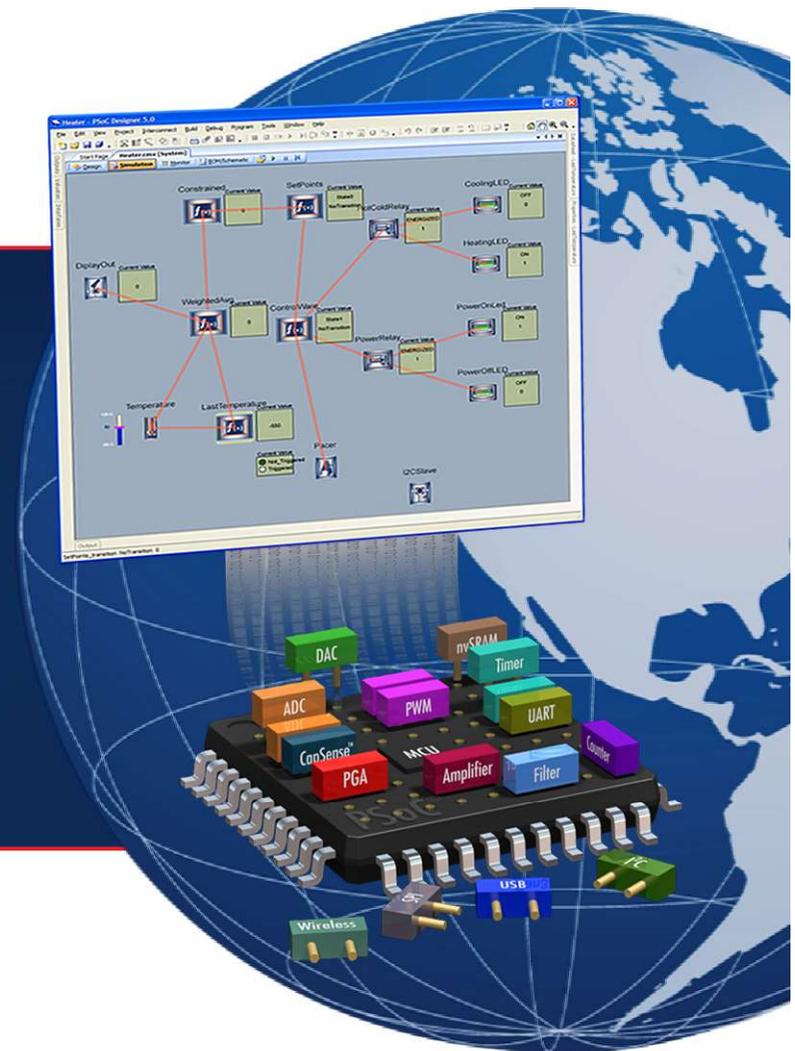


AD変換の演習

lab3_adc

*PSoC Experiment
Lab*

Experiment Course Material V 3.10
November 10th, 2020
lab3_adc.pptx (33Slides)
Renji Mikami





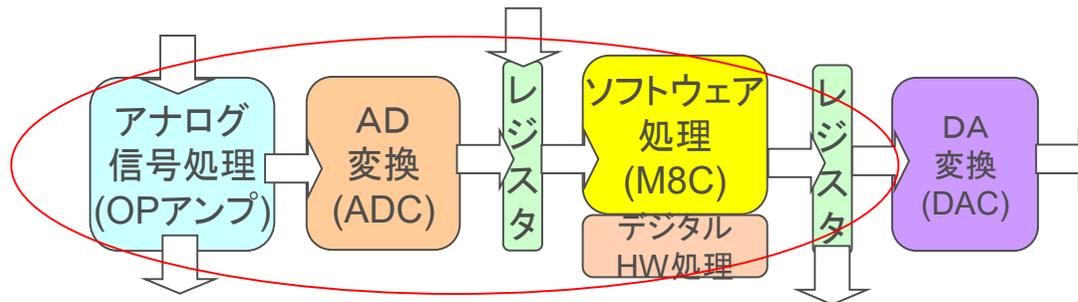
入力アナログ電圧をPGAで増幅しこれをAD変換して値をLCD表示 - デジタル電圧計を作ります.

ラボ

lab3_adc

AD変換の方法がポイントになります.

デジタル化されればパルス幅に変換できます.





AD変換によって何ができるか

これまでは、出力側のアプリケーション演習を行ってきました。出力には、アナログ出力、デジタル出力がありました。表現の形態としては、LCDディスプレイへの表示、サウンド出力、LEDの点滅などがありました。

しかしこれらは、画一的な機能の表現です。連続的な入力の変化によって出力を変化させることはありません。入力によって出力を連続的に制御したり変化させるためには、何らかの形で入力情報を連続的に取り込む必要があります。

AD変換では、アナログ的な電圧の変化をデジタル量に変換できますから、外部の変化量を電圧変化への置きかえれば、アナログ的な連続的な変化量を扱うことができます。

アナログ的な入力素子としては、さまざまなセンサーがありますが、この多くは電圧出力になっています。

よって、センサー出力 → 増幅 → AD変換 → デジタル数値化 が可能になりますから、これによりMPUで自由にデータの処理や外部制御や出力が可能になります

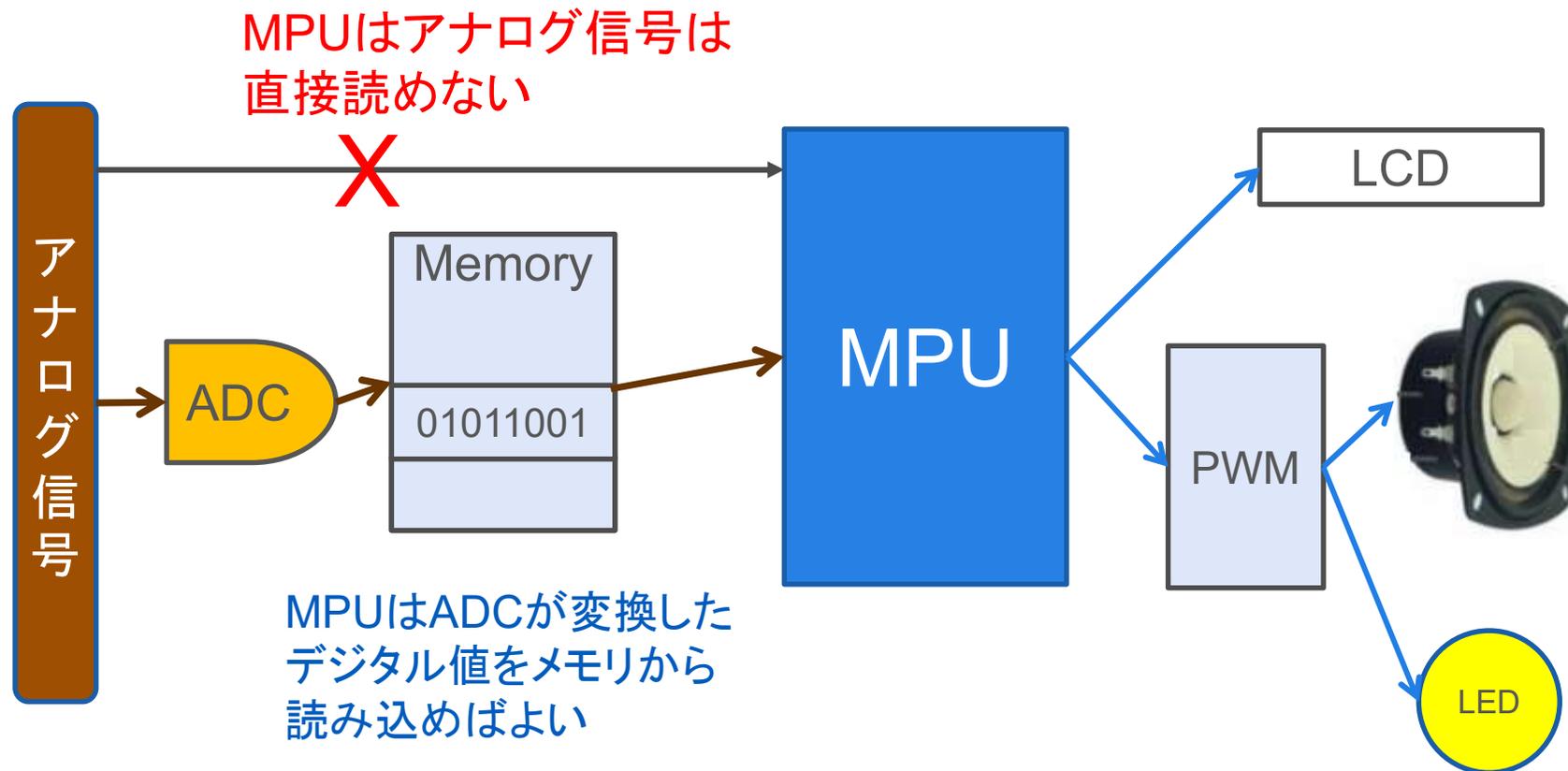
例えば、温度によって音が変わるシンセサイザ、3軸の加速度(ジャイロ)センサをコントローラにした楽器なども可能です。

MPUとアナログ入力

CB22-1R2

入力側

出力側



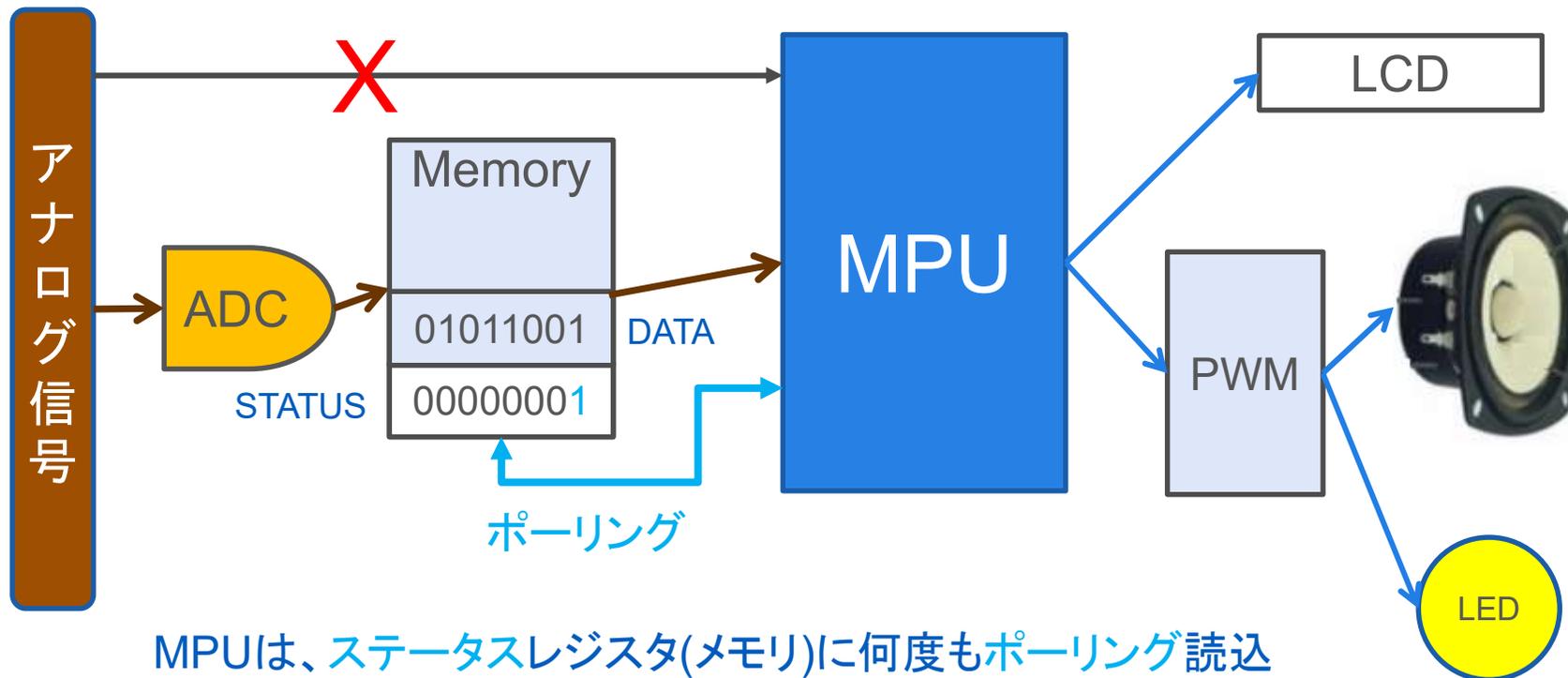
ポーリングで変換終了後のデータを 読み込む

入力側

MPUは、数10MHzで高速で動作する
しかしAD変換には時間がかかる。
(MPUに比べてとても遅い~ 1/1000)

出力側

MPUはいつAD変換が終わったかわからない

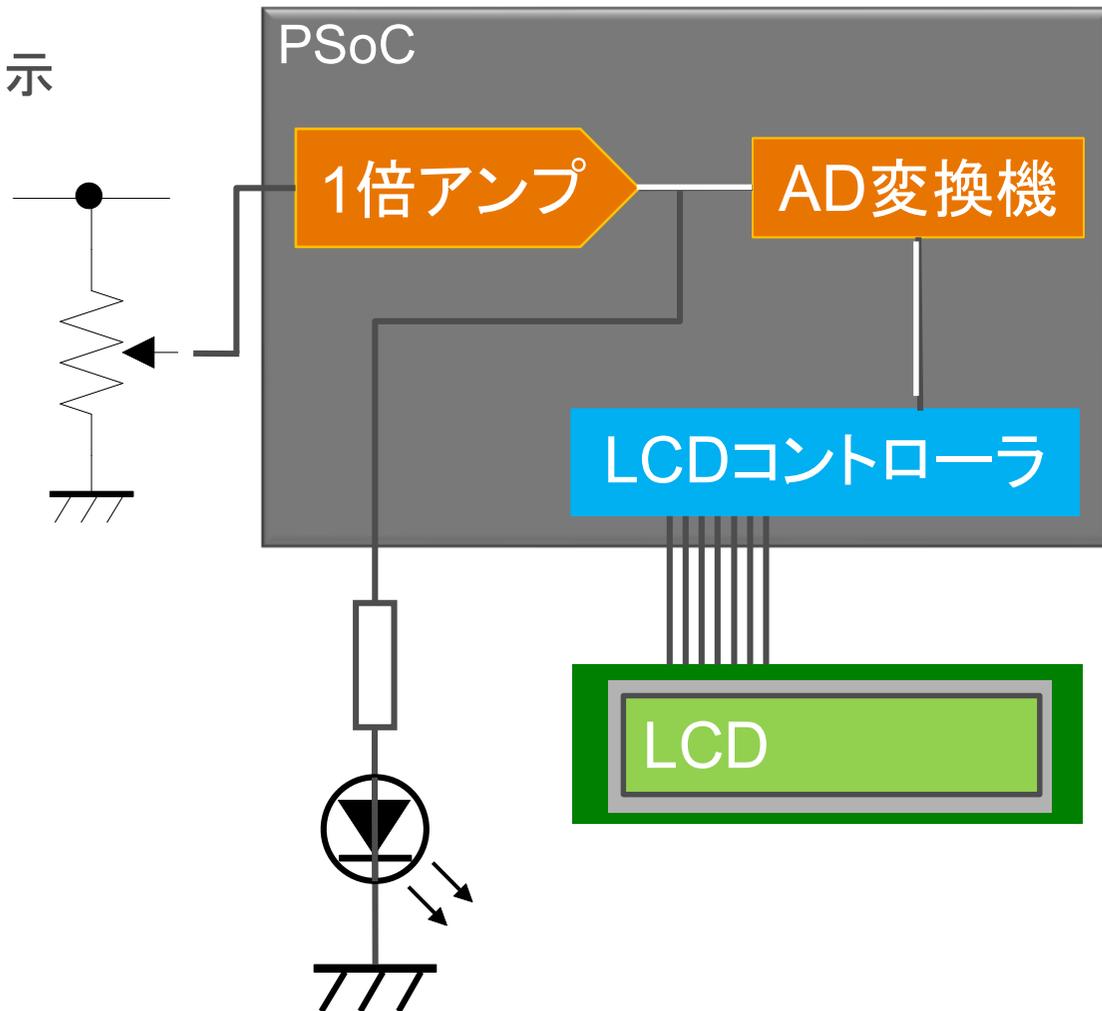


MPUは、ステータスレジスタ(メモリ)に何度もポーリング読み込みをかける。変換終了がわかった時点で、データを読みに行く



lab3_adc

- VRの電圧をLCDで表示
- VRの電圧をそのままLEDへ





3つのユーザーモジュールでの設計

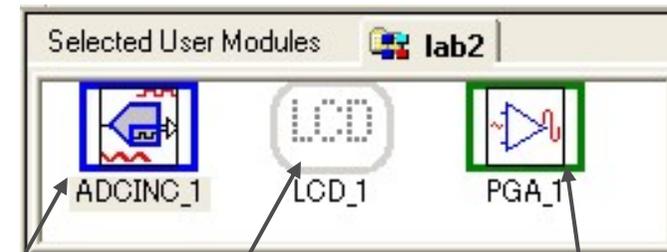
ユーザーモジュールは、パラメタライズされた機能ライブラリ(IPのようなもの)

選択したユーザーモジュールは自動的に内部リソースのコンビネーションで実現される

内部リソースは、コース・グレインで作られているので集積度が高く効率が高い

極少の配線数で機能を実現できるので、シリコンの使用効率が低い

入力アナログ電圧の
デジタル表示回路例



機能可変ADコンバータ

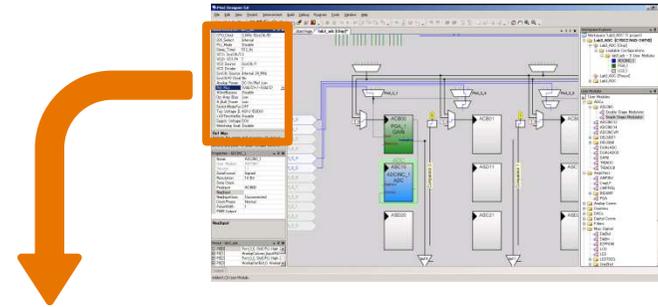
LCDドライバ

可変利得アンプ



AD変換時のRef Muxの設定

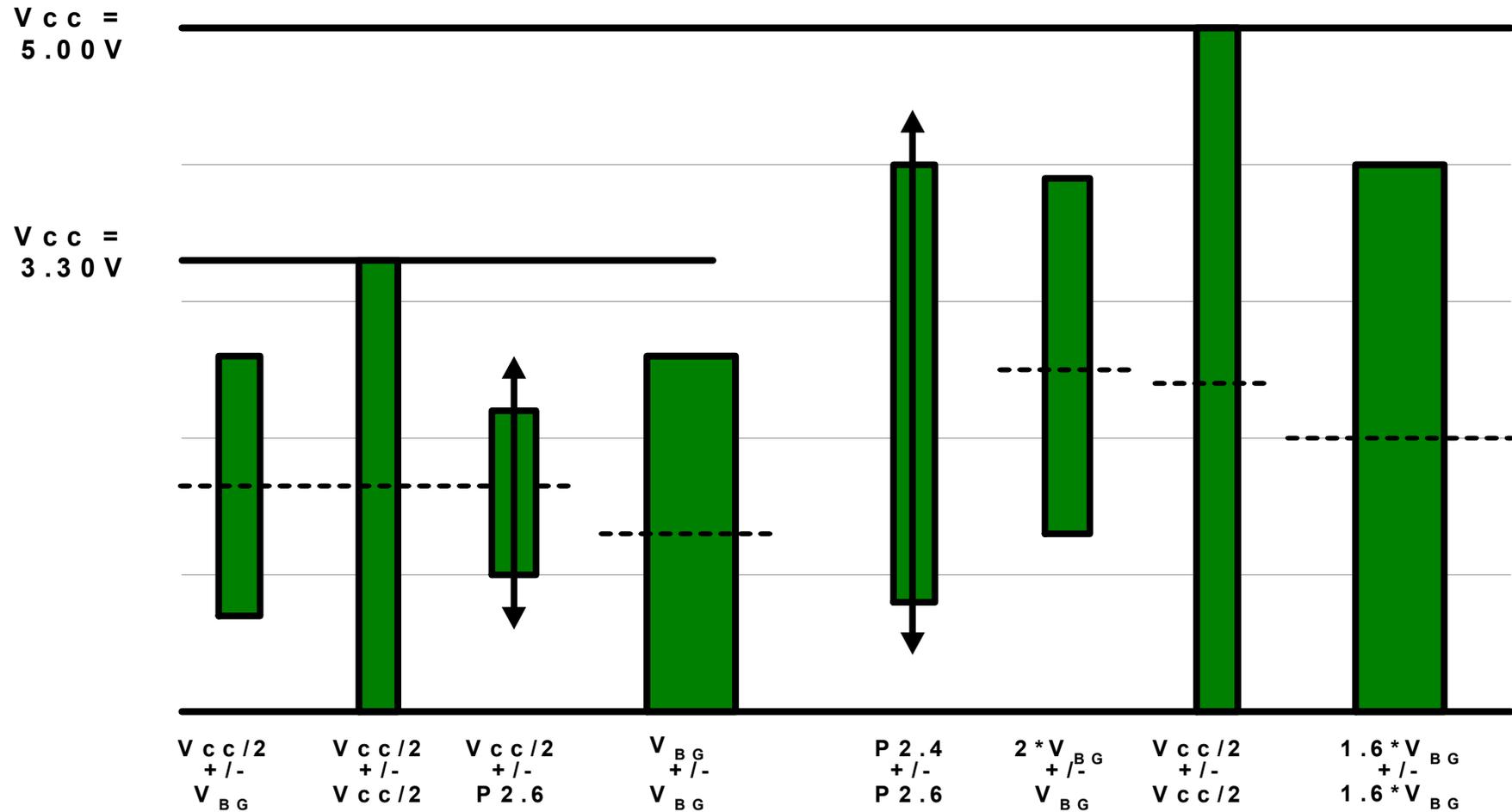
Ref Mux はアナロググラウンドのレベルを決定し
アナログ信号の上下の振幅範囲を決定します
これはPSoCのオペアンプが単電源のため
0Vをグラウンドレベルとしてマイナス側の信号を
扱えないため基準電位をかさ上げします。
このようにしてかさ上げ設定したレベルを
アナロググラウンド電位と呼んでいます。
lab_motor のPGAのRefの選択枝に
AGNDがりましたが、ここで設定します



Global Resources - lab3_adc	
CPU_Clock	3_MHz (SysClk/8)
32K_Select	Internal
PLL_Mode	Disable
Sleep_Timer	512_Hz
VC1=	SysClk/13
VC2=	VC1/N 1
VC3 Source	SysClk/1
VC3 Divider	1
SysClk Source	Internal 24_MHz
SysClk*2 Disal	No
Analog Power	SC On/Ref Low
Ref Mux	(Vdd/2)+/-(Vdd/2)
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 Enab	Disable



RefMuxの設定と範囲について





RefMuxについて

RefMuxはADの入カレンジを決定

表示	5V駆動	3.3V駆動
$[V_{dd}/2] \pm \text{BandGap}$	2.5 V \pm 1.3 V	1.65V \pm 1.3V
$[V_{dd}/2] \pm [V_{dd}/2]$	2.5 V \pm 2.5 V	1.65V \pm 1.65V
BandGap \pm BandGap	1.3 V \pm 1.3 V	1.3V \pm 1.3V
1.6 BandGap \pm 1.6 BandGap	2.08V \pm 2.08V	使用不可
2 BandGap \pm BandGap	2.6 V \pm 1.3 V	使用不可
2 BandGap \pm P2[6]	2.6 V \pm P2[6]V	2.6 V \pm P2[6]V
P2[4] \pm BandGap	P2[4] V \pm 1.3V	P2[4] V \pm 1.3V
P2[4] \pm P2[6]	P2[4]V \pm P2[6]V	P2[4]V \pm P2[6]V

BandGap電圧は内部で1.2xx..Vから昇圧した1.3Vとなります。実はこの電圧もレジスタ値でトリミングできます。

Reference 電圧を外部から入力することができますが、これができるピンはPort2[4]です。

3210EVAL1ではPort2はLCDに接続されています。



AD変換ステータスをポーリングでチェック

- ADCINC_fIsDataAvailable 値は、AD変換が終了してデータが読み込み可能な状態になったときに0になるレジスタ

このプログラムでは、ポーリングで値を読みに行き、0のとき(データが読み込み可能のとき)adc_dataにAD変換値を代入する

ADCINCユーザーモジュールは、割り込みが用意されていないのでポーリングでステータスを読む

AD変換のユーザーモジュールには12種類あり、8ビット以下のものは、割り込みが使える

```
Start Page lab3_adc [Chip] main.c
1 //-----
2 // C main line
3 //-----
4
5 #include <m8c.h> // part specific constants and
6 #include "PSoC_API.h" // PSoC API definitions for all
7
8 |
9 void main()
10 {
11     unsigned int adc_data;
12     PGA_Start(PGA_HIGHPOWER);
13     LCD_Start();
14     LCD_InitBG(LCD_SOLID_BG);
15     M8C_EnableGInt;
16     ADCINC_Start(ADCINC_HIGHPOWER);
17     ADCINC_GetSamples(0);
18     while(1){
19         while(ADCINC_fIsDataAvailable() == 0);
20         adc_data = ADCINC_wClearFlagGetData();
21         LCD_Position(0,0);
22         LCD_PrHexInt(adc_data);
23         LCD_DrawBG(1,0,16,(adc_data/50));
24     }
25 }
26
```

大文字のi



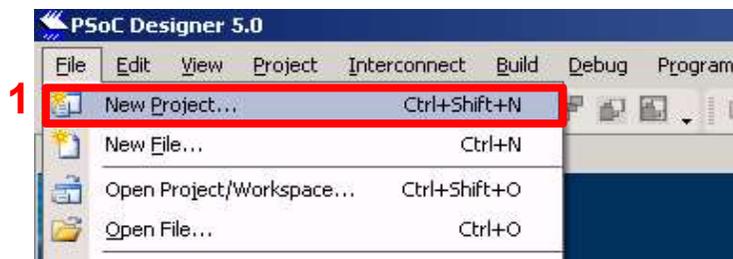
ラボ lab3_adc 手順

- 1.PGA, ADCINC, LCD ユーザーモジュールを配置
- 2.モジュール間を結線、パラメータを設定
- 3.GCとBuild
- 4.ジャンパ線で配線します
- 5.プログラムしてVRを回して値を読み取ります

解説：

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

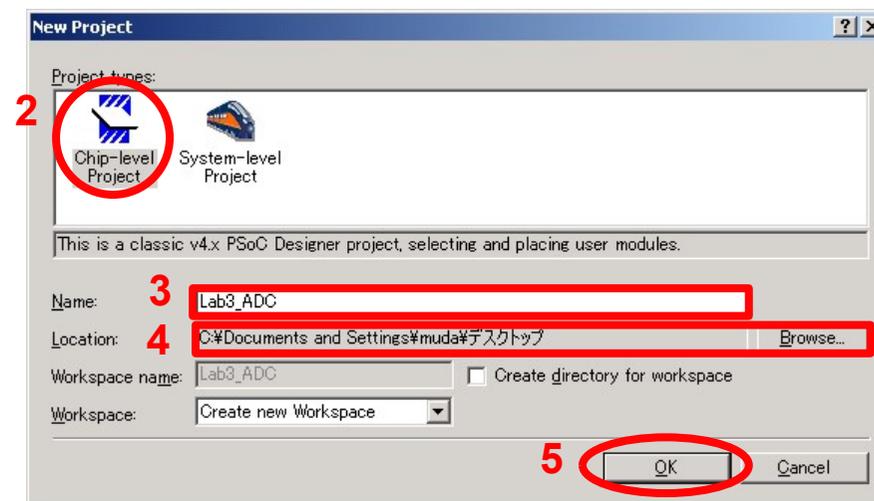
3. Name を入力

例: lab3_adc

4. Location を選択

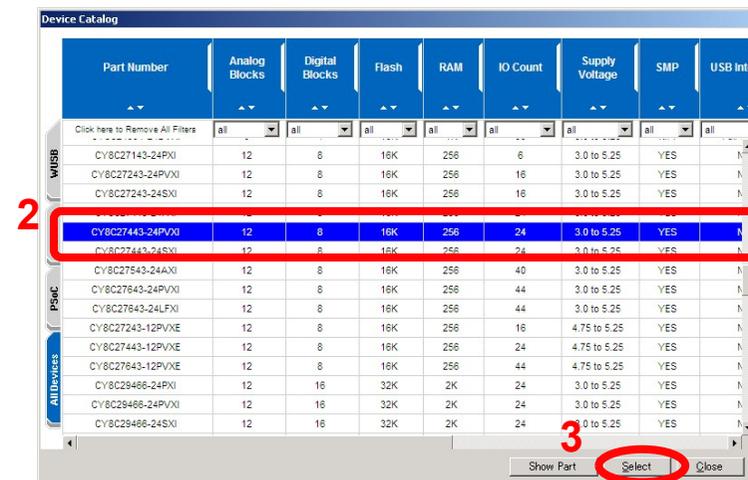
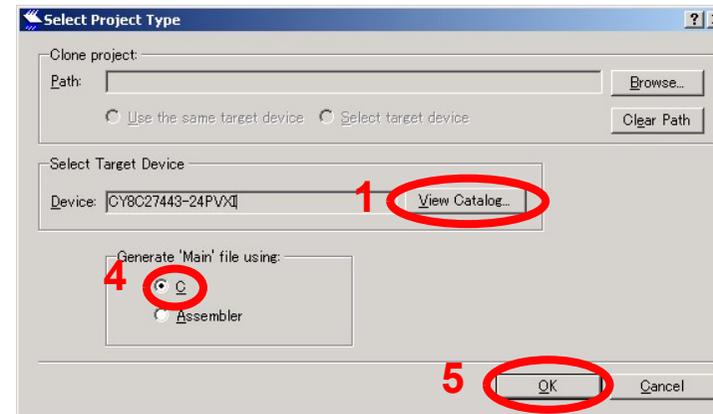
例: C:\psoc_lab\lab3_adc

5. OK をクリック



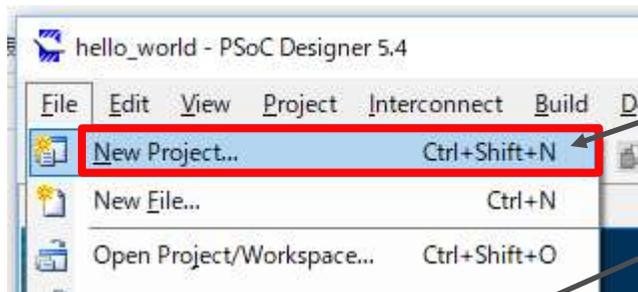
使用するPSoC、言語の選択(旧版ソフトウェアの場合)

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





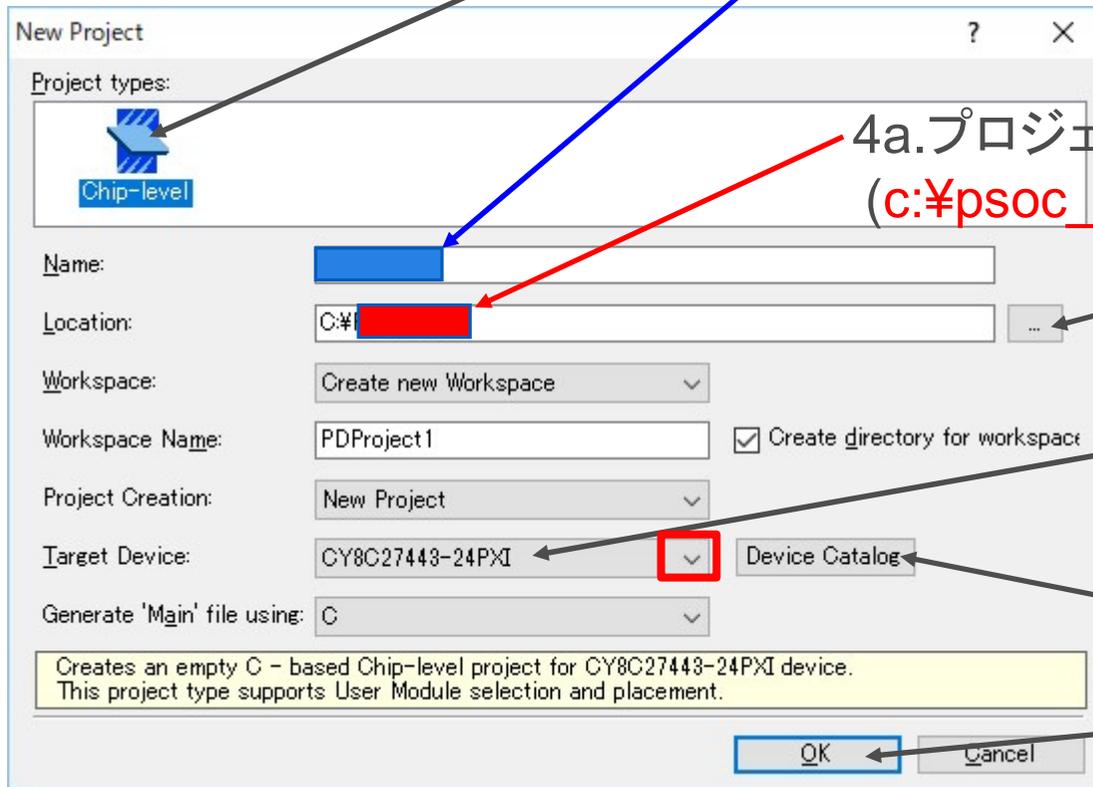
New Project: lab3_adc の作成 (新版ソフトウェアの場合)



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(lab3_adc)を入力



4a. プロジェクトを保存するディレクトリ
(c:\psoc_lab\lab3_adc)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

ユーザーモジュールの追加,配置

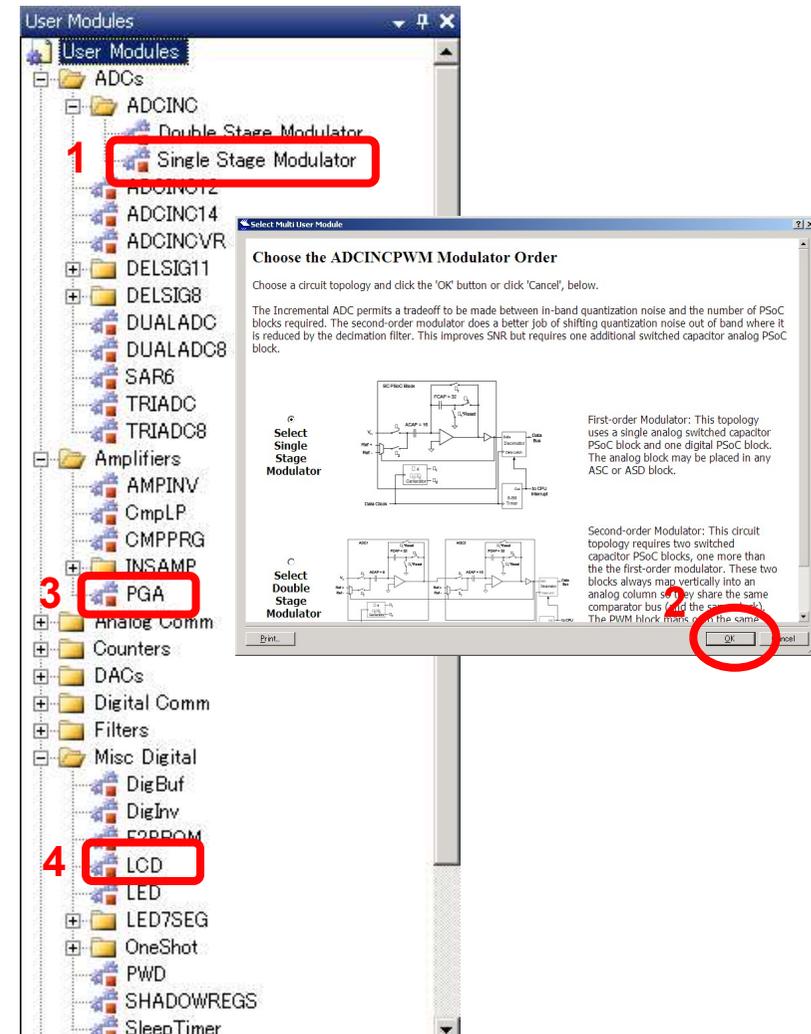
View > User Module Catalogをクリックして以下の3つのモジュールを追加

1.ADCs > ADCINC >
Single Stage Modulator
をダブルクリック

2.ポップアップウィンドはOKで閉じる

3.Amplifiers > PGA

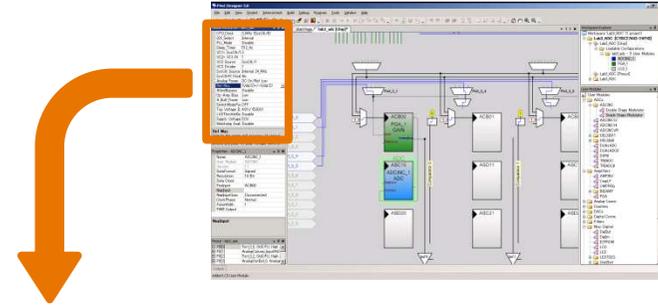
4.Misc Digital > LCD



グローバルパラメータの設定

View > Global Resource

- 1.VC1 を 3
- 2.Ref Mux を $(Vdd/2)+/-(Vdd/2)$
- 3.それ以外は 初期値



**Ref Mux はアナロググラウンドのレベルを決定し
アナログ信号の上下の振幅範囲を決定します
これはPSoCのオペアンプが単電源のため
0Vをグラウンドレベルとしてマイナス側の信号を
扱えないため基準電位をかさ上げします。
このようにしてかさ上げ設定したレベルを
アナロググラウンド電位と呼んでいます。
lab_motor のPGAのRefの選択枝に
AGNDがりましたが、ここで設定します**

Global Resources - lab3_adc	
CPU_Clock	3_MHz (SysClk/8)
32K_Select	Internal
PLL_Mode	Disable
Sleep_Timer	512_Hz
VC1=	SysClk/13
VC2= VC1/N	1
VC3 Source	SysClk/1
VC3 Divider	1
SysClk Source	Internal 24_MHz
SysClk*2 Disal	No
Analog Power	SC On/Ref Low
Ref Mux	$(Vdd/2)+/-(Vdd/2)$
AGndBypass	Disable
Op-Amp Bias	Low
A_Buff_Power	Low
SwitchModePui	OFF
Trip Voltage [L	4.81V (5.00V)
LVDThrottleBa	Disable
Supply Voltage	5.0V
Watchdog Ena1	Disable

RefMuxについて

RefMuxはADの入カレンジを決定

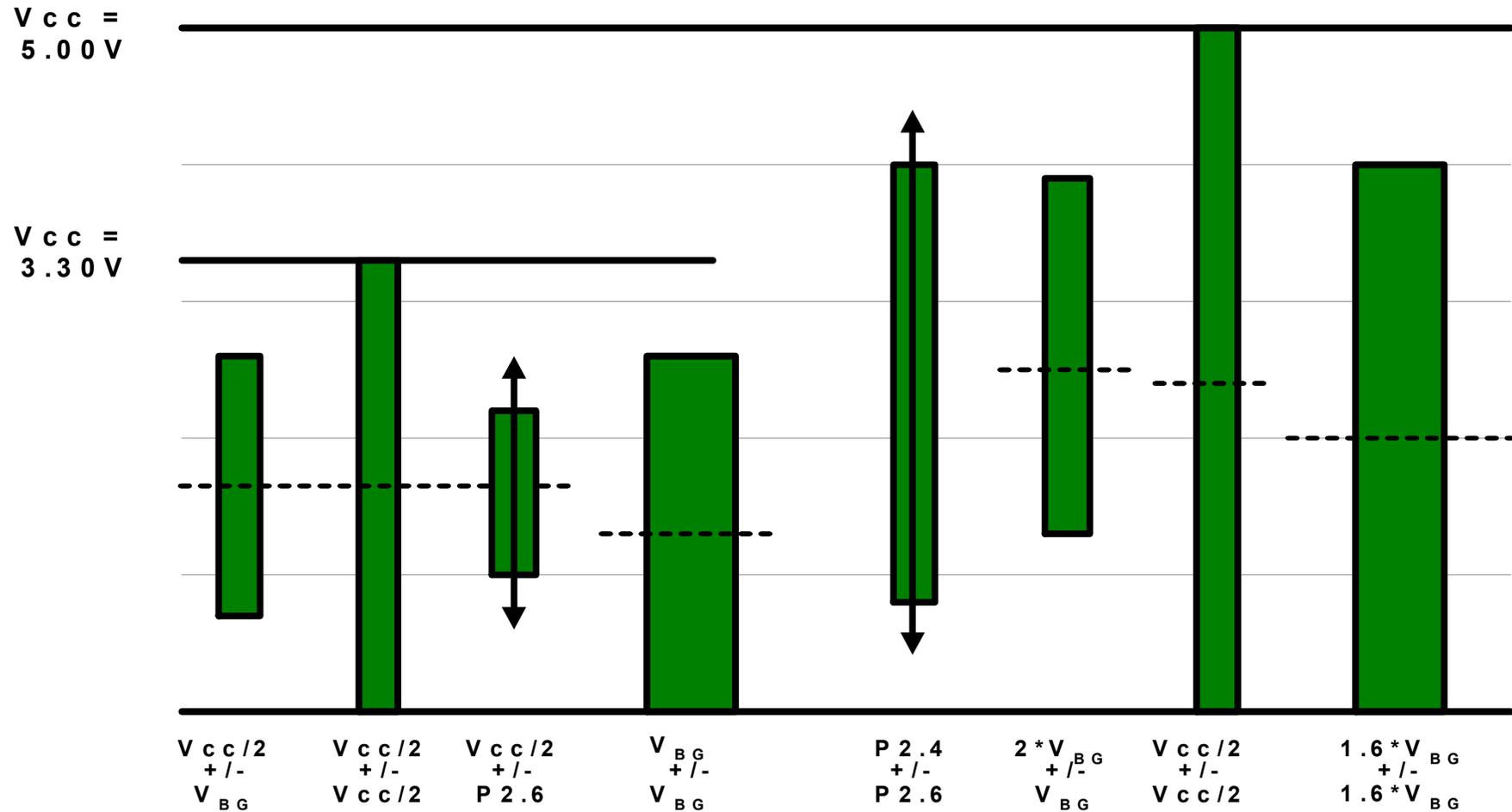
表示	5V駆動	3.3V駆動
$[V_{dd}/2] \pm \text{BandGap}$	2.5 V \pm 1.3 V	1.65V \pm 1.3V
$[V_{dd}/2] \pm [V_{dd}/2]$	2.5 V \pm 2.5 V	1.65V \pm 1.65V
BandGap \pm BandGap	1.3 V \pm 1.3 V	1.3V \pm 1.3V
1.6 BandGap \pm 1.6 BandGap	2.08V \pm 2.08V	使用不可
2 BandGap \pm BandGap	2.6 V \pm 1.3 V	使用不可
2 BandGap \pm P2[6]	2.6 V \pm P2[6]V	2.6 V \pm P2[6]V
P2[4] \pm BandGap	P2[4] V \pm 1.3V	P2[4] V \pm 1.3V
P2[4] \pm P2[6]	P2[4]V \pm P2[6]V	P2[4]V \pm P2[6]V

BandGap電圧は内部で1.2xx..Vから昇圧した1.3Vとなります。実はこの電圧もレジスタ値でトリミングできます。

Reference 電圧を外部から入力することができますが、これができるピンはPort2[4]です。

3210EVAL1ではPort2はLCDに接続されています。

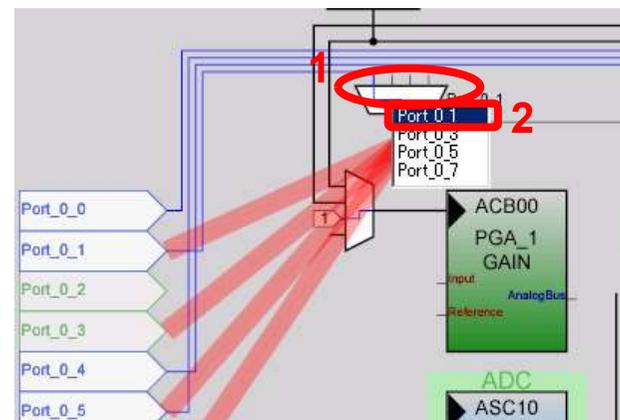
RefMuxについて



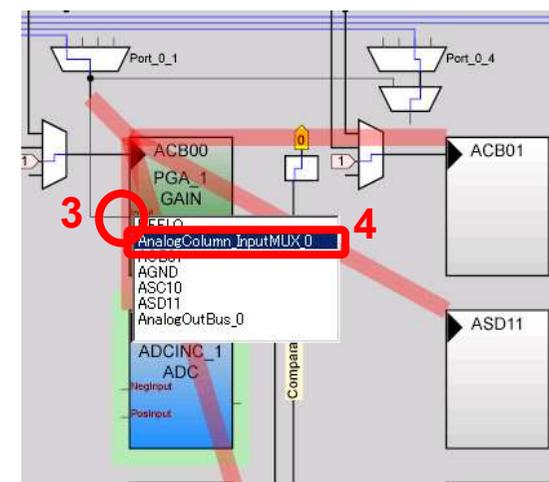
PGA入力の配線

View > Chip Editor

1. Analog_Column_InputMux_0 をクリック
2. Port0_1 を選択
3. PGAのInputを クリック
4. Analog_Column_inputMux_0を選択



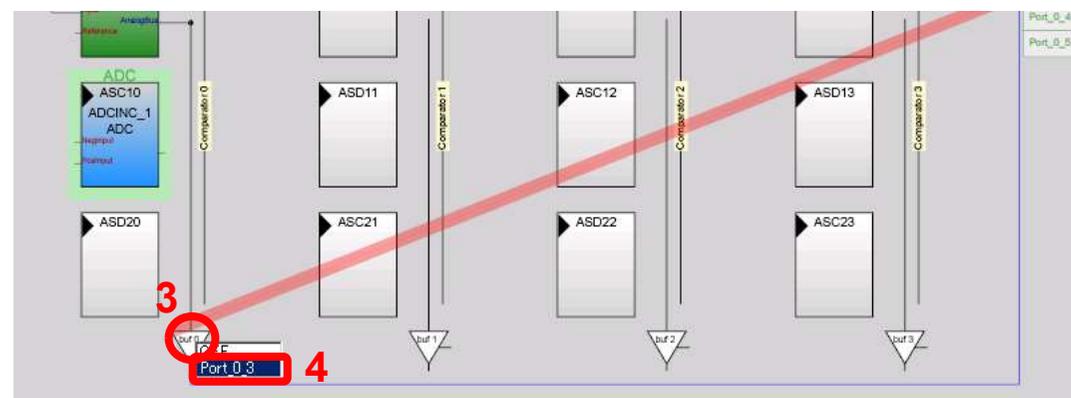
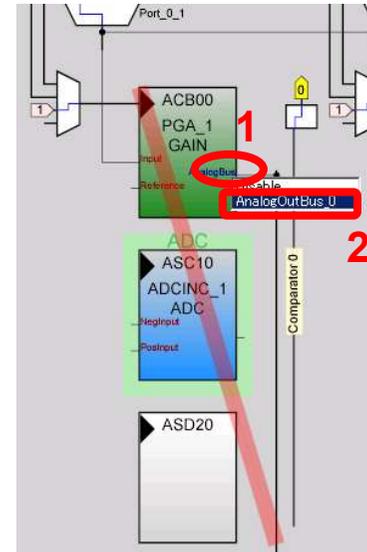
移動	Alt+ドラッグ
拡大	Ctrl+クリック Ctrl+ドラッグ
縮小	Ctrl+shift+クリック Ctrl+shift+ドラッグ



PGA出力の配線

View > Chip Editor

1. PGAのAnalogBus をクリック
2. AnalogOutBus_0 を選択
3. AnalogOutbuf_0 をクリック
4. Port_0_3 を選択



PGAパラメータの設定

View > Chip Editor

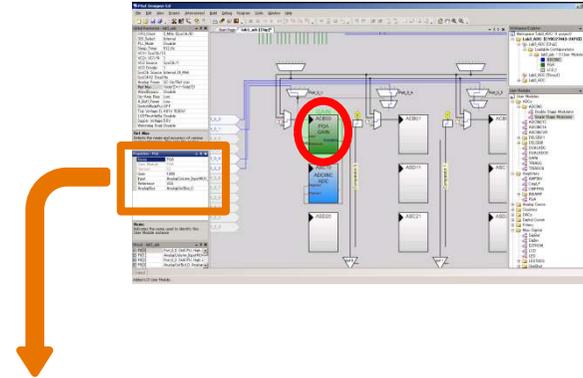
1. デジタルブロック上のPGA_1
をクリック

2. Name を PGA_1 → PGA

3. Gain を 1.000

4. Reference を VSS

それ以外は 初期値

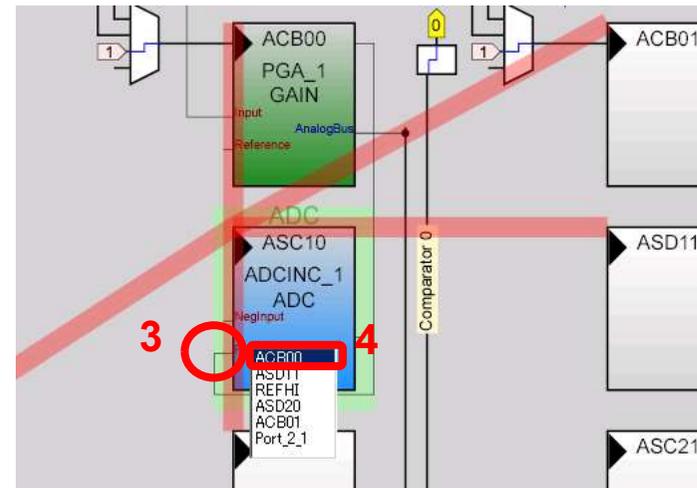
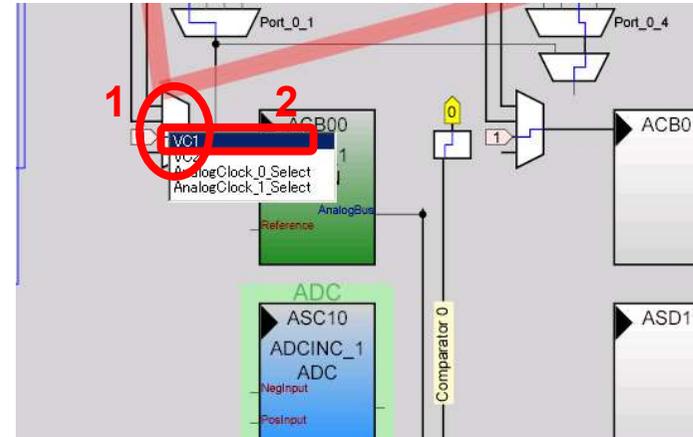


Name	PGA
User Module	PGA
Version	3.2
Gain	1.000
Input	AnalogColumn_InputMUX
Reference	VSS
Analog Bus	AnalogOutBus_0

ADCINCの配線

View > Chip Editor

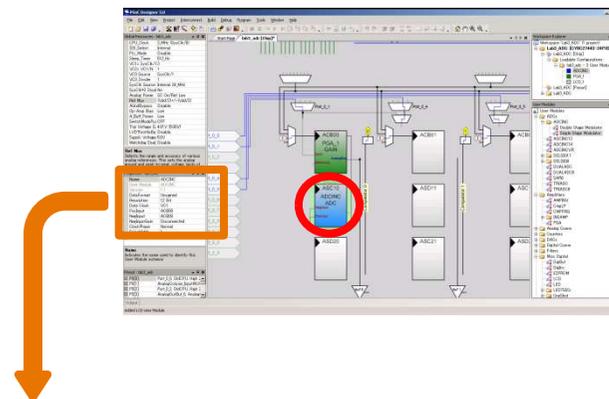
1. AnalogColumn_Clock_0 をクリック
2. VC1 を選択
3. ADCINC_1のPosInput をクリック
4. ACB00 を選択



ADCINCパラメータの設定

View > Chip Editor

1. デジタルブロック上のADCINC_1
をクリック



2. Name を ADCINC

3. DataFormat を Unsigned

4. Resolution を 12 Bit

5. Data Clock を VC1

6. NegInput を ACB00

7. PWM Output を None

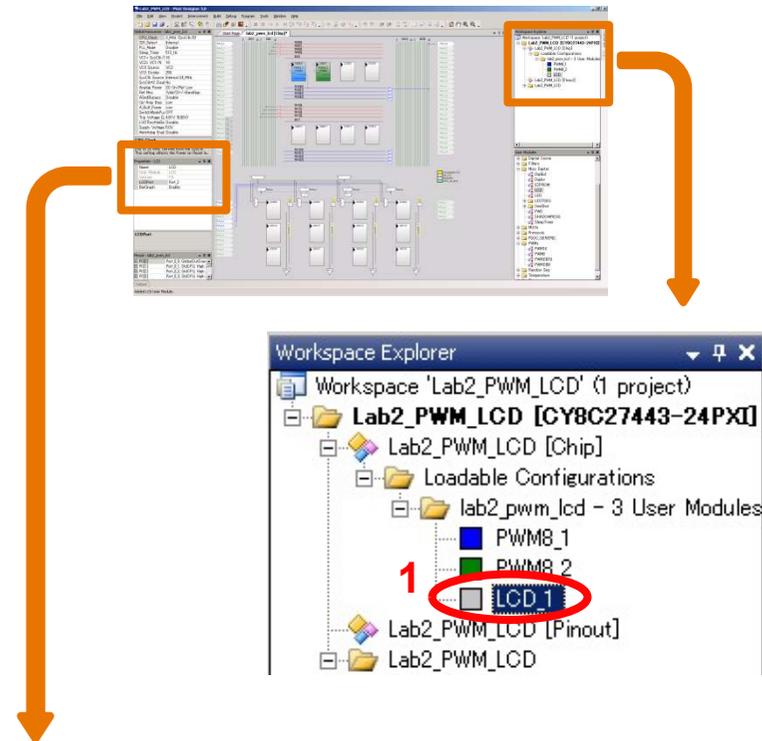
それ以外は 初期値

Properties - ADCINC	
Name	ADCINC
User Module	ADCINC
Version	1.1
DataFormat	Unsigned
Resolution	12 Bit
Data Clock	VC1
PosInput	ACB00
NegInput	ACB00
NegInputGain	Disconnected
ClockPhase	Normal
PulseWidth	1
PWM Output	None

LCDパラメータの設定

View > Chip Editor

1. 画面右上内の LCD_1 をクリック
2. LCD_1のパラメータを入力
名前の変更 LCD_1 → LCD
使用ポートの指定 Port2

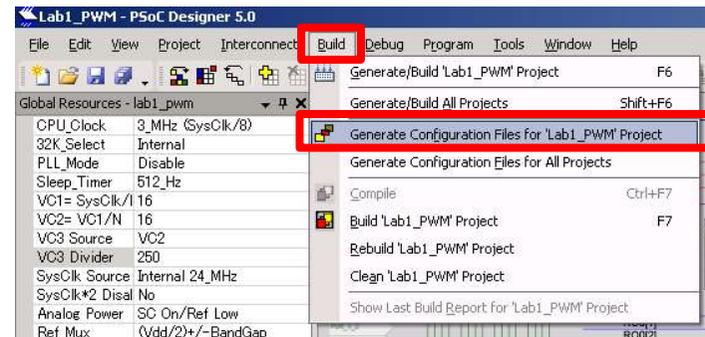


2

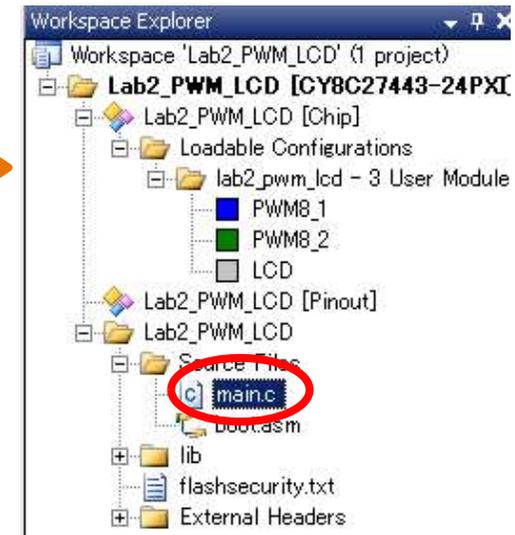
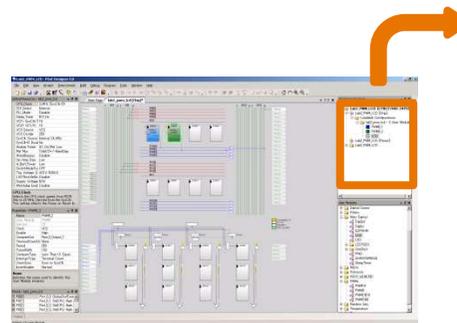
Properties - LCD	
Name	LCD
User Module	LCD
Version	1.5
LCDPort	Port 2
BarGraph	Enable

GC(Generate Configuration)

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



- GCが終了したら main.c をダブルクリック
ソースコード記述画面へ



ソースコード記述

- main関数内に
ソースコードを入力

LCD_Position 文の説明

LCD_Position(m, n)

m : 0 上の行に表示

1 下の行に表示

n : 左から n+1 文字目

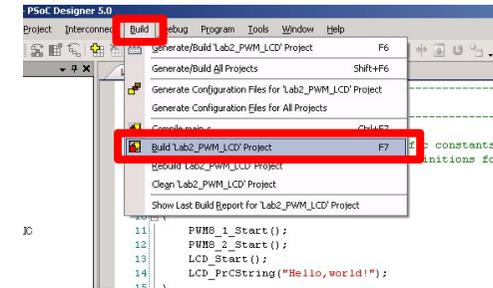
から表示(nは空白数)

```
Start Page lab3_adc [Chip] main.c
1 //-----
2 // C main line
3 //-----
4
5 #include <m8c.h> // part specific constants and
6 #include "PSoCAPI.h" // PSoC API definitions for all
7
8 |
9 void main()
10 {
11     unsigned int adc_data;
12     PGA_Start(PGA_HIGHPOWER);
13     LCD_Start();
14     LCD_InitBG(LCD_SOLID_BG);
15     MSC_EnableGInt;
16     ADCINC_Start(ADCINC_HIGHPOWER);
17     ADCINC_GetSamples(0);
18     while(1){
19         while(ADCINC_fIsDataAvailable() == 0);
20         adc_data = ADCINC_wClearFlagGetData();
21         LCD_Position(0,0);
22         LCD_PrHexInt(adc_data);
23         LCD_DrawBG(1,0,16,(adc_data/50));
24     }
25 }
26
```

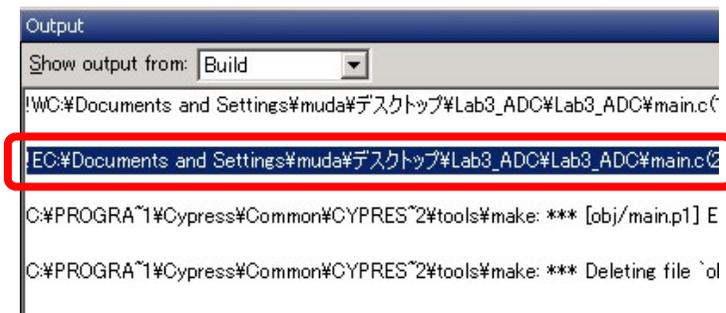
大文字のi

コンパイルとビルド

- Build > Compile 'lab3_adc' Project をクリック
- Build > Build 'lab3_adc' Project をクリック



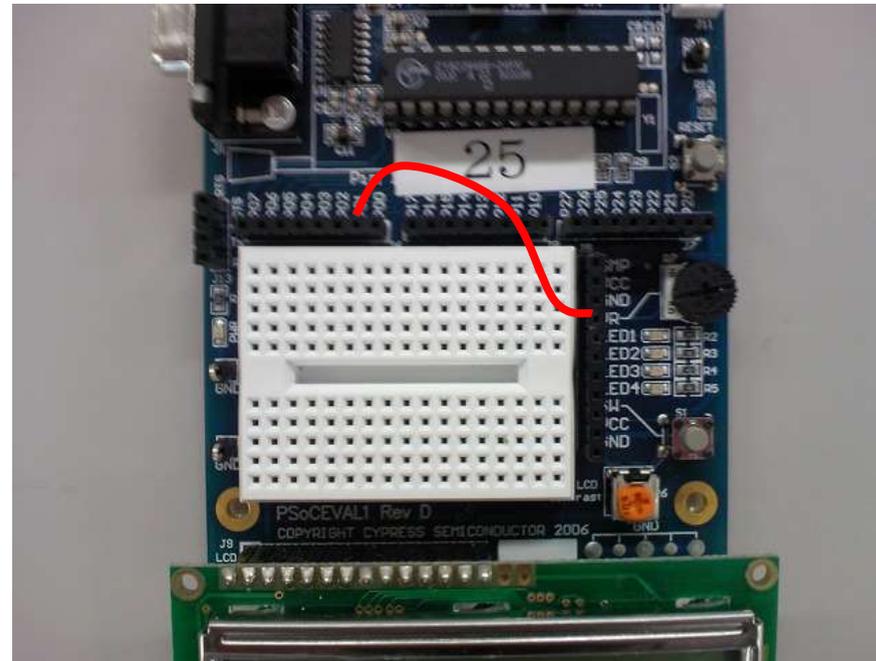
- Output Window でWarning や Errorが出たら
!W または!E の行を
ダブルクリックするとソースの
エラー原因周辺がハイライト表示される



```
17 ADCINC_GetSamples(0);
18 while(1){
19     while(ADCINC_flsDataAvailable() == 0);
20     adc_data = ADCINC_wClearFlagGetData();
21     LCD_Position(0,0);
22     LCD_PrHexInt(adc_data)
23     LCD_DrawBG(1,0,16,(adc_data/50));
24 }
25 }
26
```

MiniProgの接続,回路配線

- P01 と VRを接続
します。
- MiniProgをEval1に接続



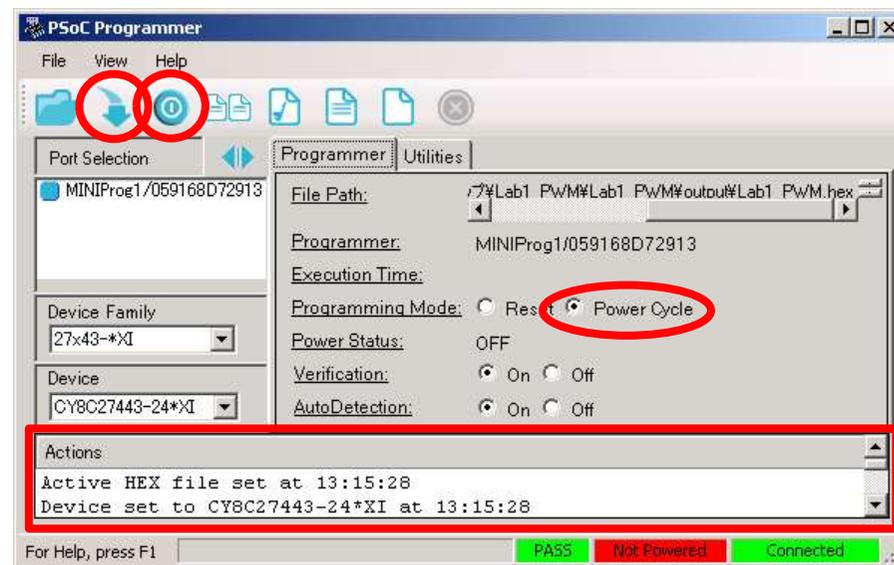
ファームウェアの書き込み

- Programming Mode
PowerCycle を選択

-  をクリックすると
書き込み開始

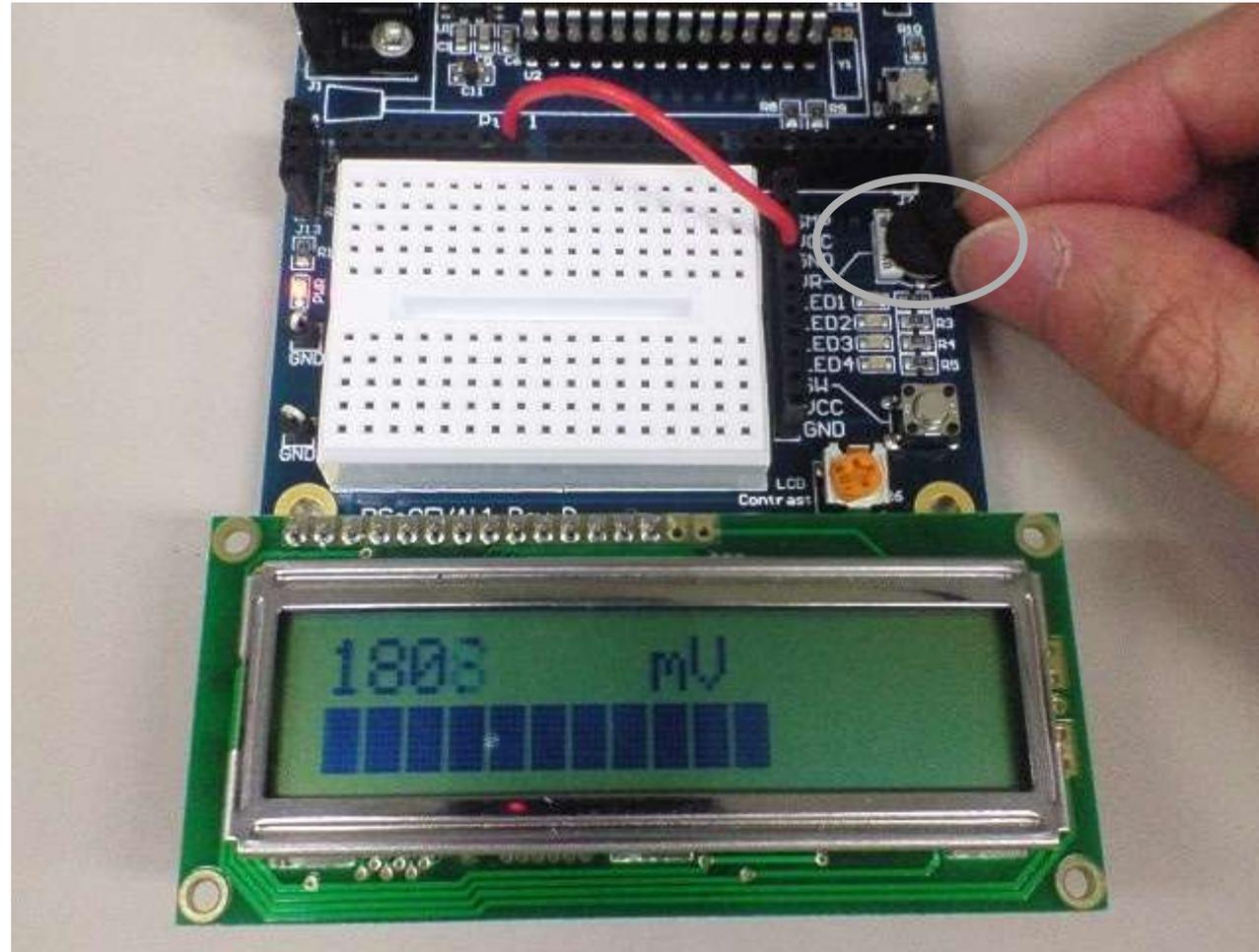
- Actions を読んで状況を確認

-  をクリックすると
MINIProgを通じて電源を供給



ボリュームをまわして動作確認

このラボの
ソースでは、
mVでの表示は
されません。



ボリュームに接触不良がある場合は、**表示が安定しない**ことがあります。その場合はボリュームを上から軽く押さえながら回してください。このノイズ電圧を**乱数のシード値**に使うこともできます。

Memo

フォローアップURL (Revised)

<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

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

<http://mikami.a.la9.jp/edu.htm>