MSX biginner's Document

MSXコンピューターの仕組み

目次

1. はじめに

本書は、MSX の基本的な動作を理解したい方々を対象に、MSX の基本的な動作の説明を纏めた資料です。

MSX の中身を知りたいけど何から調べたら良いか分からない人、簡単なプログラムは組めるようになったけど MSX がどのようにそのプログラムを実行しているのかが分からない人、が主な対象であり、各自で学習を進めるためのキーワードを覚えていただく事が主な目的です。

そのような目的のため、個々の部品の詳細には触れません。「MSX がコンピューターとして起動して操作できる、それはどういった仕組みなのか？」が焦点です。一番シンプルな MSX1 を題材とします。

2. MSX コンピューターの構成

MSX は、様々な役割の部品を組み合わせてコンピューターの動作を実現しています。ここでは、それらについて軽く触れていきたいと思います。

主な部品として、下記表（表 1 構成部品）の部品を搭載しています。

表 1 構成部品

部品名	役割
CPU（Z80）	コンピューターの頭脳
VDP（TMS9918）	表示出力用コントローラー
PSG（AY-3-8910）	音声出力用コントローラー
PPI（i8255）	汎用インターフェース
DRAM（Main RAM）	記憶装置
DRAM（Video RAM）	記憶装置
ROM（Main ROM）	不揮発性記憶装置

他にも細かい部品が多数載っていますが、概ねこれらの補助をする部品なので割愛します。では、順番に説明していきます。

2.1. CPU（Z80）

CPU は、Central Processing Unit の略称です。日本語では「中央演算処理装置」と言いますが、その名の通り全体の司令塔の役割を担っています。他の部品は、CPU から指示を受けて、指示にあった挙動をするわけです。

CPU はどうやって動いているのか？

CPU には必ずメモリが繋がっています。上の表（表 1 構成部品）にも、DRAM（Main RAM）・ROM（Main ROM）というのがありますが、それがメモリになります。

DRAM にしても ROM にしても、0 か 1 のどちらかを記憶できる記憶素子を 8 個集めて 8bit を形成し、これを沢山集めて沢山の値を記憶できるようにしています。沢山集めているどの 8bit なのか？を示す位置情報をアドレスと呼びます。1 つ 1 つにアドレス番号が振られていて、そのアドレスを指定することによって、沢山ある中の 1 つを指定するわけですね。アドレスのことを番地ということもあります。

CPU は、ROM や DRAM に対して「xxxx 番地の値を教えてください」とお願いすると、ROM や DRAM は「はい。xxxx 番地は yy です」と値を教えてくれます（図 2.1.1.）

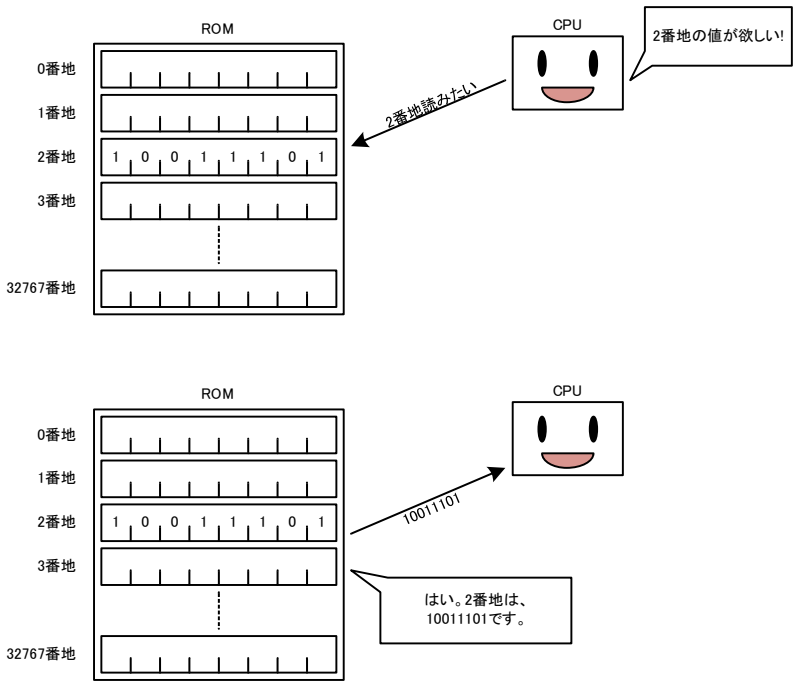

図 2.1.1. メモリアクセスのイメージ

CPU は、コンピューターの頭脳のような役割だと例えられることは多いですが、残念ながら自分で考える事はありません。中央演算処理装置という名前の通り、演算をするだけです。

何も考えない CPU に対して、人間が指示を出す必要があります。一連の指示を並べたモノを「プログラム」と呼びます。

ここで、いったん CPU から離れて数値について説明しておきます。

私たちが普段の生活で見慣れている数値は「0,1,2,3,4,5,6,7,8,9」の 10 個の記号を 1 桁として扱い、10 倍すると 1 桁増える（1 桁進む）数値の表記法になっています。左右の手の指の数が合計 10 で数えやすかったからとかそんな理由のようです。つまり、十で繰り上がるというのは数値とは無関係な理由で決められています。これを十進数表記とよびます。

では、これを「0,1」の 2 個の記号を 1 桁として扱い、2 倍すると 1 桁増える表記を採用するとどうなるのか？

0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010, 1011, ...

もの凄い勢いで桁が増えていきますが、この 0 と 1 の羅列に見えるものが、実際の数に 1:1 で対応しているので、数を表現するという意味では成り立っているわけです。

これを二進数表記と呼びます。

何らかの資料の中で具体的な数を示すために、十進数表記にするか、二進数表気にするかは、それを見る人間が見やすい方を採用すれば良い。そして、実際の記憶素子の中では二進数に相当する記憶がなされていても、全部二進数で書いてしまうと、見ている人間が見誤ります。

そこで、二進数と相性の良い十六進数表記が好まれます。

「0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F」の 16 個の記号を 1 桁として扱い、16 倍すると 1 桁増える表記が十六進数表記ですね。

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10, 11, 12, 13, ...

二進表記、十進表記、十六進表記の 10 は、それぞれ十進表記で表すと、2、10、16 になります。数値を表記する場合は、**何進表記で記述しているのか明確に区別する必要がありますね**。

（余談：たまに何進表記で書かれていても、10 なら「じゅう」と読む人が居るのですが、私は読みは十進表記に変換した値で読むか、数字をそのまま順番に読む方を好みます。十六進表記の 10 なら、ジュウロクと読むか、イチゼロと読んでいます。）

何進表記であるかを明示するために、二進表記の場合は b を、十六進表記の場合は h を末尾に付けることにします。10 は十進表記、10b は二進表記、10h は十六進表記、それぞれ十進表記では 10、2、16 ですね。

先ほどの図（図 2.1.1.）では番地を十進表記で書きましたが、**アドレスを示す数値は、特別な理由がない限りは十六進表記を使う**のが一般的です。そして、アドレスはビット幅が決まっているので、**小さい番地でも上位に 0 を付けて決まった桁数で書く**のが一般的です。

メモリと数値表記の話をしたところで、CPU の動作について掘り下げていきますね。

2.1.1. メモリアドレス

CPU には、**アドレスバス**と呼ばれる出力信号があり、Z80 の場合、16bit の幅を持つ信号となっています。これは、Z80 の出力信号として実際に Z80 の外に出ています。これが ROM や DRAM に繋がっていて、「この番地の値を読みたい」の「この番地」を示すようになっています。

16bit の幅なので、2 の 16 乗=65536 種類の番地が存在します。0 番地から 65535 番地までの 65536 種類ですね。1 アドレスに 8bit=1byte の値が割り当てられているので、65536byte＝64KB のメモリを接続できることになります。

あれれ？ MSX って、「1Mbit ROM」とかのメガロムがあるじゃない、1Mbit って 128KB だよね？ 64KB じゃ全部繋がらないじゃん！どうなってんの！？と思うかもしれません。ROM や DRAM が実際どのように繋がっているのか？については、MSX の場合、少し複雑になっていますので、後の章で詳細説明します。ここで意識していただきたいのは、**Z80 からは常に 64KB のアドレス範囲に見える**ということです。

実際にメモリから読み取った値を CPU へ伝えたり、あるいは CPU がメモリへ書き込みたい値を伝えるためにデータを流す信号線があります。これを**データバス**と呼びます。アドレスバスとデータバスですね。

2.1.2. 機械語の読み出しと実行

CPU の中にはレジスタと呼ばれる記憶素子がいくつか搭載されています。レジスタはフリップフロップと呼ばれる記憶素子で構成されており、CPU の内部状態を表現する値が記憶されています。その中の一つに、**PC レジスタ**と呼ばれるものがあります。Program Counter の略ですね。（どんな CPU にもこのレジスタは存在していますが名前が異なる場合があります。例えば、IP = Instruction Pointer だったり。他の CPU について学習する際には、PC という名前が全 CPU で共通ではないことを覚えておいて下さい。）

ＰＣレジスタは、「プログラムコードの実行位置」を示しているレジスタで、Z80 の場合、リセット直後は 0000h に初期化されています。

メモリには数値が記憶されてるんだよね？プログラムコードってどういうこと？

といった疑問を抱く方もいるかもしれません。

CPU への指示となるプログラムは、数値を並べて表現します。**電源投入直後に動くプログラムは、ROM に記録**されています。そのプログラムが起動に必要な各種初期化処理を実施しています。

MSX の場合、MainROM と呼ばれる 32KB の ROM が 0000h～7FFFh に接続されており、これが電源投入直後の「MSX System Version1.0」の表示を行ったり、DRAM の搭載量を調べる処理をしたり、MSX-BASIC が起動するのに必要な処理を実施しています。

この数値を並べて表現されたプログラムコードのことを**機械語**と呼んだりします。CPU という機械が解釈できる唯一の言語は機械語なのです。そして、機械語は CPU の種類によって異なりますので、Z80 用の機械語と、別の CPU 向けの機械語では全く数値が異なり互換性はありません。（敢えて互換性を持たせた CPU もあるわけですが、ここではそれには言及しません。）

プログラムコードが数値表現？？？

例えば、21h 34h 12h という 3byte の数値は、HL レジスタに 1234h を代入する、というプログラムコードになります。これは、「21h は後に続く 2byte を HL に代入する命令だ」と Z80 が解釈するように作りこまれています。このバリエーションが沢山あり、その組み合わせによって実行したい処理を記述したものをプログラムコードと呼んでいます。HL とはなんぞや？等疑問は尽きないかもしれませんが、ここでは「数値に何かしらの機能が割り当てられていること」を覚えていただければと思います。

2.1.3. レジスタ

Z80 には、表 1　Z80 のレジスター覧に示すレジスタが存在しています。

表 1　Z80 のレジスター覧

レジスタ名	役割
A,B,C,D,E,H,L,F	汎用レジスタ。8bit の値を記憶できる
A',B',C',D',E',H',L',F'	汎用レジスタ。裏レジスタと呼ばれるレジスタ
IX, IY	インデックスレジスタ
PC	プログラムカウンタ
SP	スタックポインタ
I	割り込みベクタレジスタ
R	リフレッシュレジスタ

Z80 には、様々な命令が用意されていて、その１つ１つに対応する数値が存在しています。メモリから汎用レジスタへ、汎用レジスタからメモリへ、汎用レジスタから汎用レジスタへ、値をコピーする命令が存在し、それにより値を移していきます。

汎用レジスタに対して様々な演算を行う命令も存在し、演算結果は再び汎用レジスタへ格納されます。

汎用レジスタは、決まった２つのペアを組み合わせることによって 16bit の値を保持できるレジスタとして使うことも出来ます。

このように様々な命令が存在し、メモリ上の PC レジスタが指し示す番地に格納されている数値（命令コード）を読み出して実行して、PC の値を +1 することを繰り返しているのが CPU です。

ジャンプ命令では、PC の値を指定の値に更新できます。そのジャンプ命令を演算結果に基づいて実行する/実行しないを選択することで、分岐することが出来ます。

Z80 の命令を説明したドキュメントは多数ありますので、詳細はそちらを参照ください。

2.1.4. 起動時の挙動

Z80 の PC レジスタは、リセット時に 0000h に初期化されます。

そして、MSX の場合、リセット時に 0000h～7FFFh には MainROM が出現しています（※詳しくは後述）。起動時に Z80 から見える 64KB のメモリ空間のイメージを、図 2.1.4-1.に示します。

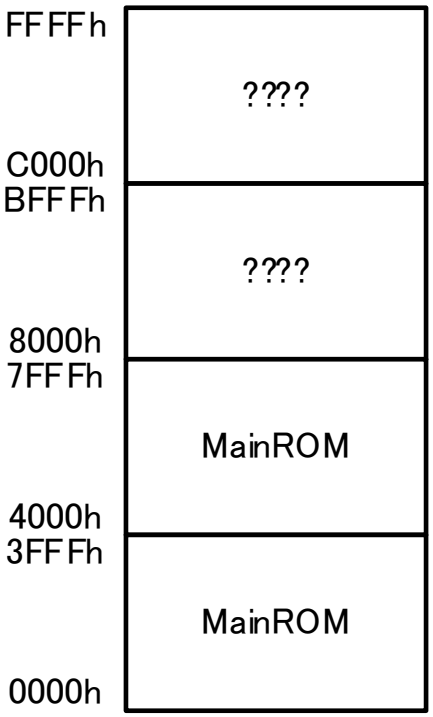

図 2.1.4-1. 起動時の Z80 メモリ空間

リセット時には、0000h～7FFFh の範囲に MainROM が見えていますが、これは ROM です。書き替えることは出来ません。
8000h～FFFFh に何がでているのかは不明です。これは、その MSX の構成によって変わります。

MainROM の内容は、全ての MSX でほぼ共通の内容になっています（一部メーカーカスタムが存在しますが、大まかな挙動に差は無いのでここでは無視してください）。

MSX 本体には、DRAM も搭載されているけど、それは何処へ行ってしまったの？

このような疑問を持つかもしれませんが、**時と場合によってメモリの構成がダイナミックに変化する**、これが MSX の最も特徴的な仕様だと思います。

起動時に RAM が使える状態にあるかどうかも怪しげです。RAM が使えないのでスタックポインタも使えません。つまり、PUSH/POP/CALL/RET のようなスタックメモリを必要とする命令が全て正常に機能しません。

さて、どういうことでしょう？

MSX の最も特徴的な仕掛けであるスロットの概念を説明します。

2.1.5. メモリ空間とスロット

スロットというとカートリッジスロットを思い浮かべる方もいるかもしれません。実は、カートリッジスロットは、MSX のスロットの概念の一部になります。

まず、Z80 の 64KB のメモリ空間を、16KB 単位に 4 つに分けて考えます（図 2.1.5-1.）。この 16KB はページと呼びます。若いバンチの方から順に page0, page1, page2, page3 となります。

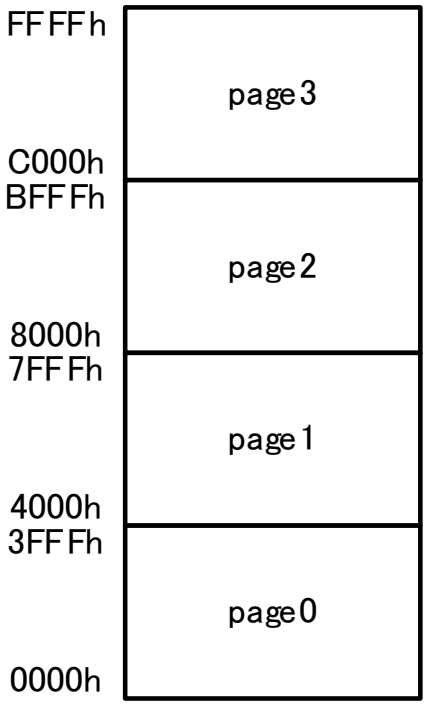

図 2.1.5-1. ページの概念

アドレス信号は、16bit あり、bit0 から順に A0, A1, A2 .... A15 と名前が付いていますが、A14, A15 の 2bit がそのまま pageN の N に対応しています。

Z80 のメモリ空間をそのまま 64KB として使うなら、A14,A15 もそのままメモリだけに繋がりますが、**MSX の場合は A14, A15 がメモリだけでなく、スロット選択レジスタにも繋がります。**

MSX には、基本スロットと拡張スロットという概念がありますが、まず基本スロットについて説明します。**基本スロットは、SLOT#0～SLOT#3 の４つ存在**します。その**夫々に 64KB の番地が存在**しています（図 2.1.5-2.）。

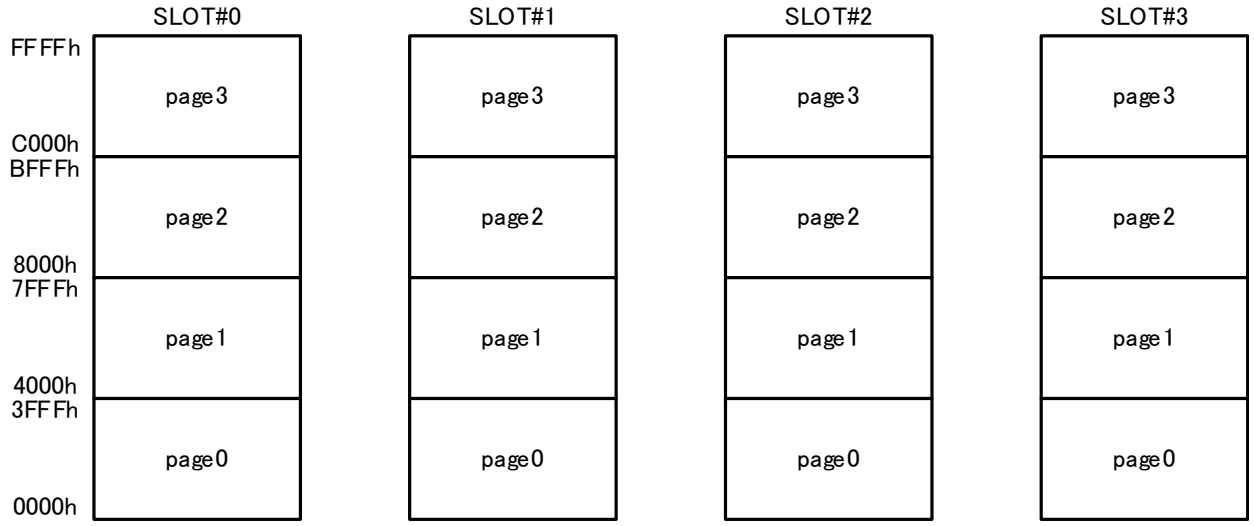

図 2.5.1-2. 基本スロット

カートリッジスロットとして現れているのは、このうちの一部です。多くの機種で、SLOT#1 がカートリッジスロット 1、SLOT#2 がカートリッジスロット 2 として、カートリッジを取り付けられるようになっています。（機種によって、SLOT#2 が無かったり、あるいは SLOT#3 もカートリッジスロットとしてでている機種もあります。）

Z80 には、64KB のアドレス空間しか無いのに、４倍もの空間をどうやって接続するのか？

ここに PPI が登場します。Programmable Peripheral Interface（i8255）ですね。表 1 構成部品に載せています。機能として表に出てこない部品ですが、MSX で最も特徴的と思われるスロットのシステムの根幹を担う部品なので載せておきました。

PPI は、1byte の値を保持しておくフリップフロップをいくつか持っています。そのうちの 1 つ Port.A を MSX では**基本スロット選択レジスタ**として使っています。1byte の各ビットは図 2.5.1-3.のような意味になっています。

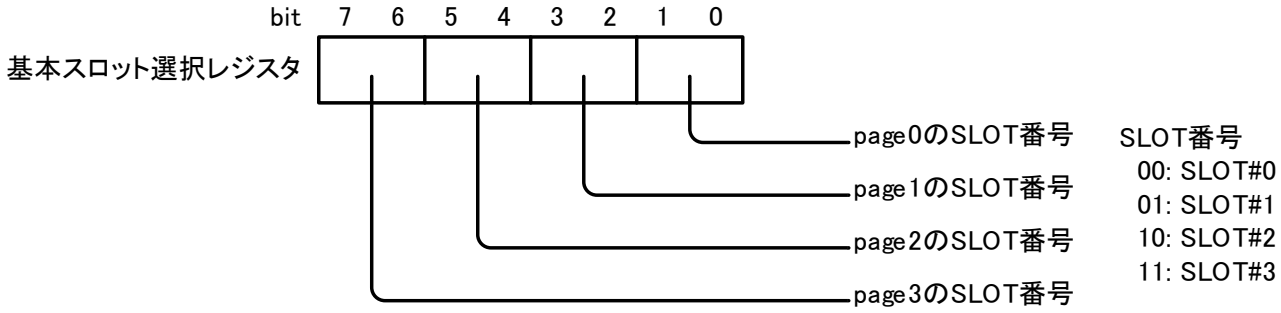

図 2.5.1-3. 基本スロット選択レジスタのビットアサイン

例えば、基本スロット選択レジスタが E4h = 11100100b となっている場合、page3 は SLOT#3，page2 は SLOT#2，page1 は SLOT#1，page0 は SLOT#0 が選択されて、**Z80 から見える 64KB は、それらが連続的に繋がった 64KB に見える**わけです（図 2.5.1-4.）。

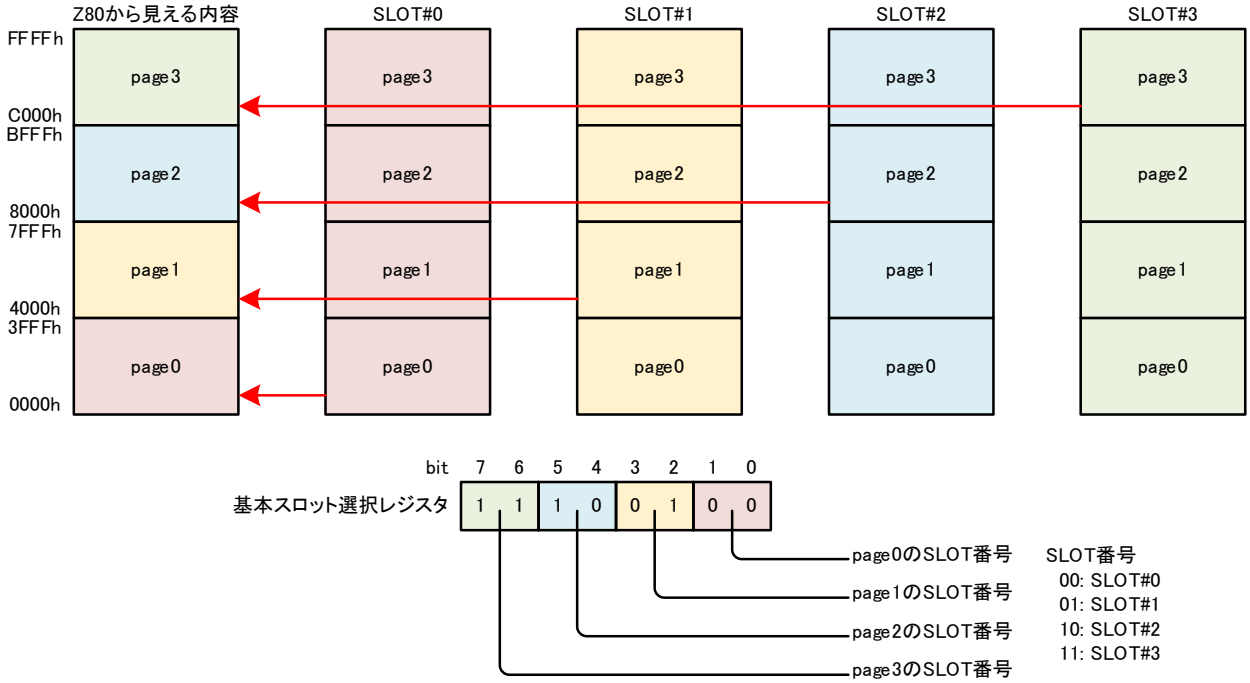

図 2.5.1-4. 基本スロット選択レジスタが E4h の場合

基本スロットレジスタは、I/O の A8h 番地にアクセスすることで読み書き出来ますが、**よほどの事情がない限りはここを直接書き替えるのは禁止**です。スロットを切り替えるための BIOS エントリーが存在するため、通常はそちらを利用して切り替えるのが一般的です。

次に拡張スロットについて説明します。

MSX のカートリッジスロットに装着して、スロットの数を増やす周辺機器を見たことがある方は、それが拡張スロットそのものです。ハードウェア的には、そちらに仕組みがありますが、MSX の BIOS にはそれを制御するための仕組みが最初から組み込まれています。先ほど述べた、**スロットを切り替えるための BIOS エントリーは、この拡張スロットも含めて切り替えることが出来ます**。

拡張スロットとは、１つの基本スロットをさらに４つに増やす仕組みです。後期の MSX では殆どの機種で本体内部にもこの仕組みが搭載されています（※論理的にそのような挙動になっているという話で、中を開けると隠しスロットが見えるわけではありません）。

拡張スロットの各 page にどの拡張スロットを出現させるのかは、拡張スロット選択レジスタで指定します。拡張スロット選択レジスタは、拡張する基本スロットの FFFFh に割り付けられ、各スロットへのアクセスより優先です。従って、拡張スロット側に装着されたカートリッジの FFFFh へはアクセス出来ません。拡張スロット選択レジスタは、読み出すと書き込んだ値を全ビット反転した値を読み出せます。BIOS はこれを使って拡張スロットの存在を確認しています。

階層的になるので分かりにくくなりますが、拡張スロットの選択のイメージを図 2.5.1-5.に示します。

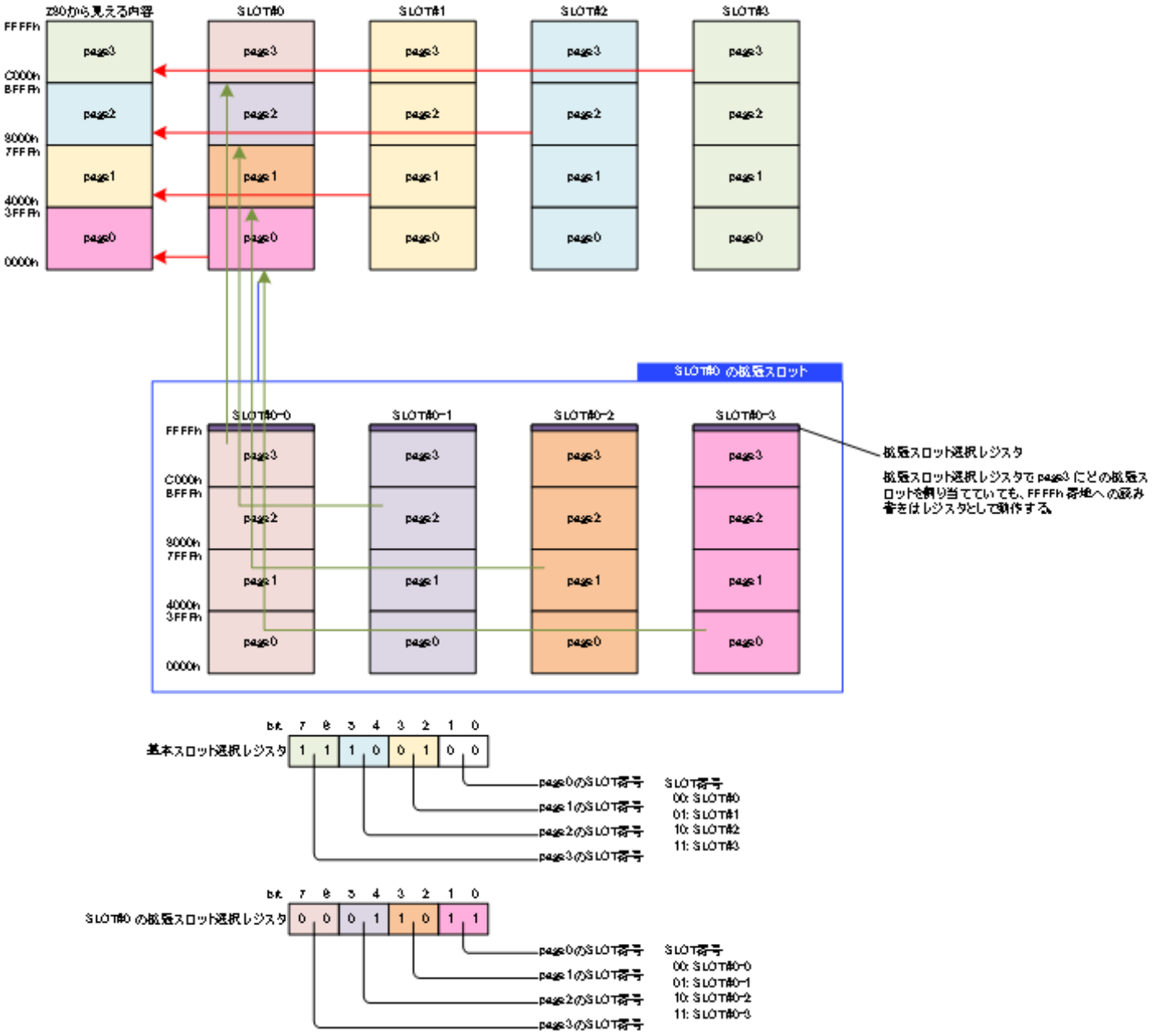

図 2.5.1-5. 基本スロットと拡張スロット

４つの基本スロットには、拡張スロットを１つだけ装着でき、１つの拡張スロットに４つまでのスロットを付けることが出来ます。もちろんカートリッジスロットとしてコネクタを付けても良いです。

スロットの切替は、このように作用します。そして、リセット後の初期状態では page0, page1 に MainROM が現れるような構成になっているため、リセット後は必ず決まった動作になります。

3. 起動シーケンス

MSX は起動すると、真っ先に MainROM が動作を開始します。

MainROM は、即座に割り込み禁止して PPI を初期化して、SLOT#0 の拡張スロットが存在するかどうか調べ、RAM を探索します。この間、確実なのは MainROM の存在だけなのでスタックは使用していません。RAM の容量（ 8K, 16K, 32K 以上 ）まで調べたら、スタックを初期化して、Z80 を割り込みモード 1 にセットします。次に、BIOS のワークエリアを初期化します。

ワークエリアを初期化したら、各スロットの 4000h（Page1 の先頭)の 2byte に "AB" が書かれているスロットが無いか調べます。あれば ROM カートリッジが装着されているモノとして、

ROM ヘッダ(表 1 ROM ヘッダの内容)を調べ、相応の処理を行います。例えば、INIT（4002h）に書かれているアドレスを CALL します。ディスクを使わないゲームであれば、ここにゲームの起動ルーチンのアドレスを書いておけば起動できます。

スロットは若い番号から順にチェックされるので、例えば SLOT#1 と SLOT#2 にゲームカートリッジを取り付けた場合、SLOT#1 の方が起動します。

表 1 ROM ヘッダの内容

アドレス	名称	内容
4000, 4001	ID	固定値 "AB"
4002, 4003	INIT	初期化ルーチンのアドレス 不要なら 0000h
4004, 4005	STATEMENT	CALL 命令拡張ルーチンのアドレス 不要なら 0000h
4006, 4007	DEVICE	OPEN "xxx:" の拡張ルーチンのアドレス 不要なら 0000h
4008, 4009	TEXT	ROM 化 BASIC コードのアドレス 8000～BFFFh の範囲 不要なら 0000h
400A～400F	-	システム予約