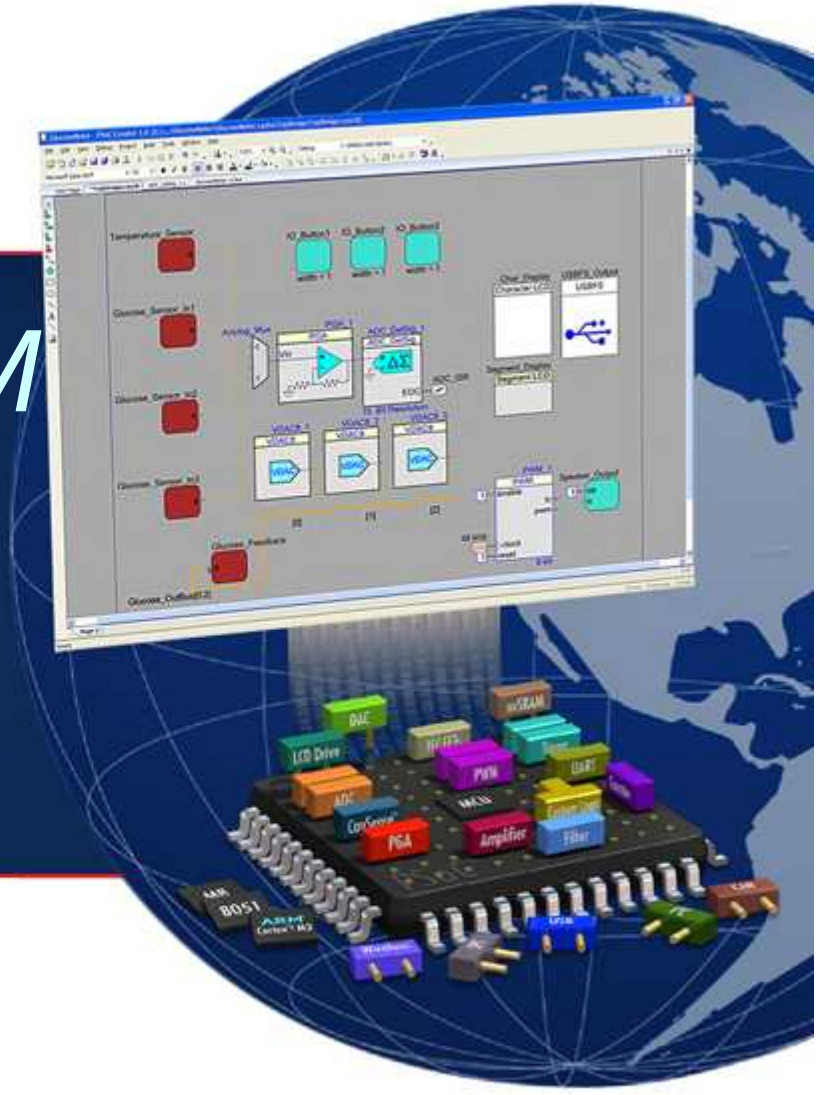
3.4

PWMによるLEDの点滅コントロール

LED Control By PWM H/W Component PSoC Experiment Lab



LED Control By Hardware Component
Experiment Course Material 3.4 V 2.10
March 7nd. 2012
PWM_LED_B_35.PPT (53 Slides)

Renji Mikami
Renji_Mikami@nifty.com

コンテンツ次スライドに統合
そのあと削除

Lab PWM_LED_B_35
PWM ユーザーモジュールの使い方
ハードウェアによる処理の理解

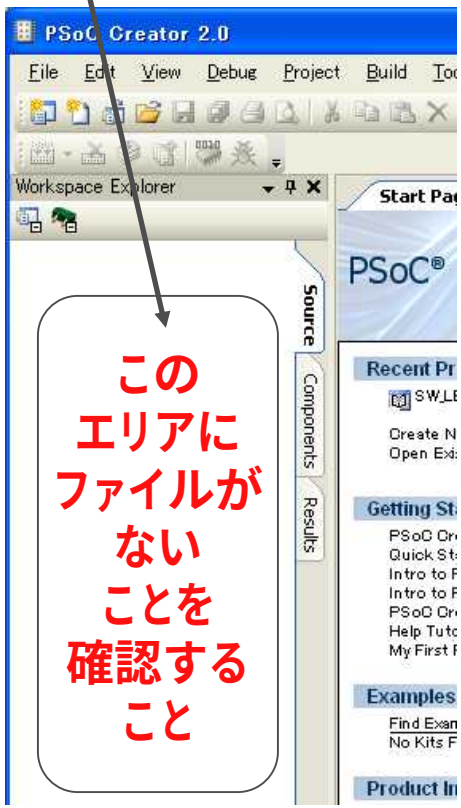
LEDの点滅（ソフトウェア編）

```
Pin_N_Write();      //ピンNに論理値を与える
CyDelay();          //ミリ秒で遅延を設定
```

PWM_LED_B_35
ラボの目的

- “ハードウェア”のコンポーネントの使用
- PWMの機能と設定の方法
- システム･クロックの設定
- 割込み
- DMA

前のプロジェクトをセーブし、
File>Close Workspace で
終了してから、次のプロジェクト
に進みます。



デザインフロー

Configure

- Start a new project
- Place components
- Configure components
- Connect components

ライブラリに登録された
機能設定可能な"部品"

コンポーネント（ハードウェア）
の配置と機能の設定

- Topdesgn.cysch(ブロック接続図)
- プロジェクト名.cydr(ピンアサイン図)

Develop

- Build hardware design and generate component APIs
- Write application code utilizing component APIs
- Compile, build and program

解説

ソフトウェアの記述(C言語)

Debug

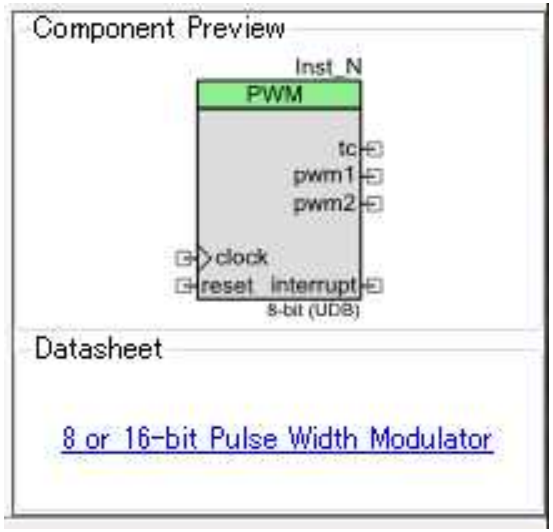
- Perform in-circuit debug using the MiniProg3 and PSoC Creator

書き込みとデバッグ

Reuse

- Capture working hardware/software designs as your own components for future use

コンポーネントのデータシート



サンプルソースコードがあります

シンボルの上にカーソルを置いて
右クリックしてメニューを開く。
Open Data Sheetを選ぶと
PWMコンポーネントの詳細が見れる

Design-Wide Resource Manager (.cydwr)

解説

Clocks(クロック)

Interrupts(ハードウェア割込み)

- Set priority and vector

DMA(Direct Memory Access設定)

- Manage DMA channels

System

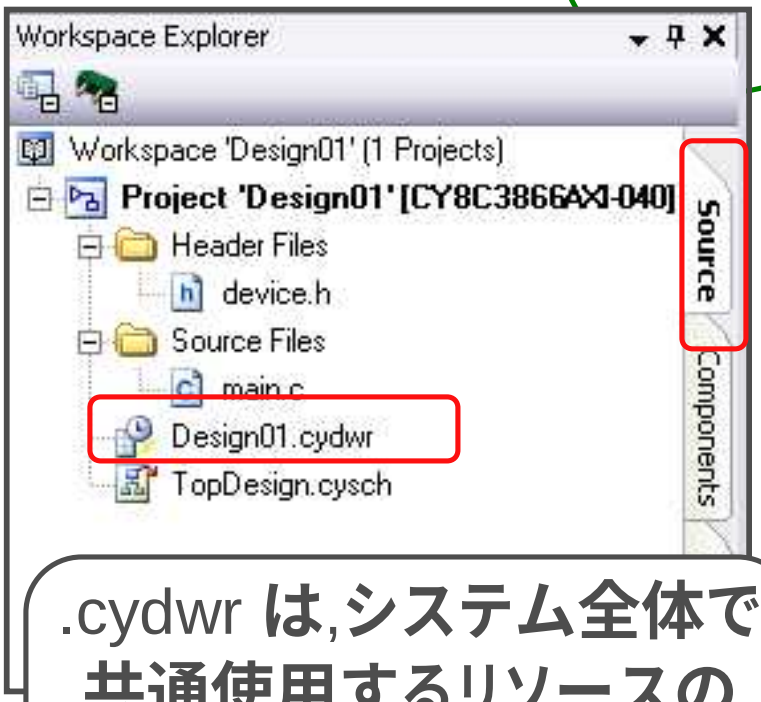
- Debug, boot parameters, sleep mode API generation, etc.

Directives

- Over-ride placement defaults

Pins

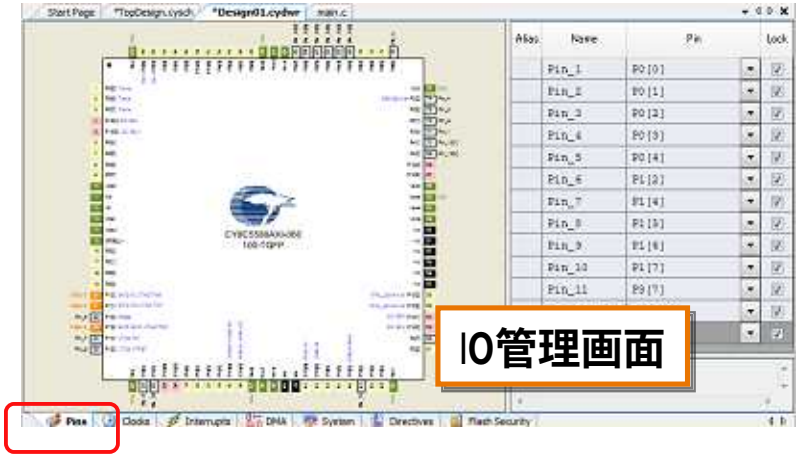
- Map I/O to physical pins and ports
- Over-ride default selections

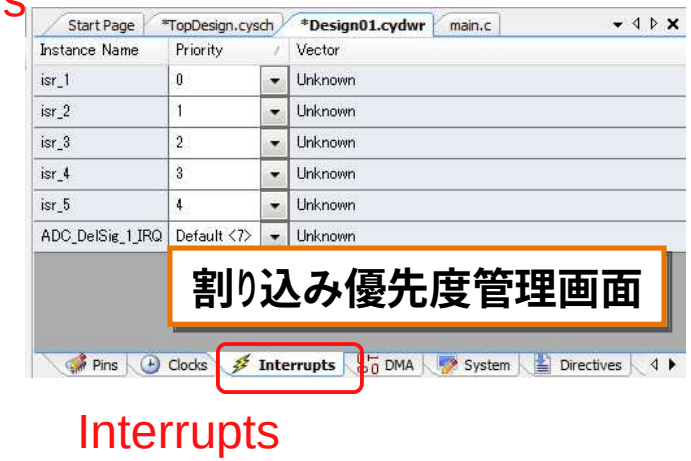


.cydwr は,システム全体で共通使用するリソースの設定を行う(各コンポーネント共用)

共有リソースエディタ(.cydwr画面から選択)

PSoC Creatorでは、デバイスの固有リソースの割り当てを行うために共有リソースエディタが用意されています。.cydwr画面を表示して**画面下のタブ**を選択することで、表示する画面の切り替えを行います。



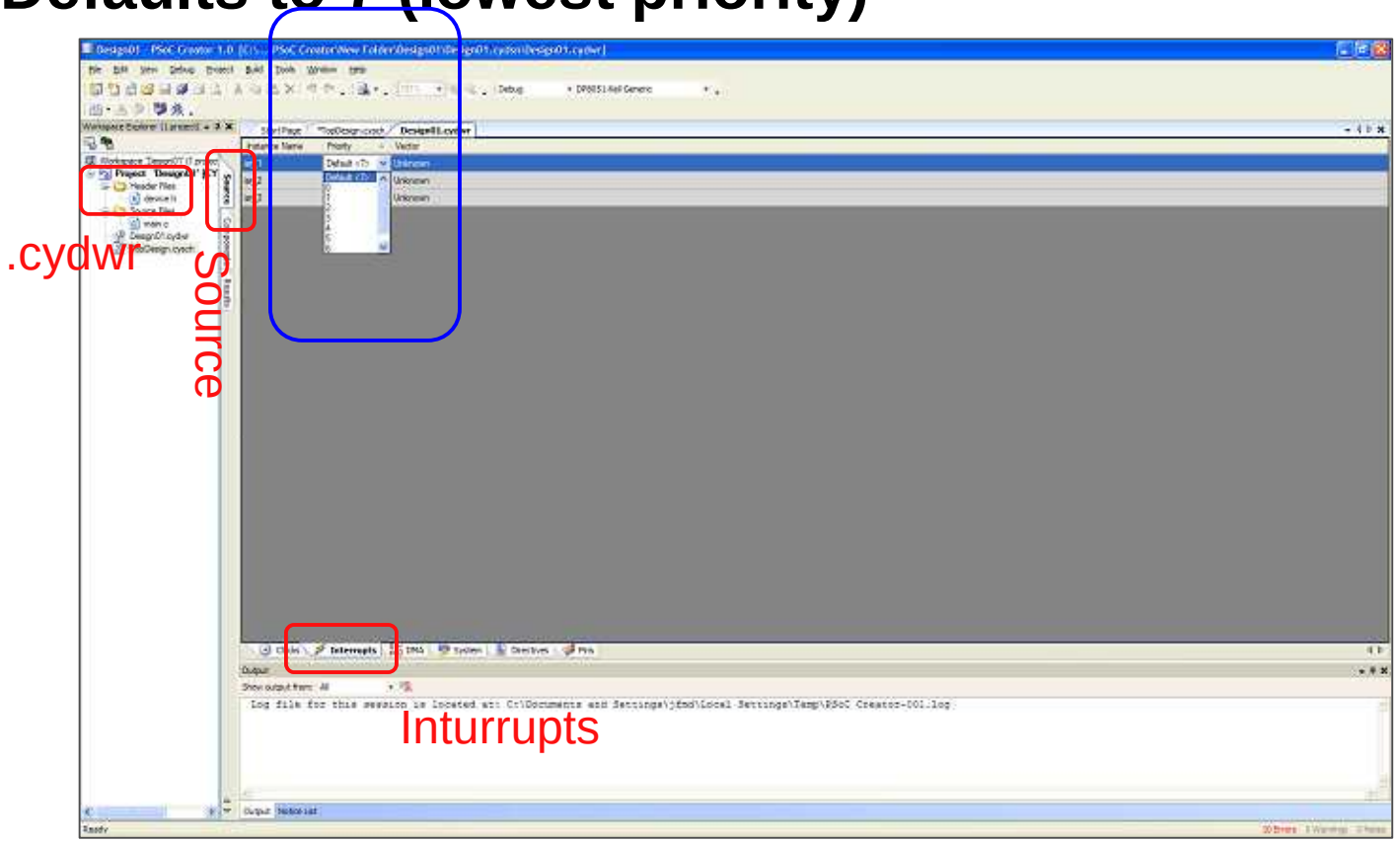


解説

ハードウェア割込みの設定

Priority may be changed

Defaults to 7 (lowest priority)



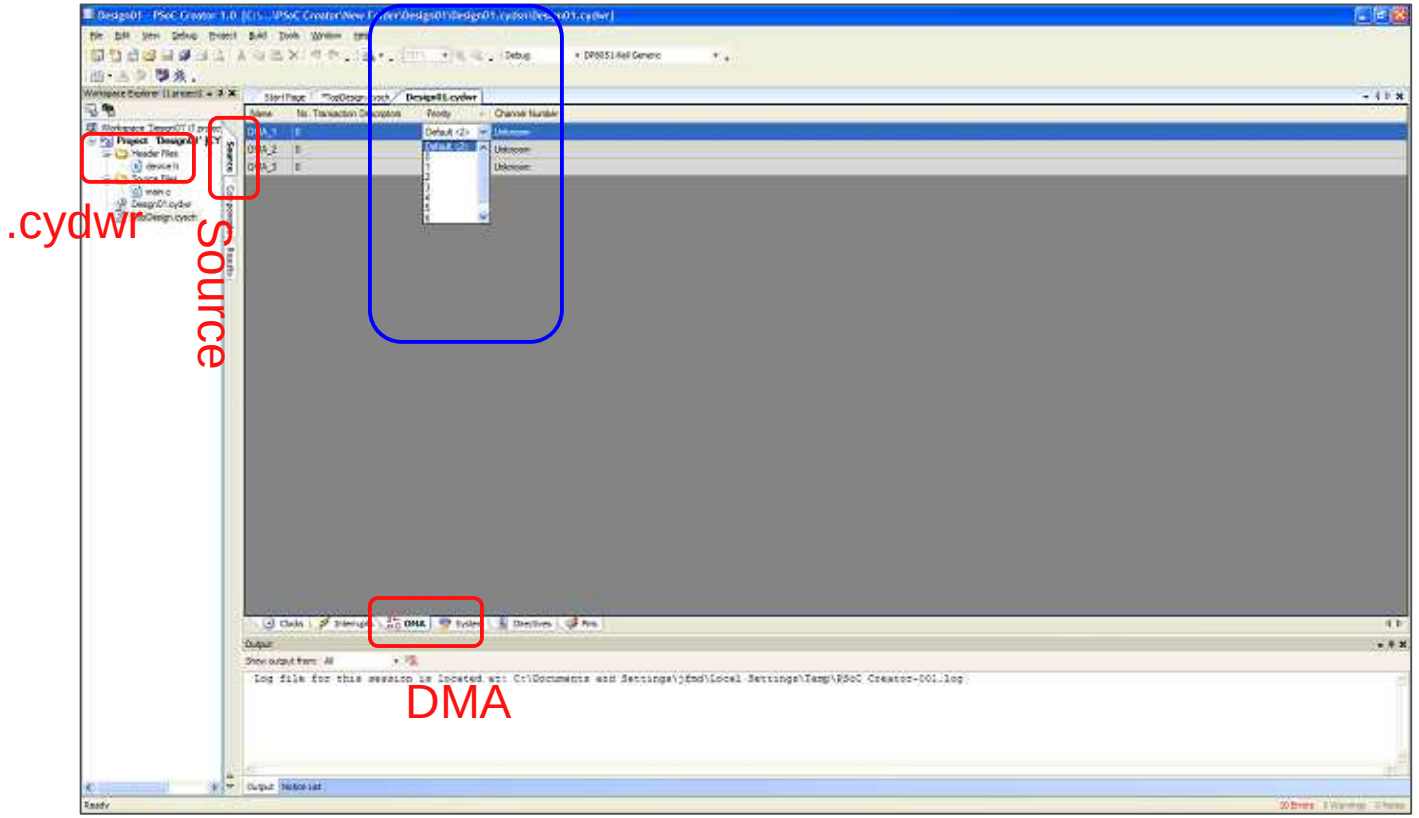
解説

DMA-プライオリティー設定

Priority may be changed

Defaults to 2 (0 & 1 can consume 100% of bandwidth)

解説



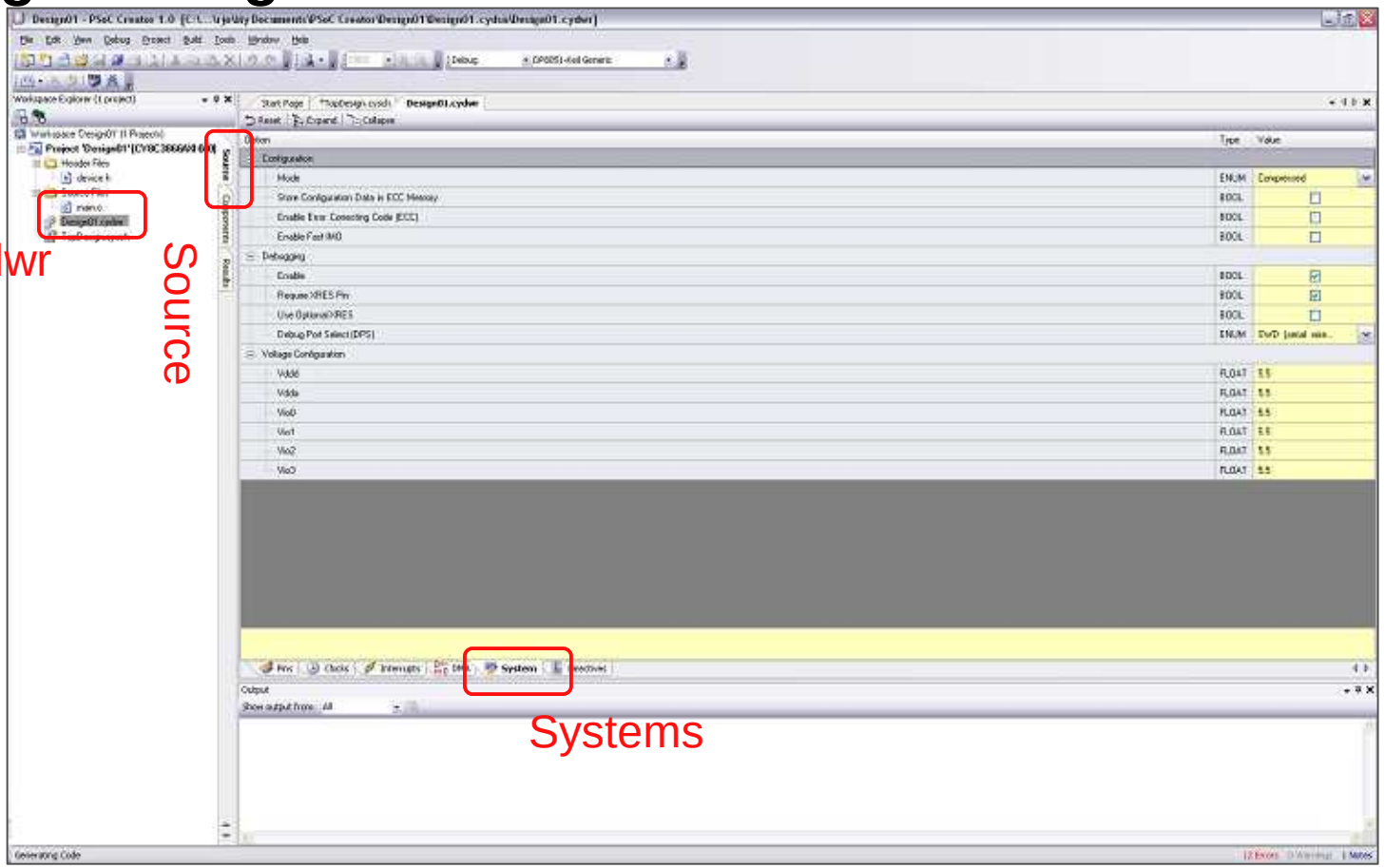
システム設定

System settings

Debug settings

Voltage Configuration

解説



クロックのコンフィギュレーション

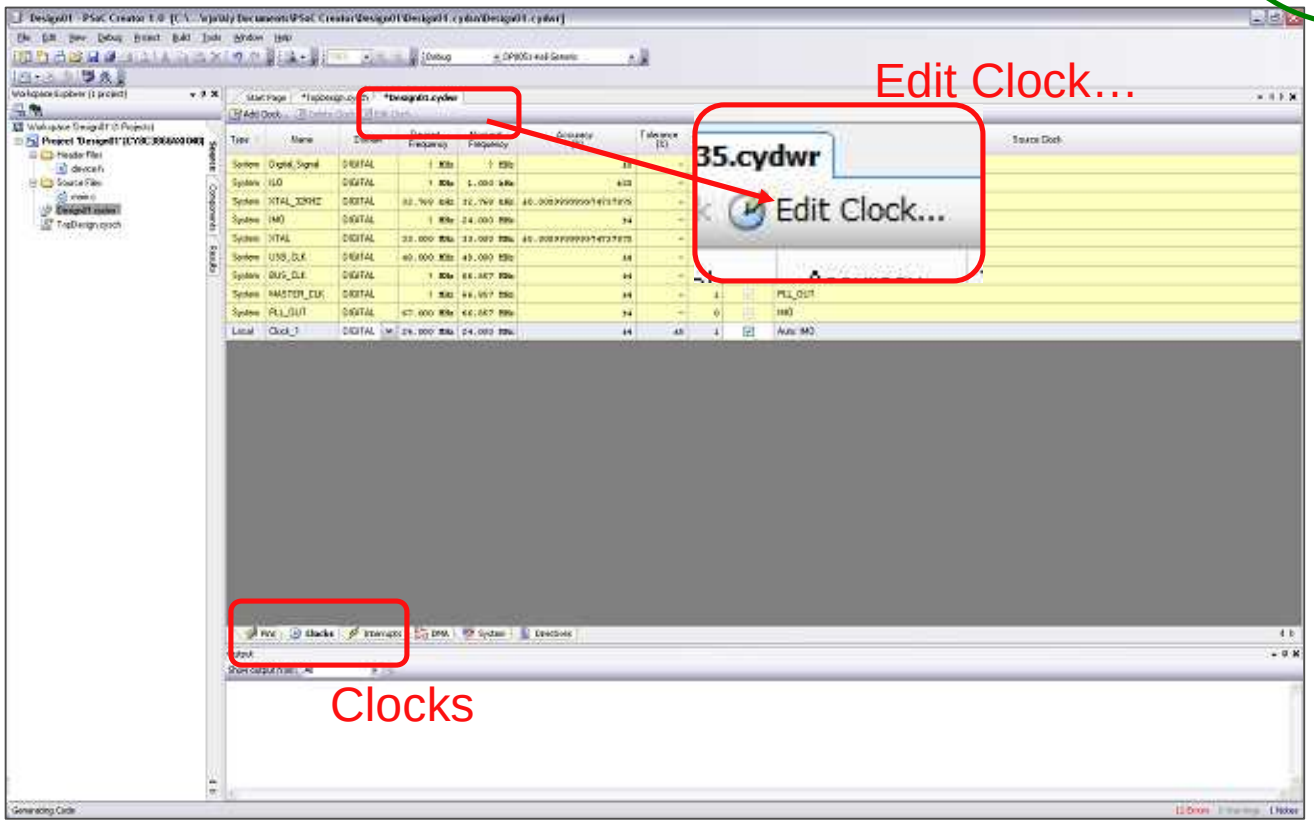
Clocks are allocated to slots in the clock tree

- 8 digital, 4 analog

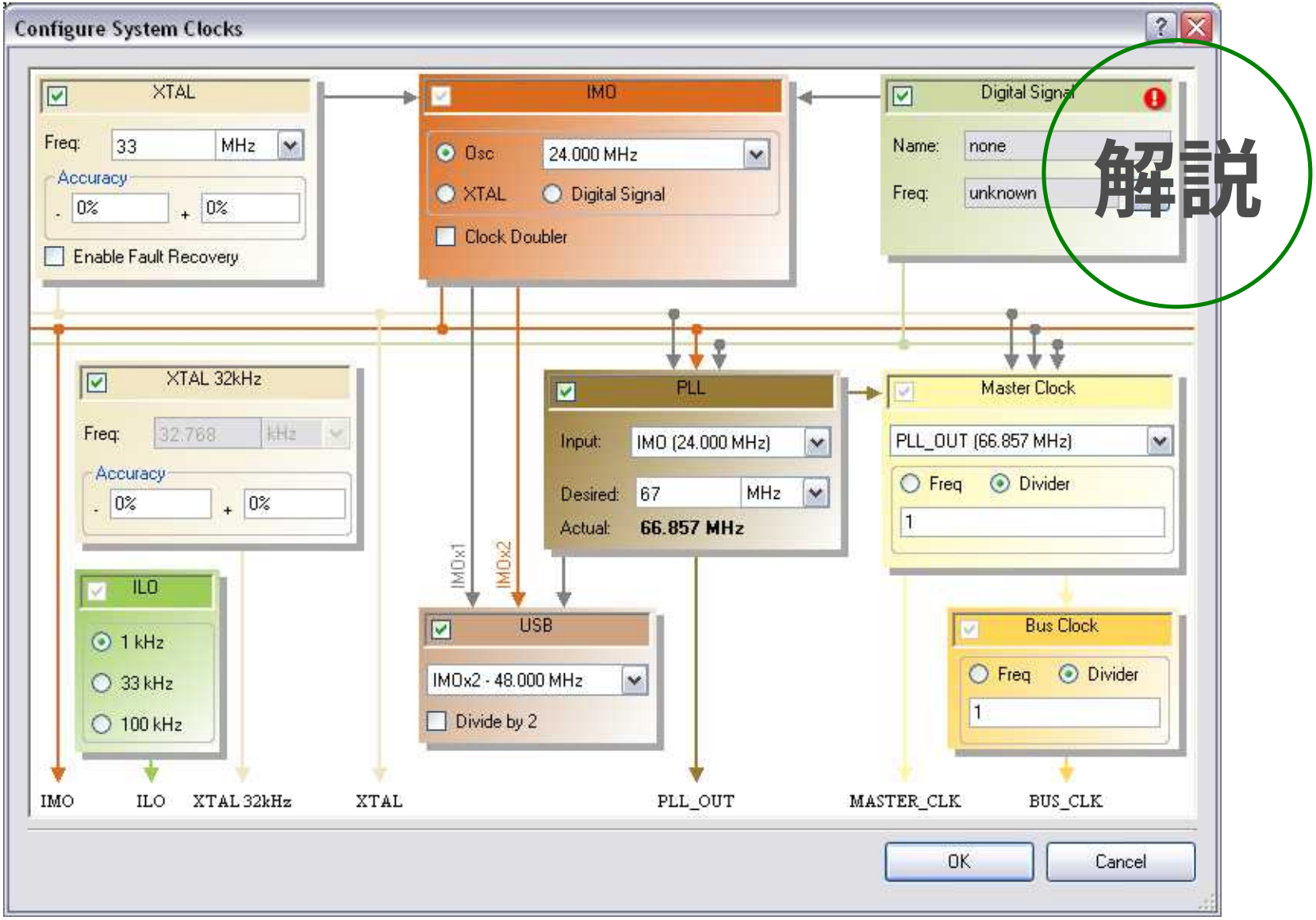
Clocks have software APIs

Reuse existing clocks to preserve resources

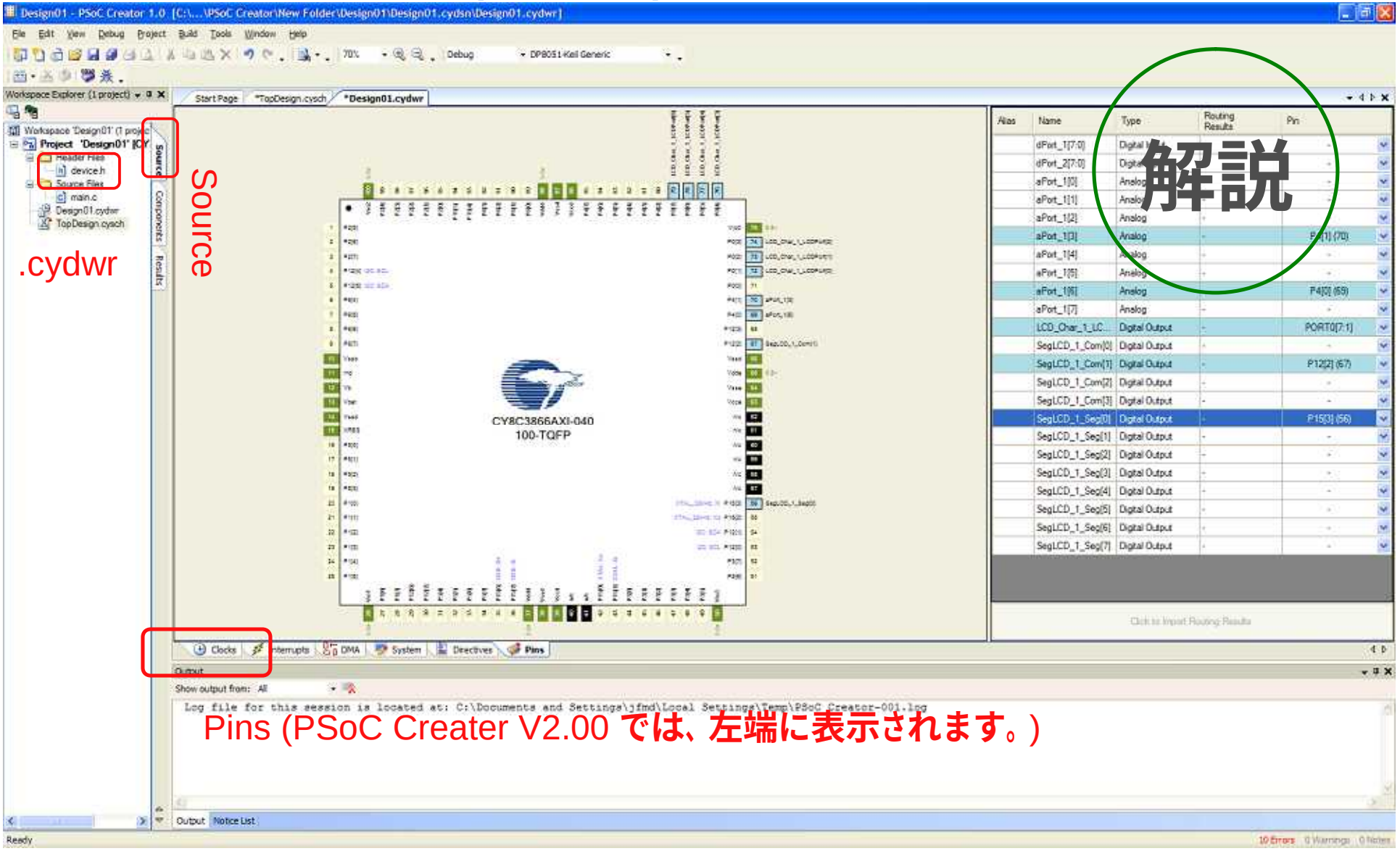
解説



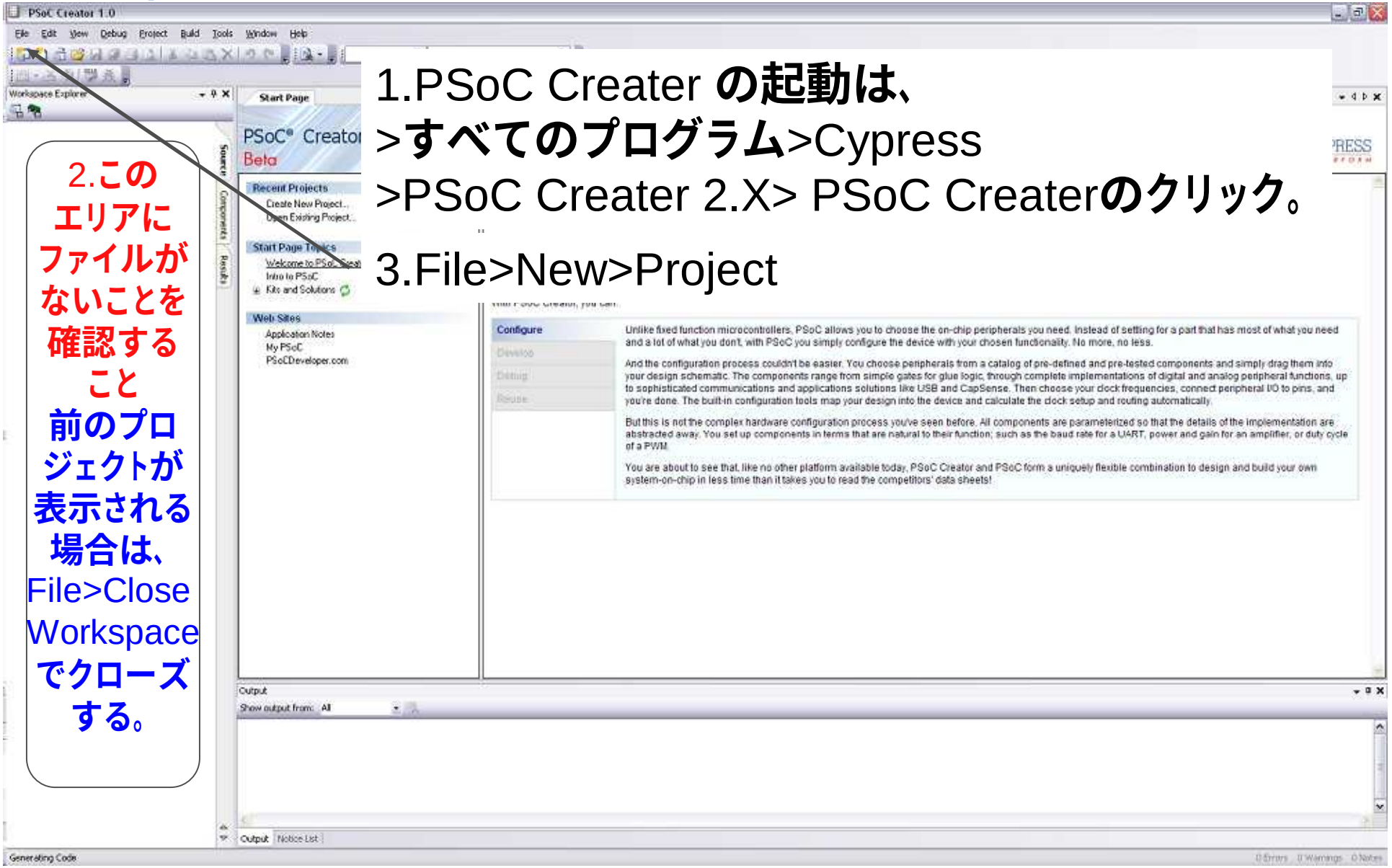
システムクロック Tree 分配設定



ピン・エディター(.を開く)



Step1.PSoC Creator Softwareの起動

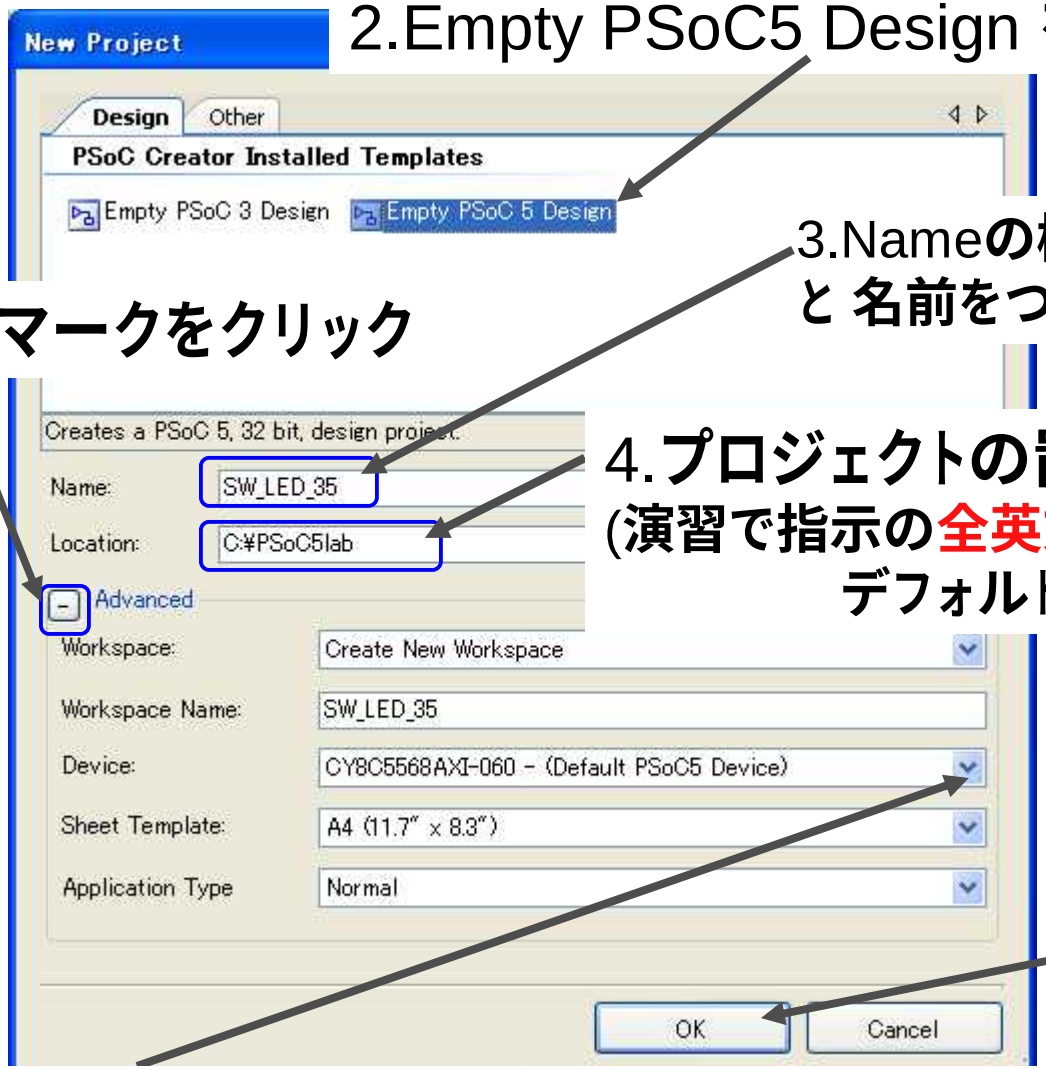


1.PSoC Creater **の起動は、**
>すべてのプログラム>Cypress
>PSoC Creater 2.X> PSoC Creater**のクリック。**

3.File>New>Project

2.**このエリアにファイルがないことを確認すること**
前のプロジェクトが表示される場合は、File>Close Workspace **でクローズする。**

Step 2:新規プロジェクトの作成2



2.Empty PSoC5 Design を選択(ハイライト化)

1.+ マークをクリック

3.Nameの欄に,PWM_LED_B_35
と 名前をつける(名前は任意)

4.プロジェクトの置き場所を指定
(演習で指示の全英文字パスディレクトリ
デフォルトは、C:¥PSoC5_Lab)

6.OKをクリック

5.デバイス指定または変更の場合は、ここから選択(次ページ)

Device Selector　デバイスの選択や変更

このアイコンをクリックすると、
表示するフィルタ項目を設定できます

フィルタ項目を選択することで、
必要な機能を搭載した
PSoC を抽出できます

Select ... evice (Other architectures available after project creation)

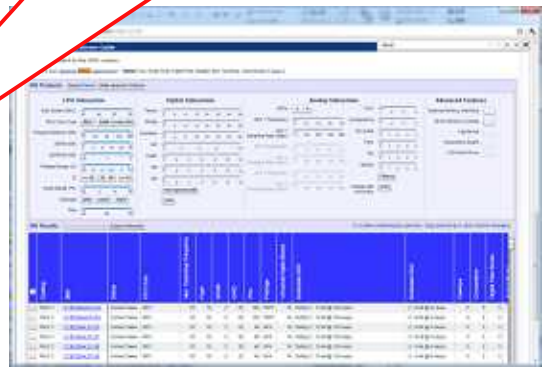
Hide/Show Columns... Reset to Defaults 18 Colum

	Architecture	CPU Speed (MHz)	Flash (KB)	SRAM (KB)	EEPROM (bytes)	Trace Buffer (KB)	DMA Channels	PLL	LCD Drive (mux ratio)	CapSense	ADC	8-bit DAC	SC/CT Blocks	...
Filters:														
CY8C5468AXI-018	PSoC5 (ARM CM3)	67	256	64	2048	-	24	1	x16	✓	2x 12-bit SAR	4	4	1
CY8C5468LTI-037	PSoC5 (ARM CM3)	67	256	64	2048	-	24	1	x16	✓	2x 12-bit SAR	4	4	1
CY8C5566AXI-061	PSoC5 (ARM CM3)	67	64	16	2048	-	24	1	x16	✓	1x 12-bit SAR 1x 20-bit Delta Sigma	4	4	1
CY8C5566LTI-017	PSoC5 (ARM CM3)	67	64	16	2048	-	24	1	x16	✓	1x 12-bit SAR 1x 20-bit Delta Sigma	4	4	1
CY8C5567AXI-019	PSoC5 (ARM CM3)	67	128	32	2048	-	24	1	x16	✓	1x 12-bit SAR 1x 20-bit Delta Sigma	4	4	1
CY8C5567LTI-079	PSoC5 (ARM CM3)	67	128	32	2048	-	24	1	x16	✓	1x 12-bit SAR 1x 20-bit Delta Sigma	4	4	1
CY8C5568AXI-060	PSoC5 (ARM CM3)	67	256	64	2048	-	24	1	x16	✓	2x 12-bit SAR 1x 20-bit Delta Sigma	4	4	1
CY8C5568LTI-114	PSoC5 (ARM CM3)	67	256	64	2048	-	24	1	x16	✓	2x 12-bit SAR 1x 20-bit Delta Sigma	4	4	1
CY8C5588AXI-060ES1	PSoC5 (ARM CM3)	80	256	64	2048	-	24	1	x16	✓	2x 12-bit SAR 1x 20-bit Delta Sigma	4	4	1
CY8C5588LTI-114ES1	PSoC5 (ARM CM3)	80	256	64	2048	-	24	1	x16	✓	2x 12-bit SAR 1x 20-bit Delta Sigma	4	4	1

28 of 28 devices found　Clear Filters

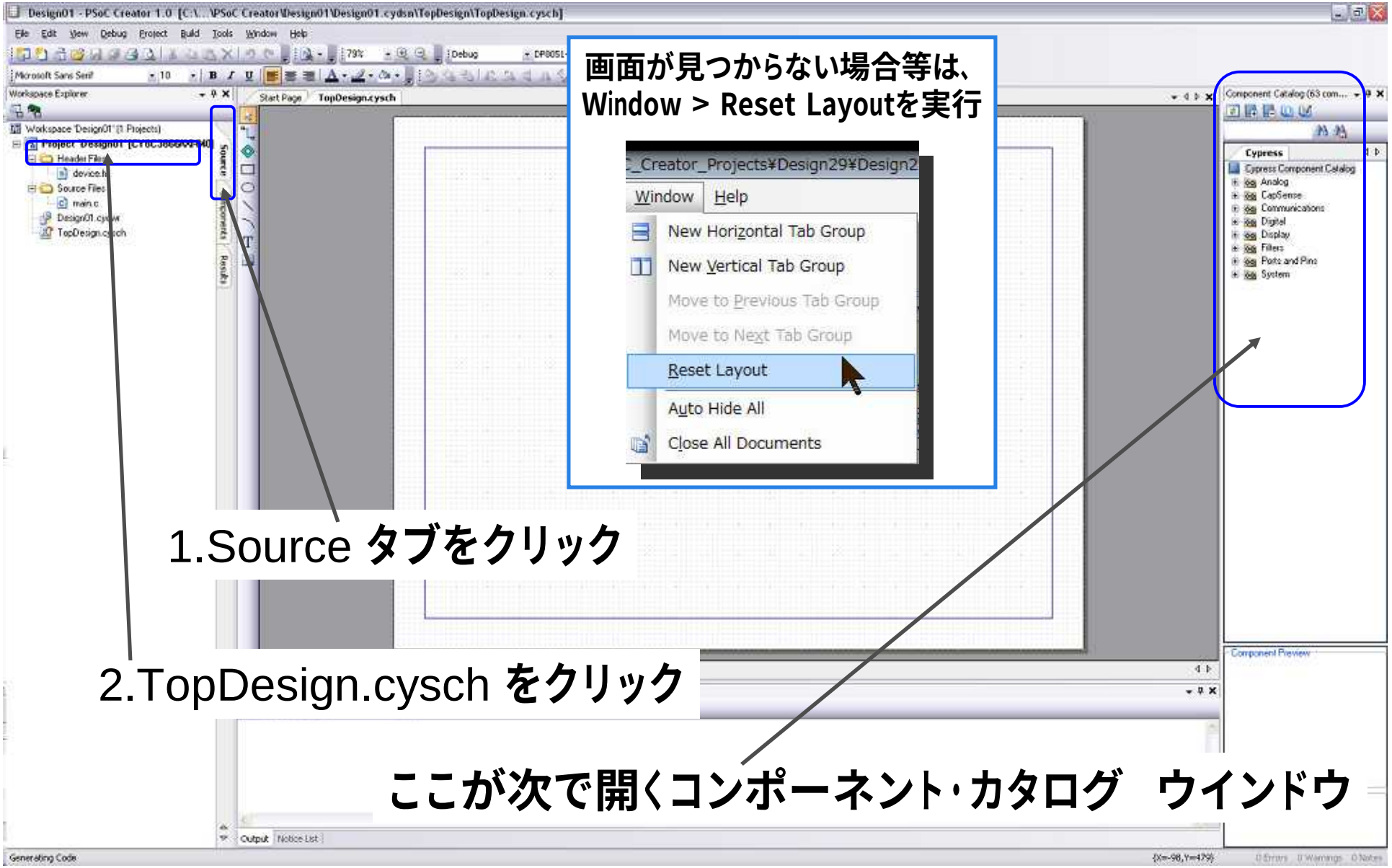
OK　Cancel

CY8C-KIT-050 **実装デバイス**は、
CY8C5588-AXI-060です。
デバイスリビジョン はここで選択できます。
ここから適切ものを選択して下さい
2011年Q4 時点では、**ES1**です。
使用する**コンポーネント**には、**バージョン**があり、**デバイスリビジョンに対応するバージョン**を選択してください。



オンライン版もあります
http://www.cypress.com/?id=2232

Step3.Design Canvas(ブロック接続図.cysch)を開く



コンポーネント・カタログ

Catalog Folders

Analog
- ADC
- Amplifier
- DAC

Digital
- Registers
- Functions
- Logic

Communication

Display

System

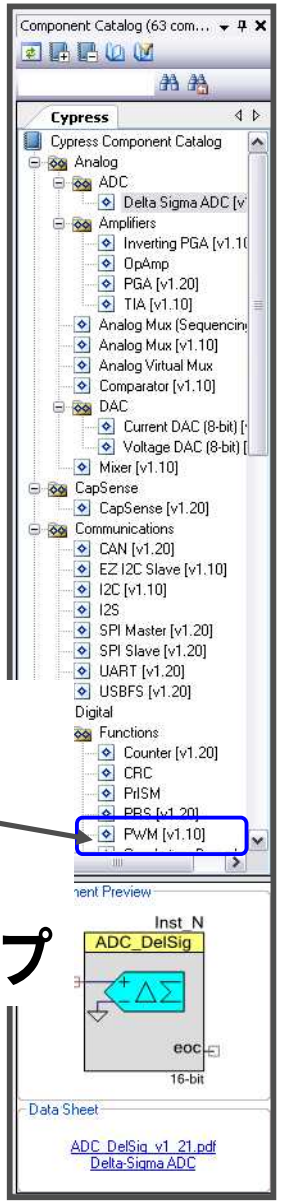
Catalog Preview

Datasheet access

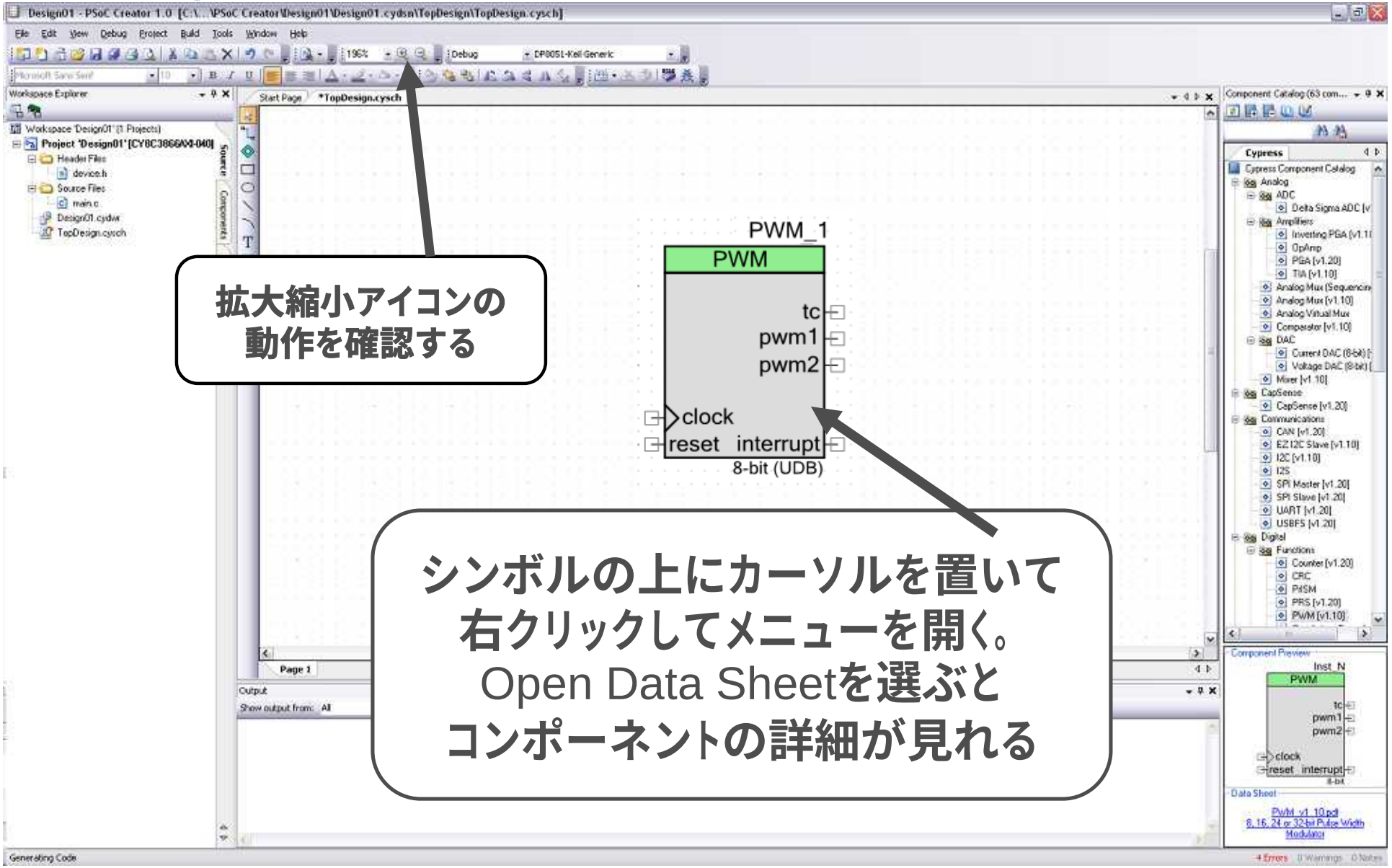
各種のハードウェアの
コンポーネントが
ライブラリ化されています

必要なものを結線して
ブロック図を作成します

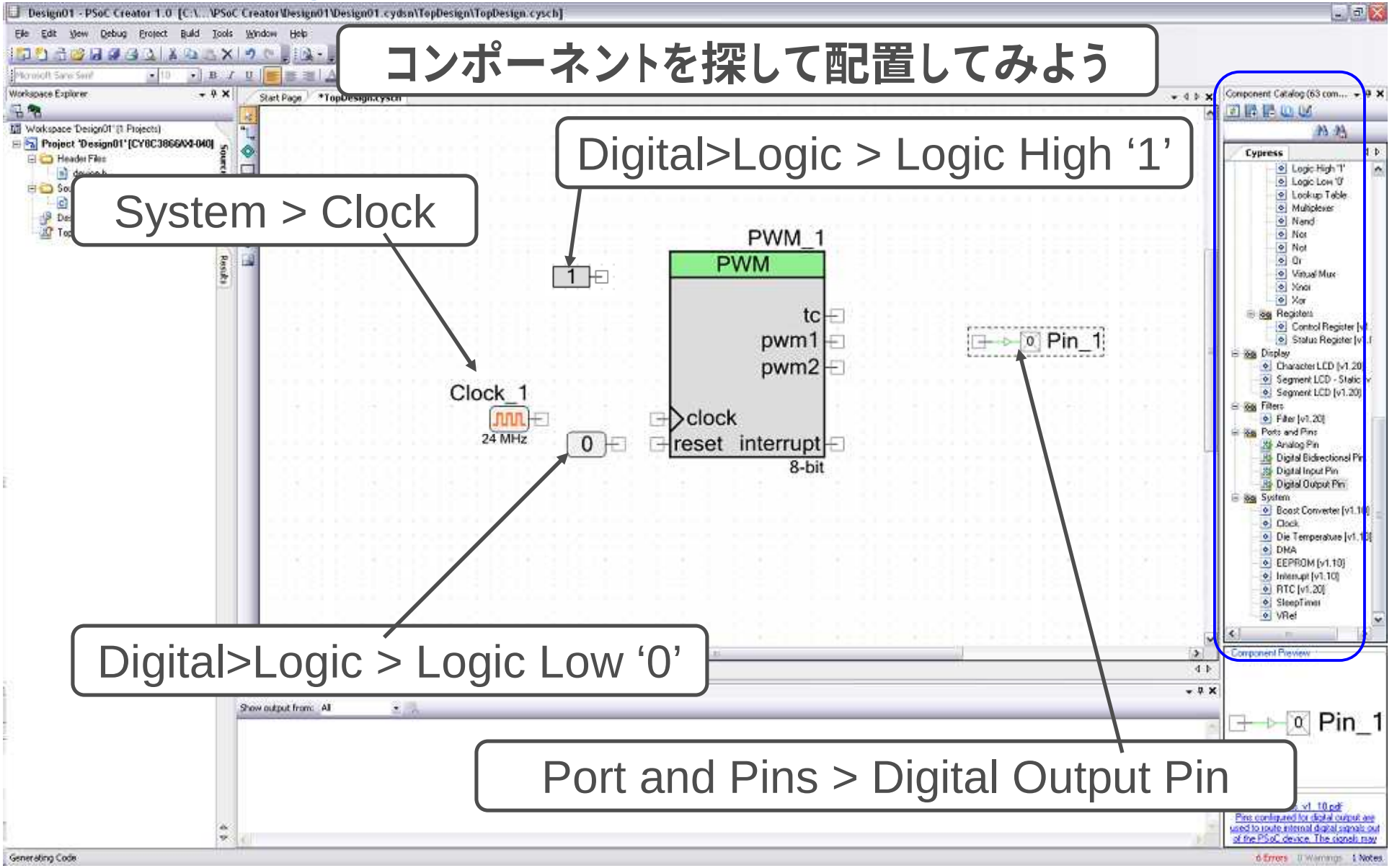
1.Digital
>Functions
>PWMをドラッグして
回路(ブロック)図上にドロップ



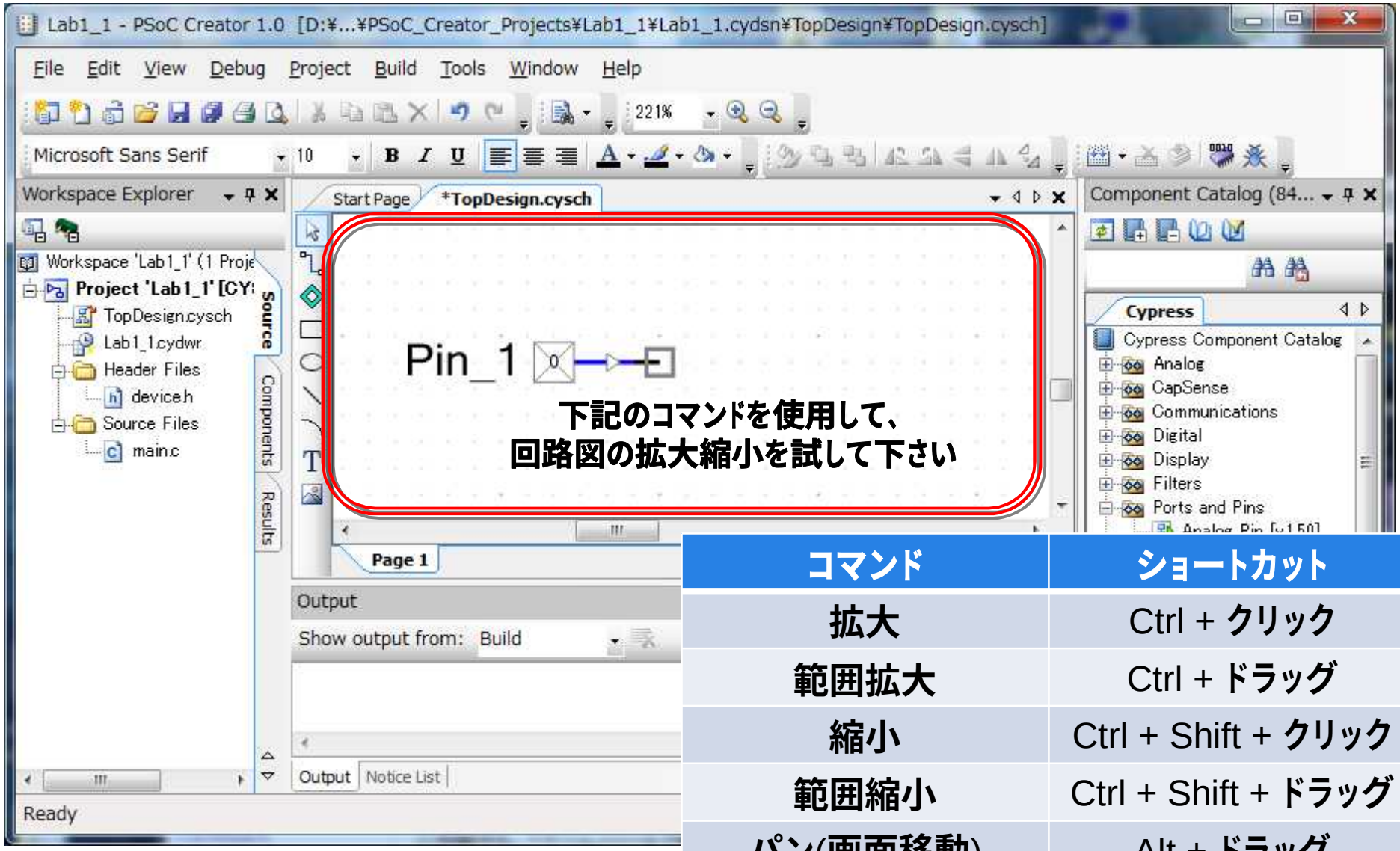
Step4.コンポーネントの追加 (PWM)



Pins, Logic and Clock コンポーネントの配置



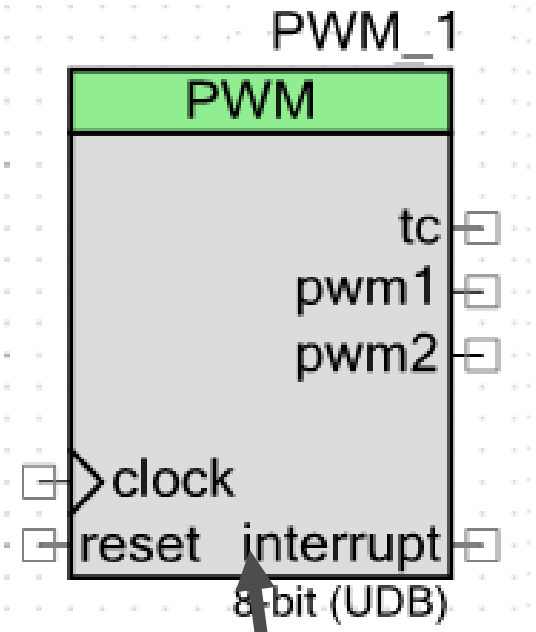
追補：回路図エディタの操作

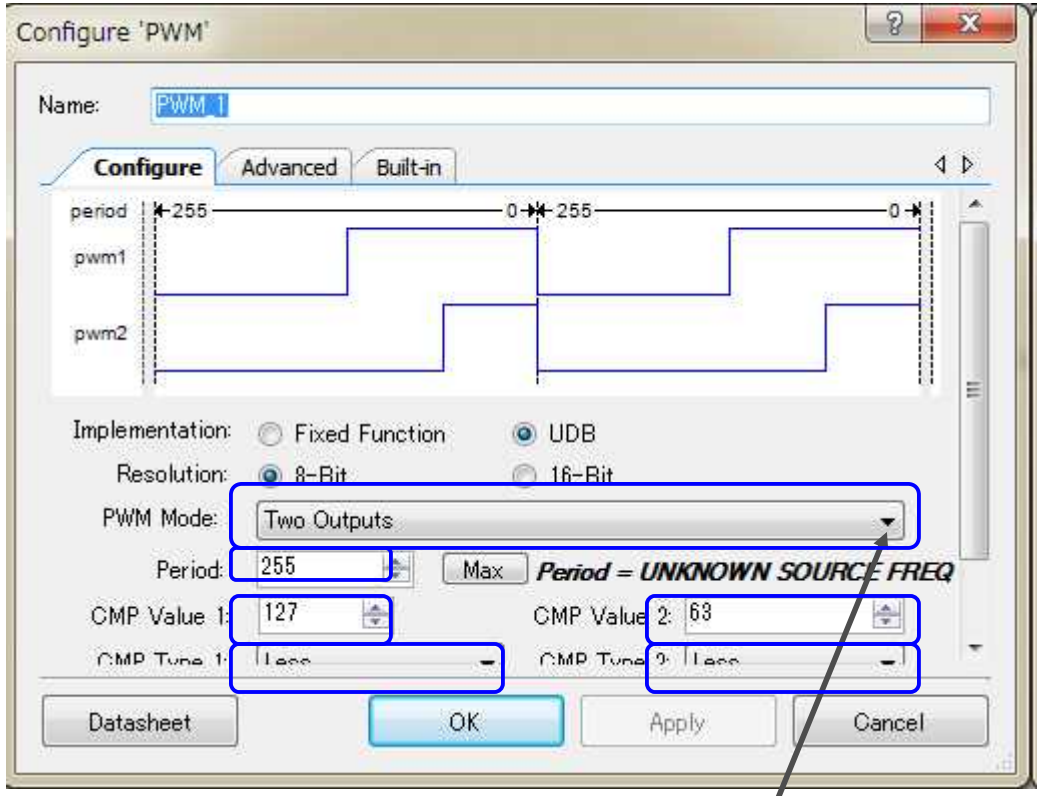


コマンド	ショートカット
拡大	Ctrl + クリック
範囲拡大	Ctrl + ドラッグ
縮小	Ctrl + Shift + クリック
範囲縮小	Ctrl + Shift + ドラッグ
パン(画面移動)	Alt + ドラッグ

Step5.コンポーネントのコンフィギュレーション

コンポーネントコンフィギュレーションダイアログを開く

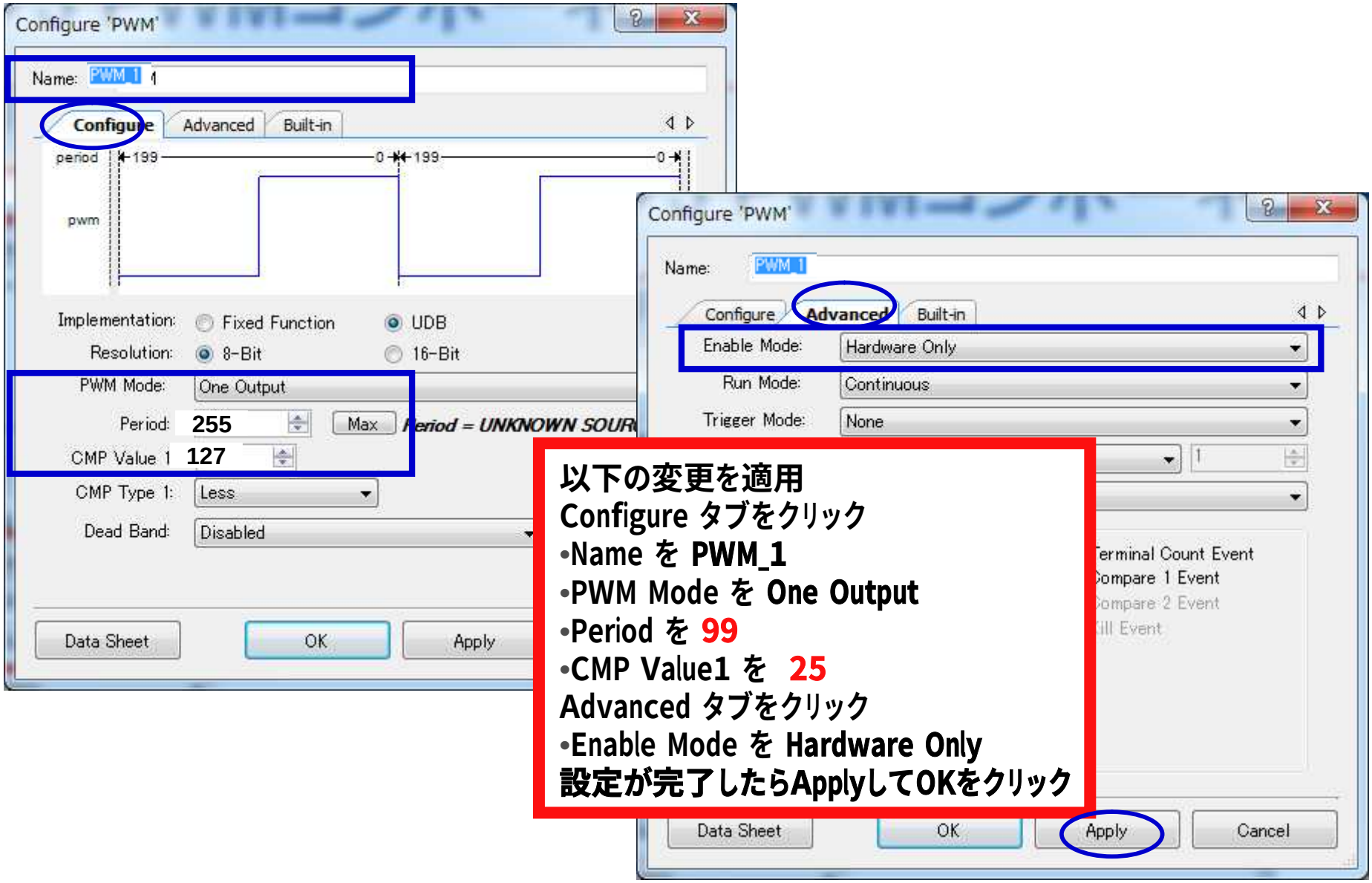




1.シンボルの上にカーソルを置いて
右クリックしてConfigureをクリック

2.One Outputに変更
表示波形が変わる

PWMコンポーネントの詳細設定

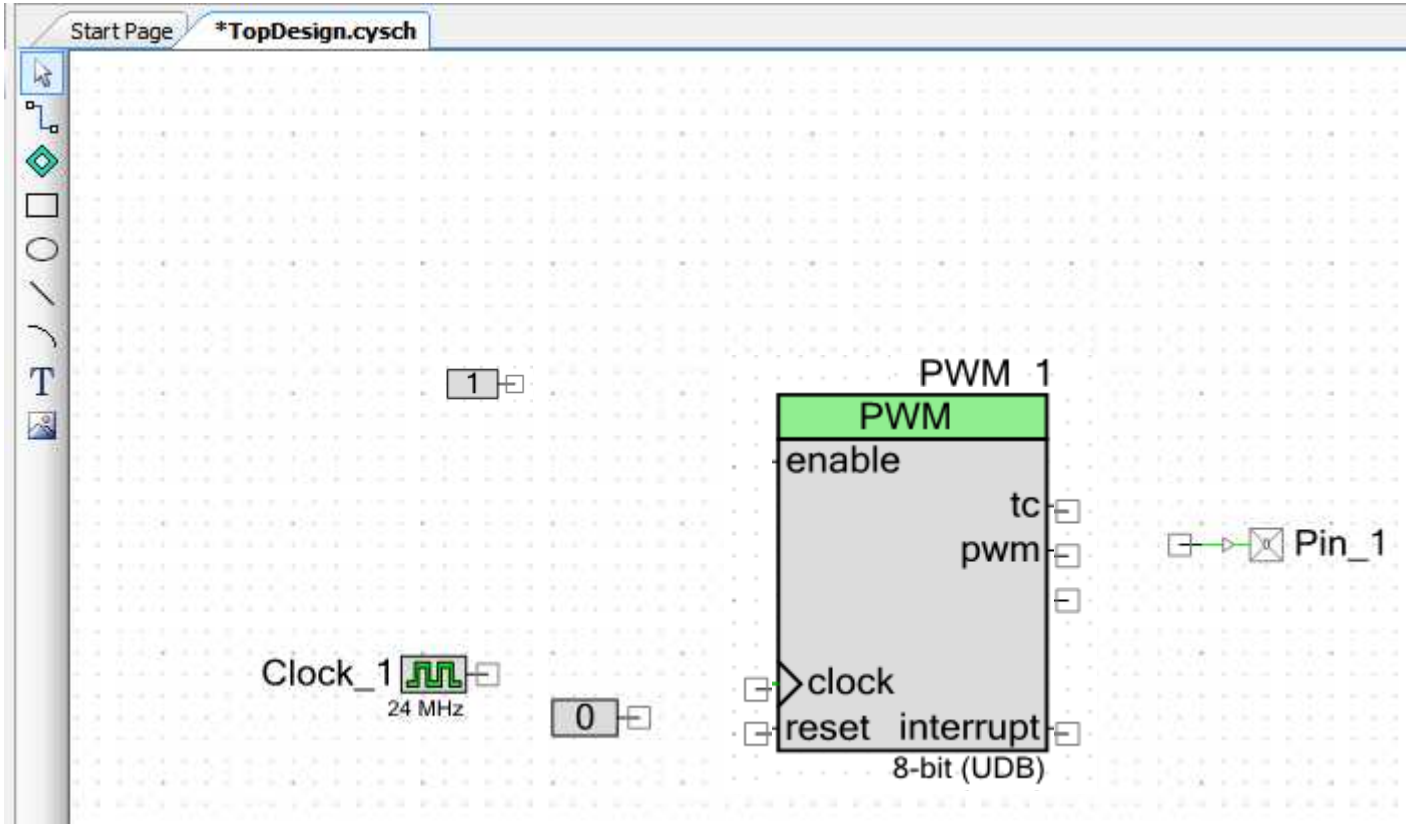


以下の変更を適用
Configure タブをクリック
- Name を PWM_1
- PWM Mode を One Output
- Period を 99
- CMP Value1 を 25

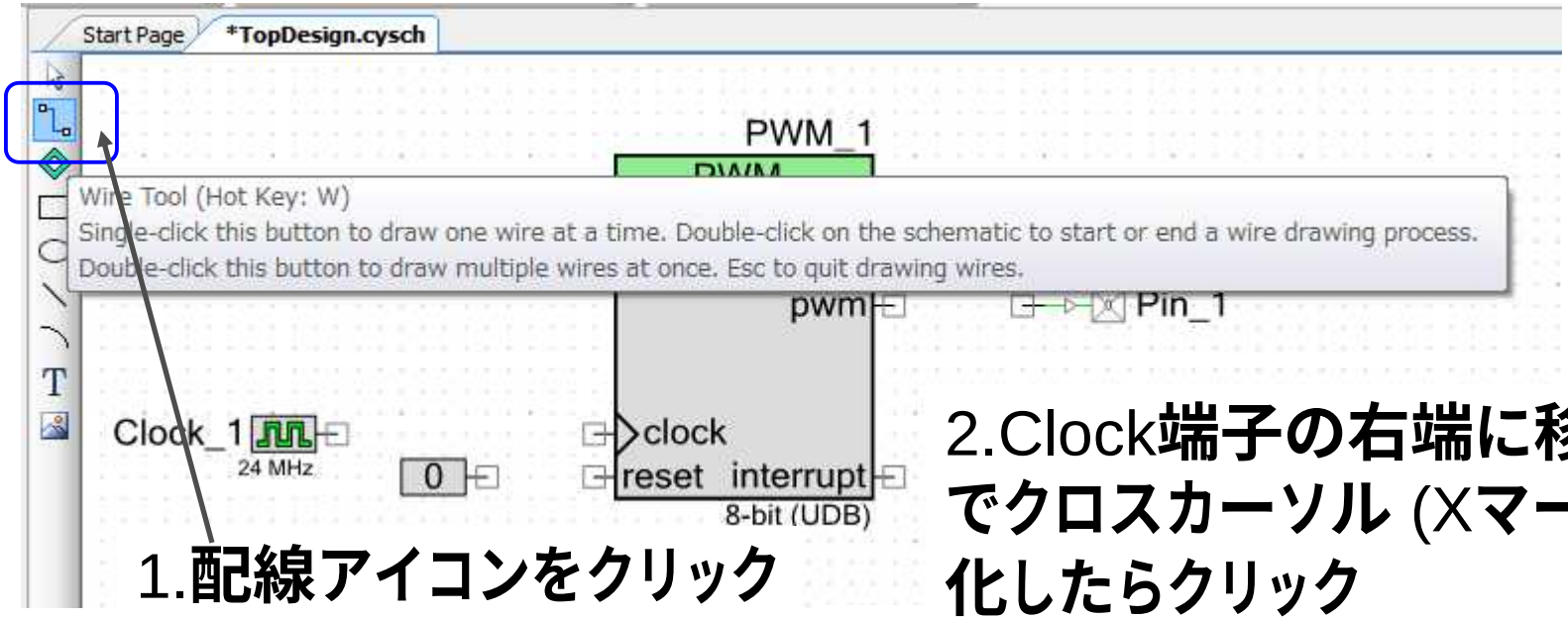
Advanced タブをクリック
- Enable Mode を Hardware Only

設定が完了したらApplyしてOKをクリック

.cysch を開いてコンポーネントを表示



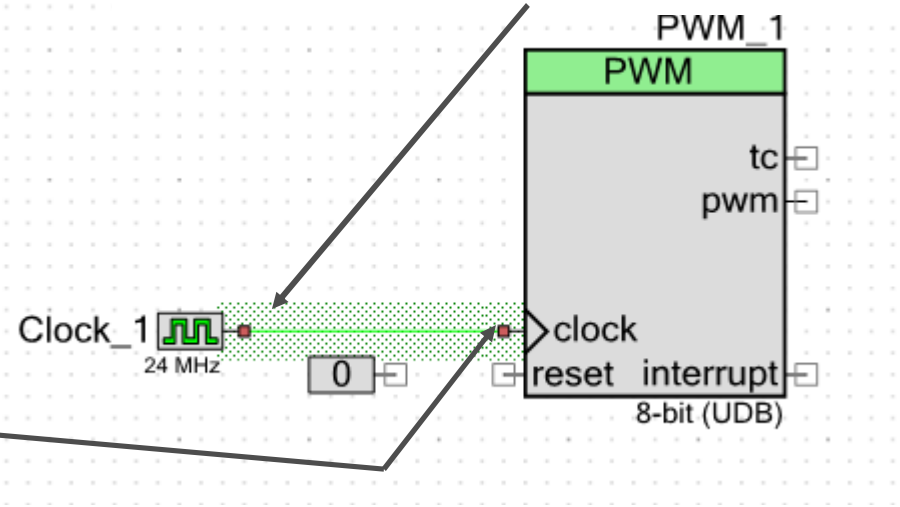
Step6.配線アイコンを選択し配線する



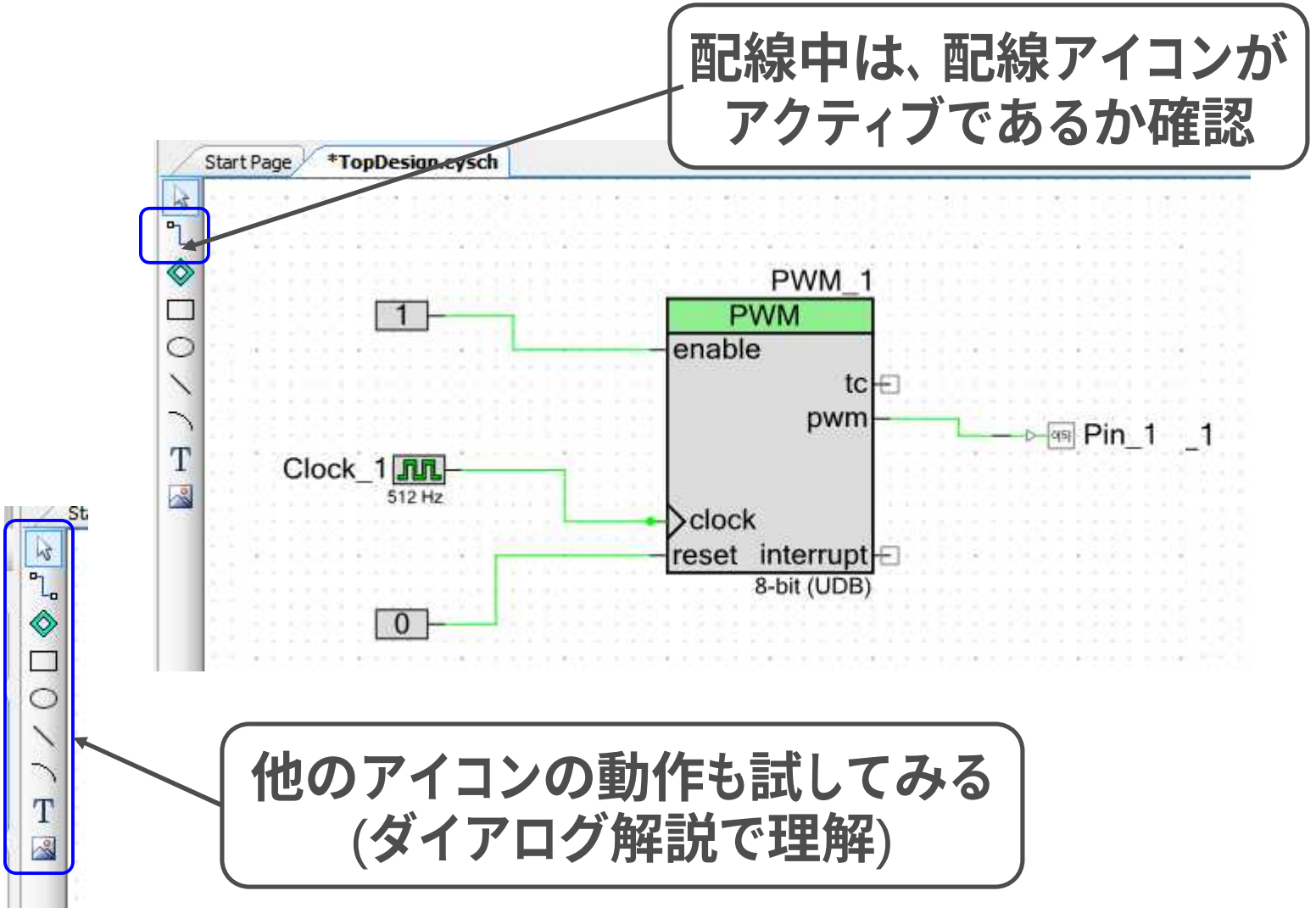
1.配線アイコンをクリック

2.Clock端子の右端に移動してクロスカーソル (Xマーク)化したらクリック

3.PWMのclock端子の左端まで配線し、
クロスカーソル化したら
クリックして配線を確定

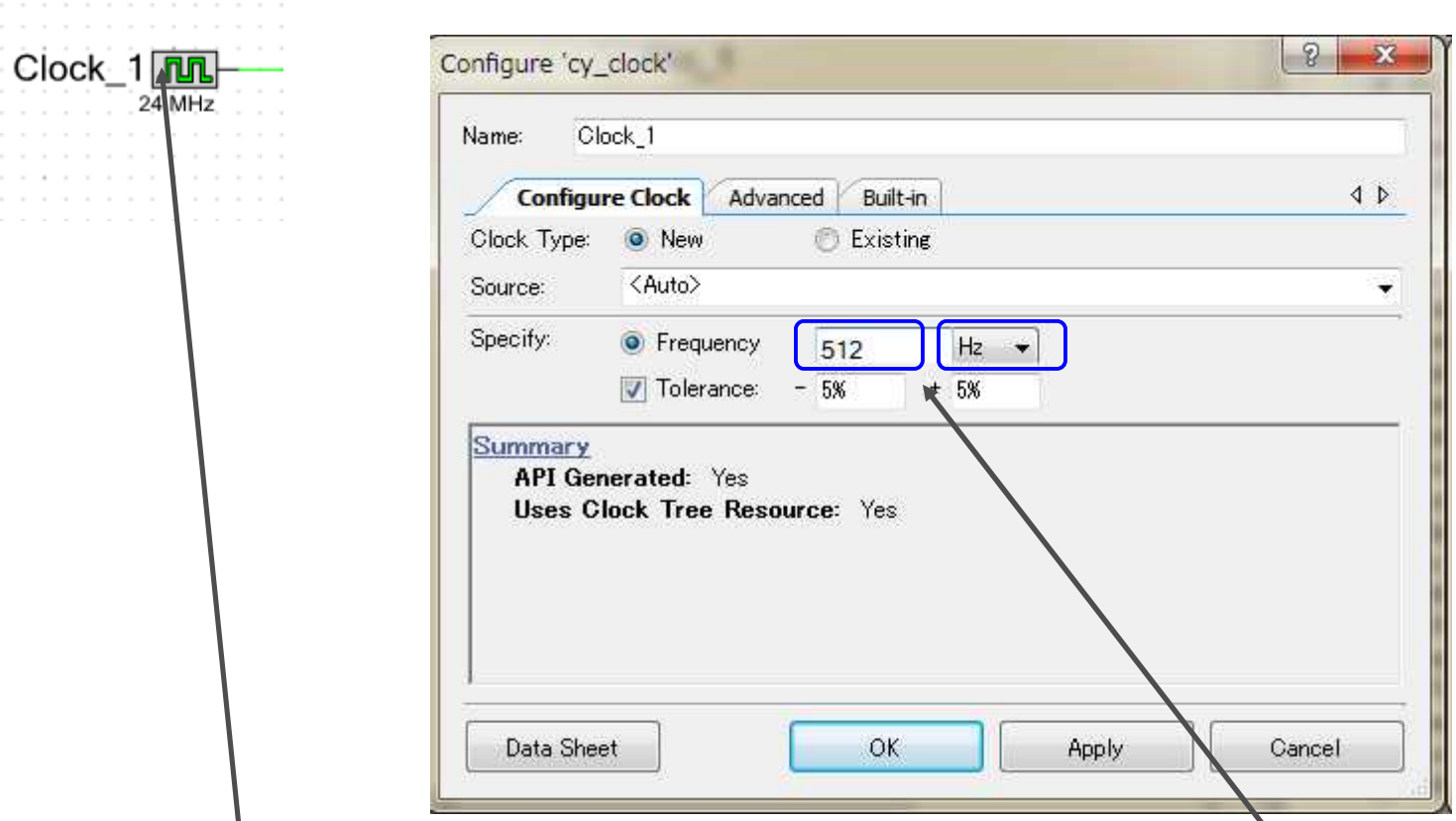


コンポーネント間配線の完了



Step7.追加コンポーネントのコンフィギュレーション

コンポーネントのシンボルをダブルクリックする

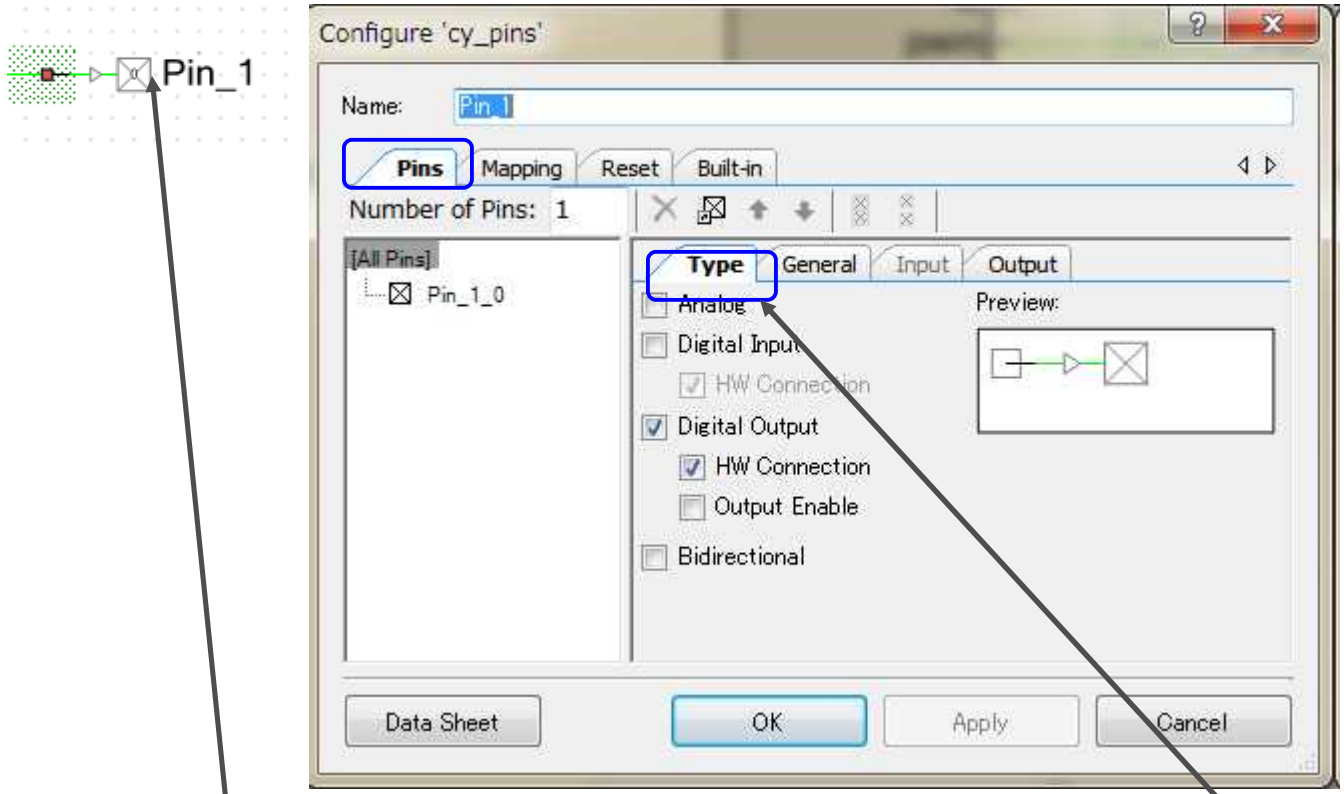


1.シンボルの上にカーソルを置いて
右クリックしてConfigureをクリック

2.周波数を100KHzに変更

各コンポーネントのコンフィギュレーション

コンポーネントのシンボルをダブルクリックする

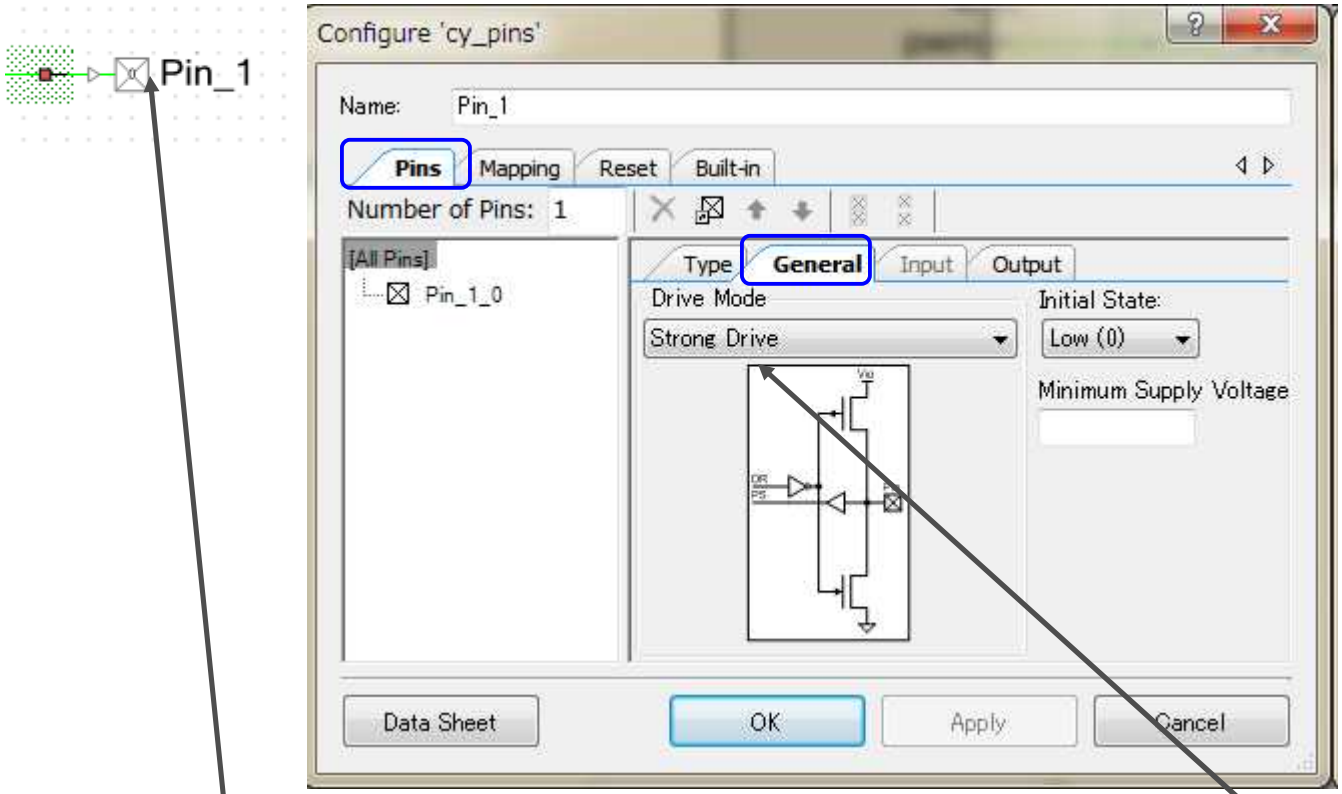


1.シンボルの上にカーソルを置いて
右クリックしてConfigureをクリック

2.Typeタブ内の設定

各コンポーネントのコンフィギュレーション

コンポーネントのシンボルをダブルクリックする

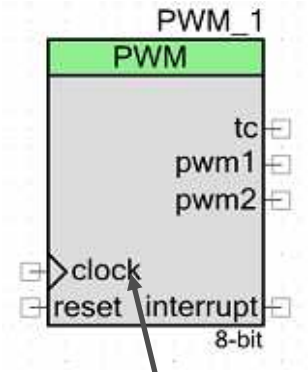


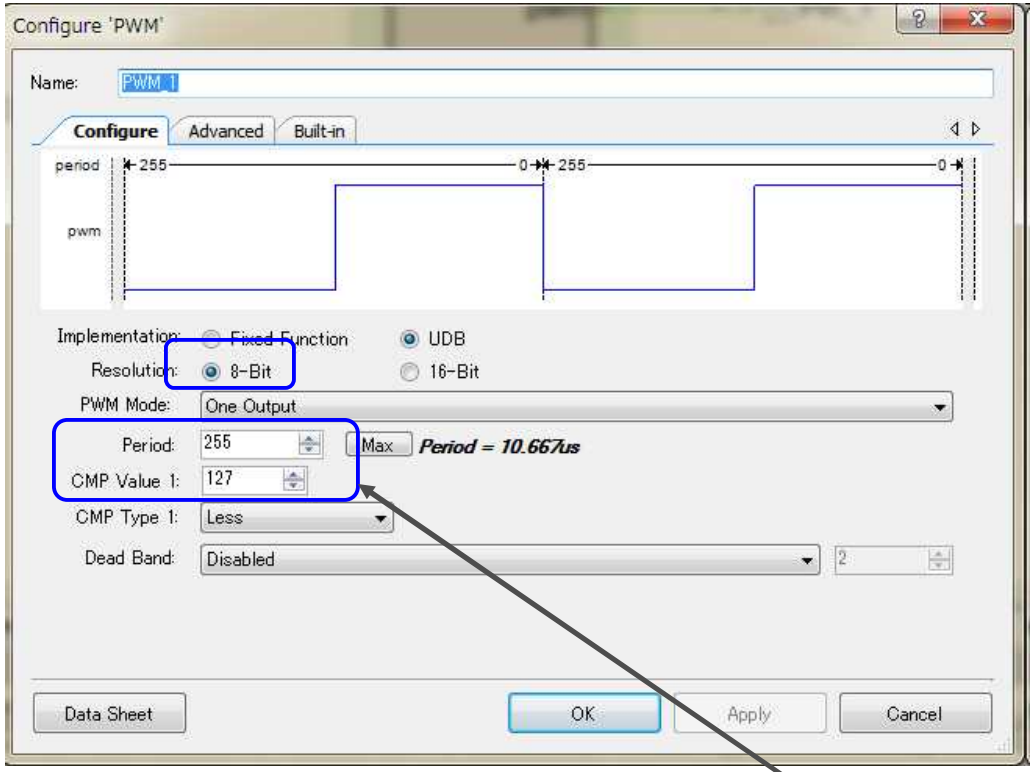
1.シンボルの上にカーソルを置いて
右クリックしてConfigureをクリック

2.Generalタブ内の設定
Strong Drive を確認

各コンポーネントのコンフィギュレーション

コンポーネントのシンボルをダブルクリックする

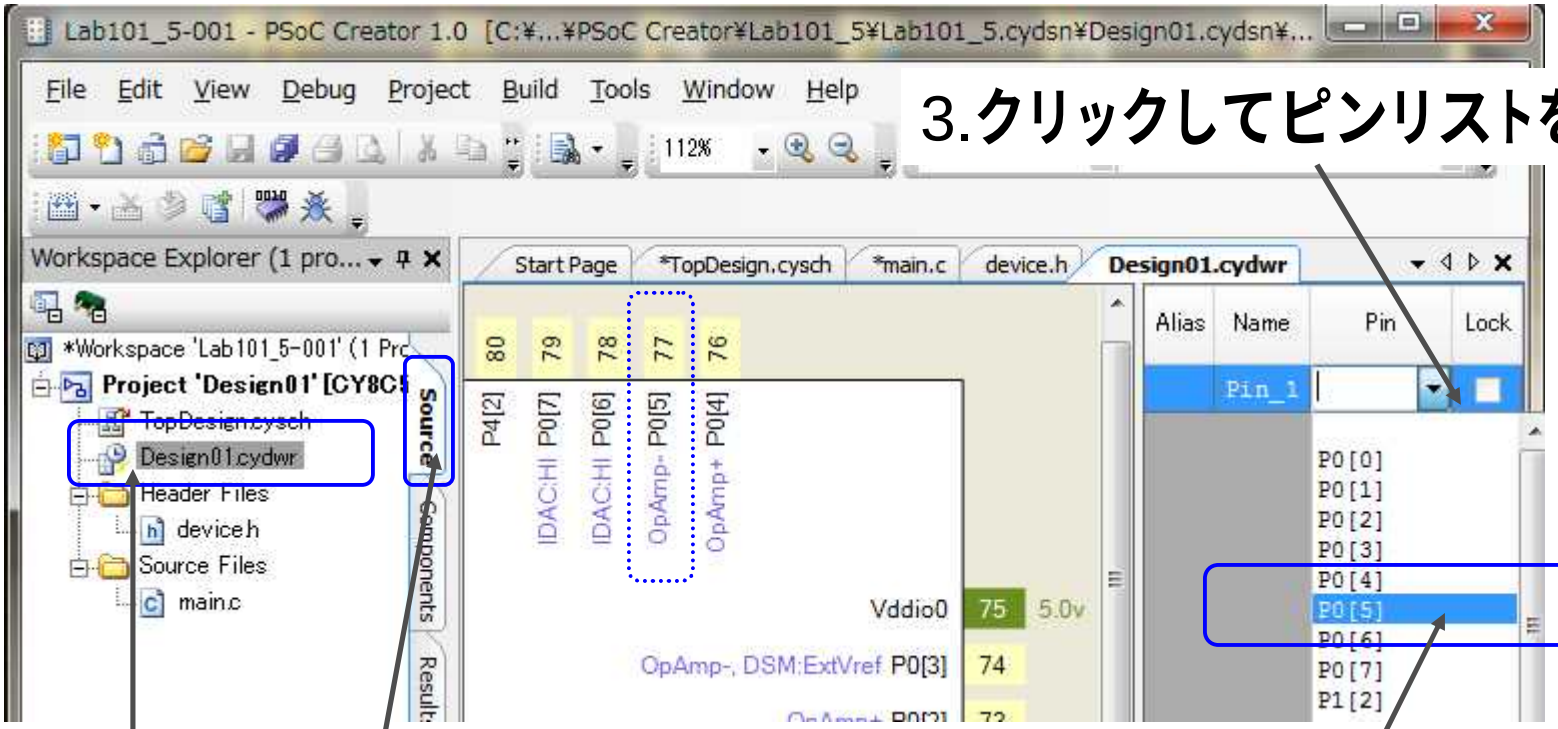




1.シンボルの上にカーソルを置いて
右クリックしてConfigureをクリック

2.Period/CMPを確認

Step8.デバイスピンのアサイン



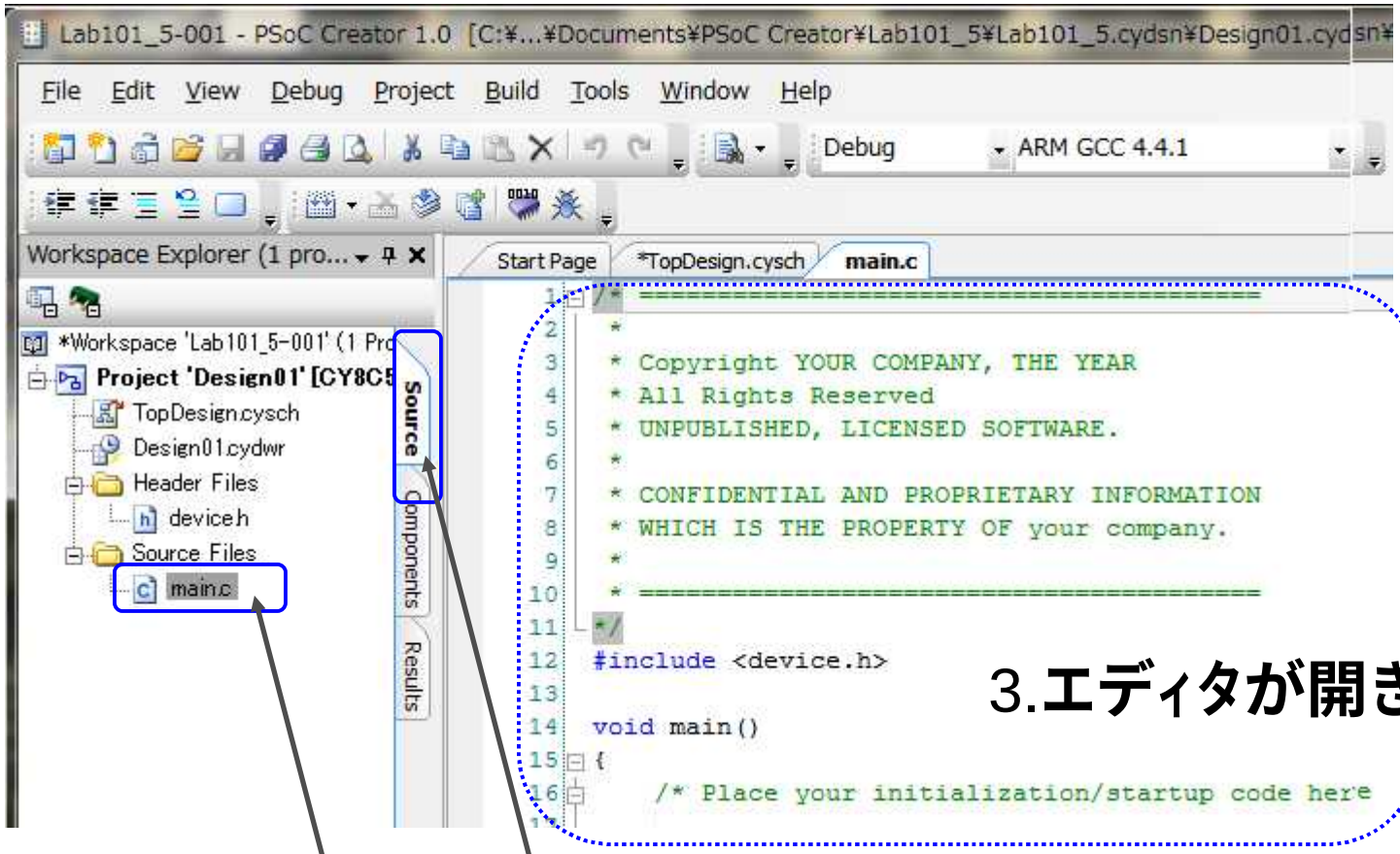
3.クリックしてピンリストを開く

1.Sourceタブをクリック

2..cydwrをダブルクリック

4.P0[5]をクリック

ソースコードエディタを開く



3.エディタが開きます

1.Sourceタブをクリック

2.main.cをダブルクリック

Step9.ソースコードの記述

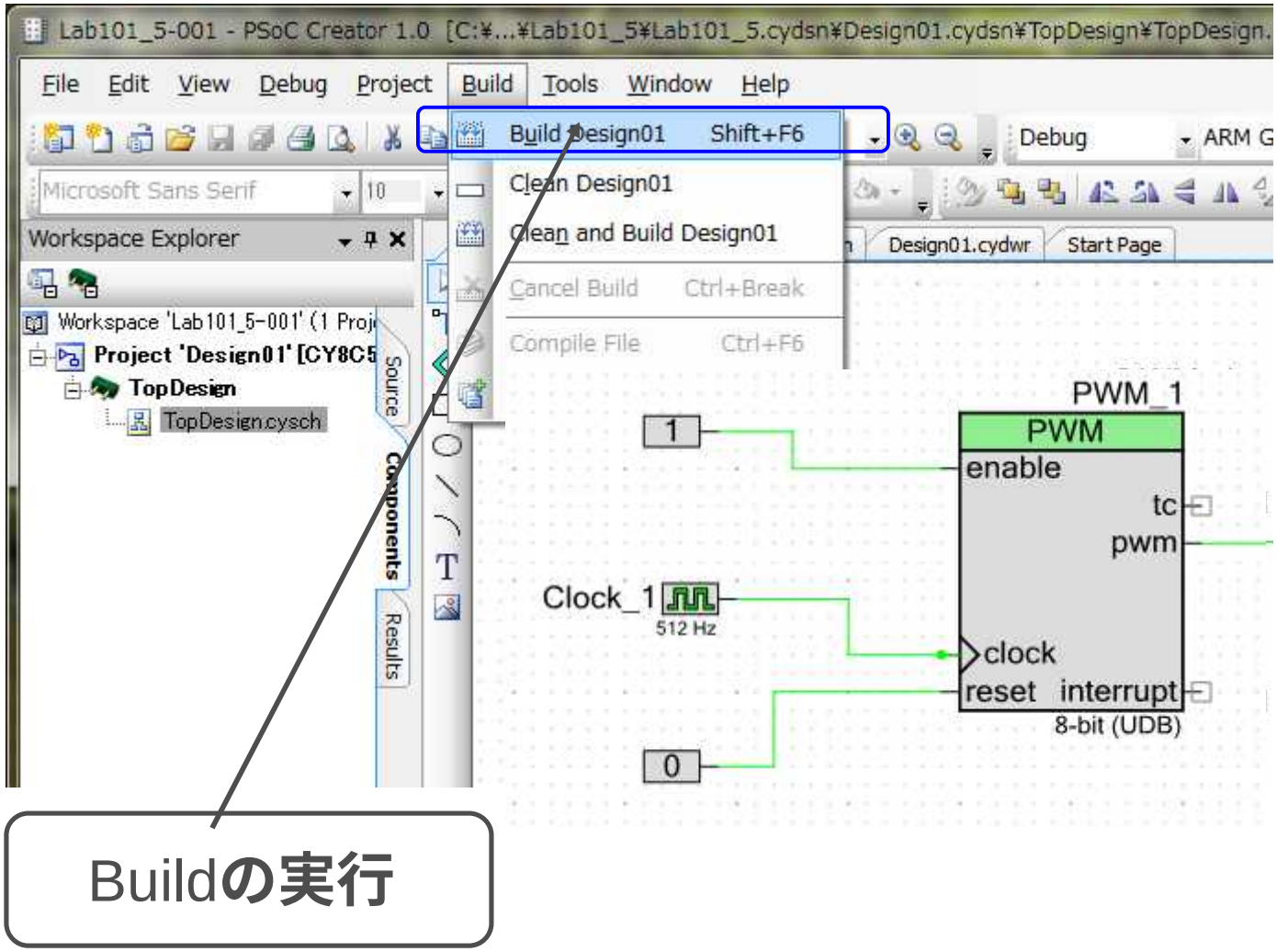
Start Page | *TopDesign.cysch | main.c

```
1  /* ============================
2   *
3   * Copyright YOUR COMPANY, THE Y
4   * All Rights Reserved
5   * UNPUBLISHED, LICENSED SOFTWAR
6   *
7   * CONFIDENTIAL AND PROPRIETARY
8   * WHICH IS THE PROPERTY OF your
9   *
10  * ============================
11  */
12 #include <device.h>
13
14 void main()
15 {
16     /* Place your initialization
17
18     /* CYGlobalIntEnable; */ /*
19     for(;;)
20     {
21         /* Place your applicatio
22     }
23 }
24
25 /* [] END OF FILE */
26
```

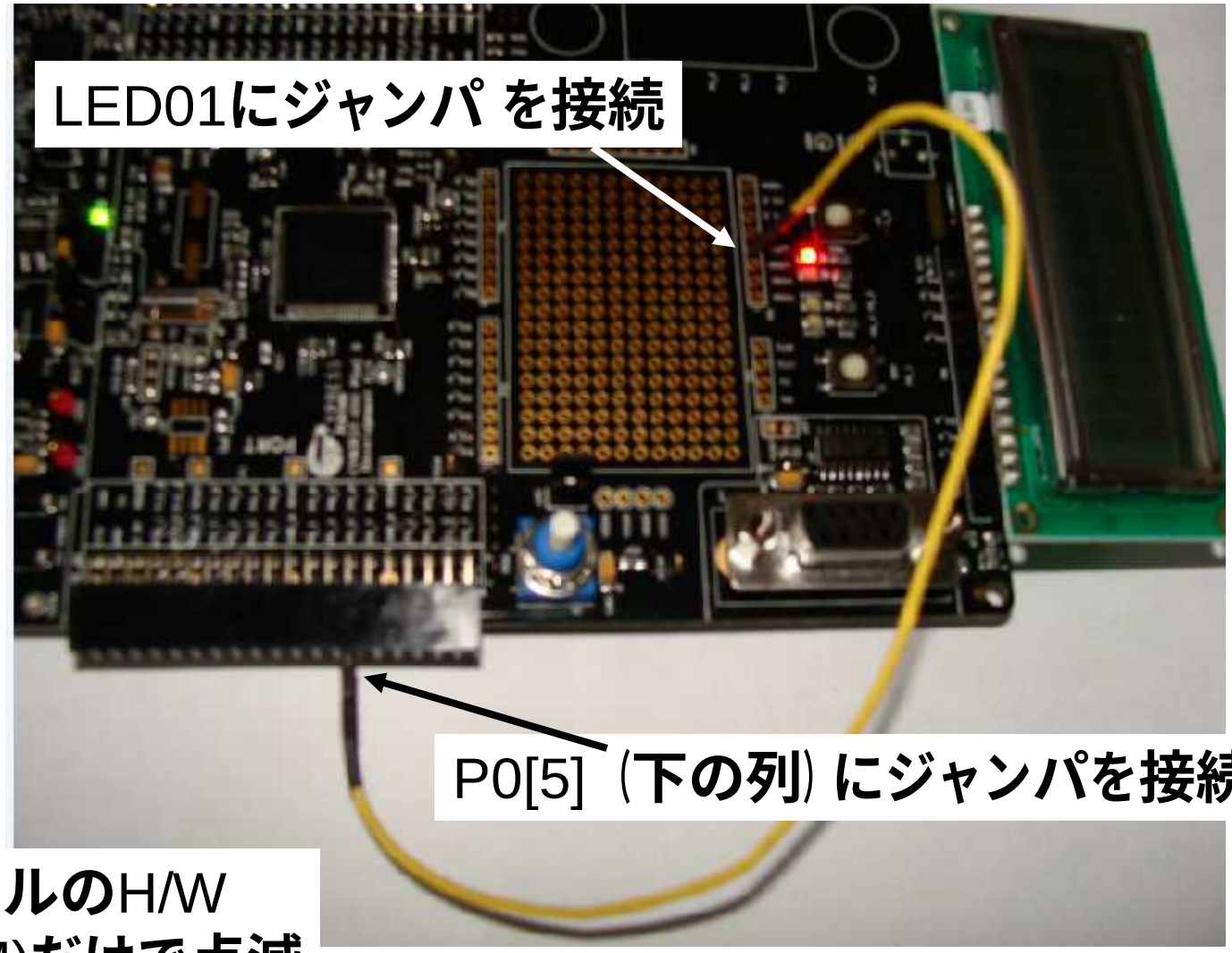
```
12 #include <device.h>
13
14 void main()
15 {
16     PWM_1_Start();
17
18     }
19
20 /* [] END OF FILE */
21
```

PWM_1_Start(); の記述

Step10.ビルドの実行

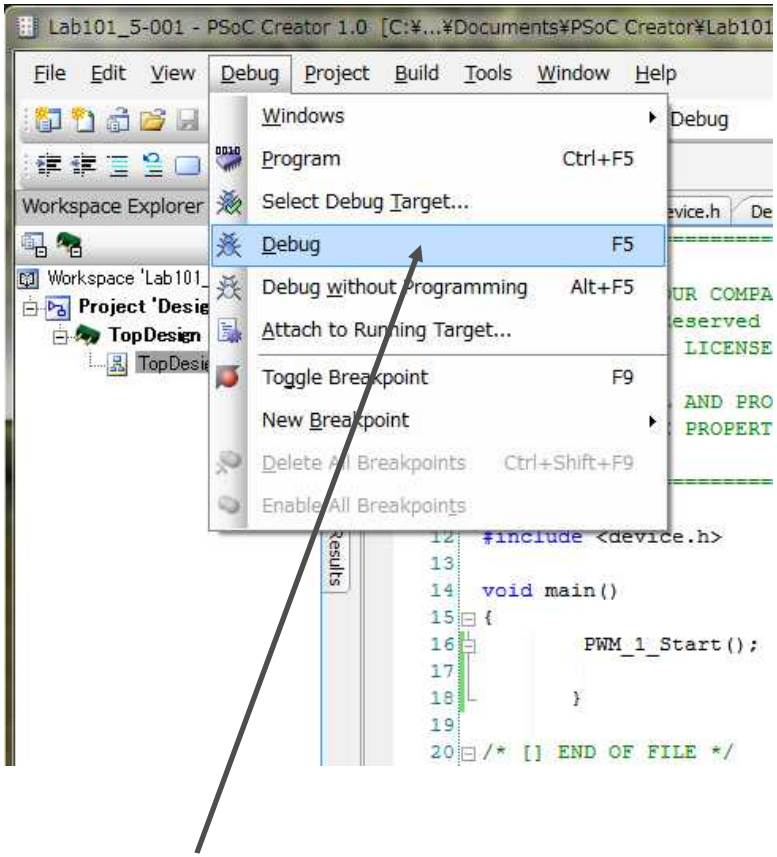


Step11.動作確認



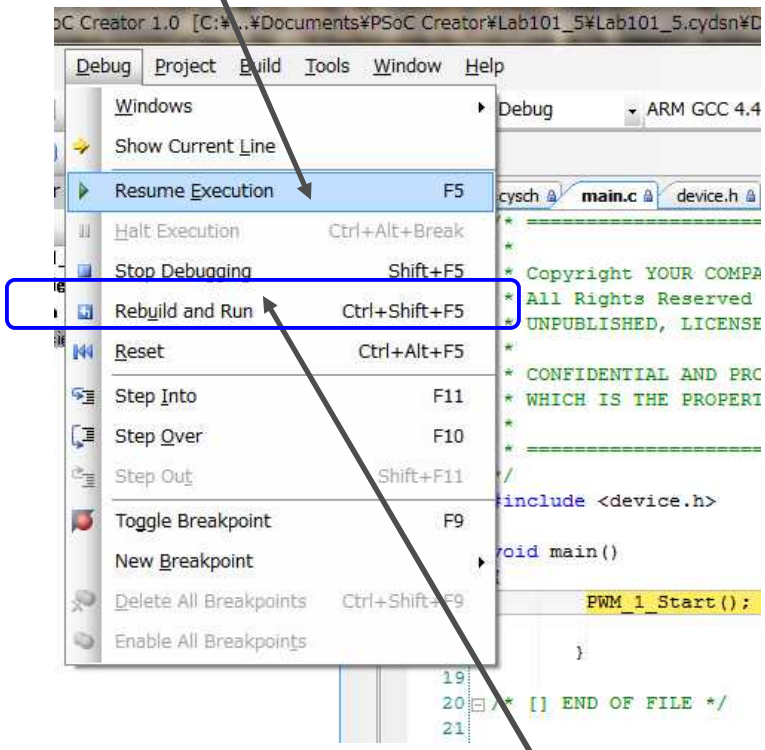
デジタルのH/W
(PWM)だけで点滅

Step12.デバッグの実行



1.デバッグプロセスの開始
(PSoC基板への書込開始)

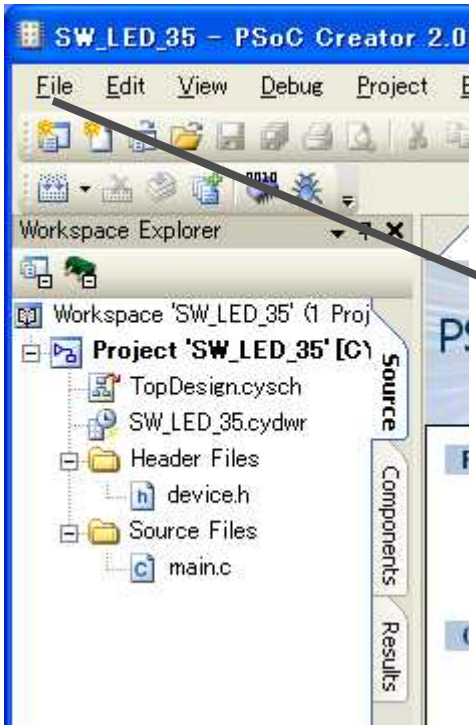
2.実デバッグの開始
(PSoC基板への書込開始)

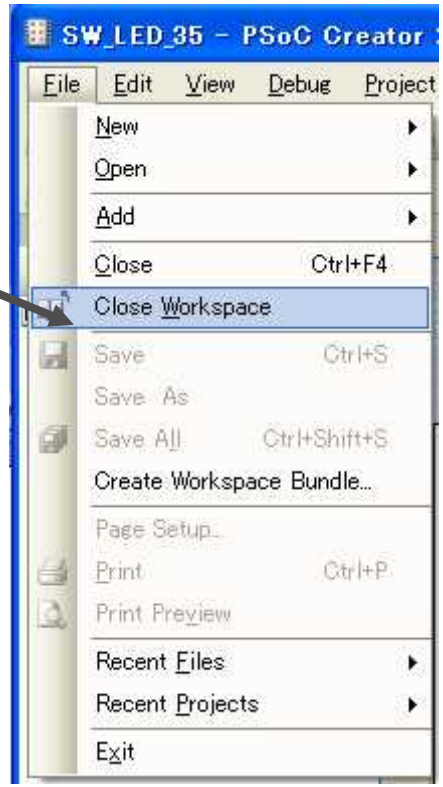


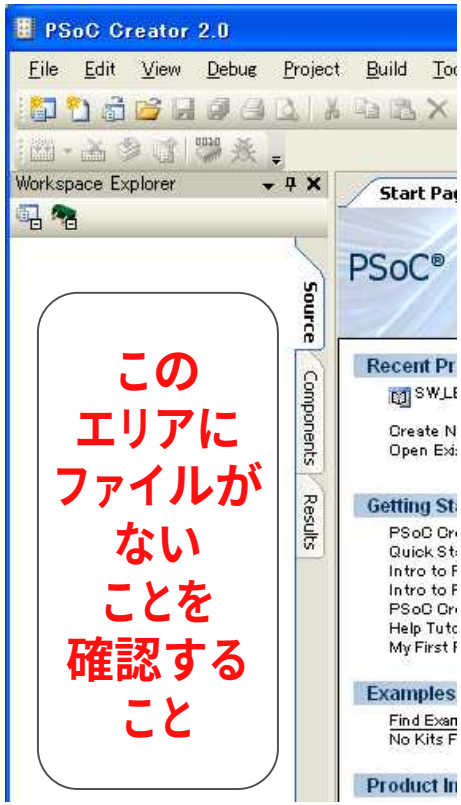
3.デバッグの終了は
Stop Debuggingをクリック

Step 13:終了:プロジェクト/ワークスペースのクローズ

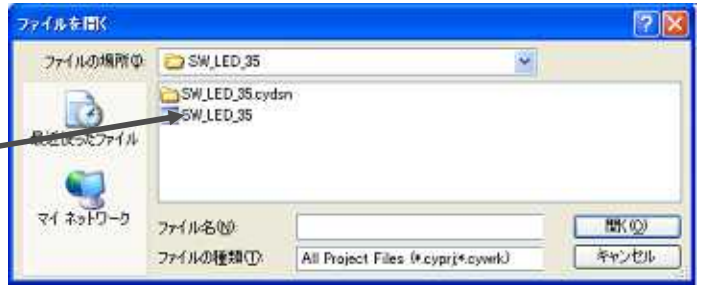
1.File > Close Workspaceを実行







プロジェクトをロードして再開する場合は、
File>Open>Project/Workspaceを実行
プロジェクト/ワークスペースを選択



ビルドのプロセス

Generate a Configuration
(ハードウェアの合成、配置配線)

- Design Elaboration
- Netlisting
- Verilog
- Logic Synthesis
- Technology Mapping
- Analog Place and Route
- Digital Packing
- Digital Placement
- Digital Routing
- <…there's more…>

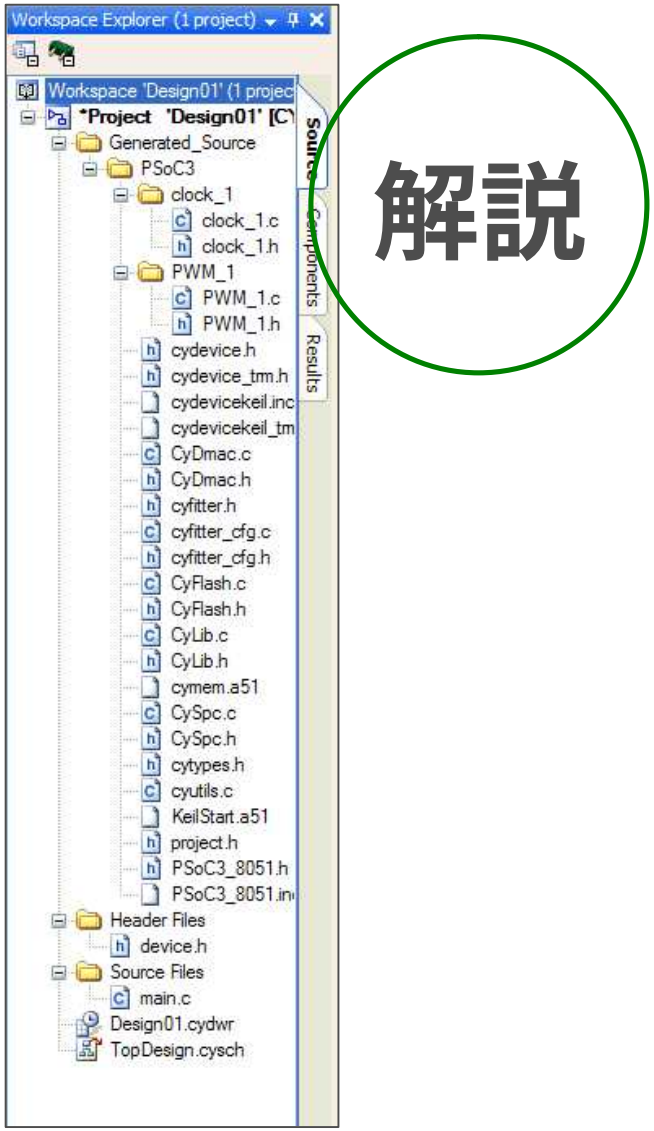
ビルドのプロセス

API Generation　(API**生成**)
Compilation
Configuration Generation
Configuration Verification

API:Application Program Interface

Development Files

Core Cypress Libraries (CyLib)
Registers, macros, types (cytypes)
Component addressing (cyfitter)



解説

サポートされるコンパイラ

Free Bundled compiler options

- PSoC 3: Cypress-Edition Keil™ CA51 Compiler Kit
- PSoC 5: GNU/CodeSourcery Sourcery G++™ Lite
- No code size restrictions, not board-locked, no time limit
- Fully integrated including full debugging support

Upgrade, more optimization/compiler-support options

- PSoC 3: Keil CA51™ Compiler Kit
- PSoC 5: Keil RealView® Microcontroller Development Kit
- Higher levels of optimization
- Direct support from the compiler vendor

Upgrade Compiler Pricing

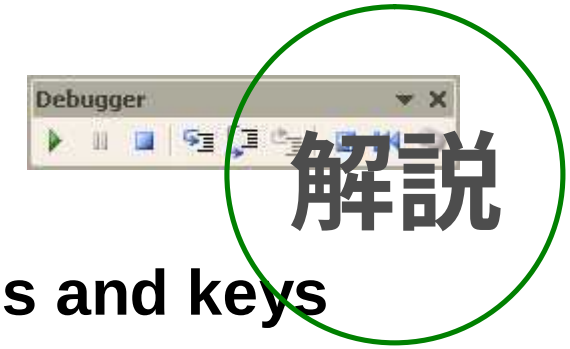
- Set and managed by our 3rd party partner, Keil
- Already own these compilers? No need to buy another license!
- Keil CA51 Compiler Kit ~$2,000
- Keil RealView MDK ~$3,000-5,000

解説

統合されているデバッガ

JTAG and SWD connection

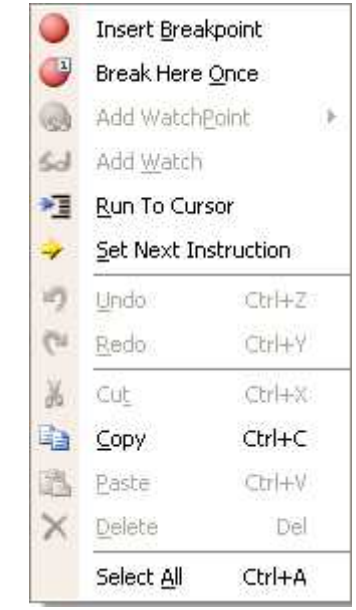
- All devices support debug
- MiniProg3 programmer / debugger



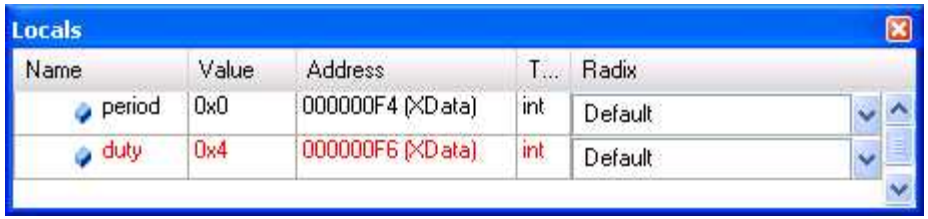
Control execution with menus, buttons and keys

Full set of debug windows

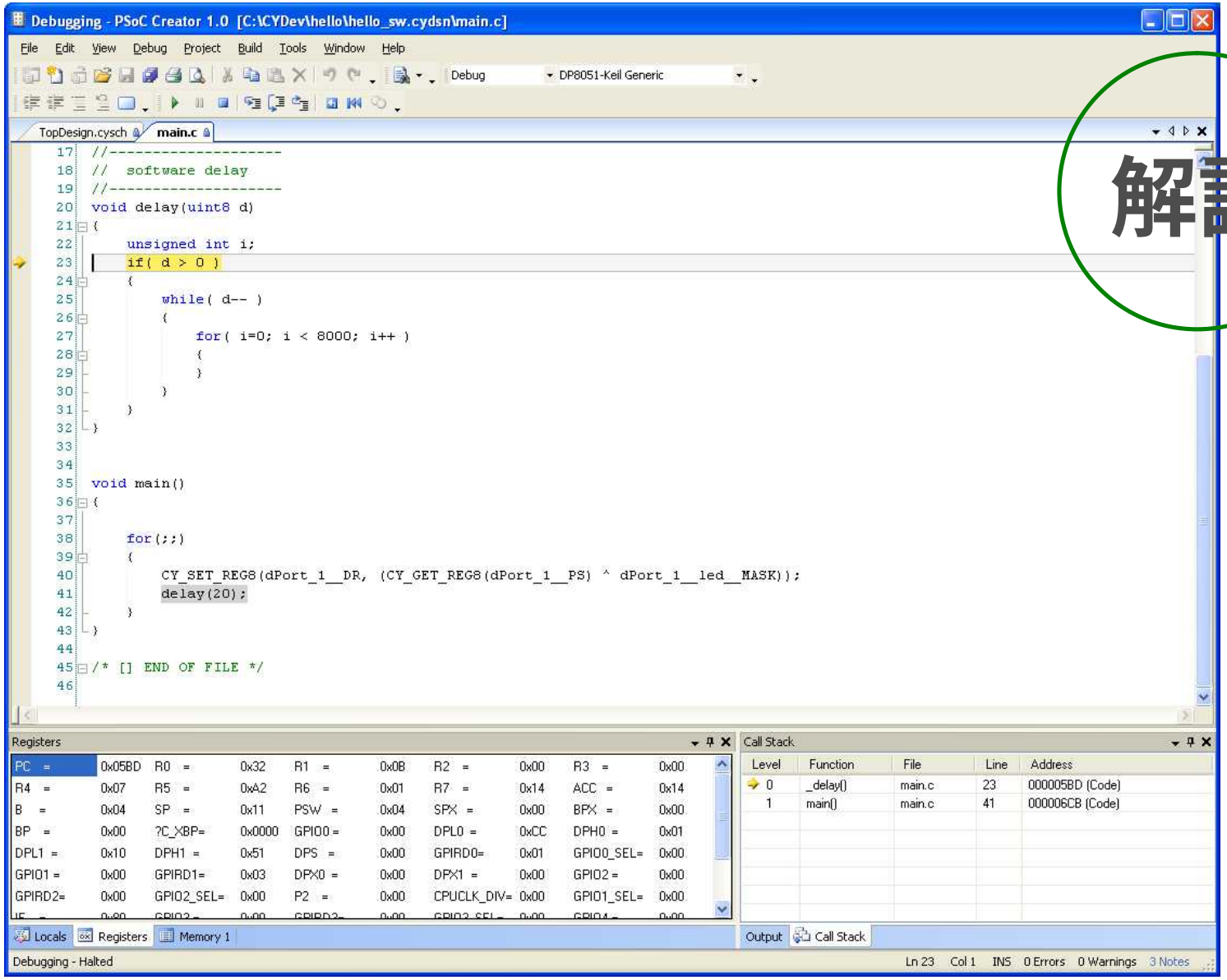
- Locals, register, call stack, watch (4), memory (4)
- C source and assembler
- Components



Set breakpoints in Source Editor



Debugger Windows



課題演習

LEDの点滅間隔を変更してみよう
最初に点滅時間の目標値(例　0.2秒)を決める
続いて、この目標値を実現するために、
設計を変更する。
デバッグで目標値になっているかを検証する。

セーブ後は、File>Close Workspace で終了します。

課題演習

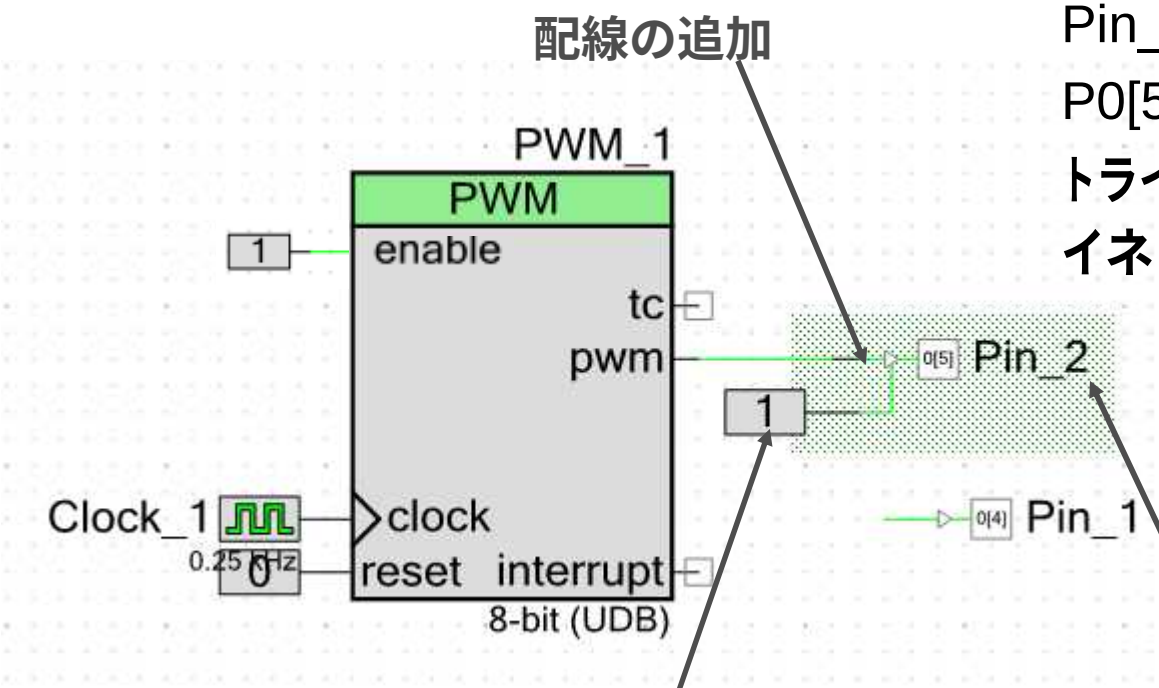
ソフトウェアによるピン制御(Pin_1)**と**
ハードウェアによる
(Pin_2)**トライステート制御の追加で**
二通りの方法でLED**を点滅させる**

Pin_1**は、**P0[4]
デジタル出力ピン

Pin_2**は、**PWM**の出力、**
P0[5]**に接続。**
トライステート制御で
イネーブル (Logic High '1')

セーブ後は、File>Close Workspace **で終了します。**

ピンを追加し回路を変更する



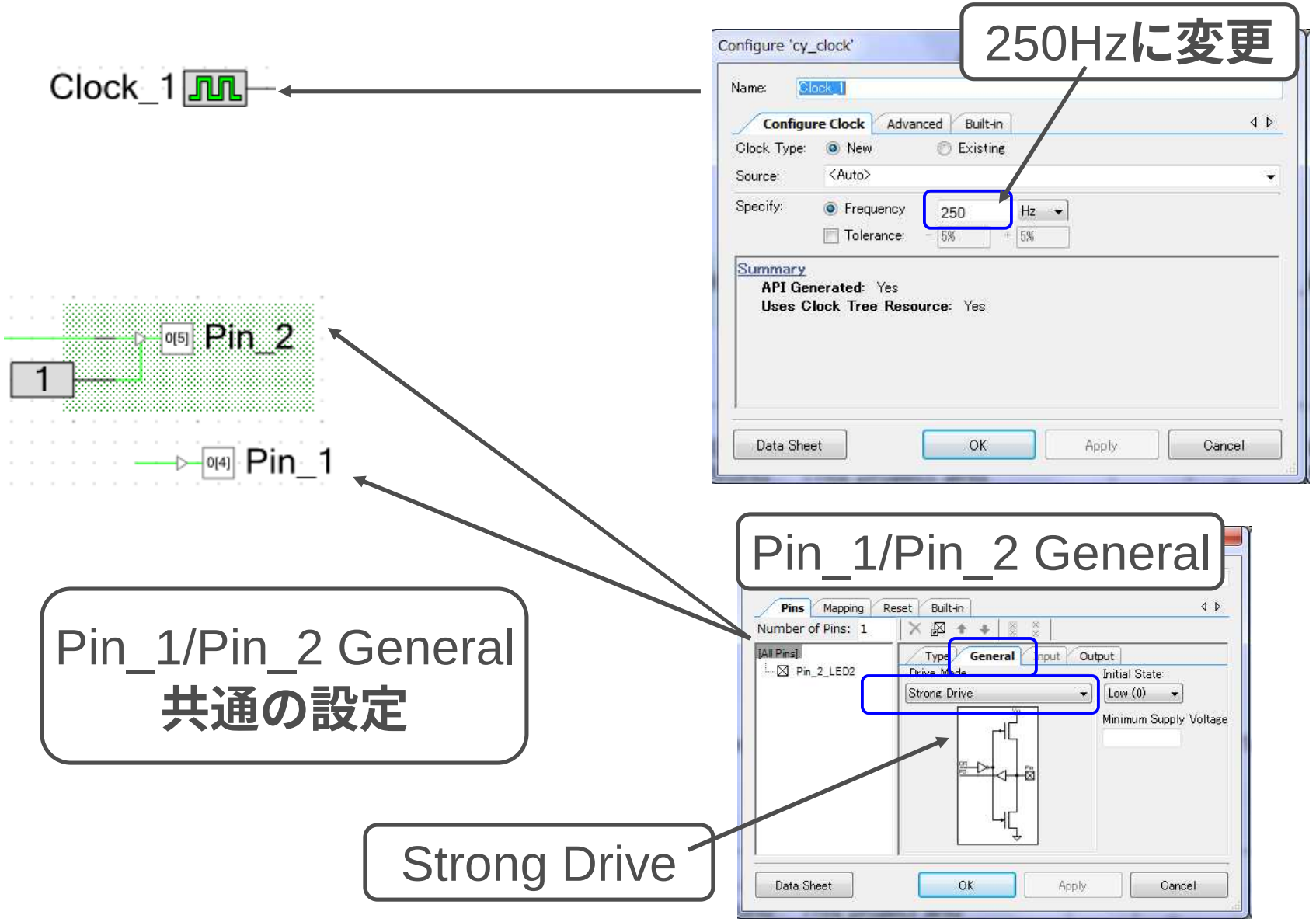
Pin_2は、PWMの出力、
P0[5]に接続。
トライステート制御で
イネーブル (Logic High ‘1’)

Pin_1は、P0[4]
デジタル出力ピン

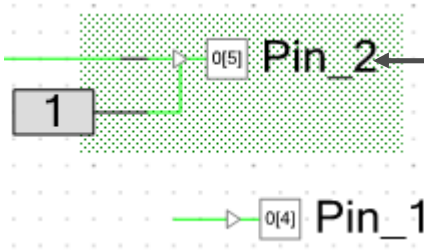
Digital>Logic > Logic High ‘1’

Port and Pins > Digital Output Pin
自動的にピン番号がインクリメント(+1)されます

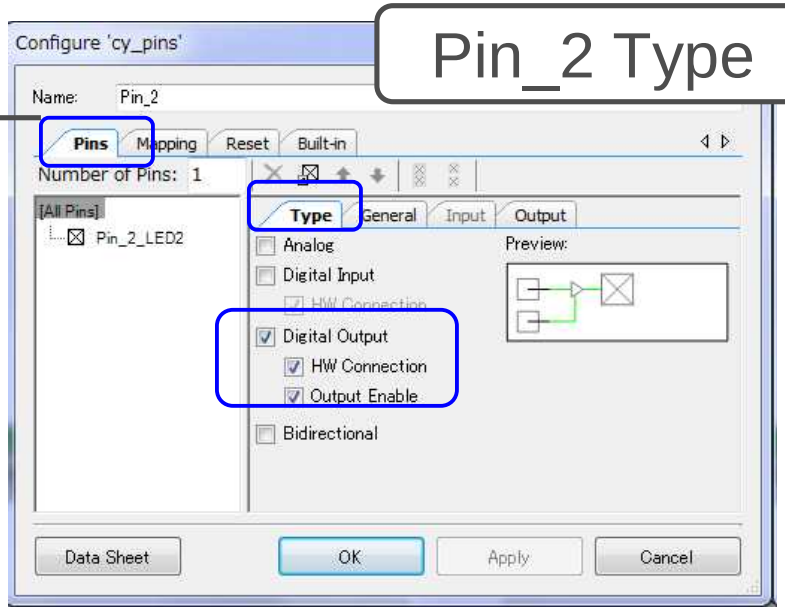
クロックの変更,Pin_1/_2共通設定



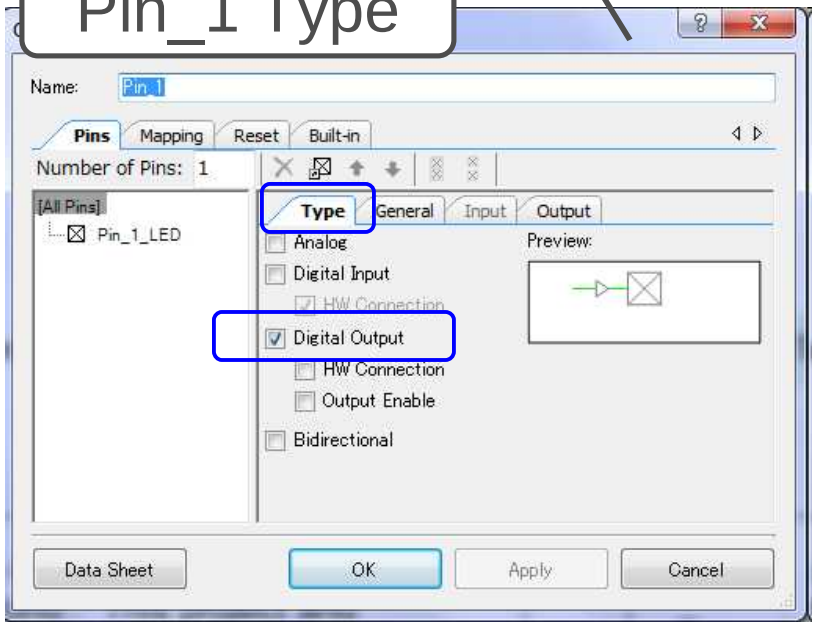
Pin_1/Pin_2のコンフィギュレーション



Pin_2 Type



Pin_1 Type

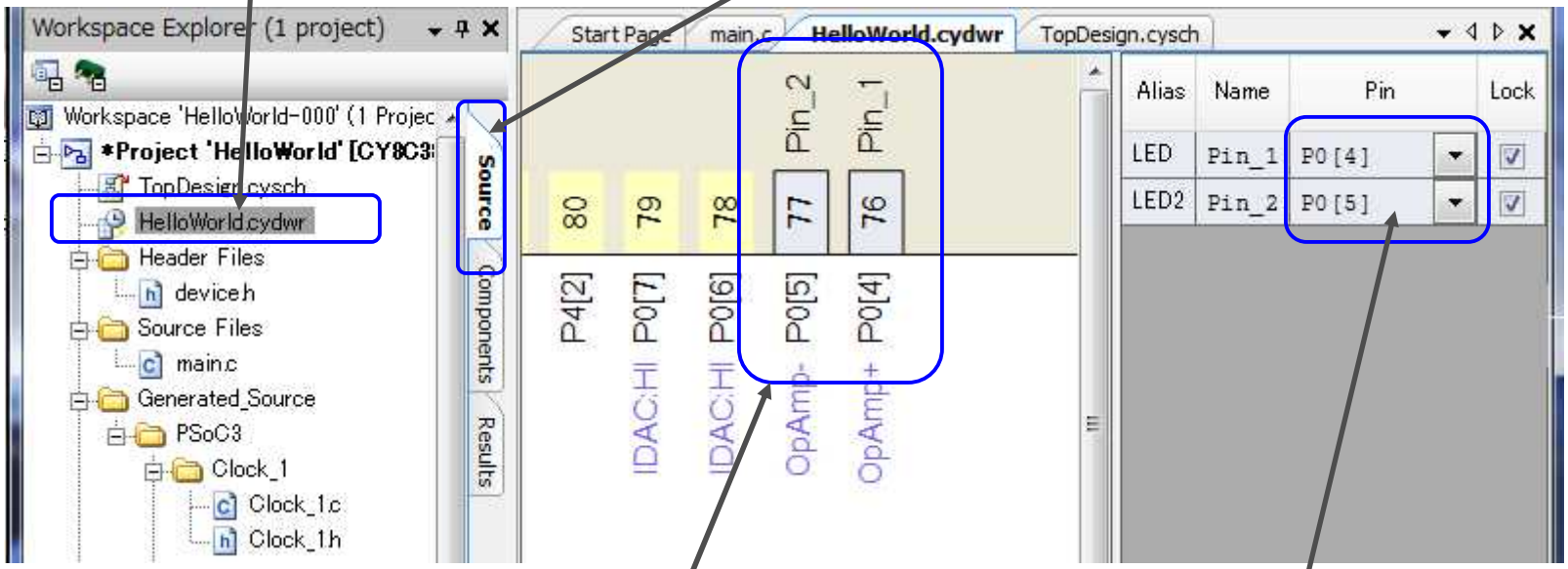


Pin_2は、PWMの出力、
P0[5]に接続。
トライステート制御で
イネーブル (Logic High ‘1’)

ピンの設定

2. .cydwrファイルを
ダブルクリック

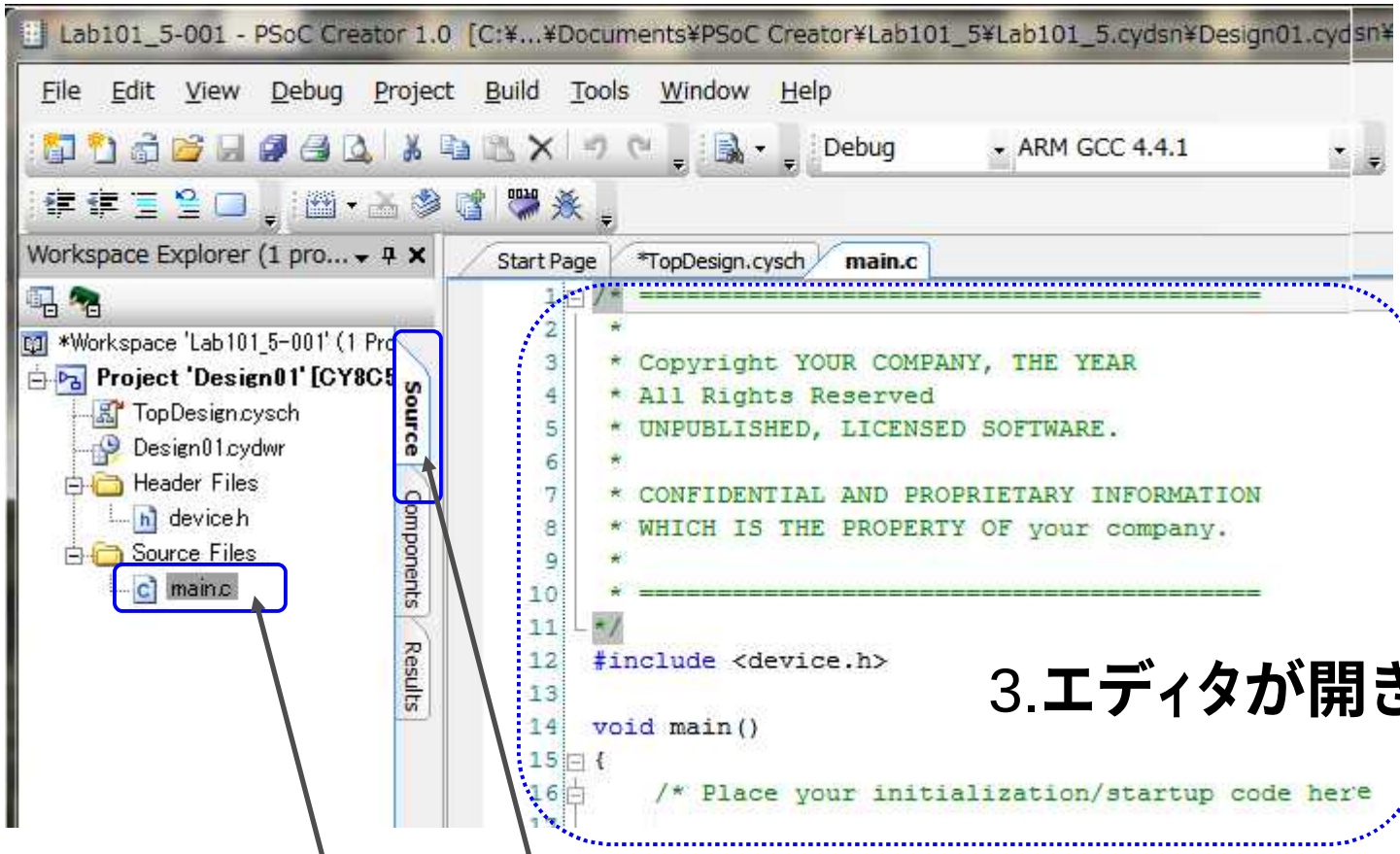
1. Source ウインドウをクリック



3.Pinのアサイン

4.Pinアサインの確認

ソースコードエディタを開く



3.エディタが開きます

1.Sourceタブをクリック

2.main.cをダブルクリック

ソースコードの変更

Start Page | *TopDesign.cysch | main.c

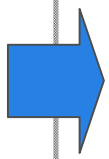
```
/* ============================
 *
 * Copyright YOUR COMPANY, THE Y
 * All Rights Reserved
 * UNPUBLISHED, LICENSED SOFTWAR
 *
 * CONFIDENTIAL AND PROPRIETARY
 * WHICH IS THE PROPERTY OF your
 *
 * ============================
*/

#include <device.h>

void main()
{
        PWM_1_Start(); /

    }

/* [] END OF FILE */
```



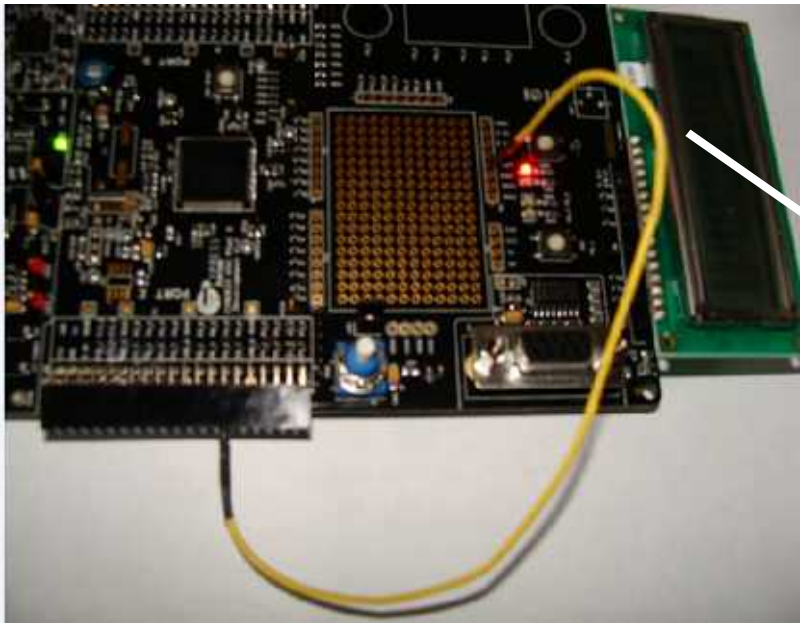
```
void delay(uint8 d)
{
    unsigned int i;
    if( d > 0 )
    {
        while( d-- )
        {
            for( i=0; i < 8000; i++ )
            {
            }
        }
    }
}

void main()
{
    PWM_1_Start(); /* Default PWM name */

    for(;;)
    {
        if( CyPins_ReadPin( Pin_1_LED ))
        {
            CyPins_ClearPin( Pin_1_LED );
        }
        else
        {
            CyPins_SetPin( Pin_1_LED );
        }
        delay(20);
    }
}

/* [] END OF FILE */
```

動作確認と演習課題



Pin_1は、P0[4]
デジタル出力ピン

Pin_2は、PWMの出力、
P0[5]に接続。
トライステート制御で
イネーブル (Logic High ‘1’)

1.基板のどこをジャンパで繋ぐか考える
2.main.cを読み機能を考える
3.デバッグ工程でブレークポイント設定して、
ステップ実行して機能を検証する

セーブ後は、File>Close Workspace で終了します。

Lab PWM_LED_35

以下いずれの場合でもLED**は、消灯しますが、**
Logic Low - ’0’ **と ハイインピーダンス** - ’Z’
を混同しないように

終了

この資料は、デバイスがES1,
ソフトウェアPSoC Creater 1.0SP2 /2.0
をベースに作成しています。
エラッタやバージョンの違いで操作や動作が
異なる場合があります。

Memo

フォローアップURL

http://mikamir.web.fc2.com/?/?.htm

?に入る文字列は、講義中に示します。

本資料は、米国および日本サイプレス社の協力と情報の提供により作成されおり、著作権は以下に帰属します。
内容は定期的に改訂されます。引用や再使用の場合はご連絡ください。

担当講師

ミカミ設計コンサルティング
〒142-0042 **東京都品川区豊町** 2-17-8

三上廉司(みかみれんじ)

Renji_Mikami@nifty.com
http://homepage3.nifty.com/western/mikamiconsult.htm
電話 080-5422-2503(au)