入出力するピンが不足した場合に入出力ピンを増やす方法

〔マイコン使用による電子部品の使い方に戻る〕

以前に、**74HC165**を使った「入力するピンが不足した場合に入力ピンを増やす方法」と、
74HC595/NJU3711を使った「出力するピンが不足した場合に出力ピンを増やす方法」を記事にしていますが、今回の"MCP23017"は16チャンネル有り其々のピンを入力／出力に割り付ける事が可能です、又、16ピン全てで割り込み入力とプルアップも出来て通信はI2Cを利用します。
尚、SPI通信を利用したい場合は"**MCP23S17**"を使いましょう。

MCP23017はこちら、MCP23S17はこちら とそれぞれ秋月電子通商で手に入ります。
MCP23017のデータシートはこちらです、 こちらはMCP23008用のデータシートですが日本語なので
参考になります。

MCP23017のレジスタについて

参照するレジスタのアドレスはデータシートを見て下さい、
ただぁ、BANK0/BANK1と切替えられます、切り替えるとレジスタのアドレスも変わってしまうので
注意が必要です、デフォルトはBANK0で、ここの"**skMCP230**"のライブラリもBANK0で作成しています。

MCP23008用データシートが日本語なのでこれを見ればレジスタの使い方も解ると思いますので、ここでは簡単に書いて置きます。
尚、レジスタ名の"**x**"には"**A**"(GPA) or "**B**"(GPB)が入ります。

ＩＯＣＯＮレジスタの構成

ビット	7	6	5	4	3	2	1	0
機能	BANK	MIRROR	SEQOP	DISSLW	HAEN	ODR	INTPOL	

Bit 7　：BANK　レジスタがアドレス指定される方法を制御します
これを切替えるとレジスタアドレスが変わってしまいます。
1 ＝BANK1に切り替えます
0 ＝BANK0に切り替えます(ここのライブラリではこちらを使います)

Bit 6　：MIRROR　割り込み発生時のINTA/INTBピンの動作を選択するビット
1 ＝INTAとINTBは内部で接続されていて同じ動作です
(**INTFA**/**INTFB**を両方読み出さないと何方で割込み発生したかは不明です)
0 ＝割り込みはINTAとINTBでそれぞれ分かれて動作します

Bit 5　：SEQOP　レジスタアドレスの自動インクリメント機能の有効無効を指定
１＝機能を無効にします
０＝機能を有効にします(ここのライブラリではこちらを使います)

Bit 4　：DISSLW　SDAピンのスルーレート機能を制御するビット
１＝制御は無効にします
０＝ハイからローに駆動するとき、SDAのスルーレートが制御されます

Bit 3　：HAEN　デバイスのアドレスを指定される方法を選択するビット(MCP23S17のみ)
１＝A0-A2のアドレスはピンで指定します
０＝A0-A2のアドレスは０で設定

Bit 2　：ODR　INTA/INTBピンをオープンドレインピンに切り替えるか選択をするビット
１＝オープンドレインに切り替えます
０＝切り替えない、アクティブのHIGH/LOW出力(TTL)

Bit 1　：INTPOL　”ODR=0”時のINTA/INTBピン出力極性を指定
１＝アクティブHIGHにします(通常はLOWで割込み発生でHIGH)
０＝アクティブLOWにします(通常はHIGHで割込み発生でLOW)

※ BANK=0とSEQOP=0は設定を変えると"skMCP230"のライブラリが動作しなくなります。

ＩＯＤＩＲｘ (I/O方向レジスタ)
０＝ピンは出力とする　１＝ピンは入力とする(デフォルト)

ＩＰＯＬｘ (入力極性ポートレジスタ)
０＝ピンの入力状態と同じ方向　１＝ピンの入力状態と反対方向(ピンがHIGHならLOWで入力される)

ＧＰＰＵｘ (プルアップ設定レジスタ)
０＝プルアップ無効とする　１＝プルアップ有効とする

ＧＰＩＯｘ (I/Oポートレジスタ)
０＝論理LOW　１＝論理HIGH

ＯＬＡＴｘ (出力ラッチレジスタ)
０＝論理LOW　１＝論理HIGH

※ ピンの電気特性はPICのピンと同等です。

割り込みレジスタについて

ＧＰＩＮＴＥＮｘ (状態変化割込み許可レジスタ)
　０＝割り込みを許可する　１＝割り込みは許可しない

ＤＥＦＶＡＬｘ (状態変化割込み方向比較レジスタ)
　０＝HIGHになったら割り込み発生(立上がり)　１＝LOWになったら割り込み発生(立下り)

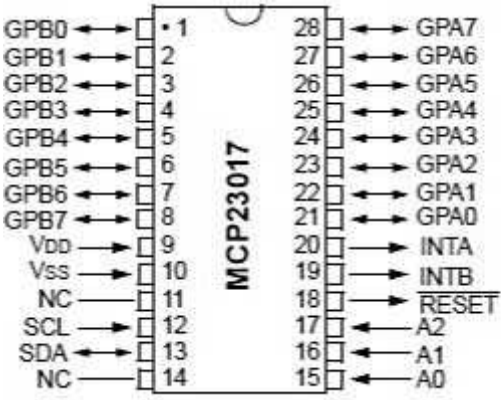
ＩＮＴＣＯＮｘ (状態変化割込み制御レジスタ)
　０＝ピンが変化すれば割り込み発生　１＝"DEFVAL"の値で割り込み発生

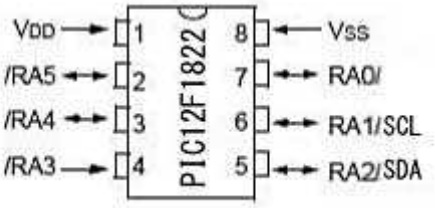
ＩＮＴＦｘ (割込みフラグレジスタ：読込みのみ)
　０＝割り込みは発生していない　１＝ピンが割り込みを発生した

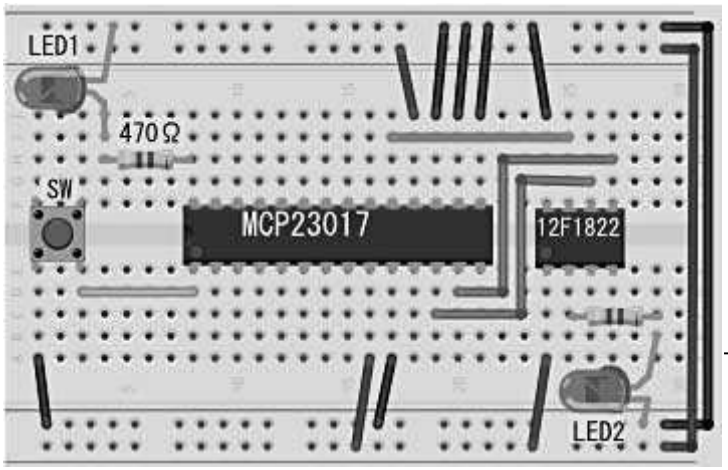
ＩＮＴＣＡＰｘ (割込みキャプチャレジスタ：読込みのみ)
　割り込みが発生した時のみビット状態をキャプチャする　０＝論理LOW　１＝論理HIGH

※ 割り込みが発生したら、"INTCAP"か"GPIO"を読み出せば割り込みはクリアされます、
　クリアされるまでは次の割り込みは発生しません。
　但し、INTA/INTBピンはピンの状態が変化するまでは出力したままです、
　例えばLOWで割込み発生しクリアしてもLOW期間中はINTピンは出力したままでHIGHで解除です。

《ＰＩＣ》







PICは12F1822を使いました。
MCP23017の電源は1.8V-5.5Vなので実験は5Vで行っています。

"**RESET**"はLOW(0V)で掛かるので、通常はHIGH(5V)に接続します。
"**A0**"･"**A1**"･"**A2**"はI2Cアドレス選択ピンです。
"**INTA**"･"**INTB**"は割込み発生で出力します。

I2C用プルアップ抵抗はPIC内蔵を使用。
SWのプルアップ抵抗はMCP23017内蔵を使用。
"**INTB**"の配線(橙色線)は、プッシュプル設定なのでプルアップは有りません。
オープンドレイン設定に変更した場合は、プルアップが必要です。

MCP23017は、"**GPA7**"ピンをLED出力とし、"**GPB0**"をスイッチ入力(プルアップ有)で設定しています。

《ダウンロードプログラムについて》

↓ここからサンプルプログラムソースファイルをダウンロードして下さい。
MCP230．lzh

プログラムソースをダウンロードしたら、**MPLAB X(v2.15)**にてプロジェクトを作成します。
以下のファイルをプロジェクトディレクトリにコピーしてプロジェクトに取込んで下さい。
次にコンパイルとＰＩＣ書き込みを実行して下さい。
MPLAB(R) XC8 C Compiler Version 1.32コンパイラを使用しています。

ダウンロードファイルを解凍すると下記の様なファイル構成です。
test1．c‥‥‥‥‥‥‥‥‥‥‥‥ 本体のサンプルプログラムソースファイル１(入出力)
test2．c‥‥‥‥‥‥‥‥‥‥‥‥ 本体のサンプルプログラムソースファイル２(割込み)
skMCP230．c‥‥‥‥‥‥‥ MCP23017と入出力を行うライブラリ関数ソースファイル
skMCP230．h‥‥‥‥‥‥‥ MCP23017と入出力を行う関数のヘッダファイル
skI2Clib．c‥‥‥‥‥‥‥‥‥‥ Ｉ２Ｃ通信を行うライブラリ関数ソースファイル
skI2Clib．h‥‥‥‥‥‥‥‥‥‥ Ｉ２Ｃ通信を行う関数のヘッダファイル

コンパイルする場合は、"test1**.**c"か"test2**.**c"の何れか一つを使いますよ、念の為。
ここの実験はPICのシステムクロック **32MHz**での実験となっています。

ｔｅｓｔ１．ｃ

このサンプルプログラムは、スイッチ(SW)を押している間だけLED1が点灯します。

ｔｅｓｔ２．ｃ

このサンプルプログラムは、割り込み実験のサンプルです。
スイッチ(GPB0ピン)はプルアップ(GPPU=1)回路なので通常はHIGH(IPOL=0)です、
押したらLOW(DEFVAL=1,INTCON=1)で割り込み発生(INTBピン)します。

PICは割り込み(RA0ピン)をループ内で監視し、発生したらLED2を１秒間点灯させています。
又、電源ON時にLED1を起動確認用で点灯させています。

以下の様にMCP23017のレジスタを設定しています。
尚、下の"GPINTENB"のビット0(GPB0)が[0]になっているので[1]に書き換えます。

```
// デバイスのＩ／Ｏレジスタコンフィグ
#define IODIRB                  0b00000001      // GPB0は入力、GPB1-GPB7ピンは出力に割当てる
#define GPPUB                   0b00000001      // GPB0は有効、GPB1-GPB7ピンはプルアップ無効とする
// 割り込みレジスタ関連のコンフィグ
#define GPINTENB                0b00000001      // GPB0は割込み有効、GPB1-GPB7ピンは割込み無効に設定
#define DEFVALB                 0b00000001      // GPB0ピン割込み時の方向(LOWで割込み発生)
#define INTCONB                 0b00000001      // GPBピン変化方向の割込み制御
// BANK0,自動インクリメント有効,SDAのスルーレートON,INT出力はアクティブHIGH,INTA/INTB個別出力
#define IOCON                   0b00000010
```

ｓｋＭＣＰ２３０．ｈ

以下のMCP23017用レジスタ設定記述が有るので、自分の回路に合わせて変更する必要が有ります。

```
// デバイスのＩ／Ｏレジスタコンフィグ
#define IODIRA                  0b00000000      // GPA0-GPA7ピンは全て出力(0=OUTPUT 1=INPUT)
#define IODIRB                  0b00000001      // GPB0は入力、GPB1-GPB7ピンは出力に割当てる
#define IPOLA                   0b00000000      // GPA0-GPA7ピンは全て正極性(0=正極性 1=逆極性)
#define IPOLB                   0b00000000      // GPB0-GPB7ピンは全て正極性に設定
#define GPPUA                   0b00000000      // GPA0-GPA7ピンは全てプルアップ無効(0=無効 1=有効)
#define GPPUB                   0b00000001      // GPB0は有効、GPB1-GPB7ピンはプルアップ無効とする
// 割り込みレジスタ関連のコンフィグ
#define GPINTENA                0b00000000      // GPA0-GPA7ピンは全て割込み無効(0=無効 1=有効)
#define GPINTENB                0b00000000      // GPB0-GPB7ピンは全て割込み無効に設定
#define DEFVALA                 0b00000000      // GPAピン(0="１"で割込み　1="０"で割込み)
#define DEFVALB                 0b00000000      // GPBピン割込み時の方向比較値
#define INTCONA                 0b00000000      // GPAピン(0=変化時に割込み　1=DEFVAL値で割込み)
#define INTCONB                 0b00000000      // GPBピン変化方向の割込み制御
// I/Oコンフィグレーションレジスタ
// BANK0,自動インクリメント有効,SDAのスルーレートON,INT出力はアクティブHIGH,INTA/INTB個別出力
#define IOCON                   0b00000010
```

Ｉ２Ｃスレーブアドレスについて

アドレスは7ビットで表します、左図の１～７ビットです。
０ビット目はR/Wでこれはスレーブに対する読書き指示ビットです。
R/W=0 ： 書き込み要求です(スレーブは受信モード)
R/W=1 ： 読み込み要求です(スレーブは送信モード)

MCP23017は"**A0**"･"**A1**"･"**A2**"端子でアドレスを変更できます。
端子を0V(VSS:GND)に接続でLOW(0)、5V(VDD)に接続でHIGH(1)となります。
[7][6][5][4][A2][A1][A0][RW] で [0][1][0][0][A2][A1][A0][RW]となり7-4ビットは固定です。

よってぇ、こんかいわぁ、"A0"･"A1"･"A2"ぜんぶぅ、GND配線だからぁ、
[0][1][0][0][0][0][0][RW]なのでぇ、”0100000”(0x20)で～すぅ。
以下の様に記述していま～ぁすぅ。
#define MCP230_ADDRESS 0b0100000 // MCP23017のI2Cアドレス(A2=0,A1=0,A0=0)
他のアドレスにしたい人は、変更してちょんまげぇ。

って事はぁ、MCP23017を８個接続可能ですね！、

8x16=128チャンネル　ウォー ш(゜ロ゜ш) スッゴイ　ヾ(- -;)　いやいや其処まで使わないから

ｓｋＭＣＰ２３０．ｃ

このライブラリはMCP23017に入出力する為の関数集です。
この関数集自体は他のPIC12F/16F/18F系統で動作すると思います。

MCP23017にアクセスを行う関数の使い方を説明します。

ans = expIO_Init( )
MCP23017の初期化を行う処理
MCP23017のピン情報は"skMCP230.h"に記述します。
ans ：0=正常 1=異常(相手からACKが返ってこない)

expIO_Write(port,pin,value)
指定したピンに出力を行う処理
port ：出力するポートを指定する 0=GPA 1=GPB
pin　：セットするビットの位置、右端(LSB)から数えて何ビット目か(0-7)
value：セットする対象となる数値、０or１

※ 一旦現状のポート内容を読んでそれと合わせて出力していますのでその分若干出力が遅くなるかもね。

ans = expIO_Read(port,pin)
指定したピンの状態を得る処理
port ：入力するポートを指定する 0=GPA 1=GPB
pin　：読み取るビットの位置、右端(LSB)から数えて何ビット目か(0-7)
ans ：返す値は読み取ったビット値 ０か１

ans = expIO_ReadINTF(port)
指定したポートの割り込み状態(INTF)を得る処理
割り込みが発生した場合は、この関数を使って割り込み情報(INTF)を読み込んで下さい、
割り込みもクリアされます。
port ：読み込むポートを指定する 0=GPA 1=GPB
ans ：返す値はINTFレジスタの値

例）MCP23017のINTB出力をPICのRA0に接続し、RA5にLEDを繋いでいる場合

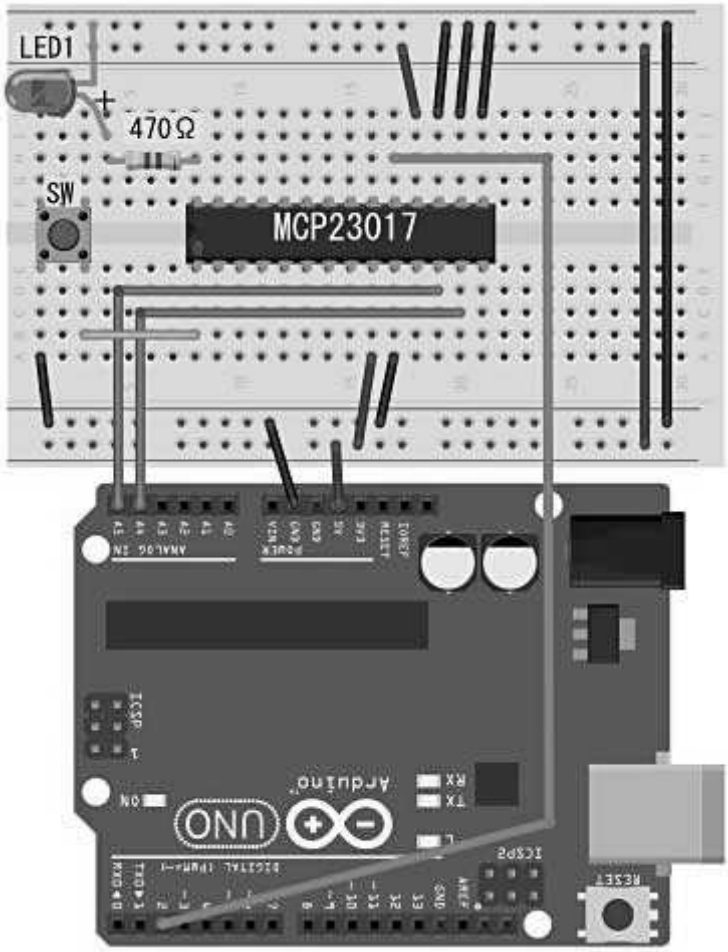
```
expIO_Init() ;
while(1) {
     if (RA0 == 1) {
          // INTBの割り込み発生
          expIO_ReadINTF(GPB) ;
          LATA5 = 1 ;
          __delay_ms(1000) ; //  １秒間点灯
     } else {
          LATA5 = 0 ;
     }
}
```

※ INTA/Bの出力をPICの状態変化ピンで受けた場合は、"expIO_ReadINTF( )"関数を
"interrupt InterFunction( )"内で処理するとI2C割込み等とダブって上手く動作しないでしょう。

ｓｋＩ２Ｃｌｉｂ．ｃ
ｓｋＩ２Ｃｌｉｂ．ｈ

この内容は”秋月電子Ｉ２Ｃ接続小型ＬＣＤモジュールに表示を行う”を参照下さい。
MCP23017のI2C通信速度は、100KHz/400KHz/1.7MHzで対応しています。

《Ａｒｄｕｉｎｏ》



電源は5Vで行っています。
MCP23017周りの配線はPICの場合と同じで、
LED2の変りはArduino付属の13番LEDです。
又、割込み発生時の"INTB"ピンはArduinoのデジタル2番ピンに接続しています。

尚、左の回路でI2C用のプルアップ抵抗が接続されていませんが、1KΩ程度でプルアップしましょう。
(実はなくても動作してたりしますがぁ．．．)

ここの"skMCP230"のライブラリでも、
Arduino Zero(M0) Proでの動作確認は行っています。
電源は3.3Vですね。
配線は左図と若干異なります、念の為。

《ダウンロードスケッチについて》

↓ここからArduino用サンプルスケッチファイルをダウンロードして下さい。
skMCP230.zip

解凍すると下の様に展開されます。

```
[skMCP230]┬┬[examples]┬┬[Interrupt]---  Interrupt.ino
          ├  skMCP230.cpp  └[SWtoLED]-----  SWtoLED.ino
          ├  skMCP230.h
          └  keywords.txt
```

ライブラリの登録方法は、ここのページの登録方法1を参照してインストールしましょう。
I2CのスケッチはArduino標準の"Wire"ライブラリを使用しています。

Interrupt．ino　・・・・・本体のサンプルスケッチ２(割込み)
SWtoLED．ino　・・・・・本体のサンプルスケッチ１(入出力)
skMCP230．cpp　・・・・・MCP23017と入出力を行う関数ライブラリソース
skMCP230．h　・・・・・・MCP23017と入出力を行う関数のヘッダファイル
keywords．txt　・・・・・・キーワードファイル

ＳＷｔｏＬＥＤ．ｃ

このサンプルスケッチは、スイッチ(SW)を押している間だけLED1が点灯します。

Ｉｎｔｅｒｒｕｐｔ．ｃ

このサンプルスケッチは、割り込み実験のサンプルです。
スイッチ(GPB0ピン)はプルアップ(GPPU=1)回路なので通常はHIGH(IPOL=0)です、
押したらLOW(DEFVAL=1,INTCON=1)で割り込み発生(INTBピン)します。

Arduinoは割り込み(2番ピン)をループ内で監視し、発生したら13番LEDを１秒間点灯させています。
又、電源ON時にLED1を起動確認用で点灯させています。

割り込みの設定は、"skMCP230.h"に以下の様にMCP23017のレジスタを設定して下さい。

```
// 割り込みレジスタ関連のコンフィグ
#define GPINTENB                0b00000001      // GPB0は割込み有効、GPB1-GPB7ピンは割込み無効に設定
#define DEFVALB                 0b00000001      // GPB0ピン割込み時の方向(LOWで割込み発生)
#define INTCONB                 0b00000001      // GPBピン変化方向の割込み制御
```

ｓｋＭＣＰ２３０．ｈ

MCP23017と入出力を行う関数のヘッダファイルです。
関数ライブラリを利用する場合は、メニューバーの「スケッチ」→「ライブラリを使用」
→「skMCP230」を
クリック操作すれば、"#include <skMCP230.h>"がスケッチに追加されます。
まぁ、手動でキー入力しても良いんですけどね。

ｓｋＭＣＰ２３０．ｃｐｐ

まず利用する場合は下記２行をスケッチの最初に記述します。
#include <Wire.h>
#include <skMCP230.h>

setup( )関数内で下の様にＩ２Ｃを使用する為の初期化処理を記述します。("SWtoLED.ino"を参

照)
Wire.begin()；　　// マスターとする

※ I2C通信速度はArduinoでは100KHzで初期化されます。

MCP23017と入出力を行う関数の使い方を説明します。

skMCP230

MCP23017と入出力を行う関数ライブラリを使用する為に必要な宣言(初期化)を行います。
skMCP230　expIO(address)；
　address ：デバイス(スレーブ)のI2Cアドレスを指定します
"expIO"の名前は任意に変更可能です。

例）

```
// デバイスの７ビットアドレス(0100A2A1A0)
#define MCP230_ADDRESS        0b0100000        // MCP23017のI2Cアドレス(A2=0,A1=0,A0=0)

skMCP230 expIO(MCP230_ADDRESS) ;        // MCP23017ライブラリの生成を行う
```

ans = expIO.Init( )

MCP23017の初期化を行う処理
MCP23017のピン情報は"skMCP230.h"に記述します。
　ans　：戻り値　0=正常終了、それ以外はI2C通信エラーです
　　1=送ろうとしたデータが送信バッファのサイズを超えた(32バイトMAX)
　　2=スレーブ・アドレスを送信し、NACKを受信した
　　3=データ・バイトを送信し、NACKを受信した
　　4=その他のエラー
　　5=データ受信エラー

expIO.pinMode(port,pin,mode)

指定ピンの動作を入力か出力に設定する処理
　port ：指定するピンのポートを指定する 0=GPA 1=GPB
　pin　：ピンのビットの位置
　mode：動作モードを指定 INPUT/OUTPUT/INPUT_PULLUP

※ ピン情報は"skMCP230.h"にて#define記述されているので、それを変更しても良いのですが、
　Arduinoらしく"pinMode"関数を作成して見ました。
　ですが割り込み関連の設定は#define記述を変更しないとダメです、
　関数にしたい人はご自分で"pinModeINT()"なんてのを作成しましょう。

expIO.Write(port,pin,value)
ans = expIO.Read(port,pin)
ans = expIO.ReadINTF(port)

これらは上記のPIC関数と同じです。

《その他》

"expIO.pinMode( )"関数はArduino側のライブラリのみ作成していますが、PICでも利用したい人は
”コピーペ”して作成して下さい。

~(-◎ω◎) ｺﾋﾟｰ ~(-◎y◎) ﾍﾟｯ　ヾ(- -;)　それはカトちゃんペッだっちゅの

以下の様なプログラムを実行(システムクロックは32MHz)し、GPA7の出力をロジアナで見て見た。

```
while(1) {
     expIO_Write(GPA,GPA7,HIGH) ;
     expIO_Write(GPA,GPA7,LOW) ;
}
```

・I2C通信速度400KHz(実際は**250KHz**)の場合
　1.226KHz(1周期：815.5us)でのパルス周波数でした。
・I2C通信速度**454.545KHz**の場合
　1.767KHz(1周期：565.8us)でのパルス周波数でした。
実際はその他のプログラムロジックが入るので実質1KHz程のパルス出力なら出来そう。
速度を求めたいなら"MCP23S17(SPI)"は通信速度10MHzなのでぇ、そうすればぁ～

スイッチが押されたかどうかはI2C通信で見に行くので、押された瞬間のチャタリングは
あまり気にしなくても良さそう。
まあ、そういう事で、割込みを含め早い応答性を求めない回路設計で利用しましょう。
