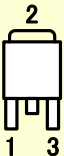


R8C/M120マイコンによる ソフトSPI 基板回路図

NJM 2845



1. Vin
2. GND
3. Vout

USB
ケーブル

両切小型
トグルSW

CP2102
モジュール



3.3V?

2極 両切小型
スライドSW

15px2

10k

1.5u

10k

1.5u

10k

1.5u

10k

1.5u

10k

1.5u

10k

1.5u

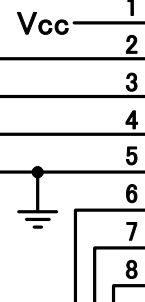
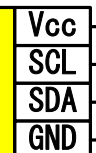
10k

1.5u

R8C/M120A

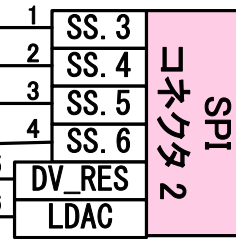
HC138

I2C
コネクタ

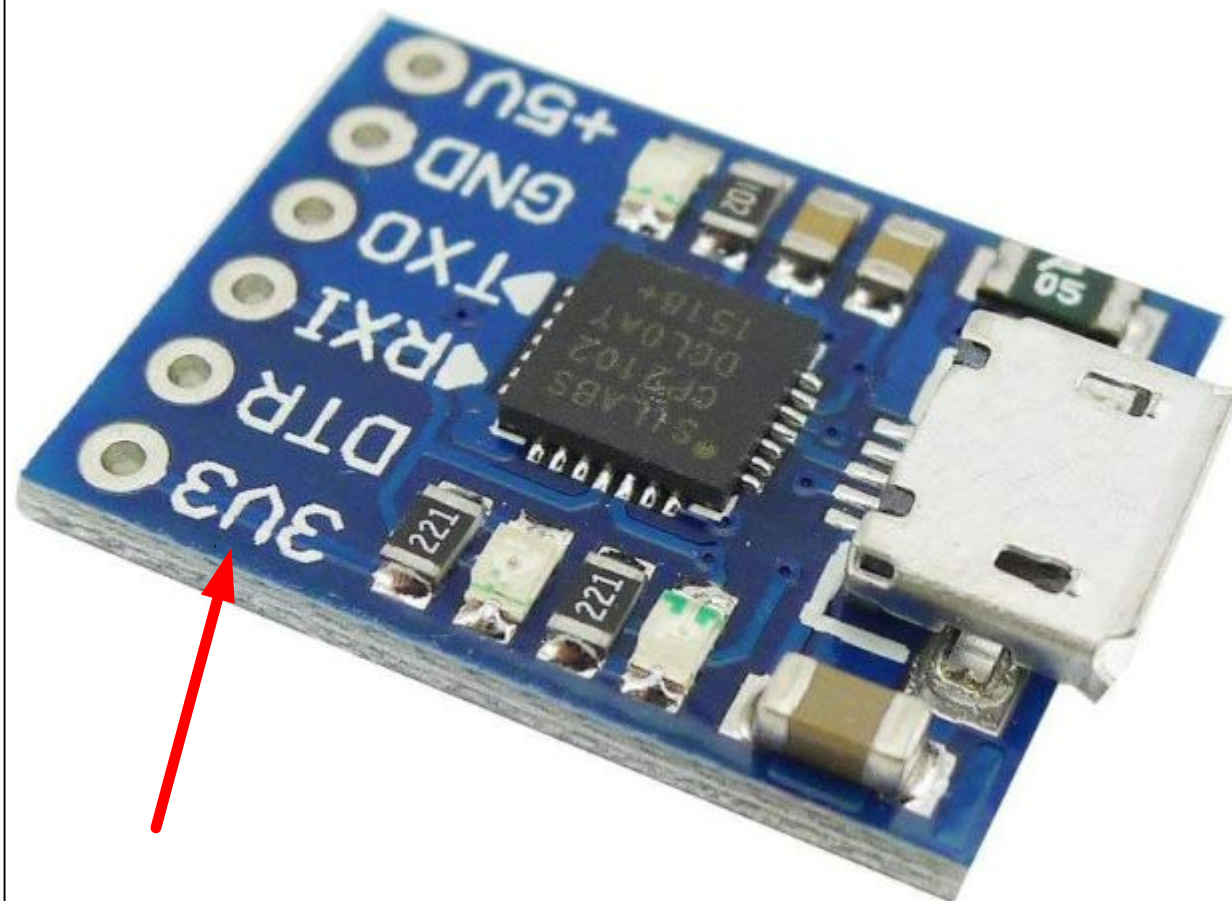


10kx8 集合R

pu1



SPI
コネクタ 2

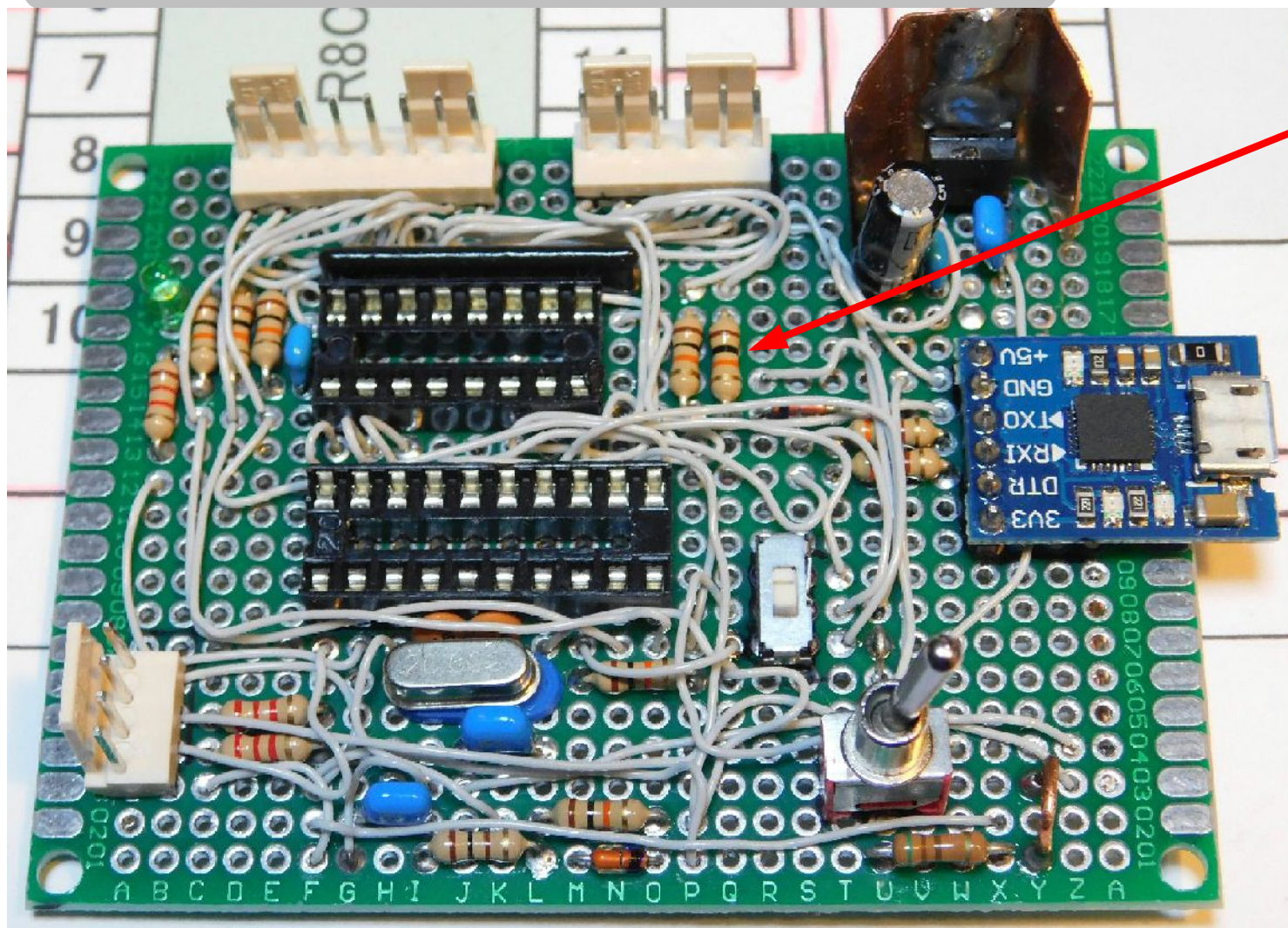


CP2102の接続端子に、3V3という端子があります。そこから、3.3Vが取り出せるのではと、思われた方も いると思います。私も、CP2102を 最初 9個購入時 そう思いました。

しかし、実際に電圧を計ってみると 4.1Vでした。 9個とも 電圧確認して 全て 4.1Vでした。

3.3V より、0.8V高い電圧では、3.3Vデバイスに 悪影響を与える恐れが ある と思い、別途 3.3Vの 3端子電源 IC を 使用しました。

DV_RESと LDACの PullUp抵抗の 取り付け



2つの PullUp 抵抗の
取り付け箇所

今回のテストに使用する SPI デバイス

今回のテストに使用する SPI デバイスは

1. **MCP4922**
2チャンネル 12bit D/Aコンバータ
2. **23LC512**
64 KByte シリアルS-RAM
3. **MCP23S17**
2Byte I/O エクスパンダ

の 3つです。

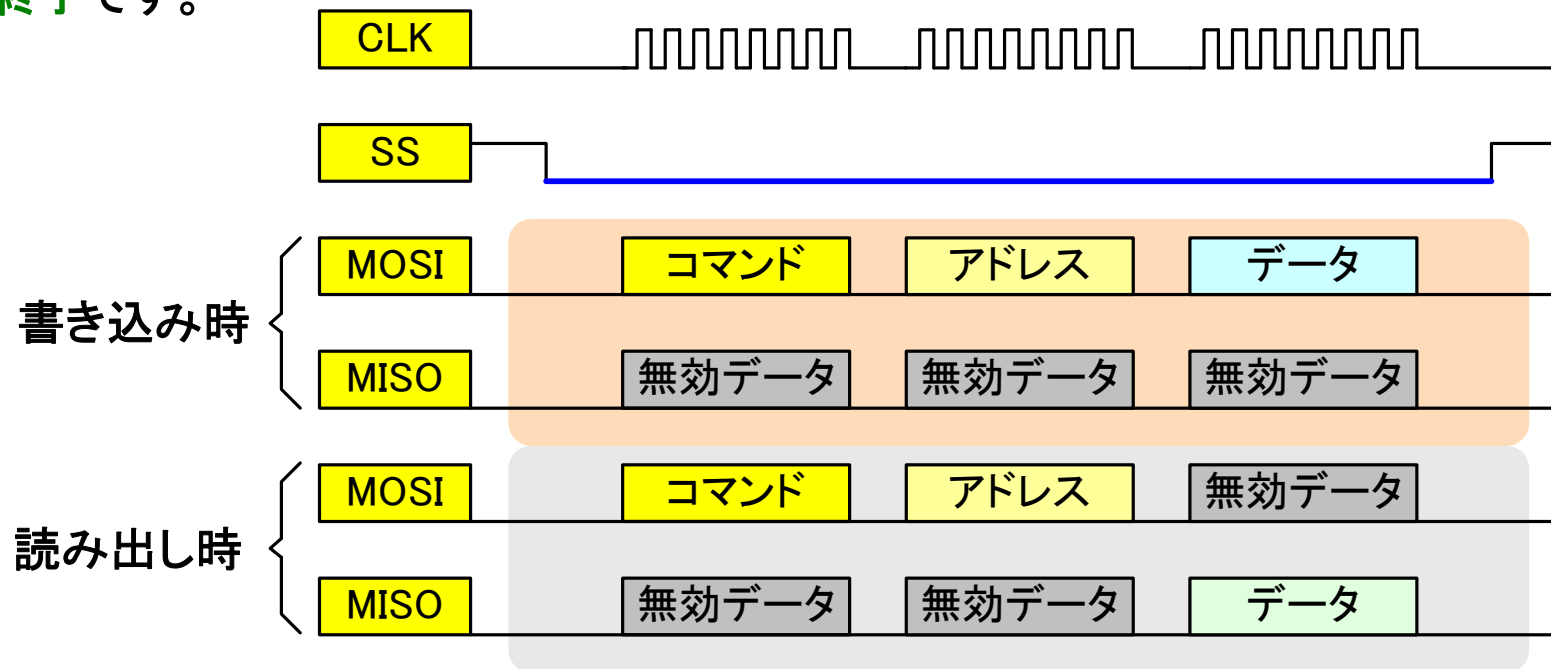
SPI アクセスの 簡単な順に 並べてみました。

その前に、SPI デバイスの データの書き込み、読み出しの電文シーケンスの説明を、行います。 例として、アクセスする電文長が、**3バイト**の **MCP23S17**を例に 説明します。

SPIの 電文シーケンス

SPIの方が、書き込み、読み出しともに **3BYTEの 転送**でシンプルです。 実は、**SPIは、3線式で 常時 全二重通信を行っている**ので 転送方向 切り替えの必要が 無いのです。

マスタ側で、データを受信したいなら、取り出せばいいし、データは、送ってくるけど必要無いなら空読みすればいい。 という事です。 それと、**I2C**には、スタート、ストップコンディションが、ありましたが、**SPI**の場合は、**SS**信号が **Low** になったら **通信の開始**で、SS信号が **Hi** になったら**通信の終了**です。



無効データは、スレーブ側が、受信中と判断しているため、スレーブ出力は0を出している事が多いです。 マスタが受信する時も、ダミーで0を出力します。

12bit D/A コンバータ MCP4922 電文

MCP4922のコマンドは、2Byteしかありません。
極めて単純です。

先頭バイトの 上位 4bitが コマンドで、下位 4bit
が 12bit D/A量子化数の 上位 4bit(b11 ~ b8)
となります、2バイト目に 12bit D/A量子化数の
残り 8bit(b7 ~ b0)が、並びます。

第1バイト							
A/B	BUF	G/A	SHDN	b11	b10	b9	b8

第2バイト							
b7	b6	b5	b4	b3	b2	b1	b0

第1バイト上位 4bitですが、A/Bは、0=VoutA、
1=VoutB です。 出力チャネルの選択です。

あとは、BUF=0、G/A=1、SHDN=1 で 問題ないよう
です。

チャネルAを 選択の場合: 0011

チャネルBを 選択の場合: 1011 に、なります。

それと、MCP4922から、アナログデータを 出力する
場合は、忘れやすいのが、電文を出して、SS 信号
を Hi に戻した後に、MCP4922の独自信号である、
LDAC 信号を Hi から Lowに する必要があります。
LDAC 信号の ダウンエッジで、アナログ信号が
更新出力されます。 LDAC信号は Lowにした後、
若干の Waitを入れて Hiに 戻して下さい。

12bit の D/Aコンバータなので、出来ればノイズの
少ないシリーズ電源を使用した方が、綺麗なアナロ
グ出力が 得られると思います。

SRAM 23LC512 の通信電文

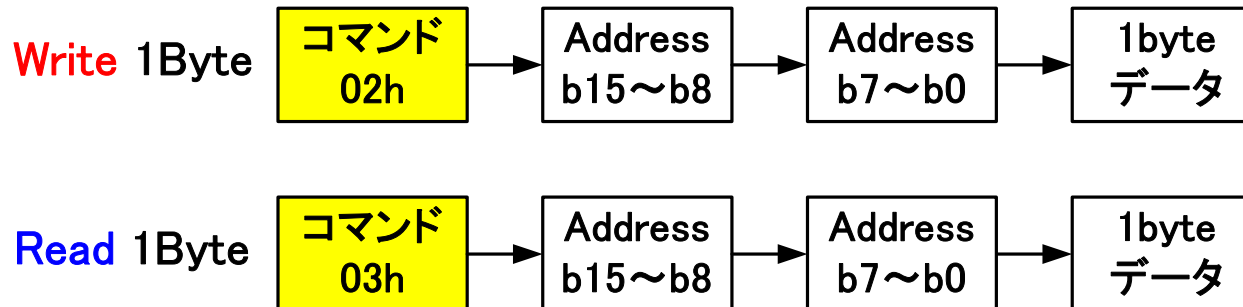
23LC512に、送る電文は、バイトモードと、ページモードと、シーケンシャルモードの 3つがあります。

今回は、単純な **バイトモード**で 通信を行います。
その場合は、**電文長は、4バイト固定**になります。

まず、**先頭に Read と Write を選択するコマンド**を送ります。23LC512は、64KByte の アドレスレンジがあるので、**アドレス情報を送るのに、2byte必要**になります。その後 **1バイトのデータ転送**を行います。

今回は、バイトモードで、転送する実験を 予定して
ますが、実使用では、**ページモード**や、**シーケンシャルモード**の方が、**転送効率が上がる**と思います。

SPIでは、選択するデバイスの SS信号を アクティブ Low にする必要が、あります。転送処理が、終了したら、SS信号を Hi に戻します。



MCP23S17と MCP23017の違い

シリアル通信のホストインターフェースが
MCP23S17が SPI で、MCP23017が I2Cです。

元々は、I2Cインタフェースのデバイスとして
作られたようで、基本的にバイト単位の
コマンド、パラメータのやり取りを行います。

データシートの タイトルが、
MCP23017／MCP23S17
16bit I/O Expander with Serial Interface と、
なっており、ホストインターフェース以外の
説明は、共通です。 いう事は、このデバイスを
制御するコマンド、パラメータの出し方は
MCP23017 と MCP23S17 で 共通です。

このデバイスで、一つ厄介なのは、 内部レジスタのアドレスが、**BANK0** と **BANK1** の2つの状態をもっている事です。 この2つの バンクのどちらを選択するかの設定が、先ほど出て来た IOCON レジスタです。 そしてこの IOCONレジスタのアドレスも、**BANK0** と **BANK1** で 異なります。

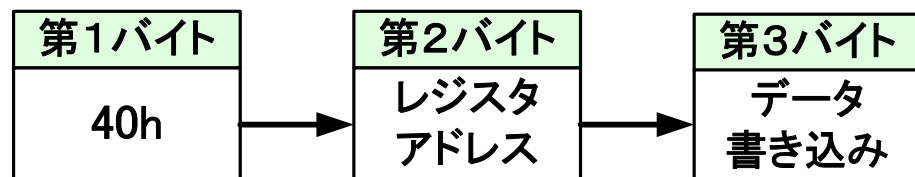
BANK0 時の **IOCON**のアドレス: **0Ah** と **0Bh**

BANK1 時の **IOCON**のアドレス: **05h** と **15h**

つまり、現在どちらの バンクになっているかが分からないと、**バンクを設定する IOCON**も **アクセス出来ない**のです。

MCP23x17の Bank設定コマンド

まず、コマンドの出し方ですが、
3バイト連続して転送します。



それと、電源ON直後のバンク設定が、どうなっているのかを、知る必要があります。

そのためには、電源ON直後のIOCONレジスタの内容を知る必要があります。データシートのP17に「制御レジスタのまとめ」の表が、有ります。IOCONの行の一番右端のPOR/RSTの値を見ます。これが電源ON直後のIOCONの値です。0000 0000 となっています。

IOCONレジスタの 各ビット:

b7	b6	b5	b4	b3	b2	b1	b0
BANK	MIRROR	SEQOP	DISSLW	HAEN	ODR	INTPOL	—

上記、IOCONの b7 が、バンクの選択ビットです。IOCONの 電源ON 初期値が 0000 0000 なので、b7 = 0 で、BANK0 が、選択されています。BANK0 の場合は IOCONのアドレスは 0Ah になります。BANK0 は、連続して複数バイトデータを書き込むと レジスタアドレスの オートインクリメントが出来るようですが、今回は単純に バイト単位で アクセスしたいので、BANK1 に変更する必要があります。IOCONの その他のビットは、データシートを参照してください。では、次のページに BANK1 に変更して、初期化するコマンドを出す手順を、示します。

MCP23x17の Bank設定含む初期化处理

MCP23S17 or MCP23017の 初期化处理を 示します。

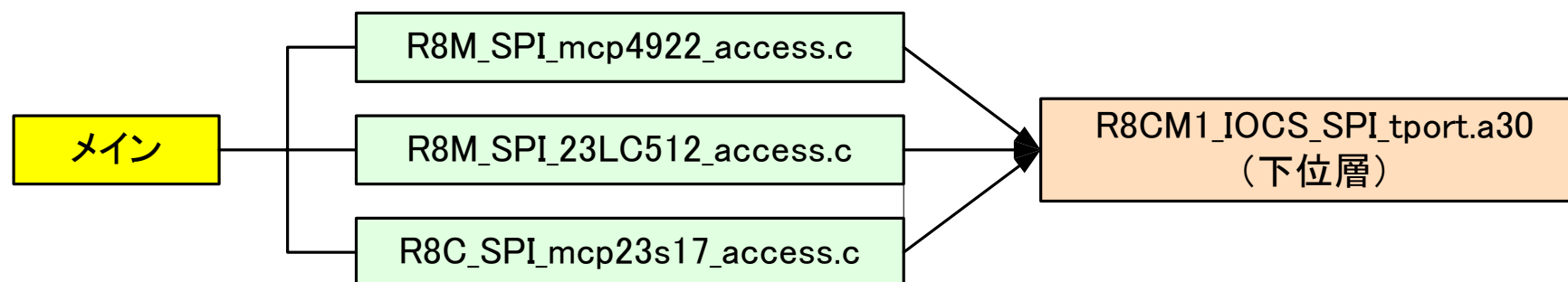
まずは、電源 ON 直後の状態で BANK0 である事を 想定して記述します。

- ① 40h、0Ah、80h // MCP23S17のモード設定 IOCON で BANK1に切り替え
 // この後、IOCON の アドレスは、05hに 変わっている。
 // BANK=1 に 変更した後に、本来の IOCON 設定を行う。
- ② 40h、05h、BCh // MCP23S17 IOCON の モード設定
- ③ 40h、01h、FFh // GPA 入力極性設定 負論理
- ④ 40h、00h、FFh // GPA 方向指定＝入力
- ⑤ 40h、19h、FFh // GPB Port Data = FF (LED点灯が負論理のため)
- ⑥ 40h、10h、00h // GPB 方向指定＝出力
- // ここで、初期化は終了(BANK1 バイトモードで 割り込み機能は 使用してない。)
- // -----
- ⑦ 40h、19h、35h // GRB LEDポートに 35h を出力
- ⑧ 41h、09h、Read Data // GRA ポートから、1バイト読み込み

尚、電源ON直後の状態で、前ページの ①～⑥ の 初期化処理を行う訳ですが、運用途中で 再初期化を行う場合は、うまく行きません。

よって、DV_RES という信号線が、有りましたが、これを、一時的に Lowにして Hi に戻す事で、MCP23x17 に 電源ON直後の状態に戻す リセットをかける事が出来ます。その後、①～⑥ の 初期化シーケンスを 行います。

今回は、R8C Mマイコン用の ソフトSPIのルーチンを用意しましたが、全体のモジュール構成を示します。



コーディングは、一通り終わってますが、実際のデバイスに接続して行う動作確認はまだ、行っていません。デバイス側基板は 以前作成した MCP4922と 23LC512と MCP23S17の基板を使用する予定ですが、一部、I2C用に改造していたので、元に戻す必要があります。