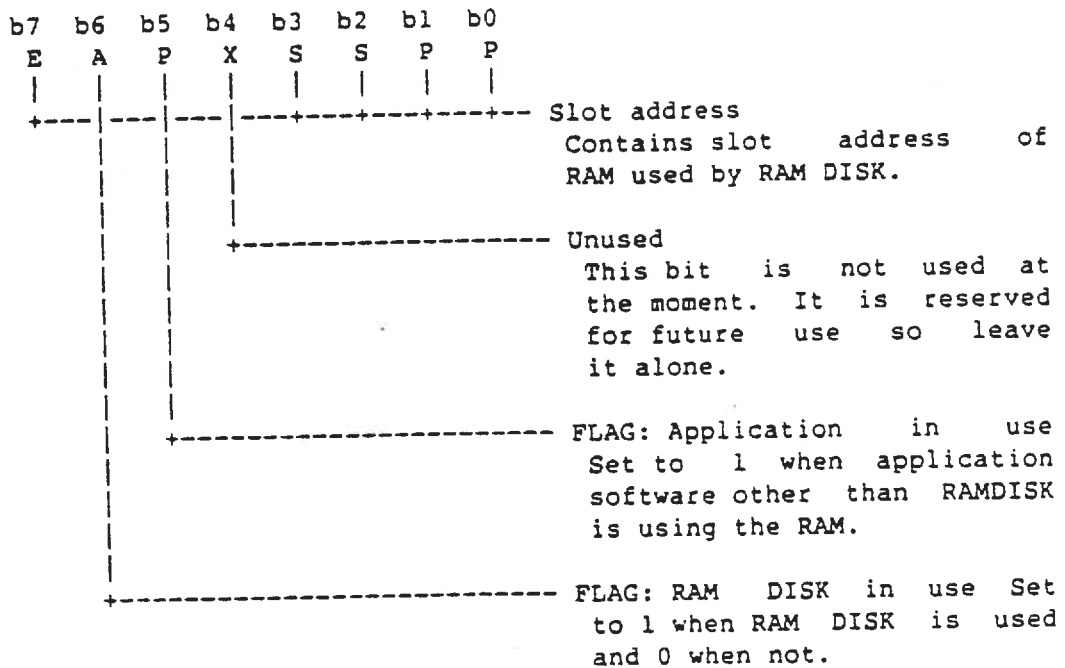


## HOW TO USE THE RAM IN PAGE 0 AND 1 ON MSX2

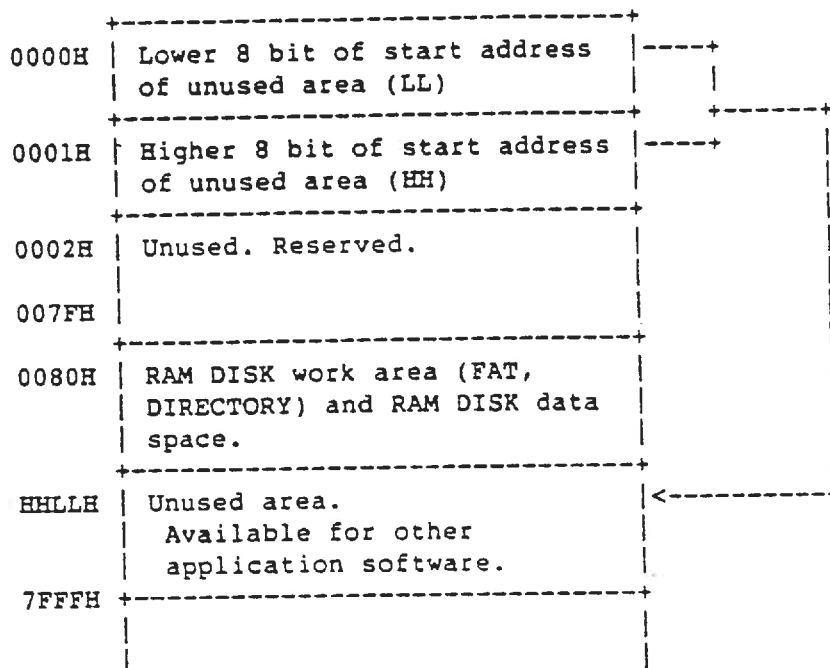
## A.1 CONTENTS OF SLTWRK

In MSX2, SLTWRK uses 1 byte which is for slot 0-0, page 0. The address is FD09H and its contents is as follows:



## A.2 RAM AREA (0000H-7FFFH)

Address 0000H and 0001H contain the RAMDISK's end address+1. It is used to find where application software can use the RAM.



A.3 HOW TO USE RAM BETWEEN 0000H AND 7FFFFH WHICH IS NOT USED BY RAMDISK

- (1) When SLTWRK (FD09H) is all zero (0).  
The RAM has not been used so by changing bits P(b0, b1) to 1, application program is able to use all the area. The application must find the slot address because the slot address of RAM is unknown in this stage. (A sample program to search the RAM slot address is attached.)
- (2) When P bits in SLTWRK are 1.  
The RAM is used by other application software, so you cannot use it.
- (3) When P bits are 0 and A bit is 1  
RAMDISK is using the RAM. The address of usable RAM below can be found in the location 0000H and 0001H.

```

                .Z80
:               CSEG
FIRSTINI:      LD      HL,EXPTBL
                LD      BC,256*4      :total number of primary slots
                XOR     A              :start with 0.0
C64K05:        AND     11B            :leave only primary slot bits
                OR      (HL)          :set MSB if slot expanded
C64K07:        PUSH    BC            :save slot counter, page address
                PUSH    HL           :save pointer to EXPTBL
                LD      H,C          :set page address to check
C64K10:        LD      L,10H
C64K20:        PUSH    AF            :save slot address
                CALL    RDSLT        :read a byte
                CPL
                LD      E,A
                POP     AF            :restore slot address
                PUSH    DE
                PUSH    AF            :save slot address
                CALL    WRSLT        :write complement
                POP     AF
                POP     DE
                PUSH    AF
                PUSH    DE
                CALL    RDSLT        :read it back
                POP     BC
                LD      B,A
                LD      A,C
                CPL
                LD      E,A
                POP     AF
                PUSH    AF
                PUSH    BC
                CALL    WRSLT        :restore old value
                POP     BC
                LD      A,C          :same?
                CP      B
                JR      NZ,C64K30    :no. RAM not equipped
                POP     AF            :restore slot address
                DEC     L
                JR      NZ,C64K20
                INC     H            :bump to next 1K block
                INC     H
                INC     H
                INC     H
                JP      P,C64K10     :check more

```

Here when RAM disk has been found

```

                POP     HL            :discard stack

```

```

      POP      HL
      LD       (DISK_SLOT).A
      JR       CONFIRM

```

```

:
: Here when RAM disk not found in the target slot
:

```

```

C64K30:
      POP      AF           ;restore slot address
      POP      HL           ;restore pointer to EXPTBL
      POP      BC           ;restore counter and start address
      AND      A            ;are we checking expanded slot?
      JP       P.C64K40     ;no

```

```

:
: Here to advance to next secondary slot
:

```

```

      ADD      A.00000100B   ;next expanded slot #
      CP       10010000B     ;more slots?
      JR       C.C64K07      ;yes

```

```

:
: Here to advance to next primary slot
:

```

```

C64K40:
      INC      HL           ;bump pointer to EXPTBL
      INC      A            ;bump slot address
      DJNZ     C64K05

```

```

JJFCERR:
      JP       JFCERR       ;we have no RAM for RAM disk

```

```

:
CONFIRM:
:

```

```

: Check if we should make the RAM disk off line
:

```

```

CHKSIZE:
      POP      HL           ;restore upper limit
      XOR      A
      LD       DE,DATATOP+OFFH
      SBC      HL,DE
      JR       C.OFFDISK    ;make the RAM disk off line

```

```

      ~~~~~
      ~~~~~
      RET

```

```

:
OFFDISK:

```

```

      XOR      A
      LD       (DISK_SLOT).A

```

```

      ~~~~~
      ~~~~~
      RET

```

```

:
END
:

```

**MSX2 BIOS functions to copy graphic data**

ASCII corporation MSFE HQ.

1st edition 8th Aug., 1985  
2nd 10th Dec., 1985

## 1. VRAM to VRAM

Name: BLTVV  
 Address: 0191H in SUBROM jump table  
 Function: Copies VRAM to VRAM  
 Entry: [HL] = 0F562H and following data

```

      ORG 0F562H
SX:   DS 2      ;source horizontal position
SY:   DS 2      ;source vertical position
DY:   DS 2      ;destination horizontal position
DY:   DS 2      ;destination vertical position
NX:   DS 2      ;number of horizontal pixels
NY:   DS 2      ;number of vertical pixels
CDUMMY: DS 1    ;dummy for OTIR (you don't need to set this)
ARG:   DS 1    ;direction and extended VRAM select (same as VDP
              ; register #45)
L_OP:  DS 1    ;logical operation code (same as VDP logical
              ; operation code)

```

Returns: carry flag is reset  
 Modifies: All

## Example:

; COPY (0,0)-(255,99) to (0,100)

```

BLTVV EQU 0191H
EXTROM EQU 015FH

LD HL,0
LD (SX),HL      ;source coordinate is (0,0)
LD (SY),HL
LD (DX),HL      ;destination coordinate is (0,100)
LD L,100
LD (DY),HL
LD HL,255-0+1   ;set width
LD (NX),HL
LD HL,99-0+1    ;set height
LD (NY),HL
XOR A           ;make direction
LD (ARG),A
LD (L_OP),A     ;operation is 'replace'
;
LD HL,0F562H
LD IX,BLTVV
CALL EXTROM
.
.

```

## 2. Main RAM and VRAM

The pixel data memory area must be defined as follows:

If you use screen mode 6

```
PXLDAT: ;any RAM address
        DW      HRZPXL      ;number of horizontal pixels
        DW      VRTPIXL     ;number of vertical pixels
        DS      (HRZPXL/4*VRTPIXL+1) ;VRAM data
```

If you use screen mode 5 or 7

```
PXLDAT: ;any RAM address
        DW      HRZPXL      ;number of horizontal pixels
        DW      VRTPIXL     ;number of vertical pixels
        DS      (HRZPXL/2*VRTPIXL+1) ;VRAM data
```

If you use screen mode 8

```
PXLDAT: ;any RAM address
        DW      HRZPXL      ;number of horizontal pixels
        DW      VRTPIXL     ;number of vertical pixels
        DS      (HRZPXL*VRTPIXL) ;VRAM data
```

## 2.1 Main RAM to VRAM

Name: BLTVM  
 Address: 0195H in SUBROM  
 Function: Copies array to VRAM  
 Entry: [HL] = 0F562H and following data

```
        ORG      0F562H

DPTR:   DS      2      ;source pointer for main memory
DUMMY:  DS      2      ;dummy for OTIR (you don't need to set this)
DX:     DS      2      ;destination horizontal position
DY:     DS      2      ;destination vertical position
NX:     DS      2      ;number of horizontal pixel (you don't need to
                    ; set this. The main RAM has this data.)
NY:     DS      2      ;number of vertical pixel (you don't need to
                    ; set this. The main RAM has this data.)
CDUMMY: DS      1      ;dummy for OTIR (you don't need to set this)
ARG:    DS      1      ;direction and extended VRAM select (same as VDP
                    ; register #45)
L_OP:   DS      1      ;logical operation code (same as VDP logical
                    ; operation code)
```

Returns: carry flag is set if the main RAM has bad number  
 of pixels  
 Modifies: All

Example:

; COPY &lt;B000H&gt; to (5,6)

```
BLTVM EQU 0195H
EXTROM EQU 015FH
```

```
LD HL,0B000H
LD (DPTR),HL ;source data is placed at B000H
LD HL,5
LD (DX),HL ;destination coordinate is (5,6)
INC HL
LD (DY),HL
XOR A ;make direction
LD (ARG),A
LD (L_OP),A ;operation is 'replace'

LD HL,0F562H
LD IX,BLTVM
CALL EXTROM
```

## 2.2 VRAM to Main RAM

```
Name: BLTMV
Address: 0199H in SUBROM
Function: Copies VRAM to array
Entry: [HL] = 0F562H and following data
```

```
ORG 0F562H
SX: DS 2 ;source horizontal position
SY: DS 2 ;source vertical position
DPTR: DS 2 ;destination memory address
DUMMY: DS 2 ;dummy for OTIR (you don't need to set this)
NX: DS 2 ;number of horizontal pixel
NY: DS 2 ;number of vertical pixel
CDUMMY: DB 1 ;dummy for OTIR (you don't need to set this)
ARG: DB 1 ;direction and extended VRAM select (same as
; VDP register #45)
```

```
Returns: carry flag is reset
Modifies: All
```

Example:

; COPY (10,20)-(50,25) TO &lt;B000H&gt;

```
BLTMV EQU 0199H
EXTROM EQU 015FH
```

```
LD HL,0B000H
LD (DPTR),HL ;source data is placed at B000H
LD HL,10
LD (SX),HL ;source coordinate is (10,20)
LD HL,20
LD (SY),HL
LD HL,50-10+1 ;set width
LD (NX),HL
LD HL,25-20+1 ;set height
LD (NY),HL
XOR A ;make direction
LD (ARG),A

LD HL,0F562H
LD IX,BLTMV
CALL EXTROM
```

.

.

## 3. DISK and VRAM

These entries (BLTV D, BLTDV, BLTMD, BLTDM) need file name to save/load data to/from disk. The file name must be set as follows:

```

      .
      .
      .
LD     HL,FILENAME      ;get pointer to file name
LD     (FNPTR),HL       ;set it to parameter area
      .
      .
      .

```

```

FILENAME:      DB      22H,'TEST.PIC',22H,0      ;"TEST.PIC", end mark
;
; Tha file name specification is same as BASIC interpreter's.
;      "d:xxxxxxxx.yyy"
;      d: = drive number (may be excepted)
;      xxxxxxxx = file name
;      yyy = extension
;

```

If any error is occurred, then this BIOS jumps the BASIC interpreter error handler. So, if you want to handle error or you use this BIOS in MSX-DOS or from ROM cartridge, you must set a hook of the error handler of the BASIC interpreter. The name of hook is 'H.ERRO', and the address is FFB1H. When the BASIC interpreter calls the hook, the stack pointer (SP) is changed from when you called these BIOS.

## 3.1 DISK to VRAM

Name: BLTVD  
 Address: 019DH in SUBROM  
 Function: Copies DISK file to VRAM  
 Entry : [HL] = 0F562H and following data

ORG 0F562H

```

FNPTR: DS      2      ;pointer to file name string
DUMMY: DS      2      ;dummy (you don't need to set this)
DX:     DS      2      ;destination horizontal position
DY:     DS      2      ;destination vertical position
NX:     DS      2      ;number of horizontal pixel (you don't need to
                        ; set this. The memory data has this data.)
NY:     DS      2      ;number of vertical pixel (you don't need to
                        ; set this. The memory data has this data.)
CDUMMY: DS      1      ;dummy for OTIR (you don't need to set this)
ARG:    DS      1      ;direction and extended VRAM select (same as VDP
                        ; register #45)
L_OP:   DS      1      ;logical operation code (same as VDP logical
                        ; operation code)

```

Return: carry flag is set if parameter error is occurred  
 Modify: All

Example:  
 ; COPY "TEST.PIC" TO (3,2)

```

BLTVD EQU 019DH
EXTROM EQU 015FH

```

```

LD      HL,FILENAME
LD      (FNPTR),HL      ;set pointer to file name
LD      HL,3
LD      (DX),HL          ;destination coordinate is (3,2)
DEC     HL
LD      (DY),HL
XOR     A                ;make direction
LD      (ARG),A
LD      (L_OP),A         ;operation is 'replace'
;
LD      HL,0F562H
LD      IX,BLTVD
CALL    EXTROM

```

```

FILENAME: DB 22H,'TEST.PIC',22H,0

```

## 3.2 VRAM to DISK

Name: BLTDV  
 Address: 01A1H in SUBROM  
 Function: Copies VRAM to DISK file  
 Entry : [HL] = 0F562H and following data

```

      ORG      0F562H

SX:    DS      2      ;source horizontal position
SY:    DS      2      ;source vertical position
FNPTR: DS      2      ;destination pointer to file name string
DUMMY: DS      2      ;dummy for OTIR (you don't need to set this)
NX:    DS      2      ;number of horizontal pixel
NY:    DS      2      ;number of vertical pixel
CDUMMY: DS      1      ;dummy for OTIR (you don't need to set this)
ARG:   DS      1      ;direction and extended VRAM select (same as VDP
                        ; register #45)

```

Return: carry flag is reset  
 Modify: All

Example:  
 ; COPY (10,20)-(50,25) TO "A:TEST.PIC"

```

BLTDV EQU 01A1H
EXTROM EQU 015FH

      LD      HL,FILENAME
      LD      (FNPTR),HL      ;set pointer to file name
      LD      HL,10
      LD      (SX),HL         ;source coordinate is (10,20)
      LD      HL,20
      LD      (SY),HL
      LD      HL,50-10+1      ;set width
      LD      (NX),HL
      LD      HL,25-20+1      ;set height
      LD      (NY),HL
      XOR     A               ;make direction
      LD      (ARG),A
      ;
      LD      HL,0F562H
      LD      IX,BLTDV
      CALL    EXTROM

```

```

      .
      .
FILENAME:DB 22H,'A:TEST.PIC',22H,0
      .

```

## 4. Main RAM and DISK

## 4.1 DISK to Main RAM

Name: BLTMD  
 Address: 01A5H in SUBROM  
 Function: Loads array data from DISK file  
 Entry: [HL] = 0F562H and following data

```

      ORG      0F562H
FNPTR: DS      2      ;pointer to file name string
SY:      DS      2      ;dummy (you don't need to set this)
SPTR:    DS      2      ;start address to load
EPTR:    DS      2      ;end address to load
  
```

Returns: carry flag is reset  
 Modifies: All

## Example:

```

BLTMD EQU      01A5H
EXTROM EQU     015FH
  
```

```

      LD      HL,FILENAME      ;set pointer to file name
      LD      (FNPTR),HL
      LD      HL,8000H         ;set start address
      LD      (SPTR),HL
      LD      HL,0AFFFH        ;set end address
      LD      (EPTR),HL
      ;
      LD      HL,0F562H
      LD      IX,BLTMD
      CALL    EXTROM
      .
      .
FILENAME:DB  22H,'TEST.BIN',22H,0
      .
      .
  
```

## 4.2 Main RAM to DISK

Name: BLTDM  
 Address: 01A9H in SUBROM  
 Function: Saves array data to DISK file  
 Entry : [HL] = 0F562H and following data

```

      ORG 0F562H
SPTR  DS 2      ;start address to save
EPTR: DS 2      ;end address to save
FNPTR: DS 2      ;pointer to file name string
  
```

Return: carry flag is reset  
 Modify: All

## Example:

```

BLTDM EQU 01A9H
EXTROM EQU 015FH
  
```

```

      LD HL,FILENAME      ;set pointer to file name
      LD (FNPTR),HL
      LD HL,8000H          ;set start address
      LD (SPTR),HL
      LD HL,0AFFFH         ;set end address
      LD (EPTR),HL
      ;
      LD HL,0F562H
      LD IX,BLTDM
      CALL EXTROM
  
```

```

FILENAME:DB 22H,'A:TEST.BIN',22H,0
  
```