MSX Assembly Page

MSX BIOS calls

This is an overview of all official MSX BIOS calls.

- MSX 1 BIOS (up to function call #159)
- MSX 2 BIOS (up to function call #177)
- MSX 2+ BIOS (up to function call #17D)
- MSX turbo R BIOS (up to function call #189)

MSX 1 BIOS Entries

RST-and other routines

CHKRAM (also called STARTUP, RESET or BOOT)

```
Address  : #0000
Function : Tests RAM and sets RAM slot for the system
Registers: All
Remark   : After this, a jump must be made to INIT, for further initialisation.
```

SYNCHR

```
Address  : #0008
Function : Tests whether the character of (HL) is the specified character
           if not, it generates SYNTAX ERROR, otherwise it goes to CHRGTR (#0010)
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)
Output   : HL is increased by one and A receives (HL), When the tested character is
           numerical, the carry flag is set the end of the statement (00h or 3Ah) causes
           the zero flag to be set
Registers: AF, HL
```

RDSLT

```
Address  : #000C
Function : Reads the value of an address in another slot
Input    : A  - ExxxSSPP  Slot-ID
           |         ||└└─ Primary slot number (00-11)
           |         └└─── Secondary slot number (00-11)
           └──────────────── Expanded slot (0 = no, 1 = yes)
           HL - Address to read
Output   : A  - Contains the value of the read address
Registers: AF, C, DE
Remark   : This routine turns off the interupt, but won't turn it on again
```

CHRGTR

```
Address  : #0010
Function : Gets the next character (or token) of the Basic text
Input    : HL - Address last character
Output   : HL - Points to the next character
           A  - Contains the character
           Carry flag set if it's a number
           Zero flag set if it's the end of the statement
Registers: AF, HL
```

WRSLT

```
Address  : #0014
Function : Writes a value to an address in another slot.
Input    : A  - Slot ID, see RDSLT
           HL - Address
           E  - Value
Registers: AF, BC, D
Remark   : See RDSLT
```

OUTDO

```
Address  : #0018
Function : Output to current output channel (printer, file, etc.)
Input    : A  - PRTFIL, PRTFLG
Remark   : Used in basic, in ML it's pretty difficult
```

CALSLT

```
Address  : #001C
Function : Executes inter-slot call.
Input    : IY - High byte with slot ID, see RDSLT
           IX - The address that will be called
Remark   : Variables can never be given in alternative registers or IX and IY
```

DCOMPR

```
Address  : #0020
Function : Compares HL with DE
Input    : HL, DE
Output   : Zero flag set if HL and DE are equal. Carry flag set if HL is less than DE.
Registers: AF
```

ENASLT

```
Address  : #0024
Function : Switches indicated slot at indicated page on perpetually
Input    : A - Slot ID, see RDSLT
           H - Bit 6 and 7 must contain the page number (00-11)
```

GETYPR

```
Address  : #0028
Function : Returns Type of DAC
Input    : DAC
Output   : S,Z,P/V, CY
Registers: AF
Remark   : Not a very clear routine to me, please mail us if you know more about it.
```

CALLF

```
Address  : #0030
Function : Executes an interslot call
Output   : Depends on the calling routine
Registers: AF, and the other registers depending on the calling routine
Remark   : The following is the calling sequence:
           RST #30
           DB destination slot ID, see RDSLT
           DW destination address
```

KEYINT

```
Address  : #0038
Function : Executes the timer interrupt process routine
```

Initialization-routines

INITIO

```
Address  : #003B
Function : Initialises the device
Registers: All
```

INIFNK

```
Address  : #003E
Function : Initialises the contents of the function keys
Registers: All
```

VDP routines

DISSCR

```
Address  : #0041
Function : Inhibits the screen display
Registers: AF, BC
```

ENASCR

```
Address  : #0044
Function : Displays the screen
Registers: AF, BC
```

WRTVDP

```
Address  : #0047
Function : Write data in the VDP-register
Input    : B  - Data to write
           C  - Number of the register
Registers: AF, BC
```

RDVRM

```
Address  : #004A
Function : Reads the content of VRAM
Input    : HL - Address read
Output   : A  - Value which was read
Registers: AF
```

WRTVRM

```
Address  : #004D
Function : Writes data in VRAM
Input    : HL - Address write
           A  - Value write
Registers: AF
```

SETRD

```
Address  : #0050
Function : Enable VDP to read
Input    : HL - For VRAM-address
Registers: AF
```

SETWRT

```
Address  : #0053
Function : Enable VDP to write
Input    : HL - Address
Registers: AF
```

FILVRM

```
Address  : #0056
Function : Fill VRAM with value
Input    : A  - Data byte
           BC - Length of the area to be written
           HL - Start address
Registers: AF, BC
```

LDIRMV

```
Address  : #0059
Function : Block transfer to memory from VRAM
Input    : BC - Block length
           DE - Start address of memory
           HL - Start address of VRAM
Registers: All
```

LDIRVM

```
Address  : #005C
Function : Block transfer to VRAM from memory
Input    : BC - Block length
           DE - Start address of VRAM
           HL - Start address of memory
Registers: All
```

CHGMOD

```
Address  : #005F
Function : Switches to given screen mode
Input    : A  - Screen mode
Registers: All
```

CHGCLR

```
Address  : #0062
Function : Changes the screen colors
Input    : Foreground color in FORCLR
           Background color in BAKCLR
           Border color in BDRCLR
Registers: All
```

NMI

```
Address  : #0066
Function : Executes non-maskable interupt handling routine
```

CLRSPR

```
Address  : #0069
Function : Initialises all sprites
Input    : SCRMOD
Registers: All
```

INITXT

```
Address  : #006C
Function : Switches to SCREEN 0 (text screen with 40×24 characters)
Input    : TXTNAM, TXTCGP
Registers: All
```

INIT32

```
Address  : #006F
Function : Switches to SCREEN 1 (text screen with 32×24 characters)
Input    : T32NAM, T32CGP, T32COL, T32ATR, T32PAT
Registers: All
```

INIGRP

```
Address  : #0072
Function : Switches to SCREEN 2 (high resolution screen with 256×192 pixels)
Input    : GRPNAM, GRPCGP, GRPCOL, GRPATR, GRPPAT
Registers: All
```

INIMLT

```
Address  : #0075
Function : Switches to SCREEN 3 (multi-color screen with 64×48 pixels)
Input    : MLTNAM, MLTCGP, MLTCOL, MLTATR, MLTPAT
Registers: All
```

SETTXT

```
Address  : #0078
Function : Switches VDP to SCREEN 0 mode
Input    : See INITXT
Registers: All
```

SETT32

```
Address  : #007B
Function : Switches VDP to SCREEN 1 mode
Input    : See INIT32
Registers: All
```

SETGRP

```
Address  : #007E
Function : Switches VDP to SCREEN 2 mode
Input    : See INIGRP
Registers: All
```

SETMLT

```
Address  : #0081
Function : Switches VDP to SCREEN 3 mode
Input    : See INIMLT
Registers: All
```

CALPAT

```
Address  : #0084
Function : Returns the address of the sprite pattern table
Input    : A  - Sprite ID
Output   : HL - For the address
Registers: AF, DE, HL
```

CALATR

```
Address  : #0087
Function : Returns the address of the sprite attribute table
Input    : A  - Sprite number
Output   : HL - For the address
Registers: AF, DE, HL
```

GSPSIZ

```
Address  : #008A
Function : Returns current sprite size
Output   : A  - Sprite size in bytes
           Carry flag set when size is 16×16 sprites otherwise Carry flag is reset
Registers: AF
```

GRPPRT

```
Address  : #008D
Function : Displays a character on the graphic screen
Input    : A  - ASCII value of the character to print
```

PSG routines

GICINI

```
Address  : #0090
Function : Initialises PSG and sets initial value for the PLAY statement
Registers: All
```

WRTPSG

```
Address  : #0093
Function : Writes data to PSG register
Input    : A  - PSG register number
           E  - Data write
```

RDPSG

```
Address  : #0096
Function : Reads value from PSG register
Input    : A  - PSG register read
Output   : A  - Value read
```

STRTMS

```
Address  : #0099
Function : Tests whether the PLAY statement is being executed as a background
           task. If not, begins to execute the PLAY statement
Registers: All
```

Console routines

CHSNS

```
Address  : #009C
Function : Tests the status of the keyboard buffer
Output   : Zero flag set if buffer is empty, otherwise not set
Registers: AF
```

CHGET

```
Address  : #009F
Function : One character input (waiting)
Output   : A  - ASCII code of the input character
Registers: AF
```

CHPUT

```
Address  : #00A2
Function : Displays one character
Input    : A  - ASCII code of character to display
```

LPTOUT

```
Address  : #00A5
Function : Sends one character to printer
Input    : A  - ASCII code of character to send
Output   : Carry flag set if failed
Registers: F
```

LPTSTT

```
Address  : #00A8
Function : Tests printer status
Output   : A  - #FF and zero flag reset if printer is ready
                #00 and zero flag set if not ready
Registers: AF
```

CNVCHR

```
Address  : #00AB
Function : Tests for the graphic header and transforms the code
Input    : A  - Character code
Output   : The carry flag is reset to not the graphic reader
           The carry flag and zero flag are set to the transformed code is set in A
           The carry flag is set and zero flag is reset to the untransformed code is set in A
Registers: AF
```

PINLIN

```
Address  : #00AE
Function : Stores in the specified buffer the character codes input until the return
```

```
           key or STOP key is pressed
Output   : HL - For the starting address of the buffer -1
           Carry flag set when it ends with the STOP key
Registers: All
```

INLIN

```
Address  : #00B1
Function : Same as PINLIN except that AUGFLG (#F6AA) is set
Output   : HL - For the starting address of the buffer -1
           Carry flag set when it ends with the STOP key
Registers: All
```

QINLIN

```
Address  : #00B4
Function : Prints a question mark and one space
Output   : HL - For the starting address of the buffer -1
           Carry flag set when it ends with the STOP key
Registers: All
```

BREAKX

```
Address  : #00B7
Function : Tests status of CTRL-STOP
Output   : Carry flag set when pressed
Registers: AF
Remark   : In this routine, interrupts are inhibited
```

ISCNTC

```
Address  : #00BA
Function : Tests status of SHIFT-STOP
```

CKCNTC

```
Address  : #00BD
Function : Same as ISCNTC. used in Basic
```

BEEP

```
Address  : #00C0
Function : Generates beep
Registers: All
```

CLS

```
Address  : #00C3
Function : Clears the screen
Registers: AF, BC, DE
Remark   : Zero flag must be set to be able to run this routine
           XOR A will do fine most of the time
```

POSIT

```
Address  : #00C6
Function : Moves cursor to the specified position
Input    : H  - Y coordinate of cursor
```

```
           L  - X coordinate of cursor
Registers: AF
```

FNKSB

```
Address  : #00C9
Function : Tests whether the function key display is active (FNKFLG)
           If so, displays them, otherwise erase them
Input    : FNKFLG (#FBCE)
Registers: All
```

ERAFNK

```
Address  : #00CC
Function : Erase functionkey display
Registers: All
```

DSPFNK

```
Address  : #00CF
Function : Displays the function keys
Registers: All
```

TOTEXT

```
Address  : #00D2
Function : Forces the screen to be in the text mode
Registers: All
```

Controller routines

GTSTCK

```
Address  : #00D5
Function : Returns the joystick status
Input    : A  - Joystick number to test (0 = cursors, 1 = port 1, 2 = port 2)
Output   : A  - Direction
Registers: All
```

GTTRIG

```
Address  : #00D8
Function : Returns current trigger status
Input    : A  - Trigger button to test
           0 = space bar
           1 = port 1, button A
           2 = port 2, button A
           3 = port 1, button B
           4 = port 2, button B
Output   : A  - #00 trigger button not pressed
                #FF trigger button pressed
Registers: AF
```

GTPAD

```
Address  : #00DB
Function : Returns current touch pad status
Input    : A  - Function call number. Fetch device data first, then read.
```

```
            [ 0]   Fetch touch pad data from port 1 (#FF if available)
            [ 1]   Read X-position
            [ 2]   Read Y-position
            [ 3]   Read touchpad status from port 1 (#FF if pressed)

            [ 4]   Fetch touch pad data from port 2 (#FF if available)
            [ 5]   Read X-position
            [ 6]   Read Y-position
            [ 7]   Read touchpad status from port 2 (#FF if pressed)

Output   : A  - Value
Registers: All
Remark   : On MSX2, function call numbers 8-23 are forwarded to
           NEWPAD in the SubROM.
```

GTPDL

```
Address  : #00DE
Function : Returns current value of paddle
Input    : A  - Paddle number
Output   : A  - Value
Registers: All
```

Tape device routines

TAPION

```
Address  : #00E1
Function : Reads the header block after turning the cassette motor on
Output   : Carry flag set if failed
Registers: All
```

TAPIN

```
Address  : #00E4
Function : Read data from the tape
Output   : A  - Read value
           Carry flag set if failed
Registers: All
```

TAPIOF

```
Address  : #00E7
Function : Stops reading from the tape
```

TAPOON

```
Address  : #00EA
Function : Turns on the cassette motor and writes the header
Input    : A  - #00 short header
             not #00 long header
Output   : Carry flag set if failed
Registers: All
```

TAPOUT

```
Address  : #00ED
Function : Writes data on the tape
Input    : A  - Data to write
```

```
Output   : Carry flag set if failed
Registers: All
```

TAPOOF

```
Address  : #00F0
Function : Stops writing on the tape
```

STMOTR

```
Address  : #00F3
Function : Sets the cassette motor action
Input    : A  - #00 stop motor
               #01 start motor
               #FF reverse the current action
Registers: AF
```

Queue routines

LFTQ

```
Address  : #00F6
Function : Gives number of bytes in queue
Output   : A  - Length of queue in bytes
Remark   : Internal use
```

PUTQ

```
Address  : #00F9
Function : Put byte in queue
Remark   : Internal use
```

Graphic routines

RIGHTC

```
Address  : #00FC
Function : Shifts screen pixel to the right
Registers: AF
```

LEFTC

```
Address  : #00FF
Function : Shifts screen pixel to the left
Registers: AF
```

UPC

```
Address  : #0102
Function : Shifts screen pixel up
Registers: AF
```

TUPC

```
Address  : #0105
Function : Tests whether UPC is possible, if possible, execute UPC
Output   : Carry flag set if operation would end outside the screen
Registers: AF
```

DOWNC

```
Address  : #0108
Function : Shifts screen pixel down
Registers: AF
```

TDOWNC

```
Address  : #010B
Function : Tests whether DOWNC is possible, if possible, execute DOWNC
Output   : Carry flag set if operation would end outside the screen
Registers: AF
```

SCALXY

```
Address  : #010E
Function : Scales X and Y coordinates
```

MAPXY

```
Address  : #0111
Function : Places cursor at current cursor address
```

FETCHC

```
Address  : #0114
Function : Gets current cursor addresses mask pattern
Output   : HL - Cursor address
           A  - Mask pattern
```

STOREC

```
Address  : #0117
Function : Record current cursor addresses mask pattern
Input    : HL - Cursor address
           A  - Mask pattern
```

SETATR

```
Address  : #011A
Function : Set attribute byte
```

READC

```
Address  : #011D
Function : Reads attribute byte of current screen pixel
```

SETC

```
Address  : #0120
Function : Returns current screen pixel of specified attribute byte
```

NSETCX

```
Address  : #0123
Function : Set horizontal screen pixels
```

GTASPC

```
Address  : #0126
Function : Gets screen relations
Output   : DE, HL
Registers: DE, HL
```

PNTINI

```
Address  : #0129
Function : Initalises the PAINT instruction
```

SCANR

```
Address  : #012C
Function : Scans screen pixels to the right
```

SCANL

```
Address  : #012F
Function : Scans screen pixels to the left
```

Misc routines

CHGCAP

```
Address  : #0132
Function : Alternates the CAPS lamp status
Input    : A  - #00 is lamp on
            not #00 is lamp off
Registers: AF
```

CHGSND

```
Address  : #0135
Function : Alternates the 1-bit sound port status
Input    : A  - #00 to turn off
            not #00 to turn on
Registers: AF
```

RSLREG

```
Address  : #0138
Function : Reads the primary slot register
Output   : A  - For the value which was read
           33221100
           ||||||└└─ Page 0 (#0000-#3FFF)
           ||||└└─── Page 1 (#4000-#7FFF)
           ||└└───── Page 2 (#8000-#BFFF)
           └└─────── Page 3 (#C000-#FFFF)
Registers: A
```

WSLREG

```
Address  : #013B
Function : Writes value to the primary slot register
Input    : A  - Value to write, see RSLREG
```

RDVDP

```
Address  : #013E
Function : Reads VDP status register
Output   : A  - Value which was read
Registers: A
```

SNSMAT

```
Address  : #0141
Function : Returns 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
```

PHYDIO

```
Address  : #0144
Function : Executes I/O for mass-storage media like disks
Input    : F  - Set carry to write, reset carry to read
           A  - Drive number (0 = A:, 1 = B:, etc.)
           B  - Number of sectors
           C  - Media ID of the disk
           DE - Begin sector
           HL - Begin address in memory
Output   : F  - Carry set on error
           A  - Error code (only if carry 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
           B  - Number of sectors actually written or read
Registers: All
Remark   : Interrupts may be disabled afterwards. On some hard disk interfaces,
           when bit 7 of register C is set, a 23-bit addressing scheme is used
           and bits 0-6 of register C contain bits 23-16 of the sector number.
```

FORMAT

```
Address  : #0147
Function : Initialises mass-storage media like formatting of disks
Registers: All
Remark   : In minimum configuration only a HOOK is available
```

ISFLIO

```
Address  : #014A
Function : Tests if I/O to device is taking place
Output   : A  - #00 if not taking place
            not #00 if taking place
Registers: AF
```

OUTDLP

```
Address  : #014D
Function : Printer output
Input    : A  - Code to print
```

```
Registers: F
Remark   : Differences with LPTOUT:
           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
```

GETVCP

```
Address  : #0150
Function : Returns pointer to play queue
Input    : A  - Channel number
Output   : HL - Pointer
Registers: AF
Remark   : Only used to play music in background
```

GETVC2

```
Address  : #0153
Function : Returns pointer to variable in queue number VOICEN (byte at #FB38)
Input    : L  - Pointer in play buffer
Output   : HL - Pointer
Registers: AF
```

KILBUF

```
Address  : #0156
Function : Clear keyboard buffer
Registers: HL
```

CALBAS

```
Address  : #0159
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
```

MSX 2 BIOS Entries

SUBROM

```
Address  : #015C
Function : Calls a routine in SUB-ROM
Input    : IX - Address of routine in SUB-ROM
Output   : Depends on the routine
Registers: Alternative registers, IY
Remark   : Use of EXTROM or CALSLT is more convenient.
           In IX an extra value to the routine can be given by first
           pushing it to the stack.
```

EXTROM

```
Address  : #015F
Function : Calls a routine in SUB-ROM. Most common way
Input    : IX - Address of routine in SUB-ROM
Output   : Depends on the routine
Registers: Alternative registers, IY
```

```
Remark   : Use: LD IX,address
               CALL EXTROM
```

CHKSLZ

```
Address  : #0162
Function : Search slots for SUB-ROM
Registers: All
```

CHKNEW

```
Address  : #0165
Function : Tests screen mode
Output   : Carry flag set if screenmode = 5, 6, 7 or 8
Registers: AF
```

EOL

```
Address  : #0168
Function : Deletes to the end of the line
Input    : H  - X coordinate of cursor
           L  - Y coordinate of cursor
Registers: All
```

BIGFIL

```
Address  : #016B
Function : Same function as FILVRM (with 16-bit VRAM-address).
Input    : HL - Address
           BC - Length
           A  - Data
Registers: AF,BC
```

NSETRD

```
Address  : #016E
Function : Same function as SETRD (with 16-bit VRAM-address).
Input    : HL - VRAM address
Registers: AF
```

NSTWRT

```
Address  : #0171
Function : Same function as SETWRT (with 16-bit VRAM-address).
Input    : HL - VRAM address
Registers: AF
```

NRDVRM

```
Address  : #0174
Function : Reads VRAM like in RDVRM (with 16-bit VRAM-address).
Input    : HL - VRAM address
Output   : A  - Read value
Registers: F
```

NWRVRM

```
Address  : #0177
Function : Writes to VRAM like in WRTVRM (with 16-bit VRAM-address).
```

```
Input    : HL - VRAM address
           A  - Value to write
Registers: AF
```

MSX 2+ BIOS Entries

RDRES

```
Address  : #017A
Function : Read value of I/O port #F4
Input    : None
Output   : A = value read
Registers: AF
```

WRRES

```
Address  : #017D
Function : Write value to I/O port #F4
Input    : A = value to write
           When bit 7 is reset it shows the MSX 2+ startup screen on boot,
           and counts and initialises the RAM.
Output   : None
Registers: None
```

MSX turbo R BIOS Entries

CHGCPU

```
Address  : #0180
Function : Changes CPU mode
Input    : A = LED 0 0 0 0 0 x x
                   |             0 0 = Z80 (ROM) mode
                   |             0 1 = R800 ROM  mode
                   |             1 0 = R800 DRAM mode
               LED indicates whether the Turbo LED is switched with the CPU
Output   : None
Registers: None
```

GETCPU

```
Address  : #0183
Function : Returns current CPU mode
Input    : None
Output   : A = 0 0 0 0 0 0 x x
                           0 0 = Z80 (ROM) mode
                           0 1 = R800 ROM  mode
                           1 0 = R800 DRAM mode
Registers: AF
```

PCMPLY

```
Address  : #0186
Function : Plays specified memory area through the PCM chip
Input    : A = v 0 0 0 0 0 x x
                 |           | |
                 |           └─┴── Quality parameter (Speed: 0 = Fast)
                 └──────────────── VRAM usage flag
           HL= Start address in RAM or VRAM
           BC= Length of area to play
```

```
            D = Bit 0 = Bit 17 of area length when using VRAM
            E = Bit 0 = Bit 17 os start address when using VRAM
Output   : Carry flag set when aborted with CTRL-STOP
Registers: All
```

PCMREC

```
Address  : #0189
Function : Records audio using the PCM chip into the specified memory area
Input    : A = v t t t t c x x
               | | | | | | | |
               | | | | | | └─┴── Quality parameter (Speed: 0 = Fast)
               | | | | | └─────── Zero-data compression
               | └─┴─┴─┴────────── Treshold
               └──────────────── VRAM usage flag
           HL= Start address in RAM or VRAM
           BC= Length of area to play
           D = Bit 0 = Bit 17 of area length when using VRAM
           E = Bit 0 = Bit 17 os start address when using VRAM
Output   : Carry flag set when aborted with CTRL-STOP
Registers: All
```

— BiFi