151-13#
EKSM     62-9     78-49   103-13#
ELB      65-13    66-16#
ELBT     66-22    67-11#   68-46
ELBTL    66-21    68-46#
ENBLCU    9-40#  106-13*  106-31*  166-24
EOL      61-25    63-29    89-16   115-14#  116-14
EOL1    115-16#  116-42
EQMT     73-21    73-32#   73-40
EQMTL    73-20    73-40#
ERM       9-41#  110-14*  110-34*  168-16*  168-38*  172-23
ERM1    240-37#  240-40
ERM2    240-46#  240-48
ERM3    241-3#   241-5
ERM4    241-11#  241-13
ERM5    241-19#  241-21
ERM6    241-27#  241-29
ERR0    240-25   240-29#
ERR1    240-43   240-51#
ERR2    240-51   241-8#
ERR3    241-8    241-16#
ERR4    241-16   241-24#
ERR5    241-24   241-32#
ERROR   161-43   207-40   240-24#
ERRORS   11-22#  160-17*  161-17*  163-53*  207-17*  207-38   236-25*  237-28*  238-44*  240-24   240-34
ERVM     62-5     91-26   145-13#
ESC       2-12#   19-38    33-55    34-40    34-55    36-2     37-25    37-26    42-7     42-10    42-13    42-16    42-19    42-22
         42-25    42-28    42-31    94-17   137-33   138-18   138-44   139-20   221-18   221-22   221-27   221-32   221-42   221-46
        221-52   222-2    222-14   222-22   223-15   223-19   223-28   223-32   223-46   224-10
ESCCOD   18-20    19-39    57-14#
ESCRET   57-29    57-32    57-35    57-37#
EUNDL    91-20   144-31#
EZM      75-35    76-33    98-31#
FCIF     16-22    26-15#   56-22
FCKB     16-41    30-17    31-15#  153-18   181-15   181-18   207-27   236-18
FCOF     16-32    29-15#
FI1      26-16    26-20#
FI2      26-20    26-23#
FI3      26-31    26-34#
FILL    170-25   170-27#
FK1      31-17    31-21#
FK2      31-33    31-36#
FKDLY    31-28    31-29    31-37    31-40    31-46#
FKDLY1   31-47    31-47#
FKL      31-27#   31-38
FKR      31-34    31-40#
FL1     170-30#  170-35
FO1      29-16    29-20#
FOR      29-28    29-32#
FREQ     10-28#  164-30   164-38   170-45   175-43   203-1*   204-24
```

CROSS REFERENCE TABLE (CREF V01-05)

```
FULLDP  10-10#   28-18   188-55*  190-25
GODES   65-16    147-26#
GOPICK  19-51    148-4    149-16#
G1DES   65-19    147-35#
G1PICK  19-48    148-10   149-20#
GATM     9-55#   109-14*  109-34*  168-17*  220-21
GDES    147-30   147-39   147-44#
GDESO   147-45#  147-49
GDES1   147-44   147-48#
GDES2   147-48   147-51#
GDES2A  147-54   147-55#
GDES3   147-46   148-1#
GDES3A  148-1    148-7#
GDESR   147-53   147-55   148-12#
GETNUM  196-31   196-44   214-17#
GNP      54-50    71-13    72-13    75-13    76-13    78-13    81-13    85-15#   87-16    87-25    88-13    89-14    90-14    90-21
         92-20    94-15    95-35    96-17   111-13   111-17   111-21   118-14   119-37   121-42   122-45   123-13   125-15   125-23
        127-14   129-14   131-14   133-14   139-15   141-32   142-35   150-16   226-15
GNP1     85-15    85-20#
GNP2     85-23    85-26#
GNPF      9-39#   58-15*   85-15    85-16*
GNUMO   214-19#  214-49
GNUM1   214-24#  214-41   214-56   215-2    215-21
GNUM2   214-28   214-34#
GNUM2A  214-34   214-36#
GNUM2B  214-35   214-40#
GNUM2L  214-42#  214-43
GNUM4   214-36   214-53#
GNUM4A  214-53   214-55#
GNUM4B  214-55   214-56#
GNUM4C  215-1    215-2#
GNUMR   214-29#  214-54
GNUMS   215-14   215-25#  215-28
GPICK   149-18   149-24#
GPICKT  149-28   149-43#
GRPH      9-20#   18-21   146-13*  146-31*  220-42*  221-1*   221-9
GSEL      9-29#  148-4    148-10   149-16*  149-20*
GSET      8-32#  148-2*   148-3*   148-7*   148-9*   149-17   149-21
HALFIA    9-18#  143-27*  144-13*  168-37*  222-53
HAZ      10-43#   17-22    18-18    33-52    34-38    34-53    35-49   124-17   124-39   137-14   160-11   174-27   174-35   193-40
HAZESC   57-25    57-36    59-14#
HAZINI  160-12   168-37#  193-41
HAZT     59-15    63-11#   64-4
HAZTL    59-14    64-4#
HCLFT    20-11   118-39#
HCLFT1  118-44   118-47#
HCRT     20-14   120-15#
HCRT1   120-20   120-23#
HEE1    116-45#  116-54
HEE1A   116-46   116-47#
HEED     63-35   116-41#
HIR     168-18   168-39#
HNDSHK   10-33#   16-29   177-28   178-17   178-44   189-4*   190-33
HSLINE    8-31#   17-35*  102-14*  154-22*  154-27*
HSMODE   10-21#   17-30   102-13*  102-32*  154-14   193-47   195-14
HSSTOP    9-52#   16-21    17-36*  102-34*  154-21*  230-45
HT        2-6#    19-44    20-16    37-22
```

CROSS REFERENCE TABLE (CREF V01-05)

```
HTO      55-17    55-20#
HT1      55-27    55-29#
HT2      55-28    55-34#
HT3      55-36    55-41#
HTAB     20-17    55-13#
HTC      54-32#   54-51   140-32
HTS      54-13#   65-43   140-38
HTS1     54-14#   54-33
I1       15-15    15-17#
I2       15-24    15-26#
I3       15-33    15-35#
I5       15-42    15-44#
IBUF      6-14#   25-16   25-33   25-34   26-24   26-31   26-32   27-21   242-6   242-7
IC1     163-52   163-55#
ICMODE    9-51#   18-38   35-3    35-12   105-13*  105-31*  234-37
ICOUNT    8-8#    25-21   25-30*  26-15    26-34*   27-16
IDT      61-40   138-15#
IDTM    138-15   138-18#
IDTT     61-55   138-37#
IDTTM   138-41   138-44#
IEDMA     2-33#  169-21  172-49
IEINIT    2-32#  161-37  170-18
IEXTIO   15-16    15-49#
IFCP     16-24    17-21#
IFETCH    8-7#    26-25   26-29*  26-30    26-32*   27-22
IFP1     17-29    17-30   17-41#
IFP1A    17-43    17-44#
IFP1B    17-41    17-49#
IFP1C    17-49    17-50#
IFP2     17-50    17-55#
IFP2A    17-57    18-1    18-4    18-7#
IFP3     17-51    18-18#
IFP3A    18-18    18-19   18-21#
IFP4     18-22    18-26#
IFP5     18-21    18-23   18-26   18-28    18-35#
IFP5A    18-30    18-36#
IFP6     17-44    17-47   18-38   18-44#
IFPA     17-21    17-22   17-25#
IFPRET   18-12    18-45#
IHI       6-16#   25-27
IKB1    167-18   167-20#
IKB2    167-26   167-28#
IKB3    167-34   167-36#
IKB4    167-42   167-44#
IKB5    167-50   167-52#
IKB6    167-20   167-28  167-36  167-44  167-52  167-55#
ILEN      6-17#   25-22   25-33   26-31
ILO       6-15#   26-20
IN3     159-38#  159-49
IN3A    160-1    160-3#
IN3B    159-57   160-1#
IN4     160-11   160-14#
IN5     160-15   160-17#
INDX1     7-11#   50-15   50-19   51-24   51-32   52-27   52-51  113-20  113-22  113-24  115-44  115-49  115-51  116-15
        116-16   116-17  116-19  116-44  116-45  116-46  116-49  118-18  118-20  119-41  119-43  121-46  121-48  122-49
        122-51   127-18  127-20  129-18  129-20  130-18  130-43  132-19  132-32  132-56  153-16  153-22  159-18  159-23
        159-36   159-49  160-30  161-24  161-29  186-36  187-9   211-18  211-28  212-16  212-21  213-26  213-29  236-16
```

CROSS REFERENCE TABLE (CREF V01-05)

```
        236-23   237-16   237-25   238-18   238-36   239-15   239-19
INDX1A    7-12#  115-46
INDX2     7-13#   52-33    52-44   131-18   131-20   133-18   133-20   153-17   153-23   172-18   172-30   173-22   186-38   187-7
        212-17   212-20   236-17   236-22   237-17   237-24   238-19   238-35
INDX2A    7-14#
INIT    159-14#  161-20   162-27
INITCR  161-33  163-14#
INITKB  157-13   161-44   167-14#  182-29   199-37   199-44
INMOV   159-20   159-25   160-25#  160-30
IPINIT    2-31#  161-31
IRQ     156-41  175-18#
IRQ1    175-34  175-36#
IRQ2    175-33   175-35   175-36   175-41#
IRQ3    175-43  175-46#
IRQ4    175-45  175-47#
IRQ4A   175-51   175-54   176-1#
IRQ5    175-44   175-46   175-50   176-4    176-8    176-11#
ISCON     2-30#  159-53
ISTORE    8-6#    25-17    25-31*   25-32    25-34*   159-16
ITMOD     2-29#  159-52
ITMR0    15-25    15-50#
KBACC    38-35#   39-18
KBBREA   39-24   153-15#  153-21
KBC0    236-17#  236-23
KBC1    236-18#  236-20   236-22
KBC2    236-19   236-22#
KBCHK   161-18   236-16#
KBCP     16-43    30-19    33-13#
KBCPLK   39-27   152-14#
KBCRT    33-48    40-13#   40-36
KBCRTL   33-47    40-36#
KBC_BE    4-31#   56-16
KBC_DC    4-33#  100-35   167-38
KBC_DK    4-38#  151-33   167-30   167-33
KBC_DR    4-37#  167-46
KBC_EC    4-32#  100-15   167-38   167-41
KBC_EK    4-39#  151-14   167-30
KBC_EL    4-34#  152-15   167-14   167-17
KBC_ER    4-36#  167-46   167-49
KBC_ID    4-30#  167-54
KBC_OF    4-40#  167-22   167-25
KBC_ON    4-41#  167-22
KBC_XL    4-35#  152-17   167-14
KBDISF    9-45#   16-40   151-13*  151-32*  167-34
KBDT     33-29    39-9#    39-38
KBDTL    33-28    39-38#
KBE1     38-14    38-17#
KBENT    38-14#   39-36
KBFNT    35-54    43-11#   43-40
KBFNTL   35-53    43-40#
KBIDNU    8-30#   35-42*
KBIN     11-38#   30-15    30-27    30-36    30-47    31-17    31-30    32-15
KBOUT    11-37#   30-24*   30-29*   30-45*   30-53*   31-27*   31-31*
KBPDT    34-25    41-16#   42-45
KBPDTL   34-24    42-45#
KBS1     37-45    37-47#
KBSCR1  154-16   154-19#
```

CROSS REFERENCE TABLE (CREF V01-05)

```
KBSCR2  154-14   154-21#
KBSCRO   39-30   154-14#
KBSPAC   37-44#   39-15
KBTAB    37-15#   39-12
KB_0      3-30#   41-18    42-3
KB_1      3-31#   41-21    42-6
KB_2      3-32#   41-24    42-9
KB_3      3-33#   41-27    42-12
KB_4      3-34#   41-30    42-15
KB_5      3-35#   41-33    42-18
KB_6      3-36#   41-36    42-21
KB_7      3-37#   34-54    41-39    42-24
KB_8      3-38#   41-42    42-27
KB_9      3-39#   41-45    42-30
KB_ACC    4-7#    39-17   181-17
KB_BRE    3-23#   39-23   153-20
KB_CAP    3-24#   39-26
KB_CNT    3-11#
KB_COM    3-43#   41-57    42-42
KB_DAS    3-42#   41-54    42-39
KB_DEC    3-40#   41-48    42-33
KB_DOW    3-15#   40-18   186-13
KB_ENT    3-41#   38-15    39-35    41-51    42-36
KB_ERA    3-19#   40-33
KB_F1     4-9#    43-13
KB_F2     4-10#   43-16
KB_F3     4-11#   43-19
KB_F4     4-12#   43-22
KB_F5     4-13#   43-25
KB_F6     4-14#   43-28
KB_F7     4-15#   43-31
KB_F8     4-16#   43-34
KB_F9     4-17#   43-37
KB_HEL    3-20#   40-30
KB_HOM    3-18#   40-27
KB_ID     4-19#   35-39
KB_LEF    3-16#   40-21   186-4
KB_POW    3-28#   39-20   236-20
KB_RIG    3-17#   40-24   186-7
KB_SCR    3-21#   39-29
KB_SET    3-22#   39-32   183-35   183-38   208-28   208-31
KB_SHF    3-10#   34-54    42-3     42-6     42-9     42-12    42-15    42-18    42-21    42-24    42-27    42-30    42-33    42-36
          42-39   42-42   183-38   208-31
KB_SPA    3-26#   39-14   208-21
KB_TAB    3-25#   39-11
KB_UP     3-14#   40-15   186-10
KCi     152-16   152-18#
KP0       33-13    33-17#
KP1       33-22    33-28#
KP10      35-25    35-29    35-31#
KP11      34-43    34-45#
KP12      34-54    35-19#
KP13      35-1     35-7#
KP14      35-3     35-5#
KP15      35-12    35-14#
KP16      34-27    34-30#
KP16A     34-33#   34-37    34-38    34-52    34-53
```

CROSS REFERENCE TABLE (CREF V01-05)

CROSS REFERENCE TABLE (CREF V01-05)

CROSS REFERENCE TABLE (CREF V01-05)

CROSS REFERENCE TABLE (CREF V01-05)

```
PDL2    128-27   128-32#
PDLA    128-20   128-21#
PFIELD   11-21#    23-14    55-22*   55-29*   55-34*   92-36*  112-14   112-15*  112-17*  113-44*  113-46*  168-39*  172-25   173-1
        226-25   226-26*  226-29*  230-21   230-22*  230-41*
PI1      25-22    25-26#
PI2      25-27    25-30#
PI3      25-33    25-36#   26-35    29-32
PIC      18-41   130-15#  131-19
PIC1    130-27#  130-43
PIL      61-26    63-44   126-18#  127-19
PILO    126-27#  126-34
PIL1    126-25   126-30#
PIL2    126-30   126-36#
PILA    126-22   126-23#
PILRET  126-28   126-40#  128-25   128-36
PILRT1  126-23   126-32   126-41#  128-21   128-28
PIR      25-24    25-37#
PK0      30-18    30-20#
PK1      30-15    30-23#
PK1A     30-27    30-32#
PK2      30-36#   30-42
PK3      30-36    30-45#
PK4      30-47#   30-48
PK5      30-47    30-51#
PKDLY    30-37    30-38    30-39    30-40    30-41    30-54#
PKL      30-34#   30-51
PKR      30-43    30-49    30-53#
PLF      22-34    47-15    47-34    47-36    49-16#   65-40   128-35
PLF1     49-17    49-19#
PLF2     49-20    49-22#
PLF3     49-21    49-24#
PLFCR    19-56    47-34#
PLFLP    49-38#   49-42
PLFRET   49-22    49-47#
PMADRH    8-25#   83-20*   83-43    83-53*   85-18*   85-21    85-29*
PMADRL    8-26#   83-19*   83-42    83-52*   85-17*   85-20    85-28*
PMBUF     6-22#   83-19    83-20    83-49    85-17    85-18
PMNUM     8-27#   83-21*   83-36    83-38*   83-50*   83-54*   84-1     84-3*    85-22    85-30*
PMVALU    8-28#   83-23*   83-31*   83-34    83-35*   83-45    83-57*
PO1      28-15    28-18#
PO2      28-18    28-21#
PO3      28-23    28-28#
PORTF     9-7#    16-28   160-6*   177-27   194-45   231-14*
PRINT   158-16   230-15#
PRLF     46-16#   61-23    65-34   126-38
PRNT1   230-28#  230-36
PRNT2   230-45   230-48#
PRNT3   230-51   230-54#
PRNTF     9-53#  174-30   174-33   174-39   230-15*  230-48*
PROTEC    9-21#
PRTF2    10-31#  160-5    231-15*
PRTYEN   10-3#   177-17   188-38   188-39*  188-47*  190-13
PRTYEV   10-5#   177-21   188-41*  188-44   188-45*  190-17
PRTYST   10-4#   177-19   188-40*  188-43   188-49*  190-15
PSD      66-18    73-17    74-17    83-17#
PSD1     83-37    83-39#
PSD2     83-49    83-52#
```

CROSS REFERENCE TABLE (CREF V01-05)

```
PSD3     83-51    83-55#
PSD4     84-2     84-4#
PSDB     83-17    83-25#
PSDB1    83-25    83-26#
PSDB2    83-27    83-28#
PSDE     83-26    83-28    83-42#
PSDF      9-38#   58-14*   73-14*   74-14*   83-17    83-18*
PSDRET   83-56    84-6#
PSDF     37-20   136-14#  136-23   137-25   138-16   138-42   139-18
PTR1      7-7#    44-17    44-18    45-18    45-19    46-24    46-27    46-34    46-35    46-39    49-28    49-31    49-39    49-40
         49-44    50-13    50-17    50-18    51-22    51-23    51-33    52-24    52-30    52-50   130-24   130-36   130-39   132-24
        132-48   132-54   159-16   159-22   159-34   159-46   159-48   160-27   160-29   161-23   161-26   161-27   186-35   186-37
        187-8    211-16   211-20   211-27   217-19   217-25   217-26   217-30   217-31   239-14   239-17   239-18
PTR1A     7-8#
PTR2      7-9#    46-25    46-26    46-33    46-36    49-29    49-30    49-38    49-41   130-25   130-33   130-40   132-29   132-51
        132-55   185-14   185-25   185-26   185-27   185-33   185-34   185-35   185-48
PTR2A     7-10#  186-49   187-1
PU1      22-31    22-37#
PU2      22-32    22-35    22-42#
PUTCHR   18-44    22-14#
R2_BAC   13-30#   16-19   157-24
R2_DCL   14-34#   16-47   158-21
R2_DIS   14-41#   99-44   158-27
R2_ESC   13-48#   57-14   157-42
R2_FIL   12-35#  112-39   114-20   115-21   116-53   156-34
R2_ICU   13-14#  101-33   157-6
R2_IKB   13-19#   39-21   157-12
R2_IN    12-11#   16-8    156-10
R2_IN2   12-17#   96-19   156-16
R2_IR0   12-40#   15-34   156-40
R2_KY1   13-36#   31-15   157-30
R2_KY2   13-42#   30-13   157-36
R2_PRN   14-29#   38-17    61-46    95-17   158-15
R2_SER   12-46#   15-43   156-46
R2_SET   13-24#   39-33   157-18
R2_STA   12-23#   15-49   156-22
R2_STO   12-29#   15-50   156-28
R2_T25   14-22#   61-43   158-9
R2_TC    14-4#    61-45    68-17   157-48
R2_TL    14-10#   61-44   157-54
R2_TP    14-16#   60-17   158-3
R2_XMT   13-4#    16-38   156-52
R2_XON   13-9#    26-21   156-57
RAM0    238-18#  238-40
RAM1    238-21#  238-35   238-36
RAM2    238-38   238-42#
RAMCHK  161-35   207-19   238-15#
RAMCON  159-17   242-6#
RAMERR  238-30   238-33   238-44#
RAMP     62-15    96-19#
REPF     10-19#  167-50   199-36*  201-2
RGL      56-19#   56-23
RGR      56-20    56-21    56-24#
RING     20-5     56-16#  207-29   228-29   230-53   240-27
RL1      46-17    46-45#
RL1A     46-45    46-47#
RLL      46-33#   46-37
```

RLR	46-42#	46-48													
RMC1	237-19#	237-24	237-25												
RMC2	237-27	237-29#													
RMS	62-14	80-14#													
RMS1	80-20	80-21#													
RMSRET	80-19	80-21	80-27#	81-14	81-18										
RMST	80-24	81-16	81-30#	82-2											
RMSTL	80-23	81-15	82-2#												
ROMCHK	161-34	207-18	237-14#												
RT1	98-32#	99-15													
RTS	11-42#	178-18*	178-45*												
RUBOUT	2-13#	17-51	214-36	220-46											
RVVA	9-15#	17-42*	17-45*	143-23*	145-13*	145-31*	222-8								
SAV2	211-20#	211-28													
SB1	150-17	150-20#													
SB2	150-21	150-26#													
SB3	150-24	150-28#													
SBC	78-43	101-13#													
SBLR	68-29	150-16#													
SBR	95-40	134-22	134-42#	188-32											
SBRO	134-48	134-49#													
SBRET	134-18	134-20	134-44	134-49	134-51#										
SBRT	150-22	150-33#													
SC1	164-30	164-32#													
SC2	164-38	164-40#													
SCH	19-34	61-22	63-14	99-42	113-27	117-13#	125-37								
SCLRL	210-18	213-19#	234-21	240-30											
SCLRLR	182-16	196-24	196-37	206-24	207-21	213-17#									
SCP	61-56	65-22	141-13#	141-33											
SCRN1	5-4#	164-26													
SCRN2A	5-5#	164-29													
SCRN2B	5-6#	164-31													
SCRN3	5-7#	164-34													
SCRN4A	5-8#	164-37													
SCRN4B	5-9#	164-39													
SCRNSA	10-30#	175-51	197-20*	198-21											
SDI	169-25	169-26	169-29#												
SERIAL	156-47	174-13#													
SERIN	174-18	174-22#													
SEROUT	174-19	174-45#													
SETCF	157-7	162-39	165-14#	202-52	203-2										
SETCRT	163-23	164-13#	165-14												
SETDIS	66-16	69-17#	73-15	74-15	77-14	80-14	95-13	124-13	134-13	138-37	140-13	143-14	147-26	147-35	
SETLCK	11-40#	182-13													
SETNOR	17-23	58-19	58-38	60-14	66-20	69-37#	73-19	74-19	77-16	80-16	95-15	124-38	134-15	138-39	
	140-15	143-16	147-28	147-37											
SETTAB	51-15#	54-15	185-49												
SETUP	157-19	182-13#													
SETUP0	182-13	182-16#													
SETUP1	182-18#	182-30	182-42	182-45	182-48										
SETUP2	182-51	182-54	185-14#												
SETUP3	182-57	183-3	188-14#	188-33	188-42	188-46	188-48	188-50	188-56	189-5					
SETUP4	183-6	183-9	193-14#	193-25	193-42	193-49	193-51	193-57							
SETUP5	183-12	183-15	196-14#	196-33	196-35	196-46	197-3	197-9	197-15	197-21					
SETUP6	183-18	183-21	199-14#	199-25	199-31	199-38	199-45								
SETUP7	183-24	183-27	202-14#	202-25	202-31	202-53	203-3								
SETUP8	183-30	183-33	206-14#	207-4	207-12	207-28	208-26								

CROSS REFERENCE TABLE (CREF V01-05)

CROSS REFERENCE TABLE (CREF V01-05)

STP2C1	185-26	185-28#			
STP2D	185-33#	186-8			
STP2D1	185-34	185-36#			
STP2E	185-41#	186-11			
STP2F	185-47#	186-14			
STP2F1	185-42	185-48#			
STP3A	188-15#	188-19			
STP3B	188-24#	189-17			
STP3B1	188-28	188-30	188-32#		
STP3C	188-38#	189-20			
STP3C1	188-38	188-43#			
STP3C2	188-43	188-47#			
STP3C3	188-44	188-49#			
STP3D	188-55#	189-23			
STP3E	189-4#	189-26			
STP4A	193-15#	193-19			
STP4B	193-24#	194-12			
STP4C	193-30#	194-15			
STP4C1	193-35	193-37#			
STP4C2	193-40	193-42#			
STP4D	193-47#	194-18			
STP4D1	193-47	193-50#			
STP4E	193-56#	194-21			
STP5A	196-15#	196-19			
STP5B	196-24#	197-33			
STP5B1	196-32	196-34#			
STP5B2	196-34	196-35#			
STP5B3	196-45	196-46#			
STP5C	197-8#	197-36			
STP5D	197-14#	197-39			
STP5E	197-20#	197-42			
STP6A	199-15#	199-19			
STP6B	199-24#	199-57			
STP6C	199-30#	200-3			
STP6D	199-36#	200-6			
STP6E	199-43#	200-9			
STP7A	202-15#	202-19			
STP7B	202-24#	203-15			
STP7C	202-30#	203-18			
STP7D	202-36#	203-21			
STP7D1	202-37	202-41#			
STP7D2	202-36	202-45#			
STP7D3	202-45	202-49#			
STP7D4	202-39	202-43	202-47	202-52#	
STP7E	203-1#	203-24			
STP8A	206-15#	206-19			
STP8B	206-24#	208-2			
STP8B1	206-31#	206-38	206-43		
STP8B3	206-42#	208-22			
STP8C	206-48#	208-5			
STP8D	207-9#	207-52			
STP8E	207-17#	207-39	208-8		
STP8E0	207-20#	207-35			
STP8E1	207-32#	207-37			
STRET	66-19	70-19	70-24#		
SUC	81-42	101-31#			
SUC1	101-14	101-33#			

CROSS REFERENCE TABLE (CREF V01-05)

```
SXC1    226-14   226-20   226-25#
SXMTC   157-49   226-14#
SXMTL   157-55   226-18   227-15#
SXON1   178-44   178-49#
SXR     178-14   178-41   178-47#  178-49
TAB      19-45    52-15#   53-23
TABTAB  10-48#   50-13    51-21    52-23   186-35
TB1     52-32    52-47#
TB2     52-40    52-42#
TB3     52-35    52-41#
TB4     52-37    52-39#
TB5     52-46    52-50#
TB6     52-31    52-33#
TBL1    52-30#   52-51
TBL2    52-35#   52-44
TBR     52-41    52-55#
TCNT    11-14#   162-15   162-37*  175-41*  175-42   175-47*  197-2*
TEMP     7-17#   33-42    33-45    35-19    35-37    35-40    35-51    36-16    36-19    45-15    45-22    52-16    52-36    52-39
        60-28    60-32    83-44    83-55    84-1     84-2     94-34    94-39    118-41   120-17   130-31   132-43   159-41   159-45
       170-24   170-25   170-35   173-4    173-10   173-12   186-34   186-40   186-48   186-57   187-6    193-31   193-37   238-27
TEMPA    7-18#   46-27*   46-39    49-31*   49-44    118-44   120-20   130-40   132-55   173-13   193-32*  238-30
TERMIN 168-15#   193-39
THOUR   11-17#   111-16*  162-18   162-34*  176-6*   176-7    176-9*   196-36*  232-25
TMIN    11-16#   111-20*  162-17   162-35*  176-2*   176-3    176-5*   196-47*  232-39
TRA     177-17   177-19   177-21   177-23#
TRANS   177-16#  178-23   178-51
TRANSM  156-53   177-14   177-14#  219-22   219-26   224-3    224-32
TROK    177-25   177-27#
TROK1   177-27   177-29#
TROK2   177-28   177-31#
TRW     177-25#  177-26   177-29
TSCROL   8-22#   46-17    46-29    49-24    49-27    49-33    121-17   125-21*  125-27   126-19   126-22   126-37*  126-42*  128-16
        128-20   128-34*  154-26
TSEC    11-15#   111-24*  162-16   162-36*  175-48*  175-49   176-1*   197-1*   232-21   232-24   232-51
UNDEF   61-13    61-30    61-31    61-32    61-33    61-34    61-35    61-36    61-37    61-38    61-47    61-49    61-50    61-51
        61-52    61-53    61-54    62-2     62-17    62-19    62-20    62-37#
UNDLNA   9-16#   143-29*  144-31*  222-29
USCP    61-57    65-25    142-13#  142-36
USCP1   142-15   142-17#
USCP2   142-16   142-18#
UXOFFS   9-28#   16-35*   16-37*   154-16   178-49
VSP     11-39#   169-27*  170-52*  175-56*
VSPF     9-54#   31-22*   161-42*  169-26   174-23*  175-55*
WEOL    62-11    75-41    97-15#
WLATCH 159-30   159-32   160-7    175-39   211-31   211-33   231-36#
WORK     7-15#   21-19    21-33    21-41    23-19    23-29    23-31    30-35    30-42    30-46    30-48    35-20    35-23    35-28
        46-31    46-37    49-35    49-42    51-17    51-31    52-18    52-43    52-48    52-49    52-53    52-55    113-19   113-25
       130-28   132-46   214-20   214-29   215-8    215-11   215-12   225-23   225-24   225-26   225-28   228-20   228-21   230-35
       230-36   237-14   237-21   237-22   238-16   238-21   238-28   238-38   238-39
WORK1    7-19#   186-39   186-52   186-54   214-17   214-42   214-45   215-17   215-19
WORK1A   7-20#
WORK2    7-21#   172-19   172-21   173-16   186-46   186-50   186-56   187-2    187-3    196-32   214-19   214-40   214-43   214-53
        215-20   240-44   240-47   241-1    241-4    241-9    241-12   241-17   241-20   241-25   241-28
WORK2A   7-22#
WORKA    7-16#   21-24    23-32    51-26    52-37    130-39   132-54   238-33
XACR    72-45    81-57    107-32#
XALF    81-54    108-32#
```

CROSS REFERENCE TABLE (CREF V01-05)

```
XATR     63-53    91-14   143-49#
XBLINA   86-8#    87-32   117-16   117-43   118-48   119-19   120-24   121-24   122-27   124-36   124-53   126-44   142-18
XERM     72-42   110-34#
XGATM    72-33   109-34#
XGM      61-21    91-32   146-31#
XHSM     61-42    81-39   102-32#  193-48
XHSM1   102-15   102-34#
XICM     61-29    72-39   105-31#
XKAM     61-12    65-31    81-51   104-31#
XKSM     62-10    81-48   103-31#
XL0     220-21   220-23   220-28#
XL0A    220-44   220-45#
XL1     220-45   220-53#
XL10    222-29   222-35#
XL10A   222-33   222-37#
XL11    222-31   222-35   222-41#
XL12    222-41   222-47#
XL12A   222-45   222-49#
XL13    222-43   222-47   222-53#
XL14    222-53   223-2#
XL14A   222-57   223-4#
XL15    222-55   223-2    223-8#
XL16    223-8    223-22#
XL16A   223-26   223-31#
XL16B   223-13   223-25   223-35#
XL17    223-10   223-17   223-20   223-22   223-29   223-33   223-39#
XL18    223-41   224-9#
XL18B   224-12   224-16#
XL18C   224-16   224-20#
XL18D   224-20   224-24#
XL18E   224-24   224-28#
XL1A    220-53   220-56#
XL2     220-51   220-55   221-1#
XL20    223-39   224-5    224-31#
XL3     220-46   221-5#
XL3A    221-16   221-21#  221-25
XL3B    221-15   221-25#
XL3C    221-14   221-31#
XL4     221-9    221-35#
XL4A    221-40   221-45#  221-50
XL4AA   221-41#  221-54
XL4B    221-39   221-50#
XL4C    221-38   222-1#
XL4SET  221-23   221-47#
XL7     221-11   221-19   221-29   221-33   221-35   221-43   221-48   222-6#
XL8     222-8    222-17#
XL8A    222-12   222-20   222-25#
XL9     222-9    222-15   222-17   222-23   222-29#
XLATCH   9-3#   150-29   231-36
XLON1   218-16   218-18#
XLONOF  195-15   195-21   198-10   198-16   198-22   200-34   200-40   201-3    201-9    203-49   204-5    218-15#
XML2    225-21#  225-24
XMP1    228-18#  228-21
XMP2    228-23#  229-19
XMP3    227-16   228-26#  229-15
XMT25   158-10   226-22   229-15#
XMTC    220-18#  225-21   226-28
```

CROSS REFERENCE TABLE (CREF V01-05)

CROSS REFERENCE TABLE (CREF V01-05)

$$INIT 0-0#