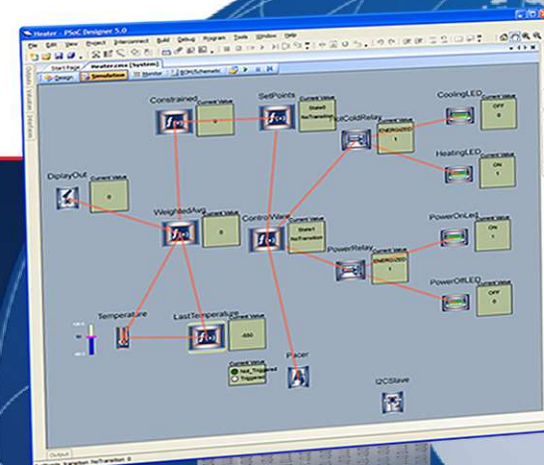


PWM 回路の演習

lab1_pwm

PSoC Experiment Lab

Experiment Course Material V1.30
October 24th, 2020
lab1_pwm.pptx (30Slides)
Renji Mikami

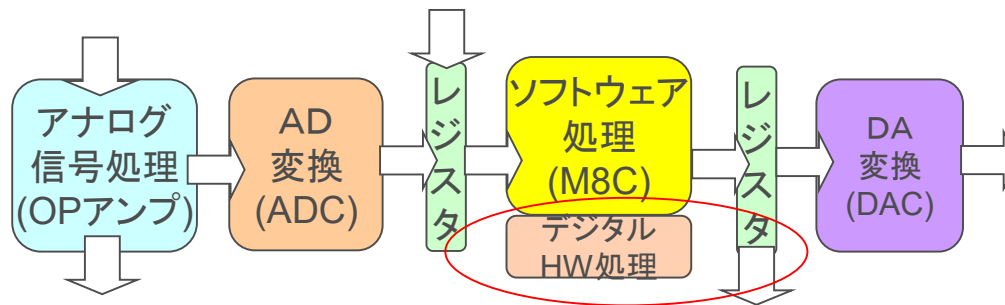




デジタル出力を使った出力の駆動, クロックリソースの演習
PWMを使用した任意の周波数(Period)と幅(Pulse Width)
を持つパルスの作り方がポイント

ラボ

lab1_pwm



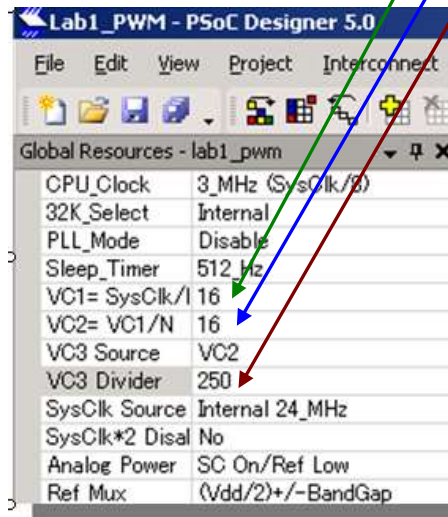


PSoCのクロック・システムについて

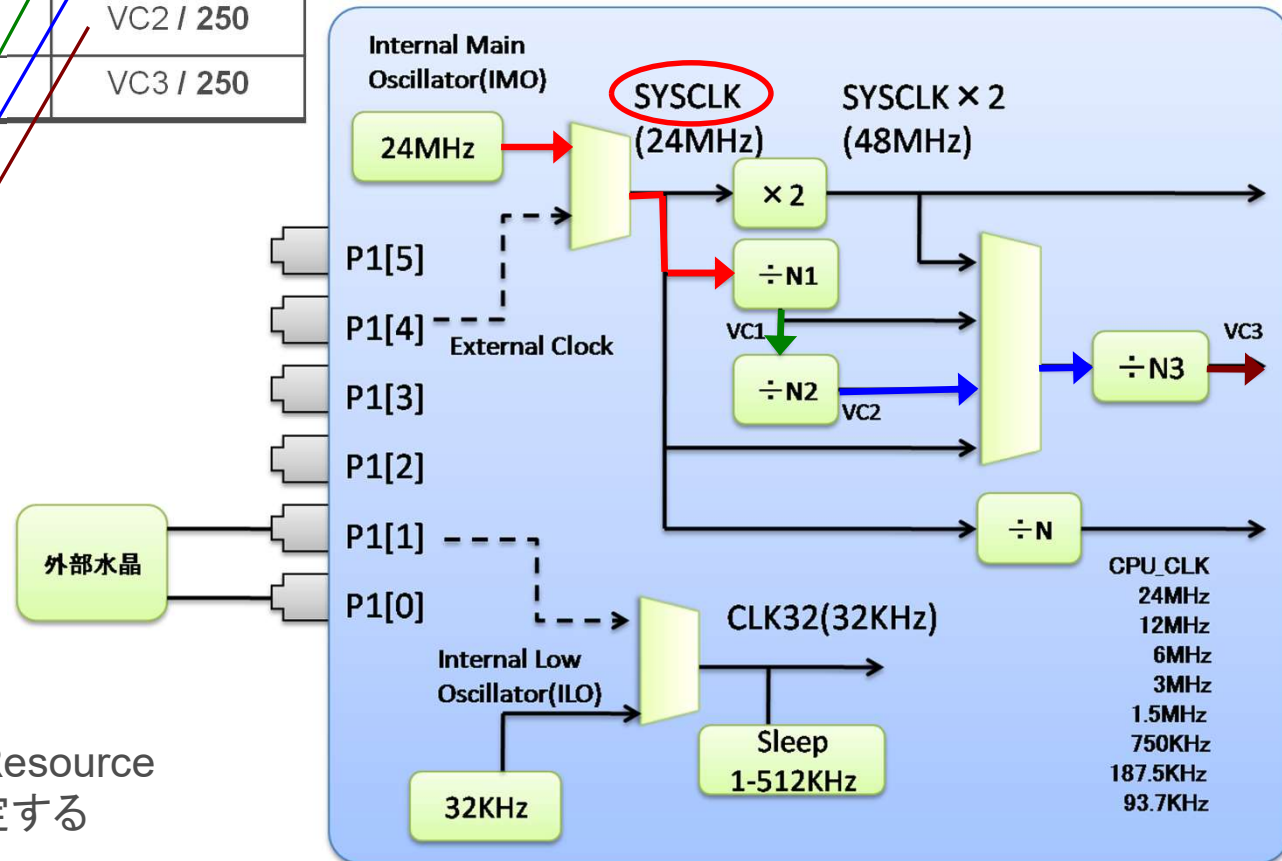
クロック	周波数	式
SysClk	24MHz	
VC1	1.5MHz	SysClk / 16
VC2	93.75KHz	VC1 / 16
VC3	375Hz	VC2 / 250
PWM8出力	1.5Hz	VC3 / 250

24MHz のSysClkを分周していき
375Hzの周波数のVC3クロックを作る

SysClk ---> VC1 ---> VC2 ---> VC3
1/16 1/16 1/250



分周比は、Global Resource
ウインドウで設定する

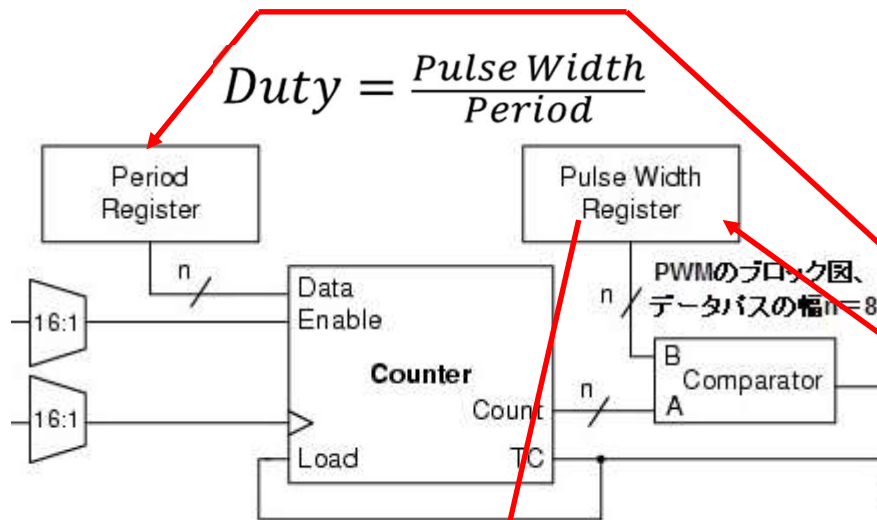
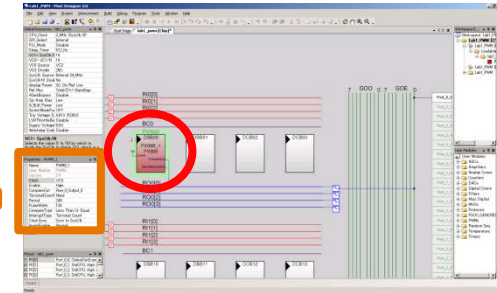
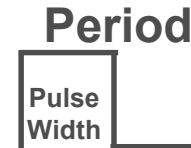




PWM8 波形決定パラメータについて

Period Register 値はPWM周波数を設定
VC3周波数が375Hz, Periodが249であればPeriod時間
は、 $1/375 * (1+249) \text{ sec} = 0.67\text{sec} \dots 1.5 \text{ Hz}$

Pulse Width Register 値はパルス幅を設定
VC3周波数が375Hz, Pulse Width が124であれば、パルス幅は、 $1/375 * (1+124) \text{ sec} = 0.33 \dots 1.5\text{Hz}$ に対して
50% duty



$$Duty = \frac{Pulse\ Width}{Period}$$

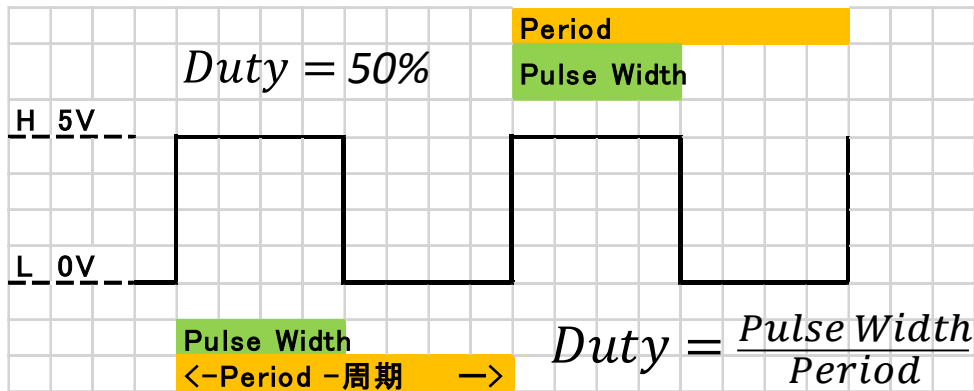
Clockの選択とPeriodで周波数、
PulseWidthでDutyを設定

Properties - PWM8_1	
Name	PWM8_1
User Module	PWM8
Version	2.5
Clock	VC3
Enable	High
CompareOut	Row_0_Output_0
TerminalCount	None
Period	249
PulseWidth	124
CompareType	Less Than Or Equal
InterruptType	Terminal Count
ClockSync	Sync to SysClk
InvertEnable	Normal

PWM波形とDuty

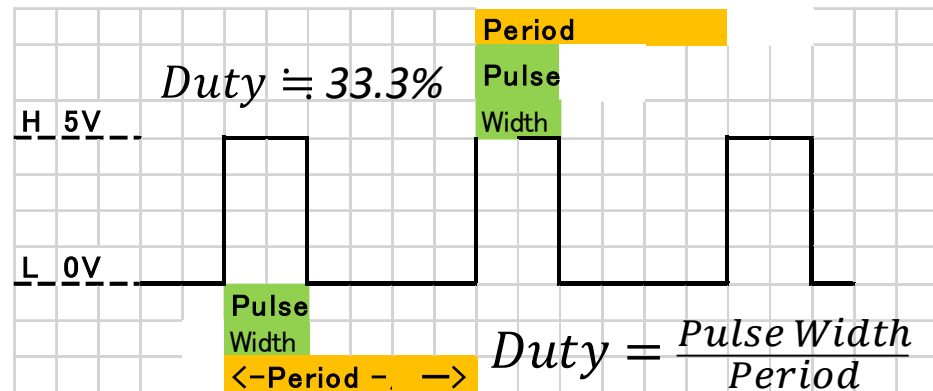
波形は、Period と Pulse Width で決まる

Duty が50%ならこのPWMの積分電圧は2.5V



PCやスマホの電源では、
スイッチングレギュレータで
PWMを使って任意の電圧を
作っている。

Duty が33.3%なら
右のPWMの積分電圧
は≒1.67V





通販のサーボモータを動かしてみる



配線の色と電源、信号

赤:+5V電源

黒:0Vグラウンド

白:制御パルス入力

秋月にて購入のGWS社のMicro STD サーボモーター
赤ラインが+5V, 黒がGND, 白が制御信号

この手のサーボの制御は,15ms から20msの周期でコントロール、パルスの幅は1ms内外と見当をつけて試してみます。

PSoCのPWM16モジュールを使用して動作させてみました。結果として,制御角度範囲は約180度, 0度に設定するパルス幅が0.7msec, 180度に設定するパルス幅が2.4msecでした。(数字はPulse Width設定値)

動作範囲を超える幅のパルスを与えると異常な動きをすること(片側によってカタカタを繰り返すなど)があります。PSoC基板では、電力が不足する場合は、外部電源を与えてください。



約2.4msec



約0.7msec



Global Resourceの設定と基本周期について

Global Resources - motor	
CPU_Clock	24_MHz (SysClk/1)
32K_Select	Internal
PLL_Mode	Disable
Sleep_Timer	512_Hz
VC1= SysClk/N	16
VC2= VC1/N	16
VC3 Source	SysClk/1
VC3 Divider	1
SysClk Source	Internal 24_MHz
SysClk*2 Disable	No
Analog Power	SC On/Ref Low
Ref Mux	(Vdd/2)+/-BandGap
AGndBypass	Disable
Op-Amp Bias	Low
A_Buff_Power	Low
SwitchModePump	OFF
Trip Voltage [LVD (SMP)]	4.81V (5.00V)
LVDThrottleBack	Disable
Supply Voltage	5.0V
Watchdog Enable	Disable

CPU Clock 24Mhz

$VC1 = 1.5\text{MHz}(24\text{MHz} \times 1/16)$

$VC2 = 93.75\text{KHz}(VC1 \times 1/16)$

VC2のクロックでPWM16を駆動

Properties - PWM16_1	
Name	PWM16_1
User Module	PWM16
Version	2.5
Clock	VC2
Enable	High
CompareOut	Row_0_Output_0
TerminalCountOut	None
Period	1835
PulseWidth	65
CompareType	Less Than Or Equal
InterruptType	Terminal Count
ClockSync	Sync to SysClk
InvertEnable	Normal

Period

PWM16でPeriodレジスタの値を設定

$93.75\text{KHz} / 1835 = \text{約}51\text{Hz} (19.6\text{msec})$

これを基本周期にしてみました。

ここではPWMの設定値は簡略化していますから正確な計算法は
かならずユーザーモジュールデータシートで確認してください



パルスの幅の設定をオシロスコープで観測した例

Pulse Width 65を設定

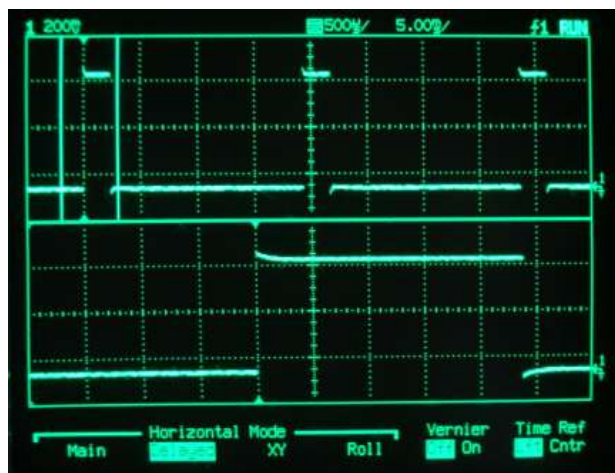
93.75KHZ = 約 0.011msecの解像度

0.011 x 65 で約0.7msecのパルスを生成

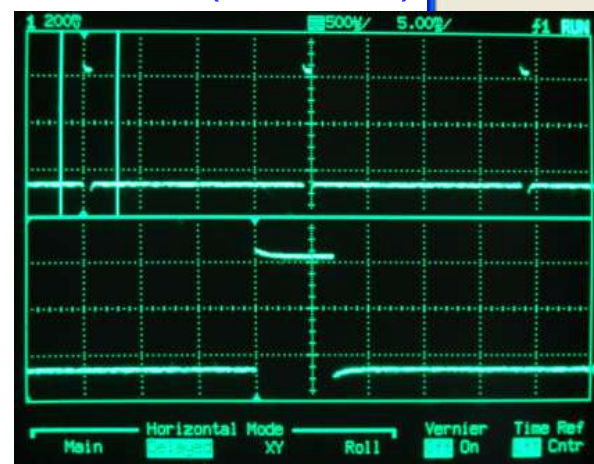
最大時は約2.4msecのためPulse Width値は225

Properties - PWM16_1	
Name	PWM16_1
User Module	PWM16
Version	2.5
Clock	VC2
Enable	High
CompareOut	Row_0_Output_0
TerminalCountOut	None
Period	1835
PulseWidth	65
CompareType	Less Than Or Equal
InterruptType	Terminal Count
ClockSync	Sync to SysClk
InvertEnable	Normal

約2.4msec (PW=225)



約0.7msec (PW=65)



パルス幅の変更で自由に回転角度を設定



プログラムソースからのAPIパラメータの制御法

PWMのプロパティ・ウィンドウで初期設定した値は, main.c から直接レジスタ値を設定することで自由に変えることができます。

プログラム上から設定するには右のように
PWM16_WritePeriod()
(Periodパラメータ設定)
PWM16_WritePulseWidth()
(Pulse幅パラメータ設定)

API関数を使用します。
PWMユーザーモジュール
データシートを参照

```
.....  
; This sample shows how to create a 33% duty cycle output  
; pulse. The clock selected should be 1000 times the required  
; period. The comparator operation is specified to be "Less than or Equal".  
.....  
/* include the Counter16 API header file */  
#include "PWM16.h"  
  
/* function prototype */  
void GenerateOneThirdDutyCycle(void);  
  
/* Divide by eight function */  
void GenerateOneThirdDutyCycle(void)  
{  
    /* set period to eight clocks */  
    PWM16_WritePeriod(999);  
    /* set pulse width to generate a 33% duty cycle */  
    PWM16_WritePulseWidth(332);  
    /* ensure interrupt is disabled */  
    PWM16_DisableInt();  
    /* start the PWM16! */  
    PWM16_Start(); }  
.....
```



ラボ lab1_pwm 手順

- 1.PWM1 ユーザーモジュールを選択
- 2.Global Resource のクロック生成を設定します
- 3.PWMのPeriod RegisterとPulse Width Register の設定を行います
- 4.LEDを点灯します.
- 5.プログラムからPWMのレジスタ値を変えてみます.

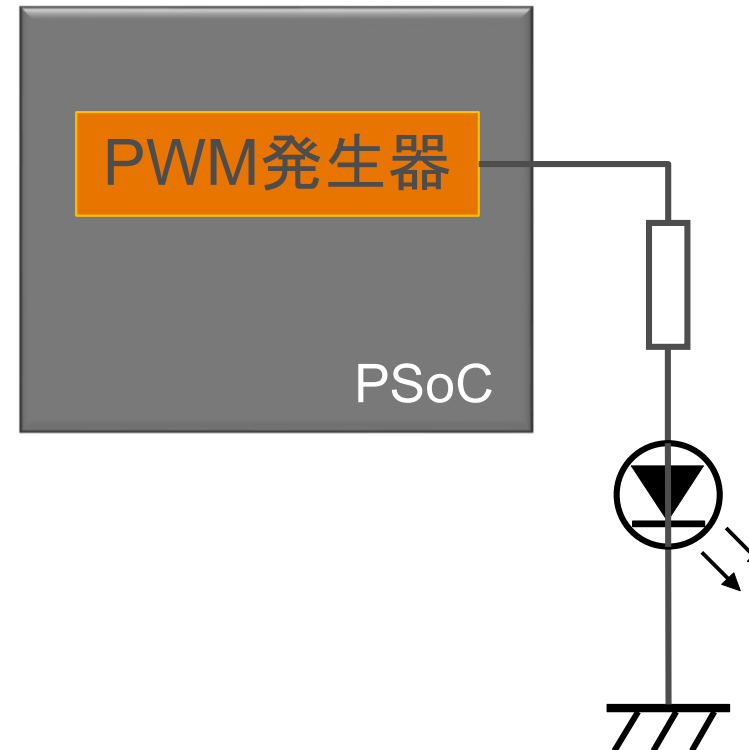
解説 :



lab1_pwm

- PWM8ユーザーモジュールを用いてLEDを1.5Hzで点滅

この設計では、
出力がデジタル
ドライブになっています。



新規プロジェクトの作成(旧版ソフトウェアの場合)

1. File > New Project をクリック

2. Chip-level Project を選択

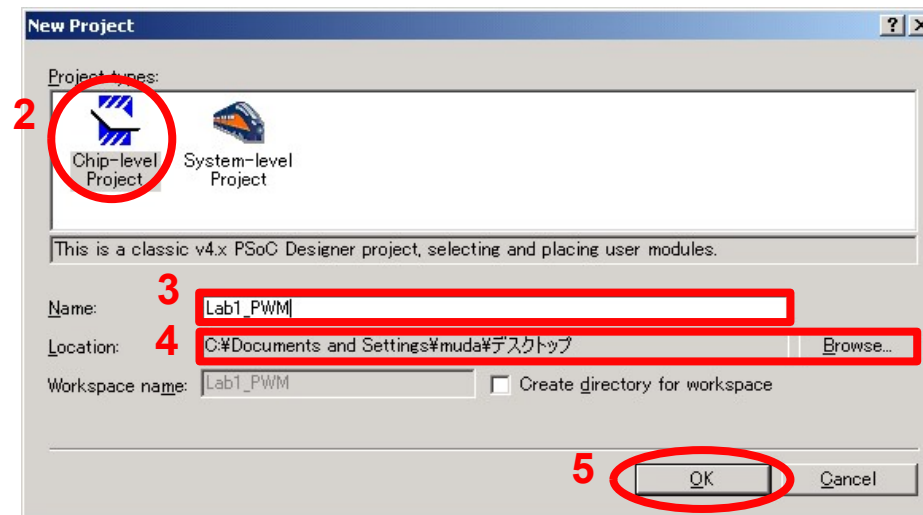
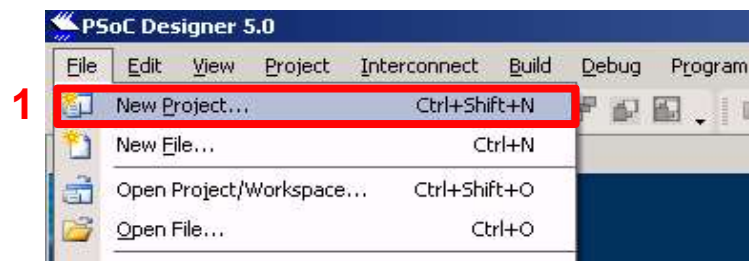
3. Name を入力

例: lab1_pwm

4. Location を選択

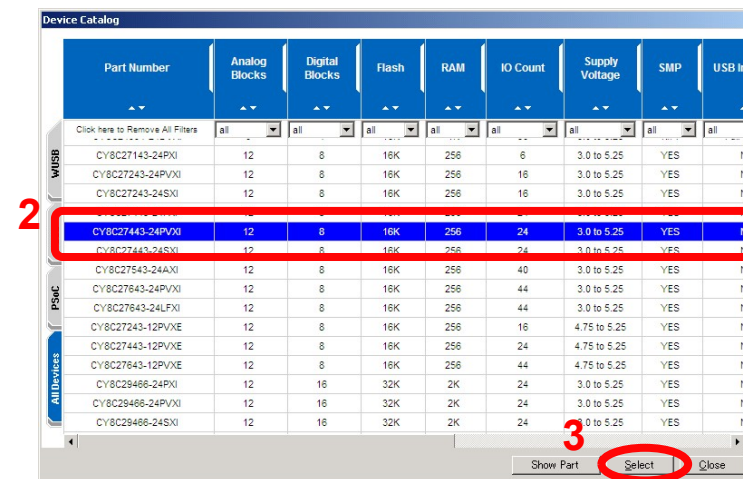
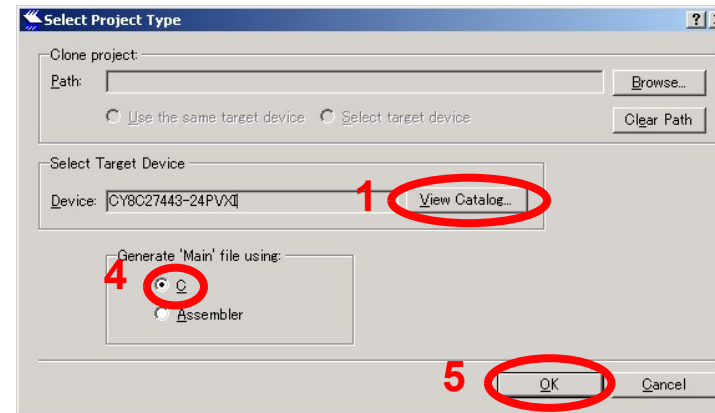
例: C:\psoc_lab\lab1_pwm

5. OK をクリック



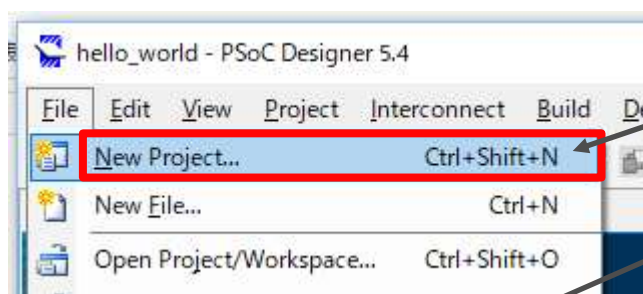
使用するPSoC、言語の選択(旧版ソフトウェアの場合)

1. View Catalog をクリック
2. CY8C27443-24PXI を選択
3. Select をクリック
4. C を選択
5. OK をクリック





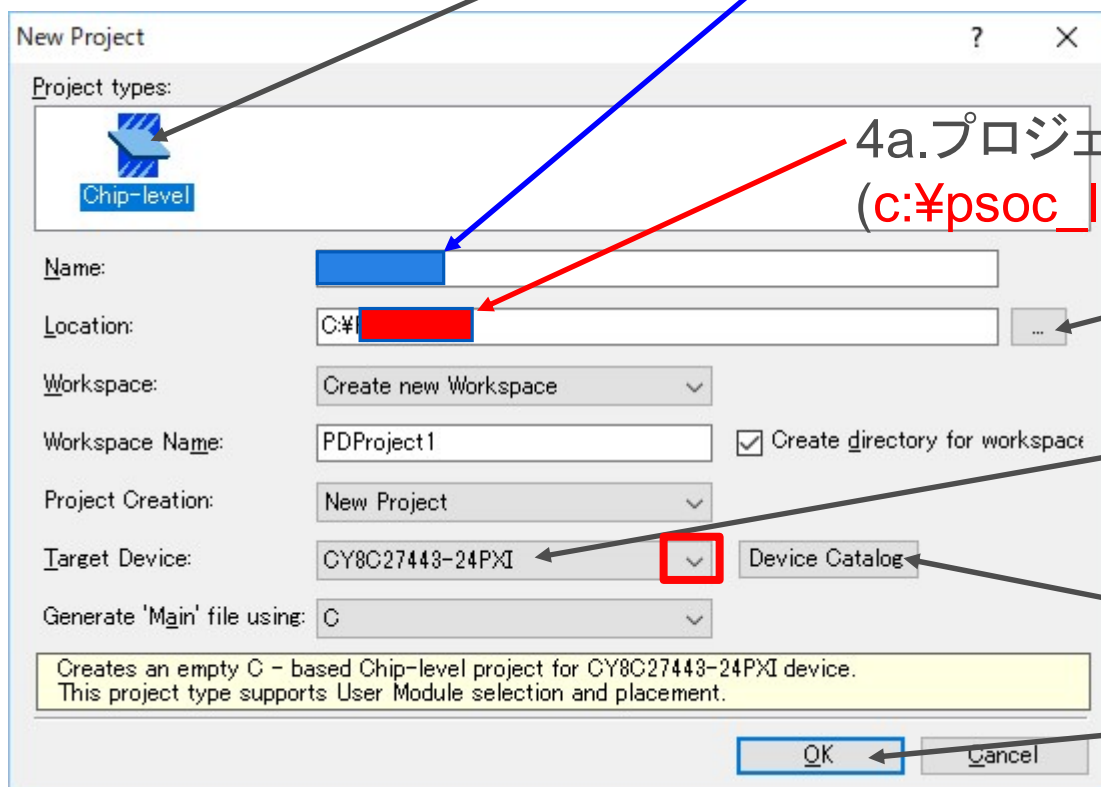
New Project: lab1_pwm の作成 (新版ソフトウェアの場合)



1. File > New Projectを
クリック

2. Chip-level Project をハイライト

3. プロジェクトの名前(lab1_pwm)を入力



4a. プロジェクトを保存するディレクトリ
(c:¥psoc_lab¥lab1_pwm)指定する。

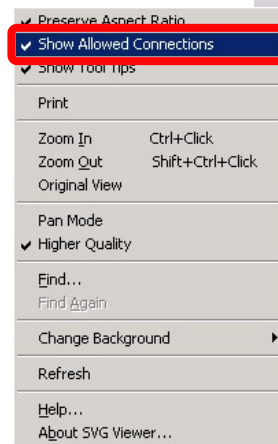
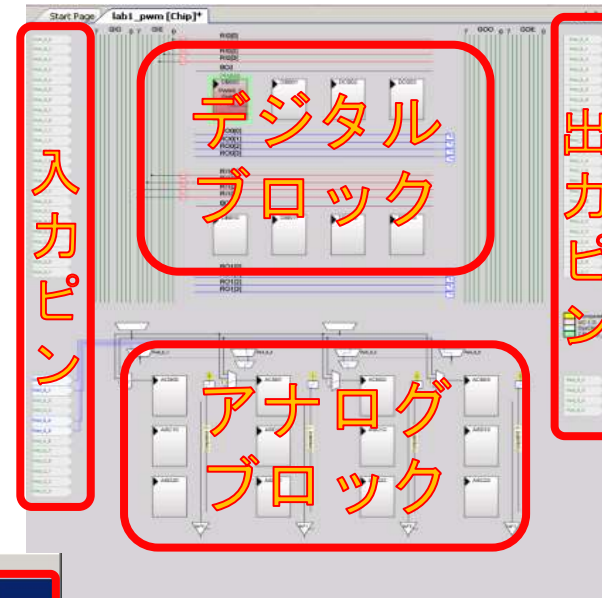
4b. 場所を選ぶときは
右の...をクリック。

5. デバイス
CY8C27443-24PXIを確認
(変更はVマークかDevice
Catalogをクリック)

6. 決まったら
OKをクリック

Chip Editor (回路図)の使い方

- Alt + ドラッグで移動
- Ctrl + クリックで拡大
- Ctrl + Shift + クリックで縮小
- 回路図上で右クリック
Show Allowed Connections
で配線候補を可視化



移動	Alt+ドラッグ
拡大	Ctrl+クリック Ctrl+ドラッグ
縮小	Ctrl+shift+クリック Ctrl+shift+ドラッグ

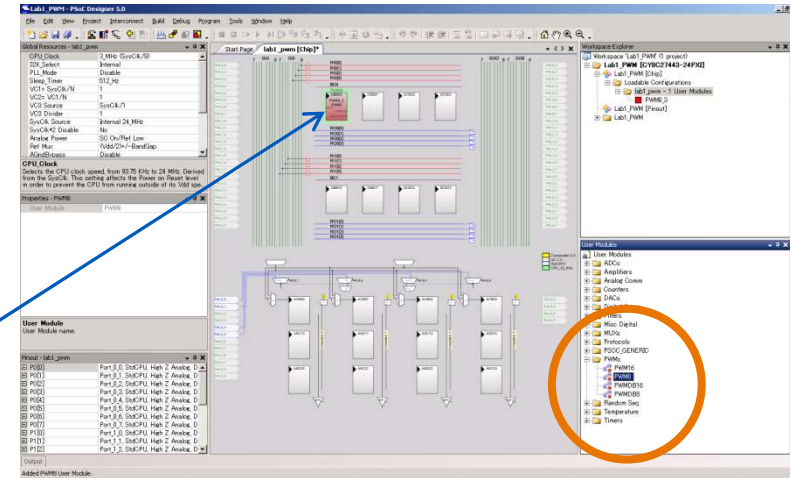
ユーザーモジュールPWM8 の追加,配置

View > User Module Catalog

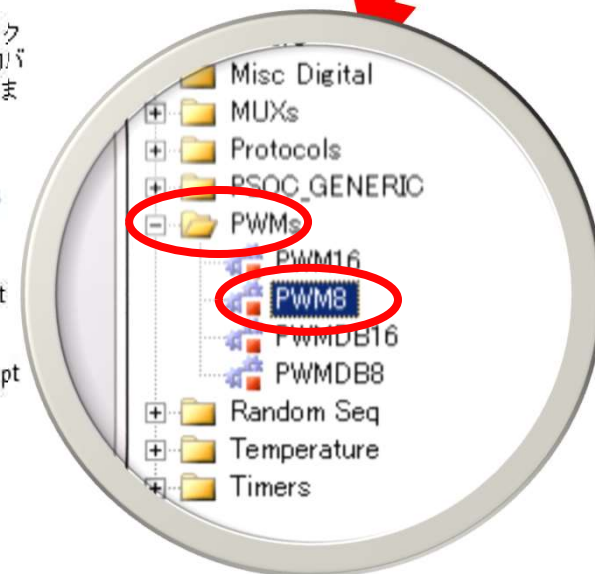
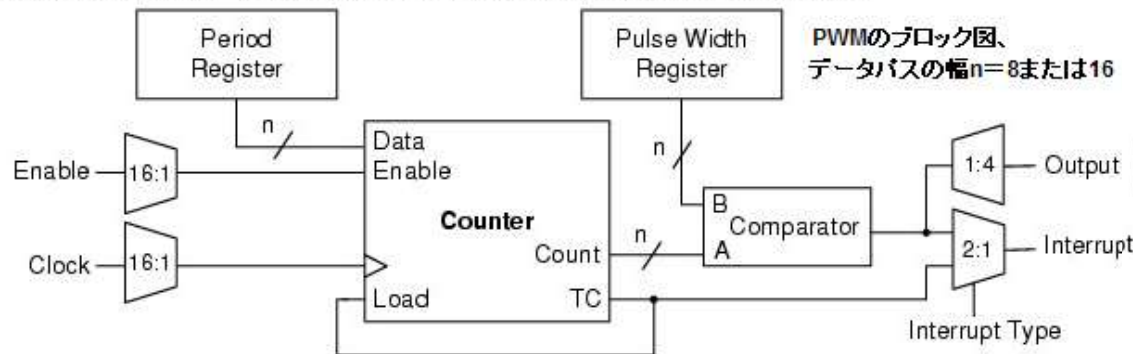
1.PWMs をクリック

2.PWM8 をダブルクリック

左上に自動的に配置されます

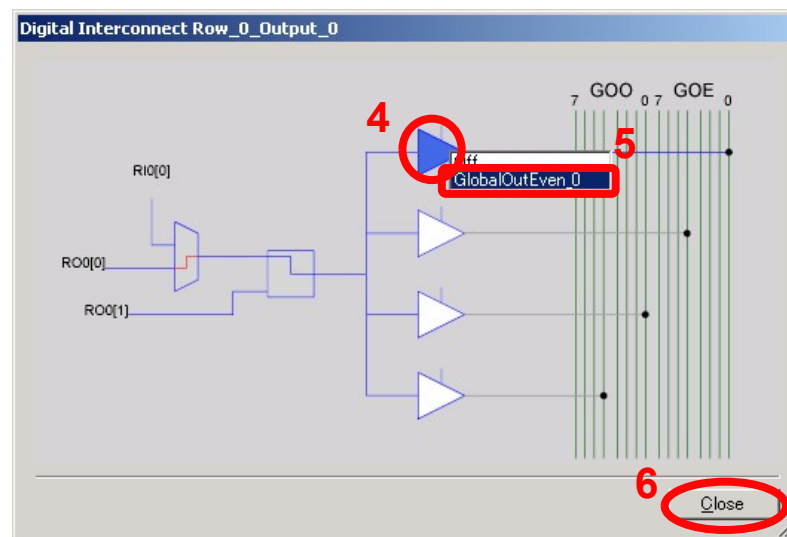
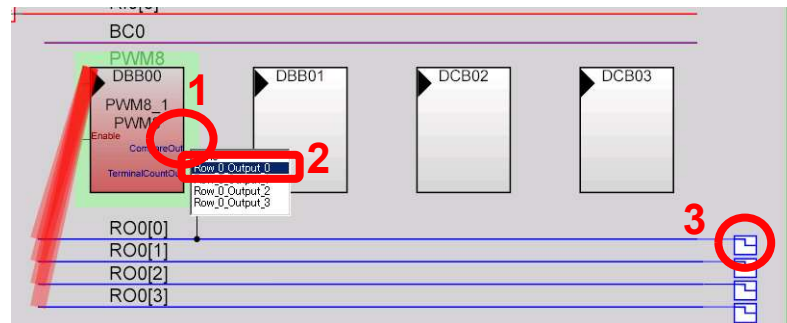


8ビットおよび16ビットのPWMユーザ モジュールは、周期とパルス幅をプログラム可能なパルス幅変調器です。クロックおよびイネーブル信号は、各種信号源から選択できます。出力信号はピンに配線するか、グローバル出力バスのいずれか1つに配線して、チップ内部の他のユーザ モジュールで使用できます。出力の立ち上がりエッジ、またはカウンタがタイマー カウント条件に達した時点での割り込みをプログラム可能です。



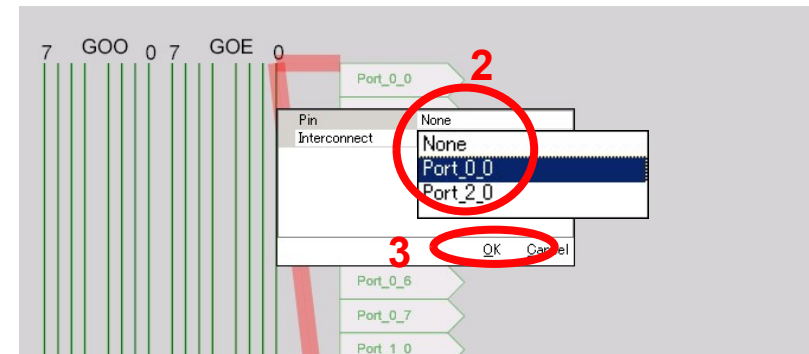
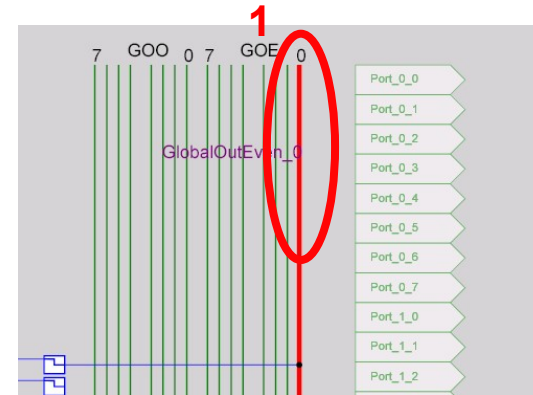
PWM8出力の配線 1

1. PWM8 CompareOut をクリック
2. Row_0_Output_0 を選択
3. RO0[0]の右端のブロックをクリック
4. 一番上のバッファをクリック
5. GlobalOutEven_0 を選択
6. Close をクリック



PWM8出力の配線 2

1. GOE 0 をクリック
2. 設定ボックスのNoneをクリック
3. ▼をクリックして、Pin を Port_0_0 に設定
4. OK をクリック



グローバルパラメータの設定

View > Global Resources

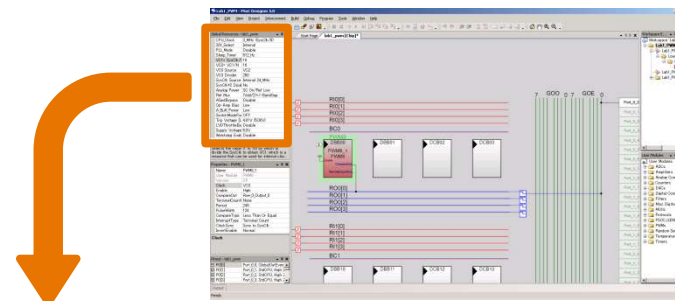
1.VC1 を 16

2.VC2 を 16

3.VC3 Source を VC2

4.VC3 Divider を 250

5.それ以外は 初期値

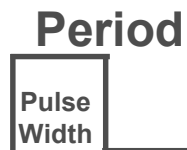
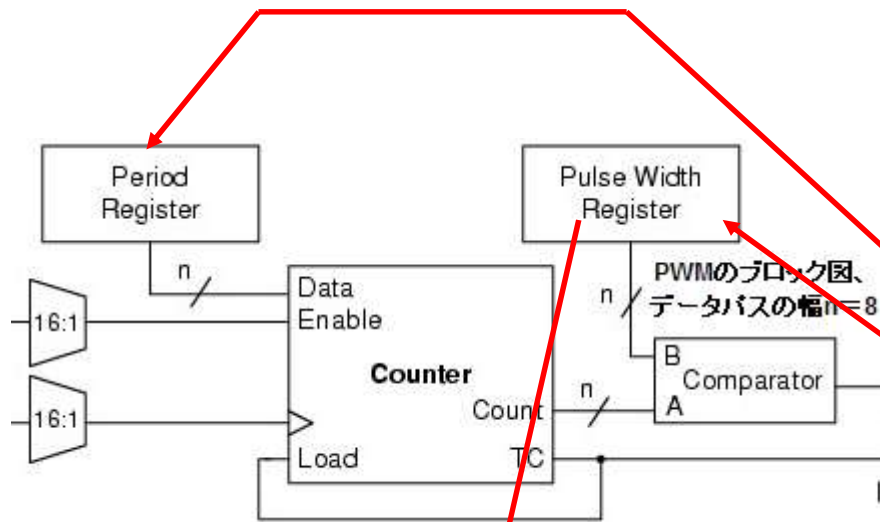
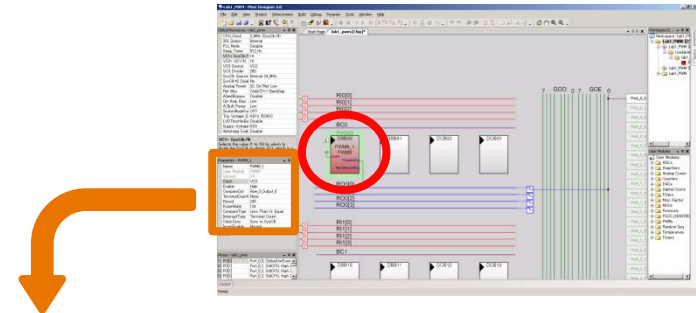


Global Resources - lab1_pwm	
CPU_Clock	3_MHz (SysClk/8)
32K_Select	Internal
PLL_Mode	Disable
Sleep_Timer	512_Hz
VC1= SysClk/N	16
VC2= VC1/N	16
VC3 Source	VC2
VC3 Divider	250
SysClk Source	Internal 24_MHz
SysClk*2 Disal	No
Analog Power	SC On/Ref Low
Ref Mux	(Vdd/2)+/-BandGap
AGndBypass	Disable
Op-Amp Bias	Low
A_Buff_Power	Low
SwitchModePur	OFF
Trip Voltage [L	4.81V (5.00V)
LVDThrottleBa	Disable
Supply Voltage	5.0V
Watchdog Ena	Disable

PWM8パラメータの設定

1. デジタルブロック上のPWM8_1をクリック

2. パラメータを入力



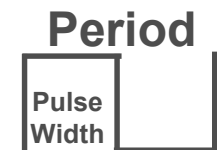
Clockの異択とPeriodで周波数、**PulseWidth**でDutyを設定

Properties - PWM8_1	
Name	PWM8_1
User Module	PWM8
Version	2.5
Clock	VC3
Enable	High
CompareOut	Row_0_Output_0
TerminalCount	None
Period	249
PulseWidth	124
CompareType	Less Than Or Equal
InterruptType	Terminal Count
ClockSync	Sync to SysClk
InvertEnable	Normal

各パラメータとクロックについて

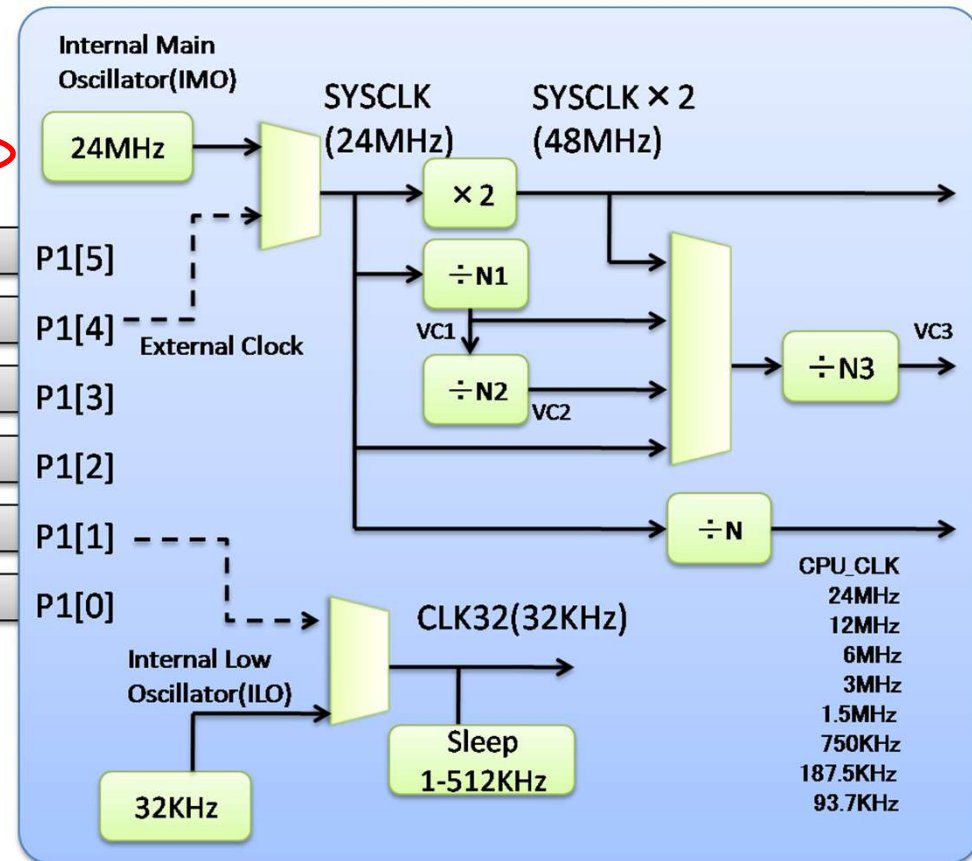
クロック	周波数	式
SysClk	24MHz	
VC1	1.5MHz	SysClk / 16
VC2	93.75KHz	VC1 / 16
VC3	375Hz	VC2 / 250
PWM8出力	1.5Hz	VC3 / 250

グローバルパラメータの クロック分周メカニズム



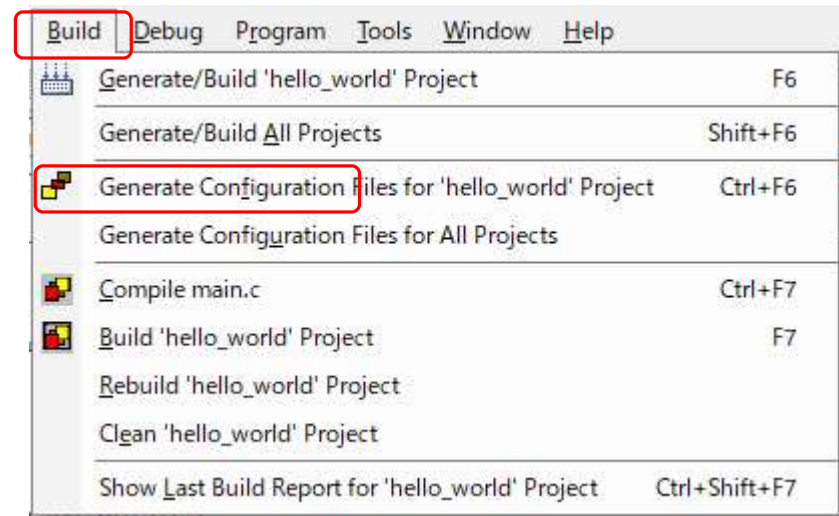
Clockの選択とPeriodで周波数、
PulseWidthでDutyを設定

外部水晶



GC(Generate Configuration)の実行

- Build >  Generate Configuration Files... をクリック



GCにより、設定ファイル、ユーザーモジュールAPIが生成される。
配線、設定の変更を加えたならGCをする必要がある。

C ソースコードの記述

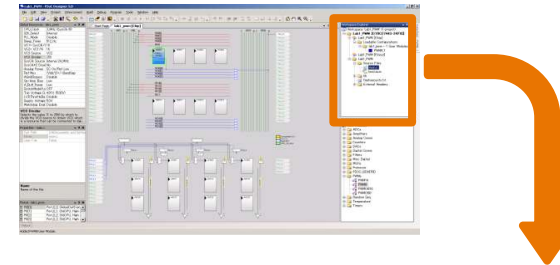
View > Chip Editor

1.Workspace Explorer 内の lab1_pwm をクリック

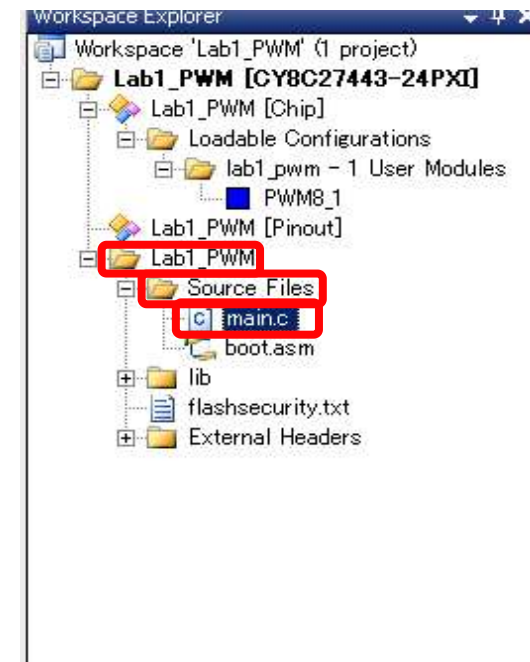
2.Source Files をクリック

3.main.c をダブルクリック

main関数内にPWM8_1_Start();
を追加

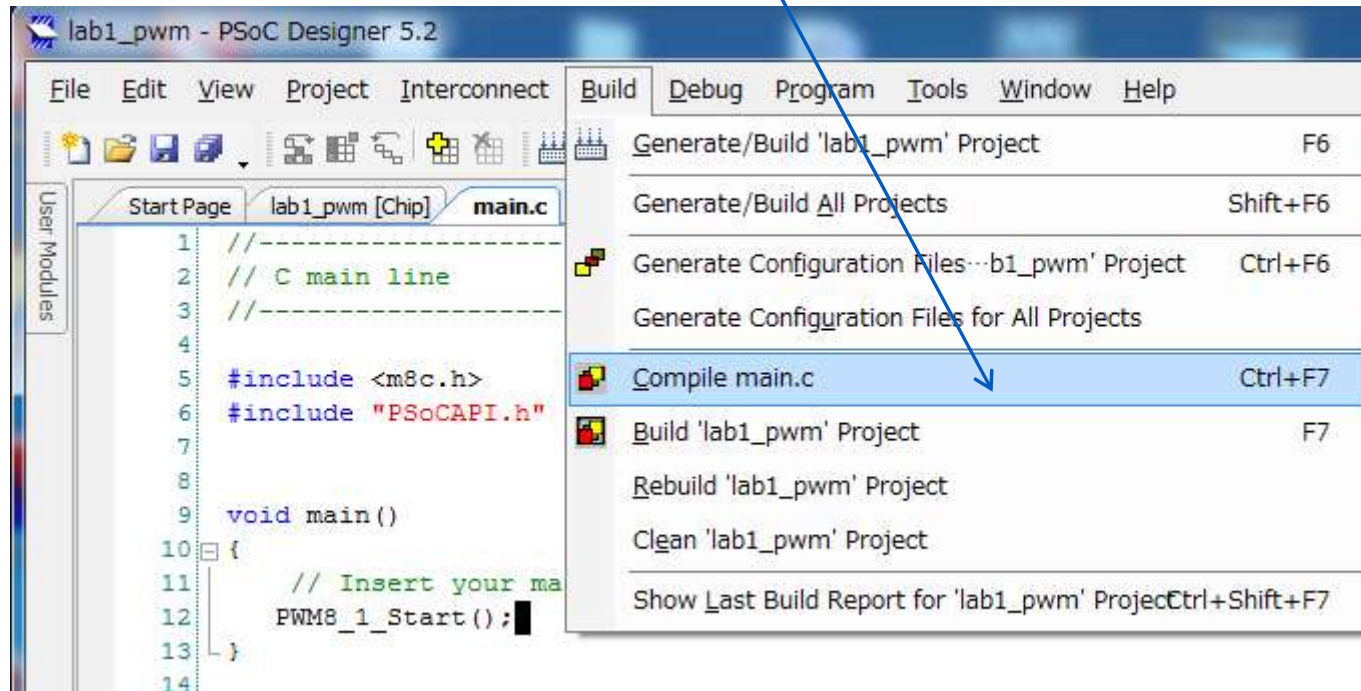


```
Start Page lab1_pwm [Chip] main.c
1 //-----
2 // C main line
3 //-----
4
5 #include <m8c.h> // part specific cons
6 #include "PSoCAPI.h" // PSoC API definitio
7
8
9 void main()
10 {
11 // Insert your main routine code here
12 PWM8_1_Start();
13 }
```





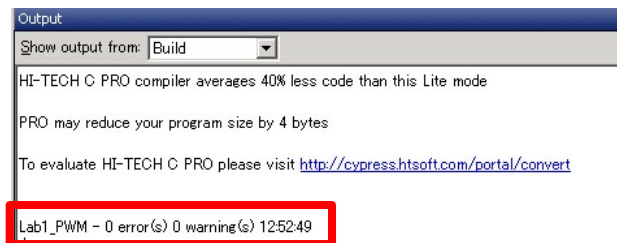
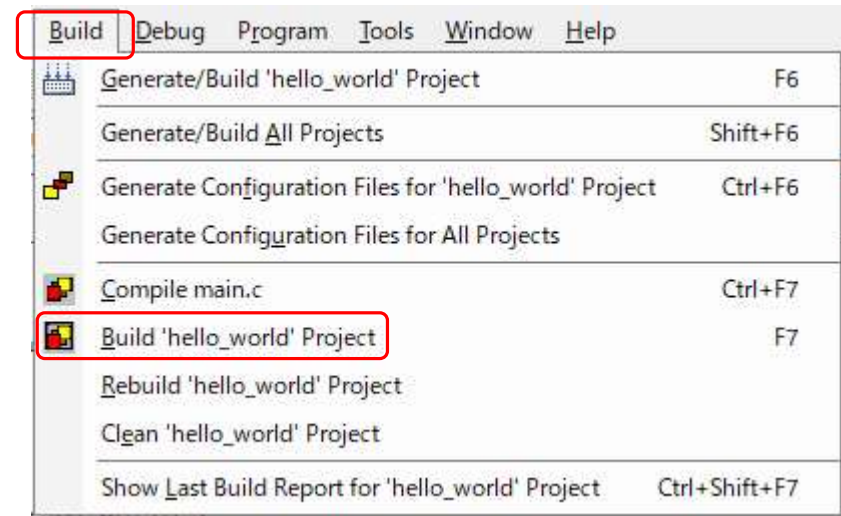
Build > Compile の実行



コンパイルエラーが出たら、Output Window のエラー行(!E!/W)をダブルクリックすると、Cソースコードのエラー行にジャンプします。(修正する場所は、他の行の場合もあります。)

ビルドの実行

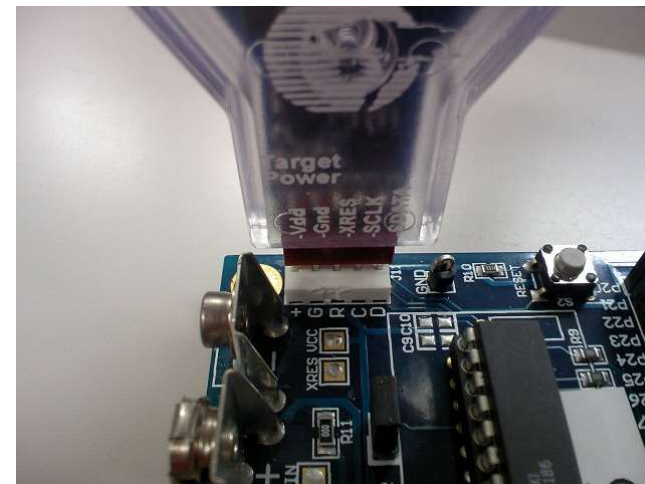
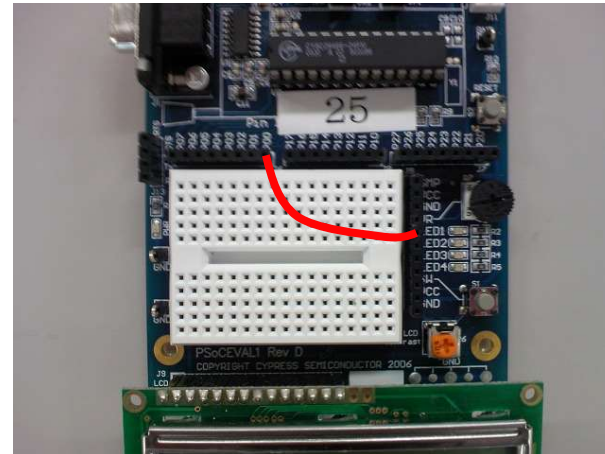
- Build > Build 'Lab1_PWM' Project をクリック



0 error(s) と出れば成功

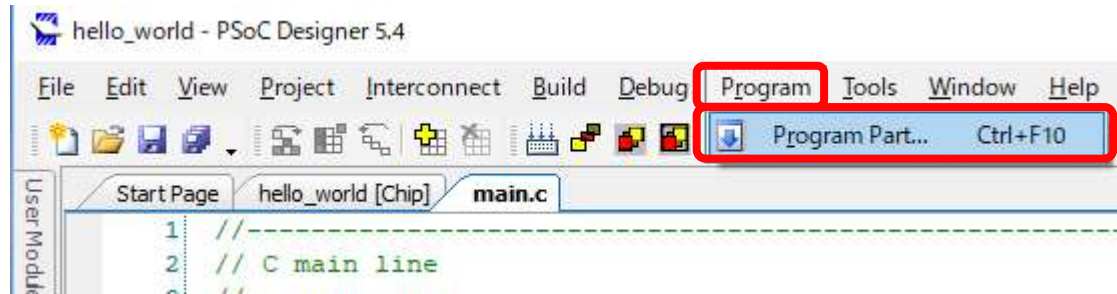
MiniProgの接続,回路配線

- P00 と LED1 をジャンプワイヤーで接続
- MiniProg を Eval1 に接続
Vdd と + が一致するように

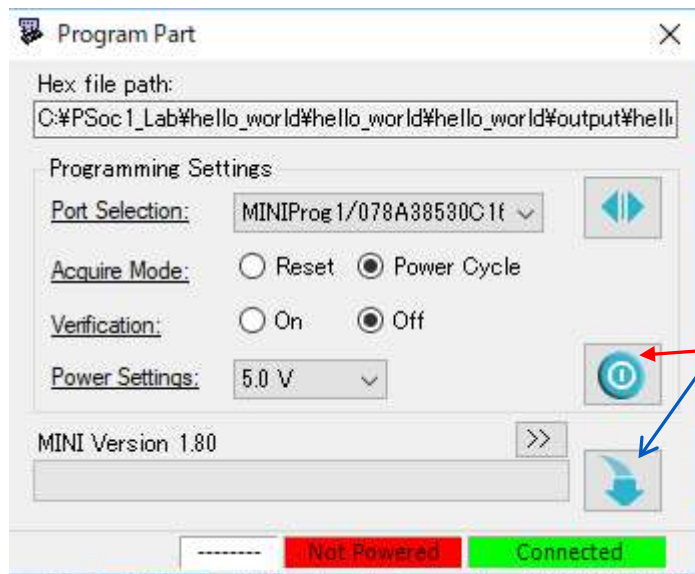




書き込み

- Program > Program Part をクリック



PSoC Designer から Program > Program Part をクリックすると、PSoC Programmer が自動的に起動し 作成されたhex ファイルがロードされる。



-  をクリックすると書き込み開始
- Actions を読んで状況を確認
-  をクリックすると MINIProgを通じて電源を供給
- LEDの点滅とその間隔を確認.

Programmer が2つ以上起動しているとエラーが発生しますので、その場合は、すべての Programmerを終了して再度PSoC DesignerからProgram タブで起動してください。

main.c の変更 (オプション課題)

約0.67秒で点滅するLED動作が確認できたら、
main.c を修正して点滅する周期を変更してみよう

Period を10Hz程度にしてPulse Width をDuty 50%にして
LEDの点灯周期を変更してみよう

500Hz程度のPWM波形を作り、この音を聴いてみよう
音を聴くには、スピーカーやイヤホンをつなぎます

自由課題

パルスモーター制御

プログラムからレジスタの値を直接書き換えることでPWM波形を自由に変更することができる. RCカーなどに応用できる.

PWMとPDM

外部を制御する場合にPDM (Pulse Density Modulation)を使用するとPWMと同じ積分値でもランダムに信号波形が変化するのでノイズスペクトラムの拡散効果がある.PSoCのPRS ユーザーモジュールにはこの機能を実装できる

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