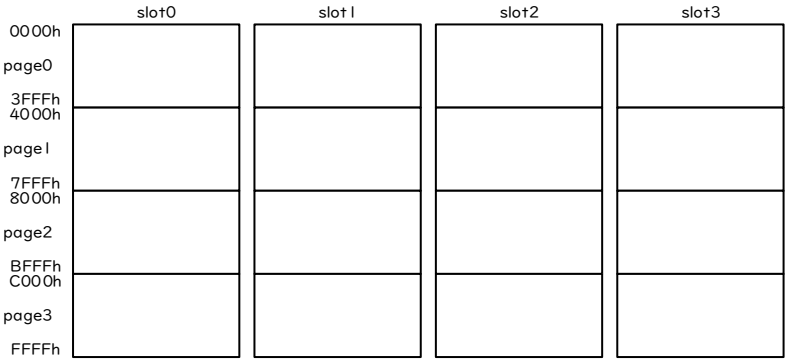
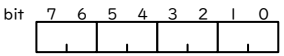
Simple MegaROM Cartridge

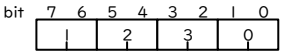
どんな MSX にも、4つの slot が存在します。これを基本スロットと呼びます。



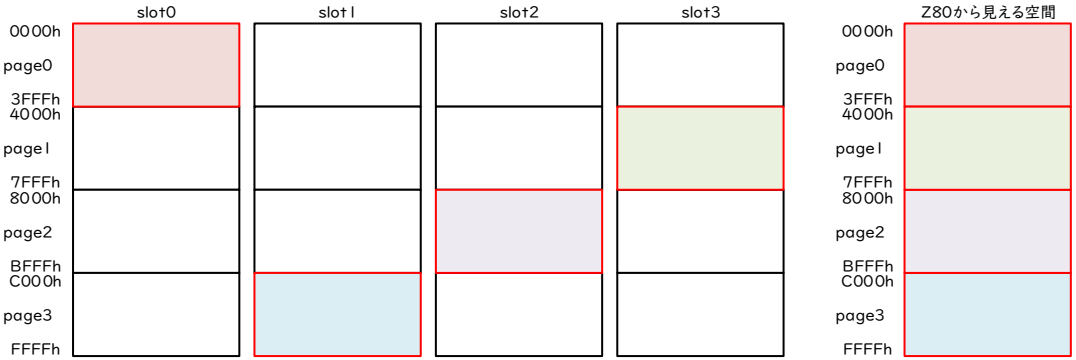
i8255(またはその互換回路)が搭載されており、その Port A と呼ばれる 8bit の記憶素子が I/O Port A8h に接続されています。
この 8bit は、2bit ずつ区切られていて、下位ビットから夫々 page0, page1, page2, page3 に選択される slot の番号になっています。



例えば、A8h が下記の値だった場合。



Z80から見える空間は下記のようになります。

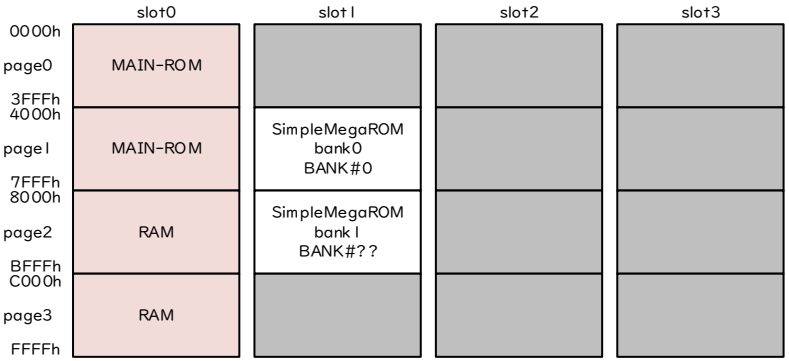


A8h を書き替えると、この構成は瞬時に切り替わります。
A8h は、通常は起動時に 00000000 に初期化されます。Z80 は起動時に 0000h から動き始めるので、
slot0 の page0 から動き始めることになります。普通の MSX は、そこに MAIN-ROM が接続されています。

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。



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。

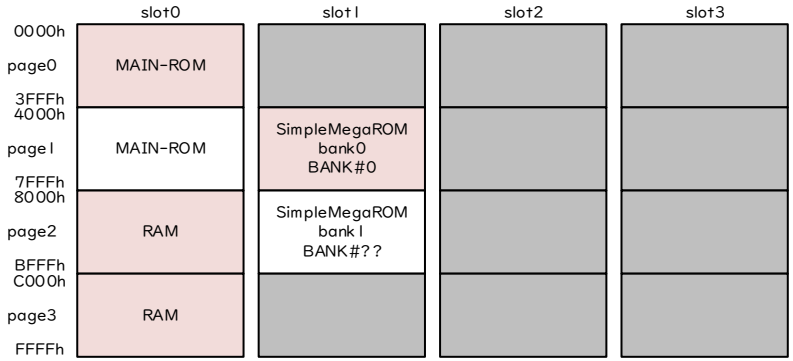
次に、全てのスロットの page1 を若い番号のスロットから順に調べていき、ROMカートリッジの存在をチェックします。
ROMカートリッジには、下記のようなヘッダ情報を書くルールになっており、BIOSはその先頭の 2byte である "AB" を調べることでROMカートリッジを検知します。

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。



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 かもしれませんし、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。)

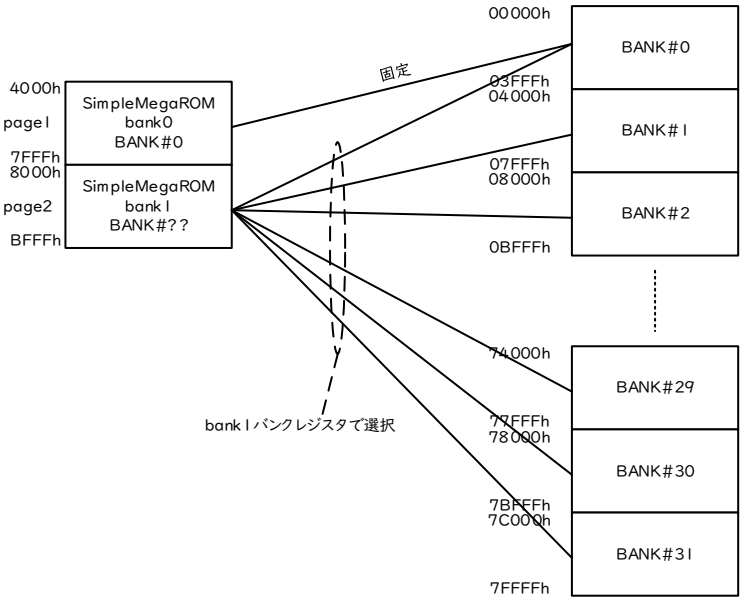
自分自身のスロットを調べるプログラムのサンプルは、後ほど提示します。
自分自身のスロットを見つけたら、必要に応じて page2 を SimpleMegaROM に切り替えて使うことになります。

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。

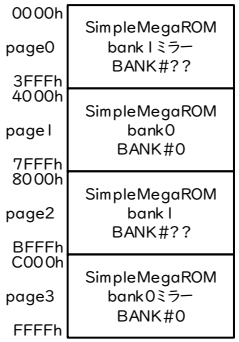
スロットの切替は、I/O の A8h を直接弄ることはオススメできません。BIOS に ENASLT, CALSLT, CALLF, RDSLT, WRSLT といったスロット操作の便利ルーチンが用意されているので、基本的にそちらを利用してください。特に、拡張スロットはこれを使わないととても大変です。

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。

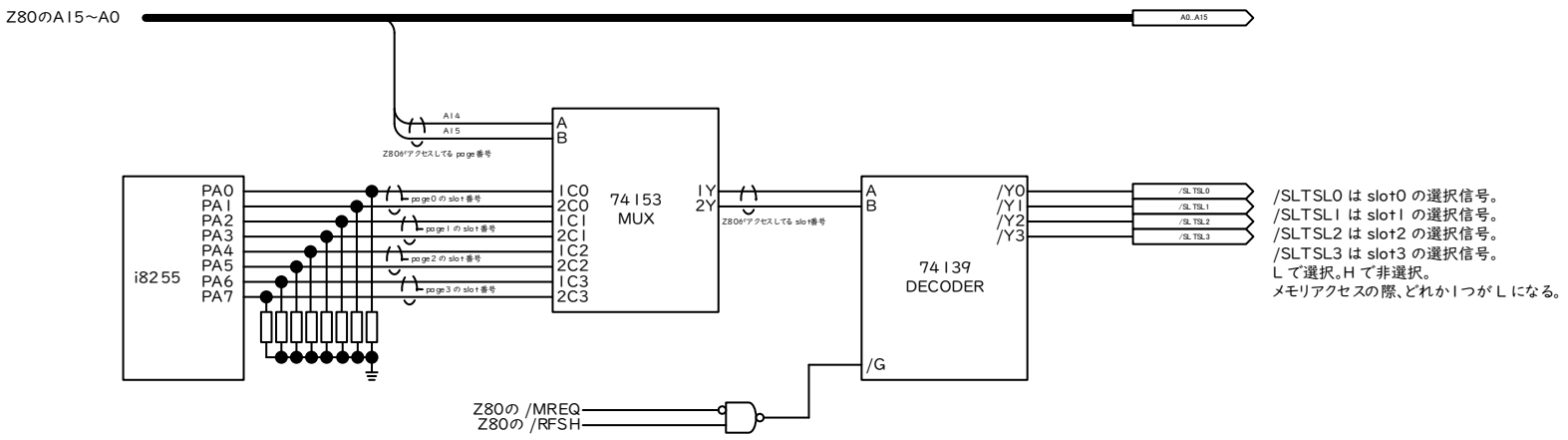


Simple MegaROM は、装着された slot の 0000h～FFFFh の空間を下記のように認識しています。



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。

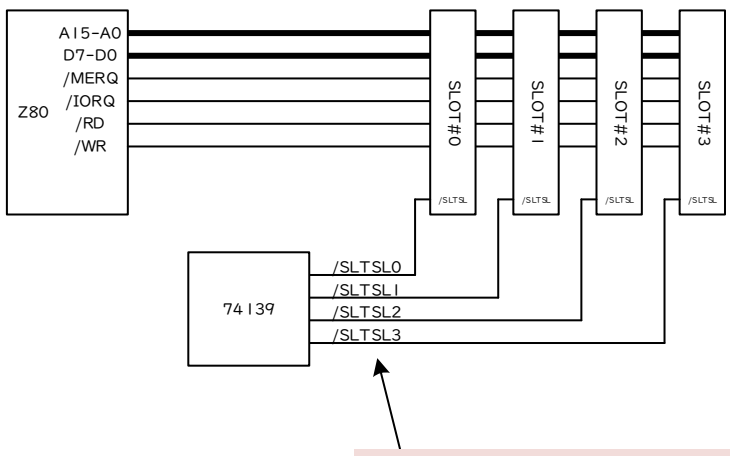
MSX本体の基本スロットの回路構成(抜粋)



/SLTSL0 は slot0 の選択信号。
/SLTSL1 は slot1 の選択信号。
/SLTSL2 は slot2 の選択信号。
/SLTSL3 は slot3 の選択信号。
L で選択。H で非選択。
メモリアクセスの際、どれか1つが L になる。

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。

MSXのカートリッジスロットに来ている各信号はどのように作られているのか？
このあたりをしっかり把握していないと、挙動を理解できない部分があるかもしれませんので、下記で簡単に解説したいと思います。



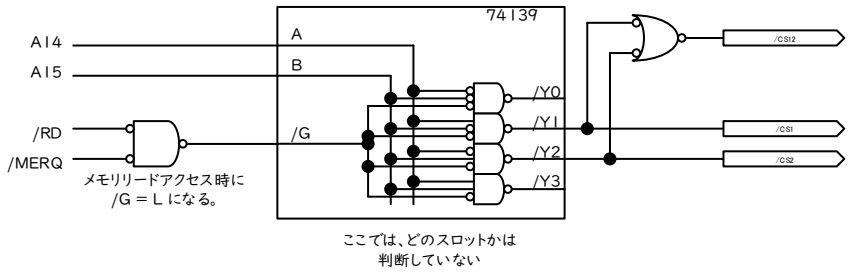
Z80がアクセスしている page を A15, A14 から判断し、
i8255 (PPI) の port A の対応ビットを選択する。
その選択した値は、「そのpageに選択されている SLOT番号」
なので、その SLOT番号をデコードして /SLTSL{SLOT番号} を
L にする。それ以外のスロットの /SLTSLx は H に。
カートリッジスロットに装着された ROM や RAM は、/SLTSL に
入ってくる値が L になってたら自分に対するアクセスだと判断する。

ご覧の通り、A, D, /MERQ, /IORQ, /RD, /WR は Z80から出てる信号がそのまま全部のスロットに繋がっています。
/SLTSL は、上記の図にあるように /MERQ が L の時しか L にならないようにロジックが組まれていますので、/SLTSL はメモリアクセスのみです。
逆に言えば、I/Oアクセスの場合は、/SLTSL を見ないので物理スロット番号は関係ないと言うことですね。

/CS1, /CS2, /CS12 という便利な信号があるのに使わないの？

/CS1, /CS2, /CS12 を生成するMSX内部のロジックには /RD が含まれており、
Readの時しか L になりません。
ROMカートリッジの場合は、Readしかしませんからこれらを有効に使えるわけですが、
Write 時に page1 と page2 を区別する目的で /CS1, /CS2 を使うことは出来ません。
Writeアクセス時には、/CS1, /CS2 は H のままなので。

/CS1, /CS2, /CS12 を生成するMSX内部のロジックには /RD が含まれており、
Readの時しか L になりません。
ROMカートリッジの場合は、Readしかしませんからこれらを有効に使えるわけですが、
Write 時に page1 と page2 を区別する目的で /CS1, /CS2 を使うことは出来ません。
Writeアクセス時には、/CS1, /CS2 は H のままなので。

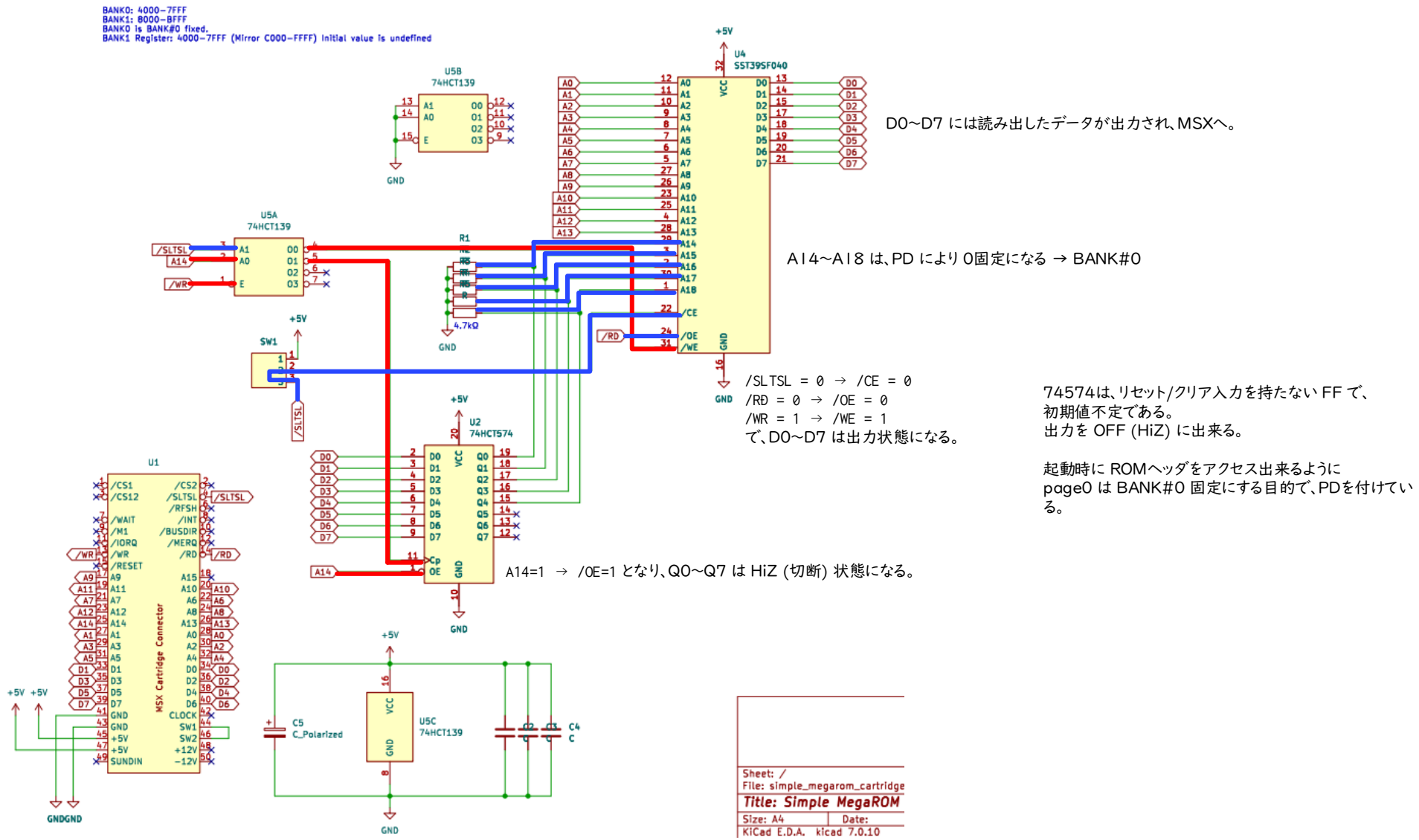


SimpleMegaROM の挙動(1) [bank0のROM読み出し]

page1 (4000-7FFFh) のリードアクセスの場合。
A15 = 0
A14 = 1
A13~A0 = 任意
/RD = 0
/WR = 1
/SLTSL = 0

0
1

ROMの A0~A13 はMSXのA0~A13 がそのまま接続されている。



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

/SLTSL = 0 → /CE = 0
/RD = 0 → /OE = 0
/WR = 1 → /WE = 1
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。

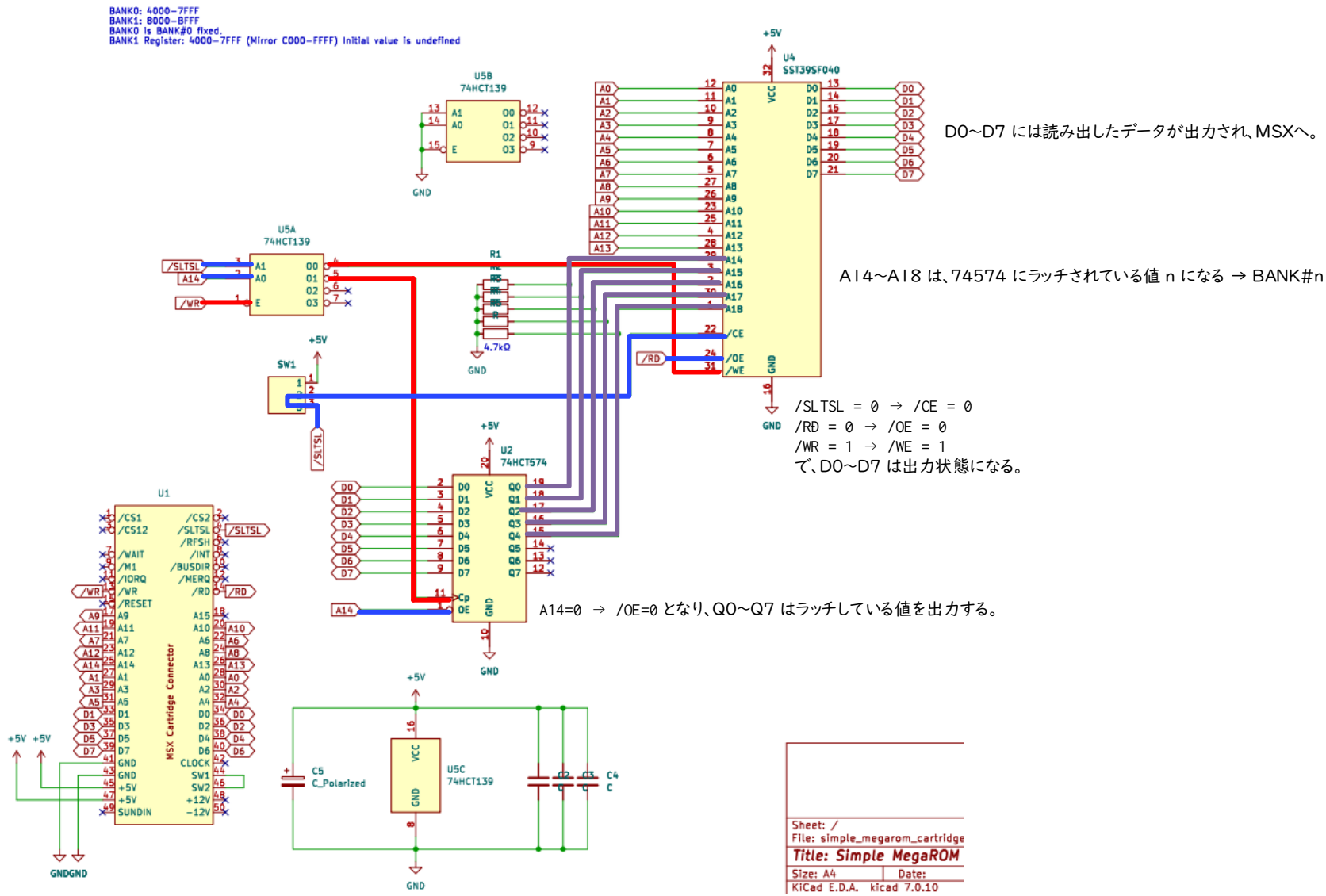
A14=1 → /OE=1 となり、Q0~Q7 は HiZ (切断) 状態になる。

SimpleMegaROM の挙動(2) [bank1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。



D0~D7 には読み出したデータが出力され、MSXへ。

A14~A18 は、74574 にラッチされている値 n になる → BANK#n

/SLTSL = 0 → /CE = 0
/RD = 0 → /OE = 0
/WR = 1 → /WE = 1
で、D0~D7 は出力状態になる。

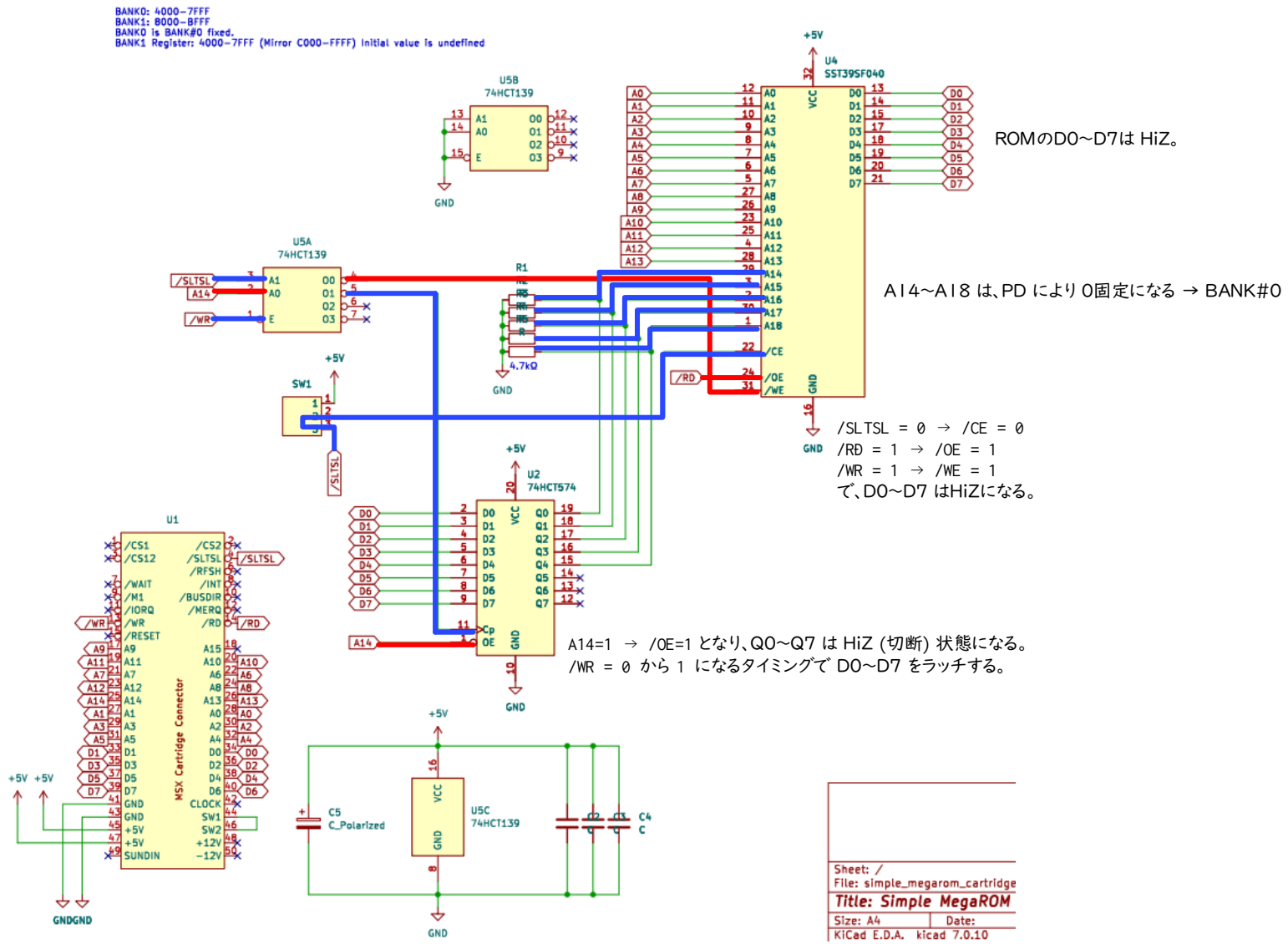
A14=0 → /OE=0 となり、Q0~Q7 はラッチしている値を出力する。

SimpleMegaROM の挙動(3) [バンクレジスタへの書き込み]

page0 (4000-7FFFh) 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。



ROMのD0~D7は HiZ。

A14~A18 は、PD により 0固定になる → BANK#0

/SLTSL = 0 → /CE = 0
/RD = 1 → /OE = 1
/WR = 1 → /WE = 1
で、D0~D7 はHiZになる。

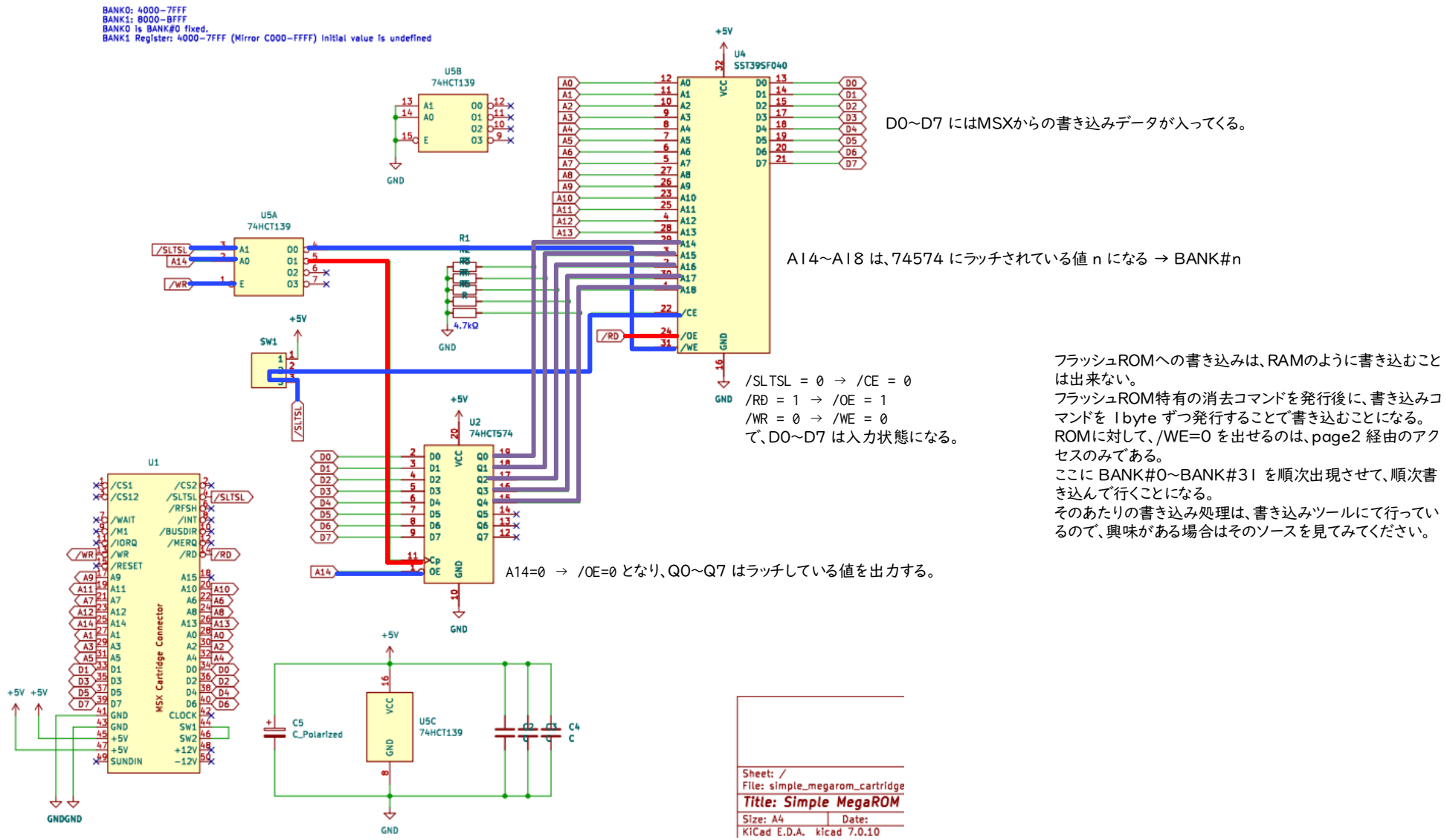
A14=1 → /OE=1 となり、Q0~Q7 は HiZ (切断) 状態になる。
/WR = 0 から 1 になるタイミングで D0~D7 をラッチする。

SimpleMegaROM の挙動(4) [bank1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。



D0~D7 にはMSXからの書き込みデータが入ってくる。

A14~A18 は、74574 にラッチされている値 n になる → BANK#n

/SLTSL = 0 → /CE = 0
/RD = 1 → /OE = 1
/WR = 0 → /WE = 0
で、D0~D7 は入力状態になる。

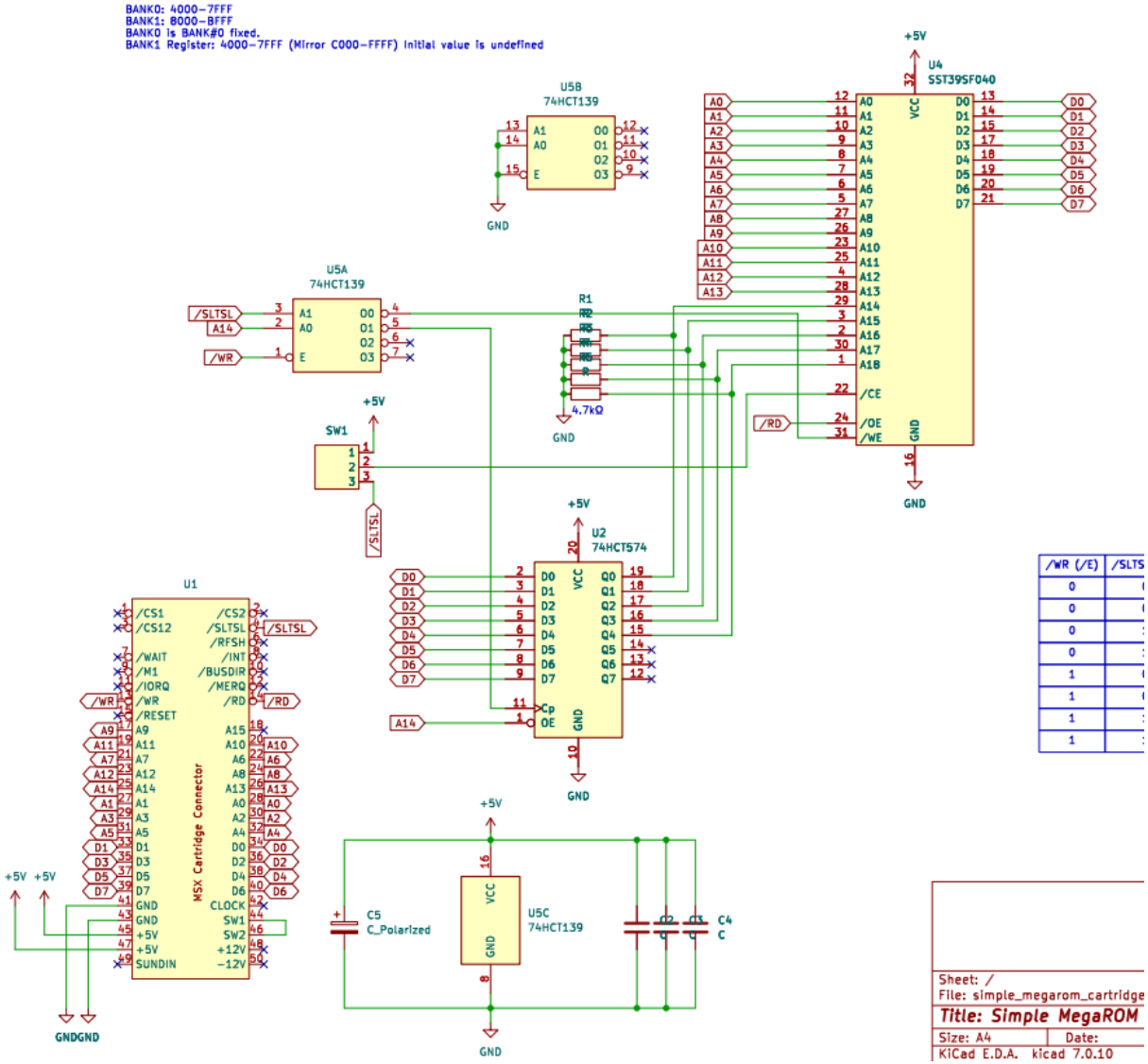
A14=0 → /OE=0 となり、Q0~Q7 はラッチしている値を出力する。

フラッシュROMへの書き込みは、RAMのように書き込むことは出来ない。
フラッシュROM特有の消去コマンドを発行後に、書き込みコマンドを 1byte ずつ発行することで書き込むことになる。
ROMに対して、/WE=0 を出せるのは、page2 経由のアクセスのみである。
ここに BANK#0~BANK#31 を順次出現させて、順次書き込んで行くことになる。
そのあたりの書き込み処理は、書き込みツールにて行っているので、興味がある場合はそのソースを見てみてください。

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

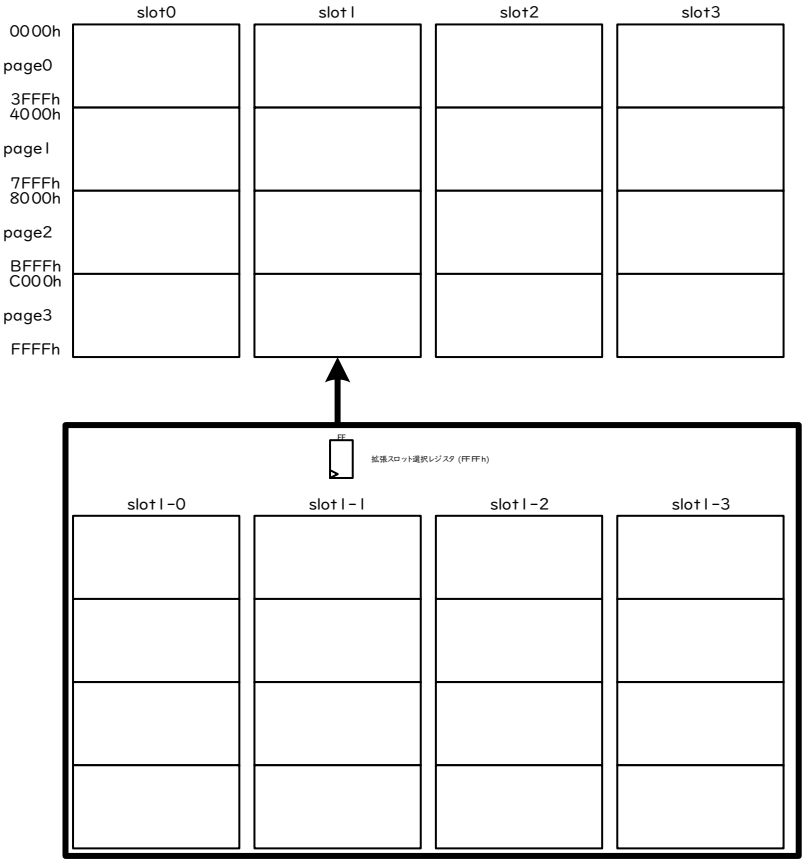
SimpleMegaROM の挙動(5) [ミラー]

回路図を見て分かるとおり、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のスロット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。
そのまま読めてしまうと、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 /SLTSL0~3 (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。

2024年6月21日 HRA!

Simple MegaROM上のプログラムで自身のスロットを検出する

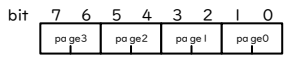
```
        .org    4000h
        .db     41h, 42h  ; ID
        .dw     init      ; INIT
        .dw     0         ; STATEMENT
        .dw     0         ; DEVICE
        .dw     0         ; TEXT
        .dw     0         ; Reserved
        .dw     0         ; Reserved
        .dw     0         ; Reserved

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 page1 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)         ; EXTTBL を見る
        and     a, 80h
        jr      z, no_extslt
        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_extslt:
        ld      a, (romslt)
        ; PAGE1 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 (FCC5h) 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。

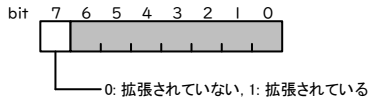
FCC5h .. slot#0の拡張スロット番号
FCC6h .. slot#1の拡張スロット番号
FCC7h .. slot#2の拡張スロット番号
FCC8h .. slot#3の拡張スロット番号

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, page1, page2, page3 の拡張スロット番号になっている。

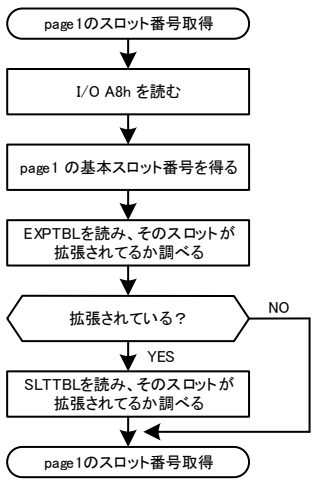


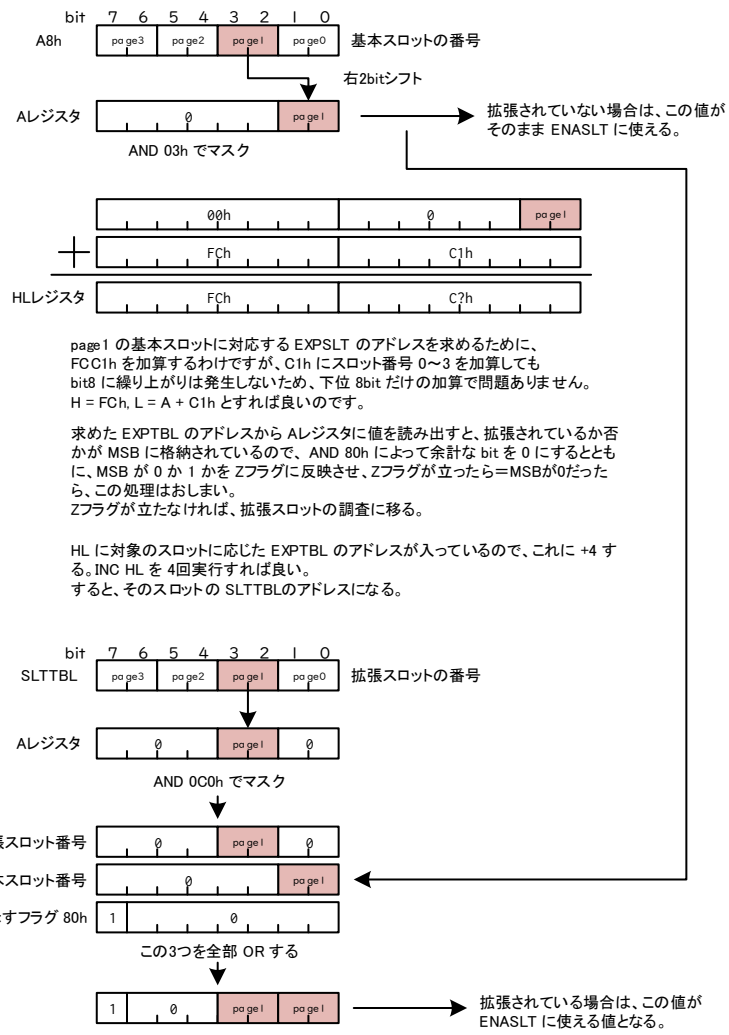
さらに、各スロットが拡張されているか否かも、BIOS のワークエリア EXPTBL(FCC1h) に記録されている。

FCC1h .. MSB 1bit が 1 なら slot0 は拡張されている
FCC2h .. MSB 1bit が 1 なら slot1 は拡張されている
FCC3h .. MSB 1bit が 1 なら slot2 は拡張されている
FCC4h .. MSB 1bit が 1 なら slot3 は拡張されている



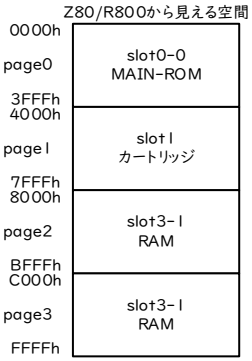
したがって、現在の page1 のスロットの番号を、ENASLTコール時に A レジスタに設定する形式の値を得るためには、下記のようにする。



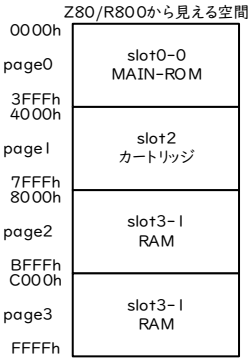


ROMの INIT が呼ばれたタイミングのスロット状態の例

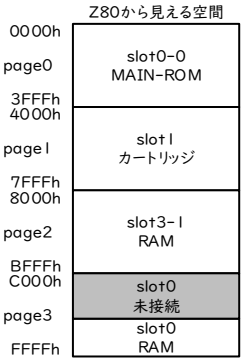
FS-A1ST/A1GT、カートリッジスロット1 に ROMを装着した場合



FS-A1ST/A1GT、カートリッジスロット2 に ROMを装着した場合



PV-7、カートリッジスロット に ROMを装着した場合



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。