Notes for MSX Software Developers [2nd edition]

25th June, 1985
2nd July, 1985
1st November, 1985

This document provides important information for anyone writing software for MSX/MSX2 to keep the compatibility between machines and versions. Some frequently asked questions and other useful information are also provided.

1.0 DETERMINING THE MSX VERSION

There are several ways to know the MSX version on which the software is running. Following is the recommended way to check the version number.

1. Main-ROM address 2DH contains the version information as follows:

CONTENTS OF 2DH	VERSION OF MSX
0	MSX-1
1	MSX-2
other	Not defined (reserved)

For software that runs in an environment where the page 0 of main-ROM is switched out with other ROM or RAM, such as software under MSX-DOS, use the inter-slot read function to address 2DH with slot address stored in RAM location FCC1H (See next note).

NOTE

In MSX2, the MAIN-ROM is not always located in the slot 0 or the slot 0-0. (0-0 means the secondary slot 0 of the primary slot 0.) The address FCC1H contains the slot address of the MAIN-ROM and the address FAF8H contains the slot address of SUB-ROM. (See 3.0 The slot address of BASIC-ROM.)
This is because of the MSX-2 adaptor for the MSX-1.

2. RAM address FAF8H contains the slot address of MSX2 SUB-ROM. Since this doesn't exist in MSX1, application software is able to know whether this is MSX1 or not by the contents of this RAM location. I.e.,

CONTENTS OF FAF8H	VERSION OF MSX
0	MSX-1
other	MSX-2 or newer

As you see, this method provides only the information just the version is MSX1 or not. The MAIN-ROM address 2CH-2DH contains more informations.

2.0 THE I/O ADDRESS OF VDP

In the MSX-2, the VDP is not always sitting in the address 98H-9BH. Therefore software that accesses the VDP directly must refer to the MAIN-ROM address 6 and 7 to know the addresses of the VDP as follows. This is because of the MSX-2 adaptor for the MSX-1.

VDP PORT	I/O address
VRAM read	(0006) in MAIN-ROM
VRAM write	(0007) in MAIN-ROM
Status read	(0006) + 1
Command write	(0007) + 1
Palette write	(0007) + 2
Indirect register access	(0007) + 3

3.0 THE SLOT ADDRESS OF THE BASIC-ROM

In the MSX-2, the MAIN-ROM is not always placed in slot 0 or slot 0-0. Also, there is another ROM, called SUB-ROM, which contains the software that supports new features and is placed in page 0 of some slot. The following RAM area contains the slot address of these ROMs.

To access the BIOS entries in these ROMs, use an inter-slot call if necessary.

Address	Label	contents
FCC1H	EXPTBL	The slot address of the MAIN-ROM
FAF8H	EXBRSA	The slot address of the SUB-ROM

4.0 EXPANDED SLOT

In the MSX2, there is a great possibility that any one of the slots has been expanded to have more slots containing more softwares on ROM such as SUB-ROM, DISK-ROM, RS232-ROM and special built in software ROM. So there should be never a software for the MSX2 that does not work with the machine that has expanded slot.

So, please check before release your software to make it sure that your software works in the expanded slot, and with the machine with

the RAM placed in the expanded slot.
Especially, do not access the address FFFFH as the normal RAM. This is the address of the expanded slot select register. There are several softwares that place the stack pointer in this address space with the instruction,

```
LD      SP,0000
```

that, of course, does not work with the machines with expanded slots.

5.0 CALLING THE BIOS

Please call the BIOS through the entry jump tables. Software that calls the BIOS directly will not work on all machines! See examples prgrams.

6.0 THE INITIAL STATE OF THE RAM

The contents of main RAM and VRAM is undetermined unless otherwise stated. There are some softwares that assumes the initial contents of RAM is zero (we don't know the reason) and, of course, it does not work in with all machines.

7.0 THE INITIAL VALUE OF THE STACK POINTER

MSX machines with built in disk drive are increasing. However, depending on the position of the slot where the disk ROM is placed, the disk ROM takes system control and initializes its work space earlier than the application software in the cartridge slot. In this case, the stack pointer points to a former address rather than that of the system without disk ROM. Because of this fact, there is software that does not work on the system with built in disk drive. Although this trouble can be avoided by starting the system with shift key pressed to disable disk system, it is recommended that, you initilize the stack pointer at the begining of your software.

8.0 HOW TO RETURN TO THE BASIC

To return to the BASIC interpreter from the application software, do the following steps. Major work area of the BASIC should be kept unchanged. (If you don't know which is the major and which is the minor, don't touch any of the BASIC work area.)
The contents of all the registers and stack are ignored.

1. Enable the MAIN-ROM slot. The slot address of the MAIN-ROM is stored in the RAM location FCC1H. (See 3.0 The slot address of

the BASIC-ROM.)

2. Jump to the location 409BH in the MAIN-ROM.

The prompt "Ok" (or in MSX-2, user defined prompt string) will be displayed.

9.0 ESCAPE SEQUENCES OF MSX

MSX supports the escape sequences listed on the table in appendix B. These functions are available for the PRINT statement of BASIC, CHPUT BIOS routine, CONOUT of MSX-DOS direct BIOS call and CONSOL OUTPUT of MSX-DOS function call.
These are subset of the DEC VT-52 terminal or HEATH H-19 terminal.

10.0 WORK SPACE OF THE DISK SYSTEM

The size of the work space of the disk system varies depending on the number and capacity of the drives. The top address of the free area for the application softwares under DISK-BASIC environment is (HIMEM)-1. (HIMEM) is a value stored in the address FC4AH(HIGH) and FC4BH(LOW).
So far, the disk system that requires the largest work space is the system with two 2DD drives. And the lowest address of the work area in this case is around DE70H.
So, with some allowance, let's make the top address of user area to DE3FH.
However, there is still a possibility that there may be a system with larger system work space. So, every application softwares must check the address stored in the HIMEM (FC4AH, FC4BH) and make sure that even in the worst case, the system does not crash. The recommended ways when the system uses more work area than the application software expects are:

1. Make the work area relocatable so that it can be located anywhere.
2. Allocate the work area from BOTTOM. This may be a good way because all the MSX2 machine has RAM from address 8000H.
3. Direct the user to reboot the system with fewer disk drives. (Refer to the next section.)
4. Halt the program after displaying the message, "Insufficient work space."

11.0 HOW TO REDUCE THE DISK DRIVES

Pressing the 'shift' key until beep sounds after the system reset (or power on) makes all the disk drives disabled. Useful when the application software does not work with the disk connected. Similarly, the 'control' key disables the two drive simulator of single drive

system and the work area of the disk system become smaller.

12.0 HOW TO KNOW IF THE DISK SYSTEM IS CONNECTED OR NOT

Check the contents of the RAM whose address is FFA7H. If 'C9H' is stored, no disk is connected, otherwise disk system is connected and initialized.

After the disk system is initialized, following address contains some informations available for the applications. If the system has less disk interfaces than four, the rest of table entries are filled with zero.

Note that the contents of this table is not initialized if no disk interface exits, so make sure there is a disk interface is connected as described before.

address	contents
FB21H	Number of drives connected to the first interface
FB22H	The slot address of the first interface
FB23H	Number of drives connected to the second interface
FB24H	The slot address of the second interface
FB25H	Number of drives connected to the 3rd interface
FB26H	The slot address of the 3rd interface
FB27H	Number of drives connected to the 4th interface
FB28H	The slot address of the 4th interface

13.0 AUTOMATIC EXECUTION OF APPLICATION SOFTWARE

For the simple application softwares like games, put start address of the software in the 'INIT' location in the ROM ID area. However, in this way, no other system softwares such as disk/RS-232C may not be initialized.

For those applications that need to have all other system softwares initialized, put inter-slot call instruction to the start address at address FEDAH. This is a hook that is called after all the system software is initialized. This method is available on the system without disk. Please refer to 'MSX-DOS BOOT PROCEDURE' in the 'MSX Technical Data Book'.

14.0 DISK ERROR HANDLING BY APPLICATION SOFTWARE

An application software may handle the disk errors. The two byte value stored in RAM whose address starting from F323H is a pointer to the pointer of the disk error handler. Change those 2 bytes so that it points the pointer to the error handler in the application software.

The kind of error is passed through register 'C' and the driver number is passed through register 'A'. The LSB of register 'C' is zero if the error occurred during the read operation and 1 if the error occurred during the write operation. Bit 1 through 3 of the register 'C' represent the error status as follows:

b3	b2	b1	Kind of error
0	0	0	Write protected
0	0	1	Not ready
0	1	0	CRC error
0	1	1	Seek error
1	0	0	Record not found
1	0	1	Write error
1	1	0	Other error

The returned value from the error handling routine determine the action taken by the DOS as follows. The contents of register 'C' and 'SP' must be kept unchanged. Other registers may be destroyed.

C	Action after the error
2	Abort
1	Retry
0	Ignore

15.0 DOS FUNCTION CALL

The MSX-DOS function call is available under the Disk Basic environment. The RAM address F37DH is the entry for DOS function call which is equivalent to the address 5 in the MSX-DOS environment.

Refer to the 'MSX-DOS SYSTEM CALL' section in the 'MSX Technical Data Book'

Note that data transfer function, i.e. disk read/write to/from page 1 of memory is not supported.

SOME IMPORTANT LOCATIONS IN MSX2

Address	contents
0006H	The I/O address of VDP read port
0007H	The I/O address of VDP write port
002DH	The version number of MSX
409BH	BASIC interpreter warm start entry
F323H	Entry to disk error handling routine
FAF8H	Slot address of SUB-ROM
FB21H	Informations of disk drivers
FC4AH	The beginning address of the system work area
FCC1H	Slot address of MAIN-ROM
FFA7H	Disk is connected if the contents of this address is not 'C9'
FEDAH	Hook for auto start
FFFFH	Slot select register for the secondary slot

ESCAPE SEQUENCES SUPPORTED BY MSX

MSX supports following escape sequences which is a subset of H19 (an upward compatible terminal of VT52).

B.1 CURSOR FUNCTIONS

```
<ESC>A                          ;Cursor up
<ESC>B                          ;Cursor down
<ESC>C                          ;Cursor right
<ESC>D                          ;Cursor left
<ESC>H                          ;Cursor home
<ESC>Y<row+20H><column+20H>     ;Locate cursor
```

B.2 ERASING AND EDITING

```
<ESC>j                          ;Clear screen
<ESC>E                          ;Clear screen
<ESC>K                          ;Erase to end-of-line
<ESC>J                          ;Erase to end-of-screen
<ESC>l                          ;Erase entire line
<ESC>L                          ;Insert a line
<ESC>M                          ;Delete a line
```

B.3 CONFIGURATION

```
<ESC>x4                         ;Set block cursor
<ESC>x5                         ;Set cursor off
<ESC>y4                         ;Set underscore cursor
<ESC>y5                         ;Set cursor on
```

HOW TO USE ZSID ON MSX-DOS

The patch information for ZSID to use under MSX-DOS

Followings are addresses and data of the patches to the ZSID.COM version 1.4 which is loaded from 100H.

Address	Original	Patch
1100	38	28
1106	39	29
13B1	FF	EF
13FE	FF	EF
1F8D	FF	EF

Original version of ZSID uses "RST 38H" instruction for break pointing, however MSX-DOS uses "RST 38H" for hardware interrupt. Above patches make ZSID to use "RST 28H" instead "RST 38H".

SAMPLE MACHINE CODE PROGRAMS

PROGRAM 1 : ENABLE SLOT FOR APPLICATION PROGRAMS

```
;       Suppose your program cartridge is 32K bytes
;       long (4000H..0BFFFH).  You set the ID at 4000H
;       and 4001H and the execution start address within
;       page 1  (4000H..7FFFH).  MSX passes control
;       to this address so the part which resides in
;       page 1 is not yet enabled at this point. You
;       have to know where you are (in what primary
;       slot, in what secondary slot) and enable the
;       part at page 1. Below is the sample program
;       to do this.
;
.Z80
ENASLT  EQU     0024H           ;enable slot
RSLREG  EQU     0138H           ;read   primary   slot   select
                        ;register
EXPTBL  EQU     0FCC1H          ;slot is expanded or not
;
;
ENAP2:
      CALL    RSLREG          ;Read primary slot #
      RRCA                    ;Move it to bit 0,1 of [Acc]
      RRCA
      AND     00000011B
      LD      C,A
      LD      B,0
      LD      HL,EXPTBL       ;See if  this  slot  is expanded
                        ;or not
      ADD     HL,BC
      LD      C,A             ;Save primary slot #
      LD      A,(HL)          ;Get the slot is expanded or not
      AND     80H
      OR      C               ;Set MSB if so
      LD      C,A             ;Save it to [C]
      INC     HL              ;Point to SLTTBL entry
      INC     HL
      INC     HL
      INC     HL
      LD      A,(HL)          ;Get what  is  currently  output
                        ;to expansion   slot    register
      AND     00001100B
      OR      C               ;Finaly form slot address
      LD      H,80H
      JP      ENASLT          ;enable page 2
;
END
```

PROGRAM 2 : SAMPLE PROGRAM TO USE HOOK

```
;=======================
; Start-up initialize entry
; This procedure will be called when system initializing.
;
      .Z80
H.KEYI  EQU     0FD9AH          ; interrupt hook
EXPTBL  EQU     0FCC1H          ; slots expanded or not
PSLTRG  EQU     0A8H            ; I/O port address of primary slot
register
EXT     MYINT                   ; my interrrupt handler
;
      CSEG
INIT:
; Please insert other initialize routine here, if you need.

; Set interrupt entry

      DI                        ; start of critical region

; Get old interrupt entry inter-slot call hook

      LD      DE,OLDINT         ; set address of old interrupt hook saved
area
      LD      HL,H.KEYI         ; set address of interrupt entry hook
      LD      BC,5              ; length of hook is 5 bytes
      LDIR                      ; transfer

; What slot address is this cartridge placed

      CALL    GTMSLT            ; get my slot address

; Set new inter-slot call of interrupt entry

      LD      (H.KEYI+1),A      ; set slot address
      LD      A,0F7H            ; `RST 30H' inter-slot call operation code
      LD      (H.KEYI),A        ; set new hook op-code
      LD      HL,INTENT         ; get our interrupt entry point
      LD      (H.KEYI+2),HL     ; set new interrupt entry point
      LD      A,0C9H            ; `RET' operation code
      LD      (H.KEYI+4),A      ; set operation code of `RET'
      EI                        ; end of critical region
      RET
```

```
;=======================
; What slot address is this cartridge placed
;  Entry: No
;  Action: Compute my slot address
;  Return: A = slot address
;  Modify: Flag

GTMSLT:
      PUSH    BC              ; save environment
      PUSH    HL
      IN      A,(PSLTRG)      ; read primary slot register
      RRCA                    ; move it to bit 0,1 of A
      RRCA
      AND     00000011B       ; get bit 1,0
      LD      C,A             ; set primary slot No.
      LD      B,0
      LD      HL,EXPTBL       ; see is this slot is expanded or not
      ADD     HL,BC
      OR      (HL)            ; set MSB if so
      LD      C,A
      INC     HL              ; point to SLTTBL entry
      INC     HL
      INC     HL
      INC     HL
      LD      A,(HL)          ; get what is currently output to
expansion
                        ; slot register

      AND     00001100B       ; get bit 3,2
      OR      C               ; finely form slot address

      POP     HL              ; restore environment
      POP     BC
      RET                     ; return to main

;=======================
; Interrupt entry
;
INTENT:
      CALL    MYINT           ; call interrupt handler
      JP      OLDINT          ; go old interrupt handler

;=======================
;       HOOK save area
;
      DSEG
OLDINT: DS      5

      END
```

PROGRAM 3 : RS232 TEST

```
;*******************************************
;***                                     ***
;***  Beginning of machine-dependent code  ***
;***                                     ***
;*******************************************

      .Z80

      ENTRY   SYSINI,RS2INZ,RS2IN,RS2OUT,TERMIN


.COMMENT %

      --
       ~
      LD      DE,NEXT
      JP      SYSINI          ;get rs232
NEXT:   CALL    RS2INZ          ;initialize rs232
      --
      ~
      CALL    RS2IN           ;read from rs232

      CALL    RS2OUT          ;write to rs232
      ~
      ~
      JP      TERMIN          ;terminate
      %

ABORT   EQU     0               ;abort system
CALSLT  EQU     01CH            ;inter slot call
ESC     EQU     01BH
EXTBIO  EQU     0FFCAH          ;extended bios call
FINIT   EQU     3               ;initialize rs-device
FOPEN   EQU     6               ;open rs-device
FCLOSE  EQU     18              ;close rs-device
FGETCH  EQU     12              ;get one character from rs-device
FPUTCH  EQU     15              ;put one character to rs-device
FLOC    EQU     24              ;get receive buffer condition
MAXBUF  EQU     254             ;input buffer length
RAWMOD  EQU     4               ;open mode < raw mode >
SLTREG  EQU     0A8H            ;slot register number
BUFSIZ  EQU     MAXBUF*2+10+40
;
;
; ENTRY:
;       DE -> RETURN ADDRESS
;
```

```
SYSINI: LD      HL,(6)          ;get top of MSX DOS
        LD      BC,BUFSIZ+512
        SBC     HL,BC
        LD      BC,08000H       ;check stack area
        XOR     A               ;must be stayed higher address than 08000H
        SBC     HL,BC
        JP      C,ERROR3        ;if error abort
        LD      HL,(6)          ;get top of MSX DOS
        LD      BC,BUFSIZ
        SBC     HL,BC
        LD      SP,HL           ;set stack pointer
        LD      (HLSAV),HL
        PUSH    DE              ;set return address
        INC     HL
        LD      (DEVTBL),HL     ;save device table address
        LD      DE,40
        ADD     HL,DE           ;get fcb address for rs232c
        LD      (RSFCB),HL      ;save it
        RET
;
GETSLT: IN      A,(SLTREG)      ;get current slot table
        INC     B               ;adjust high address
RS2IZ0: DEC     B               ;count down
        RET     Z
        SRL     A
        SRL     A               ;shift slot data
        JR      RS2IZ0
;
FUNCAL: PUSH    AF
        PUSH    DE
        LD      A,(FUNSLT)      ;get slot number
        PUSH    AF
        POP     IY
        LD      DE,(DEVTBL)
        ADD     IX,DE           ;get open address
        POP     DE
        POP     AF
        CALL    CALSLT
        EI
        RET
;
```

```
RS2INZ: LD      HL,(DEVTBL)
        LD      A,H
        RLCA
        RLCA
        AND     3
        LD      B,A
        CALL    GETSLT
        AND     3
        LD      B,A
        LD      HL,(DEVTBL)
        LD      D,8             ;select RC232C DEVICE
        LD      E,0
        PUSH    HL              ;save work address
        CALL    EXTBIO
        POP     DE
        XOR     A
        SBC     HL,DE
        JR      Z,ERROR1        ;branch if do not exist RS-cord
        LD      HL,(DEVTBL)
        LD      A,(HL)
        INC     HL
        LD      E,(HL)
        INC     HL
        LD      H,(HL)
        LD      L,E
        LD      (DEVTBL),HL
        LD      (FUNSLT),A      ;get rs232c driver slot
        CALL    RSINIT
        CALL    RSOPEN
        LD      IY,0            ;display function keys
        LD      IX,0CFH
        CALL    CALSLT
        EI
        RET
;
ERROR3: LD      DE,ERRMG3
        JR      ERRJOB
;
ERROR2: LD      DE,ERRMG2
        JR      ERRJOB
;
ERROR1: LD      DE,ERRMSG
ERRJOB: LD      C,9             ;display error message
        CALL    5
        JP      ABORT           ;go MSX DOS
;
```

```
ERRMSG: DB      'RS-DEVICE DO NOT EXIST',0DH,0AH,'$'
ERRMG2: DB      'RS-DEVICE CAN NOT OPEN',0DH,0AH,'$'
ERRMG3: DB      'DO NOT RESERVE STACK AREA',0DH,0AH,'$'
;
RSOPEN: LD      IX,FOPEN
      LD      HL,(RSFCB)      ;set fcb address
      LD      C,MAXBUF        ;set maximum buffer length
      LD      E,RAWMOD        ;set open mode
      CALL    FUNCAL
      JR      C,ERROR2
      RET
;
RSINIT: LD      HL,PRMETR       ;get slot of parameter address
      LD      A,H             ;get high address
      RLCA
      RLCA
      AND     3
      LD      B,A
      CALL    GETSLT          ;get target slot number
      AND     3
      LD      B,A
      LD      IX,FINIT
      LD      HL,PRMETR       ;parameter address
      CALL    FUNCAL
      JP      C,ERROR2
      RET
;
RS2IN:  PUSH    BC
      PUSH    DE
      PUSH    HL
      PUSH    IX
      PUSH    IY
      LD      IX,FLOC
      CALL    FUNCAL
      LD      A,H
      OR      L
      SCF
      JR      Z,RSRETN
      LD      IX,FGETCH
      CALL    FUNCAL
      JR      NC,RS2IN0       ;check error
      LD      A,3FH           ;set '?' if error
RS2IN0: LD      HL,MASK
      AND     (HL)
      JR      RSRETN
;
```

```
RS2OUT: PUSH    BC
        PUSH    DE
        PUSH    HL
        PUSH    IX
        PUSH    IY
        PUSH    AF
        LD      IX,FPUTCH
        CALL    FUNCAL
ALLRET: POP     AF
RSRETN: POP     IY
        POP     IX
        POP     HL
        POP     DE
        POP     BC
        RET
;
TERMIN: LD      IX,FCLOSE
        CALL    FUNCAL
        JP      ABORT           ;exit to MSX DOS
;
PRMETR: DEFB    '8N1XNNNN'
        DEFW    9600
        DEFW    9600
        DEFB    0
;
PRMTR2: DEFB    '8N1NNNNN'
        DEFW    9600
        DEFW    9600
        DEFB    0
FUNSLT: DEFB    1
DEVTBL: DEFW    0
RSFCB:  DEFW    0
HLSAV:  DEFW    0
MASK:   DEFB    7FH

        END
```

```
PROGRAM 4 : WHERE AM I
;       To know where you are;
;       This  routine  returns  the  slot address of the
;       following format in [Acc].
;           FxxxSSPP
;           |   ||||
;           |   ||++-- primary slot # (0-3)
;           |   ++---- secondary slot # (0-3)
;           +--------- 1 if secondary slot # specified
;       This value can later  be used as an input parameter
;       for  the RDSLT, WRSLT, CALSLT, ENASLT and 'RST 30H'
;
      .Z80
      CSEG
;
      RSLREG  EQU     138H
      EXPTBL  EQU     0FCC1H
      B8000   EQU     0       ;set this  to  non  zero  if the program
                        ;resides at 8000..0BFFFH
WHERE_AM_I:

      CALL    RSLREG          ;read primary slot #
      RRCA                    ;move it to bit 0,1 of [Acc]
      RRCA
      IF      B8000
      RRCA
      RRCA
      ENDIF
      AND     11B
      LD      C,A
      LD      B,0
      LD      HL,EXPTBL       ;see if  this  slot  is  expanded or not
      ADD     HL,BC
      LD      C,A
      LD      A,(HL)
      AND     80H
      OR      (HL)            ;set MSB if so
      LD      C,A             ;save primary slot number
      INC     HL              ;point to SLTTBL entry
      INC     HL
      INC     HL
      INC     HL
      LD      A,(HL)          ;get   what   is   currently  output  to
                        ;expansion slot register
      IF      B8000
      RRCA                    ;move it to bit 2,3 of [Acc]
      RRCA
      ENDIF
      AND     1100B
      OR      C               ;finally form slot address
      RET

      END
```

PROGRAM 5 : WORK AREA FOR CARTRIDGE SOFTWARE

```
; How to allocate work area for cartridges
;
; If the work area is greater than 2 bytes, make the SLTWRK
; point to the system variable BOTTOM (0FC48H), then update
; it by the amount of memory required. BOTTOM is set up by
; the initialization code to point to the bottom of equipped
; RAM.
;          Ex. if the program is at 4000H..7FFFH.
      .Z80
      CSEG
;
      EXT     SIZE            ;Size of memory required
      EXT     NOROOM          ;Called if out of memory
      RSLREG  EQU     138H
      EXPTBL  EQU     0FCC1H
      BOTTOM  EQU     0FC48H
      SLTWRK  EQU     0FD09H
;
      CALL    RSLREG          ;Read primary slot #
      RRCA                    ;Move it to bit 0,1
      RRCA                    ;of [Acc]
      AND     00000011B
      LD      C,A
      LD      B,0
      LD      HL,EXPTBL       ;See if this slot is
      ADD     HL,BC           ;expanded or not
      ADD     A,A
      ADD     A,A
      ADD     A,A
      ADD     A,A
      LD      C,A
      LD      A,(HL)
      ADD     A,A
      SBC     A,A             ;Form mask pattern
      AND     00001100B
      INC     HL              ;Point to SLTTBL entry
      INC     HL
      INC     HL
      INC     HL
      AND     (HL)            ;Get what is currently
                      ;output to     expansion
                      ;slot register
      OR      C
      OR      00000001B
      PAGE
;
```

```
;       Now, we have the sequence number for this
;       cartridge as follows.
;
;       00PPSSBB
;         ||||||
;         ||||++-- higher 2 bits of memory address
;         ||++---- secondary slot # (0..3)
;         ++------ primary slot # (0..3)
;
        ADD     A,A             ;Double since word table
        LD      C,A
        LD      B,0
        LD      HL,SLTWRK       ;Point to   entry   in
        ADD     HL,BC           ;SLTWRK table
        LD      BC,(BOTTOM)     ;Get current RAM bottom
        LD      (HL),C          ;Register this
        INC     HL
        LD      (HL),B
        LD      HL,SIZE
        ADD     HL,BC
        LD      A,H             ;Beyond 0EFFFH?
        CP      0F0H
        JP      NC,NOROOM       ;Yes, cannot allocate this much
        LD      (BOTTOM),HL
        RET
;
;       BOTTOM became greater than 0EFFFH, there is
;       no RAM left to be allocated.
;
;NOROOM: ;Print  messages or something

        END
```