


```
$SAVE
$NDLIST
```

```
/*
 * ASTLIB.H
```

```
 * This header describes the public entries in
 * the assembly language assist library, ASTLIB.
 * /
```

```
D$XMITC:      PROCEDURE(C)      BYTE;      EXTERNAL;
DECLARE
END;
```

```
FLAGS:        PROCEDURE
END;          WORD EXTERNAL;
```

```
$ IF EXTENDED_MONITOR
OFB:          PROCEDURE(base, offset)
DECLARE      base      WORD,
             offset    WORD;
             WORD;
$ END;
$ ENDIF
```

```
INCB:         PROCEDURE(byte$ptr)
DECLARE      byte$ptr  POINTER;
END;          EXTERNAL;
```

```
$ IF EXTENDED_MONITOR
SFB:         PROCEDURE(val, base, offset)
DECLARE      val      BYTE,
             base     WORD,
             offset   WORD;
$ END;
$ ENDIF
```

```
$XMITC:      PROCEDURE(C)      BYTE;      EXTERNAL;
DECLARE
END;
```

```
VIDEO_INTR_A: PROCEDURE
END;          EXTERNAL;
```

```
XEC:         PROCEDURE(reg$pp)  POINTER;  EXTERNAL;
DECLARE      reg$pp
END;
```

```
INIT_ITV:    PROCEDURE(ivi, ivp)
DECLARE      ivi      BYTE,
             ivp      POINTER;
             EXTERNAL;
END;
```

```
$RESTORE
```

```
$SUBTITLE('Compiler Controls')
$OPTIMIZE(3)
$NOOVERFLOW
$COMPACT
$ROM
$XREF
$LARGE ( MTR100 EXPORTS MTR_MON;
$ EXPORTS MTR_MON;
$ EXPORTS MTR_SWIM;
$ EXPORTS MTR_DCRT;
$ EXPORTS MTR_DCRD;
$ EXPORTS MTR_SCRT;
$ EXPORTS MTR_SKBD;
$ EXPORTS MTR_TTY_INTR;
$ EXPORTS MTR_TTY_POLL;
$ EXPORTS MTR_IRET)
$LARGE ( VIDEOA EXPORTS MTR_DC1;
$ EXPORTS MTR_DFC;
$ EXPORTS MTR_EDC;
$ EXPORTS MTR_FONT;
$ EXPORTS MTR_MDC;
$ EXPORTS MTR_MDL;
$ EXPORTS MTR_PROMPT;
$ EXPORTS MTR_RDC;
$ EXPORTS MTR_UIES;
$ EXPORTS MTR_XCA)
$LARGE ( FONT EXPORTS MTR_FONT)
$LARGE ( ASTLIB EXPORTS VIDEO_INTR_A)
$RESET(EXTENDED_MONITOR)
```

```

DC DC CR_DSAL LT '0DH' /* Start Low */;
DC DC CR_CSAH LT '0EH' /* Cursor Start High */;
DC DC CR_CSAL LT '0FH' /* Cursor Start Low */;

/*
 * Cursor Start Register Fields
 */
DC DC CSR_CSD LT '00#1111B' /* Cursor Start Display */;
DC DC CSR_OFF LT '01#00000B' /* Disable Cursor Disp */;
DC DC CSR_BLINK16 LT '10#000000B' /* Blink at 1/16 Field */;
DC DC CSR_BLINK LT '11#000000B' /* Blink/Enable Field */;

```

#RESTORE

1\$SAVE
\$NOLIST

DECLARE HEX_DIGIT(*) BYTE EXTERNAL DATA;

ORLF: PROCEDURE
END;

EXTERNAL;

\$IF 1=2
ORW: PROCEDURE(base, offset)
DECLARE base WORD,
offset WORD;
END;
\$ENDIF

WORD EXTERNAL;

\$IF EXTENDED_MONITOR
HOTB: PROCEDURE(char)
DECLARE char BYTE;
END;
\$ENDIF

BYTE EXTERNAL;

\$IF EXTENDED_MONITOR
IHC: PROCEDURE(delim)
DECLARE delim BYTE;
END;
\$ENDIF

BYTE EXTERNAL;

\$IF EXTENDED_MONITOR
IHV: PROCEDURE(delim, val\$p, size)
DECLARE delim BYTE,
val\$p POINTER,
size BYTE;
END;
\$ENDIF

BYTE EXTERNAL;

MOU: PROCEDURE(char)
DECLARE char BYTE;
END;

BYTE EXTERNAL;

\$IF EXTENDED_MONITOR
OHB: PROCEDURE(dat)
DECLARE dat BYTE;
END;
\$ENDIF

EXTERNAL;

\$IF EXTENDED_MONITOR
OHC: PROCEDURE(dat)
DECLARE dat BYTE;
END;
\$ENDIF

EXTERNAL;

\$IF EXTENDED_MONITOR
OHM: PROCEDURE(dat)
DECLARE dat WORD;
END;

EXTERNAL;


```

) EXTERNAL;
DC CRT0_DISPLAY STRUCTURE (
  START WORD, /* CRT-C Display Start Address */
  UPDATE BOOL /* True if Update is required */
) EXTERNAL;
DC ESCP STRUCTURE (
  COMMAND BYTE, /* Current Escape Command */
  FUNCTION WORD, /* Pointer to Escape Processor */
  MODE BYTE, /* Escape Emulation Mode */
  OPER_COUNT BYTE, /* Operand Count */
  OPER_INDEX BYTE, /* Operand Index */
  OPERAND(4) BYTE /* Operands */
) EXTERNAL;
DC H19_MODE STRUCTURE (
  ALTERNATE BOOL, /* Alternate Key-Pad Mode */
  ANSI BOOL, /* ANSI-Mode */
  AUTO_CR BOOL, /* Auto Carriage-Return on LF */
  AUTO_LF BOOL, /* Auto Line-Feed on CR */
  AUTO_REPEAT BOOL, /* Auto-Repeat Keyboard */
  BMD BOOL, /* Black and White Optimization */
  CURSOR BOOL, /* Cursor Program Value */
  CURSOR_ON BOOL, /* Cursor Enabled */
  EXPAND BOOL, /* Expand Keyboard Characters */
  GRAPHIC BOOL, /* Graphic Character Mode */
  INSERT BOOL, /* Insert Character Mode */
  KEY_EN BOOL, /* Key-Board Enabled */
  REVERSE WORD, /* Reverse Video
  /* @0000H for normal video
  /* 0FFFFH for reverse video
  /* Shifted Key-Pad Mode
  /* Status Line Enabled
  /* Key-Board Up/Down Mode
  /* Wrap Line at End-of-Line
  */
  SHIFTED BOOL, /* Shifted Key-Pad Mode */
  STATUS BOOL, /* Status Line Enabled */
  UPDN BOOL, /* Key-Board Up/Down Mode */
  WRAP BOOL /* Wrap Line at End-of-Line */
) EXTERNAL;
DC H19_PAR STRUCTURE (
  CPL BYTE, /* Characters Per Line */
  DSC BYTE, /* Displayed Scan Lines/Char */
  LPS BYTE, /* Lines Per Screen */
  PPOVRAM BYTE, /* Lines Per Screen */
  SLI BYTE, /* Lines Per Screen */
  SPC BYTE, /* Status Line Index */
  SW401 BYTE, /* Scan Line per Character */
  SW402 BYTE, /* H-19 Switch 401 */
  VRAM_SIZE BYTE, /* H-19 Switch 402
  /* @-32KBytes, 1-64KBytes
  */
) EXTERNAL;
DC XMT STRUCTURE (
  BURST BYTE, /* Char. to XMT per VSYNC */
  BURST_COUNT INTEGER, /* Remaining Char. in Burst */
  COUNT WORD, /* Characters to Transmit */
  COL BYTE, /* Horizontal Column to XMT */
  ROW BYTE, /* Vertical Row to XMT */
  COLOR BYTE, /* State of COLOR XMT'd */
  GRAPHIC BOOL, /* State of GRAPHIC XMT'd */
  REVERSE BOOL, /* State of REVERSE XMT'd */
) EXTERNAL;

```

\$RESTORE

\$SAVE
\$NOLIST

/*
*
* I3085.H

I3085.H defines hardware attributes of the
Intel 8085 micro-processor.

*/

DC	F86_OF	LT	'1000#0000#0000B';	/* Overflow	*/
DC	F86_DF	LT	'0100#0000#0000B';	/* Direction	*/
DC	F86_IF	LT	'0010#0000#0000B';	/* Interrupt En	*/
DC	F86_TF	LT	'0001#0000#0000B';	/* Trap	*/
DC	F86_SF	LT	'0000#1000#0000B';	/* Sign	*/
DC	F86_ZF	LT	'0000#0100#0000B';	/* Zero	*/
DC	F86_AF	LT	'0000#0001#0000B';	/* Aux. Carry	*/
DC	F86_PF	LT	'0000#0000#0100B';	/* Parity	*/
DC	F86_CF	LT	'0000#0000#0001B';	/* Carry	*/

\$RESTORE

SSAVE
\$NOLIST

/*
INIT.H

INIT.H defines the global initialization
routines in INIT.PLM
#

*/

! IF EXTENDED_MONITOR
INIT_8259_85: PROCEDURE EXTERNAL;
END;
! ENDIF

INIT_8259_89: PROCEDURE EXTERNAL;
END;

\$RESTORE

\$SAVE
\$NDLIST

\$ IF EXTENDED_MONITOR

I2661: PROCEDURE(base) EXTERNAL;
 DECLARE base WORD;
END;

RCHAR: PROCEDURE(base) BYTE EXTERNAL;
 DECLARE base WORD;
END;

TCHAR: PROCEDURE(base) BOOL EXTERNAL;
 DECLARE base WORD;
END;

WCHAR: PROCEDURE(base, char) EXTERNAL;
 DECLARE base WORD,
 char BYTE;
END;

\$ ENDIF
\$RESTORE

```

DC      ICM4_AE01      LT      '0000$0010B'      /* Auto. End Of Int.      /*;
DC      ICM4_86      LT      '0000$0001B'      /* MCS-86 Operation      /*;

/*
*/
*/      Output Control Words

DC      OCM1_M7      LT      '1000$0000B'      /* Mask Level 7      /*;
DC      OCM1_M6      LT      '0100$0000B'      /* Mask Level 6      /*;
DC      OCM1_M5      LT      '0010$0000B'      /* Mask Level 5      /*;
DC      OCM1_M4      LT      '0001$0000B'      /* Mask Level 4      /*;
DC      OCM1_M3      LT      '0000$1000B'      /* Mask Level 3      /*;
DC      OCM1_M2      LT      '0000$0100B'      /* Mask Level 2      /*;
DC      OCM1_M1      LT      '0000$0010B'      /* Mask Level 1      /*;
DC      OCM1_M0      LT      '0000$0001B'      /* Mask Level 0      /*;
DC      OCM1_ALL     LT      '1111$1111B'      /* Mask ALL Levels      /*;

DC      OCM2_ROT     LT      '1000$0000B'      /* Rotate      /*;
DC      OCM2_SEOI    LT      '0100$0000B'      /* Specific EOI      /*;
DC      OCM2_EOI     LT      '0010$0000B'      /* End Of Interrupt      /*;
DC      OCM2_OCM2    LT      '0000$0000B'      /* OCM2 Identifier      /*;
DC      OCM2_L2      LT      '0000$0100B'      /* SEOI Interrupt Level /*;
DC      OCM2_L1      LT      '0000$0010B'      /* SEOI Interrupt Level /*;
DC      OCM2_L0      LT      '0000$0001B'      /* SEOI Interrupt Level /*;

DC      OCM3_ESMM    LT      '0100$0000B'      /* Enable Special Mask /*;
DC      OCM3_SMM     LT      '0010$0000B'      /* Special Mask Mode   /*;
DC      OCM3_OCM3    LT      '0000$1000B'      /* OCM3 Identifier     /*;
DC      OCM3_POLL    LT      '0000$0100B'      /* Enable Polling      /*;
DC      OCM3_SRIS    LT      '0000$0010B'      /* */;

```

*RESTORE

!SAVE
\$NOLIST

/*
* KEYDEF.H

This header file defines the bits and offsets pertinent to the key-board processor.

*/

/*
* Port Definitions

DC KEY_CMND LT 'IO_KEYBRD+1' /* Command **/
DC KEY_DATA LT 'IO_KEYBRD+0' /* Data **/
DC KEY_STAT LT 'IO_KEYBRD+1' /* Status **/
*/

/*
* Status Bit Definitions

DC KS_CXRDY LT '001B' /* Character is ready **/
DC KG_RXRDY LT '010B' /* Receiver Ready **/
*/

/*
* Command Definitions

DC KC_ARF LT '02H' /* Auto-Repeat OFF **/
DC KC_ARO LT '01H' /* Auto-Repeat ON **/
DC KC_BEP LT '07H' /* Beep **/
DC KC_DI LT '0DH' /* Disable Interrupts **/
DC KC_EI LT '0CH' /* Enable Interrupts **/
DC KC_KCF LT '04H' /* Key Click OFF **/
DC KC_KCO LT '03H' /* Key Click ON **/
DC KC_CFI LT '05H' /* Clear FIFO **/
DC KC_CLK LT '06H' /* Click **/
DC KC_KBD LT '09H' /* Key-Board DISABLE **/
DC KC_KBE LT '08H' /* Key-Board ENABLE **/
DC KC_MNS LT '0BH' /* Normal Scan Mode **/
DC KC_MUD LT '0AH' /* Key Up/Down Mode **/
DC KC_RES LT '00H' /* Reset **/
*/

/*
* Key Code Definitions

DC KY_ENTER LT '08DH' /* Enter **/
DC KY_HELP LT '075H' /* HELP **/
DC KY_F0 LT '076H' /* F0 **/
DC KY_F12 LT '0A2H' /* F12 **/
DC KY_ID_CHAR LT '0A3H' /* Insert/Delete Char. **/
DC KY_ID_LINE LT '0A4H' /* Insert/Delete Line **/
DC KY_UP LT '0A5H' /* Arrow Up **/
DC KY_DOWN LT '0A6H' /* Arrow Down **/
DC KY_RIGHT LT '0A7H' /* Arrow Right **/
DC KY_LEFT LT '0A8H' /* Arrow Left **/
DC KY_HOME LT '0A9H' /* Home **/
*/

\$SAVE
\$NOLIST

```
/*  
 * LEXCAL.H  
 * LEXCAL.H defines a number of standard lexical  
 * headers.  
 */
```

```
DECLARE DC LITERALLY 'DECLARE';  
DECLARE LT LITERALLY 'LITERALLY';
```

```
DC BOOL LT 'BYTE' /* Boolean Variables */;
```

```
DC FALSE LT '000H' /* Boolean False */;  
DC TRUE LT '0FFH' /* Boolean True */;
```

\$RESTORE

\$SAVE
\$NDLIST

/*
* MTR100.H
*
* MTR100.H is the file defining the globals to
* be found in MTR100.A86.
*
*/

MTR_SORT: PROCEDURE EXTERNAL;
END;

MTR_SKED: PROCEDURE EXTERNAL;
END;

MTR_TTY_INTR: PROCEDURE EXTERNAL;
END;

MTR_TTY_POLL: PROCEDURE BOOL EXTERNAL;
END;

MTR_IRET: PROCEDURE EXTERNAL;
END;

\$ IF EXTENDED_MONITOR
R3085: PROCEDURE EXTERNAL;
END;
\$ ENDIF

SOS: PROCEDURE EXTERNAL;
END;

\$RESTORE

\$SAVE
\$NOLIST

/*
* SSWDEF.H

* SSWDEF.H defines the H-17 compatible software
* switch settings.
*/

DC

SM_401_BLOCK_CURSOR LT '001H' /* Block Cursor */
SM_401_KEY_CLICK LT '002H' /* Key Click */
SM_401_WRAP_LINE LT '004H' /* Wrap Line at Right-Hand Margin*/
SM_401_AUTO_LF LT '008H' /* Auto Line-Feed on Carriage-Return*/
SM_401_AUTO_CR LT '010H' /* Auto Carriage-Return on Line-Feed*/
SM_401_ANSI LT '020H' /* ANSI Escape Processing Mode */
SM_401_SHIFTED LT '040H' /* Key-Pad shifted Mode */
SM_401_50HZ LT '080H' /* 50-Hertz CRT Refresh */

DC

SM_402_BAUD_RATE LT '00FH' /* Baud Rate */
SM_402_PARITY_ENABLE LT '010H' /* Parity Enable */
SM_402_ODD_PARITY LT '020H' /* Odd Parity */
SM_402_STICK_PARITY LT '040H' /* Stick Parity */
SM_402_FULL_DUPLEX LT '080H' /* Full Duplex */

\$RESTORE


```

$SAVE
$NCLIST

/*
*      SURLIB.H
*
*      This file describes the global Subroutine Library
*      entry points.
*/

$ IF EXTENDED_MONITOR
IHA:      PROCEDURE(delim,addr$p)          BYTE EXTERNAL;
DECLARE   delim      BYTE,
addr$p   POINTER;
END;
$ ENDIF

$ IF 1=2
IHA:      PROCEDURE(base$p,offset$p)      EXTERNAL;
DECLARE   base$p     POINTER,
offset$p  POINTER;
END;
$ ENDIF

$ IF EXTENDED_MONITOR
IRI:      PROCEDURE
END;
$ ENDIF          BYTE EXTERNAL;

$ IF EXTENDED_MONITOR
OHA:      PROCEDURE(addr$p)              EXTERNAL;
DECLARE   addr$p     POINTER;
END;
$ ENDIF

$ IF EXTENDED_MONITOR
RNC:      PROCEDURE(delim)              BYTE EXTERNAL;
DECLARE   delim      BYTE;
END;
$ ENDIF

$ IF EXTENDED_MONITOR
RUBOUT:   PROCEDURE
END;
$ ENDIF          EXTERNAL;

$RESTORE

```

```
XMTSC:
DECLARE
PROCEDURE (row, col, count)
row
col
count
BYTE,
BYTE,
WORDS
EXTERNAL:
END;
$RESTORE
```

\$RESTORE

;; E2661 - EPCI 2661

;; The following definitions are for the Signetics
;; 2661 EPCI (Enhanced Programmable Communications
;; Interface).
;;

```
SR      RECORD  DSR:1, DCD:1, FE_SYN:1, OVRN:1, PE_DLE:1, TXEMT_DSCHG:1, RXRDY:1, TXRDY:1
EP_RHR  EQU     0      ; Receiver Holding Register
EP_THR  EQU     0      ; Transmitter Holding Register
EP_STA  EQU     1      ; Status Register
EP_SYN1 EQU     1      ; SYN1
EP_SYN2 EQU     1      ; SYN2
EP_DLE  EQU     1      ; DLE
EP_MR1  EQU     2      ; Mode Register 1
EP_MR2  EQU     2      ; Mode Register 2
EP_CR   EQU     3      ; Command Register
CONSOL  EQU     0ESH   ; Console 2661
```

```

EXTRN  CR-LF:NEAR          ; PL/M-86 CR-LF
EXTRN  INIT:NEAR          ; PL/M-86 Initialize Hardware
EXTRN  WRITEC:NEAR       ; PL/M-86 Write Character
ENDS
)FI

```

```

%IF(%NES(DATA,%SEGMENT_LABEL)) THEN (
DATA
  SEGMENT WORD PUBLIC 'DATA'
  EXTRN  HORZ_CHAR:BYTE   ; Column Index
  EXTRN  VERT_LINE:BYTE  ; Row Index
  EXTRN  REGP:DWORD      ; Pointer to Saved Processor Registers
  EXTRN  RESETF:BYTE     ; Hardware Reset Flag

  EXTRN  DCI:DWORD       ; Display Character Initialization
  EXTRN  DFC:DWORD       ; Display Font Character
  EXTRN  D_XMTC:DWORD    ; Dumb Terminal Transmit Character
  EXTRN  EDC:DWORD       ; Erase Display Character
  EXTRN  EMEC:DWORD      ; Extend-Mode Escape Character
  EXTRN  FONT:DWORD      ; Pointer to Font Table
  EXTRN  MDC:DWORD       ; Move Display Character
  EXTRN  MDL:DWORD       ; Move Display Line
  EXTRN  PROMPT:DWORD    ; Display Monitor Prompt
  EXTRN  RDC:DWORD       ; Read Display Character
  EXTRN  S_XMTC:DWORD    ; Smart Terminal Transmit Character
  EXTRN  UIES:DWORD      ; Un-Implemented Escape Sequence
  EXTRN  XCA:DWORD       ; Transmit Character Attributes

```

```

  EXTRN  BOOT_PAR:BOOT_PAR_STRUC
  EXTRN  COLOR:COLOR_STRUC
  EXTRN  CRTC_CURSOR:CRTC_STRUC
  EXTRN  CRTC_DISPLAY:CRTC_STRUC
  EXTRN  ESCP:ESCP_STRUC
  EXTRN  H19_MODE:H19_MODE_STRUC
  EXTRN  H19_PAR:H19_PAR_STRUC
  EXTRN  XMT:XMT_STRUC
ENDS
DATA
)FI

```

```

%IF(%NES(VIDEQA,%SEGMENT_LABEL)) THEN (
VIDEQA
  SEGMENT BYTE PUBLIC 'CODE'
VIDEQA
)FI
ENDS

```

```

GROUP  GROUP  MTR100, CODE, ASTLIB, VIDEQA

```

```

#RESTORE

```

```
;$EJECT
; Interrupt Usage
```

```
;
; This common deck defines the MTR-100 Interrupt
; usage. Note that the interrupt vector for int-
; erupt 255 is used to store a global data seg-
; ment value. This is somewhat undesirable to
; reserve such a location, however, since all
; other values are indexed off of it, it seems
; alright. The offset must be zero so that any
; spurious interrupts will jump to the first addr-
; ess of the data segment. If these interrupts
; are to be handled, the data segment vector may
; be patched.
```

```
INIVEC SEGMENT PARA AT PAR_INT
```

```
ITV_DBZ DD 1 DUP(?) ; Divide by zero Interrupt
ITV_SST DD 1 DUP(?) ; Single Step
ITV_NMI DD 1 DUP(?) ; Non-Maskable Interrupt
ITV_OBI DD 1 DUP(?) ; One Byte Instruction
ITV_I0V DD 1 DUP(?) ; Interrupt On Overflow
```

```
ITV_5 DD 1 DUP(?) ; Interrupt 5
DD (32-6) DUP(?) ; Reserved Interrupts
ITV_32 DD 1 DUP(?) ; Interrupt 32
DD (255-33) DUP(?) ; Allocate space for available user interrupts
ITV_255 DD 1 DUP(?) ; Interrupt 255 is used for MTR-100 DS
```

```
MTR_DS_0 EQU WORD PTR ITV_255+0 ; MTR-100 Data Segment Offset
MTR_DS_S EQU WORD PTR ITV_255+2 ; (must be 0)
; MTR-100 Data Segment
```

```
INIVEC ENDS
```

\$SAVE
\$NOGEN

Global Structure Definitions

```
;;  
;  
BOOT__PAR_STRUC STRUC  
  INDEX DB ? ; Device Index  
  PORT DB ? ; Base Port Address  
  STRING DB 80 DUP(?) ; Parameter String Passed  
  UNIT DB ? ; Unit of Device Booted  
ENDS  
BOOT__PAR_STRUC
```

```
COLOR_STRUC STRUC  
  FORE DB ? ; Foreground Color  
  BACK DB ? ; Background Color  
  MKL DB ? ; SHL(Back,3) OR Fore  
  CLEAR DB ? ; Color to Clear  
  PRINTED DB ? ; Color to Fully Paint  
  P_FONT DB ? ; Color to set to Font Pattern  
  COMP_FONT DB ? ; Color to set to Complement of Font Pattern  
  COLOR_STRUC ENDS
```

```
CRTC_STRUC STRUC  
  START DW ? ; CRTC-***** Start Address  
  UPDATE DB ? ; True if update is required  
CRTC_STRUC ENDS
```

```
ESCP_STRUC STRUC  
  COMMAND DB ? ; Current Escape Command  
  FUNCTION DW ? ; Pointer to Escape Processor  
  MODE DB ? ; Escape Emulation Mode  
  OPER_COUNT DB ? ; Operand Count  
  OPER_INDEX DB ? ; Operand Index  
  OPERAND DB ? ; Operands  
  ESCP_STRUC ENDS
```

```
HI?_MODE_STRUC STRUC  
  ALTERNATE DB ? ; Alternate Key-pad Mode  
  ANSI DB ? ; ANSI Mode  
  AUTO_CR DB ? ; Auto Carriage-Return on Line-Feed  
  AUTO_LF DB ? ; Auto Line-Feed on Carriage-Return  
  AUTO_REPEAT DB ? ; Auto-Repeat Keyboard  
  BMO DB ? ; Black and White Optimization  
  CURSOR DB ? ; Programmed Cursor Value  
  CURSOR_ON DB ? ; Cursor Enabled  
  EXPAND DB ? ; Expand Key-Board Characters  
  GRAPHIC DB ? ; Graphic Character Mode  
  INSERT DB ? ; Insert Character Mode  
  KEY_EN DB ? ; Key-Board Enable  
  REVERSE DB ? ; Key-Board Enable  
  SHIFTED DB ? ; Reverse Video  
  STATUS DB ? ; Shifted Key-Pad Mode  
  UPDN DB ? ; Status-Line Enabled  
  WRAP DB ? ; Key-Board Up/Down Mode  
  WRAP DB ? ; Wrap at End-of-Line  
ENDS  
HI?_MODE_STRUC
```

```
HI?_PAR_STRUC STRUC  
  CPL DB ? ; Characters Per Line  
  DSC DB ? ; Displayed Scan Lines per Character  
  LPS DB ? ; Lines Per Screen  
  PVRAM DB ? ; Planes of Video RAM  
  SLI DB ? ; Status Line Index  
  SPC DB ? ; Scan Lines per Character
```

SERIES-III 8086/3087/8088 MACRO ASSEMBLER V1.0 ASSEMBLY OF MODULE GLOBAL
OBJECT MODULE PLACED IN :F2:GLOBAL.OBJ
INVOCATION LINE CONTROLS: PRINT(:TO:) XREF ERRORPRINT(:TO:)

LOC OBJ LINE SOURCE

LOC	OBJ	LINE	SOURCE
		14	%*DEFINE(SEGMENT_LABEL)(DATA)
		15	\$ INCLUDE (:F2:EXTERN.COM)
		16	\$SAVE
		17	\$NOGEN
		18	
		19	%*DEFINE(EXTENDED_MONITOR)(0) %' EXTENDED_MODE is FALSE
00FF		20	PLM_TRUE EQU OFFH ; PL/M-86 Boolean TRUE
		21	
		22	; Structure Definitions
		23	
		24	\$ INCLUDE (:F2:STRDEF.COM')
		25	\$SAVE
		26	\$NOGEN
		27	
		28	
		29	; Global Structure Definitions
		30	
		31	BOOT_PAR_STRUC STRUC
0000		32	INDEX DB ? ; Device Index
0001		33	PORT DB ? ; Base Port Address
0002		34	STRING DB @0 DUP(?) ; Parameter String Passed
0052		35	UNIT DB ? ; Unit of Device Booted
		36	BOOT_PAR_STRUC ENDS
		37	
		38	COLOR_STRUC STRUC
0000		39	FGRE DB ? ; Foreground Color
0001		40	BACK DB ? ; Background Color
0002		41	MSK DB ? ; SHL(Back,3) OR Fore
0003		42	CLEAR DB ? ; Color to Clear
0004		43	PAINTED DB ? ; Color to Fully Paint
0005		44	PFONT DB ? ; Color to set to Font Pattern
0006		45	COMP_FONT DB ? ; Color to set to Complement of Font Pattern
		46	COLOR_STRUC ENDS
		47	
		48	CRTC_STRUC STRUC
0000		49	START DW ? ; CRT-C Start Address
0002		50	UPDATE DB ? ; True if update is required
		51	CRTC_STRUC ENDS
		52	
		53	ESCP_STRUC STRUC
0000		54	COMMAND DB ? ; Current Escape Command
0001		55	FUNCTION DW ? ; Pointer to Escape Processor
0003		56	MODE DB ? ; Escape Emulation Mode
0004		57	OPER_COUNT DB ? ; Operand Count
0005		58	OPER_INDEX DB ? ; Operand Index
0006		59	OPERAND DB ? ; Operands
		60	ESCP_STRUC ENDS
		61	
		62	H19_MODE_STRUC STRUC
0000		63	ALTERNATE DB ? ; Alternate Key-pad Mode
0001		64	ANSI DB ? ; ANSI Mode
0002		65	AUTO_CR DB ? ; Auto Carriage-Return on Line-Feed
0003		66	AUTO_LF DB ? ; Auto Line-Feed on Carriage-Return
0004		67	AUTO_REPEAT DB ? ; Auto-Repeat Keyboard

```

LOC OBJ          LINE          SOURCE
=====
=1 123           %IF (ZNES(MTR100, %SEGMENT_LABEL)) THEN (
=1 124           ; INTVEC      SEGMENT PUBLIC 'DATA'
=1           EXTRN      ITV_STG:DWORD
=1           EXTRN      ITV_OBI:DWORD
=1           EXTRN      DS_PTR_O:WORD
=1           EXTRN      DS_PTR_S:WORD
=1           ;
=1           ; INTVEC
=1           ; MONENT
=1           SEGMENT PARA PUBLIC
=1           EXTRN      MTR_RES: FAR
=1           EXTRN      MTR_MON: FAR
=1           EXTRN      MTR_SWIM: FAR
=1           EXTRN      MTR_DCRT: FAR
=1           EXTRN      MTR_SCRT: FAR
=1           EXTRN      MTR_DKBD: FAR
=1           EXTRN      MTR_SKBD: FAR
=1           ENDS
=1           MONENT
=1           MTR100
=1           SEGMENT PARA PUBLIC 'CODE'
=1           %IF (XEXTENDED_MONITOR) THEN(
=1           EXTRN R3085: NEAR
=1           )FI
=1           )FI
=1           SDS: NEAR      ; Set Data Segment
=1           ENDS
=1           MTR100
=1           )FI
=1 145           %IF (ZNES(MTR101, %SEGMENT_LABEL)) THEN (
=1 146           ; CODE      SEGMENT BYTE PUBLIC 'CODE'
=1 147           EXTRN      MTR101: NEAR
=1           EXTRN      VIDEO_INTR: NEAR
=1           EXTRN      D_CRT: NEAR
=1           EXTRN      D_KBD: NEAR
=1           EXTRN      S_CRT: NEAR
=1           EXTRN      S_KBD: NEAR
=1           EXTRN      TXMT: NEAR
=1           EXTRN      TTY_INTR: NEAR
=1           EXTRN      TTY_POLL: NEAR
=1           EXTRN      CRLF: NEAR
=1           EXTRN      INIT: NEAR
=1           EXTRN      WRITEC: NEAR
=1           ENDS
=1           CRLF: NEAR      ; PL/M-86 CR-LF
=1           INIT: NEAR     ; PL/M-86 Initialize Hardware
=1           WRITEC: NEAR  ; PL/M-86 Write Character
=1           ENDS
=1 164           %IF (ZNES(DATA, %SEGMENT_LABEL)) THEN (
=1 165           ; DATA      SEGMENT PUBLIC 'DATA'
=1 166           EXTRN      HORZ_CHAR: BYTE
=1           EXTRN      VERT_LINE: BYTE
=1           EXTRN      REGP: DWORD
=1           EXTRN      RESETF: BYTE
=1           EXTRN      DCI: DWORD
=1           EXTRN      DFC: DWORD
=1           EXTRN      D_XMTC: DWORD
=1           ;
=1           ; Column Index
=1           ; Row Index
=1           ; Pointer to Saved Processor Registers
=1           ; Hardware Reset Flag
=1           ; Display Character Initialization
=1           ; Display Font Character
=1           ; Dumb Terminal Transmit Character

```

LOC	OBJ	LINE	SOURCE
		181	%DEFINE(DPB(name, count))(%name DB %count DUP(?)
		182	PUBLIC %name)
		183	%DEFINE(DPD(name))(%name DD 1 DUP(?)
		184	PUBLIC %name)
		185	%DEFINE(DPS(name, sname))(%name DB SIZE(%sname) DUP(?)
		186	PUBLIC %name)
		187	%DEFINE(DPW(name, count))(%name DW %count DUP(?)
		188	PUBLIC %name)
		189	
		190	
		191	\$EJECT

```

LOC  OBJ          LINE      SOURCE
291          not to move.
292          ;;;
293          ;;;
294          ;;;
295          ;;;
296          ;;;
297          ;;;
298          ;;;
299          ;;;
300          ;;;
301          ;;;
302          ;;;
303          ;;;
304          ;;;
305          ;;;
306          ;;;
307          ;;;
308          ;;;
309          ;;;
310          ;;;
311          ;;;
312          ;;;
313          ;;;
314          ;;;
315          ;;;
316          ;;;
317          ;;;
318          ;;;
319          ;;;
320          ;;;
321          ;;;
322          ;;;
323          ;;;
324          ;;;
325          ;;;
326          ;;;
327          ;;;
328          ;;;
329          ;;;
330          ;;;
331          ;;;
332          ;;;
333          ;;;
334          ;;;
335          ;;;
336          ;;;
337          ;;;
338          ;;;
339          ;;;
340          ;;;
341          ;;;
342          ;;;
343          ;;;
344          ;;;
345          ;;;
346          ;;;

;
; Miscellaneous Hardware Latch Values
;
; Base Internal Interrupt Level
; Base S-100 Interrupt Level
; Processor Swap Port
; Pointer to Saved Processor Registers
; Reset Flag
;
; Terminal Emulator Variables
;
; Color Structure
; Cursor Start Address Structure
; Display Start Address Structure
; Escape Processing Structure
; H-19 Mode Definitions
; H-19 Parameters
; Transmit Character Structure

DATA      ENDS
END

```

NAME	TYPE	VALUE	ATTRIBUTES, XREFS
H19_MODE_STRUC.	V BYTE	02B2H	DATA PUBLIC 332# 333
H19_PAR_STRUC.	V BYTE	02C4H	DATA PUBLIC 335# 336
H19_PAR_STRUC.	STRUC		SIZE=0009H #FIELDS=9
HORZ_CHAR.	V BYTE	0291H	DATA PUBLIC 270# 271
INDEX.	V BYTE	0000H	S FIELD 32#
INIT.	L NEAR	0000H	EXTRN 160#
INSERT.	V BYTE	000AH	S FIELD 73#
ITV_OBI.	V DWORD	0000H	EXTRN 127#
ITV_SST.	V DWORD	0000H	EXTRN 126#
KEY_EN.	V BYTE	000BH	S FIELD 74#
KMAP.	V BYTE	0091H	DATA PUBLIC 264# 265
LPS.	V DWORD	0002H	S FIELD 85#
MDC.	V DWORD	0073H	DATA PUBLIC 237# 238
MDL.	V DWORD	0077H	DATA PUBLIC 240# 241
MODE.	V BYTE	0003H	S FIELD 56#
MOMENT.	SEGMENT		SIZE=0000H PARA PUBLIC 131# 139
MSK.	V BYTE	0002H	S FIELD 41#
MTR_DCRT.	L FAR	0000H	EXTRN 135#
MTR_DKBD.	L FAR	0000H	EXTRN 137#
MTR_FONT.	V BYTE	0000H	EXTRN 119#
MTR_MON.	L FAR	0000H	EXTRN 133#
MTR_RES.	L FAR	0000H	EXTRN 132#
MTR_SCR.	L FAR	0000H	EXTRN 136#
MTR_SKBD.	L FAR	0000H	EXTRN 138#
MTR_SWIM.	L FAR	0000H	EXTRN 134#
MTR100.	SEGMENT		SIZE=0000H PARA PUBLIC 'CODE' 140# 143 175
MTR101.	L NEAR	0000H	EXTRN 149#
OPER_COUNT.	V BYTE	0004H	S FIELD 57#
OPER_INDEX.	V BYTE	0005H	S FIELD 58#
OPERAND.	V BYTE	0006H	S FIELD 59#
PAINTED.	V BYTE	0004H	S FIELD 43#
POINT.	V BYTE	0005H	S FIELD 44#
PLM_TRUE.	NUMBER	00FFH	20#
PORT.	V BYTE	0001H	S FIELD 33#
POVRAM.	V DWORD	0003H	S FIELD 86#
PROMPT.	V DWORD	007BH	DATA PUBLIC 243# 244
PSP.	V BYTE	0295H	DATA PUBLIC 308# 309
RDC.	V DWORD	007FH	DATA PUBLIC 246# 247
REGP.	V DWORD	0296H	DATA PUBLIC 311# 312
RESETF.	V BYTE	029AH	DATA PUBLIC 314# 315
REVERSE.	V WORD	000CH	S FIELD 75#
ROM.	V BYTE	0006H	S FIELD 99#
S_CRT.	L NEAR	0000H	EXTRN 153#
S_KBD.	L NEAR	0000H	EXTRN 154#
S_XMTC.	V DWORD	0083H	DATA PUBLIC 249# 250
SXS.	L NEAR	0000H	EXTRN 142#
SHIFTED.	V BYTE	000EH	S FIELD 76#
SLI.	V BYTE	0004H	S FIELD 87#
SPC.	V BYTE	0005H	S FIELD 88#
START.	V WORD	0000H	S FIELD 49#
STATUS.	V BYTE	000FH	S FIELD 77#
STRING.	V BYTE	0002H	S FIELD 34#
SW401.	V BYTE	0006H	S FIELD 89#
SW402.	V BYTE	0007H	S FIELD 90#

SERIES-111 8086/8087/80386 MACRO ASSEMBLER V1.0 ASSEMBLY OF MODULE MAIN
OBJECT MODULE PLACED IN : F2:MTR100.OBJ
INVOCATION LINE CONTROLS: PRINT(:TD:) XREF ERRPRPRINT(:TD:)

LOC OBJ LINE SOURCE

```

LOC   OBJ          LINE   SOURCE
-----
00FB  =1          27 +1  $ INCLUDE (:F2:ISDEF.COM)
00FB  =1          28      ;;          185_DEF - Intel 8085/8080 Instruction Definitions
00FB  =1          29      ;
00FB  =1          30      ;          addr      - 16-bit Address
00FB  =1          31      ;          data      - 8-bit Data
00FB  =1          32      ;          port      - 8-bit Port Address
00FB  =1          33      ;
00FB  =1          34      ;
00FB  =1          35      185_DI      EQU      3630      ; DI      - Disable Interrupts
00FB  =1          36      185_EI      EQU      3730      ; EI      - Enable Interrupts
00FB  =1          37      185_JMP      EQU      3030      ; JMP     addr    - Jump to Address
00FB  =1          38      185_LHLD     EQU      0520      ; LHLD   addr    - Load HL Direct
00FB  =1          39      185_MVIA     EQU      0760      ; MVI    A,data  - Set A to data
00FB  =1          40      185_OUT      EQU      3230      ; OUT    port    - Output A to port
00FB  =1          41      185_POP_AF   EQU      3610      ; POP    PSM     - Pop Processor Status Word
00FB  =1          42      185_POP_BC   EQU      3010      ; POP    BC      - Pop BC Register Pair
00FB  =1          43      185_POP_DE   EQU      3210      ; POP    DE      - Pop DE Register Pair
00FB  =1          44      185_POP_HL   EQU      3410      ; POP    HL      - Pop HL Register Pair
00FB  =1          45      185_PUSH_AF  EQU      3650      ; PUSH   PSM     - Push Processor Status Word
00FB  =1          46      185_RET      EQU      3110      ; RET     - Return from Procedure
00FB  =1          47      185_SHLD     EQU      0420      ; SHLD   addr    - Store HL Direct
00FB  =1          48      185_SPHL     EQU      3710      ; SPHL
00FB  =1          49 +1  $ INCLUDE (:F2:IODEF.COM)
00FB  =1          50      ;;          IODEF.COM
00FB  =1          51      ;
00FB  =1          52      ;          'IODEF.COM' contains the few I/O constants required
00FB  =1          53      ;          by assembly language routines.
00FB  =1          54      ;
00FB  =1          55      ;
00FB  =1          56      10_DIP      EQU      00FH      ; DIP-Switch Port
00FB  =1          57      10_DIAG     EQU      00F6H     ; Diagnostic Board Switch
00FB  =1          58      10_SWAP     EQU      00FEH     ; SWAP Port
00FB  =1          59      ;
00FB  =1          60      Misc.
00FB  =1          61      ;
00FB  =1          62      DIP_AUTO_ROOT EQU      00001000B ; Auto-Boot Dip Switch
00FB  =1          63 +1  $ INCLUDE (:F2:PARDEF.COM)
00FB  =1          64 +1  $EJECT

```

```
LOC ORU          LINE      SOURCE
=====
0000 (1         =1 107   ;;
      )         =1 108   ; Interrupt Usage
      )         =1 109   ;
      )         =1 110   ; This common deck defines the MTR-100 Interrupt
      )         =1 111   ; usage. Note that the interrupt vector for int-
      )         =1 112   ; errupt 255 is used to store a global data seg-
      )         =1 113   ; ment value. This is somewhat undesirable to
      )         =1 114   ; reserve such a location, however, since all
      )         =1 115   ; other values are indexed off of it, it seems
      )         =1 116   ; alright. The offset must be zero so that any
      )         =1 117   ; spurious interrupts will jump to the first add-
      )         =1 118   ; ress of the data segment. If these interrupts
      )         =1 119   ; are to be handled, the data segment vector may
      )         =1 120   ; be patched.
      )         =1 121   ;
      )         =1 122   ;
      )         =1 123   ;
      )         =1 124   ;
0004 (1         =1 125   INTVEC SEGMENT PARA AT PAR_INT
      )         =1 126   ; ITV_SST DD 1 DUP(?) ; Single Step
      )         =1 127   ; ITV_NMI DD 1 DUP(?) ; Non-Maskable Interrupt
      )         =1 128   ; ITV_OBI DD 1 DUP(?) ; One Byte Instruction
      )         =1 129   ; ITV_IOV DD 1 DUP(?) ; Interrupt On Overflow
      )         =1 130   ; ITV_S DD 1 DUP(?) ; Interrupt S
      )         =1 131   ; ITV_32 DD (32-6) DUP(?) ; Reserved Interrupts
      )         =1 132   ; ITV_32 DD 1 DUP(?) ; Interrupt 32
      )         =1 133   ; ITV_255 DD (255-33) DUP(?) ; Allocate space for available user interrupts
03FC (1         =1 134   ; ITV_255 DD 1 DUP(?) ; Interrupt 255 is used for MTR-100 DS
      )         =1 135   ;
      )         =1 136   ; MTR_DS_0 EQU WORD PTR ITV_255+0 ; MTR-100 Data Segment Offset
      )         =1 137   ; MTR_DS_S EQU WORD PTR ITV_255+2 ; (must be 0)
03FE           =1 138   ; INTVEC ENDS PUBLIC ITV_SST ; Single Step
      )         =1 139   ;
      )         =1 140   ;
      )         =1 141   ;
```



```

LOC  OBJ  LINE  SOURCE
-----
00000  0000  150  %*DEFINE(SEGMENT_LABEL)(MTR100)
00001  0001  151  +1 $INCLUDE(:F2:EXTERN.COM)
00002  0002  152  +1 $SAVE
00003  0003  153  +1 $NOGEN
00004  0004  154  %*DEFINE(EXTENDED_MONITOR)(0) %' EXTENDED_MODE is FALSE
00005  0005  155  %*DEFINE(EXTENDED_MONITOR)(0) %' EXTENDED_MODE is FALSE
00006  0006  156  PLM_TRUE EQU 0FFH ; PL/M-86 Boolean TRUE
00007  0007  157  ; Structure Definitions
00008  0008  158  $ INCLUDE (:F2:STRDEF.COM')
00009  0009  159  $SAVE
00010  0010  160  $NOGEN
00011  0011  161  ; Global Structure Definitions
00012  0012  162  ;
00013  0013  163  ;
00014  0014  164  ;
00015  0015  165  ;
00016  0016  166  ;
00017  0017  167  ;
00018  0018  168  ;
00019  0019  169  ;
00020  0020  170  ;
00021  0021  171  ;
00022  0022  172  ;
00023  0023  173  ;
00024  0024  174  ;
00025  0025  175  ;
00026  0026  176  ;
00027  0027  177  ;
00028  0028  178  ;
00029  0029  179  ;
00030  0030  180  ;
00031  0031  181  ;
00032  0032  182  ;
00033  0033  183  ;
00034  0034  184  ;
00035  0035  185  ;
00036  0036  186  ;
00037  0037  187  ;
00038  0038  188  ;
00039  0039  189  ;
00040  0040  190  ;
00041  0041  191  ;
00042  0042  192  ;
00043  0043  193  ;
00044  0044  194  ;
00045  0045  195  ;
00046  0046  196  ;
00047  0047  197  ;
00048  0048  198  ;
00049  0049  199  ;
00050  0050  200  ;
00051  0051  201  ;
00052  0052  202  ;
00053  0053  203  ;
00054  0054  204  ;

```



```

LOC OBJ          LINE  SOURCE
326             ;;;      Initial 8085 Boot Code
327             ;
328             ; This code is initially invoked after reset. Since
329             ; the first processor to run is the 8085, this code
330             ; switches to the 8086 processor.
331             ;
332             ; Initially, the 8K-byte MTR-100 ROM is replicated
333             ; throughout the memory space. Since the 8085 resets
334             ; location zero, this code must be the first in the
335             ; ROM. It promptly switches to the 8086, which it is
336             ; assumed will turn off the ROM mapping. The 8085 is
337             ; left with the instruction pointer at 0.
338             ;
339             ;
340             ;
341             ;
342             ;
343             ;
344             ;
345             ;
346             ;
347             ;
348 +1           $ EJECT

-----
0000 3E          SEGMENT BYTE AT PAR_885 ; Start the segment at base of ROM
0001 80          DB 185_MVIA,80H ; MVI A,80H
0002 03          DB 185_JMP ; JMP 0FFFEH
0003 FEF8       DW 0FFFEH
-----

```

LOC	OBJ	LINE	SOURCE
		373	;;; MTR_ENT - Monitor Entry Points
		374	;
		375	;
		376	These vectors are provided as alternate entry points
		377	into the monitor routines.
		378	;
		379	MOMENT SEGMENT PARA PUBLIC
		380	
		381	ASSUME NOTHING
		382	
0000		383	MTR_RES LABEL FAR
		384	PUBLIC MTR_RES
0000	EA0000	385	JMP FAR PTR MTR
		386	
0005		387	MTR_MON LABEL FAR
		388	PUBLIC MTR_MON
0005	EA9C00	389	JMP FAR PTR MON
		390	
000A		391	MTR_SWIM LABEL FAR
		392	PUBLIC MTR_SWIM
000A	EA2B00	393	JMP FAR PTR SWIM
		394	
000F		395	MTR_DCRT LABEL FAR
		396	PUBLIC MTR_DCRT
000F	EAB300	397	JMP FAR PTR DCRT
		398	
0014		399	MTR_DKBD LABEL FAR
		400	PUBLIC MTR_DKBD
0014	EA9200	401	JMP FAR PTR DKBD
		402	
0019		403	MTR_SCRT LABEL FAR
		404	PUBLIC MTR_SCRT
0019	EAB300	405	JMP FAR PTR SCRT
		406	
001E		407	MTR_SKBD LABEL FAR
		408	PUBLIC MTR_SKBD
001E	EAAE00	409	JMP FAR PTR SKBD
		410	
0023		411	MTR_TTY_INTR LABEL FAR
		412	PUBLIC MTR_TTY_INTR
0023	EAC200	413	JMP FAR PTR TTYINTR
		414	
0028		415	MTR_TTY_POLL LABEL FAR
		416	PUBLIC MTR_TTY_POLL
0028	EAE000	417	JMP FAR PTR TTYPOLL
		418	
002D		419	MTR_IHRET LABEL FAR
		420	PUBLIC MTR_IHRET
002D	EACB00	421	JMP FAR PTR IHRET
		422	
		423	MOMENT ENDS
		424	;\$ EJECT

Monitor Entry Point Segment
 Monitor Entry Point
 Reset the monitor entry point
 Monitor Entry Point
 Monitor Software Interrupt Entry
 Dumb Terminal Processor
 Dumb Key-Board Processor
 Smart Terminal Processor
 Smart Keyboard Processor
 Terminal Interrupt Handler
 Terminal Poll Function
 Interrupt Return

```

LOC  OBJ  LINE  SOURCE
467      ; SWIM      Software Interrupt Monitor
468      ;
469      ; SWIM is the software interrupt monitor entry point. It
470      ; is presumed that both the flags and return address are
471      ; on the top of the stack. Furthermore, it is assumed that
472      ; the return address is a far return, i.e. that both the code
473      ; segment and instruction pointer are on the stack.
474      ;
475      ; Entry: (SP+4) = Original Flags
476      ;         (SP+2) = Original Code Segment
477      ;         (SP+0) = Original Instruction Pointer
478      ;         All other registers as initialized
479      ;
480      SWIM:  PUSH  AX
481            PUSH  BX
482            PUSH  CX
483            PUSH  DX
484            MOV   AX,SP
485            ADD  AX,(4+3)*2      ; Account for all parameters on stack
0034 50      AX
486            PUSH BP
487            PUSH SI
488            PUSH DI
489            PUSH DI
490
491            ; CS is at bottom of stack
0038 1E      CS
492            PUSH DS
0039 16      DS
493            PUSH SS
003A 06      SS
494            PUSH ES
495
003B E86400  CALL  SDS      ; Set Data Segment
003E 89260000 MOV  WORD PTR DS:REGP,SP      ; Save Pointer to Registers
0042 8C160200 MOV  WORD PTR DS:REGP+2,SS
496
0046 E30000  SWIMI: CALL  MTR101      ; CALL MTR101
500      JMP   SWIMI      ; Continue processing monitor commands
0049 EBF8      $ EJECT
502 +1

```

LOC	OBJ	LINE	SOURCE
		547	:: EDIAG -- Boot Diagnostics
		548	::
		549	:: EDIAG is performs any power-on diagnostics.
		550	::
		551	::
0087		552	BDIAG PROC NEAR
		553	
		554	RET
		555	
0087	C3	556	BDIAG ENDP
		557	+1 \$ EJECT

LOC	OBJ	LINE	SOURCE	TEXT
		583	::	DKBD ... Dumb Key-Board
		584	;	
		585	;	DKBD is the assembly language entry point to read
		586	;	and translate a character from the keyboard. This
		587	;	entry point is for 'dumb', ie. naive processing.
		588	;	Each keyboard character returns a unique entry.
		589	;	
		590	;	Entry: NONE
		591	;	
		592	;	Exit: AL = Key-Board Value for character
		593	;	
		594	;	Uses: ???
		595	;	
		596	;	
0072		597	DKBD	PROC FAR
0072	1E	598		PUSH DS
0073	E80C00	599		CALL SDS ; Set Data Segment
		600		
0074	50	601		PUSH AX
0077	E80000	602		CALL D_KBD
		603		
007A	1F	604		POP DS
007B	CB	605		RET
		606		
		607	DKBD	ENDP
		608	+1 \$EJECT	

LOC	OBJ	LINE	SOURCE
		661	TTYPOLL -- Terminal Poll
		662	;
		663	TTYPOLL is a far call to the TTY_POLL routine.
		664	;
		665	Entry: NONE
		666	;
		667	Exit: AX = TRUE if updates will occur
		668	;
		669	;
		670	;
		671	;
009E		672	TTYPOLL PROC NEAR
009E	E	673	PUBLIC TTYPOLL
00A1	C3	674	CALL TTY_POLL
		675	RET
		676	TTYPOLL ENDP
		677	+1 \$ EJECT

LDC OBJ	LINE	SOURCE
	733 +1	* EJECT

LOC	OBJ	LINE	SOURCE	
		761	;; SKBD ... Smart Keyboard	
		762	;	
		763	;	
		764	;	
		765	;	
		766	;	
		767	;	
		768	;	
		769	Exit: AL = Key-Board Value for character	
		770	;	
		771	;	
		772	Uses: ???	
		773	;	
		774	SKBD PROC FAR	
00AE	1E	775	FUSH DS	
00AF	ESF0FF	776	CALL SDS	; Set Data Segment
		777		
00B2	50	778	PUSH AX	
00B3	E30000	779	CALL S_KBD	
		780		
00B6	1F	781	POP DS	
		782	RET	
00B7	0B	783		
		784	SKBD ENDP	
		785	+1 \$ EJECT	

SKBD is a routine to return H19 compatible key codes from the keyboard. This entails emulating the keyboard modes and escape sequence generation.

Entry: NONE

Exit: AL = Key-Board Value for character

Uses: ???

; Set Data Segment

LOC	OBJ	LINE	SOURCE
		816	
		817	
		818	
		819	
		820	
		821	;; TTYINTR - Terminal Interrupt
		822	;;
		823	;; TTYINTR is the global entry vector for the Terminal Interrupt
		824	;; processor. This routine saves all registers, sets the correct
		825	;; Data Segment, and invokes TTY_INTR for the interrupt processing
		826	;;
		827	;; Entry: NONE
		828	;;
		829	;; Exit: NONE
		830	;;
		831	;; Uses: NONE
		832	;;
00C2		833	TTYINTR PROC FAR
		834	
		835	XIF(ZEXTENDED_MONITOR
		836	0)THEN(
		837	PUSH AX
		838	PUSH BX
		839	PUSH CX
		840	PUSH DX
		841	; PUSH SP
		842	PUSH BP
		843	PUSH SI
		844	PUSH DI
		845	PUSHF
		846	; PUSH CS
		847	; PUSH DS
		848	PUSH SS
		849	PUSH ES
		850)FI
		851	+1
		852	
00C2 1E		853	PUSH DS
00C3 E8DCFF		854	CALL SDS
00C6 E80000	E	855	CALL TTY_INTR
00C9 1F		856	POP DS
		857	
		858	XIF(ZEXTENDED_MONITOR
		859	0)THEN(
		860	POP ES
		861	POP SS
		862	; POP DS
		863	; POP CS
		864	POPF
		865	POP DI
		866	POP SI
		867	POP BP
		868	; POP SP
		869	POP DX
		870	POP CX

; Set the Data Segment

LOC	OBJ	LINE	SOURCE
		884	
		885	
		886	
		887	
		888	; MIRET -- Monitor Interrupt Return
		889	;
		890	; MIRET is simply a vector pointing to an Interrupt
		891	; Return instruction.
		892	;
		893	; Entry: Stack clean except for Interrupt Data, ie.
		894	; flags, CS, and IP.
		895	;
		896	; Exit: From interrupt
		897	;
		898	
00CB		899	MIRET PROC FAR
		900	
		901	IRET
		902	
00CB CF		903	MIRET ENDP ; Return from interrupt
		904	
		905	
		906	
		907	
		908	MTR100 ENDS
		909	
		910	
		911	
		912	
		913	END

NAME	TYPE	VALUE	ATTRIBUTES, XREFS
TTY_POLL	L NEAR	0000H	EXTRN 273# 674
TTYINTR	L FAR	00C2H	MTR100 413 833# 873
TTYROLL	L NEAR	009EH	MTR100 PUBLIC 417 672# 673 676
TXMT	L NEAR	0000H	EXTRN 271#
UIES	V DWORD	0000H	EXTRN 300#
UNIT	V BYTE	0052H	S FIELD 171#
UPDATE	V BYTE	0002H	S FIELD 186#
UPDN	V BYTE	0010H	S FIELD 214#
VERT_LINE	V BYTE	0000H	EXTRN 285#
VIDEO_INTR	L NEAR	0000H	EXTRN 266#
VIDEO_INTR_A	L FAR	0000H	EXTRN 248#
VIDEOD	SEGMENT		SIZE=0000H BYTE PUBLIC 'CODE' 316# 317 321
VRAM_SIZE	V BYTE	0008H	S FIELD 227#
WRAP	V BYTE	0011H	S FIELD 215#
WRITEC	L NEAR	0000H	EXTRN 277#
XCA	V DWORD	0000H	EXTRN 301#
XMT	V TBYTE	0000H	EXTRN 310#
XMT_COLOR	V BYTE	0007H	S FIELD 236#
XMT_GRAPHIC	V BYTE	0008H	S FIELD 237#
XMT_REVERSE	V BYTE	0009H	S FIELD 238#
XMT_STRUC	STRUC		SIZE=000AH #FIELDS=8 230 239# 310

ASSEMBLY COMPLETE, NO ERRORS FOUND

SERIES-III 8086/8087/8088 MACRO ASSEMBLER V1.0 ASSEMBLY OF MODULE ASTLIB
 OBJECT MODULE PLACED IN :F2:ASTLIB.OBJ
 INVOCATION LINE CONTROLS: PRINT(:TO:) XREF ERRORPRINT(:TO:)

```

LOC OBJ          LINE  SOURCE
-----
      1 +1 $TITLE('MTR100 Z-Machine Monitor ROM: Assist Library')
      2 +1 $GEN
      3 +1 $SYMBOLS
      4 +1 $XREF
      5    ;;
      6    MTR-100 Assembly Language Assist Library
      7    ;
      8    ; This library provides the assembly language assistance
      9    ; required by the PL/M-86 code in MTR-101. These routines
     10    ; are meant to be called from PL/M-86, and assume that the
     11    ; model of computation is COMPACT.
     12    ;
     13    ; By      Gregg Chandler
     14    ; Date    1-Jan-1982
     15    ;
     16    ; Copyright(C) 1982, Heath Co.
     17
     18 %DEFINE(SEGMENT_LABEL)(ASTLIB)
     19 +1 $INCLUDE(:F2:EXTERN.COM)
     20 +1 $SAVE
     21 +1 $NOGEN
     22
     23 %DEFINE(EXTENDED_MONITOR)(0) %' EXTENDED_MODE is FALSE
     24
     25 PLM_TRUE EQU %FFH ; PL/M-86 Boolean TRUE
     26
     27 ; Structure Definitions
     28
     29 $ $SAVE
     30 $ $NOGEN
     31
     32 ;; Global Structure Definitions
     33 ;
     34
     35 BOOT_PAR_STRUC STRUC
     36     INDEX DB ? ; Device Index
     37     PORT DB ? ; Base Port Address
     38     STRNG DB 80 DUP(?) ; Parameter String Passed
     39     UNIT DB ? ; Unit of Device Booted
     40     BOOT_PAR_STRUC ENDS
     41
     42 COLOR_STRUC STRUC
     43     FORE DB ? ; Foreground Color
     44     BACK DB ? ; Background Color
     45     MSK DB ? ; SHL(Back,3) OR Fore
     46     CLEAR DB ? ; Color to Clear
     47     PAINTED DB ? ; Color to Fully Paint
     48     P_FONT DB ? ; Color to set to Font Pattern
     49     COMP_FONT DB ? ; Color to set to Complement of Font Pattern
     50
     51
     52
     53
     54
     55
     56
     57
     58
     59
     60
     61
     62
     63
     64
     65
     66
     67
     68
     69
     70
     71
     72
     73
     74
     75
     76
     77
     78
     79
     80
     81
     82
     83
     84
     85
     86
     87
     88
     89
     90
     91
     92
     93
     94
     95
     96
     97
     98
     99
    100
    
```

LOC	OBJ	LINE	SOURCE
0008		=2	XMT_GRAPHIC DB
0009		=2	XMT_REVERSE DB
		=2	XMT_STRUC ENDS
		=2	#RESTORE
		=2	Global's Definitions
		=1	%IF (%NES(ASLIB, %SEGMENT_LABEL)) THEN (
		=1	ASLIB
		=1	SEGMENT BYTE PUBLIC 'CODE'
		=1	EXTRN VIDEO_INTR_A: FAR
		=1	ENDS
		=1	ASTLIB
		=1	FI
		=1	115
		=1	116
		=1	117
		=1	%IF (%NES(FONTAB, %SEGMENT_LABEL)) THEN (
		=1	FONTAB
		=1	SEGMENT BYTE PUBLIC 'DATA'
		=1	EXTRN MTR_FONT: BYTE
		=1	ENDS
		=1	FONTAB
		=1	FI
		=1	122
		=1	123
		=1	124
		=1	%IF (%NES(MTR100, %SEGMENT_LABEL)) THEN (
		=1	INTVEC
		=1	SEGMENT PUBLIC 'DATA'
		=1	EXTRN ITV_SST: DWORD
		=1	EXTRN ITV_OBI: DWORD
		=1	EXTRN DS_PTR_0: WORD
		=1	EXTRN DS_PTR_S: WORD
		=1	ENDS
		=1	INTVEC
		=1	MONENT
		=1	SEGMENT PARA PUBLIC
		=1	EXTRN MTR_RES: FAR
		=1	EXTRN MTR_MON: FAR
		=1	EXTRN MTR_SWIM: FAR
		=1	EXTRN MTR_DCRT: FAR
		=1	EXTRN MTR_SCRT: FAR
		=1	EXTRN MTR_DKBD: FAR
		=1	EXTRN MTR_SKBD: FAR
		=1	ENDS
		=1	MONENT
		=1	SEGMENT PARA PUBLIC 'CODE'
		=1	EXTRN %IF (%EXTENDED_MONITOR) THEN (
		=1	EXTRN R3085: NEAR
		=1	FI
		=1	145
		=1	146
		=1	147
		=1	MTR100
		=1	FI
		=1	SDS: NEAR
		=1	Set Data Segment
		=1	ENDS
		=1	%IF (%NES(MTR101, %SEGMENT_LABEL)) THEN (
		=1	CODE
		=1	SEGMENT BYTE PUBLIC 'CODE'
		=1	EXTRN MTR101: NEAR
		=1	EXTRN VIDEO_INTR: NEAR
		=1	EXTRN D_CRT: NEAR
		=1	EXTRN D_KBD: NEAR
		=1	EXTRN S_CRT: NEAR
		=1	PL/M-86 Monitor Loop
		=1	PL/M-86 Video Interrupt
		=1	PL/M-86 Dumb Terminal
		=1	PL/M-86 Dumb Key-Board
		=1	PL/M-86 Smart Keyboard Emulator

LOC	OBJ	LINE	SOURCE
0000		209	
0002		210	; INCLUDE ('F2:PARDEF.COM')
0004		211	; INCLUDE ('F2:INTDEF.COM')
0006		212	
0008		213	
000A		214	REG STRUC
000C		215	
0010		216	ES_ DW ? ; Extra Segment
0012		217	SS_ DW ? ; Stack Segment
0014		218	DS_ DW ? ; Data Segment
0016		219	DI_ DW ? ; Destination Index
0018		220	SI_ DW ? ; Source Pointer
001A		221	BP_ DW ? ; Base Pointer
001C		222	SP_ DW ? ; Stack Pointer
001E		223	DX_ DW ? ; Data
0020		224	CX_ DW ? ; Count
0022		225	BX_ DW ? ; Base
0024		226	AX_ DW ? ; Accumulator
0026		227	IP_ DW ? ; Instruction Pointer
0028		228	CS_ DW ? ; Code Segment
002A		229	FLAG_ DW ? ; Flags
002C		230	
002E		231	REG ENDS
0030		232	
0032		233	
0034		234	NAME ASTLIB ; Name the module
0036		235	
0038		236	ASTLIB SEGMENT BYTE PUBLIC 'CODE' ; Concatenate with the Code Segment
003A		237	ASSUME CS:COROUP,DS:DATA ; Assume MTR-100 Segments
003C		238	+1 \$EJECT

LOC OBJ	LINE	SOURCE
000E	281	;;
	282	;;
	283	;;
	284	;;
	285	;;
	286	;;
	287	;;
	288	;;
	289	;;
	290	;;
	291	;;
	292	;;
	293	;;
	294	;;
	295	;;
	296	;;
	297	;;

000E 9C PROC NEAR
 000F 58 PUBLIC FLAGS
 0010 C3 PUSHF
 POP AX ; AX = Flags
 RET ;
 ENDP
 \$ EJECT +1

FLAGS ... Return 0086 Flags

FLAGS returns the current 0086 flag settings as a WORD.

Entry: NONE

Exit: AX = Entry flags

PROC NEAR PUBLIC FLAGS

PUSHF POP AX ; AX = Flags

RET ;

FLAGS

ENDP

297 +1

\$ EJECT

LOC	OBJ	LINE	SOURCE
		341	;; INCB -- Increment Byte
		342	;;
		343	;; INCB increments the specified byte. The primary
		344	;; use for this routine is to force a value to be
		345	;; updated, rather than letting the compiler optimize
		346	;; variable references to said variable.
		347	;;
		348	;;
		349	;;
		350	;;
		351	;;
		352	;; USAGE: CALL INCB(byte#ptr);
		353	;;
		354	;; Entry: byte#ptr -- Address of byte to increment
		355	;;
		356	;;
0000		357	INCB_P STRUC
0002		358	DW ?
0004		359	DW ?
		360	INCB_BYTEPTR DD ?
		361	INCB_P ENDS
		362	
0011		363	INCB PROC NEAR
		364	PUBLIC INCB
0011 55		365	PUSH BP
0012 8BEC		366	MOV BP,SP
		367	
		368	
0014 C47E04		369	LES DI,[BP],INCB_BYTEPTR
0017 26FE05		370	INC BYTE PTR ES:[DI]
		371	
001A 5D		372	POP BP
001B C20400		373	RET 4
		374	INCB ENDP
		375	+1 \$EJECT

LOC	OBJ	LINE	SOURCE
		423	;; SXMTC ... Smart Terminal Transmit Character
		424	;;
		425	;; 'SXMTC' Is invoked by the smart keyboard/terminal to
		426	;; transmit generated characters. This routine merely
		427	;; transforms the PL/M-86 calling convention into a more
		428	;; conventional assembly language one by moving the sup-
		429	;; plied byte from the stack to the AL register, and
		430	;; then invokes the routine pointed to by 'S_XMTC'.
		431	;;
		432	;;
		433	;; Usage: CALL \$SXMTC(c);
		434	;;
		435	;; Parameters:
		436	;;
		437	;; c - Character generated by the emulator
		438	;;
		439	;; Exit: NONE
		440	;;
		441	;;
		442	;;
0000		443	SXMTCP STRUC
0002		444	DW ?
0004		445	DW ?
0005		446	DB ?
		447	DB ?
		448	ENDS
		449	;;
001E		450	;;
001E 55		451	SXMTC PROC NEAR
001F 8BEC		452	PUBLIC SXMTC
		453	PUSH BP
		454	MOV BP,SP
		455	;;
0021 8A4604		456	MOV AL, [BP+1.SXMTC_C
0024 FF1E0000	E	457	CALL DS:S_XMTC
		458	;; ; Invoke vectored routine
0028 5D		459	POP BP
0029 C20200		460	RET 2
		461	SXMTC ENDP
		462 +1	\$EJECT

LOC	OBJ	LINE	SOURCE
		515	;; XEC - Execute
		516	;;
		517	;;
		518	;;
		519	;;
		520	;; XEC is called to begin execution at the specified address.
		521	;; This routine assumes the registers are saved in the order
		522	;; specified in SWIM in the MTR100 module. It is necessary
		523	;; to play some tricks to account for programs which may have
		524	;; modified the Stack Segment.
		525	;;
		526	;; Usage: CALL XEC(regfp)
		527	;;
		528	;; Entry: regfp = Base Address of register structure POINTER
		529	;;
		530	;; Exit: to new address
		531	;;
		532	;;
		533	XECP STRUC
0000		534	DW ?
0002		535	DW ?
0004		536	DD ? ; Pointer to register structure
		537	XECP ENDS
		538	
		539	
0042		540	XEC PROC NEAR
		541	PUBLIC XEC
0042 55		542	PUSH BP
0043 8BEC		543	MOV BP,SP
		544	
		545	;; Initialize Software Interrupt Vectors
		546	;;
		547	;;
		548	;;
		549	;;
		550	MOV AX,SEG ITV_SST
		551	MOV ES,AX ; ES = Interrupt Segment
		552	MOV AX,OFFSET MTR_SWIM
		553	MOV WORD PTR ES:ITV_SST+0,AX
		554	MOV WORD PTR ES:ITV_ORI+0,AX
		555	MOV AX,SEG MTR_SWIM
		556	MOV WORD PTR ES:ITV_SST+2,AX
		557	MOV WORD PTR ES:ITV_ORI+2,AX
		558)FI
		559	;;
		560	;; Restore Saved Registers
		561	;;
0045 C57604		562	LDS SI,[BP],XECP_REGP ; DS:SI = Register Structure
		563	;;
		564	;;
		565	;;
		566	;;
		567	;;
		568	;;
		569	;;

LOC	OBJ	LINE	SOURCE
0000		604	;;
0002		605	;; INIT_ITV -- Initialize Interrupt Vectors
0004		606	;;
0008		607	;; INIT_ITV initializes the specified interrupt vector
		608	;; to the specified pointer value.
		609	;; Usage: CALL INIT_ITV(ivi,ivp);
		610	;; Parameters
		611	;; ivi = Interrupt Vector Index (BYTE)
		612	;; ivp = Pointer to Interrupt Vector Routine (POINTER)
		613	;; Exit: Specified Interrupt Vector Initialized to specified
		614	;; pointer value.
		615	;;
		616	;;
		617	;;
		618	;;
		619	;;
		620	;;
		621	;;
		622	;;
		623	;;
		624	INIT_ITVP STRUC
		625	DW ?
		626	DW ?
		627	DD ?
		628	DB ?
		629	DB ?
		630	ENDS
		631	; An entire word is on the stack
0061		632	INIT_ITV PROC NEAR
0061	55	633	PUBLIC INIT_ITV
0062	8BEC	634	PUSH BP
0064	1E	635	MOV BP,SP
		636	PUSH DS
		637	;
		638	; Compute address of interrupt vector
		639	;
0065	8A4608	640	MOV AL,[BP],IVI
0068	02C0	641	ADD AL,AL
006A	02C0	642	ADD AL,AL
006C	B400	643	MOV AH,0
006E	8BF8	644	MOV DI,AX
		645	; DI = 4 * IVI
0070	B8-----	646	MOV AX,SEG ITV,_SST
0073	8ED8	647	DS,AX
0075	C44604	648	AX,[BP],IVP
0078	8705	649	LDI+0],AX
007A	8C4502	650	LDI+2],ES
		651	; Set Segment
007D	1F	652	POP DS
007E	5D	653	POP BP
007F	C20600	654	RET 06H
		655	ENDP
		656	;
		657	;
		658	;

LOC	OBJ	LINE	SOURCE
		661	
		662	
		663	
		664	
		665	
		666	;; @MOVE -- Move
		667	;;
		668	;; @MOVE moves the specified bytes from one location to
		669	;; another. Though the 8086 has instructions tuned for
		670	;; this operation, it is necessary to first compute the
		671	;; direction of the move---whether from high to low, or
		672	;; low to high. Since the true 20-bit addresses must be
		673	;; used for comparison, this takes a little work.
		674	;;
		675	;; NOTE: This routine will not wrap out of
		676	;; any of the segment boundaries,
		677	;; rather, it wraps around within
		678	;; them.
		679	;;
		680	;; Entry: DS = Source Segment
		681	;; SI = Source Offset
		682	;; ES = Destination Segment
		683	;; DI = Destination Offset
		684	;; CX = Byte Count
		685	;;
		686	;; Exit: Byte moved, SI and DI advanced, CX=0
		687	;;
		688	;; Uses: ALL
		689	;;
		690	;;
		691	;; %IF(%EXTENDED_MONITOR
		692	;; 0) THEN (
		693	;; +1
		694	;; @MOVE PROC NEAR
		695	;;
		696	;; AND CX,CX
		697	;; JNZ MOV0 ; There are bytes to move
		698	;;
		699	;; RET ; No bytes to move
		700	;;
		701	;; MOV0: PUSH CX
		702	;;
		703	;; ; Compute Absolute Source Address
		704	;;
		705	;; MOV AX,DS
		706	;; MOV DL,AH
		707	;; MOV CL,4
		708	;; SHL AX,CL
		709	;; SHR DL,CL
		710	;; ADD AX,SI
		711	;; ADD DL,0 ; DL,AH,AL = Source Address
		712	;;
		713	;; ; Compute Absolute Destination Address
		714	;;
		715	;; MOV BX,ES

XREF SYMBOL TABLE LISTING

NAME	TYPE	VALUE	ATTRIBUTES, XREFS
73SEG	SEGMENT	0000H	SIZE=0000H PARA PUBLIC
ALTERNATE	V BYTE	0000H	S FIELD 67#
ANSI	V BYTE	0001H	S FIELD 68#
ASTLIB	SEGMENT	0002H	SIZE=0082H BYTE PUBLIC 'CODE' 205# 236 756
AUTO_CR	V BYTE	0003H	S FIELD 69#
AUTO_LF	V BYTE	0004H	S FIELD 70#
AUTO_REPEAT	V BYTE	0004H	S FIELD 71#
AX	V WORD	0014H	S FIELD 226#
BACK	V BYTE	0001H	S FIELD 44#
BASE	V WORD	0006H	S FIELD 316#
BBCOUNT	V WORD	0001H	S FIELD 100#
BOOT_PAR	V 83	0000H	EXTRN 187#
BOOT_PAR_STRUC	STRUC		SIZE=0053H #FIELDS=4 35 40# 187
BP	V WORD	000AH	S FIELD 221#
BURST	V BYTE	0000H	S FIELD 99#
BWD	V BYTE	0005H	S FIELD 72#
BX	V WORD	0012H	S FIELD 225#
CORGRP	GROUP		MTR100 CODE ASTLIB VIDEQA 205# 237
CLEAR	V BYTE	0003H	S FIELD 46#
CODE	SEGMENT		SIZE=0000H BYTE PUBLIC 'CODE' 148# 162 205
COL	V BYTE	0005H	S FIELD 102#
COLOR	V	0000H	EXTRN 188#
COLOR_STRUC	STRUC		SIZE=0007H #FIELDS=7 42 50# 188
COMMAND	V BYTE	0000H	S FIELD 58#
COMP_FONT	V BYTE	0006H	S FIELD 49#
COUNT	V WORD	0003H	S FIELD 101#
CPL	V BYTE	0000H	S FIELD 87#
CRLF	L NEAR	0000H	EXTRN 159#
CRTC_CURSOR	V	0000H	EXTRN 189#
CRTC_DISPLAY	V	0000H	EXTRN 190#
CRTC_STRUC	STRUC		SIZE=0003H #FIELDS=2 52 55# 189 190
CS	V WORD	0018H	S FIELD 228#
CURSQR	V BYTE	0006H	S FIELD 73#
CURSQR_ON	V BYTE	0007H	S FIELD 74#
CX	V WORD	0010H	S FIELD 224#
D_CRT	L NEAR	0000H	EXTRN 151#
D_KBD	L NEAR	0000H	EXTRN 152#
D_XMTC	V DWORD	0000H	EXTRN 175# 274
DATA	SEGMENT		SIZE=0000H WORD PUBLIC 'DATA' 167# 195 237
DCI	V DWORD	0000H	EXTRN 173#
DHC	V DWORD	0000H	EXTRN 174#
DI	V WORD	0006H	S FIELD 219#
DS	V WORD	0004H	S FIELD 218#
DSC	V BYTE	0001H	S FIELD 88#
DX	V WORD	000EH	S FIELD 223#
DXMTC	L NEAR	0000H	ASTLIB PUBLIC 268# 269 278
DXMTC_C	V BYTE	0004H	S FIELD 263# 273
DXMTCF	STRUC		SIZE=0006H #FIELDS=4 260 265#
EDC	V DWORD	0000H	EXTRN 176#
EMEC	V DWORD	0000H	EXTRN 177#
ES	V WORD	0000H	S FIELD 216#

NAME	TYPE	VALUE	ATTRIBUTES, XREFS
PROMPT.	V DWORD	00000H	EXTRN 181#
RDC	V DWORD	00000H	EXTRN 182#
REG	STRUC		SIZE=001CH #FIELDS=14 214 231# 578
REGP	V DWORD	00000H	EXTRN 170#
RESETF	V BYTE	00000H	EXTRN 171#
REVERSE	V WORD	0000CH	S FIELD 79#
ROW	V BYTE	00006H	S FIELD 103#
S_CRT	L NEAR	00000H	EXTRN 153#
S_KBD	L NEAR	00000H	EXTRN 154#
S_XMTC	V DWORD	00000H	EXTRN 183# 457
SYS	L NEAR	00000H	EXTRN 142#
SHIFTED	V BYTE	0000EH	S FIELD 80#
SI	V WORD	00003H	S FIELD 220#
SLI	V BYTE	00004H	S FIELD 91#
SP	V WORD	0000CH	S FIELD 222# 577
SPC	V BYTE	00005H	S FIELD 92#
SS	V WORD	0002H	S FIELD 217# 576
START	V WORD	00000H	S FIELD 53#
STATUS	V BYTE	0000FH	S FIELD 81#
STRNG	V BYTE	0002H	S FIELD 38#
SW401	V BYTE	0006H	S FIELD 93#
SW402	V BYTE	0007H	S FIELD 94#
SXMTC	L NEAR	001EH	ASTLIB PUBLIC 451# 452 461
SXMTC_C	V BYTE	0004H	S FIELD 446# 456
SXMTCP	STRUC		SIZE=0064H #FIELDS=4 443 448#
TTY_INTR	L NEAR	00000H	EXTRN 156#
TTY_POLL	L NEAR	00000H	EXTRN 157#
TXMT	L NEAR	00000H	EXTRN 155#
UIES	V DWORD	00000H	EXTRN 184#
UNIT	V BYTE	0052H	S FIELD 39#
UPDATE	V BYTE	0002H	S FIELD 54#
UPDN	V BYTE	0010H	S FIELD 82#
VAL	V WORD	0008H	S FIELD 317#
VERT_LINE	V BYTE	00000H	EXTRN 169#
VIDEO_INTR	L NEAR	00000H	EXTRN 150# 492
VIDEO_INTR_A	L FAR	002CH	ASTLIB PUBLIC 474# 475 509
VIDEOA	SEGMENT		SIZE=0000H BYTE PUBLIC 'CODE' 200# 201 205
VRAM_SIZE	V BYTE	0008H	S FIELD 95#
WRAP	V BYTE	0011H	S FIELD 83#
WRITEC	L NEAR	0000H	EXTRN 161#
XCA	V DWORD	0000H	EXTRN 185#
XEC	L NEAR	0042H	ASTLIB PUBLIC 540# 541 598
XECP	STRUC		SIZE=0008H #FIELDS=3 533 537#
XECP_REGP	V DWORD	0004H	S FIELD 536# 562
XMT	V TBYTE	0000H	EXTRN 174#
XMT_COLOR	V BYTE	0007H	S FIELD 104#
XMT_GRAPHIC	V BYTE	0008H	S FIELD 105#
XMT_REVERSE	V BYTE	0009H	S FIELD 106#
XMT_STRUC	STRUC		SIZE=000AH #FIELDS=8 98 107# 194

ASSEMBLY COMPLETE, NO ERRORS FOUND

SERIES-III 3086/3087/3088 MACRO ASSEMBLER V1.0 ASSEMBLY OF MODULE VIDEOA
 OBJECT MODULE PLACED IN :F2:VIDEOA.OBU
 INVOCATION LINE CONTROLS: PRINT(:TO:) XREF ERRORPRINT(:TO:)

LOC	OBU	LINE	SOURCE
		1 +1	\$TITLE('MTR100 Z-Machine Monitor ROM: Assembly Video Library')
		2 +1	\$GEN
		3 +1	\$SYMBOLS
		4 +1	\$XREF
		5	;;
		6	MTR100 Video Library Assembly Procedures
		7	;
		7	Copyright(C) 1982, by:
		8	;
		9	Heath Co.
		10	Benton Harbor, MI
		11	47022
		12	;
		13	;
		14	;
		15	This library implements the critical assembly language
		16	routines necessary for the video library. These routines
		17	assume the PL/M-86 calling convention and the COMPACT
		18	model of computation, however, they are invoked via "far"
		19	calls.
		20	;
		21	These routines assumes that the CRT-C is programmed for 9
		22	scan lines per character, and that the current VRMM par-
		23	ameters are valid. The video address is computed as:
		24	vert*16+128+char = vert*8*256+char
		25	;
		26	;
		27	;
		28	By: G. Chandler
		29	Date: 7-Jan-1982
		30	;
		31 +1	\$ EJECT

LOC	OBJ	LINE	SOURCE
		=1	;; Paragraph Usage
		=1	;
		=1	;
		=1	;
		=1	Due to reset hardware conditions, various paragraphs within the ROM are hard-coded to represent certain values. The current paragraph usage is as follows:
		=1	;
		=1	;
		=1	Paragraph Usage
		=1	;
		=1	;
		=1	;
		=1	;
		=1	Interrupt Vectors
		=1	User Program
		=1	;
		=1	User Program
		=1	Video Ram: Blue Plane
		=1	Video Ram: Red Plane
		=1	Video Ram: Green Plane
		=1	Reserved
		=1	;
		=1	Reserved
		=1	F0FFFH
		=1	MTR-100: 8085 Processor Switch Code
		=1	MTR-100: MTR-100 Entry Vectors
		=1	MTR-100: MTR-100 Code
		=1	;
		=1	;
		=1	;
		=1	;
		=1	;
		=1	MTR-100: Jump Far to MTR-100 Start
		=1	;
		=1	;
00200		=1	SS_SIZE EQU 2*16
00400		=1	DS_SIZE EQU 4*16
		=1	;
		=1	;
		=1	;
		=1	;
		=1	;
		=1	;
		=1	;
00000		=1	PAR_INT EQU 00000H
00400		=1	PAR_USR EQU 00040H
C0000		=1	PAR_VID EQU 0C000H
D0000		=1	PAR_BLU EQU 0D000H
E0000		=1	PAR_RED EQU 0E000H
		=1	PAR_GRN EQU 0E000H
		=1	;
		=1	;
FE000		=1	PAR_BSS EQU 0FE00H
FE010		=1	PAR_MTR EQU 0FE01H
FFFF0		=1	PAR_BSS EQU 0FFFFH
		=1	;
		=1	;
		=1	;
		=1	;
		=1	;
		=1	;
		=1	%*DEFINE(SEGMENT_LABEL) (VIDEQA)
		=1	77 +1 \$ INCLUDE (:FZ:EXTERN.COM)
		=1	78 +1 \$ SAVE
		=1	79 +1 \$NOGEN
		=1	80
		=1	81
		=1	;
		=1	%*DEFINE(EXTENDED_MONITOR) (0) %' EXTENDED_MODE is FALSE
00FF0		=1	PLM_TRUE EQU 0FFFH ; PL/M-86 Boolean TRUE
		=1	82
		=1	83
		=1	84
		=1	85
		=1	;
		=1	Structure Definitions
		=1	86 \$ INCLUDE (:FZ:STRDEF.COM')
		=2	87 \$SAVE
		=2	88 \$NOGEN

LOC	OBJ	LINE	SOURCE
00000		144	H19_PAR_STRUC STRUC
00001		145	CPL DB
00002		146	DISC DB
00003		147	LPS DB
00004		148	POVRAM DB
00005		149	SLI DB
00006		150	SPC DB
00007		151	SM401 DB
00008		152	SM402 DB
		153	VRAM_SIZE DB
		154	H19_PAR_STRUC ENDS
		155	
		156	XMT_STRUC STRUC
00001		157	BURST DB
00003		158	BCCOUNT DW
00005		159	COUNT DW
00006		160	COL DB
00007		161	ROW DB
00008		162	XMT_COLOR DB
00009		163	XMT_GRAPHIC DB
		164	XMT_REVERSE DB
		165	XMT_STRUC ENDS
		166	
		167	
		168	#RESTORE
		169	
		170	
		171	
		172	
			; Global's Definitions
			;
			XIF(%NES(ASTLIB,%SEGMENT_LABEL)) THEN (
			ASTLIB SEGMENT BYTE PUBLIC 'CODE'
			EXTRN VIDEO_INTR_A:FAR
			ENDS
			ASTLIB
			FI
);
			XIF(%NES(FONTAB,%SEGMENT_LABEL)) THEN (
			FONTAB SEGMENT BYTE PUBLIC 'DATA'
			EXTRN MTR_FONT:BYTE
			ENDS
			FONTAB
			FI
);
			XIF(%NES(MTR100,%SEGMENT_LABEL)) THEN (
			; INTVEC SEGMENT PUBLIC 'DATA'
			EXTRN ITV_SST:DWORD
			EXTRN ITV_OBI:DWORD
			EXTRN DS_PTR_O:WORD
			EXTRN DS_PTR_S:WORD
			ENDS
			; INTVEC
);
			; INTVEC
			MONENT
			SEGMENT PARA PUBLIC
			EXTRN MTR_RES:FAR
			EXTRN MTR_MON:FAR
			EXTRN MTR_SWIM:FAR
			EXTRN MTR_DCRT:FAR
			EXTRN MTR_SCRT:FAR
			; Monitor Reset
			; Monitor CALL Entry
			; Monitor Software Interrupt Entry
			; Monitor Dumb Terminal Emulator
			; Monitor Smart Terminal Emulator

```

LOC OBJ          LINE          SOURCE
=====
=1              =1              EXTRN      CRIC_DISPLAY:CRIC_STRUC
=1              =1              EXTRN      ESCP:ESCP_STRUC
=1              =1              EXTRN      H19_MODE:H19_MODE_STRUC
=1              =1              EXTRN      H19_PAR:H19_PAR_STRUC
=1              =1              EXTRN      XMT:XMT_STRUC
=1              =1              ENDS
=1              =1              DATA
=1              =1              )FI
=1              =1              259
=1              =1              260
=1              =1              261
=1              =1              %IF(ZNES(VIDEOA,%SEGMENT_LABEL)) THEN (
=1              =1              VIDEOA
=1              =1              SEGMENT BYTE PUBLIC 'CODE'
=1              =1              VIDEOA
=1              =1              ENDS
=1              =1              )FI
=1              =1              262
=1              =1              263
=1              =1              264
=1              =1              265
=1              =1              266
=1              =1              267
=1              =1              268
=1              =1              269
=1              =1              270
=1              =1              271
=1              =1              272
=1              =1              273
=1              =1              274
=1              =1              275
=1              =1              276
=1              =1              277
=1              =1              278
=1              =1              279
=1              =1              280
=1              =1              281
=1              =1              282
=1              =1              283
=1              =1              284
=1              =1              285
=1              =1              286
=1              =1              287
=1              =1              288
=1              =1              289
=1              =1              290
=1              =1              291
=1              =1              292
=1              =1              293
=1              =1              294
=1              =1              295
=1              =1              296
=1              =1              297
=1              =1              298
=1              =1              299
=1              =1              300
=1              =1              301
=1              =1              302 +1
=1              =1              $EJECT

; Color Bit Definitions in Video Command Port
CB_BLK EQU 000B ; Black
CB_BLU EQU 100B ; Blue
CB_RED EQU 001B ; Red
CB_GRN EQU 010B ; Green
CB_MAG EQU CB_RED+CB_BLU ; Magenta
CB_CYN EQU CB_GRN+CB_BLU ; Cyan
CB_YEL EQU CB_GRN+CB_RED ; Yellow
CB_WHT EQU CB_GRN+CB_RED+CB_BLU ; White

SLPC EQU 9+2 ; Scan Lines Per Character, 9 for char. + 2 for attrib.
VID_CMD EQU 0D8H ; Video Command Port

; Concatenate with the Code Segment
VIDEOA SEGMENT BYTE PUBLIC 'CODE'
ASSUME CS:CGROUP,DS:DATA
CGROUP GROUP MTR100, CODE,ASTLIB,VIDEOA
$RESTORE
NAME VIDEOA
ENDS
    
```

LOC	OBJ	LINE	SOURCE
		317	;; MTR_DFC -- Display Font Character
		318	;;
		319	MTR_DFC displays the indexed font character at the specified
		320	screen address. The value of compf is XOR'd with each of
		321	the bytes displayed, thus, the character will be displayed
		322	straight if compf is 0, or it will be displayed in reverse
		323	video if compf is 0FFFFFFH.
		324	;;
		325	Usage: CALL MTR_DFC(horz,vert,findx,compf)
		326	;;
		327	Entry:
		328	horz = Horizontal character index within line [0,79] BYTE
		329	vert = Vertical character index within screen [0,24] BYTE
		330	findx = Index into Font ROM for the character [0,127] BYTE
		331	compf = Mask to be XOR'd with display bytes WORD
		332	;;
		333	Globals:
		334	Color structure initialized
		335	;;
		336	Exit: NONE
		337	;;
		338	;;
		339	P_DFC
0000		340	STRUC
0002		341	DW ?
0004		342	DW ?
0006		343	DW ? ; A real WORD
0008		344	DB ? ; Font Character Index
0009		345	DB ? ; Vertical Character Line
000A		346	DB ? ; Vertical Character Line
000B		347	DB ? ; Vertical Character Line
000C		348	DB ? ; Horizontal Character Column
000D		349	DB ?
		350	P_DFC ENDS
		351	;;
		352	;;
0001		353	MTR_DFC PROC FAR
0001	55	354	PUBLIC MTR_DFC
0002	8BEC	355	PUSH BP
		356	MOV BP,SP
0004	1E	357	PUSH DS
0005	E4D8	358	IN AL,VID_CMD
0007	50	359	PUSH AX
		360	; Save current parameters
		361	; Compute offset of character in Video RAM
		362	; Compute offset of character in Font ROM
0008	8A5E0C	363	MOV BL,[BP].DFC_HORZ ; BL = Horizontal Character Index
000B	8A7E0A	364	MOV BH,[BP].DFC_VERT ; BH = Vertical Line Index
000E	E80C03	365	CALL CCA ; Compute Character Address
		366	; Compute offset of character in Font ROM
		367	; Compute offset of character in Font ROM
0011	8A4E08	368	MOV CL,[BP].DFC_FINDX ; CL = Font ROM Character Index
0014	B500	369	MOV CH,0
		370	; CL = Font ROM Character Index
		371	; CH,0

LOC	OBJ	LINE	SOURCE
006A	5B	427	
006B	EBB602	428	POP BX ; BL = COLOR.FRONT
006E	7403	429	CALL SWP ; Set Write Parameters
0070	EB3700	430	JZ DF04 ; No Foreground Color
0073	B307	431	CALL DF05 ; Display the character
0075	EBAC02	432	CALL LABEL NEAR
0078	EB13	433	MOV BL,CL_MHT
		434	CALL SWP ; Write to all planes of VRAM
		435	JMP SHORT DF04_6
007A	A00200	436	DF04_3: MOV AL,COLOR_MSK
007D	50	437	PUSH AX ; Save Color Mask
007E	8B4E06	438	MOV CX,[BP].DF0_COMPF ; CX = Complement Flag
0081	C5060000	439	LDS AX,Font ; DS:SI = Pointer to Font Character
0085	03F0	440	ADD SI,AX ; Set Write Parameters to all Green
0087	EB1602	441	CALL SWPG
008A	EB1D00	442	CALL DF05
008D		443	CALL LABEL NEAR
		444	DF04_6: LABEL
		445	
		446	
		447	
		448	
		449	Save Character Attributes
008D	8A4608	449	MOV AL,[BP].DF0_FINDX ; Save font Index
0070	26385804	450	MOV ES:BYTE PTR EDI+9*1281,AL
		451	
0095	58	452	POP AX ; AL = COLOR_MSK
0096	243F	453	AND AL,00111111B ; Set Color Mask
0098	80E580	454	CH,080H
009B	0AC5	455	OR AL,CH ; Set Reverse Video Flag
009D	263858005	456	MOV ES:BYTE PTR EDI+10*1281,AL ; Save Character Attributes
		457	
00A2	58	458	POP AX
00A3	E6D8	459	VID_CMD,AL ; Restore original video register
00A5	1F	460	DS
		461	
00A6	5D	462	POP BP
00A7	CA0800	463	RET 8
		464	
		465	
		466	
		467	
		468	
00AA	56	469	DF05: PROC NEAR
		470	PUSH SI
		471	
00AB	AD	472	LODSW ; AX = Font character (0,1)
00AC	33C1	473	XOR AX,CX ; Complement bits for reverse video
00AE	263805	474	ES:BYTE PTR EDI+0*1281,AL
00B1	2638A58000	475	MOV ES:BYTE PTR EDI+1*1281,AH
00B6	AD	476	LODSW ; AX = Font character (2,3)
00B7	33C1	477	XOR AX,CX
00B9	2638350001	478	ES:BYTE PTR EDI+2*1281,AL
00BE	2638A58001	479	MOV ES:BYTE PTR EDI+3*1281,AH
00C3	AD	480	LODSW ; AX = Font character (4,5)
00C4	33C1	481	XOR AX,CX

LOC	OBJ	LINE	SOURCE
		519	;; MTR_EDC -- Erase Display Character
		520	;;
		521	MTR_EDC erases the specified character from the screen.
		522	In this case, it merely zeroes out video RAM.
		523	;;
		524	NOTE: This routine assumes that all of the char-
		525	acters are on the same line! If they are
		526	not, bad video RAM may be smashed due to
		527	the memory mapping scheme (in a 32KB system).
		528	;;
		529	Usage: CALL MTR_EDC(line, char, count);
		530	;;
		531	Parameters:
		532	;;
		533	line -- Line of character to erase, BYTE, [0-24]
		534	char -- First Character to erase, BYTE, [0-79]
		535	count -- Number of characters to erase, BYTE, [0-79]
		536	;;
		537	Exit: NONE
		538	;;
		539	;;
0000		540	P_EDC STRUC
0002		541	DW ?
0004		542	DW ?
0006		543	DW ?
0007		544	EDC_COUNT ?
0008		545	EDC_CHAR ?
0009		546	EDC_LINE ?
000A		547	EDC_LINE ?
000B		548	EDC_LINE ?
		549	DB ?
		550	P_EDC ENDS
		551	;;
		552	;;
010E		553	MTR_EDC PROC FAR
010E	55	554	PUBLIC MTR_EDC
010F	8BEC	555	PUSH BP
		556	MOV BP, SP
		557	;;
0111	E4D8	558	IN AL, VID_CMD
0113	50	559	PUSH AX
		560	;; Save Video Parameters
		561	;;
		562	Fetch Parameters
0114	8A7E0A	563	MOV BH, [BP].EDC_LINE; BH = Row
0117	8A5E08	564	MOV BL, [BP].EDC_CHAR; BL = Column
011A	E80102	565	CALL CCA_
011D	8A5606	566	MOV DL, [BP].EDC_COUNT
0120	B600	567	DI, 0
0122	B90900	568	MOV CX, SLP0-2 ; DX = Character Count
		569	;; CX = Scan Lines per Character
		570	;; Zero the characters one scan line at a time
0125	HC	571	CLD ; Auto-Increment
0126	51	572	PUSH CX ; Save Scan Line Count
		573	EDC1: ;

LOC	OBJ	LINE	SOURCE
		628 +1	\$EJECT

LOC	OBJ	LINE	SOURCE
018A	8BF3	684	MOV SI,BX ; SI = Source Character Address
018C	8A5606	685	MOV DL,(BP),MDC_COUNT ; DX = Move Count
018F	B600	686	MOV DH,0 ; CX = Scan Lines per Character
0191	B90B00	687	MOV CX,SLPC
0194	33FA00	688	CMP DX,0
0197	744E	689	JE MDC4 ; No characters to move
0199	FC	690	CLD ; Assume Auto-Increment
019A	B80000	691	MOV AX,0 ; Assume no negative word adjust
019D	3BFE	692	CMP DI,SI
019F	7213	693	JB MDC1
		694	
		695	
		696	
		697	
01A1	FD	698	STD ; Destination >= Source
01A2	BF0FFF	699	MOV AX,-1 ; Auto-Decrement
01A5	03FA	700	ADD DI,DX ; Back each address up one byte before movsw
01A7	4F	701	DEC DI
01A8	03F2	702	ADD SI,DX
01AA	4E	703	DEC SI ; Advance Pointers to the end of strings
		704	
01AB	8A1E0500	705	MOV BL,H19_MODE,BMO
01AF	30FFFF	706	CMP BL,PLM_TRUE
01B2	741D	707	JZ MDC2 ; Optimize Black & White
		708	
		709	
		710	
		711	
01B4	51	712	MDC1: PUSH CX ; Move Color Characters
01B5	B304	713	MOV BL,CL_GRN ; Move Green Plane
01B7	E83500	714	CALL MDC5 ; Move Red Plane
01BA	B302	715	MOV BL,CL_RED ; Move Red Plane
01BC	E83000	716	CALL MDC5 ; Move Blue Plane
01BF	B301	717	MOV BL,CL_BLU ; Move Blue Plane
01C1	E82E00	718	CALL MDC5
01C4	59	719	POP CX
		720	
01C5	81C78000	721	ADD DI,128
01C9	81C68000	722	ADD SI,128
01CD	E2E5	723	LOOP MDC1
01CF	EB16	724	JMP SHORT MDC4
		725	
		726	
		727	
		728	
		729	
		730	
		731	
		732	
		733	
		734	
		735	
		736	
		737	
		738	
01D1	50	727	MDC2: PUSH AX ; Set Multiple Write for all planes
01D2	E88E01	728	CALL SWPG ; Source Plane = Destination Plane
01D5	8ED8	729	MOV DS,AX
01D7	58	730	POP AX
		731	
01D8	51	732	MDC3: PUSH CX ; Set Multiple Write for all planes
01D9	E81A00	733	CALL MDC6 ; Source Plane = Destination Plane
01DC	59	734	POP CX
01DD	81C78000	735	ADD DI,128
01E1	81C68000	736	ADD SI,128
01E5	E2F1	737	LOOP MDC3
		738	

LOC	OBJ	LINE	SOURCE
		776	;;
		777	MTR_MD_L ... Move Display Line
		778	;
		779	MTR_MD_L moves the line of characters to the spec-
		780	ified destination line.
		781	;
		782	;
		783	;
		784	;
		785	Usage: CALL MTR_MD_L(line_src,line_dst);
		786	;
		787	Parameters:
		788	;
		789	line_src -- Source Line
		790	line_dst -- Destination Line
		791	;
		792	Exit: NONE
		793	;
		794	;
		795	P_MD_L STRUC
0000		796	DW ?
0002		797	DW ?
0004		798	DW ?
0006		799	DB ? ; Destination Line
0007		800	DB ?
0008		801	DB ? ; Source Line
0009		802	DB ?
		803	P_MD_L ENDS
		804	
		805	
0208		806	MTR_MD_L PROC FAR
		807	PUBLIC MTR_MD_L
0209	SBEC	808	PUSH BP
		809	MOV BP,SP
020B	IE	810	PUSH DS
		811	IN AL,VID_CMD
020C	EAD8	812	PUSH AX
		813	
		814	;
		815	Initialize from Parameters
020F	SA7E06	816	MOV BH,[BP].MDL_DST
		817	MOV BL,0
0214	E80701	818	CALL CCA_
		819	MOV BH,[BP].MDL_SRC
021A	ESFA00	820	CALL CCA
		821	MOV SI,BX
		822	;
021F	B600	823	MOV DH,0
		824	MOV DL,H19_PAR.CPL
0221	8A160000	825	MOV CX,SLPC
		826	MOV CLD
0228	FC	827	CLD
		828	MOV BL,H19_MODE.BMO
022D	80FEFF	829	CMP BL,PLM_TRUE
		830	JIE MDL2 ; Optimize this line move

LOC	OBJ	LINE	SOURCE
027D	C3	885	RET
		886	
		887	MDL5 ENDP
		888	
		889	
		890	
		891	
		892	+1 \$EJECT

LOC	OBJ	LINE	SOURCE		
02BE	00				
02BF	00				
02C0	00				
02C1	C0	948		DB	11000000B,00111111B,11111111B,11111100B
02C2	3F				
02C3	FF				
02C4	FC	949		DB	01111100B,11000000B,01000000B,00000010B
02C5	7C				
02C6	C0				
02C7	40				
02C8	02				
02C9	47	950		DB	01000111B,00000000B,00111111B,11111100B
02CA	00				
02CB	3F				
02CC	FC	951		DB	01000100B,00001100B,00100010B,00000000B
02CD	44				
02CE	0E				
02CF	22				
02D0	00				
02D1	47	952		DB	01000111B,00000001B,11111100B,00000000B
02D2	01				
02D3	FC				
02D4	00				
02D5	7C	953		DB	01111100B,11000001B,00000100B,00000000B
02D6	C1				
02D7	04				
02D8	00				
02D9	C0	954		DB	11000000B,00111111B,11111000B,00000000B
02DA	3F				
02DB	F8				
02DC	00				
02DD	00	955		DB	00000000B,00000000B,00000000B,00000000B
02DE	00				
02DF	00				
02E0	00				

956
 957 MTR_FRMPT ENDP
 958
 959
 960
 961
 962 +1 \$EJECT

LOC	OBJ	LINE	SOURCE
		1018	MTR_RDC ENDP
		1019	
		1020	
		1021	
		1022	
		1023 +1	\$EJECT

LOC	OBJ	LINE	SOURCE
		1065	;;
		1066	;;
		1067	;;
		1068	;;
		1069	;;
		1070	;;
		1071	;;
		1072	;;
		1073	;;
		1074	;;
		1075	;;
		1076	;;
		1077	;;
		1078	;;
		1079	;;
		1080	;;
		1081	;;
0317	D0E7	1082	CCA
0317	D0E7	1083	PROC
0319	D0E7	1084	SHL
031B	D0E7	1085	SHL
031D	C3	1086	RET
		1087	ENDP
		1088	
		1089	
031E	E9F6FF	1090	CCA
031E	E9F6FF	1091	PROC
0321	8BFB	1092	CALL
0323	C3	1093	MOV
		1094	RET
		1095	ENDP
		+1	CCA
			\$EJECT

-- Compute Character Address

CCA returns the character offset in BX based on the character Row and Column.

Entry: BH -- Character Row
BL -- Character Column

Exit: BX -- Character Offset within Video Plane

Uses: BH

NEAR
BH,1
BH,1
BH,1

NEAR
CCA
DI,BX

LOC	OBJ	LINE	SOURCE					
0350	0000	1151	SWPB	DM	00000H			; Black
0352	00C0	1152		DM	PAR_BLU			; Blue
0354	00D0	1153		DM	PAR_RED			; Red
0356	00D0	1154		DM	PAR_RED			; Magenta
0358	00E0	1155		DM	PAR_GRN			; Green
035A	00E0	1156		DM	PAR_GRN			; Cyan
035C	00E0	1157		DM	PAR_GRN			; Yellow
035E	00E0	1158		DM	PAR_GRN			; White
		1159						
		1160	SWP	ENDP				
		1161						
		1162						
		1163						
		1164						
0360		1165	SWPG	PROC	NEAR			
0360	E4D8	1166		IN	AL,VID_CMD			
0362	240F	1167		AND	AL,00001111B			
0364	E6D8	1168		OUT	VID_CMD,AL			
0366	B800E0	1169		MOV	AX,FAR_GRN			
0369	23C0	1170		AND	AX,AX			
036B	8EC0	1171		MOV	ES,AX			
036D	C3	1172		RET				; FLAGS = NZ
		1173						
		1174	SWPG	ENDP				
		1175						
		1176						
		1177						
		1178						
		1179	VIDEQA	ENDS				
		1180						
		1181		END				

NAME	TYPE	VALUE	ATTRIBUTES, XREFS
DCI	V DWORD	0000H	EXTRN 235#
DFC	V DWORD	0000H	EXTRN 236#
DFC_COMPF	V WORD	0006H	S FIELD 343# 413 439
DFC_FINDX	V BYTE	0008H	S FIELD 344# 370 449
DFC_HORZ	V BYTE	000CH	S FIELD 348# 364
DFC_VERT	V BYTE	000AH	S FIELD 346# 365
DFC1	L NEAR	003AH	VIDEOA 392 395#
DFC2	L NEAR	0048H	VIDEOA 401 404#
DFC3	L NEAR	006AH	VIDEOA 420 424#
DFC4	L NEAR	0073H	VIDEOA 430 432#
DFC4_3	L NEAR	007AH	VIDEOA 386 437#
DFC4_6	L NEAR	008DH	VIDEOA 435 444#
DFC5	L NEAR	00AAH	VIDEOA 422 431 443 469# 495
DFC6	L NEAR	00E7H	VIDEOA 394 403 500# 517
DS_SIZE	NUMBER	0040H	63#
DSC	V BYTE	0001H	S FIELD 146#
EDC	V DWORD	0000H	EXTRN 238#
EDC_CHAR	V BYTE	0008H	S FIELD 546# 564
EDC_COUNT	V BYTE	0005H	S FIELD 544# 566
EDC_LINE	V BYTE	000AH	S FIELD 548# 563
EDC1	L NEAR	0126H	VIDEOA 573# 593
EDC2	L NEAR	013AH	VIDEOA 579 582#
EDC3	L NEAR	0146H	VIDEOA 586 589#
EDC4	L NEAR	016BH	VIDEOA 581 588 600 604 616# 627
EDC5	L NEAR	0173H	VIDEOA 621 623#
EMEC	V DWORD	0000H	EXTRN 239#
ESCP	V TBYTE	0000H	EXTRN 253# 1038
ESCP_STRUC	STRUC		SIZE=000AH #FIELDS=6 115 122# 253
EXPAND	V BYTE	0008H	S FIELD 133#
FONT	V DWORD	0000H	EXTRN 240# 414 440
FONTAB	SEGMENT		SIZE=0000H BYTE 'PUBLIC 'DATA' 180# 182
FORE	V BYTE	0000H	S FIELD 101#
FUNCTION	V WORD	0001H	S FIELD 117#
GRAPHIC	V BYTE	0009H	S FIELD 134#
H19_MODE	V 18	0000H	EXTRN 254# 384 705 828 1037
H19_MODE_STRUC	STRUC		SIZE=0012H #FIELDS=17 124 142# 254
H19_PAR	V 9	0000H	EXTRN 255# 824
H19_PAR_STRUC	STRUC		SIZE=0009H #FIELDS=9 144 154# 255
HORIZ_CHAR	V BYTE	0000H	EXTRN 230# 911
INDEX	V BYTE	0000H	S FIELD 74#
INIT	L NEAR	0000H	EXTRN 222#
INSERT	V BYTE	000AH	S FIELD 135#
ITV_OBI	V DWORD	0000H	EXTRN 189#
ITV_SST	V DWORD	0000H	EXTRN 188#
KEY_EN	V BYTE	0008H	S FIELD 136#
LFS	V BYTE	0002H	S FIELD 147#
MDC	V DWORD	0000H	EXTRN 241#
MDC_COUNT	V DWORD	0006H	S FIELD 659# 685
MDC_DST	V BYTE	0003H	S FIELD 661# 681
MDC_LINE	V BYTE	000CH	S FIELD 665# 680
MDC_SRC	V BYTE	000AH	S FIELD 663# 683
MDC1	L NEAR	0184H	VIDEOA 674 711# 722
MDC2	L NEAR	01D1H	VIDEOA 707 727#
MDC3	L NEAR	01D8H	VIDEOA 732# 737
MDC4	L NEAR	01E7H	VIDEOA 690 723 739#

NAME	TYPE	VALUE	ATTRIBUTES, XREFS
PAR_VID	NUMBER	C000H	67#
PFRONT	V BYTE	0005H	S FIELD 106# 410 928
PLM_TRUE	NUMBER	00F7H	S2# 385 706 829
PORT	V BYTE	0001H	S FIELD 95#
POVRAM	V BYTE	0003H	S FIELD 148#
PROMPT	V DWORD	0000H	EXTRN 243#
RDC	V DWORD	0000H	EXTRN 244#
RDC_HORZ	V BYTE	0006H	S FIELD 939# 1006
RDC_VERT	V BYTE	0008H	S FIELD 991# 1007
RDC_XC_PTR	V DWORD	0004H	S FIELD 993# 1011
REG	V DWORD	0000H	EXTRN 232#
RESETF	V BYTE	0000H	EXTRN 233#
REVERSE	V WORD	000CH	S FIELD 137#
ROW	V BYTE	0006H	S FIELD 161#
S_CRT	L NEAR	0000H	EXTRN 215#
S_KBD	L NEAR	0000H	EXTRN 216#
S_XMTC	L DWORD	0000H	EXTRN 245#
SDS	L NEAR	0000H	EXTRN 204#
SHIFTED	V BYTE	000EH	S FIELD 138#
SLI	V BYTE	0004H	S FIELD 149#
SLPC	NUMBER	000BH	275# 568 687 825 930
SPC	V BYTE	0005H	S FIELD 150#
SS_SIZE	NUMBER	0020H	62#
START	V WORD	0000H	S FIELD 111#
STATUS	V BYTE	000FH	S FIELD 139#
STRNG	V BYTE	0002H	S FIELD 96#
SW401	V BYTE	0005H	S FIELD 151#
SW402	V BYTE	0007H	S FIELD 152#
SWP	L NEAR	0324H	VIDEOA 391 400 419 429 434 578 585 598 752 872 929 1119# 1160
SWP1	L NEAR	032AH	VIDEOA 1121 1124#
SWPA	V BYTE	0348H	VIDEOA 1128 1140#
SWPB	V WORD	0350H	VIDEOA 1133 1151#
SWPG	L NEAR	0360H	VIDEOA 442 728 850 1004 1165# 1174
TTY_INTR	L NEAR	0000H	EXTRN 218#
TTY_POLL	L NEAR	0000H	EXTRN 219#
TXMT	L NEAR	0000H	EXTRN 217# 1056
UIES	V DWORD	0000H	EXTRN 246#
UNIT	V BYTE	0052H	S FIELD 97#
UPDATE	V BYTE	0002H	S FIELD 112#
UPDN	V BYTE	0010H	S FIELD 140#
VERT_LINE	V BYTE	0000H	EXTRN 231# 910
VID_CMD	NUMBER	00DSH	277# 359 459 558 607 675 741 811 862 905 941 1001 1015 1126 1131 1166 1168
VIDEO_INTR	L NEAR	0000H	EXTRN 212#
VIDEO_INTR_A	L FAR	0000H	EXTRN 174#
VIDEOA	SEGMENT		SIZE=036EH BYTE 'PUBLIC 'CODE' 264# 271 1179
VRAM_SIZE	V BYTE	0008H	S FIELD 153#
WRAP	V BYTE	0011H	S FIELD 141#
WRITEC	L NEAR	0000H	EXTRN 223# 921
XCA	V DWORD	0000H	EXTRN 247#
XMT	V TBYTE	0000H	EXTRN 256#
XMT_COLOR	V BYTE	0007H	S FIELD 162#
XMT_GRAPHIC	V BYTE	0008H	S FIELD 163#
XMT_REVERSE	V BYTE	0007H	S FIELD 164#
XMT_STRUC	STRUC		SIZE=000AH #FIELDS=8 156 165# 256

SERIES-III 8086/8087/8088 MACRO ASSEMBLER V1.0 ASSEMBLY OF MODULE B207A
OBJECT MODULE PLACED IN : F2:B207A.OBJ
INVOCATION LINE CONTROLS: PRINT(::TO:) XREF ERRORPRINT(::TO:)

LOC OBJ LINE SOURCE

LOC	OBJ	LINE	SOURCE	DB	?	;
0002		50	MSK	DB	?	; SHL(Back,3) OR Fore
0003		51	CLEAR	DB	?	; Color to Clear
0004		52	PAINTED	DB	?	; Color to Fully Paint
0005		53	PFONT	DB	?	; Color to set to Font Pattern
0006		54	COMP_FONT	DB	?	; Color to set to Complement of Font Pattern
		55	COLOR_STRUC	ENDS		
		56				
		57	CRTC_STRUC	STRUC		
0000		58	START	DW	?	; CRT-C..... Start Address
0002		59	UPDATE	DB	?	; True if update is required
		60	CRTC_STRUC	ENDS		
		61				
		62	ESCP_STRUC	STRUC		
0000		63	COMMAND	DB	?	; Current Escape Command
0001		64	FUNCTION	DW	?	; Pointer to Escape Processor
0003		65	MODE	DB	?	; Escape Emulation Mode
0004		66	OPER_COUNT	DB	?	; Operand Count
0005		67	OPER_INDEX	DB	?	; Operand Index
0006		68	OPERAND	DB	?	; Operands
		69	ESCP_STRUC	ENDS	4 DUP(?)	
		70				
		71	H19_MODE_STRUC	STRUC		
0000		72	ALTERNATE	DB	?	; Alternate Key-pad Mode
0001		73	ANSI	DB	?	; ANSI Mode
0002		74	AUTO_CR	DB	?	; Auto Carriage-Return on Line-Feed
0003		75	AUTO_LF	DB	?	; Auto Line-Feed on Carriage-Return
0004		76	AUTO_REPEAT	DB	?	; Auto-Repeat Keyboard
0005		77	BWD	DB	?	; Black and White Optimization
0006		78	CURSOR	DB	?	; Programmed Cursor Value
0007		79	CURSOR_ON	DB	?	; Cursor Enabled
0008		80	EXPAND	DB	?	; Expand Key-Board Characters
0009		81	GRAPHIC	DB	?	; Graphic Character Mode
000A		82	INSERT	DB	?	; Insert Character Mode
000B		83	KEY_EN	DB	?	; Key-Board Enable
000C		84	REVERSE	DW	?	; Reverse Video
000E		85	SHIFTED	DB	?	; Shifted Key-Pad Mode
000F		86	STATUS	DB	?	; Status-Line Enabled
0010		87	UPDN	DB	?	; Key-Board Up/Down Mode
0011		88	WRAP	DB	?	; Wrap at End-of-Line
		89	H19_MODE_STRUC	ENDS		
		90				
		91	H19_PAR_STRUC	STRUC		
0000		92	CPL	DB	?	; Characters Per Line
0001		93	DSC	DB	?	; Displayed Scan Lines per Character
0002		94	LPS	DB	?	; Lines Per Screen
0003		95	FOVRAM	DB	?	; Planes of Video RAM
0004		96	SLI	DB	?	; Status Line Index
0005		97	SPC	DB	?	; Scan Lines per Character
0006		98	SW401	DB	?	; H19-Switch 401
0007		99	SW402	DB	?	; H19-Switch 402
0008		100	VRAM_SIZE	DB	?	; 0-32KBytes, 1-64KBytes
		101	H19_PAR_STRUC	ENDS		
		102				
		103	XMT_STRUC	STRUC		
0000		104	BURST	DB	?	; Characters to Transmit per VSYNC

LOC	OBJ	LINE	SOURCE
=1			EXTRN MTR101:NEAR ; PL/M-86 Monitor Loop
=1			EXTRN VIDE0_INTR:NEAR ; PL/M-86 Video Interrupt
=1			EXTRN D_CRTI:NEAR ; PL/M-86 Dumb Terminal
=1			EXTRN D_KBD:NEAR ; PL/M-86 Dumb Key-Board
=1			EXTRN S_CRT:NEAR ; PL/M-86 Smart Terminal
=1			EXTRN S_KBD:NEAR ; PL/M-86 Smart Key-Board
=1			EXTRN TXMT:NEAR ; PL/M-86 Transmit Screen Character
=1			EXTRN TTY_INTR:NEAR ; PL/M-86 Terminal Interrupt
=1			EXTRN TTY_POLL:NEAR ; PL/M-86 Interrupt Poll
=1			EXTRN CRLE:NEAR ; PL/M-86 CR-LF
=1			EXTRN INIT:NEAR ; PL/M-86 Initialize Hardware
=1			EXTRN WRITEC:NEAR ; PL/M-86 Write Character
=1			ENDS
=1			CODE
=1)FI
169			
=1		170	
=1		171	
=1			%IF(%NES(DATA,%SEGMENT_LABEL)) THEN (
=1			DATA
=1			SEGMENT WORD PUBLIC 'DATA'
=1			HORZ_CHAR:BYTE ; Column Index
=1			VERT_LINE:BYTE ; Row Index
=1			RESETF:BYTE ; Hardware Reset Flag
=1			EXTRN DCI:DWORD ; Display Character Initialization
=1			EXTRN DFC:DWORD ; Display Font Character
=1			EXTRN D_XMTC:DWORD ; Dumb Terminal Transmit Character
=1			EXTRN EDC:DWORD ; Erase Display Character
=1			EXTRN EMEC:DWORD ; Extend-Mode Escape Character
=1			EXTRN FONT:DWORD ; Pointer to Font Table
=1			EXTRN MDC:DWORD ; Move Display Character
=1			EXTRN MDL:DWORD ; Move Display Line
=1			EXTRN PROMPT:DWORD ; Display Monitor Prompt
=1			EXTRN RDC:DWORD ; Read Display Character
=1			EXTRN S_XMTC:DWORD ; Smart Terminal Transmit Character
=1			EXTRN UIES:DWORD ; Un-Implemented Escape Sequence
=1			EXTRN XCA:DWORD ; Transmit Character Attributes
=1			EXTRN BOOT_PAR:BOOT_PAR_STRUC
=1			EXTRN COLOR:COLOR_STRUC
=1			EXTRN CRIC_CURSOR:CRIC_STRUC
=1			EXTRN CRIC_DISPLAY:CRIC_STRUC
=1			EXTRN ESCP:ESCP_STRUC
=1			EXTRN HI9_MODE:HI9_MODE_STRUC
=1			EXTRN HI9_PAR:HI9_PAR_STRUC
=1			EXTRN XMT:XMT_STRUC
=1			ENDS
=1			DATA
=1)FI
202			
=1		203	
=1		204	
=1			%IF(%NES(VIDE0,%SEGMENT_LABEL)) THEN (
=1			VIDE0
=1			SEGMENT BYTE PUBLIC 'CODE'
=1			VIDE0
=1)FI
208			

LOC	OBJ	LINE	SOURCE
		242	;
		243	;
		244	;
		245	;
		246	;
		247	;
		248	;
		249	;
		250	;
		251	;
		252	;
0000		253	CSP
		254	PROD NEAR
		255	PUBLIC CSP
0000	8A160100	256	MOV DL,BOOT_PAK,PORT
0004	8600	257	MOV DH,0 ; DX = Device Port
0005	EC	258	IN AL,DX
0007	C3	259	RET
		260	
		261	CSP ENDP
		262	+1 \$EJECT

Controller Status Port
 CSP returns the value for the controller status port

Entry: NONE
 Exit: AL = Controller Status Port Value
 Uses: AL, DH

PROD NEAR
 PUBLIC CSP

MOV DL,BOOT_PAK,PORT
 MOV DH,0 ; DX = Device Port
 IN AL,DX
 RET

CSP ENDP
 +1 \$EJECT

LOC	OBJ	LINE	SOURCE
		300	;;
		301	RDTRACKA -- Read Track
		302	;
		303	RDTRACKA is the assembly language assist routine
		304	for RDTRACK. This routine is required to perform
		305	the actual disk transfer, as the code may be opt-
		306	imized.
		307	;
		308	Usage: CALL RDTRACKA(base_port,cmd,cnt,ptr\$data)
		309	;
		310	Entry: base_port = Base Port Address for device
		311	cmd = command
		312	cnt = Byte Count
		313	ptr\$data = Address to save data at
		314	;
		315	Exit: AX = Byte Count
		316	;
		317	Uses: ???
		318	;
		319	;
		320	RDTRACKA STRUC
		321	DW ?
		322	DW ?
		323	DD ? ; Address to save data at
		324	DW ? ; Byte Count
		325	DW ? ; Command
		326	DW ? ; Base Port Address for device
		327	RDTRACKA ENDS
		328	;
		329	;
0019		330	RDTRACKA PROC NEAR
		331	PUBLIC RDTRACKA
		332	;
0019 55		333	PUSH BP
001A 8BEC		334	MOV BP,SP
		335	;
001C 8B560C		336	MOV DX,[BP],BASE_PORT
001F 8B460A		337	MOV AX,[BP],CMD
0022 EE		338	OUT DX,AL
		339	;
		340	CLD
0023 FC		341	XOR BX,BX
0024 33DB		342	MOV CX,[BP],CNT
0026 8B4E08		343	LES DI,[BP],PTR_DATA
0029 C47E04		344	ADD DX,DI
002C 83C203		345	IN AL,DX
002F EC		346	STOSB
0030 AA		347	INC BX
0031 43		348	LOOP RDT1
0032 E2FB		349	;
		350	MOV AX,BX
0034 8BC3		351	POP BP
0036 5D		352	RET 10
0037 C20A00		353	;
		354	RDTRACKA ENDP

```

; Set auto-increment
; BX = 0
; CX = Byte Count
; ES:DI = Address to store data at
; DX = Data Port Address
; Input Data Byte
; Read another sector
; AX = Actual read count
; Discard the parameters
    
```

LOC	OBJ	LINE	SOURCE
003A	BR581B70	360	MDR - Wait for Drive Ready
003E	B9041E	361	MDR waits until the selected drive is ready by waiting
0041	E81600	362	for at least 7 index hole transitions.
0044	720D	363	Entry: NONE
0046	75F9	364	Exit: TIME_OUT = results of loop time-out.
0048	E80F00	365	Uses: NONE
004B	7206	366	
004D	74F9	367	
004F	FEC9	368	
0051	75EE	369	
0053	0ADD	370	
0055	891E0000	371	
0057	C3	372	
1B58		373	
005A		374	
005A	4B	375	
005B	7507	376	
005D	FEC0	377	
005F	7414	378	
0061	BR581B	379	
0064	53	380	
0065	51	381	
0066	E80000	382	
0067	59	383	
0069		384	
006A		385	
006B		386	
006C		387	
006D		388	
006E		389	
006F		390	
0070		391	
0071		392	
0072		393	
0073		394	
0074		395	
0075		396	
0076		397	
0077		398	
0078		399	
0079		400	
0080		401	
0081		402	
0082		403	
0083		404	
0084		405	
0085		406	
0086		407	
0087		408	
0088		409	
0089		410	
0090		411	
0091		412	
0092		413	
0093		414	

LOC	OBJ	LINE	SOURCE
		432	WNB ; Wait for NOT Busy
		433	;
		434	;
		435	WNB waits for the controller to deassert busy. This is
		436	necessary before any valid commands are sent. The only
		437	exception is the force interrupt command, which naturally
		438	is sent when the device is busy.
		439	;
		440	Entry: NONE
		441	;
		442	Exit: NONE
		443	;
		444	Uses: ???----(because CBA may use a11)
		445	;
0077		446	WNB PROC NEAR
		447	PUBLIC WNB
		448	
0077	ESGAF	449	CALL CSP
007A	AS01	450	TEST AL, JNS, BSY
007C	7407	451	JZ WNB1 ; Busy is de-asserted
		452	
007E	ES0000	453	CALL CBA
0081	D0D8	454	ROR AL, 1
0083	73F2	455	WNB ; The user has not aborted yet
		456	
0085	C3	457	WNB1: RET
		458	
		459	WNB ENDP
		460	
		461	
		462	
		463	
		464	ASTLIB ENDS
		465	
		466	END

NAME	TYPE	VALUE	ATTRIBUTES, XREFS
EXPAND.	V BYTE	0000H	S FIELD 80#
FD_CNND	NUMBER	0000H	225#
FD_DATA	NUMBER	0003H	228# 344
FD_SECT	NUMBER	0002H	227#
FD_STAT	NUMBER	0000H	224#
FD_TRACK	NUMBER	0001H	226#
FONT.	V DWORD	0000H	EXTRN 183#
FONTAR.	SEGMENT		SIZE=0000H BYTE PUBLIC 'DATA' 123# 125
FORE.	V BYTE	0000H	S FIELD 48#
FUNCTION.	V WORD	0001H	S FIELD 64#
GRAPHIC.	V BYTE	0009H	S FIELD 81#
H19_MODE.	V 18	0000H	EXTRN 177#
H19_MODE_STRUC.	STRUC		SIZE=0012H #FIELDS=17 71 89# 197
H19_PAR.	V	0000H	EXTRN 178#
H19_PAR_STRUC.	STRUC		SIZE=0009H #FIELDS=9 91 101# 193
HORZ_CHAR	V BYTE	0000H	EXTRN 173#
INDEX.	V BYTE	0000H	S FIELD 41#
INIT.	L NEAR	0000H	EXTRN 165#
INSERT.	V BYTE	000AH	S FIELD 82#
ITV_ORI	V DWORD	0000H	EXTRN 132#
ITV_SST	V DWORD	0000H	EXTRN 131#
KEY_EN.	V BYTE	000BH	S FIELD 83#
LPS.	V BYTE	0002H	S FIELD 94#
MUC.	V DWORD	0000H	EXTRN 184#
MDL.	V DWORD	0000H	EXTRN 185#
MODE.	V BYTE	0003H	S FIELD 65#
MONENT.	SEGMENT		SIZE=0000H PARA PUBLIC 136# 144
MSK.	V BYTE	0002H	S FIELD 50#
MTR_DGRT.	L FAR	0000H	EXTRN 140#
MTR_DKBD.	L FAR	0000H	EXTRN 142#
MTR_FONT.	V BYTE	0000H	EXTRN 124#
MTR_MON.	L FAR	0000H	EXTRN 138#
MTR_RES.	L FAR	0000H	EXTRN 137#
MTR_SCR.	L FAR	0000H	EXTRN 141#
MTR_SKBD.	L FAR	0000H	EXTRN 143#
MTR_SWIM.	L FAR	0000H	EXTRN 139#
MTR100.	SEGMENT		SIZE=0000H PARA PUBLIC 'CODE' 145# 143 210
MTR101.	L NEAR	0000H	EXTRN 154#
OPER_COUNT.	V BYTE	0004H	S FIELD 66#
OPER_INDEX.	V BYTE	0005H	S FIELD 67#
OPERAND	V BYTE	0006H	S FIELD 68#
PAINTED	V BYTE	0004H	S FIELD 52#
PRINT.	V BYTE	0005H	S FIELD 53#
PLM_THUE.	NUMBER	00FH	29#
PORT.	V BYTE	0001H	S FIELD 42# 256
POVRAM.	V BYTE	0003H	S FIELD 95#
PROMPT.	V DWORD	0000H	EXTRN 186#
PTR_DATA.	V DWORD	0004H	S FIELD 323# 343
RDC.	V DWORD	0000H	EXTRN 187#
RDT1.	L NEAR	002FH	ASTLIB 345# 348
RDTA.	STRUC		SIZE=000EH #FIELDS=6 320 327#
RDTACKA.	L NEAR	0019H	ASTLIB PUBLIC 330# 331 334
RECP.	V DWORD	0000H	EXTRN 175#
RESETF.	V BYTE	0000H	EXTRN 176#
REVERSE	V WORD	000CH	S FIELD 84#

SERIES-III 8086/386/387/388 MACRO ASSEMBLER V1.0 ASSEMBLY OF MODULE FONT
OBJECT MODULE PLACED IN :F2:FONT2.OBJ
INVOCATION LINE CONTROLS: PRINT(:TO:) XREF ERRORPRINT(:TO:)

LOC OBJ LINE SOURCE

LOC OBJ LINE SOURCE

```

51 ; *****
52 ; *****
53 ; *****
54 ; *****
55 ; *****
56 ; *****
57 ; *****
58 ; *****
59 ; *****
60 ; *****
61 ; *****
62 ; *****
63 ; *****
64 ; *****
65 ; *****
66 ; *****
67 ; *****
68 ; *****
69 ; *****
70 ; *****
71 ; *****
72 ; *****
73 ; *****
74 ; *****
75 ; *****
76 ; *****
77 ; *****
78 ; *****
79 ; *****
80 ; *****
81 ; *****
82 ; *****
83 ; *****
84 ; *****
85 ; *****
86 +1 $EJECT

```

Basically, each of the characters consists of a ten element array which is to be displayed on the screen. Most of the work here was done by Steve Parker for the CP/M 8086 assembler. Some of the descenders have been modified ('p', 'q', and 'g'). The format has been modified for Intel Assembler compatibility.

G. Chandler

```

%*DEFINE(GRAPHIC)(0) %' No Graphic Characters
%*DEFINE(LOWER)(1) %' Lower Case

```

FONTAB SEGMENT BYTE PUBLIC 'DATA'

NAME FONT

```

MTR_FONT LABEL BYTE ; Base Address of Font ROM
PUBLIC MTR_FONT ; Font ROM is a global

```

```

00000
86 +1 $EJECT

```

LOC	OBJ	LINE	SOURCE
0030	04	142	DB 00000100B
0031	08	143	DB 00001000B
0032	10	144	DB 00010000B
0033	26	145	DB 00100100B
0034	06	146	DB 00000110B
0035	00	147	DB 00000000B
		148	;"
0036	00	149	DB 00000000B
0037	08	150	DB 00001000B
0038	14	151	DB 00010100B
0039	14	152	DB 00010100B
003A	18	153	DB 00011000B
003B	2A	154	DB 00101010B
003C	24	155	DB 00100100B
003D	1A	156	DB 00011010B
003E	00	157	DB 00000000B
		158	;"
003F	00	159	DB 00000000B
0040	0C	160	DB 00001000B
0041	03	161	DB 00001000B
0042	10	162	DB 00010000B
0043	00	163	DB 00000000B
0044	00	164	DB 00000000B
0045	00	165	DB 00000000B
0046	00	166	DB 00000000B
0047	00	167	DB 00000000B
		168	;"
0048	00	169	DB 00000000B
0049	04	170	DB 00001000B
004A	08	171	DB 00001000B
004B	10	172	DB 00010000B
004C	10	173	DB 00010000B
004D	10	174	DB 00010000B
004E	03	175	DB 00001000B
004F	04	176	DB 00001000B
0050	00	177	DB 00000000B
		178	;"
0051	00	179	DB 00000000B
0052	10	180	DB 00010000B
0053	08	181	DB 00001000B
0054	04	182	DB 00000100B
0055	04	183	DB 00000100B
0056	04	184	DB 00000100B
0057	03	185	DB 00001000B
0058	10	186	DB 00010000B
0059	00	187	DB 00000000B
		188	;"
005A	00	189	DB 00000000B
005B	00	190	DB 00000000B
005C	08	191	DB 00001000B
005D	2A	192	DB 00101010B
005E	1C	193	DB 00011100B
005F	2A	194	DB 00101010B
0060	08	195	DB 00001000B
0061	00	196	DB 00000000B

LOC	OBJ	LINE	SOURCE
0093	26	252	DB 00100110B
0094	2A	253	DB 00101010B
0095	32	254	DB 00110010B
0096	22	255	DB 00100010B
0097	1C	256	DB 00011100B
0098	00	257	DB 00000000B
0099	00	258	DB "1"
009A	08	259	DB 00000000B
009B	18	260	DB 00001000B
009C	08	261	DB 00011000B
009D	08	262	DB 00001000B
009E	08	263	DB 00001000B
009F	08	264	DB 00001000B
00A0	1C	265	DB 00001000B
00A1	00	266	DB 00011100B
00A2	00	267	DB 00000000B
00A3	1C	268	DB "2"
00A4	22	269	DB 00000000B
00A5	02	270	DB 00011100B
00A6	04	271	DB 00100010B
00A7	08	272	DB 00000010B
00A8	10	273	DB 00000100B
00A9	3E	274	DB 00001000B
00AA	00	275	DB 00010000B
00AB	00	276	DB 00111110B
00AC	3E	277	DB 00000000B
00AD	04	278	DB "3"
00AE	08	279	DB 00000000B
00AF	04	280	DB 00111110B
00B0	02	281	DB 00000100B
00B1	22	282	DB 00001000B
00B2	1C	283	DB 00000100B
00B3	00	284	DB 00000010B
00B4	00	285	DB 00100010B
00B5	04	286	DB 00011100B
00B6	0C	287	DB 00000000B
00B7	14	288	DB "4"
00B8	24	289	DB 00000000B
00B9	3E	290	DB 00000100B
00BA	04	291	DB 00001100B
00BB	04	292	DB 00010100B
00BC	00	293	DB 00100100B
00BD	00	294	DB 00111110B
00BE	3E	295	DB 00000100B
00BF	20	296	DB 00000100B
00C0	3C	297	DB 00000000B
00C1	02	298	DB "5"
00C2	02	299	DB 00000000B
00C3	22	300	DB 00111110B
00C4	1C	301	DB 00100000B
		302	DB 00100000B
		303	DB 00111100B
		304	DB 00000010B
		305	DB 00000010B
		306	DB 00100010B
			DB 00011100B

LOC	OBJ	LINE	SOURCE
00F6	18	362	DB 00011000B
00F7	00	363	DB 00000000B
00F8	18	364	DB 00011000B
00F9	18	365	DB 00011000B
00FA	08	366	DB 00001000B
00FB	10	367	DB 00010000B
00FC	00	368	DB "<"
00FD	02	369	DB 00000000B
00FE	04	370	DB 00000010B
00FF	08	371	DB 00000100B
0100	10	372	DB 00001000B
0101	08	373	DB 00010000B
0102	04	374	DB 00001000B
0103	02	375	DB 00000100B
0104	00	376	DB 00000010B
0105	00	377	DB 00000000B
0106	00	378	DB "="
0107	00	379	DB 00000000B
0108	00	380	DB 00000000B
0109	00	381	DB 00000000B
010A	00	382	DB 00000000B
010B	00	383	DB 00111110B
010C	00	384	DB 00000000B
010D	00	385	DB 00111110B
010E	00	386	DB 00000000B
010F	20	387	DB 00000000B
0110	10	388	DB ">"
0111	08	389	DB 00000000B
0112	04	390	DB 00100000B
0113	08	391	DB 00010000B
0114	10	392	DB 00001000B
0115	20	393	DB 00000100B
0116	00	394	DB 00001000B
0117	00	395	DB 00010000B
0118	1C	396	DB 00100000B
0119	22	397	DB 00000000B
011A	02	398	DB "?"
011B	04	399	DB 00000000B
011C	08	400	DB 00011100B
011D	00	401	DB 00100010B
011E	08	402	DB 00000010B
011F	00	403	DB 00000100B
0120	00	404	DB 00001000B
0121	0C	405	DB 00000000B
0122	12	406	DB 00001000B
0123	26	407	DB 00000000B
0124	2A	408	DB "e"
0125	2E	409	DB 00000000B
0126	20	410	DB 00001100B
0127	1E	411	DB 00010010B
		412	DB 00100110B
		413	DB 00401010B
		414	DB 00101110B
		415	DB 00100000B
		416	DB 00011110B

LOC	OBJ	LINE	SOURCE
0159	20	472	DB 00100000B
015A	3C	473	DB 00111100B
015B	20	474	DB 00100000B
015C	20	475	DB 00100000B
015D	20	476	DB 00100000B
015E	00	477	DB 00000000B
015F	00	478	DB "G"
0160	1C	479	DB 00000000B
0161	22	480	DB 00011100B
0162	20	481	DB 00100010B
0163	26	482	DB 00100000B
0164	22	483	DB 00100110B
0165	22	484	DB 00100010B
0166	1E	485	DB 00100010B
0167	00	486	DB 00011110B
0168	00	487	DB 00000000B
0169	22	488	DB "H"
016A	22	489	DB 00000000B
016B	22	490	DB 00100010B
016C	3E	491	DB 00100010B
016D	22	492	DB 00100010B
016E	22	493	DB 00111110B
016F	22	494	DB 00100010B
0170	00	495	DB 00100010B
0171	00	496	DB 00100010B
0172	1C	497	DB 00000000B
0173	08	498	DB "I"
0174	08	499	DB 00000000B
0175	08	500	DB 00011100B
0176	08	501	DB 00001000B
0177	08	502	DB 00001000B
0178	1C	503	DB 00001000B
0179	00	504	DB 00001000B
017A	00	505	DB 00001000B
017B	0E	506	DB 00011100B
017C	04	507	DB 00000000B
017D	04	508	DB "J"
017E	04	509	DB 00000000B
017F	04	510	DB 00001100B
0180	24	511	DB 00000100B
0181	18	512	DB 00000100B
0182	00	513	DB 00000100B
0183	00	514	DB 00000100B
0184	22	515	DB 00000100B
0185	24	516	DB 00011000B
0186	28	517	DB 00100000B
0187	30	518	DB "K"
0188	28	519	DB 00000000B
0189	24	520	DB 00100010B
018A	22	521	DB 00100100B
		522	DB 00101000B
		523	DB 00110000B
		524	DB 00101000B
		525	DB 00100100B
		526	DB 00100010B

LDC OBJ	LINE	SOURCE
01BC 22	582	DB 00100010B
01BD 22	583	DB 00100010B
01BE 2A	584	DB 00101010B
01BF 24	585	DB 00100100B
01C0 1A	586	DB 00011010B
01C1 00	587	DB 00000000B
	588	"R"
01C2 00	589	DB 00000000B
01C3 3C	590	DB 00111100B
01C4 22	591	DB 00100010B
01C5 22	592	DB 00100010B
01C6 3C	593	DB 00111100B
01C7 28	594	DB 00101000B
01C8 24	595	DB 00100100B
01C9 22	596	DB 00100010B
01CA 00	597	DB 00000000B
	598	"S"
01CB 00	599	DB 00000000B
01CC 1C	600	DB 00011100B
01CD 22	601	DB 00100010B
01CE 20	602	DB 00100000B
01CF 1C	603	DB 00011100B
01D0 02	604	DB 00000010B
01D1 22	605	DB 00100010B
01D2 1C	606	DB 00011100B
01D3 00	607	DB 00000000B
	608	"T"
01D4 00	609	DB 00000000B
01D5 3E	610	DB 00111110B
01D6 08	611	DB 00001000B
01D7 08	612	DB 00001000B
01D8 08	613	DB 00001000B
01D9 08	614	DB 00001000B
01DA 08	615	DB 00001000B
01DB 08	616	DB 00001000B
01DC 00	617	DB 00000000B
	618	"U"
01DD 00	619	DB 00000000B
01DE 22	620	DB 00100010B
01DF 22	621	DB 00100010B
01E0 22	622	DB 00100010B
01E1 22	623	DB 00100010B
01E2 22	624	DB 00100010B
01E3 22	625	DB 00100010B
01E4 1C	626	DB 00011100B
01E5 00	627	DB 00000000B
	628	"V"
01E6 00	629	DB 00000000B
01E7 22	630	DB 00100010B
01E8 22	631	DB 00100010B
01E9 22	632	DB 00100010B
01EA 14	633	DB 00010100B
01EB 14	634	DB 00010100B
01EC 08	635	DB 00001000B
01ED 08	636	DB 00001000B

LOC	OBJ	LINE	SOURCE
021F	10	692	DB 00010000B
0220	08	693	DB 00001000B
0221	04	694	DB 00000100B
0222	02	695	DB 00000010B
0223	01	696	DB 00000001B
0224	00	697	DB 00000000B
0225	00	698	DB "J"
0226	38	699	DB 00000000B
0227	08	700	DB 00111000B
0228	08	701	DB 00001000B
0229	08	702	DB 00001000B
022A	08	703	DB 00001000B
022B	08	704	DB 00001000B
022C	38	705	DB 00001000B
022D	00	706	DB 00111000B
022E	00	707	DB 00000000B
022F	08	708	DB "A"
0230	14	709	DB 00000000B
0231	22	710	DB 00001000B
0232	00	711	DB 00010100B
0233	00	712	DB 00100010B
0234	00	713	DB 00000000B
0235	00	714	DB 00000000B
0236	00	715	DB 00000000B
0237	00	716	DB 00000000B
0238	00	717	DB " "
0239	00	718	DB " "
023A	00	719	DB 00000000B
023B	00	720	DB 00000000B
023C	00	721	DB 00000000B
023D	00	722	DB 00000000B
023E	00	723	DB 00000000B
023F	00	724	DB 00000000B
0240	00	725	DB 00000000B
0241	18	726	DB 01111111B
0242	08	727	DB 00000000B
0243	04	728	DB " "
0244	00	729	DB 00000000B
0245	00	730	DB 00011000B
0246	00	731	DB 00001000B
0247	00	732	DB 00000100B
0248	00	733	DB 00000000B
0249	00	734	DB 00000000B
024A	00	735	DB 00000000B
024B	00	736	DB 00000000B
024C	00	737	DB 00000000B
024D	00	738	DB 00000000B
024E	00	739	DB 00011000B
024F	00	740	DB 00001000B
0250	00	741	DB 00000100B
0251	00	742	DB 00000000B
0252	00	743	DB 00000000B
0253	00	744	DB 00000000B
0254	00	745	DB 00011100B
0255	00	746	DB 00000010B

```

%IF (XLMER
  740 +1
  741 "a"
  742 ;
  743 ;
  744 ;
  745 ;
  746 ;

```

LOC	OBJ	LINE	SOURCE
		802	DB 00000000B
		803	DB 00000000B
		804	DB 00000000B
		805	DB 00011110B
		806	DB 00100100B
		807	DB 00111000B
		808	DB 00011100B
		809	DB 00100010B
		810	DB 00011100B
		811	; "h"
		812	DB 00000000B
		813	DB 00100000B
		814	DB 00100000B
		815	DB 00111100B
		816	DB 00100010B
		817	DB 00100010B
		818	DB 00100010B
		819	DB 00100010B
		820	DB 00000000B
		821	; "i"
		822	DB 00000000B
		823	DB 00001000B
		824	DB 00000000B
		825	DB 00011000B
		826	DB 00001000B
		827	DB 00001000B
		828	DB 00001000B
		829	DB 00011100B
		830	DB 00000000B
		831	; "j"
		832	DB 00000000B
		833	DB 00000010B
		834	DB 00000000B
		835	DB 00000010B
		836	DB 00000010B
		837	DB 00000010B
		838	DB 00000010B
		839	DB 00100010B
		840	DB 00011100B
		841	; "k"
		842	DB 00000000B
		843	DB 00100000B
		844	DB 00100000B
		845	DB 00100100B
		846	DB 00101000B
		847	DB 00110100B
		848	DB 00100010B
		849	DB 00100010B
		850	DB 00000000B
		851	; "l"
		852	DB 00000000B
		853	DB 00011000B
		854	DB 00001000B
		855	DB 00001000B
		856	DB 00001000B

LOC	OBJ	LINE	SOURCE
		912	DB 00000000B
		913	DB 00000000B
		914	DB 00000000B
		915	DB 00101100B
		916	DB 00110010B
		917	DB 00100000B
		918	DB 00100000B
		919	DB 00100000B
		920	DB 00000000B
		921	; "s"
		922	DB 00000000B
		923	DB 00000000B
		924	DB 00000000B
		925	DB 00011100B
		926	DB 00100000B
		927	DB 00011100B
		928	DB 00000010B
		929	DB 00011100B
		930	DB 00000000B
		931	; "t"
		932	DB 00000000B
		933	DB 00010000B
		934	DB 00010000B
		935	DB 00111000B
		936	DB 00010000B
		937	DB 00010000B
		938	DB 00010010B
		939	DB 00001100B
		940	DB 00000000B
		941	; "u"
		942	DB 00000000B
		943	DB 00000000B
		944	DB 00000000B
		945	DB 00100010B
		946	DB 00100010B
		947	DB 00100010B
		948	DB 00100110B
		949	DB 00011010B
		950	DB 00000000B
		951	; "v"
		952	DB 00000000B
		953	DB 00000000B
		954	DB 00000000B
		955	DB 00100010B
		956	DB 00100010B
		957	DB 00100010B
		958	DB 00010100B
		959	DB 00001000B
		960	DB 00000000B
		961	; "w"
		962	DB 00000000B
		963	DB 00000000B
		964	DB 00000000B
		965	DB 00100010B
		966	DB 00100010B

LOC	OBJ	LINE	SOURCE
0249	00	1022	DB 00000000B
024A	00	1023	DB 00011000B
024B	00	1024	DB 000000100B
024C	1C	1025	DB 00000100B
024D	02	1026	DB 000000010B
024E	1E	1027	DB 00000100B
024F	22	1028	DB 000000100B
0250	1E	1029	DB 00011000B
0251	00	1030	DB 00000000B
		1031	; "v"
		1032	DB 00000000B
		1033	DB 00110000B
		1034	DB 01001001B
		1035	DB 00000110B
		1036	DB 00000000B
		1037	DB 00000000B
		1038	DB 00000000B
		1039	DB 00000000B
		1040	DB 00000000B
		1041	DB 00000000B
		1042);F1
		1043	+1
		1044	; "a"
		1044	DB 00000000B
		1045	DB 00000000B
		1046	DB 00000000B
		1047	DB 00011100B
		1048	DB 00000010B
		1049	DB 00011110B
		1050	DB 00100010B
		1051	DB 00011110B
		1052	DB 00000000B
		1053	; "b"
		1054	DB 00000000B
		1055	DB 00100000B
		1056	DB 00100000B
		1057	DB 00111100B
		1058	DB 00100010B
		1059	DB 00100010B
		1060	DB 00100010B
		1061	DB 00111100B
		1062	DB 00000000B
		1063	; "c"
		1064	DB 00000000B
		1065	DB 00000000B
		1066	DB 00000000B
		1067	DB 00011100B
		1068	DB 00100010B
		1069	DB 00100000B
		1070	DB 00100000B
		1071	DB 00011100B
		1072	DB 00000000B
		1073	; "d"
		1074	DB 00000000B
		1075	DB 00000010B
		1076	DB 00000010B

LOC	OBJ	LINE	SOURCE
02979	00	1132 +1	DB 00000000B
02979	00	1133 +1	; "j"
02979	00	1134 +1	DB 00000000B
02979	00	1135 +1	DB 000000010B
02979	00	1136 +1	DB 000000000B
02979	00	1137 +1	DB 000000010B
02979	00	1138 +1	DB 000000010B
02979	00	1139 +1	DB 000000010B
02979	00	1140 +1	DB 000000010B
02979	00	1141 +1	DB 00100010B
02979	00	1142 +1	DB 00011100B
02979	00	1143 +1	; "k"
02979	00	1144 +1	DB 00000000B
02979	00	1145 +1	DB 00100000B
02979	00	1146 +1	DB 00100000B
02979	00	1147 +1	DB 00100100B
02979	00	1148 +1	DB 00101000B
02979	00	1149 +1	DB 00110100B
02979	00	1150 +1	DB 00100010B
02979	00	1151 +1	DB 00100010B
02979	00	1152 +1	DB 00000000B
02979	00	1153 +1	; "l"
02979	00	1154 +1	DB 00000000B
02979	00	1155 +1	DB 00011000B
02979	00	1156 +1	DB 000001000B
02979	00	1157 +1	DB 000001000B
02979	00	1158 +1	DB 000001000B
02979	00	1159 +1	DB 000001000B
02979	00	1160 +1	DB 000001000B
02979	00	1161 +1	DB 00011100B
02979	00	1162 +1	DB 00000000B
02979	00	1163 +1	; "m"
02979	00	1164 +1	DB 00000000B
02979	00	1165 +1	DB 00000000B
02979	00	1166 +1	DB 000000000B
02979	00	1167 +1	DB 00110100B
02979	00	1168 +1	DB 00101010B
02979	00	1169 +1	DB 00101010B
02979	00	1170 +1	DB 00101010B
02979	00	1171 +1	DB 00101010B
02979	00	1172 +1	DB 00000000B
02979	00	1173 +1	; "n"
02979	00	1174 +1	DB 00000000B
02979	00	1175 +1	DB 00000000B
02979	00	1176 +1	DB 00000000B
02979	00	1177 +1	DB 00111100B
02979	00	1178 +1	DB 00100010B
02979	00	1179 +1	DB 00100010B
02979	00	1180 +1	DB 00100010B
02979	00	1181 +1	DB 00100010B
02979	00	1182 +1	DB 00000000B
02979	00	1183 +1	; "o"
02979	00	1184 +1	DB 00000000B
02979	00	1185 +1	DB 00000000B
02979	00	1186 +1	DB 00000000B

LOC	OBJ	LINE	SOURCE
02FC	00	1242 +1	DB 00000000B
02FD	00	1243 +1	; "u"
02FE	00	1244 +1	DB 00000000B
02FF	00	1245 +1	DB 00000000B
0300	22	1246 +1	DB 00000000B
0301	22	1247 +1	DB 00100010B
0302	22	1248 +1	DB 00100010B
0303	26	1249 +1	DB 00100010B
0304	1A	1250 +1	DB 00100110B
0305	00	1251 +1	DB 00011010B
0306	00	1252 +1	DB 00000000B
0307	00	1253 +1	; "v"
0308	00	1254 +1	DB 00000000B
0309	22	1255 +1	DB 00000000B
030A	22	1256 +1	DB 00000000B
030B	22	1257 +1	DB 00100010B
030C	14	1258 +1	DB 00100010B
030D	08	1259 +1	DB 00100010B
030E	00	1260 +1	DB 00010100B
030F	00	1261 +1	DB 00001000B
0310	00	1262 +1	DB 00000000B
0311	00	1263 +1	; "w"
0312	22	1264 +1	DB 00000000B
0313	22	1265 +1	DB 00000000B
0314	2A	1266 +1	DB 00000000B
0315	2A	1267 +1	DB 00100010B
0316	14	1268 +1	DB 00100010B
0317	00	1269 +1	DB 00101010B
0318	00	1270 +1	DB 00010100B
0319	00	1271 +1	DB 00010100B
031A	00	1272 +1	DB 00000000B
031B	22	1273 +1	; "x"
031C	14	1274 +1	DB 00000000B
031D	08	1275 +1	DB 00000000B
031E	14	1276 +1	DB 00000000B
031F	22	1277 +1	DB 00100010B
0320	00	1278 +1	DB 00010100B
0321	00	1279 +1	DB 00001000B
0322	00	1280 +1	DB 00010100B
0323	00	1281 +1	DB 00100010B
0324	22	1282 +1	DB 00000000B
0325	22	1283 +1	; "y"
0326	22	1284 +1	DB 00000000B
0327	1E	1285 +1	DB 00000000B
0328	02	1286 +1	DB 00000000B
0329	1C	1287 +1	DB 00100010B
032A	00	1288 +1	DB 00100010B
032B	00	1289 +1	DB 00100010B
032C	00	1290 +1	DB 00100010B
		1291 +1	DB 00011110B
		1292 +1	DB 00000010B
		1293 +1	DB 00011100B
		1294 +1	; "z"
		1295 +1	DB 00000000B
		1296 +1	DB 00000000B

LOC	OBJ	LINE	SOURCE
		1352	DB 00111110B
		1353	DB 00111110B
		1354	DB 00011100B
		1355	DB 00000000B
		1356	DB 00000000B
		1357	; graphic " _ "
		1358	DB 11111111B
		1359	DB 01111111B
		1360	DB 00111111B
		1361	DB 00011111B
		1362	DB 00001111B
		1363	DB 00000111B
		1364	DB 00000011B
		1365	DB 00000001B
		1366	DB 00000000B
		1367	; graphic " . "
		1368	DB 00011000B
		1369	DB 00011000B
		1370	DB 00011000B
		1371	DB 00011000B
		1372	DB 00011000B
		1373	DB 00011000B
		1374	DB 00011000B
		1375	DB 00011000B
		1376	DB 00011000B
		1377	; graphic " a "
		1378	DB 00000000B
		1379	DB 00000000B
		1380	DB 00000000B
		1381	DB 00000000B
		1382	DB 11111111B
		1383	DB 00000000B
		1384	DB 00000000B
		1385	DB 00000000B
		1386	DB 00000000B
		1387	; graphic " b "
		1388	DB 00011000B
		1389	DB 00011000B
		1390	DB 00011000B
		1391	DB 00011000B
		1392	DB 11111111B
		1393	DB 00011000B
		1394	DB 00011000B
		1395	DB 00011000B
		1396	DB 00011000B
		1397	; graphic " c "
		1398	DB 00000000B
		1399	DB 00000000B
		1400	DB 00000000B
		1401	DB 00000000B
		1402	DB 11111000B
		1403	DB 00011000B
		1404	DB 00011000B
		1405	DB 00011000B
		1406	DB 00011000B

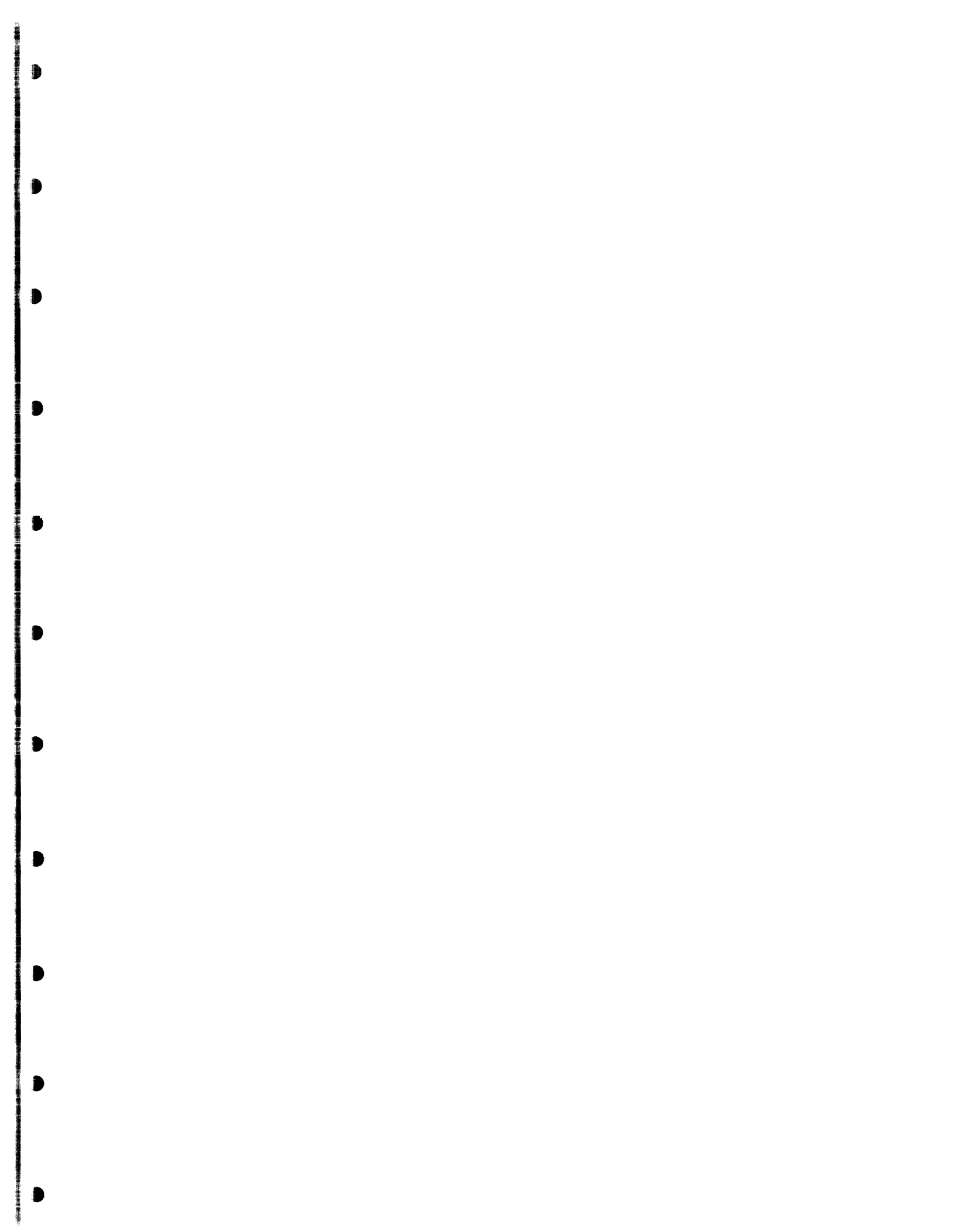
```

LOC OBJ          LINE          SOURCE
1462             DB 10101010B
1463             DB 01010101B
1464             DB 10101010B
1465             DB 01010101B
1466             DB 10101010B
1467             ; graphic "j"
1468             %IF(ZNES(0,1))
1469             THEN (
1470             DB 11110000B
1471             DB 11110000B
1472             DB 11110000B
1473             DB 11110000B
1474             DB 11111111B
1475             DB 00001111B
1476             DB 00001111B
1477             DB 00001111B
1478             DB 00001111B )
1479             ELSE (
1480             DB 00000000B
1481             DB 00000000B
1482             DB 00001000B
1483             DB 00000000B
1484             DB 00111110B
1485             DB 00000000B
1486             DB 00001000B
1487             DB 00000000B
1488             DB 00000000B )FI
1489             ; graphic "k"
1490             DB 00000000B
1491             DB 00000000B
1492             DB 00001000B
1493             DB 00001000B
1494             DB 00101010B
1495             DB 00011100B
1496             DB 00001000B
1497             DB 00000000B
1498             DB 00000000B
1499             ; graphic "l"
1500             DB 00000000B
1501             DB 00000000B
1502             DB 00000000B
1503             DB 00000000B
1504             DB 00001111B
1505             DB 00001111B
1506             DB 00001111B
1507             DB 00001111B
1508             DB 00001111B
1509             ; graphic "m"
1510             DB 00000000B
1511             DB 00000000B
1512             DB 00000000B
1513             DB 00000000B
1514             DB 11110000B
1515             DB 11110000B
1516             DB 11110000B

```


LOC	OBJ	LINE	SOURCE
		1572	DB 00000000B
		1573	DB 00000000B
		1574	DB 11111111B
		1575	DB 00011000B
		1576	DB 00011000B
		1577	DB 00011000B
		1578	DB 00011000B
		1579	; graphic "t"
		1580	DB 00011000B
		1581	DB 00011000B
		1582	DB 00011000B
		1583	DB 00011000B
		1584	DB 11111000B
		1585	DB 00011000B
		1586	DB 00011000B
		1587	DB 00011000B
		1588	DB 00011000B
		1589	; graphic "u"
		1590	DB 00011000B
		1591	DB 00011000B
		1592	DB 00011000B
		1593	DB 00011000B
		1594	DB 11111111B
		1595	DB 00000000B
		1596	DB 00000000B
		1597	DB 00000000B
		1598	DB 00000000B
		1599	; graphic "v"
		1600	DB 00011000B
		1601	DB 00011000B
		1602	DB 00011000B
		1603	DB 00011000B
		1604	DB 00011111B
		1605	DB 00011000B
		1606	DB 00011000B
		1607	DB 00011000B
		1608	DB 00011000B
		1609	; graphic "w"
		1610	DB 10000001B
		1611	DB 11000011B
		1612	DB 01100110B
		1613	DB 00111100B
		1614	DB 00011000B
		1615	DB 00111100B
		1616	DB 01100110B
		1617	DB 11000011B
		1618	DB 10000001B
		1619	; graphic "x"
		1620	DB 00000001B
		1621	DB 00000011B
		1622	DB 000000110B
		1623	DB 00001100B
		1624	DB 00011000B
		1625	DB 00110000B
		1626	DB 01100000B

LOC	OBJ	LINE	SOURCE
		1682	DB 00111100B
		1683	DB 00111100B
		1684	DB 00011100B
		1685	DB 00001100B
		1686	DB 00001100B
		1687	DB 00000000B
		1688	DB 00000000B
		1689)FI
		1690	+1
		1691	
		1692	FONTAB ENDS
		1693	
		1694	END



```
= $SAVE  
= $NOLIST  
INCLUDE (':F2:GLOBAL.H')  
= $SAVE  
= $NOLIST  
INCLUDE (':F2:ISSDEF.H')  
= $SAVE  
= $NOLIST  
INCLUDE (':F2:INIT.H')  
= $SAVE  
= $NOLIST  
INCLUDE (':F2:IODDEF.H')  
= $SAVE  
= $NOLIST  
INCLUDE (':F2:ITCDEF.H')  
= $SAVE  
= $NOLIST  
INCLUDE (':F2:KEYDEF.H')  
= $SAVE  
= $NOLIST  
INCLUDE (':F2:KEYLIB.H')  
= $SAVE  
= $NOLIST  
INCLUDE (':F2:MISC.H')  
= $SAVE  
= $NOLIST  
INCLUDE (':F2:MTR100.H')  
= $SAVE  
= $NOLIST  
INCLUDE (':F2:P6821.H')  
= $SAVE  
= $NOLIST  
INCLUDE (':F2:REGDEF.H')  
= $SAVE  
= $NOLIST  
INCLUDE (':F2:VIDEO.H')  
= $SAVE  
= $NOLIST  
$EJECT
```

```
306 1  COMMAND:      PROCEDURE      EXTERNAL;  
307 2  END;  
308 1  C_BOOT:      PROCEDURE      EXTERNAL;  
309 2  END;  
310 1  INIT:        PROCEDURE      EXTERNAL;  
311 2  END;  
      $ IF EXTENDED_MONITOR  
      $STEP:        PROCEDURE      EXTERNAL;  
      $ END;  
      $ ENDIF  
312 1  DECLARE      STEP_COUNT      WORD EXTERNAL;
```

\$SUBTITLE('MTR-101 - Monitor Loop')

```

323 1 VIDEO_INTR: PROCEDURE PUBLIC;
324 2 DECLARE TEMP BYTE;
325 2 DO;
326 3 CALL SDS; /* Set the Data Segment */

/*
/* Vertical Retrace Interrupt
*/
327 3 IF ( INPUT(GEN_CNT) AND P6821_IRQ2 ) <> 0 THEN
328 3 DO;
329 4 TEMP=INPUT(GEN_DAT); /* Clear Interrupt */
330 4 OUTPUT(GEN_DAT)=(INPUT(GEN_DAT) AND (NOT 0010$0000B));
331 4 TEMP=INPUT(IO_DIP); /* Dummy I/O for 6821 */
332 4 OUTPUT(GEN_DAT)=(INPUT(GEN_DAT) OR 0010$0000B);
333 4 CALL MTR_TTY_INTR; /* Vertical Retrace */
334 4 END;

/*
/* Key-Board Interrupt
*/
335 3 IF ( INPUT(KEY_STAT) AND KS_CXRDY ) <> 0 THEN
336 3 DO;
337 4 KEY_CHAR=INPUT(KEY_DATA);
338 4 KEY_AVAIL=TRUE;
339 4 END;
340 3 END;
341 2 END;

```

*EJECT

```

; STATEMENT # 6
; STATEMENT # 20
; STATEMENT # 39
; STATEMENT # 53
; STATEMENT # 67
; STATEMENT # 108
; STATEMENT # 109
; STATEMENT # 111
; STATEMENT # 138
; STATEMENT # 193
; STATEMENT # 229
; STATEMENT # 237
; STATEMENT # 243
; STATEMENT # 255
; STATEMENT # 265
; STATEMENT # 275
; STATEMENT # 304
; STATEMENT # 314

MTR101 PROC NEAR
0024 55          PUSH BP
0025 8BEC      MOV BP, SP
; STATEMENT # 315

0027 B007      MOV AL, 7H
0029 50        AX
002A E80000    CALL WRITEC
; STATEMENT # 316

002D A00000    MOV AL, RESETF
0030 D0D8      RCR AL, 1
0032 7309     JNB @1
0034 E4FF     IN @FFH
0036 A808     TEST AL, 8H
0038 7403     JZ @1
; STATEMENT # 317

003A E30000    CALL C_BOOT
; STATEMENT # 318

003D C6060000 @1:
@2: MOV RESETF, 0H
; STATEMENT # 320

0042 E80000    CALL COMMAND
; STATEMENT # 321

0045 EBFB      JMP @2
; STATEMENT # 322

MTR101 ENDP
; STATEMENT # 323

VIDEO_INTR PROC NEAR
0047 55          PUSH BP
0048 8BEC      MOV BP, SP
; STATEMENT # 326

004A E80000    CALL SDS
; STATEMENT # 327

004D E4E1      IN @E1H
004F A840      TEST AL, 40H
0051 741B     JZ @4
; STATEMENT # 329

0053 E4E0      IN @E0H

```

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
291	00000H	1	BACK
7	00000H	1	BEL.
94	00000H	1	BILL
4	00000H	4	BOOL
74	00000H	83	BOOT_PAR
	00000H	1	INDEX
	00010H	1	PORT
	00020H	80	STRING
	00520H	1	UNIT
8	00000H	1	BS
95	00000H	1	BSIL
27	00000H	4	BYTEPTR.
30	00000H	1	C.
22	00000H	1	C.
9	00000H	1	CAN.
279	00000H	1	CHAR
58	00000H	1	CHAR
63	00000H	1	CHAR
236	00000H	1	CMND
68	00000H	1	CODE_SEG
303	00000H	1	COL.
102	00000H	7	COLOR.
	00000H	1	FORE
	00010H	1	BACK
	00020H	1	MASK
	00030H	1	CLEAR.
	00040H	1	PAINTED.
	00050H	1	FONT
	00060H	1	COMP_FONT
306	00000H	1	COMMAND.
313	00000H	35	COPYRIGHT.
303	00000H	2	COUNT.
273			CPU_AF
275			CPU_CF
268			CPU_DF
269			CPU_IF
267			CPU_OF
274			CPU_PF
271			CPU_SF
270			CPU_TF
272			CPU_ZF
10			CR
55	00000H		CRLF
103	00000H	3	CRTC_CURSOR.
	00000H	2	START.
	00020H	1	UPDATE.
104	00000H	3	CRTC_DISPLAY
	00000H	2	START.
	00020H	1	UPDATE
276	00000H	1	CURSOR
308	00000H	1	C_BOOT
			BYTE IN PROC (SCA) PARAMETER
			LITERALLY '007H' 315
			BYTE EXTERNAL(36)
			LITERALLY 'BYTE' 98 103 104 106 108 230 231 250
			285
			STRUCTURE EXTERNAL(16)
			BYTE
			BYTE
			BYTE ARRAY(80)
			BYTE
			LITERALLY '008H'
			BYTE EXTERNAL(37)
			POINTER IN PROC (INCB) PARAMETER
			BYTE IN PROC (SXMT) PARAMETER
			BYTE IN PROC (DXMT) PARAMETER
			LITERALLY '018H'
			BYTE IN PROC (D CRT) PARAMETER
			BYTE IN PROC (MCU) PARAMETER
			BYTE IN PROC (WRITC) PARAMETER
			BYTE IN PROC (WRITK) PARAMETER
			LITERALLY '0FE05H'
			BYTE IN PROC (XMTSC) PARAMETER
			STRUCTURE EXTERNAL(41)
			BYTE
			BYTE
			BYTE
			BYTE
			BYTE
			BYTE
			PROCEDURE EXTERNAL(71) STACK=0000H
			BYTE ARRAY(35) DATA
			WORD IN PROC (XMTSC) PARAMETER
			LITERALLY '00000000\$00010000B'
			LITERALLY '00000000\$00000001B'
			LITERALLY '00000100\$00000000B'
			LITERALLY '00000010\$00000000B'
			LITERALLY '00000001\$00000000B'
			LITERALLY '00000000\$00000100B'
			LITERALLY '00000000\$10000000B'
			LITERALLY '00000001\$00000000B'
			LITERALLY '00000000\$01000000B'
			LITERALLY '00DH' 313
			STRUCTURE EXTERNAL(8) STACK=0000H
			STRUCTURE EXTERNAL(42)
			WORD
			BYTE
			STRUCTURE EXTERNAL(43)
			WORD
			STRUCTURE EXTERNAL(43)
			WORD
			PROCEDURE EXTERNAL(59) STACK=0000H
			PROCEDURE EXTERNAL(72) STACK=0000H
			317

15	00000H	18	FF	LITERALLY '00CH'
24	00000H	1	FLAGS	PROCEDURE WORD EXTERNAL(1) STACK=0000H
80	00000H	1	FONT	POINTER EXTERNAL(22)
89	00000H	2	FONTSIZE	WORD EXTERNAL(31)
291	00000H	1	FORE	BYTE IN PROC (SCA) PARAMETER
241			GEN_CNT	LITERALLY '10_GENERAL+1'
242			GEN_DIR	LITERALLY '10_GENERAL'
243			GEN_MODE	LITERALLY '10_GENERAL'
106	00000H	18	H19_MODE	STRUCTURE EXTERNAL(45)
	00000H	1	ALTERNATE	BYTE
	00010H	1	ANSI	BYTE
	00020H	1	AUTO_CR	BYTE
	00030H	1	AUTO_LF	BYTE
	00040H	1	AUTO_REPEAT	BYTE
	00050H	1	BWD	BYTE
	00060H	1	CURSOR	BYTE
	00070H	1	CURSOR_ON	BYTE
	00080H	1	EXPAND	BYTE
	00090H	1	GRAPHIC	BYTE
	000AH	1	INSERT	BYTE
	000BH	1	KEY_EN	BYTE
	000CH	2	REVERSE	WORD
	000EH	1	SHIFTED	BYTE
	000FH	1	STATUS	BYTE
	0010H	1	UPDN	BYTE
	0011H	1	WRAP	BYTE
107	00000H	9	H19_PAR	STRUCTURE EXTERNAL(46)
	00000H	1	CPL	BYTE
	00010H	1	DSC	BYTE
	00020H	1	LPS	BYTE
	00030H	1	POVRAM	BYTE
	00040H	1	SLI	BYTE
	00050H	1	SPC	BYTE
	00060H	1	SM401	BYTE
	00070H	1	SM402	BYTE
	00080H	1	VRAM_SIZE	BYTE
54	00000H	1	HEX_DIGIT	BYTE ARRAY(0) EXTERNAL(7) DATA
92	00000H	1	HORZ_CHAR	BYTE EXTERNAL(34)
16			HT	LITERALLY '007H'
109			I85_JMP	LITERALLY '03030'
153			ICW1_A7A5	LITERALLY '1110*0000B'
156			ICW1_AD14	LITERALLY '0000*0100B'
158			ICW1_IC4	LITERALLY '0000*0001B'
154			ICW1_ICM1	LITERALLY '0001*0000B'
155			ICW1_L1TM	LITERALLY '0000*1000B'
157			ICW1_SML	LITERALLY '0000*0010B'
159			ICW2_A15A8	LITERALLY '1111*1111B'
167			ICW3_S0	LITERALLY '0000*0001B'
166			ICW3_S1	LITERALLY '0000*0010B'
165			ICW3_S2	LITERALLY '0000*0100B'
164			ICW3_S3	LITERALLY '0000*1000B'
163			ICW3_S4	LITERALLY '0001*0000B'
162			ICW3_S5	LITERALLY '0010*0000B'
161			ICW3_S6	LITERALLY '0100*0000B'
160			ICW3_S7	LITERALLY '1000*0000B'
172			ICW4_S6	LITERALLY '0000*0001B'

CROSS-REFERENCE LISTING

Address	Hex	Symbol	Label	Address	Hex	Symbol	Label
0000CH	2	SP	WORD				
0000EH	2	DX	WORD				
00010H	2	CX	WORD				
00012H	2	BX	WORD				
00014H	2	AX	WORD				
00016H	2	IP	WORD				
00018H	2	CS	WORD				
0001AH	2	FLAGS	WORD				
00000H	4	REGP	POINTER IN PROC (XEC) PARAMETER	35			
00000H	4	REGP	POINTER EXTERNAL(39)				
00000H	4	RESETE	BYTE EXTERNAL(40)	316	318*		
00000H	1	REV_SCROLL	PROCEDURE EXTERNAL(69) STACK=0000H				
00000H	1	ROM	BYTE IN PROC (XMTSC) PARAMETER	303			
00000H	1	SCA	PROCEDURE EXTERNAL(65) STACK=0000H				
00000H	1	SCP	PROCEDURE EXTERNAL(66) STACK=0000H				
00000H	1	SCP1	PROCEDURE EXTERNAL(67) STACK=0000H				
00000H	1	SCROLL	PROCEDURE EXTERNAL(68) STACK=0000H				
00000H	2	SDS	PROCEDURE EXTERNAL(58) STACK=0000H	326			
00000H	2	STEP_COUNT	WORD EXTERNAL(74)				
00000H	4	STRNGP	POINTER IN PROC (WRITES) PARAMETER	66			
00000H	4	SXMTSC	PROCEDURE EXTERNAL(3) STACK=0000H				
00000H	4	S_INT0	LITERALLY '24'				
00000H	4	S_ITC_0	LITERALLY '10_INT_SL+0'				
00000H	4	S_ITC_1	LITERALLY '10_INT_SL+1'				
00007H	1	S_XMTC	POINTER EXTERNAL(27)				
00007H	1	TEMP	BYTE IN PROC (VIDEO_INTR)	329*	331*		
00007H	1	TESTK	PROCEDURE BYTE EXTERNAL(50) STACK=0000H				
00000H	6	TRUE	LITERALLY '0FFH'	338			
00000H	2	TTY_INTR	PROCEDURE EXTERNAL(62) STACK=0000H				
00000H	4	TTY_POLL	PROCEDURE BYTE EXTERNAL(63) STACK=0000H				
00000H	4	UIES	POINTER EXTERNAL(28)				
00000H	1	VALU	BYTE IN PROC (SCP1) PARAMETER	296			
00000H	1	VECTORS	POINTER ARRAY(13) EXTERNAL(30)				
00000H	1	VERT_LINE	BYTE EXTERNAL(14)				
00000H	1	VIDEO_INTR	PROCEDURE PUBLIC STACK=0006H				
00047H	57	VIDEO_INTR_A	PROCEDURE EXTERNAL(4) STACK=0000H				
00000H	18	VT	LITERALLY '00BH'				
00000H	5	WIP	BYTE ARRAY(5) EXTERNAL(13)				
00000H	62	WRITEC	PROCEDURE EXTERNAL(11) STACK=0000H	315			
00000H	65	WRITES	PROCEDURE EXTERNAL(12) STACK=0000H				
00000H	235	WRITK	PROCEDURE EXTERNAL(52) STACK=0000H				
00000H	67	XCA	POINTER EXTERNAL(29)				
00000H	4	XCOLOR_BACK	LITERALLY '00*111*000B'				
00000H	100	XCOLOR_FORE	LITERALLY '00*000*111B'				
00000H	100	XEC	PROCEDURE EXTERNAL(5) STACK=0000H				
00000H	108	XMT	STRUCTURE EXTERNAL(47)				
00000H	1	BURST	BYTE				
00000H	1	RCOUNT	INTEGER				
00000H	2	COUNT	WORD				
00000H	1	COL	BYTE				
00000H	1	COLM	BYTE				
00000H	1	RCOLM	BYTE				
00000H	1	COLOR	BYTE				
00000H	1	GRAPHIC	BYTE				
00000H	1	REVERSE	BYTE				
00000H	1	XMTSC	PROCEDURE EXTERNAL(70) STACK=0000H				

SERIES-111 PL/M-86 V2.0 COMPILATION OF MODULE COMMAND
OBJECT MODULE PLACED IN :F2:COMMAND.OBJ
COMPILER INVOKED BY: PLM86.86 :F2:COMMAND.PLM CODE PRINT(:TO:)

```
$TITLE('MTR-100 Z-Machine Monitor ROM: Command Processor')
$SUBTITLE('Compiler Controls')
$ INCLUDE ('F2:COMCON.H')
$SUBTITLE('Compiler Controls')
$OPTIMIZE(3)
$NOOVERFLOW
$COMPACT
$ROM
$XREF
$XLARGE(MTR100 EXPORTS MTR_MON;
MTR_MON;
EXPORTS MTR_SWIM;
MTR_SWIM;
EXPORTS MTR_DCRT;
MTR_DCRT;
EXPORTS MTR_DKBD;
MTR_DKBD;
EXPORTS MTR_SCRT;
MTR_SCRT;
EXPORTS MTR_SKBD;
MTR_SKBD;
EXPORTS MTR_TTY_INTR;
MTR_TTY_INTR;
EXPORTS MTR_TTY_POLL;
MTR_TTY_POLL;
EXPORTS MTR_IRET);
$XLARGE(VIDEQA EXPORTS MTR_DC1;
MTR_DC1;
EXPORTS MTR_DFC;
MTR_DFC;
EXPORTS MTR_EDC;
MTR_EDC;
EXPORTS MTR_FONT;
MTR_FONT;
EXPORTS MTR_MDIC;
MTR_MDIC;
EXPORTS MTR_MDL;
MTR_MDL;
EXPORTS MTR_PROMPT;
MTR_PROMPT;
EXPORTS MTR_RDC;
MTR_RDC;
EXPORTS MTR_QIES;
MTR_QIES;
EXPORTS MTR_XCA);
$XLARGE(FONT EXPORTS MTR_FONT)
$XLARGE(ASTLIB EXPORTS VIDEO_INTR_A)
$RESET(EXTENDED_MONITOR)
$NOINTVECTOR
$SUBTITLE('External Definitions')
COMMAND;
DO;
1
$ INCLUDE ('F2:LEXCAL.H')
$SAVE
$NQLIST
$ INCLUDE ('F2:ASCII.H')
$SAVE
$NQLIST
$ INCLUDE ('F2:ASTLIB.H')
$SAVE
$NQLIST
$ INCLUDE ('F2:CRTC.H')
$SAVE
$NQLIST
```

```

$ INCLUDE ('F2:SUBLIB.H')
= $SAVE
= $NOLIST
$ INCLUDE ('F2:VIDEO.H')
= $SAVE
= $NOLIST

```

```

$ IF EXTENDED_MONITOR
DC BUFFER STRUCTURE (
IN_POINTER BYTE,
OUT_POINTER BYTE,
COUNT BYTE,
CHAR(100) BYTE);
$ ENDIF

```

```

333 1 D_KBD: PROCEDURE(C) EXTERNAL;
334 2 DECLARE C BYTE;
335 2 END;
336 1 MTR_SWIM: PROCEDURE EXTERNAL INTERRUPT INT_STEP ;
337 2 END;

```

```

$ IF EXTENDED_MONITOR
P8085:
DECLARE AF WORD, EXTERNAL;
BC WORD,
DE WORD,
HL WORD,
PC WORD,
SP WORD,
PAGE WORD;
$ END;
ENDIF

```

```

338 1 BOOT_207_5: PROCEDURE EXTERNAL;
339 2 END;
340 1 BOOT_207_8: PROCEDURE EXTERNAL;
341 2 END;
$EJECT

```

```

$      IF EXTENDED_MONITOR
DECLARE CMD(*)  BYTE DATA(
'Boot',0,
'Color Bar',0,
'Continue',0,
'Diagnose',0,
'Dump',0,
'Examine',0,
'Execute',0,
'Fill',0,
'Input',0,
'Move',0,
'Output',0,
'Register',0,
'Step',0,
'Swap',0,
'Terminal',0,
'Trace',0,
'Version',0,
0,0);

$      ELSE
CMD_STR(4)      STRUCTURE (
CHAR            BYTE,
STRING         POINTER)
DATA (
'B',@CS_BOOT,
'V',@CS_VERS);
$      CS_BOOT(*)  BYTE DATA ('Boo','t'+EOS);
$      CS_VERS(*)  BYTE DATA ('Version 1.','2'+EOS);
$      ENDIF

344  1  1  DC      (ADDR_1,ADDR_2,ADDR_3)  STRUCTURE (
BAROBT         BOOL,
BYTE_1         BYTE,
C              BYTE,
CMD_IDX        BYTE,
CMD_PTR        BYTE,
DELM           BYTE,
I              WORD,
INP_PTR        WORD,
J              BYTE,
K              WORD,
LINE(80)       WORD,
MATCH          BYTE,
PTR#P          POINTER,
PTR_B          BASED_PTR#P,
SELEC          SELECTOR,
STALLED        BOOL,
STEP_COUNT     WORD)
346  1  1  DC      ADDR;
347  1  1  DC      /* True if Boot Aborted

$      /* I/O is currently stalled */

```

```

$      IF EXTENDED_MONITOR
COMMAND:      PROCEDURE      BYTE;      PUBLIC;
DECLARE      C
DO;
CALL DPT???;      /* Display Prompt      */
MATCH, INP_PTR, CMD_PTR, CMD_IDX=0;
DO WHILE CMD(CMD_PTR) <> 0 ;
IF MATCH=0 THEN
    C = MCU(READC);
MATCH=0;
IF (C=DEL OR C=BS) AND INP_PTR=0 THEN
    CALL WRITEC(BEL);
ELSE IF C=DEL OR C=BS THEN
    DO;
    CALL RUBOUT;
    INP_PTR=INP_PTR-1;
    TMP_IDX, TMP_PTR=0;
    CALL FNM;
    CMD_PTR=TMP_PTR+INP_PTR;
    CMD_IDX=TMP_IDX;
    END;
ELSE IF C = MCU(CMD(CMD_PTR)) THEN
    DO;
    CALL WRITEC(CMD(CMD_PTR));
    LINE(INP_PTR)=C;
    TMP_PTR, CMD_PTR=CMD_PTR+1;
    TMP_INP, INP_PTR=INP_PTR+1;
    TMP_IDX=CMD_IDX;
    CALL ANC;
    CALL FNM;
    BYTE_1=MCU(CMD(CMD_PTR));
    DO WHILE BYTE_1<>0 AND BYTE_1=MCU(CMD(TMP_PTR+TMP_INP)) ;
    CALL ANC;
    CALL FNM;
    END;
    IF CMD(TMP_PTR) = 0 THEN
    DO;
    MATCH=1;
    C=MCU(CMD(CMD_PTR+TMP_INP));
    END;
    END;
ELSE
    DO;
    TMP_INP=INP_PTR;
    TMP_PTR=CMD_PTR;
    TMP_IDX=CMD_IDX;
    CALL ANC;
    CALL FNM;
    DO WHILE CMD(TMP_PTR)<>0 AND C<>MCU(CMD(TMP_PTR+TMP_INP)) ;
    CALL ANC;
    
```

```

    TMP_IDX=TMP_IDX+1;
    TMP_INP=0;
  END;
END;

```

FNM: PROCEDURE;

```

DO;
  TMP_INP=0;
  DO WHILE CMD(TMP_PTR+TMP_INP)<>0 AND TMP_INP<INP_PTR ;
    DO WHILE CMD(TMP_PTR+TMP_INP)<>0 AND
      MCU(CMD(TMP_PTR+TMP_INP))=LINE(TMP_INP) AND
      TMP_INP<INP_PTR ;
      TMP_INP=TMP_INP+1;
    END;
  IF TMP_INP=INP_PTR OR CMD(TMP_PTR)=0
  THEN
    RETURN;
  ELSE
    DO;
      TMP_PTR=TMP_PTR+TMP_INP;
      CALL ANC;
    END;
  END;
END;

```

* ELSE

COMMAND: PROCEDURE PUBLIC; /* Display Prompt

*/

```

353 3 3 CMD_IDX = 2;
354 3 DO WHILE CMD_IDX = 2 ;
355 4 C = MCU(READC);
356 4 IF C = 'B' THEN
357 4 CMD_IDX = 0;
358 4 ELSE IF C = 'V' THEN
359 4 CMD_IDX = 1;
360 4 ELSE
361 4 CALL WRITEC(BEL);
362 4 END;
363 3 CALL WRITEC(' ');
364 3 DO CASE CMD_IDX ;
365 4 CALL C_BOOT;
366 4 CALL C_VERS;
367 4 END;

```

```

368 3 END;
369 2 END;

```

* ENDF


```

370 1 C_BOOT:
371 2 DECLARE BOOT ADDRESS,
C BYTE,
PRIMARY BOOL;
DO;
PUBLIC;
/* Address of Boot Routine
/* Temporarily Input Character
/* TRUE for boot from Prim. Dev */
372 2 DO;
ECHO: PROCEDURE;
DO;
CALL WRITEC(C);
C = MCU(READC);
END;
373 3
374 4
375 5
376 5
377 5
378 4
379 3 /* Default to the DIP-Switch
BOOT_PAR_INDEX = INPUT(IO_DIP) AND DIP_DEVICE_MASK;
380 3 IF BOOT_PAR_INDEX >= LENGTH(BOOT_DEVICE) THEN
381 3 DO;
382 4 CALL WRITES(@('BEL,CR,LF','Illegal Boot Configuratio','n'+EOS));
383 4 RETURN;
384 4 END;
385 3 BABORT = FALSE;
386 3 BOOT_PAR_UNIT = 0;
387 3 BOOT_PAR_STRNG(0) = 0;
388 3 PRIMARY = TRUE;
IF NOT RESETF THEN
DO;
C=MCU(READC);
IF KY_F0+1 <= C AND C < KY_F0+1+LENGTH(BOOT_DEVICE) THEN
DO;
BOOT_PAR_INDEX = C - KY_F0 - 1;
CALL WRITES(@('ESC','p','f'+EOS));
CALL WRITEC('0'+BOOT_PAR_INDEX+1);
CALL WRITES(@('ESC','q'+EOS));
C = MCU(READC);
END;
IF '0' <= C AND C<'0'+BOOT_DEVICE(BOOT_PAR_INDEX).NUM_UNITS THEN
DO;
BOOT_PAR_UNIT = C - '0';
CALL ECHO;
END;
IF C = 'S' THEN
DO;
PRIMARY = FALSE;
CALL ECHO;
END;
IF C = '.' THEN
DO;
I=0;
CALL WRITEC(C);
C=READC;
389 3
390 3
391 4
392 4
393 4
394 5
395 5
396 5
397 5
398 5
399 5
400 4
401 4
402 5
403 5
404 5
405 4
406 4
407 5
408 5
409 5
410 4
411 4
412 5
413 5
414 5

```

*EJECT

#EJECT

```
* IF EXTENDED_MONITOR  
C_DIAG: PROCEDURE;  
DO;  
END;  
END;  
* ENDIF
```

*EJECT

```
CALL WRITEC(C);  
ELSE  
CALL WRITES(@(ESC,'F1',ESC,'G'+EOL)); /* Checkerboard */  
ADDR_3.OFFS=ADDR_3.OFFS+1;  
END;  
END;  
END;  
* ENDIF  
  
*EJECT
```

```
/*
 *      Fill      nnn:mmm -- pppp , qq
 */
$      IF EXTENDED_MONITOR
C_FILL:      PROCEDURE;
DO;
DELIM=IHA(' ', @ADDR_1);
IF DELIM<>' ' THEN
RETURN;
CALL WRITEC(DELIM);

DELIM=IHV(' ', @ADDR_2.OFFS, 2);
IF DELIM<>' ' THEN
RETURN;
CALL WRITEC(DELIM);

DELIM=IHV(CR, @WORD_1, 1);
IF DELIM<>CR THEN
RETURN;

DO WHILE ADDR_1.OFFS <> ADDR_2.OFFS ;
CALL SFB(LOW(WORD_1), ADDR_1.BASE, ADDR_1.OFFS);
ADDR_1.OFFS=ADDR_1.OFFS+1;
END;
CALL SFB(LOW(WORD_1), ADDR_1.BASE, ADDR_1.OFFS);
END;
$      ENDIF

$EJECT
```

```

/*
 *      Move      nnnn:mmmm -- pppp , qqqq
 */
$      IF EXTENDED_MONITOR
C_MOVE:      PROCEDURE;
DO;
  DELIM=IHA(' ', @ADDR_1);
  IF DELIM<>' ' THEN
    RETURN;
  CALL WRITEC(DELIM);

  DELIM=IHV(' ', @ADDR_2.OFFS, 2);
  IF DELIM<>' ' THEN
    RETURN;
  CALL WRITEC(DELIM);

  DELIM=IHV(CR, @WORD_1, 2);
  IF DELIM<>CR THEN
    RETURN;

  IF (ADDR_1.OFFS<=ADDR_2.OFFS AND
      ADDR_1.OFFS<=WORD_1 AND WORD_1<=ADDR_2.OFFS) OR
      (ADDR_1.OFFS>=ADDR_2.OFFS AND
      (ADDR_2.OFFS<=WORD_1 OR WORD_1<=ADDR_1.OFFS))
  THEN
    DO;
      I=-1;
      WORD_1=WORD_1+ADDR_2.OFFS-ADDR_1.OFFS;
      WORD_2=ADDR_1.OFFS;
      ADDR_1.OFFS=ADDR_2.OFFS;
      ADDR_2.OFFS=WORD_2;
    END;
  ELSE
    I=1;

  DO WHILE ADDR_1.OFFS <> ADDR_2.OFFS ;
  CALL SFB(GFB(ADDR_1.BASE, ADDR_1.OFFS), ADDR_1.BASE, WORD_1);
  ADDR_1.OFFS=ADDR_1.OFFS+I;
  WORD_1=WORD_1+I;
  END;
  CALL SFB(GFB(ADDR_1.BASE, ADDR_1.OFFS), ADDR_1.BASE, WORD_1);
  END;
$      ENDIF

```

*EJECT

```

$ IF EXTENDED_MONITOR
  PROCEDURE;
  DO;
    STEP_COUNT=0;
    DELIM=IHV(CR,@WORD_1,2);
    IF DELIM <> CR THEN
      RETURN;
    CALL CRLF;

    STEP_COUNT = WORD_1 ;
    CALL SSTEP;
  END;
END;
$ ENDIF

$ IF EXTENDED_MONITOR
  PROCEDURE PUBLIC;
  DO;
    WORD_1=REG_IP;
    DO WHILE PREFIX(GFB(REG_CS,WORD_1)) ;
      WORD_1=WORD_1+1;
    END;
    IF GFB(REG_CS,WORD_1)=08EH AND
      (GFB(REG_CS,WORD_1+1) AND 00#011#000B)=00#010#000B
    THEN
      DO;
        CALL WRITES('@Bad to trap through mov to SS',CR,LF+EOS);
        REG_FLAGS = REG_FLAGS AND (NOT CPU_TF) ;
      END;
    ELSE
      DO;
        REG_FLAGS = REG_FLAGS OR CPU_TF ;
        CALL XEC(REG*P);
      END;
    END;
  END;
END;
$

PREFIX: PROCEDURE(op_code)
DECLARE op_code BYTE;
DO;
  IF (op_code AND 111#00#111B)=001#00#110B THEN
    RETURN(TRUE);
  ELSE IF op_code = 0F0H THEN
    THEN

```



```
* IF EXTENDED_MONITOR
C_SWAP: PROCEDURE;
DO;
  DELIM=IHA(' ', @ADDR_1);
  IF DELIM <> ' ' THEN
    RETURN;
  DELIM=IHV(CR, @WORD_1, 2);
  IF DELIM <> CR THEN
    RETURN;
CALL P8085(ADDR_1, BASE, WORD_1, ADDR_1, OFFS, 0, 0, 0, 0);
END;
* ENDIF

*EJECT
```

```
CALL WRITES('@' AX BX CX DX SI DI BP',CR,LF+EOS));  
CALL ORV(REG.AX);  
CALL ORV(REG.BX);  
CALL ORV(REG.CX);  
CALL ORV(REG.DX);  
CALL ORV(REG.SI);  
CALL ORV(REG.DI);  
CALL OHM(REG.BP);  
CALL CRLF;  
CALL CRLF;  
CALL WRITES('@' IP CS SP SS DS ES Flags',CR,LF+EOS));  
CALL ORV(REG.IP);  
CALL ORV(REG.CS);  
CALL ORV(REG.SP);  
CALL ORV(REG.SS);  
CALL ORV(REG.DS);  
CALL ORV(REG.ES);  
CALL OHM(REG.FLAGS);  
CALL CRLF;  
CALL CRLF;  
END;  
ENDIF
```

#EJECT

```
END;
END;
OUTPUT(IO_SER_A+3)=INPUT(IO_SER_A+3) AND 0FEH;
END;
```

```
ABP:
DECLARE PROCEDURE(ptr) WORD;
DO:
  ptr < 79
  THEN
    RETURN(ptr+1);
  ELSE
    RETURN(0);
END;
END;
```

```
RESUME:
PROCEDURE;
DO:
  CALL T_XMTC(XON);
  STALLED = FALSE;
END;
END;
```

```
STALL:
PROCEDURE;
DO:
  STALLED = TRUE;
  CALL T_XMTC(XOFF);
END;
END;
```

```
T_XMTC:
PROCEDURE(c);
DECLARE c BYTE;
DO:
  CALL MCHAR(IO_SER_A,c);
END;
END;
* ENDIF
```

\$EJECT

```

$      IF EXTENDED_MONITOR
C_EXECUTE:      PROCEDURE;
DO;
  DELIM=IHAYCR,eADDR_1);
  IF DELIM<>YCR THEN
    RETURN;
  REG_CS=ADDR_1.BASE;
  REG_IP=ADDR_1.OFFS;
  CALL XEC(REG#P);
END;
$      ENDIF

```

464 1 END;

```

00C0 A02A00 MOV AL,CMD_IDX
00C3 B105 MOV CL,5H
00C5 F6E1 MUL CL
00C7 89C3 MOV BX,AX
00C9 2EC4870B00 LES AX,CS:CMD_STRCBX+1HJ
00CE 06 PUSH ES ; 1
00CF 50 PUSH AX ; 2
00D0 E90000 CALL WRITES
; STATEMENT # 363
00D3 B020 MOV AL,20H
00D5 50 PUSH AX ; 1
00D6 E90000 CALL WRITTEC
; STATEMENT # 364
00D9 8A1E2A00 MOV BL,CMD_IDX
00DD B700 MOV BH,0H
00DF D1E3 SHL BX,1
00E1 2EFA7E600 JMP CS:@IBXJ
; STATEMENT # 365
00E6 EA00 DW @9
00E8 EF00 DW @10
; STATEMENT # 366
00EA E90700 CALL C_BOOT
; STATEMENT # 369
00ED 5D POP BP
00EE C3 RET
; STATEMENT # 370
00FF E9C201 CALL C_VERS
; STATEMENT # 373
00F2 5D POP BP
00F3 C3 RET
; STATEMENT # 374
00F4 55 PROC NEAR
00F5 8BEC MOV BP,SP
; STATEMENT # 375
0276 55 PROC NEAR
0277 8BEC MOV BP,SP
; STATEMENT # 376
0279 FF368300 PUSH C
027D E90000 CALL WRITTEC
; STATEMENT # 377
0280 E90000 CALL READC
0283 50 AX
0284 E90000 CALL MCU
0287 A28300 MOV C,AL
; STATEMENT # 378
028A 5D POP BP
028B C3 RET
; STATEMENT # 379
ECHO ENDP
00F7 E4FF IN @FFH
00F9 2407 AND AL,7H
00FB A20000 MOV BOOT_PAR,AL
  
```

```

015F A28300      MOV      C,AL      ; STATEMENT # 400
                                @13:
0162 B130      MOV      CL,30H
0164 A08300      MOV      AL,C
0167 3AC8      CMP      CL,AL
0169 7721      JA      @14
016B A00000      MOV      AL,BOOT_PAR
016E B205      MOV      DL,5H
0170 F6E2      MULL   DL
0172 89C3      MOV      BX,AX
0174 2E8A70200  MOV      AL,CS:BOOT_DEVICE(BX+2H)
0179 02C1      ADD     AL,CL
017B 8A168300  MOV      DL,C
017E 3AD0      CMP     DL,AL
0181 7309      JNB    @14
                                ; STATEMENT # 402
0183 2AD1      SUB     DL,CL
0185 80165200  MOV     BOOT_PAR+52H,DL
                                ; STATEMENT # 403
0189 E8EA00      CALL   ECHO
                                ; STATEMENT # 405
                                @14:
018C 803E83053  CMP     C,53H
0191 7508      JNZ    @15
                                ; STATEMENT # 407
0193 C606840000  MOV     PRIMARY,0H
                                ; STATEMENT # 408
0198 E8DB00      CALL   ECHO
                                ; STATEMENT # 410
                                @15:
019B A08300      MOV     AL,C
019E 3C3A      CMP     AL,3AH
01A0 7539      JNZ    @16
                                ; STATEMENT # 412
01A2 C7060C000000  MOV     I,0H
                                ; STATEMENT # 413
01A8 50      PUSH   AX
                                ; 1
01A9 E80000      CALL   WRITEC
                                ; STATEMENT # 414
01AC E80000      CALL   READC
01AF A28300      MOV     C,AL
                                ; STATEMENT # 415
                                @17:
01B2 A08300      MOV     AL,C
01B5 3C0D      CMP     AL,0DH
01B7 7419      JZ     @18
01B9 A10C00      MOV     AX,I
01BC 83F84E      CMP     AX,4EH
01BF 7311      JNB    @18
                                ; STATEMENT # 416
01C1 87C3      MOV     BX,AX
01C3 8A0E8300  MOV     CL,C
01C7 888F0200  MOV     BOOT_PAR(BX+2H),CL
                                ; STATEMENT # 417

```

```

0231 732D JNB @24
0233 A02700 MOV AL,BARBRT
0236 D0D8 RCR AL,1
0238 7226 JB @24
; STATEMENT # 438

023A B80000 MOV AX,0H
023D C41E0000 LES BX,REGP
0241 26894718 ES:REGIBX+19HJ,AX
0245 268907 ES:REGIBXJ,AX
0248 26894704 ES:REGIBX+4HJ,AX
; STATEMENT # 439

024C 26C747160004 MOV ES:REGIBX+16HJ,400H
; STATEMENT # 440

0252 E80000 CALL ORLF
; STATEMENT # 441

0255 C4060000 LES AX,REGP
0259 06 PUSH ES ; 1
025A 50 PUSH AX ; 2
025B E80000 CALL XEC
; STATEMENT # 443

025E 5D POP BP
025F C3 RET

; STATEMENT # 444

0260 A02700 MOV AL,BARBRT
0263 D0D8 RCR AL,1
0265 7305 JNB @26
; STATEMENT # 445

0267 B86500 MOV AX,OFFSET(@@LONG*CONSTANT*0065H)
026A EB03 JMP @31
; STATEMENT # 445

; STATEMENT # 446

026C B87200 MOV AX,OFFSET(@@LONG*CONSTANT*0072H)
; STATEMENT # 446

; STATEMENT # 447

026F 0E PUSH CS ; 1
0270 50 PUSH AX ; 2
0271 E80000 CALL WRITES
; STATEMENT # 447

; STATEMENT # 448

0274 5D POP BP
0275 C3 RET
; STATEMENT # 448

; STATEMENT # 449

027C 55 PROC NEAR
028D 8BEC MOV BP,SP
; STATEMENT # 451

028F E80000 CALL FLAGS
0292 A32400 MOV TFLAGS,AX
; STATEMENT # 452

0295 FB STI
; STATEMENT # 453

0296 A90002 TEST AX,200H
0299 7501 JNZ @28
; STATEMENT # 453
    
```

DEFN ADDR SIZE NAME, ATTRIBUTES, AND REFERENCES

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
303	0000H	4	ADDR_1
346	0000H	2	BASE
	0002H	2	OFFS
346	0004H	4	ADDR_2
	0000H	2	BASE
	0002H	2	OFFS
346	0003H	4	ADDR_3
	0000H	2	BASE
	0002H	2	OFFS
347	0027H	1	BARBRT
319	0000H	1	BACK
7	0000H	7	BLK
126	0000H	1	BILL
6	0000H	1	BOUL
371	0022H	2	BOOT
338	0000H	2	BOOT_207_5
340	0000H	2	BOOT_207_8
342	0000H	10	BOOT_DEVICE
	0000H	2	PROC
	0002H	1	NUM_UNITS
	0003H	2	DADDR
106	0000H	83	BOOT_PAR
	0000H	1	INDEX
	0001H	1	PORT
	0002H	80	STRNG
	0052H	1	UNIT
8	0000H	1	BS
127	0000H	1	BSIL
348	0026H	1	BSTATUS
27	0000H	4	BYTEPTR
347	0028H	1	BYTE_1
22	0000H	1	C
347	0029H	1	C
371	0083H	1	C
30	0000H	1	C
334	0000H	1	C
9	023CH	40	CAN
307	0000H	1	CBA
95	0000H	1	CHAR
90	0000H	1	CHAR
347	002AH	1	CMD_IDX
347	002BH	1	CMD_PTR
344	000AH	20	CMD_STR
	0000H	1	CHAR
	0001H	4	STRNG
279	0000H	1	CMND
100	0000H	1	CODE_SEG
331	0000H	1	COL
			LITERALLY 'STRUCTURE (
			BASE
			WORD, OFFS
			WORD, OFFS
			WORD, OFFS
			STRUCTURE
			WORD
			STRUCTURE
			WORD
			WORD
			STRUCTURE
			WORD
			WORD
			385* 436 443 456* 457
			BYTE IN PROC (SCA) PARAMETER
			LITERALLY '007H'
			360 382 425 444 445
			PARAMETER
			LITERALLY 'BYTE'
			130 135 136 138 140 273 274 287
			LITERALLY 'BYTE'
			313 343 347 371 448
			WORD IN PROC (C_BOOT)
			432* 435
			PROCEDURE EXTERNAL(72) STACK=0000H
			342
			PROCEDURE EXTERNAL(73) STACK=0000H
			342
			STRUCTURE ARRAY(2) DATA
			380 392
			WORD
			432
			WORD
			400
			BYTE ARRAY(2)
			430 431
			STRUCTURE EXTERNAL(16)
			379* 380 394* 396 400 430 431 432
			BYTE
			430* 431*
			BYTE ARRAY(80)
			387* 415 416* 421*
			386* 402*
			BYTE
			LITERALLY '008H'
			436
			BYTE EXTERNAL(37)
			436
			BYTE PUBLIC
			PARAMETER
			POINTER IN PROC (INCB)
			27
			BYTE
			IN PROC (DXMTC) PARAMETER
			22
			355* 356 358
			BYTE IN PROC (C_BOOT)
			375 376* 391* 392 394 398* 400
			402 405 410 413 414* 415 416 418 419* 423
			BYTE IN PROC (SXMTC) PARAMETER
			30
			PARAMETER
			BYTE IN PROC (D_KBD) PARAMETER
			334
			LITERALLY '018H'
			PROCEDURE BYTE PUBLIC STACK=0004H
			307
			BYTE IN PROC (D_CRT) PARAMETER
			95
			90
			BYTE IN PROC (WRITEC) PARAMETER
			90
			353* 354 357* 357* 362 364
			BYTE IN PROC (MCU) PARAMETER
			STRUCTURE ARRAY(4) DATA
			362
			BYTE
			POINTER
			362
			BYTE IN PROC (WRITK) PARAMETER
			279
			LITERALLY '0FE05H'
			331
			BYTE IN PROC (XMTSC) PARAMETER

107	0000H	4	DCL.	139	140	141	142	143	144	145	146	147	148	149	150
11	002CH	1	DEL.	151	152	153	154	155	156	157	158	159	160	161	162
347	002CH	4	DFC.	163	164	165	166	167	168	169	170	171	172	173	174
108	0000H			175	176	177	178	179	180	181	182	183	184	185	186
72			DIP_0.	187	188	189	190	191	192	193	194	195	196	197	198
73			DIP_1.	199	200	201	202	203	204	205	206	207	208	209	210
74			DIP_2.	211	212	213	214	215	216	217	218	219	220	221	222
75			DIP_3.	223	224	225	226	227	228	229	230	231	232	233	234
76			DIP_4.	235	236	237	238	239	240	241	242	243	244	245	246
77			DIP_5.	247	248	249	250	251	252	253	254	255	256	257	258
30			DIP_50HZ	259	260	261	262	263	264	265	266	267	268	269	270
78			DIP_5.	271	272	273	294	295	296	297	298	299	300	301	343
79			DIP_6.	344	345	346	347	348							
84			DIP_7.												
85			DIP_AUTO_BOOT.												
81			DIP_DEVICE_MASK.												
82			DIP_RSV_1.												
83			DIP_RSV_2.												
			DIP_RSV_3.												
123	0000H	256	DMAP												
105	0000H	2	DS_SIZE.												
21	0000H		DXMTC.												
306	0000H		D_CRT.												
333	0000H		D_KBD.												
309	0000H		D_TTY_RES.												
109	0000H	4	D_XMTC												
373	0276H	22	ECHQ												
110	0000H	4	EDC												
111	0000H	4	EMEC												
315	0000H		ENABLE_LINES												
12			EOL.												
13			EOS.												
14			ESC.												
137	0000H	10	ESCP												
	0000H	1	COMMAND.												
	0001H	2	FUNCTION.												
	0003H	1	MODE												
	0004H	1	OPER_COUNT.												
	0005H	1	OPER_INDEX.												
	0006H	4	OPERAND.												
147			F86_AF												
142			F86_CF												
143			F86_DF												
			F86_IF												
			POINTER EXTERNAL (18)												
			LITERALLY '0000\$0001B'												
			LITERALLY '0000\$0010B'												
			LITERALLY '0000\$1000B'												
			LITERALLY '0001\$0000B'												
			LITERALLY '0010\$0000B'												
			LITERALLY '1\$000\$0\$000B'												
			LITERALLY '0100\$0000B'												
			LITERALLY '100\$0000B'												
			LITERALLY '0\$000\$1\$000B'												
			LITERALLY '0\$100\$0\$000B'												
			LITERALLY '0\$010\$0\$000B'												
			LITERALLY '0\$001\$0\$000B'												
			BYTE ARRAY (256) EXTERNAL (33)												
			WORD EXTERNAL (15)												
			PROCEDURE EXTERNAL (0) STACK=0000H												
			PROCEDURE EXTERNAL (59) STACK=0000H												
			PROCEDURE EXTERNAL (70) STACK=0000H												
			PROCEDURE EXTERNAL (60) STACK=0000H												
			POINTER EXTERNAL (19)												
			PROCEDURE IN PROC (0_BOOT) STACK=0006H										403	408	
			POINTER EXTERNAL (20)												
			POINTER EXTERNAL (21)												
			PROCEDURE EXTERNAL (63) STACK=0000H												
			LITERALLY '080H'				345	382	395	397	425	444	445		
			LITERALLY '01BH'				395	397							
			STRUCTURE EXTERNAL (44)												
			BYTE												
			WORD												
			BYTE												
			BYTE												
			BYTE												
			BYTE ARRAY (4)												
			LITERALLY '0000\$0001\$0000B'												
			LITERALLY '0000\$0000\$0001B'												
			LITERALLY '0100\$0000\$0000B'												
			LITERALLY '0010\$0000\$0000B'												

204	ICW3_S6	LITERALLY '0100\$0000B'	
203	ICW3_S7	LITERALLY '1000\$0000B'	
215	ICW4_86	LITERALLY '0000\$0001B'	
214	ICW4_AE01	LITERALLY '0000\$0010B'	
212	ICW4_BUF	LITERALLY '0000\$1000B'	
211	ICW4_FNM	LITERALLY '0001\$0000B'	
213	ICW4_MS	LITERALLY '0000\$0100B'	
170	IL_KV	LITERALLY '6'	
184	IL_PERR	LITERALLY '0'	
185	IL_PSWAP	LITERALLY '1'	
191	IL_PTR	LITERALLY '7'	
187	IL_S100	LITERALLY '3'	
188	IL_SER_A	LITERALLY '4'	
189	IL_SER_B	LITERALLY '5'	
186	IL_TIMER	LITERALLY '2'	
26	INCB	PROCEDURE EXTERNAL (2) STACK=0000H	
37	INIT_ITV	PROCEDURE EXTERNAL (6) STACK=0000H	
0000H	INPUT	BUILTIN	
002DH	INP_PTR	379	
150	INT_DBZ	LITERALLY '0'	
154	INT_IOF	LITERALLY '4'	
152	INT_NMI	LITERALLY '2'	
153	INT_OBI	LITERALLY '3'	
151	INT_SSTEP	LITERALLY '1'	336
175	IO_CRTC	LITERALLY '0DDCH'	
155	IO_DIP	LITERALLY '00F7H'	379
172	IO_GENERAL	LITERALLY '00E0H'	
157	IO_HIADR	LITERALLY '00FDH'	
166	IO_INT_MS	LITERALLY '00F2H'	
167	IO_INT_SL	LITERALLY '00F0H'	
165	IO_KEYBRD	LITERALLY '00F4H'	
174	IO_LTPN	LITERALLY '00DEH'	
158	IO_MEM	LITERALLY '00FCH'	
171	IO_PRINTER	LITERALLY '00E2H'	
160	IO_RSRV_1	LITERALLY '00FAH'	
162	IO_RSRV_2	LITERALLY '00F3H'	
163	IO_RSRV_3	LITERALLY '00F7H'	
164	IO_RSRV_4	LITERALLY '00F6H'	
173	IO_RSRV_5	LITERALLY '00DFH'	
177	IO_RSRV_6	LITERALLY '00CFH'	
169	IO_SER_A	LITERALLY '00E8H'	
168	IO_SER_B	LITERALLY '00ECH'	
161	IO_SOUND	LITERALLY '00F9H'	
156	IO_SWAP	LITERALLY '00FEH'	
170	IO_TIMER	LITERALLY '00E4H'	
159	IO_TSTAT	LITERALLY '00FBH'	
176	IO_VIDED	LITERALLY '00D3H'	
179	IO_Z207_0	LITERALLY '00D8H'	342
178	IO_Z207_1	LITERALLY '00B8H'	342
181	IO_Z217_0	LITERALLY '00B8H'	
180	IO_Z217_1	LITERALLY '00A8H'	
38	IWI	BYTE IN PROC (INIT_ITV) PARAMETER	38
38	IVP	POINTER IN PROC (INIT_ITV) PARAMETER	38
347	J	WORD	
347	K	WORD	
242	KC_ARF	LITERALLY '02H'	

32	0000H	VIDEO_INTR_A	PROCEDURE EXTERNAL(4) STACK=0000H				
44		VID_CDC	LITERALLY '10_VIDEO+1'				
43		VID_CDD	LITERALLY '10_VIDEO+0'				
42		VID_CMD	LITERALLY '10_VIDEO+0'				
45		VRM_DAT	LITERALLY '10_VIDEO+2'				
47		VRM_MPC	LITERALLY '10_VIDEO+3'				
46		VRM_MPD	LITERALLY '10_VIDEO+2'				
18		VT	LITERALLY '00BH'				
103	0000H	WIP	BYTE ARRAY(5) EXTERNAL(13)				
349	001AH	WORD_1	WORD				
349	001CH	WORD_2	WORD				
349	001EH	WORD_3	WORD				
349	0020H	WORD_4	WORD				
94	0000H	WRITEC	PROCEDURE EXTERNAL(11) STACK=0000H	360	363	375	396
97	0000H	WRITES	418				
			PROCEDURE EXTERNAL(12) STACK=0000H	362	382	395	397
278	0000H	WRITE	444	445			
119	0000H	XCA	PROCEDURE EXTERNAL(51) STACK=0000H				
133		XCOLOR_BACK	POINTER EXTERNAL(29)				
132		XCOLOR_FORE	LITERALLY '001111\$00B'				
34	0000H	XEC	LITERALLY '001000\$111B'				
140	0000H	XMT	PROCEDURE EXTERNAL(5) STACK=0000H	441			
			STRUCTURE EXTERNAL(47)				
			BYTE				
			INTEGER				
			WORD				
			BYTE				
			BYTE				
			BYTE				
			BYTE				
			BYTE				
			PROCEDURE EXTERNAL(69) STACK=0000H				
330	0000H	XMTSC	LITERALLY '013H'				
19		XOHF	LITERALLY '011H'				
20		XON	LITERALLY '10\$00\$00B'				
131		XREVERSE	LITERALLY '10\$00\$00B'				

MODULE INFORMATION:

CODE AREA SIZE = 02R9H 697D
 CONSTANT AREA SIZE = 0000H 0D
 VARIABLE AREA SIZE = 0085H 133D
 MAXIMUM STACK SIZE = 000AH 10D
 2222 LINES READ
 0 PROGRAM WARNINGS
 0 PROGRAM ERRORS

END OF PL/M-86 COMPILATION

```
= $SAVE  
= $NOLIST  
= $NOLIST INCLUDE (':F2:H19CRT.H')  
= $SAVE  
= $NOLIST INCLUDE (':F2:IODEF.H')  
= $SAVE  
= $NOLIST INCLUDE (':F2:IOLIB.H')  
= $SAVE  
= $NOLIST INCLUDE (':F2:ITCDEF.H')  
= $SAVE  
= $NOLIST INCLUDE (':F2:MISC.H')  
= $SAVE  
= $NOLIST INCLUDE (':F2:MTR100.H')  
= $SAVE  
= $NOLIST INCLUDE (':F2:P6821.H')  
= $SAVE  
= $NOLIST INCLUDE (':F2:VIDEOA.H')  
= $SAVE  
= $NOLIST
```

231 1 DECLARE INITIAL_VECTORS(*) POINTER DATA (

```
@MTR_DCI,  
@MTR_DFC,  
@MTR_SKBD,  
@MTR_EDC,  
0,  
@MTR_FONT,  
@MTR_MDC,  
@MTR_MDJ,  
@MTR_PROMPT,  
@MTR_RDIC,  
@MTR_SCRT,  
@MTR_UIES,  
@MTR_XCA  
);
```

\$SUBTITLE('Initialization Routines')

MTR-101 Z-Machine Monitor ROM: Initialization Routines
 Initialization Routines

```

/*
 * INIT_68A21' initializes the General/Printer 68A21.
 *
 * NOTE: To provide a viable system, the interrupts
 * in the data port should always be enabled.
 * This is so that it is never necessary to
 * examine this port before outputting to it.
 * Such an output might accidentally clear a
 * pending interrupt. Instead, any interrupt
 * enable/disable should be done in the con-
 * trol port of the 6821.
 *
 * The various ports and pins are initialized as follows:
 *
 * Port A
 * bits 1--0: Output Printer Data, bits 1--0
 * bit 2: Output,1; NOT Strobe
 * bit 3: Output,1; NOT Prime
 * bit 4: Input; Vertical Synchronization
 * bit 5: Output,1; Video Interrupt Enable
 * bit 6: Input; Light Pen Switch
 * bit 7: Output,1; Light Pen Interrupt Enable
 *
 * CA1: Low to High; Light Pen Strobe
 * CA2: Low to High; Video Interrupt
 *
 * Port B
 * bit 0: Input; Busy
 * bit 1: Input; NOT Fault
 * bits 7--2: Output,0; Printer Data, bits 7--2
 *
 * CB1: Input; Printer Acknowledge
 * CB2: Input; Printer Busy
 */
242 1 INIT_68A21: PROCEDURE;
243 2 DO;
/*
 * Initialize Port A
 */
244 3 OUTPUT(GEN_CNT)=(P6821_C2_1 OR P6821_OR_A OR P6821_C1_1);
245 3 OUTPUT(GEN_DAT)=0000011100B; /* Initialize Data */
246 3 OUTPUT(GEN_CNT)=(P6821_C2_1 OR P6821_C1_1);
247 3 OUTPUT(GEN_DIR)=10101111B; /* Set direction */
248 3 OUTPUT(GEN_CNT)=(P6821_C2_1 OR P6821_OR_A OR P6821_C1_1);
249 3 OUTPUT(GEN_DAT)=101011100B; /* Enable Interrupts */
/*

```

```

/*
 * INIT_8259_85 - Initialize 8259 for 8085
 *
 * INIT_8259_85 initializes the 8259's for the 8085 mode
 * of operation.
 */

```

\$ IF EXTENDED_MONITOR

INIT_8259_85: PROCEDURE PUBLIC;

```

DO;
  OUTPUT(M_ITC_0)=ICW1_ICW1 OR ICW1_LTIM OR ICW1_IC4;
  OUTPUT(M_ITC_1)=0;
  OUTPUT(M_ITC_1)=ICW3_S3;
  OUTPUT(M_ITC_1)=ICW4_FNM OR ICW4_AEOI;
  OUTPUT(M_ITC_1)=OCW1_ALL; /* Mask all interrupts */
  OUTPUT(S_ITC_0)=ICW1_ICW1 OR ICW1_LTIM OR ICW1_IC4;
  OUTPUT(S_ITC_1)=0FFH;
  OUTPUT(S_ITC_1)=3;
  OUTPUT(S_ITC_1)=ICW4_FNM OR ICW4_AEOI;
  OUTPUT(S_ITC_1)=OCW1_ALL; /* Mask all interrupts */
END;

```

\$ ENDIF

\$EJECT

```

271 1 DECLARE INITIAL_DS STRUCTURE (
      WIP(5) BYTE,
      VERSION BYTE,
      DS_SIZE WORD)
      DATA (
        @EAH,@00H,@00H,@FFH,@FFH, /* WIP(0-4)
        MTR_VERSION, /* BCD Monitor Version
        1024 /* Maximum Size for DS
      );

272 1 INIT_DATA_SEGMENT: PROCEDURE;
273 2 DO;
274 3 CALL MOV@INITIAL_DS,@WIP(0),SIZE(INITIAL_DS);
      *
      * IF @=1 /* Far Jump to MTR-100 Reset
        WIP(0)=@EAH;
        WIP(1),WIP(2)=0;
        WIP(3),WIP(4)=@FFH;
        VERSION=MTR_VERSION;
        DS_SIZE=1024;
      *
      * ENDIF /* BCD Monitor Version Identifier
        BIL=M_INT0; /* Master Interrupt Level
        BSIL=S_INT0; /* Base Slave Interrupt Level
        FONTSIZE=@357H; /* Font Table Size in Bytes
      *
      * END;
  
```

REJECT


```

290 1 INIT_TERMINAL: PROCEDURE;
291 2 DECLARE I WORD;
292 2 DO;
    /*
293 3 * Initialize RAM Vectors to ROM Routines
    */
    CALL MOVW(@INITIAL_VECTORS,@VECTORS,SIZE(VECTORS)/2);

    /*
294 3 * Initialize H-19 Parameters
    */
    HI9_PAR.CPL = 80; /* 80 Characters Per Line */
    HI9_PAR.LPS = 25; /* 25 Lines Per Screen */
    HI9_PAR.SLI = 24; /* 24 is Status Line Index */
    HI9_PAR.DSC = 9; /* Displayed Scan Lines Per Character */
    HI9_PAR.SPC = 11; /* 11 Scan Lines Per Character */
    XMT_BURST = 960/60; /* 9600 Baud, or 16 Char/Sec */

    /*
300 3 * Initialize Keyboard and Display Maps
    */
    DO I=0 TO LAST(DMAP) ;
301 4 KMAP(I),DMAP(I)=I;
302 4 END;
    IF 0=1
303 3 DO I='a'-' ' TO DEL-' ' ;
304 4 DMAP(I)=I-(DEL+' ');
305 4 END;
    IF 0=0
    ENDIF
    ENDIF
    ENDIF
    ENDIF
    ENDIF

    /*
306 3 * Reset Terminal to Power-Up Configuration
    */
    CALL P_RES;

    /*
307 3 * Enable VSYNC Interrupts
    */
    OUTPUT(GEN_CNT)=INPUT(GEN_CNT) OR P6821_C2_0;
308 3 END;
309 2 END;
310 1 END;
  
```

ASSEMBLY LISTING OF OBJECT CODE

```

006E E6E3 OUT 0E3H ; STATEMENT # 251
0070 B000 MOV AL,0H
0072 E6E2 OUT 0E2H ; STATEMENT # 252
0074 B012 MOV AL,12H
0076 E6E3 OUT 0E3H ; STATEMENT # 253
0078 B0FC MOV AL,0FCH
007A E6E2 OUT 0E2H ; STATEMENT # 254
007C B016 MOV AL,16H
007E E6E3 OUT 0E3H ; STATEMENT # 256
0080 SD POP BP
0081 C3 RET BP
; STATEMENT # 257
INIT_68A21 ENDP
; STATEMENT # 257
INIT_8259_88 PROC NEAR
0082 55 PUSH BP
0083 8BEC MOV BP,SP
; STATEMENT # 259
0085 B019 MOV AL,19H
0087 E6F2 OUT 0F2H ; STATEMENT # 260
0089 A00000 MOV AL,BIIL
008C B1F8 MOV CL,0F8H
008E 22C1 AND AL,CL
0090 E6F3 OUT 0F3H ; STATEMENT # 261
0092 B008 MOV AL,8H
0094 E6F3 OUT 0F3H ; STATEMENT # 262
0096 B013 MOV AL,13H
0098 E6F3 OUT 0F3H ; STATEMENT # 263
009A B0FF MOV AL,0FFH
009C E6F3 OUT 0F3H ; STATEMENT # 264
009E B019 MOV AL,19H
00A0 E6F0 OUT 0F0H ; STATEMENT # 265
00A2 A00000 MOV AL,BIIL
00A5 22C1 AND AL,CL
00A7 E6F1 OUT 0F1H ; STATEMENT # 266
00A9 B003 MOV AL,3H
00AB E6F1 OUT 0F1H ; STATEMENT # 267
00AD B013 MOV AL,13H
00AF E6F1 OUT 0F1H ; STATEMENT # 268
00B1 B0FF MOV AL,0FFH
00B3 E6F1 OUT 0F1H ; STATEMENT # 270
00B5 SD POP BP

```

```

0112 FE060200      INC      I          ; STATEMENT # 286
0116 75CE          JNZ      @1         ; STATEMENT # 287
                                @2:
0118 A00000        MOV      AL,BILL      ; STATEMENT # 287
011B 0406          ADD      AL,6H
011D 50            PUSH     AX            ; 1
011E BA0000        MOV      DX,SEG(VIDEO_INTR_A)
0121 B80000        MOV      AX,OFFSET(VIDEO_INTR_A)
0124 52            PUSH     DX            ; 2
0125 50            PUSH     AX            ; 3
0126 E80000        CALL     INIT_IV      ; STATEMENT # 289
0129 5D            POP      BP            ; STATEMENT # 289
012A C3            RET

                                INIT_IV
012B 55            INT_TERMINAL
012C 8BEC          PUSH     BP            ; STATEMENT # 290
                                PROC NEAR
                                BP,SP
012E BE0000        MOV      SI,OFFSET(INITIAL_VECTORS) ; STATEMENT # 293
0131 BF0000        MOV      DI,OFFSET(VECTORS)
0134 B91A00        MOV      CX,1AH
0137 1E            PUSH     DS            ; 1
0138 07            POP      ES            ; 1
0139 0E            PUSH     CS            ; 1
013A 1F            POP      DS            ; 1
013B FC            CLD
013C F2A5          REP      MOVSM
013E 06            PUSH     ES            ; 1
013F 1F            POP      DS            ; 1
0140 C606000050     MOV      HI9_PAR,50H   ; STATEMENT # 294
0145 C606020019     MOV      HI9_PAR+2H,19H ; STATEMENT # 295
014A C606040018     MOV      HI9_PAR+4H,18H ; STATEMENT # 296
014F C606010009     MOV      HI9_PAR+1H,9H  ; STATEMENT # 297
0154 C60605000B     MOV      HI9_PAR+5H,0BH ; STATEMENT # 298
0159 C606000010     MOV      XMT,10H       ; STATEMENT # 299
015E C70600000000   MOV      I,0H          ; STATEMENT # 300
                                @3:
0164 A10000        MOV      AX,I
0167 3DF00         CMP      AX,0FFH
016A 7710          JA      @4
                                ; STATEMENT # 301
016C 89C3          MOV      BX,AX
016E 88870000     DMAC[BX],AL
0172 88870000     MOV      KMAP[BX],AL
                                ; STATEMENT # 302

```

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
7	00000H	1	BEL.
67	00000H	1	BILL.
4	00000H	83	BOUL.
47	00000H	1	ROOT_PAR.
	00000H	1	INDEX.
	00001H	1	PORT.
	00020H	80	STRING.
	00520H	1	UNIT.
8	00000H	1	BS.
68	00000H	1	BSIL.
27	00000H	4	BYTEPTR.
30	00000H	1	C.
22	00000H	1	C.
7	00000H	1	CAN.
214	00000H	1	CHAR.
83	00000H	1	CHAR.
211	00000H	1	CHAR.
217	00000H	1	CHAR_DST.
217	00000H	1	CHAR_SRC.
41	00000H	1	CODE_SEG.
86	00000H	1	COL.
75	00000H	7	COLOR.
	00001H	1	FORE.
	00011H	1	BACK.
	00020H	1	MASK.
	00030H	1	CLEAR.
	00040H	1	PAINTED.
	00050H	1	FONT.
	00060H	1	COMP_FONT.
211	00000H	2	COMP_F.
217	00000H	1	COUNT.
214	00000H	1	COUNT.
10	00000H	1	CR.
76	00000H	3	CRTC_CURSOR.
	00000H	2	START.
	00020H	1	UPDATE.
77	00000H	3	CRTC_DISPLAY.
	00000H	2	START.
	00020H	1	UPDATE.
2	00020H	1	DC.
			LITERALLY '007H'
			BYTE EXTERNAL(31)
			LITERALLY 'BYTE'
			STRUCTURE EXTERNAL(11)
			BYTE
			BYTE
			BYTE ARRAY(80)
			BYTE
			LITERALLY '003H'
			BYTE EXTERNAL(32)
			POINTER IN PROC (INCB) PARAMETER
			BYTE IN PROC (SXMTIC) PARAMETER
			BYTE IN PROC (DXMTC) PARAMETER
			LITERALLY '018H'
			BYTE IN PROC (MTR_EDC) PARAMETER
			BYTE IN PROC (S_CRT) PARAMETER
			BYTE IN PROC (MTR_DFC) PARAMETER
			BYTE IN PROC (MTR_MDC) PARAMETER
			BYTE IN PROC (MTR_MDC) PARAMETER
			LITERALLY '0FE05H'
			BYTE IN PROC (DCA) PARAMETER
			STRUCTURE EXTERNAL(36)
			BYTE
			BYTE
			BYTE
			BYTE
			BYTE
			WORD IN PROC (MTR_DFC) PARAMETER
			BYTE IN PROC (MTR_MDC) PARAMETER
			BYTE IN PROC (MTR_EDC) PARAMETER
			LITERALLY '00DH'
			STRUCTURE EXTERNAL(37)
			WORD
			BYTE
			STRUCTURE EXTERNAL(38)
			WORD
			WORD
			LITERALLY 'DECLARE'
			10 11 12 13 14 15 16 17 18 19 39 40
			41 42 43 44 45 46 47 48 49 50 51 52
			53 54 55 56 57 58 59 60 61 62 63 64
			65 66 67 68 69 70 71 72 73 74 75 76
			77 78 79 80 97 98 99 100 101 102 103 104
			105 106 107 108 109 110 111 112 113 114 115 116
			117 118 119 120 121 122 123 124 125 126 127 128
			129 130 131 132 133 134 135 136 137 138 139 140
			141 142 143 144 145 146 147 148 149 150 151 152
			153 154 155 156 157 158 159 160 161 162 163 164
			165 166 167 168 169 170 171 172 173 174 175 176
			177 178 179 180 181 182 183 184 185 186 187 188
			189 190 191 192 193 194 195 196 197 198 199 200

Address	Hex	Label	Value	Address	Hex	Label	Value
225	0000H	VRAM_SIZE	1	293	0000H	PROCEDURE STACK=0000H	293
211	0000H	HORZ	1	233	0000H	PROCEDURE PUBLIC STACK=0002H	233
65	0000H	HORZ_CHAR	1	234	0000H	PROCEDURE STACK=0004H	234
16	0000H	HT	1	236	0000H	PROCEDURE EXTERNAL (6) STACK=0000H	236
281	0002H	I	1	237	0000H	PROCEDURE STACK=0004H	237
291	0000H	I	1	307	0000H	BUILTIN	307
139	0000H	ICM1_A7A5	1	244	00FFH	LITERALLY '00FFH'	244
142	0000H	ICM1_A014	1	245	00E0H	LITERALLY '00E0H'	245
144	0000H	ICM1_IC4	1	246	00FDH	LITERALLY '00FDH'	246
140	0000H	ICM1_ICM1	1	259	00F2H	LITERALLY '00F2H'	259
141	0000H	ICM1_LTIM	1	261	00F2H	LITERALLY '00F2H'	261
143	0000H	ICM1_SINGL	1	262	00F2H	LITERALLY '00F2H'	262
145	0000H	ICM2_A15A8	1	263	00F2H	LITERALLY '00F2H'	263
153	0000H	ICM3_S0	1	264	00F2H	LITERALLY '00F2H'	264
152	0000H	ICM3_S1	1	265	00F2H	LITERALLY '00F2H'	265
151	0000H	ICM3_S2	1	266	00F2H	LITERALLY '00F2H'	266
150	0000H	ICM3_S3	1	267	00F2H	LITERALLY '00F2H'	267
149	0000H	ICM3_S4	1	267	00F2H	LITERALLY '00F2H'	267
148	0000H	ICM3_S5	1	267	00F2H	LITERALLY '00F2H'	267
147	0000H	ICM3_S6	1	267	00F2H	LITERALLY '00F2H'	267
146	0000H	ICM3_S7	1	267	00F2H	LITERALLY '00F2H'	267
156	0000H	ICM4_S6	1	267	00F2H	LITERALLY '00F2H'	267
157	0000H	ICM4_AE01	1	267	00F2H	LITERALLY '00F2H'	267
155	0000H	ICM4_BUF	1	267	00F2H	LITERALLY '00F2H'	267
154	0000H	ICM4_FNM	1	267	00F2H	LITERALLY '00F2H'	267
156	0000H	ICM4_MS	1	267	00F2H	LITERALLY '00F2H'	267
133	0000H	IL_KV	1	267	00F2H	LITERALLY '00F2H'	267
127	0000H	IL_PENR	1	267	00F2H	LITERALLY '00F2H'	267
123	0000H	IL_PSWAP	1	267	00F2H	LITERALLY '00F2H'	267
134	0000H	IL_PTR	1	267	00F2H	LITERALLY '00F2H'	267
130	0000H	IL_S100	1	267	00F2H	LITERALLY '00F2H'	267
131	0000H	IL_SER_A	1	267	00F2H	LITERALLY '00F2H'	267
132	0000H	IL_SER_B	1	267	00F2H	LITERALLY '00F2H'	267
129	0000H	IL_TIMER	1	267	00F2H	LITERALLY '00F2H'	267
26	0000H	INCB	1	267	00F2H	LITERALLY '00F2H'	267
232	003CH	INIT	21	267	00F2H	LITERALLY '00F2H'	267
271	0034H	INIT_DS	8	267	00F2H	LITERALLY '00F2H'	267
271	0000H	WIP	5	267	00F2H	LITERALLY '00F2H'	267
271	0005H	VERSION	1	267	00F2H	LITERALLY '00F2H'	267
231	0004H	DS_SIZE	2	267	00F2H	LITERALLY '00F2H'	267
242	003CH	INIT	49	267	00F2H	LITERALLY '00F2H'	267
257	0082H	INIT_68A21	53	267	00F2H	LITERALLY '00F2H'	267
272	00B7H	INIT_8259_88	39	267	00F2H	LITERALLY '00F2H'	267
37	0000H	INIT_DATA_SEGMENT	77	267	00F2H	LITERALLY '00F2H'	267
280	00DEH	INIT_IV	77	267	00F2H	LITERALLY '00F2H'	267
290	012BH	INIT_TERMINAL	129	267	00F2H	LITERALLY '00F2H'	267
118	0000H	INPUT	1	267	00F2H	LITERALLY '00F2H'	267
98	0000H	ID_CRIC	1	267	00F2H	LITERALLY '00F2H'	267
115	0000H	ID_DIP	1	267	00F2H	LITERALLY '00F2H'	267
100	0000H	ID_GENERAL	1	267	00F2H	LITERALLY '00F2H'	267
109	0000H	ID_HIADR	1	267	00F2H	LITERALLY '00F2H'	267
109	0000H	ID_INT_MS	1	267	00F2H	LITERALLY '00F2H'	267

186	00000H	MTR_SCRT	PROCEDURE EXTERNAL (50)	STACK=0000H	231
188	00000H	MTR_SKB0	PROCEDURE EXTERNAL (51)	STACK=0000H	231
190	00000H	MTR_TTY_INTR	PROCEDURE EXTERNAL (52)	STACK=0000H	231
192	00000H	MTR_TTY_POLL	PROCEDURE BYTE EXTERNAL (53)	STACK=0000H	
227	00000H	MTR_UIES	PROCEDURE EXTERNAL (63)	STACK=0000H	231
42		MTR_VERSION	LITERALLY '12H'	271	
229	00000H	MTR_XCA	PROCEDURE EXTERNAL (64)	STACK=0000H	231
125		M_INT0	LITERALLY '16'	275	
135		M_ITC_0	LITERALLY 'IO_INT_MS+0'	259	
136		M_ITC_1	LITERALLY 'IO_INT_MS+1'	260	261 262 263
167		OCM1_ALL	LITERALLY '11111111B'	263	268
166		OCM1_M0	LITERALLY '000010001B'		
165		OCM1_M1	LITERALLY '000010001B'		
164		OCM1_M2	LITERALLY '000010001B'		
163		OCM1_M3	LITERALLY '000010001B'		
162		OCM1_M4	LITERALLY '000110000B'		
161		OCM1_M5	LITERALLY '001010000B'		
160		OCM1_M6	LITERALLY '010010000B'		
159		OCM1_M7	LITERALLY '100010000B'		
170		OCM2_E01	LITERALLY '001010000B'		
174		OCM2_L0	LITERALLY '000010001B'		
173		OCM2_L1	LITERALLY '000010001B'		
172		OCM2_L2	LITERALLY '000010001B'		
171		OCM2_OCM2	LITERALLY '000010000B'		
168		OCM2_ROT	LITERALLY '100010000B'		
169		OCM2_SE01	LITERALLY '010010000B'		
175		OCM3_ESNM	LITERALLY '010010000B'		
177		OCM3_DCM3	LITERALLY '000011000B'		
178		OCM3_POLL	LITERALLY '000010100B'		
176		OCM3_SNM	LITERALLY '001010000B'		
179		OCM3_SRS	LITERALLY '000010010B'		
		OUTPUT	BUILTIN		
			253* 254* 259* 260* 261* 262* 263* 264* 265* 266* 267* 268*		
			307*		
198		P6821_A	LITERALLY '0'		
199		P6821_B	LITERALLY '2'		
207		P6821_C1_0	LITERALLY '001H'	244	246 248 250 252 254
206		P6821_C1_1	LITERALLY '002H'	244	246 248 250 252 254
204		P6821_C2_0	LITERALLY '003H'	307	
203		P6821_C2_1	LITERALLY '010H'	244	246 248 250 252 254
202		P6821_C2_2	LITERALLY '020H'		
200		P6821_IR01	LITERALLY '080H'		
201		P6821_IR02	LITERALLY '040H'	244	248 250 254
205		P6821_GRA	LITERALLY '004H'	244	248 250 254
180		PRN_CNT	LITERALLY 'IO_PRINTER+1'	251	252 254
181		PRN_DIR	LITERALLY 'IO_PRINTER'	253	
182		PRN_DIR	LITERALLY 'IO_PRINTER'	253	
56	00000H	PROMPT	POINTER EXTERNAL (20)		
69	00000H	PSP	BYTE EXTERNAL (33)		
70	00000H	P_ELLIN	PROCEDURE EXTERNAL (46)	STACK=0000H	
72	00000H	P_EPAD	PROCEDURE EXTERNAL (47)	STACK=0000H	
88	00000H	P_HOM	PROCEDURE EXTERNAL (45)	STACK=0000H	
94	00000H	P_RES	PROCEDURE EXTERNAL (48)	STACK=0000H	306
57	00000H	RDC	POINTER EXTERNAL (21)		
35	00000H	REGP	POINTER IN PROC (XEC) PARAMETER		35
70	00000H	REGP	POINTER EXTERNAL (34)		

SERIES-111 PL/M-86 V2.0 COMPILATION OF MODULE FNCLIB
OBJECT MODULE PLACED IN :F2:FNCLIB.OBJ
COMPILER INVOKED BY: PLM86.86 :F2:FNCLIB.PLM CODE PRINT(:TO:)

```
1 $TITLE('MTR-101 Z-Machine Monitor ROM: Function Library')
  $SUBTITLE('Compiler Controls')
  $INCLUDE('F2:COMMON.H')
  $SUBTITLE('Compiler Controls')
  $OPTIMIZE(3)
  $NOOVERFLOW
  $COMPACT
  $ROM
  $XREF
  $LARGE(MTR100 EXPORTS MTR_MON;
  = $ EXPORTS MTR_MON;
  = $ EXPORTS MTR_SWIM;
  = $ EXPORTS MTR_DCRT;
  = $ EXPORTS MTR_DKED;
  = $ EXPORTS MTR_SCRIPT;
  = $ EXPORTS MTR_SKED;
  = $ EXPORTS MTR_TTY_INTR;
  = $ EXPORTS MTR_TTY_POLL;
  = $ EXPORTS MTR_IRET)
  $LARGE(VIDEQA EXPORTS MTR_DC1;
  = $ EXPORTS MTR_DFC;
  = $ EXPORTS MTR_EDC;
  = $ EXPORTS MTR_FONT;
  = $ EXPORTS MTR_MIC;
  = $ EXPORTS MTR_MDL;
  = $ EXPORTS MTR_PROMPT;
  = $ EXPORTS MTR_RDCC;
  = $ EXPORTS MTR_ULES;
  = $ EXPORTS MTR_XCA)
  $LARGE(ASLIB EXPORTS VIDEO_INTR_A)
  $RESET(EXTENDED_MONITOR)
  $SUBTITLE('External Definitions')
  FNCLIB:
  DO;
  $INCLUDE('F2:LEXCAL.H')
  $SAVE
  $NCLIST
  $INCLUDE('F2:IODEF.H')
  $SAVE
  $NCLIST
  $INCLUDE('F2:ASCII.H')
  $SAVE
  $NCLIST
  $INCLUDE('F2:ASTLIB.H')
  $SAVE
  $NCLIST
```

```

/*
 * CRLF -- Carriage Return / Line-Feed
 *
 * 'CRLF' outputs a carriage-return/line-feed
 * sequence to the console.
 */
CRLF: PROCEDURE PUBLIC;
DO;
CALL WRITEC(CR);
IF NOT HI9_MODE.AUTO_LF THEN
CALL WRITEC(LF);
END;
183 1
184 2
185 3
186 3
187 3
188 3
189 2

```

```

/*
 * GFW -- Get Far Word
 *
 * GFW gets the specified word based on the supplied
 * paragraph/segment and offset.
 */
NOTE: The actual routine which performs the work
is contained in 'ASTLIB', the assembly lan-
guage assist library.
Entry: base = paragraph/segment
offset = Offset
Exit: Returns WORD addressed by the base/offset pair
*/
IF 1=2

```

```

GFW: PROCEDURE(base,offset) WORD PUBLIC;
DECLARE base WORD,
offset WORD;
DO;
RETURN(GFB(base,offset)+SHL(GFB(base,offset+1),8));
END;
$
ENDIF

```

/*


```

* IF EXTENDED_MONITOR
*
* PROCEDURE(delim) BYTE PUBLIC;
* DECLARE delim BYTE;
* DECLARE C BYTE;
*
* DO;
*   C=MCU(READC);
*   DO WHILE (C<>delim) AND (C<>DEL) AND (C<>BS) AND (NOT VHC(C)) ;
*     CALL WRITC(BEL);
*     C=MCU(READC);
*   END;
*   RETURN(C);
* END;
* END;
* ENDIF
    
```

```

/*
* IHV -- Input Hex Value
*
* IHV inputs a hex value from the console. The
* value is to be terminated by the supplied de-
* limiter. Other interpretable characters are
* DEL and BS. If DEL or BS are hit, the routine
* returns immediately, otherwise, the value at
* the supplied address is modified according to
* the attribute specified. If 'size' is 1, then
* a byte value is assumed. If 'size' is 2, then
* a word value is assumed.
*
* NOTE: Delimiters which are valid hex characters
* should be avoided, and that at least one
* valid hex character is required before
* the delimiter is accepted.
*
* Entry: delim = required value delimiter
*         val$ = value pointer
*         size = 1 for byte value, 2 for word value
*
* Exit: Returns the actual delimiter entered
*        val$ modified if the terminating character
*        was not 'delim'.
*
* */
    
```

```

* IF EXTENDED_MONITOR
    
```

```

IHV: PROCEDURE(delim, val$, size) BYTE PUBLIC;
      DECLARE delim BYTE;
      DECLARE val$ POINTER;
      DECLARE size BYTE;
      DECLARE C BYTE;
    
```

```

193 3 IF 'a' <= char AND char <= 'z'
      THEN
194 3 RETURN (char-'a'+1);
195 3 ELSE
196 3 RETURN (char);
197 2 END;
    
```

```

/*
*
* OHB outputs a hex byte to the console.
*
* Entry: dat = byte value to output
*
* Exit: NONE
*
*/
    
```

```

$ IF EXTENDED_MONITOR
OHB: PROCEDURE(dat) PUBLIC;
DECLARE dat BYTE;

DO;
CALL OHB(SHR(dat,4));
CALL OHB(dat AND 00FH);
END;
ENDIF
    
```

```

/*
*
* OHC -- Output Hex Character
*
* OHC outputs a hex character to the console.
* This routine must be called carefully, as
* the data supplied is not checked for valid-
* ity.
*
* Entry: dat = Binary value to output the hex character for
*
* Exit: NONE
*
*/
    
```

```

$ IF EXTENDED_MONITOR
OHC: PROCEDURE(dat) PUBLIC;
DECLARE dat BYTE;
    
```

```

*
* WRITES outputs a string of text to the console.
* This is accomplished by outputting the bytes spec-
* ified by the supplied string pointer, until a
* terminating byte is found. A terminator is any
* byte with the MSB set.
*
* WRITEC is invoked for the console output.
*
* Entry: string#p = pointer to terminated byte string
*
* Exit: NONE
*
*/
WRITEC:
    PUBLIC:
    POINTER;
    INTEGER;
    BYTE;
    DECLARE string#p
    DECLARE I
    STRING BASED string#p(1)
213 1
214 2
215 2
    IF @=1
    DO;
    I = 0;
    DO WHILE (STRING(I) AND EOL)=0 ;
    CALL WRITEC(STRING(I));
    I = I + 1;
    END;
    CALL WRITEC(STRING(I) AND 07FH);
    END;
    ELSE
    DO;
    I = -1;
    WRITES1;
    DO;
    I = I+1 ;
    CALL WRITEC(STRING(I));
    IF (STRING(I) AND EOS)=0 THEN
    GO TO WRITES1;
    END;
    END;
    ENDDIF
216 2
217 3
218 3
219 4
220 4
221 4
222 4
223 4
224 3
    END;
225 2
226 1
    END;

```

```

0046 8BEC      MOV      BP,SP      ; STATEMENT # 201
0048 E80000    CALL     READK
004B A20200    MOV      T,AL      ; STATEMENT # 202
004E B1B0      MOV      CL,@B0H
0050 3AC8      CMP      CL,AL
0052 7704      JA       @5
0054 3CB9      CMP      AL,@B7H
0056 760F      JBE      @6
0058 A00200    @5:  MOV      AL,T
005B 3C8D      CMP      AL,8DH
005D 7408      JZ       @6
005F 3CAD      CMP      AL,@ADH
0061 7404      JZ       @6
0063 3CAE      CMP      AL,@AEH
0065 7505      JNZ     @4
0067 80260207F AND     T,7FH      ; STATEMENT # 203
006C A00200    @4:  MOV      AL,T
006F 5D       POP      BP
0070 C3       RET
0071 55       WRITTEC  PROC NEAR
0072 8BEC      MOV      BP,SP      ; STATEMENT # 210
0074 FF7604    PUSH    [BP],CHAR; 1
0077 E80000    CALL    S_CRT      ; STATEMENT # 212
007A 5D       POP      BP
007B C20200    RET      2H
007E 55       WRITTEC  PROC NEAR
007F 8BEC      MOV      BP,SP      ; STATEMENT # 213
0081 C706000FFF MOV     I,0FFFFH  ; STATEMENT # 217
0087 A10000    MOV     AX,I      ; STATEMENT # 219
008A 40       INC     AX
008B A30000    MOV     I,AX
008E 89C6      MOV     SI,AX      ; STATEMENT # 220
0090 C4E04    LES     BX,[BP],STRNGP
0093 26FF30    PUSH   ES:[BX],STRNGCSI
0096 E8DBFF    CALL   WRITTEC      ; STATEMENT # 221
    
```

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
34	0000H	1	BEL.
107	0000H	1	BILL
4	0000H	1	BOOL
87	0000H	83	BOOT_PAR
	0000H	1	INDEX
	0001H	1	PORT
	0002H	80	STRING
	0052H	1	UNIT
35	0000H	1	BS
108	0000H	1	BSTL
54	0000H	4	BYTEPTR
57	0000H	1	C
49	0000H	1	C
36	0004H	1	CAN
191	0004H	1	CHAR
208	0004H	1	CHAR
123	0000H	1	CHAR
180	0000H	1	CMND
81	0000H	1	CODE_SEQ
126	0000H	1	COL
115	0000H	7	COLOR
	0000H	1	FORE
	0001H	1	BACK
	0002H	1	MASK
	0003H	1	CLEAR
	0004H	1	PAINTED
	0005H	1	FONT
	0006H	1	COMP_FONT
37	0010H	24	CR
183	0000H	3	CRLE
116	0000H	2	CRTC_CURSOR
	0000H	2	START
	0002H	1	UPDATE
117	0000H	3	CRTC_DISPLAY
	0000H	2	START
	0002H	1	UPDATE
2	0002H	1	DC
125	0000H	4	DCA
88	0000H	4	DCL
38	0000H	4	DEL
89	0000H	4	DFC

LITERALLY '007H'									
BYTE EXTERNAL(30)									
LITERALLY 'BYTE'	111	116	117	119	121	174	175		
STRUCTURE EXTERNAL(10)									
BYTE									
BYTE									
BYTE ARRAY(80)									
BYTE									
LITERALLY '008H'									
BYTE EXTERNAL(31)									
POINTER IN PROC (INCB) PARAMETER									
BYTE IN PROC (SXMTD) PARAMETER									
BYTE IN PROC (DXMTD) PARAMETER									
LITERALLY '018H'									
BYTE IN PROC (MCU) PARAMETER AUTOMATIC									
BYTE IN PROC (WRITD) PARAMETER AUTOMATIC									
BYTE IN PROC (S CRT) PARAMETER									
BYTE IN PROC (WRITK) PARAMETER									
LITERALLY '0FE0SH'									
BYTE IN PROC (DCA) PARAMETER									
STRUCTURE EXTERNAL(35)									
BYTE									
BYTE									
BYTE									
BYTE									
BYTE									
LITERALLY '00DH'									
PROCEDURE PUBLIC STACK=0006H									
STRUCTURE EXTERNAL(36)									
WORD									
BYTE									
STRUCTURE EXTERNAL(37)									
WORD									
BYTE									
LITERALLY 'DECLARE'									
10	11	12	13	14	15	16	17	18	19
22	23	24	25	26	27	28	29	30	31
34	35	36	37	38	39	40	41	42	43
46	46	67	68	69	70	71	72	73	74
77	78	79	80	81	82	83	84	85	86
89	90	91	92	93	94	95	96	97	98
101	102	103	104	105	106	107	108	109	110
113	114	115	116	117	118	119	120	137	138
141	142	143	144	145	146	147	148	149	150
153	154	155	156	157	158	159	160	161	162
165	166	167	168	169	170	171	172	173	

182	000001H	16	HEX_DIGIT.	BYTE ARRAY(16) PUBLIC DATA			
105	000001H	1	HORZ_CHAR.	BYTE EXTERNAL(28)			
43	000003H	1	VRAM_SIZE.	BYTE			
215	000004H	2	I.	INTEGER IN PROC (WRITES)	217*	219*	219 220 221
53	000001H		INCB.	PROCEDURE EXTERNAL(2) STACK=0000H			
64	000001H		INIT_ITV.	PROCEDURE EXTERNAL(6) STACK=0000H			
27			IO_CRTC.	LITERALLY '00DCH'			
7			IO_DIP.	LITERALLY '00FFH'			
24			IO_GENERAL.	LITERALLY '00E0H'			
9			IO_HIADR.	LITERALLY '00F0H'			
18			IO_INT_MS.	LITERALLY '00F2H'			
19			IO_INT_SL.	LITERALLY '00F0H'			
17			IO_KEYBRD.	LITERALLY '00F4H'			
26			IO_LTPN.	LITERALLY '00DEH'			
10			IO_MEM.	LITERALLY '00FCH'			
23			IO_PRINTER.	LITERALLY '00E2H'			
12			IO_RSRV_1.	LITERALLY '00FAH'			
14			IO_RSRV_2.	LITERALLY '00F3H'			
15			IO_RSRV_3.	LITERALLY '00F7H'			
16			IO_RSRV_4.	LITERALLY '00F6H'			
25			IO_RSRV_5.	LITERALLY '00DFH'			
29			IO_RSRV_6.	LITERALLY '00C0H'			
21			IO_SER_A.	LITERALLY '00E3H'			
20			IO_SER_B.	LITERALLY '00E3H'			
13			IO_SOUND.	LITERALLY '00F9H'			
8			IO_SWAP.	LITERALLY '00E4H'			
22			IO_TIMER.	LITERALLY '00FBH'			
11			IO_TSTAT.	LITERALLY '00D8H'			
28			IO_VIDEO.	LITERALLY '00B0H'			
31			IO_Z207_0.	LITERALLY '00B0H'			
30			IO_Z207_1.	LITERALLY '00A8H'			
33			IO_Z217_0.	LITERALLY '00A8H'			
32			IO_Z217_1.	LITERALLY '00ACH'			
65	000004H	1	IVI.	BYTE IN PROC (INIT_ITV) PARAMETER			65
65	000004H	4	IVP.	POINTER IN PROC (INIT_ITV) PARAMETER			65
143			KC_ARF.	LITERALLY '02H'			
144			KC_ARO.	LITERALLY '01H'			
145			KC_BEP.	LITERALLY '07H'			
150			KC_CFI.	LITERALLY '05H'			
151			KC_CLK.	LITERALLY '06H'			
146			KC_DI.	LITERALLY '0DH'			
147			KC_ET.	LITERALLY '0CH'			
152			KC_KBD.	LITERALLY '09H'			
153			KC_KBE.	LITERALLY '08H'			
148			KC_KCF.	LITERALLY '04H'			
149			KC_KCO.	LITERALLY '03H'			
154			KC_MNS.	LITERALLY '0BH'			
155			KC_MUD.	LITERALLY '0AH'			

214	00004H	4	SIRNGP	214	220	221
56	00000H		SXMTG.			
122	00000H		S_CRT.			
98	00000H	4	S_XMTC	210		
179	00020H	1	T.	201*	202	203
136	00000H		TESTK.			204
175	00000H		TRUE			
6	00000H	4	UIES			
97	00000H	52	VECTORS.			
101	00000H	1	VERSION.			
85	00000H	1	VERT_LINE.			
106	00000H	1	VIDEO_INTR_A			
59	00000H		VT			
45	00000H	5	WIP.			
84	00000H	13	WRITEC			
207	00710H	44	WRITES.	185	187	220
213	007EH		WRITES1.			
179	00000H		XCA.			
100	00000H	4	XCOLOR_BACK.			
114			XCOLOR_FORE.			
113			XEC.			
61	00000H	10	XMT.			
121	00000H	1	BURST.			
	00001H	2	BCOUNT.			
	0003H	2	COUNT.			
	0005H	1	COL.			
	0006H	1	ROW.			
	0007H	1	COLOR.			
	0008H	1	GRAPHIC.			
	0009H	1	REVERSE.			
46			XOFF.			
47			XON.			
112			XREVERSE			

MODULE INFORMATION:

CODE AREA SIZE = 000AAH 170D
 CONSTANT AREA SIZE = 00000H 0D
 VARIABLE AREA SIZE = 0003H 3D
 MAXIMUM STACK SIZE = 0010H 16D
 1125 LINES READ
 0 PROGRAM WARNINGS
 0 PROGRAM ERRORS

END OF PL/M-86 COMPILATION

```
/*  
* I2661  
* * I2661 initializes the specified 2661 EPCI.  
* *  
* * Entry: base = Base address of port  
* * Exit: NONE  
* */
```

```
$ IF EXTENDED_MONITOR  
I2661: PROCEDURE(base) PUBLIC;  
DECLARE base WORD;
```

```
DO;  
OUTPUT(0EAH)=04EH;  
OUTPUT(0EAH)=05DH;  
OUTPUT(0EBH)=027H;  
END;  
ENDIF
```

```
/*  
* RCHAR  
* * RCHAR returns a character from the specified EPCI.  
* * RCHAR waits for the character task-time.  
* *  
* * Entry: byte = The base address of the EPCI  
* * Exit: Byte returned, with the parity bit stripped  
* */
```

```
$ IF EXTENDED_MONITOR  
RCHAR: PROCEDURE(base) BYTE PUBLIC;  
DECLARE base WORD;
```

```
DO;  
DO WHILE NOT TCHAR(base) ;  
END;  
RETURN (INPUT(0EBH) AND 07FH);  
END;
```



```
MOCHAR: PROCEDURE(base, char) PUBLIC:
DECLARE base WORD,
char BYTE;
00;
OUTPUT(0ESH+3)=INPUT(0ESH+3) OR 1;
DO WHILE (INPUT(0E9H) AND @01H)=0;
END;
OUTPUT(0E9H)=char;
END;
* ENDIF
7 1 END;
```

DEFN ADDR SIZE NAME, ATTRIBUTES, AND REFERENCES

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
4			BOOL LITERALLY 'BYTE'
2			DC LITERALLY 'DECLARE'
5	0000H		FALSB. LITERALLY '000H'
3			IOLIB. PROCEDURE STACK=0000H
6			LI LITERALLY 'LITERALLY'
			TRUE LITERALLY '0FFH'

MODULE INFORMATION:

CODE AREA SIZE = 0000H 0D
 CONSTANT AREA SIZE = 0000H 0D
 VARIABLE AREA SIZE = 0000H 0D
 MAXIMUM STACK SIZE = 0000H 0D
 191 LINES READ
 0 PROGRAM WARNINGS
 0 PROGRAM ERRORS

END OF PL/M-86 COMPILATION

) AVAIL ROOM PUBLIC;

\$SUBTITLE('Key-Board Procedures')

```

76 1 1 READK: PROCEDURE BYTE PUBLIC;
77 2 2 DECLARE C BYTE;
78 3 3 DO;
79 4 4 DO WHILE NOT TESTK ;
80 5 5 END;
81 6 6 DISABLE;
82 7 7 IF KEY.AVAIL
83 8 8 THEN
84 9 9 DO;
85 10 10 C=KEY.CHAR;
86 11 11 KEY.AVAIL=FALSE;
87 12 12 END;
88 13 13 ELSE
89 14 14 C=INPUT(KEY..DATA);
90 15 15 ENABLE;
91 16 16 RETURN(C);
END;
  
```

*REJECT

```

; STATEMENT # 6
; STATEMENT # 33
; STATEMENT # 69
; STATEMENT # 71
; STATEMENT # 73
; STATEMENT # 75
; STATEMENT # 76
; STATEMENT # 79
; STATEMENT # 81
; STATEMENT # 82
; STATEMENT # 84
; STATEMENT # 85
; STATEMENT # 87
; STATEMENT # 88
; STATEMENT # 89
; STATEMENT # 91
; STATEMENT # 92

```

TESTK PROC NEAR

0000 55 PUSH BP

0001 8BEC MOV BP,SP

; STATEMENT # 73

0003 E4F5 IN OF5H
 0005 2401 AND AL,1H
 0007 08C0 OR AL,AL
 0009 B0FF MOV AL,0FFH
 000B 7501 JNZ \$+3H
 000D 40 INC AX
 000E 0A060100 OR AL,KEY+1H
 0012 5D POP BP
 0013 C3 RET

; STATEMENT # 75

TESTK ENDP

PROC NEAR

0014 55 PUSH BP

0015 8BEC MOV BP,SP

; STATEMENT # 79

@1:

0017 E8E6FF CALL TESTK

001A D0D8 RCR AL,1

001C 73F9 JNB @1

; STATEMENT # 81

001E FA CLI

; STATEMENT # 82

001F A00100 MOV AL,KEY+1H

0022 D0D8 RCR AL,1

0024 730D JNB @3

; STATEMENT # 84

0026 A00000 MOV AL,KEY

0029 A20200 MOV C,AL

; STATEMENT # 85

002C C606010000 MOV KEY+1H,0H

0031 EB05 JMP @4

; STATEMENT # 87

0033 E4F4 IN OF4H

0035 A20200 MOV C,AL

; STATEMENT # 88

@4:

0038 FB STI

; STATEMENT # 89

0039 A00200 MOV AL,C

003C 5D POP BP

003D C3 RET

; STATEMENT # 91

READK ENDP

PROC NEAR

003E 55 PUSH BP

; STATEMENT # 92

WRITK

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
4	0002H	1	BOOL
77	0004H	1	C
93	0004H	1	CMND
2			DC
5			FALSE
27			INPUT
7			IO_CRIC
24			IO_DIP
9			IO_GENERAL
18			IO_HIADR
19			IO_INIT_MS
17			IO_INT_3L
26			IO_KEYERD
10			IO_LFPN
23			IO_MEM
12			IO_PRINTER
14			IO_RSRV_1
15			IO_RSRV_2
16			IO_RSRV_3
25			IO_RSRV_4
29			IO_RSRV_5
21			IO_RSRV_6
20			IO_SERV_A
13			IO_SERV_B
8			IO_SOUND
22			IO_SWAP
11			IO_TIMER
28			IO_VSTAT
31			IO_VIDEO
30			IO_ZZ07_0
33			IO_ZZ07_1
32			IO_ZZ17_0
39			IO_ZZ17_1
40			KC_ANF
41			KC_AR0
46			KC_REP
47			KC_CFI
42			KC_CLK
43			KC_DI
48			KC_EI
49			KC_KBD
44			KC_KBE
45			KC_KCF
50			KC_KCO
51			KC_MNS
52			KC_MUD
			KC_RES
			LITERALLY 'BYTE'
			BYTE IN PROC (READ)
			BYTE IN PROC (WRITE)
			LITERALLY 'DECLARE'
			10 11 12 13 14 15 16 17 18 19 20 21
			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 61 62 63 64 65 66 67 68 69
			LITERALLY '000H'
			BULLFIN
			LITERALLY '00DCH'
			LITERALLY '00FH'
			LITERALLY '00FH'
			LITERALLY '00FH'
			LITERALLY '00FDH'
			LITERALLY '00F2H'
			LITERALLY '00F4H'
			LITERALLY '00F4H'
			LITERALLY '00F4H'
			LITERALLY '00FCH'
			LITERALLY '00FCH'
			LITERALLY '00F2H'
			LITERALLY '00F3H'
			LITERALLY '00F7H'
			LITERALLY '00F6H'
			LITERALLY '00F6H'
			LITERALLY '00C0H'
			LITERALLY '00E8H'
			LITERALLY '00E8H'
			LITERALLY '00F9H'
			LITERALLY '00F4H'
			LITERALLY '00FBH'
			LITERALLY '00B8H'
			LITERALLY '00B0H'
			LITERALLY '00B8H'
			LITERALLY '00B8H'
			LITERALLY '00A8H'
			LITERALLY '00A8H'
			LITERALLY '02H'
			LITERALLY '01H'
			LITERALLY '07H'
			LITERALLY '05H'
			LITERALLY '06H'
			LITERALLY '0DH'
			LITERALLY '0CH'
			LITERALLY '09H'
			LITERALLY '08H'
			LITERALLY '04H'
			LITERALLY '03H'
			LITERALLY '0BH'
			LITERALLY '0AH'
			LITERALLY '00H'

SERIES-III PL/M-86 V2.0 COMPILATION OF MODULE SUBLIB
 OBJECT MODULE PLACED IN :F2:SUBLIB.OBJ
 COMPILER INVOKED BY: PLM86.86 :F2:SUBLIB.PLN CODE PRINT(:TO:)

```

1
$TITLE('MIR-101 Z-Machine Monitor ROM: Subroutine Library')
$SUBTITLE('Compiler Controls')
$INCLUDE('F2:COMMON.H')
$SUBTITLE('Compiler Controls')
$OPTIMIZE(3)
$NOOVERFLOW
$COMPACT
$FROM
$XREF
$LARGE(MTR100 EXPORTS MTR_MON;
EXPORTS MTR_MON;
EXPORTS MTR_SWIM;
EXPORTS MTR_DCRT;
EXPORTS MTR_DKRD;
EXPORTS MTR_SCRT;
EXPORTS MTR_SKED;
EXPORTS MTR_TTY_INTR;
EXPORTS MTR_TTY_POLL;
EXPORTS MTR_IRET)
$LARGE(VIDEQA EXPORTS MTR_DC1;
EXPORTS MTR_DFC;
EXPORTS MTR_EDC;
EXPORTS MTR_FONT;
EXPORTS MTR_MDC;
EXPORTS MTR_MDL;
EXPORTS MTR_PROMPT;
EXPORTS MTR_RDC;
EXPORTS MTR_LIES;
EXPORTS MTR_XCA)
$LARGE(FONT EXPORTS MTR_FONT)
$LARGE(ASTLIB EXPORTS VIDEO_INTR_A)
$RESET(EXTENDED_MONITOR)
$SUBTITLE('Definitions')
SUBLIB:
DU;
$INCLUDE('F2:LEXCAL.H')
$SAVE
$NOLIST
$INCLUDE('F2:STRUCT.H')
$SAVE
/*
* STRUCT.H
*
* This file defines all of the structures for
* MIR-100.
*

```

```

/*
**      IHA      - Input Hex Address
**
**      IHA inputs a hex address from the console. The
**      format enforced is:
**
**      nnnn:mmm
**
**      where 'nnnn' is the paragraph, and 'mmm' is the
**      offset within the paragraph.
**
**      NOTE: Any partially entered address may result
**      in partially modified parameters. Further...
**      more, the same cautions about 'delim' as
**      outlined in 'IHV()' apply since 'IHV()' is
**      invoked.
**
**      Entry:  delim = Required delimiter
**              addr$* = Pointer to address structure
**
**      Exit:   Returns actual delimiter
**
**      */
$      IF EXTENDED_MONITOR
IHA:  PROCEDURE(delim, addr$*)      BYTE PUBLIC;
      DECLARE      delim          BYTE;
                   addr$*        POINTER;
      DECLARE      DELIMITER      BYTE;
                   I              BYTE;
                   ADDR_          BASED addr$*  ADDR;

      DO:
      I=0;
      DO WHILE 0=0 ;
      IF I=0
      THEN
      DO:
      DELIMITER=IHV('?', @ADDR_..BASE, 2);
      IF DELIMITER<>'?' THEN
      RETURN(DELIMITER);
      I=1;
      ELSE
      DO:
      CALL WRITEC(DELIMITER);
      DELIMITER=IHV(delim, @ADDR_..OFFS, 2);
      IF DELIMITER<>delim
      THEN
      DO:
      CALL RUBOUT;
    
```



```

$      IF EXTENDED_MONITOR
      DECLARE REG_SPEC (*) BYTE DATA
      ( 'ESSSDSDISIPSPDXCXBAXI'PCSF1DLDHCLCHBLHALAH' );
/*
*      IRI      - Input Register Identifier
*      *      IRI inputs a valid register name, and returns the
*      *      appropriate Index into the register structure.
*      *      Entry: NONE
*      *      Exit: Returns -1 for delete or back-space, other wise
*      *      register index into structure.
*      */

IRI:
DECLARE      PROCEDURE      BYTE,
C            BYTE,
C1          BYTE,
I          BYTE,
J          BYTE,
MATCH      BYTE,
PTR        BYTE;
      BYTE PUBLIC;

      DO;
      I = 0 ;
      DO WHILE 0=0 ;
      IF I<2 THEN
      C = MCU(READC);
      IF I=0 AND (C=DEL OR C=RS) THEN
      RETURN(-1);
      ELSE IF C=DEL OR C=RS THEN
      DO;
      CALL RUBOUT;
      I = I - 1;
      END;
      ELSE IF I=0 THEN
      DO;
      MATCH = 0 ;
      DO J=0 TO LENGTH(REG_SPEC)--2 BY 2 ;
      IF C = REG_SPEC(J) THEN
      DO;
      MATCH = MATCH + 1 ;
      PTR = J ;
      END;
      IF MATCH = 0 THEN
      MATCH = 0 THEN
      CALL WRITEC(BEL);
      ELSE IF MATCH = 1 THEN
      DO;
      CALL WRITEC(C);
      C1 = C ;
      
```

```

/*
 *
 * OHA      -- Output Hex Address
 *
 * OHA outputs the specified hex address in the same
 * format specified input by 'IHA()':
 *
 *      nnnn:mmm
 *
 * where 'nnnn' is the segment, and 'mmm' is the
 * offset within the segment.
 *
 * Entry:  addr$P = Pointer to address structure
 *
 */

```

```

$
  IF EXTENDED_MONITOR
    OHA:
      PROCEDURE(addr$P)      PUBLIC;
      DECLARE  addr$P        POINTER;
      DECLARE  ADDR_         BASED addr$P  ADDR;
      DO;
        CALL OHM(ADDR_+BASE);
        CALL WRITEC(' ');
        CALL OHM(ADDR_+OFFS);
      END;
  END;
$
  ENDIF

```

```

/*
 *
 * RNC      -- Require Next Character
 *
 * RNC requires the next character from the input device
 * to be the specified one. If the character is DEL or
 * BS, RNC returns this character as the delimiter. If
 * the character is any character besides the specified
 * delimiter, RNC outputs a bell, and waits.
 *
 * Entry:  delim = Character required
 *
 * Exit:   Returns the delimiter actually read as a BYTE
 *
 */

```

```

$
  IF EXTENDED_MONITOR
    RNC:
      PROCEDURE(delim)      PUBLIC;
      DECLARE  delim        BYTE;
      DECLARE  C             BYTE;

```

```
/*  
* RUBOUT ... Rub out a character  
*  
* RUBOUT rubs out a character currently displayed on  
* the screen. This is done by simply outputting a  
* backspace, space, and backspace.  
*  
* Entry: NONE  
*  
* Exit: NONE  
*  
*/
```

```
* IF EXTENDED_MONITOR  
RUBOUT: PROCEDURE PUBLIC;  
DO;  
CALL WRITES(@CS, ' ', BSHEOS);  
END;  
END;  
* ENDIF
```

36 1 END;

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
7	00000H		ADDR
8	00000H		BEQ
4	00000H		BEQ
9	00000H		BEQ
10	00000H		CAN
26	00000H	1	CHAR
31	00000H	1	CHAR
11	00000H		CR
23	00000H		CRLF
2	00000H		DC
12	00000H		DEL
13	00000H		EDL
14	00000H		EDS
15	00000H		ESC
5	00000H		FALSE
16	00000H		FF
22	00000H		HEX_DIGIT
17	00000H		HT
18	00000H		LF
3	00000H		LT
25	00000H		MCU
28	00000H		READC
34	00000H	4	STRNGP
6	00000H		SUBLIB
19	00000H		TRUE
30	00000H		VT
33	00000H		WRITEC
20	00000H		WRITES
21	00000H		XOFF
			XON

MODULE INFORMATION:

CODE AREA SIZE = 00000H 0D
 CONSTANT AREA SIZE = 00000H 0D
 VARIABLE AREA SIZE = 00000H 0D
 MAXIMUM STACK SIZE = 00000H 0D
 501 LINES READ
 0 PROGRAM WARNINGS
 0 PROGRAM ERRORS

END OF PL/M-86 COMPILATION

```
= $SAVE INCLUDE (':F2:COLORS.H')
=$NQLIST
=$SAVE INCLUDE (':F2:CRTC.H')
=$NQLIST
=$SAVE INCLUDE (':F2:DIPDEF.H')
=$NQLIST
=$SAVE INCLUDE (':F2:FNCLIB.H')
=$NQLIST
=$SAVE INCLUDE (':F2:GLOBAL.H')
=$NQLIST
=$SAVE INCLUDE (':F2:H19CRT.H')
=$NQLIST
=$SAVE INCLUDE (':F2:I8086.H')
=$NQLIST
=$SAVE INCLUDE (':F2:ITCDEF.H')
=$NQLIST
=$SAVE INCLUDE (':F2:KEYDEF.H')
=$NQLIST
=$SAVE INCLUDE (':F2:KEYLIB.H')
=$NQLIST
=$SAVE INCLUDE (':F2:MISC.H')
=$NQLIST
=$SAVE INCLUDE (':F2:MTR100.H')
=$NQLIST
=$SAVE INCLUDE (':F2:P6821.H')
=$NQLIST
=$SUBTITLE('Global Declarations')
```

```

/*
 *
 * INIT_VIDEO      -- Initialize Video
 *
 * INIT_VIDEO initializes the video parameters. Initially,
 * the CRT-C parameters are set. Then the 6821 controlling
 * access to the video command bits is initialized. The
 * 6821's must be carefully initialized to avoid accidentally
 * 'glitching' the screen. Then, the Video RAM Mapping Mod-
 * ule is initialized. Since it only controls access to the
 * pages of video RAM from the CPU, one need not take as much
 * care in the initialization.
 *
 * If the first byte of the blue page is modifiable, then
 * it is assumed that all of the color planes are present,
 * and update from all of the planes is enabled, otherwise,
 * only the green plane is enabled (though the multiple-write
 * bits for all planes are set.
 *
 * Entry:  NONE
 * Exit:   NONE
 */
INIT_VIDEO:
DO;
DECLARE
  I          I          BYTE,
  P1#PTR    P1#PTR    POINTER,
  P1         P1         BASED P1#PTR  BYTE,
  P2#PTR    P2#PTR    POINTER,
  P2         P2         BASED P2#PTR  BYTE,
  P3         P3         BASED P1#PTR(1) BYTE,
  T1        T1        BYTE,
  T2        T2        BYTE;
/*
 * Initialize CRT-Controller, and sub-system
 */
IF (INPUT(IO_DIP) AND DIP_50HZ) <> 0
THEN
  P1#PTR=@CRTC_REG_50(0);
ELSE
  P1#PTR=@CRTC_REG_60(0);
DO I=0 TO 15;
  CALL UCOR(I,P3(I));
END;
CALL 16821;
OUTPUT(VRMM_DAT)=0;
/*
 * Initialize Terminal Emulation Parameters

```

330 1 INIT_VIDEO: PROCEDURE;
 331 2 DO;
 332 3 DECLARE

I I BYTE,
 P1#PTR P1#PTR POINTER,
 P1 P1 BASED P1#PTR BYTE,
 P2#PTR P2#PTR POINTER,
 P2 P2 BASED P2#PTR BYTE,
 P3 P3 BASED P1#PTR(1) BYTE,
 T1 T1 BYTE,
 T2 T2 BYTE;

333 3
 334 3 P1#PTR=@CRTC_REG_50(0);
 335 3 ELSE
 336 3 P1#PTR=@CRTC_REG_60(0);
 337 4 DO I=0 TO 15;
 338 4 CALL UCOR(I,P3(I));
 339 3 END;
 340 3 CALL 16821;
 340 3 OUTPUT(VRMM_DAT)=0;

/*
 * Initialize Terminal Emulation Parameters

```

/*
 * 16821
 *
 * '16821' initializes the both ports of the
 * 6821 used primarily for video applications.
 * Port A is used for video commands, and Port
 * B is used for the Video RAM Mapping Module
 * offset.
 *
 */
16821: PROCEDURE;
DO;
  /* Initialize CA2 for input, and access peripheral register A */
  OUTPUT(VID_CDC)=(P6821_ORA OR P6821_C2_1 OR P6821_C2_0);
  /* Clear peripheral reg. to reset condition, all signals active low */
  OUTPUT(VID_CMD)=0FFH;
  /* Initialize Data Direction Register CA2 for output, and */
  /* Data Direction Register A access */
  OUTPUT(VID_CDC)=(P6821_C2_2 OR P6821_C2_1 OR P6821_C2_0);
  /* Set all bits for output */
  OUTPUT(VID_CDD)=0FFH;
  /* Set access to peripheral register A */
  OUTPUT(VID_CDC)=(P6821_C2_2 OR P6821_C2_1 OR P6821_C2_0 OR P6821_ORA);
  /* Set data direction for all lines as output */
  OUTPUT(VRMM_MPC)=0;
  OUTPUT(VRMM_MPD)=0FFH;
  /* Set access to peripheral register B and zero adder */
  OUTPUT(VRMM_MPC)=(P6821_ORA OR P6821_C2_2 OR P6821_C2_1 OR P6821_C2_0);
  OUTPUT(VRMM_DAT)=0;
END;
383: END;
384: END;

```

\$EJECT

```

402 1 TTY_INTR: PUBLIC;
403 2 DECLARE TEMP BYTE,
404 2 DO; TFLAGS WORD;
405 3 TFLAGS = FLAGS; /* Save Current Processor Flags */
406 3 DISABLE; /* Disable ALL Interrupts */

/*
/* Update the CRT-C Display Start Address
*/
IF CRTC_DISPLAY.UPDATE THEN
DO;
OUTPUT(CRTC_RSEL)=CR_DSARH;
OUTPUT(CRTC_RVAL)=HIGH(CRTC_DISPLAY.START);
OUTPUT(CRTC_RSEL)=CR_DSAL;
OUTPUT(CRTC_RVAL)=LOW(CRTC_DISPLAY.START);
CRTC_DISPLAY.UPDATE=FALSE;
END;

/*
/* Update the CRT-C Cursor Start Address
*/
IF CRTC_CURSOR.UPDATE THEN
DO;
OUTPUT(CRTC_RSEL)=CR_CSARH;
OUTPUT(CRTC_RVAL)=HIGH(CRTC_CURSOR.START);
OUTPUT(CRTC_RSEL)=CR_CSAL;
OUTPUT(CRTC_RVAL)=LOW(CRTC_CURSOR.START);
CRTC_CURSOR.UPDATE=FALSE;
END;

/*
/* Transmit any required characters
*/
IF XMT.COUNT <> 0 THEN
DO;
XMT.BCOUNT = XMT.BCOUNT + INT(XMT.BURST);
DO WHILE XMT.BCOUNT > 0 AND XMT.COUNT <> 0 ;
CALL XCA;
XMT.COUNT = XMT.COUNT - 1;
XMT.COL = XMT.COL + 1;
IF XMT.COL = HI9_PAR.CPL THEN
DO;
XMT.COL = 0;
XMT.ROW = XMT.ROW + 1;
END;
END;
IF XMT.COUNT = 0 THEN
CALL S%XMT(CR);
END;
423 3
424 3
425 4
426 4
427 5
428 5
429 5
430 5
431 5
432 6
433 6
434 6
435 5
436 4
437 4
438 4

```



```
443 1 TTY_POLL; PROCEDURE BOOL PUBLIC;  
444 2 RETURN(CRTC_CURSOR_UPDATE OR CRTC_DISPLAY_UPDATED);  
445 2 END;
```

```
446 1 D_TTY_RES; PROCEDURE WORD; PUBLIC;  
447 2 DECLARE TFLAGS;  
448 2 DO;
```

```
/* Disable VSYNC, Keyboard, and Light-Pen Interrupts  
*/  
449 3 TFLAGS = FLAGS;  
450 3 DISABLE;  
451 3 OUTPUT(M_ITC_1)=INPUT(M_ITC_1) OR OCW1_M6;  
452 3 IF (TFLAGS AND F86_IF) <> 0 THEN  
453 3 ENABLE;
```

```
/* Initialize Dumb Terminal Sub-system  
*/  
454 3 CALL INIT_VIDEO;  
455 3 CALL D_TTY_INIT;
```

```
/* Re-Enable VSYNC, Keyboard, and Light-Pen Interrupts  
*/  
456 3 TFLAGS = FLAGS;  
457 3 DISABLE;  
458 3 OUTPUT(M_ITC_1) = INPUT(M_ITC_1) AND (NOT OCW1_M6);  
459 3 IF (TFLAGS AND F86_IF) <> 0 THEN  
460 3 ENABLE;  
461 3 END;  
462 2 END;
```

* EJECT

```
499 1 D_KBD; PROCEDURE(C) PUBLIC;  
500 2 DECLARE C BYTE;  
501 3 DO;  
502 3 CALL D$XMT(C(KMAP(C)));  
503 3 END;  
504 2 END;
```

\$EJECT

```
534 3 IF (CA AND (XCOLOR_FORE OR XCOLOR_BACK)) <> XMT.COLOR THEN
535 3 DO:
536 4 XMT.COLOR = CA AND (XCOLOR_FORE OR XCOLOR_BACK);
537 4 CALL TXMTC(ESC);
538 4 CALL TXMTC('m');
539 4 CALL TXMTC('0'+(CA AND XCOLOR_FORE));
540 4 CALL TXMTC('0'+SHR(CA AND XCOLOR_BACK,3));
541 4 END;
542 3 IF ((DEL-'')<C) AND XMT.GRAPHIC
543 3 THEN
544 3 CALL TXMTC((C-(DEL+''))+'');
545 3 ELSE
546 2 CALL TXMTC(C+'');
546 2 END;
546 2 END;
```

*EJECT

```

565 1 REV_SCROLL: PROCEDURE PUBLIC;
566 2 DECLARE TFLAGS WORD;
567 2 DO;
568 3 CALL MDL(H19_PAR.SLI,H19_PAR.SLI-1); /* Move Status Line
569 3 OUTPUT(VRMM_DAT)=INPUT(VRMM_DAT)-5; /* Retreat Mapping Offset
570 3 CALL EDC(0,0,H19_PAR.CPL); /* Clear top line */
571 3 TFLAGS = FLAGS;
572 3 DISABLE;
573 3 CRTC_DISPLAY.START=CRTC_DISPLAY.START-80; /* Retreat Start Address */
574 3 CRTC_DISPLAY.UPDATE=TRUE;
575 3 IF (TFLAGS AND F86_IF) <> 0 THEN
576 3 ENABLE;
577 3 END;
578 2 END;

579 1 SCROLL: PROCEDURE PUBLIC;
580 2 DECLARE TFLAGS WORD;
581 2 DO;
582 3 IF H19_MODE.STATUS THEN
583 3 CALL MDL(H19_PAR.SLI,H19_PAR.SLI+1); /* Move Status Line */
584 3 CALL EDC(H19_PAR.SLI,0,H19_PAR.CPL); /* Erase the new bottom line */
585 3 OUTPUT(VRMM_DAT)=INPUT(VRMM_DAT)+5; /* Advance the mapping offset */
586 3 TFLAGS = FLAGS;
587 3 DISABLE;
588 3 /* Advance Display Start Address */
589 3 CRTC_DISPLAY.START=CRTC_DISPLAY.START+80;
590 3 CRTC_DISPLAY.UPDATE=TRUE;
591 3 IF (TFLAGS AND F86_IF) <> 0 THEN
592 3 ENABLE;
593 3 END;
593 2 END;

```

*EJECT

```

631 1 1 SCPI; PROCEDURE(mask,valu) PUBLIC;
632 2 2 DECLARE mask BYTE,
valu BYTE;
633 2 2 DO;
634 3 3 HI?_MODE_CURSOR = (HI?_MODE_CURSOR AND mask) OR valu;
635 3 3 CALL SCPI;
636 3 3 END;
637 2 2 END;

638 1 1 XMTSC; PROCEDURE(row,col,count) PUBLIC;
639 2 2 DECLARE row BYTE,
col BYTE,
count WORD;
640 2 2 DO;
641 3 3 XMT_ROW = row;
642 3 3 XMT_COL = col;
643 3 3 XMT_BCOUNT = 0;
644 3 3 XMT_GRAPHIC,
XMT_REVERSE = FALSE;
645 3 3 XMT_COLOR = BLACK*8 + WHITE;
646 3 3 XMT_COUNT = count;
647 3 3 END;
648 2 2 END;
  
```

*EJECT

```
657 2 DECLARE TFLAGS WORD;  
658 2 DO;  
659 3 TFLAGS = FLAGS; /* Save Current Processor Flags */  
660 3 DISABLE; /* Disable all interrupts */  
661 3 OUTPUT(CRTC_RSEL)=reg; /* Select specified register */  
662 3 OUTPUT(CRTC_RVAL)=val; /* Set new value */  
663 3 IF (TFLAGS AND F96_1F) <> 0 THEN  
664 3 ENABLE;  
665 3 END;  
666 2 END;  
  
667 1 END;
```

```

0060 A20F00      MOV      ; STATEMENT # 341
                                H19_MODE+0FH,AL
                                ; STATEMENT # 342
0063 B107      MOV      CL,7H
0065 51        PUSH     CX ; 1
0066 50        PUSH     AX ; 2
0067 E80705    CALL     SCA      ; STATEMENT # 343
006A F1E00000  CALL     DCI      ; STATEMENT # 344
006E E80000    CALL     P_HQM    ; STATEMENT # 345
0071 E80000    CALL     P_EPAO   ; STATEMENT # 346
0074 B0F0      MOV      AL,0FH
0076 50        PUSH     AX ; 1
0077 B0B0      MOV      AL,80H
0079 50        PUSH     AX ; 2
007A E8A505    CALL     DVC      ; STATEMENT # 347
007D B900C0    MOV      CX,0C00H
0080 B80000    MOV      AX,0H
0083 A30000    MOV      P1PTR,AX
0086 870E0200  MOV      P1PTR+2H,CX ; STATEMENT # 348
008A 87C3      MOV      BX,AX
008C 8EC1      MOV      ES,CX
008E 268A17   MOV      DL,ES:P1[BX]
0091 8B161500  MOV      TI,DL
                                ; STATEMENT # 349
0095 06        ES      ; 1
0096 50        PUSH     AX ; 2
0097 E80000    CALL     INCB
                                ; STATEMENT # 350
009A C4E00000  LES      BX,P1PTR
009E 268A07   MOV      AL,ES:P1[BX]
00A1 3A061500  CMP      AL,TI
00A5 7507      JNZ     @5
                                ; STATEMENT # 351
00A7 C60603001 MOV      H19_PAR+3H,1H
                                ; STATEMENT # 352
00AC EB0F      JMP     @6
                                ; STATEMENT # 353
00AE C60603003 MOV      H19_PAR+3H,3H
                                ; STATEMENT # 354
00B3 A01500      MOV      AL,TI
00B6 C4E00000  LES      BX,P1PTR
00BA 268B07   MOV      ES:P1[BX],AL
                                ; STATEMENT # 356
@6:
00BD B900E0      MOV      CX,0E00H
00C0 B80000    MOV      AX,0H
00C3 A30000    MOV      P1PTR,AX
00C6 870E0200  MOV      P1PTR+2H,CX
                                ; STATEMENT # 357

```

```

0138 E9E704 CALL OVC ; STATEMENT # 371
0139 50 POP BP
013C C3 RET
INIT_VIDED ENDP
; STATEMENT # 372
16921 PROC NEAR
013D 55 PUSH BP
013E 9BEC MOV BP,SP
; STATEMENT # 374
0140 B01C MOV AL,1CH
0142 E6D9 OUT 0DD9H
; STATEMENT # 375
0144 B0FF MOV AL,0FFH
0146 E6D8 OUT 0D8H
; STATEMENT # 376
0148 B038 MOV AL,38H
014A E6D9 OUT 0D9H
; STATEMENT # 377
014C B0FF MOV AL,0FFH
014E E6D8 OUT 0D8H
; STATEMENT # 378
0150 B03C MOV AL,3CH
0152 E6D9 OUT 0D9H
; STATEMENT # 379
0154 B000 MOV AL,0H
0156 E6D8 OUT 0D8H
; STATEMENT # 380
0158 B0FF MOV AL,0FFH
015A E6DA OUT 0DAH
; STATEMENT # 381
015C B03C MOV AL,3CH
015E E6D8 OUT 0D8H
; STATEMENT # 382
0160 B000 MOV AL,0H
0162 E6DA OUT 0DAH
; STATEMENT # 384
0164 5D POP BP
0165 C3 RET
16921 ENDP
; STATEMENT # 385
0_TTY_INIT PROC NEAR
0166 55 PUSH BP
0167 9BEC MOV BP,SP
; STATEMENT # 388
0169 B30000 MOV AX,0H
016C A30000 CRTD_DISPLAY,AX
016F A30C00 H19_MODE+0CH,AX
0172 A30300 XMT+3H,AX
; STATEMENT # 389
0175 A00500 MOV AL,H19_PAR+5H
0178 2C03 SUB AL,3H
017A 0C40 OR AL,40H
017C A20600 H19_MODE+6H,AL
; STATEMENT # 390
017F E000 MOV AL,0H
    
```



```

023E E80000 CALL FLAGS ; STATEMENT # 456
0291 A30A00 MOV TFLAGS,AX ; STATEMENT # 457
0294 FA CLI ; STATEMENT # 458
0295 E4F3 IN 0F3H ; STATEMENT # 459
0297 24BF AND AL,0BFH
0299 E6F3 OUT 0F3H
029B F706A00002 TEST TFLAGS,200H ; STATEMENT # 460
02A1 7401 JZ @21 ; STATEMENT # 461
02A3 FB STI ; STATEMENT # 462
02A4 5D POP BP ; STATEMENT # 463
02A5 C3 RET ; STATEMENT # 464
02A6 55 PROC NEAR
02A7 8BEC PUSH BP
MOV BP,SP
02A9 8A4604 MOV AL,EBP,C
02AC 3C20 CMP AL,20H ; STATEMENT # 465
02AE 7204 JB @23
02B0 3C7F CMP AL,7FH
02B2 7556 JNZ @22
02B4 E007 MOV AL,7H ; STATEMENT # 466
02B6 384604 CMP EBP,C,AL
02B9 7506 JNZ @24
02BB 50 PUSH AX ; STATEMENT # 467
02BC E80000 CALL WRTIK ; 1
; STATEMENT # 470
02BE EB47 JMP @18 ; STATEMENT # 471
02C1 807E0408 CMP EBP,C,SH ; STATEMENT # 472
02C5 750D JNZ @26
02C7 823E000000 CMP HORZ_CHAR,0H
02CC 7406 JZ @26 ; STATEMENT # 473
02CE FE0E0000 DEC HORZ_CHAR ; STATEMENT # 474
02D2 EBEB JMP @18 ; STATEMENT # 475
02D4 807E0409 CMP EBP,C,7H ; STATEMENT # 476
02D8 7505 JNZ @26
02DA E83301 CALL P_HT ; STATEMENT # 477
02DD EBEO JMP @18 ; STATEMENT # 478
  
```

```

034A A00000 MOV AL,H19_PAR
034D HEC8 DEC AL
034F 38060000 CMP HORZ_CHAR,AL
0353 7306 JNB @37
; STATEMENT # 492
0355 HE060000 INC HORZ_CHAR
; STATEMENT # 493
0359 EB0A JMP @35
; STATEMENT # 493
035B A01100 MOV AL,H19_MODE+1H
035E D0D8 RCR AL,1
0360 7303 JNB @35
; STATEMENT # 494
0362 E80000 CALL ORLF
; STATEMENT # 495
0365 E8C501 CALL CURSOR
; STATEMENT # 498
0368 50 POP BP
0369 C20200 RET ZH
; STATEMENT # 499
; STATEMENT # 499
036C 55 PROC NEAR
036D 88EC PUSH BP
MOV BP,SP
; STATEMENT # 502
036F 3A5E04 MOV BL,EBP+4
0372 B700 MOV BH,0H
0374 FFB70000 PUSH KMAPLTX+1
0378 E80000 CALL DXMTC
; STATEMENT # 504
037B 5D POP BP
037C C20200 RET ZH
; STATEMENT # 505
; STATEMENT # 505
037F 55 PROC NEAR
0380 88EC PUSH BP
MOV BP,SP
; STATEMENT # 507
; STATEMENT # 507
034F 55 PROC NEAR
034F 88EC PUSH BP
MOV BP,SP
; STATEMENT # 510
0352 FF7604 PUSH CBJ+C
; 1
0355 E80000 CALL SXMTC
; STATEMENT # 511
0358 FF0E0100 DEC XMT+1H
; STATEMENT # 513
035C 5D POP BP
035D C20200 RET ZH
; STATEMENT # 513
; STATEMENT # 513
0382 B80C00 MOV AX,OFFSET(XC)
; STATEMENT # 515
0385 1E US ; 1
0386 50 PUSH AX
; 2
0387 FFB60600 PUSH XMT+6H
; 3
  
```

```

03E0 EB02      JMP      @27      ; STATEMENT # 532
@44:
03E1 B071      MOV      AL,71H
03E2 E85A00    PUSH     AX      ; 1
@45:
CALL     TXMTC
; STATEMENT # 534

@43:
03F5 A01A00    MOV      AL,CA
03F6 B13F      MOV      CL,3FH
03F7 22C1      AND     AL,CL
03F8 3A060700  CMP     AL,XMT+7H
0400 7429      JZ      @46
; STATEMENT # 536
0402 A20700    MOV      XMT+7H,AL
; STATEMENT # 537
0405 B01E      MOV      AL,1BH
0407 50       PUSH     AX      ; 1
0408 E84400    CALL     TXMTC
; STATEMENT # 538
040B B06D      MOV      AL,6DH
040D 50       PUSH     AX      ; 1
040E E33E00    CALL     TXMTC
; STATEMENT # 539
0411 A01A00    MOV      AL,CA
0414 2407      AND     AL,7H
0416 0430      ADD     AL,30H
0418 50       PUSH     AX      ; 1
0419 E83300    CALL     TXMTC
; STATEMENT # 540
041C A01A00    MOV      AL,CA
041F 2438      AND     AL,38H
0421 B103      MOV      CL,3H
0423 D2E8      SHR     AL,CL
0425 0430      ADD     AL,30H
0427 50       PUSH     AX      ; 1
0428 E82400    CALL     TXMTC
; STATEMENT # 542

@46:
042B B15F      MOV      CL,5FH
042D A01700    MOV      AL,C
0430 3AC8      CMP     CL,AL
0432 7310      JNB     @47
0434 A00800    MOV      AL,XMT+8H
0437 D0D8      RCR     AL,1
0439 7309      JNB     @47
; STATEMENT # 543
043B A01700    MOV      AL,C
043E 2C60      SUB     AL,60H
0440 045E      ADD     AL,5EH
0442 EB05      JMP     @29
; STATEMENT # 544
0444 A01700    MOV      AL,C
@47:

```

```

04A6 7503 JNZ @53 ; STATEMENT # 562
04A8 E83D00 CALL SCROLL ; STATEMENT # 562
      @53:
04AB 5D POP BP ; STATEMENT # 564
04AC C3 RET ; STATEMENT # 564
      P_LF ENDP
      REV_SCROLL
04AD 55 PROC NEAR ; STATEMENT # 565
04AE 8BEC PUSH BP
      MOV BP,SP
04B0 A00400 MOV AL,H1?_PAR+4H ; STATEMENT # 568
04E3 50 PUSH AX ; 1
04E4 FEC8 DEC AL ; 1
04E6 B400 MOV AH,0H
04E8 50 PUSH AX ; 2
04B9 F1E000 CALL MDL ; STATEMENT # 569
04BD E4DA IN 0DAH ; STATEMENT # 569
04BF 2C05 SUB AL,5H
04C1 E6DA OUT 0DAH
04C3 B000 MOV AL,0H ; STATEMENT # 570
04C5 50 PUSH AX ; 1
04C6 50 PUSH AX ; 2
04C7 FF360000 MOV H1?_PAR ; 3
04CB F1E000 CALL EDC ; STATEMENT # 571
04CF E80000 CALL FLAGS ; STATEMENT # 571
04D2 A30E00 MOV TFLAGS,AX
04D5 FA CLI ; STATEMENT # 572
04D6 832E000050 SUB CRIC_DISPLAY,50H ; STATEMENT # 573
04DB C6060200FF MOV CRIC_DISPLAY+2H,0FFH ; STATEMENT # 574
04E0 A70002 TEST AX,200H ; STATEMENT # 575
04E3 7401 JZ @55 ; STATEMENT # 576
04E5 FB STI ; STATEMENT # 576
      @55:
04E6 5D POP BP ; STATEMENT # 578
04E7 C3 RET ; STATEMENT # 578
      REV_SCROLL ENDP
04E8 55 PROC NEAR ; STATEMENT # 579
04E9 8BEC PUSH BP
      MOV BP,SP
04EB A00F00 MOV AL,H1?_MODE+0FFH ; STATEMENT # 582
04EE D0D8 RCR AL,1
04F0 730D JNB @56 ; STATEMENT # 582
    
```

```

054E A30000 MOV CRTD_CURSOR,AX ; STATEMENT # 600
0551 D6060200FF MOV CRTD_CURSOR+2H,0FFH ; STATEMENT # 601
0556 F70612000002 TEST TFLAGS,200H ; STATEMENT # 602
055C 7401 JZ 058 ; STATEMENT # 602
055E FB STI ; STATEMENT # 602
058:
055F 5D POP BP ; STATEMENT # 604
0560 C3 RET ; STATEMENT # 604
CURSOR ENDP
ENABLE_LINES ; STATEMENT # 605
0561 55 BP PROC NEAR
0562 8BEC PUSH BP
MOV BP,SP ; STATEMENT # 608
0564 B006 MOV AL,6H
0566 50 AX ; 1
0567 FF7604 PUSH [BP],LINE$ 2
056A E80100 CALL UCCR
; STATEMENT # 610
056D 5D POP BP
056E C20200 RET 2H
ENABLE_LINES ENDP ; STATEMENT # 611
SDA
0571 55 PROC NEAR
0572 8BEC PUSH BP
MOV BP,SP ; STATEMENT # 614
0574 844606 MOV AL,[BP],FORE ; STATEMENT # 615
0577 A20000 MOV COLOR,AL
; STATEMENT # 615
057A 844604 MOV AL,[BP],BACK
057D A20100 MOV COLOR+1H,AL
; STATEMENT # 616
0580 B103 MOV CL,3H
0582 D2E0 SHL AL,CL
0584 844E06 MOV CL,[BP],FORE
0587 0AC1 OR AL,CL
0589 A20200 MOV COLOR+2H,AL
; STATEMENT # 617
058C E207 MOV DL,7H
058E 3ACC2 CMP AL,DL
0590 B0FF MOV AL,0FFH
0592 7401 JZ $+3H
0594 40 INC AX
0595 20060500 H19_MODE+5H,AL
; STATEMENT # 618
0599 844604 MOV AL,[BP],BACK
059C 0AC1 OR AL,CL
059E 32C2 XOR AL,DL
05A0 A20300 MOV COLOR+3H,AL
; STATEMENT # 619
05A3 844604 MOV AL,[BP],BACK

```

```

05F8 55          XMTSC      PROC NEAR
05F9 8BEC        MOV      BP,SP          ; STATEMENT # 641
05FB 8A4603      MOV      AL,[BP],ROM
05FE A20600      MOV      XMT+6H,AL
0601 8A4606      MOV      AL,[BP],COL    ; STATEMENT # 642
0604 A20500      MOV      XMT+5H,AL
0607 B30000      MOV      AX,0H          ; STATEMENT # 643
060A A30100      MOV      XMT+1H,AX
060D A20700      MOV      XMT+7H,AL
0610 A20E00      MOV      XMT+8H,AL
0613 C606070007  MOV      XMT+7H,7H      ; STATEMENT # 645
0618 8B4604      MOV      AX,[BP],COUNT ; STATEMENT # 646
061B A30300      MOV      XMT+3H,AX
061E 5D          POP      BP             ; STATEMENT # 648
061F C20600      RET      6H
XMTSC      ENDP
; STATEMENT # 649

0622 55          PROC NEAR
0623 8BEC        PUSH     BP
0625 E4D8        MOV      0D8H
0627 8A4E06      MOV      CL,[BP],MASK ; STATEMENT # 652
062A F6D1        MOV      NOT CL
062C 22C1        AND     NOT CL
062E 8A4E04      MOV      CL,[BP],VALLU
0631 F6D1        MOV      NOT CL
0633 224E06      AND     CL,[BP],MASK
0636 0AC1        OR      AL,CL
0638 E6D8        OUT     0D8H
063A 5D          POP      BP             ; STATEMENT # 654
063B C20400      RET     4H
UVC       ENDP
; STATEMENT # 655

063E 55          PROC NEAR
063F 8BEC        PUSH     BP
0641 51          MOV      BP,SP
0642 E80000      CALL    FLAGS          ; STATEMENT # 659
0645 8746FE      MOV     [BP],TFLAGS,AX ; STATEMENT # 660
0648 FA          CLI
0649 8A4606      MOV     AL,[BP],REG    ; STATEMENT # 661
064C E6D8        OUT     0D8H          ; STATEMENT # 662
    
```

DEFN ADDR SIZE NAME, ATTRIBUTES, AND REFERENCES

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
612	00004H	1	BACK
34	00000H	1	BEL
161	00000H	1	BILL
67			BLACK
68			BLUE
4			BOOL
141	00000H	83	BOOT_PAR
	00000H	1	INDEX
	00001H	1	PORT
	00024H	80	STRMG
	00524H	1	UNIT
35		1	BS
162	00000H	1	BSIL
			BUILPTR
54	00000H	4	BYTEPTR
500	00004H	1	C
508	00004H	1	C
464	00004H	1	C
49	00000H	1	C
506	00174H	1	C
57	00000H	1	C
506	001A4H	1	CA
36			CAN
130	00000H	1	CHAR
125	00000H	1	CHAR
177	00000H	1	CHAR
298	00000H	1	CHAR
135			CODE_SEG
180	00000H	1	COL
639	00006H	1	COL
169	00000H	7	COLOR
			FORE
			BACK
			MASK
			CLEAR
			PAINTED
			FONT
			COMP_FONT
639	00004H	2	COUNT
37			CR
122	00000H	3	CRLF
170	00000H	3	CRTC_CURSOR
			START
			UPDATE
171	00000H	3	CRTC_DISPLAY
			START
			UPDATE
323	00000H	16	CRTC_REG_50
329	0010H	16	CRTC_REG_60
75			CRTC_RSEL
			BYTE IN PROC (SCA) PARAMETER AUTOMATIC
			LITERALLY '007H' 468
			BYTE EXTERNAL(36) 342 645
			LITERALLY '0#0#0B' 165 170 171 173 175 292 293 312
			LITERALLY '0#0#1B' 443
			LITERALLY 'BYTE' 470
			STRUCTURE EXTERNAL(16)
			BYTE
			BYTE
			BYTE ARRAY(80)
			LITERALLY '008H' 470
			BYTE EXTERNAL(37)
			BUILTIN
			347 356 357
			POINTER IN PROC (INCB) PARAMETER
			472 474 480 490
			BYTE IN PROC (DXMT) PARAMETER 49
			BYTE IN PROC (TXMT) 516* 518 542 543 544
			BYTE IN PROC (SXMT) PARAMETER 57
			BYTE IN PROC (TXMT) 517* 526 534 536 539 540
			LITERALLY '018H'
			PARAMETER 130
			BYTE IN PROC (WRIT) PARAMETER 125
			BYTE IN PROC (MCU) PARAMETER 177
			BYTE IN PROC (S_CRT) PARAMETER 298
			BYTE IN PROC (WRITE) PARAMETER
			LITERALLY '0FE5H'
			PARAMETER 180
			BYTE IN PROC (DCA) PARAMETER
			STRUCTURE EXTERNAL(41) 639 642
			PARAMETER AUTOMATIC
			STRUCTURE EXTERNAL(42)
			614* 616 618 619 620
			615* 616 618 619 621
			616* 617
			618*
			619* 620 621
			620*
			621*
			WORD IN PROC (XMTSC) PARAMETER AUTOMATIC
			LITERALLY '00DH' 437 480
			PROCEDURE EXTERNAL(8) STACK=0000H 494
			STRUCTURE EXTERNAL(42)
			418 420 599*
			WORD 397* 415 421* 444 600*
			STRUCTURE EXTERNAL(43)
			WORD 338* 410 412 573* 573 588* 588 599
			397* 407 413* 444 574* 589*
			BYTE ARRAY(16) DATA 334
			335
			BYTE ARRAY(16) DATA 409 411 417 419 661
			LITERALLY '10_CRTC+0'

140	0000H	2	DS_SIZE	WORD EXTERNAL(15)					
48	0000H		DXMTC	PROCEDURE EXTERNAL(0) STACK=0000H				502	
463	02A6H	198	D_CRT	PROCEDURE PUBLIC STACK=0010H					
479	036CH	19	D_KBD	PROCEDURE PUBLIC STACK=0009H					
385	0166H	96	D_TTY_INIT	PROCEDURE STACK=0006H	455				
446	026FH	55	D_TTY_RES	PROCEDURE PUBLIC STACK=000CH					
144	0000H	4	D_XMTC	POINTER EXTERNAL(19)					
145	0000H	4	EDC	POINTER EXTERNAL(20)	570	584			
146	0000H	4	EMEC	POINTER EXTERNAL(21)					
605	0561H	16	ENABLE_LINES	PROCEDURE PUBLIC STACK=000AH					
39			EOL	LITERALLY '080H'					
40			EOS	LITERALLY '030H'					
41			ESC	LITERALLY '01BH'	521	529	537		
172	0000H	10	ESCP	STRUCTURE EXTERNAL(44)					
	0000H	1	COMMAND	BYTE					
	0001H	2	FUNCTION	WORD					
	0003H	1	MODE	BYTE					
	0004H	1	OPER_COUNT	BYTE					
	0005H	1	OPER_INDEX	BYTE					
	0006H	4	OPERAND	BYTE ARRAY(4)					
178			F86_AF	LITERALLY '0000*0001*0000B'					
200			F86_CF	LITERALLY '0000*0000*0000B'					
173			F86_DF	LITERALLY '0100*0000*0000B'					
174			F86_IF	LITERALLY '0010*0000*0000B'	439	452	459	575	590
				643					601
192			F86_OF	LITERALLY '1000*0000*0000B'					
179			F86_Pf	LITERALLY '0000*0000*0100B'					
196			F86_Sf	LITERALLY '0000*1000*0000B'					
195			F86_Tf	LITERALLY '0001*0000*0000B'					
197			F86_Zf	LITERALLY '0000*0100*0000B'					
5			FALSE	LITERALLY '000H'	341	390	395	397	413
42			FF	LITERALLY '00CH'					421
51	0000H		FLAGS	PROCEDURE WORD EXTERNAL(1) STACK=0000H	405	449	456	571	
				586	597	659			
147	0000H	4	FONT	POINTER EXTERNAL(22)					
156	0000H	2	FONT_SIZE	WORD EXTERNAL(31)					
612	0006H	1	FORE	BYTE IN PROC (SCA) PARAMETER AUTOMATIC	612	614			
303			GEN_CNT	LITERALLY '10_GENERAL*1'					
304			GEN_DAT	LITERALLY '10_GENERAL'					
305			GEN_DIR	LITERALLY '10_GENERAL'					
71			GREEN	LITERALLY '18*0B'					
173	0000H	18	H19_MODE	STRUCTURE EXTERNAL(45)					
	0000H	1	ALTERNATE	BYTE					
	0001H	1	ANSI	BYTE					
	0002H	1	AUTO_CR	390* 476					
	0003H	1	AUTO_LF	BYTE					
	0004H	1	AUTO_REPEAT	390* 483					
	0005H	1	BWD	BYTE					
	0006H	1	CURSOR	617* 617					
	0007H	1	CURSOR_ON	389* 627	628	634*	634		
	0008H	1	EXPAND	391* 626					
	0009H	1	GRAPHIC	BYTE					
	000AH	1	INSERT	390* 483					
	000BH	1	KEY_EN	BYTE					
	000CH	2	REVERSE	391*					
	000EH	1	SHIFTED	383* 490					

7			IO_DIP	LITERALLY '00FEH'	333							
24			IO_GENERAL	LITERALLY '00E0H'								
7			IO_HIADR	LITERALLY '00FDH'								
18			IO_INT_MS	LITERALLY '00F2H'	451	458						
19			IO_INT_SL	LITERALLY '00F0H'								
17			IO_KEYBRD	LITERALLY '00F4H'								
26			IO_LIFN	LITERALLY '00DEH'								
10			IO_MEM	LITERALLY '00FCH'								
23			IO_PRINTER	LITERALLY '00E2H'								
12			IO_RSRV_1	LITERALLY '00FAH'								
14			IO_RSRV_2	LITERALLY '00FBH'								
15			IO_RSRV_3	LITERALLY '00F7H'								
16			IO_RSRV_4	LITERALLY '00F6H'								
25			IO_RSRV_5	LITERALLY '00DFH'								
29			IO_RSRV_6	LITERALLY '00C0H'								
21			IO_SER_A	LITERALLY '00E8H'								
20			IO_SER_B	LITERALLY '00ECH'								
13			IO_SOUND	LITERALLY '00F9H'								
8			IO_SWAP	LITERALLY '00FEH'								
22			IO_TIMER	LITERALLY '00E4H'								
11			IO_TSTAT	LITERALLY '00F8H'	340	374	375	376	377	378	379	380
28			IO_VIDEO	LITERALLY '00D8H'								
31			IO_ZZ07_0	381 382 569 585 652								
30			IO_ZZ07_1	LITERALLY '00B0H'								
33			IO_ZZ17_0	LITERALLY '00B8H'								
32			IO_ZZ17_1	LITERALLY '00A8H'								
65	0000H	1	IVL	BYTE IN PROC (INIT_ITV) PARAMETER						65		
65	0000H	4	IVP	POINTER IN PROC (INIT_ITV) PARAMETER						65		
261			KC_ARF	LITERALLY '02H'								
262			KC_AND	LITERALLY '01H'								
263			KC_BEP	LITERALLY '07H'	469							
268			KC_CFI	LITERALLY '05H'								
269			KC_CLK	LITERALLY '06H'								
264			KC_DI	LITERALLY '0DH'								
265			KC_EI	LITERALLY '0CH'	396							
270			KC_KBD	LITERALLY '07H'								
271			KC_KBE	LITERALLY '03H'								
266			KC_KCF	LITERALLY '04H'								
267			KC_KCQ	LITERALLY '03H'								
272			KC_MNS	LITERALLY '0BH'								
273			KC_MUD	LITERALLY '0AH'								
274			KC_RES	LITERALLY '00H'	392							
292	0000H	2	KEY	STRUCTURE EXTERNAL(55)								
	0000H	1	CHAR	BYTE								
	0001H	1	AVAIL	BYTE								
256			KEY_CMND	LITERALLY 'ID_KEYBRD+1'	395#							
257			KEY_DATA	LITERALLY 'ID_KEYBRD+0'								
258			KEY_STAT	LITERALLY 'ID_KEYBRD+1'								
157	0000H	256	KMAP	BYTE ARRAY(256) EXTERNAL(32)	502							
259			KS_CXRDY	LITERALLY '001B'								
260			KS_KXRDY	LITERALLY '010B'								
286			KY_BREAK	LITERALLY '0AAH'								
282			KY_DOWN	LITERALLY '0AAH'								
275			KY_ENTER	LITERALLY '08DH'								
277			KY_F0	LITERALLY '096H'								

238	OCM1_M4	LITERALLY '0001\$0000B'				
237	OCM1_M5	LITERALLY '0010\$0000B'				
236	OCM1_M6	LITERALLY '0100\$0000B'	451	458		
235	OCM1_M7	LITERALLY '1000\$0000B'				
246	OCM2_E01	LITERALLY '0010\$0000B'				
250	OCM2_L0	LITERALLY '0000\$0001B'				
249	OCM2_L1	LITERALLY '0000\$0010B'				
248	OCM2_L2	LITERALLY '0000\$0100B'				
247	OCM2_OCM2	LITERALLY '0000\$0000B'				
244	OCM2_ROT	LITERALLY '1000\$0000B'				
245	OCM2_SE01	LITERALLY '0100\$0000B'				
251	OCM3_ESM1	LITERALLY '0100\$0000B'				
253	OCM3_OCM3	LITERALLY '0000\$1000B'				
254	OCM3_POLL	LITERALLY '0000\$0100B'				
252	OCM3_SMM	LITERALLY '0010\$0000B'				
255	OCM3_SRRS	LITERALLY '0000\$0010B'				
	OUTPUT	BULLITT				
		340* 374* 375* 376* 377* 378* 379* 380* 381*				
		382* 409* 410* 411* 412* 417* 418* 419* 420* 451* 458* 569*				
		385* 652* 661* 662*				
449	0622H	PROCEDURE PUBLIC STACK=0006H				
		BYTE BASED(P1PTR) IN PROC (INIT_VIDEO)				
332	0000H	358 360 362 365*	346	368	369	
		350 356* 358 360 362	348	349	350	354*
		BYTE BASED(P2PTR) IN PROC (INIT_VIDEO)				
332	0000H	346*	334*	335*	337	347* 348 349
		346*				
		POINTER IN PROC (INIT_VIDEO)				
332	0000H	357* 359 361 362				
		357				
		BYTE BASED(P1PTR) ARRAY(1) IN PROC (INIT_VIDEO)				
318	P6821_A	LITERALLY '0'				
319	P6821_B	LITERALLY '2'				
327	P6821_C1_0	LITERALLY '001H'				
326	P6821_C1_1	LITERALLY '002H'	374	376	378	381
324	P6821_C2_0	LITERALLY '008H'	374	376	378	381
323	P6821_C2_1	LITERALLY '010H'	376	378	381	
322	P6821_C2_2	LITERALLY '020H'	376	378	381	
320	P6821_IR01	LITERALLY '030H'				
321	P6821_IR02	LITERALLY '040H'				
325	P6821_ORA	LITERALLY '004H'	374	378	381	
300	PRN_CNT	LITERALLY '00PRINTER+1'				
301	PRN_DAT	LITERALLY 'ID_PRINTER'				
302	PRN_DIR	LITERALLY 'ID_PRINTER'				
	PROMPT	POINTER EXTERNAL (25)				
150	0000H	BYTE EXTERNAL (38)				
163	0000H	PROCEDURE EXTERNAL (51) STACK=0000H				
184	0000H	PROCEDURE EXTERNAL (52) STACK=0000H				345
186	0000H	PROCEDURE EXTERNAL (50) STACK=0000H				344
182	0000H	PROCEDURE EXTERNAL (50) STACK=0000H				344
347	0460H	PROCEDURE STACK=0002H	473			398
357	0487H	PROCEDURE STACK=0004H	473	484		
188	0000H	PROCEDURE EXTERNAL (53) STACK=0000H				
151	0000H	POINTER EXTERNAL (26)				
127	0000H	515				
295	0000H	PROCEDURE BYTE EXTERNAL (10) STACK=0000H				394
67	0000H	PROCEDURE BYTE EXTERNAL (57) STACK=0000H				
	REG	LITERALLY '0\$1\$0B'				
656	0006H	BYTE IN PROC (UCCR) PARAMETER AUTOMATIC				656
62	0000H	POINTER IN PROC (XEC) PARAMETER				62
164	0000H	POINTER EXTERNAL (39)				

CROSS-REFERENCE LISTING

59	0000H	VIDEO	PROCEDURE STACK=0000H
79	0000H	VIDEO_INTR_A	PROCEDURE EXTERNAL(4) STACK=0000H
78	0000H	VID_CDC	LITERALLY '10_VIDC0+1'
77	0000H	VID_CDD	LITERALLY '10_VIDC0+0'
80	0000H	VID_CMD	LITERALLY '10_VIDC0+0'
82	0000H	VRM_DAT	LITERALLY '10_VIDC0+2'
81	0000H	VRM_MPC	LITERALLY '10_VIDC0+3'
45	0000H	VRM_MPD	LITERALLY '10_VIDC0+2'
74	0000H	VT	LITERALLY '00BH'
138	0000H	WHITE	LITERALLY '1\$1\$1B'
129	0000H	WIP	BYTE ARRAY(5) EXTERNAL(13)
132	0000H	WRITC	PROCEDURE EXTERNAL(11) STACK=0000H
297	0000H	WRITES	PROCEDURE EXTERNAL(12) STACK=0000H
506	0000H	WRITK	PROCEDURE EXTERNAL(58) STACK=0000H
154	0000H	XC	WORD IN PROC (TXMT)
168	0000H	XCA	POINTER EXTERNAL(29)
167	0000H	XCOLOR_BACK	LITERALLY '00\$111\$000B'
61	0000H	XCOLOR_FORE	LITERALLY '00\$100\$111B'
175	0000H	XEC	PROCEDURE EXTERNAL(5) STACK=0000H
10	0000H	XMT	STRUCTURE EXTERNAL(47)
1	0000H	BURST	BYTE
2	0000H	COUNT	INTEGER
2	0000H	COUNT	WORD
1	0000H	COL	BYTE
1	0000H	COL	BYTE
1	0000H	ROM	BYTE
1	0000H	ROM	BYTE
1	0000H	COLOR	BYTE
1	0000H	GRAPHIC	BYTE
1	0000H	GRAPHIC	BYTE
1	0000H	REVERSE	BYTE
42	05F8H	XMTSC	PROCEDURE PUBLIC STACK=0008H
46		XOPF	LITERALLY '013H'
47		XON	LITERALLY '011H'
166		XREVERSE	LITERALLY '10\$1000\$000B'
73		YELLOW	LITERALLY '1\$1\$0B'

MODULE INFORMATION:

CUBE AREA SIZE = 0661H 1693D
 CONSTANT AREA SIZE = 0000H 0D
 VARIABLE AREA SIZE = 001CH 28D
 MAXIMUM STACK SIZE = 0010H 16D
 1758 LINES READ
 0 PROGRAM WARNINGS
 0 PROGRAM ERRORS

END OF PL/M-86 COMPILATION

Definitions

```

= $SAVE
= $NOLIST
= $NOLIST INCLUDE (':F2:KEYDEF.H')
= $SAVE
= $NOLIST INCLUDE (':F2:KEYLIB.H')
= $SAVE
= $NOLIST INCLUDE (':F2:VIDEO.H')
= $SAVE
= $NOLIST
= $ SET(COMPRESS)
/*
*/
Internal Terminal Emulation Modes
*/
186 1 DC M_NRM LT '0' /* Normal character display */;
187 1 DC M_ESC LT '1' /* Looking for Escape command */;
188 1 DC M_OPR LT '2' /* Accepting Sequence Operands */;
189 1 DC M_ANS LT '3' /* ANSI-Mode Sequence Processing */;

190 1 DECLARE LINE_INDEX BYTE /* Scratch Line Index */
PUBLIC;
191 1 DECLARE SAVED_HORZ_CHAR BYTE /* Saved Horizontal Char. pos. */
PUBLIC;
192 1 DECLARE SAVED_VERT_LINE_BYTE /* Saved Vertical Line pos. */
PUBLIC;

193 1 DECLARE ESC_TABLE_1(*) WORD DATA
.P_EAM, /* Enter ANSI Mode */
.P_EAKM, /* Enter Alternate Key-Pad Mode */
.P_XAKM, /* Exit Alternate Key-Pad Mode */
.P_UIM, /* Un-Implemented */
.P_EICM, /* Enter Insert Character Mode */
.P_CUP, /* Cursor Up */
.P_CDN, /* Cursor Down */
.P_CRT, /* Cursor Right */
.P_CLF, /* Cursor Left */
.P_ERPA, /* Erase Page */
.P_ERGM, /* Enter Graphics Mode */
.P_XGM, /* Exit Graphics Mode */
.P_HOM, /* Home the Cursor */
.P_RLF, /* Reverse Line Feed */
.P_EEOP, /* Erase to End-Of-Page */
.P_EEOL, /* Erase to End-Of-Line */
.P_IL, /* Insert Line */
.P_DL, /* Delete Line */
.P_DC, /* Delete Character */
.P_XICM, /* Exit Insert Character Mode */
.P_ICA, /* Direct Cursor Addressing */

194 1 DECLARE WORD DATA (
/* Direct Cursor Addressing */

```

```

196 1 1 S_TTY_INIT: PROCEDURE PUBLIC;
197 2 DO;
198 3 ESCP.MODE = M_NRM;
199 3 H19.MODE.EXPAND = TRUE;
200 3 H19.MODE.ANSI,
H19.MODE.GRAPHIC,
H19.MODE.ALTERNATE,
H19.MODE.SHIFTED = FALSE;
201 3 CALL P_SCP; /* Initialize Saved Cursor Position */
202 3 END; /*
203 2 END;

204 1 S_CRT: PROCEDURE(char) PUBLIC;
205 2 DECLARE char BYTE;
206 2 DO;
207 3 IF XMT.COUNT = 0 THEN
208 3 DO;
209 4 char = char AND 07FH;
210 4 DO CASE ESCP.MODE :
/*
*/
/* Normal Character
*/
DO;
IF char <> ESC THEN
DO;
IF H19.MODE.GRAPHIC AND ('^'<=char AND char<DEL)
THEN
CALL D_CRT(char^'+DEL+1);
ELSE
CALL D_CRT(char);
END;
ELSE IF H19.MODE.ANSI THEN
ESCP.MODE = M_ANSI;
ELSE
ESCP.MODE = M_ESC;
END;
END;
/*
*/
/* Escape Sequence Command
*/
DO;
ESCP.COMMAND = char ;
ESCP.OPER_COUNT = DOC(char) ;
ESCP.OPER_INDEX = 0 ;
ESCP.MODE = M_OPR ;
IF ESCP.OPER_COUNT = 0 THEN
CALL PES;
END;
/*
*/
Escape Sequence Operands

```

```

/*
 *
 * DCA -- Direct Cursor Addressing
 *
 * DCA performs the direct cursor addressing
 * function. This involves setting the cursor
 * to the specified address. This cursor will
 * NOT be moved to the status line if the status
 * line is not enabled. Any illegal line spec-
 * ification results in the cursor remaining at
 * the current line. Any illegal column spec-
 * ification results in positioning the cursor
 * at the end of the current line.
 *
 * Parameters:
 *
 * row -- Vertical Row [0,h19_par.s11]
 * col -- Horizontal Column [0,h19_par.cp1-1]
 */
JCA:
DECLARE   row      BYTE;
          col      BYTE;
DO;
IF (0 <= row AND row < H19_PAR.SL1) OR
   (row=H19_PAR.SL1 AND H19_MODE.STATUS) THEN
  VERT_LINE = row;
IF (0 <= col AND col < H19_PAR.CP1)
  THEN
  HORZ_CHAR = col;
ELSE
  HORZ_CHAR = H19_PAR.CP1-1;
END;
END;
244 1
245 2
246 2
247 3
248 3
249 3
250 3
251 3
252 3
253 2

```

```

/*
 *
 * DDC -- Determine Operand Count
 *
 * DDC determines the operand count for the specified escape
 * sequence.
 *
 * Parameters:
 *
 * cmd = Escape Sequence or Command to return
 *       the number of parameters for.
 *
 * Exit:
 *
 * Returns the number of parameters (characters) to be

```

```

/*
 * - Process Escape Sequence
 *
 * Description
 *
 * PES processes the current escape sequence. When PES
 * is invoked, it is assumed that all of the required
 * parameters/operands have been input.
 *
 * Entry:
 *
 * ESCP.COMMAND = Escape Sequence Command
 * ESCP.OPERAND(i) = Operands for this escape sequence
 *
 * Exit:
 *
 * Emulation mode, 'ESCP.MODE', reset to normal character mode
 */

```

```

266 1 PES:
267 2 DO;
268 3 PROCEDURE;
269 3 IF ESCP.MODE = M_NRM;
270 3 ' <<=ESCP.COMMAND AND ESCP.COMMAND<='0' THEN
271 3 ESCP.FUNCTION = ESC_TABLE_1(ESCP.COMMAND<');
272 3 'Y<=ESCP.COMMAND AND ESCP.COMMAND<=' ' THEN
273 3 ESCP.FUNCTION = ESC_TABLE_2(ESCP.COMMAND<'Y');
274 3 ELSE IF ESCP.COMMAND = 'b' THEN
275 3 ESCP.FUNCTION = 'P_EDOP';
276 3 ELSE IF 'i<=ESCP.COMMAND AND ESCP.COMMAND<=' THEN
277 3 ESCP.FUNCTION = ESC_TABLE_3(ESCP.COMMAND<'i');
278 3 ELSE IF '#? = ESCP.COMMAND THEN
279 3 ESCP.FUNCTION = 'P_XMTP';
280 3 ELSE
281 3 ESCP.FUNCTION = 'P_UIM';
282 3 CALL ESCP.FUNCTION;
283 3 CALL CURSOR;
283 2 END;

```

REJECT


```
290 1 P_UIM: PROCEDURE PUBLIC;  
291 2 DO;  
292 3 CALL UIES;  
293 3 END;  
294 2 END;
```

*EJECT

346 2 END;

```
$IF EXTENDED_MONITOR  
P_EHSM: PROCEDURE EXTERNAL;  
END;  
P_XHEM: PROCEDURE EXTERNAL;  
END;  
$ENDIF
```

```
347 1 P_DK: PROCEDURE EXTERNAL;  
348 2 END;  
349 1 P_EK: PROCEDURE EXTERNAL;  
350 2 END;  
351 1 P_EKPS: PROCEDURE EXTERNAL;  
352 2 END;  
353 1 P_XKPS: PROCEDURE EXTERNAL;  
354 2 END;  
355 1 P_EVVM: PROCEDURE EXTERNAL;  
356 2 END;  
357 1 P_XRVM: PROCEDURE EXTERNAL;  
358 2 END;  
359 1 P_EMRA: PROCEDURE EXTERNAL;  
360 2 END;  
361 1 P_XWRAP: PROCEDURE EXTERNAL;  
362 2 END;
```

```
$IF EXTENDED_MONITOR  
P_BAUD: PROCEDURE EXTERNAL;  
END;  
$ENDIF
```

```
363 1 P_RES: PROCEDURE EXTERNAL;  
364 2 END;  
365 1 P_RM: PROCEDURE EXTERNAL;  
366 2 END;  
367 1 P_SM: PROCEDURE EXTERNAL;  
368 2 END;  
369 1 P_COLOR: PROCEDURE EXTERNAL;  
370 2 END;  
371 1 P_IDENT: PROCEDURE EXTERNAL;  
372 2 END;  
373 1 P_IVT52: PROCEDURE EXTERNAL;  
374 2 END;  
375 1 P_XMTC: PROCEDURE EXTERNAL;  
376 2 END;  
377 1 P_XMTC: PROCEDURE EXTERNAL;  
378 2 END;  
379 1 P_XMISL: PROCEDURE EXTERNAL;  
380 2 END;  
381 1 P_XMTP: PROCEDURE EXTERNAL;  
382 2 END;
```

```
$ ELSE  
$ EJECT
```

```
/*  
* P_CUP -- Perform Cursor Up  
*  
* P_CUP moves the cursor up one line. This will NOT  
* move the cursor off of the status line. If the  
* cursor is currently on line-0, then no action is  
* taken.  
*/
```

```
P_CUP: PROCEDURE;  
DO;  
IF VERT_LINE < H19_PAR_SLI THEN  
CALL DCA(VERT_LINE-1,HORZ_CHAR);  
END;  
END;
```

```
/*  
* P_CPR -- Cursor Position Report  
*  
* 'P_CPR' reports the cursor position in the format  
* expected by the Direct Cursor Addressing escape  
* sequence.  
*/
```

```
P_CPR: PROCEDURE;  
DO;  
CALL TEC;  
CALL S%XMITC('Y');  
CALL S%XMITC(VERT_LINE+'');  
CALL S%XMITC(HORZ_CHAR+'');  
END;
```

```
/*  
* P_DCA -- Perform Director Cursor Addressing  
*  
* P_DCA sets the cursor to the specified address.  
* this routine invokes the 'DCA' routine to per-  
* form the cursor addressing, and assumes that  
* the operands to the escape sequence are in the  
* operand buffer. The first byte is assumed to  
* be the row, and the second the column. The bytes  
* are as in the H-19/VT-52, which means that the  
* values are offset by the value of the space char...
```

```
P_ERLF:      PROCEDURE;  
DO;  
IF  
    VERT_LINE = H19_PAR.SLI THEN  
    RETURN;  
ELSE IF VERT_LINE <> 0 THEN  
    VERT_LINE = VERT_LINE - 1;  
ELSE  
    CALL REV_SCROLL;  
END;  
END;  
END;
```

```
/*  
* P_SCP -- Save Cursor Position  
*  
* P_SCP saves the current cursor position so that  
* it may be restored with the appropriate escape  
* sequence. It is important to note that the be-  
* haviour to be emulated stipulates that the pos-  
* itions NOT be stacked, therefore, only one set  
* of save locations is used.  
*  
*  
*  
*/  
P_SCP:      PROCEDURE;  
DO;  
    SAVED_HORZ_CHAR = HORZ_CHAR;  
    SAVED_VERT_LINE = VERT_LINE;  
END;  
END;
```

```
#EJECT
```

```
/*  
* P_EROL -- Process Erase to Beginning Of Line  
*  
* P_EROL erases to the beginning of the current line,  
* including the current character.  
*  
*  
*/
```

```
P_EROL:      PROCEDURE;  
DO;  
    CALL EDC(VERT_LINE,0,HORZ_CHAR+1);  
END;  
END;
```

```
ENDS;  
ENDS;  
ENDS;
```

```
/*  
* P_ELIN -- Process Erase Line  
*  
* P_ELIN erases the entire current line.  
*/
```

```
P_ELIN: PROCEDURE PUBLIC;  
DO;  
CALL EDL(VERT_LINE);  
ENDS;  
ENDS;
```

```
/*  
* P_EPAG -- Erase Page  
*  
* P_EPAG erases the entire page.  
*/
```

```
P_EPAG: PROCEDURE PUBLIC;  
DO;  
IF VERT_LINE = H19_PAR.SLI  
THEN  
CALL P_ELIN;  
ELSE  
DO;  
DO LINE_INDEX=0 TO H19_PAR.SLI-1;  
CALL EDL(LINE_INDEX);  
ENDS;  
CALL P_HOM;  
IF NOT H19_MODE.STATUS THEN  
H19_MODE.BWD = (COLOR.MASK = 7);  
ENDS;  
ENDS;  
ENDS;
```

```
/*  
* P_DC -- Perform Delete Character  
*  
* P_DC deletes the character at the current character  
* position. This is done by moving the rest of the  
* line to the current character position, and deleting
```

```

THEN
DO;
DO LINE_INDEX = VERT_LINE+1 TO H19_PAR.SLI-1 ;
CALL MDL(LINE_INDEX,LINE_INDEX-1);
END;
CALL EDL(H19_PAR.SLI-1);
END;
ELSE
CALL P_ELIN;
CALL DCA(VERT_LINE,0);
END;
ENDIF

```

```

/*
*
* P_IL -- Perform Insert Line
*
* P_IL performs the insert line function. A blank line
* line is inserted at the current cursor position after
* the remaining lines have been moved down.
*/

```

```

$ IF 0=1
P_IL: PROCEDURE;
DO;
IF VERT_LINE=H19_PAR.SLI THEN
DO;
CALL P_ELIN;
RETURN;
END;

```

```

IF VERT_LINE < H19_PAR.SLI/2
THEN
DO;
CALL REV_SCROLL;
DO LINE_INDEX=1 TO VERT_LINE ;
CALL MDL(LINE_INDEX,LINE_INDEX-1);
END;
ELSE

```

```

DO;
LINE_INDEX=H19_PAR.SLI-1;
DO WHILE LINE_INDEX <> VERT_LINE ;
CALL MDL(LINE_INDEX-1,LINE_INDEX);
LINE_INDEX=LINE_INDEX-1;
END;

```

```

END;
CALL P_ELIN;
END;
ELSE
P_IL: PROCEDURE;

```

```
/*  
 * P_EAM -- Enter ANSI Mode  
 *  
 * 'P_EAM' enters ANSI Mode, which means that the  
 * extended mode flag is set for external user  
 * processing.  
 *  
 */
```

```
P_EAM: PROCEDURE;  
DO;  
  H19_MODE.ANSI = TRUE;  
END;  
END;
```

```
/*  
 * P_EICM -- Process Enter Insert Character Mode  
 *  
 * P_EICM processes the insert character mode escape  
 * sequence. It merely sets the global boolean to  
 * true.  
 *  
 */
```

```
P_EICM: PROCEDURE;  
DO;  
  H19_MODE.INSERT = TRUE;  
END;  
END;
```

```
/*  
 * P_XICM -- Process Exit Insert Character Mode  
 *  
 * P_XICM processes the Exit insert character mode  
 * escape sequence. It merely sets the global bool-  
 * ean to FALSE.  
 *  
 */
```

```
P_XICM: PROCEDURE;  
DO;  
  H19_MODE.INSERT = FALSE;  
END;  
END;
```

```
*
*      P_XHSM - Exit Hold-Screen Mode
*
*      'P_XHSM' exits the Hold-Screen Mode by setting the
*      boolean H19_MODE.HOLD to FALSE.
*
*/

$      IF EXTENDED_MONITOR
P_XHSM:
  DO;
  END;
$      ENDIF

/*
*      P_DK - Disable Keyboard
*
*      'P_DK' disables the key-board.
*
*/

P_DK:
  DO;
  CALL WRITK(KC_KBD);
  H19_MODE.KEY_EN = FALSE;
  END;

/*
*      P_EK - Enable Key-Board
*
*      'P_EK' enables the key-board.
*
*/

P_EK:
  DO;
  CALL WRITK(KC_KBE);
  H19_MODE.KEY_EN = TRUE;
  END;

/*
*      P_EKPS - Enter Key-Pad Shifted Mode
*
*      'P_EKPS' enters the key-pad shifted mode by
*      merely setting the boolean 'H19_MODE.SHIFTED'
*/
```



```
*  
# zeroes for invoking the display font character  
# procedure.  
*/
```

```
P_XRVM: PROCEDURE;  
DO;  
  H19_MODE.REVERSE = 00000H;  
END;  
*/
```

```
/*  
* P_EWRAP -- Enter Wrap Mode  
*  
* 'P_EWRAP' enters the H19 character wrap mode  
* by setting the global H19_MODE.WRAP to TRUE.  
*  
*/
```

```
P_EWRAP: PROCEDURE;  
DO;  
  H19_MODE.WRAP = TRUE;  
END;  
*/
```

```
/*  
* P_XWRAP -- Exit Wrap Mode  
*  
* 'P_XWRAP' exits the H19 character wrap mode  
* by setting the global H19_MODE.WRAP to FALSE.  
*  
*/
```

```
P_XWRAP: PROCEDURE;  
DO;  
  H19_MODE.WRAP = FALSE;  
END;  
*/
```

#EJECT

```
/*  
* P_BAUD -- Process Baud Rate  
*  
*/
```

```

*
* > -- Disable Function-Key Expansion
* @ -- Disable Key-Board Up/Down Mode
*/

```

P_ARM: PROCEDURE;

```

DO;
IF '1' <= ESCP.OPERAND(0) AND ESCP.OPERAND(0) <= 'e' THEN
DO CASE ESCP.OPERAND(0) = '1' ;

```

```

/* 1 */
DO;
H19_MODE.STATUS = FALSE;
CALL EDL(H19_PAR.SLI);
CALL ENABLE_LINES(H19_PAR.SLI);
END;

```

```

/* 2 */
CALL WRITR(KC_KCO);
/* 3 */
CALL P_UIM;
/* 4 */
CALL SCP1(NOT CSR_CSD,
H19_PAR.SPC-3);

```

```

/* 5 */
DO;
H19_MODE.CURSOR_ON = TRUE;
CALL SCP;
END;

```

```

/* 6 */
CALL P_XRPS;
/* 7 */
CALL P_XAKM;

```

```

/* 8 */
H19_MODE.AUTO_LF = FALSE;
/* 9 */
H19_MODE.AUTO_CR = FALSE;

```

```

/* : */
CALL P_UIM;
/* ; */
CALL SCP1(CSR_CSD,
CSR_BLINK16);

```

```

/* < */
CALL WRITR(KC_ARO);
/* = */
CALL P_UIM;

```

```

/* > */
CALL P_UIM;
/* ? */
H19_MODE.EXPAND = FALSE;

```

```

/* @ */
DO;
H19_MODE.UPDN = FALSE;
CALL WRITR(KC_MNS);
END;

```

END;

```
CALL P_EKPS; /* 6 */
CALL P_EAKM; /* 7 */
H19_MODE.AUTO_LF = TRUE; /* 8 */
H19_MODE.AUTO_CR = TRUE; /* 9 */
CALL P_UIM; /* : */
CALL SCPI(NOT CSR_BLINK,0); /* ; */
CALL WRITK(KC_ARF); /* < */
CALL P_UIM; /* = */
CALL P_UIM; /* > */
H19_MODE.EXPAND = TRUE; /* ? */
DO; /* @ */
H19_MODE.UPDN = TRUE;
CALL WRITK(KC_MUD);
END;
END;
END;
```

#EJECT

```
P_COLOR: PROCEDURE;
DO;
IF ('0' <= ESCP_OPERAND(0) AND ESCP_OPERAND(0) <='7') AND
('0' <= ESCP_OPERAND(1) AND ESCP_OPERAND(1) <='7')
CALL SCA(ESCP_OPERAND(0)..'0',ESCP_OPERAND(1)..'0');
END;
END;
```

```
P_IDENT: PROCEDURE;
DO;
IF ESCP_OPERAND(0) = '0' THEN
DO;
CALL TEC;
CALL S$XMT('1');
CALL S$XMT('E');
END;
END;
```

```
CALL XMTSC(H19_PAR.SL1,0,H19_PAR.CPL);  
ELSE  
CALL S*XMTC(CR);  
END;  
END;
```

```
P_XMTP: PROCEDURE;  
DO;  
CALL XMTSC(0,0,H19_PAR.CPL*(H19_PAR.LPS-1));  
END;  
END;
```

```
* ENDIF
```

```
383 1 END;
```

```

00BD B15E MOV CL,SEH
00BF 8A4604 MOV AL,[BP].CHAR
00C2 3AC8 CMP CL,AL
00C4 770F JA @6
00C6 B27F MOV DL,ZFH
00C8 3ACC CMP AL,DL
00CA 7309 JNB @6
; STATEMENT # 215
00CC 2AC1 SUB AL,CL
00CE 02C2 ADD AL,DL
00D0 FED0 INC AL
00D2 50 AX ; 1
00D3 EB03 JMP @2
; STATEMENT # 216
00D5 FF7604 @6: PUSH [BP].CHAR; 1
00D8 E90000 @2: CALL D_CRT
; STATEMENT # 218
00DB EB57 @8: JMP @1
00DD A00100 @5: MOV AL,H19_MODE+1H
00E0 D0D8 RCR AL,1
00E2 7307 JNB @7
; STATEMENT # 219
00E4 C606030003 MOV ESCP+3H,3H
; STATEMENT # 220
00E7 EB49 @9: JMP @1
00EB C606030001 MOV ESCP+3H,1H
; STATEMENT # 222
00F0 EB42 @11: JMP @1
; STATEMENT # 223
00F2 8A4604 MOV AL,[BP].CHAR
00F5 A20000 MOV ESCP,AL
; STATEMENT # 224
00F8 50 PUSH AX
00F9 EB7F00 CALL DDG
00FC A20400 MOV ESCP+4H,AL
; STATEMENT # 225
00FF C606050000 MOV ESCP+5H,0H
; STATEMENT # 226
0104 C606030002 MOV ESCP+3H,2H
; STATEMENT # 227
0109 08C0 OR AL,AL
010B EB17 @13: JMP @7
; STATEMENT # 231
010D 8A1E0500 MOV BL,ESCP+5H
0111 B700 BH,0H
0113 8A4604 AL,[BP].CHAR
0116 88870600 MOV ESCP[BP+6H],AL
; STATEMENT # 232
011A FE03 INC BL
    
```

```

0239 55      PUSH    BP
023A 8BEC    MOV     BP,SP      ; STATEMENT # 287
023C FF7604  PUSH    [BP+1,LINE+1]
023F B000    MOV     AL,0H
0241 50      PUSH    AX           ; 2
0242 FE360000  PUSH   HI7_PAR ; 3
0246 F1E0000  CALL   EDC
                ; STATEMENT # 289
024A 5D      POP     BP
024B C20200  RET     2H
                ; STATEMENT # 290
                EDI
                ENDP
                ; STATEMENT # 290
024E 55      PROC    NEAR
024F 8BEC    PUSH   BP
0251 F1E0000  MOV     BP,SP
                ; STATEMENT # 292
0255 5D      CALL   UIES
                ; STATEMENT # 294
0256 C3      POP     BP
                RET
                ENDP
                ; STATEMENT # 303
                P_UJM
    
```

CROSS-REFERENCE LISTING

Cursor	Offset	Symbol	Address	Symbol	Address	Symbol	Address	Symbol	Address
69	0000H	CURSOR	000000B	LITERALLY '01#0000B'					
157	0000H	CURSOR	000000B	PROCEDURE EXTERNAL(46) STACK=0000H					
2		DC		LITERALLY 'DECLARE'	3	4	5	6	7
				10 11 12 13 14 15 16 17 18 19 20 21 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 61 62 63 64					
				65 66 67 68 69 70 71 72 73 74 75 76					
				77 78 79 80 81 82 83 84 85 86 87 88					
				89 90 91 92 93 94 95 96 97 98 99 100					
				101 102 103 104 105 106 107 108 109 110 111 112					
				113 114 115 116 117 118 119 120 121 122 123 124					
				125 126 127 128 129 130 131 132 133 134 135 136					
				137 138 139 140 141 142 143 144 145 146 147 148					
				185 186 187 188					
244	0138H	DCA		PROCEDURE PUBLIC STACK=0006H					
79	0000H	DC1		PROCEDURE EXTERNAL(11)					
11	0000H	DEL		LITERALLY '07FH'	214	215			
80	0000H	DFC		POINTER EXTERNAL(12)					
95	0000H	DMAF		BYTE ARRAY(256) EXTERNAL(27)					
255	017BH	DOC		PROCEDURE BYTE STACK=0004H		224			
77	0000H	DS_SIZE		WORD EXTERNAL(9)					
21	0000H	DXMIC		PROCEDURE EXTERNAL(0) STACK=0000H					
159	0000H	O_CRT		PROCEDURE EXTERNAL(47) STACK=0000H					
162	0000H	D_TTY_RES		PROCEDURE EXTERNAL(48) STACK=0000H					
31	0000H	O_XMITC		POINTER EXTERNAL(13)					
82	0000H	EDC		POINTER EXTERNAL(14)	287				
284	0239H	EMEC		PROCEDURE PUBLIC STACK=000EH					
83	0000H	ENABLE_LINES		POINTER EXTERNAL(15)					
163	0000H	EOL		PROCEDURE EXTERNAL(51) STACK=0000H					
12		EOS		LITERALLY '080H'					
13		ESC		LITERALLY '080H'					
14		ESC		LITERALLY '01BH'	212				
109	0000H	ESCP		STRUCTURE EXTERNAL(38)					
		COMMAND		BYTE 223* 237* 269 270 271 272 273 275 276 277					
		FUNCTION		WORD 270* 272* 274* 276* 278* 279* 280					
		MODE		BYTE 198* 210 219* 220* 226* 268*					
		OPER_COUNT		BYTE 224* 227 233					
		OPER_INDEX		BYTE 225* 232* 232 233					
		OPERAND		BYTE ARRAY(4)	231*				
173	0000H	ESC_TABLE_1		WORD ARRAY(20) DATA	270				
174	0023H	ESC_TABLE_2		WORD ARRAY(7) DATA	272				
175	0035H	ESC_TABLE_3		WORD ARRAY(21) DATA	276				
5		FALSE		LITERALLY '000H'	200				
15		FF		LITERALLY '00CH'					
24	0000H	FLAGS		PROCEDURE WORD EXTERNAL(1) STACK=0000H					
84	0000H	FONT		POINTER EXTERNAL(16)					
93	0000H	FONT_SIZE		WORD EXTERNAL(25)					
172	0000H	FOKE		BYTE IN PROC (SCA) PARAMETER	172				
		H19CRT		PROCEDURE STACK=0000H					
110	0000H	H19_MODE		STRUCTURE EXTERNAL(39)					
		ALTERNATE		BYTE 200*					
		ANSI		BYTE 200* 218					
		AUTO_CR		BYTE					
		AUTO_LF		BYTE					
		AUTO_REPEAT		BYTE					

CROSS-REFERENCE LISTING

Line	Address	Label	Symbol	Start	End	Start	End	Start	End
135	0000H	KY_F12	LITERALLY '0A2H'	11	12	13	14	15	16
133	0000H	KY_HELP	LITERALLY '095H'	42	43	44	45	46	47
142	0000H	KY_HOME	LITERALLY '0A9H'	54	55	56	57	58	59
136	0000H	KY_ID_CHAR	LITERALLY '0A3H'	66	67	68	69	70	71
137	0000H	KY_ID_LINE	LITERALLY '0A4H'	113	114	115	116	117	118
146	0000H	KY_KP_0	LITERALLY '0B0H'	125	126	127	128	129	130
147	0000H	KY_KP_9	LITERALLY '0B9H'	137	138	139	140	141	142
144	0000H	KY_KP_MINUS	LITERALLY '0ADH'	186	187	188	189		
145	0000H	KY_KP_PERIOD	LITERALLY '0AEH'						
141	0000H	KY_LEFT	LITERALLY '0A8H'						
140	0000H	KY_RIGHT	LITERALLY '0A7H'						
148	0000H	KY_SHIFT_OFFSET	LITERALLY '0A5H'						
138	0000H	KY_UP	LITERALLY '00AH'						
17	0004H	LF	BYTE IN PROC (EDL) PARAMETER AUTOMATIC						
285	0000H	LINE	BYTE IN PROC (ENABLE_LINES) PARAMETER						
169	0000H	LINES	BYTE IN PROC (ENABLE_LINES) PARAMETER						
190	0000H	LINE_INDEX	BYTE PUBLIC						
3	0000H	LF	LITERALLY 'LITERALLY'	11	12	13	14	15	16
177	0000H	MASK	POINTER EXTERNAL(17)	177					
85	0000H	MDC	POINTER EXTERNAL(18)						
86	0000H	MDL	LITERALLY '54H'						
74		MTR_ISSUE	LITERALLY '12H'						
73		MTR_VERSION	LITERALLY '3'						
189		M_ANS	LITERALLY '1'						
187		M_ESC	LITERALLY '0'						
186		M_NRM	LITERALLY '2'						
188		M_OPR	BYTE ARRAY(14) DATA						
254	0060H	OPC	POINTER STACK=0004H	260	263				
266	01ADH	PES	POINTER EXTERNAL(19)	228	234				
87	0000H	PSP	BYTE EXTERNAL(32)						
100	0000H	P_CDN	PROCEDURE EXTERNAL(58) STACK=0000H	193					
295	0000H	P_CLF	PROCEDURE EXTERNAL(59) STACK=0000H	193					
297	0000H	P_COLGR	PROCEDURE EXTERNAL(95) STACK=0000H	195					
369	0000H	P_CPLR	PROCEDURE EXTERNAL(62) STACK=0000H	195					
303	0000H	P_CPR	PROCEDURE EXTERNAL(60) STACK=0000H	193					
299	0000H	P_CRT	PROCEDURE EXTERNAL(61) STACK=0000H	193					
301	0000H	P_CUP	PROCEDURE EXTERNAL(74) STACK=0000H	193					
327	0000H	P_DC	PROCEDURE EXTERNAL(63) STACK=0000H	194					
305	0000H	P_DCA	PROCEDURE EXTERNAL(84) STACK=0000H	195					
347	0000H	P_DK	PROCEDURE EXTERNAL(75) STACK=0000H	193					
329	0000H	P_DL	PROCEDURE EXTERNAL(77) STACK=0000H	193					
333	0000H	P_EAKM	PROCEDURE EXTERNAL(79) STACK=0000H	193					
337	0000H	P_EAM	PROCEDURE EXTERNAL(68) STACK=0000H	195					
315	0000H	P_EBOL	PROCEDURE EXTERNAL(70) STACK=0000H	193					
317	0000H	P_EBOP	PROCEDURE EXTERNAL(71) STACK=0000H	193					
319	0000H	P_EEDL	PROCEDURE EXTERNAL(82) STACK=0000H	193					
321	0000H	P_EEOP							
343	0000H	P_EGM							


```

77 0000H 1 VERT_LINE . . . . . BYTE EXTERNAL(29) 248*
57 0000H VIDC_BLUE . . . . . LITERALLY '00000100B'
55 0000H VIDC_FLASH . . . . . LITERALLY '00001000B'
38 0000H VIDC_GREEN . . . . . LITERALLY '00000010B'
50 0000H VIDC_FRAME . . . . . LITERALLY '10000000B'
59 0000H VIDC_REDE . . . . . LITERALLY '00000001B'
52 0000H VIDC_WBLU . . . . . LITERALLY '01000000B'
53 0000H VIDC_WGRN . . . . . LITERALLY '00100000B'
54 0000H VIDC_WRED . . . . . LITERALLY '00010000B'
51 0000H VIDC_WXXX . . . . . LITERALLY '01110000B'
56 0000H VIDC_XXYE . . . . . LITERALLY '00000111B'
32 0000H VIDE0_INTR_A . . . . . PROCEDURE EXTERNAL(4) STACK=0000H
44 0000H VID_CDC . . . . . LITERALLY '10_VIDE0+1'
43 0000H VID_CDD . . . . . LITERALLY '10_VIDE0+0'
42 0000H VID_CMD . . . . . LITERALLY '10_VIDE0+0'
45 0000H VRMM_GAT . . . . . LITERALLY '10_VIDE0+2'
47 0000H VRMM_MPC . . . . . LITERALLY '10_VIDE0+3'
46 0000H VRMM_MPD . . . . . LITERALLY '10_VIDE0+2'
18 0000H VT . . . . . LITERALLY '00BH'
75 0000H WIP . . . . . BYTE ARRAY(5) EXTERNAL(7)
154 0000H WRITK . . . . . PROCEDURE EXTERNAL(45) STACK=0000H
91 0000H XCA . . . . . POINTER EXTERNAL(23)
105 0000H XCOLOR_BACK . . . . . LITERALLY '00#111#000B'
104 0000H XCOLOR_FORE . . . . . LITERALLY '00#000#111B'
34 0000H XEC . . . . . PROCEDURE EXTERNAL(5) STACK=0000H
112 0000H XMT . . . . . STRUCTURE EXTERNAL(41)
      10 XMT . . . . . BYTE
      1 BURST . . . . . INTEGER
      2 BCOUNT . . . . . WORD 207
      2 CCOUNT . . . . . BYTE
      1 COL . . . . . BYTE
      1 ROW . . . . . BYTE
      1 COLOR . . . . . BYTE
      1 GRAPHIC . . . . . BYTE
      1 REVERSE . . . . . BYTE
183 0000H XMTSC . . . . . PROCEDURE EXTERNAL(57) STACK=0000H
19 0000H XOFF . . . . . LITERALLY '013H'
20 0000H XON . . . . . LITERALLY '011H'
103 0000H XREVERSE . . . . . LITERALLY '10#000#000B'

```

MODULE INFORMATION:

```

CODE AREA SIZE = 0257H 597D
CONSTANT AREA SIZE = 0000H 0D
VARIABLE AREA SIZE = 0004H 4D
MAXIMUM STACK SIZE = 000EH 14D
2134 LINES READ
0 PROGRAM WARNINGS
0 PROGRAM ERRORS

```

END OF PL/M-86 COMPILATION

SERIES: III 8086/8087/8088 MACRO ASSEMBLER V1.0 ASSEMBLY OF MODULE H19_CRT_A86
 OBJECT MODULE PLACED IN : F2:H19A.OBJ
 INVOCATION LINE CONTROLS: PRINT(:TO:) XREF ERRORPRINT(:TO:)

LOC	OBJ	LINE	SOURCE
		1	;; h19crt.a86
		2	;;
		3	;;
		4	;; These routines are optimized PL/M-86 routines.
		5	;; The output of the PL/M-86 compiler has been
		6	;; extensively 'filtered' to produce files which
		7	;; contain optimized code. The basic optimization
		8	;; was to eliminate the procedure prologue/epilogue
		9	;; code for procedures which pass no parameters.
		10	;; This consists of reducing the following:
		11	;;
		12	xxxx PROC NEAR
		13	PUSH BP
		14	MOV BP,SP
		15	:
		16	POP BP
		17	RET
		18	xxxx ENDP
		19	;;
		20	to:
		21	;;
		22	xxxx PROC NEAR
		23	:
		24	:
		25	RET
		26	xxxx ENDP
		27	;;
		28	;; This saves at least four bytes of code for each
		29	;; of the procedures involved.
		30	;;
		31	;;
		32	;; \$ EJECT

LOC	OBJ	LINE	SOURCE
00003		=2	AUTO_LF DB ? ; Auto Line-Feed on Carriage-Return
00004		=2	AUTO_REPEAT DB ? ; Auto-Repeat Keyboard
00005		=2	BWD DB ? ; Black and White Optimization
00006		=2	CURSOR DB ? ; Programmed Cursor Value
00007		=2	CURSOR_ON DB ? ; Cursor Enabled
00008		=2	EXPAND DB ? ; Expand Key-Board Characters
00009		=2	GRAPHIC DB ? ; Graphic Character Mode
0000A		=2	INSERT DB ? ; Insert Character Mode
0000B		=2	KEY_EN DB ? ; Key-Board Enable
0000C		=2	REVERSE DB ? ; Reverse Video
0000E		=2	SHIFTED DB ? ; Shifted Key-Pad Mode
0000F		=2	STATUS DB ? ; Status-Line Enabled
0010		=2	UPDN DB ? ; Key-Board Up/Down Mode
0011		=2	WRAP DB ? ; Wrap at End-of-Line
		=2	H19_MCODE_STRUC ENDS
		=2	H19_PAR_STRUC STRUC
0000		=2	CPL DB ? ; Characters Per Line
0001		=2	DSC DB ? ; Displayed Scan Lines per Character
0002		=2	LPS DB ? ; Lines Per Screen
0003		=2	POVRAM DB ? ; Planes of Video RAM
0004		=2	SLI DB ? ; Status Line Index
0005		=2	SPC DB ? ; Scan Lines per Character
0006		=2	SM401 DB ? ; H19-Switch 401
0007		=2	SM402 DB ? ; H19-Switch 402
0008		=2	VRAM_SIZE DB ? ; 0-32KBytes, 1-64KBytes
		=2	H19_PAR_STRUC ENDS
		=2	XMT_STRUC STRUC
0000		=2	BURST DB ? ; Characters to Transmit per VSYNC
0001		=2	RCOUNT DB ? ; Remaining Characters in current Burst
0003		=2	COUNT DB ? ; Characters left to Transmit
0005		=2	COL DB ? ; Horizontal Column to Transmit
0006		=2	ROW DB ? ; Vertical Row to Transmit
0007		=2	XMT_COLOR DB ? ; Current COLOR State Transmitted
0008		=2	XMT_GRAPHIC DB ? ; Current GRAPHIC State Transmitted
0009		=2	XMT_REVERSE DB ? ; Current REVERSE State Transmitted
		=2	XMT_STRUC ENDS
		=2	\$RESTORE
		=2	; Global's Definitions
		=1	XIF(ZNES(ASTLIB,ZSEGMENT_LABEL)) THEN (
		=1	ASTLIB SEGMENT BYTE PUBLIC 'CODE'
		=1	EXTRN VIDEO_INTR_A:AFAR
		=1	ASTLIB ENDS
		=1)FI
		=1	XIF(ZNES(FONTAB,ZSEGMENT_LABEL)) THEN (
		=1	FONTAB SEGMENT BYTE PUBLIC 'DATA'
		=1	EXTRN MTR_FONT:BYTE
		=1	FONTAB ENDS

LOC	OBJ	LINE	SOURCE
=1			EXTRN DFC:DWORD ; Display Font Character
=1			EXTRN D_XMTC:DWORD ; Dumb Terminal Transmit Character
=1			EXTRN EDC:DWORD ; Erase Display Character
=1			EXTRN EMEC:DWORD ; Extend-Mode Escape Character
=1			EXTRN FONT:DWORD ; Pointer to Font Table
=1			EXTRN MDC:DWORD ; Move Display Character
=1			EXTRN MDL:DWORD ; Move Display Line
=1			EXTRN PROMPT:DWORD ; Display Monitor Prompt
=1			EXTRN RDC:DWORD ; Read Display Character
=1			EXTRN S_XMTC:DWORD ; Smart Terminal Transmit Character
=1			EXTRN UIES:DWORD ; Un-Implemented Escape Sequence
=1			EXTRN XCA:DWORD ; Transmit Character Attributes
=1			EXTRN BOOT_PAR:BOOT_PAR_STRUC
=1			EXTRN COLOR:COLOR_STRUC
=1			EXTRN CRIC_CURSOR:CRIC_STRUC
=1			EXTRN CRIC_DISPLAY:CRIC_STRUC
=1			EXTRN ESCP:ESCP_STRUC
=1			EXTRN H19_MODE:H19_MODE_STRUC
=1			EXTRN H19_PAR:H19_PAR_STRUC
=1			EXTRN XMT:XMT_STRUC
=1			ENDS
		218	DATA
		219)FI
		220	XIF(2NES(VIDEQA,2SEGMENT LABEL)) THEN (
			VIDEQA SEGMENT BYTE PUBLIC 'CODE'
			VIDEQA)FI
			ENDS
		224	
		225	
		226	CGROUP GROUP MTR100, CODE,ASTLIB,VIDEQA
		227	
		228	
		229	\$RESTORE
		230	
		231	\$EJECT

LOC	OBJ	LINE	SOURCE	ENDP	STATEMENT #
0018		287	P_CDN	ENDP	
		288			
		289	P_CLF	PROC NEAR	STATEMENT # 301
		290		PUBLIC P_CLF	
		291		PUSH BP	
		292		MOV BP, SP	
		293			
0018	A00000	294		MOV AL, HORIZ_CHAR	STATEMENT # 303
001B	0AC0	295		OR AL, AL	
001D	740A	296		JZ @34	
		297			
001F	FF360000	298		PUSH WORD PTR VERT_LINE, 1	STATEMENT # 304
0023	FEC8	299		DEC AL	
0025	50	300		PUSH AX	
0026	E80000	301		CALL DCA	STATEMENT # 2
0029		302	@34:		
		303			
		304		POP BP	STATEMENT # 306
0029	C3	305		RET	
		306		ENDP	
		307			
002A		308	P_CRT	PROC NEAR	STATEMENT # 307
		309		PUBLIC P_CRT	
		310		PUSH BP	
		311		MOV BP, SP	
		312			
002A	FF360000	313		PUSH WORD PTR VERT_LINE, 1	STATEMENT # 309
002E	A00000	314		MOV AL, HORIZ_CHAR	
0031	FEC0	315		INC AL	
0033	50	316		PUSH AX	STATEMENT # 2
0034	E80000	317		CALL DCA	
		318			
		319		POP BP	STATEMENT # 311
0037	C3	320		RET	
		321		ENDP	
		322	P_CRT		
0038		323		PROC NEAR	STATEMENT # 312
		324		PUBLIC P_CUP	
		325		PUSH BP	
		326		MOV BP, SP	
		327			
0038	A00000	328		MOV AL, VERT_LINE	STATEMENT # 314
003B	3A060400	329		CMP AL, H19_PAR_SLI	
003F	730A	330		JNB @35	
		331			
		332			STATEMENT # 315
0041	FEC8	333		DEC AL	
0043	50	334		PUSH AX	STATEMENT # 1
0044	FF360000	335		PUSH WORD PTR HORIZ_CHAR, 2	
0048	E80000	336		CALL DCA	
004B		337	@35:		
		338		POP BP	STATEMENT # 317
		339		RET	
		340		ENDP	
		341			STATEMENT # 318

LOC	OBJ	LINE	SOURCE
0082		397	P_HDM ENDP
		398	PROC NEAR ; STATEMENT # 336
		399	P_RCP PUBLIC P_RCP
		400	PUSH BP
		401	MOV BP,SP
		402	;
		403	; STATEMENT # 338
0082	FF360000	404	PUSH WORD PTR SAVED_VERT_LINE; 1
0086	FF360000	405	PUSH WORD PTR SAVED_HORZ_CHAR; 2
008A	E80000	406	CALL DCA
		407	; STATEMENT # 340
		408	POP BP
008D	C3	409	RET
		410	ENDP
		411	; STATEMENT # 341
008E		412	P_RLF PROC NEAR
		413	PUBLIC P_RLF
		414	PUSH BP
		415	MOV BP,SP
		416	;
		417	; STATEMENT # 343
008E	A00000	418	MOV AL,VERT_LINE
0091	3A060400	419	CALL AL,H19_PAR_SLI
0095	7501	420	JNZ @36
		421	; STATEMENT # 344
		422	POP BP
0097	C3	423	RET
		424	; STATEMENT # 345
0098	803E000000	425	CMPL JZ @36
009D	7405	426	JZ @38
		427	; STATEMENT # 346
009F	FE0E0000	428	DEC VERT_LINE
		429	; STATEMENT # 347
		430	POP BP
00A3	C3	431	RET
00A4		432	;
00A4	E80000	433	CALL @38
		434	CALL REV_SCROLL
		435	; STATEMENT # 349
00A7	C3	436	POP BP
		437	RET
		438	ENDP
00A8		439	P_RLF ; STATEMENT # 350
		440	PROC NEAR
		441	P_SCP PUBLIC P_SCP
		442	PUSH BP
		443	MOV BP,SP
		444	; STATEMENT # 352
00A8	A00000	445	MOV AL,HORZ_CHAR
00AB	A20000	446	MOV SAVED_HORZ_CHAR,AL
		447	; STATEMENT # 353
00AE	A00000	448	MOV AL,VERT_LINE
00B1	A20000	449	MOV SAVED_VERT_LINE,AL
		450	; STATEMENT # 355
00B4	C3	451	POP BP
			RET

LOC	OBJ	LINE	SOURCE
00F7	FF360000	507	PUSH BP
00FB	A00000	508	MOV BP,SP
00FE	50	509	;
00FF	A00000	510	PUSH WORD PTR VERT_LINE; 1
0102	2A060000	511	MOV AL,HORZ_CHAR
0106	50	512	PUSH AX
0107	FF1E0000	513	MOV AL,H19_PAR,CPL
		514	SUB AL,HORZ_CHAR
		515	PUSH AX
		516	CALL EDC
		517	;
010B	C3	518	POP BP
		519	RET
		520	ENDP
		521	;
010C		522	PROC NEAR
		523	PUBLIC P_EEOP
		524	PUSH BP
		525	MOV BP,SP
		526	;
		527	CALL P_EEOL
		528	;
		529	MOV AL,VERT_LINE
		530	INC AL
		531	MOV LINE_INDEX,AL
		532	;
		533	MOV AL,H19_PAR,SLI
		534	DEC AL
		535	MOV CL,LINE_INDEX
		536	CL,AL
		537	CMF CL,AL
		538	JA @44
		539	;
		540	PUSH CX
		541	CALL EDI
		542	;
		543	INC LINE_INDEX
		544	JNZ @43
		545	;
		546	POP BP
		547	RET
		548	ENDP
		549	;
012E	C3	550	PROC NEAR
		551	PUBLIC P_ELIN
		552	PUSH BP
		553	MOV BP,SP
		554	;
		555	MOV WORD PTR VERT_LINE; 1
		556	CALL EDI
		557	;
		558	POP BP
		559	RET
		560	ENDP
		561	;

LOC	OBJ	LINE	SOURCE
0181	B400	617	MOV AH,0H
0183	50	618	PUSH AX ; 2
0184	A00000	619	MOV AL,HORZ_CHAR
0187	50	620	PUSH AX ; 3
0188	A00000	621	MOV AL,H19_PAR_CPL
018B	2A060000	622	SUB AL,HORZ_CHAR
018F	FEC8	623	DEC AL
0191	B400	624	MOV AH,0H
0193	50	625	PUSH AX ; 4
0194	FF1E0000	626	CALL MDC
0198	FF360000	627	CALL ; STATEMENT # 407
019C	A00000	628	PUSH WORD PTR VERT_LINE; 1
019F	FEC8	629	MOV AL,H19_PAR_CPL
01A1	B400	630	DEC AL
01A3	50	631	MOV AH,0H
01A4	B001	632	PUSH AX ; 2
01A6	50	633	MOV AL,1H
01A7	FF1E0000	634	PUSH AX ; 3
		635	CALL EDC
		636	CALL ; STATEMENT # 409
		637	POP BP
01AB	C3	638	RET
		639	ENDP
01AC		640	P_DC ; STATEMENT # 410
		641	PROC NEAR P_DL
		642	PUBLIC P_DL
		643	PUSH BP
		644	MOV BP,SP
		645	CALL ; STATEMENT # 412
01AC	A00000	646	MOV AL,VERT_LINE
01AF	3A060400	647	CALL AL,H19_PAR_SLI
01B3	742E	648	JZ @50
		649	CALL ; STATEMENT # 414
		650	INC AL
01B5	FEC0	651	MOV LINE_INDEX,AL
01B7	A20000	652	MOV ; STATEMENT # 415
01BA	A00400	653	MOV AL,H19_PAR_SLI
01BD	FEC8	654	DEC AL
01BF	8A0E0000	655	MOV CL,LINE_INDEX
01C3	3AC8	656	CMP CL,AL
01C5	7710	657	JJA @52
		658	CALL ; STATEMENT # 415
01C7	51	659	PUSH CX ; 1
01C8	FEC9	660	DEC CL
01CA	B500	661	MOV CH,0H
01CC	51	662	PUSH CX ; 2
01CD	FF1E0000	663	CALL MDL
		664	CALL ; STATEMENT # 416
01D1	FE060000	665	INC LINE_INDEX
01D5	75E3	666	JNZ @51
01D7	A00400	667	MOV ; STATEMENT # 417
01DA	FEC8	668	DEC AL
01DC	50	670	PUSH AX ; 1
		671	CALL ; STATEMENT # 417

LOC	OBJ	LINE	SOURCE
0229		727	P_EAKM PROC NEAR
		728	PUBLIC P_EAKM
		729	PUSH BP
		730	MOV BP,SP
		731	;
0229	C6060000FF	732	MOV H19_MODE,ALTERNATE,0FFH
		733	;
		734	POP BP
		735	RET
022E	C3	736	ENDP
		737	;
022F		738	PROC NEAR
		739	PUBLIC P_XAKM
		740	PUSH BP
		741	MOV BP,SP
		742	;
022F	C6060000FF	743	MOV H19_MODE,ALTERNATE,0H
		744	;
		745	POP BP
		746	RET
0234	C3	747	ENDP
		748	;
0235		749	PROC NEAR
		750	PUBLIC P_EAM
		751	PUSH BP
		752	MOV BP,SP
		753	;
0235	C6060100FF	754	MOV H19_MODE,ANG1,0FFH
		755	;
		756	POP BP
		757	RET
023A	C3	758	ENDP
		759	;
023B		760	PROC NEAR
		761	PUBLIC P_EICM
		762	PUSH BP
		763	MOV BP,SP
		764	;
023B	C6060A00FF	765	MOV H19_MODE,INSERT,0FFH
		766	;
		767	POP BP
		768	RET
0240	C3	769	ENDP
		770	;
0241		771	PROC NEAR
		772	PUBLIC P_XICM
		773	PUSH BP
		774	MOV BP,SP
		775	;
0241	C6060A0000	776	MOV H19_MODE,INSERT,0H
		777	;
		778	POP BP
		779	RET
0246	C3	780	ENDP
		781	;

; STATEMENT # 462

LOC	OBJ	LINE	SOURCE
0268	C6060E00FF	837	MOV BP,SP ; STATEMENT # 486
		838	MOV H19_MODE,SHIFTED,0FFH ; STATEMENT # 488
0270	C3	840	POP BP ; STATEMENT # 488
		841	RET
0271	C3	842	ENDP
		843	PROC NEAR
		844	P_XKPS
		845	PUBLIC P_XKPS
		846	PUSH BP
		847	MOV BP,SP
0271	C6060E0000	849	MOV H19_MODE,SHIFTED,0H ; STATEMENT # 491
		850	POP BP ; STATEMENT # 493
		851	RET
0276	C3	852	ENDP
		853	PROC NEAR
		854	P_XKPS
		855	PUBLIC P_XKPS
0277	C3	856	PROC NEAR
		857	P_ERVM
		858	PUBLIC P_ERVM
		859	PUSH BP
		860	MOV BP,SP
0277	C7060C00FFFF	861	MOV H19_MODE,REVERSE,0FFFFH ; STATEMENT # 496
		862	POP BP ; STATEMENT # 498
		863	RET
027D	C3	864	ENDP
		865	PROC NEAR
		866	P_XRVM
		867	PUBLIC P_XRVM
		868	PUSH BP
		869	MOV BP,SP
027E	C3	871	MOV H19_MODE,REVERSE,0H ; STATEMENT # 501
		872	POP BP ; STATEMENT # 503
		873	RET
0284	C3	874	ENDP
		875	PROC NEAR
		876	P_ENRAP
		877	PUBLIC P_ENRAP
		878	PUSH BP
		879	MOV BP,SP
0285	C6061100FF	881	MOV H19_MODE,WRAP,0FFH ; STATEMENT # 506
		882	POP BP ; STATEMENT # 508
		883	RET
028A	C3	884	ENDP
		885	PROC NEAR
		886	P_XWRAP
		887	PUBLIC P_XWRAP
		888	PUSH BP
028B	C3	889	MOV H19_MODE,WRAP,0FFH ; STATEMENT # 509
		890	POP BP
		891	RET

LOC	OBJ		LINE	SOURCE			
02D1	2903	R	947		MOV		
02D3	2F03	R	948		MOV		
02D5			949	@60:			
			950				
02D5	C6060F0000	E	951		MOV		
			952				
02D6	FF360400	E	953		PUSH		
02D6	E80000	E	954		CALL		
			955				
02E1	FF360400	E	956		PUSH		
02E5	E80000	E	957		CALL		
			958				
			959		POP		
02E8	C3		960		RET		
02E9			961	@61:			
02E9	B003		962		MOV		
02EB	EB4990		963		JMP		
			964				
02EE			965	@63:			
02EE	B0E0		966		MOV		
02F0	50		967		PUSH		
02F1	A00500	E	968		MOV		
02F4	2C03		969		SUB		
02F6	EB2390		970		JMP		
			971	@64:			
			972				
02F9	C6060700FF	E	973		MOV		
			974				
02FE	E80000	E	975		CALL		
			976				
			977		POP		
0301	C3		978		RET		
0302	E86CFF		979	@65:			
			980		CALL		
			981				
0305	C3		982		POP		
0306	EB26FF		983		RET		
			984	@66:			
			985		CALL		
			986				
0309	C3		987		POP		
030A	C606030000	E	988		RET		
			989	@67:			
			990		MOV		
			991				
030F	C3		992		POP		
0310	C606020000	E	993		RET		
			994	@68:			
			995		MOV		
			996				
0315	C3		997		POP		
			998		RET		
			999				
0316	B01F		1000	@70:			
			1001		MOV		

CGROUP:@74
CGROUP:@75

; STATEMENT # 526
H19_MODE,STATUS,0H
; STATEMENT # 527

WORD PTR H19_PAR,SLI; 1
EDL
; STATEMENT # 528

WORD PTR H19_PAR,SLI; 1
ENABLE_LINES
; STATEMENT # 530

BP
; STATEMENT # 532
AL,0EH
AX
; 1

AL,H19_PAR,SPC
AL,3H
@26

; STATEMENT # 534
H19_MODE,CURSUR_ON,0FFH
; STATEMENT # 535

SCP
; STATEMENT # 537
BP

CALL
P_XKPS
; STATEMENT # 538

BP
; STATEMENT # 539
P_XAKM

BP
; STATEMENT # 540
H19_MODE,AUTO_LF,0H

BP
; STATEMENT # 541
H19_MODE,AUTO_CR,0H

BP
; STATEMENT # 542
AL,1FH

LOC	OBJ	LINE	SOURCE
0358	7803	1057	@78: DW
0359	7503	1059	LB2 DW
035C	CB03	1060	DM
035E	9A03	1061	DM
0362	9F03	1062	DM
0364	AC03	1063	DM
0366	B003	1064	DM
0368	B603	1065	DM
036A	BC03	1066	DM
036C	BC03	1067	DM
036E	C603	1068	DM
0370	CB03	1069	DM
0372	CB03	1070	DM
0374	CF03	1071	DM
0376	DS03	1072	DM
0378		1073	DM
0378	A00F00	1074	@79: ; STATEMENT # 559
037B	D018	1075	MOV AL, H19_MODE, STATUS
037D	7261	1076	RCR AL, 1
		1077	JB @76
		1078	
		1079	
037F	C6060F00FF	1080	MOV ; STATEMENT # 561
		1081	H19_MODE, STATUS, 0FFH
0384	FF360400	1082	; STATEMENT # 562
		1083	WORD PTR H19_PAR, SLI; 1
0388	EB0000	1083	ED
		1084	
038B	A00400	1085	MOV ; STATEMENT # 563
		1086	AL, H19_PAR, SLI
038E	FEC0	1086	INC AL
0390	50	1087	PUSH AX
		1088	; 1
0391	EB0000	1088	CALL ENABLE_LINES
		1089	; STATEMENT # 566
		1090	
		1091	POP BP
0394	C3	1091	RET
0395	B004	1092	
0395	B004	1093	@31: MOV AL, 4H
0397	EB4390	1094	JMP @42
		1095	; STATEMENT # 568
		1096	
039A	BEF0	1096	@33: MOV AL, 0E0H
039A	BEF0	1097	JMP @37
039C	EB2090	1098	
039F		1099	@34: ; STATEMENT # 570
		1100	
039F	C606070000	1101	MOV H19_MODE, CURSOR_ON, 0H
		1102	; STATEMENT # 571
03A4	EB0000	1103	CALL SCP
		1104	; STATEMENT # 573
		1105	
03A7	C3	1105	POP BP
03A8		1106	RET
03A8	EB00FE	1107	@35: CALL P_EKPS
		1108	; STATEMENT # 574
		1109	
		1110	POP BP
		1111	RET

LOC	OBJ	LINE	SOURCE	LOC	OBJ	LINE	SOURCE
03E1	A00600	1171	MOV	03E1	A00600	1171	MOV
03E4	B130	1172	MOV	03E4	B130	1172	MOV
03E6	3AC8	1173	CMP	03E6	3AC8	1173	CMP
03E8	7720	1174	JA	03E8	7720	1174	JA
03EA	B237	1175	MOV	03EA	B237	1175	MOV
03EC	3AC2	1176	CMP	03EC	3AC2	1176	CMP
03EE	771A	1177	JA	03EE	771A	1177	JA
03F0	A00700	1178	MOV	03F0	A00700	1178	MOV
03F3	3AC8	1179	CMP	03F3	3AC8	1179	CMP
03F5	7713	1180	JA	03F5	7713	1180	JA
03F7	3AC2	1181	CMP	03F7	3AC2	1181	CMP
03F9	770F	1182	JA	03F9	770F	1182	JA
03FB	A00600	1183	MOV	03FB	A00600	1183	MOV
03FE	2AC1	1184	SUB	03FE	2AC1	1184	SUB
0400	50	1185	PUSH	0400	50	1185	PUSH
0401	A00700	1187	MOV	0401	A00700	1187	MOV
0404	2AC1	1188	SUB	0404	2AC1	1188	SUB
0406	50	1189	PUSH	0406	50	1189	PUSH
0407	E80000	1190	CALL	0407	E80000	1190	CALL
040A		1191		040A		1191	
040A	C3	1192	POP	040A	C3	1192	POP
		1193	RET			1193	RET
		1194				1194	
		1195				1195	
040B		1196		040B		1196	
		1197				1197	
		1198				1198	
		1199				1199	
		1200				1200	
		1201				1201	
040B	E03E060030	1202	CMP	040B	E03E060030	1202	CMP
0410	7521	1203	JNZ	0410	7521	1203	JNZ
		1204				1204	
0412	E32F00	1205	CALL	0412	E32F00	1205	CALL
		1206				1206	
0415	B069	1207	MOV	0415	B069	1207	MOV
0417	50	1208	PUSH	0417	50	1208	PUSH
0418	E80000	1209	CALL	0418	E80000	1209	CALL
		1210				1210	
041B	B045	1211	MOV	041B	B045	1211	MOV
041H	50	1212	PUSH	041H	50	1212	PUSH
041E	E80000	1213	CALL	041E	E80000	1213	CALL
		1214				1214	
0421	A00300	1215	MOV	0421	A00300	1215	MOV
0424	0430	1216	ADD	0424	0430	1216	ADD
0426	50	1217	PUSH	0426	50	1217	PUSH
0427	E80000	1218	CALL	0427	E80000	1218	CALL
		1219				1219	
042A	A00800	1220	MOV	042A	A00800	1220	MOV
042D	0441	1221	ADD	042D	0441	1221	ADD

AL, ESCP, OPERAND ; STATEMENT # 592
 AL, CL ; STATEMENT # 595
 AX ; 1
 AL, BYTE PTR ESCP+7H ; 1
 AX, CL ; 2
 SCA ; STATEMENT # 595
 BP ; STATEMENT # 595
 RET ; STATEMENT # 596
 ENDP ; STATEMENT # 596
 PROC NEAR ; STATEMENT # 596
 P_IDENT ; STATEMENT # 596
 PUBLIC P_IDENT ; STATEMENT # 596
 PUSH BP ; STATEMENT # 596
 MOV BP, SP ; STATEMENT # 596
 ; STATEMENT # 596
 ESCP, OPERAND, 30H ; STATEMENT # 600
 JNZ @97 ; STATEMENT # 600
 TEC ; STATEMENT # 601
 ; STATEMENT # 601
 AL, 69H ; STATEMENT # 601
 ; 1 ; STATEMENT # 602
 SXMT ; STATEMENT # 602
 ; STATEMENT # 602
 AL, 45H ; STATEMENT # 602
 ; 1 ; STATEMENT # 603
 SXMT ; STATEMENT # 603
 ; STATEMENT # 603
 AL, H19_FAR, POV RAM ; STATEMENT # 603
 ; 1 ; STATEMENT # 604
 AX ; STATEMENT # 604
 SXMT ; STATEMENT # 604
 ; STATEMENT # 604
 AL, H19_FAR, VRAM_SIZE ; STATEMENT # 604
 ; STATEMENT # 604
 AL, 41H ; STATEMENT # 604