

Simple MegaROM Cartridge

どんな MSX にも、4つの slot が存在します。これを基本スロットと呼びます。

	slot0	slot1	slot2	slot3
0000h				
page0				
3FFFh				
4000h				
page1				
7FFFh				
8000h				
page2				
BFFFh				
C000h				
page3				
FFFFh				

i8255(またはその互換回路)が搭載されており、その Port A と呼ばれる 8bit の記憶素子が I/O Port A8h に接続されています。
この 8bit は、2bit ずつ区切られていて、下位ビットから夫々 page0, page1, page2, page3 に選択される slot の番号になっています。

bit	7	6	5	4	3	2	1	0

例えば、A8h が下記の値だった場合。

bit	7	6	5	4	3	2	1	0
				2	3			0

Z80から見える空間は下記ようになります。

	slot0	slot1	slot2	slot3	Z80から見える空間
0000h					
page0					
3FFFh					
4000h					
page1					
7FFFh					
8000h					
page2					
BFFFh					
C000h					
page3					
FFFFh					

A8h を書き替えると、この構成は瞬時に切り替わります。
A8h は、通常は起動時に 00000000 に初期化されます。Z80 は起動時に 0000h から動き始めるので、slot0 の page0 から動き始めることになります。普通の MSX は、そこに MAIN-ROM が接続されています。

Simple MegaROM Cartridge 起動時の挙動

slot0 に MAIN-ROM と RAM、slot1 がカートリッジスロット、slot2 と slot3 は未使用の MSX を例に考えてみます。
カートリッジスロットに Simple MegaROM を接続している想定です。
赤く色が付いている page が、Z80 から見える空間だと思ってください。つまり、A8h で指定されている slot ですね。

起動時は下記のようになっています。

	slot0	slot1	slot2	slot3
0000h	MAIN-ROM			
page0				
3FFFh	MAIN-ROM	SimpleMegaROM bank0 BANK#0		
4000h				
page1	RAM	SimpleMegaROM bank1 BANK#??		
7FFFh				
8000h	RAM			
page2				
BFFFh	RAM			
C000h				
page3	RAM			
FFFFh				

MAIN-ROM の page0 の方は、起動してすぐに割り込み禁止にして CHKRAM というルーチンに入り、RAM の存在する slot を探し始めます。slot0 の page2、page3 に RAM が無い機種や、他の slot にもっと大容量の RAM がある場合もあるためです。
大容量の方を選択します。最大 32KB。
上記の構成の場合、slot0 の page2、page3 にしか RAM が存在しないので、page2、page3 は slot0 が選択され、RAM が 32KB 見える状態になっています。

MAIN-ROM上のプログラム(BIOS)は、ここでようやく RAM が使えるようになるので、スタックポインタの初期化や、BIOSのための RAM初期化を行います。

次に、全てのスロットの page1 を若い番号のスロットから順に調べていき、ROMカートリッジの存在をチェックします。
ROMカートリッジには、下記のようなヘッダ情報を書くルールになっており、BIOSはその先頭の 2byte である "AB" を調べることでROMカートリッジを検知します。

4000h	ID
4002h	INIT
4004h	STATEMENT
4006h	DEVICE
4008h	TEXT
400Ah	予約
400Ch	予約
400Eh	予約

IDは、"AB" (41h, 42h) を書く決まりになっています。
INIT は、そのカートリッジの初期化ルーチンのアドレスを書いておきます。ゲームならゲームの開始アドレスをここに書いておきます。
STATEMENT は、MSX-BASIC の CALL命令拡張を行う場合に、そのルーチンのアドレスを書いておく場所です。
DEVICE は、MSX-BASIC の OPEN "XXX;" のようなデバイス拡張を行う場合に、そのルーチンのアドレスを書いておく場所です。
TEXT は、MSX-BASIC の BASICプログラムをROM化した場合に、そのアドレスを書いておく場所です。
いずれも、必要ない項目は 0000h を書いておけば無視されます。

BIOS は、32KB 存在し、slot0 の page1 にも BIOSの後半が存在しています。
しかし、ここには "AB" は書かれていないため、ここを ROMカートリッジと誤認することはありません。
slot0 の次は、slot1 を調べます。
すると、SimpleMegaROM の bank0 が見えるようになります。

BIOSは、slot1 の page1 を調べるために、page1 を slot1 に切り替えます。
BIOS も、Z80の機械語プログラムなので、当然 Z80の空間に出現させないとアクセス出来ないからです。

	slot0	slot1	slot2	slot3
0000h	MAIN-ROM			
page0				
3FFFh	MAIN-ROM	SimpleMegaROM bank0 BANK#0		
4000h				
page1	RAM	SimpleMegaROM bank1 BANK#??		
7FFFh				
8000h	RAM			
page2				
BFFFh	RAM			
C000h				
page3				
FFFFh				

この状態で、bank #0 の先頭に "AB" が書いてあれば、SimpleMegaROM は「ROMカートリッジ」と認識して貰えます。
すると、次に INIT が 0000h でなければ、そこを CALL してくれるので、bank0 に出現している BANK#0 (4000h-7FFFh) を INIT に書いておけば、そこからゲームなどのプログラムを動かすことが出来るわけです。

ただ、注意しなければならないのは、page2 は RAM (slot0) が出現していて、bank1 (slot1) ではないという点です。
BANK#0 は、4000h-7FFFh に出現している前提で、「自分自身のスロットを調べるプログラム」を置いておく必要があります。
なぜならば、SimpleMegaROM が装着されている slot が slot1 とは限らないからです。
slot2 や slot3 かもしれませんが、拡張スロット上の slot1-1 かもしれません。(拡張スロットについては後ほど説明します。)

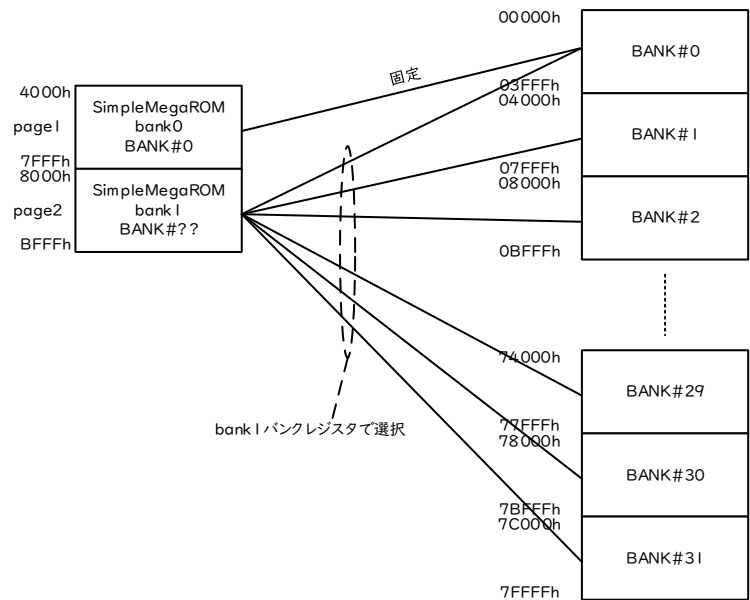
自分自身のスロットを調べるプログラムのサンプルは、後ほど提示します。
自分自身のスロットを見つけたら、必要に応じて page2 を SimpleMegaROM に切り替えて使うことになります。

SimpleMegaROM は、bank0 (page1)、bank1 (page2) が存在しますが、bank0 は BANK#0 固定です。
bank1 は、BANK#0~BANK#31 を選択して出現させられます。
bank0, bank1 とともに BANK#0 にすることもできます。

スロットの切替は、I/O の A8h を直接弄ることはオススメできません。BIOS に ENASLT, CALSLT, CALLF, RDSLT, WRSLT といったスロット操作の便利ルーチンが用意されているので、基本的にそちらを利用してください。特に、拡張スロットはこれを使わないととても大変です。

Simple MegaROM Cartridge バンクの構成

Simple MegaROM は、512KB の ROM を搭載しており、これを 16KB 単位で 32bank に分割管理されています。

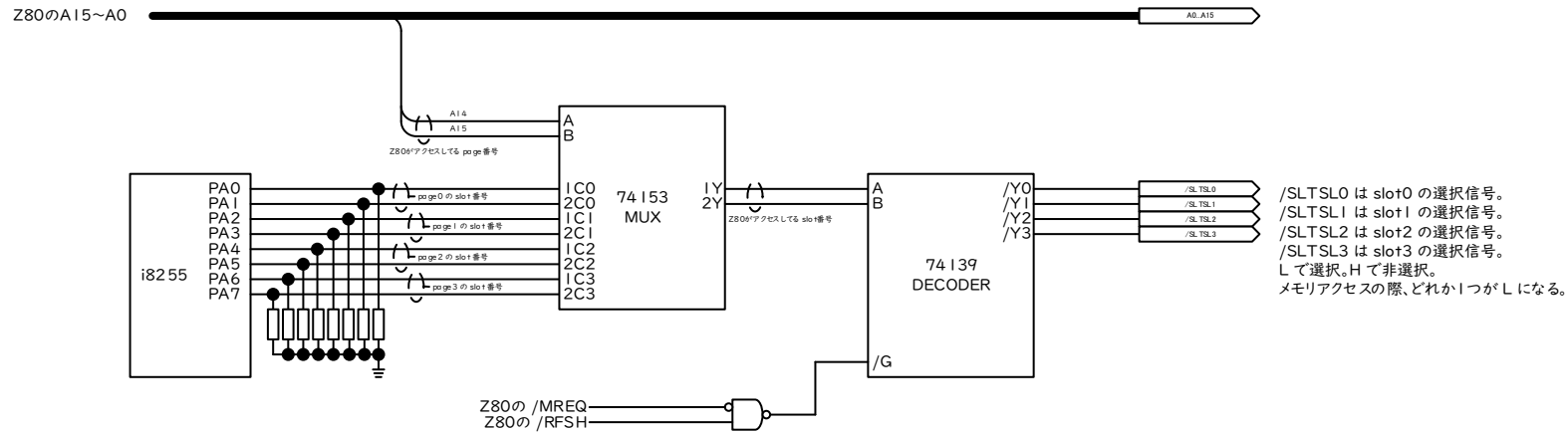


Simple MegaROM は、装着された slot の 0000h~FFFFh の空間を下記のように認識しています。

0000h	SimpleMegaROM
page0	bank 1 ミラー
3FFFh	BANK #??
4000h	SimpleMegaROM
page 1	bank 0
7FFFh	BANK #0
8000h	SimpleMegaROM
page2	bank 1
BFFFh	BANK #??
C000h	SimpleMegaROM
page3	bank 0 ミラー
FFFFh	BANK #0

MSX 本体は、i8255 portA によって選択されているスロット番号をデコードして、対応するスロットに /SLTSL = L を出すことで、「あなたがアクセスされていますよ」と通知します。この信号は、全てのスロットに個別に存在しています。

MSX本体の基本スロットの回路構成(抜粋)



Z80は、プログラムコードや、データをメモリから読み書きする際に、そのアドレスを A0~A15 の 16bit で出力してきます。この上位 2bit を page番号と見なし、i8255 portA の示す値から、対応するスロットを選択しています。スロットに装着されたカートリッジは、/SLTSL が L であることを認知して、読み書きに対応することになります。/SLTSL には、/MREQ が加味されているので、/MREQ は見なくても問題ありません。ROMカートリッジで ROM IC しか付いていないモノが多いのも、面倒な選択ロジックは MSX本体側が受け持ってくれているからなのです。

SimpleMegaROM の挙動(1) [bank0のROM読み出し]

page1 (4000-7FFFh) のリードアクセスの場合。

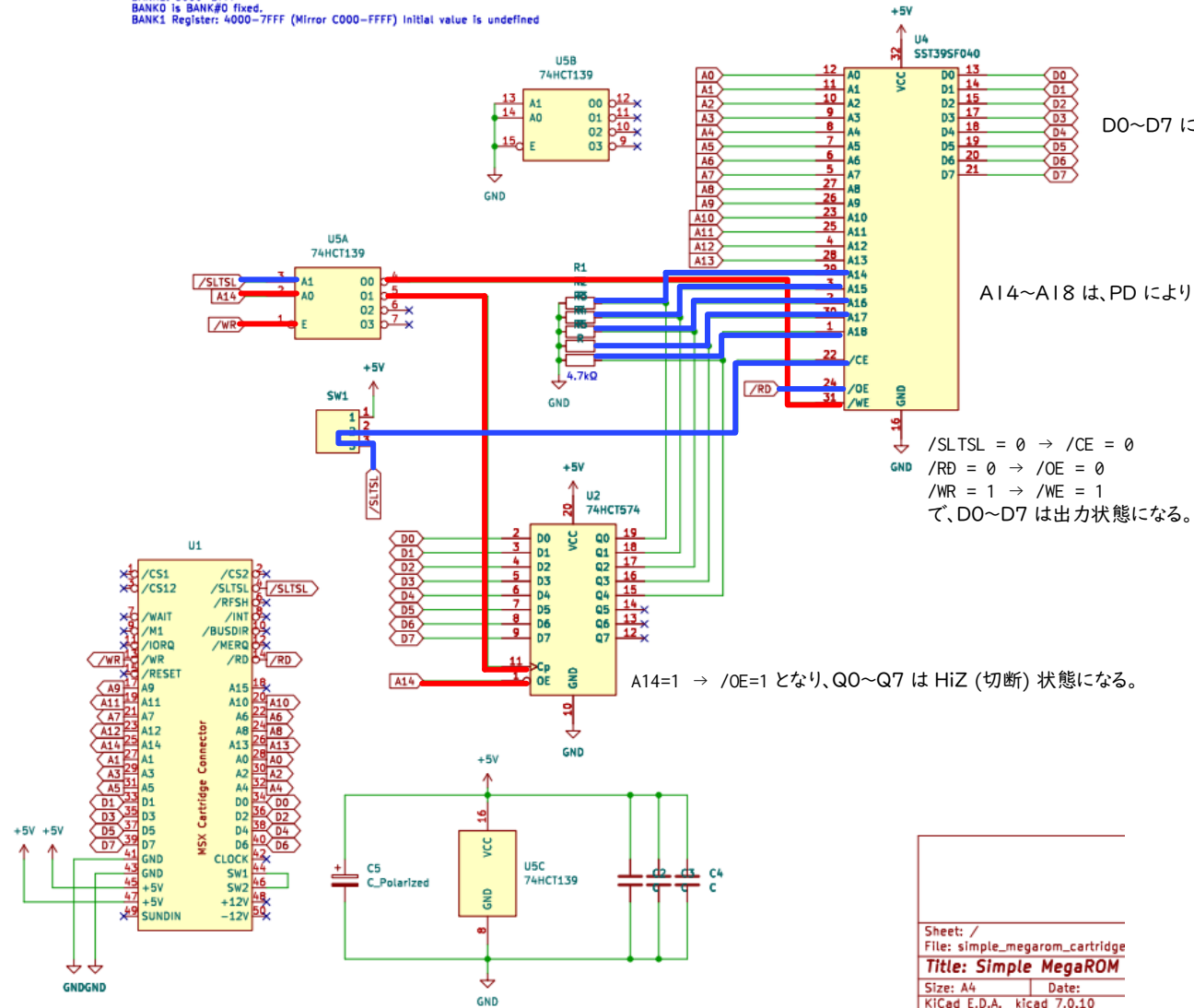
$$A15 = 0$$
$$A|4 = 1$$

A13~A0 = 任意

$$/RD = 0$$
$$/WR = 1$$
$$/SLTSL = 0$$

ROMのA0~A13はMSXのA0~A13がそのまま接続されている。

BANK0: 4000-7FFF
BANK1: 8000-BFFF
BANK0 is BANK#0 fixed.
BANK1 Register: 4000-7FFF (Mirror C000-FFFF) Initial value is undefined



D0~D7 には読み出したデータが出力され、MSXへ。

A14~A18 は、PD により 0 固定になる → BANK#0

で、D0~D7 は出力状態になる。

74574は、リセット/クリア入力を持たないFFで、初期値不定である。
出力を OFF (HiZ) に出来る。

起動時に ROMヘッダをアクセス出来るように
page0 は BANK#0 固定にする目的で、PDを付けてい
る。

A14=1 → /OE=1 となり、Q0~Q7 は HiZ (切断) 状態になる。

SimpleMegaROM の挙動(2) [bank I のROM読み出し]

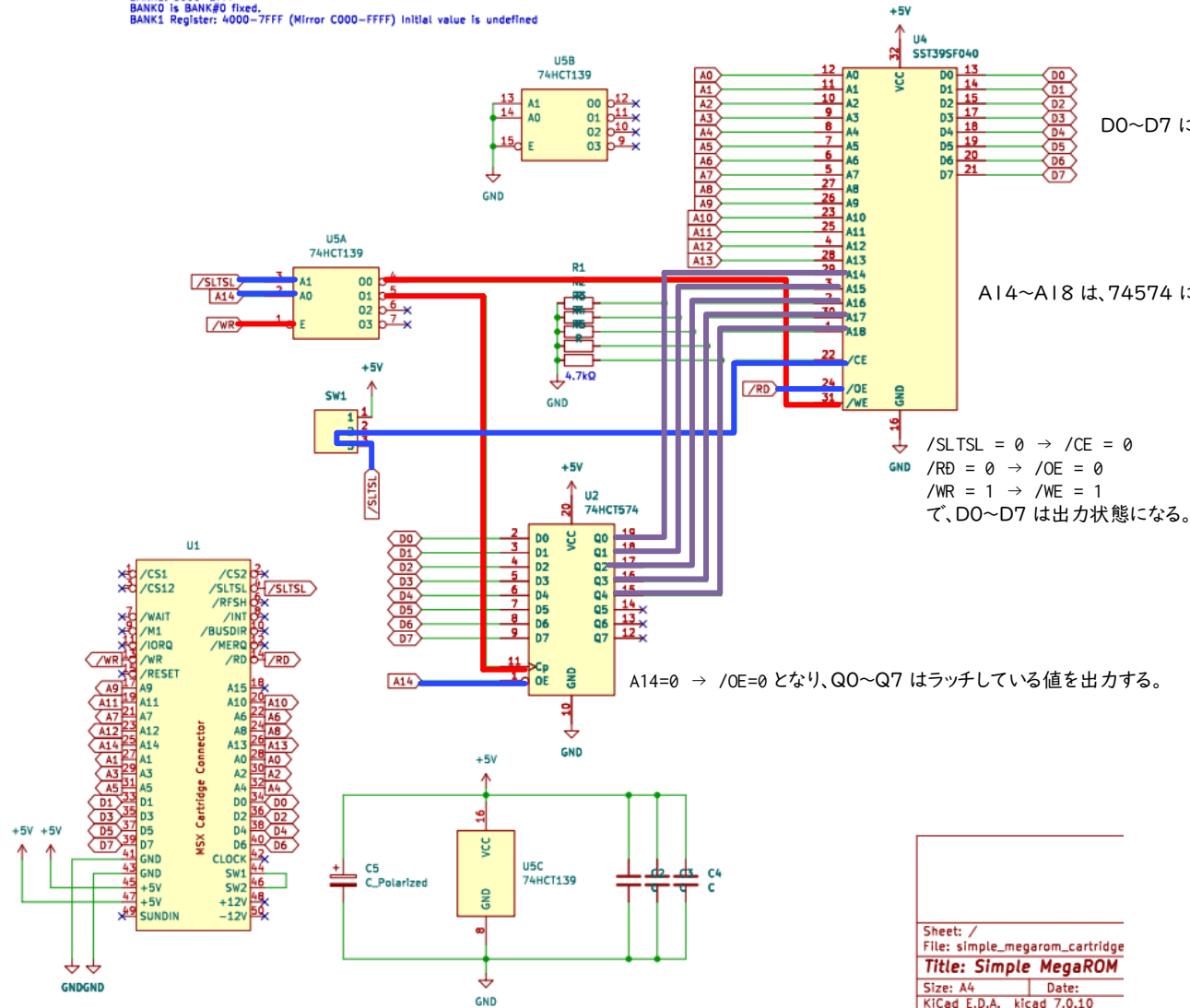
page2 (8000-BFFFh) のリードアクセスの場合。

A15 = 1
A14 = 0
A13~A0 = 任意
/RD = 0
/WR = 1
/SLTSL = 0

0
1
0 or 1

ROMの A0~A13 はMSXのA0~A13 がそのまま接続されている。

BANK0: 4000~7FFF
BANK1: 8000~BFFF
BANK0 is BANK#0 fixed.
BANK1 Register: 4000~7FFF (Mirror C000~FFFF) Initial value is undefined



Sheet: /
File: simple_megarom_cartridge
Title: Simple MegaROM
Size: A4 Date:
KiCad E.D.A. kicad 7.0.10

SimpleMegaROM の挙動(3) [バンクレジスタへの書き込み]

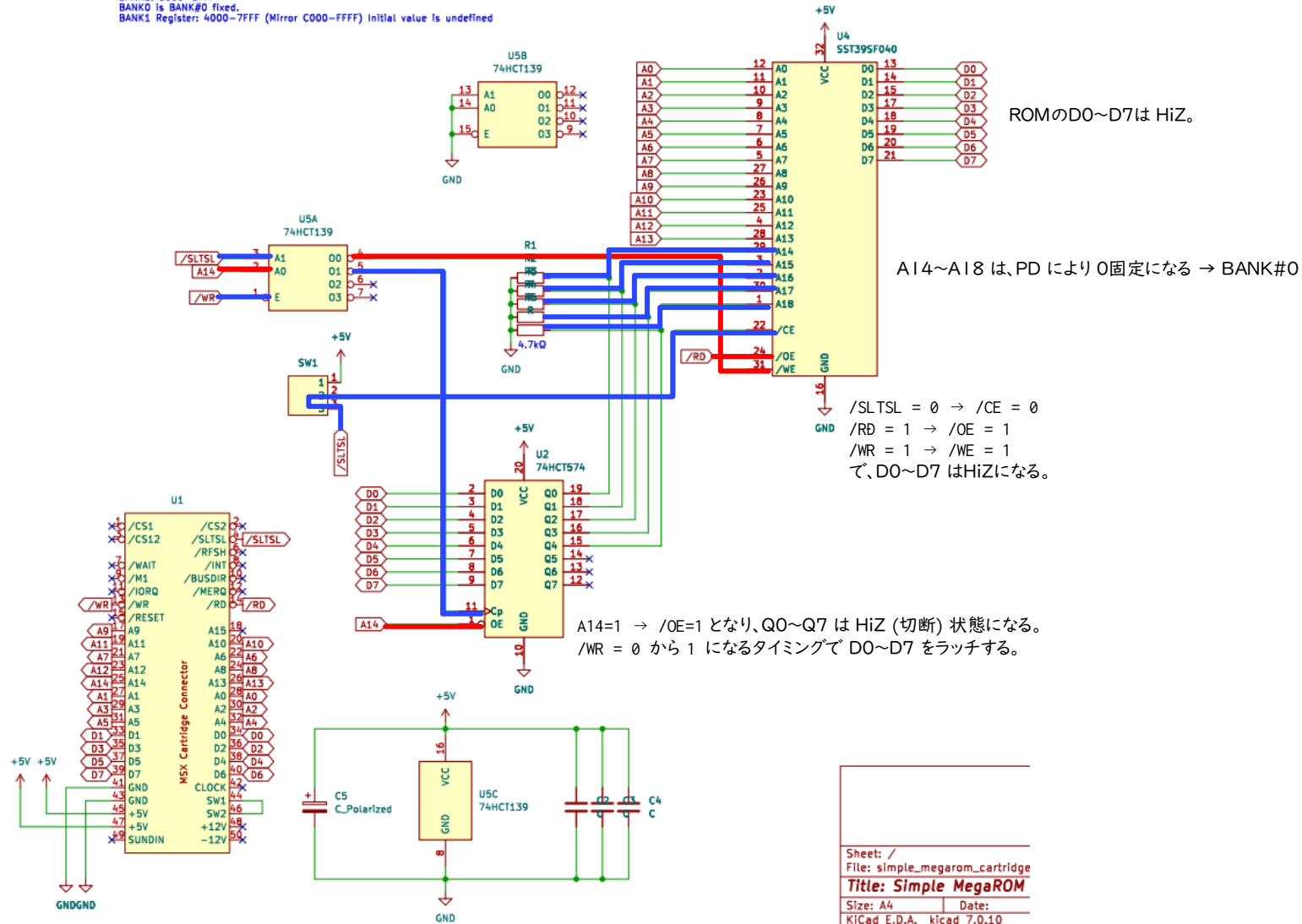
page0 (4000-7FFF) のライトアクセスの場合。

A15 = 0
A14 = 1
A13~A0 = 任意
/RD = 1
/WR = 0
/SLTSL = 0
D7~D0 = 任意

0
1

ROMの A0~A13 はMSXのA0~A13 がそのまま接続されている。

BANK0: 4000-7FFF
BANK1: 8000-BFFF
BANK0 is BANK#0 fixed.
BANK1 Register: 4000-7FFF (Mirror C000-FFFF) Initial value is undefined



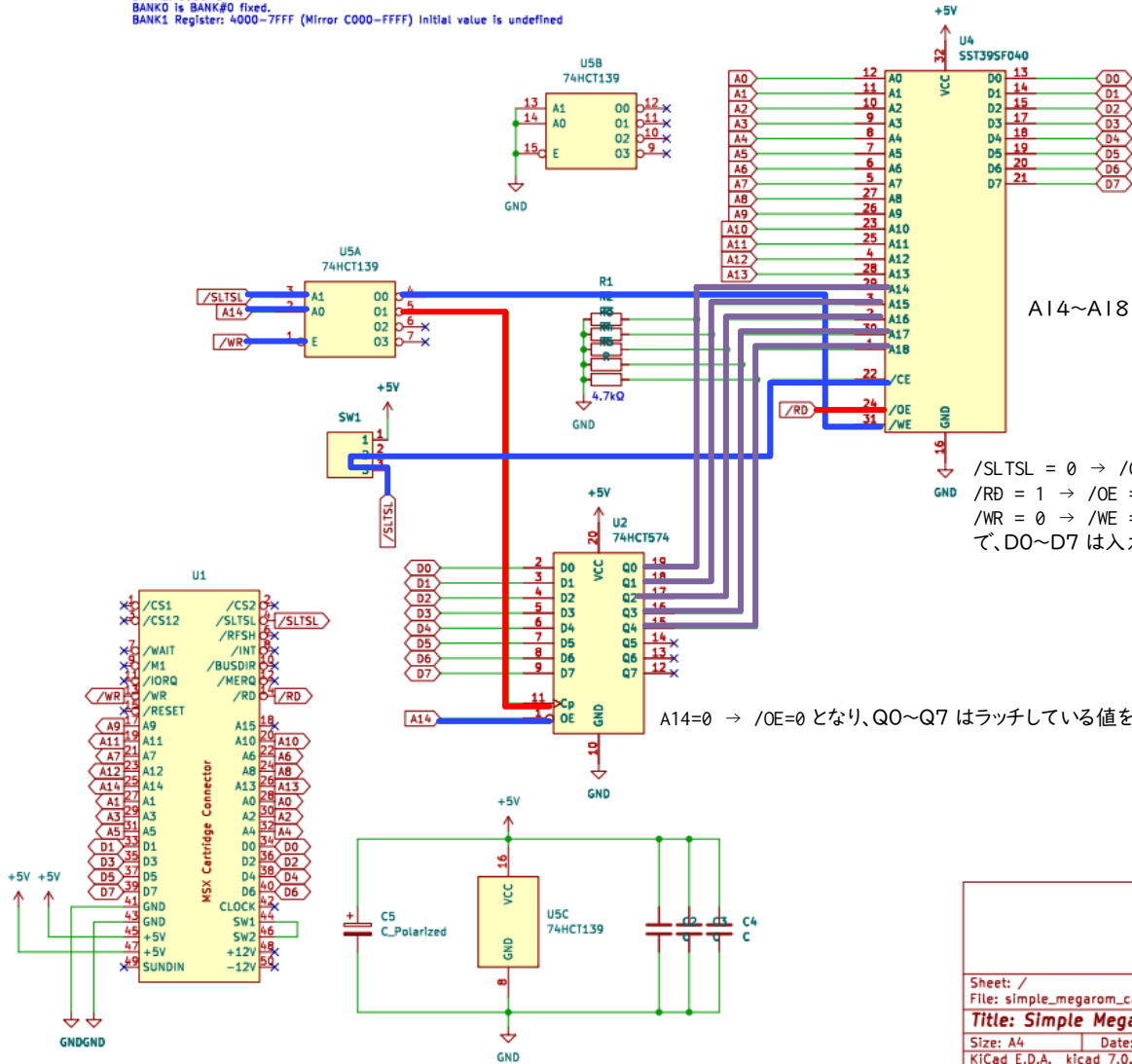
SimpleMegaROM の挙動(4) [bank I のROM書き込み]

page2 (8000-BFFFh) のライトアクセスの場合。
A15 = 1
A14 = 0
A13~A0 = 任意
/RD = 1
/WR = 0
/SLTSL = 0

0
1
0 or 1

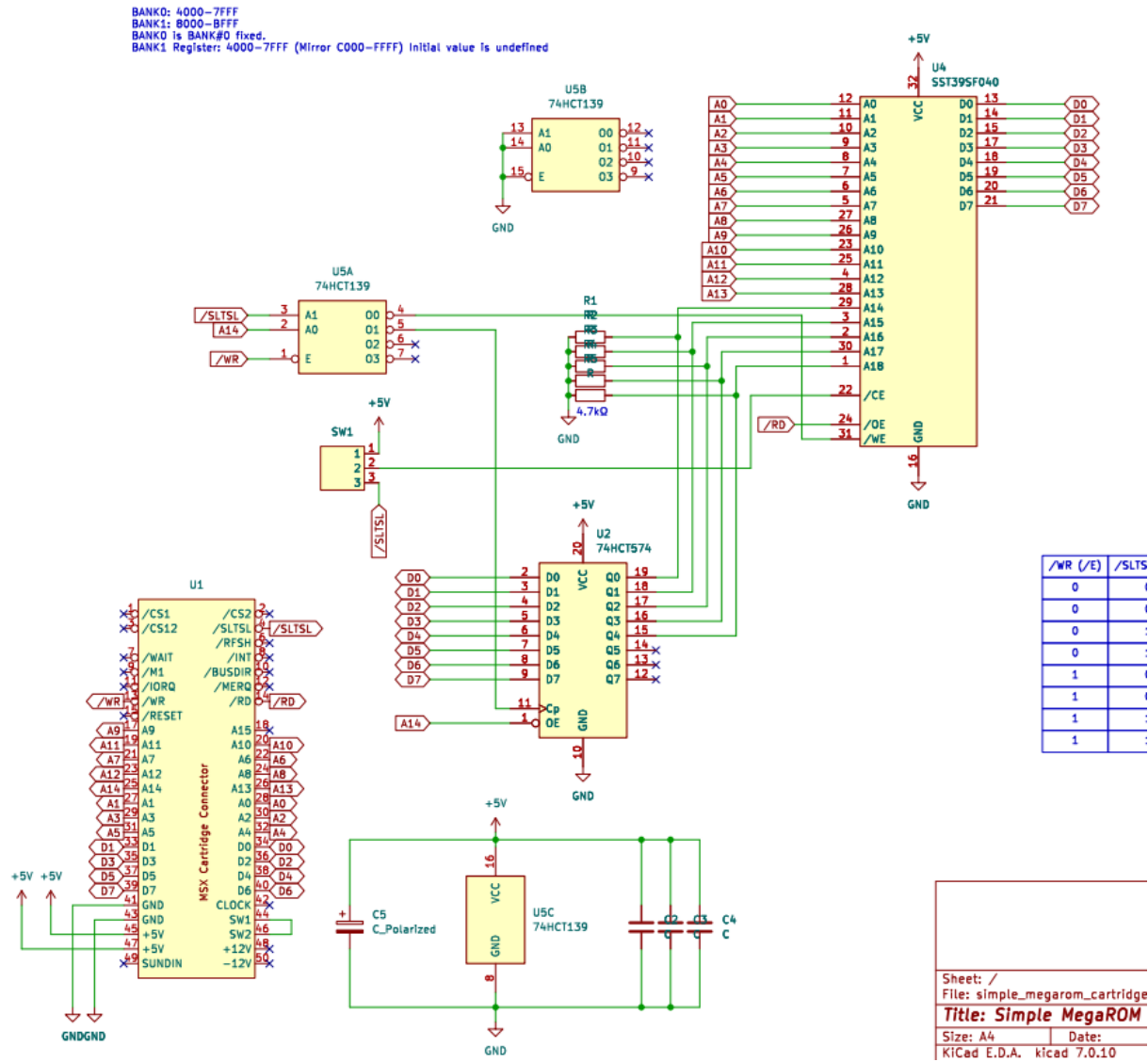
ROMの A0~A13 はMSXのA0~A13 がそのまま接続されている。

BANK0: 4000-7FFF
BANK1: 8000-BFFF
BANK0 is BANK#0 fixed.
BANK1 Register: 4000-7FFF (Mirror C000-FFFF) Initial value is undefined



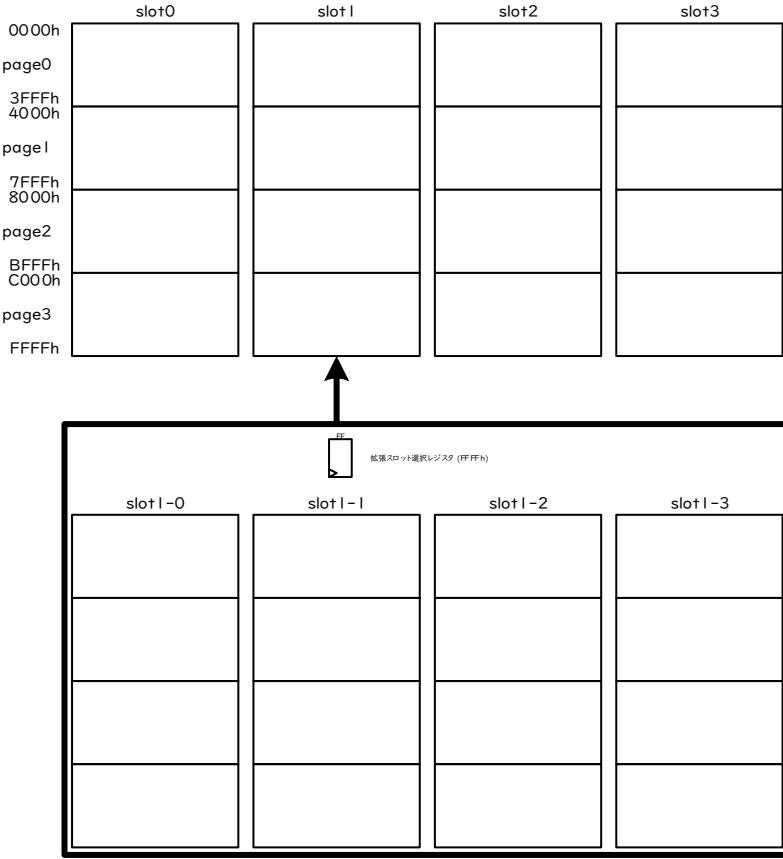
SimpleMegaROM の挙動(5) [ミラー]

回路図を見て分かりますとおり、A15 は全く使用していないため、page0 と page2、page1 とpage3 は全く区別していません。
つまり、page0 へのアクセスは page2 へのアクセスと同じ意味を持ちます。
また、page3へのアクセスは page1 へのアクセスと同じ意味を持ちます。
通常、起動時には page0 は BIOS が、page3 はスタックを置いているRAMの slots を選択している状態であり、
特に page3 を切り替えることは殆ど無いため、これを敢えてミラーとして使う事は無いでしょう(使っても良いですけど)。



MSXの拡張スロット

MSXには、拡張スロットという仕組みも定義されています。
例えば、slot1 に拡張スロットが装着されている場合、下記のようなイメージになります。



拡張スロットが接続されているスロットに対して、メモリアクセスで FFFFh 番地にアクセスすると拡張スロット選択レジスタにアクセス出来ます。
このレジスタは、8bit の値を持ち、2bit 単位で区切られており、下位ビットから page0, page1, page2, page3 の拡張スロット番号を保持しています。
この拡張スロット選択レジスタで選択されている拡張スロットが、装着されている基本スロットから見える形になります。

拡張スロット選択レジスタは、読み出すことも出来ます。読み出すと全ビット反転して読み出されます。

反転されるのは、「拡張スロット選択レジスタですよ!」との意味を示しています。
そのまま読めようと、単なる RAMカートリッジが基本スロットに直接刺さっている場合に、RAMだから書いた値が読めたのか、拡張スロット選択レジスタだから書いた値が読めたのか、区別できないためですね。書いた値の反転が読めれば、拡張スロット選択レジスタと判定できるわけです。

そして、拡張スロットの page3 にどのスロットが選択されていても、FFFFh に対するアクセスは拡張スロット選択レジスタになります。
つまり、**拡張スロットに装着した ROMやRAMの FFFFh にはアクセス出来ません。**
各拡張スロットにも /SLTSLO~3 (この 0~3 は基本スロットではなく拡張スロットの番号) が存在していて、それが L になっているスロットにアクセスすることになりますが、FFFFh だけは、この信号がでません。拡張スロット選択レジスタの回路が遮断します。

例えば、MapperRAM のように、16KB 単位の RAMを任意の page に出現させられるようなモノの場合、拡張スロット切替の度に FFFFh に出現しているマッパーセグメント (MapperRAMのバンクのようなモノ) の FFFFhが拡張スロット番号で上書きされると、そのセグメントを別のpageに出現させたときにそこが壊れている状態になってしまうためです。なので、遮断されるように拡張スロットは作られています。

後期の MSXでは、本体内は、拡張スロットになっているのが当たり前の状態になっていました。MAIN-ROM でさえ、SLOT #0-0 に装着されている状態ですね。

Simple MegaROM上のプログラムで自身のスロットを検出する

```
.org 4000h
.db 41h, 42h ; ID
.dw init ; INIT
.dw 0 ; STATEMENT
.dw 0 ; DEVICE
.dw 0 ; TEXT
.dw 0 ; Reserved
.dw 0 ; Reserved
.dw 0 ; Reserved

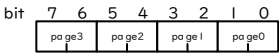
init:
; ROM のスロットを調べる
di
in a, (0A8h) ; 現在このプログラムがある page 1 のスロットを調べる
rrca
rrca
and a, 3
ld (romslt), a
add a, 0C1h
ld h, 0FCh
ld l, a
ld a, (hl) ; EXT_TBL を見る
and a, 80h
jr z, no_extsl
inc hl
inc hl
inc hl
inc hl
ld a, (hl) ; 拡張スロット選択レジスタ
and a, 0Ch
ld b, a
ld a, (romslt)
or a, b
or a, 80h
ld (romslt), a
no_extsl:
ld a, (romslt)
; PAGE 1 をカートリッジのスロットにする
ld h, 80h
call enaslt
```

MSXのBIOS には、スロット切替のためのルーチンが用意されていて、これを利用することによって面倒なスロット切替を、簡単に行える。
しかし、残念ながら現在のスロット番号を得るルーチンは用意されていない。

現在の基本スロットの 番号を得るためには、I/O A8h を読む必要がある。
現在の拡張スロットの 番号を得る場合は、ワークエリア SLTTBL (FC05h) にバックアップが書き込まれているので、そこを参照すれば良い。

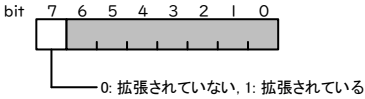
FC05h ... slot#0の拡張スロット番号
FC06h ... slot#1の拡張スロット番号
FC07h ... slot#2の拡張スロット番号
FC08h ... slot#3の拡張スロット番号

この拡張スロット番号は、2bit ずつ区切られていて、下位から page0, page 1, page 2, page 3 の拡張スロット番号になっている。

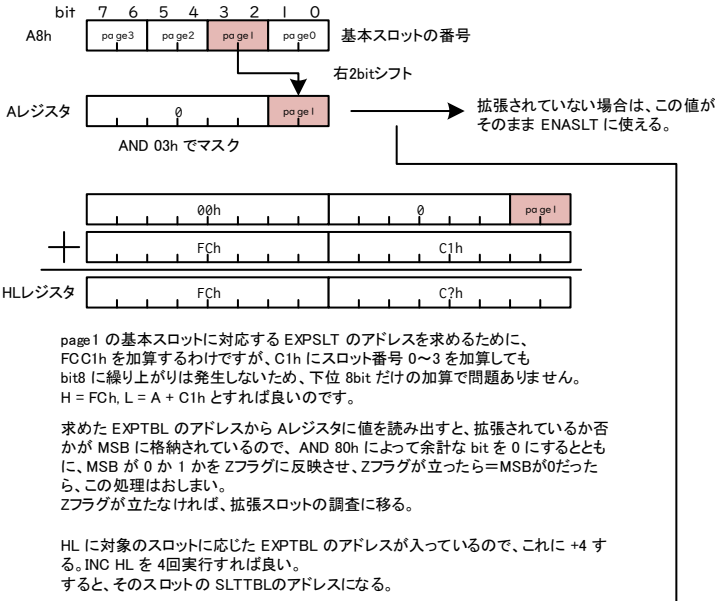
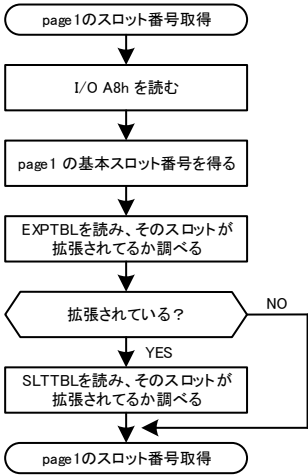


さらに、各スロットが拡張されているか否かも、BIOS のワークエリア EXPTBL(FC01h) に記録されている。

FC01h ... MSB 1bit が 1 なら slot 0 は拡張されている
FC02h ... MSB 1bit が 1 なら slot 1 は拡張されている
FC03h ... MSB 1bit が 1 なら slot 2 は拡張されている
FC04h ... MSB 1bit が 1 なら slot 3 は拡張されている

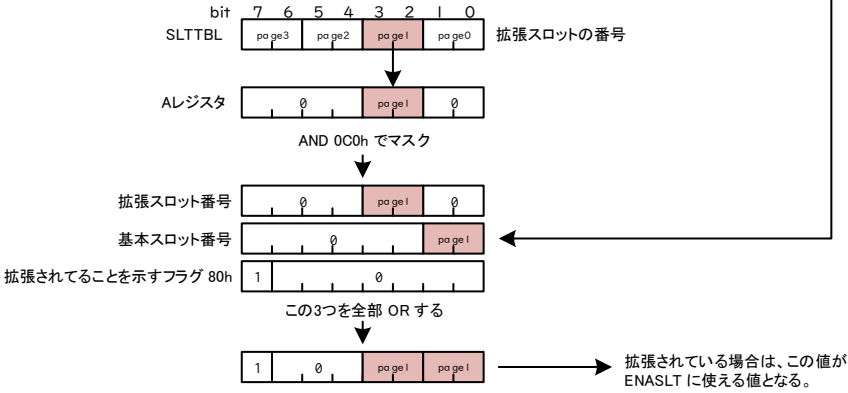


したがって、現在の page 1 のスロットの 番号を、ENASLTコール時に A レジスタに設定する形式の値を得るためには、下記のようにする。



求めた EXPTBL のアドレスから A レジスタに値を読み出すと、拡張されているか否かが MSB 1に格納されているので、AND 80h によって余計な bit を 0 にするとともに、MSB が 0 か 1 かを Zフラグに反映させ、Zフラグが立ったら＝MSBが0だったら、この処理はおしまい。
Zフラグが立たなければ、拡張スロットの調査に移る。

HL に対象のスロットに応じた EXPTBL のアドレスが入っているので、これに +4 する。INC HL を 4回実行すれば良い。
すると、そのスロットの SLTTBLのアドレスになる。



ROMの INIT が呼ばれたタイミングのスロット状態の例

FS-A1ST/A1GT、カートリッジスロット1 に ROMを装着した場合

Z80/R800から見える空間	
0000h	slot0-0 MAIN-ROM
page0	
3FFFh 4000h	
page1	slot1 カートリッジ
7FFFh 8000h	slot3-1 RAM
page2	
BFFFh C000h	slot3-1 RAM
page3	
FFFFh	

FS-A1ST/A1GT、カートリッジスロット2 に ROMを装着した場合

Z80/R800から見える空間	
0000h	slot0-0 MAIN-ROM
page0	
3FFFh 4000h	
page1	slot2 カートリッジ
7FFFh 8000h	slot3-1 RAM
page2	
BFFFh C000h	slot3-1 RAM
page3	
FFFFh	

PV-7、カートリッジスロット1 に ROMを装着した場合

Z80から見える空間	
0000h	slot0-0 MAIN-ROM
page0	
3FFFh 4000h	
page1	slot1 カートリッジ
7FFFh 8000h	slot3-1 RAM
page2	
BFFFh C000h	slot0 未接続
page3	slot0 RAM
FFFFh	

この機種だけ特殊で RAM が 8KB しかない。
page3 の後半半分(E000h-FFFFh)だけ 8KB の RAM
が見えている状態。
従って、全ての MSX に対応させるためには、RAM は
E000h-FFFEhしか使えない。
FFFFh は拡張スロットレジスタの可能性があるので、
RAMとして使用してはならない。