7章 MSX拡張BIOS仕様

7. MSX Extended BIOS specification

7.1 概要

7.1 Summary

MSXではハードウェアを拡張する際は、そのハードウェアをカートリッジスロットに接続します。そして、そのハードウェアを制御するソフトウェアは、ROMの形でそのハードウェア上に実装されます。

もし、その制御用ソフトウェアが拡張BASICステートメントの形式であれば、ユーザーはプログラミングのときにスロットの切り換えなどを意識する必要はありません。

しかし、機械語やそれに準ずる言語（例えば、C言語など）でそのハードウェアや制御用ソフトウェアをアクセスする場合、どのスロットに呼び出し先のハードウェアが接続されているかがわからなければなりません。

この問題を解決するために定められたのが、拡張BIOSという手法です。拡張BIOSコールを使えば、

- ■何の拡張ハードウェアがシステムに実装されているか
- ■その数はいくつか
- ■どのカートリッジスロットに接続されているか

などを調べることができます。

この章では、拡張BIOSコールを使用する方法と拡張BIOSそのものを作成するために必要な情報を解説します。

In MSX, when extending hardware, the hardware is connected into a cartridge slot.
And software which controls the hardware is mounted on the hardware in the form of ROM.
If the software for control is the form of an extended BASIC statement, the user does not need to be conscious of the change of a slot, etc. at the time of programming.
However, when accessing the hardware and the software for control in the languages (for example, C language etc.) according to a machine language or it, you have to understand to which slot the hardware of the callee is connected.
The technique of extended BIOS was set in order to solve this problem. If an extended BIOS call is used

- ■ or what extension hardware is mounted in the system
- ■ how many is the number?
- ■ It can be investigated to which cartridge slot it is connected.

This chapter explains information required in order to create the method and the extended BIOS itself which use an extended BIOS call.

7.2 拡張BIOSにおけるデバイス

7.2 Devices in the extended BIOS

拡張BIOSでは、拡張ハードウェアとその制御用ソフトウェアの組み合わせを「デバイス」、とし、0～255の番号を割り当てています、違う種類のデバイスに同じ番号をつけることはできません。したがって、デバイス番号はアスキーが登録・管理しています。

デバイス番号を拡張BIOSに渡すには、呼び出しの際にDレジスタに番号を入れます。

現在登録されているデバイス番号は以下のとおりです。

In extended BIOS, the same number as the device of a different kind which makes a "device" combination of extension hardware and its software for control, and is assigning the number of 0?255 cannot be given.
In case of the extended BIOS, a combination of an extended hardware and its control software is treated as a “device” and a number between 0 and 255 is allotted to it. A same number cannot be allotted to different types of device.
Therefore, ASCII has registered and managed the device number.
Therefore, ASCII is in charge of registering and controlling the device number.
In order to pass a device number to extended BIOS, a number is put into D register in the case of

a call.
The device number registered now is as follows.

表7.100登録されているデバイス一覧
Table 7.100 Registered devices-details

番号 Number	デバイス Device
0	全てのデバイスを意味する Broadcast to all devices
1〜3	未使用 Unused
4	MSX-DOS2(メモリマッパーサポート) MSX-DOS2 (Memory mapper support)
5〜7	未使用 Unused
8	RS232C、MSX-MODEM
9	未使用 Unused
10	MSX-AUDIO
11	MSX-MIDI
12〜15	未使用 Unused
16	MSX-JE
17	漢字ドライバ Kanji driver
18〜254	未使用 Unused
255	システムエクスクルーシブ System exclusive

Unofficial devices created by the MSX community

Number	Device
22	UNAPI (http://msx.konamiman.com)

7.3 拡張BIOSの呼び出し

7.3 Extended BIOS call

ここでは、拡張BIOSの呼び出し方について説明します。
【HOKVLD(FB20H)】のビット0を調べ、「0」ならば拡張BIOSはありません。「1」ならば拡張BIOSが設定されているので、ルールに従い【EXTBIO(FFCAH)】を呼び出します。
詳しくは、各デバイスの拡張BIOSの説明をご参照下さい。
リスト7.6は、アプリケーションプログラムが拡張BIOSの「割り込み禁止の宣言」を呼び出す例です。
Here, It's described how to call the extended BIOS.
Check the bit 0 [HOKVLD (FB20H)] of, there is no extended BIOS if "0". Extended BIOS is set so if "1", call [EXTBIO (FFCAH)] in accordance with the rules.
For more information, please refer to the description of the extended BIOS of each device.
List 7.6 is an example of the application program calls a "declaration of interrupt disabled" extended BIOS.

```
hokvld equ    0fb20h        ;address of Extended BIOS valid flag
extbio equ    0ffcah        ;entry address of extended BIOS

        ld    a,(hokvld)    ;get valid flag
```

```
        rrca                    ;is the extended BIOS valid?
        jr      nc,noextbio     ;no..
        ld      d,0             ;broad-cast command
        ld      e,2             ;say 'disable interrupt'
        call    extbio          ;call extended BIOS
                .
                .
;
;Enters here when no extended BIOS
;
noextbio:
        di
        halt                    ;halt here
        jr      noextbio
```

7.4 拡張BIOSの実現方法

7.4 The realization method of extended BIOS

ここでは、拡張BIOSを実現するためのデバイス側のプログラムについて説明します。
デバイス側のソフトウェアは、まず次のような手順で環境を整えます。
1. 【HOKVLD(OFB20H)】の内容を調べ、「0」ならばこれを「1」にして、【EXTBIO(0FFCAH)】からの29バイトにC9H(RET命令)を書く。
2. 自分の後に【EXTBIO(0FFCAH)】を使用するデバイス側のプログラムのために、【EXTBIO(0FFCAH)】からの5バイトを自分のワークエリアにセーブする。
処理の最後で、セーブしたワークエリアにジャンプすることで、複数のデバイスのデバイスが同じワークエリア（EXTBIO）を使うことができる（「INIEXBIO.MAC」参照）。
3. 自分の拡張BIOSのエントリへのインタースロットコール命令を【EXTBIO(0FFCAH)】からに書く。
4. DISINT、ENAINTのプログラムを図7.52の指定アドレスに書く。

Here, the program on the device side for realizing extended BIOS is explained.
1. If it "0" Comes to investigate the contents of [HOKVLD (OFB20H)], this will be set to "1" and C9H (RET command) will be written to 29 bytes from [EXTBIO (0FFCAH)].
2. Save 5 bytes from [EXTBIO (0FFCAH)] to your work area for the program on the device side which uses [EXTBIO (0FFCAH)] after itself.
3. Write the inter-slot call instruction to the entry of your extended BIOS from [EXTBIO (0FFCAH)].
4. Write the program of DISINT and ENAINT to the specification address of Fig. 7.52.

0FFCAH	拡張BIOSのエントリ（5バイト） Extended BIOS entry (5 bytes)
0FFCFH	DISINTのエントリ（5バイト） DISINT entry (5 bytes)
0FFD4H	ENAINTのエントリ （5バイト） ENAINT entry (5 bytes)
0FFD9H~0FFE6H	DISINT/ENAINTプログラムエリア （14バイト） DISINT/ENAINT program areas (14 bytes)

図7.52 拡張BIOSエントリの記憶領域
Figure 7.52 The storage area of an extended BIOS entry

添付のフロッピーディスクに、拡張BIOSを設定するためのサンプルプログラム「INIEXBIO.MAC」が入っていますので、参照して下さい。

このプログラム中にGETSLTというルーチンがあります.これはスロット切り換えをしないで、CPUが直接アクセスできるメモリ（メインRAM）のスロットアドレスを得るものです。このプログラムは以降のプログラム例でも使います。
スロットアドレスとは各スロットにつけられたアドレスで、8ビットで表されます。その形式は以下のとおりです。
Since the sample program "INIEXBIO.MAC" for setting up extended BIOS is contained in the attached floppy disk, please refer to it.
There's a routine called GETSLT is in this program . It obtains the slot address of the memory (main RAM) which can carry out direct access of the CPU without carrying out a slot change. This program also uses subsequent example programs.
At the address attached to each slot and slot address, it is represented by 8 bits. Its format is as follows.

7	6	5	4	3	2	1	0
F	0	0	0	S	S	P	P

PP = Primary slot address
SS = Secondary slot address
F = Expanded slot flag

図7.53 スロットアドレスの形式
Figure 7.53 Slot address format

7.5 拡張BIOSの機能

7.5 Function of the extended BIOS

拡張BIOSの機能は以下の3つに分けられます。

1. デバイス番号0によるブロードキャストコマンド（すべてのデバイスの呼び出し）
2. デバイス番号による個々のデバイスの呼び出し
3. デバイス番号255によるシステムエクスクルーシブ（そのデバイスのメーカーが独自の拡張BIOSを組み入れる場合）

なお、拡張BIOSには、以下ような制限があります。
拡張BIOSはインタースロットコールで呼び出され、しかもネストしています。そのためデバイスが1つ呼ばれるたびに最低16バイトのスタックエリアが必要になります。また、80000H～BFFFHに拡張BIOSのプログラムが配置されていることもあります。したがって、拡張BIOSを呼び出す際のスタックポインタはC000Hよりも上位のアドレスになければなりません。
デバイスの制御プログラム（拡張BIOS）は、ブロードキャストコマンドと各々のデバイスの呼び出しコマンドを必ずサポートして下さい。
機能の選択はEレジスタに機能番号を入れて行います、各機能は以下で説明します。

Extended BIOS functions are divided into the following three.

1. Broadcasting Command by Device Number 0 (Call of All Device)
2. Call the device by device number
3. System by Device Number 255 -- Exclusive (when Maker of the Device Incorporates Original Extended BIOS)

Furthermore, in the extended BIOS there are the following limitations.

Extended BIOS is called by the inter slot call, yet we are nested. Therefore, whenever one device is called, at least 16 bytes of stack area is needed. Moreover, the program of extended BIOS may

be placed at 80000H~BFFFH. Therefore, the stack pointer at the time of calling extended BIOS must be in the address of a higher rank rather than C000H.
Device control program (Extended BIOS) must always support broadcast command and call commands for each device.
This is done by putting the function number in the E register the choice of function, can be found below each function.

7.5.1 ブロードキャストコマンド

7.5.1 Broadcast command

デバイス番号0による全デバイスの選択です。ブロードキャストコマンドには、以下の機能があります。
It selects all the devices by using the device number 0. A broadcasting command has the following functions.

機能番号 Function number	意味 Meaning
0	デバイス番号の取得　Get the device number
1	トラップ使用数の取得　Gets the number of traps used
2	割り込み禁止の宣言　Interrupt disable declaration
3	割り込み許可の宣言　Interrupt enable declaration

機能番号0

Function number 0

機能 Function

デバイス番号を取得します、
Get the device number

解説 Commentary

システムにどんなデバイスがあるかを調べます、各々のデバイスは呼び出し元が用意したテーブルに自分のデバイス番号を書き込んだ後、テーブルのポインタを1つインクリメントして、次のデバイスに制御を渡します。
呼び出し元に制御が戻ったとき、ポインタは最後に書き込まれたアドレス+1を指しています。
テーブルのポインタはBレジスタとHLレジスタで指定します。Bレジスタはテーブルが存在するメモリのスロットアドレスで、HLレジスタはメモリアドレスです。

Check what devices are present in the system. Each device will write their device number in the tables provided by the caller, increment the table pointer and passes the control to the next device.
When control returns to the caller, the pointer points to the last written address +1.
The pointer to the Table address is specified in B and HL registers. B register contains the slot address and the HL register is the memory address.

■テーブルの例（デバイスが4つ存在している場合）
■ Example of a table (when four devices exist)

Upper address HL at the time of return->	
	Device number

	Device number
	Device number
HL at the time of entry->	Device number
Lower address	

リスト7.7は、アプリケーションプログラムがブロードキャストコマンドの「デバイス番号の取得」を呼び出す例です。
List 7.7 is the example application program calls a "get device number" in the broadcast command.

リスト7.7 デバイス番号の取得例
List 7.7 Getting a device number example

```
;
; GETDEV - Get device number
; Entry  :none
; Return :carry flag is set if no devices
; Modify :all
;
extbio  equ     0ffcah          ;entry address of extended BIOS
getdev:
        ld      bc,table
        call    getslt          ;get slot address of the TABLE
        ld      h,b
        ld      l,c
        ld      b,a
        ld      d,0             ;broad-cast command
        ld      e,0             ;'get device number' command
        push    hl              ;save TABLE address
        call    extbio
        pop     de              ;restore TABLE address
        or      a
        sbc     hl,de           ;how many devices ?
        ret     nz              ; there are some devices
        scf                     ; no devices
        ret
table:
        ds      32              ;assume maximum 32 devices
```

機能番号1

Function number 1

機能 Function

トラップ使用数を取得します。
Gets the number of traps used.

解説 Commentary

MSX BASICには、「ON STOP」や「ON SPRITE」などのトラップ機能があります。
このエントリはトラップ機能を外部で拡張するために使用します。MSX BASICは内部で26個のイベントを管理しています。その内訳は以下のとおりです。

MSX BASIC has trap functions, such as "ON STOP" and "ON SPRITE." This entry is used in order to extend a trap function externally. MSX BASIC has managed 26 events inside. The items are as follows.

イベント番号 Event number	用途 Use
0〜9	ON KEY GOSUB
10	ON STOP GOSUB
11	ON SPRITE GOSUB
12〜16	ON STRIG GOSUB
17	ON INTERVAL GOSUB
18〜23	拡張デバイス用 Extended devices
24〜25	システム予約（使用禁止） Reserved for the system (do not use)

18番目から23番目までを拡張デバイスに使用することができます。トラップを使用するデバイスはイニシャライズ時にこの機能を呼び出し、他のデバイスが幾つ使うかを調べて、18+(この機能の返り値)番目のトラップから使用します。
この機能を呼び出す際には。Aレジスタに「0」をセットして下さい。また、戻り値（各デバイスが使っているトラップの数）はAレジスタに入ります。
リスト7.8は、アプリケーションプログラムがブロードキャストコマンドの「トラップ使用数の取得」を呼び出す例です。
From the 18th to the 23rd can be used for an extended device. Call this function at initialization, checks how many other devices use, and uses it from the trap 18 + (return value of this function).

リスト7.8 トラップ使用数の取得例
List 7.8 Number of used traps retrieval example

```
; GETTRP  - get how many traps are already used
; Entry   : none
; Return  : [A] = number of traps are already used
; Modify  : [Flag]、[DE]
;
gettrp:
      xor   a                 ;clear number of traps
      ld    d,0               ;broad-cast command
      ld    e,1               ;'get number of traps' command
      call  extbio
      ret
```

機能番号2

Function number 2

機能 Function

割り込みの禁止を宣言します。
Declare a ban on interrupt.
Disable interrupts.

解説 Commentary

CPUの割り込み機能を利用するデバイスがある場合、別のデバイスが割り込みを一定期間以上禁止すると割り込み処理に不都合が生じることがあります。例えば、RS-232Cカートリッジは受信割り込みを使用するので、別のデバイスが長期間割り込みを禁止してしまうと、受信デー

タを取りこぼしてしまいます。
これを避けるために、長期間割り込みを禁止する場合は、その前にこのエントリを呼び出します。割り込みを禁止する時間の長さは1mS以上の場合とします（1mS以上割り込みが禁止されると9600bpsの通信で受信ができなくなる）。
呼ばれたデバイス側では、割り込みが禁止されても都合の良いように内部で処理し（RS-232CではXOFFを送るなど）戻ります。割り込みを許可するときは、次の「割り込み許可の宣言」を必ず呼び出して下さい。
リスト7.9は、アプリケーションプログラムがブロードキャストコマンドの「割り込み禁止の宣言」を呼び出す例です。
この機能には、レジスタへの設定値や戻り値はありません。
When there is a device using the interruption function of CPU, if another device forbids interruption more than a certain fixed period, inconvenience may arise in interruption processing. For example, RS-232C cartridge because it uses the receive interrupt, another device and would prohibit the long-term question interrupt, it will occasionally drop the received data.
To avoid this, if you want to disable interrupts a long period of time, call this entry before it. Length of time that interrupts are disabled (which can not be received by the communication of 9600bps interrupt is prohibited 1mS or more) to the case of 1mS or more.
In the device side, which is called (for example, send the XOFF in RS-232C) treated with internal as convenient to return even if the interrupt is prohibited. When you enable the interrupt, you must call the "Enable interrupts" below.
List 7.9 is an example of the application program calls a "disable interrupts" broadcast command.
This function has no return value and set value to the register.

リスト7.9 割り込み禁止の宣言例
List 7.9 Example requesting interrupt to be disabled

```
;
; DISINT   - disable interrupt while long time
; Entry    : none
; Return   : none
; Modify   : none
disint:
      ld     d,0                ;broad-cast command
      ld     e,2                ;'disable interrupt' command
      call   EXTBIO
      ret
```

機能番号3

Function number 3

機能 Function

割り込みの許可を宣言します。
Enable interrupts.

解説 Commentary

機能番号2で禁止宣言されていた割り込みを許可します。
リスト7.10は、アプリケーションプログラムがブロードキャストコマンドの「割り込み許可の宣言」を呼び出す例です。
この機能には、レジスタへの設定値や戻り値はありません。
Re-enable the interrupts that were disabled by the function number 2.
List 7.10 is an example of the application program calls the "Enable Interrupts" broadcast command.
This function has no return value and set value to the register.

リスト7.10 割り込み許可の宣言例

List 7.10 Example requesting interrupt to be enabled

```
;
; ENAINT  - enable interrupt
; Entry   : none
; Return  : none
; Modify  : none
;
disint:
disint:
      ld     d,0                ;broad-cast command
      ld     e,3                ;'enable interrupt' command
      call   extbio
      ret
```

7.5.2 各デバイスに対するコマンド

7.5.2 Commands for each device

各々のデバイスを選択して発行するコマンドです、各デバイスに対しては、機能番号0のみが決められており、その他の機能はデバイスによって異なります。
詳しくは、各デバイスの拡張BIOSの説明を参照して下さい。
Is the command to be issued by selecting each device, for each device, only function number 0 has been determined, all other functions depend on the device.
For more information, see the description of the extended BIOS of each device.

機能番号0

Function number 0

機能 Function

エントリアドレスを取得します。

解説 Commentary

各デバイスが持っているBIOSのジャンプテーブルの先頭アドレスとスロットアドレスを調べます。各々のデバイスはデバイス番号を見て、自分が処理すべき命令かどうかを判断し、違うときは次のデバイスに制御を渡します。自分が処理すべき命令ならば、呼び出し元が用意したテーブルにジャンプテーブルの先頭アドレスとスロットアドレスを書き込んだ後、次のデバイスに制御を渡します。
呼び出し元に制御が戻ったとき、ポインタは最後に書き込まれたアドレス+1を指しています。
テーブルのポインタはBレジスタとHLレジスタで指定します。Bレジスタはテーブルのあるメモリのスロットアドレスで、HLレジスタはメモリアドレスです。

■テーブル例（同じ番号のデバイスが2つ存在している場合）

The head address and slot address of a jump table of BIOS which each device has are found out.
Each device looks at a device number, it judges whether it is the command which he should process, and when different, it passes control to the following device.
Control is passed to the following device after writing the head address and slot address of a jump table in the table which the caller prepared, if it is the command which he should process.
When control returns to the caller a pointer refers to the last written address + 1.
The pointer of a table is specified by B register and HL register.
B register is a slot address of a memory with a table, and HL register is a memory address.

■ table example (when two devices of the same number exist)

Upper address HL at the time of return->	
	Reserved for system (always 0)
	Address of the jump table LSB
	Address of the jump table MSB
	Slot address of the device
	Reserved for system (always 0)
	Address of the jump table LSB
	Address of the jump table MSB
HL at the time of entry Lower address	Slot address of the device

リスト7.11は、アプリケーションプログラムが各デバイスの「エントリアドレスの取得」を呼び出す例です。
List 7.11 is an example to which an application program calls "obtaining entry address" for each device.

リスト7.11 各デバイスのエントリアドレスの取得
List 7.11 Obtaining the entry address of each device

```
;
; GETENT  - Get entry address
; Entry   :none
; Return  :carry flag is set if no devices
; Modify  :all
;
device  equ   8                 ;device #8 is RS-232C
getent:
        ld    bc,table
        call  getsit            ;get slot address of TABLE
        ld    h,b
        ld    l,c
        ld    b,a
        ld    d,device          ;set device number
        ld    e,0               ;'get entry address' command
        push  hl                ;save TABLE address
        call  extbio
        pop   de                ;restore TABLE address
        or    a
        sbc   hl,de             ;how many devices ?
        ret   nz                ; there are some devices
        scf                     ; no devices
        ret
table:
        ds    32*4              ;assume maximum 32 devices
```

7.5.3 システムエクスクルーシブ

7.5.3 System exclusive

機能番号0
Function number 0

機能 Function

エントリアドレスを取得します。
Gets the entry address.

解説 Commentary

各デバイスが持っているメーカー独自の拡張BIOSのジャンプテーブルの先頭アドレスとスロットアドレスおよびメーカーコードを調べます。各々のデバイスは、呼び出し元が用意したテーブルにジャンプテーブルの先頭アドレスとスロットアドレスおよびメーカーコードを書き込んだ後、次のデバイスに制御を渡します。呼び出し元に制御が戻ったとき、ポインタは最後に書き込まれたアドレス+1を指しています。全てのデバイスが値を返してきますので、特定のデバイスを選択するときは、「7.5.2 各デバイスに対するコマンド（機能番号O）」と併用するようにします。
テーブルのポインタはBレジスタとHLレジスタで指定します。Bレジスタはテーブルのあるメモリのスロットアドレスで、HLレジスタはメモリアドレスです。

■テーブルの例（メーカー独自のデバイスが2つ存在している場合）

　リスト7.12は、アプリケーションプログラムがシステムエクスクルーシブの「エントリアドレスの取得」を呼び出す例です。
リスト7.12 システムエクスクルーシブのエントリアドレスの取得例

I examine the manufacturer code and slot address and the start address of the jump table of extended BIOS proprietary to have each device. After writing the manufacturer code and slot address and the start address of the jump table to table the caller were prepared, each device, and transfers control to the next device. When control returns to the caller, the pointer points to the address +1 the last written.
When choosing a specific device, it is made to use together with "7.5.2 Command (functional number 0) to each device", since all the devices return a value.
The pointer of a table is specified by B register and HL register.
B register is a slot address of a memory with a table, and HL register is a memory address.

■ table example (when two devices of the same number exist)

List 7.12 is an example of the application program calls the "acquisition of entry address" of the system exclusive.
Get an example of an entry address of 7.12 system exclusive list

```
;
; SYSENT  - Get entry address
; Entry   :none
; Return  :carry flag is set if no devices
; Modify  :all
;
systent:
        ld    bc,table
        call  getsit              ;get slot address of TABLE
        ld    h,b
        ld    l,c
        ld    b,a
        ld    d,0ffh              ;system exclusive
```

```
        ld    e,0                ;'get entry address' command
        push  hl                 ;save TABLE address
        call  extbio
        pop   de                 ;restore TABLE address
        or    a
        sbc   hl,de              ;how many devices ?
        ret   nz                 ; there are some devices
        scf                      ; no devices
        ret
table:
        ds    32*5               ;assume maximum 32 devices
```

表7.101 メーカーコード一覧
Table 7.101 Manufacturer Code List

コード Code	メーカー名 Manufacturer name
0	アスキー ASCII
1	マイクロソフト Microsoft
2	キヤノン Canon
3	カシオ計算機 Casio Computer Co., Ltd.
4	富士通 Fujitsu Limited
5	富士通ゼネラル Fujitsu General Limited
6	日立製作所 Hitachi, Ltd.
7	京セラ Kyocera Corporation
8	松下電器産業 Matsushita Electric Industrial
9	三菱電機 Mitsubishi Electric Corporation
10	日本電気 NEC
11	ヤマハ Yamaha
12	日本ビクター Victor Company of Japan (JVC)
13	フィリップス Philips
14	パイオニア Pioneer
15	三洋電機 Sanyo
16	シャープ Sharp
17	ソニー Sony Corporation
18	スペクトラビデオ Spectravideo
19	東芝 Toshiba
20	ミツミ電機 Mitsumi Electric Co., Ltd.
21	テレマティカ Telematica
22	グラディエンテ Gradiente
23	シャープドブラジル Sharp do Brasil
24	GOLD STAR Goldstar (LG Electronics)
25	DAEWOO
26	Samsung