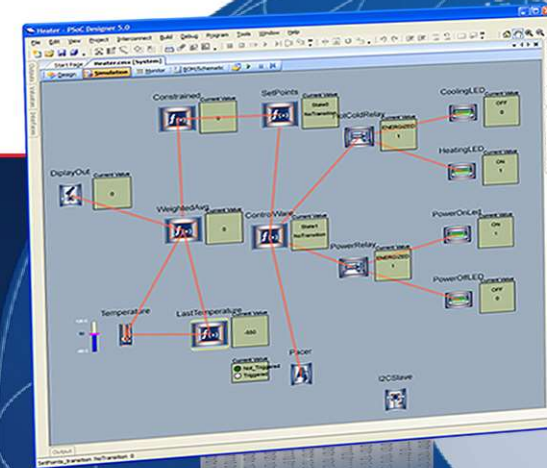


API関数を使った
LCDディスプレイへの表示

hello_world

PSoC Experiment Lab

Experiment Course Material V2.20
October 3rd, 2020
hello_world.pptx (28Slides)



Renji Mikami
Renji_Mikami(at_mark)nifty.com



ラボで使用するファイルのルール

Labで使用する資料,ファイルのディレクトリの作成ルールは、以下のとおりとします。絶対にまちがわないように注意してください

各自の演習ファイルの置き先:C:¥psoc_lab¥などのユニークな名前をつけたマスタディレクトリを作成します。(前の組が演習で作成した同名のC:¥psoc_lab¥がある場合は削除してかまいません)

各ラボのファイル群はC:¥psoc_lab¥のサブディレクトリに作成してください。例：ラボ名 hello_world の場合は各自で作成するのはファイル・ディレクトリ C:¥psoc_lab¥hello_world となります

解答例となる”完成プロジェクト”は、デスクトップのpsoc_lab_master(またはpsoc_lab_master201X)というディレクトリにあります。ない場合にはWEBのサポートURLからダウンロードしてデスクトップに置いてください。

使用ツール中では、テキスト中の”¥”文字は、“バックスラッシュ”になります

空白を含む日本語文字(2ByteCode)、半角カタカナ、は使用しないでください。

半角英数文字(1ByteCode)を使用してください。空白を使用したい場合は半角記号の_(アンダースコア、アンダーバー)を使ってください。

よくあるプロジェクト階層の失敗例

以前作成したプロジェクトの下の階層に誤って新規プロジェクトを作成することがあります。この場合、ファイルシステムが Corrupt して、ビルドシステムがエラーを起こします。正しく生成されたプロジェクトを開くファイルは、第2階層にあるアイコン付のhello_world.app です。このアイコンファイルが第3階層以下に作られている場合はプロジェクトの作成誤りです。C:\psoc_lab\hello_world\のディレクトリにあることを確認してください。

システムは第3階層にも同名のhello_world

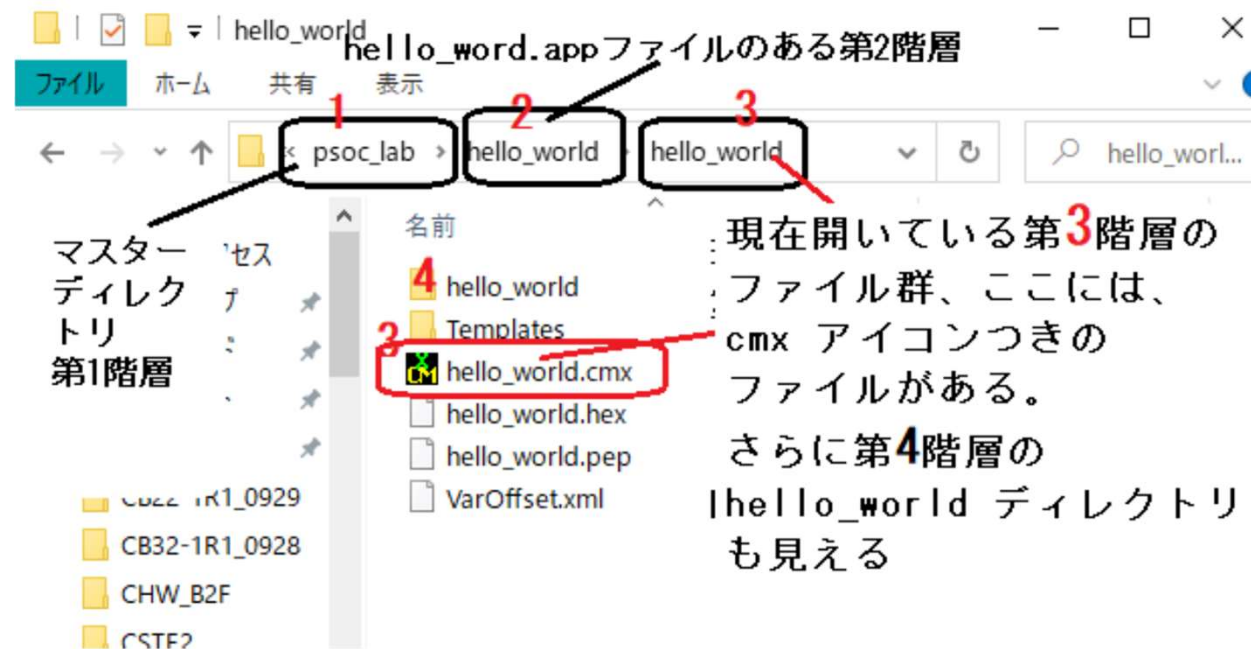
ディレクトリを自動生成しますから注意してください



プロジェクト階層のチェックと直し方

プロジェクト名が、hello_wold の場合、psoc_lab のマスターディレクトリのあとに何階層にもhello_wold サブディレクトリが生成されます。アイコン付のhello_world.app が第2階層にない場合は、PSoC_Designer に戻って、プロジェクト自体を削除して、できれば別名(hello_wold2など)で、第2階層に新しいサブディレクトリができていることを確認して進めてください。別名(hello_wold2など)を使うことにより問題の発見が容易になります。

参考までに、第3階層には、hellow_world.cmx という別のアイコンファイルが生成されますので問題解決のヒントにしてください。

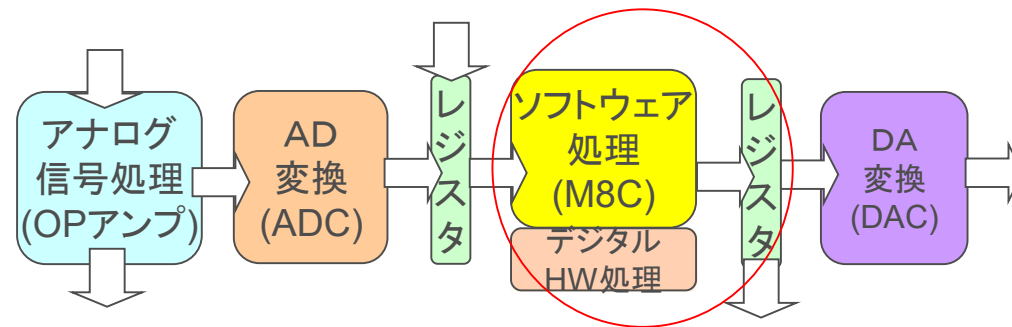




ラボ

hello world

LCDに文字を表示させます. 処理フローの演習





ラボ hello_world 手順

1.PSoC Designer を起動してProjectを作成します.

2.User Moduleから”LCD”を選択して配置します.このときモジュール名は自動的に**インスタンス番号が追加された名前**で登録されます.1個目の場合は**LCD_1**、2個目の場合は**LCD_2**、と自動的に番号が追加されます。この名前は変更できません。

この選択モジュール名とmain.cソース内で使用するモジュール名が一致しないとエラーになりますから注意してください.またPDFからソースをコピーした場合、見えないコードが付加される場合がありますから注意してください。

3.LCDのパラメータを設定してPSoCのGPIOのPort2に割り当てます.

4.main.cにソースを記載します.

LCD_Start(); 文は自動インスタンス番号追加により、モジュール名は**LCD_1** になりますから**LCD_1_Start()** と書いてください.

API関数も同様に**LCD_1_PrCString()**; となります. ()内は引数になります。

()内に記述された文字列が表示されます。各自好きな文字列を表示させてください。

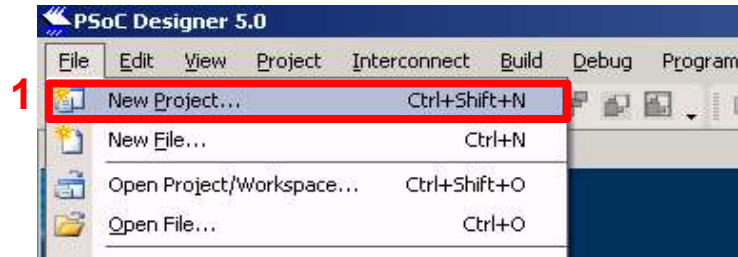
5.Generate Configurationを実行します

6.PSoC Programmerで書き込みして動作を確認します.

解説 :PSoC Designer では同じ機能のユーザーモジュールを複数使用するため識別のための番号が自動的に追加されます。

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

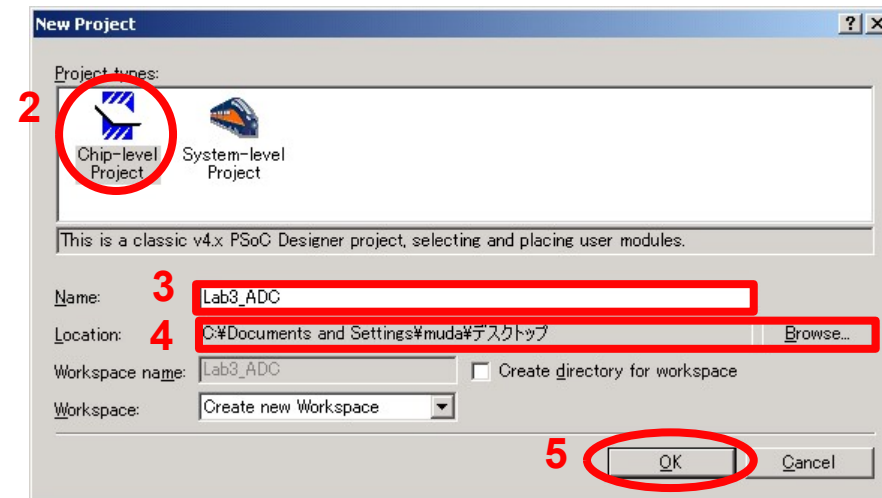
3. Name を入力

例: hello_woeld

4. Location を選択

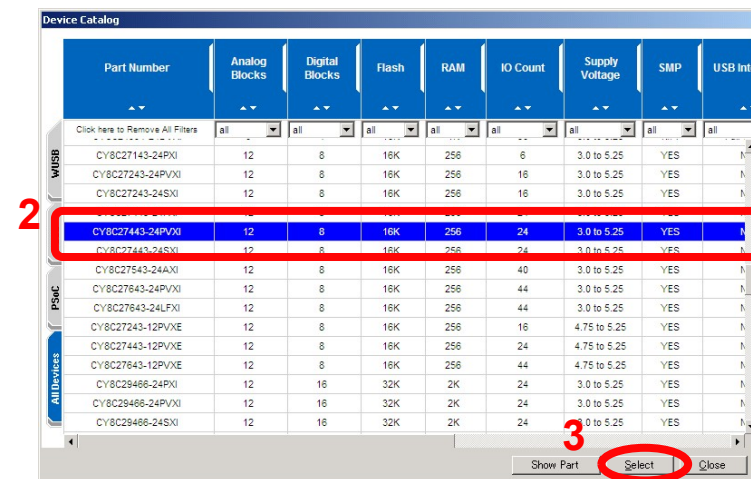
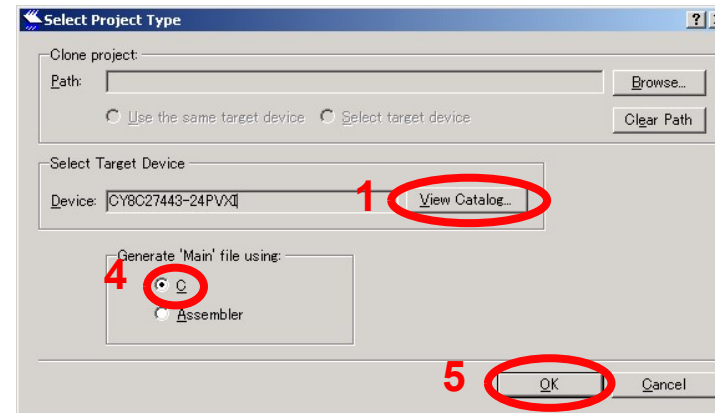
例: C:¥psoc_lab¥hello_world

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)

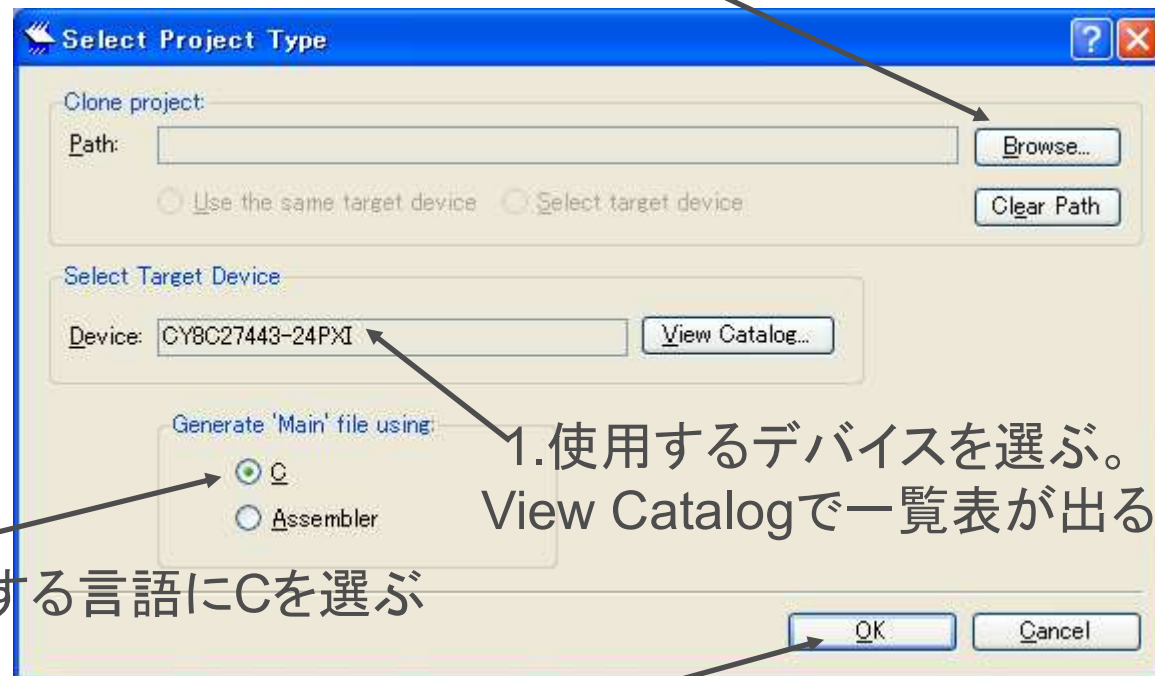
1. View Catalog をクリック
2. CY8C27443-24PXI を選択
3. Select をクリック
4. C を選択
5. OK をクリック





クローンの作製(参考-オプション)

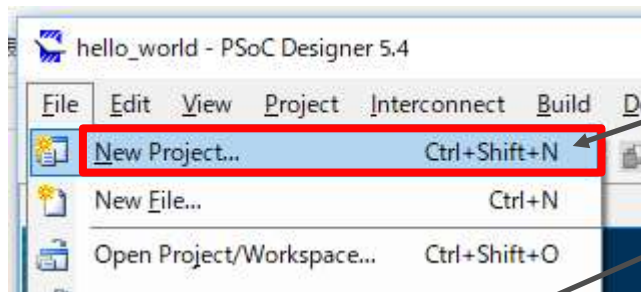
クローン化に関しては、Lab2_pwm_lcd を参照してください
既存のプロジェクトをコピーして使用する場合には、クローン作製をします
コピー元のプロジェクトの場所をBrowseするとコピーが作成されます。
使用デバイスを変更する場合などに使います。



- 1.使用するデバイスを選ぶ。
View Catalogで一覧表が出る
- 2.設計に使用する言語にCを選ぶ
- 3.決まったらOKをクリック



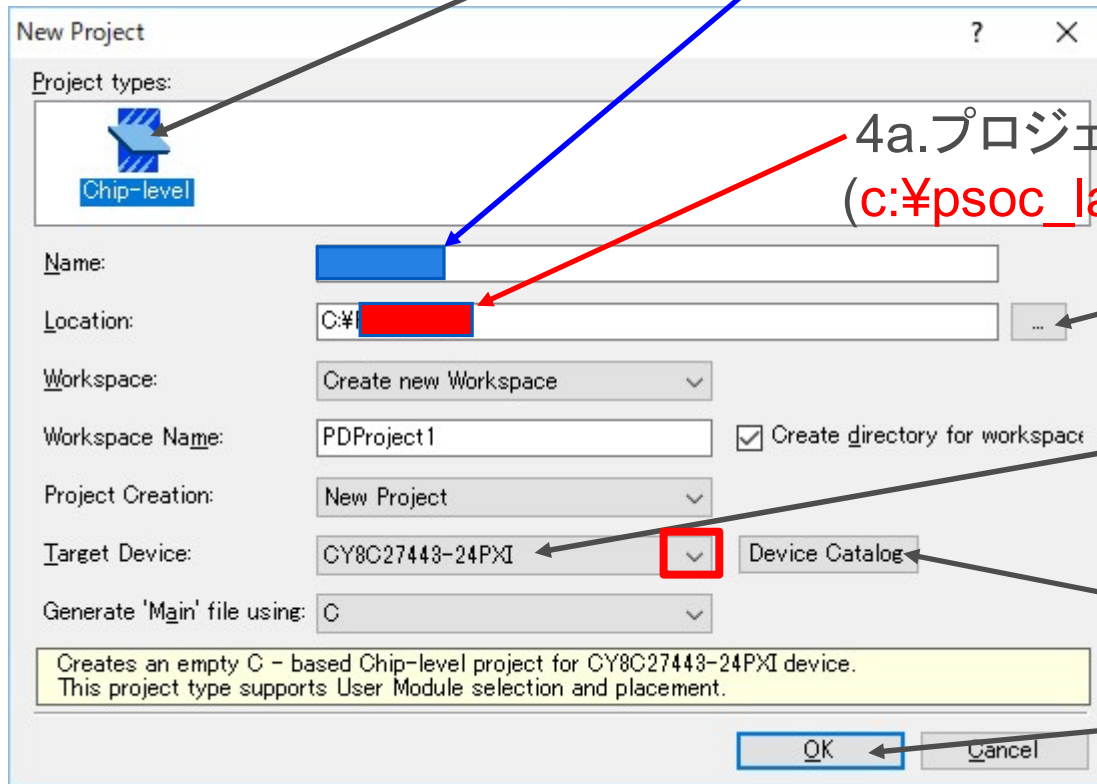
New Project: hello_worldの作成 (新版ソフトウェアの場合)



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(hello_world)を入力



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

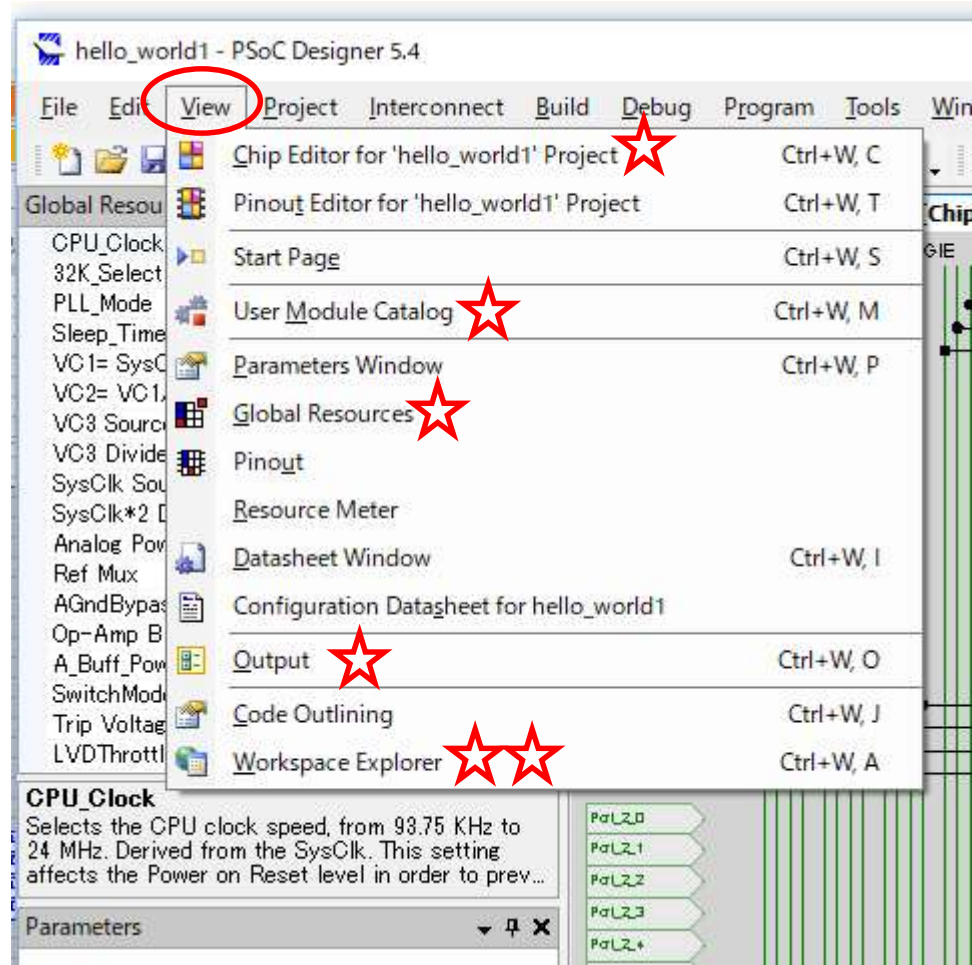
4b. 場所を選ぶときは
右の...をクリック。

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



PSoC Designerの各ウィンドウの開き方

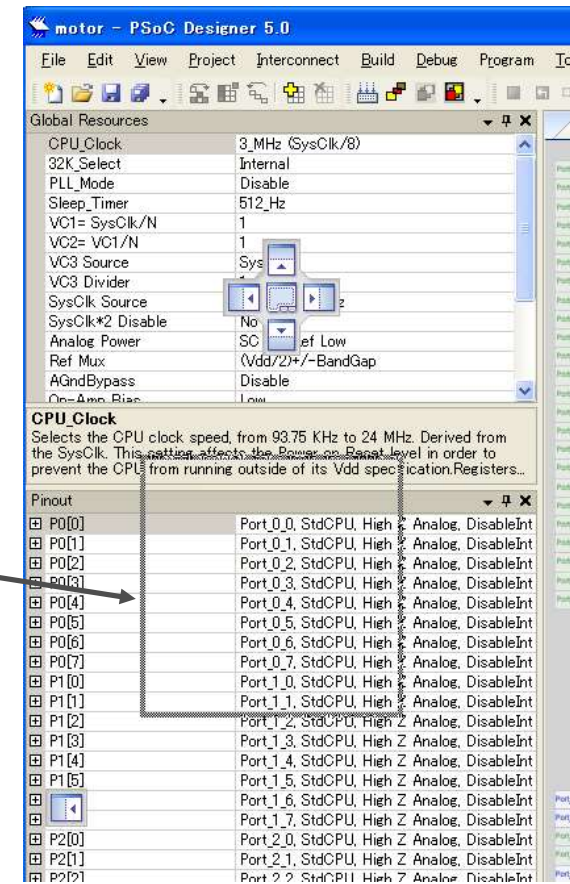
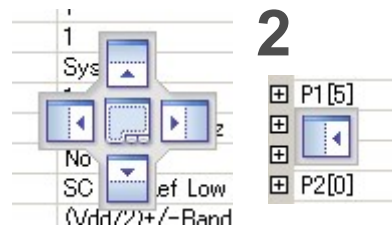
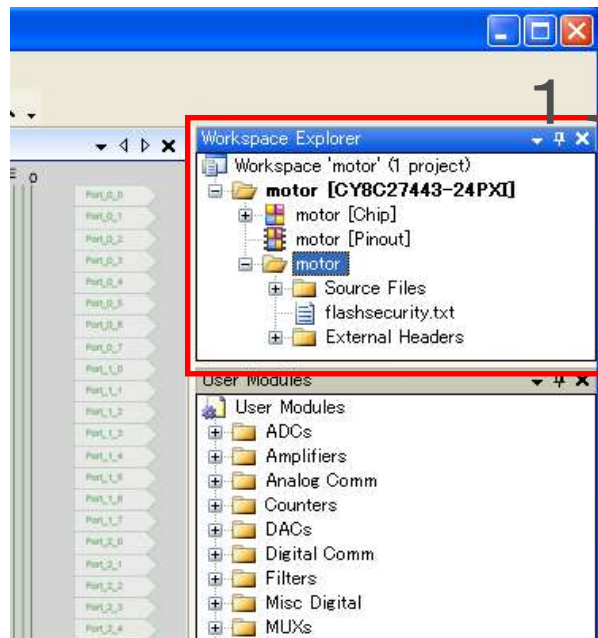


- 必要なウィンドウを開くためには、Viewタブをクリックして選択する。以下がよく使うもの。Workspace Explorerはいつも開いておくとよい。
- Workspace Explorer
- Output Window
- Chip Editor
- User Module Catalog
- Global Resource



ウィンドウの場所を自由に変更してみる

- 1. ためにWorkspace Explorer をクリックしたままで、画面の右端とか左端まで動かしてみる。
- 2. 任意の場所でクリックを離すとその場所にウィンドウが開く(フローティング)



- 2. 上の十文字や方向タブの場所でクリックを離すとPSoC Designerにドックした状態でウィンドウが固定される. ちょっとやってみよう

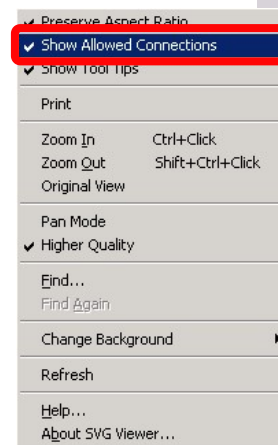


Chip Editor (回路図)の使い方

- 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



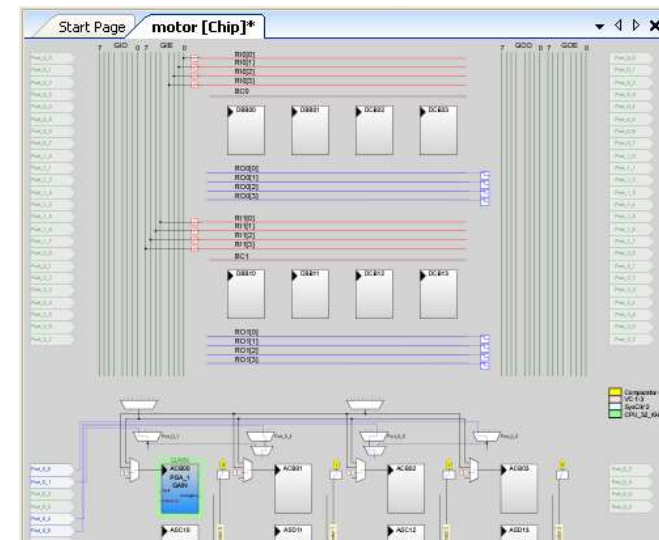
LCDだけの場合はハードウェア・リソースを使用しませんから回路図上にモジュールや配線はありません

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



ズームとパンモード切替

- 拡大、縮小アイコンでChip Editorをズームできます。任意の場所を拡大するには、Ctrlキーを押しながらマウスで拡大部分指定します。
- 手のアイコンは、矢印カーソルを出して配線を行うモードと画面を移動するパンモードを切り替えます。
- 手のマークのカーソルが出ているパンモードでは、画面の移動ができますが配線はできません。
- 矢印のカーソルが出ているときには、配線ができますが、画面は固定されます。
- 移動とズームをためしてみてください

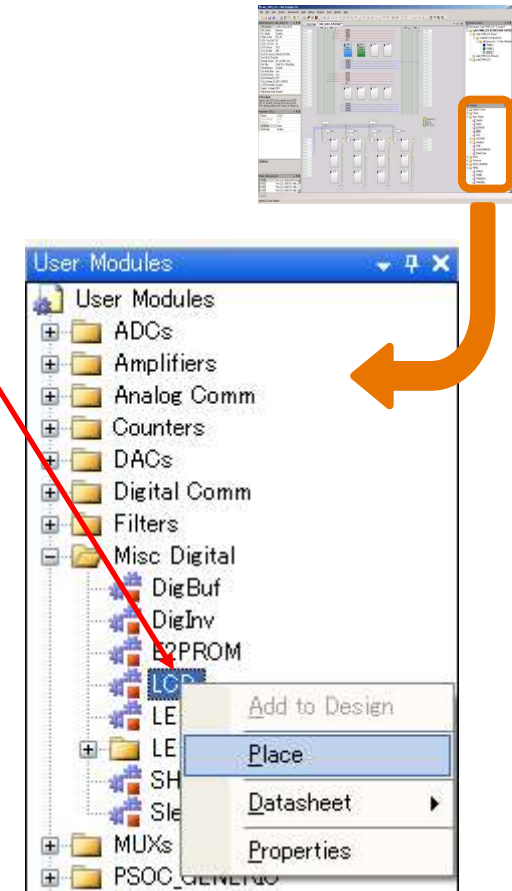
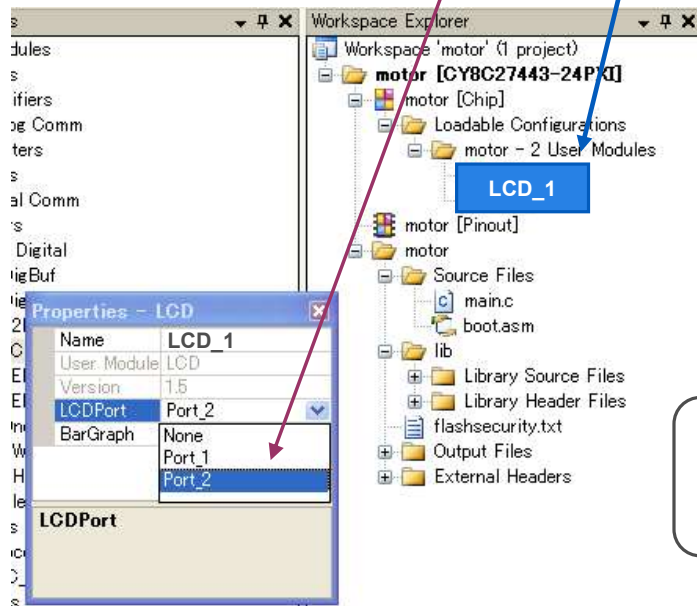


Chip Editor ウィンドウ



User Module CatalogからLCDを選ぶ

- ユーザーモジュールカタログのウィンドウを開き Misc Digital グループの下のLCDをハイライト.
- 右クリックでDatasheet を開くことができる. また Place を選ぶかLCDをダブルクリック選択. **自動的にLCD_1**とリネームされます.
- Workspace Explorer のLCD_1をハイライトするとProperties ウィンドウにLCD_1のパラメータが表示されるのでPort_2を選択

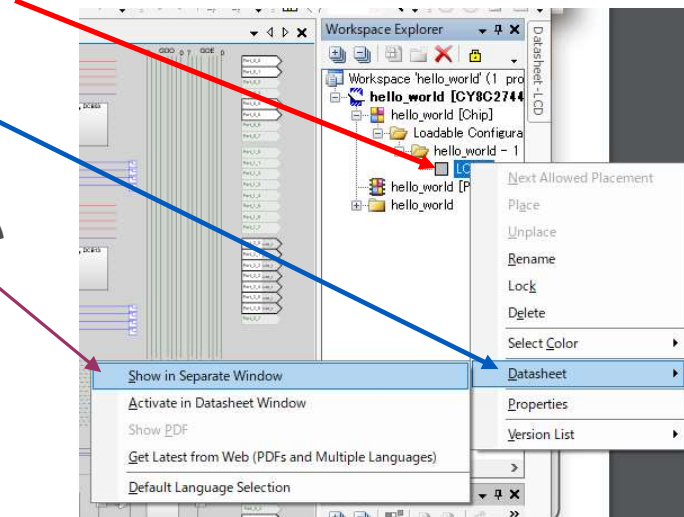


通常のユーザーモジュールの場合、内部のハードウェア・リソースが割り当てられるが、LCDの場合はAPI(デバイス・ドライバ)だけのため内部のハードウェア・リソースは消費されません



LCDに使えるAPI関数を調べてみる

- Workpsce ExploreからLCD_1をクリックしてハイライト表示させ、続いて右クリックで、Datasheet、さらにShow in Separate Windowsを選択するとCharacter LCD Datasheet が開きます。
- Datasheet - LCD ウィンドウを大きく開いてくださいLCD User Module で使えるAPI関数とサンプルソースコードを見ることができます。



Datasheet - LCD

Application Programming Interface

The Application Programming Interface (API) routines are provided as part of a module at a higher level. This section specifies the interface to each function file.

Note: In this, as in all user module APIs, the values of the A and X register responsibility of the calling function to preserve the values of A and X before "registers are volatile" policy was selected for efficiency reasons and has been compiler automatically takes care of this requirement. Assembly language too. Though some user module API function may leave A and X unchanged.

For Large Memory Model devices, it is also the caller's responsibility to preserve MVW_PP registers. Even though some of these registers may not be modified in future releases.

Here are the API programming routines provided for the LCD User Module:

Basic Character LCD Functions

LCD_Start

Description:
Initializes LCD to use the multi-line 4-bit interface. This function

C Prototype:
`void LCD_Start(void);`

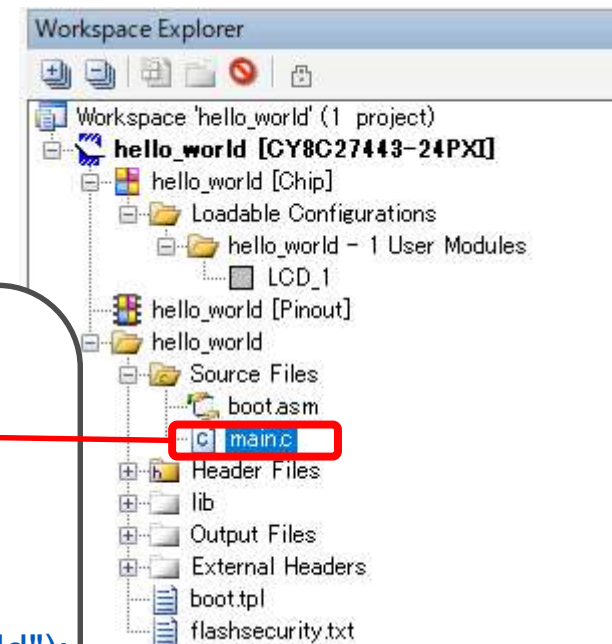
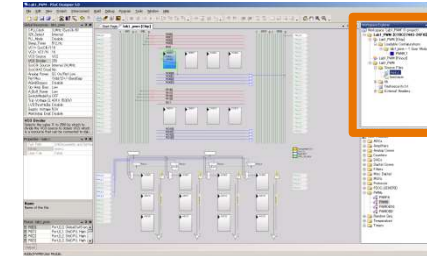
Assembly:
`lcall LCD_Start`

Resources	PSoC® Blocks			API Memory (Bytes)		Pins (per External I/O)
	Digital	Analog CT	Analog SC	Flash	RAM	
CY8C29/27/26/25/24/22/21xxx, CY8C23x33, CY7C603xx/64215, CYWUSB6953, CY8C20x34, CY8CLED02/04/08/16, CY8C21x45, CY8C22x45, CY8C28x45, CY8CPLC20, CY8CLED16P01, CY8C28xxx, CY8C21x12						
Bar Graph Enabled	0	0	0	646	0	7 from One Port
Bar Graph Disabled	0	0	0	434	0	7 from One Port



main.c ソース式にAPI関数で記述

```
hello_world - PSoC Designer 5.4
File Edit View Project Interconnect Build Debug Program Tools
Start Page hello_world [Chip] main.c*
1 //-----
2 // C main line
3 //-----
4
5 #include <m8c.h> // part specific constants
6 #include "PSoCAPI.h" // PSoC API definitions
7 void main()
8 {
9     // Insert your main routine code here.
10 LCD_1_Start();
11 LCD_1_Position(0,0);
12 LCD_1_PrCString("Hello World");
13 }
```

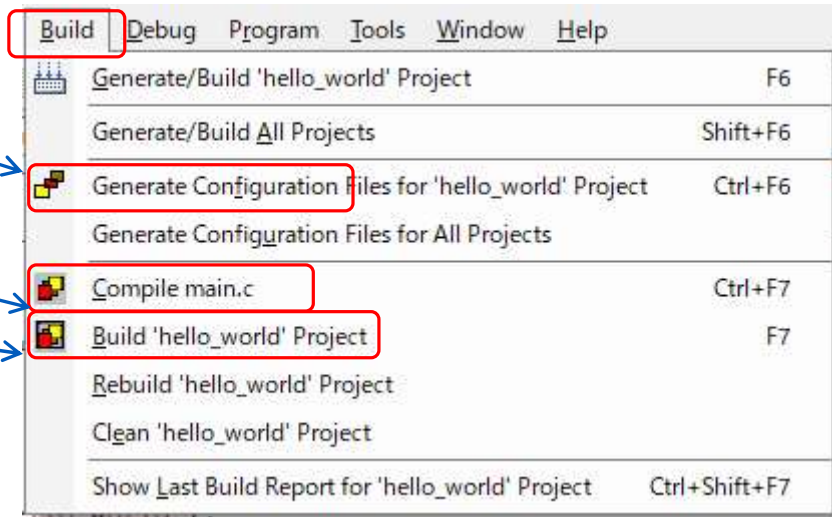


右の記述を追加して
LCDに”Hello PSoC!”
と表示させる
Position(0,0)の意味は
LCD表示開始位置
最初の0: 1行目 (1なら2行目)
次の0: 1文字目 (5なら6文字目)
Datasheet ウィンドウで
API関数を確認してみよう

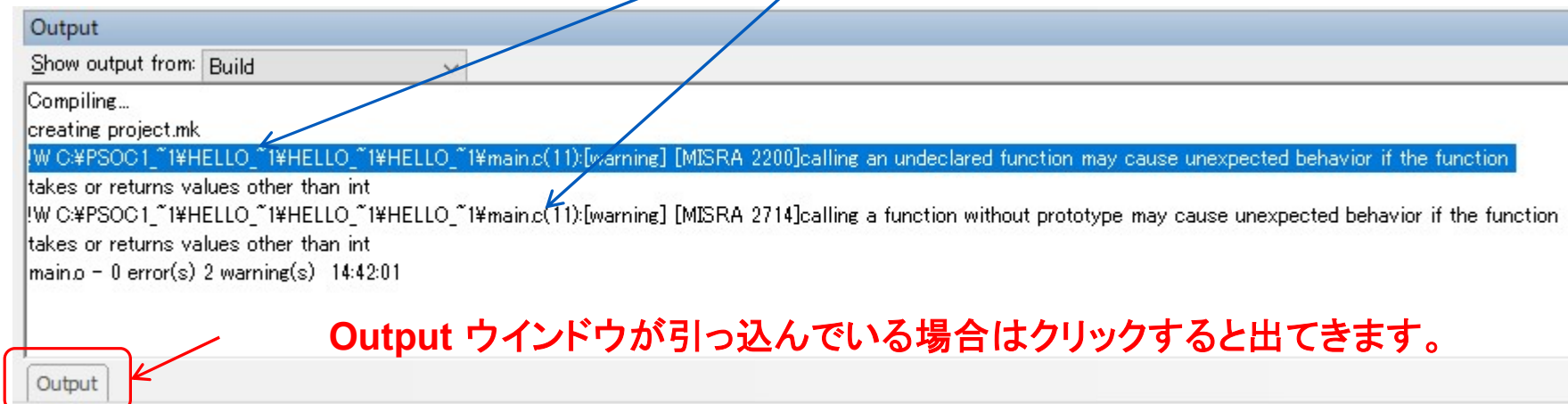
```
void main()
{
LCD_1_Start();
LCD_1_Position(0,0);
LCD_1_PrCString("Hello World");
}
```

★ 書き込みデータの作成

1. Build > Generate Configuration の実行
2. Build > Compile の実行
3. Build > Build の実行

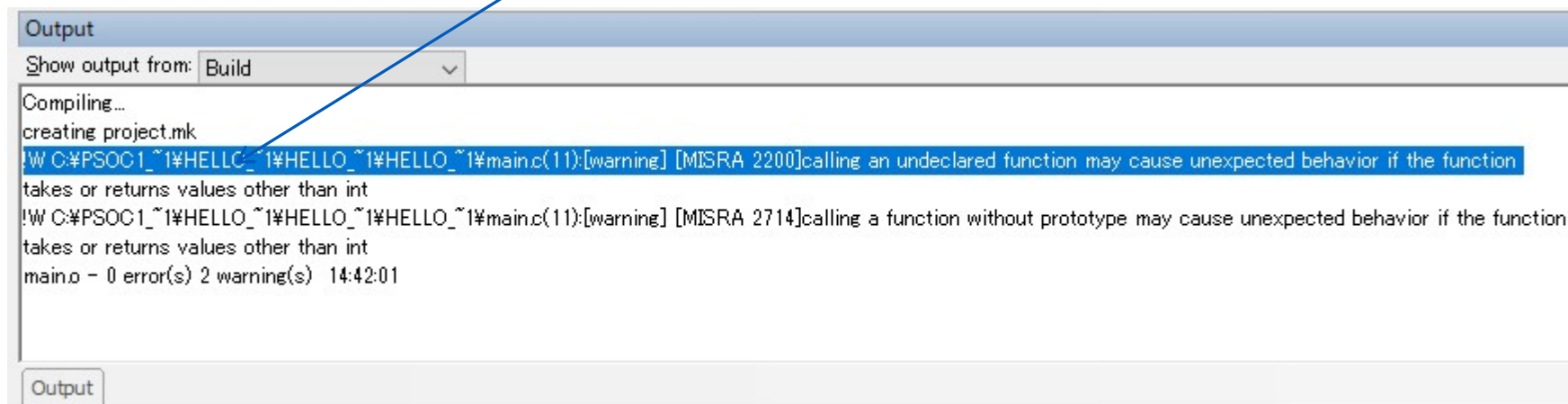


必ず順番どおりに作業をすすめること
エラーが出たらその時点でデバッグする
コンパイル・エラーは、Output Window の ! Wマーク
の行をクリックするとエラーの原因付近を表示するので、ソースコードをデバッグ



★ デバッグ

!W マークのある行を上から順にクリックする



```
Output
Show output from: Build
Compiling...
creating project.mk
!W C:\PSOC1\~1\HELLO_~1\HELLO_~1\main.c(11):[warning] [MISRA 2200]calling an undeclared function may cause unexpected behavior if the function
takes or returns values other than int
!W C:\PSOC1\~1\HELLO_~1\HELLO_~1\main.c(11):[warning] [MISRA 2714]calling a function without prototype may cause unexpected behavior if the function
takes or returns values other than int
main.o - 0 error(s) 2 warning(s) 14:42:01
```

自動的にソースコードのエラーの原因付近を表示、ソースをデバッグ
このケースでは、PrCStringの大文字 P を小文字 p とタイプミス

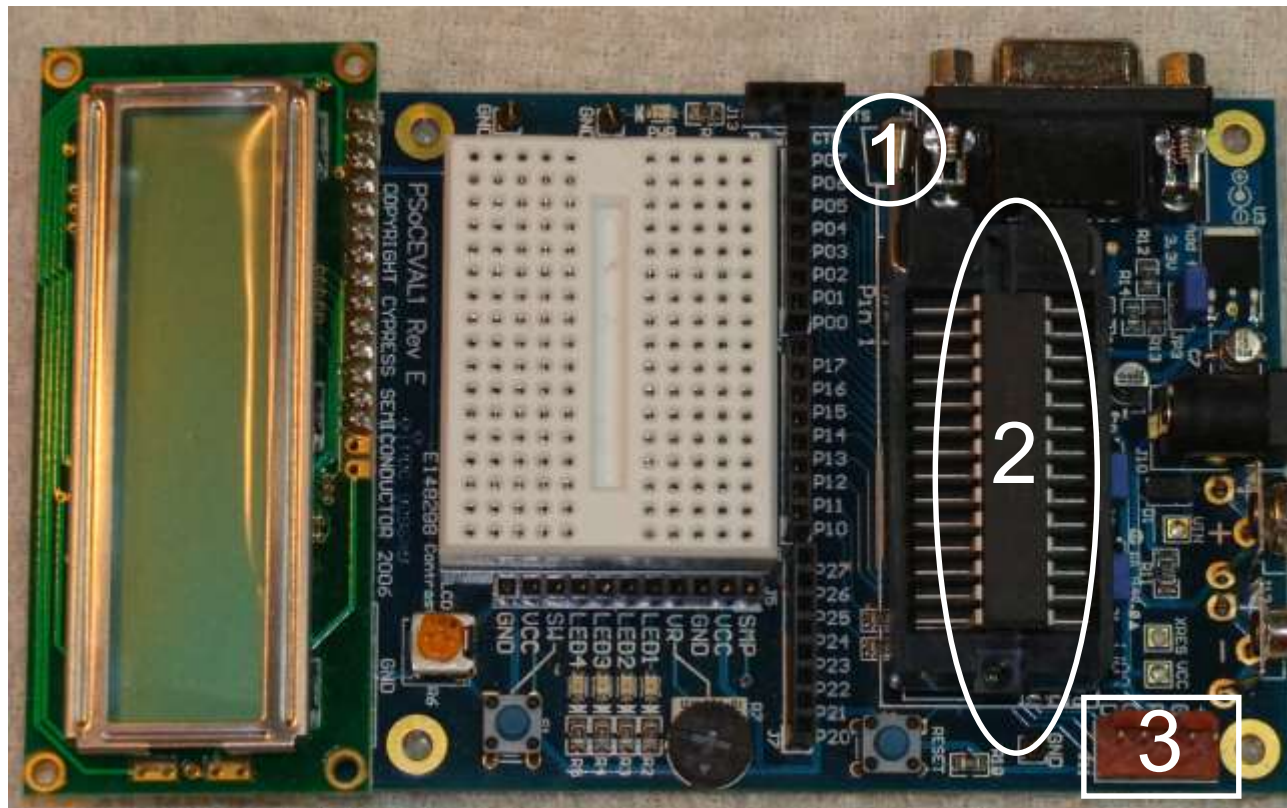


```
5 #include <m8c.h> // part specific constants and macros
6 #include "PSoCAPI.h" // PSoC API definitions for all User Modules
7 void main(void)
8 {
9 LCD_1_Start();
10 LCD_1_Position(0,0);
11 LCD_1_prCString("Hello World");
12 }
13
```

★ 基板の準備

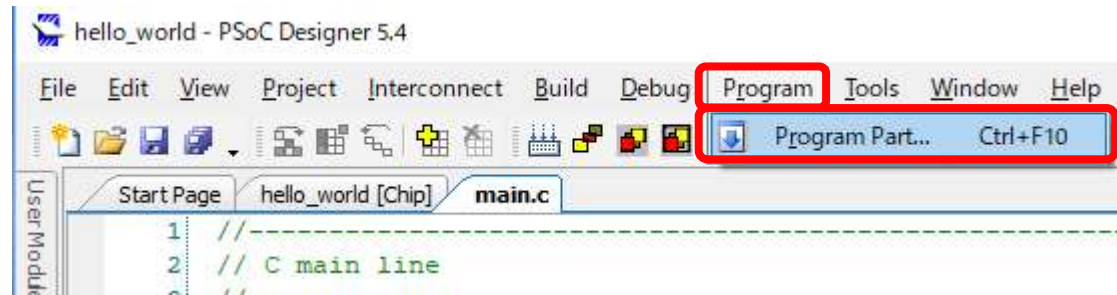
レバー①を上立てソケットを開きデバイス27443を実装②し、レバーを倒してデバイスを中央でしっかり固定してください。

続いてMIniProgを③ピン位置がずれないようにさしこんでください。

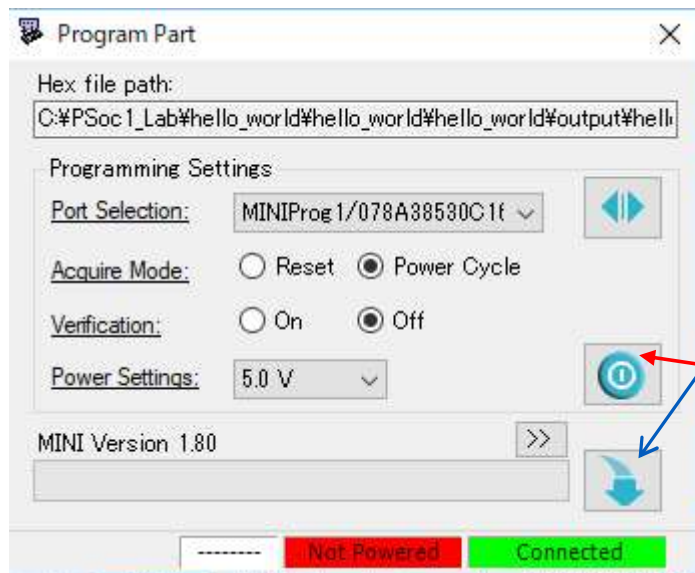




書き込み

- 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。

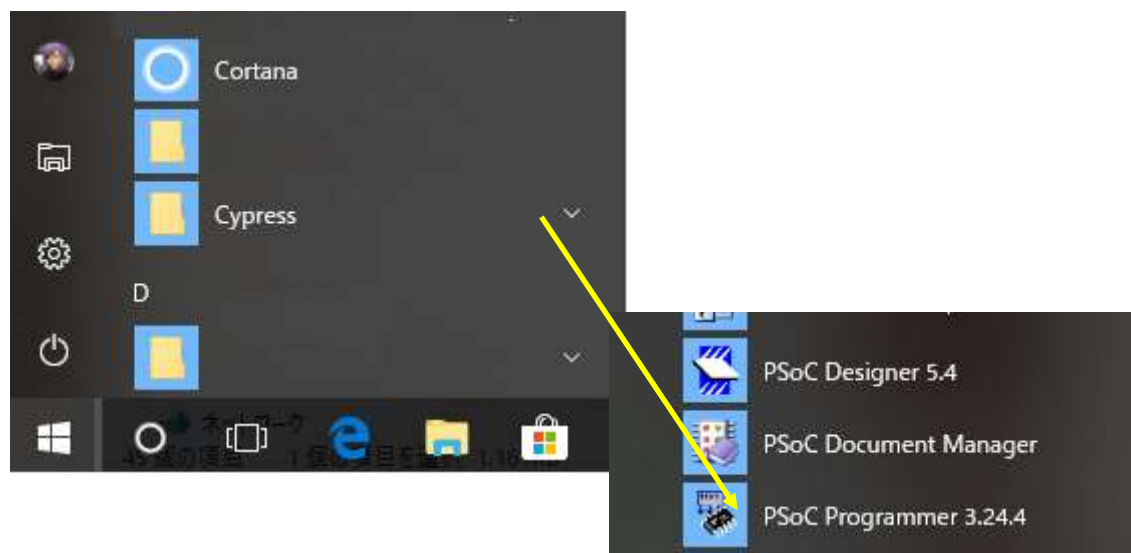


-  をクリックすると書き込み開始
- Actions を読んで状況を確認
-  をクリックすると MINIProgを通じて電源を供給
- LCDにHello PSoC!と表示されていれば完成です。

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。

追補ページ(参考/オプション): PSoC Programmer の単独起動について

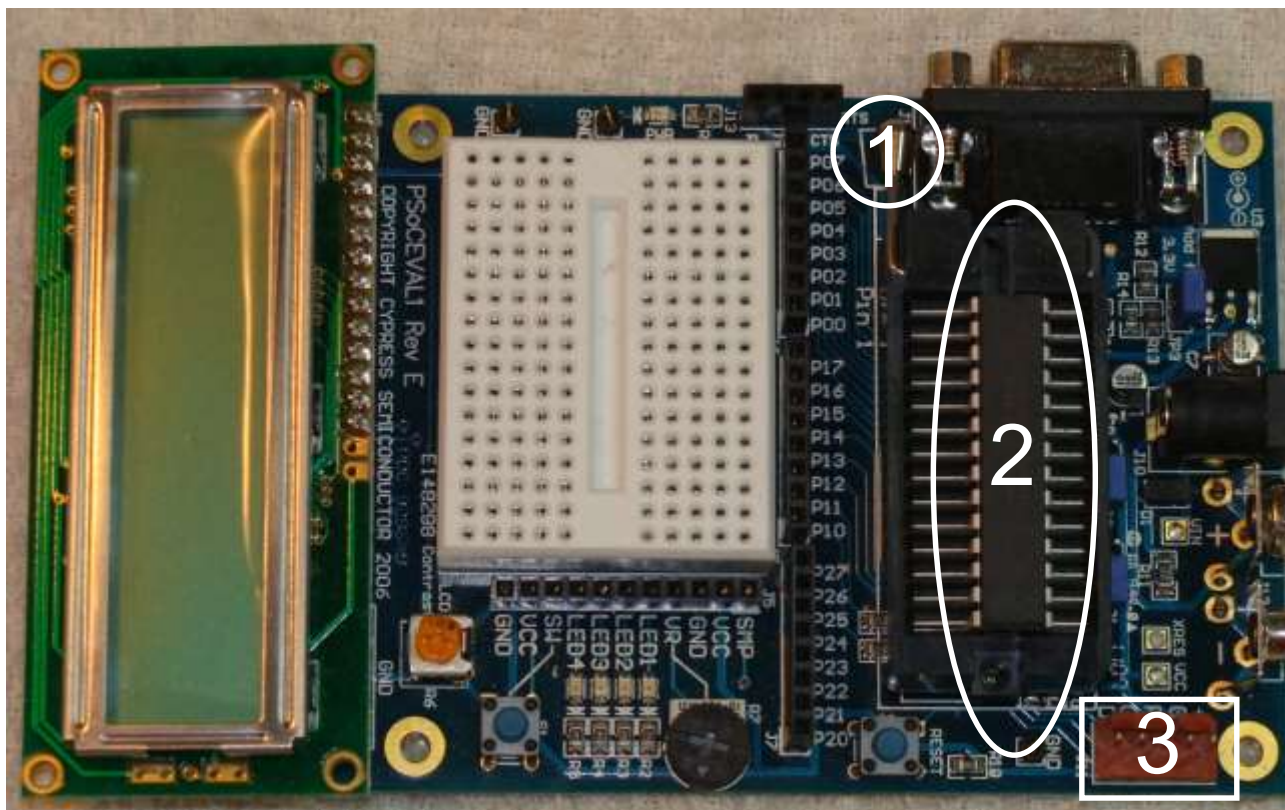
- PSoC Programmer を単独起動し、HEXファイルをロードして書き込む方法(Widows10) :すべてのアプリ>Cypress>PSoC Programmer
- HEXファイルは、¥output サブディレクトリの下にあります
- 例 C:¥PSoC_Lab¥hello_world¥rld¥hello_world
¥output¥hello_world.hex



★ 追補ページ(参考/オプション): 基板の準備

レバー①を上にしてソケットを開きデバイス27443を実装②し、レバーを倒してデバイスを中央でしっかり固定してください。

続いてMIniProgを③ピン位置がずれないようにさしこんでください。





追補ページ: Windows7の場合

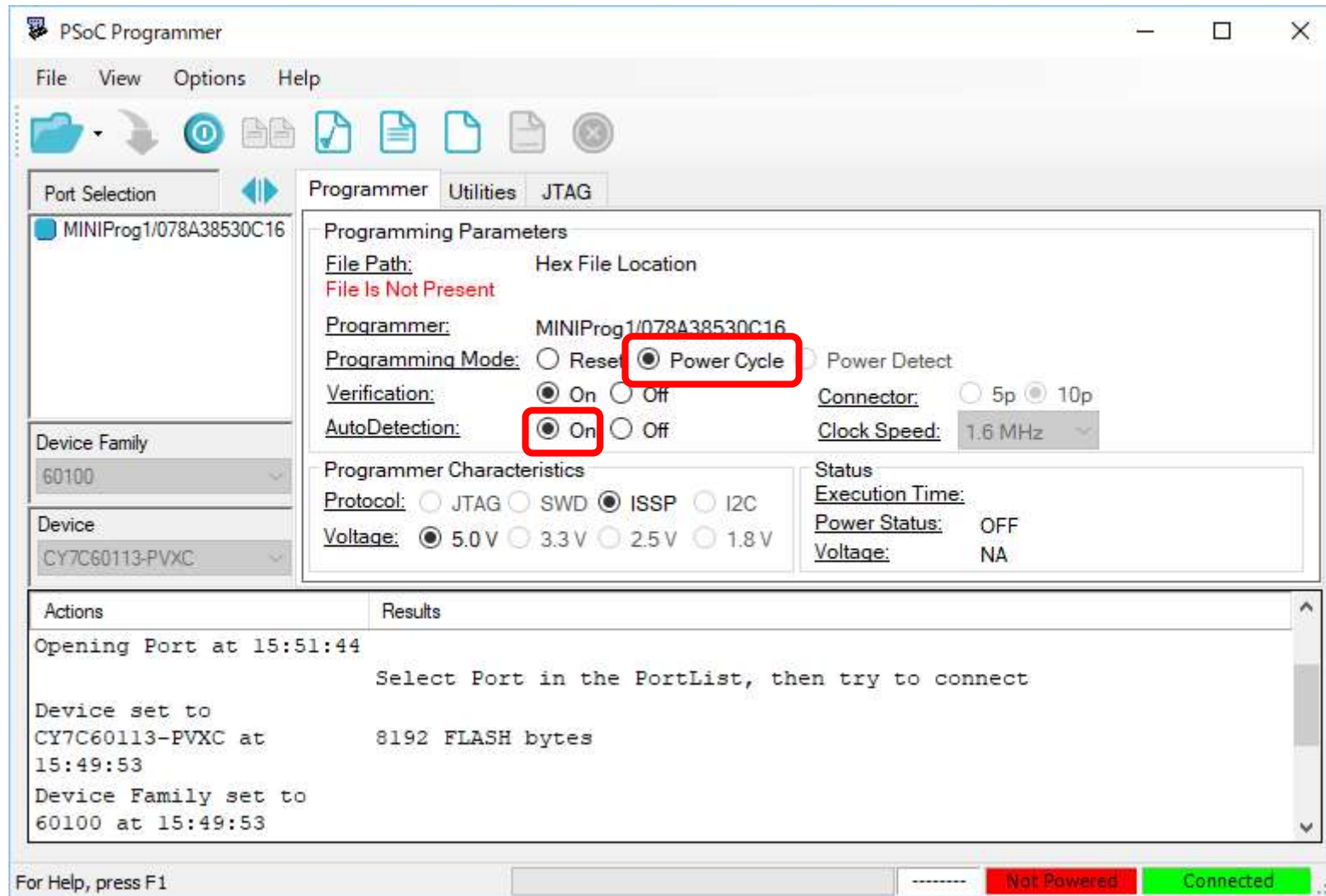
- スタート>すべてのプログラム> Cypress>PSoC Programming>PSoC Programmer をクリック(win7の場合)

左の画面が現れます
MiniProgが認識されない
場合は,Part Selectionウ
ィンドウに表示されている
番号表示のある
MiniProg1をクリックして選
択してください

★ 追補ページ: Programmerの設定の確認

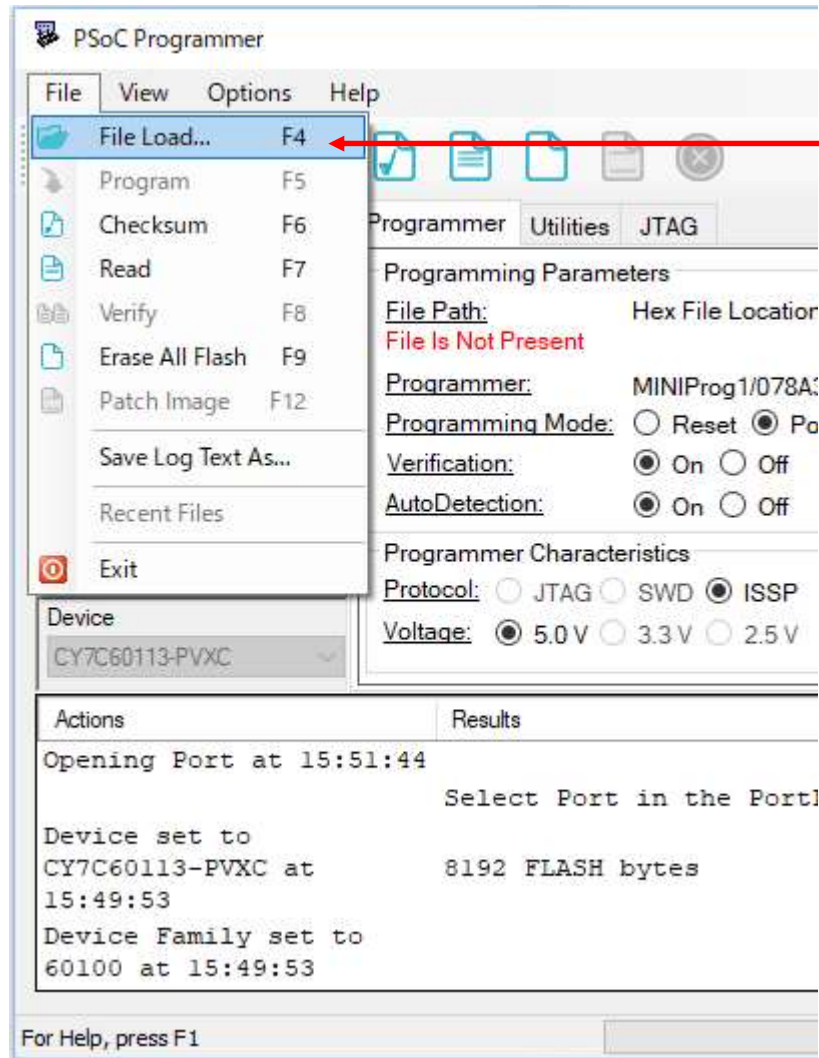
Programming Mode: PowerCycle を選択

AutoDetection: On を選択





追補ページ: hello_world.hexファイルのロード



File > Load をクリック

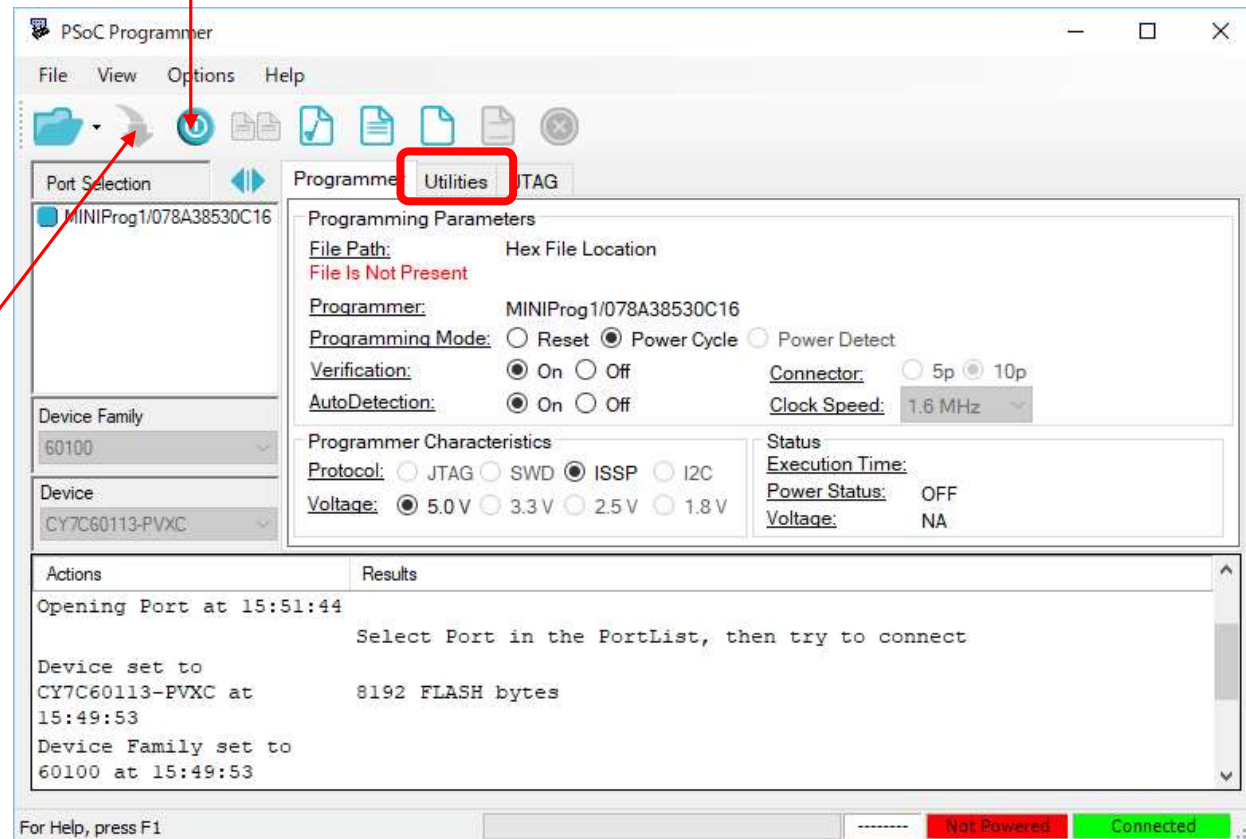
...¥output¥hello_world.hex
を開く



追補ページ:hexファイルの書き込み/動作しない場合

- MINIProgのファームウェアが古い場合、Utilities > UpgradeFirmware でアップデートします。
- MiniProgを認識しない場合にはUSBポートから引き抜いてもう一度挿しなおして再認識させてください。(認識には時間がかかりますから少し待ってください)
- デバイスは、一度書き込めばUSB電源をONにするだけで動作します。毎回プログラムする必要はありません。基板は、DCアダプタ、006P 9V電池でも動作します。

書き込みは
ここをクリック



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 (Default - Recommended)

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

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