

Z-29
TERMINAL
ROM SOURCE LISTING

Z-29 Terminal

ROM Source Listing

This document contains the source listing of the code contained in the Z-29 Terminal's ROM.

Reproduction of this code (software) in any form, except with written consent from Zenith Data Systems Corporation, is expressly prohibited.

NOTICE

Zenith Data Systems Corporation makes no express or implied warranties with regard to this software, including merchantability, fitness for any purpose, or non-infringement of patents, copyrights, or other proprietary rights of others. Zenith Data Systems Corporation shall not have any liability or responsibility to sublicensees, including end-users, for damages of any kind, including special, indirect, or consequential damages, arising out of or resulting from any program, services, or materials made available hereunder or the use or modification thereof.

The information in this publication is considered to be correct and up to date when it is first published. Zenith Data Systems Corporation reserves the right to change the product at any time without incurring any obligation to incorporate new features in products already sold. These changes may or may not be reflected in this or future editions of this documentation.

595-3053-01
Copyright 1983
Zenith Data Systems Corporation
All Rights Reserved
First Printing
Printed in the
United States of America

ZENITH | data
systems

```
1          ;;; Z-29 Terminal Firmware Version 1.03
2          ;
3          ;
4          ; Written by R. Nick K. Boland February 10, 1983
5          ;
6          ; Copyright (C) 1983
7          ; Zenith Data Systems
8          ; Saint Joseph, Michigan
9          ;
10         ;
11         ; Initial consultation by Tom Blog
12         ; Initial hardware by Michael Gorbutt
13         ; Hardware finished by Anthony M. Olson
14         ;
15         ;
16         ; Many things have been used to compact the program. Look
17         ; for these situations. The most common are where subroutines
18         ; use parts of other subroutines. Other instances are where
19         ; a single LJMP could be used and many AJMP's used to reference
20         ; the long jump.
21         ;
22         ;
23         ; ENABL LC
24         ; NLIST TOC
25         ; NAME Z29ROM
26         ; SBTTL *** Z-29 Computer Terminal Firmware Version 1.03 ***
```

```

1      ; ASCII character equivalences
2      ;
3      0000      NULL      EQU      00000000B      ; Null
4      0007      BELL      EQU      00000111B      ; Bell
5      0008      BS        EQU      00001000B      ; Backspace
6      0009      HT        EQU      00001001B      ; Horizontal tab
7      000A      LF        EQU      00001010B      ; Line feed
8      000D      CR        EQU      00001101B      ; Carriage return
9      0011      XON       EQU      00010001B      ; DC1 (XON)
10     0013      XOFF      EQU      00010011B      ; DC3 (XOFF)
11     0018      CAN       EQU      00011000B      ; Cancel
12     001B      ESC       EQU      00011011B      ; Escape
13     007F      RUBOUT    EQU      01111111B      ; Rubout / delete
14
15
16     ; Constants
17     ;
18     0018      MAXLINE   EQU      24              ; Max number of lines is 0..24
19     0019      NUMLINES  EQU      25              ; Number of lines is 1..25
20     004F      MAXCHAR   EQU      79              ; Max number of chars/line 0..79
21     0050      NUMCHARS  EQU      80              ; Number of characters is 1..80
22
23
24
25     ;; 8051 processor constants
26     ;
27     0000      BANK0     EQU      00000000B      ; Select register bank 0
28     0008      BANK1     EQU      00001000B      ; Bank 1
29     0023      ITMOD     EQU      00100011B      ; Init value for -TMOD-
30     005A      ISCON     EQU      01011010B      ; Init value for -SCON-
31     0003      IPINIT    EQU      00000011B      ; Init value for interrupt priorities
32     0095      IEINIT    EQU      10010101B      ; Init value for interrupts
33     0082      IEDMA     EQU      10000010B      ; Init value for interrupts during DMA

```

```

1          ; ; Keyboard data
2          ; ;
3          ; ; Data from the keyboard other than ASCII data is in the format
4          ; ; of 'iscXXXXXB' where 's' = shift status, 'c' = control status and
5          ; ; the rest of the word defines the function. Exceptions to this
6          ; ; are the access function, power up function and ID number.
7          ; ;
8          ; ; OaaaaaaaaB ASCII data to be transmitted
9          ; ;
10         0040 KB_SHFT EQU 01000000B ; Keyboard shift mask
11         0020 KB_CNTR EQU 00100000B ; Keyboard control mask
12
13
14         0080 KB_UP EQU 10000000B ; Up arrow key
15         0081 KB_DOWN EQU 10000001B ; Down arrow key
16         0082 KB_LEFT EQU 10000010B ; Left arrow key
17         0083 KB_RIGHT EQU 10000011B ; Right arrow key
18         0084 KB_HOME EQU 10000100B ; Home key
19         0085 KB_ERASE EQU 10000101B ; Erase key
20         0086 KB_HELP EQU 10000110B ; Help key
21         0087 KB_SCROLL EQU 10000111B ; Scroll key
22         0088 KB_SETUP EQU 10001000B ; Setup key
23         0089 KB_BREAK EQU 10001001B ; Break key
24         008A KB_CAPS EQU 10001010B ; Caps lock key
25         008B KB_TAB EQU 10001011B ; Tab key
26         008C KB_SPACE EQU 10001100B ; Space bar
27
28         008F KB_POWER EQU 10001111B ; Keyboard just powered up
29
30         0090 KB_0 EQU 10010000B ; Numeric keypad 0 key
31         0091 KB_1 EQU 10010001B ; Numeric keypad 1 key
32         0092 KB_2 EQU 10010010B ; Numeric keypad 2 key
33         0093 KB_3 EQU 10010011B ; Numeric keypad 3 key
34         0094 KB_4 EQU 10010100B ; Numeric keypad 4 key
35         0095 KB_5 EQU 10010101B ; Numeric keypad 5 key
36         0096 KB_6 EQU 10010110B ; Numeric keypad 6 key
37         0097 KB_7 EQU 10010111B ; Numeric keypad 7 key
38         0098 KB_8 EQU 10011000B ; Numeric keypad 8 key
39         0099 KB_9 EQU 10011001B ; Numeric keypad 9 key
40         009A KB_DEC EQU 10011010B ; Numeric keypad decimal point key
41         009B KB_ENT EQU 10011011B ; Numeric keypad enter key
42         009C KB_DASH EQU 10011100B ; Numeric keypad dash key
43         009D KB_COMMA EQU 10011101B ; Numeric keypad comma key

```

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

```

1          ; Access code and keys that follow
2          ;
3          ;
4          ; The access code from the keyboard is very much like an escape
5          ; character. It indicates that the second character from the
6          ; keyboard will be the data with the following definitions.
7          009F          KB_ACC          EQU    10011111B      ; Access code
8          ;
9          0080          KB_F1           EQU    10000000B      ; Function key #1
10         0081          KB_F2           EQU    10000001B      ; Function key #2
11         0082          KB_F3           EQU    10000010B      ; Function key #3
12         0083          KB_F4           EQU    10000011B      ; Function key #4
13         0084          KB_F5           EQU    10000100B      ; Function key #5
14         0085          KB_F6           EQU    10000101B      ; Function key #6
15         0086          KB_F7           EQU    10000110B      ; Function key #7
16         0087          KB_F8           EQU    10000111B      ; Function key #8
17         0088          KB_F9           EQU    10001000B      ; Function key #9
18         ;
19         0098          KB_ID            EQU    10011000B      ; ID number 10011xxx8
20         ;
21         ;
22         ;
23         ;
24         ; Keyboard commands
25         ;
26         ; These are commands that are sent from the terminal to the
27         ; keyboard. They tell the keyboard to enter and exit different
28         ; modes and to perform different functions.
29         ;
30         0080          KBC_ID           EQU    10000000B      ; Send keyboard ID number
31         0081          KBC_BELL         EQU    10000001B      ; Ring bell
32         0082          KBC_EC           EQU    10000010B      ; Enable key click
33         0083          KBC_DC           EQU    10000011B      ; Disable key click
34         0084          KBC_EL           EQU    10000100B      ; Enter caps locked
35         0085          KBC_XL           EQU    10000101B      ; Exit caps locked
36         0086          KBC_ER           EQU    10000110B      ; Enable auto repeat mode
37         0087          KBC_DR           EQU    10000111B      ; Disable auto repeat mode
38         0088          KBC_DK           EQU    10001000B      ; Disable keyboard
39         0089          KBC_EK           EQU    10001001B      ; Enable keyboard
40         008A          KBC_OFFLN        EQU    10001010B      ; Turn on offline LED
41         008B          KBC_ONLN        EQU    10001011B      ; Turn off offline LED

```

```
1          ;;          CRT video controller equivalences
2          ;
3          0000      CRESET      EQU      00000000B      ; Reset 8275
4          004F      SCRNI       EQU      01001111B      ; Normal rows, 80 chars/row
5          0098      SCRNI2A     EQU      10011000B      ; 3 VRTC, 25 rows/screen
6          00DC      SCRNI2B     EQU      11011100B      ; 4 VRTC, 29 rows/screen
7          0099      SCRNI3      EQU      10011001B      ; line 10 underline, 10 scan lines
8          004E      SCRNI4A     EQU      01001110B      ; attributes, cursor, 30 HRTC
9          004F      SCRNI4B     EQU      01001111B      ; attributes, cursor, 32 HRTC
10
11         0020      CSTRT       EQU      00100000B      ; Start display, 0 delay, 1/burst
12         0040      CSTOP      EQU      01000000B      ; Stop display
13         0080      CCSR       EQU      10000000B      ; Load cursor position
14         00A0      CINT       EQU      10100000B      ; Enable end of frame interrupt
15         00E0      CSET       EQU      11100000B      ; Preset counters
```

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

```

1      ;
2      ; External memory address map
3      ;
4      2000      DDATREG      EQU      8*1024      ; 8275 data register
5      2001      DCMDREG      EQU      8*1024 + 1   ; 8275 command register
6      2001      DSTAREG      EQU      8*1024 + 1   ; 8275 status register
7
8      3000      MLATCH       EQU      12*1024      ; Memory latch
9
10     4000      ATRIBMEM     EQU      16*1024      ; Attribute memory
11
12     5000      CHARMEM     EQU      20*1024      ; Character memory
13     5780      L25MEM      EQU      20*1024 + 1920 ; Line 25 memory
14
15     57D0      IBUF        EQU      20*1024 + 2000 ; Input buffer 32 bytes
16     0009      ILO         EQU      ?           ; Input queue lo water mark
17     0015      IHI         EQU      21           ; Input queue hi water mark
18     0020      ILEN        EQU      32           ; Input queue length
19
20     57F0      OBUF        EQU      20*1024 + 2032 ; Output buffer 8 bytes
21     57F7      OLEN        EQU      OBUF + 7     ; Output queue length
22
23     57F8      PMBUF       EQU      20*1024 + 2040 ; ANSI parameter buffer 8 bytes
24
25     6000      DMAMEM     EQU      24*1024      ; DMA memory
26
27     7000      EAROM       EQU      28*1024      ; EAROM memory
28
29     ; External memory constants
30     ;
31     00EF      ATRANL      EQU      11101111B    ; Mask -CHARMEM- to -ATRIBMEM-
32     00CF      DMAANL      EQU      11001111B    ; Mask -DMAMEM- to -ATRIBMEM-
33     0010      CHRORL      EQU      00010000B    ; Mask -ATRIBMEM- to -CHARMEM-
34     0020      DMAORL      EQU      00100000B    ; Mask -ATRIBMEM- to -DMAMEM-

```



```
1          ;;          Internal RAM memory setup
2          ;
3
4
5          ;          Register bank #0 used in background jobs
6          ;
7          REG        PTR1      EQU    R0          ; Work pointer #1          (R0)
8          0000       PTR1A     EQU    00H        ; Address of -PTR1-
9          REG        PTR2      EQU    R1          ; Work pointer #2          (R1)
10         0001       PTR2A     EQU    01H        ; Address of -PTR2-
11         REG        INDX1     EQU    R2          ; Index register 1        (R2)
12         0002       INDX1A    EQU    02H        ; Address of -INDX1-
13         REG        INDX2     EQU    R3          ; Index register 2        (R3)
14         0003       INDX2A    EQU    03H        ; Address of -INDX2-
15         REG        WORK      EQU    R4          ; Work register           (R4)
16         0004       WORKA     EQU    04H        ; Address of -WORK-
17         REG        TEMP      EQU    R5          ; Temporary holding register (R5)
18         0005       TEMPA     EQU    05H        ; Address of -TEMP-
19         REG        WORK1     EQU    R6          ; Work register 1         (R6)
20         0006       WORK1A    EQU    06H        ; Address of -WORK1-
21         REG        WORK2     EQU    R7          ; Work register 2         (R7)
22         0007       WORK2A    EQU    07H        ; Address of -WORK2-
23
24
25         ;          Register bank #1 used in DMA routine
26         ;
27         REG        DMPTR     EQU    R0          ; Pointer into line order  (R0)
28         0008       DMPTRA    EQU    08H        ; Address of -DMPTR-
29         REG        DMLN      EQU    R1          ; Current DMA line number  (R1)
30         0009       DMLNA     EQU    09H        ; Address of -DMLN-
```



```

1          ;                               Flags that define byte 25H
2          ;
3          0028      PRTYENA      BIT      28H      ; 0 = no parity      1 = ok parity
4          0029      PRTYSTK      BIT      29H      ; 0 = normal parity  1 = stick
5          002A      PRTYEVN      BIT      2AH      ; 0 = odd parity     1 = even parity
6          002B      AUTOWRAP     BIT      2BH      ; 0 = no auto wrap   1 = wrap around
7          002C      AUTOCR      BIT      2CH      ; 0 = no auto CR on LF 1 = auto CR
8          002D      AUTOLF      BIT      2DH      ; 0 = no auto LF on CR 1 = auto LF
9          002E      ONLINE      BIT      2EH      ; 0 = off line      1 = on line
10         002F      FULLDPLX     BIT      2FH      ; 0 = half duplex   1 = full duplex
11
12
13         ;                               Flags that define byte 26H
14         ;
15         0030      KPADSHTF     BIT      30H      ; 0 = keypad ordinary 1 = shifted
16         0031      KPADALTF     BIT      31H      ; 0 = keypad ordinary 1 = alternate
17         0032      CRUL        BIT      32H      ; 0 = block cursor   1 = underline
18         0033      CLKF        BIT      33H      ; 0 = no key click   1 = click
19         0034      REPF        BIT      34H      ; 0 = no auto repeat  1 = auto repeat
20         0035      CAPLOCKF     BIT      35H      ; 0 = caps unlocked  1 = locked
21         0036      HSMODEF     BIT      36H      ; 0 = not hold screen 1 = hold screen
22         0037      L25ON       BIT      37H      ; 0 = status line off 1 = status on
23
24
25         ;                               Flags that define byte 27H
26         ;
27         0038      CRNEK       BIT      38H      ; 0 = blinking cursor 1 = non blink
28         0039      FREQ        BIT      39H      ; 0 = 50 Hz          1 = 60 Hz
29         003A      MONITOR     BIT      3AH      ; 0 = not monitor mode 1 = monitor
30         003B      SCRNSAVE    BIT      3BH      ; 0 = no screen saver 1 = saver on
31         003C      PRTF2      BIT      3CH      ; 0 = main port      1 = auxillary
32         003D      ACHR2      BIT      3DH      ; 0 = normal char set 1 = alternate
33         003E      HNDSHK     BIT      3EH      ; 0 = software       1 = hardware
34
35
36         ;                               Flags that define byte 28H
37         ;
38         0028      MODE        EQU      28H      ; Mode terminal is in
39
40         0040      ANSI        BIT      40H      ; 0 = other modes    1 = ANSI mode
41         0041      ZDS        BIT      41H      ; 0 = other modes    1 = ZDS mode
42         0042      ADM3       BIT      42H      ; 0 = other modes    1 = ADM 3 mode
43         0043      HAZ        BIT      43H      ; 0 = other modes    1 = Haz 1500
44
45
46         ;                               Variable tab table
47         ;
48         0029      TABTAB     EQU      29H      ; Variable tab table 10 bytes to 32H

```

```

1          ;          General use variables from 33H to 46H
2          ;
3          0033      BAUDRATE      EQU    33H          ; Serial port baud rate
4          0034      NVRSUM        EQU    34H          ; Non-volatile RAM checksum
5
6          0035      XXPOS         EQU    35H          ; Saved X-position
7          0036      XYPOS         EQU    36H          ; Saved Y-position
8
9          0037      BLNKRATE      EQU    37H          ; Blinking rate of screen
10         0038      BLNKCNT       EQU    38H          ; Running count for blink rate
11
12         0039      AUTOCNT       EQU    39H          ; Auto off timer count
13
14         003A      TCNT          EQU    3AH          ; Running clock count
15         003B      TSEC          EQU    3BH          ; Seconds counter
16         003C      TMIN          EQU    3CH          ; Minute counter
17         003D      THOUR         EQU    3DH          ; Hour counter
18
19         003E      DISPSEC       EQU    3EH          ; Last count of seconds displayed
20
21         003F      PFIELD        EQU    3FH          ; Protected fields byte
22         0040      ERRORS        EQU    40H          ; Powerup diagnostic error class
23
24
25         ;          Screen line table
26         ;
27         0047      LORDER        EQU    47H          ; Line order list      25 bytes to 5FH
28
29
30         ;          Stack area
31         ;
32         005F      STACK         EQU    5FH          ; Stack pointer address 32 bytes to 7FH
33
34
35         ;          8051 Processor bit port defines
36         ;
37         0090      KBOUT         BIT    P1.0         ; Keyboard output line
38         0091      KBIN         BIT    P1.1         ; Keyboard input line
39         0092      VSP          BIT    P1.2         ; 0 = normal video      1 = suppressed
40         0093      SETLCK       BIT    P1.3         ; 0 = setup mode locked 1 = normal
41         0094      DTR          BIT    P1.4         ; Data terminal ready
42         0095      RTS          BIT    P1.5         ; Ready to send
43         ;          BIT    P1.6         ; Expansion socket
44         0097      CLKRUN       BIT    P1.7         ; Clock run for CRT controller
45
46         ;TXD          BIT    P3.1         ; Predefined serial port transmit pin
47         00B4      DMATYPE      BIT    P3.4         ; 0 = clear memory     1 = DMA CRTC
48         00B5      CTS         BIT    P3.5         ; Clear to send

```

```
1 .....; .....ROM.#2.VECTORS.....
2 .....; .....
3 .....; .....These.locations.are.provided.to.facilitate.future.modifications.....
4 .....; .....without.a.change.to.the.masked.processor.Each.location.is.....
5 .....; .....accessed.with.a.-CALL- and.control.is.gained.back.with.a.-RET-.....
6 .....; .....
7 .....; .....
8 .....; .....Initialization.of.internal.RAM.and.any.other.devices.that.....
9 .....; .....may.be.added,.....
10 .....; .....
11 .....1000.....R2_IN.....EQU.....1000H.....
12 .....; .....
13 .....; .....Same.as.R2_IN.but.does.not.destroy.current.settings.....
14 .....; .....such.as.CAPS.LOCK,ON/OFF.LINE,TIME.on.clock,etc.....
15 .....; .....
16 .....; .....
17 .....1003.....R2_IN2.....EQU.....1003H.....
18 .....; .....
19 .....; .....
20 .....; .....Start.DMA.hook.This.is.jumped.to.at.the.besinnins.of.each.....
21 .....; .....DMA.request.from.the.CRTC.....
22 .....; .....
23 .....1006.....R2_STARTDMA.....EQU.....1006H.....
24 .....; .....
25 .....; .....
26 .....; .....Stop.DMA.hook.This.is.jumped.to.at.the.besinnins.of.each.....
27 .....; .....timer.0.interrupt.....
28 .....; .....
29 .....1009.....R2_STOPDMA.....EQU.....1009H.....
30 .....; .....
31 .....; .....
32 .....; .....Fill.screen.hook.This.is.jumped.to.start.the.fill.screen.....
33 .....; .....DMA.feature.A.check.for.protected.fields.is.made.....
34 .....; .....
35 .....100C.....R2_FILL.....EQU.....100CH.....
36 .....; .....
37 .....; .....
38 .....; .....IRQ.hook.This.is.jumped.to.at.the.besinnins.of.each.IRQ.....
39 .....; .....
40 .....100F.....R2_IRQ.....EQU.....100FH.....
41 .....; .....
42 .....; .....
43 .....; .....Serial.hook.This.is.jumped.to.at.the.besinnins.of.each.....
44 .....; .....serial.interrupt.....
45 .....; .....
46 .....1012.....R2_SERIAL.....EQU.....1012H.....
```

```
1      ; Transmit hook. This performs the transmit character
2      ; function of the terminal for the serial port.
3
4      1015      R2_XMT      EQU      1015H
5
6
7      ; Send an XON to the host.
8      ;
9      1018      R2_XON      EQU      1018H
10
11
12     ; Initialize the cursor format.
13     ;
14     101B      R2_ICUR     EQU      101BH
15
16
17     ; Initialize the keyboard
18     ;
19     101E      R2_IKB      EQU      101EH
20
21
22     ; SETUP mode vector
23     ;
24     1021      R2_SETUP    EQU      1021H
25
26
27     ; Background loop hook. This is called once every time through
28     ; the main background loop.
29     ;
30     1024      R2_BACK     EQU      1024H
31
32
33     ; Keyboard input hook. This is called once before a character
34     ; is received from the keyboard.
35     ;
36     1027      R2_KY1      EQU      1027H
37
38
39     ; Keyboard output hook. This is called once at the beginning
40     ; of each transfer to the keyboard.
41     ;
42     102A      R2_KY2      EQU      102AH
43
44
45     ; Escape parsing hook. This is called after an -ESC- is
46     ; received.
47     ;
48     102D      R2_ESC      EQU      102DH
```

```
1 ; Transmit character. This performs the function of
2 ; transmitting the character at the current position.
3 ;
4 1030 R2_TC EQU 1030H
5
6
7 ; Transmit line. This performs the function of transmitting
8 ; the line at the current position.
9 ;
10 1033 R2_TL EQU 1033H
11
12
13 ; Transmit page. This performs the function of transmitting
14 ; the page to the port that is not selected.
15 ;
16 1036 R2_TP EQU 1036H
17
18
19 ; Transmit 25th line. This performs the function of transmitting
20 ; the 25th line to the port that is not selected.
21 ;
22 1039 R2_T25L EQU 1039H
23
24
25 ; Print page. This performs the function of transmitting the
26 ; page to the opposite port in a form proper for printing (uses
27 ; CR, LF at the end of each line).
28 ;
29 103C R2_PRNT EQU 103CH
30
31
32 ; Display clock. This updates the time on the 25th line.
33 ;
34 103F R2_DCLK EQU 103FH
35
36
37 ; Display 25th status line. This displays the system status
38 ; on the 25th line if the option is enabled and the user does
39 ; not have the 25th line enabled.
40 ;
41 1042 R2_DISP EQU 1042H
```


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

```

1          ;;          Initialize system
2          ;
3          ;          Set stack pointer and call routine to set up everything
4
5          0030          ORG          30H          ; Somewhat arbitrary
6
7          0030  75      81      5F      START:    MOV      SP, #STACK          ; Initialize stack pointer
8          0033  12      1000          LCALL   R2_IN          ; Initialize everything else
9
10
11
12
13          ;;          MAIN - main control loop
14          ;
15          ;          *MAIN* is in charge of giving control to the proper routine
16          ;          whenever there is a character present in one of the FIFOs.
17
18
19          0036  12      1024          MAIN:      LCALL   R2_BACK          ; Background hook
20
21          0039  20      22      06          JB      HSSTOP, MN1          ; IF not held screen THEN
22          003C  31      F6          ACALL   FCIF          ; fetch char from input FIFO
23          003E  40      02          JC      MN1          ; IF character THEN
24          0040  11      79          ACALL   IFCP          ; processes character
25
26          0042  30      2E      20          MN1:   JNB     ONLINE, MN2          ; IF online AND
27          0045  30      13      1D          JNB     OKTRANS, MN2          ; IF output port free THEN
28          0048  20      02      03          JB      PORTF, MN1A          ; IF auxiliary port OR
29          004B  30      3E      03          JNB     HNDSHK, MN1B          ; IF hardware handshake THEN
30          004E  20      B5      14          MN1A:   JB      CTS, MN2          ; IF not CTS THEN skip
31          0051  20      16      11          MN1B:   JB      XOFFRCVD, MN2          ; IF not -XOFF- received THEN
32          0054  51      4C          ACALL   FCOF          ; fetch char from output FIFO
33          0056  40      0D          JC      MN2          ; IF character THEN
34          0058  B4      13      02          CJNE   A, #XOFF, MN1C          ; IF char = xoff THEN
35          005B  D2      11          SETB   UXOFFSENT          ; flag user xoff sent
36          005D  B4      11      02          MN1C:   CJNE   A, #XON, MN1D          ; IF char = xon THEN
37          0060  C2      11          CLR    UXOFFSENT          ; clear user xoff sent
38          0062  12      1015          MN1D:   LCALL   R2_XMT          ; transmit character
39
40          0065  20      1F      06          MN2:   JB      KBDISF, MN3          ; IF keyboard enabled THEN
41          0068  51      B6          ACALL   FCKB          ; fetch char from keyboard
42          006A  40      02          JC      MN3          ; IF keyboard data THEN
43          006C  51      EE          ACALL   KBCP          ; process keyboard data
44
45          006E  30      37      C5          MN3:   JNB     L25ON, MAIN          ; IF 25th status enabled AND
46          0071  20      15      C2          JB      L25EN, MAIN          ; IF 25th line disabled THEN
47          0074  12      103F          LCALL   R2_DCLK          ; display clock every second
48
49          0077  80      BD          SJMP   MAIN          ; Go forever

```

```

1          ;;          IFCP - input FIFO character Processor
2          ;
3          ;
4          ;          *IFCP* interprets characters which have been received (or
5          ;          struck on the keyboard if off-line or keyed as a local
6          ;          function).  If the character is part of an escape sequence
7          ;          then control is given directly to that routine by means of
8          ;          the -dispatch address-.  If it is a control character then it
9          ;          is looked up in *CTLTAB*.  If found the program jumps to the
10         ;          appropriate routine, otherwise the character is discarded.  If
11         ;          it is a displayable character it is placed on the screen
12         ;          after checking for graphics mode, attributes, etc.
13         ;
14         ;          ENTRY  (A) = character from input FIFO
15         ;
16         ;          EXIT   none
17         ;
18         ;          USES   all
19         ;
20
21 0079 B4 18 06 IFCP: CJNE A, #CAN, IFPA ; IF canceled AND
22 007C 20 43 03 JB HAZ, IFPA ; IF not Haz mode THEN
23 007F 12 082F LCALL SETNORM ; Restore dispatch address
24
25 0082 C0 16 IFPA: PUSH DSADRL ; Jump to the
26 0084 C0 15 PUSH DSADRH ; -dispatch address-
27 0086 22 RET
28
29 0087 B4 0D 12 NORM: CJNE A, #CR, IFP1 ; IF carriage return THEN
30 008A 30 36 0F JNB HSMODEF, IFP1 ; IF hold screen mode THEN
31 008D E5 12 MOV A, YPOS
32 008F 04 INC A ; set y-pos to check
33 0090 B5 18 00 CJNE A, BFIX, NRMO ; with bottom fixed region
34 0093 40 05 02 NRMO: JC NRM1 ; IF ypos >= bottom THEN
35 0095 D5 1E 02 DJNZ HSLINE, NRM1 ; decrement line
36 0098 D2 22 02 SETB HSSTOPF ; IF zero THEN stop disp
37 009A 74 0D 02 NRM1: MOV A, #CR ; restore carriage return
38
39 ;          Check for monitor mode and control characters
40 ;
41 009C 30 3A 0D IFP1: JNB MONITOR, IFP1B ; IF monitor THEN
42 009F C2 08 CLR RVVA ; start with normal display
43 00A1 B4 20 00 CJNE A, #', IFP1A
44 00A4 50 5E IFP1A: JNC IFP6 ; IF control character THEN
45 00A6 D2 08 SETB RVVA ; make it reverse video
46 00A8 44 40 ORL A, #01000000B ; convert to ordinary char
47 00AA 80 58 SJMP IFP6 ; skip ahead
48
49 00AC B4 20 00 IFP1B: CJNE A, #', IFP1C ; Normal place to jump to
50 00AF 40 03 IFP1C: JC IFP2 ; IF control character OR
51 00B1 B4 7F 21 CJNE A, #RUBOUT, IFP3 ; IF rubout THEN
52
53 ;          Character is a control character
54 ;
55 00B4 75 F0 08 IFP2: MOV B, #CTAB1L ; Get table info
56 00B7 90 0119 MOV DPTR, #CTAB1
57 00BA 20 40 12 JB ANSI, IFP2A ; IF not ANSI mode OR

```

```

1 00BD 20 41 0F JB ZDS, IFF2A ; 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, IFF2A ; 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 IFF2A: 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 IFFRET ; 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 IFF3: JNB HAZ, IFF3A ; IF Hazeltine mode AND
19 00D8 B4 7E 03 CJNE A, #10, IFF3A ; IF char = 10 THEN
20 00DB 02 067B LJMP ESCCODE ; do escape stuff
21 00DE 30 0D 13 IFF3A: JNB GRPH, IFF5 ; IF in graphics mode THEN
22 00E1 B4 5E 06 CJNE A, #5EH, IFF4
23 00E4 20 0F 0D JB ALTGRPH, IFF5 ; 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 IFF4: JC IFF5 ; IF in graphics range THEN
27 00EC 54 1F ANL A, #00011111B ; mask to graphic character
28 00EE 30 0F 03 JNB ALTGRPH, IFF5 ; IF alt graphic set THEN
29 00F1 D3 SETB C ; set carry and
30 00F2 80 02 SJMP IFF5A ; place into alt mode
31
32
33 ; Check for insert character mode
34 ;
35 00F4 A2 0C IFF5: MOV C, ALTCHARA ; Get alt char mode
36 00F6 92 E7 IFF5A: MOV ACC.7, C ; Place into data byte
37
38 00F8 30 21 09 JNB ICMODEF, IFF6 ; 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 IFF6: ACALL PUTCHR ; Put character onto screen
45 0106 22 IFFRET: RET

```

```

1          ;;          CTAB1/2/3 - control character tables
2          ;
3          ;          *CTAB1* contains all control codes for the ANSI and ZDS
4          ;          emulation modes.
5          ;
6          ;          *CTAB2* contains all control codes for the ADM 3 emulation
7          ;          mode.
8          ;
9          ;          *CTAB3* contains all control codes for the Hazeltine 1500
10         ;          emulation mode.
11         ;
12         ;          Table entries are: The control character followed by the
13         ;          address of the routine which performs the requested function.
14         ;
15
16         0107          CTAB2          EQU          $
17
18         0107          0B          DB          OBH          ; Vertical tab
19         0108          0C8D          DW          CUP          ; Cursor up
20
21         010A          0C          DB          OCH          ; Form feed
22         010B          0C64          DW          CRT          ; Cursor right
23
24         010D          0E          DB          OEH          ; Shift out
25         010E          0FB3          DW          EKI          ; Enable keyboard input
26
27         0110          0F          DB          OFH          ; Shift in
28         0111          0FB9          DW          DKI          ; Disable keyboard input
29
30         0113          1A          DB          1AH          ; Substitute
31         0114          0BAC          DW          CLRS          ; Clear screen
32
33         0116          1E          DB          1EH          ; Record separator
34         0117          0C36          DW          SCH          ; Set cursor home
35
36         0119          CTAB1          EQU          $
37
38         0119          1B          DB          ESC          ; Escape
39         011A          067B          DW          ESCCODE          ; Escape code processor
40
41         011C          08          DB          BS          ; Backspace
42         011D          0C3D          DW          PBS          ; Perform backspace
43
44         011F          09          DB          HT          ; Horizontal tabulation
45         0120          05C1          DW          TAB          ; Tab to next column
46
47         0122          0E          DB          OEH          ; Shift out
48         0123          0F81          DW          G1PICK          ; Pick G1 character set
49
50         0125          0F          DB          OFH          ; Shift in
51         0126          0F7B          DW          G0PICK          ; Pick G0 character set
52
53         0128          CTAB3          EQU          $
54
55         0128          0A          DB          LF          ; Line feed
56         0129          0555          DW          PLFCR          ; Perform LF and/or CR
57

```

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	EQU	(\$-CTAB1)/3	; Table length
8		000E	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	EQU	(\$-CTAB3)/3	; Table length

```

1          ;:          STAB - search table
2          ;
3          ;
4          ;          *STAB* searches tables of specified length for a match
5          ;          with the first character in an entry.
6          ;
7          ;          ENTRY (A) = character to match
8          ;          (B) = table length in number of entries
9          ;          (DPTR) = table address
10         ;
11         ;          EXIT (A) = original character
12         ;          (B) = number of entries left to search + 1
13         ;          (DPTR) = first address after character match
14         ;          'C' = set if no match, clear otherwise
15         ;
16         ;          USES A, B, DPTR, WORK
17
18
19 013A    FC          STAB:    MOV    WORK, A          ; Save character to match
20
21 013B    E4          ST1:    CLR    A          ; REPEAT
22 013C    93          MOV    A, @A+DPTR          ; set character from table
23 013D    A3          INC    DPTR          ; bump to address high byte
24 013E    B5          CJNE   A, WORKA, ST2       ; compare char with work
25 0141    E4          CLR    A          ; IF character = work THEN
26 0142    93          MOV    A, @A+DPTR          ; set address high byte
27 0143    C0          PUSH   ACC          ; save on stack
28 0145    A3          INC    DPTR          ; bump to address low byte
29 0146    E4          CLR    A
30 0147    93          MOV    A, @A+DPTR          ; set address low byte
31 0148    F5          MOV    DPL, A          ; save address low byte
32 014A    D0          POP    DPH          ; pop address high byte
33 014C    EC          MOV    A, WORK          ; restore original character
34 014D    C3          CLR    C          ; clear 'C' for ok
35 014E    22          RET
36
37 014F    A3          ST2:    INC    DPTR          ; bump to address low byte
38 0150    A3          INC    DPTR          ; bump to next entry if any
39 0151    D5          DJNZ   B, ST1          ; UNTIL #entries = 0
40
41 0154    EC          MOV    A, WORK          ; Restore original character
42 0155    D3          SETB  C          ; Flas 'C' for no match
43 0156    22          RET

```

```

1          ;|          PUTCHR - put character onto screen
2          ;
3          ;          *PUTCHR* puts the character onto the screen and checks for
4          ;          wrap around at the end of a line for auto CR or LF.
5          ;
6          ;
7          ;          ENTRY   (A) = character
8          ;
9          ;          EXIT    none
10         ;
11         ;          USES   A, B, DPTR, TEMP, PTR1
12
13
14 0157    D2    14          PUTCHR:    SETB   DX          ; Flag that cursor moved
15
16 0159    C0    E0          PUSH    ACC          ; Save character
17
18 015B    31    A9          ACALL   SPF          ; Skip any protected fields
19
20 015D    85    14    82    MOV     DPL, LINADL          ; Get line address
21 0160    85    13    83    MOV     DPH, LINADH
22 0163    53    83    EF    ANL    DPH, #ATRANL          ; Mask to attribute memory
23 0166    E5    21          MOV     A, ATTRIBUTES
24 0168    F0          MOVX   @DPTR, A          ; Output current attributes
25 0169    43    83    10    ORL    DPH, #CHRORL          ; Mask to character memory
26 016C    D0    E0          POP     ACC          ; Restore character
27
28 016E    F0          MOVX   @DPTR, A          ; Store into screen memory
29
30 016F    E5    11          MOV     A, XPOS
31 0171    B4    4F    09    CJNE   A, #MAXCHAR, PU1          ; IF cursor at end of line AND
32 0174    30    2B    0F    JNB   AUTOWRAP, PU2          ; IF auto wrap on THEN
33 0177    B1    5C          ACALL   PCR          ; perform carriage return
34 0179    B1    61          ACALL   PLF          ; perform line feed
35 017B    80    09          SJMP   PU2          ; exit out
36
37 017D    05    11          PU1:    INC     XPOS          ; Bump -XPOS-
38 017F    A3          INC     DPTR          ; Bump location in memory
39 0180    85    82    14    MOV     LINADL, DPL          ; Save new line address
40 0183    85    83    13    MOV     LINADH, DPH
41
42 0186    21    A9          PU2:    AJMP   SPF          ; Skip any protected fields

```



```

1          ; ; CPF - check protected fields
2          ; ;
3          ; ; *CPF* checks to see if the current position is protected.
4          ; ; If so it returns a value in the carry flag.
5          ; ;
6          ; ;
7          ; ; ENTER  XPOS and YPOS at position
8          ; ;
9          ; ; EXIT   (C) = set if protected, clear otherwise
10         ; ;
11         ; ; USES   A, B, DPTR, TEMP, WORK, PTR1
12
13
14 0188    E5    3F          CPF:    MOV    A, PFIELD      ; Get protected fields
15 018A    70    02          JNZ    CPF1         ; IF none protected THEN
16 018C    C3          CPF1:   CLR    C              ; clear carry
17 018D    22          RET     ; and return
18
19 018E    FC          CPF1:   MOV    WORK, A      ; Save value
20 018F    E5    11          MOV    A, XPOS
21 0191    85    12    F0    MOV    B, YPOS
22 0194    B1    03          ACALL  CLA              ; Calculate line address
23
24 0196    53    83    EF    ANL    DPH, #ATRANL    ; Mask to attribute memory
25 0199    E0          MOVX   A, @DPTR      ; Get current attribute
26 019A    54    0F          ANL    A, #00001111B ; Mask to set only attributes
27
28 019C    70    05          JNZ    CPF2         ; IF no attributes set THEN
29 019E    BC    80    EB    CJNE   WORK, #80H, CPF1    ; IF not protected THEN return
30 01A1    80    04          SJMP  CPF2         ; ELSE return protected
31 01A3    5C          CPF2:   ANL    A, WORK      ; ELSE mask with protected
32 01A4    B5    04    E5    CJNE   A, WORKA, CPF1    ; IF no match THEN return
33
34 01A7    D3          CPF2:   SETB  C              ; Set carry
35 01A8    22          RET     ; Return

```

```

1          ;:          SPF - skip over protected fields
2          ;
3          ;          *SPF* checks to see if the current position is protected.....
4          ;          If so then the cursor is moved to the next unprotected
5          ;          character position on the screen.... If there are no unprotected
6          ;          positions from the current cursor position to the end of the
7          ;          screen then the cursor is moved to the HOME position.
8          ;          The screen address is only updated if the cursor position
9          ;          changes.
10         ;
11         ;
12         ;          ENTRY   none
13         ;
14         ;          EXIT    none
15         ;
16         ;          USES   A, B, DPTR, WORK, TEMP, PTR1
17         ;
18         ;
19 01A9    31    88          SPF:          ACALL   CPF          ; Check protected fields
20 01AB    40    01          JC          SPO          ; IF not protected THEN
21 01AD    22          RET          ; return
22
23 01AE    E5    11          SPO:          MOV     A, XPOS
24 01B0    B4    4F    11    CJNE    A, #MAXCHAR, SP2      ; IF end of line THEN
25 01B3    75    11    00    MOV     XPOS, #0      ; beginning of line
26 01B6    E5    12          MOV     A, YPOS
27 01B8    B4    17    05    CJNE    A, #MAXLINE-1, SP1  ; IF end of screen THEN
28 01BB    75    12    00    MOV     YPOS, #0      ; beginning of screen
29 01BE    81    EA          AJMP   BLINAD      ; new address and return
30
31 01C0    05    12          SP1:          INC     YPOS          ; ELSE bump line number
32 01C2    80    02          SJMP   SP3          ; calc address and go
33
34 01C4    05    11          SP2:          INC     XPOS          ; ELSE bump column number
35 01C6    91    EA          SP3:          ACALL  BLINAD      ; calc new address
36 01C8    80    DF          SJMP   SPF          ; keep on going

```

```

1          ;;          PCIF - place character in input FIFO
2          ;;
3          ;;          *PCIF* places a single character into the input FIFO.
4          ;;
5          ;;
6          ;;          ENTRY (A) = character
7          ;;
8          ;;          EXIT (A) = character
9          ;;          'C' = set if no room, clear otherwise
10         ;;          'FO' = set if XOFF is needed, clear otherwise
11         ;;
12         ;;          USES A B, DPTR
13
14
15 01CA    C2    AF          PCIF:    CLR    EA          ; Lock out entries
16 01CC    75    83    57    MOV    DPH, #HIGH (IBUF)
17 01CF    85    0C    82    MOV    DPL, ISTORE          ; Load input buffer pointer
18 01D2    F0          MOVX   @DPTR, A          ; Write character into FIFO
19 01D3    F5    F0    MOV    B, A          ; Save temporary
20
21 01D5    E5    0E          MOV    A, ICOUNT
22 01D7    B4    20    03    CJNE   A, #ILEN, PI1          ; IF queue is full THEN
23 01DA    D3          SETB   C          ; set 'C' for no room
24 01DB    80    14          SJMP   PIR          ; and exit out
25
26 01DD    C2    D5          PI1:    CLR    FO          ; No -XOFF- until otherwise
27 01DF    B4    15    02    CJNE   A, #IHI, PI2          ; IF queue at XOFF point THEN
28 01E2    D2    D5          SETB   FO          ; flag that -XOFF- is needed
29
30 01E4    05    0E          PI2:    INC    ICOUNT          ; Bump count
31 01E6    05    0C          INC    ISTORE          ; Bump store and check wrap
32 01E8    E5    0C          MOV    A, ISTORE
33 01EA    B4    F0    03    CJNE   A, #LOW (IBUF+ILEN), PI3
34 01ED    75    0C    D0    MOV    ISTORE, #LOW (IBUF)
35
36 01F0    C3          PI3:    CLR    C          ; Flag everything OK
37 01F1    D2    AF    PIR:    SETB   EA          ; Enable entries
38 01F3    E5    F0    MOV    A, B          ; Restore character
39 01F5    22          RET

```

```

1          ;:          FCIF = fetch character from input FIFO
2          ;
3          ;          *FCIF* is used to fetch a single character from the input
4          ;          FIFO if any are available.
5          ;
6          ;
7          ;          ENTRY none
8          ;
9          ;          EXIT (A) = character if available
10         ;          'C' = set if no character, clear otherwise
11         ;
12         ;          USES A, B, DPTR
13
14
15 01F6    E5    0E          FCIF:    MOV    A, ICOUNT
16 01F8    70    02          JNZ    F11          ; IF no data in queue THEN
17 01FA    D3          SETB   C          ; set carry and return
18 01FB    22          RET
19
20 01FC    B4    09    03    F11:    CJNE   A, #ILO, F12          ; IF low water mark THEN
21 01FF    12    1018    LCALL  R2_XON          ; send -XON- now
22
23 0202    C2    AF          F12:    CLR    EA          ; Lock out entries
24 0204    75    83    57    MOV    DPH, #HIGH (IBUF)
25 0207    85    0D    82    MOV    DPL, IFETCH          ; Load input buffer pointer
26 020A    E0          MOVX  A, @DPTR          ; Read character from FIFO
27 020B    F5    F0          MOV    B, A          ; Save temporary
28
29 020D    05    0D          INC    IFETCH          ; Bump fetch and check wrap
30 020F    E5    0D          MOV    A, IFETCH
31 0211    B4    F0    03    CJNE   A, #LOW (IBUF+ILEN), F13
32 0214    75    0D    D0    MOV    IFETCH, #LOW (IBUF) ; Reset value of fetch pointer
33
34 0217    15    0E          F13:    DEC    ICOUNT          ; Decrement count
35 0219    21    F0          AJMP  P13          ; Return (saves ROM)

```

```
1          ;;          LCIF - look at next character on input FIFO
2          ;
3          ;          *LCIF* returns the next character from the input FIFO if there
4          ;          is one but does not take it off the FIFO. This is needed for
5          ;          'look ahead' applications.
6          ;
7          ;
8          ;          ENTRY   none
9          ;
10         ;          EXIT   (A) = character if available
11         ;          'C' = set if no character, clear otherwise
12         ;
13         ;          USES   A, DPTR
14         ;
15
16 021B    E5    0E          LCIF:    MOV    A, ICOUNT
17 021D    70    02          JNZ   LI1          ; IF no data in queue THEN
18 021F    D3          SETB  C          ; set carry and return
19 0220    22          RET
20
21 0221    75    83    57    LI1:    MOV    DPH, #HIGH (IBUF)
22 0224    85    0D    82          MOV    DPL, IFETCH          ; Load input buffer pointer
23 0227    E0          MOVX  A, @DPTR          ; Read character from FIFO
24 0228    C3          CLR   C          ; Flag everything ok
25 0229    22          RET
```

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

```

1      ;:      PCOF - place character in output FIFO
2      ;
3      ;      *PCOF* places a single character into the output FIFO and
4      ;      checks for online and half duplex.
5      ;
6      ;
7      ;      ENTRY (A) = character
8      ;
9      ;      EXIT (A) = character
10     ;      ^C = set if no room in FIFO, clear otherwise
11     ;
12     ;      USES A, B, DPTR
13
14
15 022A 20 2E 02 PCOF: JB ONLINE, P01 ; IF offline THEN
16 022D 80 9B SJMP PCIF ; Place on input and return
17
18 022F 20 2F 02 P01: JB FULLDPLX, P02 ; IF in half duplex mode THEN
19 0232 31 CA ACALL PCIF ; Place on input also
20
21 0234 F5 F0 P02: MOV B, A ; Save temporary
22 0236 E5 0F MOV A, OSTORE
23 0238 B4 F7 04 CJNE A, #LOW(OLEN), P03 ; IF store = queue length THEN
24 023B D3 SETB C ; set ^C for full
25 023C E5 F0 MOV A, B ; restore char and return
26 023E 22 RET
27
28 023F C3 P03: CLR C ; ELSE clear ^C
29 0240 75 83 57 MOV DPH, #HIGH(OBUF) ; set up the output
30 0243 85 0F 82 MOV DPL, OSTORE ; buffer pointer
31 0246 E5 F0 MOV A, B ; restore character
32 0248 F0 MOVX @DPTR, A ; write character to buffer
33 0249 05 0F INC OSTORE ; bump store and return
34 024B 22 RET

```

```

1          ; ; FCOF - fetch character from output FIFO
2          ; ;
3          ; ; *FCOF* fetches a character from the output FIFO if any
4          ; ; are available.
5          ; ;
6          ; ;
7          ; ; ENTRY none
8          ; ;
9          ; ; EXIT (A) = character if available
10         ; ; 'C' = set if no characters, clear otherwise
11         ; ;
12         ; ; USES A, B, DPTR
13
14
15 024C E5 OF FCOF: MOV A, OSTORE
16 024E B5 10 02 CJNE A, OFETCH, F01 ; IF store = fetch THEN
17 0251 D3 SETB C ; set 'C' and return
18 0252 22 RET
19
20 0253 75 83 57 F01: MOV DPH, #HIGH (OBUF)
21 0256 85 10 82 MOV DPL, OFETCH ; Load output buffer pointer
22 0259 E0 MOVX A, @DPTR ; Get character from buffer
23 025A F5 F0 MOV B, A ; Save temporary
24 025C C2 AF CLR EA ; Lock out entries
25 025E 05 10 INC OFETCH ; Bump fetch pointer
26 0260 E5 OF MOV A, OSTORE
27
28 0262 B5 10 06 CJNE A, OFETCH, FOR ; IF store = fetch THEN
29 0265 75 10 F0 MOV OFETCH, #LOW (OBUF) ; reset the values of
30 0268 75 OF F0 MOV OSTORE, #LOW (OBUF) ; fetch and store
31
32 026B 21 F0 FOR: AJMP P13 ; Return (saves ROM)

```

```

1          ;:          PCKB - place character out to the keyboard
2          ;
3          ;          *PCKB* outputs bytes to the keyboard using a special handshake.
4          ;
5          ;
6          ;          ENTRY (A) = byte to output
7          ;
8          ;          EXIT none
9          ;
10         ;          USES A, B, WORK
11
12
13 026D    12    102A    PCKB:    LCALL    R2_KY2    ; Keyboard hook
14
15 0270    30    91      0C      JNB     KBIN, PK1    ; IF data THEN
16 0273    C0    E0      PUSH    ACC        ; save byte
17 0275    51    B4      ACALL   FCKB        ; fetch char from keyboard
18 0277    40    02      JC      PK0        ; IF none now THEN exit
19 0279    51    EE      ACALL   KBOP        ; ELSE process character
20 027B    D0    E0      PK0:    POP     ACC
21 027D    80    EE      Sjmp   PCKB        ; and try again
22
23 027F    C2    AF      PK1:    CLR     EA        ; Disable interrupts
24 0281    C2    90      CLR     KBOUT     ; Request keyboard attention
25 0283    75    F0      MOV     B, #8     ; Initialize index for 8 bits
26
27 0286    30    91      06      JNB     KBIN, PK1A  ; IF contention THEN
28 0289    D2    AF      SETB   EA        ; enable interrupts
29 028B    D2    90      SETB   KBOUT     ; wipe out request
30 028D    80    DE      Sjmp   PCKB        ; try again
31
32 028F    D2    AF      PK1A:  SETB   EA        ; Enable interrupts
33
34 0291    13      PKL:    RRC     A        ; REPEAT rotate next bit in
35 0292    7C    00      MOV    WORK, #0   ; init count for safety check
36 0294    20    91      0E      PK2:    JB     KBIN, PK3   ; REPEAT wait for request
37 0297    51    B5      ACALL  PKDLY     ; delay some time
38 0299    51    B5      ACALL  PKDLY
39 029B    51    B5      ACALL  PKDLY
40 029D    51    B5      ACALL  PKDLY
41 029F    51    B5      ACALL  PKDLY
42 02A1    DC    F1      DJNZ   WORK, PK2 ; UNTIL check is done
43 02A3    80    0E      Sjmp   PKR
44
45 02A5    92    90      PK3:    MOV    KBOUT, C   ; output data
46 02A7    7C    00      MOV    WORK, #0   ; init count for safety check
47 02A9    30    91      04      PK4:    JNB   KBIN, PK5   ; REPEAT wait for signal
48 02AC    DC    FB      DJNZ   WORK, PK4 ; UNTIL safety check done
49 02AE    80    03      Sjmp   PKR
50
51 02B0    D5    F0      DE      PK5:    DJNZ   B, PKL    ; UNTIL all 8 bits are done
52
53 02B3    D2    90      PKR:    SETB  KBOUT     ; Restore status
54 02B5    22      PKDLY:  RET

```



```

1          ;:          FCKB - fetch character from keyboard
2          ;
3          ;
4          ;          *FCKB* takes characters from the keyboard using a special
5          ;          handshaking.
6          ;
7          ;          ENTRY   none
8          ;
9          ;          EXIT   (A) = character if available
10         ;          'C' = set if no character, clear otherwise
11         ;
12         ;          USES   A, B, DPTR
13         ;
14         ;
15 02B4    12    1027    FCKB:    LCALL   R2_KY1          ; Keyboard hook
16         ;
17 02B9    20    91      02      JB     KBIN, FK1          ; IF no data THEN
18 02BC    D3                    SETB   C              ; set carry and return
19 02BD    22                    RET
20         ;
21 02BE    75    39      00      FK1:   MOV    AUTOCNT, #0
22 02C1    D2    24                    SETB   VSPF          ; Toggle screen saver
23         ;
24 02C3    E4                    CLR    A              ; Init receiving byte
25 02C4    75    F0      08      MOV    B, #8          ; Init index counter
26         ;
27 02C7    C2    90      FKL:   CLR    KBOUT          ; REPEAT request data
28 02C9    51    E1                    ACALL  FKDLY          ; delay 200 usec
29 02CB    51    E1                    ACALL  FKDLY
30 02CD    A2    91      MOV    C, KBIN          ; set data bit
31 02CF    D2    90      SETB   KBOUT          ; data received
32 02D1    92    E7      MOV    ACC, C          ; Place data into word
33 02D3    D5    F0      02      DJNZ  B, FK2          ; IF all 8 bits THEN
34 02D6    80    05      SJMP   FKR          ; exit out
35         ;
36 02D8    03      FK2:   RR     A              ; rotate to next bit
37 02D9    51    E1                    ACALL  FKDLY          ; delay 100 usec
38 02DB    80    EA      SJMP   FKL          ; UNTIL forever
39         ;
40 02DD    51    E1      FKR:   ACALL  FKDLY          ; Delay last time to let clear
41 02DF    C3                    CLR    C              ; Clear 'C' for ok
42 02E0    22                    RET
43         ;
44         ;          Delay routine
45         ;
46 02E1    75    83      1E      FKDLY:  MOV    DPH, #30          ; Delay of 100 usec
47 02E4    D5    83      FD      FKDLY1: DJNZ  DPH, FKDLY1
48 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
```

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

```

1          ; ; KBCP - keyboard character processor
2          ;
3          ; ; *KBCP* processes any characters received from the keyboard.
4          ;
5          ;
6          ; ; ENTRY (A) = byte from keyboard
7          ;
8          ; ; EXIT none
9          ;
10         ; ; USES all
11
12
13 02EE 20 E7 03 KBCP: JB ACC.7, KPO ; IF ASCII character THEN
14 02F1 41 2A AJMP PCOF ; output char and return
15 02F3 22 KPR: RET
16
17 02F4 A2 E6 KPO: MOV C, ACC.6
18 02F6 92 1C MOV SHIFT, C ; Save shift value
19 02F8 A2 E5 MOV C, ACC.5
20 02FA 92 1D MOV CONTLF, C ; Save control value
21 02FC 54 9F ANL A, #10011111B ; Mask off shift and control
22 02FE 30 1E 02 JNB ACCESSF, KP1 ; IF access in progress THEN
23 0301 61 CD AJMP KPA ; do a long jump
24
25
26         ; ; No access in progress
27         ;
28 0303 75 F0 09 KP1: MOV B, #KBDTL ; (B) = table length
29 0306 90 044B MOV DPTR, #KBDT ; (DPTR) = keyboard data table
30 0309 31 3A ACALL STAB ; Search the table
31 030B 40 02 JC KP2 ; IF found THEN
32
33
34         ; ; Command from keyboard
35         ;
36 030D E4 CLR A
37 030E 73 JMP @A+DPTR ; Jump to routine
38
39
40         ; ; Cursor keys
41         ;
42 030F FD KP2: MOV TEMP, A
43 0310 E5 0F MOV A, OSTORE ; IF output FIFO not empty THEN
44 0312 B5 10 DE CJNE A, OFETCH, KPR ; loose character and return
45 0315 ED MOV A, TEMP
46
47 0316 75 F0 07 MOV B, #KBCRTL ; (B) = table length
48 0319 90 0466 MOV DPTR, #KBCRT ; (DPTR) = kb cursor table
49 031C 31 3A ACALL STAB ; Search the table
50 031E 40 2F JC KP6 ; IF found THEN
51 0320 20 42 D0 JB ADM3, KPR ; IF ADM.3 mode THEN exit
52 0323 20 43 CD JB HAZ, KPR ; IF Haz mode THEN exit
53 0326 C0 82 PUSH DPL ; save values
54 0328 C0 83 PUSH DPH
55 032A 74 1B MOV A, #ESC
56 032C 91 14 ACALL KPCOF ; output escape
57 032E 30 40 04 JNB ANSI, KP3 ; IF in ANSI mode THEN

```

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

```

1 0331 74 5B      MOV    A, #'['
2 0333 91 14      ACALL  KPCOF          ; output '['
3 0335 D0 E0      POP    ACC          ;
4 0337 60 12      JZ     KP4          ;
5 0339 30 1C      JNB    SHIFTF, KP4 ; IF erase key AND
6 033C 30 40      JNB    ANSI, KP5   ; IF shift key down 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     POP    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 ;
18 ; Numeric keypad key
19 034F A2 1C      KP6:  MOV    C, SHIFTF
20 0351 30 30      JNB    KPADSHFT, KP7 ; IF keypad is shifted THEN
21 0354 B3 1C      CPL    C            ; invert shift function
22 0355 92 1C     KP7:  MOV    SHIFTF, C    ; save final result
23 0357 92 E6     MOV    ACC, C       ; place into data word also
24 0359 75 F0      MOV    B, #KBDL     ; (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      KP16: JB    SHIFTF, KP8   ; IF keypad unshifted THEN
31
32 0368 20 31      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      KP9:  JB    ADM3, KP16A   ; IF ADM 3 mode THEN ordinary
38 0374 20 43      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, #'0'      ; IF in ANSI mode THEN
43 037F 20 40      JB    ANSI, KP11   ; output '0'
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 ;
51 ; Insert mode key
52 038A 20 42      KP8:  JB    ADM3, KP16A   ; IF ADM 3 mode THEN ordinary
53 038D 20 43      JB    HAZ, KP16A   ; IF Haz mode THEN ordinary
54 0390 B4 D7      CJNE  A, #KB_L7 + KB_SHFT, KP12
55 0393 74 1B     MOV   A, #ESC
56 0395 91 14     ACALL KPCOF        ; Output ESC
57

```

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

```

1 0397 20 40 09 JB ANSI, KP13 ; IF in ZDS mode THEN
2 039A 74 40 MOV A, #'e'
3 039C 30 21 02 JNB ICMODEF, KP14 ; check if in IC mode
4 039F 74 4F MOV A, #'D'
5 03A1 81 14 KP14: AJMP KPCOF ; output char and return
6
7 03A3 74 5B KP13: MOV A, #'I' ; IF in ANSI mode THEN
8 03A5 91 14 ACALL KPCOF ; output 'I'
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, #'1'
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, #'t'
27 03C3 91 14 ACALL KPCOF ; IF ANSI THEN output 'I'
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, #1111000B ; 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 ; set 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 18          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, #'0'
6 03F9 91 14          ACALL  KPCOF ; output '0'
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          POP    ACC ; IF in ZDS mode THEN
13 0404 D0 E0          POP    ACC ; set 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           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  PCDF ; output char and return
28
29 0419 21 CA          KPC1: AJMP  PCIF ; ELSE do local and return

```

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

```

1          ;;          KBTAB - keyboard tab routine
2          ;
3          ;          *KBTAB* checks to see if the SHIFT key is down. If it is
4          ;          down then it checks for ZDS or ANSI mode. If it is in one
5          ;          of those modes then the sequence to for backtab is sent.
6          ;
7          ;
8          ;          ENTRY   none
9          ;
10         ;          EXIT    none
11         ;
12         ;          USES    all
13
14
15 041B 30 1C OF KBTAB: JNB SHIFT, KT1 ; IF not SHIFT THEN just TAB
16 041E 90 0431 MOV DPTR, #KTM1 ; set ANSI sequence
17 0421 20 40 06 JB ANSI, KTO ; IF ANSI mode THEN do it
18 0424 30 41 06 JNB ZDS, KT1 ; IF not ZDS THEN just TAB
19 0427 90 0434 MOV DPTR, #KTM2 ; ELSE set ZDS sequence
20 042A 02 0E6D LUMP PSOF ; output back tab sequence
21
22 042D 74 09 KT1: MOV A, #HT ;
23 042F 41 2A AJMP PCOF ; ELSE output horizontal tab
24
25 0431 1B 5B DA KTM1: DB ESC, ('Z'+80H) ; ESC [ Z
26 0434 1B AD KTM2: DB ESC, ('^'+80H) ; ESC ^
27
28
29
30
31         ;;          KBSpace - keyboard space routine
32         ;
33         ;          *KBSpace* outputs a space unless the control key is down
34         ;          in which case it outputs a null character.
35         ;
36         ;
37         ;          ENTRY   none
38         ;
39         ;          EXIT    none
40         ;
41         ;          USES    all
42
43
44 0436 74 20 KBSpace: MOV A, # ' ' ; Default to space
45 0438 30 1D 01 JNB CNTRL, KBS1 ; IF control space THEN
46 043B E4 CLR A ; output a null
47 043C 41 2A KBS1: AJMP PCOF ; Output character and return

```



```

1          ;;          KBDT - keyboard data table
2          ;
3          ;          *KBDT* contains the commands received from the keyboard
4          ;
5          ;          Table entries are: Keyboard data word followed by the
6          ;          address of the routine which performs the requested function.
7
8
9          044B          KBDT          EQU          $
10
11 044B          8B          DB          KB_TAB          ; Tab key
12 044C          041B          DW          KBTAB
13
14 044E          8C          DB          KB_SPACE          ; Space bar
15 044F          0436          DW          KBSPACE
16
17 0451          9F          DB          KB_ACC          ; Access code
18 0452          0448          DW          KBACC
19
20 0454          8F          DB          KB_POWER          ; Keyboard just powered up
21 0455          101E          DW          R2_IKB
22
23 0457          89          DB          KB_BREAK          ; Break pressed
24 0458          0FCC          DW          KBBREAK
25
26 045A          8A          DB          KB_CAPS          ; Caps locked mode
27 045B          0F8F          DW          KBCPLK
28
29 045D          87          DB          KB_SCROLL          ; Scroll key pressed
30 045E          0FE3          DW          KBSCROLL
31
32 0460          88          DB          KB_SETUP          ; Setup key pressed
33 0461          1021          DW          R2_SETUP
34
35 0463          9B          DB          KB_ENT          ; Enter key pressed
36 0464          043E          DW          KBENT
37
38          0009          KBDTL          EQU          ($-KBDT)/3          ; Table length

```

```

1
2
3      ;
4      ;
5      ;
6      ;
7      ;
8      ;
9      ;
10     ;
11     ;
12
13     0466      KBCRT      EQU      $
14
15     0466      80
16     0467      00      41
17
18     0469      81
19     046A      00      42
20
21     046C      82
22     046D      00      44
23
24     046F      83
25     0470      00      43
26
27     0472      84
28     0473      00      48
29
30     0475      86
31     0476      00      7E
32
33     0478      85
34     0479      45      4A
35
36     0007      KBCRTL     EQU      ($-KBCRT)/3

```

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.

; UP arrow key
; Down arrow key
; Left arrow key
; Right arrow key
; Home key
; Help key
; Erase key
; Table length

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

```

1          ;;          KBPDT - keyboard key pad table
2          ;;
3          ;;          *KBPDT* contains the ASCII values to be transmitted by the
4          ;;          terminal when received from the keyboard.
5          ;;
6          ;;          Table entries are: Keyboard data word followed by the
7          ;;          ASCII character to be transmitted if in unshifted followed
8          ;;          by the ASCII character to be transmitted if in any mode.
9          ;;          The shifted key format uses keyboard data + shift function
10         ;;          followed by teh escape sequence or ASCII character to be
11         ;;          transmitted. The exception to this is the insert character
12         ;;          key. In this case the first character is the character for
13         ;;          entering IC mode and the second is for leaving IC mode.
14
15
16         047B          KBPDT          EQU          $
17
18         047B          90          DB          KB_0          ; Keypad 0 key
19         047C          30          70          DB          '0', 'P'
20
21         047E          91          DB          KB_1          ; Keypad 1 key
22         047F          31          71          DB          '1', 'q'
23
24         0481          92          DB          KB_2          ; Keypad 2 key
25         0482          32          72          DB          '2', 'r'
26
27         0484          93          DB          KB_3          ; Keypad 3 key
28         0485          33          73          DB          '3', 's'
29
30         0487          94          DB          KB_4          ; Keypad 4 key
31         0488          34          74          DB          '4', 't'
32
33         048A          95          DB          KB_5          ; Keypad 5 key
34         048B          35          75          DB          '5', 'u'
35
36         048D          96          DB          KB_6          ; Keypad 6 key
37         048E          36          76          DB          '6', 'v'
38
39         0490          97          DB          KB_7          ; Keypad 7 key
40         0491          37          77          DB          '7', 'w'
41
42         0493          98          DB          KB_8          ; Keypad 8 key
43         0494          38          78          DB          '8', 'x'
44
45         0496          99          DB          KB_9          ; Keypad 9 key
46         0497          39          79          DB          '9', 'y'
47
48         0499          9A          DB          KB_DEC          ; Keypad decimal point key
49         049A          2E          6E          DB          '.', 'n'
50
51         049C          9B          DB          KB_ENT          ; Keypad enter key
52         049D          0D          4D          DB          CR, 'M'
53
54         049F          9C          DB          KB_DASH          ; Keypad dash (minus) key
55         04A0          2D          6D          DB          '-', 'm'
56
57         04A2          9D          DB          KB_COMMA          ; Keypad comma key

```

1	04A3	2C	6C	DB	`,,`, `1`	
2						
3	04A5	D0		DB	KB_0 + KB_SHFT	; Keypad 0 key shifted
4	04A6	30	00	DB	`0`, NULL	
5						
6	04A8	D1		DB	KB_1 + KB_SHFT	; Keypad 1 key shifted
7	04A9	1B	4C	DB	ESC, `L`	
8						
9	04AB	D2		DB	KB_2 + KB_SHFT	; Keypad 2 key shifted
10	04AC	1B	42	DB	ESC, `B`	
11						
12	04AE	D3		DB	KB_3 + KB_SHFT	; Keypad 3 key shifted
13	04AF	1B	4D	DB	ESC, `M`	
14						
15	04B1	D4		DB	KB_4 + KB_SHFT	; Keypad 4 key shifted
16	04B2	1B	44	DB	ESC, `D`	
17						
18	04B4	D5		DB	KB_5 + KB_SHFT	; Keypad 5 key shifted
19	04B5	1B	48	DB	ESC, `H`	
20						
21	04B7	D6		DB	KB_6 + KB_SHFT	; Keypad 6 key shifted
22	04B8	1B	43	DB	ESC, `C`	
23						
24	04BA	D7		DB	KB_7 + KB_SHFT	; Keypad 7 key shifted
25	04BB	1B	40	DB	ESC, `@`	
26						
27	04BD	D8		DB	KB_8 + KB_SHFT	; Keypad 8 key shifted
28	04BE	1B	41	DB	ESC, `A`	
29						
30	04C0	D9		DB	KB_9 + KB_SHFT	; Keypad 9 key shifted
31	04C1	1B	4E	DB	ESC, `N`	
32						
33	04C3	DA		DB	KB_DEC + KB_SHFT	; Keypad decimal key shifted
34	04C4	2E	00	DB	`,` , NULL	
35						
36	04C6	DB		DB	KB_ENT + KB_SHFT	; Keypad enter key shifted
37	04C7	0D	00	DB	CR, NULL	
38						
39	04C9	DC		DB	KB_DASH + KB_SHFT	; Keypad dash key shifted
40	04CA	2D	00	DB	`,` , NULL	
41						
42	04CC	DD		DB	KB_COMMA + KB_SHFT	; Keypad comma key shifted
43	04CD	2C	00	DB	`,` , NULL	
44						
45		001C		EQU	(\$-KBPDT)/3	; Table length

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

```

1          ;;          KBFNT - keyboard function table
2          ;
3          ;          *KBFNT* contains the ASCII values to be transmitted by the
4          ;          terminal when received from the keyboard.
5          ;
6          ;          Table entries are: Keyboard data word followed by the
7          ;          ASCII character to be transmitted if in ANSI mode followed
8          ;          by the ASCII character to be transmitted in any other mode.
9
10         11          04CF          KBFNT          EQU          $
11
12         13         04CF          80          DB          KB_LF1          ; Function key #1
13         14         04D0          53          53          DB          'S', 'S'          ; ESC O S, ESC S
14
15         16         04D2          81          DB          KB_LF2          ; Function key #2
16         17         04D3          54          54          DB          'T', 'T'          ; ESC O T, ESC T
17
18         19         04D5          82          DB          KB_LF3          ; Function key #3
19         20         04D6          55          55          DB          'U', 'U'          ; ESC O U, ESC U
20
21         22         04D8          83          DB          KB_LF4          ; Function key #4
22         23         04D9          56          56          DB          'V', 'V'          ; ESC O V, ESC V
23
24         25         04DB          84          DB          KB_LF5          ; Function key #5
25         26         04DC          57          57          DB          'W', 'W'          ; ESC O W, ESC W
26
27         28         04DE          85          DB          KB_LF6          ; Function key #6 (blue)
28         29         04DF          50          50          DB          'P', 'P'          ; ESC O P, ESC P
29
30         31         04E1          86          DB          KB_LF7          ; Function key #7 (red)
31         32         04E2          51          51          DB          'Q', 'Q'          ; ESC O Q, ESC Q
32
33         34         04E4          87          DB          KB_LF8          ; Function key #8 (gray)
34         35         04E5          52          52          DB          'R', 'R'          ; ESC O R, ESC R
35
36         37         04E7          88          DB          KB_LF9          ; Function key #9
37         38         04E8          58          C9          DB          'X', ('I'+80H)          ; ESC O X, ESC O I
38
39         40          0009          KBFNTL          EQU          ($-KBFNT)/3          ; Table length

```

```

1          ;
2          ;
3          ;
4          ;
5          ;
6          ;
7          ;
8          ;
9          ;
10         ;
11         ;
12         ;
13         ;
14 04EA    C0      F0          BLINAD:    PUSH    B          ; Save register
15 04EC    74      47          MOV     A, #LORDER ; Get base of table
16 04EE    25      12          ADD     A, YPOS    ; Add in offset
17 04F0    F8          MOV     PTR1, A      ; Place into pointer
18 04F1    E6          MOV     A, @PTR1    ; Fetch pointer
19 04F2    75      F0      50    MOV     B, #NUMCHARS ; Set up for multiply
20 04F5    A4          MUL     AB          ; Do it
21 04F6    25      11          ADD     A, XPOS    ; Add in XPOS offset
22 04F8    F5      14          MOV     LINADL, A  ; Save low order byte
23 04FA    E5      F0          MOV     A, B       ; Get high byte
24 04FC    34      50          ADDC   A, #HIGH (CHARMEM) ; Add carry and offset
25 04FE    F5      13          MOV     LINADH, A  ; Save high order byte
26 0500    D0      F0          POP     B          ; Restore
27 0502    22          RET

```

```
1          ;;          CLA - calculate line address.....
2          ;;
3          ;;          *CLA* inputs line coordinates and calculates the current
4          ;;          address of that coordinate to character memory.
5          ;;
6          ;;
7          ENTRY   (A) = X-position.....
8                (B) = Y-position
9          ;;
10         EXIT    (DPTR) = current address
11         ;;
12         USES    A, B, DPTR, TEMP, PTR1
13
14
15 0503  FD          CLA:      MOV    TEMP, A          ; Save -XPOS-
16 0504  74          MOV    A, #LORDER      ; Get base of table
17 0506  25 F0      ADD    A, B            ; Add in -YPOS- offset
18 0508  F8          MOV    PTR1, A        ; Move into pointer
19 0509  E6          MOV    A, @PTR1       ; Get value from table
20 050A  75 F0 50   MOV    B, #NUMCHARS      ; Set up for multiply
21 050D  A4          MUL    AB            ; Do it.....
22 050E  2D          ADD    A, TEMP        ; Add in -XPOS- offset
23 050F  F5 82     MOV    DPL, A          ; Save low byte
24 0511  E5 F0     MOV    A, B            ; Get high byte
25 0513  34 50     ADDC   A, #HIGH (CHARMEM) ; Add memory offset and carry
26 0515  F5 83     MOV    DPH, A          ; Save high byte
27 0517  22          RET
```

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

```

1          ;:          PRLF - perform a reverse line feed
2          ;
3          ;
4          ;          *PRLF* moves the cursor up one line in the video memory.
5          ;          If the cursor was already on the top line of the display,
6          ;          the display is scrolled down and the new line is filled
7          ;          with spaces.
8          ;
9          ;
10         ;          ENTRY   none
11         ;
12         ;          EXIT    none
13         ;
14         ;          USES   all
15
16 0518    E5      12          PRLF:      MOV     A, YPOS          ; Get Y-position
17 051A    B5      17      2B          CJNE   A, TSCROLL, RL1    ; IF top of scroll region THEN
18 051D    85      18      F0          MOV     B, BFIX          ; clear bottom line
19 0520    15      F0          DEC     B                ; of the scrolling region
20 0522    12      0B%        LCALL  BLRLINE
21 0525    74      47          MOV     A, #LORDER      ; Get top of table
22 0527    25      18          ADD     A, BFIX          ; Add in offset
23 0529    14          DEC     A
24 052A    F8          MOV     PTR1, A
25 052B    F9          MOV     PTR2, A
26 052C    19          DEC     PTR2
27 052D    86      05          MOV     TEMPA, @PTR1    ; Save bottom to put on top
28 052F    E5      18          MOV     A, BFIX
29 0531    95      17          SUBB   A, TSCROLL
30 0533    14          DEC     A                ; Calc # times through loop
31 0534    FC          MOV     WORK, A
32 0535    C2      AF          CLR     EA                ; Disable for flicker
33 0537    E7          MOV     A, @PTR2          ; REPEAT
34 0538    F6          MOV     @PTR1, A          ; exchange values
35 0539    18          DEC     PTR1
36 053A    19          DEC     PTR2                ; bump down
37 053B    DC      FA          DJNZ   WORK, RLL        ; UNTIL work = 0
38
39 053D    A6      05          MOV     @PTR1, TEMPA    ; Put last value in
40 053F    D2      AF          SETB   EA                ; Enable interrupts again
41 0541    E5      12          MOV     A, YPOS          ; Keep same position
42 0543    F5      12          RLL:   MOV     YPOS, A
43 0545    91      EA          ACALL  BLINAD          ; Build line address
44 0547    22          RET                ; Return
45 0548    B4      00      01      RL1:   CJNE   A, #0, RL1A    ; IF top of screen THEN
46 054B    22          RET                ; return
47 054C    14          RL1A:  DEC     A                ; ELSE bump up
48 054D    80      F4          SJMP  RLR                ; and return

```



```
1          ;:          PCRLF - perform carriage return and/or line feed
2          ;
3          ;          *PCRLF* performs a carriage return and then performs
4          ;          a line feed if the auto line feed function is selected.
5          ;
6          ;
7          ;          ENTRY none
8          ;
9          ;          EXIT none
10         ;
11         ;          USES A, B, DPTR, TEMP, PTR1
12
13
14 054F B1 5C          PCRLF:          ACALL PCR          ; Perform carriage return
15 0551 20 2D 0D      JB AUTOLF, PLF      ; IF auto LF THEN do *PLF*
16 0554 22          RET          ; ELSE return
17
18
19
20
21         ;:          PLFCR - perform line feed and/or carriage return
22         ;
23         ;          *PLFCR* performs a carriage return prior to performing a
24         ;          line feed if the auto carriage return function is selected.
25         ;
26         ;
27         ;          ENTRY none
28         ;
29         ;          EXIT none
30         ;
31         ;          USES A, B, DPTR, TEMP, PTR1
32
33
34 0555 30 2C 09      PLFCR:          JNB AUTO CR, PLF      ; IF no auto CR THEN do *PLF*
35 0558 B1 5C          ACALL PCR          ; ELSE do carriage return
36 055A 80 05          SJMP PLF          ; line feed and return
```

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

```
1          ;:          PCR - perform carriage return
2          ;
3          ;          *PCR* moves the cursor to the beginning of the current line.
4          ;
5          ;
6          ;          ENTRY  none
7          ;
8          ;          EXIT   none
9          ;
10         ;          USES   A
11
12
13 055C    75    11    00    PCR:    MOV    XPOS, #0          ; Return to beginning of line
14 055F    81    EA                    AJMP   BLINAD         ; Build line address and return
```

```

1          ;:          PLF - perform line feed
2          ;
3          ;
4          ;          *PLF* moves the cursor down one line in the display memory.
5          ;          Cursor never moves out of the top fixed region. The display
6          ;          is scrolled up one line if cursor is just above the bottom
7          ;          fixed region. The cursor never moves into the 25th line.
8          ;
9          ;          ENTRY   none
10         ;
11         ;          EXIT    none
12         ;
13         ;          USES   all
14
15
16 0561    E5      12          PLF:      MOV     A, YPOS      ; Get Y-position
17 0563    B4      18      01      CJNE   A, #MAXLINE, PLF1 ; IF on 25th line THEN
18 0566    22          RET          ; return
19 0567    04          PLF1:     INC     A          ; ELSE bump value
20 0568    B5      18      02      CJNE   A, BFIX, PLF2   ; IF bottom scroll region THEN
21 056B    80      04          SJMP   PLF3          ; scroll as usual
22 056D    B4      18      24      PLF2:     CJNE   A, #MAXLINE, PLFRET ; IF not 24th line THEN bump
23 0570    22          RET          ; ELSE return
24 0571    85      17      F0      PLF3:     MOV     B, TSCROLL ; Set up to clear
25 0574    12      0B96    LCALL  CLRLINE ; Top line of scroll region
26 0577    74      47          MOV     A, #LORDER ; Calculate pointers needed
27 0579    25      17          ADD     A, TSCROLL ; For adjusting -LORDER-
28 057B    F8          MOV     PTR1, A
29 057C    F9          MOV     PTR2, A
30 057D    09          INC     PTR2
31 057E    86      05          MOV     TEMP1, @PTR1 ; Save top value for later
32 0580    E5      18          MOV     A, BFIX
33 0582    95      17          SUBB   A, TSCROLL ; Calc number of times
34 0584    14          DEC     A ; To go through loop
35 0585    FC          MOV     WORK, A
36 0586    C2      AF          CLR     EA ; Stops any blinking
37
38 0588    E7          PLFLP:    MOV     A, @PTR2 ; REPEAT
39 0589    F6          MOV     @PTR1, A ; switch entries
40 058A    08          INC     PTR1
41 058B    09          INC     PTR2 ; bump pointers
42 058C    DC      FA          DJNZ  WORK, PLFLP ; UNTIL work = 0
43
44 058E    A6      05          MOV     @PTR1, TEMP1 ; Place last value
45 0590    D2      AF          SETB  EA ; Enable interrupts
46 0592    E5      12          MOV     A, YPOS ; Keep same position
47 0594    F5      12          PLFRET: MOV     YPOS, A ; Update -YPOS-
48 0596    81      EA          AJMP  BLINAD ; Build line address and return
  
```

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

```
1          ;:          CLRRTABS - clear all tabs
2          ;
3          ;          *CLRRTABS* 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          CLRRTABS:  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          CLRRTB:   MOV    @PTR1, A              ; REPEAT zero byte
18 059E    08          INC    PTR1                        ; bump pointer
19 059F    DA      FC          DJNZ  INDX1, CLRRTB      ; UNTIL index = 0
20 05A1    22          RET
```

```

1          ; ;          SETTAB - change tab settings
2          ; ;
3          ; ;          *SETTAB* sets or clears a tab position depends on the
4          ; ;          values passed to it.
5          ; ;
6          ; ;
7          ; ;          ENTRY  (A) = tab position from 0..79
8          ; ;          *FO* = value to place into tab position
9          ; ;
10         ; ;          EXIT   none
11        ; ;
12        ; ;          USES   A, B, PTR1, WORK, INDX1
13
14
15 05A2    F5    F0          SETTAB:    MOV    B, A          ; Save position to set/clear
16 05A4    54    F8          ANL    A, #11111000B ; Make it a multiple of 8
17 05A6    FC          MOV    WORK, A      ; Save as a start for counting
18 05A7    03          RR    A          ; Divide by 8
19 05A8    03          RR    A
20 05A9    03          RR    A
21 05AA    24    29          ADD    A, #TABTAB ; Add offset to tab table
22 05AC    F8          MOV    PTR1, A      ; Init pointer
23 05AD    E6          MOV    A, @PTR1 ; Get tab byte
24 05AE    7A    08          MOV    INDX1, #8 ; Loop through all 8 bits
25 05B0    C5    F0          STB1:    XCH    A, B          ; Get count and save byte
26 05B2    B5    04    04    CJNE   A, WORKA, STB2 ; If position if found
27 05B5    A2    D5          MOV    C, FO      ; Get value desired
28 05B7    92    F7          MOV    B.7, C    ; Place into tab byte
29 05B9    C5    F0          STB2:    XCH    A, B          ; Change back
30 05BB    23          RL    A          ; Rotate to next tab position
31 05BC    0C          INC    WORK      ; Bump count
32 05BD    DA    F1          DJNZ  INDX1, STB1 ; Until all 8 bits done
33 05BF    F4          MOV    @PTR1, A    ; Replace the tab byte
34 05C0    22          RET
  
```

```

1          ;:          TAB - horizontal tab routine
2          ;
3          ;
4          ;          *TAB* gets current position and then does a search for the
5          ;          next tab stop.  If none is found then the default is the last
6          ;          column position.
7          ;
8          ;
9          ;          ENTRY   none
10         ;
11         ;
12         ;          EXIT   none
13         ;
14         ;          USES   A, B, PTR1, WORK, TEMP, INDX1/2
15 05C1    E5    11          TAB:    MOV     A, XPOS          ; Get current position
16 05C3    FD          MOV     TEMP, A          ; Save position to tab from
17 05C4    54    F8          ANL     A, #11111000B        ; Make a multiple of 8
18 05C6    FC          MOV     WORK, A        ; Save as a start for count
19 05C7    03          RR      A              ; Divide by 8
20 05C8    03          RR      A
21 05C9    03          RR      A
22 05CA    F5    F0          MOV     B, A              ; Save temporary byte offset
23 05CC    24    29          ADD     A, #TABTAB        ; Add offset to tab table
24 05CE    F8          MOV     PTR1, A        ; Initialize pointer
25 05CF    74    0A          MOV     A, #10
26 05D1    95    F0          SUBB   A, B              ; Calc # times needed in loop
27 05D3    FA          MOV     INDX1, A       ; Place into index
28 05D4    C2    D5          CLR     F0              ; F0 = hunting for current PoD
29
30 05D6    E6          TBL1:    MOV     A, @PTR1        ; REPEAT set tab byte
31 05D7    30    D5    02    JNB     F0, TB6        ; IF not looking for position
32 05DA    60    17          JZ     TB1          ; quick check to exit
33 05DC    7B    08          TB6:    MOV     INDX2, #8        ; look through the 8 bits
34
35 05DE    20    D5    09    TBL2:    JB      F0, TB3        ; IF looking for current
36 05E1    CD          XCH     A, TEMP        ; set original position
37 05E2    B5    04    02    CJNE   A, WORKA, TB4        ; IF equal to original
38 05E5    D2    D5          SETB   F0              ; flag it
39 05E7    CD          XCH     A, TEMP        ; set current position
40 05E8    80    03          Sjmp   TB2          ; ELSE looking for tab
41 05EA    20    E7    0F    TB3:    JB      ACC.7, TBR        ; IF tab set THEN exit
42 05ED    23          TB2:    RL      A              ; rotate to next
43 05EE    0C          INC     WORK          ; bump count
44 05EF    DB    ED          DJNZ   INDX2, TBL2        ; UNTIL bit count = 0
45
46 05F1    80    04          Sjmp   TB5          ; IF need to bump by eight
47 05F3    74    08          TB1:    MOV     A, #8
48 05F5    2C          ADD     A, WORK        ; bump count by eight
49 05F6    FC          MOV     WORK, A
50 05F7    08          TB5:    INC     PTR1          ; bump pointer to next byte
51 05F8    DA    DC          DJNZ   INDX1, TBL1        ; UNTIL table is done
52
53 05FA    7C    4F          MOV     WORK, #MAXCHAR ; Default value if not found
54
55 05FC    8C    11          TBR:    MOV     XPOS, WORK ; Set next tab position
56 05FE    91    EA          ACALL  BLINAD        ; Build new address
57 0600    21    A9          JMP     SPF          ; Skip any protected fields

```

```

1          ;;          BTAB - horizontal back tab routine
2          ;
3          ;          *BTAB* sets current position and then does a search for the
4          ;          previous tab stop. If none is found then the default is the
5          ;          first column position.
6          ;
7          ;
8          ;          ENTRY   none
9          ;
10         ;          EXIT    none
11         ;
12         ;          USES    A, B, PTR1, WORK, TEMP, INDX1/2
13         ;
14         ;
15 0602    E5    11          BTAB:    MOV     A, XPOS          ; Position to compare with
16 0604    60    19          ;          JZ     BTR           ; IF first column THEN exit
17 0606    14          ;          DEC    A             ; Decrement for compare
18 0607    75    11    00    ;          MOV    XPOS, #0        ; Start looking from first pos
19         ;
20 060A    85    11    F0    BTL:    MOV    B, XPOS          ; REPEAT save previous position
21 060D    C0    E0          ;          PUSH   ACC         ; save compare value
22 060F    C0    F0          ;          PUSH   B           ; save previous position
23 0611    B1    C1          ;          ACALL  TAB         ; tab to next stop
24 0613    D0    F0          ;          POP    B           ; restore previous position
25 0615    D0    E0          ;          POP    ACC        ; restore compare value
26 0617    B5    11    00    ;          CJNE  A, XPOS, BTL1
27 061A    50    EE          BTL1:   JNC    BTL           ; UNTIL next >= current pos
28         ;
29 061C    85    F0    11    ;          MOV    XPOS, B         ; Get previous position
30 061F    81    EA          BTR:    JMP    BLINAD        ; Calc new address and return
  
```

```

1      ; ;
2      ;
3      ; ;
4      ; ;
5      ; ;
6      ; ;
7      ; ;
8      ; ;
9      ; ;
10     ; ;
11     ; ;
12     ; ;
13 0621 D2 D5 HTS: SETB F0 ; Set tab
14 0623 E5 11 HTS1: MOV A, XPOS ; Get position
15 0625 A1 A2 AJMP SETTAB ; Execute and return
16
17
18
19
20     ; ;
21     ; ;
22     ; ;
23     ; ;
24     ; ;
25     ; ;
26     ; ;
27     ; ;
28     ; ;
29     ; ;
30     ; ;
31     ; ;
32 0627 C2 D5 HTC: CLR F0 ; Clear tab
33 0629 80 F8 SJMP HTS1 ; Execute and return
34
35
36
37
38     ; ;
39     ; ;
40     ; ;
41     ; ;
42     ; ;
43     ; ;
44     ; ;
45     ; ;
46     ; ;
47     ; ;
48     ; ;
49     ; ;
50 062B 12 09D1 ATBC: LCALL GNP ; Get parameter
51 062E 60 F7 JZ HTC ; IF 0 THEN hor tab clear
52 0630 B4 03 02 CJNE A, #3, AT2 ; ELSE IF 3 THEN
53 0633 A1 98 AJMP CLRTABS ; clear all tabs
54 0635 22 AT2: RET ; ELSE return

```



```

1          ;;          HTAB - Hazeltine tab routine
2          ;;
3          ;;          *HTAB* performs the Hazeltine tab function.
4          ;;
5          ;;
6          ;;          ENTRY   none
7          ;;
8          ;;          EXIT    none
9          ;;
10         ;;          USES   all
11
12
13 0636    C2    D5          HTAB:    CLR    F0          ; Flag = FALSE
14 0638    E5    11          MOV    A, XPOS
15 063A    25    12          ADD    A, YPOS          ; Check for home position
16
17 063C    70    02          JNZ   HTO          ; IF home position THEN
18 063E    D2    D5          SETB  F0          ;   Flag = TRUE
19
20 0640    C0    11          HTO:    PUSH   XPOS          ; Save current position
21 0642    C0    12          PUSH   YPOS
22 0644    75    3F    80    MOV    PFIELD, #10000000B ; Make normal protected
23 0647    31    A9          ACALL SPF          ; Skip over foreground
24 0649    E5    11          MOV    A, XPOS
25 064B    25    12          ADD    A, YPOS          ; Check for home position
26
27 064D    70    03          JNZ   HT1          ; IF home position AND
28 064F    30    D5    09    JNB   F0, HT2          ; IF that is old pos THEN
29 0652    75    3F    08    HT1:    MOV    PFIELD, #00001000B ; make half protected
30 0655    31    A9          ACALL SPF          ; skip over background
31 0657    E5    11          MOV    A, XPOS
32 0659    25    12          ADD    A, YPOS          ; check for home position
33
34 065B    75    3F    00    HT2:    MOV    PFIELD, #0          ; Clear Protected fields
35
36 065E    60    05          JZ    HT3          ; IF position not home THEN
37 0660    D0    E0          POP   ACC          ; remove old position
38 0662    D0    E0          POP   ACC          ; and return
39 0664    22          RET
40
41 0665    D0    12          HT3:    POP   YPOS          ; ELSE
42 0667    D0    11          POP   XPOS          ; use previous position
43 0669    81    EA          AJMP  BLINAD        ; update and return

```

```

1          ;:          RING - rings bell
2          ;
3          ;
4          ;          *RING* rings the bell on the keyboard. If the input FIFO
5          ;          has a bell as the next character it is dropped. This is
6          ;          so that the terminal can keep up with multiple bells and
7          ;          communicate to the keyboard.
8          ;
9          ;
10         ;          ENTRY   none
11         ;
12         ;          EXIT    none
13         ;
14         ;          USES    none
15
16 066B    74      81          RING:      MOV     A, #KBC_BELL
17 066D    51      6D          ACALL    PCKB          ; Output command to ring bell
18
19 066F    51      1B          RGL:      ACALL    LCIF          ; REPEAT set next input char
20 0671    40      07          JC      RGR          ; IF no character THEN exit
21 0673    B4      07      04          CJNE   A, #BELL, RGR  ; IF character = bell THEN
22 0676    31      F6          ACALL    FCIF          ; fetch character and drop
23 0678    80      F5          SJMP   RGL          ; UNTIL char <> bell
24 067A    22          RGR:      RET

```

```

1          ;;          ESCCODE - escape code received
2          ;;
3          ;;          *ESCCODE* determines which mode the terminal is in and
4          ;;          sets the dispatch address to the proper routine.
5          ;;
6          ;;
7          ;;          ENTRY none
8          ;;
9          ;;          EXIT none
10         ;;
11         ;;          USES none
12         ;;
13         ;;
14 067B    12    102D    ESCCODE:    LCALL    R2_ESC          ; Escape hook
15
16 067E    75    16     97          MOV     DSADRL, #LOW (ANSIESC) ; ANSI dispatch address
17 0681    75    15     06          MOV     DSADRH, #HIGH (ANSIESC)
18
19         IFC     NE, HIGH(ANSIESC)-HIGH(ZDSESC)
20         ERROR  ;'ZDSESC' IS NOT THE SAME AS 'ANSIESC'
21         ENDC
22         IFC     NE, HIGH(ANSIESC)-HIGH(ADM3ESC)
23         ERROR  ;'ADM3ESC' IS NOT THE SAME AS 'ANSIESC'
24         ENDC
25         IFC     NE, HIGH(ANSIESC)-HIGH(HAZESC)
26         ERROR  ;'HAZESC' IS NOT THE SAME AS 'ANSIESC'
27         ENDC
28
29 0684    20    40     0F          JB     ANSI, ECRET          ; IF not ANSI mode THEN
30 0687    75    16     B8          MOV     DSADRL, #LOW (ZDSESC) ; ZDS dispatch address
31
32 068A    20    41     09          JB     ZDS, ECRET          ; IF not ZDS mode THEN
33 068D    75    16     A7          MOV     DSADRL, #LOW (ADM3ESC) ; ADM 3 dispatch address
34
35 0690    20    42     03          JB     ADM3, ECRET        ; IF not ADM 3 mode THEN
36 0693    75    16     B0          MOV     DSADRL, #LOW (HAZESC) ; Hazeltine dispatch address
37 0696    22          ESCRET:    RET          ; return

```

```

1      ;: ANSIESC - ANSI escape parser
2      ;
3      ; *ANSIESC* parses the first character following an ESC when
4      ; in the ANSI mode.
5      ;
6      ;
7      ; ENTRY (A) = character to parse
8      ;
9      ; EXIT none
10     ;
11     ; USES all
12
13
14 0697 C2 18 ANSIESC: CLR PSDF ; Flag first time through *PSD*
15 0699 C2 19 CLR GNPFL ; Flag first time through *GNP*
16 069B 75 F0 0C MOV B, #ANSITL ; ANSI mode table parameters
17 069E 90 0798 MOV DPTR, #ANSIT
18
19 06A1 12 082F AESC1: LCALL SETNORM ; Restore dispatch address
20 06A4 02 0836 LJMP STJMP ; Jump to any routine
21
22
23
24
25     ;: ADM3ESC - ADM 3 escape parser
26     ;
27     ; *ADM3ESC* parses the character received to do the proper
28     ; routine, if any, for ADM 3 mode.
29     ;
30     ;
31     ; ENTRY (A) = first character after ESC sequence
32     ;
33     ; EXIT none
34     ;
35     ; USES all
36
37
38 06A7 12 082F ADM3ESC: LCALL SETNORM ; Restore dispatch address
39
40 06AA B4 3D 31 CJNE A, #'=', ZE2 ; IF not '=' THEN return
41 06AD 02 0CD6 LJMP CUA ; ELSE same as cursor address

```

```
1          ;; HAZESC - Hazeltine 1500 escape (tilde) parser
2          ;
3          ; *HAZESC* parses the character received to do the proper
4          ; routine, if any, for Hazeltine 1500 mode.
5          ;
6          ;
7          ; ENTRY (A) = first character after ESC (tilde) sequence
8          ;
9          ; EXIT none
10         ;
11         ; USES all
12         ;
13         ;
14 06B0 75 F0 10 HAZESC: MOV B, #HAZTL ; Hazeltine table parameters
15 06B3 90 0768 MOV DPTR, #HAZT
16 06B6 80 E9 SJMP AESC1 ; Look through table
```

```

1      ;
2      ;
3      ;
4      ;
5      ;
6      ;
7      ;
8      ;
9      ;
10     ;
11     ;
12     ;
13     ;
14 06B8 12 082F      ZDSESC:      LCALL  SETNORM      ; Restore dispatch address
15     ;
16 06BB B4 23 03      CJNE   A, #'#', ZE1A      ; IF transmit base THEN
17 06BE 02 1036      JMP    R2_TP             ; do it
18 06C1 B4 2D 02      ZE1A:  CJNE   A, #'-', ZE1B      ; IF back tab THEN
19 06C4 C1 02        AJMP   BTAB             ; do it
20 06C6 B4 2E 03      ZE1B:  CJNE   A, #'./', ZE1C     ; IF tab functions THEN
21 06C9 02 0ED3      JMP    ZTAB             ; do it
22     ;
23 06CC C3           ZE1C:      CLR    C               ; Set up to
24 06CD 94 3C        SUBB   A, #'<'         ; Normalize into range
25 06CF 40 0D        JC     ZE2             ; IF out of range THEN return
26     ;
27 06D1 23           RL     A               ; ELSE mult by 2
28 06D2 FD           MOV   TEMP, A          ; save value
29 06D3 24 0A        ADD   A, #10           ; add code offset to low
30 06D5 83           MOV   A, @A+PC         ; set low byte
31 06D6 C0 E0        PUSH  ACC             ; put onto stack
32 06D8 ED           MOV   A, TEMP         ; set value back
33 06D9 24 03        ADD   A, #3           ; add code offset to high
34 06DB 83           MOV   A, @A+PC         ; set high byte
35 06DC C0 E0        PUSH  ACC             ; put onto stack
36 06DE 22           ZE2:      RET              ; Jump to address
37     ;
38     ;
39     ;
40     ;
      IFC   NE, ZDST-$
      ERROR ;TABLE 'ZDST' MUST FOLLOW
      ENDC

```

```

1          ; ; ZDST - ZDS escape table
2          ; ;
3          ; ; *ZDST* contains the second character in the range of
4          ; ; all escape sequences the Z-29 will respond to.
5          ; ;
6          ; ; Table entries are: Address of routine which performs the
7          ; ; requested operation in order of their ASCII sequence.
8          ; ;
9          ; ;
10         06DF 0B03          ZDST: DW EAM          ; < enter ANSI mode
11         06E1 0B4E          DW EKAM         ; = enter alt keypad mode
12         06E3 0B51          DW XFAM         ; > exit alt keypad mode
13         06E5 0767          DW UNDEF        ; ? undefined
14         06E7 0B54          DW EICM         ; @ enter insert char mode
15         06E9 0C8D          DW CUP          ; A cursor up
16         06EB 0CA8          DW CDN          ; B cursor down
17         06ED 0C64          DW CRT          ; C cursor right
18         06EF 0C3D          DW CLFT         ; D cursor left
19         06F1 0BAC          DW CLRS         ; E clear entire display
20         06F3 0F39          DW EGM          ; F enter graphic mode
21         06F5 0F3C          DW XGM          ; G exit graphic mode
22         06F7 0C36          DW SCH          ; H set cursor home
23         06F9 0518          DW PRLF         ; I reverse index
24         06FB 0C08          DW EED          ; J erase cursor to EOS
25         06FD 0BE0          DW EOL          ; K erase cursor to EOL
26         06FF 0D47          DW PIL          ; L perform insert line
27         0701 0D88          DW PDL          ; M perform delete line
28         0703 0DF4          DW PDC          ; N delete char
29         0705 0B58          DW XICM         ; O exit insert char mode
30         0707 0767          DW UNDEF        ; P undefined (blue)
31         0709 0767          DW UNDEF        ; Q undefined (red)
32         070B 0767          DW UNDEF        ; R undefined (gray)
33         070D 0767          DW UNDEF        ; S undefined (F1)
34         070F 0767          DW UNDEF        ; T undefined (F2)
35         0711 0767          DW UNDEF        ; U undefined (F3)
36         0713 0767          DW UNDEF        ; V undefined (F4)
37         0715 0767          DW UNDEF        ; W undefined (F5)
38         0717 0767          DW UNDEF        ; X undefined
39         0719 0CD6          DW CUA          ; Y VT52 cursor addr. (param)
40         071B 0EAB          DW IDT          ; Z VT52 identify
41         071D 0B3C          DW EHSM         ; [ Enter hold screen mode
42         071F 0B43          DW XHSM         ; \ Exit hold screen mode
43         0721 1039          DW R2_T25L      ; ] xmit 25th line
44         0723 1033          DW R2_TL        ; ^ transmit line
45         0725 1030          DW R2_TC        ; _ transmit character
46         0727 103C          DW R2_PRNT      ; ` print page
47         0729 0767          DW UNDEF        ; a undefined
48         072B 0BF1          DW EBD          ; b erase display to cursor
49         072D 0767          DW UNDEF        ; c undefined
50         072F 0767          DW UNDEF        ; d undefined
51         0731 0767          DW UNDEF        ; e undefined
52         0733 0767          DW UNDEF        ; f undefined
53         0735 0767          DW UNDEF        ; g undefined
54         0737 0767          DW UNDEF        ; h undefined
55         0739 0EB3          DW IDTT         ; i identify terminal type
56         073B 0EE8          DW SCP          ; j save cursor position
57         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
26
27
28
29
30
31
32
33
34
35
36

```

; ; UNDEF - undefined escape sequence
;
; *UNDEF* is used in ZDS escape parsing for undefined entries
;
;
; ENTRY none
;
; EXIT none
;
; USES none

```

```

37 0767 22 UNDEF: RET ; Undefined ESC sequence

```


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

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

```

1          ;;          ANSIT - ANSI escape table
2          ;
3          ;          *ANSIT* contains the second character of the ANSI escape table
4          ;
5          ;          Table entries are: The second character of the ANSI escape
6          ;          sequence followed by the address of the routine which performs
7          ;          the requested function.
8
9
10         0798          ANSIT          EQU          $
11
12 0798     5B          DB          '[/'          ; ESC [
13 0799     07BC       DW          ELB          ; Escape left bracket
14
15 079B     28          DB          '('/'          ; ESC (
16 079C     0F3F       DW          GODES       ; G0 designator
17
18 079E     29          DB          ')'/'          ; ESC )
19 079F     0F47       DW          GIDES       ; G1 designator
20
21 07A1     37          DB          '['/'          ; ESC [
22 07A2     0EE8       DW          SCP          ; Save cursor position
23
24 07A4     38          DB          '['/'          ; ESC 8
25 07A5     0EF4       DW          USCP        ; Unsave cursor position
26
27 07A7     3D          DB          '='/'          ; ESC =
28 07A8     0B4E       DW          EKAM        ; Enter keypad alternate mode
29
30 07AA     3E          DB          '>'/'          ; ESC >
31 07AB     0B51       DW          XKAM        ; Exit keypad alternate mode
32
33 07AD     4D          DB          'M'/'          ; ESC M
34 07AE     0518       DW          PRLF        ; Reverse index (rev line feed)
35
36 07B0     63          DB          'c'/'          ; ESC c
37 07B1     0AF6       DW          ARAMP       ; ANSI reset to power up
38
39 07B3     44          DB          'D'/'          ; ESC D
40 07B4     0561       DW          PLF          ; Perform line feed
41
42 07B6     48          DB          'H'/'          ; ESC H
43 07B7     0621       DW          HTS         ; Horizontal tab set
44
45 07B9     23          DB          '#'/'          ; ESC # 7
46 07BA     0ADF       DW          APRNT       ; Print page
47
48         000C          ANSITL          EQU          ($-ANSIT)/3          ; Table length

```

```

1          ;|          ELB - escape left bracket
2          ;
3          ;          *ELB* is a continuation of the escape sequence processing
4          ;          for ANSI escape sequences. The final character of the
5          ;          sequence is decoded with or without a preceding parameter
6          ;          string and control is passed on to the associated routine.
7          ;
8          ;
9          ;          ENTRY none
10         ;
11         ;          EXIT to requested routine
12         ;
13         ;          USES all
14
15
16 07BC    12    082A    ELB:    LCALL    SETDISP    ; Set dispatch and return
17
18 07BF    12    0971    LCALL    PSD        ; Get parameter string
19 07C2    50    78      JNC      STRET      ; IF not done THEN return
20 07C4    12    082F    LCALL    SETNORM    ; ELSE restore dispatch address
21 07C7    75    F0      MOV     R, #ELBTL   ; (R) = table length
22 07CA    90    07D0    MOV     DPTR, #ELBT ; (DPTR) = ESC [ table
23 07CD    02    0836    LJMPL  STJMP      ; Jump to any routine

```

```

1          ;;          ELBT - escape left bracket table
2          ;
3          ;
4          ;          *ELBT* contains the third and/or final character of the
5          ;          ANSI escape sequence for ESC [ Pn F
6          ;
7          ;          Table entries are: Final character of ESC [ Pn F sequence
8          ;          followed by address of the routine which performs the
9          ;          requested function.
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	EQU	(\$-ELBT)/3	; Table length

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

```

1          ;;          SETDISP - set dispatch address
2          ;
3          ;
4          ;          *SETDISP* sets the dispatch address to the address just
5          ;          following the call to this routine. This routine then
6          ;          executes a -RET-. Be carefull with this. A call to this
7          ;          routine does not return to back to the routine that called
8          ;          this one.
9          ;
10         ;          ENTRY   none
11         ;
12         ;          EXIT    none
13         ;
14         ;          USES    none
15         ;
16         ;
17 082A    D0    15          SETDISP:   POP    DSADRH          ; Get high byte of data
18 082C    D0    16          POP    DSADRL          ; Get low byte of data
19 082E    22          RET
20
21
22
23
24         ;;          SETNORM - set dispatch normal
25         ;
26         ;
27         ;          *SETNORM* sets the dispatch address for processing characters
28         ;          back to the 'normal' address.
29         ;
30         ;          ENTRY   none
31         ;
32         ;          EXIT    none
33         ;
34         ;          USES    none
35         ;
36         ;
37 082F    75    16    87          SETNORM:  MOV    DSADRL, #LOW (NORM)  ; Restore dispatch address
38 0832    75    15    00          MOV    DSADRH, #HIGH (NORM)
39 0835    22          RET

```

```

1          ;:          STJMP - search table and jump
2          ;          STCALL - Jump to routine in (DPTR)
3          ;
4          ;
5          ;          *STJMP* does a search table operation. If the data was found
6          ;          it does a jump to the address in the (DPTR), otherwise it
7          ;          does a return.
8          ;
9          ;          *STCALL only does a jump to the address in the (DPTR).
10         ;
11         ;
12         ;          ENTRY... (DPTR) = address to jump to
13         ;
14         ;          EXIT... none
15         ;
16         ;          USES... A
17
18 0836    31    F7          STJMP:    ACALL  XSTAB          ; Search the table
19 0838    40    02          JC        STRET          ; IF carry is set THEN return
20
21 083A    E4          STCALL:    CLR        A
22 083B    73          JMP        @A+DPTR          ; ELSE jump to address
23
24 083C    22          STRET:    RET

```

; Return

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

```

1          ;;          ASM - ANSI set mode sequence
2          ;
3          ;          *ASM* set the specified mode(s).
4          ;
5          ;
6          ;          ENTRY   none
7          ;
8          ;          EXIT    none
9          ;
10         ;          USES   all
11
12
13 083D    31    D1          ASM:    ACALL  GNP          ; Get next (or first) parameter
14 083F    40    0E          JC      ASMR          ; IF parameter THEN
15 0841    75    F0    05    MOV    B, #ASMTL      ; (B) = table length
16 0844    90    0850      MOV    DPTR, #ASMT      ; (DPTR) = "ESC [ Pn h" table
17 0847    31    F7          ACALL  XSTAB        ; search table
18 0849    40    04          JC      ASMR          ; IF found THEN
19 084B    11    3A          ACALL  STCALL       ; jump to routine
20 084D    80    EE          SJMP  ASM          ; and keep going
21
22 084F    22          ASMR:    RET
23
24
25
26
27         ;;          ASMT - ANSI set mode sequence table
28         ;
29         ;          *ASMT* contains valid parameters for "ESC [ Pn h" sequence.
30
31
32         0850          ASMT    EQU    $
33
34 0850    01          DB      1          ; ESC [ 1 h
35 0851    0B6E      DW      EGATM       ; Enter guarded area transfer
36
37 0853    02          DB      2          ; ESC [ 2 h
38 0854    0FB9      DW      DK1          ; Disable keyboard input
39
40 0856    04          DB      4          ; ESC [ 4 h
41 0857    0B54      DW      EICM        ; Enter insert character mode
42
43 0859    06          DB      6          ; ESC [ 6 h
44 085A    0B74      DW      EERM        ; Enter erasure mode
45
46 085C    14          DB      20         ; ESC [ 20 h
47 085D    0B62      DW      EACR        ; Enable auto CR on LF
48
49         0005          ASMTL   EQU    ($-ASMT)/3      ; Table length

```

```

1      ;:          ARM - ANSI reset mode sequence
2      ;
3      ;          *ARM* reset the specified mode(s).
4      ;
5      ;
6      ;          ENTRY   none
7      ;
8      ;          EXIT    none
9      ;
10     ;          USES   all
11     ;
12     ;
13 085F  31    D1      ARM:      ACALL  GNP          ; REPEAT set parameter
14 0861  40    EC          JC      ASMR          ; IF no parameter THEN return
15 0863  75    F0    05      MOV    B, #ARMTL    ; (B) = table length
16 0866  90    0871      MOV    DPTR, #ARMT    ; (DPTR) = "ESC [ Pn i" table
17 0869  31    F7      ACALL  XSTAB          ; search table
18 086B  40    E2          JC      ASMR          ; IF not found THEN return
19 086D  11    3A      ACALL  STCALL         ; call routine
20 086F  80    EE          SJMP   ARM          ; UNTIL forever
21
22
23
24
25     ;:          ARMT - ANSI reset mode sequence table
26     ;
27     ;          *ARMT* contains valid parameters for "ESC [ Pn i" sequence
28     ;
29
30     0871      ARMT      EQU      $
31
32 0871  01          DB      1          ; ESC [ 1 ]
33 0872  0B71      DW      XGATM        ; Exit guarded area transfer
34
35 0874  02          DB      2          ; ESC [ 2 ]
36 0875  0FB3      DW      EKI         ; Enable keyboard input
37
38 0877  04          DB      4          ; ESC [ 4 ]
39 0878  0B58      DW      XICM        ; Exit insert character mode
40
41 087A  06          DB      6          ; ESC [ 6 ]
42 087B  0B77      DW      XERM        ; Exit erasure mode
43
44 087D  14          DB      20         ; ESC [ 20 ]
45 087E  0B65      DW      XACR        ; Disable auto CR on LF
46
47     0005      ARMTL     EQU      ($-ARMT)/3 ; Table length

```

```

1          ;;          AIM - ANSI mode #1
2          ;
3          ;          *AIM* inputs the parameter string and final character for
4          ;          the ESC [ ? Pn F sequence.
5          ;
6          ;
7          ;          ENTRY none
8          ;
9          ;          EXIT none
10         ;
11         ;          USES all
12         ;
13         ;
14 0880 C2 18          AIM: CLR PSDF          ; Clear *PSD* flag
15 0882 11 2A        ACALL SETDISP        ; Set dispatch and return
16         ;
17 0884 31 71        ACALL PSD            ; Get parameter string
18 0886 50 C7        JNC ASMR             ; IF not done THEN return
19 0888 11 2F        ACALL SETNORM        ; ELSE restore dispatch
20 088A 75 F0 02     MOV B, #EQMTL        ; (B) = table length
21 088D 70 0892     MOV DPTR, #EQMT      ; (DPTR) = ESC [ ? table
22 0890 01 36        AJMP STJMP          ; Jump to any routine
23         ;
24         ;
25         ;
26         ;
27         ;          EQMT - escape [ ? Pn Ps table
28         ;
29         ;          *EQMT* contains the fourth and/or final character
30         ;
31         ;
32         0892          EQMT EQU $
33         ;
34 0892 68           DB 'h'              ; ESC [ ? Pn h
35 0893 08B0        DW AISM              ; ANSI set mode #1
36         ;
37 0895 6C           DB '<|'            ; ESC [ ? Pn |
38 0896 08CC        DW AIRM              ; ANSI reset mode
39         ;
40         0002          EQMTL EQU ($-EQMT)/3 ; Table length

```

```

1      ;
2      ;
3      ;
4      ;
5      ;
6      ;
7      ;
8      ;
9      ;
10     ;
11     ;
12     ;
13     ;
14 0898 C2 18 A2M: CLR PSDF ; Clear *PSD* flag
15 089A 11 2A ACALL SETDISP ; Set dispatch and return
16     ;
17 089C 31 71 ACALL PSD ; Get parameter string
18 089E 50 22 JNC AISR ; IF not done THEN return
19 08A0 11 2F ACALL SETNORM ; ELSE restore dispatch
20 08A2 75 F0 02 MOV B, #EGTTL ; (B) = table length
21 08A5 90 08AA MOV DPTR, #EGTT ; (DPTR) = ESC [ > table
22 08A8 01 36 AJMP STJMP ; Jump to any routine
23     ;
24     ;
25     ;
26     ;
27     ;
28     ;
29     ;
30     ;
31     ;
32     08AA EGTT EQU $
33     ;
34 08AA 68 DB '<h>' ; ESC [ > Pn h
35 08AB 08FF DW A2SM ; ANSI set mode #1
36     ;
37 08AD 6C DB '<1>' ; ESC [ > Pn 1
38 08AE 0944 DW A2RM ; ANSI reset mode
39     ;
40     0002 EGTTL EQU ($-EGTT)/3 ; Table length

```

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

```

1          ; ; AISM - ANSI mode #1 set mode sequence
2          ;
3          ; *AISM* set the specified mode(s).
4          ;
5          ;
6          ; ENTRY none
7          ;
8          ; EXIT none
9          ;
10         ; USES A, B, DPTR
11
12
13 08B0 31 D1 AISM: ACALL GNP ; Get next (or first) parameter
14 08B2 40 0E JC AISR ; IF parameter THEN
15 08B4 75 F0 03 MOV B, #A1STL ; (B) = table length
16 08B7 90 08C3 MOV DPTR, #A1ST ; (DPTR) = ESC [ ? Pn h table
17 08BA 31 F7 ACALL XSTAB ; Search table
18 08BC 40 04 JC AISR ; IF found THEN
19 08BE 11 3A ACALL STCALL ; Jump to routine
20 08C0 80 EE SUMP AISM ; and keep going
21
22 08C2 22 AISR: RET
23
24
25
26
27         ; ; A1ST - ANSI mode #1 set mode table
28         ;
29         ; *A1SMT* contains valid parameters for ESC [ ? Pn h sequence.
30
31
32         08C3 A1ST EQU $
33
34 08C3 02 DB 2 ; ESC [ ? 2 h
35 08C4 0B07 DW EZM ; Enter ZDS mode
36
37 08C6 03 DB 3 ; ESC [ ? 3 h
38 08C7 0BAC DW CLRS ; Enter 132 column mode
39
40 08C9 07 DB 7 ; ESC [ ? 7 h
41 08CA 0AED DW WEOL ; Wrap at end of line
42
43         0003 A1STL EQU ($-A1ST)/3 ; Table length

```

```

1          ;: AIRM - ANSI mode #1 reset mode sequence
2          ;
3          ; *AIRM* resets the specified mode(s).
4          ;
5          ;
6          ; ENTRY none
7          ;
8          ; EXIT none
9          ;
10         ; USES A, B, DPTR
11         ;
12         ;
13 08CC    31    D1          AIRM: ACALL GNP          ; REPEAT set parameter
14 08CE    40    F2          JC A1SR          ; IF no parameter THEN exit
15 08D0    75    F0    03   MOV B, #A1RTL ; (B) = table length
16 08D3    90    08DE      MOV DPTR, #A1RT ; (DPTR) = ESC [ ? Pn ] table
17 08D6    31    F7          ACALL XSTAB      ; Search table
18 08D8    40    E8          JC A1SR          ; IF found THEN
19 08DA    11    3A          ACALL STCALL    ; Jump to routine
20 08DC    80    EE          SJMP AIRM      ; UNTIL forever
21
22
23
24
25         ;: A1RT - ANSI mode #1 reset mode table
26         ;
27         ; *A1RT* contains valid parameters for ESC [ ? Pn ] sequence.
28
29
30         08DE          A1RT EQU $
31
32 08DE    02          DB 2          ; ESC [ ? 2 ]
33 08DF    0B07      DW EZM        ; Enter VT52 mode
34
35 08E1    03          DB 3          ; ESC [ ? 3 ]
36 08E2    0BAC      DW CLRS      ; Enter 80 column mode
37
38 08E4    07          DB 7          ; ESC [ ? 7 ]
39 08E5    0B00      DW DEOL      ; Discard past end of line
40
41         0003          A1RTL EQU ($-A1RT)/3 ; Table length

```

```

1          ;;          SMS - set mode sequence
2          ;
3          ;          *SMS* sets 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 08E7    11    2A          SMS:          ACALL  SETDISP          ; Set dispatch and return
15
16 08E9    11    2F          ACALL  SETNORM          ; Restore dispatch address
17 08EB    C3
18 08EC    94    31          CLR      C          ; Subtract offset
19 08EE    40    0E          SUBB   A, #1          ; Check for lower bounds
20 08F0    B4    09    00          JC     SMSRET
21 08F3    50    09          CJNE  A, #('9'-'0'), SMS1
22 08F5    04          JNC   SMSRET          ; Check for upper bounds
23 08F6    75    F0    09          INC   A          ; Bump up into range
24 08F9    90    0911          MOV   B, #SMSTL          ; (B) = table length
25 08FC    01    36          MOV   DPTR, #SMST          ; (DPTR) = ESC x Pn table
26
27 08FE    22          AJMP  STJMP          ; Jump to any routine
SMSRET:          RET

```

```

1      ;
2      ;
3      ;
4      ;
5      ;
6      ;
7      ;
8      ;
9      ;
10     ;
11     ;
12     ;
13 08FF 31    D1      A2SM:  ACALL  GNP      ; Get next (or first) parameter
14 0901 40    FB      JC      SMSRET    ; IF parameter THEN
15 0903 75    F0      MOV     B, #SMSTL  ; (B) = table length
16 0906 90    0911   MOV     DPTR, #SMST  ; (DPTR) = ESC [ > Pn h table
17 0909 31    F7      ACALL  XSTAB    ; Search table
18 090B 40    F1      JC      SMSRET    ; IF found THEN
19 090D 11    3A      ACALL  STCALL   ; Jump to routine
20 090F 80    EE      JMP     A2SM      ; and keep going
21
22
23
24
25     ;
26     ;
27     ;
28     ;
29     ;
30
31     0911      SMST      EQU      $
32
33 0911 01      DB      1      ; ESC x 1  ESC [ > 1 h
34 0912 0B0B   DW      E25L    ; Enable 25th line
35
36 0914 02      DB      2      ; ESC x 2  ESC [ > 2 h
37 0915 0B2D   DW      NKC     ; No keyboard click
38
39 0917 03      DB      3      ; ESC x 3  ESC [ > 3 h
40 0918 0B3C   DW      EHSM    ; Enter hold screen mode
41
42 091A 04      DB      4      ; ESC x 4  ESC [ > 4 h
43 091B 0B33   DW      SBC     ; Set "block" cursor
44
45 091D 05      DB      5      ; ESC x 5  ESC [ > 5 h
46 091E 0B5F   DW      DC      ; Disable cursor
47
48 0920 06      DB      6      ; ESC x 6  ESC [ > 6 h
49 0921 0B48   DW      EKSM    ; Enter keypad shifted mode
50
51 0923 07      DB      7      ; ESC x 7  ESC [ > 7 h
52 0924 0B4E   DW      EKAM    ; Enter keypad alternate mode
53
54 0926 08      DB      8      ; ESC x 8  ESC [ > 8 h
55 0927 0B68   DW      EALF    ; Enable auto LF on CR
56
57 0929 09      DB      9      ; ESC x 9  ESC [ > 9 h

```


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

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          SUBB   A, #'1'          ; Subtract offset
19 0933    40          JC      RMSRET          ; Check for lower bounds
20 0935    B4          09    00          CJNE  A, #'9'-'0', RMS1
21 0938    50          09          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 Pr table
25 0941    01          36          AJMP  STJMP          ; Jump to any routine
26         ;
27 0943    22          RMSRET:         RET

```

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

```

1          ;:          A2RM - ANSI mode #2 reset mode sequence
2          ;
3          ;          *A2RM* reset the specified mode(s).
4          ;
5          ;
6          ;          ENTRY   none
7          ;
8          ;          EXIT    none
9          ;
10         ;          USES   A, B, DPTR
11         ;
12         ;
13 0944    31    D1          A2RM:  ACALL  GNP          ; Get next (or first) parameter
14 0946    40    FB          JC      RMSRET         ; IF parameter THEN
15 0948    75    F0    09    MOV     B, #RMSTL        ; (B) = table length
16 094B    90    0956      MOV     DPTR, #RMST      ; (DPTR) = ESC [ > Pn l table
17 094E    31    F7          ACALL  XSTAB         ; Search table
18 0950    40    F1          JC      RMSRET         ; IF found THEN
19 0952    11    3A          ACALL  STCALL        ; Jump to routine
20 0954    80    EE          SJMP   A2RM         ; and keep going
21
22
23
24
25         ;:          RMST - ANSI reset mode sequence table
26         ;
27         ;          *RMST* contains valid parameters for ESC [ > Pn l sequence.
28         ;
29
30         0956          RMST      EQU     $
31
32 0956    01          DB      1          ; ESC y 1   ESC [ > 1 l
33 0957    0B15      DW      D25L        ; Disable 25th line
34
35 0959    02          DB      2          ; ESC y 2   ESC [ > 2 l
36 095A    0B26      DW      EKC        ; Enable keyboard click
37
38 095C    03          DB      3          ; ESC y 3   ESC [ > 3 l
39 095D    0B43      DW      XHSM       ; Exit hold screen mode
40
41 095F    04          DB      4          ; ESC y 4   ESC [ > 4 l
42 0960    0B37      DW      SUC        ; Set "underscore" cursor
43
44 0962    05          DB      5          ; ESC y 5   ESC [ > 5 l
45 0963    0B5C      DW      EC        ; Enable cursor
46
47 0965    06          DB      6          ; ESC y 6   ESC [ > 6 l
48 0966    0B4B      DW      XKSM       ; Exit keypad shifted mode
49
50 0968    07          DB      7          ; ESC y 7   ESC [ > 7 l
51 0969    0B51      DW      XKAM       ; Exit keypad alternate mode
52
53 096B    08          DB      8          ; ESC y 8   ESC [ > 8 l
54 096C    0B6B      DW      XALF       ; Disable auto LF on CR
55
56 096E    09          DB      9          ; ESC y 9   ESC [ > 9 l
57 096F    0B65      DW      XACR       ; Disable auto CR on LF

```

1
2

0009

RMSTL

EQU

($\$-RMST$)/3

; Table length

```

1          ;;          PSD - parameter string decoder
2          ;
3          ;          *PSD* inputs a parameter string of decimal numbers separated
4          ;          by a semicolon (to a maximum of 8) until the final character
5          ;          of the escape sequence (non decimal and not a semicolon) is
6          ;          inputed.
7          ;
8          ;
9          ;          ENTRY (A) = next parameter character
10         ;
11         ;          EXIT (A) = last character inputed (Ps usually)
12         ;          'C' = set if done decoding, clear otherwise
13         ;
14         ;          USES A, B, DPTR, TEMP
15
16
17 0971    20    18    0E    PSD:    JB    PSDF, PSDB          ; IF first time through THEN
18 0974    D2    18          SETB   PSDF          ; toggle flag
19 0976    75    1A    F8    MOV    PMADRL, #LOW (PMBUF) ; set parameter pointer
20 0979    75    19    57    MOV    PMADRH, #HIGH (PMBUF)
21 097C    75    1B    FF    MOV    PMNUM, #-1        ; init number of parameters
22
23 097F    75    1C    00    MOV    PMVALUE, #0      ; init parameter value
24
25 0982    B4    3A    00    PSDB:    CJNE   A, #'9'+1, PSDB1
26 0985    50    1B          JNC    PSDE          ; IF below upper range AND
27 0987    B4    30    00    PSDB1:  CJNE   A, #'0', PSDB2
28 098A    40    16          JC     PSDE          ; IF above lower range THEN
29
30 098C    94    30          SUBB   A, #'0'        ; remove ASCII offset
31 098E    C5    1C          XCH   A, PMVALUE    ; set previous value
32 0990    75    F0    0A    MOV    B, #10
33 0993    A4          MUL   AB          ; value := value * 10
34 0994    25    1C          ADD   A, PMVALUE    ; value := value + newparm
35 0996    F5    1C          MOV   PMVALUE, A    ; save new parameter value
36 0998    E5    1B          MOV   A, PMNUM
37 099A    B4    FF    03    CJNE   A, #-1, PSD1   ; IF parmnum = -1 THEN
38 099D    75    1B    00    MOV    PMNUM, #0     ; parmnum := 0
39 09A0    C3          PSD1:  CLR   C          ; flag = still parsing string
40 09A1    22          RET
41
42 09A2    85    1A    82    PSDE:  MOV    DPL, PMADRL   ; ELSE set parameter address
43 09A5    85    19    83    MOV    DPH, PMADRH
44 09A8    FD          MOV   TEMP, A      ; save character temporary
45 09A9    E5    1C          MOV   A, PMVALUE   ; set current parameter value
46 09AB    F0          MOVX  @DPTR, A      ; save in parameter buffer
47 09AC    A3          INC   DPTR        ; bump parameter address
48 09AD    E5    82          MOV   A, DPL       ; set low byte address to
49 09AF    B4    00    05    CJNE   A, #LOW(PMBUF+8), PSD2 ; IF overflow THEN
50 09B2    75    1B    08    MOV    PMNUM, #8    ; give max number of parameters
51 09B5    80    08          SJMP  PSD3        ; and skip
52 09B7    85    82    1A    PSD2:  MOV    PMADRL, DPL  ; ELSE replace
53 09BA    85    83    19    MOV    PMADRH, DPH  ; parameter address
54 09BD    05    1B          INC   PMNUM       ; bump number of parameters
55 09BF    ED          PSD3:  MOV   A, TEMP     ; restore original character
56 09C0    B4    3B    0C    CJNE   A, #';', PSDRET ; IF ';' THEN
57 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
```

```

1          ;;          GNP - set next parameter
2          ;
3          ;          *GNP* sets the next (or first) parameter from the parameter
4          ;          buffer.
5          ;
6          ;
7          ;          ENTRY none
8          ;
9          ;          EXIT (A) = next (or first) parameter, 0 if no more
10         ;          'C' = set if no parameters left, clear otherwise
11         ;
12         ;          USES A, DPTR
13
14
15 09D1 20 19 08 GNP: JB GNP, GNP1 ; IF first time since ESC THEN
16 09D4 D2 19 SETB GNP ; toggle flag
17 09D6 75 1A F8 MOV PMADRL, #LOW (PMBUF) ; set parameter address
18 09D9 75 19 57 MOV PMADRH, #HIGH (PMBUF)
19
20 09DC 85 1A 82 GNP1: MOV DPL, PMADRL ; Get pointer to next parameter
21 09DF 85 19 83 MOV DPH, PMADRH
22 09E2 E5 1B MOV A, PMNUM ; Get number of parameters 0..7
23 09E4 70 02 JNZ GNP2 ; IF no more or none at all
24 09E6 D3 SETB C ; 'C' = no more left
25 09E7 22 RET
26 09E8 E0 GNP2: MOVX A, @DPTR ; ELSE set parameter
27 09E9 A3 INC DPTR ; bump pointer
28 09EA 85 82 1A MOV PMADRL, DPL ; and save
29 09ED 85 83 19 MOV PMADRH, DPH
30 09F0 15 1B DEC PMNUM ; dec number of parameters
31 09F2 C3 CLR C ; 'C' = parameter available
32 09F3 22 RET
  
```

```
1      ;:      XBLINAD - build line address
2      ;
3      ;      *XBLINAD* is a jump to the routine *BLINAD*. It is used to
4      ;      save bytes so that routines may use a short jump to here
5      ;      instead of a long jump to the other routine.
6      ;
```

```
8 09F4 02 04EA XBLINAD: LJMPL BLINAD
```

```
13     ;:      XSTAB - search table
14     ;
15     ;      *XSTAB* is a jump to the routine *STAB*. It is used to
16     ;      save bytes so that routines may use a short jump to here
17     ;      instead of a long jump to the other routine.
18     ;
```

```
20 09F7 02 013A XSTAB: LJMPL STAB
```


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

```

1          ;;          APCA - ANSI perform cursor addressing
2          ;
3          ;
4          ;          *APCA* checks to see that the line and column values are legal
5          ;          and sets the cursor to the requested address.  If the values
6          ;          are out of range, the old line number remains and/or the cursor
7          ;          goes to the last column on the line.
8          ;
9          ;          ENTRY   none
10         ;
11         ;          EXIT    none
12         ;
13         ;          USES   all
14         ;
15
16 09FA 31  D1          APCA:      ACALL  GNP          ; Get next parameter
17 09FC 60  0A          ;          JZ      ACA2          ; IF non-zero THEN
18 09FE 14          ;          DEC     A          ; bring into range 0..24
19 09FF B4  18  04          ;          CJNE   A, #MAXLINE, ACA1 ; check range
20 0A02 30  15  05          ;          JNB    L25EN, ACA3 ; IF line 25 disbl'd THEN skip
21 0A05 D3          ;          SETB   C          ; trick ckeck below
22 0A06 50  02          ACA1:      JNC     ACA3          ; IF in range THEN
23 0A08 F5  12          ACA2:      MOV    YPOS, A          ; set Y-position
24
25 0A0A 31  D1          ACA3:      ACALL  GNP          ; Get next parameter
26 0A0C 60  08          ;          JZ      ACA5          ; IF non-zero THEN
27 0A0E 14          ;          DEC     A          ; bring into range 0..79
28 0A0F B4  50  00          ;          CJNE   A, #MAXCHAR+1, ACA3A ; check range
29 0A12 40  02          ACA3A:     JC     ACA5          ; IF not in range THEN
30 0A14 74  4F          ;          MOV    A, #MAXCHAR ; go to end of line
31 0A16 F5  11          ACA5:      MOV    XPOS, A          ; Set X-position
32 0A18 21  F4          ;          AJMP   XBLINAD ; Update and return

```

```

1      ; ; EID = erase in display
2      ;
3      ; *EID* erases the amount of the screen specified by Pn.
4      ;
5      ;
6      ; ENTRY none
7      ;
8      ; EXIT none
9      ;
10     ; USES all
11
12
13 0A1A 31 D1 EID: ACALL GNP ; Get parameter
14 0A1C 70 02 JNZ EID1 ; IF zero THEN
15 0A1E 81 08 AJMP EED ; erase to eos and return
16
17 0A20 14 EID1: DEC A
18 0A21 70 02 JNZ EID2 ; IF one THEN
19 0A23 61 F1 AJMP EBD ; erase from beginning & ret
20
21 0A25 14 EID2: DEC A
22 0A26 70 02 JNZ EIR ; IF two THEN
23 0A28 61 AC AJMP CLRS ; erase screen and return
24
25 0A2A 22 EIR: RET

```

```
1          ;;          EIL - erase in line
2          ;
3          ;          *EIL* erases the portion of the current line as specified
4          ;          by Pn.
5          ;
6          ;
7          ;          ENTRY  none
8          ;
9          ;          EXIT   none
10         ;
11         ;          USES   all
12         ;
13         ;
14 0A2B    31    D1    EIL:    ACALL  GNP          ; Get parameter
15 0A2D    70    02    JNZ    EIL1         ; IF zero THEN
16 0A2F    61    E0    AJMP   EOL          ; erase to eol and return
17
18 0A31    14          EIL1:   DEC     A          ;
19 0A32    70    02    JNZ    EIL2         ; IF one THEN
20 0A34    61    CB    AJMP   EBL          ; erase from beginning & ret
21
22 0A36    14          EIL2:   DEC     A          ;
23 0A37    70    F1    JNZ    EIR          ; IF two THEN
24 0A39    61    DB    AJMP   EEL          ; erase entire line and ret
```

```
1 ; ; ASGM - ANSI set graphics mode
2 ;
3 ; *ASGM* sets or resets any or all of the attributes and
4 ; graphics modes.
5 ;
6 ;
7 ; ENTRY none
8 ;
9 ; EXIT none
10 ;
11 ; USES all
12 ;
13
14 0A3B 31 D1 ASGM: ACALL GNP ; Get first parameter
15
16 0A3D 75 F0 07 ASG1: MOV B, #ASGMTL ; (B) = table length
17 0A40 90 0A50 MOV DPTR, #ASGMT ; (DPTR) = set mode table
18 0A43 31 F7 ACALL XSTAB ; Search table
19 0A45 40 02 JC ASG0 ; IF found THEN
20 0A47 51 4E ACALL ASG0 ; call the routine
21 0A49 31 D1 ASG0: ACALL GNP ; Get next parameter
22 0A4B 50 F0 JNC ASG1 ; IF done THEN return
23 0A4D 22 RET ; ELSE keep looping
24
25 0A4E E4 ASG0: CLR A
26 0A4F 73 JMP @A+DPTR ; Jump to routine
```

```

1          ;;          ASGMT - ANSI set graphics mode table
2          ;
3          ;          *ASGMT* contains the address of the routines which set the
4          ;          requested attribute or graphic mode.
5          ;
6          ;          Table entries are: The third character of the ANSI escape
7          ;          sequence followed by the address of the routine which performs
8          ;          the requested function.
9
10
11          0A50          ASGMT          EQU          $
12
13          0A50          00          DB          0          ; ESC [ 0 m
14          0A51          0F26          DW          XATR          ; Exit all attributes
15
16          0A53          02          DB          2          ; ESC [ 2 m
17          0A54          0F2A          DW          EHALF          ; Enter half intensity mode
18
19          0A56          04          DB          4          ; ESC [ 4 m
20          0A57          0F2D          DW          EUNDL          ; Enter underline mode
21
22          0A59          05          DB          5          ; ESC [ 5 m
23          0A5A          0F30          DW          EBLNK          ; Enter blink mode
24
25          0A5C          07          DB          7          ; ESC [ 7 m
26          0A5D          0F33          DW          ERVM          ; Enter reverse video mode
27
28          0A5F          0A          DB          10          ; ESC [ 10 m
29          0A60          0F39          DW          EGM          ; Enter graphics mode
30
31          0A62          0B          DB          11          ; ESC [ 11 m
32          0A63          0F3C          DW          XGM          ; Exit graphics mode
33
34          0007          ASGMTL          EQU          ($-ASGMT)/3          ; Table length

```

```

1      ;: ASPF - ANSI set protected fields
2      ;
3      ;
4      ; *ASPF* selects the attributes which imply protection. Multiple
5      ; attributes may be selected in a single sequence. This sequence
6      ; does not change any attributes of the characters displayed on
7      ; the screen or of the characters received. The sequence merely
8      ; changes the way the characters with the specified attributes
9      ; are interpreted by the different modes.
10     ;
11     ;
12     ; ENTRY none
13     ;
14     ; EXIT none
15     ;
16     ; USES all
17
18 0A65 75 F0 00 ASPF: MOV B, #0 ; Initialize byte
19
20 0A68 31 D1 ASPF1: ACALL GNP ; Get next parameter
21 0A6A 40 22 JC ASPF2 ; Exit when done
22 0A6C B4 00 03 CJNE A, #0, ASPF1A
23 0A6F 75 F0 00 MOV B, #0 ; None protected
24 0A72 B4 02 02 ASPF1A: CJNE A, #2, ASPF1B
25 0A75 D2 F3 SETB B.3 ; Half intensity
26 0A77 B4 04 02 ASPF1B: CJNE A, #4, ASPF1C
27 0A7A D2 F1 SETB B.1 ; Under line
28 0A7C B4 05 02 ASPF1C: CJNE A, #5, ASPF1D
29 0A7F D2 F2 SETB B.2 ; Blink
30 0A81 B4 07 02 ASPF1D: CJNE A, #7, ASPF1E
31 0A84 D2 F0 SETB B.0 ; Reverse video
32 0A86 B4 FE 03 ASPF1E: CJNE A, #254, ASPF1F
33 0A89 75 F0 80 MOV B, #10000000B ; No attributes are protected
34 0A8C 80 DA ASPF1F: SJMP ASPF1
35
36 0A8E 85 F0 3F ASPF2: MOV PFIELD, B ; Final change
37 0A91 22 RET

```

```
1          ;;          CBA - convert byte to ASCII
2          ;
3          ;          *CBA* converts a byte into its ASCII equivalent.
4          ;
5          ;
6          ;          ENTRY   (A) = byte to convert
7          ;
8          ;          EXIT    (B) = hundreds digit
9          ;          (DPH) = tens digit
10         ;          (DPL) = ones digit
11         ;
12         ;          USES   A, B, DPTR
13
14
15 0A92 75 F0 64 CBA:  MOV B, #100
16 0A95 84          DIV AB
17 0A96 C0 E0          PUSH ACC          ; Save hundreds amount
18 0A98 E5 F0          MOV A, B
19 0A9A 75 F0 0A      MOV B, #10
20 0A9D 84          DIV AB          ; (A) = tens (B) = ones
21 0A9E 24 30        ADD A, #'0'          ; Make tens digit ASCII
22 0AA0 F5 83        MOV DPH, A          ; Save it
23 0AA2 E5 F0        MOV A, B
24 0AA4 24 30        ADD A, #'0'          ; Make ones digit ASCII
25 0AA6 F5 82        MOV DPL, A          ; Save it
26 0AA8 D0 E0        POP ACC
27 0AAA 24 30        ADD A, #'0'          ; Make hundreds digit ASCII
28 0AAC F5 F0        MOV B, A          ; Save it
29 0AAE 22          RET
```

```

1      ;:          ACPR - ANSI cursor position report
2      ;
3      ;
4      ;          *ACPR* outputs the current cursor position in the ANSI form
5      ;          of ESC [ P1 ; Pc R      Please note the tricky code to save a
6      ;          few bytes of code.
7      ;
8      ;
9      ;          ENTRY   none
10     ;
11     ;
12     ;          EXIT    none
13     ;
14     ;
15     OAAF  31    D1          ACPR:  ACALL  GNP          ; Get parameter
16     OAB1  B4    06    35    CJNE  A, #6, APR        ; IF not 6 THEN return
17     OAB4  74    1B          MOV   A, #ESC
18     OAB6  51    CC          ACALL ACP3            ; Output an ESC
19     OAB8  74    5B          MOV   A, #'['
20     OABA  51    CC          ACALL ACP3            ; Output a '['
21     OABC  E5    12          MOV   A, YPOS
22     OABE  04          INC   A                    ; Bump to normalize
23     OABF  51    CF          ACALL ACP1            ; Output Y-position
24     OAC1  74    3B          MOV   A, #'';
25     OAC3  51    CC          ACALL ACP3            ; Output seperator
26     OAC5  E5    11          MOV   A, XPOS
27     OAC7  04          INC   A                    ; Bump to normalize
28     OAC8  51    CF          ACALL ACP1            ; Output X-position
29     OACA  74    52          MOV   A, #'R'
30     ;
31     OACC  02    022A        ACP3:  LJMPL PCDF          ; Output 'R' and return
32     ;
33     OACF  51    92          ACP1:  ACALL CBA          ; Convert position to ASCII
34     OAD1  AD    82          MOV   TEMP, DPL        ; Save one's digit
35     OAD3  E5    83          MOV   A, DPH          ; Get ten's digit
36     OAD5  B4    31    00    CJNE  A, #'1', ACP1A
37     OAD8  40    02          ACP1A: JC   ACP2          ; IF not zero THEN
38     OADA  51    CC          ACALL ACP3            ; output ten's digit
39     OADC  ED          ACP2:  MOV   A, TEMP
40     OADD  80    ED          SJMPL ACP3            ; Output one's digit and return

```


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

```

1          ; ;          APRNT - ANSI print page
2          ;
3          ;
4          ;          *APRNT* prints the page to the opposite port.
5          ;
6          ;
7          ;          ENTRY none
8          ;
9          ;          EXIT none
10         ;
11         ;          USES all
12         ;
13 OADF    11    2A          APRNT:    ACALL  SETDISP          ; Set dispatch and return
14
15 OAE1    11    2F          ACALL  SETNORM          ; Restore dispatch address
16 OAE3    B4    37    03    CJNE   A, #7, APR          ; IF ESC # 7 THEN
17 OAE6    02    103C      JMP    R2, PRNT          ; Print page
18 OAE9    22          APR:    RET
19
20
21
22
23         ; ;          ASBR - ANSI set baud rate
24         ;
25         ;          *ASBR* sets the baud rate to that specified by Pn.
26         ;
27         ;
28         ;          ENTRY none
29         ;
30         ;          EXIT none
31         ;
32         ;          USES all
33
34
35 OAEA    31    D1          ASBR:    ACALL  GNP          ; Get parameter
36 OAEC    50    01          JNC   ASBR1          ; IF no parameter THEN
37 OAE6    04          INC   A          ; default to 1
38 OAEF    B4    0E    00    ASBR1:  CJNE   A, #14, ASBR2
39 OAF2    50    F5          ASBR2:  JNC   APR          ; IF in range (0..13) THEN
40 OAF4    C1    4B          AJMP  SBR          ; set baud rate and return

```

```
1 ; ; ARAMP - ANSI reset all modes to power up configuration
2 ; ;
3 ; ; RAMP - reset all modes to power up configuration
4 ; ;
5 ; ;
6 ; ; *ARAMP* and *RAMP* resets all flass etc. to the configuration
7 ; ; on power up.
8 ; ;
9 ; ;
10 ; ; ENTRY none
11 ; ;
12 ; ; EXIT none
13 ; ;
14 ; ; USES all
15 ; ;
16 ; ;
17 0AF6 31 D1 ARAMP: ACALL GNP ; Get any parameter
18 0AF8 50 EF RAMP: JNC APR ; IF no parameters THEN
19 0AFA 02 1003 JMP R2_IN2 ; reset and return
```

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

```
1          ;;          WEOL - wrap at end of line
2          ;
3          ;          *WEOL* sets the flag which causes the terminal to perform
4          ;          a carriage return and line feed immediatly after the 80th
5          ;          character before a line is displayed.
6          ;
7          ;
8          ;          ENTRY   none
9          ;
10         ;          EXIT    none
11        ;
12        ;          USES    none
13
14
15 0AFD    D2    2B          WEOL:    SETB   AUTOWRAP          ; Flag auto wrap
16 0AFF    22          RET
17
18
19
20
21        ;;          DEOL - discard at end of line
22        ;
23        ;          *DEOL* resets the wrap around at end of line flag.
24        ;          characters past column 79 are placed in column 79 until
25        ;          a carriage return is received.
26        ;
27        ;
28        ;          ENTRY   none
29        ;
30        ;          EXIT    none
31        ;
32        ;          USES    none
33
34
35 0B00    C2    2B          DEOL:    CLR    AUTOWRAP          ; Clear auto wrap
36 0B02    22          RET
```

```
1 ; ; EAM - enter ANSI mode
2 ; ;
3 ; ;
4 ; ; *EAM* places the terminal in the ANSI mode for escape codes.
5 ; ;
6 ; ;
7 ; ; ENTRY none
8 ; ;
9 ; ; EXIT none
10 ; ;
11 ; ; USES none
12 ; ;
13 OB03 75 28 01 EAM: MOV MODE, #00000001B ; Place in ANSI mode
14 OB06 22 RET
15
16
17
18
19 ; ;
20 ; ; EZM - enter ZDS mode
21 ; ;
22 ; ; *EZM* takes the terminal out of ANSI mode and into ZDS mode.
23 ; ;
24 ; ;
25 ; ; ENTRY none
26 ; ;
27 ; ; EXIT none
28 ; ;
29 ; ; USES none
30 ; ;
31 OB07 75 28 02 EZM: MOV MODE, #00000010B ; Take out of ANSI mode
32 OB0A 22 RTI: RET
```

```

1          ;;          E25L - enable 25th line
2          ;
3          ;          *E25L* enables the display of the 25th line only if it has
4          ;          not been previously enabled. The 25th line is cleared at
5          ;          the time it is enabled.
6          ;
7          ;
8          ;          ENTRY   none
9          ;
10         ;          EXIT    none
11         ;
12         ;          USES    A, B, DPTR, TEMP, PTR1
13         ;
14         ;
15         ;          15      FC      E25L:      JB      L25EN, RT1      ; IF enabled THEN return
16         ;          16      D2      SETB     L25EN          ; ELSE enable 25th line
17         ;          17      75      F0      18      MOV     B, #MAXLINE
18         ;          18      B1      96      AJMP    CLRLINE          ; Clear 25th line
19         ;
20         ;
21         ;
22         ;
23         ;          ;;          D25L - disable 25th line
24         ;
25         ;          ;          *D25L* disables the display of the 25th line and clears it.
26         ;          ;          If the cursor is on the 25th line then it is sent to the
27         ;          ;          home position.
28         ;
29         ;
30         ;          ;          ENTRY   none
31         ;
32         ;          ;          EXIT    none
33         ;
34         ;          ;          USES    A, B, DPTR, TEMP, PTR1
35         ;
36         ;
37         ;          37      C2      15          D25L:      CLR     L25EN          ; Disable 25th line
38         ;          38      B7      75      18      MOV     B, #MAXLINE
39         ;          39      B1A     71      96      ACALL  CLRLINE          ; Clear 25th line
40         ;          40      B1C     E5      12      MOV     A, YPOS
41         ;          41      B1E     B4      18      02      CJNE   A, #MAXLINE, XR2_DISP ; IF cursor on 25th line THEN
42         ;          42      B21     91      36      ACALL  SCH              ; set cursor home
43         ;
44         ;          44      B23     02      1042     XR2_DISP:  LJMPL  R2_DISP          ; Display any status

```

```
1          ; ; EKC - enable keyboard click
2          ; ;
3          ; ;
4          ; ; *EKC* resets the flag that disables keyboard click and
5          ; ; outputs the command to the keyboard.
6          ; ;
7          ; ; ENTRY none
8          ; ;
9          ; ; EXIT none
10         ; ;
11         ; ; USES all
12
13
14 0B26 D2 33 EKC: SETB CLKF ; Turn on key click
15 0B28 74 82 MOV A, #KBC_EC
16 0B2A 02 026D XPCKB: JMP PCKB ; Output command and return
17
18
19
20
21         ; ; NKC - no keyboard click
22         ; ;
23         ; ; *NKC* sets the flag that disables keyboard click and
24         ; ; outputs the command to the keyboard.
25         ; ;
26         ; ;
27         ; ; ENTRY none
28         ; ;
29         ; ; EXIT none
30         ; ;
31         ; ; USES all
32
33
34 0B2D C2 33 NKC: CLR CLKF ; Turn off key click
35 0B2F 74 83 MOV A, #KBC_DC
36 0B31 80 F7 SJMP XPCKB ; Output command and return
```

```
1          ;;          SBC - set block cursor
2          ;
3          ;          *SBC* programs the CRTC to display a block cursor.
4          ;
5          ;
6          ;          ENTRY   none
7          ;
8          ;          EXIT    none
9          ;
10         ;          USES    A, B, DPTR
11         ;
12         ;
13 0B33    C2    32          SBC:          CLR    CRUL          ; Clear underline flag
14 0B35    80    02          SUMP        SUC1          ; Initialize cursor format
15
16
17
18
19         ;;          SUC - set underscore cursor
20         ;
21         ;          *SUC* programs the CRTC to display an underline cursor.
22         ;
23         ;
24         ;          ENTRY   none
25         ;
26         ;          EXIT    none
27         ;
28         ;          USES    A, B, DPTR
29         ;
30         ;
31 0B37    D2    32          SUC:          SETB   CRUL          ; Set underline flag
32
33 0B39    02    101B       SUC1:        JMP     R2_ICUR       ; Initialize cursor format
```

```

1      ; ;      EHSM - enter hold screen mode
2      ;
3      ;      *EHSM* sets the hold screen mode flag.
4      ;
5      ;
6      ;      ENTRY   none
7      ;
8      ;      EXIT    none
9      ;
10     ;      USES   none
11
12
13     OB3C  D2    36      EHSM:      SETB   HSMODEF      ; Enter mode
14     OB3E  75    1E    01      MOV    HSLINE, #1      ; Set number of lines
15     OB41  80    02      SJMP   XHSM1      ; Clear stop display and return
16
17
18
19
20     ; ;      XHSM - exit hold screen mode
21     ;
22     ;      *XHSM* clear hold screen mode flag.
23     ;
24     ;
25     ;      ENTRY   none
26     ;
27     ;      EXIT    none
28     ;
29     ;      USES   none
30
31
32     OB43  C2    36      XHSM:      CLR    HSMODEF      ; Exit mode
33
34     OB45  C2    22      XHSM1:     CLR    HSSTOPF     ; Clear stop display flag
35     OB47  22

```


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

```
1 ; ; EKSM - enter keypad shifted mode
2 ; ;
3 ; ; *EKSM* places the numeric keypad into the shifted mode.
4 ; ;
5 ; ;
6 ; ; ENTRY none
7 ; ;
8 ; ; EXIT none
9 ; ;
10 ; ; USES none
11 ; ;
12 ; ;
13 OB48 D2 30 EKSM: SETB KPADSHTF ; Set keypad shifted
14 OB4A 22 RET
15 ; ;
16 ; ;
17 ; ;
18 ; ;
19 ; ; XKSM - exit keypad shifted mode
20 ; ;
21 ; ; *XKSM* takes the numeric keypad out of the shifted mode.
22 ; ;
23 ; ;
24 ; ; ENTRY none
25 ; ;
26 ; ; EXIT none
27 ; ;
28 ; ; USES none
29 ; ;
30 ; ;
31 OB4B C2 30 XKSM: CLR KPADSHTF ; Clear keypad shifted
32 OB4D 22 RET
```

```
1 ;; EKAM - enter keypad alternate mode
2 ;
3 ;
4 ; *EKAM* places the numeric keypad into the alternate mode.
5 ;
6 ;
7 ; ENTRY none
8 ;
9 ; EXIT none
10 ;
11 ; USES none
12 ;
13 0B4E D2 31 EKAM: SETB KPADALTF ; Set keypad alternate
14 0B50 22 RET
15
16
17
18
19 ;; XKAM - exit keypad alternate mode
20 ;
21 ;
22 ; *XKAM* takes the numeric keypad out of the alternate mode.
23 ;
24 ;
25 ; ENTRY none
26 ;
27 ; EXIT none
28 ;
29 ; USES none
30 ;
31 0B51 C2 31 XKAM: CLR KPADALTF ; Clear keypad alternate
32 0B53 22 RET
```

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

```

1          ;;          EICM - enter insert character mode
2          ;
3          ;          *EICM* places terminal into insert character mode.
4          ;
5          ;
6          ;          ENTRY   none
7          ;
8          ;          EXIT    none
9          ;
10         ;          USES    all
11         ;
12         ;
13 0B54    D2    21          EICM:    SETB    ICMODEF          ; Set insert mode flas
14 0B56    61    23          AJMP    XR2_DISP          ; Display any change in status
15
16
17
18
19         ;;          XICM - exit insert character mode
20         ;
21         ;          *XICM* takes terminal out of the insert character mode.
22         ;
23         ;
24         ;          ENTRY   none
25         ;
26         ;          EXIT    none
27         ;
28         ;          USES    all
29         ;
30         ;
31 0B58    C2    21          XICM:    CLR     ICMODEF          ; Clear insert mode flas
32 0B5A    61    23          AJMP    XR2_DISP          ; Display any change in status

```

```
1 ; ; EC - enable cursor
2 ;
3 ; *EC* enables the display of the cursor.
4 ;
5 ;
6 ; ENTRY none
7 ;
8 ; EXIT none
9 ;
10 ; USES none
11 ;
12 ;
13 OB5C D2 1A EC: SETB ENBLCUR ; Enable display of cursor
14 OB5E 22 RET
15 ;
16 ;
17 ;
18 ;
19 ; ; DC - disable cursor
20 ;
21 ; *DC* disables the display of the cursor
22 ;
23 ;
24 ; ENTRY none
25 ;
26 ; EXIT none
27 ;
28 ; USES none
29 ;
30 ;
31 OB5F C2 1A DC: CLR ENBLCUR ; Disable display of cursor
32 OB61 22 RET
```

```
1          ;;          EACR - enable auto carriage return
2          ;
3          ;          *EACR* enables the terminal to perform an automatic carriage
4          ;          return upon receipt of a line feed character.
5          ;
6          ;
7          ;          ENTRY none
8          ;
9          ;          EXIT none
10         ;
11         ;          USES none
12
13
14 OB62 D2 2C          EACR:          SETB  AUTOOCR          ; Enter auto CR mode
15 OB64 22          RET
16
17
18
19
20         ;;          XACR - exit auto carriage return
21         ;
22         ;          *XACR* clears the auto carriage return flag.
23         ;
24         ;
25         ;          ENTRY none
26         ;
27         ;          EXIT none
28         ;
29         ;          USES none
30
31
32 OB65 C2 2C          XACR:          CLR    AUTOOCR          ; Exit auto CR mode
33 OB67 22          RET
```

```
1 ; EALF - enable auto line feed
2 ;
3 ; *EALF* enables the terminal to perform an automatic
4 ; line feed upon receipt of a carriage return.
5 ;
6 ;
7 ; ENTRY none
8 ;
9 ; EXIT none
10 ;
11 ; USES none
12 ;
13 ;
14 OB68 D2 2D EALF: SETB AUTOLF ; Enter auto LF mode
15 OB6A 22 RET
16 ;
17 ;
18 ;
19 ;
20 ; XALF - exit auto line feed mode
21 ;
22 ; *XALF* clears the auto line feed mode flag.
23 ;
24 ;
25 ; ENTRY none
26 ;
27 ; EXIT none
28 ;
29 ; USES none
30 ;
31 ;
32 OB6B C2 2D XALF: CLR AUTOLF ; Exit auto LF mode
33 OB6D 22 RET
```

```
1          ; ; EGATM - enter guarded area transfer mode
2          ; ;
3          ; ;
4          ; ; *EGATM* places the terminal in a mode such that all data
5          ; ; is to be transmitted in a data stream such as transmit pase.
6          ; ;
7          ; ; ENTRY none
8          ; ;
9          ; ; EXIT none
10         ; ;
11         ; ; USES none
12         ; ;
13         ; ;
14 0B6E C2 25 EGATM: CLR GATM ; Clear guarded area transfer
15 0B70 22 RET
16
17
18
19
20         ; ; XGATM - exit guarded area transfer mode
21         ; ;
22         ; ; *XGATM* places the terminal in a mode such that only
23         ; ; unprotected data is to be transmitted in a data stream
24         ; ; such as transmit pase.
25         ; ;
26         ; ;
27         ; ; ENTRY none
28         ; ;
29         ; ; EXIT none
30         ; ;
31         ; ; USES none
32         ; ;
33         ; ;
34 0B71 D2 25 XGATM: SETB GATM ; Set guarded area transfer
35 0B73 22 RET
```

```

1          ;|          EERM - enter erasure mode
2          ;
3          ;          *EERM* places the terminal in a mode such that all data is
4          ;          erased in functions such as erase screen.
5          ;
6          ;
7          ;          ENTRY  none
8          ;
9          ;          EXIT   none
10         ;
11         ;          USES   none
12         ;
13
14 OB74    C2    1B          EERM:          CLR    ERM          ; Clear erasure mode
15 OB76    22
16
17
18
19
20         ;|          XERM - exit erasure mode
21         ;
22         ;          *XERM* places the terminal in a mode such that only
23         ;          unprotected data is erased in functions such as erase
24         ;          screen.
25         ;
26         ;
27         ;          ENTRY  none
28         ;
29         ;          EXIT   none
30         ;
31         ;          USES   none
32         ;
33
34 OB77    D2    1B          XERM:          SETB   ERM          ; Set erasure mode
35 OB79    22

```



```

1          ; ; PCLK - program clock
2          ;
3          ; *PCLK* sets the internal clock.
4          ;
5          ;
6          ; ENTRY none
7          ;
8          ; EXIT none
9          ;
10         ; USES A, DPTR
11
12
13 0B7A 31 D1 PCLK: ACALL GNP ; Get hour parameter
14 0B7C B4 18 00 CJNE A, #24, PCL1A
15 0B7F 50 14 PCL1A: JNC PCL2 ; IF bad parameter THEN exit
16 0B81 F5 3D MOV THOUR, A
17 0B83 31 D1 ACALL GNP ; Get minutes parameter
18 0B85 B4 3C 00 CJNE A, #60, PCL1B
19 0B88 50 0B PCL1B: JNC PCL2 ; IF bad parameter THEN exit
20 0B8A F5 3C MOV TMIN, A
21 0B8C 31 D1 ACALL GNP ; Get seconds parameter
22 0B8E B4 3C 00 CJNE A, #60, PCL1C
23 0B91 50 02 PCL1C: JNC PCL2 ; IF bad parameter THEN exit
24 0B93 F5 3B MOV TSEC, A
25 0B95 22 PCL2: RET

```

```

1          ;;          CLRLINE - clear line completely.
2          ;
3          ;
4          ;          *CLRLINE* writes spaces into the entire line indicated and
5          ;          ignores any settings of Protected fields.
6          ;
7          ;          ENTRY (B) = line number to erase
8          ;
9          ;          EXIT none
10         ;
11         ;          USES A, B, DPTR, TEMP, PTR1, WORK2, INDX2
12         ;
13
14 0B96 C0 3F          CLRLINE:  PUSH  PFIELD          ; Save protected fields
15 0B98 75 3F 00     MOV    PFIELD, #0      ; Make everything unprotected
16 0B9B 71 A0        ACALL  CLR           ; Clear line
17 0B9D D0 3F        POP    PFIELD         ; Restore protected fields
18 0B9F 22          RET
19
20
21
22
23         ;;          CLRL - clear line
24         ;
25         ;          *CLRL* writes spaces into the entire line indicated.
26         ;
27         ;
28         ;          ENTRY (B) = line number to erase
29         ;
30         ;          EXIT none
31         ;
32         ;          USES A, B, DPTR, TEMP, PTR1, WORK2, INDX2
33         ;
34
35 0BA0 E4          CLRL:      CLR    A
36 0BA1 12 0503     LCALL  CLA           ; Calculate address
37 0BA4 74 50        MOV    A, #NUMCHARS  ; 80 characters
38 0BA6 75 F0 00     MOV    B, #0         ; No attributes
39 0BA9 02 100C     LJMP  R2_FILL       ; Fill with spaces and return

```

```

1          ;;          CLR5 - clear screen
2          ;
3          ;
4          ;          *CLR5* clears entire display except the 25th line. IF cursor
5          ;          is on the 25th line then just the 25th line is cleared.
6          ;
7          ;          ENTRY   none
8          ;
9          ;          EXIT    none
10         ;
11         ;          USES   all
12         ;
13
14  OBAC   E5   12          CLR5:   MOV     A, YPOS
15  OBAB   B4   18   04     CJNE   A, #MAXLINE, CS1      ; IF ypos = 25th line THEN
16  OBB1   F5   F0         MOV     B, A          ; setup for subroutine
17  OBB3   80   EB         SJMP   CLRL          ; clear line and return
18
19  OBB5   7C   18          CS1:   MOV     WORK, #NUMLINES-1      ; Get number of lines
20  OBB7   7A   00         MOV     INDX1, #0          ; Index for -LORDER-
21
22  OBB9   8A   F0          CS2:   MOV     B, INDX1          ; REPEAT setup for call
23  OBBB   71   A0         ACALL  CLRL          ; to clear a line
24  OBBD   0A          INC     INDX1          ; bump indxi
25  OBBE   DC   F9         DJNZ   WORK, CS2       ; UNTIL work = 0
26
27  OBC0   81   36         AJMP   SCH          ; Set cursor home and return
28
29
30
31
32         ;;          CLF0 - clear foreground
33         ;
34         ;          *CLF0* clears entire display except for half intensity spaces.
35         ;
36         ;
37         ;          ENTRY   none
38         ;
39         ;          EXIT    none
40         ;
41         ;          USES   all
42         ;
43
44  OBC2   75   3F   08     CLF0:   MOV     PFIELD, #00001000B      ; Make half intensity protected
45  OBC5   71   AC          ACALL  CLR5          ; Clear screen
46  OBC7   75   3F   00     MOV     PFIELD, #0          ; Make nothing protected
47  OBCA   22          RET
  
```

```

1          ;:          EBL - erase from beginning of line
2          ;
3          ;          *EBL* erases from the beginning of the current line to the
4          ;          cursor. Cursor position does not change.
5          ;
6          ;
7          ;          ENTRY   none
8          ;
9          ;          EXIT    none
10         ;
11         ;          USES   A, B, DPTR, TEMP, PTR1, WORK2, INDX2
12         ;
13
14 OBCB    E4          EBL:    CLR    A
15 OBCC    85          12      F0    MOV    B, YPOS
16 OBCF    12          0503    LCALL  CLA          ; Calc address for (0,YPOS)
17 OBD2    E5          11      MOV    A, XPOS      ; Get number of characters
18 OBD4    04          INC     A          ; to erase
19 OBD5    75          F0      00    MOV    B, #0        ; Clear attributes
20 OBD8    02          100C    LJMPL R2_FILL     ; Clear and return
21
22
23
24
25         ;:          EEL - erase entire line
26         ;
27         ;          *EEL* clears the entire line where the cursor resides.
28         ;          Cursor position does not change.
29         ;
30         ;
31         ;          ENTRY   none
32         ;
33         ;          EXIT    none
34         ;
35         ;          USES   A, B, DPTR, TEMP, PTR1, WORK2, INDX2
36         ;
37
38 OBDB    85          12      F0    EEL:    MOV    B, YPOS      ; Get current position
39 OBDE    80          C0      SJMPL CLRL        ; Clear line and return

```

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

```

1          ;;          EOL - erase to end of line
2          ;
3          ;          *EOL* erases from the cursor to the end of the current line.
4          ;          Cursor position does not change.
5          ;
6          ;
7          ;          ENTRY none
8          ;
9          ;          EXIT none
10         ;
11         ;          USES A, B, DPTR, TEMP, PTR1, WORK2, INDX2
12
13
14 OBE0    75    F0    00    EOL:    MOV    B, #0          ; Clear attributes
15
16 OBE3    74    50          EOL1:   MOV    A, #NUMCHARS ; Calculate number to erase
17 OBE5    C3          CLR    C
18 OBE6    95    11          SUBB   A, XPOS
19 OBE8    85    14    82          MOV    DPL, LINADL ; Get line address
20 OBEB    85    13    83          MOV    DPH, LINADH
21 OBEE    02    100C          LJMP   R2-FILL ; Erase to end and return
22
23
24
25
26         ;          EBD - erase from beginning of display
27         ;
28         ;          *EBD* clears the screen from 'HOME' position to the current
29         ;          cursor position. The exception is the 25th line. Cursor
30         ;          position does not change.
31         ;
32         ;
33         ;          ENTRY none
34         ;
35         ;          EXIT none
36         ;
37         ;          USES A, B, DPTR, TEMP, PTR1, INDX1, WORK2, INDX2
38
39
40 OBF1    E5    12          EBD:    MOV    A, YPOS
41 OBF3    B4    18    02          CJNE   A, #MAXLINE, EB1 ; IF 25th line THEN
42 OBF6    80    D3          SJMP   EBL ; erase just this line
43
44 OBF8    7A    00          EB1:    MOV    INDX1, #0 ; Start at top
45 OBF9    E5    12          EB2:    MOV    A, YPOS ; REPEAT
46 OBFc    B5    02    02          CJNE   A, INDX1A, EB3 ; IF index = current THEN
47 OBFf    80    CA          SJMP   EBL ; erase up to cursor
48
49 OC01    8A    F0          EB3:    MOV    B, INDX1 ; ELSE set up and
50 OC03    71    A0          ACALL  CLRl ; clear entire line
51 OC05    0A          INC    INDX1 ; bump index counter
52 OC06    80    F2          SJMP   EB2 ; UNTIL forever

```

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

```

1          ;:          EED - erase to end of display
2          ;
3          ;          *EED* clears the screen from the current cursor position to
4          ;          the end of the screen.  Cursor position does not change.
5          ;
6          ;
7          ;          ENTRY  none
8          ;
9          ;          EXIT   none
10         ;
11         ;          USES   A, B, DPTR, TEMP, PTR1, INDX1, WORK2, INDX2
12         ;
13
14 0C08    71    E0          EED:          ACALL  EOL          ; Clear to end of line
15 0C0A    AA    12          MOV     INDX1, YPOS      ; Set loop counter
16 0C0C    0A          EE1:          INC     INDX1          ; REPEAT bump to next line
17 0C0D    BA    18    00    CJNE   INDX1, #MAXLINE, EE1A
18 0C10    50    06    EE1A:         JNC    EER            ; IF end of disp THEN exit
19 0C12    8A    F0          MOV     B, INDX1        ; set line to clear
20 0C14    71    A0          ACALL  CLRL           ; clear line
21 0C16    80    F4          SJMP   EE1            ; UNTIL forever
22 0C18    22          EER:          RET
23
24
25
26
27         ;:          HEED - Hazeltine erase to end of display (with background)
28         ;
29         ;          *HEED* clears the screen from the current cursor position
30         ;          to the end of the screen with half intensity spaces.  The
31         ;          cursor position does not change.
32         ;
33         ;
34         ;          ENTRY  none
35         ;
36         ;          EXIT   none
37         ;
38         ;          USES   A, B, DPTR, TEMP, PTR1, INDX1, WORK2, INDX2
39
40
41 0C19    75    F0    08    HEED:         MOV     B, #00001000B ; Get half intensity
42 0C1C    71    E3          ACALL  EOL1         ; Clear to end of line
43
44 0C1E    AA    12          MOV     INDX1, YPOS      ; Set loop counter
45 0C20    0A          HEE1:          INC     INDX1          ; REPEAT bump to next line
46 0C21    BA    18    00    CJNE   INDX1, #MAXLINE, HEE1A
47 0C24    50    F2    HEE1A:         JNC    EER            ; IF end of disp THEN exit
48 0C26    E4          CLR     A
49 0C27    8A    F0          MOV     B, INDX1        ; set line to clear
50 0C29    12    0503       LCALL  CLA            ; calculate line address
51 0C2C    74    50          MOV     A, #NUMCHARS    ; complete line
52 0C2E    75    F0    08    MOV     B, #00001000B ; set half intensity
53 0C31    12    100C       LCALL  R2_FILL        ; fill with spaces
54 0C34    80    EA          SJMP   HEE1           ; UNTIL forever

```

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

```

1          ;;          SCH - set cursor home
2          ;
3          ;          *SCH* Places the cursor at line zero, column zero.
4          ;
5          ;
6          ;          ENTRY  none
7          ;
8          ;          EXIT   none
9          ;
10         ;          USES   A, DPTR, PTR1
11
12
13 0C36    E4          SCH:    CLR    A
14 0C37    F5    11     MOV    XPOS, A          ; Zero X-Position
15 0C39    F5    12     MOV    YPOS, A          ; Zero Y-Position
16 0C3B    21    F4     AJMP   XBLINAD        ; Update address and return
17
18
19
20
21         ;;          PBS - perform back space
22         ;
23         ;          CLFT - cursor left
24         ;
25         ;          *PBS* - *CLFT* steps the cursor one position to the left, but
26         ;          does not wrap around to the previous line after reaching
27         ;          column zero.
28         ;
29         ;          ENTRY  none
30         ;
31         ;          EXIT   none
32         ;
33         ;          USES   A, DPTR, PTR1
34
35
36         OC3D        PBS    EQU    $
37         OC3D        CLFT   EQU    $
38 0C3D    E5    11     MOV    A, XPOS          ; Get X-Position
39 0C3F    70    01     JNZ    PBS1            ; IF -XPOS- is zero THEN
40 0C41    22          RET                    ; return
41
42 0C42    15    11     PBS1:  DEC    XPOS          ; ELSE decrement -XPOS-
43 0C44    21    F4     AJMP   XBLINAD        ; build address and return

```

```

1          ;:          ACLFT - ANSI cursor left
2          ;
3          ;
4          ;          *ACLFT* moves the cursor towards the beginning of a line
5          ;          the specified Pn number of times.
6          ;
7          ;
8          ;          ENTRY   none
9          ;
10         ;
11         ;          EXIT    none
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          ;              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         ;
35         ;          EXIT    none
36         ;
37         ;          USES   A, DPTR, PTR1, TEMP
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, TEMP, 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

```


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

```

1          ;;          CRT - cursor right
2          ;
3          ;
4          ;          *CRT* moves the cursor right one column.  If the cursor is in
5          ;          the last column then the cursor will not be advanced.
6          ;
7          ;          ENTRY  none
8          ;
9          ;          EXIT   none
10         ;
11         ;          USES   A, DPTR
12         ;
13
14 0C64    E5    11    CRT:    MOV    A, XPOS    ; Get X-position
15 0C66    B4    4F    01    CJNE   A, #MAXCHAR, CRT1 ; IF end of line THEN
16 0C69    22                    RET            ; return
17
18 0C6A    05    11    CRT1:   INC    XPOS    ; ELSE bump position
19 0C6C    21    F4    AJMP   XBLINAD ; update and return
20
21
22
23
24         ;;          ACRT - ANSI cursor right
25         ;
26         ;
27         ;          *ACRT* moves the cursor towards the end of the line
28         ;          the specified Pn number of times.
29         ;
30         ;          ENTRY  none
31         ;
32         ;          EXIT   none
33         ;
34         ;          USES   A, DPTR, INDX1
35
36
37 0C6E    31    D1    ACRT:   ACALL  GNP      ; Get parameter
38 0C70    70    01    JNZ   ACR1    ; IF no parameter or 0 THEN
39 0C72    04                    INC    A        ; bump to 1
40
41 0C73    FA                    ACRT1:  MOV    INDX1, A ; Set up index
42 0C74    91    64    ACRT:   ACALL  CRT      ; REPEAT cursor right one
43 0C76    DA    FC    DJNZ   INDX1, ACR ; UNTIL count = 0
44 0C78    22                    RET

```

```
1      ;:          HCRT = Hazeltine cursor right
2      ;
3      ;          #HCRT* moves the cursor right one column.  If the cursor is
4      ;          in the last column then the cursor moves to the beginning of
5      ;          the next line.
6      ;
7      ;
8      ;          ENTRY   none
9      ;
10     ;          EXIT    none
11     ;
12     ;          USES    A, DPTR, PTR1
13
14
15 0C79  E5    11          HCRT:      MOV     A, XPOS          ; Get X-Position
16 0C7B  B4    4F    E6      CJNE    A, #MAXCHAR, CRT      ; IF not eol THEN use *CRT*
17 0C7E  AD    12          MOV     TEMP, YPOS        ; Save Y-Position
18 0C80  91    A8          ACALL   CDN             ; Move cursor down
19 0C82  E5    12          MOV     A, YPOS
20 0C84  B5    05    01      CJNE    A, TEMP, HCRT1      ; IF xpos did not change THEN
21 0C87  22          RET                     ; do nothing
22
23 0C88  75    11    00      HCRT1:   MOV     XPOS, #0        ; ELSE move to beginning
24 0C8B  21    F4          AJMP   XBLINAD         ; update and return
```

```

1          ;;          CUP - cursor up
2          ;
3          ;          *CUP* moves the cursor up one line on the display. The
4          ;          cursor may move out of but not into fixed regions and
5          ;          never past line zero.
6          ;
7          ;
8          ;          ENTRY none
9          ;
10         ;          EXIT none
11         ;
12         ;          USES A, DPTR, PTR1
13
14
15 0C8D E5 12          CUP:      MOV A, YPOS          ; Get Y-position
16 0C8F 60 03          JZ CUP1          ; IF top of screen THEN return
17 0C91 B5 17 01      CUNE      A, TSCROLL, CUP1A      ; IF top of scroll region THEN
18 0C94 22            CUP1:     RET                  ; return
19
20 0C95 B4 18 01      CUP1A:    CUNE A, #MAXLINE, CUP1B    ; IF 25th line THEN
21 0C98 22            RET                  ; return
22
23 0C99 15 12          CUP1B:    DEC YPOS          ; ELSE dec position
24 0C9B 21 F4          AJMP XBLINAD      ; update and return
25
26
27
28
29         ;;          ACUP - ANSI cursor up
30         ;
31         ;          *ACUP* moves the cursor toward the top of the display
32         ;          the specified Pn number of times.
33         ;
34         ;
35         ;          ENTRY none
36         ;
37         ;          EXIT none
38         ;
39         ;          USES A, DPTR, INDX1, PTR1
40
41
42 0C9D 31 D1          ACUP:     ACALL GNP          ; Get parameter
43 0C9F 70 01          JNZ ACU1          ; IF no parameter or 0 THEN
44 0CA1 04            INC A          ; bump to 1
45
46 0CA2 FA          ACU1:     MOV INDX1, A          ; Set up index
47 0CA3 91 8D          ACU:      ACALL CUP          ; REPEAT cursor up one line
48 0CA5 DA FC          DJNZ INDX1, ACU      ; UNTIL count = 0
49 0CA7 22            RET

```

```

1      ; ;          CDN - cursor down
2      ;
3      ;          *CDN* moves the cursor down one line on the display...The
4      ;          cursor may move out of but not into fixed regions and
5      ;          never past line 24.
6      ;
7      ;
8      ;          ENTRY   none
9      ;
10     ;          EXIT    none
11     ;
12     ;          USES    A, DPTR, PTR1
13     ;
14     ;
15     OCA8  E5    12    01    CDN:    MOV    A, YPOS          ; Get Y-position
16     OCAA  B4    18    01    CJNE   A, #MAXLINE, CDN1      ; IF 25th line THEN return
17     OCAD  22
18     ;
19     OCAE  B4    17    01    CDN1:   CJNE   A, #MAXLINE-1, CDN2  ; IF 24th line THEN return
20     OCB1  22
21     ;
22     OCB2  04    18    01    CDN2:   INC    A                ; Bump value
23     OCB3  B5    18    01    CJNE   A, BFIX, CDN3          ; IF scroll region THEN return
24     OCB6  22
25     ;
26     OCB7  05    12    01    CDN3:   INC    YPOS          ; ELSE bump position
27     OCB9  21    F4    01    AJMP   XBLINAD             ; Update and return
28     ;
29     ;
30     ;
31     ;
32     ; ;          ACDN - ANSI cursor down
33     ;
34     ;          *ACDN* moves the cursor toward the bottom of the display
35     ;          the specified Pn number of lines.
36     ;
37     ;
38     ;          ENTRY   none
39     ;
40     ;          EXIT    none
41     ;
42     ;          USES    A, DPTR, INDX1, PTR1
43     ;
44     ;
45     OCB8  31    D1    01    ACDN:   ACALL  GNP              ; Get parameter
46     OCB9  70    01    01    JNZ   ACD1          ; IF no parameters or 0 THEN
47     OCBF  04
48     ;
49     OCC0  FA
50     OCC1  91    A8    01    ACD1:   MOV    INDX1, A        ; Set up index
51     OCC3  DA    FC    01    ACD:   ACALL  CDN              ; REPEAT cursor down one line
52     OCC5  22
           DJNZ  INDX1, ACD          ; UNTIL count = 0
           RET

```

```
1          ;;          APBT - ANSI perform back tab
2          ;
3          ;          *APBT* moves the cursor back the number of tab stops indicated.
4          ;
5          ;
6          ;          ENTRY none
7          ;
8          ;          EXIT none
9          ;
10         ;          USES all
11
12
13 OCC6    31    D1    APBT:    ACALL  GNP          ; Get parameter
14 OCC8    70    01    JNZ     APBT1       ; IF no parameter or 0 THEN
15 OCCA    04    INC     A              ; bump to 1
16
17 OCCB    C0    E0    APBT1:   PUSH   ACC          ; REPEAT
18 OCCD    12    0602  LCALL  BTAB        ; perform back tab
19 OCCD0   D0    E0    POP     ACC
20 OCD2    D5    E0    F6    DJNZ  ACC, APBT1  ; UNTIL count = 0
21 OCD5    22    RET
```

```

1      ; ; CUA - cursor addressing routine
2      ;
3      ; *CUA* does direct cursor addressing for a few emulation modes.
4      ;
5      ;
6      ; ENTRY none
7      ;
8      ; EXIT none
9      ;
10     ; USES all
11
12
13     OCD6 11 2A CUA: ACALL SETDISP ; Set dispatch and return
14
15     OCD8 75 16 04 MOV DSADRL, #LOW (CUC) ; Next dispatch address
16     OCDB 75 15 0D MOV DSADRH, #HIGH (CUC)
17     OCDE 30 43 11 JNB HAZ, CUB0 ; IF Hazeltine THEN
18     OCE1 B4 60 00 CJNE A, #96, CUA1
19     OCE4 40 02 CUA1: JC CUA2 ; IF value >= 96 THEN
20     OCE6 94 60 SUBB A, #96 ; subtract offset
21     OCE8 B4 50 00 CUA2: CJNE A, #NUMCHARS, CUA2A
22     OCEB 40 02 CUA2A: JC CUA3 ; IF value >= numchars THEN
23     OCED 74 4F MOV A, #MAXCHAR ; move to end of line
24     OCEF 80 2F CUA3: SJMP CUR2 ; update and return
25
26     OCF1 22 CUR: RET
27
28     OCF2 C3 CUB0: CLR C
29     OCF3 94 20 SUBB A, #' / ; ELSE subtract offset
30     OCF5 40 FA JC CUR ; check for lower bounds
31     OCF7 B4 18 04 CJNE A, #MAXLINE, CUB1 ; check for upper bounds
32     OCFA 30 15 F4 JNB L25EN, CUR ; IF 25th disabled THEN ret
33     OCFD D3 SETB C ; ELSE fall through
34     OCFE 50 F1 CUB1: JNC CUR ; IF upper bounds ok THEN
35     OD00 F5 12 CUR1: MOV YPOS, A ; replace Y-position
36     OD02 21 F4 AJMP XBLINAD ; build line address
37
38     OD04 11 2F CUC: ACALL SETNORM ; Next dispatch address
39     OD06 30 43 0B JNB HAZ, CUC0 ; IF Hazeltine THEN
40     OD09 54 1F ANL A, #00011111B ; do mod function
41     ODOB B4 18 00 CJNE A, #MAXLINE, CUC1
42     OD0E 40 02 CUC1: JC CUC1A ; IF value >= 24 THEN
43     OD10 74 17 MOV A, #MAXLINE-1 ; move to last line
44     OD12 80 EC CUC1A: SJMP CUR1 ; update and return
45
46     OD14 C3 CUC0: CLR C
47     OD15 94 20 SUBB A, #' / ; Subtract offset
48     OD17 40 05 JC CUC2 ; Check for lower bounds
49     OD19 B4 50 00 CJNE A, #NUMCHARS, CUC0A
50     OD1C 40 02 CUC0A: JC CUC2 ; Check for upper bounds
51     OD1E 74 4F CUC2: MOV A, #MAXCHAR ; IF bad range THEN move to end
52     OD20 F5 11 CUR2: MOV XPOS, A ; Replace X-position
53     OD22 21 F4 AJMP XBLINAD ; Build line address and return

```

```

1          ;:          DSR - define scrolling region
2          ;
3          ;          *DSR* defines the scrolling regions top and bottom line.
4          ;          The region must be at least two lines long and can not include
5          ;          the 25th line.
6          ;
7          ;
8          ;          ENTRY   none
9          ;
10         ;
11         ;          EXIT   none
12         ;
13         ;          USES   A, DPTR
14
15 OD24    31    D1    DSR:      ACALL  GNP          ; Get first parameter
16 OD26    60    01          JZ     DSRO          ; IF parameter and not 0 THEN
17 OD28    14          DEC     A              ; normalize to 0..24
18
19 OD29    B4    17    00    DSRO:  CJNE   A, #MAXLINE-1, DSROA
20 OD2C    50    02          JNC   DSROA          ; IF top >= 22 THEN
21 OD2E    F5    17          MOV   TSCROLL, A      ; change value
22
23 OD30    31    D1    DSR1:    ACALL  GNP          ; Get next parameter
24 OD32    60    01          JZ     DSR1A          ; IF parameter and not 0 THEN
25 OD34    14          DEC     A              ; normalize to 0..24
26
27 OD35    B5    17    01    DSR1A: CJNE   A, TSCROLL, DSR2
28 OD38    D3          SETB  C              ; IF bottom = top OR
29
30 OD39    40    05          JC     DSRERR          ; IF bottom < top OR
31 OD3B    B4    18    00    DSR2:  CJNE   A, #MAXLINE, DSR2A
32 OD3E    40    02          JC     DSROK          ; IF bottom >= 25 THEN error
33
34 OD40    74    17          DSRERR: MOV   A, #MAXLINE-1 ; Default error value
35 OD42    04          DSR0K:  INC   A              ; Bump for top of bottom fixed
36 OD43    F5    18          MOV   BFIX, A      ; Save value
37 OD45    81    36          AJMP  SCH          ; Set cursor home and return

```

```

1          ;;          PIL - perform insert line
2          ;
3          ;
4          ;          *PIL* inserts a blank line at the cursor position after
5          ;          moving the remaining lines down one line. Everything is
6          ;          done relative to the region the cursor is in, top fixed,
7          ;          bottom fixed, 25th line, or scrolling region. No region
8          ;          outside of the current region is effected. The cursor is
9          ;          moved to the beginning of the current line.
10         ;
11         ;
12         ;          ENTRY none
13         ;
14         ;          EXIT none
15         ;
16         ;          USES all
17
18 0D47 C0 12          PIL:  PUSH  YPOS          ; Save values
19 0D49 C0 17          PUSH  TSCROLL
20 0D4B C0 18          PUSH  BFIX
21 0D4D E5 12          MOV   A, YPOS          ; Get position
22 0D4F B5 17 00      CJNE  A, TSCROLL, PILA  ; IF upper fixed region THEN
23 0D52 40 21          PILA:  JC    PILRT1          ; just return
24 0D54 15 18          DEC   BFIX
25 0D56 B5 18 07      CJNE  A, BFIX, PIL1  ; IF last line scrolling region
26 0D59 85 12 F0      MOV   B, YPOS
27 0D5C 71 96          PILO:  ACALL CLRLINE  ; just clear the line
28 0D5E 80 12          SUMP  PILRET
29
30 0D60 40 08          PIL1:  JC    PIL2          ; IF lower fixed THEN
31
32 0D62 B4 18 10      CJNE  A, #MAXLINE, PILRT1  ; IF not 25th line THEN return
33 0D65 75 F0 18      MOV   B, #MAXLINE  ; ELSE clear 25th line
34 0D68 80 F2          SUMP  PILO          ; and return
35
36 0D6A 05 18          PIL2:  INC   BFIX
37 0D6C 85 12 17      MOV   TSCROLL, YPOS  ; Insert line uses
38 0D6F 12 0518      CALL  PRLF          ; Reverse line feed routine
39
40 0D72 75 11 00      PILRET: MOV  XPOS, #0          ; Move to beginning of line
41 0D75 D0 18          PILRT1: POP  BFIX          ; Restore values
42 0D77 D0 17          POP  TSCROLL
43 0D79 D0 12          POP  YPOS
44 0D7B 21 F4          AJMP  XBLINAD        ; Build line address and return

```


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

```
1          ;;          APIL - ANSI perform insert line
2          ;
3          ;          *APIL* inserts the specified Pn number of lines at the
4          ;          current cursor position.
5          ;
6          ;
7          ;          ENTRY none
8          ;
9          ;          EXIT none
10         ;
11         ;          USES all
12
13
14 0D7D 31 D1 APIL: ACALL GNP ; Get next parameter
15 0D7F 70 01 JNZ APILO ; IF no parameter or 0 THEN
16 0D81 04 INC A ; bump to 1
17
18 0D82 FA APIL0: MOV INDX1, A ; Set up index
19 0D83 B1 47 APIL1: ACALL PIL ; REPEAT insert one line
20 0D85 DA FC DJNZ INDX1, APIL1 ; UNTIL count = 0
21 0D87 22 RET
```

```

1          ; ; PDL = perform delete line.
2          ; ;
3          ; ; *PDL* moves the remaining lines of the current region UP
4          ; ; one line, clears the last line of the region, and moves
5          ; ; the cursor to the beginning of the current line.
6          ; ;
7          ; ;
8          ; ; ENTRY none
9          ; ;
10         ; ; EXIT none
11        ; ;
12        ; ; USES all
13       ; ;
14      ; ;
15 OD88 C0 12          PDL: PUSH YPOS ; Save values
16 OD8A C0 17          PUSH TSCROLL
17 OD8C C0 18          PUSH BFIX
18 OD8E E5 12          MOV A, YPOS ; Get current position
19 OD90 F5 F0          MOV B, A ; Save it in (B) also
20 OD92 B5 17 00      CJNE A, TSCROLL, PDLA ; IF upper fixed THEN
21 OD95 40 DE          PDLA: JC PILRT1 ; Just return
22 OD97 04             INC A
23 OD98 B5 18 04      CJNE A, BFIX, PDL1 ; IF last line of scroll THEN
24 OD9B 71 96          PDLO: ACALL CLRLINE ; clear current line
25 OD9D 80 D3          SJMP PILRET ; and return
26
27 OD9F 40 08          PDL1: JC PDL2 ; IF bottom fixed THEN
28 ODA1 B4 19 D1      CJNE A, #NUMLINES, PILRT1 ; IF 25th line THEN
29 ODA4 75 F0 18      MOV B, #MAXLINE ; clear 25th line
30 ODA7 80 F2          SJMP PDLO ; and return
31
32 ODA9 85 18 12      PDL2: MOV YPOS, BFIX ; Put cursor above fixed
33 ODAC 15 12          DEC YPOS
34 ODAE 85 F0 17      MOV TSCROLL, B ; Get saved current line
35 ODB1 12 0561       LCALL PLF ; Line feed to simulate delete
36 ODB4 A1 72          AJMP PILRET ; Return

```

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

```
1          ;;          APDL - ANSI perform delete line
2          ;
3          ;          *APDL* deletes the specified Pn number of lines at and
4          ;          following the cursor position.
5          ;
6          ;
7          ;          ENTRY none
8          ;
9          ;          EXIT none
10         ;
11         ;          USES A, B, DPTR, INDX1
12
13
14 ODB6 31 D1 APDL: ACALL GNP ; Get parameter
15 ODB8 70 01 JNZ APDLO ; IF no parameter or 0 THEN
16 ODBA 04 INC A ; bump to 1
17
18 ODBB FA APDLO: MOV INDX1, A ; Set up index
19 ODBC B1 88 APDL1: ACALL PDL ; REPEAT delete one line
20 ODBE DA FC DJNZ INDX1, APDL1 ; UNTIL count = 0
21 ODC0 22 RET
```

```

1      ;:          PIC - perform insert character
2      ;
3      ;          *PIC* move all characters that are at and to the right of the
4      ;          cursor one column to the right.  Cursor position does not
5      ;          change.
6      ;
7      ;
8      ;          ENTRY   none
9      ;
10     ;          EXIT    none
11     ;
12     ;          USES    all
13     ;
14     ;
15     ODC1  74      50      PIC:      MOV      A, #NUMCHARS      ; Calc number of moves to make
16     ODC3  C3
17     ODC4  95      11          CLR      C
18     ODC6  FA          SUBB     A, XPOS
19          MOV      INDX1, A
20     ODC7  E5      11          MOV      A, XPOS          ; Get current address
21     ODC9  85      12      F0      MOV      B, YPOS
22     ODCC  12      0503      LCALL   CLA
23
24     ODCF  78      20          MOV      PTR1, #
25     ODD1  79      00          MOV      PTR2, #0      ; Init value for 1st slot
26
27     ODD3  E0          PIC1:    MOVX     A, @DPTR
28     ODD4  FC          MOV      WORK, A      ; Save lower character
29     ODD5  53      83      EF      ANL     DPH, #ATRANL
30     ODD8  E0          MOVX     A, @DPTR
31     ODD9  FD          MOV      TEMP, A      ; Save lower attribute
32
33     ODDA  E9          MOV      A, PTR2
34     ODDB  F0          MOVX     @DPTR, A      ; Write lower attribute
35     ODDC  43      83      10      ORL     DPH, #CHRORL
36     ODDF  E8          MOV      A, PTR1
37     ODE0  F0          MOVX     @DPTR, A      ; Write lower character
38
39     ODE1  A8      04          MOV      PTR1, WORKA   ; Exchange character
40     ODE3  A9      05          MOV      PTR2, TEMPA   ; Exchange attribute
41
42     ODE5  A3          INC     DPTR          ; Bump up
43     ODE6  DA      EB          DJNZ   INDX1, PIC1    ; UNTIL all done
44     ODES  22          RET

```

```

1          ;;          APIC - ANSI perform insert character
2          ;
3          ;          *APIC* inserts the specified number of characters into
4          ;          the line.
5          ;
6          ;
7          ;          ENTRY   none
8          ;
9          ;          EXIT    none
10         ;
11         ;          USES   all
12         ;
13         ;
14 ODE9    31      D1      APIC:      ACALL   GNP          ; Get parameter
15 ODEB    70      01      APIC:      JNZ    APIC0       ; IF no parameter or 0 THEN
16 ODED    04      APIC:      INC     A          ; bump to 1
17
18 ODEE    FB      C1      APIC0:     MOV    INDX2, A      ; Setup index
19 ODEF    B1      FC      APIC1:     ACALL  PIC          ; REPEAT perform insert char
20 ODF1    DB      APIC1:     DJNZ   INDX2, APIC1   ; UNTIL count = 0
21 ODF3    22      APIC1:     RET

```

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

```

1          ;:          PDC - perform delete character
2          ;
3          ;
4          ;          *PDC* deletes the character at the cursor position by moving
5          ;          the remaining characters on the line to the left one column
6          ;          and inserting a space in the last column on the line.  Cursor
7          ;          position does not change.
8          ;
9          ;
10         ;          ENTRY  none
11         ;
12         ;          EXIT   none
13         ;
14         ;          USES   all
15
16 0DF4    74    4F          PDC:    MOV    A, #MAXCHAR          ; Calc number of moves
17 0DF6    C3          CLR    C
18 0DF7    95    11          SUBB   A, XPOS
19 0DF9    FA          MOV    INDX1, A
20 0DFA    74    4F          MOV    A, #MAXCHAR          ; Get address at end of line
21 0DFC    85    12    F0    MOV    B, YPOS
22 0DFF    12    0503      LCALL  CLA
23 0E02    E0          MOVX   A, @DPTR
24 0E03    F8          MOV    PTR1, A          ; Save upper character
25 0E04    74    20          MOV    A, #'
26 0E06    F0          MOVX   @DPTR, A          ; Clear last character
27 0E07    53    83    EF    ANL    DPH, #ATRANL
28 0E0A    E0          MOVX   A, @DPTR
29 0E0B    F9          MOV    PTR2, A          ; Save upper attribute
30 0E0C    E4          CLR    A
31 0E0D    F0          MOVX   @DPTR, A          ; Clear last attribute
32 0E0E    EA          MOV    A, INDX1
33 0E0F    60    20          JZ     PDC2            ; Exit out if last column
34
35 0E11    E5    82          PDC1:  MOV    A, DPL          ; Decrement DPTR without
36 0E13    24    FF          ADD    A, #-1          ; Using a "-CLR C"
37 0E15    F5    82          MOV    DPL, A          ; Instruction to save
38 0E17    E5    83          MOV    A, DPH          ; Every bit of time
39 0E19    34    FF          ADDC  A, #-1
40 0E1B    F5    83          MOV    DPH, A
41
42 0E1D    E0          MOVX   A, @DPTR
43 0E1E    FD          MOV    TEMP, A          ; Save lower attribute
44 0E1F    43    83    10    ORL    DPH, #CHRORL
45 0E22    E0          MOVX   A, @DPTR
46 0E23    FC          MOV    WORK, A          ; Save lower character
47
48 0E24    E8          MOV    A, PTR1
49 0E25    F0          MOVX   @DPTR, A          ; Write character
50 0E26    53    83    EF    ANL    DPH, #ATRANL
51 0E29    E9          MOV    A, PTR2
52 0E2A    F0          MOVX   @DPTR, A          ; Write attribute
53
54 0E2B    A8    04          MOV    PTR1, WORKA      ; Exchange character
55 0E2D    A9    05          MOV    PTR2, TEMP      ; Exchange attribute
56 0E2F    DA    E0          DJNZ  INDX1, PDC1      ; UNTIL all done
57 0E31    22          PDC2:  RET

```

```
1      ; ; APDC - ANSI perform delete character
2      ;
3      ; *APDC* deletes the specified number of characters from the
4      ; line.
5      ;
6      ;
7      ; ENTRY none
8      ;
9      ; EXIT none
10     ;
11     ; USES all
12     ;
13
14 0E32 31 D1 APDC: ACALL GNP ; Get parameter
15 0E34 70 01 JNZ APDC0 ; IF no parameter or 0 THEN
16 0E36 04 INC A ; bump to 1
17
18 0E37 FB APDC0: MOV INDX2, A ; Setup index
19 0E38 B1 F4 APDC1: ACALL PDC ; REPEAT perform delete char
20 0E3A DB FC DJNZ INDX2, APDC1 ; UNTIL count = 0
21 0E3C 22 RET
```

```

1      ;:          ZDSSBR - ZDS set baud rate
2      ;
3      ;          *ZDSSBR* sets the baud rate of the UART built into the 8051.
4      ;
5      ;
6      ;          ENTRY   none
7      ;
8      ;          EXIT    none
9      ;
10     ;          USES   all
11     ;
12     ;
13     0E3D    11    2A          ZDSSBR:    ACALL  SETDISP          ; Set dispatch and return
14     ;
15     0E3F    11    2F          ACALL  SETNORM          ; Restore dispatch address
16     0E41    C3          CLR      C
17     0E42    94    40          SUBB   A, #'0'          ; Subtract offset
18     0E44    40    18          JC     SBRET          ; Check for lower bounds
19     0E46    B4    0E    00          CJNE  A, #'('-'@'), ZDS0 ; Check for upper bounds
20     0E49    50    13          ZDS0:    JNC     SBRET
21     ;
22     ;          IFC     NE, SBR-$
23     ;          ERROR  ;ROUTINE 'SBR' MUST FOLLOW
24     ;          ENDC
25     ;
26     ;
27     ;
28     ;
29     ;:          SBR - set baud rate
30     ;
31     ;          *SBR* selects a baud rate given the value of which rate to
32     ;          select.
33     ;
34     ;
35     ;          ENTRY   (A) = which baud rate (0..13)
36     ;
37     ;
38     ;          EXIT    none
39     ;
40     ;          USES   A, DPTR
41     ;
42     0E4B    90    0E5F          SBR:    MOV    DPTR, #BRTAB          ; Point to baud rate table
43     0E4E    93          MOVVC  A, @A+DPTR          ; Get value from table
44     0E4F    60    0D          JZ     SBRET          ; IF zero THEN exit
45     0E51    F5    33          MOV    BAUDRATE, A          ; Save baud rate
46     0E53    F5    8D          MOV    TH1, A              ; Change baud rate
47     0E55    C2    9F          CLR   SMO                  ; Place UART into mode 1
48     0E57    B4    53    00          CJNE  A, #83, SBRO
49     0E5A    50    02          SBRO:  JNC     SBRET          ; IF baud <= 300 THEN
50     0E5C    D2    9F          SETB  SMO                  ; Place UART into mode 3
51     0E5E    22          SBRET:  RET

```



```
1          ;:          BRTAB - baud rate table
2          ;
3          ;
4          ;          *BRTAB* contains the timer values for baud rates
5
6          0E5F          BRTAB          EQU          $
7
8 0E5F          01          DB          1          ; 75 baud
9 0E60          52          DB          82          ; 110 baud
10 0E61          80          DB          128         ; 150 baud
11 0E62          C0          DB          192         ; 300 baud
12 0E63          E0          DB          224         ; 600 baud
13 0E64          F0          DB          240         ; 1200 baud
14 0E65          F5          DB          245         ; 1800 baud
15 0E66          00          DB          0          ; 2000 baud (not available)
16 0E67          F8          DB          248         ; 2400 baud
17 0E68          00          DB          0          ; 3600 baud (not available)
18 0E69          FC          DB          252         ; 4800 baud
19 0E6A          00          DB          0          ; 7200 baud (not available)
20 0E6B          FE          DB          254         ; 9600 baud
21 0E6C          FF          DB          255         ; 19200 baud
```

```

1 ..... ;: PSOF = put string in output FIFO
2 ..... ;
3 ..... ; *PSOF* places the character string pointed to by DPTR into
4 ..... ; the output FIFO. Final character has bit seven set.
5 ..... ;
6 ..... ;
7 ..... ; ENTRY (DPTR) = start of string
8 ..... ;
9 ..... ; EXIT none
10 ..... ;
11 ..... ; USES A, B, DPTR
12 ..... ;
13 ..... ;
14 0E6D E4 PSOF: CLR A ; REPEAT
15 0E6E 93 MOVC A, @A+DPTR ; set character
16 0E6F 20 E7 OD JB ACC.7, SOR ; IF bit 7 set THEN exit
17 0E72 C0 82 PUSH DPL ; save DPTR
18 0E74 C0 83 PUSH DPH
19 0E76 D1 A6 ACALL CPR2 ; place character on FIFO
20 0E78 D0 83 POP DPH ; restore DPTR
21 0E7A D0 82 POP DPL
22 0E7C A3 INC DPTR ; bump pointer
23 0E7D 80 EE SJMP PSOF ; UNTIL done
24 ..... ;
25 0E7F 54 7F SOR: ANL A, #7FH ; Mask off 7th bit
26 0E81 80 23 SJMP CPR2 ; Place last char on FIFO

```

```

1          ;;          CPR - cursor position report
2          ;
3          ;          *CPR* outputs the cursor position in the format of the mode
4          ;          currently set.
5          ;
6          ;
7          ;          ENTRY   none
8          ;
9          ;          EXIT    none
10         ;
11         ;          USES   all
12         ;
13
14 0E83    30    43    11    CPR:    JNB    HAZ, CPR1    ; IF Hazeltine THEN
15 0E86    E5    11          MOV    A, XPOS    ; set -xpos-
16 0E88    B4    20    00    CJCNE  A, #32, CPRO ;
17 0E8B    50    02    CPR0:    JNC    CPROA    ; IF value < 32 THEN
18 0E8D    24    60          ADD    A, #96    ; normalize
19 0E8F    D1    A6    CPR0A:  ACALL  CPR2    ; output value
20 0E91    E5    12          MOV    A, YPOS    ; set -ypos-
21 0E93    24    60          ADD    A, #96    ; normalize
22 0E95    80    0F          SJMP   CPR2    ; output value and return
23
24 0E97    90    0EA9    CPR1:  MOV    DPTR, #CPRM ; ELSE
25 0E9A    D1    6D          ACALL  PSQF    ; output ESC.Y
26 0E9C    E5    12          MOV    A, YPOS    ; set line number
27 0E9E    24    20          ADD    A, #' '    ; convert to ASCII
28 0EA0    D1    A6          ACALL  CPR2    ; output it
29 0EA2    E5    11          MOV    A, XPOS    ; set column number
30 0EA4    24    20          ADD    A, #' '    ; convert to ASCII
31 0EA6    02    022A    CPR2:  JMP    PCQF    ; output it and return
32
33 0EA9    1B    D?    CPRM:    DB     ESC, ('Y'+80H) ; ESC.Y

```

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

```

1          ;:          IDT - identify terminal
2          ;
3          ;          *IDT* identifies the terminal as a DEC VT52 so that existing
4          ;          DEC software which interrogates the console type will operate
5          ;          as it would with a VT52.
6          ;
7          ;
8          ;          ENTRY   none
9          ;
10         ;          EXIT    none
11         ;
12         ;          USES    all
13         ;
14         ;
15 OEAB    90      OEBO    IDT:      MOV    DPTR, #IDTM
16 OEAE    C1      6D      AJMP   PSOF          ; Output string and return
17
18 OEBO    1B      2F      CB      IDTM:   DB    ESC, '/', ('K'+80H) ; ESC / K
19
20
21
22
23         ;:          IDTT - identify terminal type
24         ;
25         ;          *IDTT* identifies the terminal as a Z-29 so that future
26         ;          software can interrogate the console type and see what
27         ;          version of the terminal is running.
28         ;
29         ;
30         ;          ENTRY   none
31         ;
32         ;          EXIT    none
33         ;
34         ;          USES    all
35         ;
36         ;
37 OEBC    11      2A      IDTT:   ACALL  SETDISP          ; Set dispatch and return
38
39 OEBC    11      2F      ACALL  SETNORM          ; Restore dispatch address
40 OEBC    B4      30      CJNE   A, #0, ZAR          ; IF not '0' THEN return
41 OEBA    90      OEBC    MOV    DPTR, #IDTTM
42 OEBC    C1      6D      AJMP   PSOF          ; Output string and return
43
44 OEBC    1B      69      42      IDTTM:  DB    ESC, 'iB', ('0'+80H) ; ESC i B 0
45 OEC2    B0

```

```
1          ;;          ADA - ANSI device attributes
2          ;
3          ;          *ADA* identifies the terminal as a DEC VT100 so that existing
4          ;          DEC software which interrogates the console type will operate
5          ;          as it would with a VT100.
6          ;
7          ;
8          ;          ENTRY   none
9          ;
10         ;          EXIT    none
11         ;
12         ;          USES    all
13         ;
14
15 OEC3    31    D1          ADA:          ACALL   GNP          ; Get parameter
16 OEC5    70    5E          ;          JNZ     ZAR          ; IF no parameter or 0 THEN
17 OEC7    90    OECC       ;          MOV     DPTR, #ADAM
18 OECA    C1    6D          ;          AJMP   PSOF         ; output string and return
19
20 OECC    1B    5B    3F    ADAM:         DB     ESC, '[?1;2', ('c'+80H)
    OECF    31    3B    32
    OED2    E3
```

```

1          ;:          ZTAB - ZDS mode tab functions
2          ;
3          ;          *ZTAB* does the tab function requested.
4          ;
5          ;
6          ;          ENTRY   none
7          ;
8          ;          EXIT    none
9          ;
10         ;          USES   all
11
12
13 0ED3     11     2A          ZTAB:   ACALL   SETDISP          ; Get dispatch address and return
14
15 0ED5     11     2F          ACALL   SETNORM          ; Restore dispatch address
16 0ED7     75     F0     03     MOV     B, #ZTABTL
17 0EDA     70     0EDF     MOV     DPTR, #ZTABT
18 0EDD     01     36          AJMP   STJMP          ; Jump to any routine
19
20
21
22
23         ;:          ZTABT - ZDS mode tab table
24         ;
25         ;          *ZTABT* contains valid parameters for the set/clear tab
26         ;          functions in ZDS mode.
27
28
29         0EDF          ZTABT     EQU     $
30
31 0EDF     30          DB     '0'          ; ESC . 0
32 0EE0     0627       DW     HTC          ; Horizontal tab clear
33
34 0EE2     33          DB     '3'          ; ESC . 3
35 0EE3     0599       DW     CLR TABS     ; Clear all tabs
36
37 0EE5     38          DB     '8'          ; ESC . 8
38 0EE6     0621       DW     HTS          ; Horizontal tab set
39
40         0003          ZTABTL     EQU     ($-ZTABT)/3          ; Table length

```

```

1          ;;          SCP - save cursor position
2          ;
3          ;          *SCP* saves current cursor position.
4          ;
5          ;
6          ;          ENTRY  none
7          ;
8          ;          EXIT   none
9          ;
10         ;          USES   none
11
12
13 0EE8    85    11    35    SCP:    MOV    XXPOS, XPOS          ; Save values
14 0EEB    85    12    36    MOV    XYPOS, YPOS
15 0EEE    22    RET
16
17
18
19
20         ;          ASCP - ANSI save cursor position
21         ;
22         ;          *ASCP* saves current position and attributes.
23         ;
24         ;
25         ;          ENTRY  none
26         ;
27         ;          EXIT   none
28         ;
29         ;          USES   all
30
31
32 0EEF    31    D1    ASCP:    ACALL  GNP          ; Get parameter if any
33 0EF1    40    F5    JC      SCP          ; IF no parameter THEN ok
34 0EF3    22    RET

```

```

1      ;:          USCP - unsave cursor position
2      ;
3      ;          *USCP* returns to position saved with the use of *SCP*.
4      ;
5      ;
6      ;          ENTRY   none
7      ;
8      ;          EXIT    none
9      ;
10     ;          USES   all
11     ;
12     ;
13 0EF4 85      35      11      USCP:      MOV     XPOS, XXPOS      ; Restore values
14 0EF7 E5      36
15 0EF9 B4      18      03
16 0EFC 30      15      02
17 0EFF F5      12
18 0F01 21      F4      USCP1:      MOV     YPOS, A          ; ELSE set position
19                                USCP2:      AJMP   XBLINAD       ; build address and return
20
21
22
23     ;:          AUSCP - ANSI unsave cursor position
24     ;
25     ;          *AUSCP* returns to position saved with the use of *SPCP*.
26     ;
27     ;
28     ;          ENTRY   none
29     ;
30     ;          EXIT    none
31     ;
32     ;          USES   all
33     ;
34     ;
35 0F03 31      D1      AUSCP:      ACALL  GNP              ; Get parameter if any
36 0F05 40      ED
37 0F07 22

```



```

1          ;;          ZATR - ZDS select attributes
2          ;
3          ;
4          ;          *ZATR* selects which attributes are to be turned on or off
5          ;          using the configuration of the lower 5 bits of the character.
6          ;
7          ;          ENTRY  none
8          ;
9          ;          EXIT   none
10         ;
11         ;          USES   all
12         ;
13         ;
14 0F08    11    2A          ZATR:      ACALL  SETDISP          ; Set dispatch and return
15
16 0F0A    11    2F          ACALL  SETNORM          ; Restore dispatch address
17 0F0C    C3
18 0F0D    94    30          CLR    C
19 0F0F    40    14          SUBB  A, #'0'
20 0F11    B4    20    00          JC   ZAR          ; IF below range THEN exit
21 0F14    50    0F          CJNE A, #32, ZATRO
22 0F16    13          ZATRO:      JNC   ZAR          ; IF above range THEN exit
23 0F17    92    08          RRC  A          ; Get reverse video
24 0F19    13          MOV  RVVA, C
25 0F1A    92    0A          RRC  A          ; Get blinkins
26 0F1C    13          MOV  BLINKA, C
27 0F1D    92    0B          RRC  A          ; Get half intensity
28 0F1F    13          MOV  HALFIA, C
29 0F20    92    09          RRC  A          ; Get underline
30 0F22    13          MOV  UNDLNA, C
31 0F23    92    0C          RRC  A          ; Get alternate char set
32 0F25    22          MOV  ALTCHARA, C
33         ;          ZAR:      RET
34
35
36
37         ;;          XATR - exit all attributes
38         ;
39         ;          *XATR* turns off all attributes so that none are active.
40         ;
41         ;
42         ;          ENTRY  none
43         ;
44         ;          EXIT   none
45         ;
46         ;          USES   none
47         ;
48         ;
49 0F26    53    21    F0          ZATR:      ANL  ATTRIBUTES, #0F0H          ; Mask off all attributes
50 0F29    22          RET

```

```

1      ;:      EHALF - enter half intensity mode.
2      ;
3      ;      *EHALF* places the terminal into half intensity mode.
4      ;
5      ;
6      ;      ENTRY   none
7      ;
8      ;      EXIT    none
9      ;
10     ;      USES   none
11     ;
12     ;
13     OF2A  D2    OB      EHALF:      SETB   HALFIA      ; Set half intensity attribute
14     OF2C  22
15     ;
16     ;
17     ;
18     ;
19     ;:      EUNDL - enter underline mode
20     ;
21     ;      *EUNDL* places the terminal into underline mode.
22     ;
23     ;
24     ;      ENTRY   none
25     ;
26     ;      EXIT    none
27     ;
28     ;      USES   none
29     ;
30     ;
31     OF2D  D2    09      EUNDL:      SETB   UNDLNA      ; Set underline attribute
32     OF2F  22
33     ;
34     ;
35     ;
36     ;
37     ;:      EBLNK - enter blink mode
38     ;
39     ;      *EBLNK* places the terminal into blink mode
40     ;
41     ;
42     ;      ENTRY   none
43     ;
44     ;      EXIT    none
45     ;
46     ;      USES   none
47     ;
48     ;
49     OF30  D2    0A      EBLNK:      SETB   BLINKA      ; Set blink attribute
50     OF32  22

```

```
1          ;;          ERVM - enter reverse video mode
2          ;
3          ;          *ERVM* places the terminal into reverse video mode.
4          ;
5          ;
6          ;          ENTRY   none
7          ;
8          ;          EXIT    none
9          ;
10         ;          USES    none
11
12
13 0F33  D2   08          ERVM:          SETB   RVVA          ; Set reverse video attribute
14 0F35  22
15
16
17
18
19         ;;          XRVM - exit reverse video mode
20         ;
21         ;          *XRVM* removes the terminal from reverse video mode.
22         ;
23         ;
24         ;          ENTRY   none
25         ;
26         ;          EXIT    none
27         ;
28         ;          USES    none
29
30
31 0F36  C2   08          XRVM:          CLR    RVVA          ; Clear reverse video attribute
32 0F38  22
```

```
1      ;:      EGM - enter graphics mode
2      ;
3      ;:      *EGM* places terminal into graphics mode.
4      ;
5      ;
6      ;
7      ;      ENTRY   none
8      ;
9      ;      EXIT    none
10     ;
11     ;      USES   none
12     ;
13     OF39  D2    OD      EGM:      SETB   GRPH      ; Set graphics mode
14     OF3B  22
15
16
17
18
19     ;:      XGM - exit graphic mode
20     ;
21     ;:      *XGM* exits the terminal out of graphic character mode.
22     ;
23     ;
24     ;
25     ;      ENTRY   none
26     ;
27     ;      EXIT    none
28     ;
29     ;      USES   none
30     ;
31     OF3C  C2    OD      XGM:      CLR    GRPH      ; Clear graphic mode
32     OF3E  22
```

```

1          ;;          GODES - G0 designator
2          ;          GIDES - G1 designator
3          ;
4          ;          *GODES* and *GIDES* assign a graphic character set to be
5          ;          associated with itself. If the character set is not found
6          ;          then no action is taken. The values returned for the
7          ;          different character sets are as follows:
8          ;
9          ;          Character | Internal
10         ;          received | value and meaning
11         ;          -----
12         ;          A          | 0 = United States (should be United Kingdom)
13         ;          B          | 0 = United States
14         ;          0          | 1 = United States with alternate graphic characters
15         ;          1          | 2 = Alternate character set
16         ;          2          | 3 = Alternate with alternate graphic characters
17         ;
18         ;
19         ;          ENTRY none
20         ;
21         ;          EXIT none
22         ;
23         ;          USES all
24         ;
25         ;
26 0F3F 11 2A          GODES: ACALL SETDISP          ; Set dispatch and return
27
28 0F41 11 2F          ACALL SETNORM          ; Restore dispatch address
29 0F43 C2 D5          CLR F0          ; Flag for GODES
30 0F45 80 08          SJMP GDES          ; Do the stuff
31
32
33
34
35 0F47 11 2A          GIDES: ACALL SETDISP          ; Set dispatch and return
36
37 0F49 11 2F          ACALL SETNORM          ; Restore dispatch address
38 0F4B D2 D5          SETB F0          ; Flag for GIDES
39 0F4D 80 00          SJMP GDES          ; Do the stuff
40
41
42
43
44 0F4F B4 41 03      GDES: CJNE A, #'A', GDES1          ; IF char = 'A' THEN
45 0F52 E4           GDES0: CLR A          ; value of zero
46 0F53 80 10          SJMP GDES3
47
48 0F55 B4 42 02      GDES1: CJNE A, #'B', GDES2          ; IF char = 'B' THEN
49 0F58 80 F8          SJMP GDES0          ; value of zero
50
51 0F5A C3           GDES2: CLR C          ; Subtract offset
52 0F5B 94 30          SUBB A, #'0'
53 0F5D 40 1B          JC GDESR          ; IF out of bounds THEN return
54 0F5F B4 03 00      GDES2A: CJNE A, #3, GDES2A          ; IF out of bounds THEN return
55 0F62 50 14          JNC GDESR
56 0F64 04           INC A          ; Bump into range
57

```



```

1          ; ;
2          ; ;
3          ; ;
4          ; *GOPICK* and *GIPICK* place the terminal into the mode
5          ; indicated by what has been set for the particular graphic
6          ; set.
7          ; ;
8          ; ;
9          ; ENTRY none
10         ; ;
11         ; EXIT none
12         ; ;
13         ; USES all
14         ; ;
15         ; ;
16 0F7B C2 12 GOPICK: CLR GSEL ; Clear select flag
17 0F7D E5 1F MOV A, GSET ; Get graphic set chosen
18 0F7F 80 05 SUMP GPICK
19         ; ;
20 0F81 D2 12 GIPICK: SETB GSEL ; Set select flag
21 0F83 E5 1F MOV A, GSET
22 0F85 C4 SWAP A ; Get graphic set chosen
23         ; ;
24 0F86 53 21 4F GPICK: ANL ATTRIBUTES, #01001111B ; Clean different modes out
25         ; ;
26 0F89 54 0F ANL A, #00001111B ; Mask to get number of set
27         ; ;
28 0F8B 90 0F92 MOV DPTR, #GPICKT ; Get address of table
29 0F8E 93 MOV A, @A+DPTR ; Get value from table
30 0F8F 42 21 ORL ATTRIBUTES, A ; Set as indicated by table
31 0F91 22 RET
32         ; ;
33         ; ;
34         ; ;
35         ; ;
36         ; ;
37         ; ;
38         ; *GPICKT* contains values that will be ORed into the
39         ; attribute byte to set the different character set modes
40         ; as indicated by the graphic mode defined above.
41         ; ;
42         ; ;
43         ; OF92 GPICKT EQU $
44         ; ;
45 0F92 00 DB 00000000B ; 0 - norm char
46         ; ;
47 0F93 A0 DB 10100000B ; 1 - norm char and alt graphic set
48         ; ;
49 0F94 10 DB 00010000B ; 2 - alt char set
50         ; ;
51 0F95 B0 DB 10110000B ; 3 - alt char and graphic set

```

```

1          ;:          SBLR - set blink rate
2          ;
3          ;
4          ;          *SBLR* sets the blink rate in increments of 1/30 second. The
5          ;          default value is 32/30 second. If a value of 255 is received
6          ;          then all blinking data is suppressed. If a value of 0
7          ;          is received then all blinking data is continuously on.
8          ;
9          ;
10         ;          ENTRY   none
11         ;
12         ;          EXIT    none
13         ;
14         ;          USES    all
15
16 0F96    31    D1          SBLR:          ACALL   GNP
17 0F98    50    02          JNC      SB1          ; IF no parameter THEN
18 0F9A    74    20          MOV     A, #32          ; set to default value
19
20 0F9C    F5    37          SB1:          MOV     BLNKRATE, A          ; Set blink rate
21 0F9E    60    06          JZ      SB2          ; IF not zero THEN
22 0FA0    B4    FF    0C          CJNE   A, #255, SBRT          ; IF max value THEN
23 0FA3    D3          SETB   C          ; flas off and
24 0FA4    80    01          SJMP   SB3          ; set output
25
26 0FA6    C3          SB2:          CLR     C          ; ELSE IF zero THEN flas on
27
28 0FA7    92    01          SB3:          MOV     BLNKF, C          ; Set output of blink
29 0FA9    E5    20          MOV     A, XLATCH
30 0FAB    90    3000          MOV     DPTR, #MLATCH
31 0FAE    F0          MOVX   @DPTR, A          ; Set latch
32
33 0FAF    75    38    00          SBRT:          MOV     BLNKCNT, #0          ; Zero count and return
34 0FB2    22          RET

```



```
1          ;;          EKI - enable keyboard input
2          ;
3          ;          *EKI* resets the flag which disables input from the keyboard.
4          ;
5          ;
6          ;          ENTRY   none
7          ;
8          ;          EXIT    none
9          ;
10         ;          USES   all
11
12
13 0FB3    C2    1F          EKI:    CLR    KBDISF
14 0FB5    74    89          MOV    A, #KBC_EK          ; Enable keyboard
15 0FB7    61    2A          AJMP   XPCKB          ; Output and return
16
17
18
19
20         ;;          DKI - disable keyboard input
21         ;
22         ;          *DKI* sets the flag which disables input from the keyboard.
23         ;
24         ;
25         ;          ENTRY   none
26         ;
27         ;          EXIT    none
28         ;
29         ;          USES   all
30
31
32 0FB9    D2    1F          DKI:    SETB   KBDISF
33 0FBB    74    88          MOV    A, #KBC_DK          ; Disable keyboard
34 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 OFBF B2 35 KBCPLK: CPL CAPLOCKF ; Tossle caps locked flas
15 OFC1 74 84 MOV A, #KBC_EL
16 OFC3 20 35 02 JB CAPLOCKF, KC1
17 OFC6 74 85 MOV A, #KBC_XL
18 OFC8 71 2A KC1: ACALL XPCKB ; Output command to keyboard
19 OFCA 61 23 AJMP XR2_DISP ; Diselay status line & return

```

```

1          ; ;          KBBREAK - keyboard break key
2          ;
3          ;
4          ;          *KBBREAK* indicates that the break key on the keyboard has
5          ;          been depressed. The terminal should then hold the serial
6          ;          output line low for a length of time.
7          ;
8          ;          ENTRY   none
9          ;
10         ;          EXIT    none
11        ;
12        ;          USES    all
13
14
15 OFCC    C2    B1          KBBREAK:    CLR    TXD          ; Set break
16 OFCE    7A    00          MOV    INDX1, #0
17 OFDO    7B    50          MOV    INDX2, #80
18 OFD2    12    02B6       BR1:      LCALL  FCKB          ; REPEAT look at keyboard
19 OFD5    40    05          JC     BR2
20 OFD7    B4    89    02    CJNE   A, #KB_BREAK, BR2      ; IF another break THEN
21 OFDA    80    F0          SJMP  KBBREAK        ; start again
22 OFDC    DA    F4          BR2:      DJNZ  INDX1, BR1      ; UNTIL done
23 OFDE    DB    F2          DJNZ  INDX2, BR1      ; UNTIL all done
24 OFE0    D2    B1          SETB  TXD          ; Back to normal
25 OFE2    22          RET

```

```

1 ; KBSCROLL = keyboard "NO SCROLL" key
2 ;
3 ; *KBSCROLL* indicates that the "NO SCROLL" key on the
4 ; keyboard has been depressed.
5 ;
6 ;
7 ; ENTRY none
8 ;
9 ; EXIT none
10 ;
11 ; USES all
12 ;
13 ;
14 OFE3 20 36 0A KBSCROLL: JB HSMODEF, KBSCR2 ; IF not hold screen mode THEN
15 OFE6 74 13 ; ; start with XOFF
16 OFE8 30 11 02 JNB UXOFFSENT, KBSCR1 ; IF user XOFF last sent THEN
17 OFEB 74 11 MOV A, #XON ; send an XON now instead
18 ;
19 OFED 02 022A KBSCR1: LJMPC PCOF ; place in output FIFO and return
20 ;
21 OFF0 C2 22 KBSCR2: CLR HSSTOPF ; ELSE allow screen to run
22 OFF2 05 1E INC HSLINE ; bump line count
23 OFF4 30 1C 07 JNB SHIFTF, KSR ; IF shift down THEN
24 OFF7 E5 18 MOV A, BFIX ; calculate number
25 OFF9 C3 CLR C ; between pages
26 OFFA 95 17 SUBB A, TSCROLL
27 OFFC F5 1E MOV HSLINE, A
28 OFFE 22 KSR: RET

```

1

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

```
1          ;:          ROM #2 vectors
2          ;
3          ;
4          ;          These locations are provided to facilitate future modifications
5          ;          without a change to the masked processor. Each location is
6          ;          accessed with a -CALL- and control is gained back with a -RET-.
7          ;
8          ;          Initialization of terminal.
9          ;
10         1000          ORG      R2_IN
11 1000 02 10AF          JMP      ALLINIT
12
13
14         ;          Soft initialization of terminal.
15         ;
16         1003          ORG      R2_IN2
17 1003 02 10E4          JMP      SOFTINIT
18
19
20         ;          Start DMA routine.
21         ;
22         1006          ORG      R2_STARTDMA
23 1006 02 11DC          LJMP     STARTDMA
24
25
26         ;          Stop DMA routine.
27         ;
28         1009          ORG      R2_STOPDMA
29 1009 02 11FD          LJMP     STOPDMA
30
31
32         ;          Fill screen routine.
33         ;
34         100C          ORG      R2_FILL
35 100C 02 1262          LJMP     STFILL
36
37
38         ;          Interrupt request.
39         ;
40         100F          ORG      R2_IRQ
41 100F 02 1329          LJMP     IRQ
42
43
44         ;          Serial interrupt.
45         ;
46         1012          ORG      R2_SERIAL
47 1012 02 12CB          LJMP     SERIAL
48
49
50         ;          Transmit character from serial port.
51         ;
52         1015          ORG      R2_XMT
53 1015 02 139B          LJMP     TRANSMIT
54
55
56         ;          Send XON to host
57         1018          ORG      R2_XON
```

```
.....
1 1018 02 13D5 LJMP DOXDN
2
3
4 ; Initialize the cursor format.
5 ;
6 101B 101B ORG R2_ICUR
7 101B 02 117B LJMP SETCF
8
9
10 ; Initialize the keyboard
11 ;
12 101E 101E ORG R2_IKB
13 101E 02 119E LJMP INITKB
14
15
16 ; Setup mode vector
17 ;
18 1021 1021 ORG R2_SETUP
19 1021 02 1417 LJMP SETUP
20
21
22 ; Background loop hook.
23 ;
24 1024 1024 ORG R2_BACK
25 1024 22 RET
26
27
28 ; Keyboard input hook.
29 ;
30 1027 1027 ORG R2_KY1
31 1027 22 RET
32
33
34 ; Keyboard output hook.
35 ;
36 102A 102A ORG R2_KY2
37 102A 22 RET
38
39
40 ; Escape parsing hook.
41 ;
42 102D 102D ORG R2_ESC
43 102D 22 RET
44
45
46 ; Transmit character.
47 ;
48 1030 1030 ORG R2_TC
49 1030 02 108B LJMP SXMTC
50
51
52 ; Transmit line.
53 ;
54 1033 1033 ORG R2_TL
55 1033 02 10A8 LJMP SXMTL
56
57
.....
```

```
1 ; Transmit page.
2 ;
3 1036 02 1DAC ORG R2_TP
4 1036 02 1DAC LJMP XMTP
5
6
7 ; Transmit 25th line.
8 ;
9 1039 02 1DC6 ORG R2_T25L
10 1039 02 1DC6 LJMP XMT25
11
12
13 ; Print page.
14 ;
15 103C 02 1DD2 ORG R2_PRNT
16 103C 02 1DD2 LJMP PRINT
17
18
19 ; Display clock.
20 ;
21 103F 02 1E1F ORG R2_DCLK
22 103F 02 1E1F LJMP DCLK
23
24
25 ; Display status.
26 ;
27 1042 02 1E69 ORG R2_DISP
28 1042 02 1E69 LJMP DISPSTAT
```



```

1          ; ;          INIT - initialize terminal
2          ; ;
3          ; ;          *INIT* sets up internal RAM from the ROM and EAROM, reconfigures
4          ; ;          screen tables and initializes processor bytes.
5          ; ;
6          ; ;
7          ; ;          ENTRY none
8          ; ;
9          ; ;          EXIT none
10         ; ;
11         ; ;          USES all
12
13
14 1045    75    D0    00    INIT:    MOV    PSW, #BANK0    ; Select bank 0
15
16 1048    78    0C
17 104A    90    1FD1
18 104D    7A    19
19
20 104F    11    A7
21
22 1051    78    35
23 1053    7A    0B
24
25 1055    11    A7
26
27         ; ;          Recall from EAROM
28         ; ;
29 1057    C2    03
30 1059    12    1E18
31 105C    D2    03
32 105E    12    1E18
33
34 1061    78    25
35 1063    90    7000
36 1066    7A    10
37
38 1068    E0
39 1069    54    0F    IN3:    MOVX   A, @DPTR    ; REPEAT set upper nibble
40 106B    C4
41 106C    FD
42 106D    A3
43 106E    E0
44 106F    54    0F
45 1071    4D
46 1072    F6
47 1073    A3
48 1074    08
49 1075    DA    F1
50
51 1077    C2    8C
52 1079    75    89    23
53 107C    75    98    5A
54
55 107F    E5    33
56 1081    F5    8D
57 1083    B4    53    00    CJNE   A, #83, IN3B

```



```

1          ;:          ALLINIT - initialize all of terminal
2          ;
3          ;          *ALLINIT* initializes everything including the CRTC.
4          ;
5          ;
6          ;          ENTRY   none
7          ;
8          ;          EXIT    none
9          ;
10         ;          USES   all
11
12
13 10AF    C2    AF          ALLINIT:    CLR    EA          ; Disable interrupts
14
15 10B1    75    90    CF          MOV    P1, #11001111B ; Initialize port #1
16
17 10B4    75    40    00          MOV    ERRORS, #0 ; Initialize errors here
18 10B7    12    1EE4          CALL   KBCHK ; Check keyboard first (timing)
19
20 10BA    11    45          ACALL  INIT ; Initialize everything
21
22 10BC    E4          CLR    A
23 10BD    78    47          MOV    PTR1, #LORDER ; Setup to initialize
24 10BF    7A    19          MOV    INDX1, #NUMLINES ; the -LORDER- table entries
25
26 10C1    F6          ALLI1:    MOV    @PTR1, A ; REPEAT place entry into table
27 10C2    08          INC    PTR1 ; bump RAM pointer
28 10C3    04          INC    A ; bump count
29 10C4    DA    FB          DJNZ  INDX1, ALLI1 ; UNTIL done
30
31 10C6    75    B8    03          MOV    IP, #IPINIT ; Interrupt priorities
32
33 10C9    31    1E          ACALL  INITCRT ; Start CRT controller
34 10CB    12    1ED9          CALL   ROMCHK ; Check ROM
35 10CE    12    1EF1          CALL   RAMCHK ; Check RAM
36
37 10D1    75    A8    95          MOV    IE, #IEINIT ; Enable interrupts
38
39 10D4    12    0BAC          CALL   CLRS ; Clear screen
40 10D7    75    F0    18          MOV    B, #MAXLINE
41 10DA    12    0B96          CALL   CLRLINE ; Clear 25th line
42 10DD    D2    24          SETB  VSPF ; Flash screen to be turned on
43 10DF    12    1F2C          CALL   ERROR ; Display any errors
44 10E2    21    9E          AJMP  INITKB ; Initialize keyboard & return

```

```

1          ;:          SOFTINIT - soft initialize terminal
2          ;
3          ;          *SOFTINIT* initializes the terminal but does not destroy the
4          ;          current settings such as caps lock, on/off line, and time on
5          ;          the clock.
6          ;
7          ;
8          ;          ENTRY   none
9          ;
10         ;          EXIT    none
11        ;
12        ;          USES    all
13        ;
14
15 10E4    C0    3A          SOFTINIT:    PUSH    TCNT          ; Save clock information
16 10E6    C0    3B          PUSH    TSEC
17 10E8    C0    3C          PUSH    TMIN
18 10EA    C0    3D          PUSH    THOUR
19 10EC    A2    2E          MOV     C, ONLINE
20 10EE    92    E0          MOV     ACC.0, C          ; Save online flag
21 10F0    A2    35          MOV     C, CAPLOCKF
22 10F2    92    E1          MOV     ACC.1, C          ; Save caps locked flag
23 10F4    C0    E0          PUSH   ACC
24
25 10F6    75    90    CB    MOV     P1, #11001011B    ; Initialize port #1
26
27 10F9    11    45          ACALL  INIT          ; Initialize everything
28
29 10FB    D0    E0          POP     ACC
30 10FD    A2    E1          MOV     C, ACC.1
31 10FF    92    35          MOV     CAPLOCKF, C      ; Restore caps locked flag
32 1101    A2    E0          MOV     C, ACC.0
33 1103    92    2E          MOV     ONLINE, C        ; Restore online flag
34 1105    D0    3D          POP     THOUR          ; Restore clock information
35 1107    D0    3C          POP     TMIN
36 1109    D0    3B          POP     TSEC
37 110B    D0    3A          POP     TCNT
38
39 110D    31    7B          ACALL  SETCF          ; Set cursor format
40
41 110F    12    0BAC        CALL   CLRS          ; Clear screen
42 1112    20    37    06    JB     L25ON, SOFTI1    ; IF 25th line is off THEN
43 1115    75    F0    18    MOV     B, #MAXLINE
44 1118    02    0B96        JMP     CLRLINE        ; clear 25th line
45
46 111B    02    1E69        SOFTI1:    JMP     DISPSTAT      ; ELSE display status info

```

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

```

1          ;;;          INITCRT - initialize CRT controller
2          ;
3          ;
4          ;          *INITCRT* does everything needed to set the CRT controller
5          ;          up and running properly.
6          ;
7          ;          ENTRY   none
8          ;
9          ;          EXIT    none
10         ;
11         ;          USES   A, DPTR
12         ;
13         ;
14 111E    C0    A8          INITCRT:    PUSH    IE          ; Save interrupt setup
15 1120    C2    AF          CLR        EA          ; Disable interrupts
16
17 1122    75    09    02    MOV        DMLNA, #2          ; Start at first line to DMA
18 1125    75    0B    A0    MOV        DMADRL, #LOW (DMAMEM+160)
19 1128    75    0A    60    MOV        DMADRH, #HIGH (DMAMEM+160)
20
21 112B    D2    97          SETB    CLKRUN          ; Let the clock run
22
23 112D    31    52          ACALL    SETCRT          ; Set CRT controller
24
25 112F    D2    88          SETB    ITO          ; Int 0 transition activated
26 1131    53    88    D1    ANL        TCON, #11010001B ; Clear ITI, IEO, IEI, TFO
27
28 1134    90    2001    MOV        DPTR, #DCMDREG
29 1137    74    E0    MOV        A, #CSET
30 1139    F0          MOVX    @DPTR, A          ; Preset counters
31
32 113A    C2    97          CLR        CLKRUN          ; Turn off clock
33
34 113C    74    20    MOV        A, #CSTRT
35 113E    F0          MOVX    @DPTR, A          ; Start CRT controller
36
37 113F    D2    97          SETB    CLKRUN          ; Start clock
38
39 1141    74    A0    MOV        A, #CINT
40 1143    F0          MOVX    @DPTR, A          ; Enable VRTC interrupts
41
42          IFC        NE, DSTAREG-DCMDREG
43          ERROR    ; 'DSTAREG' and 'DCMDREG' are different
44          ENDC
45
46          ;          MOV        DPTR, #DSTAREG
47
48 1144    E0          MOVX    A, @DPTR          ; Get status register
49 1145    44    B3    ORL        A, #10110011B ; Fill in dont cares
50 1147    64    08    XRL        A, #00001000B ; Toggle proper positions
51 1149    F4    CPL        A
52 114A    60    03    JZ        IC1          ; IF not zero THEN
53 114C    43    40    04    ORL        ERRORS, #00000100B ; flas error #2
54
55 114F    D0    A8          POP        IE          ; Enable old interrupt status
56 1151    22          RET

```

```

1          ;:          SETCRT - set CRT controller (reset 8276)
2          ;
3          ;          *SETCRT* initializes the CRT controller screen parameter bytes.
4          ;
5          ;
6          ;          ENTRY   none
7          ;
8          ;          EXIT    none
9          ;
10         ;          USES   A, DPTR
11         ;
12         ;
13 1152    90    2001    SETCRT:    MOV    DPTR, #DSTAREG
14 1155    E0                MOVX   A, @DPTR                ; Clear status register
15
16         ;          IFC    NE,DCMDREG-DSTAREG
17         ;          ERROR  !'DCMDREG' and 'DSTAREG' are different
18         ;          ENDC
19
20         ;          MOV    DPTR, #DCMDREG
21
22 1156    74    00                MOV    A, #CRESET
23 1158    F0                MOVX   @DPTR, A                ; Reset and stop display
24
25 1159    90    2000
26 115C    74    4F                MOV    DPTR, #DDATREG
27 115E    F0                MOV    A, #SCRN1
28         ;          MOVX   @DPTR, A                ; Output screen parameter bytes
29 115F    74    98
30 1161    20    39    02                MOV    A, #SCRN2A
31 1164    74    DC                JB     FREQ, SC1                ; Check for 50/60 Hz
32 1166    F0                MOV    A, #SCRN2B
33         ;          MOVX   @DPTR, A
34 1167    74    99
35 1169    F0                MOV    A, #SCRN3
36         ;          MOVX   @DPTR, A
37 116A    74    4E
38 116C    20    39    02                MOV    A, #SCRN4A
39 116F    74    4F                JB     FREQ, SC2                ; Check for 50/60 Hz
40 1171    A2    38                MOV    A, #SCRN4B
41 1173    92    E5                MOV    C, CRNBK                ; Get cursor no blink flag
42 1175    A2    32                MOV    ACC.5, C
43 1177    92    E4                MOV    C, CRUL                ; Get cursor underline flag
44 1179    F0                MOV    ACC.4, C
45 117A    22                MOVX   @DPTR, A
          ;          RET

```

```
1          ;;          SETCF - set cursor format
2          ;
3          ;          *SETCF* initializes the CRT controller for the cursor format
4          ;          indicated.
5          ;
6          ;
7          ;          ENTRY none
8          ;
9          ;          EXIT none
10         ;
11         ;          USES A, DPTR
12
13
14 117B    31    52          SETCF:          ACALL SETCRT          ; Initialize screen registers
15
16 117D    90    2001        MOV DPTR, #DCMDREG
17 1180    74    20          MOV A, #CSTRT
18 1182    F0          MOVX @DPTR, A          ; Start CRT controller
19
20 1183    22          RET
```

```

1          ;:          CURSOR - update cursor position
2          ;
3          ;
4          ;          *CURSOR* sends the current cursor address to the CRTC and
5          ;          disables cursor if data is > 600 baud or display of cursor
6          ;          is disabled.
7          ;
8          ;
9          ;          ENTRY   none
10         ;
11         ;
12         ;          EXIT   none
13         ;
14         ;          USES   A, DPTR
15 1184    90    2001    CURSOR:    MOV    DPTR, #DCMDREG
16 1187    74    80      MOV    A, #CCRSR
17 1189    F0      MOVX   @DPTR, A          ; Load cursor position
18
19 118A    90    2000    MOV    DPTR, #DDATREG
20 118D    E5    11      MOV    A, XPOS
21 118F    F0      MOVX   @DPTR, A          ; Load cursor X-position
22
23 1190    E5    12      MOV    A, YPOS
24 1192    30    1A    03    JNB   ENBLCUR, CR1      ; IF cursor display disabled OR
25 1195    30    14    02    JNB   DX, CR2          ; IF char rcvd since last time
26 1198    74    1A      MOV    A, #NUMLINES + 1      ; put in non-existing region
27 119A    F0      MOVX   @DPTR, A          ; ELSE put cursor Y-position
28
29 119B    C2    14      CLR   DX          ; and clear -DX-
30 119D    22      RET

```



```

1          ;:          INITKB - initialize keyboard
2          ;
3          ;          *INITKB* resets the keyboard to whatever settings the terminal
4          ;          has at the current moment.
5          ;
6          ;
7          ;          ENTRY   none
8          ;
9          ;          EXIT    none
10         ;
11         ;          USES   all
12         ;
13
14 119E          INITKB:      IFC      NE,KBC_XL - KBC_EL - 1
15                ERROR     ;:This code will not work'
16                ENDC
17 119E      74      84
18 11A0      20      35      01      MOV     A, #KBC_EL
19 11A3      04                JB      CAPLOCKF, IKB1      ; IF not caps locked THEN
20 11A4      31      C8      IKB1:   ACALL   IKB6                ; output caps unlocked
21                                     ; (save bytes this way)
22
23                IFC      NE,KBC_ONLN - KBC_OFFLN - 1
24                ERROR     ;:This code will not work'
25                ENDC
26 11A6      74      8A      01      MOV     A, #KBC_OFFLN
27 11A8      30      2E                JNB    ONLINE, IKB2      ; IF on line THEN
28 11AB      04                INC     A                ; output on line command
29 11AC      31      C8      IKB2:   ACALL   IKB6
30
31                IFC      NE,KBC_EK - KBC_DK - 1
32                ERROR     ;:This code will not work'
33                ENDC
34 11AE      74      88      01      MOV     A, #KBC_DK
35 11B0      20      1F                JB      KBDISF, IKB3      ; IF keyboard disabled THEN
36 11B3      04                INC     A                ; enable keyboard
37 11B4      31      C8      IKB3:   ACALL   IKB6
38
39                IFC      NE,KBC_DC - KBC_EC - 1
40                ERROR     ;:This code will not work'
41                ENDC
42 11B6      74      82      01      MOV     A, #KBC_EC
43 11B8      20      33                JB      CLKF, IKB4        ; IF key click disabled THEN
44 11BB      04                INC     A                ; output no key click
45 11BC      31      C8      IKB4:   ACALL   IKB6
46
47                IFC      NE,KBC_DR - KBC_ER - 1
48                ERROR     ;:This code will not work'
49                ENDC
50 11BE      74      86      01      MOV     A, #KBC_ER
51 11C0      20      34                JB      REPF, IKB5        ; IF auto repeat disabled THEN
52 11C3      04                INC     A                ; no auto repeat
53 11C4      31      C8      IKB5:   ACALL   IKB6
54
55 11C6      74      80                MOV     A, #KBC_ID        ; Have it send it's ID number
56 11C8      02      026D      IKB6:   JMP     PCKB

```

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

```

1 ..... ;: ..... TERMINIT - terminal initialization .....
2 ..... ; .....
3 ..... ; ..... *TERMINIT* initializes the terminal for ZDS and ANSI .....
4 ..... ; ..... emulation mode. This places the terminal in whatever mode .....
5 ..... ; ..... the terminal must be in to emulate those modes. ....
6 ..... ; .....
7 ..... ; .....
8 ..... ; ..... ENTRY none .....
9 ..... ; .....
10 ..... ; ..... EXIT none .....
11 ..... ; .....
12 ..... ; ..... USES all .....
13 ..... ; .....
14 ..... ; .....
15 11CB 53 21 F0 ..... TERMINIT: ANL ATTRIBUTES, #OF0H ; Clear attributes .....
16 11CE D2 1B ..... SETB ERM ; Set erasure mode .....
17 11D0 D2 25 ..... SETB GATM ; Set guarded area xmt mode .....
18 11D2 80 04 ..... SJMP HIR ; Clear protected and return .....
19 ..... ; .....
20 ..... ; .....
21 ..... ; .....
22 ..... ; .....
23 ..... ;: ..... HAZINIT - Hazeltine initialization .....
24 ..... ; .....
25 ..... ; ..... *HAZINIT* initializes the terminal for the Hazeltine 1500 .....
26 ..... ; ..... emulation mode. This places the terminal in whatever mode .....
27 ..... ; ..... the terminal must be in to emulate the Hazeltine 1500. ....
28 ..... ; .....
29 ..... ; .....
30 ..... ; ..... ENTRY none .....
31 ..... ; .....
32 ..... ; ..... EXIT none .....
33 ..... ; .....
34 ..... ; ..... USES all .....
35 ..... ; .....
36 ..... ; .....
37 11D4 D2 0B ..... HAZINIT: SETB HALFIA ; Starts in background .....
38 11D6 D2 1B ..... SETB ERM ; Reset erasure mode .....
39 11D8 75 3F 00 ..... HIR: MOV PFIELD, #0 ; No protected fields .....
40 11DB 22 ..... RET .....

```

```

1          ; ;          STARTDMA - start psuedo DMA transfer
2          ;
3          ;          *STARTDMA* can only be interrupted by the stop DMA routine.
4          ;          It is initiated by pushing the address to start DMA onto the
5          ;          stack and executing a -RET-. The hardware takes over and
6          ;          executes an instruction that is hard wired to the data bus
7          ;          and initiates reads from the video memory to the CRT controller.
8          ;
9          ;
10         ;          ENTRY   none
11         ;
12         ;          EXIT    none
13         ;
14         ;          USES    none
15         ;
16         ;
17 11DC    C0    D0          STARTDMA:  PUSH   PSW          ; Save processor status
18 11DE    C0    E0          PUSH   ACC
19 11E0    C0    F0          PUSH   B
20
21 11E2    75    A8    82    MOV     IE, #IEDMA      ; Select interrupts for DMA
22 11E5    D2    B4          SETB   DMATYPE      ; Set type for DMA transfer
23
24 11E7    E5    09          MOV     A, DMLNA
25 11E9    70    05          JNZ    SD1          ; IF DMA for line #0 THEN
26 11EB    30    24    02    JNB    VSPF, SD1      ; IF no suppression THEN
27 11EE    C2    92          CLR     VSP          ; always turn video on
28
29 11F0    C0    0B          SD1:    PUSH   DMADRL      ; Push DMA start address
30 11F2    C0    0A          PUSH   DMADRH      ; onto the stack
31 11F4    75    8C    FF    MOV     TH0, #0FFH  ; Initialize timer values
32 11F7    75    8A    D6    MOV     TLO, #0D6H  ; For 80 characters
33 11FA    D2    8C          SETB   TR0          ; Start timer running
34 11FC    32          RETI          ; Start psuedo DMA

```

```

1          ; ;          STOPDMA - stop pseudo DMA transfer
2          ;
3          ;
4          ;          *STOPDMA* is the other half of the DMA transfer routine.
5          ;          It stops the timer, cleans the stack, sets -DMATYPE- to it's
6          ;          normal value, and enables interrupts. If it was filling memory
7          ;          then it finishes up that otherwise it continues and calculates
8          ;          the next line number, address, and indexes for DMA.
9          ;
10         ;
11         ;          ENTRY   none
12         ;
13         ;          EXIT    none
14         ;
15         ;          USES    none
16
17 11FD    C2    8C          STOPDMA:   CLR    TRO          ; 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            ; set last address from
23 1209    D0    82          POP     DPL            ; the stack
24 120B    AD    F0          MOV     TEMP, B        ; set 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, #1        ; ELSE set up and
28 1213    53    83    CF    ANL     DPH, #DMAANL
29 1216    43    83    10    DRL     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    10    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          JC      DM3            ; do usual stuff
50 123A    B9    1A    00    DM1C:   CJNE   DMLN, #NUMLINES+1, DM1D
51 123D    40    1C          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          JC      DM3            ; IF line > 25 THEN
57 1248    79    00          DM2:    MOV     DMLN, #0    ; wrap around

```



```

1          ;:          STFILL - start filling with spaces
2          ;
3          ;          *STFILL* takes the number of characters and location in
4          ;          screen memory and fills them with 20H and places the
5          ;          attribute given into attribute memory. It takes into
6          ;          account the mode that ERM is in.
7          ;
8          ;
9          ;          ENTRY (A) = number of characters less than 127
10         ;          (B) = attribute to write
11         ;          (DPTR) = character memory address to start from
12         ;
13         ;          EXIT none
14         ;
15         ;          USES A, B, DPTR, WORK2, TEMP, INDX2, TIMER 0
16         ;
17
18 1262 FB          STFILL: MOV INDX2, A          ; Save count
19 1263 AF F0      MOV WORK2, B          ; Save attributes
20
21 1265 BF 00 3D   CJNE WORK2, #0, SFB    ; IF attributes THEN special
22
23 1268 30 1B 04  JNB ERM, SFA          ; IF not erasure mode THEN
24
25 126B E5 3F     MOV A, PFIELD          ; IF protected fields THEN
26 126D 70 36     JNZ SFB              ; special
27
28 126F 53 83 EF   SFA: ANL DPH, #ATRANL    ; convert to DMA memory
29 1272 43 83 20   ORL DPH, #DMAORL
30 1275 EB        MOV A, INDX2          ; set count back
31 1276 23        RL A              ; multiply by 2
32 1277 75 F0 03  MOV B, #3          ; set up for divide by 3
33 127A 84        DIV AB           ; place remainder in (B)
34 127B C5 F0     XCH A, B          ; switch to work with (B)
35 127D B4 02 06  CJNE A, #2, SF1     ; IF remainder = 2 THEN
36 1280 05 F0     INC B              ; bump clock value
37 1282 74 00     MOV A, #0         ; remainder := 0
38 1284 80 07     SJMP SF2          ; ELSE
39
40 1286 B4 00 04   SF1: CJNE A, #0, SF2    ; IF remainder = 0 THEN
41 1289 74 02     MOV A, #2          ; remainder := 2
42 128B 15 F0     DEC B              ; decrement clock count
43 128D C5 F0     SF2: XCH A, B      ; switch back
44 128F F4        CPL A              ; take 2's complement of (A)
45 1290 04        INC A
46
47 1291 C2 AF     CLR EA              ; disable interrupts
48 1293 00        NOP              ; give time to clear
49 1294 75 A8 82  MOV IE, #IEDMA     ; init for DMA
50 1297 C2 B4     CLR DMATYPE       ; set type for clear memory
51 1299 C0 82     PUSH DPL          ; push DMA start address
52 129B C0 83     PUSH DPH          ; onto the stack
53 129D 75 8C FF  MOV TH0, #OFFH     ; set timer 0
54 12A0 F5 8A     MOV TLO, A          ; for number of characters
55 12A2 D2 8C     SETB TRO         ; start timer running
56 12A4 22        RET              ; start DMA
57

```



```

1          ; SERIAL - serial port interrupt routine
2          ;
3          ; *SERIAL* processes all serial port interrupts.
4          ;
5          ;
6          ;
7          ENTRY    none
8          ;
9          EXIT     none
10         ;
11        USES     none
12
13 12CB    C0    D0          SERIAL:    PUSH    PSW          ; Save registers
14 12CD    C0    E0          PUSH    ACC
15 12CF    C0    F0          PUSH    B
16 12D1    C0    82          PUSH    DPL
17 12D3    C0    83          PUSH    DPH
18 12D5    20    98    05    JB     RI, SERIN      ; Check receiver interrupt
19 12D8    20    99    38    JB     TI, SEROUT     ; Check transmit interrupt
20 12DB    80    3A          SJMP   SRT          ; This should never happen
21
22 12DD    75    39    00    SERIN:    MOV     AUTOCNT, #0
23 12E0    D2    24          SETB   VSPF        ; Change auto off
24 12E2    C2    98          CLR    RI          ; Clear receiver int flas
25 12E4    E5    99          MOV    A, SBUF     ; Get character
26 12E6    54    7F          ANL   A, #01111111B ; Mask off parity
27 12E8    20    43    02    JB     HAZ, SIO     ; IF not Hazeltine THEN
28 12EB    60    2A          JZ     SRT          ; IF null THEN exit
29
30 12ED    20    23    03    SIO:     JB     PRNTRF, S11   ; IF not printing THEN
31 12F0    30    2E    24    JNB    ONLINE, SRT ; IF offline THEN exit
32 12F3    B4    13    0B    S11:     CJNE   A, #XOFF, S12 ; IF -XOFF- received THEN
33 12F6    20    23    06    JB     PRNTRF, S11A ; IF not printing THEN
34 12F9    20    42    05    JB     ADM3, S12    ; IF ADM 3A THEN skip
35 12FC    20    43    02    JB     HAZ, S12     ; IF Hazeltine THEN skip
36 12FF    D2    16          S11A:   SETB   XOFFRCVED   ; flas for XOFF
37 1301    B4    11    02    S12:     CJNE   A, #XON, S13 ; IF -XON- received THEN
38 1304    C2    16          CLR    XOFFRCVED  ; flas for no XOFF
39 1306    20    23    0E    S13:     JB     PRNTRF, SRT ; IF printing THEN exit
40 1309    12    01CA        LCALL  PCIF        ; Put character in input FIFO
41 130C    30    D5    08    JNB    FO, SRT     ; IF -XOFF- needed THEN
42 130F    71    C4          ACALL  DOXOFF      ; send an -XOFF- and exit
43 1311    80    04          SJMP   SRT
44
45 1313    C2    99          SEROUT:  CLR    TI          ; Clear transmit int flas
46 1315    D2    13          SETB   OKTRANS    ; Set ok to transmit
47
48 1317    D0    83          SRT:     POP    DPH        ; Restore registers
49 1319    D0    82          POP    DPL
50 131B    D0    F0          POP    B
51 131D    D0    E0          POP    ACC
52 131F    D0    D0          POP    PSW
53 1321    71    26          ACALL  SRT1       ; Double check
54 1323    D2    AF          SETB   EA
55 1325    22          RET
56 1326    C2    AF          SRT1:   CLR    EA
57 1328    32          RETI          ; And return

```


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

```

1          ; ;          IRQ - interrupt request
2          ;
3          ;
4          ;          *IRQ* is executed when the CRTC starts to display the 25th
5          ;          line. At this point the DMA line number must have been set
6          ;          to zero or some error has occurred. This routine is a safety
7          ;          check for any such errors and will take at most one frame to
8          ;          straighten out any such errors. It also takes care of any
9          ;          clock timings and blinkins.
10         ;
11         ;          ENTRY   none
12         ;
13         ;          EXIT    none
14         ;
15         ;          USES    none
16         ;
17
18 1329    C0    D0          IRQ:    PUSH    PSW          ; Save registers
19 132B    C0    E0          PUSH    ACC
20 132D    C0    F0          PUSH    B
21 132F    C0    82          PUSH    DPL
22 1331    C0    83          PUSH    DPH
23
24 1333    75    09    00          MOV    DMLNA, #0          ; Safety check
25
26 1336    31    84          ACALL   CURSOR          ; Display cursor
27
28 1338    90    2001        MOV    DPTR, #DSTAREG
29 133B    E0          MOVX   A, @DPTR          ; Clear any CRTC flags
30
31 133C    05    38          INC    BLNKCNT          ; Bump blink count
32 133E    E5    37          MOV    A, BLNKRATE
33 1340    60    10          JZ     IRQ2
34 1342    B4    FF    02          CJNE  A, #0FFH, IRQ1    ; IF not full count THEN
35 1345    80    0B          SJMP  IRQ2
36 1347    B5    38    08    IRQ1: CJNE  A, BLNKCNT, IRQ2    ; IF time to blink THEN
37 134A    75    38    00          MOV    BLNKCNT, #0     ; reset count
38 134D    B2    01          CPL   BLINKF           ; toggle blink flag
39 134F    12    1E18        CALL  WLATCH
40
41 1352    05    3A          IRQ2:  INC    TCNT          ; Bump time count
42 1354    E5    3A          MOV    A, TCNT
43 1356    20    39    05          JB     FREQ, IRQ3      ; IF 50 Hz AND 1 second OR
44 1359    B4    32    34          CJNE  A, #50, IRQ5
45 135C    80    03          SJMP  IRQ4
46 135E    B4    3C    2F    IRQ3:  CJNE  A, #60, IRQ5      ; IF 60 Hz AND 1 second THEN
47 1361    75    3A    00    IRQ4:  MOV    TCNT, #0        ; initialize count
48 1364    05    3B          INC    TSEC           ; bump seconds
49 1366    E5    3B          MOV    A, TSEC
50 1368    B4    3C    25          CJNE  A, #60, IRQ5      ; IF one minute THEN
51 136B    30    3B    0B          JNB   SCRNSAVE, IRQ4A  ; IF screen saver THEN
52 136E    05    39          INC    AUTOCNT        ; bump auto off
53 1370    E5    39          MOV    A, AUTOCNT
54 1372    B4    0F    04          CJNE  A, #15, IRQ4A    ; IF saver time up THEN
55 1375    C2    24          CLR   VSPF           ; flag it is off
56 1377    D2    92          SETB  VSP            ; and turn off display
57

```


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

```

1          ;:          TRANSMIT - transmit to host
2          ;
3          ;
4          ;          *TRANSMIT* waits for an empty output holdins register, sets
5          ;          parity, and outputs the character to the serial port.
6          ;
7          ;          ENTRY   (A) = character to output
8          ;
9          ;          EXIT    (A) = character with correct parity
10         ;
11         ;          USES   A
12
13
14 139B    20    16    FD    TRANSMIT:  JB    XOFFRCVD, TRANSMIT    ; Wait until no -XOFF-
15
16 139E    C3          TRANS:      CLR    C                    ; Start with no parity
17 139F    30    28    0A          JNB   PRTYENA, TRA          ; IF parity enabled THEN
18 13A2    D3          SETB   C                    ; do set parity
19 13A3    20    29    06          JB    PRTYSTK, TRA          ; IF not stick parity THEN
20 13A6    A2    D0          MOV    C, P                    ; do even parity
21 13A8    20    2A    01          JB    PRTYEVN, TRA         ; IF not even parity THEN
22 13AB    B3          CPL     C                    ; do odd parity
23 13AC    92    E7          TRA:      MOV    ACC.7, C          ; Put in parity bit
24
25 13AE    20    13    03          TRW:      JB    OKTRANS, TROK          ; IF ok to transmit OR
26 13B1    30    99    FA          JNB   TI, TRW          ; IF transmit buffer empty THEN
27 13B4    20    02    03          TROK:     JB    PORTF, TROK1         ; IF auxiliary port OR
28 13B7    30    3E    03          JNB   HNDSHK, TROK2        ; IF hardware handshake THEN
29 13BA    20    B5    F1          TROK1:    JB    CTS, TRW          ; IF not CTS THEN do again
30
31 13BD    C2    99          TROK2:    CLR    TI                    ; clear buffer empty flag
32 13BF    C2    13          CLR    OKTRANS          ; make not ok to transmit
33 13C1    F5    99          MOV    SBUF, A          ; output data
34 13C3    22          RET

```

```

1          ;:          DOXOFF - do send XOFF to host
2          ;
3          ;          *DOXOFF* preempts outgoing data and outputs an XOFF to host
4          ;          setting the proper flags.
5          ;
6          ;
7          ;          ENTRY none
8          ;
9          ;          EXIT none
10         ;
11         ;          USES A
12         ;
13
14 13C4    20    10    1A    DOXOFF:    JB    XOFFSENT, SXR          ; IF XOFF already sent THEN exit
15 13C7    D2    10                SETB  XOFFSENT          ; Flag that handshake was done
16
17 13C9    30    3E    05                JNB   HNDSHK, XMTXOFF      ; IF hardware handshake THEN
18 13CC    D2    95                SETB  RTS              ; set handshake lines
19 13CE    D2    94                SETB  DTR
20 13D0    22                RET
21
22 13D1    74    13                XMTXOFF:    MOV   A, #XOFF          ; ELSE output an XOFF
23 13D3    61    9E                AJMP  TRANS           ; transmit it and return
24
25
26
27
28         ;:          DOXON - do send XON to host
29         ;
30         ;          *DOXON* preempts outgoing data and outputs an XON to host
31         ;          checking that it does not override any user XOFF.
32         ;
33         ;
34         ;          ENTRY none
35         ;
36         ;          EXIT none
37         ;
38         ;          USES A
39
40
41 13D5    30    10    09    DOXON:    JNB   XOFFSENT, SXR          ; IF no XOFF sent THEN exit
42 13D8    C2    10                CLR   XOFFSENT         ; Clear flag on handshake
43
44 13DA    30    3E    05                JNB   HNDSHK, SXON1     ; IF hardware handshake THEN
45 13DD    C2    95                CLR   RTS              ; clear handshake lines
46 13DF    C2    94                CLR   DTR
47 13E1    22                SXR:    RET
48
49 13E2    20    11    FC    SXON1:    JB    UXOFFSENT, SXR          ; IF no user XOFF THEN
50 13E5    74    11                XMTXON:    MOV   A, #XON          ; send an XON
51 13E7    61    9E                AJMP  TRANS           ; transmit it and return

```

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

```

1          ;;          LOOKUP - table lookup and jump
2          ;
3          ;          *LOOKUP* looks for the entry in the table passed to it. If
4          ;          found the return address is removed and control given to the
5          ;          routine. If not found then the main table is searched. If
6          ;          found the return address is removed and control given to the
7          ;          routine. If not found then control is given back to the
8          ;          callins routine.
9          ;
10         ;
11         ;          ENTRY   (B) = table length
12         ;          (DPTR) = beginning of table
13         ;
14         ;          EXIT   return if not found in tables
15         ;
16         ;          USES   all
17         ;
18
19 13E9    91    04          LOOKUP:    ACALL  XSTABX          ; Search users table
20 13EB    50    0B          JNC    LKUP1          ; IF not found THEN
21 13ED    75    F0    13    MOV    B, #LOTL
22 13F0    90    1432    MOV    DPTR, #LOT
23 13F3    91    04          ACALL  XSTABX          ; search main table
24 13F5    50    01          JNC    LKUP1          ; IF not found THEN
25 13F7    22          RET          ; return
26
27 13F8    D0    E0          LKUP1:    POP    ACC          ; ELSE found THEN
28 13FA    D0    E0          POP    ACC          ; remove return address
29 13FC    E4          CLR    A
30 13FD    73          JMP    @A+DPTR       ; jump to routine

```

```
1 ;: XSTX - transmit string to line 25 following CALL
2 ;
3 ; *XSTX* is a jump to the routine *XST*... It is used to save
4 ; bytes so that routines may use a short jump to here instead
5 ; of a long jump to the other routine.
6 ;
7 ;
8 13FE 02 1B9D XSTX: LJMPL XST ; Jump to routine
9 ;
10 ;
11 ;
12 ;
13 ;: XSTATX - transmit string to line 25
14 ;
15 ; *XSTATX* is a jump to the routine *XSTAT*... It is used to
16 ; save bytes so that routines may use a short jump to here
17 ; instead of a long jump to the other routine.
18 ;
19 ;
20 1401 02 1B88 XSTATX: LJMPL XSTAT ; Jump to routine
21 ;
22 ;
23 ;
24 ;
25 ;: XSTABX - search table
26 ;
27 ; *XSTABX* is a jump to the routine *XSTAB*... It is used to
28 ; save bytes so that routines may use a short jump to here
29 ; instead of a long jump to the other routine.
30 ;
31 ;
32 1404 02 013A XSTABX: LJMPL XSTAB ; Jump to routine
33 ;
```

```

1          ; ; SFCKB - setup fetch character from keyboard
2          ;
3          ; *SFCKB* waits for characters from the keyboard. It does
4          ; not return until a character is returned. All access codes
5          ; and the characters that follow are dropped.
6          ;
7          ;
8          ; ENTRY none
9          ;
10         ; EXIT (A) = character from keyboard
11        ;
12        ; USES A, B, DPTR
13
14
15 1407 12 02B6 SFCKB: CALL FCKB ; Wait for key
16 140A 40 FB JC SFCKB
17 140C B4 9F 07 CJNE A, #KB_ACC, SFCKB2 ; IF access code THEN
18 140F 12 02B6 SFCKB1: CALL FCKB ; drop next character
19 1412 40 FB JC SFCKB1
20 1414 80 F1 SJMP SFCKB ; and keep on going
21 1416 22 SFCKB2: RET ; ELSE return

```

```

1          ;;;          SETUP1 - setup mode line #1
2          ;
3          ;          *SETUP1* runs the terminal setup mode program.
4          ;
5          ;
6          ;          ENTRY   none
7          ;
8          ;          EXIT    none
9          ;
10         ;          USES   all
11         ;
12         ;
13 1417    20    93    01    SETUP1:   JB      SETLCK, SETUP0      ; Check for locked setup
14 141A    22
15
16 141B    12    1B2C    SETUP0:   CALL    SCLRRLR      ; Setup clear line reverse
17
18 141E    91    6E          SETUP1:   ACALL   L1          ; Display first line
19 1420    91    07          STR1A:   ACALL   SFCKB      ; Fetch character from keyboard
20 1422    75    F0    01          MOV     B, #L1TL
21 1425    90    146B          MOV     DPTR, #L1T
22 1428    71    E9          ACALL   LOOKUP      ; Lookup command
23 142A    80    F4          SJMP    STP1A      ; Keep on going
24
25
26         ;          Toggle on/off line
27         ;
28 142C    B2    2E          STP1B:   CPL     ONLINE      ; Toggle on/off line
29 142E    31    9E          CALL    INITKB      ; Reconfigur keyboard
30 1430    80    EC          SJMP    SETUP1
31
32
33
34
35         ;;;          LOT - every line table
36         ;
37         ;          *LOT* contains the jump table for every line.
38         ;
39         ;          1432    LOT          EQU     $
40
41 1432    0D          DB     CR
42 1433    141E          DW     SETUP1      ; Main setup line
43
44 1435    61          DB     'a'
45 1436    141E          DW     SETUP1      ; Main setup line
46
47 1438    41          DB     'A'
48 1439    141E          DW     SETUP1      ; Main setup line
49
50 143B    74          DB     't'
51 143C    14D7          DW     SETUP2      ; Set tabs
52
53 143E    54          DB     'T'
54 143F    14D7          DW     SETUP2      ; Set tabs
55
56 1441    62          DB     'b'
57 1442    1542          DW     SETUP3      ; Protocol #1

```



```

1
2 1444 42 DB 'B'
3 1445 1542 DW SETUP3 ; Protocol #1
4
5 1447 63 DB 'c'
6 1448 16B0 DW SETUP4 ; Protocol #2
7
8 144A 43 DB 'C'
9 144B 16B0 DW SETUP4 ; Protocol #2
10
11 144D 64 DB 'd'
12 144E 1795 DW SETUP5 ; Misc #1
13
14 1450 44 DB 'D'
15 1451 1795 DW SETUP5 ; Misc #1
16
17 1453 65 DB 'e'
18 1454 1865 DW SETUP6 ; Keyboard
19
20 1456 45 DB 'E'
21 1457 1865 DW SETUP6 ; Keyboard
22
23 1459 66 DB 'f'
24 145A 18F4 DW SETUP7 ; Misc #2
25
26 145C 46 DB 'F'
27 145D 18F4 DW SETUP7 ; Misc #2
28
29 145F 67 DB 'g'
30 1460 19DD DW SETUP8 ; Misc #3
31
32 1462 47 DB 'G'
33 1463 19DD DW SETUP8 ; Misc #3
34
35 1465 88 DB KB_SETUP ; Setup key
36 1466 1AF6 DW SETUPRET ; Return
37
38 1468 C8 DB KB_SETUP + KB_SHFT ; Shift setup key
39 1469 1B02 DW SETUPSAVE ; Save and return
40
41 0013 LOTL EQU ($-LOT)/3 ; Table #0 length
42
43
44
45
46 ;; LIT - line #1 table
47 ;
48 ; *LIT* contains the jump table for line #1 in setup mode.
49
50 146B LIT EQU $
51
52 146B 31 DB '1' ; Option #1
53 146C 142C DW STP1B ; Toggle on/off line
54
55 0001 LITL EQU ($-LIT)/3 ; Table length
56
57

```

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

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15 146E E4 L1: CLR A ; Position
16
17 146F 71 FE ACALL XSTX
18 1471 2A 2A 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
19
20 1495 90 14C7 MOV DPTR, #LN1B1
21 1498 20 2E 03 JB ONLINE, L1A ; Get proper message for
22 149B 90 14CF MOV DPTR, #LN1B2 ; on/off line
23 149E 91 01 L1A: ACALL XSTATX ; Next part
24
25 14A0 71 FE ACALL XSTX
26 14A2 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
27
28 14C6 22 RET
29
30 ; Line #1 setup mode messages
31 ;
32 14C7 6F 6E 20 LN1B1: DB 'on line', (''+80H)
14CA 6C 69 6E
14CD 65 A0
33 14CF 6F 66 66 LN1B2: DB 'off lin', ('e'+80H)

```

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

14D2	20	6C	69
14D5	6E	E5	

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

```

1          ;:          SETUP2 - setup mode line #2.
2          ;
3          ;          *SETUP2* is the setup mode line for set/clear tabs.
4          ;          -PTR2- is used as the cursor position.
5          ;
6          ;
7          ;          ENTRY   none
8          ;
9          ;          EXIT    none
10         ;
11         ;          USES   all
12         ;
13
14 14D7    79    00          SETUP2:  MOV    PTR2, #0          ; -PTR2- is cursor position
15 14D9    B1    0D          STP2A:  ACALL  L2          ; Display line #2
16 14DB    91    07          STP2B:  CALL   SFCKB         ; Fetch character from keyboard
17 14DD    75    F0    04          MOV    B, #L2TL
18 14E0    90    1501        MOV    DPTR, #L2T
19 14E3    71    E9          ACALL  LOOKUP         ; Look up command
20 14E5    80    F4          SJMP   STP2B         ; Keep on going
21
22
23         ;          Move left
24         ;
25 14E7    19          STP2C:  DEC    PTR2          ; Decrement
26 14E8    B9    FF    01          CJNE   PTR2, #(0-1), STP2C1 ; IF wrap around THEN
27 14EB    09          INC    PTR2          ; bump back up
28 14EC    80    EB          STP2C1: SJMP   STP2A         ; keep going
29
30
31         ;          Move right
32         ;
33 14EE    09          STP2D:  INC    PTR2          ; Bump
34 14EF    B9    50    01          CJNE   PTR2, #80, STP2D1 ; IF wrap around THEN
35 14F2    19          DEC    PTR2          ; bump back down
36 14F3    80    E4          STP2D1: SJMP   STP2A         ; keep going
37
38
39         ;          Set tab
40         ;
41 14F5    D2    D5          STP2E:  SETB   F0
42 14F7    80    02          SJMP   STP2F1         ; Set tab
43
44
45         ;          Clear tab
46         ;
47 14F9    C2    D5          STP2F:  CLR    F0
48 14FB    E9          STP2F1: MOV    A, PTR2         ; Get position
49 14FC    12    05A2        CALL   SETTAB         ; Set/clear tab
50 14FF    80    D8          SJMP   STP2A         ; Keep going
51
52
53
54
55         ;:          L2T - line #2 table
56         ;
57         ;          *L2T* contains the jump table for line #2 in setup mode.

```

```

1
2      1501          L2T          EQU      $
3
4 1501      82          DB      KBLEFT      ; Cursor left
5 1502      14E7       DW      STP2C       ; Move left
6
7 1504      83          DB      KB_RIGHT   ; Cursor right
8 1505      14EE       DW      STP2D       ; Move right
9
10 1507      80          DB      KB_UP     ; Cursor up
11 1508      14F5       DW      STP2E       ; Set tab
12
13 150A      81          DB      KB_DOWN   ; Cursor down
14 150B      14F9       DW      STP2F       ; Clear tab
15
16          0004          L2TL          EQU      ($-L2T)/3      ; Table length
17
18
19
20
21          ; ;          L2 - line #2
22          ; ;
23          ; ;          *L2* displays setup information for line #2 set/clear tabs.
24          ; ;
25          ; ;
26          ; ;          ENTRY      none
27          ; ;
28          ; ;          EXIT      none
29          ; ;
30          ; ;          USES      all
31          ; ;
32
33 150D      90          5780          L2:      MOV      DPTR, #L25MEM      ; Work with 25th line
34 1510      7D          00          MOV      TEMP, #0          ; Counter variable
35 1512      78          29          MOV      PTR1, #TABTAB      ; Point to tab table
36 1514      7A          0A          MOV      INDX1, #10        ; 10 bytes
37 1516      E6          L2A:      MOV      A, @PTR1          ; REPEAT set byte
38 1517      7B          08          MOV      INDX2, #8        ; 8 bits
39 1519      FE          MOV      WORK1, A          ; save value for later
40 151A      ED          L2B:      MOV      A, TEMP          ; REPEAT
41 151B      04          INC      A          ; set and adjust position
42 151C      75          F0          0A          MOV      B, #10          ; calculate character
43 151F      84          DIV      AB
44 1520      E5          F0          MOV      A, B
45 1522      24          30          ADD      A, #'0'          ; convert to ASCII
46 1524      FF          MOV      WORK2, A          ; and save it
47
48 1525      ED          MOV      A, TEMP
49 1526      B5          01          02          CJNE     A, PTR2A, L2C    ; IF cursor = position THEN
50 1529      7F          09          MOV      WORK2, #09H     ; change to cursor char
51
52 152B      EE          L2C:      MOV      A, WORK1          ; restore tab table value
53 152C      33          RLC      A
54 152D      FE          MOV      WORK1, A          ; replace new value
55 152E      50          08          JNC     L2D              ; IF tab is set here THEN
56 1530      7F          00          MOV      WORK2, #00H     ; change to tab char
57 1532      ED          MOV      A, TEMP

```


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

```

1          ; ;          SETUP3 - setup mode line #3
2          ;
3          ;          *SETUP3* runs the terminal setup mode to change baud rate,
4          ;          parity, duplex, and handshake.
5          ;
6          ;
7          ;          ENTRY none
8          ;
9          ;          EXIT none
10         ;
11         ;          USES all
12         ;
13         ;
14 1542    B1    98          SETUP3:    ACALL    L3          ; Display third line
15 1544    91    07          STP3A:    ACALL    SFCKB        ; Fetch character from keyboard
16 1546    75    F0    04          MOV     B, #L3TL
17 1549    90    158C        MOV     DPTR, #L3T
18 154C    71    E9          ACALL    LOOKUP        ; Look up command
19 154E    80    F4          SJMP     STP3A        ; Keep on going
20
21         ;
22         ;          Change baud rate
23         ;
24 1550    E5    33          STP3B:    MOV     A, BAUDRATE    ; Search for current baud rate
25 1552    75    F0    0E          MOV     B, #L3ATL
26 1555    90    161E        MOV     DPTR, #L3AT
27 1558    91    04          ACALL    XSTABX
28 155A    40    06          JC      STP3B1        ; IF found THEN
29 155C    E5    F0          MOV     A, B          ; set count + 1
30 155E    B4    0E    01          CJNE   A, #L3ATL, STP3B1 ; IF time for wrap THEN
31 1561    F4          CLR     A          ; zero
32 1562    12    0E4B        STP3B1:  CALL    SBR          ; Set baud rate
33 1565    80    DB          SJMP     SETUP3
34
35         ;
36         ;          Change parity
37         ;
38 1567    20    28    08          STP3C:  JB      PRTYENA, STP3C1 ; IF disabled THEN
39 156A    D2    28          SETB   PRTYENA        ; enable parity
40 156C    C2    29          CLR    PRTYSTK        ; normal parity
41 156E    C2    2A          CLR    PRTYEVN        ; odd parity
42 1570    80    D0          SJMP   SETUP3
43 1572    20    29    07          STP3C1: JB     PRTYSTK, STP3C2 ; IF normal parity AND
44 1575    20    2A    08          JB     PRTYEVN, STP3C3 ; IF odd parity THEN
45 1578    D2    2A          SETB   PRTYEVN        ; make even parity
46 157A    80    C6          SJMP   SETUP3
47 157C    C2    28          STP3C2: CLR   PRTYENA        ; IF stick THEN disable parity
48 157E    80    C2          SJMP   SETUP3
49 1580    D2    29          STP3C3: SETB   PRTYSTK        ; IF even THEN stick parity
50 1582    80    BE          SJMP   SETUP3
51
52         ;
53         ;          Toggle duplex
54         ;
55 1584    B2    2F          STP3D:  CPL     FULLDPLX      ; Toggle full/half duplex
56 1586    80    BA          SJMP   SETUP3
57

```

```

1
2
3 ; Toggle handshake
4 1588 B2 3E STP3E: CPL HNSHK ; Toggle soft/hard handshake
5 158A 80 B6 S JMP SETUP3
6
7
8
9
10 ; ; L3T - line #3 table
11 ; ;
12 ; ; *L3T* contains the jump table for line #3 in setup mode.
13
14 158C L3T EQU $
15
16 158C 31 DB '1' ; Option #1
17 158D 1550 DW STP3B ; Change baud rate
18
19 158F 32 DB '2' ; Option #2
20 1590 1567 DW STP3C ; Change parity
21
22 1592 33 DB '3' ; Option #3
23 1593 1584 DW STP3D ; Toggle duplex
24
25 1595 34 DB '4' ; Option #4
26 1596 1588 DW STP3E ; Toggle handshake
27
28 0004 L3TL EQU (L3T)/3 ; Table length
29
30
31
32
33 ; ; L3 - line #3
34 ; ;
35 ; ; *L3* displays setup information for line #3.
36 ; ;
37 ; ;
38 ; ; ENTRY none
39 ; ;
40 ; ; EXIT none
41 ; ;
42 ; ; USES all
43
44
45 1598 E4 L3: CLR A ; Position
46
47 1599 71 FE ACALL XSTX
48 159B 2A 4D 45 DB '*MENU B* 1. BAUD', (7'+80H)
49 159E 4E 55 20
50 15A1 42 2A 20
51 15A4 20 20 31
52 15A7 2E 20 42
53 15AA 41 55 44
54 15AD A0
55
56
57
58
59
60 15AE E5 33 MOV A, BAUDRATE
61 15B0 75 F0 0E MOV B, #L3ATL

```


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

```

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)
15C8 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 91 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', (''+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)
.....

```

..... Z29ROM; MICROBENCH 8051 CROSS ASSEMBLER (V1)-218 24-FEB-83 15:30:06 PAGE 191+

..... *** Z-2? COMPUTER TERMINAL FIRMWARE VERSION 1.03 ***

..... 165A 30 A0


```

1          ; ;          SETUP4 - setup mode line #4
2          ;
3          ;          *SETUP4* runs the terminal setup mode to change port, mode
4          ;          hold screen mode, and monitor mode.
5          ;
6          ;
7          ;          ENTRY   none
8          ;
9          ;          EXIT    none
10         ;
11         ;          USES   all
12         ;
13         ;
14 16B0    D1    F8          SETUP4:    ACALL   L4          ; Display fourth line
15 16B2    91    07          STP4A:    CALL    SFCKB        ; Fetch character from keyboard
16 16B4    75    F0    04          MOV     B, #L4TL
17 16B7    90    16EC        MOV     DPTR, #L4T
18 16BA    71    E9          CALL    LOOKUP        ; Look up command
19 16BC    80    F4          SJMP   STP4A        ; Keep on going
20
21         ;
22         ;          Tossle main port
23         ;
24 16BE    12    1E0F        STP4B:    CALL    CMPORT        ; Tossle port
25 16C1    80    ED          SJMP   SETUP4
26
27         ;
28         ;          Tossle ANSI/ZDS/ADM 3/Hazeltine mode
29         ;
30 16C3    E5    28          STP4C:    MOV     A, MODE        ; Get current mode
31 16C5    FD          MOV     TEMP, A        ; Save it
32 16C6    53    05    F0          ANL    TEMP, #11110000B ; Mask off mode portion
33 16C9    23          RL     A                ; Rotate to next mode
34 16CA    54    0E          ANL    A, #00001110B   ; Mask off extra stuff
35 16CC    70    01          JNZ    STP4C1         ; IF zero THEN
36 16CE    04          INC     A                ; bump to one
37 16CF    4D          STP4C1:   ORL    A, TEMP        ; Combine both halves
38 16D0    F5    28          MOV     MODE, A        ; Save new current mode
39 16D2    31    CB          ACALL  TERMINIT       ; Initialize for any terminal
40 16D4    30    43    02          JNB    HAZ, STP4C2    ; IF Hazeltine THEN
41 16D7    31    D4          ACALL  HAZINIT        ; initialize for Hazeltine
42 16D9    80    D5          STP4C2:  SJMP   SETUP4
43
44         ;
45         ;          Tossle hold screen mode
46         ;
47 16DB    30    36    05          STP4D:    JNB    HSMODEF, STP4D1 ; IF hold screen THEN exit
48 16DE    12    0B43          CALL   XHSM
49 16E1    80    CD          SJMP   SETUP4
50 16E3    12    0B3C          STP4D1:  CALL   EHSM          ; ELSE enter hold screen
51 16E6    80    C8          SJMP   SETUP4
52
53         ;
54         ;          Tossle monitor mode
55         ;
56 16E8    B2    3A          STP4E:    CPL    MONITOR        ; Tossle monitor mode
57 16EA    80    C4          SJMP   SETUP4

```

```

1
2
3
4
5
6          ;:          L4T - Line #4 table
7          ;
8          ;          *L4T* contains the jump table for line #4 in setup mode
9          16EC          L4T          EQU          $
10
11 16EC          31          DB          /1/
12 16ED          16BE          DW          STP4B          ; Toggle ports
13
14 16EF          32          DB          /2/
15 16F0          16C3          DW          STP4C          ; Toggle mode
16
17 16F2          33          DB          /3/
18 16F3          16DB          DW          STP4D          ; Toggle hold screen
19
20 16F5          34          DB          /4/
21 16F6          16E8          DW          STP4E          ; Toggle monitor mode
22
23          0004          L4TL          EQU          ($-L4T)/3          ; Table length
24
25
26
27
28          ;:          L4 - line #4
29          ;
30          ;          *L4* displays setup information for line #4.
31          ;
32          ;
33          ;          ENTRY none
34          ;
35          ;          EXIT none
36          ;
37          ;          USES all
38          ;
39
40 16F8          E4          L4:          CLR          A
41 16F9          71          FE          ACALL         XSTX
42 16FB          2A          4D          45          DB          /*MENU C* 1. PORT', (/'+80H)
43          16FE          4E          55          20          1701          43          2A          20
44          1704          20          20          31
45          1707          2E          20          50
46          170A          4F          52          54
47          170D          A0
48
49 170E          90          1709          MOV          DPTR, #LN4B1          ; Normal
50 1711          30          02          03          JNB          PORTE, L4A
51 1714          90          170F          MOV          DPTR, #LN4B2          ; Auxiliary
52 1717          91          01          L4A:          ACALL         XSTATX          ; Port message
53
54 1719          71          FE          ACALL         XSTX
55 171B          20          20          20          DB          /* 2. MODE', (/'+80H)
56 171E          32          2E          20

```

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

.....1721 4D 4F 44
.....1724 45 A0

1

```

2 1726 90 1775 MOV DPTR, #LN4D1 ; ANSI mode
3 1729 20 40 0F JB ANSI, L4B
4 172C 90 177D MOV DPTR, #LN4D2 ; ZDS mode
5 172F 20 41 09 JB ZDS, L4B
6 1732 90 1785 MOV DPTR, #LN4D3 ; ADM 3 mode
7 1735 20 42 03 JB ADM3, L4B
8 1738 90 178D MOV DPTR, #LN4D4 ; Hazeltine mode
9 173B 91 01 L4B: ACALL XSTATX ; ANSI/ZDS/ADM 3/Haz message
10
11 173D 71 FE ACALL XSTX
12 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

```

13

```

14 174F A2 36 MOV C, HSMODEF
15 1751 12 1BBF LCALL XLONOFF ; Hold on/off message
16
17 1754 71 FE ACALL XSTX
18 1756 20 20 20 DB ' 4. MONITOR', (''+80H)
1759 34 2E 20
175C 4D 4F 4E
175F 49 54 4F
1762 52 A0

```

19

```

20 1764 A2 3A MOV C, MONITOR
21 1766 02 1BBF LJMPL XLONOFF ; Monitor mode on/off and return
22
23
24
25

```

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

; ; Line #4 setup mode messages

;

```

29 1769 6E 6F 72 LN4B1: DB 'norma', (''+80H)
176C 6D 61 EC
30 176F 61 75 78 LN4B2: DB 'auxil', (''+80H)
1772 69 6C A0
31 1775 61 6E 73 LN4D1: DB 'ansi', (''+80H)
1778 69 20 20
177B 20 A0
32 177D 7A 65 6E LN4D2: DB 'zenith', (''+80H)
1780 69 74 68
1783 20 A0
33 1785 61 64 6D LN4D3: DB 'adm 3a', (''+80H)
1788 20 33 61
178B 20 A0
34 178D 68 61 7A LN4D4: DB 'haz 150', (''+80H)
1790 20 31 35
1793 30 B0

```


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

```

1          ;:          SETUP5 - setup mode line #5
2          ;
3          ;          *SETUP5* runs the terminal setup mode to change set clock,
4          ;          toggle status line, setup wrap around, and screen saver.
5          ;
6          ;
7          ;          ENTRY   none
8          ;
9          ;          EXIT    none
10         ;
11         ;          USES   all
12
13
14 1795     12     1808     SETUP5:   CALL    L5          ; Display fifth line
15 1798     91     07          STP5A:   CALL    SFCKB       ; Fetch character from keyboard
16 179A     75     F0     04          MOV    B, #L5TL
17 179D     90     17FC        MOV    DPTR, #L5T
18 17A0     71     E9          CALL   LOOKUP      ; Look up command
19 17A2     80     F4          SJMP   STP5A       ; Keep on going
20
21
22         ;          ;          Set clock
23         ;
24 17A4     12     1B2C     STP5B:   CALL    SCLRLR      ; Setup clear line reverse
25 17A7     E4          CLR    A
26
27 17A8     71     FE          ACALL  XSTX
28 17AA     45     6E     74          DB     'Enter hours ', (>+80H)
    17AD     65     72     20
    17B0     68     6F     75
    17B3     72     73     20
    17B6     BE
29
30 17B7     74     0E          MOV    A, #14
31 17B9     12     1B45        CALL   GETNUM      ; Get number
32 17BC     BF     00     02        CJNE  WORK2, #0, STP5B1 ; IF zero THEN skip
33 17BF     80     D4          SJMP   SETUP5
34 17C1     B4     18     00        STP5B1: CJNE  A, #24, STP5B2
35 17C4     50     CF     STP5B2: JNC    SETUP5      ; IF number <= 24 THEN
36 17C6     F5     3D          MOV    THOUR, A
37 17C8     12     1B2C        CALL   SCLRLR      ; Setup clear line reverse
38 17CB     E4          CLR    A
39
40 17CC     71     FE          ACALL  XSTX
41 17CE     45     6E     74          DB     'Enter minutes ', (>+80H)
    17D1     65     72     20
    17D4     6D     69     6E
    17D7     75     74     65
    17DA     73     20     BE
42
43 17DD     74     10          MOV    A, #16
44 17DF     12     1B45        CALL   GETNUM      ; set number
45 17E2     B4     3C     00        CJNE  A, #60, STP5B3
46 17E5     50     AE     STP5B3: JNC    SETUP5      ; IF number <= 59 THEN
47 17E7     F5     3C          MOV    TMIN, A
48
49 17E9     E4          CLR    A

```

*** Z-2? COMPUTER TERMINAL FIRMWARE VERSION 1.03 ***

```

1 17EA F5 3B MOV TSEC, A ; clear seconds
2 17EC F5 3A MOV TCNT, A ; and count
3 17EE 80 A5 SJMP SETUP5
4
5
6 ; Tossle 25th status line
7 ;
8 17F0 B2 37 STP5C: CPL L25ON ; Tossle status line
9 17F2 80 A1 SJMP SETUP5
10
11
12 ; Tossle wrap
13 ;
14 17F4 B2 2B STP5D: CPL AUTOWRAP ; Tossle auto wrap
15 17F6 80 9D SJMP SETUP5
16
17
18 ; Tossle screen saver mode
19 ;
20 17F8 B2 3B STP5E: CPL SCRNSAVE ; Tossle screen saver
21 17FA 80 99 SJMP SETUP5
22
23
24
25
26 ;; L5T - line #5 table
27 ;
28 ; *L5T* contains the JUMP table for line #5 in setup mode.
29 ;
30 17FC L5T EQU $
31
32 17FC 31 DB '1'
33 17FD 17A4 DW STP5B ; Set clock
34
35 17FF 32 DB '2'
36 1800 17F0 DW STP5C ; Tossle status line
37
38 1802 33 DB '3'
39 1803 17F4 DW STP5D ; Tossle wrap around
40
41 1805 34 DB '4'
42 1806 17F8 DW STP5E ; Tossle screen saver
43
44 0004 L5TL EQU ($-L5T)/3 ; Table length
45
46
47
48
49 ;; L5 - line #5
50 ;
51 ; *L5* displays setup information for line #5.
52 ;
53 ;
54 ; ENTRY none
55 ;
56 ; EXIT none
57 ;

```

```

1          ;          USES    all
2
3
4 1808     E4          L5:     CLR    A
5
6 1809     71         9D          ACALL  XST
7 180B     2A         4D         45          DB    '*MENU D*  1. SET CLOCK  2. STATUS LINE', ( '+'80H)
8 180E     4E         55         20
9 1811     44         2A         20
10 1814     20         20         31
11 1817     2E         20         53
12 181A     45         54         20
13 181D     43         4C         4F
14 1820     43         4B         20
15 1823     20         20         32
16 1826     2E         20         53
17 1829     54         41         54
18 182C     55         53         20
19 182F     4C         49         4E
20 1832     45         A0
21
22 1834     A2         37          MOV    C, L250N
23 1836     12         1BBF       LCALL  XLONOFF          ; Status line on/off
24
25 1839     71         9D          ACALL  XST
26 183B     20         20         20          DB    '* 3. WRAP', ( '+'80H)
27 183E     33         2E         20
28 1841     57         52         41
29 1844     50         A0
30
31 1846     A2         2B          MOV    C, AUTOWRAP
32 1848     12         1BBF       LCALL  XLONOFF          ; Wrap around on/off
33
34 184B     71         9D          ACALL  XST
35 184D     20         20         20          DB    '* 4. SCREEN SAVER', ( '+'80H)
36 1850     34         2E         20
37 1853     53         43         52
38 1856     45         45         4E
39 1859     20         53         41
40 185C     56         45         52
41 185F     A0
42
43 1860     A2         3B          MOV    C, SCRNSAVE
44 1862     02         1BBF       LJMP   XLONOFF          ; Screen saver on/off and return

```

```

1 ..... ;; ..... SETUP6: = setup mode line #6
2 ..... ;
3 ..... ; ..... *SETUP6* runs the terminal setup mode to change keypad shift,
4 ..... ; ..... keypad alternate, auto repeat, and key click.
5 ..... ;
6 ..... ;
7 ..... ; ..... ENTRY: none
8 ..... ;
9 ..... ; ..... EXIT: none
10 ..... ;
11 ..... ; ..... USES: all
12 ..... ;
13 ..... ;
14 1865 11 97 SETUP6: ACALL L6 ; Display sixth line
15 1867 12 1407 STP6A: CALL SFCKB ; Fetch character from keyboard
16 186A 75 F0 04 MOV B, #L6TL
17 186D 90 188B MOV DPTR, #L6T
18 1870 12 13E9 CALL LOOKUP ; Look up command
19 1873 80 F2 SJMP STP6A ; Keep on going
20 ..... ;
21 ..... ; ..... Toggle keypad shifted
22 ..... ;
23 ..... ;
24 1875 B2 30 STP6B: CPL KPASHTF ; Toggle keypad shifted
25 1877 80 EC SJMP SETUP6
26 ..... ;
27 ..... ; ..... Toggle keypad alternate
28 ..... ;
29 ..... ;
30 1879 B2 31 STP6C: CPL KPADALTF ; Toggle keypad alternate
31 187B 80 E8 SJMP SETUP6
32 ..... ;
33 ..... ; ..... Toggle auto repeat
34 ..... ;
35 ..... ;
36 187D B2 34 STP6D: CPL REPF ; Toggle auto repeat
37 187F 12 119E CALL INITKB ; Reconfigure keyboard
38 1882 80 E1 SJMP SETUP6
39 ..... ;
40 ..... ; ..... Toggle key click
41 ..... ;
42 ..... ;
43 1884 B2 33 STP6E: CPL CLKF ; Toggle key click
44 1886 12 119E CALL INITKB ; Reconfigure keyboard
45 1889 80 DA SJMP SETUP6
46 ..... ;
47 ..... ;
48 ..... ;
49 ..... ; ..... L6T - line #6 table
50 ..... ;
51 ..... ; ..... *L6T* contains the jump table for line #6 in setup mode.
52 ..... ;
53 ..... ;
54 ..... 188B L6T EQU $
55 ..... ;
56 188B 31 DB '1'
57 188C 1875 DW STP6B ; Toggle keypad shifted

```

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

```

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, KPADSHFT
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', (''+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

```

1          ;:          SETUP7 - setup mode line #7
2          ;
3          ;          *SETUP7* runs the terminal setup mode to change auto carriage
4          ;          return, auto line feed, cursor, and frequency.
5          ;
6          ;
7          ;          ENTRY   none
8          ;
9          ;          EXIT    none
10         ;
11         ;          USES   all
12         ;
13         ;
14 18F4    31    3F          SETUP7:    ACALL   L7          ; Display seventh line
15 18F6    12    1407      STP7A:    CALL    SFCKB      ; Fetch character from keyboard
16 18F9    75    F0      04          MOV    B, #L7TL
17 18FC    90    1933      MOV    DPTR, #L7T
18 18FF    12    13E9      CALL   LOOKUP     ; Look up command
19 1902    80    F2          SJMP   STP7A      ; Keep on going
20
21         ;
22         ;          Tossle auto carriage return
23         ;
24 1904    B2    2C          STP7B:    CPL     AUTOOCR  ; Tossle auto CR
25 1906    80    EC          SJMP   SETUP7
26
27         ;
28         ;          Tossle auto line feed
29         ;
30 1908    B2    2D          STP7C:    CPL     AUTOLF  ; Tossle auto LF
31 190A    80    E8          SJMP   SETUP7
32
33         ;
34         ;          Tossle cursor
35         ;
36 190C    20    38      0D          STP7D:    JB     CRNBK, STP7D2 ; IF blinking AND
37 190F    20    32      04          JB     CRUL, STP7D1  ; IF block THEN
38 1912    D2    32          SETB  CRUL          ; set to blink line
39 1914    80    11          SJMP  STP7D4
40
41 1916    C2    32          STP7D1:   CLR    CRUL          ; IF blink line THEN
42 1918    D2    38          SETB  CRNBK         ; set to block
43 191A    80    0B          SJMP  STP7D4
44
45 191C    20    32      04          STP7D2:   JB     CRUL, STP7D3 ; IF block THEN
46 191F    D2    32          SETB  CRUL          ; set to underline
47 1921    80    04          SJMP  STP7D4
48
49 1923    C2    32          STP7D3:   CLR    CRUL          ; IF underline THEN
50 1925    C2    38          CLR    CRNBK         ; set to blink block
51
52 1927    12    117B      STP7D4:   CALL   SETCF      ; Set cursor format
53 192A    80    C8          SJMP  SETUP7
54
55         ;
56         ;          Tossle line frequency
57         ;

```


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

```

1 192C B2 39 STP7E: CPL FREQ ; Toggle frequency
2 192E 12 117B CALL SETCF ; Set cursor format
3 1931 80 C1 SJMP SETUP7
4
5
6
7
8 ;; L7T = line #7 table
9 ;
10 ; *L7T* contains the Jump table for line #7 in setup mode.
11
12 1933 L7T EQU $
13
14 1933 31 DB '1'
15 1934 1904 DW STP7B ; Toggle auto carriage return
16
17 1936 32 DB '2'
18 1937 1908 DW STP7C ; Toggle auto line feed
19
20 1939 33 DB '3'
21 193A 190C DW STP7D ; Toggle cursor
22
23 193C 34 DB '4'
24 193D 192C DW STP7E ; Toggle line freq 50/60 Hz
25
26 0004 L7TL EQU ($-L7T)/3 ; Table length
27
28
29
30
31 ;; L7 - line #7
32 ;
33 ; *L7* displays setup information for line #7
34 ;
35 ;
36 ; ENTRY none
37 ;
38 ; EXIT none
39 ;
40 ; USES all
41 ;
42
43 193F E4 L7: CLR A ; Position
44
45 1940 71 9D ACALL XST
46 1942 2A 4D 45 DB '*MENU F* 1. AUTO CR', (0'+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
47
48 1958 A2 2C MOV C, AUTO CR
49 195A 71 BF ACALL XLONOFF ; Auto CR on/off
50

```

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

```

1 195C 71 9D ACALL XST
2 195E 20 20 20 DB ' 2. AUTO LF', (''+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', (''+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 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 ACALL XSTAT ; Cursor message
19
20 1998 71 9D ACALL XST
21 199A 20 20 20 DB ' 4. FREQ', (''+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 ACALL XSTAT
27
28 19B0 71 9D ACALL XST
29 19B2 30 20 48 DB ' 0 H', (''+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 'blk blk', ('k'+80H)
   19BA 6B 20 62
   19BD 6C 63 EB
40 19C0 62 6C LN7D2: DB 'blk lin', ('e'+80H)
   19C3 6B 20 6C
   19C6 69 6E E5
41 19C9 62 6C 6F LN7D3: DB 'block ', (''+80H)

```

Z29ROM: MICROBENCH 8051 CROSS ASSEMBLER (V1)-218 24-FEB-83 15:30:06 PAGE 204+

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

19CC	63	6B	20
19CF	20	20	A0

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

1	19D2	75	6E	64	LN7D4:	DB	<u>'</u> ('e'+80H)
	19D5	65	72	6C			
	19D8	69	6E	E5			
2	19DB	B5			LN7F1:	DB	('5'+80H)
3	19DC	B6			LN7F2:	DB	('6'+80H)

```

1          ; ;          SETUP8 - setup mode line #8
2          ;
3          ;          *SETUP8* runs the terminal setup mode for simple terminal
4          ;          diagnostics.
5          ;
6          ;
7          ;          ENTRY none
8          ;
9          ;          EXIT none
10         ;
11         ;          USES all
12         ;
13
14 19DD 51 8C          SETUP8:  ACALL  L8          ; Display eighth line
15 19DF 12 1407       STP8A:   CALL   SFCKB          ; Fetch character from keyboard
16 19E2 75 F0 04     MOV     B, #L8TL
17 19E5 90 1A74     MOV     DPTR, #L8T
18 19E8 12 13E9     CALL   LOOKUP          ; Look up command
19 19EB 80 F2       SJMP    STP8A          ; Keep on going
20
21
22         ;          Fill screen
23         ;
24 19ED 71 2C       STP8B:   ACALL  SCLRRLR        ; Setup clear line reverse
25 19EF E4         CLR     A          ; Position
26
27 19F0 71 9D       ACALL  XST
28 19F2 50 72 65    DB     'Press RETURN to exit', ('t'+80H)
29 19F5 73 73 20
30 19F8 52 45 54
31 19FB 55 52 4E
32 19FE 20 74 6F
33 1A01 20 65 78
34 1A04 69 F4
35
36 1A06 74 45       MOV     A, #'E'          ; Display 'E' first time
37 1A08 90 5000     STP8B1:  MOV     DPTR, #CHARMEM    ; Point to character memory
38 1A0B 71 21       ACALL  SFILL          ; Fill screen with char
39
40 1A0D 12 1407     CALL   SFCKB          ; Fetch character from keyboard
41 1A10 75 F0 04     MOV     B, #L8TL2
42 1A13 90 1A80     MOV     DPTR, #L8T2
43 1A16 12 013A     CALL   STAB          ; Search the table
44 1A19 40 ED       JC     STP8B1        ; IF not found THEN display it
45 1A1B E4         CLR     A
46 1A1C 73         JMP     @A+DPTR        ; Jump to routine
47
48 1A1D 74 20       STP8B3:  MOV     A, #' '          ; Change funny space to real
49 1A1F 80 E7       SJMP    STP8B1        ; Fill screen
50
51
52         ;          Change attribute
53         ;
54 1A21 E5 21       STP8C:   MOV     A, ATTRIBUTES    ; Get current attributes
55 1A23 04         INC     A          ; Bump to next attribute
56 1A24 54 0F       ANL     A, #00001111B  ; Mask just attributes
57 1A26 53 21 F0     ANL     ATTRIBUTES, #11110000B ; Get attributes ready

```

```

1 1A29 42 21 ORL ATTRIBUTES, A ; Move new attributes in
2 1A2B 90 4000 MOV DPTR, #ATTRMEM ; 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 ALCHARA, 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 SCLRRL ; setup clear line reverse
22 1A46 E4 CLR A ; position
23
24 1A47 71 9D ACALL XST
25 1A49 43 6F 6E DB 'Continuous test, (10+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 ;
48 ; *L8T* contains the jump table for line #8 in setup mode.
49 1A74 L8T EQU $
50
51 1A74 31 DB '1'
52 1A75 1A32 DW STP8D ; Toggle character set
53

```

```

1 1A77 32 DB '2'
2 1A78 19ED DW STP8B ; Fill screen
3
4 1A7A 33 DB '3'
5 1A7B 1A21 DW STP8C ; Change attributes
6
7 1A7D 34 DB '4'
8 1A7E 1A3A DW STP8E ; Continuous test
9 0004 L8TL EQU ($-L8T)/3 ; Table length
10
11
12
13
14 ; L8T2 - line #8 table 2
15 ;
16 ; *L8T2* contains the Jump table for line #8 in fill screen
17 ; option.
18
19 1A80 L8T2 EQU $
20
21 1A80 8C DB KB_SPACE
22 1A81 1A1D DW STP8B3 ; Access code (ignore)
23
24
25 1A83 0D DB CR
26 1A84 19DD DW SETUP8 ; Carriage return
27
28 1A86 88 DB KB_SETUP
29 1A87 1AF6 DW SETUPRET ; Exit setup mode
30
31 1A89 C8 DB KB_SETUP + KB_SHFT
32 1A8A 1B02 DW SETUPSAVE ; Exit setup mode and save
33
34 0004 L8TL2 EQU ($-L8T2)/3 ; Table length
35
36
37
38
39 ; L8 - line #8
40 ;
41 ; *L8* displays setup information for line #8.
42 ;
43 ;
44 ; ENTRY none
45 ;
46 ; EXIT none
47 ;
48 ; USES all
49
50
51 1A8C E4 L8: CLR A ; Position
52
53 1A8D 71 9D ACALL XST
54 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', (''+80H)
 1AF0 65 72 6E
 1AF3 61 74 E5

```

```
1          ;:          SETUPRET - return from setup mode
2          ;
3          ;          *SETUPRET* exits setup mode without permanently saving
4          ;          any values that have changed.
5          ;
6          ;
7          ;          ENTRY   none
8          ;
9          ;          EXIT    none
10         ;
11         ;          USES   none
12
13
14 1AF6    30    37    06    SETUPRET:  JNB    L250N, SRET      ; IF 25th line is on AND
15 1AF9    20    15    03          JB     L25EN, SRET      ; IF not enabled for user THEN
16 1AFC    02    1E69          JMP    DISPSTAT      ; display stats and return
17 1AFF    E4          SRET:      CLR    A
18 1B00    80    2C          SJMP   SCLRL      ; ELSE clear line and return
```

```

1          ;;          SETUPSAVE - save setup configuration
2          ;
3          ;          *SETUPSAVE* returns all values back to the EAROM.
4          ;
5          ;
6          ;          ENTRY   none
7          ;
8          ;          EXIT    none
9          ;
10         ;          USES   all
11         ;
12         ;
13 1B02    F1     22          SETUPSAVE:  ACALL  NVRCHK          ; Checksum the NVRAM
14 1B04    F5     34          MOV     NVRSUM, A          ; Update checksum value
15
16 1B06    78     25          MOV     PTR1, #25H          ; EAROM store
17 1B08    90     7000       MOV     DPTR, #EAROM
18 1B0B    7A     10          MOV     INDX1, #16          ; 16 bytes
19
20 1B0D    E6          SAV2:    MOV     A, @PTR1          ; REPEAT
21 1B0E    C4          SWAP     A
22 1B0F    F0          MOVX    @DPTR, A          ; store upper nibble
23 1B10    A3          INC     DPTR
24 1B11    C4          SWAP     A
25 1B12    F0          MOVX    @DPTR, A          ; store lower nibble
26 1B13    A3          INC     DPTR
27 1B14    08          INC     PTR1          ; bump pointers
28 1B15    DA     F6       DJNZ   INDX1, SAV2      ; UNTIL done
29
30 1B17    C2     00          CLR     EAROMSAVE
31 1B19    D1     18          ACALL  WLATCH          ; Save EAROM
32 1B1B    D2     00          SETB   EAROMSAVE
33 1B1D    D1     18          ACALL  WLATCH          ; By toggling flas
34
35 1B1F    41     F6       AJMP   SETUPRET        ; Return from setup mode

```

```
1      ;: SFILL - screen fill
2      ;
3      ; *SFILL* fills the complete screen with the data given it
4      ; into the memory location given it. It does not touch the
5      ; 25th line however.
6      ;
7      ;
8      ; ENTRY (A) = character to write
9      ; (DPTR) = memory location to start at
10     ;
11     ; EXIT none
12     ;
13     ; USES DPTR, INDX1/2
14
15
16 1B21 7A 18 SFILL: MOV INDX1, #MAXLINE ; Go through all lines
17 1B23 7B 50 SFIL1: MOV INDX2, #NUMCHARS ; REPEAT
18 1B25 F0 SFIL2: MOVX @DPTR, A ; REPEAT write character
19 1B26 A3 INC DPTR ; bump pointer
20 1B27 DB FC DJNZ INDX2, SFIL2 ; UNTIL line is done
21 1B28 DA F8 DJNZ INDX1, SFIL1 ; UNTIL screen is done
22 1B2B 22 RET
```

```

1          ;:          SCLRLR - setup clear line reverse video
2          ;          SCLRL  - setup clear line
3          ;
4          ;          *SCLRLR* clears 25th line with reverse video spaces.
5          ;
6          ;          *SCLRL* clears 25th line and places the attributes passed
7          ;          to in on that line.
8          ;
9          ;
10         ;          ENTRY  (A) = lower four bits used for attributes
11         ;
12         ;          EXIT   (A) = unchanged
13         ;
14         ;          USES   all
15
16
17 1B2C    74      01          SCLRLR:    MOV     A, #00000001B      ; Set for reverse video
18
19 1B2E    C0      E0          SCLRL:    PUSH    ACC                ; Save attributes
20 1B30    75      F0      18          MOV     B, #MAXLINE      ; Clear line before display
21 1B33    12      0B96        CALL    CLRLINE
22 1B36    D0      E0          POP     ACC                ; Restore attributes
23
24 1B38    90      5780        MOV     DPTR, #L25MEM
25 1B3B    53      83      EF          ANL    DPH, #ATRANL      ; Point to attribute memory
26 1B3E    7A      50          MOV     INDX1, #NUMCHARS
27 1B40    F0          SLP:    MOVX   @DPTR, A            ; REPEAT write attribute
28 1B41    A3          INC    DPTR                ; bump RAM pointer
29 1B42    DA      FC          DJNZ   INDX1, SLP        ; UNTIL done
30 1B44    22          RET
  
```

```

1      ; ; GETNUM = set number string
2      ;
3      ;
4      ; *GETNUM* sets a number from 0..255 from the keyboard. The
5      ; keys "RETURN" finishes the number and "BACKSPACE" or
6      ; "DELETE" clears the complete number entered. The fourth
7      ; character received automatically makes the routine return.
8      ;
9      ;
10     ; ENTRY (A) = position on 25th line to start
11     ;
12     ; EXIT (A) = number received (0..255)
13     ; (WORK2) = number of characters picked up
14     ;
15     ; USES all
16     ;
17 1B45 FE GETNUM: MOV WORK1, A ; Original position
18
19 1B46 7F 00 GNUM0: MOV WORK2, #0 ; Position count
20 1B48 7C 00 MOV WORK, #0 ; Runnings total
21
22 ;
23 ; Get character
24 1B4A 12 1407 GNUM1: CALL SFCKB ; Get next character
25
26 ;
27 ; Check for end of line
28 1B4D B4 0D 02 CJNE A, #CR, GNUM2 ; IF carriage return THEN
29 1B50 EC GNUMR: MOV A, WORK ; set number and return
30 1B51 22 RET
31
32 ;
33 ; Check for backspace or delete
34 1B52 B4 08 02 GNUM2: CJNE A, #BS, GNUM2A ; IF backspace OR
35 1B55 80 03 SJMP GNUM2B ;
36 1B57 B4 7F 0F GNUM2A: CJNE A, #RUBOUT, GNUM4 ; IF delete THEN
37
38 ;
39 ; Clear current number
40 1B5A EF GNUM2B: MOV A, WORK2
41 1B5B 60 ED JZ GNUM1 ; IF no chars THEN skip it
42 1B5D 1E GNUM2L: DEC WORK1 ; REPEAT decrement position
43 1B5E DF FD DJNZ WORK2, GNUM2L ; UNTIL count = 0
44
45 1B60 EE MOV A, WORK1 ; set position
46 1B61 71 9D ACALL XST ; wipe out old number
47 1B63 20 20 DB / / , ( / '+80H)
48 1B66 A0
49 1B67 80 DD SJMP GNUM0 ; start over again
50
51 ;
52 ; Check number and position
53 1B69 BF 03 02 GNUM4: CJNE WORK2, #3, GNUM4A ; IF all chars THEN
54 1B6C 80 E2 SJMP GNUMR ; exit
55 1B6E B4 3A 00 GNUM4A: CJNE A, #'9'+1, GNUM4B
56 1B71 50 D7 GNUM4B: JNC GNUM1 ; Check upper bounds

```

```

1 1B73 B4 30 00 CJNE A, #'0', GNUM4C
2 1B76 40 D2 GNUM4C: JC GNUM1 ; Check lower bounds
3
4 ; Calculate next number and display
5 ;
6 1B78 94 30 SUBB A, #'0' ; Subtract ASCII offset
7 1B7A C0 E0 PUSH ACC ; Save it
8 1B7C CC XCH A, WORK
9 1B7D 75 F0 0A MOV B, #10
10 1B80 A4 MUL AB ; num := num * 10
11 1B81 2C ADD A, WORK
12 1B82 FC MOV WORK, A ; num := num + work
13 1B83 D0 E0 POP ACC ; Restore number
14 1B85 90 1B93 MOV DPTR, #GNUMS
15 1B88 25 82 ADD A, DPL
16 1B8A F5 82 MOV DPL, A ; Bump address (caution)
17 1B8C EE MOV A, WORK1 ; Get position
18 1B8D 71 A8 ACALL XSTAT ; Display
19 1B8F 0E INC WORK1 ; Bump position
20 1B90 0F INC WORK2 ; Bump count
21 1B91 80 B7 SJMP GNUM1 ; UNTIL done
22
23 ; Must not cross any page boundaries
24 ;
25 1B93 B0 B1 B2 GNUMS: DB ('0'+80H), ('1'+80H), ('2'+80H), ('3'+80H), ('4'+80H)
1B96 B3 B4
26 1B98 B5 B6 B7 DB ('5'+80H), ('6'+80H), ('7'+80H), ('8'+80H), ('9'+80H)
1B9B B8 B9
27
28 IFC NE, HIGH($)-HIGH(GNUMS)
29 ERROR ; 'GNUMS' crosses a page boundary
30 ENDC

```

```
1 ;: XST - transmit string to 25th status line
2 ;
3 ;
4 ; *XST* places a string of displayable characters that
5 ; follow the CALL onto the 25th status line of the screen.
6 ;
7 ;
8 ; ENTRY (A) = position on 25th line
9 ; String to be sent must immediately follow the call
10 ; to this routine. Final character must have bit
11 ; seven set.
12 ;
13 ; EXIT (A) = next position on 25th line
14 ;
15 ; USES A, DPTR, PTR1
16 ;
17 1B9D D0 83 XST: POP DPH ; Get return address
18 1B9F D0 82 POP DPL
19
20 1BA1 71 A8 ACALL XSTAT ; Output string
21
22 1BA3 C0 82 PUSH DPL ; Place return address
23 1BA5 C0 83 PUSH DPH ; back onto the stack
24 1BA7 22 RET
```


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

```

1          ;:          XSTAT - transmit string to 25th status line
2          ;
3          ;
4          ;          *XSTAT* places a string of displayable characters onto
5          ;          the 25th status line of the screen.
6          ;
7          ;
8          ;          ENTRY (A) = position on 25th line
9          ;          (DPTR) = address of string, final character must have
10         ;          bit seven set.
11         ;
12         ;          EXIT (A) = next position on 25th line
13         ;          (DPTR) = position following string
14         ;
15         ;          USES A, DPTR, PTR1
16
17 1BA8 75 A0 57 XSTAT: MOV P2, #HIGH (L25MEM) ; Set address of memory
18 1BAB 24 80 ADD A, #LOW (L25MEM) ; Add in offset to low byte
19 1BAD F8 MOV PTR1, A ; Point to memory
20
21 1BAE E4 XSTL: CLR A ; Get data byte
22 1BAF 93 MOV A, @A+DPTR ; Bump ROM pointer
23 1BB0 A3 INC DPTR ; Bump ROM pointer
24 1BB1 20 E7 04 JB ACC.7, XSTR ; IF bit set THEN exit out
25 1BB4 F2 MOVX @PTR1, A ; Output data
26 1BB5 08 INC PTR1 ; Bump RAM pointer
27 1BB6 80 F6 SJMP XSTL ; Go forever
28
29 1BB8 54 7F XSTR: ANL A, #7FH ; Mask off high bit
30 1BBA F2 MOVX @PTR1, A ; Output last data
31 1BBB E8 MOV A, PTR1 ; Get pointer
32 1BBC 24 81 ADD A, #-LOW (L25MEM) + 1 ; Subtract offset
33 1BBE 22 RET ; Return

```

```
1 ; XLONOFF - transmit string "ON" or "OFF" to 25th status line
2 ;
3 ;
4 ; *XLONOFF* displays "ON" or "OFF" as indicated by the carry
5 ; flag.
6 ;
7 ;
8 ; ENTRY (A) = position on 25th line
9 ; (c) = 0 indicates "OFF" 1 indicates "ON"
10 ;
11 ;
12 ; EXIT (A) = next position on 25th line
13 ;
14 ;
15 1BBF 90 1BC9 XLONOFF: MOV DPTR, #LNOFF ; Get the "OFF" message
16 1BC2 50 03 JNC XLON1 ; IF carry set THEN
17 1BC4 90 1BCC MOV DPTR, #LNON ; set the "ON" message
18 1BC7 61 A8 AJMP XSTAT ; Display message and return
19
20
21
22
23 1BC9 6F 66 E6 LNOFF: DB /off/, (/f/+80H)
24 1BCC 6F 6E A0 LNON: DB /on/, (/v/+80H)
```

```

1          ;;          XMTS - transmit string
2          ;
3          ;
4          ;          *XMTS* transmits a string of characters out of the terminal
5          ;          regardless of whether the terminal is on line or not.
6          ;
7          ;
8          ;          ENTRY  Strings to be sent must immediately follow the call
9          ;          to this routine.  Final character must have bit
10         ;          seven set.
11         ;
12         ;          EXIT   none
13         ;
14         ;          USES   A, DPTR
15
16 1BCF    D0    83          XMTS:      POP    DPH          ; (DPTR) = first character
17 1BD1    D0    82          POP    DPL
18
19 1BD3    E4          XMTS1:     CLR    A
20 1BD4    93          MOV    A, @A+DPTR      ; REPEAT fetch character
21 1BD5    20    E7    06    JB    ACC.7, XMTS2     ; IF bit 7 set THEN exit
22 1BD8    12    139B    CALL   TRANSMIT      ; transmit character
23 1BDB    A3          INC    DPTR          ; bump pointer
24 1BDC    80    F5    SJMP   XMTS1      ; UNTIL forever
25
26 1BDE    12    139B    XMTS2:     CALL   TRANSMIT      ; Transmit last character
27 1BE1    74    01    MOV    A, #1
28 1BE3    73    JMP    @A+DPTR      ; Return to code after string

```



```

1 1C26 D2 OD XL2: SETB GRPH ; Set graphic flag
2
3 ;
4 ; Save character to output
5 1C28 C0 E0 XL3: PUSH ACC
6
7 ;
8 ; See if character is graphic character
9 1C2A 20 OD 30 JB GRPH, XL4 ; IF not graphic AND
10
11 1C2D 30 F5 5D JNB B.5, XL7 ; IF previous was graphic THEN
12 1C30 C2 F5 CLR B.5 ; clear graphic flag
13
14 1C32 30 40 22 JNB ANSI, XL3C ; IF ANSI mode THEN
15 1C35 20 F4 13 JB B.4, XL3B ; IF previous not alt THEN
16 1C38 20 0C 09 JB ALTCHARA, XL3A ; IF this not alt THEN
17 1C3B 71 CF ACALL XMTS ; transmit "EXIT GRAPHIC MODE"
18 1C3D 1B 5B 31 DB ESC, '[11', ('m'+80H)
19 1C40 31 ED
19 1C42 80 49 SJMP XL7
20
21 1C44 71 CF XL3A: ACALL XMTS ; ELSE this is alt THEN
22 1C46 1B 2B B1 DB ESC, '[' , ('1'+80H) ; transmit "ENTER ALT MODE"
23 1C49 80 2E SJMP XL4SET ; flag in alt now
24
25 1C4B 20 0C F6 XL3B: JB ALTCHARA, XL3A ; ELSE IF this not alt THEN
26 1C4E 71 CF ACALL XMTS ; transmit "ENTER NORMAL"
27 1C50 1B 2B C2 DB ESC, '[' , ('B'+80H)
28 1C53 C2 F4 CLR B.4 ; flag in normal now
29 1C55 80 36 SJMP XL7
30
31 1C57 71 CF XL3C: ACALL XMTS ; ELSE transmit in ZDS mode
32 1C59 1B C7 DB ESC, '[' , ('G'+80H)
33 1C5B 80 30 SJMP XL7
34
35 1C5D 20 F5 2D XL4: JB B.5, XL7 ; IF previous not graphic THEN
36 1C60 D2 F5 SETB B.5 ; flag changes
37
38 1C62 30 40 24 JNB ANSI, XL4C ; IF ANSI mode THEN
39 1C65 20 F4 15 JB B.4, XL4B ; IF previous normal THEN
40 1C68 20 0C 09 JB ALTCHARA, XL4A ; IF not alt char THEN
41 1C6B 71 CF XL4AA: ACALL XMTS ; transmit "ENTER GRAPHIC"
42 1C6D 1B 5B 31 DB ESC, '[10', ('m'+80H)
43 1C70 30 ED
43 1C72 80 19 SJMP XL7
44
45 1C74 71 CF XL4A: ACALL XMTS ; ELSE
46 1C76 1B 2B B0 DB ESC, '[' , ('O'+80H) ; transmit "ENTER ALT GRAPHIC"
47 1C79 D2 F4 XL4SET: SETB B.4 ; flag in alt now
48 1C7B 80 10 SJMP XL7
49
50 1C7D 20 0C F4 XL4B: JB ALTCHARA, XL4A ; ELSE this is normal THEN
51 1C80 71 CF ACALL XMTS ; transmit "ENTER NORMAL"
52 1C82 1B 2B C2 DB ESC, '[' , ('B'+80H)
53 1C85 C2 F4 CLR B.4 ; flag in normal now
54 1C87 80 E2 SJMP XL4AA ; and place into graphic
55

```

```

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" flas
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, ('a'+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 flas 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 flas
33 1CB8 80 05 SJMP XL10A ; flas changes
34
35 1CBA 20 F1 04 XL10: JB B.1, XL11 ; IF previous not underline
36 1CBD D2 F1 SETB B.1 ; flas 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 flas
45 1CC9 80 05 SJMP XL12A ; flas changes
46
47 1CCB 20 F2 04 XL12: JB B.2, XL13 ; IF previous not blinking THEN
48 1CCE D2 F2 SETB B.2 ; flas 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 flas
57 1CDA 80 05 SJMP XL14A ; flas changes

```

```

1
2 1CDC 20 F3 04 XL14: JB B.3, XL15 ; IF previous not half THEN
3 1CDF D2 F3 SETB B.3 ; flag changes
4 1CE1 D2 D5 XL14A: SETB F0
5
6 ;
7 ; See if character is alternate graphic
8 1CE3 20 0C 19 XL15: JB ALTCHARA, XL16 ; IF not alt AND
9
10 1CE6 30 F4 31 JNB B.4, XL17 ; IF previous was alt THEN
11 1CE9 C2 F4 CLR B.4 ; clear alt flag
12
13 1CEB 30 40 2A JNB ANSI, XL16B ; IF ANSI mode THEN
14 1CEE 71 CF ACALL XMTS ; transmit "EXIT EVERYTHING"
15 1CF0 1B 28 C2 DB ESC, '(' , ('B'+80H)
16
17 1CF3 30 F5 24 JNB B.5, XL17 ; IF graphics mode THEN
18 1CF6 71 CF ACALL XMTS ; transmit "ENTER GRAPHIC MODE"
19 1CF8 1B 5B 31 DB ESC, '[' , ('m'+80H)
20 1CFD 80 1B SJMP XL17
21
22 1CFF 20 F4 18 XL16: JB B.4, XL17 ; IF previous not alt THEN
23 1D02 D2 F4 SETB B.4 ; flag changes
24
25 1D04 30 40 11 JNB ANSI, XL16B ; IF ANSI mode THEN
26 1D07 20 F5 07 JB B.5, XL16A ; IF no graphics THEN
27 1D0A 71 CF ACALL XMTS ; transmit "ENTER ALT MODE"
28 1D0C 1B 28 B1 DB ESC, '(' , ('1'+80H)
29 1D0F 80 09 SJMP XL17
30
31 1D11 71 CF XL16A: ACALL XMTS ; ELSE transmit "ENTER ALT GRAPHIC"
32 1D13 1B 28 B0 DB ESC, '(' , ('0'+80H)
33 1D16 80 02 SJMP XL17
34
35 1D18 D2 D5 XL16B: SETB F0
36
37 ;
38 ; Transmit any changes
39 1D1A 30 D5 46 XL17: JNB F0, XL20 ; IF no changes THEN skip
40
41 1D1D 20 40 1F JB ANSI, XL18 ; Check for mode
42
43 ;
44 ; ZDS mode changes
45 1D20 71 CF ACALL XMTS ; set attributes
46 1D22 1B F3 DB ESC, ('s'+80H)
47 1D24 C0 F0 PUSH B ; save attributes
48 1D26 E5 F0 MOV A, B ; reorder bits
49 1D28 13 RRC A ; set reverse video
50 1D29 13 RRC A ; set underline
51 1D2A 92 F3 MOV B.3, C ; put in position
52 1D2C 13 RRC A ; set blink
53 1D2D 92 F1 MOV B.1, C ; put in position
54 1D2F 13 RRC A ; set half intensity
55 1D30 92 F2 MOV B.2, C ; put in position
56 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

```


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

```

1          ;;          XMTL - transmit line
2          ;          XMTL1 - entry to transmit line
3          ;
4          ;          *XMTL* transmits the entire contents of the line as defined
5          ;          by transmit character *XMTC*.
6          ;
7          ;          *XMTL1* needs a current transmit mode.
8          ;
9          ;
10         ;          ENTRY (A) = current transmit mode (*XMTL1*)
11         ;          YPOS points to screen line
12         ;
13         ;          EXIT (A) = current transmit mode (updated)
14         ;
15         ;          USES all
16         ;
17
18 1D74    E4          XMTL:    CLR    A          ; Start with no attributes
19 1D75    C0      11    XMTL1:  PUSH   XPOS      ; Save current position
20 1D77    75      11      00    MOV   XPOS, #0 ; Start at beginning of line
21 1D7A    71      E4          XML2:  ACALL  XMTC      ; REPEAT transmit character
22 1D7C    05      11          INC   XPOS      ; bump position
23 1D7E    AC      11          MOV   WORK, XPOS
24 1D80    BC      50      F7    CJNE  WORK, #NUMCHARS, XML2 ; UNTIL done
25 1D83    D0      11          POP   XPOS      ; Restore position
26 1D85    FC          MOV   WORK, A    ; Save (A)
27 1D86    12      04EA    CALL  BLINAD   ; Update address
28 1D89    EC          MOV   A, WORK   ; Restore (A)
29 1D8A    22          RET          ; Return

```

```

1          ;          ;          ;          ;          ;          ;          ;
2          ;          ;          ;          ;          ;          ;          ;
3          ;          ;          ;          ;          ;          ;          ;
4          ;          ;          ;          ;          ;          ;          ;
5          ;          ;          ;          ;          ;          ;          ;
6          ;          ;          ;          ;          ;          ;          ;
7          ;          ;          ;          ;          ;          ;          ;
8          ;          ;          ;          ;          ;          ;          ;
9          ;          ;          ;          ;          ;          ;          ;
10         ;          ;          ;          ;          ;          ;          ;
11         ;          ;          ;          ;          ;          ;          ;
12         ;          ;          ;          ;          ;          ;          ;
13         ;          ;          ;          ;          ;          ;          ;
14 1D8B    30      40      0F      SXMTC:  JNB   ANSI, SXC1      ; IF ANSI mode THEN
15 1D8E    12      09D1    0F      CALL  GNP           ; set parameter
16 1D91    60      19      0F      JZ    XMTF           ; IF zero THEN xmt page
17 1D93    14      0F      0F      DEC   A             ;
18 1D94    60      12      0F      JZ    SXMTL         ; IF one THEN xmt line
19 1D96    14      0F      0F      DEC   A             ;
20 1D97    60      04      0F      JZ    SXC1         ; IF two THEN xmt character
21 1D99    14      0F      0F      DEC   A             ;
22 1D9A    60      2A      0F      JZ    XMT25        ; IF three THEN xmt 25th line
23 1D9C    22      0F      0F      RET   A             ; ELSE return
24         ;          ;          ;          ;          ;          ;          ;
25 1D9D    C0      3F      00      SXC1:  PUSH  PFIELD      ; Save protected fields
26 1D9F    75      3F      00      MOV  PFIELD, #0    ; Make none protected
27 1DA2    E4      0F      0F      CLR  A             ; No attributes set
28 1DA3    71      E4      0F      ACALL XMTC         ; Transmit character
29 1DA5    D0      3F      0F      POP  PFIELD       ; Restore protected fields
30 1DA7    22      0F      0F      RET   A             ;

```

```
1          ;;          SXMTL - special transmit line
2          ;
3          ;          *SXMTL* transmits the entire contents of the line as defined
4          ;          by transmit character *XMTC*. The end of the line is signaled
5          ;          by a carriage return.
6          ;
7          ;
8          ;          ENTRY   none
9          ;
10         ;          EXIT    none
11         ;
12         ;          USES    all
13         ;
14         ;
15 1DAS    B1    74          SXMTL:    ACALL  XMTL          ; Transmit line
16 1DAA    80    14          SJMP    XMP3          ; Send CR, bell, and return
```

```

1      ;:          XMTP - transmit page
2      ;
3      ;          *XMTP* transmits the entire contents of the page (excluding
4      ;          25th line) as defined by transmit character *XMTC*. The end
5      ;          of the page is signaled by a carriage return.
6      ;
7      ;
8      ;          ENTRY   none
9      ;
10     ;          EXIT    none
11     ;
12     ;          USES    all
13
14
15 1DAC  E4          XMTP:      CLR     A          ; Start with no attributes set
16 1DAD  C0          PUSH     YPOS         ; Save current position
17 1DAF  75 12 00    MOV      YPOS, #0
18 1DB2  B1 75      XMP1:      ACALL    XMTL1        ; REPEAT transmit line
19 1DB4  05 12      INC      YPOS         ; bump position
20 1DB6  AC 12      MOV      WORK, YPOS
21 1DB8  BC 18 F7    CJNE     WORK, #MAXLINE, XMP1 ; UNTIL done
22
23 1DBB  D0 12      XMP2:      POP      YPOS         ; Restore position
24 1DBD  12 04EA    CALL     BLINAD        ; Update address
25
26 1DC0  71 CF      XMP3:      ACALL    XMTS         ; Send CR to terminate page
27 1DC2  8D        DB      (CR+80H)
28
29 1DC3  02 066B    JMP     RING          ; Signal done to user

```

```
1          ;;          XMT25 - transmit 25th line
2          ;
3          ;
4          ;          *XMT25* transmits the 25th line in the same manner as *XMTL*
5          ;          but only if the 25th line is enabled. The end of the line
6          ;          is signaled by a carriage return.
7          ;
8          ;          ENTRY   none
9          ;
10         ;          EXIT    none
11         ;
12         ;          USES   all
13
14
15 1DC6    30    15    F7    XMT25:    JNB    L25EN, XMP3    ; IF 25th line enabled THEN
16 1DC9    C0    12                PUSH   YPOS    ; save current position
17 1DCB    75    12    18    MOV    YPOS, #MAXLINE ; do 25th line
18 1DCE    B1    74                ACALL  XMTL    ; transmit 25th line
19 1DD0    80    E9                SJMP   XMP2    ; restore, update, return
```

```

1          ;:          PRINT - print page
2          ;
3          ;
4          ;          *PRINT* acts exactly like *XMT* (transmit page) except that
5          ;          the data goes out to the opposite port and every line ends with
6          ;          a carriage return and line feed.
7          ;
8          ;
9          ;          ENTRY none
10         ;
11         ;
12         ;          EXIT none
13         ;
14         ;
15 1DD2    D2    23          PRINT:    SETB    PRNPF          ; Flag printing
16
17 1DD4    12    13C4       LCALL    DOXOFF         ; Output an XOFF
18
19 1DD7    D1    0F          ACALL    CMPORT         ; Toggle port
20
21 1DD9    C0    3F          PUSH    PFIELD         ; Save protected fields
22 1DDB    75    3F    00    MOV     PFIELD, #0      ; Make all unprotected
23
24 1DEE    E4          CLR     A              ; Start with no attributes set
25 1DDF    C0    12          PUSH    YPOS           ; Save current position
26 1DE1    75    12    00    MOV     YPOS, #0
27
28 1DE4    B1    75          PRNT1:    ACALL    XMTL1         ; REPEAT transmit line
29 1DE6    C0    E0          PUSH    ACC           ; save attributes
30 1DE8    71    CF          ACALL    XMT5         ; transmit CR and LF
31 1DEA    0D    8A          DB     CR, (LF+80H)
32 1DEC    D0    E0          POP     ACC           ; restore attributes
33
34 1DEE    05    12          INC     YPOS          ; bump position
35 1DF0    AC    12          MOV     WORK, YPOS
36 1DF2    BC    18    EF    CJNE    WORK, #MAXLINE, PRNT1 ; UNTIL done
37
38 1DF5    D0    12          POP     YPOS          ; Restore position
39 1DF7    12    04EA       LCALL    BLINAD       ; Update address
40
41 1DFA    D0    3F          POP     PFIELD        ; Restore protected fields
42
43 1DFC    D1    0F          ACALL    CMPORT         ; Toggle port again
44
45 1DFE    20    22    03    JB     HSSTOPF, PRNT2 ; IF not hold screen THEN
46 1E01    12    13D5       LCALL    DOXON        ; output an -XON-
47
48 1E04    C2    23          PRNT2:    CLR     PRNPF         ; Flag done printing
49
50 1E06    12    02E8       LCALL    LKB         ; Look at keyboard
51 1E09    50    03          JNC    PRNT3         ; IF no keyboard data THEN
52
53 1E0B    02    066B       LJMP   RING         ; signal done to user
54 1E0E    22          PRNT3:    RET

```

```

1          ;;          CMPORT - complement port
2          ;
3          ;
4          ;          *CMPORT* toggles direction of the I/O port.
5          ;
6          ;
7          ;          ENTRY   none
8          ;
9          ;          EXIT    none
10         ;
11         ;          USES   A, DPTR
12         ;
13 1E0F    30    13    FD    CMPORT:    JNB    OKTRANS, CMPORT    ; Wait until transmitter empty
14 1E12    B2    02                CPL    PORTF                ; Toggle port flas
15 1E14    B2    3C                CPL    PRTF2                ; Toggle EAROM flas
16 1E16    C2    18                CLR    XOFFRCVED          ; Clear any received -XOFF-
17
18         ;          SJMP   WLATCH                ; Write and return
19
20
21
22
23         ;;          WLATCH - write latch
24         ;
25         ;          -WLATCH- writes the current value of the byte *XLATCH* out
26         ;          to the memory mapped latch.
27         ;
28         ;
29         ;          ENTRY   none
30         ;
31         ;          EXIT    none
32         ;
33         ;          USES   A, DPTR
34
35
36 1E18    E5    20                WLATCH:    MOV    A, XLATCH                ; Get byte
37 1E1A    90    3000              MOV    DPTR, #MLATCH          ; Get address
38 1E1D    F0                MOVX   @DPTR, A                ; Output data to latch
39 1E1E    22                RET

```

```

1      ;:          DCLK - Display clock
2      ;
3      ;          DCLK0 - always display clock
4      ;
5      ;          *DCLK* displays the current settings of the clock on the 25th
6      ;          line once every second only.
7      ;
8      ;          *DCLK0* always displays the current settings of the clock on
9      ;          the 25th line.
10     ;
11     ;
12     ;
13     ;          ENTRY  none
14     ;
15     ;          EXIT   none
16     ;
17     ;          USES   A, B, DPTR
18     ;
19
20 1E1F  E5  3E          DCLK:  MOV  A, DISPSEC
21 1E21  B5  3B  01     CJNE  A, TSEC, DCLK0  ; IF no change in seconds THEN
22 1E24  22             RET          ; exit
23
24 1E25  85  3B  3E     DCLK0: MOV  DISPSEC, TSEC  ; Init *DISPSEC*
25 1E28  E5  3D         MOV  A, THOUR
26 1E2A  12  0A92      CALL  CBA          ; Convert hours to ASCII
27 1E2D  C0  82         PUSH  DPL          ; Save ones digit
28 1E2F  E5  83         MOV  A, DPH
29 1E31  B4  30  02     CJNE  A, #'0', DCLK1  ; IF tens digit = '0' THEN
30 1E34  74  20             MOV  A, #' '      ; suppress leading digit
31 1E36  90  57C8      DCLK1: MOV  DPTR, #L25MEM+72
32 1E39  F0             MOVX @DPTR, A      ; Place on screen
33 1E3A  D0  E0         POP  ACC
34 1E3C  A3             INC  DPTR
35 1E3D  F0             MOVX @DPTR, A      ; Place on screen
36 1E3E  74  3A         MOV  A, #' ':
37 1E40  A3             INC  DPTR
38 1E41  F0             MOVX @DPTR, A      ; Place ':' on screen
39 1E42  E5  3C         MOV  A, TMIN
40 1E44  12  0A92      CALL  CBA          ; Convert minutes to ASCII
41 1E47  C0  82         PUSH  DPL          ; Save ones digit
42 1E49  E5  83         MOV  A, DPH
43 1E4B  90  57CB      MOV  DPTR, #L25MEM+75
44 1E4E  F0             MOVX @DPTR, A      ; Place tens digit on screen
45 1E4F  D0  E0         POP  ACC
46 1E51  A3             INC  DPTR
47 1E52  F0             MOVX @DPTR, A      ; Place ones digit on screen
48 1E53  74  3A         MOV  A, #' ':
49 1E55  A3             INC  DPTR
50 1E56  F0             MOVX @DPTR, A      ; Place ':' on screen
51 1E57  E5  3B         MOV  A, TSEC
52 1E59  12  0A92      CALL  CBA          ; Convert seconds to ASCII
53 1E5C  C0  82         PUSH  DPL          ; Save ones digit
54 1E5E  E5  83         MOV  A, DPH
55 1E60  90  57CE      MOV  DPTR, #L25MEM+78
56 1E63  F0             MOVX @DPTR, A      ; Place tens digit on screen
57 1E64  D0  E0         POP  ACC

```


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

1	1E66	A3	INC	DPTR	
2	1E67	F0	MOVX	@DPTR, A	; Place ones digit on screen
3	1E68	22	RET		

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

```

1          ;:          DISPSTAT - display status line
2          ;
3          ;
4          ;          *DISPSTAT* displays everything that can be displayed onto
5          ;          the 25th line if the option is enabled. This includes
6          ;          clock, caps locked, on/off line indicators, and insert
7          ;          character mode.
8          ;
9          ;          ENTRY   none
10         ;
11         ;          EXIT    none
12         ;
13         ;          USES   all
14
15
16 1E69    30    37    03    DISPSTAT:  JNB    L250N, DSPR    ; IF line 25 off OR
17 1E6C    30    15    01          JNB    L25EN, DSP0    ; IF line 25 enabled THEN
18 1E6F    22          DSPR:         RET          ; return
19
20 1E70    74    08          DSP0:         MOV    A, #00001000B
21 1E72    71    2E          ACALL   SCLRL    ; Clear with half intensity
22
23 1E74    D1    25          ACALL   DCLKO    ; Display clock
24
25 1E76    E4          CLR    A
26 1E77    90    1E94        MOV    DPTR, #MSGCL    ; Caps locked
27 1E7A    30    35    02    JNB    CAPLOCKF, DSP1
28 1E7D    71    A8          ACALL   XSTAT
29
30 1E7F    74    10          DSP1:         MOV    A, #16
31 1E81    90    1EA5        MOV    DPTR, #MSGOL    ; Offline
32 1E84    20    2E    02    JB     ONLINE, DSP2
33 1E87    71    A8          ACALL   XSTAT
34
35 1E89    74    1D          DSP2:         MOV    A, #29
36 1E8B    90    1EB3        MOV    DPTR, #MSGIC    ; Insert character
37 1E8E    30    21    02    JNB    ICMODEF, DSP3
38 1E91    71    A8          ACALL   XSTAT    ; Output
39
40 1E93    22          DSP3:         RET

```

1 ; Caps locked message

2 ;
3 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

4 ;
5 ;
6 ; Off line message

7 ;
8 1EA5 00 20 20 MSGCL: DB 00H, / OFF LINE /, (00H+80H)
1EA8 4F 46 46
1EAB 20 4C 49
1EAE 4E 45 20
1EB1 20 80

9 ;
10 ;
11 ; Insert character message

12 ;
13 1EB3 00 20 20 MSGCL: DB 00H, / INSERT MODE /, (00H+80H)
1EB6 4F 4E 53
1EB9 45 52 54
1EBC 20 4D 4F
1EBF 44 45 20
1EC2 20 80

```

1      ;:          KBCHK - keyboard check
2      ;
3      ;          *KBCHK* waits for the keyboard to send -KB_POWER- within
4      ;          75 ms. If it is not found then an error condition exists.
5      ;          This routine must only be done after a hardware reset so that
6      ;          it is known that the keyboard is going to send the data.
7      ;
8      ;
9      ;          ENTRY   none
10     ;
11     ;          EXIT    none
12     ;
13     ;          USES   all
14     ;
15
16 1EC4 7A 0D      KBCHK:  MOV   INDX1, #13
17 1EC6 7B 00      KBC0:  MOV   INDX2, #0          ; REPEAT init count
18 1EC8 12 02B6    KBC1:  CALL  FCKB          ; REPEAT set any data
19 1ECB 40 04      JC     KBC2          ; IF data AND
20 1ECD B4 BF F8    CJNE  A, #KB_POWER, KBC1 ; IF power up THEN
21 1ED0 22      RET          ; return
22 1ED1 DB F5      KBC2:  DJNZ  INDX2, KBC1 ; UNTIL done
23 1ED3 DA F1      DJNZ  INDX1, KBC0 ; UNTIL all done
24
25 1ED5 43 40 08    ORL   ERRORS, #00001000B ; ELSE error #3
26 1ED8 22      RET

```

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

```
1          ; ROMCHK - ROM checksum
2          ;
3          ; -ROMCHK- does a checksum on the ROM and compares it with
4          ; the known value in ROM.
5          ;
6          ;
7          ; ENTRY none
8          ;
9          ; EXIT none
10         ;
11         ; USES all
12
13
14 1ED9 7C 00 ROMCHK: MOV WORK, #0 ; Init checksum
15 1EDB 90 0000 MOV DPTR, #0 ; Init pointer
16 1EDE 7A 20 MOV INDX1, #20H ; First index
17 1EE0 7B 00 MOV INDX2, #00H ; Init count
18
19 1EE2 E4 RMC1: CLR A ; REPEAT
20 1EE3 93 MOV A, @A+DPTR ; set ROM byte
21 1EE4 2C ADD A, WORK ; add to previous amount
22 1EE5 FC MOV WORK, A ; replace
23 1EE6 A3 INC DPTR ; bump pointer
24 1EE7 DB F9 DJNZ INDX2, RMC1 ; UNTIL done
25 1EE9 DA F7 DJNZ INDX1, RMC1 ; UNTIL all done
26
27 1EEB 60 03 JZ RMC2 ; IF error THEN
28 1EED 43 40 01 ORL ERRORS, #00000001B ; error #0
29 1EF0 22 RMC2: RET
```

```

1      ;:          RAMCHK - RAM check
2      ;
3      ;          -RAMCHK- checks the RAM for any failures.  It writes two
4      ;          different values into character and attribute RAM and reads
5      ;          them back.  Any errors are flagged.
6      ;
7      ;
8      ;          ENTRY   none
9      ;
10     ;          EXIT    none
11     ;
12     ;          USES    a'1
13
14
15     1EF1    90    5000    RAMCHK:    MOV    DPTR, #CHARMEM    ; Point to character RAM
16     1EF4    7C    55          MOV    WORK, #01010101B    ; First pattern
17
18     1EF6    7A    08          RAM0:    MOV    INDX1, #08H    ; First index
19     1EF8    7B    00          MOV    INDX2, #00H    ; Second index
20
21     1EFA    EC          RAM1:    MOV    A, WORK    ; REPEAT
22     1EFB    F0          MOVX   @DPTR, A    ; write pattern
23     1EFC    53    83    EF    ANL    DPH, #ATRANL    ; move to attribute RAM
24     1EFF    F0          MOVX   @DPTR, A    ; write pattern
25     1F00    E0          MOVX   A, @DPTR    ; read pattern
26     1F01    54    0F    ANL    A, #0FH    ; mask
27     1F03    FD    MOV    TEMP, A    ; save
28     1F04    EC    MOV    A, WORK    ; set original pattern
29     1F05    54    0F    ANL    A, #0FH    ; mask it also
30     1F07    B5    05    14    CJNE   A, TEMP, RAMERR    ; IF error THEN exit
31     1F0A    43    83    10    ORL    DPH, #CHQRJL    ; move to character RAM again
32     1F0D    E0          MOVX   A, @DPTR    ; read pattern
33     1F0E    B5    04    0D    CJNE   A, WORK, RAMERR    ; IF error THEN exit
34     1F11    A3    INC    DPTR    ; bump pointer
35     1F12    DB    E6    DJNZ   INDX2, RAM1    ; UNTIL done
36     1F14    DA    E4    DJNZ   INDX1, RAM1    ; UNTIL all done
37
38     1F16    BC    55    04    CJNE   WORK, #01010101B, RAM2    ; IF pattern is original THEN
39     1F19    7C    AA    MOV    WORK, #10101010B    ; change to second pattern
40     1F1B    80    D9    SJMP   RAM0    ; and do it again
41
42     1F1D    22          RAM2:    RET    ; IF no error THEN return
43
44     1F1E    43    40    02    RAMERR:  ORL    ERRORS, #00000010B    ; ELSE flas error #1 and return
45     1F21    22          RET

```

```

1          ; ; NVRCHK - non-volatile RAM checksum
2          ;
3          ; *NVRCHK* does a checksum on the NVRAM and returns the value.
4          ;
5          ;
6          ; ENTRY none
7          ;
8          ; EXIT (A) = checksum value of NVRAM
9          ;
10         ; USES A, WORK, INDX1, PTR1
11
12
13 1F22 E4          NVRCHK: CLR A ; Init for checksum
14 1F23 78 25      MOV PTR1, #25H
15 1F25 7A 0F      MOV INDX1, #15
16
17 1F27 26          NVR1: ADD A, @PTR1 ; REPEAT add next value
18 1F28 08          INC PTR1 ; bump RAM pointer
19 1F29 DA FC      DJNZ INDX1, NVR1 ; UNTIL done
20 1F2B 22          RET

```

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

```

1          ;:          ERROR - diagnostic error
2          ;
3          ;
4          ;          -ERROR- outputs the error messages for any errors that
5          ;          may have been flagged in the -ERRORS- byte. The status
6          ;          line is displayed if there are no errors. A bell is
7          ;          given when finished and no errors occurred.
8          ;
9          ;          Format of the error byte is as follows:
10         ;
11         ;          BIT      0 = ROM checksum error
12         ;                   1 = RAM error
13         ;                   2 = CRTC error
14         ;                   3 = Keyboard error
15         ;                   4 = Non-volatile RAM checksum error
16         ;
17         ;
18         ;          ENTRY   none
19         ;
20         ;          EXIT    none
21         ;
22         ;          USES    all
23
24 1F2C    E5    40          ERROR:    MOV     A, ERRORS
25 1F2E    70    05          JNZ     ERRO          ; IF no errors THEN
26 1F30    D1    69          ACALL   DISPSTAT      ; display status
27 1F32    02    066B      LJMP    RING          ; ring bell and return
28
29 1F35    E4          ERROR:    CLR     A
30 1F36    71    2E          ACALL   SCLRL          ; Clear line full intensity
31 1F38    D1    25          ACALL   DCLKO          ; Display clock
32 1F3A    D1    0F          ACALL   CMPORT          ; Change ports
33
34 1F3C    85    40    F0      MOV     B, ERRORS          ; Get errors
35
36 1F3F    71    CF          ACALL   XMTS          ; Output message
37 1F41    45    52    52      ERM1:    DB     'ERROR -', ( '/' +80H)
38 1F44    4F    52    20
39 1F47    2D    A0
40
41 1F49    E4          CLR     A
42 1F4A    90    1F41      MOV     DPTR, #ERM1
43 1F4D    71    A8          ACALL   XSTAT          ; Output to 25th line
44
45 1F4F    30    F0    17      JNB    B.0, ERR1          ; IF ROM checksum error THEN
46 1F52    FF          MOV     WORK2, A
47 1F53    71    CF          ACALL   XMTS          ; output message
48 1F55    52    4F    4D      ERM2:    DB     'ROM checksum ', ( '/' +80H)
49 1F58    20    63    68
50 1F5B    65    63    6B
51 1F5E    73    75    6D
52 1F61    2F    A0
53
54 1F63    EF          MOV     A, WORK2
55 1F64    90    1F55      MOV     DPTR, #ERM2
56 1F67    71    A8          ACALL   XSTAT          ; output to 25th line
57
58
59
60
61 1F69    30    F1    14      ERM1:    JNB    B.1, ERR2          ; IF RAM error THEN

```


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

```

1 1F6C FF          MOV    WORK2, A
2 1F6D 71 CF      ACALL  XMTS          ; output message
3 1F6F 52 41 4D   ERM3:  DB    'RAM fault ', (''+80H)
  1F72 20 66 61
  1F75 75 6C 74
  1F78 20 A0
4 1F7A EF          MOV    A, WORK2
5 1F7B 90 1F6F    MOV    DPTR, #ERM3
6 1F7E 71 A8      ACALL  XSTAT          ; output to 25th line
7
8 1F80 30 F2 15   ERR2:  JNB    B.2, ERR3          ; IF CRTC error THEN
9 1F83 FF          MOV    WORK2, A
10 1F84 71 CF     ACALL  XMTS          ; output message
11 1F86 43 52 54   ERM4:  DB    'CRTC error ', (''+80H)
  1F89 43 20 65
  1F8C 72 72 6F
  1F8F 72 20 A0
12 1F92 EF          MOV    A, WORK2
13 1F93 90 1F86    MOV    DPTR, #ERM4
14 1F96 71 A8      ACALL  XSTAT          ; output to 25th line
15
16 1F98 30 F3 13   ERR3:  JNB    B.3, ERR4          ; IF keyboard error THEN
17 1F9B FF          MOV    WORK2, A
18 1F9C 71 CF     ACALL  XMTS          ; output message
19 1F9E 4B 65 79   ERM5:  DB    'Keyboard ', (''+80H)
  1FA1 62 6F 61
  1FA4 72 64 20
  1FA7 A0
20 1FA8 EF          MOV    A, WORK2
21 1FA9 90 1F9E    MOV    DPTR, #ERM5
22 1FAC 71 A8      ACALL  XSTAT          ; output to 25th line
23
24 1FAE 30 F4 18   ERR4:  JNB    B.4, ERR5          ; IF NVRAM error THEN
25 1FB1 FF          MOV    WORK2, A
26 1FB2 71 CF     ACALL  XMTS          ; output message
27 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
28 1FC3 EF          MOV    A, WORK2
29 1FC4 90 1FB4    MOV    DPTR, #ERM6
30 1FC7 71 A8      ACALL  XSTAT          ; output to 25th line
31
32 1FC9 71 CF     ERM5:  ACALL  XMTS          ; Output CR, LF's
33 1FCB 0D 0A 0D   DB    CR, LF, CR, (LF+80H)
  1FCE 8A
34
35 1FCF C1 0F      AJMP  CMPORT          ; Change ports back

```

```

1          ;:          RAMCONST - RAM constants
2          ;
3          ;          *RAMCONST* is the table with all of the RAM constants that
4          ;          the terminal will power up with.
5          ;
6 1FD1     D0          RAMCONST:  DB      LOW (IBUF)          ; ISTORE
7 1FD2     D0          DB      LOW (IBUF)          ; IFETCH
8 1FD3     00          DB      0                    ; ICOUNT
9
10 1FD4     F0          DB      LOW (OBUF)          ; OSTORE
11 1FD5     F0          DB      LOW (OBUF)          ; OFETCH
12
13 1FD6     00          DB      0                    ; XPOS
14 1FD7     00          DB      0                    ; YPOS
15 1FD8     50          DB      HIGH (CHARMEM)       ; LINADH
16 1FD9     00          DB      LOW (CHARMEM)        ; LINADL
17
18 1FDA     00          DB      HIGH (NORM)          ; DSPADH
19 1FDB     87          DB      LOW (NORM)           ; DSPADL
20
21 1FDC     00          DB      0                    ; TSCROLL
22 1FDD     18          DB      MAXLINE              ; BFIX
23
24 1FDE     00          DB      0                    ; PMADRH
25 1FDF     00          DB      0                    ; PMADRL
26 1FE0     00          DB      0                    ; PMNUM
27 1FE1     00          DB      0                    ; PMVALUE
28
29 1FE2     00          DB      0                    ; KBIDNUM
30 1FE3     01          DB      1                    ; HSLINE
31 1FE4     00          DB      0                    ; GSET
32
33 1FE5     39          DB      00111001B           ; XLATCH      bits #1
34 1FE6     00          DB      00000000B           ; ATTRIBUTES  bits #2
35 1FE7     08          DB      00001000B           ; Bits #3
36 1FE8     0C          DB      00001100B           ; Bits #4
37 1FE9     30          DB      00110000B           ; Bits #5
38
39 1FEA     00          DB      0                    ; XXPOS
40 1FEB     00          DB      0                    ; XYPOS
41
42 1FEC     20          DB      32                    ; BLNKRATE
43 1FED     00          DB      0                    ; BLNKCOUNT
44 1FEE     00          DB      0                    ; AUTOCNT
45
46 1FEF     00          DB      0                    ; TCNT
47 1FF0     00          DB      0                    ; TSEC
48 1FF1     00          DB      0                    ; TMIN
49 1FF2     00          DB      0                    ; THOUR
50
51 1FF3     FF          DB      255                  ; DISPSEC
52
53 1FF4     00          DB      0                    ; PFIELD

```


SYMBOL TABLE

AC	%00D6	01	ACA1	0A04	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	ACL0	0C4B	03
ACPR	0AAF	03	ACPT	0ACF	03	ACP1A	0ADB	03	ACP2	0ADC	03	ACP3	0ACC	03
ACR	0C74	03	ACRT	0C6E	03	ACR1	0C73	03	ACU	0CA3	03	ACUP	0C7D	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	AFDC	0E32	03	APDC0	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	APIL0	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	ASG0	0A4E	03	ASG0	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	
ATTRIB	= 4000		ATTRIB	= 0021		AT2	0635	03	AUSCP	0F03	03	AUTOCN	= 0039	
AUTOGR	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	BLNKN	= 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	CCSR	= 0080		CDN	0CAB	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
CPR0	0E8B	03	CPR0A	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	CUB0	0CF2	03
CUB1	0CFE	03	CUC	0D04	03	CUC0	0D14	03	CUC0A	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	DCLK0	1E25	03	DCLK1	1E36	03
DCMDRE	= 2001		DDATRE	= 2000		DEOL	0B00	03	DISPSE	= 003E		DISPST	1E69	03
DKI	0FB9	03	DMAANL	= 00CF		DMAHRH	= 000A		DMADRL	= 000B		DMAMEM	= 6000	
DMAORL	= 0020		DMA1YP	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	DSR0K	0D42	03	DSR0	0D29	03
DSR0A	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	F13	0217	03	FKDLY	02E1	03
FKDLY1	02E4	03	FKL	02C7	03	FKR	02DD	03	FK1	02BE	03	FK2	02D8	03
FL1	1219	03	F0R	026B	03	F01	0253	03	FREQ	0039	01	FULLDP	002F	01
FO	%00D5	01	GATM	0025	01	GDES	0F4F	03	GDES R	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
HNSHK	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
ITO	%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_LEC	= 0082		KBC_LEK	= 0089		KBC_LEL	= 0084	
KBC_ER	= 0086		KBC_ID	= 0080		KBC_OF	= 008A		KBC_ON	= 008B		KBC_XL	= 0085	
KBC0	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
KBSCRO	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_LERA	= 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_LUP	= 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	LN0FF	1BC9	03	LN0N	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	1987	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	L0TL	= 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	= 166C	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	MNIC	005D	03	MN1D	0062	03
MN2	0065	03	MN3	006E	03	MODE	= 0028		MONIT0	003A	01	MSGCL	1E94	03
MSGIC	1EB3	03	MSGOL	1EA5	03	NKC	0B2D	03	NORM	0087	03	NRMO	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	PDO	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	PK0	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	P01	022F	03	P02	0234	03	P03	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		FT0	%00B9	01
PT1	%00BB	01	PUTCHR	0157	03	PUI	017D	03	PU2	0186	03	PX0	%00B8	01
PX1	%00BA	01	PO	%0080	00	P1	%0090	00	P2	%00A0	00	P3	%00B0	00
RAMCHK	1EF1	03	RAMCON	1FB1	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	080A	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_ST0=	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
SIO	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	SFF	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	1744	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	SXMTG	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	TBS	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	TLO	%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	TRO	%008C	01
TR1	%008E	01	TSCROL=	0017		TSEC =	003B		TXD	%00B1	01	TQ	%00B4	01
T1	%00B5	01	UNDEF	0767	03	UNDLNA	0009	01	USCP	0EF4	03	USCP1	0EFF	03
USCP2	0F01	03	UXOFFS	0011	01	VSP	0072	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	0F24	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	XL9	1CA0	03	XL9A	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
XPQS	= 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	XXPQS	= 0035		XYPQS	= 0036		YPOS	= 0012		ZAR	0F25	03
ZATR	0F08	03	ZATRO	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

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*					
E2SL	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.)

HT0	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	8-17#	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#												
INS	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.)

L6T	199-17	199-54#	200-11			
L6TL	199-16	200-11#				
L7	202-14	203-43#				
L7C1	204-10	204-15#				
L7C2	204-12	204-14	204-16	204-18#		
L7D	204-24	204-26#				
L7T	202-17	203-12#	203-26			
L7TL	202-16	203-26#				
L8	206-14	208-51#				
L8A	209-3	209-5#				
L8T	206-17	207-49#	208-9			
L8T2	206-36	208-19#	208-34			
L8TL	206-16	208-9#				
L8TL2	206-35	208-34#				
LCIF	27-16#	56-19				
LF	2-7#	19-55	230-31	241-33	241-33	
LI1	27-17	27-21#				
LINADH	8-16#	22-21	22-40*	44-25*	115-20	
LINADL	8-17#	22-20	22-39*	44-22*	115-19	
LKR	32-14#	207-34	230-50			
LKBR	32-15	32-17#				
LKUP1	172-20	172-24	172-27#			
LN1B1	184-20	184-32#				
LN1B2	184-22	184-33#				
LN3B	191-2	191-51#				
LN3B1	191-5	191-52#				
LN3B10	191-32	192-4#				
LN3B11	191-35	192-5#				
LN3B12	191-38	192-6#				
LN3B13	191-41	192-7#				
LN3B14	190-5	191-11	191-17	191-23	192-8#	
LN3B3	191-8	191-53#				
LN3B5	191-14	191-54#				
LN3B7	191-20	192-1#				
LN3B8	191-26	192-2#				
LN3B9	191-29	192-3#				
LN3C1	190-12	192-9#				
LN3C2	190-14	192-10#				
LN3C3	190-16	192-11#				
LN3C4	190-18	192-12#				
LN3E1	190-24	192-13#				
LN3E2	190-26	192-14#				
LN3G1	190-32	192-15#				
LN3G2	190-34	192-16#				
LN4B1	194-44	195-29#				
LN4B2	194-46	195-30#				
LN4D1	195-2	195-31#				
LN4D2	195-4	195-32#				
LN4D3	195-6	195-33#				
LN4D4	195-8	195-34#				
LN7D1	204-11	204-39#				
LN7D2	204-13	204-40#				
LN7D3	204-15	204-41#				
LN7D4	204-17	205-1#				
LN7F1	204-23	205-2#				
LN7F2	204-25	205-3#				
LN8B1	209-2	209-19#				

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#		

##INIT 0-0#

