MSX2 TECHNICAL HANDBOOK

Edited by: ASCII Systems Division
Published by: ASCII Coprporation - JAPAN
First edition: March 1987

Text file typed by: Nestor Soriano (Konami Man) - SPAIN
                  October 1997

Changes from the original in APPENDIX 1:

- In description of ENASLT, the needed input in HL has been added.

- In description of GETYPR, the Input field has been added.

- In description of INITXT (MAIN), the reference to "INIPLOT" has been corrected to "INIPLT".

- In description of SUBROM routine, the mark "*1" has been erased.

- In description of INITXT (SUB), the needed input in LINL40 has been added.

- Description of PHYDIO routine has been added.

Changes from the original in APPENDIX 2:

- In the explanation before Figure A.3, the indication about the excess 64 method has been added.

- In Figure A.3, in the third byte, "63rd power of 10" has been corrected to "-63rd power of 10".

- In the explanation before Figure A.3, the indication about the excess 64 method has been added.

- In Figure A.3, in the third byte, "63rd power of 10" has been corrected to "-63rd power of 10".

-=-

APPENDIX 1 - BIOS LISTING

This section lists the 126 BIOS entries available to the user.

There are two kinds of BIOS routines, the ones in MAIN-ROM and the ones in SUB-ROM. They each have different calling sequences which will be described later. The following is the entry notation.

```
Label name (address)    *n
  Function:   descriptions and notes about the function
  Input:      parameters used by call
  Output:     parameters returned by call
  Registers:  registers which will be used (original contentes are lost)
```

The value of *n has the following meanings.

```
    *1 ... same as MSX1
    *2 ... call SUB-ROM internally in screen modes 5 to 8
    *3 ... always call SUB-ROM
    *4 ... do not call SUB-ROM while screen modes 4 to 8 are changed
```

Routines without "*n" are appended for MSX2.

MAIN-ROM

To call routines in MAIN-ROM, the CALL or RTS instruction is used as an ordinary subroutine call.

* RSTs

Among the following RSTs, RST 00H to RST 28H are used by the BASIC interpreter. RST 30H is used for inter-slot calls and RST 38H is used for hardware interrupts.

CHKRAM (0000H) *1
  Function: tests RAM and sets RAM slot for the system
  Input: none
  Output: none
  Registers: all

SYNCHR (0008H) *1
  Funtcion: tests whether the character of [HL] is the specified character. If not, it generates SYNTAX ERROR, otherwise it goes to CHRGTR (0010H).
  Input: set the character to be tested in [HL] and the character to be compared next to RST instruction which calls this routine (inline parameter).

```
Example:    LD    HL,LETTER
            RST   08H
            DB    "A"
                   .
                   .
                   .
      LETTER: DB  "B"
```

  Output: HL is increased by one and A receives [HL]. When the tested character is numerical, the CY flag is set; the end of the statement (00H or 3AH) causes the Z flag to be set.
  Registers: AF, HL

RDSLT (000CH) *1
  Function: selects the slot corresponding to the value of A and reads one byte from the memory of the slot. When this routine is called, the interrupt is inhibited and remains inhibited

```
            even after execution ends.
  Input:    A for the slot number.

             F000EEPP
             -   ----
             |   ||++-------------- Basic slot number (0 to 3)
             |   ++---------------- Expansion slot number (0 to 3)
             +--------------------- "1" when using expansion slot

            HL for the address of memory to be read
  Output:   the value of memory which has been read in A
  Registers:      AF, BC, DE


CHRGTR (0010H)          *1
  Function: gets a character (or a token) from BASIC text
  Input:    [HL] for the character to be read
  Output:   HL is incremented by one and A receives [HL]. When the
            character is numerical, the CY flag is set; the end of the
            statement causes the Z flag to be set.
  Registers:      AF, HL


WRSLT (0014H)           *1
  Function: selects the slot corresponding to the value of A and writes
            one byte to the memory of the slot. When this routine is
            called, interrupts are inhibited and remain so even after
            execution ends.
  Input:    specifies a slot with A (same as RDSLT)
  Output:   none
  Registers:      AF, BC, D


OUTDO (0018H)           *2
  Funtion:  sends the value to current device
  Input:    A for the value to be sent
               sends output to the printer when PTRFLG (F416H) is other
                than 0
               sends output to the file specified by PTRFIL (F864H) when
                PTRFIL is other than 0
  Output:   none
  Registers:      none


CALSLT (001CH)          *1
  Function: calls the routine in another slot (inter-slot call)
  Input:    specify the slot in the 8 high order buts of the IY register
            (same as RDSLT). IX is for the address to be called.
  Output:   depends on the calling routine
  Registers:      depends on the calling routine


DCOMPR (0020H)          *1
  Function: compares the contents of HL and DE
  Input:    HL, DE
  Output:   sets the Z flag for HL = DE, CY flag for HL < DE
  Registers:      AF
```

```
ENASLT (0024H)          *1
  Function: selects the slot corresponding to the value of A and enables
            the slot to be used. When this routine is called, interrupts
            are inhibited and remain so even after execution ends.
  Input:    specify the slot by A (same as RDSLT)
            specify the page to switch the slot by 2 high order bits
            of HL
  Output:   none
  Registers:      all


GETYPR (0028H)          *1
  Function: returns the type of DAC (decimal accumulator)
  Input:    none
  Output:   S, Z, P/V flags are changed depending on the type of DAC:

            integer type                  single precision real type
               C = 1                      C = 1
               S = 1 *                    S = 0
               Z = 0                      Z = 0
               P/V = 1                    P/V = 0 *

            string type             double precision real type
               C = 1                      C = 0 *
               S = 0                      S = 0
               Z = 1 *                    Z = 0
               P/V = 1                    P/V = 1

            Types can be recognised by the flag marked by "*".
  Registers:      AF


CALLF (0030H)           *1
  Function: calls the routine in another slot. The following is the
            calling sequence:

                  RST   30H
                  DB    n     ;n is the slot number (same as RDSLT)
                  DW    nn    ;nn is the called address

  Input:    In the method described above
  Output:   depends on the calling routine
  Registers:      AF, and other registers depending on the calling routine


KEYINT (0038H)          *1
  Function: executes the timer interrupt process routine
  Input:    none
  Output:   none
  Register: none



* I/O initialisation


INITIO (003BH)          *1
```

  Function: initialises the device
  Input: none
  Output: none
  Registers: all

INIFNK (003EH)   *1
  Function: initialises the contents of function keys
  Input: none
  Output: none
  Registers: all

* VDP access

DISSCR (0041H)   *1
  Function: inhibits the screen display
  Input: none
  Output: none
  Registers: AF, BC

ENASCR (0044H)   *1
  Function: displays the screen
  Input: none
  Output: none
  Registers: all

WRTVDP (0047H)   *2
  Function: writes data in the VDP register
  Input: C for the register number, B for data; the register number
            is 0 to 23 and 32 to 46
  Output: none
  Registers: AF, BC

RDVRM (004AH)   *1
  Function: reads the contents of VRAM. This is for TMS9918, so only the
            14 low order bits of the VRAM address are valid. To use all
            bits, call NRDVRM.
  Input: HL for VRAM address to be read
  Output: A for the value which was read
  Registers: AF

WRTVRM (004DH)   *1
  Function: writes data in VRAM. This is for TMS9918, so only the 14 low
            order bits of the VRAM address are valid. To use all bits,
            call NWRVRM.
  Input: HL for VRAM address, A for data
  Output: none
  Registers: AF

SETRD (0050H)   *1

```
  Function: sets VRAM address to VDP and enables it to be read. This is
            used to read data from the sequential VRAM area by using the
            address auto-increment function of VDP. This enables faster
            readout than using RDVRM in a loop. This is for TMS9918, so
            only the 14 low order bits of VRAM address are valid. To use
            all bits, call NSETRD.
  Input:    HL for VRAM address
  Output:   none
  Registers:      AF


SETWRT (0053H)          *1
  Function: sets VRAM address to VDP and enables it to be written. The
            purpose is the same as SETRD. This is for TMS9918, so only
            the 14 low order bits of VRAM address are valid. To use all
            bits, call NSETRD.
  Input:    HL for VRAM address
  Output:   none
  Registers:      AF


FILVRM (0056H)          *4
  Function: fills the specified VRAM area with the same data. This is for
            TMS9918, so only the 14 low order bits of the VRAM address
            are valid. To use all bits, see BIGFIL.
  Input:    HL for VRAM address to begin writing, BC for the length of
            the area to be written, A for data.
  Output:   none
  Registers:      AF, BC


LDIRMV (0059H)          *4
  Function: block transfer from VRAM to memory
  Input:    HL for source address (VRAM), DE for destination address
            (memory), BC for the length. All bits of the VRAM address
            are valid.
  Output:   none
  Registers:      all


LDIRVM (005CH)          *4
  Function: block transfer from memory to VRAM
  Input:    HL for source address (memory), DE for destination address
            (VRAM), BC for the length. All bits of the VRAM address are
            valid.
  Output:   none
  Registers:      all


CHGMOD (005FH)          *3
  Function: changes the screen mode. The palette is not initialised. To
            initialise it, see CHGMDP in SUB-ROM.
  Input:    A for the screen mode (0 to 8)
  Output:   none
  Registers:      all


CHGCLR (0062H)          *1
```

```
  Function: changes the screen colour
  Input:    A for the mode
               FORCLR (F3E9H) for foreground color
               BAKCLR (F3EAH) for background color
               BDRCLR (F3EBH) for border colour
  Output:   none
  Registers:      all


NMI (0066H)       *1
  Function: executes NMI (Non-Maskable Interrupt) handling routine
  Input:    none
  Output:   none
  Registers:      none


CLRSPR (0069H)          *3
  Function: initialises all sprites. The sprite pattern is cleared to
            null, the sprite number to the sprite plane number, the
            sprite colour to the foregtound colour. The vertical location
            of the sprite is set to 209 (mode 0 to 3) or 217
            (mode 4 to 8).
  Input:    SCRMOD (FCAFH) for the screen mode
  Output:   none
  Registers:      all


INITXT (006CH)          *3
  Function: initialises the screen to TEXT1 mode (40 x 24). In this
            routine, the palette is not initialised. To initialise the
            palette, call INIPLT in SUB-ROM after this call.
  Input:    TXTNAM (F3B3H) for the pattern name table
            TXTCGP (F3B7H) for the pattern generator table
            LINL40 (F3AEH) for the length of one line
  Output:   none
  Registers:      all


INIT32 (006FH)          *3
  Function: initialises the screen to GRAPHIC1 mode (32x24). In this
            routine, the palette is not initialised.
  Input:    T32NAM (F3BDH) for the pattern name table
            T32COL (F3BFH) for the colour table
            T32CGP (F3C1H) for the pattern generator table
            T32ATR (F3C3H) for the sprite attribute table
            T32PAT (F3C5H) for the sprite generator table
  Output:   none
  Registers:      all


INIGRP (0072H)          *3
  Function: initialises the screen to the high-resolution graphics mode.
            In this routine, the palette is not initialised.
  Input:    GRPNAM (F3C7H) for the pattern name table
            GRPCOL (F3C9H) for the colour table
            GRPCGP (F3CBH) for the pattern generator table
            GRPATR (F3CDH) for the sprite attribute table
            GRPPAT (F3CFH) for the sprite generator table
```

```
  Output:   none
  Registers:      all


INIMLT (0075H)          *3
  Function: initialises the screen to MULTI colour mode. In this routine,
            the palette is not initialised.
  Input:    MLTNAM (F3D1H) for the pattern name table
            MLTCOL (F3D3H) for the colour table
            MLTCGP (F3D5H) for the pattern generator table
            MLTATR (F3D7H) for the sprite attribute table
            MLTPAT (F3D9H) for the sprite generator table
  Output:   none
  Registers:      all


SETTXT (0078H)          *3
  Function: set only VDP in TEXT1 mode (40x24)
  Input:    same as INITXT
  Output:   none
  Registers:      all


SETT32 (007BH)          *3
  Function: set only VDP in GRAPHIC1 mode (32x24)
  Input:    same as INIT32
  Output:   none
  Registers:      all


SETGRP (007EH)          *3
  Function: set only VDP in GRAPHIC2 mode
  Input:    same as INIGRP
  Output:   none
  Registers:      all


SETMLT (0081H)          *3
  Function: set only VDP in MULTI colour mode
  Input:    same as INIMLT
  Output:   none
  Registers:      all


CALPAT (0084H)          *1
  Funtion:  returns the address of the sprite generator table
  Input:    A for the sprite number
  Output:   HL for the address
  Registers:      AF, DE, HL


CALATR (0087H)          *1
  Function: returns the address of the sprite attribute table
  Input:    A for the sprite number
  Output:   HL for the address
  Registers:      AF, DE, HL
```

GSPSIZ (008AH) *1
  Function: returns the current sprite size
  Input: none
  Output: A for the sprite size (in bytes). Only when the size is
           16 x 16, the CY flag is set; otherwise the CY flag is reset.
  Registers: AF

GRPPRT (008DH) *2
  Function: displays a character on the graphic screen
  Input: A for the character code. When the screen mode is 0 to 8,
           set the logical operation code in LOGOPR (FB02H).
  Output: none
  Registers: none

* PSG

GICINI (0090H) *1
  Function: initialises PSG and sets the initial value for the PLAY
           statement
  Input: none
  Output: none
  Registers: all

WRTPSG (0093H) *1
  Function: writes data in the PSG register
  Input: A for PSG register number, E for data
  Output: none
  Registers: none

RDPSG (0096H) *1
  Function: reads the PSG register value
  Input: A for PSG register number
  Output: A for the value which was read
  Registers: none

STRTMS (0099H) *1
  Function: tests whether the PLAY statement is being executed as a
           background task. If not, begins to execute the PLAY statement
  Input: none
  Output: none
  Registers: all

* Keyboard, CRT, printer input-output

CHSNS (009CH) *1
  Function: tests the status of the keyboard buffer
  Input: none
  Output: the Z flag is set when the buffer is empty, otherwise the

```
            Z flag is reset
  Registers:      AF


CHGET (009FH)           *1
  Function: one character input (waiting)
  Input:    none
  Output:   A for the code of the input character
  Registers:      AF


CHPUT (00A2H)           *1
  Function: displays the character
  Input:    A for the character code to be displayed
  Output:   none
  Registers:      none


LPTOUT (00A5H)          *1
  Function: sends one character to the printer
  Input:    A for the character code to be sent
  Output:   if failed, the CY flag is set
  Registers:      F


LPTSTT (00A8H)          *1
  Function: tests the printer status
  Input:    none
  Output:   when A is 255 and the Z flag is reset, the printer is READY.
            when A is 0 and the Z flag is set, the printer is NOT READY.
  Registers:      AF


CNVCHR (00ABH)          *1
  Function: test for the graphic header and transforms the code
  Input:    A for the character code
  Output:   the CY flag is reset to not the graphic header
            the CY flag and the Z flag are set to the transformed code
            is set in A
            the CY flag is set and the CY flag is reset to the
            untransformed code is set in A
  Registers:      AF


PINLIN (00AEH)          *1
  Function: stores in the specified buffer the character codes input
            until the return key or STOP key is pressed.
  Input:    none
  Output:   HL for the starting address of the buffer minus 1, the CY
            flag is set only when it ends with the STOP key.
  Registers:      all


INLIN (00B1H)           *1
  Function: same as PINLIN except that AUTFLG (F6AAH) is set
  Input:    none
  Output:   HL for the starting address of the buffer minus 1, the CY
            flag is set only when it ends with the STOP key.
```

```
  Registers:      all


QINLIN (00B4H)          *1
  Function: executes INLIN with displaying "?" and one space
  Input:    none
  Output:   HL for the starting address of the buffer minus 1, the CY
            flag is set only when it ends with the STOP key.
  Registers:      all


BREAKX (00B7H)          *1
  Function: tests Ctrl-STOP key. In this routine, interrupts are
            inhibited.
  Input:    none
  Output:   the CY flag is set when pressed
  Registers:      AF


BEEP (00C0H)            *3
  Function: generates BEEP
  Input:    none
  Output:   none
  Registers:      all


CLS (00C3H)       *3
  Function: clears the screen
  Input:    set zero flag
  Output:   none
  Registers:      AF, BC, DE


POSIT (00C6H)           *1
  Function: moves the cursor
  Input:    H for the X-coordinate of the cursor, L for the Y-coordinate
  Output:   none
  Registers:      AF


FNKSB (00C9H)           *1
  Function: tests whether the function key display is active (FNKFLG).
            If so, displays them, otherwise erases them.
  Input:    FNKFLG (FBCEH)
  Output:   none
  Registers:      all


ERAFNK (00CCH)          *1
  Function: erases the function key display
  Input:    none
  Output:   none
  Registers:      all


DSPFNK (00CFH)          *2
  Function: displays the function keys
  Input:    none
```

```
  Output:   none
  Registers:      all


TOTEXT (00D2H)          *1
  Function: forces the screen to be in the text mode
  Input:    none
  Output:   none
  Registers:      all



* Game I/O access


GTSTCK (00D5H)          *1
  Function: returns the joystick status
  Input:    A for the joystick number to be tested
  Output:   A for the joystick direction
  Registers:      all


GTTRIG (00D8H)          *1
  Function: returns the trigger button status
  Input:    A for the trigger button number to be tested
  Output:   When A is 0, the trigger button is not being pressed.
            When A is FFH, the trigger button is being pressed.
  Registers:      AF


GTPAD (00DBH)           *1
  Function: returns the touch pad status
  Input:    A for the touch pad number to be tested
  Output:   A for the value
  Registers:      all


GTPDL (00DEH)           *2
  Function: returns the paddle value
  Input:    A for the paddle number
  Output:   A for the value
  Registers:      all



* Cassette input-output routine


TAPION (00E1H)          *1
  Function: reads the header block after turning the cassette motor ON.
  Input:    none
  Output:   if failed, the CY flag is set
  Registers:      all


TAPIN (00E4H)           *1
  Function: reads data from the tape
  Input:    none
```

```
  Output:   A for data. If failed, the CY flag is set.
  Registers:      all


TAPIOF (00E7H)          *1
  Function: stops reading the tape
  Input:    none
  Output:   none
  Registers:      none


TAPOON (00EAH)          *1
  Function: writes the header block after turning the cassette motor ON
  Input:    A = 0, short header; A <> 0, long header
  Output:   if failed, the CY flag is set
  Registers:      all


TAPOUT (00EDH)          *1
  Function: writes data on the tape
  Input:    A for data
  Output:   if failed, the CY flag is set
  Registers:      all


TAPOOF (00F0H)          *1
  Function: stops writing to the tape
  Input:    A for data
  Output:   if failed, the CY flag is set
  Registers:      all


STMOTR (00F3H)          *1
  Function: sets the cassette motor action
  Input:    A = 0       ->    stop
            A = 1       ->    start
            A = 0FFH    ->    reverse the current action
  Output:   none
  Registers:      AF



* Miscellaneous


CHGCAP (0132H)          *1
  Function: alternates the CAP lamp status
  Input:    A = 0       ->    lamp off
            A <>0       ->    lamp on
  Output:   none
  Registers:      AF


CHGSND (0135H)          *1
  Function: alternates the 1-bit sound port status
  Input:    A = 0       ->    OFF
            A <>0       ->    ON
  Output:   none
```

```
  Registers:      AF


RSLREG (0138H)          *1
  Function: reads the contents of current output to the basic slot
            register
  Input:    none
  Output:   A for the value which was read
  Registers:      A


WSLREG (013BH)          *1
  Function: writes to the primary slot register
  Input:    A for the value to be written
  Output:   none
  Registers:      none


RDVDP (013EH)           *1
  Function: reads VDP status register
  Input:    none
  Output:   A for the value which was read
  Registers:      A


SNSMAT (0141H)          *1
  Function: reads the value of the specified line from the keyboard
            matrix
  Input:    A for the specified line
  Output:   A for data (the bit corresponding to the pressed key will
            be 0)
  Registers:      AF, C


PHYDIO (0144H)
  Function: Physical input/output for disk devices
  Input:    A for the drive number (0 = A:, 1 = B:,...)
            B for the number of sector to be read from or written to
            C for the media ID
            DE for the first sector number to be read rom or written to
            HL for the startinga address of the RAM buffer to be
               read from or written to specified sectors
            CY set for sector writing; reset for sector reading
  Output:   CY set if failed
            B for the number of sectors actually read or written
            A for the error code (only if CY set):
                  0 = Write protected
                  2 = Not ready
                  4 = Data error
                  6 = Seek error
                  8 = Record not found
                   10 = Write error
                   12 = Bad parameter
                   14 = Out of memory
                   16 = Other error
  Registers:      all
```

ISFLIO (014AH) *1
  Function: tests whether the device is active
  Input: none
  Output: A = 0 -> active
           A <>0 -> inactive
  Registers: AF

OUTDLP (014DH) *1
  Function: printer output.Different from LPTOUT in the following points:
            1. TAB is expanded to spaces
            2. For non-MSX printers, hiragana is transformed to
             katakana and graphic characters are transformed to
             1-byte characters.
            3. If failed, device I/O error occurs.
  Input: A for data
  Output: none
  Registers: F

KILBUF (0156H) *1
  Function: clears the keyboard buffer
  Input: none
  Output: none
  Registers: HL

CALBAS (0159H) *1
  Function: executes inter-slot call to the routine in BASIC interpreter
  Input: IX for the calling address
  Output: depends on the called routine
  Registers: depends on the called routine

* Entries appended for MSX2

SUBROM (015CH)
  Function: executes inter-slot call to SUB-ROM
  Input: IX for the calling address and, at the same time, pushes IX
          on the stack
  Output: depends on the called routine
  Registers: background registers and IY are reserved

EXTROM (015FH)
  Function: executes inter-slot call to SUB-ROM
  Input: IX for the calling address
  Output: depends on the called routine
  Registers: background registers and IY are reserved

EOL (0168H)
  Function: deletes to the end of the line
  Input: H for X-coordinate of the cursor, L for Y-coordinate
  Output: none
  Registers: all

BIGFIL (016BH)
  Function: same function as FILVRM. Differences are as follows:
              In FILVRM, it is tested whether the screen mode is 0 to 3.
              If so, it treats VDP as the one which has only 16K bytes
              VRAM (for the compatibility with MSX1). In BIGFIL, the
              mode is not tested and actions are carried out by the
              given parameters.
  Input: same as FILVRM
  Output: same as FILVRM
  Registers: same as FILVRM

NSETRD (016EH)
  Function: enables VRAM to be read by setting the address
  Input: HL for VRAM address
  Output: none
  Registers: AF

NSTWRT (0171H)
  Function: enables VRAM to be written by setting the address
  Input: HL for VRAM address
  Output: none
  Registers: AF

NRDVRM (0174H)
  Function: reads the contents of VRAM
  Input: HL for VRAM address to be read
  Output: A for the value which was read
  Registers: F

NWRVRM (0177H)
  Function: writes data in VRAM
  Input: HL for VRAM address, A for data
  Output: none
  Registers: AF

SUB-ROM

The calling sequence of SUB-ROM is as follows:

```
            .
            .
            .
      LD    IX, INIPLT
                  ; Set BIOS entry address
      CALL  EXTROM
                  ; Returns here
            .
            .
            .
```

When the contents of IX should not be destroyed, use the call as shown below.

```
            .
            .
            .
INIPAL: PUSH     IX
                  ; Save IX
      LD    IX, INIPLT
                  ; Set BIOS entry address
      JP    SUBROM
                 ;Returns caller of INIPAL
            .
            .
            .
```

```
GRPRT (0089H)
  Function: one character output to the graphic screen (active only in
            screen modes 5 to 8)
  Input:    A for the character code
  Output:   none
  Registers:      none
```

```
NVBXLN (00C9H)
  Function: draws a box
  Input:    start point: BC for X-coordinate, DE for Y-coordinate
            end point:   GXPOS (FCB3H) for X-coordinate
                         GYPOS (FCB5H) for Y-coordinate
            colour:      ATRBYT (F3F3H) for the attribute
            logical operation code: LOGOPR (FB02H)
  Output:   none
  Registers:      all
```

```
NVBXFL (00CDH)
  Function: draws a painted box
  Input:    start point: BC for X-coordinate, DE for Y-coordinate
            end point:   GXPOS (FCB3H) for X-coordinate
                         GYPOS (FCB5H) for Y-coordinate
            colour:      ATRBYT (F3F3H) for the attribute
            logical operation code: LOGOPR (FB02H)
  Output:   none
  Registers:      all
```

```
CHGMOD (00D1H)
  Function: changes the screen mode
  Input:    A for the screen mode (0 to 8)
  Output:   none
  Registers:      all
```

```
INITXT (00D5H)
  Function: initialises the screen to TEXT1 mode (40 x 24)
  Input:    TXTNAM (F3B3H) for the pattern name table
            TXTCGP (F3B7H) for the pattern generator table
```

```
            LINL40 (F3AEH) for the length of one line
  Output:   none
  Registers:      all


INIT32 (00D9H)
  Function: initialises the screen to GRAPHIC1 mode (32x24)
  Input:    T32NAM (F3BDH) for the pattern name table
            T32COL (F3BFH) for the colour table
            T32CGP (F3C1H) for the pattern generator table
            T32ATR (F3C3H) for the sprite attribute table
            T32PAT (F3C5H) for the sprite generator table
  Output:   none
  Registers:      all


INIGRP (00DDH)
  Function: initialises the screen to the high-resolution graphics mode
  Input:    GRPNAM (F3C7H) for the pattern name table
            GRPCOL (F3C9H) for the colour table
            GRPCGP (F3CBH) for the pattern generator table
            GRPATR (F3CDH) for the sprite attribute table
            GRPPAT (F3CFH) for the sprite generator table
  Output:   none
  Registers:      all


INIMLT (00E1H)
  Function: initialises the screen to MULTI colour mode
  Input:    MLTNAM (F3D1H) for the pattern name table
            MLTCOL (F3D3H) for the colour table
            MLTCGP (F3D5H) for the pattern generator table
            MLTATR (F3D7H) for the sprite attribute table
            MLTPAT (F3D9H) for the sprite generator table
  Output:   none
  Registers:      all


SETTXT (00E5H)
  Function: sets VDP in the text mode (40x24)
  Input:    same as INITXT
  Output:   none
  Registers:      all


SETT32 (00E9H)
  Function: ses VDP in the text mode (32x24)
  Input:    same as INIT32
  Output:   none
  Registers:      all


SETGRP (00EDH)
  Function: sets VDP in the high-resolution mode
  Input:    same as INIGRP
  Output:   none
  Registers:      all
```

SETMLT (00F1H)
  Function: sets VDP in MULTI COLOUR mode
  Input: same as INIMLT
  Output: none
  Registers: all

CLRSPR (00F5H)
  Function: initialises all sprites. The sprite pattern is set to null,
            sprite number to sprite plane number, and sprite colour to
            the foregtound colour. The vertical location of the sprite
            is set to 217.
  Input: SCRMOD (FCAFH) for the screen mode
  Output: none
  Registers: all

CALPAT (00F9H)
  Funtion: returns the address of the sprite generator table
            (this routine is the same as CALPAT in MAIN-ROM)
  Input: A for the sprite number
  Output: HL for the address
  Registers: AF, DE, HL

CALATR (00FDH)
  Function: returns the address of the sprite attribute table
            (this routine is the same as CALATR in MAIN-ROM)
  Input: A for the sprite number
  Output: HL for the address
  Registers: AF, DE, HL

GSPSIZ (0101H)
  Function: returns the current sprite size
            (this routine is the same as GSPSIZ in MAIN-ROM)
  Input: none
  Output: A for the sprite size. The CY flag is set only for the size
            16 x 16.
  Registers: AF

GETPAT (0105H)
  Function: returns the character pattern
  Input: A for the character code
  Output: PATWRK (FC40H) for the character pattern
  Registers: all

WRTVRM (0109H)
  Function: writes data in VRAM
  Input: HL for VRAM address (0 TO FFFFH), A for data
  Output: none
  Registers: AF

RDVRM (010DH)

```
  Function: reads the contents of VRAM
  Input:    HL for VRAM address (0 TO FFFFH) to be read
  Output:   A for the value which was read
  Registers:      AF


CHGCLR (0111H)
  Function: changes the screen colour
  Input:    A for the mode
            FORCLR (F3E9H) for the foreground color
            BAKCLR (F3EAH) for the background color
            BDRCLR (F3EBH) for the border colour
  Output:   none
  Registers:      all


CLSSUB (0115H)
  Function: clears the screen
  Input:    none
  Output:   none
  Registers:      all


DSPFNK (011DH)
  Function: displays the function keys
  Input:    none
  Output:   none
  Registers:      all


WRTVDP (012DH)
  Function: writes data in the VDP register
  Input:    C for the register number, B for data
  Output:   none
  Registers:      AF, BC


VDPSTA (0131H)
  Function: reads the VDP register
  Input:    A for the register number (0 to 9)
  Output:   A for data
  Registers:      F


SETPAG (013DH)
  Function: switches the page
  Input:    DPPAGE (FAF5H) for the display page number
            ACPAGE (FAF6H) for the active page number
  Output:   none
  Registers:      AF


INIPLT (0141H)
  Function: initialises the palette(the current palette is saved in VRAM)
  Input:    none
  Output:   none
  Registers:      AF, BC, DE
```

```
RSTPLT (0145H)
  Function: restores the palette from VRAM
  Input:    none
  Output:   none
  Registers:      AF, BC, DE


GETPLT (0149H)
  Function: obtains the colour code from the palette
  Input:    D for the palette number (0 to 15)
  Output:   4 high order bits of B for red code
            4 low order bits of B for blue code
            4 low order bits of C for green code
  Registers:      AF, DE


SETPLT (014DH)
  Function: sets the colour code to the palette
  Input:    D for the palette number (0 to 15)
            4 high order bits of A for red code
            4 low order bits of A for blue code
            4 low order bits of E for green code
  Output:   none
  Registers:      AF


BEEP (017DH)
  Function: generates BEEP
  Input:    none
  Output:   none
  Registers:      all


PROMPT (0181H)
  Function: displays the prompt
  Input:    none
  Output:   none
  Registers:      all


NEWPAD (01ADH)
  Function: reads the status of mouse or light pen
  Input:    call with setting the following data in A;
            descriptions in parenthesis are return values.
               8 .......... light pen check (valid at 0FFH)
               9 .......... returns X-coordinate
              10 .......... returns Y-coordinate
              11 .......... returns the light pen switch status
                        (0FFH, when pressed)
              12 .......... whether the mouse is connected to the
                        port 1 (valid at 0FFH)
              13 .......... returns the offset in X direction
              14 .......... returns the offset in Y direction
              15 .......... (always 0)
              16 .......... whether the mouse is connected to the
                        port 2 (valid at 0FFH)
              17 .......... returns the offset in X direction
```

```
              18 .......... returns the offset in Y direction
              19 .......... (always 0)
  Output:   A
  Registers:      all


CHGMDP (01B5H)
  Function: changes VDP mode. The palette is initialised.
  Input:    A for the screen mode (0 to 8)
  Output:   none
  Registers:      all


KNJPRT (01BDH)
  Function: sends a kanki to the graphic screen (modes 5 to 8)
  Input:    BC for JIS kanji code, A for the display mode. The display
            mode has the following meaning, similar to the PUT KANJI
            command of BASIC.
               0 .......... display in 16 x 16 dot
               1 .......... display even dots
               2 .......... display odd dots


REDCLK (01F5H)
  Function: reads the clock data
  Input:    C for RAM address of the clock

              00MMAAAA
                ------
                ||++++--- Address (0 to 15)
                ++------- Mode (0 to 3)
  Output:   A for the data which were read (only 4 low order bits are
            valid)
  Registers:      F


WRTCLK (01F9H)
  Function: writes the clock data
  Input:    A for the data to be written, C for RAM address of the clock
  Output:   none
  Registers:      F


================================================================================

Changes from the original in APPENDIX 2:

- In the explanation before Figure A.3, the indication about the excess 64
method has been added.

- In Figure A.3, in the third byte, "63rd power of 10" has been corrected to
"-63rd power of 10".

- In the explanation before Figure A.3, the indication about the excess 64
method has been added.

- In Figure A.3, in the third byte, "63rd power of 10" has been corrected to
"-63rd power of 10".
```

-=-

APPENDIX 2 - MATH-PACK

The Math-Pack is the core for the mathematical routines of MSX-BASIC and, by calling these routines from an assembly language program, floating-point operations and trigonometrical functions are available.

Any operations involving real numbers in Math-Pack are done in BCD (Binary Coded Decimal). There are two ways of expressing a real number, "single precision" and "double precision"; a single precision real number (6 digits) is expressed by 4 bytes and a double precision real number (14 digits) by 8 bytes (see Figure A.1 and Figure A.2).

Figure A.1 BCD format for expressing real numbers

```
                 MSB   7       6     5     4      3   2     1     0     LSB
  ---        ---  -------------------------------------------------
   ^        ^     |sign |           exponent        |  0
   |        |     -------------------------------------------------
   |        |     -------------------------------------------------
   |        |     |  mantissa 1st place   |  mantissa 2nd place   |  1
   |       single       -------------------------------------------------
   |      precision     -------------------------------------------------
   |        |     |  mantissa 3rd place   |  mantissa 4th place   |  2
   |        |     -------------------------------------------------
   |        |     -------------------------------------------------
   |        V     |  mantissa 5th place   |  mantissa 6th place   |  3
 double        ---      -------------------------------------------------
precision         -------------------------------------------------
   |              |  mantissa 7th place   |  mantissa 8th place   |  4
   |              -------------------------------------------------
   |              -------------------------------------------------
   |              |  mantissa 9th place   |  mantissa 10th place  |  5
   |              -------------------------------------------------
   |              -------------------------------------------------
   |              |  mantissa 11th place  |  mantissa 12th place  |  6
   |              -------------------------------------------------
   |              -------------------------------------------------
   V              |  mantissa 13th place  |  mantissa 14th place  |  7
  ---             -------------------------------------------------
```

Figure A.2 Examples of expressions for real numbers

Example of the single precision expression

```
       123456 --> 0.123456 E+6

                   1     2       3   4
                ------------------------
            DAC |  46 |  12 |  34 |  56 |
                ------------------------
```

Example of the double precision expression

123456.78901234 --> 0.12345678901234 E+6

	1	2	3	4	5	6	7	8
DAC	46	12	34	56	78	90	12	34

A real number consists of a sign, an exponent, and a mantissa. The sign represents the sign of the mantissa; 0 for positive, 1 for negative. The exponent is a binary expression and can be expressed as a power from +63 to -63, with an excess of 64 (see Figure A.3). Figure A.4 shows the valid range of double precision real numbers.

Figure A.3 Exponent format

sign	exponent							meaning
0	0	0	0	0	0	0	0	 0
1	0	0	0	0	0	0	0	..... undefined (-0?)
x	0	0	0	0	0	0	1	..... -63rd power of 10
x	1	0	0	0	0	0	0	..... 0th power of 10
x	1	1	1	1	1	1	1	..... +63rd power of 10

Note: "x" is 1 or 0, both of which are allowed.

Figure A.4 Valid range for double precision real numbers

	7	6	5	4	3	2	1	0	(byte)
DAC	FF	99	99	99	99	99	99	99	-0.99999999999999 E+63
				.					
				.					
				.					
	81	10	00	00	00	00	00	00	-0.10000000000000 E-63
	00	x	x	x	x	x	x	x	0
	01	10	00	00	00	00	00	00	+0.10000000000000 E-63
				.					
				.					

```
                  .
    ------------------------------------------------
    |  7F |  99 |  99 |  99 |  99 |  99 |  99 |  +0.99999999999999 E+63
    ------------------------------------------------
```

In Math-Pack, the memory is predefined for operation. This memory area is called "DAC (Decimal ACumulator (F7F6H)" and the area which reserves the numerical value to be operated is called "ARG (F847H)". For example, in multiplication, the product of the numbers in DAC and ARG is calculated and the result is returned in the DAC.

In the DAC, single precision real numbers, double precision real numbers, and two-byte integers can be stored. In order to distinguish them, "VALTYP (F663H)" is used and its value is 4 for single precision real numbers, 8 for double precision real numbers, and 2 for two-byte integers.

Single and double precision numbers must be stored from the top of the DAC. For two-byte integers, the low and high bytes should be stored in DAC + 2 and DAC + 3.

Since Math-Pack is an internal routine of BASIC, when an error occurs (such as division by 0 or overflow), control automatically jumps to the corresponding error routine, then returns to BASIC command level. To prevent this, change H.ERRO (FFB1H).

* Math-Pack work area

Label	Address	Size	Meaning
VALTYP	F663H	1	format of the number in DAC
DAC	F7F6H	16	floating point accumulator in BCD format
ARG	F847H	16	argument of DAC

* Math-Pack entry

Basic operation

Label	Address	Function
DECSUB	268CH	DAC <-- DAC - ARG
DECADD	269AH	DAC <-- DAC + ARG
DECNRM	26FAH	normalises DAC (*1)
DECROU	273CH	rounds DAC
DECMUL	27E6H	DAC <-- DAC * ARG
DECDIV	289FH	DAC <-- DAC / ARG

Note: These operations treat numbers in DAC and ARG as the double precision number. Registers are not preserved.
*1 Excessive zeros in mantissa are removed. (0.00123 --> 0.123 E-2)

Function 1

Label	Address	Function	Register modified
COS	2993H	DAC <-- COS(DAC)	all
SIN	29ACH	DAC <-- SIN(DAC)	all
TAN	29FBH	DAC <-- TAN(DAC)	all
ATN	2A14H	DAC <-- ATN(DAC)	all
LOG	2A72H	DAC <-- LOG(DAC)	all
SQR	2AFFH	DAC <-- SQR(DAC)	all
EXP	2B4AH	DAC <-- EXP(DAC)	all
RND	2BDFH	DAC <-- RND(DAC)	all

Note: These processing routines all have the same function names as those in BASIC. "All" registers are A, B, C, D, E, H, and L.

Function 2

Label	Address	Function	Register modified
SIGN	2E71H	A <-- sign of DAC	A
ABSFN	2E82H	DAC <-- ABS(DAC)	all
NEG	2E8DH	DAC <-- NEG(DAC)	A,HL
SGN	2E97H	DAC <-- SGN(DAC)	A,HL

Note: Except for SIGN, these processing routines all have the same function names as those in BASIC. Registers are A, B, C, D, E, H, and L. Note that for SGN, the result is represented as a 2-byte integer.

Movement

Label	Address	Function	Object	Reg. mod.
MAF	2C4DH	ARG <-- DAC	double prec.	A,B,D,E,H,L
MAM	2C50H	ARG <-- (HL)	double prec.	A,B,D,E,H,L
MOV8DH	2C53H	(DE) <-- (HL)	double prec.	A,B,D,E,H,L
MFA	2C59H	DAC <-- ARG	double prec.	A,B,D,E,H,L
MFM	2C5CH	DAC <-- (HL)	double prec.	A,B,D,E,H,L
MMF	2C67H	(HL) <-- DAC	double prec.	A,B,D,E,H,L
MOV8HD	2C6AH	(HL) <-- (DE)	double prec.	A,B,D,E,H,L
XTF	2C6FH	(SP) <--> DAC	double prec.	A,B,D,E,H,L
PHA	2CC7H	ARG <-- (SP)	double prec.	A,B,D,E,H,L
PHF	2CCCH	DAC <-- (SP)	double prec.	A,B,D,E,H,L
PPA	2CDCH	(SP) <-- ARG	double prec.	A,B,D,E,H,L
PPF	2CE1H	(SP) <-- DAC	double prec.	A,B,D,E,H,L
PUSHF	2EB1H	DAC <-- (SP)	single prec.	D,E
MOVFM	2EBEH	DAC <-- (HL)	single prec.	B,C,D,E,H,L
MOVFR	2EC1H	DAC <-- (CBED)	single prec.	D,E
MOVRF	2ECCH	(CBED) <-- DAC	single prec.	B,C,D,E,H,L
MOVRMI	2ED6H	(CBED) <-- (HL)	single prec.	B,C,D,E,H,L

MOVRM	2EDFH	(BCDE) <-- (HL)	single prec.	B,C,D,E,H,L
MOVMF	2EE8H	(HL) <-- DAC	single prec.	A,B,D,E,H,L
MOVE	2EEBH	(HL) <-- (DE)	single prec.	B,C,D,E,H,L
VMOVAM	2EEFH	ARG <-- (HL)	VALTYP	B,C,D,E,H,L
MOVVFM	2EF2H	(DE) <-- (HL)	VALTYP	B,C,D,E,H,L
VMOVE	2EF3H	(HL) <-- (DE)	VALTYP	B,C,D,E,H,L
VMOVFA	2F05H	DAC <-- ARG	VALTYP	B,C,D,E,H,L
VMOVFM	2F08H	DAC <-- (HL)	VALTYP	B,C,D,E,H,L
VMOVAF	2F0DH	ARG <-- DAC	VALTYP	B,C,D,E,H,L
VMOVMF	2F10H	(HL) <-- DAC	VALTYP	B,C,D,E,H,L

Note: (HL), (DE) means the values in memory pointed to by HL or DE. Four register names in the parentheses are the single precision real numbers which indicate (sign + exponent), (mantissa 1st and 2nd places), (mantissa 3th and 4th places), (mantissa 5th and 6th places) from left to right. Where the object is VALTYP, the movement (2, 4, 8 bytes) is according to the type indicated in VALTYP (F663H).

Comparison

Label	Address	Object	Left	Right	Reg. mod.
FCOMP	2F21H	single prec. real number	CBED	DAC	HL
ICOMP	2F4DH	2-byte integer	DE	HL	HL
XDCOMP	2F5CH	double prec. real number	ARG	DAC	all

Note: Results will be in A register. Meanings of A register are:

A = 1 --> left < right
A = 0 --> left = right
A = -1 --> left > right

In the comparison of single precision real numbers, CBED means that each register has single precision (sign + exponent), (mantissa 1st and 2nd places), (mantissa 3th and 4th places), and (mantissa 5th and 6th places).

Floating-point input/output

Label	Address	Function
FIN	3299H	Stores a string representing the floating-point number in DAC, converting it in real.
Entry condition	HL <--	Starting address of the string
	A <--	First character of the string
Return condition	DAC <--	Real number
	C <--	FFH: without a decimal point
		0: with a decimal point
	B <--	Number of places after the decimal point
	D <--	Number of digits

Label	Address	Function
FOUT	3425H	Converts the real number in DAC to the string (unformatted)
PUFOUT	3426H	Converts the real number in DAC to the string (formatted)

Entry condition A <-- format
bit 7 0: unformatted 1: formatted
bit 6 0: without commas 1: with commas every three digits
bit 5 0: meaningless 1: leading spaces are padded with "."
bit 4 0: meaningless 1: "$" is added before the numerical value
bit 3 0: meaningless 1: "+" is added even for positive values
bit 2 0: meaningless 1: the sign comes after the value
bit 1 unused
bit 0: 0: fixed point 1: floating-point
B <-- number of digits before and not including the decimal point
C <-- number of digits after and including the decimal point
Return condition HL <-- starting address of the string

Label	Address	Function
FOUTB	371AH	Converts 2-byte integer in DAC+2, 3 to a binary expression string.
FOUTO	371EH	Converts 2-byte integer in DAC+2, 3 to an octal expression string.
FOUTH	3722H	Converts 2-byte integer in DAC+2, 3 to a hexadecimal expression string.

Entry condition DAC + 2 <-- 2-byte integer
VALTYP <-- 2
Return condition HL <-- starting address of the string

Note: no strings are reserved. The starting address of the string in the output routine is normally in FBUFFR (from F7C5H). In some cases it may differ slightly. For the integer in DAC + 2, VALTYP (F663H) must be 2, even in cases other than FOUTB, FOUTO and FOUTH.

Type conversion

Label	Address	Function
FRCINT	2F8AH	Converts DAC to a 2-byte integer (DAC + 2, 3)
FRCSNG	2FB2H	Converts DAC to a single precision real number
FRCDBL	303AH	Converts DAC to a double precision real number
FIXER	30BEH	DAC <-- SGN(DAC) * INT(ABS(DAC))

Note: after execution, VALTYP (F663H) will contain the number (2, 4 or 8) representing DAC type. No registers are reserved.

Integer operation

Label	Address	Function	Registers modified
UMULT	314AH	DE <-- BC * DE	A, B, C, D, E
ISUB	3167H	HL <-- DE - HL	all
IADD	3172H	HL <-- DE + HL	all
IMULT	3193H	HL <-- DE * HL	all
IDIV	31E6H	HL <-- DE / HL	all
IMOD	323AH	HL <-- DE mod HL	alle
		(DE <-- DE/HL)	

Power

Label	Address	Function	Base	Exp.	Result
SGNEXP	37C8H	power of single-prec. real	DAC	ARG	DAC
DBLEXP	37D7H	power of double-prec. real	DAC	ARG	DAC
INTEXP	383FH	power of 2-byte integer	DE	HL	DAC

Note: No registers are reserved.

===

Changes from the original in APPENDIX 3:

none

-=-

APPENDIX 3 - BIT BLOCK TRANSFER

The bit block transfer corresponds to the COPY command in BASIC and is used to transfer data from RAM, VRAM, and the disk. It is easily executed by the routine in expansion ROM and available from the assembly language program. Since it is in expansion ROM, use SUBROM or EXTROM of BIOS for this routine.

1. Transferring in VRAM

* BLTVV (0191H/SUB)

  Function: transfers data in VRAM area

  Input: HL register <-- F562H
  The following parameters should be set:

  * SX (F562H, 2)   X-coordinate of the source
  * SY (F564H, 2)   Y-coordinate of the source
  * DX (F566H, 2)   X-coordinate of the destination

```
            * DY (F568H, 2)   Y-coordinate of the destination
            * NX (F56AH, 2)   number of dots in the X direction
            * NY (F56CH, 2)   number of dots in the Y direction
            * CDUMMY (F56EH, 1)     dummy (not required to be set)
            * ARG (F56FH, 1)  selects the direction and expansion
                              RAM (same as VDP R#45)
            * LOGOP (F570H, 1)      logical operation code (same as the
                              logical operation code of VDP)

  Output:   the CY flag is reset

  Registers:      all
```

2. Transferring data between RAM and VRAM

To use the routines below, the following memory space should be allocated as graphic area for screen modes.

* screen mode 6
  number of dots in X direction times number of dots in Y direction/4 + 4

* screen mode 5 or 7
  number of dots in X direction times number of dots in Y direction/2 + 4

* screen mode 8
  number of dots in X direction times number of dots in Y direction/2 + 4

Note to raise fractions.

For disk or RAM, data to indicate the size is added as the array data. The first two bytes of data indicate the number of dots in X direction; the next two bytes indicate the number of dots in the Y direction.

* BLTVM (0195H/SUB)

```
  Function: transfers the array to VRAM

  Input:    HL register <-- F562H
            The following parameters should be set:

            * DPTR (F562H, 2) source address of memory
            * DUMMY (F564H, 2)      dummy (not required to be set)
            * DX (F566H, 2)   X-coordinate of the destination
            * DY (F568H, 2)   Y-coordinate of the destination
            * NX (F56AH, 2)   number of dots in the X direction
                              (not required to be set; this is
                              already in the top of data to be
                              transferred)
            * NY (F56CH, 2)   number of dots in the Y direction
                              (not required to be set; this is
                              already in the top of data to be
                              transferred)
            * CDUMMY (F56EH, 1)     dummy (not required to be set)
            * ARG (F56FH, 1)  selects the direction and expansion
                              RAM (same as VDP R#45)
```

```
            * LOGOP (F570H, 1)      logical operation code (same as the
                                    logical operation code of VDP)

  Output:   the CY flag is set when the number of data bytes to be
            transferred is incorrect

  Registers:      all


* BLTMV (0199H/SUB)

  Function: transfers to the array from VRAM

  Input:    HL register <-- F562H
            The following parameters should be set:

            * SX (F562H, 2)   X-coordinate of the source
            * SY (F564H, 2)   Y-coordinate of the source
            * DPTR (F566H, 2) destination address of memory
            * DUMMY (F568H, 2)      dummy (not required to be set)
            * NX (F56AH, 2)   number of dots in the X direction
            * NY (F56CH, 2)   number of dots in the Y direction
            * CDUMMY (F56EH, 1)     dummy (not required to be set)
            * ARG (F56FH, 1)  selects the direction and expansion
                              RAM (same as VDP R#45)

  Output:   the CY flag is reset

  Registers:      all


3. Transferring between the disk and RAM or VRAM

The filename should be set first to use the disk (specify the filename as
BASIC). The following is an example:

            .
            .
            .
      LD    HL,FNAME          ; Get pointer to file name
      LD    (FNPTR),HL        ; Set it to parameter area
            .
            .
            .
FNAME:      DB 22H,"B:TEST.PIC",22H,0       ; "TEST.PIC", end mark

When an error occurs, control jumps to the error handler of the BASIC
interpreter. Set the hook to handle the error in the user program or to call
this routine from MSX-DOS or a ROM cartridge. This hook is H.ERRO (FFB1H).


* BLTVD (019DH/SUB)

  Function: transfers from disk to VRAM

  Input:    HL register <-- F562H
            The following parameters should be set:
```

```
            * FNPTR (F562H, 2)      address of the filename
            * DUMMY (F564H, 2)      dummy (not required to be set)
            * DX (F566H, 2)   X-coordinate of the destination
            * DY (F568H, 2)   Y-coordinate of the destination
            * NX (F56AH, 2)   number of dots in the X direction
                              (not required to be set; this is
                              already in the top of data to be
                              transferred)
            * NY (F56CH, 2)   number of dots in the Y direction
                              (not required to be set; this is
                              already in the top of data to be
                              transferred)
            * CDUMMY (F56EH, 1)     dummy (not required to be set)
            * ARG (F56FH, 1)  selects the direction and expansion
                              RAM (same as VDP R#45)
            * LOGOP (F570H, 1)      logical operation code (same as the
                              logical operation code of VDP)

  Output:   the CY flag is set when there is an error in the parameter

  Registers:      all


* BLTDV (01A1H/SUB)

  Function: transfers from VRAM to disk

  Input:    HL register <-- F562H
            The following parameters should be set:

            * SX (F562H, 2)   X-coordinate of the source
            * SY (F564H, 2)   Y-coordinate of the source
            * FNPTR (F566H, 2)      address of the filename
            * DUMMY (F568H, 2)      dummy (not required to be set)
            * NX (F56AH, 2)   number of dots in the X direction
            * NY (F56CH, 2)   number of dots in the Y direction
            * CDUMMY (F56EH, 1)     dummy (not required to be set)

  Output:   the CY flag is reset

  Registers:      all


* BLTMD (01A5H/SUB)

  Function: loads array data from disk

  Input:    HL register <-- F562H
            The following parameters should be set:

            * FNPTR (F562H, 2)      address of the filename
            * SY (F564H, 2)   dummy (not required to be set)
            * SPTR (F566H, 2) the starting address for loading
            * EPTR (F568H, 2) the end address for loading

  Output:   the CY flag is reset

  Registers:      all
```

* BLTDM (01A9H/SUB)

  Function: saves array data to disk

  Input: HL register <-- F562H
            The following parameters should be set:

            * SPTR (F562H, 2) the starting address for saving
            * EPTR (F564H, 2) the end address for saving
            * FNPTR (F566H, 2)   address of the filename

  Output: the CY flag is reset

  Registers: all

==========

Changes from the original in APPENDIX 4:

- Address of FLAGS variable is corrected from FB1BH to FB1CH.

- Address of MCLLEN variable is corrected from FB39H to FB3BH.

- Address of H.FIEL hook is corrected from DE2BH to FE2BH.

-=-

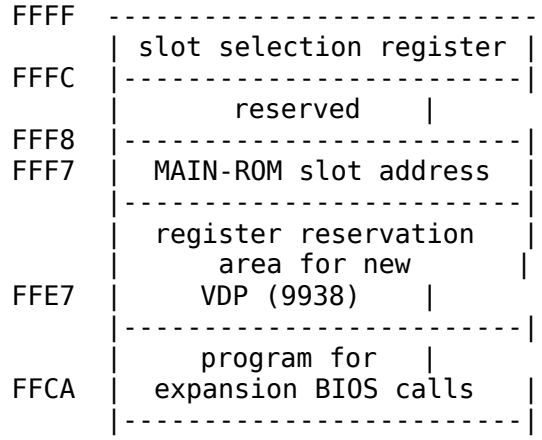
APPENDIX 4 - WORK AREA LISTING

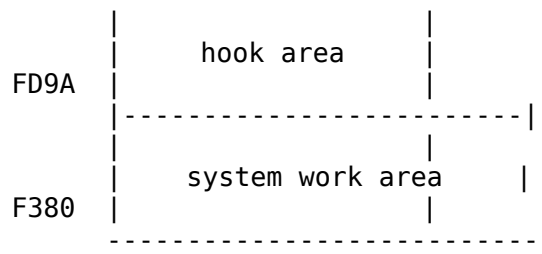
Figure A.5 shows the map of the MSX2 work area. In this section, the system work area and hook from F380H to FFCAH in the figure are described. The following notation is used. Length is in bytes.

      Label name (address, length)

            Initial value, contents, purpose

   Figure A.5 Work area





* Subroutines for read/write calls of the inter-slot

RDPRIM (F380H, 5)
  contents: read from basic slot

WRPRIM (F385H, 7)
  contents: write to basic slot

CLPRIM (F38CH, 14)
  contents: basic slot call

* Starting address of assembly language program of USR function, text screen

USRTAB (F39AH, 20)
  initial value: FCERR
  contents: starting address of assembly language program of USR function
            (0 to 9); the value before defining assembly language program
            points to FCERR (475AH).

LINL40 (F3AEH, 1)
  initial value: 39
  contents: screen width per line at SCREEN 0 (set by WIDTH statement
            at SCREEN 0)

LINL32 (F3AfH, 1)
  initial value: 32
  contents: screen width per line at SCREEN 1 (set by WIDTH statement
            at SCREEN 1)

LINLEN (F3B0H, 1)
  initial value: 29
  contents: current screen width per line

CRTCNT (F3B1H, 1)
  initial value: 24
  contents: number of lines of current screen

CLMLST (F3B2H, 1)
  initial value: 14
  contents: horizontal location in the case that items are divided by
            commas in PRINT statement

* Work for initialisation

- SCREEN 0

TXTNAM (F3B3H, 2)
  initial value: 0000H
  contents: pattern name table

TXTCOL (F3B5H, 2)
  contents: unused

TXTCGP (F3B7H, 2)
  initial value: 0800H
  contents: pattern generator table

TXTATR (F3B9H, 2)
  contents: unused

TXTPAT (F3BBH, 2)
  contents: unused

- SCREEN 1

T32NAM (F3BDH, 2)
  initial value: 1800H
  contents: pattern name table

T32COL (F3BFH, 2)
  initial value: 2000H
  contents: colour table

T32CGP (F3C1H, 2)
  initial value: 0000H
  contents: pattern generator table

T32ATR (F3C3H, 2)
  initial value: 1B00H
  contents: sprite attribute table

T32PAT (F3C5H, 2)
  initial value: 3800H
  contents: sprite generator table

- SCREEN 2

GRPNAM (F3C7H, 2)
  initial value: 1800H
  contents: pattern name table

GRPCOL (F3C9H, 2)
  initial value: 2000H
  contents: colour table

GRPCGP (F3CBH, 2)
  initial value: 0000H
  contents: pattern generator table

GRPATR (F3CDH, 2)
  initial value: 1B00H
  contents: sprite attribute table

GRPPAT (F3CFH, 2)
  initial value: 3800H
  contents: sprite generator table

- SCREEN 3

MLTNAM (F3D1H, 2)
  initial value: 0800H
  contents: pattern name table

MLTCOL (F3D3H, 2)
  contents: unused

MLTCGP (F3D5H, 2)
  initial value: 0000H
  contents: pattern generator table

MLTATR (F3D7H, 2)
  initial value: 1B00H
  contents: sprite attribute table

MLTPAT (F3D9H, 2)
  initial value: 3800H

```
  contents: sprite generator table



* Other screen settings


CLIKSW (F3DBH, 1)
  initial value:  1
  contents: key click switch (0 = OFF, otherwise = ON), set by
            <key click switch> of SCREEN statement


CSRY (F3DCH, 1)
  initial value:  1
  contents: Y-coordinate of cursor


CSRX (F3DDH, 1)
  initial value:  1
  contents: X-coordinate of cursor


CNSDFG (F3DEH, 1)
  initial value:  0
  contents: function key display switch (0 = display, otherwise = no
            display), set by KEY ON/OFF statement



* Area to save VDP registers


RG0SAV (F3DFH, 1)
  initial value:  0


RG1SAV (F3E0H, 1)
  initial value:  E0H


RG2SAV (F3E1H, 1)
  initial value:  0


RG3SAV (F3E2H, 1)
  initial value:  0


RG4SAV (F3E3H, 1)
  initial value:  0


RG5SAV (F3E4H, 1)
  initial value:  0


RG6SAV (F3E5H, 1)
```

  initial value: 0

RG7SAV (F3E6H, 1)
  initial value: 0

STATFL (F3E7H, 1)
  initial value: 0
  contents: stores VDP status (contents of status register 0, in MSX2)

TRGFLG (F3E8H, 1)
  initial value: FFH
  contents: stores trigger button status of joystick

FORCLR (F3E9H, 1)
  initial value: 15
  contents: foreground colour; set by colour statement

BAKCLR (F3EAH, 1)
  initial value: 4
  contents: background colour; set by colour statement

BDRCLR (F3EBH, 1)
  initial value: 7
  contents: border colour; set by colour statement

MAXUPD (F3ECH, 3)
  initial value: JP 0000H (C3H, 00H, 00H)
  contents: used by CIRCLE statement internally

MINUPD (F3EFH, 3)
  initial value: JP 0000H (C3H, 00H, 00H)
  contents: used by CIRCLE statement internally

ATRBYT (F3F2H, 1)
  initial value: 15
  contents: colour code in using graphics

* Work area for PLAY statement

QUEUES (F3F3H, 2)
  initial value: QUETAB (F959H)
  contents: points to queue table at the execution of PLAY statement

FRCNEW (F3F5H), 1)
  initial value: 255

  contents: used by BASIC interpreter internally

* Work area for key input

SCNCNT (F3F6H, 1)
  initial value: 1
  contents: interval for the key scan

REPCNT (F3F7H, 1)
  initial value: 50
  contents: delay until the auto-repeat of the key begins

PUTPNT (F3F8H, 2)
  initial value: KEYBUF (FBF0H)
  contents: points to address to write in the key buffer

GETPNT (F3FAH, 2)
  initial value: KEYBUF (FBF0H)
  contents: points to address to read from key buffer

* Parameters for Cassette

CS120 (F3FCH, 5*2)

- 1200 baud

  contents: 83 (LOW01) ..... Low width representing bit 0
            92 (HIGH01) ..... High width representing bit 0
            38 (LOW11) ..... Low width representing bit 1
            45 (HIGH11) ..... High width representing bit 1
            HEADLEN * 2/256 ..... High bytes (HEDLEN = 2000)
                                  of header bits for short
                                  header

- 2400 baud

  contents: 37 (LOW02) ..... Low width representing bit 0
            45 (HIGH02) ..... High width representing bit 0
            14 (LOW12) ..... Low width representing bit 1
            22 (HIGH12) ..... High width representing bit 1
            HEADLEN * 4/256 ..... High bytes (HEDLEN = 2000)
                                  of header bits for short
                                  header

LOW (F406H, 2)
  initial value: LOW01, HIGH01 (by default, 1200 baud)
  contents: width of LOW and HIGH which represents bit 0 of current baud
            rate; set by <cassette baud rate> of SCREEN statement

HIGH (F408H, 2)
  initial value: LOW11, HIGH11 (by default, 1200 baud)
  contents: width of LOW and HIGH which represents bit 1 of current baud
            rate; set by <cassette baud rate> of SCREEN statement

HEADER (F40AH, 1)
  initial value: HEADLEN * 2/256 (by default, 1200 baud)
  contents: header bit for the short header of current baud rate
            (HEADLEN = 2000); set by <cassette baud rate> of SCREEN
            statement

ASPCT1 (F40BH, 1)
  contents: 256/aspect ratio; set by SCREEN statement to use in CIRCLE
            statement

ASPCT2 (F40DH, 1)
  contents: 256 * aspect ratio; set by SCREEN statement to use in CIRCLE
            statement

ENDPRG (F40FH, 5)
  initial value: ":"
  contents: false end of program for RESUME NEXT statement

* Work used by BASIC internally

ERRFLG (F414H, 1)
  contents: area to store the error number

LPTPOS (F415H, 1)
  initial value: 0
  contents: printer head location

PRTFLG (F416H, 1)
  contents: flag whether to send to printer

NTMSXP (F417H, 1)
  contents: printer (0 = printer for MSX, otherwise not)

RAWPRT (F418H, 1)
  contents: non-zero when printing in raw-mode

VLZADR (F419H, 2)
  contents: address of character to be replaced by VAL function

VLZDAT (F41BH, 1)
  contents: character to be replaced with 0 by VAL function

CURLIN (F41CH, 2)
  contents: currently executing line number of BASIC

KBUF (F41FH, 318)
  contents: crunch buffer; translated into intermediate language from
            BUF (F55EH)

BUFMIN (F55DH, 1)
  initial value: ","
  contents: used in INPUT statement

BUF (F55EH, 258)
  contents: buffer to store characters typed; where direct statements
            are stored in ASCII code

ENDBUF (F660H, 1)
  contents: prevents overflow of BUF (F55EH)

TTYPOS (F661H, 1)
  contents: virtual cursor location internally retained by BASIC

DIMFLG (F662H, 1)
  contents: used by BASIC internally

VALTYP (F663H, 1)
  contents: used to identify the type of variable

DORES (F664H, 1)
  contents: indicates whether stored word can be crunched

DONUM (F665H, 1)
  contents: flag for crunch

CONTXT (F666H, 2)
  contents: stores text address used by CHRGET

CONSAV (F668H, 1)
  contents: stores token of constant after calling CHRGET

CONTYP (F669H, 1)
  contents: type of stored constant

CONLO (F66AH, 8)
  contents: value of stored constant

MEMSIZ (F672H, 2)
  contents: highest address of memory used by BASIC

STKTOP (F674H, 2)
  contents: address used as stack by BASIC; depending on CLEAR statement

TXTTAB (F676H, 2)
  contents: starting address of BASIC text area

TEMPPT (F768H, 2)
  initial value: TEMPST (F67AH)
  contents: starting address of unused area of temporary descriptor

TEMPST (F67AH, 3 * NUMTMP)
  contents: area for NUMTEMP

DSCTMP (F698H, 3)
  contents: string descriptor which is the result of string function

FRETOP (F69BH, 2)
  contents: starting address of unused area of string area

TEMP3 (F69DH, 2)
  contents: used for garbage collection or by USR function

TEMP8 (F69FH, 2)
  contents: for garbage collection

ENDFOR (F6A1H, 2)
  contents: stores next address of FOR statement (to begin execution from
            the next of FOR statement at loops)

DATLIN (F6A3H, 2)
  contents: line number of DATA statement read by READ statement

SUBFLG (F6A5H, 1)
  contents: flag for array for USR function

FLGINP (F6A6H, 1)
  contents: flag used in INPUT or READ

TEMP (F6A7H, 2)
  contents: location for temporary reservation for statement code; used
            for variable pointer, text address, and others

PTRFLG (F6A9H, 1)
  contents: 0 if there is not a line number to be converted,otherwise not

AUTFLG (F6AAH, 1)
  contents: flag for AUTO command validity (non-zero = valid, otherwise
            invalid)

AUTLIN (F6ABH, 2)
  contents: last input line number

AUTINC (F6ADH, 2)
  initial value: 10
  contents: increment value of line number of AUTO command

SAVTXT (F6AFH, 2)
  contents: area to store address of currently executing text; mainly
            used for error recovery by RESUME statement

ERRLIN (F6B3H, 2)
  contents: line number where an error occurred

DOT (F6B5H, 2)
  contents: last line number which was displayed in screen or entered

ERRTXT (F6B7H, 2)
  contents: text address which caused an error; mainly used for error
            recovery by RESUME statement

ONELIN (F6B9H, 2)
  contents: text address to which control jumps at error; set by ON
            ERROR GOTO statement

ONEFLG (F6BBH, 1)
  contents: flag which indicates error routine execution
            (non-zero = in execution, otherwise not)

TEMP2 (F6BCH, 2)
  contents: for temporary storage

OLDLIN (F6BEH, 2)

  contents: line number which was terminated by Ctrl+STOP, STOP
            instruction, END instruction, or was executed last

OLDTXT (F6C0H, 2)
  contents: address to be executed next

VARTAB (F6C2H, 2)
  contents: starting address of simple variable; executing NEW statement
            causes [contents of TXTTAB(F676H) + 2] to be set

ARYTAB (F6C4H, 2)
  contents: starting address of array table

STREND (F6C6H, 2)
  contents: last address of memory in use as text area or variable area

DATPTR (F6C8H, 2)
  contents: text address of data read by executing READ statement

DEFTBL (F6CAH, 26)
  contents: area to store type of variable for one alphabetical
            character; depends on type declaration such as CLEAR, DEFSTR,
            !, or #

* Work for user function parameter

PRMSTK (F6E4H, 2)
  contents: previous definition block on stack (for garbage collection)

PRMLEN (F6E6H, 2)
  contents: number of bytes of objective data

PARM1 (F6E8H, PRMSIZ)
  contents: objective parameter definition table; PRMSIZ is number of
            bytes of definition block, initial value is 100

PRMPRV (F74CH, 2)
  initial value: PRMSTK
  contents: pointer to previous parameter block (for garbage collection)

PRMLN2 (F74EH, 2)
  contents: size of parameter block

PARM2 (F750H, 100)

```
  contents: for parameter storage


PRMFLG (F7B4H, 1)
  contents: flag to indicate whether PARM1 was searched


ARYTA2 (F7B5H, 2)
  contents: end point of search


NOFUNS (F7B7H, 1)
  contents: 0 if there is not an objective function


TEMP9 (F7B8H, 2)
  contents: location of temporary storage for garbage collection


FUNACT (F7BAH, 2)
  contents: number of objective functions


SWPTMP (F7BCH, 8)
  contents: location of temporary storage of the value of the first
            variable of SWAP statement


TRCFLG (F7C4H, 1)
  contents: trace flag (non-zero = TRACE ON, 0 = TRACE OFF)



* Work for Math-Pack


FBUFFR (F7C5H, 43)
  contents: used internally by Math-Pack


DECTMP (F7F0H, 2)
  contents: used to transform decimal integer to floating-point number


DECTM2 (F7F2H, 2)
  contents: used at division routine execution


DECCNT (F7F4H, 2)
  contents: used at division routine execution


DAC (F7F6H, 16)
  contents: area to set the value to be calculated


HOLD8 (F806H, 48)
  contents: register storage area for decimal multiplication
```

HOLD2 (F836H, 8)
  contents: used internally by Math-Pack

HOLD (F83EH, 8)
  contents: used internally by Math-Pack

ARG (F847H, 16)
  contents: area to set the value to be calculated with DAC (F7F6H)

RNDX (F857H, 8)
  contents: stores last random number in double precision real number;
            set by RND function

* Data area used by BASIC interpreter

MAXFIL (F85FH, 1)
  contents: maximum file number; set by MAXFILES statement

FILTAB (F860H, 2)
  contents: starting address of file data area

NULBUF (F862H, 2)
  contents: points to buffer used in SAVE and LOAD by BASIC interpreter

PTRFIL (F864H, 2)
  contents: address of file data of currently accessing file

RUNFLG (F866H, 2)
  contents: non-zero value if program was loaded and executed; used
            by R option of LOAD statement

FILNAM (F866H, 11)
  contents: area to store filename

FILNM2 (F871H, 11)
  contents: area to store filename

NLONLY (F87CH, 1)
  contents: non-zero value if program is being loaded

SAVEND (F87DH, 2)
  contents: end address of assembly language program to be saved

FNKSTR (F87FH, 160)
  contents: area to store function key string (16 character x 10)

CGPNT (F91FH, 3)
  contents: address to store character font on ROM

NAMBAS (F922H, 2)
  contents: base address of current pattern name table

CGPBAS (F924H, 2)
  contents: base address of current pattern generator table

PATBAS (F926H, 2)
  contents: base address of current sprite generator table

ATRBAS (F928H, 2)
  contents: base address of current sprite attribute table

CLOC (F92AH, 2)
  contents: used internally by graphic routine

CMASK (F92CH, 1)
  contents: used internally by graphic routine

MINDEL (F92DH, 1)
  contents: used internally by graphic routine

MAXDEL (F92FH, 2)
  contents: used internally by graphic routine

* Data area used by CIRCLE statement

ASPECT (F931H, 2)
  contents: aspect ratio of the circle; set by <ratio> of CIRCLE
            statement

CENCNT (F933H, 2)
  contents: used internally by CIRCLE statement

CLINEF (F935H, 1)
  contents: flag whether a line is drawn toward the center; specified
            by <angle> of CIRCLE statement

CNPNTS (F936H, 2)
  contents: point to be plotted

CPLOTF (F938H, 1)
  contents: used internally by CIRCLE statement

CPCNT (F939H, 2)
  contents: number of one eight of the circle

CPNCNT8 (F93BH, 2)
  contents: used internally by CIRCLE statement

CPCSUM (F93DH, 2)
  contents: used internally by CIRCLE statement

CSTCNT (F93FH, 2)
  contents: used internally by CIRCLE statement

CSCLXY (F941H, 1)
  contents: scale of x and y

CSAVEA (F942H, 2)
  contents: reservation area of ADVGRP

CSAVEM (F944H, 1)
  contents: reservation area of ADVGRP

CXOFF (F945H, 2)
  contents: x offset from the center

CYOFF (F947H, 2)
  contents: y offset from the center

* Data area used in PAINT statement

LOHMSK (F949H, 1)
  contents: used internally by PAINT statement

LOHDIR (F94AH, 1)
  contents: used internally by PAINT statement

LOHADR (F94BH, 2)
  contents: used internally by PAINT statement

LOHCNT (F94DH, 2)
  contents: used internally by PAINT statement

SKPCNT (F94FH, 2)
  contents: skip count

MIVCNT (F951H, 2)
  contents: movement count

PDIREC (F953H, 1)
  contents: direction of the paint

LFPROG (F954H, 1)
  contents: used internally by PAINT statement

RTPROG (F955H, 1)
  contents: used internally by PAINT statement

* Data area used in PLAY statement

MCLTAB (F956H, 2)
  contents: points to the top of the table of PLAY macro or DRAW macro

MCLFLG (F958H, 1)
  contents: assignment of PLAY/DRAW

QUETAB (F959H, 24)
  contents: queue table
            +0: PUT offset
            +1: GET offset
            +2: backup character
            +3: queue length
            +4: queue address
            +5: queue address

QUEBAK (F971H, 4)
  contents: used in BCKQ

VOICAQ (F975H, 128)
  contents: queue of voice 1 (1 = a)

VOICBQ (F9F5H, 128)
  contents: queue of voice 2 (2 = b)

VOICCQ (FA75H, 128)
  contents: queue of voice 3 (3 = c)

* Work area added in MSX2

DPPAGE (FAF5H, 1)
  contents: display page number

ACPAGE (FAF6H, 1)
  contents: active page number

AVCSAV (FAF7H, 1)
  contents: reserves AV control port

EXBRSA (FAF8H, 1)
  contents: SUB-ROM slot address

CHRCNT (FAF9H, 1)
  contents: character counter in the buffer; used in Roman-kana
            translation (value is 0 <=n <=2)

ROMA (FAFAH, 2)
  contents: area to store character in the buffer; used in Roman-kana
            translation (Japan version only)

MODE (FAFCH, 1)
  contents: mode switch for VRAM size

            (0000WVV0)

                 |||
                 |++--- 00 = 16K VRAM
                 | 01 = 64K VRAM
                 | 11 = 128K VRAM
                 |
                 +----- 1 = mask, 0 = no mask
                      Flags whether to specify VRAM address
                      ANDed with 3FFFH in SCREEN 0 to 3;
                      in SCREEN 4 to 8, never masked

NORUSE (FAFDH, 1)
  contents: unused

```
XSAVE (FAFEH, 2)
  contents: [ I OOOOOOO XXXXXXXX ]


YSAVE (FB00H, 2)
  contents: [ x OOOOOOO YYYYYYYY ]

            I = 1 lightpen interrupt request
            OOOOOOO = unsigned offset
            XXXXXXX = X-coordinate
            YYYYYYY = Y-coordinate


LOGOPR (FB02H, 1)
      contents:   logical operation code


* Data area used by RS-232C


RSTMP (FB03H, 50)
  contents: work area for RS-232C or disk


TOCNT (FB03H, 1)
  contents: used internally by RS-232C routine


RSFCB (FB04H, 2)
  contents: FB04H + 0:  LOW address of RS-232C
            FB04H + 1:  HIGH address of RS-232C


RSIQLN (FB06H, 5)
  contents: used internally by RS-232C routine


MEXBIH (FB07H, 5)
  contents: FB07H +0:  RST 30H (0F7H)
            FB07H +1:  byte data
            FB07H +2:  (Low)
            FB07H +3:  (High)
            FB07H +4:  RET (0C9H)


OLDSTT (FB0CH, 5)
  contents: FB0CH +0:  RST 30H (0F7H)
            FB0CH +1:  byte data
            FB0CH +2:  (Low)
            FB0CH +3:  (High)
            FB0CH +4:  RET (0C9H)


OLDINT (FB12H, 5)
  contents: FB12H +0:  RST 30H (0F7H)
            FB12H +1:  byte data
            FB12H +2:  (Low)
            FB12H +3:  (High)
```

```
            FB12H +4:  RET (0C9H)


DEVNUM (FB17H, 1)
  contents: used internally by RS-232C routine


DATCNT (FB18H, 3)
  contents: FB18H +0:  byte data
            FB18H +1:  byte pointer
            FB12H +2:  byte pointer


ERRORS (FB1BH, 1)
  contents: used internally by RS-232C routine


FLAGS (FB1CH, 1)
  contents: used internally by RS-232C routine


ESTBLS (FB1DH, 1)
  contents: used internally by RS-232C routine


COMMSK (FB1EH, 1)
  contents: used internally by RS-232C routine


LSTCOM (FB1FH, 1)
  contents: used internally by RS-232C routine


LSTMOD (FB20H, 1)
  contents: used internally by RS-232C routine



* Data area used by DOS


reserved (FB21H to FB34H)
  contents: used by DOS



* Data area used by PLAY statement
   (the following is the same as with MSX1)


PRSCNT (FB35H, 1)
  contents: D1 to D0    string parse
            D7 = 0            1 pass


SAVSP (FB36H, 2)
  contents: reserves stack pointer in play
```

VOICEN (FB38H, 1)
  contents: current interpreted voice

SAVVOL (FB39H, 2)
  contents: reserves volume for the pause

MCLLEN (FB3BH, 1)
  contents: used internally by PLAY statement

MCLPTR (FB3CH, 1)
  contents: used internally by PLAY statement

QUEUEN (FB3EH, 1)
  contents: used internally by PLAY statement

MUSICF (FC3FH, 1)
  contents: interrupt flag for playing music

PLYCNT (FB40H, 1)
  contents: number of PLAY statements stored in the queue

* Offset from voice static data area
  (offset is in decimal)

METREX (+0, 2)
  contents: timer count down

VCXLEN (+2, 1)
  contents: MCLLEN for this voice

VCXPTR (+3, 2)
  contents: MCLPTR for this voice

VCXSTP (+5, 2)
  contents: reserves the top of the stack pointer

QLENGX (+7, 1)
  contents: number of bytes stored in the queue

NTICSX (+8, 2)
  contents: new count down

TONPRX (+10, 2)
  contents: area to set tone period

AMPPRX (+12, 1)
  contents: discrimination of volume and enveloppe

ENVPRX (+13, 2)
  contents: area to set enveloppe period

OCTAVX (+15, 1)
  contents: area to set octave

NOTELX (+16, 1)
  contents: area to set tone length

TEMPOX (+17, 1)
  contents: area to set tempo

VOLUMX (+18, 1)
  contents: area to set volume

ENVLPX (+19, 14)
  contents: area to set enveloppe wave form

MCLSTX (+33, 3)
  contents: reservation area of stack

MCLSEX (+36, 1)
  contents: initialisation stack

VCBSIZ (+37, 1)
  contents: static buffer size

* Voice static data area

VCBA (FB41H, 37)
  contents: static data for voice 0

VCBB (FB66H, 37)
  contents: static data for voice 1

VCBC (FB8BH, 37)
  contents: static data for voice 2

* Data area

ENSTOP (FBB0H, 1)
  contents: flag to enable warm start by [SHIFT+Ctrl+Kana key]
            (0 = disable, otherwise enable)

BASROM (FBB1H, 1)
  contents: indicates BASIC text location (0 = on RAM, otherwise in ROM)

LINTTB (FBB2H, 24)
  contents: line terminal table; area to keep information about
            each line of text screen

FSTPOS (FBCAH, 2)
  contents: first character location of line from INLIN (00B1H) of BIOS

CODSAV (FBCCH, 1)
  contents: area to reserve the character where the cursor is stacked

FNKSW1 (FBCDH, 1)
  contents: indicates which function key is displayed at KEY ON
            (1 = F1 to F5 is displayed, 0 = F6 to F10 is displayed)

FNKFLG (FBCEH, 10)
  contents: area to allow, inhibit, or stop the execution of the line
            defined in ON KEY GOSUB statement, or to reserve it for each
            function key; set by KEY(n)ON/OFF/STOP statement
            (0 = KEY(n)OFF/STOP, 1= KEY(n)ON)

ONGSBF (FBD8H, 1)
  contents: flag to indicate whether event waiting in TRPTBL (FC4CH)
            occurred

CLIKFL (FBD9H, 1)
  contents: key click flag

OLDKEY (FBDAH, 11)
  contents: key matrix status (old)

NEWKEY (FBE5H, 11)
  contents: key matrix status (new)

KEYBUF (FBF0H, 40)

contents: key code buffer

LINWRK (FC18H, 40)
contents: temporary reservation location used by screen handler

PATWRK (FC40H, 8)
contents: temporary reservation location used by pattern converter

BOTTOM (FC48H, 2)
contents: installed RAM starting (low) address; ordinarily 8000H in MSX2

HIMEM (FC4AH, 2)
contents: highest address of available memory; set by <memory upper limit> of CLEAR statement

TRAPTBL (FC4CH, 78)
contents: trap table used to handle interrupt; one table consists of three bytes, where first byte indicates ON/OFF/STOP status and the rest indicate the text address to be jumped to

FC4CH to FC69H (3 * 10 bytes)	used in ON KEY GOSUB
FC6AH to FC6CH (3 * 1 byte)	used in ON STOP GOSUB
FC6DH to FC6FH (3 * 1 byte)	used in ON SPRITE GOSUB
FC70H to FC7EH (3 * 5 bytes)	used in ON STRIG GOSUB
FC7FH to FC81H (3 * 1 byte)	used in ON INTERVAL GOSUB
FC82H to FC99H	for expansion

RTYCNT (FC9AH, 1)
contents: used internally by BASIC

INTFLG (FC9BH, 1)
contents: if Ctrl+STOP is pressed, setting 03H here causes a stop

PADY (FC9CH, 1)
contents: Y-coordinate of the paddle)

PADX (FC9DH, 1)
contents: X-coordinate of the paddle)

JIFFY (FC9EH, 2)
contents: used internally by PLAY statement

INTVAL (FCA0H, 2)
contents: interval period; set by ON INTERVAL GOSUB statement

```
INTCNT (FCA2H, 2)
  contents: counter for interval


LOWLIM (FCA4H, 1)
  contents: used during reading from cassette tape


WINWID (FCA5H, 1)
  contents: used during reading from cassette tape


GRPHED (FCA6H, 1)
  contents: flag to send graphic character (1 = graphic character,
            0 = normal character)


ESCCNT (FCA7H, 1)
  contents: area to count from escape code


INSFLG (FCA8H, 1)
  contents: flag to indicate insert mode (0 = normal mode,
            otherwise = insert mode)


CSRSW (FCA9H, 1)
  contents: whether cursor is displayed (0 = no, otherwise = yes);
            set by <cursor swicth> of LOCATE statement


CSTYLE (FCAAH, 1)
  contents: cursor shape (0 = block, otherwise = underline)


CAPST (FCABH, 1)
  contents: CAPS key status (0 = CAP OFF, otherwise = CAP ON)


KANAST (FCACH, 1)
  contents: kana key status (0 = kaka OFF, otherwise = kana ON)


KANAMD (FCADH, 1)
  contents: kana key arrangement status (0 = 50-sound arrangement,
            otherwise = JIS arrangement)


FLBMEM (FCAEH, 1)
  contents: 0 when loading BASIC program


SCRMOD (FCAFH, 1)
  contents: current screen mode number


OLDSCR (FCB0H, 1)
  contents: screen mode reservation area
```

CASPRV (FCB1H, 1)
  contents: character reservation area used by CAS:

BRDATR (FCB2H, 1)
  contents: border colour code used by PAINT; set by <border colour>
            in PAINT statement

GXPOS (FCB3H, 2)
  contents: X-coordinate

GYPOS (FCB5H, 2)
  contents: Y-coordinate

GRPACX (FCB7H, 2)
  contents: graphic accumulator (X-coordinate)

GRPACY (FCB9H, 2)
  contents: graphic accumulator (Y-coordinate)

DRWFLG (FCBBH, 1)
  contents: flag used in DRAW statement

DRWSCL (FCBCH, 1)
  contents: DRAW scaling factor (0 = no scaling, otherwise = scaling)

DRWANG (FCBDH, 1)
  contents: angle at DRAW

RUNBNF (FCBEH, 1)
  contents: flag to indicate BLOAD in progress, BSAVE in progress,
            or neither

SAVENT (FCBFH, 2)
  contents: starting address of BSAVE

EXPTBL (FCC1H, 4)
  contents: flag table for expansion slot; whether the slot is expanded

SLTTBL (FCC5H, 4)
  contents: current slot selection status for each expansion slot
            register

SLTATR (FCC9H, 64)

  contents: reserves attribute for each slot

SLTWRK (FD09H, 128)
  contents: allocates specific work area for each slot

PROCNM (FD89H, 16)
  contents: stores name of expanded statement (after CALL statement) or
            expansion device (after OPEN); 0 indicates the end

DEVICE (FD99H, 1)
  contents: used to identify cartridge device

* Hooks

H.KEYI (FD9AH)
  meaning: beginning of MSXIO interrupt handling
  purpose: adds the interrupt operation such as RS-232C

H.TIMI (FD9FH)
  meaning: MSXIO timer interrupt handling
  purpose: adds the timer interrupt handling

H.CHPH (FDA4H)
  meaning: beginning of MSXIO CHPUT (one character output)
  purpose: connects other console device

H.DSPC (FDA9H)
  meaning: beginning of MSXIO DSPCSR (cursor display)
  purpose: connects other console device

H.ERAC (FDAEH)
  meaning: beginning of MSXIO ERACSR (erase cursor)
  purpose: connects other console device

H.DSPF (FDB3H)
  meaning: beginning of MSXIO DSPFNK (function key display)
  purpose: connects other console device

H.ERAF (FDB8H)
  meaning: beginning of MSXIO ERAFNK (erase function key)
  purpose: connects other console device

H.TOTE (FDBDH)
  meaning: beginning of MSXIO TOTEXT (set screen in text mode)
  purpose: connects other console device

H.CHGE (FDC2H)
  meaning: beginning of MSXIO CHGET (get one character)
  purpose: connects other console device

H.INIP (FDC7H)
  meaning: beginning of MSXIO INIPAT (character pattern initialisation)
  purpose: uses other character set

H.KEYC (FDCCH)
  meaning: beginning of MSXIO KEYCOD (key code translation)
  purpose: uses other key arrangement

H.KYEA (FDD1H)
  meaning: beginning of MSXIO NMI routine (Key Easy)
  purpose: uses other key arrangement

H.NMI (FDD6H)
  meaning: beginning of MSXIO NMI (non-maskable interrupt)
  purpose: handles NMI

H.PINL (FDDBH)
  meaning: beginning of MSXIO PINLIN (one line input)
  purpose: uses other console input device or other input method

H.QINL (FDE0H)
  meaning: beginning of MSXINL QINLIN (one line input displaying "?")
  purpose: uses other console input device or other input method

H.INLI (FDE5H)
  meaning: beginning of MSXINL INLIN (one line input)
  purpose: uses other console input device or other input method

H.ONGO (FDEAH)
  meaning: beginning of MSXSTS INGOTP (ON GOTO)
  purpose: uses other interrupt handling device

H.DSKO (FDEFH)
  meaning: beginning of MSXSTS DSKO$ (disk output)
  purpose: connects disk device

H.SETS (FDF4H)
  meaning: beginning of MSXSTS SETS (set attribute)
  purpose: connects disk device

H.NAME (FDF9H)

```
  meaning:  beginning of MSXSTS NAME (rename)
  purpose:  connects disk device


H.KILL (FDFEH)
  meaning:  beginning of MSXSTS KILL (delete file)
  purpose:  connects disk device


H.IPL (FE03H)
  meaning:  beginning of MSXSTS IPL (initial program loading)
  purpose:  connects disk device


H.COPY (FE08H)
  meaning:  beginning of MSXSTS COPY (file copy)
  purpose:  connects disk device


H.CMD (FE0DH)
  meaning:  beginning of MSXSTS CMD (expanded command)
  purpose:  connects disk device


H.DSKF (FE12H)
  meaning:  beginning of MSXSTS DSKF (unusde disk space)
  purpose:  connects disk device


H.DSKI (FE17H)
  meaning:  beginning of MSXSTS DSKI (disk input)
  purpose:  connects disk device


H.ATTR (FE1CH)
  meaning:  beginning of MSXSTS ATTR$ (attribute)
  purpose:  connects disk device


H.LSET (FE21H)
  meaning:  beginning of MSXSTS LSET (left-padded assignment)
  purpose:  connects disk device


H.RSET (FE26H)
  meaning:  beginning of MSXSTS RSET (right-padded assignment)
  purpose:  connects disk device


H.FIEL (FE2BH)
  meaning:  beginning of MSXSTS FIELD (field)
  purpose:  connects disk device


H.MKI$ (FE30H)
  meaning:  beginning of MSXSTS MKI$ (create integer)
  purpose:  connects disk device
```

H.MKS$ (FE35H)
  meaning: beginning of MSXSTS MKS$ (create single precision real)
  purpose: connects disk device

H.MKD$ (FE3AH)
  meaning: beginning of MSXSTS MKD$ (create double precision real)
  purpose: connects disk device

H.CVI (FE3FH)
  meaning: beginning of MSXSTS CVI (convert integer)
  purpose: connects disk device

H.CVS (FE44H)
  meaning: beginning of MSXSTS CVS (convert single precision real)
  purpose: connects disk device

H.CVD (FE49H)
  meaning: beginning of MSXSTS CVS (convert double precision real)
  purpose: connects disk device

H.GETP (FE4EH)
  meaning: SPDSK GETPTR (get file pointer)
  purpose: connects disk device

H.SETF (FE53H)
  meaning: SPCDSK SETFIL (set file pointer)
  purpose: connects disk device

H.NOFO (FE58H)
  meaning: SPDSK NOFOR (OPEN statement without FOR)
  purpose: connects disk device

H.NULO (FE5DH)
  meaning: SPCDSK NULOPN (open unused file)
  purpose: connects disk device

H.NTFL (FE62H)
  meaning: SPCDSK NTFLO (file number is not 0)
  purpose: connects disk device

H.MERG (FE67H)
  meaning: SPCDSK MERGE (program file merge)
  purpose: connects disk device

H.SAVE (FE6CH)
  meaning: SPCDSK SAVE (save)

```
  purpose:  connects disk device


H.BINS (FE71H)
  meaning:  SPCDSK BINSAV (save in binary)
  purpose:  connects disk device


H.BINL (FE76H)
  meaning:  SPCDSK BINLOD (load in binary)
  purpose:  connects disk device


H.FILE (FD7BH)
  meaning:  SPCDSK FILES (displey filename)
  purpose:  connects disk device


H.DGET (FE80H)
  meaning:  SPCDSK DGET (disk GET)
  purpose:  connects disk device


H.FILO (FE85H)
  meaning:  SPCDSK FILOU1 (file output)
  purpose:  connects disk device


H.INDS (FE8AH)
  meaning:  SPCDSK INDSKC (disk attribute input)
  purpose:  connects disk device


H.RSLF (FE8FH)
  meaning:  SPCDSK; re-select previous drive
  purpose:  connects disk device


H.SAVD (FE94H)
  meaning:  SPCDSK; reserve current disk
  purpose:  connects disk device


H.LOC (FE99H)
  meaning:  SPCDSK LOC function (indicate location)
  purpose:  connects disk device


H.LOF (FE9EH)
  meaning:  SPCDSK LOC function (file length)
  purpose:  connects disk device


H.EOF (FEA3H)
  meaning:  SPCDSK EOF function (end of file)
  purpose:  connects disk device
```

H.FPOS (FEA8H)
  meaning: SPCDSK FPOS function (file location)
  purpose: connects disk device

H.BAKU (FEADH)
  meaning: SPCDSK BAKUPT (backup)
  purpose: connects disk device

H.PARD (FEB2H)
  meaning: SPCDEV PARDEV (get peripheral name)
  purpose: expands logical device name

H.NODE (FEB7H)
  meaning: SPCDEV NODEVN (no device name)
  purpose: sets default device name to other device

H.POSD (FEBCH)
  meaning: SPCDEV POSDSK
  purpose: connects disk device

H.DEVN (FEC1H)
  meaning: SPCDEV DEVNAM (process device name)
  purpose: expands logical device name

H.GEND (FEC6H)
  meaning: SPCDEV GENDSP (FEC6H)
  purpose: expands logical device name

H.RUNC (FECBH)
  meaning: BIMISC RUNC (clear for RUN)

H.CLEAR (FED0H)
  meaning: BIMISC CLEARC (clear for CLEAR statement)

H.LOPD (FED5H)
  meaning: BIMISC LOPDFT (set loop and default value)
  purpose: uses other default value for variable

H.STKE (FEDAH)
  meaning: BIMISC STKERR (stack error)

H.ISFL (FEDFH)
  meaning: BIMISC ISFLIO (file input-output or not)

H.OUTD (FEE4H)
  meaning: BIO OUTDO (execute OUT)

H.CRDO (FEE9H)
  meaning: BIO CRDO (execute CRLF)

H.DSKC (FEEEH)
  meaning: BIO DSKCHI (input disk attribute)

H.DOGR (FEF3H)
  meaning: GENGRP DOGRPH (execute graphic operation)

H.PRGE (FEF8H)
  meaning: BINTRP PRGEND (program end)

H.ERRP (FEFDH)
  meaning: BINTRP ERRPTR (error display)

H.ERRF (FF02H)
  meaning: BINTRP

H.READ (FF07H)
  meaning: BINTRP READY

H.MAIN (FF0CH)
  meaning: BINTRP MAIN

H.DIRD (FF11H)
  meaning: BINTRP DIRDO (execute direct statement)

H.FINI (FF16H)
  meaning: BINTRP

H.FINE (FF1BH)
  meaning: BINTRP

H.CRUN (FF20H)
  meaning: BINTRP

H.CRUN (FF20H)
  meaning: BINTRP

H.CRUS (FF25H)
  meaning: BINTRP

H.ISRE (FF2AH)
  meaning: BINTRP

H.NTFN (FF2FH)
  meaning: BINTRP

H.NOTR (FF34H)
  meaning: BINTRP

H.SNGF (FF39H)
  meaning: BINTRP

H.NEWS (FF3EH)
  meaning: BINTRP

H.GONE (FF43H)
  meaning: BINTRP

H.CHRG (FF48H)
  meaning: BINTRP

H.RETU (FF4DH)
  meaning: BINTRP

H.PRTF (FF52H)
  meaning: BINTRP

H.COMP (FF57H)
  meaning: BINTRP

H.FINP (FF5CH)
  meaning: BINTRP

H.TRMN (FF61H)
  meaning: BINTRP

H.FRME (FF66H)
  meaning: BINTRP

H.NTPL (FF6BH)
  meaning: BINTRP

H.EVAL (FF70H)
  meaning: BINTRP

H.OKNO (FF75H)
  meaning: BINTRP

H.FING (FF7AH)
  meaning: BINTRP

H.ISMI (FF7FH)
  meaning: BINTRP ISMID$ (MID$ or not)

H.WIDT (FF84H)
  meaning: BINTRP WIDTHS (WIDTH)

H.LIST (FF89H)
  meaning: BINTRP LIST

H.BUFL (FF8EH)
  meaning: BINTRP BUFLIN (buffer line)

H.FRQI (FF93H)
  meaning: BINTRP FRQINT

H.SCNE (FF98H)
  meaning: BINTRP

H.FRET (FF9DH)
  meaning: BINTRP FRETMP

H.PTRG (FFA2H)
  meaning: BIPTRG PTRGET (get pointer)
  purpose: uses variable other than default value

H.PHYD (FFA7H)
  meaning: MSXIO PHYDIO (physical disk input-output)
  purpose: connects disk device

H.FORM (FFACH)
  meaning: MSXIO FORMAT (format disk)
  purpose: connects disk device

H.ERRO (FFB1H)
  meaning: BINTRP ERROR
  purpose: error handling for application program

H.LPTO (FFB6H)
  meaning: MSXIO LPTOUT (printer output)
  purpose: uses printer other than default value

H.LPTS (FFBBH)
  meaning: MSXIO LPTSTT (printer status)
  purpose: uses printer other than default value

H.SCRE (FFC0H)
  meaning: MSXSTS SCREEN statement entry
  purpose: expands SCREEN statement

H.PLAY (FFC5H)
  meaning: MSXSTS PLAY statement entry
  purpose: expands PLAY statement

* For expanded BIOS

FCALL (FFCAH)
  contents: hook used by expanded BIOS

DISINT (FFCFH)
  contents: used by DOS

ENAINT (FFD4H)
  contents: used by DOS

===

Changes from the original in APPENDIX 5:

- The original VRAM mapping figures have been converted to simple text tables.

- In SCREEN 0 (WIDTH 80) map, different end addresses for the blink table are indicated for 24 lines mode and 26.5 lines mode.

-=-

APPENDIX 5 - VRAM MAP

* SCREEN 0 (WIDTH 40) / TEXT 1

```
0000H - 03BFH     -->   Pattern name table
0400H - 042FH     -->   Palette table
0800H - 0FFFH     -->   Pattern generator table
```

* SCREEN 0 (WIDTH 80) / TEXT 2

```
0000H - 077FH     -->   Pattern name table
0800H - 08EFH     -->   Blink table (24 lines mode)
      090DH                 (26.5 lines mode)
0F00H - 0F2FH     -->   Palette table
1000H - 17FFH     -->   Pattern generator table
```

* SCREEN 1 / GRAPHIC 1

```
0000H - 07FFH     -->   Pattern generator table
1800H - 1AFFH     -->   Pattern name table
1B00H - 1B7FH     -->   Sprite attribute table
2000H - 201FH     -->   Colour table
2020H - 204FH     -->   Palette table
3800H - 3FFFH     -->   Sprite generator table
```

* SCREEN 2 / GRAPHIC 2

```
0000H - 07FFH     -->   Pattern generator table 1
0800H - 0FFFH     -->   Pattern generator table 2
1000H - 17FFH     -->   Pattern generator table 3
1800H - 18FFH     -->   Pattern name table 1
1900H - 19FFH     -->   Pattern name table 2
1A00H - 1AFFH     -->   Pattern name table 3
1B00H - 1B7FH     -->   Sprite attribute table
1B80H - 1BAFH     -->   Palette table
2000H - 27FFH     -->   Colour table 1
2800H - 2FFFH     -->   Colour table 2
3000H - 37FFH     -->   Colour table 3
3800H - 3FFFH     -->   Sprite generator table
```

* SCREEN 3 / MULTI COLOUR

```
0000H - 07FFH     -->   Pattern generator table
0800H - 0AFFH     -->   Pattern name table
1B00H - 1B7FH     -->   Sprite attribute table
2020H - 204FH     -->   Palette table
3800H - 3FFFH     -->   Sprite generator table
```

* SCREEN 4 / GRAPHIC 3

```
0000H - 07FFH     -->   Pattern generator table 1
0800H - 0FFFH     -->   Pattern generator table 2
1000H - 17FFH     -->   Pattern generator table 3
1800H - 18FFH     -->   Pattern name table 1
1900H - 19FFH     -->   Pattern name table 2
1A00H - 1AFFH     -->   Pattern name table 3
1B80H - 1BAFH     -->   Palette table
1C00H - 1DFFH     -->   Sprite colour table
1E00H - 1E7FH     -->   Sprite attribute table
2000H - 27FFH     -->   Colour table 1
2800H - 2FFFH     -->   Colour table 2
```

```
3000H - 37FFH     -->   Colour table 3
3800H - 3FFFH     -->   Sprite generator table


* SCREEN 5, 6 / GRAPHIC 4, 5

0000H - 5FFFH     -->   Pattern name table (192 lines)
      69FFH                     (212 lines)
7400H - 75FFH     -->   Sprite colour table
7600H - 767FH     -->   Sprite attribute table
7680H - 76AFH     -->   Palette table
7A00H - 7FFFH     -->   Sprite generator table


* SCREEN 7, 8 / GRAPHIC 6, 7

0000H - BFFFH     -->   Pattern name table (192 lines)
      D3FFH                     (212 lines)
F000H - F7FFH     -->   Sprite generator table
F800H - F9FFH     -->   Sprite colour table
FA00H - FA7FH     -->   Sprite attribute table
FA80H - FAAFH     -->   Palette table


================================================================================

Changes from the original in APPENDIX 6:

none

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

APPENDIX 6 - I/O MAP


00H to 3FH  user defined

40H to 7FH  reserved

80H to 87H  for RS-232C
      80H   8251 data
      81H   8251 status/command
      82H   status read/interrupt mask
      83H   unused
      84H   8253
      85H   8253
      86H   8253
      87H   8253

88H to 8BH  VDP (9938) I/O port for MSX1 adaptor
            This is V9938 I/O for MSX1. To access VDP directly,
            examine 06H and 07H of MAIN-ROM to confirm the port
            address

8CH to 8DH  for the modem

8EH to 8FH  reserved
```

```
90H to 91H  printer port
      90H   bit 0: strobe output (write)
            bit 1: status input (read)
      91H   data to be printed

92H to 97H  reserved

98H to 9BH  for MSX2 VDP (V9938)
      98H   VRAM access
      99H   command register access
      9AH   palette register access (write only)
      9BH   register pointer (write only)

9CH to 9FH  reserved

A0H to A3H  sound generator (AY-3-8910)
      A0H   address latch
      A1H   data read
      A2H   data write

A4H to A7H  reserved

A8H to ABH  parallel port (8255)
      A8H   port A
      A9H   port B
      AAH   port C
      ABH   mode set

ACH to AFH  MSX engine (one chip MSX I/O)

B0H to B3H  expansion memory (SONY specification) (8255)
      A8H   port A, address (A0 to A7)
      A9H   port B, address (A8 to A10, A13 to A15), control R/"
      AAH   port C, address (A11 to A12), data (D0 - D7)
      ABH   mode set

B4H to B5H  CLOCK-IC (RP-5C01)
      B4H   address latch
      B5H   data

B6H to B7H  reserved

B8H to BBH  lightpen control (SANYO specification)
      B8H   read/write
      B9H   read/write
      BAH   read/write
      BBH   write only

BCH to BFH  VHD control (JVC) (8255)
      BCH   port A
      BDH   port B
      BEH   port C

C0H to C1H  MSX-Audio

C2H to C7H  reserved
```

```
C8H to CFH  MSX interface

D0H to D7H  floppy disk controller (FDC)
            The floppy disk controller can be interrupted by an
            external signal. Interrupt is possible only when the
            FDC is accessed. Thus, the system can treat different
            FDC interfaces.

D8 to D9H   kanji ROM (TOSHIBA specification)
     D8H    b5-b0       lower address (write only)
     D9H    b5-b0       upper address (write)
            b7-b0       data (read)

DAH to DBH  for future kanji expansion

DCH to F4H  reserved

F5H         system control (write only)
            setting bit to 1 enables available I/O devices
      b0    kanji ROM
      b1    reserved for kanji
      b2    MSX-AUDIO
      b3    superimpose
      b4    MSX interface
      b5    RS-232C
      b6    lightpen
      b7    CLOCK-IC (only on MSX2)
            Bits to void the conflict between internal I/O
            devices or those connected by cartridge. The bits
            can disable the internal devices. When BIOS is initialised,
            internal devices are valid if no external devices are
            connected. Applications may not write to or read from here.

F8H         colour bus I/O

F7H         A/V control
      b0    audio R           mixing ON (write)
      b1    audio L           mixing OFF (write)
      b2    select video input      21p RGB (write)
      b3    detect video input      no input (read)
      b4    AV control        TV (write)
      b5    Ym control        TV (write)
      b6    inverse of bit 4 of VDP register 9 (write)
      b7    inverse of bit 5 of VDP register 9 (write)

F8H to FBH  reserved

FCH to FFH  memory mapper


================================================================================

Changes from the original in APPENDIX 8:

none

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

APPENDIX 8 - CONTROL CODES

Code (dec)	Code (hex)	Function	Corresponding key(s)
0	00H		CTRL + @
1	01H	header at input/output of graphic characters	CTRL + A
2	02H	move cursor to the top of the previous word	CTRL + B
3	03H	end the input-waiting state	CTRL + C
4	04H		CTRL + D
5	05H	delete below cursor	CTRL + E
6	06H	move cursor to the top of the next word	CTRL + F
7	07H	speaker output (same as the BEEP statement)	CTRL + G
8	08H	delete a character before cursor	CTRL + H or BS
9	09H	move to next horizontal tab stop	CTRL + I or TAB
10	0AH	line feed	CTRL + J
11	0BH	home cursor	CTRL + K or HOME
12	0CH	clear screen and home cursor	CTRL + L or CLS
13	0DH	carriage return	CTRL + M or RETURN
14	0EH	move cursor to the end of line	CTRL + N
15	0FH		CTRL + O
16	10H		CTRL + P
17	11H		CTRL + Q
18	12H	insert mode ON/OFF	CTRL + R or INS
19	13H		CTRL + S
20	14H		CTRL + T
21	15H	delete one line from screen	CTRL + U
22	16H		CTRL + V
23	17H		CTRL + W

24	18H		CTRL + X or SELECT
25	19H		CTRL + Y
26	1AH		CTRL + Z
27	1BH		CTRL + [or ESC
28	1CH	move cursor right	CTRL + \ or RIGHT
29	1DH	move cursor left	CTRL +] or LEFT
30	1EH	move cursor up	CTRL + ^ or UP
31	1FH	move cursor down	CTRL + _ or DOWN
127	7FH	delete character under cursor	DEL

==========

Changes from the original in APPENDIX 10:

none

-=-

APPENDIX 10 - ESCAPE SEQUENCES

* Cursor movement

<ESC> A move cursor up
<ESC> B move cursor down
<ESC> C move cursor right
<ESC> D move cursor left
<ESC> H move cursor home
<ESC> Y <Y-coordinate+20H> <X-coordinate+20H>
         move cursor to (X, Y)

* Edit, delete

<ESC> j clear screen
<ESC> E clear screen
<ESC> K delete to end of line
<ESC> J delete to end of screen
<ESC> L insert one line
<ESC> M delete one line

* Miscellaneous

<ESC> x4 set block cursor

```
<ESC> x5    hide cursor
<ESC> y4    set underline cursor
<ESC> y5    display cursor
```

==========

APPENDIX 7 - CARTRIDGE HARDWARE

and

APPENDIX 9 - CHARACTER SET

are not available here