レジスタの読み書きによる外部IOのリードライト演習

gpio_poll

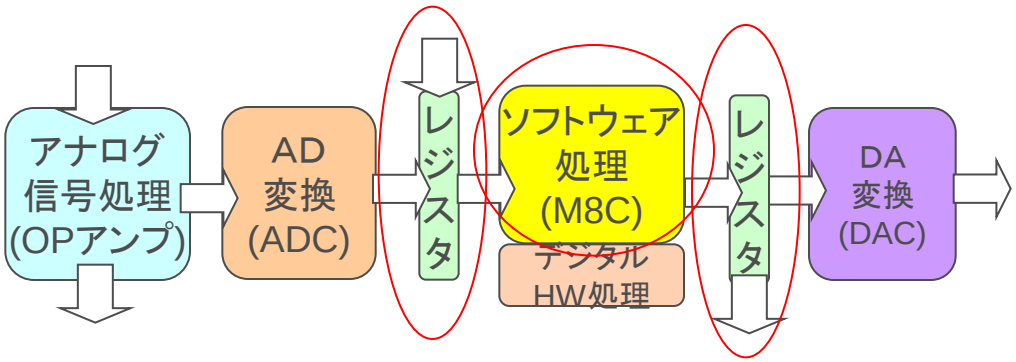
PSoC Experiment Lab *(with 4X4 MatrixKeypad)*

Experiment Course Material V1.11
April 9th., 2019
gpio_poll.pptx (16Slides)
Renji Mikami

ラボ

gpio_poll

レジスタを直接読み書きしてIOをリードライトする演習



PSoCに外部スイッチをつけるには 1

PSoCの入出力ピンに外部スイッチを接続すればシステム動作をコントロールすることができます。一見簡単そうですがデバイス外部信号の入出力インターフェースにはさまざまなポイントがあります.

1.入出力ピンのデータの読み込み

PSoCの入出力ピンはレジスタに割り付けられています.MCUは8ビット単位でこのレジスタを読み書きします.データを操作するピンが8ピンに満たない場合は他のピンのデータはマスクする必要があります.（読み込み時にはマスクで必要なピンのデータだけを取り出します.書き込み時にはまずバイト単位でレジスタのデータを読み込み変更するビットの値だけを変えてその上でバイト書き込みをします-PSoCの入出力ピン-GPIOの項目を参照してください）

2.プルアップとプルダウン

デバイスのピンの値を論理1(電圧H)論理0(電圧L)として扱うためには入出力のピンをプルアップまたはプルダウンしてスイッチを押していない状態の電圧を確定してやる必要があります.回路がオープンの状態(ハイ・インピーダンス)では論理値は0にも1にも確定しません

PSoCに外部スイッチをつけるには 2

3.ポーリングと割り込み

MCUから外部ピンの状態を監視する方法には、プログラムから定期的にレジスタの値を読み込む“ポーリング”と外部のハードウェアから動作中のMCUに対してハードウェア割り込みをかける“ハードウェア・インタラプト”という方法があります.(シンプルに割り込みと呼ぶ場合もありますが,ソフトウェア割り込みと区別する意味で,ここではハードウェア・インタラプトと呼んでいます-ハードウェア割り込みはMCUアーキテクチャレベルのものですがソフトウェア割り込みはオペレーティング・システムレベルのものです)通常は処理の速度の問題がない場合には簡単なポーリングにします.

4.チャタリング

外部スイッチは接点部分で電流のON/OFFを実現していますが、この接点ではON/OFF動作時の短い時間に”チャタリング”現象が発生し数ミリから数十ミリ秒の間電圧0と1の間で不安定な論理値を示します。これは回路上のノイズとして基板上の付加回路で解決したりソフトウェアのプログラムで解決します。

GPIO(入出力)関連レジスタ

入出力のピンは,レジスタにアサインされています。
レジスタにデータを書き込めば、ピンにデータが出力されます
ピンの状態を知るには、レジスタの値を読み込みします

ポートの入出力にはPRTxDRレジスタ変数を使用します。
　PRT1DR ： ポート1のデータリード・ライト・レジスタ

その他　PSoC Designerで設定可能なレジスタ
　PRTxDM2/DM1/DM0：動作モード設定
　PRTxIE　：割り込み発生許可・不許可レジスタ
　PRTxIC1/IC0：割り込みモード設定レジスタ
　PRTxGS：ポートをCPU・デジタルブロックで使うかを
　選択するレジスタ
＊xにはポート番号（0,1,2,3…)を入れる

GPIOの使い方

ポート(8ビット単位)の読み込み

```
unsigned char prtData,pinData;
prtData = PRT1DR;                    // Port1の状態を読み込み
pinData = PRT1DR & 0b00000001; // ビットマスクでP1.0の値を得る
pinData = PRT1DR & 0b00000010; //ビットマスクでP1.1の値を得る
```

ポート(8ビット単位)への書き込み

```
PRT1DR |= 0x01;                      // Port1.0をHighにセット
（PRT1DR = PRT1DR | 0x01;          // Port1.0をHighにセット）
PRT1DR |= 0x02;                      // Port1.1をHighにセット
PRT1DR &= 0b11111011;                // Port1.2をLowにセット
```

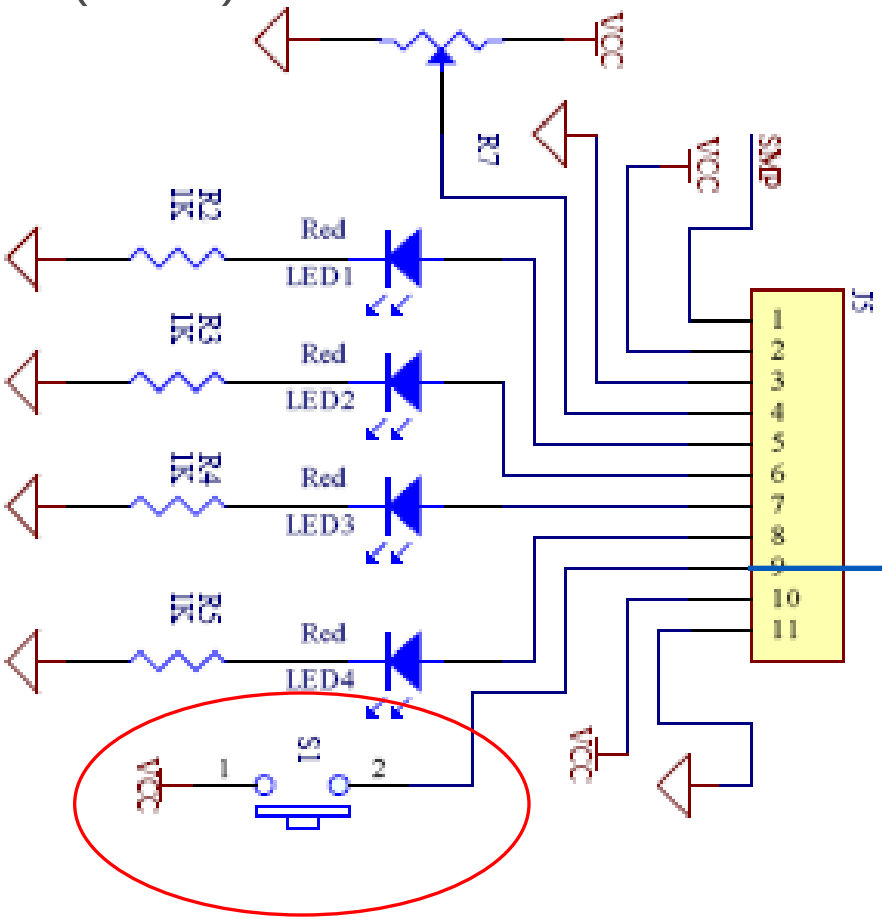
注意点

M8CにはGPIOの1ピンに対するビット命令がありません。ポート単位8ビットの読み書きしかできませんから,ビットマスクで1バイト単位でR/Wします)

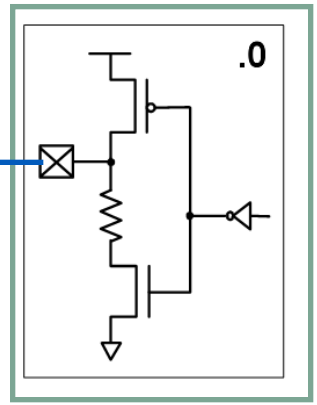
尚入力の値は実際の外部ピンの状態をそのまま読み込みます.*出力値についてはGPIOのドライブロジックに対してデータを出力しますから外部ピンの状態(値)はドライブロジックの回路に依存します.*

GPIOのプルアップとプルダウン

- S1は基板上でVCC側(5V)に接続されているので,デバイスのGPIOはプルダウンVSS(0V)側に設定しておきます。
- こうするとSWがOFFのときはピンは0V(LOW)になり,SWをONにすると5V(HIGH)になります。



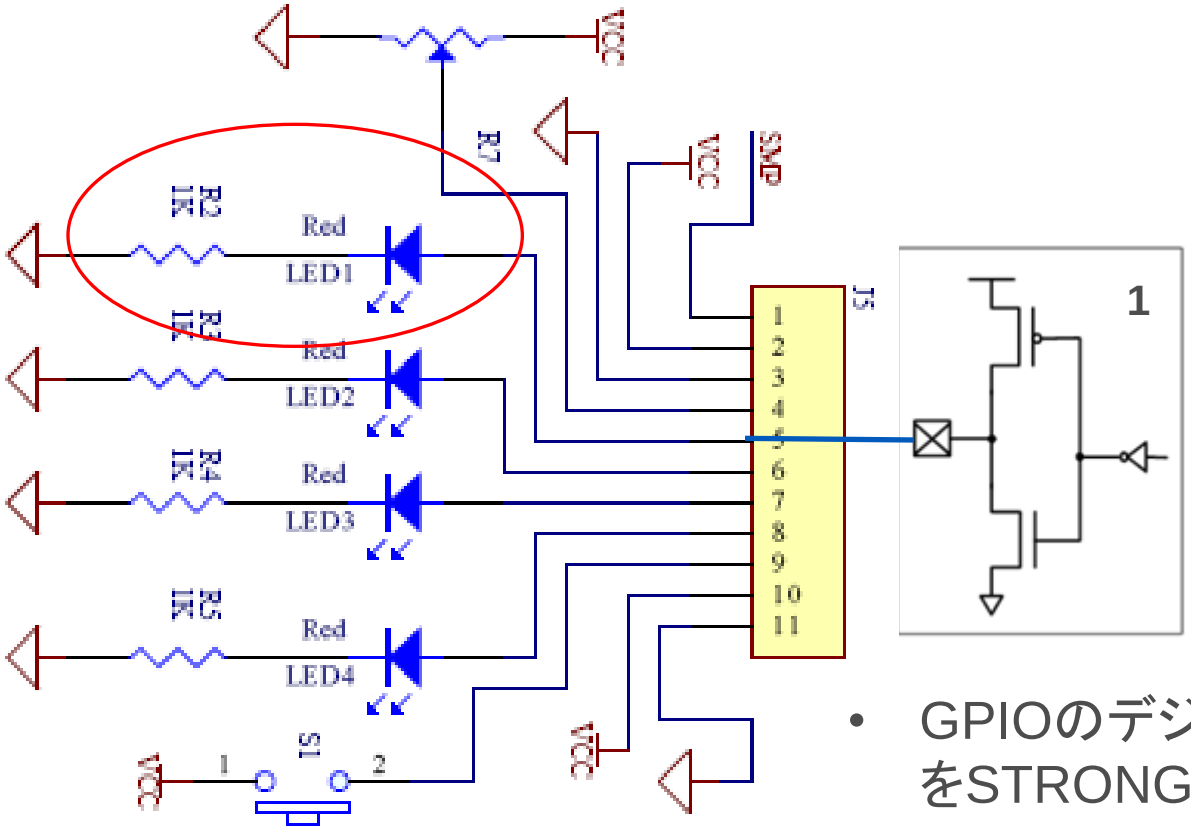
- もし基板がVSS(0V)側にピンを配線してある場合はプルアップします。この場合はON時の論理も逆になります



GPIOのプルダウン設定

基板のLED配線とGPIOの設定

- LEDは基板上でVSS(0V)側に接続されているのでデバイスのGPIOはSTRONG(出力)モードに設定する. これで出力ピンをスイッチ・ドライブしてLED電流をON/OFFできる.



- GPIOのデジタル入出力部をSTRONG設定

システム・リソース GPIOの設定

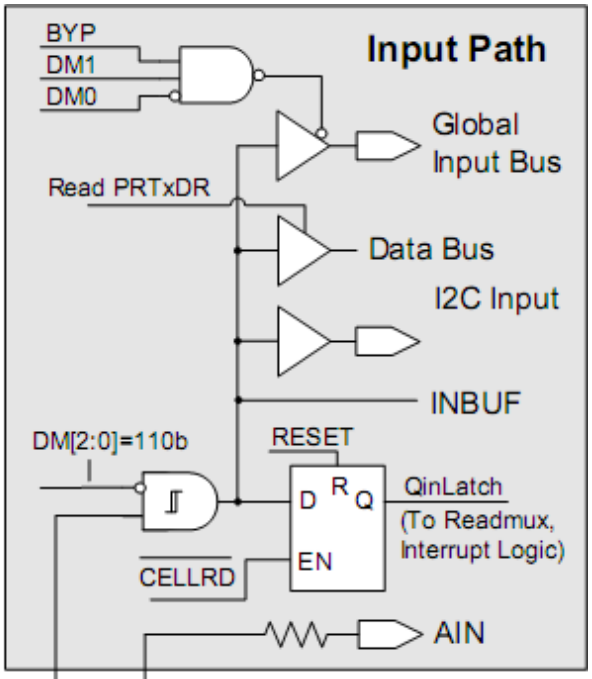
Pinout - gpio_test

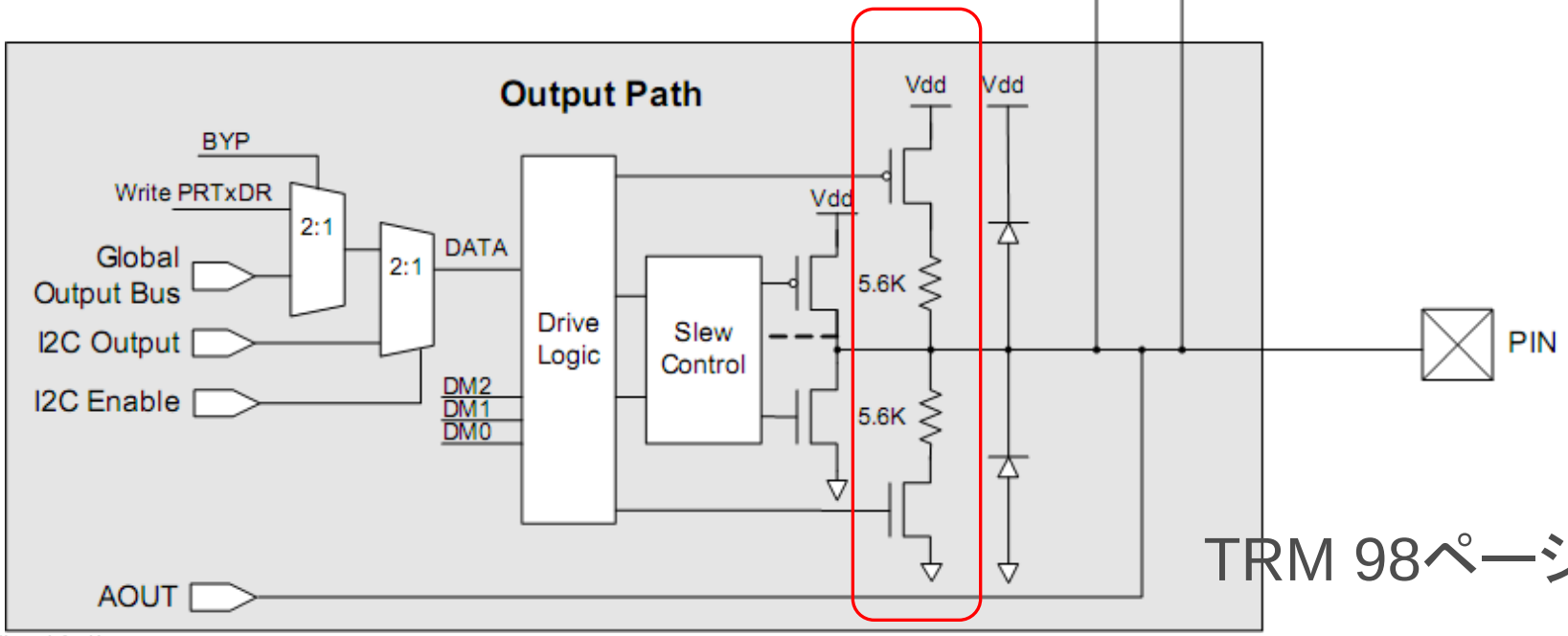
⊞ P0[7]	Port_0_7, StdCPU, High Z Analog, DisableInt
⊞ P1[0]	Port_1_0, StdCPU, Strong, DisableInt
⊞ P1[1]	Port_1_1, StdCPU, High Z Analog, DisableInt
⊞ P1[2]	Port_1_2, StdCPU, High Z Analog, DisableInt
⊞ P1[3]	Port_1_3, StdCPU, High Z Analog, DisableInt
⊞ P1[4]	Port_1_4, StdCPU, Pull Down, DisableInt
⊞ P1[5]	Port_1_5, StdCPU, High Z Analog, DisableInt
⊞ P1[6]	Port_1_6, StdCPU, High Z Analog, DisableInt

- P1[0] LED用　　Output
- P1[4] Switch用　　Input

GPIOの実際の構造

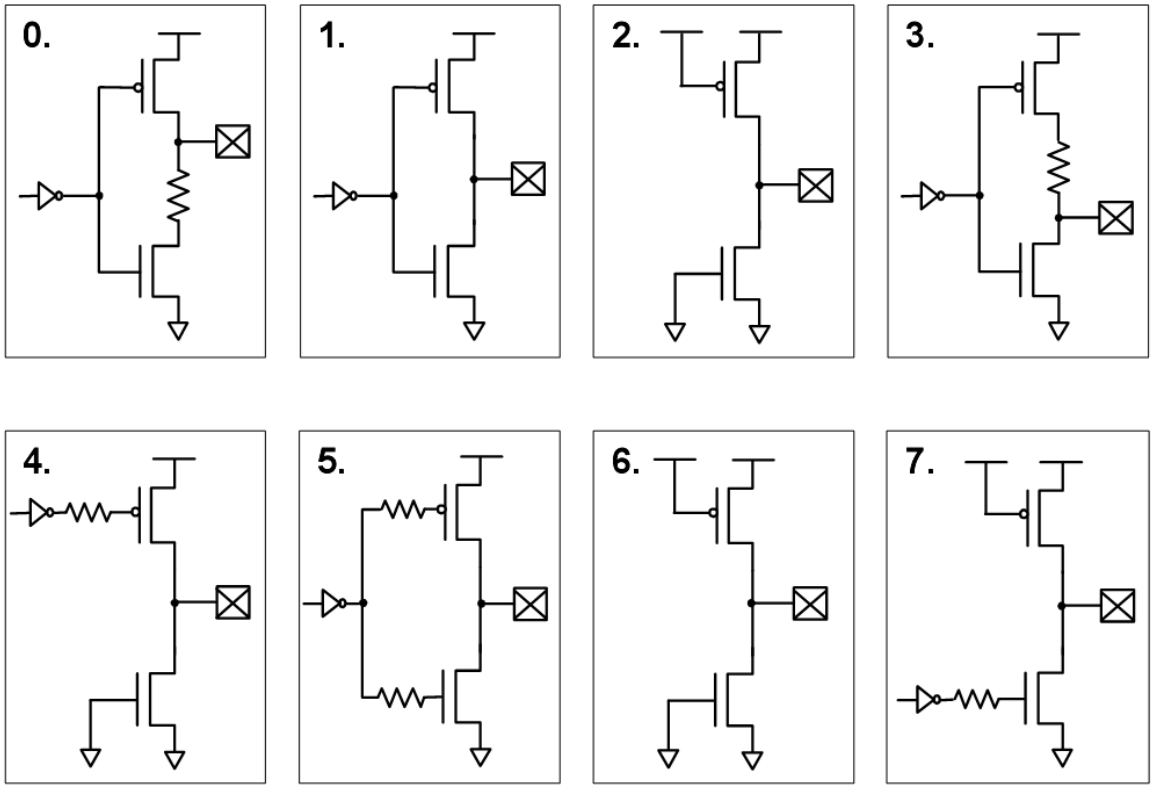
GPIOは、デジタルとアナログ双方の入力と出力が可能な構造になっています
そのため回路的には複雑ですが、
デジタルで使用する場合とアナログで使用する場合に分けて考えればすっきりします
詳細は、TRM (Technical Refeence Manual)を参照してください





TRM 98ページ

GPIOのDrive Mode(TRM 98ページ)



番号	説明
0	プルダウン
1	ストロング (出力用)
2	ハイインピーダンス
3	プルアップ
4	オープンエミッタ
5	Slowストロング (出力用)
6	ハイインピーダンスアナログ
7	オープンコレクタ

- Motor プロジェクトではアナログ回路からの出力をI/Oピンに出したので、ドライブロジック出力部はハイインピーダンスにしました

SW(P14)を押すとLED1(P10)が消える回路を作ってください

```
//-------------------------------------------------------------------------
// C main line  (参考ソースです。このソースの動きを考えて// これを修正して動かしてください)
//-------------------------------------------------------------------------
#include <m8c.h>        // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all User Modules
#define SW 0b00100000
#define LED 0b00000010
void main()
{
        PRT1DR &= (~SW);
        while(1){
                if(PRT0DR & SW)
                        {
                        PRT0DR &= (~LED);
                        PRT0DR &= (~SW);
                        }
                else {  PRT0DR |= LED;
                        PRT0DR &= (~SW);
                    }
            }
}
```

PRT1DR &= (~LED);は

PRT1DR = PRT1DR & (~LED);

と同じです

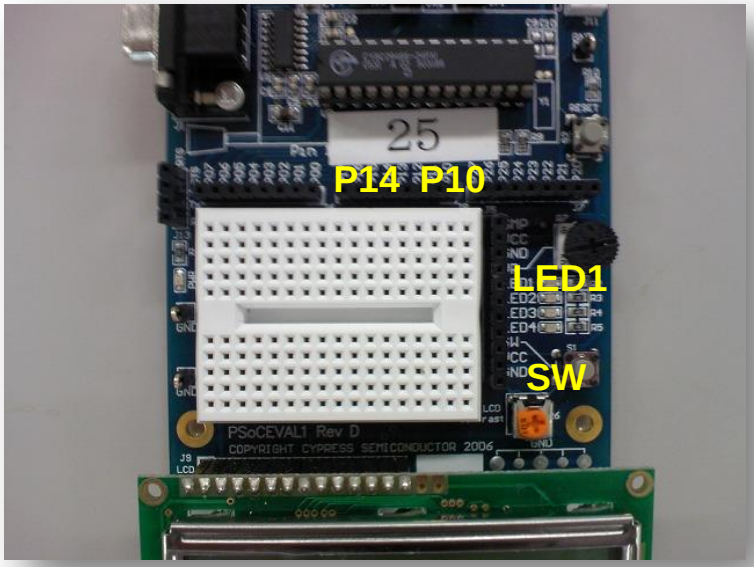
PRT1DR &= (~SW);は

PRT1DR = PRT1DR & (~SW);

と同じです。

動作を確認してみましょう

P10 → LED1
P14 → SW(CY3210)
をジャンパーで接続します。
通常はLEDがONしています。
SWを押すとLEDが消えますか？



CY3210 PSoCEval1

続いて
通常はLEDがOFFしていて
SWを押すとLEDがONにするには
どうしたらいいでしょうか

この設計では
CPUが常にSWの状態を読み込みに行きその状態によってLEDを変化させます
常にCPUからSWの状態を見にいくような方法をポーリングといいます。これに対してインタラプト(ハードウェア割り込み)という、より高度な処理方法もあります。

もっと勉強したい人へのヒント

これまでポーリングと割り込みによる処理方法を解説しましたが、これらは、シングル・プロセス(一つのアプリケション・プログラムしか走っていない状態)あるいは、オペレーティング・システムが介在していない場合の例です。オペレーティング・システムのもとで、複数のプロセスやスレッドが並行して動作している場合はどうなるでしょうか？以下に勉強のヒントになるキーワードを上げておきますので、各自で勉強を深めるときのよりどころとしてください。

Process(プロセス)

Thread(スレッド)

Fiber(ファイバー)

並列処理時の同期と排他制御

Deadlock(デッドロック)の発生

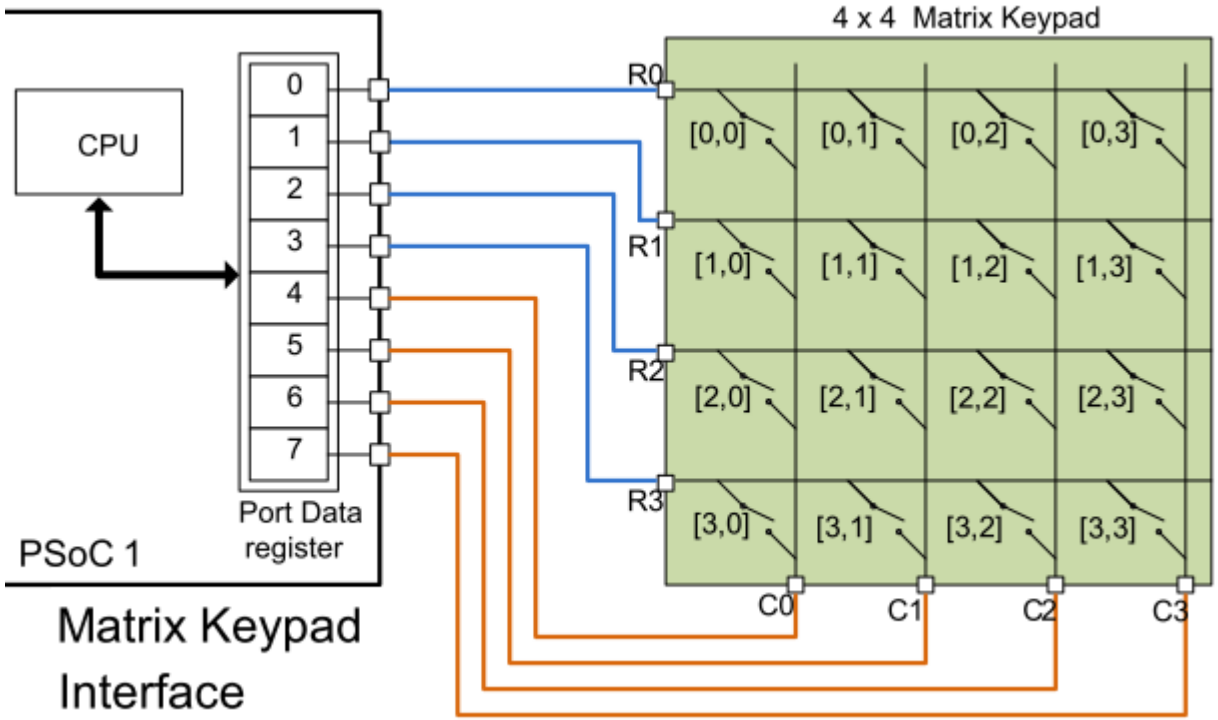
Semafo(セマフォ)

Mutex(Mutual exclusion)

応用編 (自由研究用4X4マトリックス・キーパッド)

プロジェクト・ファイルは、

psoc_lab_master_20XX/MatrixKeypad にあります。(XX>=14)



port0[0]-[3]が、行R0-R3にport0[4]-[7]が、列C0-C3に接続されています。

チャタリング対策にタイマー割込みで、スイッチをスキャンします。

Memo

フォローアップURL

http://mikami.a.la9.jp/meiji/MEIJI.HTM

担当講師

三上廉司(みかみれんじ)

Renji_Mikami(at_mark)nifty.com (Default - Recommended)

mikami(at_mark)meiji.ac.jp (Alternative)

http://mikami.a.la9.jp/_edu.htm