

MegaCON K-Type Compatible

- ・K社の SCC搭載ゲームカートリッジに搭載のメガコン機能と互換性アリ
- ・SCC音源部は搭載していない

【カートリッジスロット と RasPiPico の接続信号】

- 4 : /SLTSL
- 12: /MERQ
- 13: /WR
- 14: /RD
- 18: A15
- 25: A14
- 26: A13
- 23: A12
- 33: D1
- 34: D0
- 35: D3
- 36: D2
- 37: D5
- 38: D4
- 39: D7
- 40: D6
- 41: GND
- 43: GND
- 45: +5V
- 46: +5V

【RasPiPico <=> ROM の接続信号】

- MA20~MA13 <=> A20~A13
- /OE <=> /OE
- /OE <=> /OE
- /WE <=> /WE

【カートリッジスロット <=> ROM の接続信号】

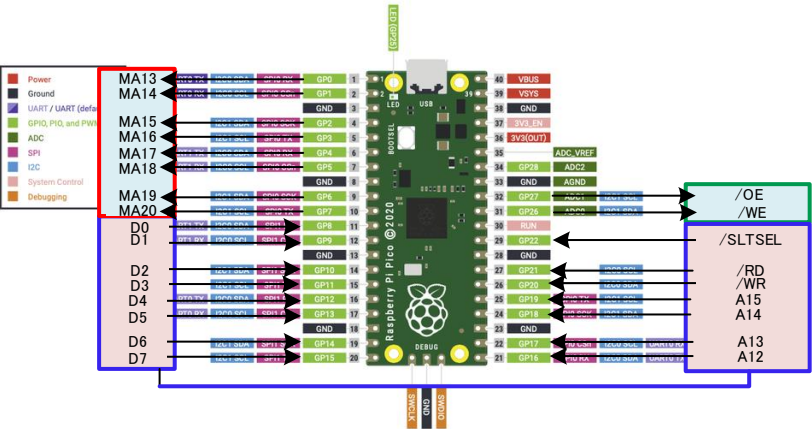
- A12~A0 <=> A12~A0
- D7~D0 <=> D7~D0
- +5V <=> VDD
- GND <=> VSS

【考え方】

A14, A13 の値によって 4つのバンクレジスタの中から1つが選択される。
選択されたバンクレジスタの値が MA20~MA13 として出力される。

Z80のDRAMリフレッシュのタイミングを考慮すると、Z80のクロックの半分程度の時間で MA20~MA13 を切り替える必要が出てくる。
これを緩和するために、/CE, /OE, /WR は Picoで作り直した信号を ROMに出す。

Raspberry Pi Pico Pinout



MSXスロットと接続

フラッシュROMと接続

ROM /CS は、MSXスロット /SLTSEL と直結

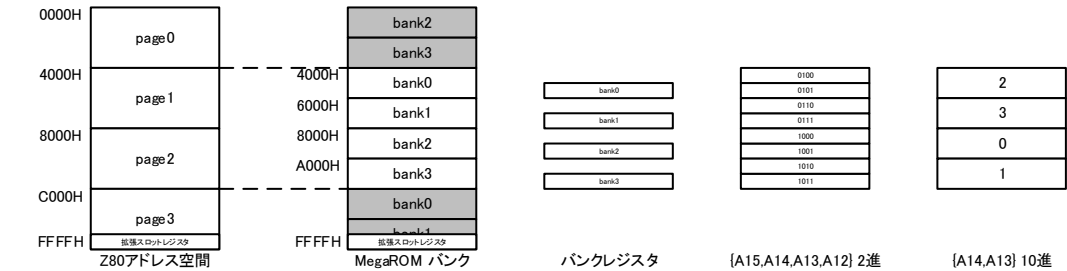


Raspberry Pi

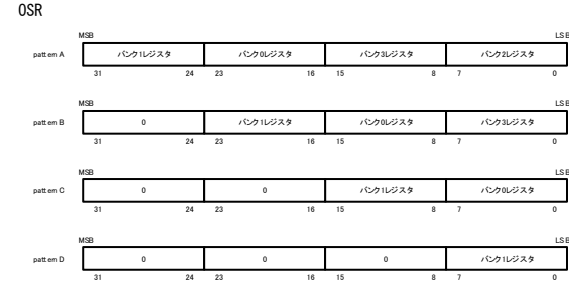
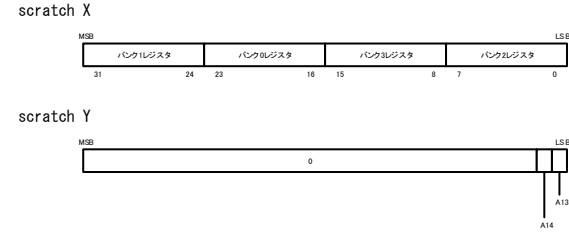
pio_bank_address (PIO0 SM0)

【役割】
/SLTSEL=0の立ち下りをキャッチして、A14, A13に対応するバンクレジスタ値を MA20～MA13 へ出力すること。

【MSXからみたバンク構成】



【各種マッピング】



```
.program pio_bank_address
.wrap_target
wait 0 gpio 22          ; /SLTSEL=0 を待つ
mov osr, pins           ; OSR = { GPI016, ..., GPI00, GPI031, ..., GPI018, GPI017 }
out isr, 1              ; ISR = { 0, 0, 0, ..., 0, GPI017 }
out Y, 1                ; Y = { 0, 0, 0, ..., 0, GPI018 }

pull noblock            ; OSR = (TX FIFO is empty) ? X : TX FIFO
mov X, osr              ; X = OSR ※ OSR の値を X にバックアップしておく
jmp !Y out_skip_bit1    ; if Y==0 goto out_skip_bit1
out NULL, 16

out_skip_bit1:
mov Y, isr              ; if Y==0 goto out_skip_bit0
jmp !Y out_skip_bit0
out NULL, 8

out_skip_bit0:
out pins, 8
wait 1 gpio 22          ; /SLTSEL=1 を待つ
.wrap
```

13 words

【I/Oマッピング】

sm_config_set_in_pins(&c, 17);
GPI017, GPI018 を IN の下位 2bit に割り付ける。
GPI018 が A14, GPI017 が A13 に接続されている想定。

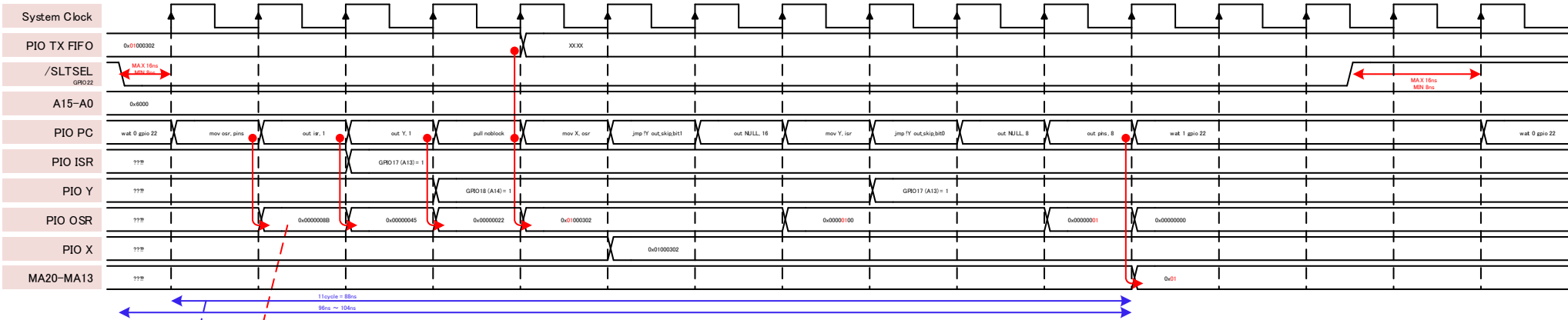
sm_config_set_out_pins(&c, 0, 8);
GPI00～GPI07 を OUT の下位 8bit に割り付ける。
GPI00～GPI07 が MA20～MA13 に接続されている想定。

【実行の前提】

pio_sm_set_enabled() で実行する前に、pio_sm_put_blocking() で FIFO にバンクレジスタの初期値を詰めておくこと。

【タイミングチャート】

(1) 一番時間がかかるケース : { A14, A13 } = { 1, 1 }



mov命令の第2オペランドに pins が指定されると、sm_config_set_in_pins(&c, 17) で設定したシフトが適用されて、第1オペランドに指定のレジスタへ格納される。
デフォルトは右シフトで、右からはみ出したビットは上位に繋げられる。

LSB
[GPI016, GPI015, ..., GPI00, 0, 0, GPI029, GPI028, ..., GPI017]
MSB

A13 = GPI017 = 1, A14 = GPI018 = 1 の例であるため、このタイミング (/SLTSEL = GPI022 = 0, /RD = GPI021 = 0, /WR = GPI020 = 1, D = GPI026 = 16 = 0, A12 = GPI016 = 0, A15 = GPI019 = 0, GPI024 = 1) では、下記のようなビット列になる。
0000_0000_0000_0000_0000_0000_1000_1011 = 0x000000B

/SLTSEL の立ち下りを、wait 0 gpio 22 で待機している。
System Clock による同期動作による wait と、
そのクロックとは非同期の /SLTSEL のタイミングであるため、
wait の検知タイミングの直後の /SLTSEL は、その次の検知タイミングまで待たされる。
よって、検知に最大で 8ns のズレが発生する可能性がある。

wait命令自体が 1cycle の動作時間を消費するので、実際の /SLTSEL の立ち下りから、wait命令終了までは 8ns ～ 16ns の時間が掛かることになる。

これに加え、wait命令の次の命令から、out pins, 8 の実行完了までが 11cycle = 88ns かかるため、/SLTSEL立ち下りから MA出力更新までは 96ns～104ns の時間が掛かることになる。

pio_bank_write (PIO0 SM1)

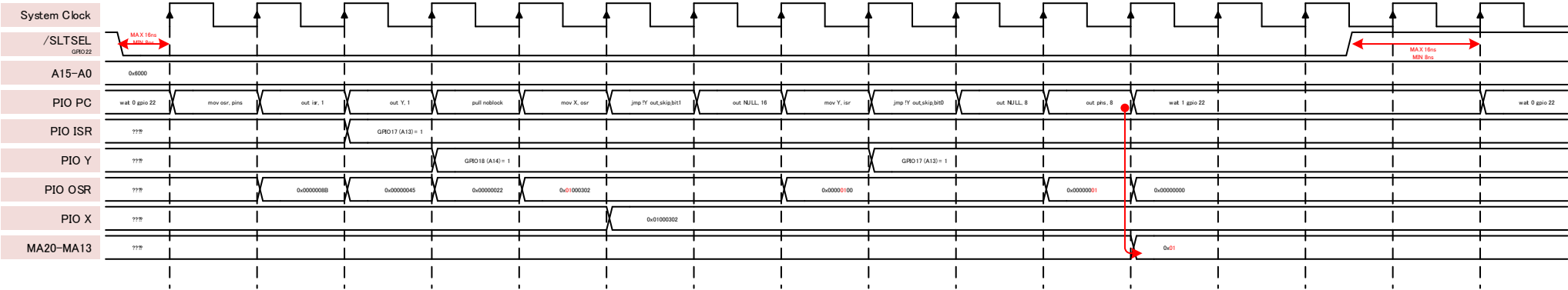
【役割】
/SLTSELの立ち下りをキャッチして、/WR=0 なら CPUへ D7-D0, A15-A12 を通知する。

```
.program pio_bank_write

.wrap_target
wait_sltssel0:
    wait 0 gpio 22      : /SLTSEL=0 を待機する
    mov  osr, pins      : OSR = { GPI07, ..., GPI010, GPI09, GPI08 }    ※ GPI08 = D0
    out  isr, 12        : ISR = { GPI019, ..., GPI08 }  (A15-A12, D7-D0)
    out  X, 1           : X = GPI020  (/WR)
    jmp  !X wr_active
wr_inactive:
    jmp  wait_sltssel1
wr_active:
    push
wait_sltssel1:
    wait 1 gpio 22      : /SLTSEL=1 を待機する
.wrap
```

8 words

【I/Oマッピング】
sm_config_set_in_pins(&c, 8);
GPI08 (D0) が入力時の LSB に来るように設定する。



pio_we_oe (PIO1 SM0)

【役割】
/SLTSELの立ち下がりをキャッチして、A15,A14,/WR,/RD を見て /WE,/OE を作る。

A15,A14	/WE	/OE
00	1	1
01	/WR	/RD
10	/WR	/RD
11	1	1

```
.program pio_we_oe

.wrap_target
    set pins, 3      : { GPIO27,GPIO26 } = { 1,1 }
wait_sltsel0:
    wait 0 gpio 22    : /SLTSEL=0 を待機する
    mov osr, pins     : OSR = { GPIO17, ... ,GPIO22,GPIO21,GPIO20,GPIO19,GPIO18 }
    out X,1           : X = GPIO18 (A14)
    out Y,1           : Y = GPIO19 (A15)
    jmp X!=Y wr_active
wr_inactive:
    set pins, 3      : { GPIO27,GPIO26 } = { 1,1 }
    jmp wait_sltsel1
wr_active:
    out X,2 [7]       : X = { GPIO21,GPIO20 } ( [/RD,/WR] ) with delay 7cyc
    mov pins, X
wait_sltsel1:
    wait 1 gpio 22    : /SLTSEL=1 を待機する
.wrap
```

11 words

【I/Oマッピング】
sm_config_set_in_pins(&c, 18);
GPIO18 (A14) が IN の bit0 に来るように設定する。
GPIO18 が A14, GPIO19 が A15, GPIO20 が /WR, GPIO21 が /RD に接続されている想定。

sm_config_set_out_pins(&c, 26, 2);
GPIO26, GPIO27 を OUT命令の出力対象とする。bit0 = GPIO26, bit = GPIO27 の 2bit

sm_config_set_set_pins(&c, 26, 2);
GPIO26, GPIO27 を SET命令の出力対象とする。bit0 = GPIO26, bit = GPIO27 の 2bit