



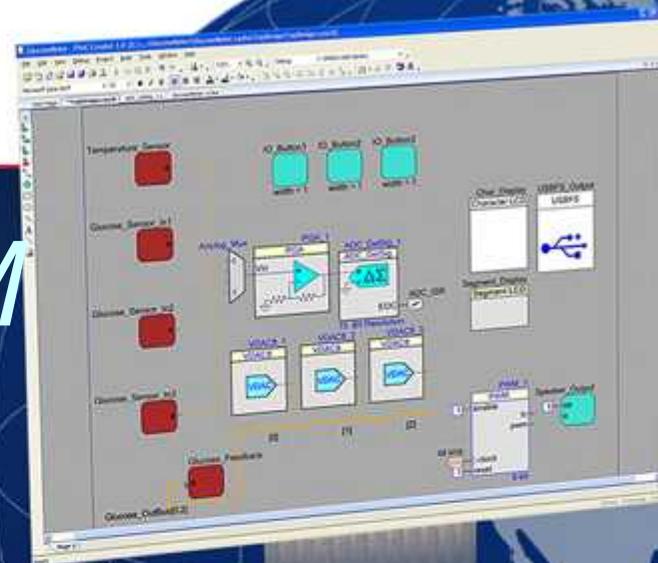
3.4

3

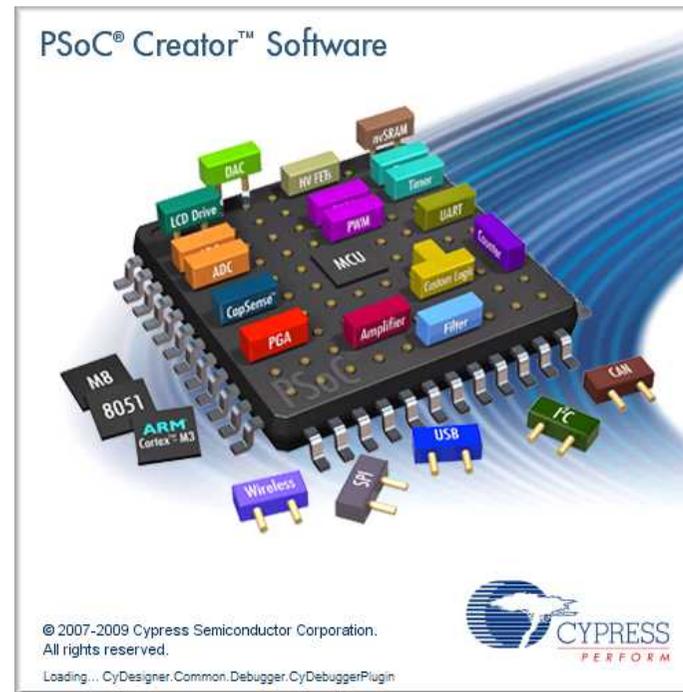
PWMによるLEDの点滅コントロール

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

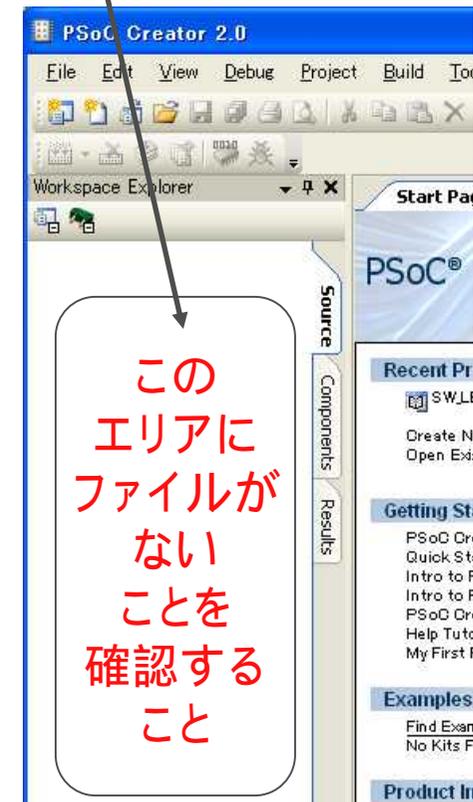
LEDの点滅(ソフトウェア編)

```
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
- 割り込み
- 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

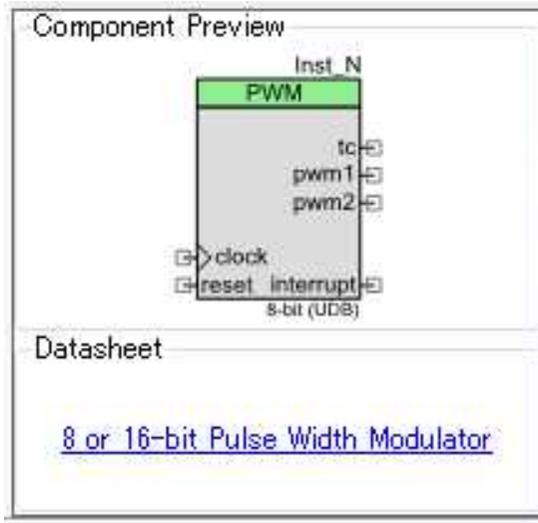
Reuse

書き込みとデバッグ

- 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



CYPRESS PSoC Creator Component Data Sheet

Pulse Width Modulator (PWM) 1.10

Features

- 8 or 16-Bit Resolution
- Multiple Pulse Width Output Modes
- Configurable Trigger
- Configurable Capture
- Configurable Hardware/Software Enable
- Configurable Dead-Band
- Multiple Configurable Kill Modes
- Customized Configuration tool

General Description

The PWM component provides compare outputs to generate single or continuous timing and control signals in hardware. The PWM is designed to provide an easy method of generating complex real time events accurately with minimal CPU intervention. The PWM features may be combined with other analog and digital components to create custom peripherals.

The PWM generates up to 2 left or right aligned PWM outputs or 1 center aligned or dual edged PWM output. The PWM outputs are double buffered to avoid glitches due to duty cycle changes while running. Left aligned PWMs are used for most general purpose PWM uses. Right aligned PWMs are typically only used in special cases which require alignment opposite of left aligned PWMs. Center aligned PWMs are most often used in AC motor control to maintain phase alignment. Dual edge PWMs are optimized for power conversion where phase alignment must be maintained.

Optional deadband provides complementary outputs with adjustable dead time where both are low between each transition. The complementary outputs and dead time are most used to drive power devices in half bridge configurations to avoid shoot through currents causing damage. A kill input is also available that immediately disables the deadband when enabled. Three kill modes are available to support multiple use scenarios.

Software dither modes are provided to increase PWM flexibility. The first dither mode is effective resolution by 2-bits when resources or clock frequency preclude a standard dither mode in the PWM counter. The second dither mode uses a digital input to select one of two PWM outputs on a cycle by cycle basis typically used to provide fast transient response to load converts.

PRELIMINARY

Cypress Semiconductor Corporation • 198 Champion Court • San Jose, CA 95134-1709 • 408-943-2600
Number: 001-46827 Rev. 1A
Revised August 27, 2009

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(ハードウェア割り込み)

- Set priority and vector

DMA(Direct Memory Access設定)

- Manage DMA channels

System

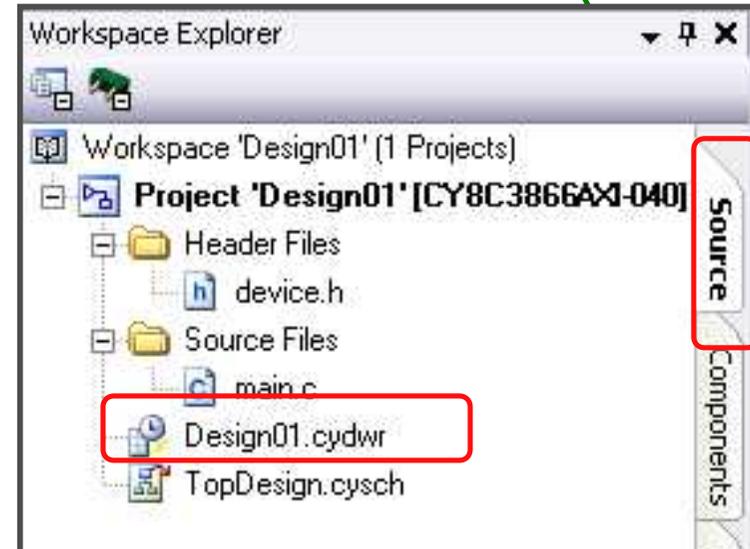
- Debug, boot parameters, sleep mode API generation, etc.

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)

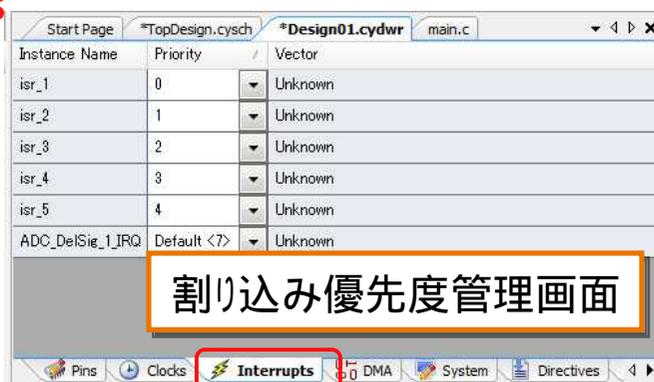
共有リソースエディタ(.cydwr画面から選択)

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。



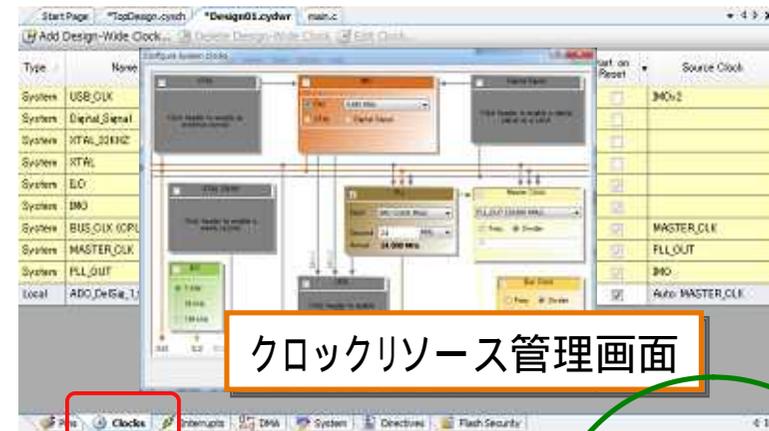
IO管理画面

Pins



割り込み優先度管理画面

Interrupts



クロックリソース管理画面

Clocks



DMA優先度管理画面

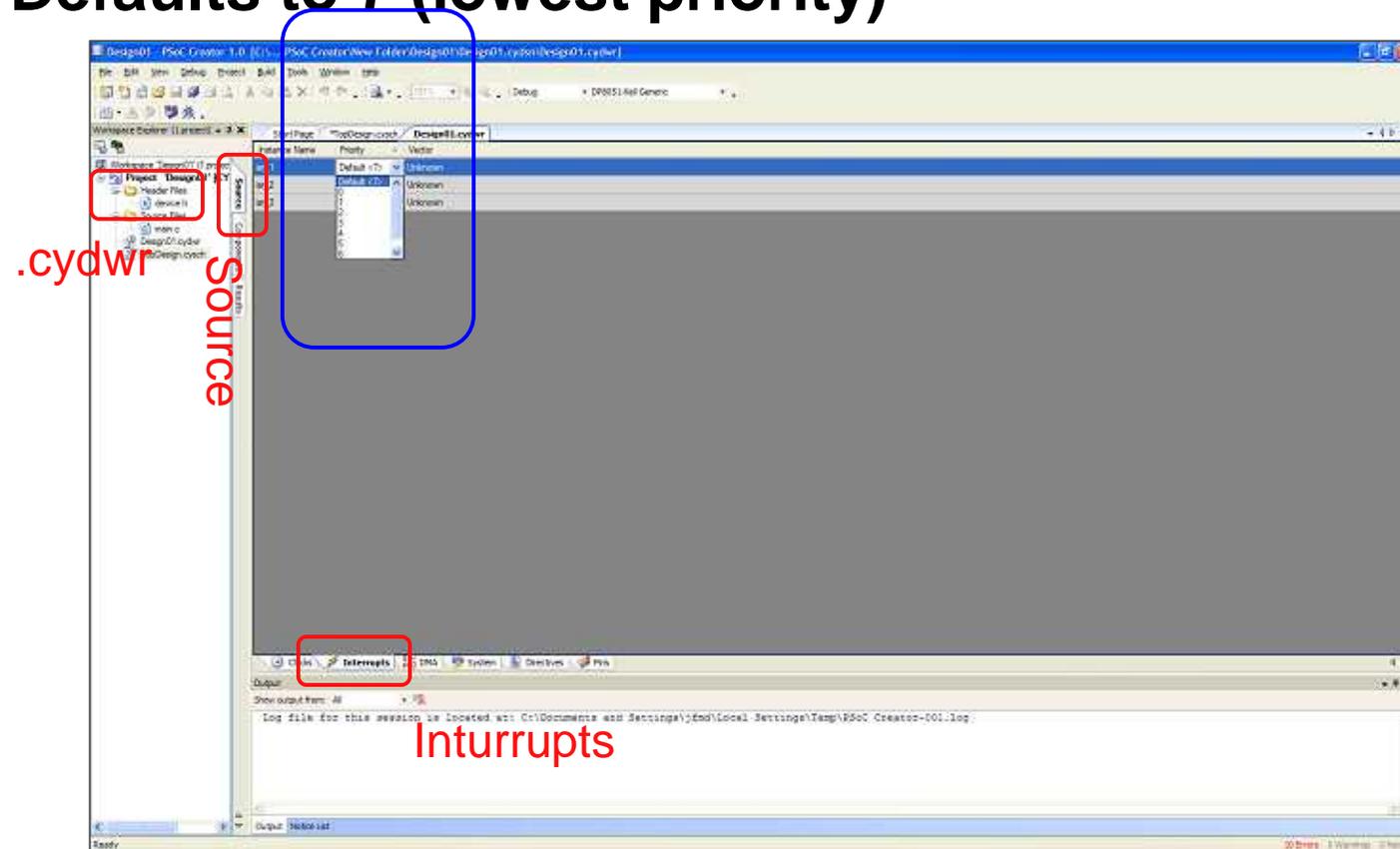
DMA



ハードウェア割り込みの設定

Priority may be changed

Defaults to 7 (lowest priority)



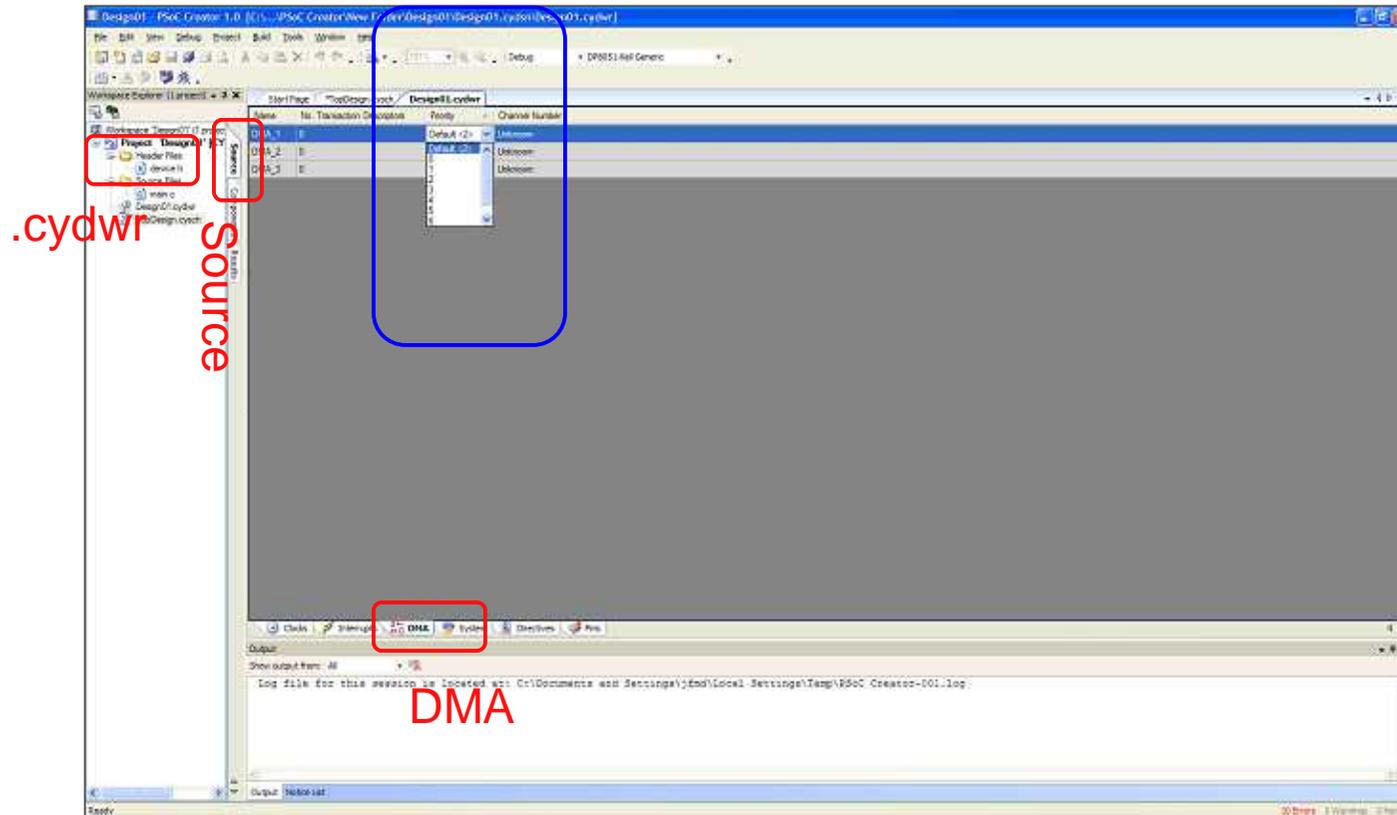
解説

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

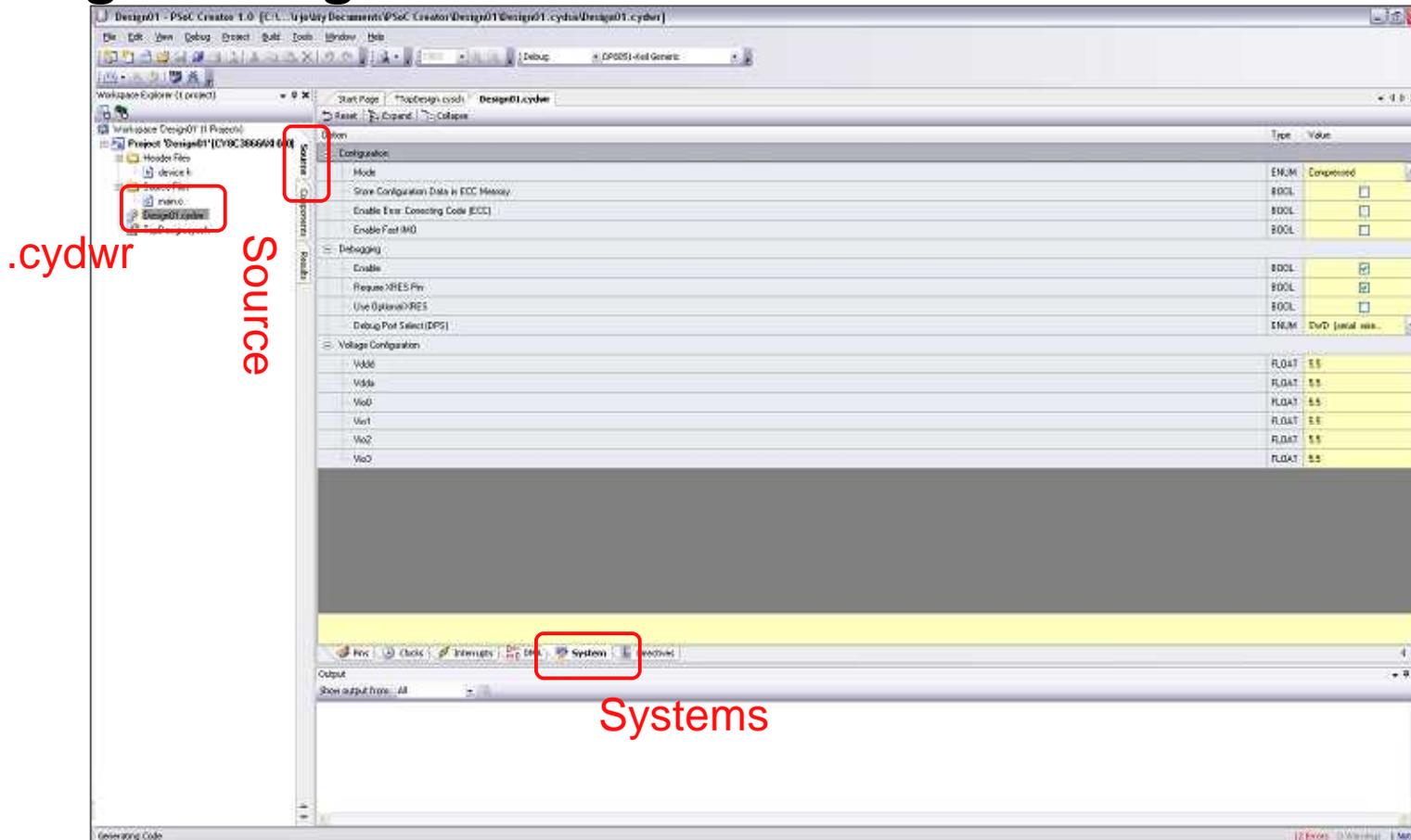


システム設定

System settings

Debug settings

Voltage Configuration



クロックのコンフィギュレーション

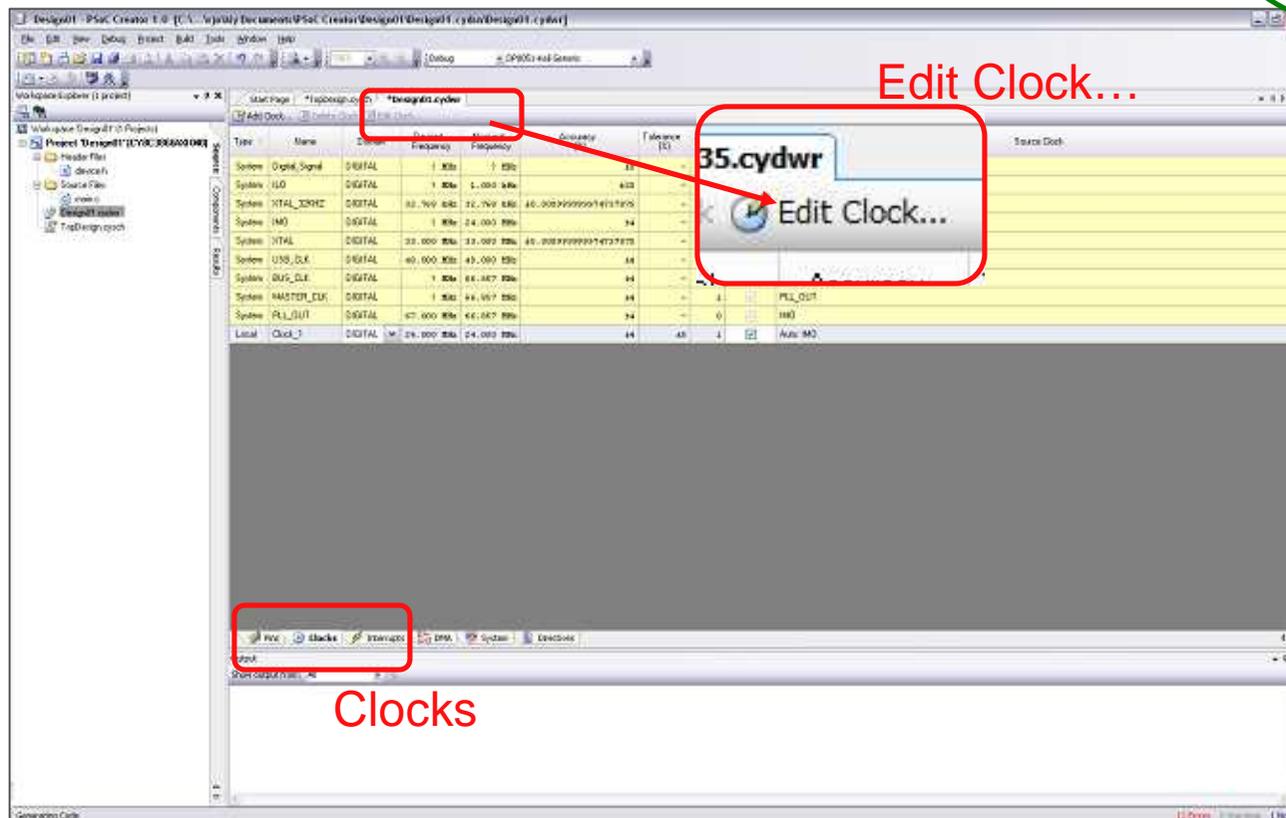
Clocks are allocated to slots in the clock tree

- 8 digital, 4 analog

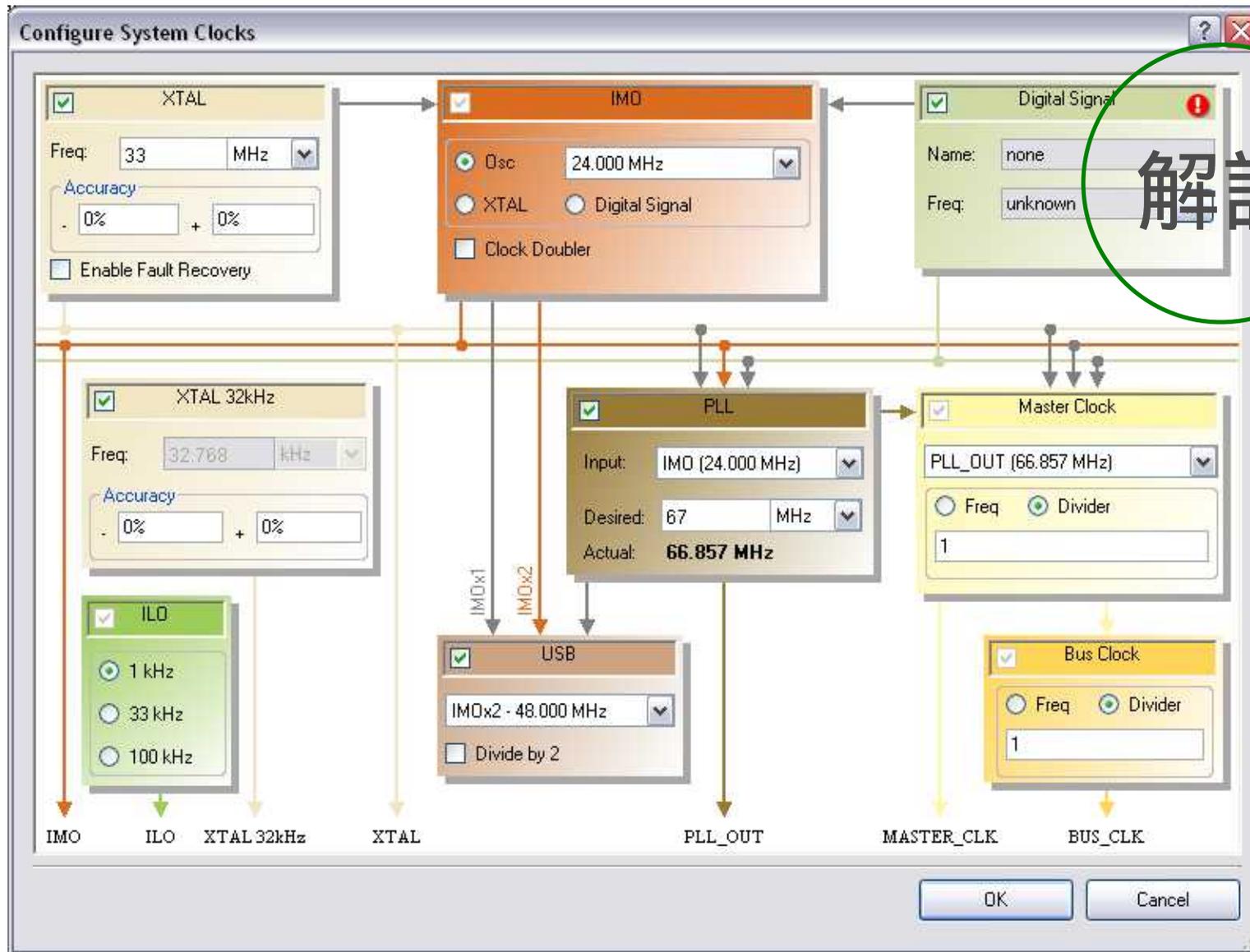
Clocks have software APIs

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)

Source

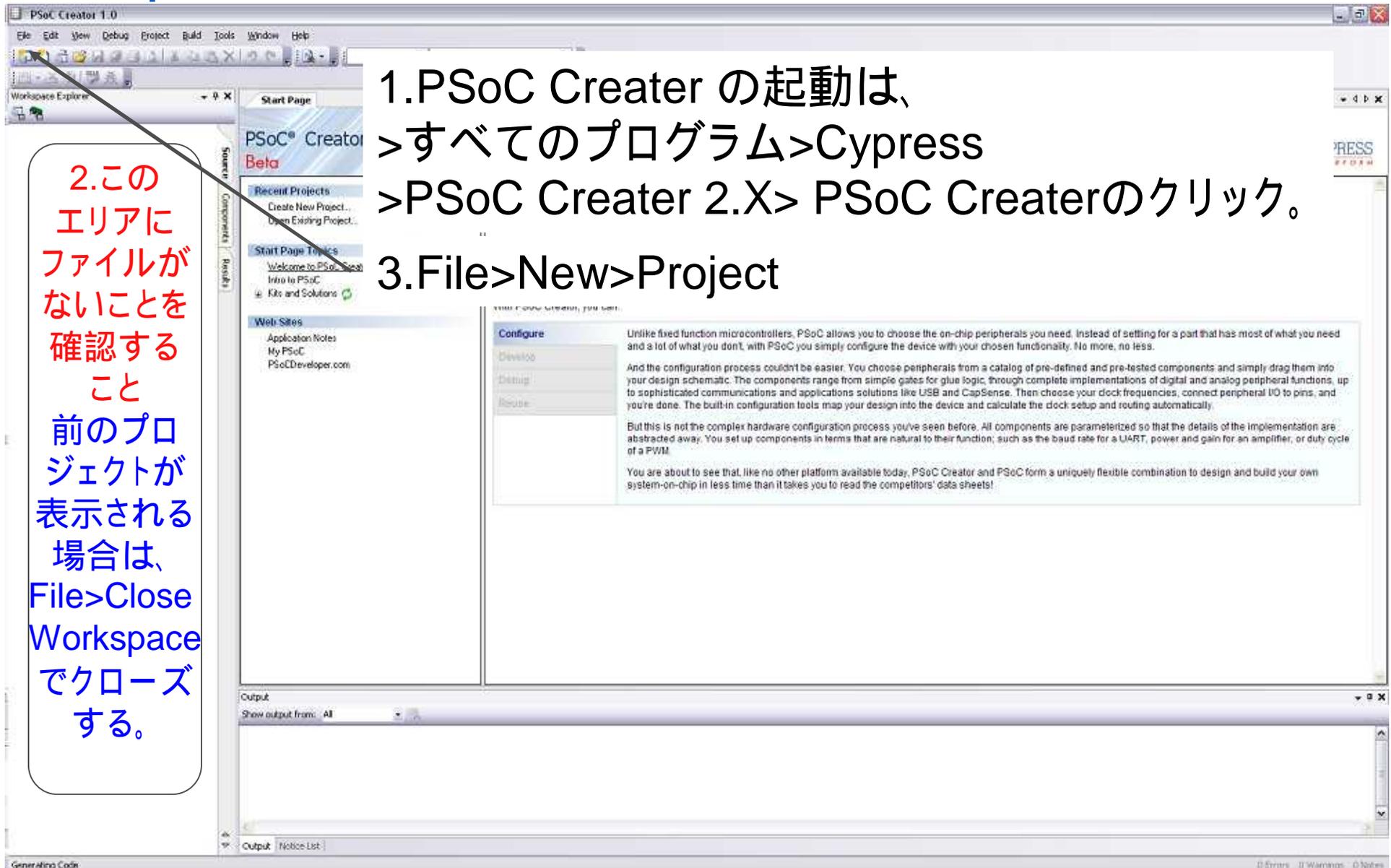
.cydwr

解説

| Alias | Name | Type | Routing Results | Pin |
|-------|------------------|----------------|-----------------|-------------|
| | dPort_1[7:0] | Digital Input | - | - |
| | dPort_2[7:0] | Digital | - | - |
| | aPort_1[0] | Analog | - | - |
| | aPort_1[1] | Analog | - | - |
| | aPort_1[2] | Analog | - | - |
| | aPort_1[3] | Analog | - | P4[1] (70) |
| | aPort_1[4] | Analog | - | - |
| | aPort_1[5] | Analog | - | - |
| | aPort_1[6] | Analog | - | P4[2] (69) |
| | aPort_1[7] | Analog | - | - |
| | LCD_Char_1_UC... | Digital Output | - | PORT0[7:1] |
| | SegLCD_1_Com[0] | Digital Output | - | - |
| | SegLCD_1_Com[1] | Digital Output | - | P12[2] (67) |
| | SegLCD_1_Com[2] | Digital Output | - | - |
| | SegLCD_1_Com[3] | Digital Output | - | - |
| | SegLCD_1_Seg[0] | Digital Output | - | P15[3] (56) |
| | SegLCD_1_Seg[1] | Digital Output | - | - |
| | SegLCD_1_Seg[2] | Digital Output | - | - |
| | SegLCD_1_Seg[3] | Digital Output | - | - |
| | SegLCD_1_Seg[4] | Digital Output | - | - |
| | SegLCD_1_Seg[5] | Digital Output | - | - |
| | SegLCD_1_Seg[6] | Digital Output | - | - |
| | SegLCD_1_Seg[7] | Digital Output | - | - |

Pins (PSoC Creator V2.00 では、左端に表示されます。)

Step1.PSoC Creator Softwareの起動



The screenshot shows the PSoC Creator 1.0 software interface. The main window displays the 'Start Page' with various options like 'Recent Projects', 'Start Page Topics', and 'Web Sites'. A 'Configure' panel is visible on the right, showing options for 'Device', 'Debug', and 'Route'. The bottom status bar indicates 'Generating Code'.

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

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。

3.File>New>Project

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)

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)

6.OKをクリック

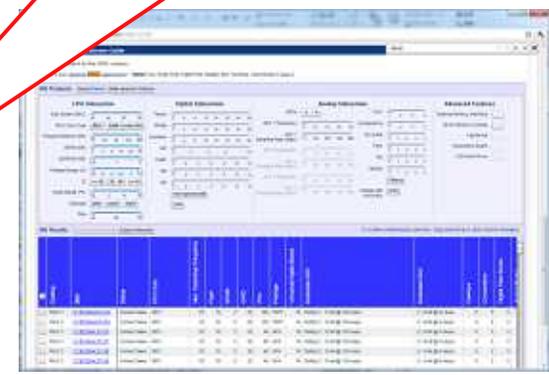
Device Selector デバイスの選択や変更

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

| Part Number | Architecture | CPU Speed (MHz) | Flash (KB) | SRAM (KB) | EEPROM (bytes) | Trace Buffer (KB) | DMA Channels | PLL | LED Drive (mux ratio) | CapSense | A/D | UART DAC | SPI/CT Blocks | Other |
|---------------------------|------------------------|-----------------|------------|-----------|----------------|-------------------|--------------|----------|-----------------------|----------|--|----------|---------------|----------|
| 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 |

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。
ここから適切な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

画面が見つからない場合等は、
Window > Reset Layoutを実行

1.Source タブをクリック

2.TopDesign.cysch をクリック

ここが次で開くコンポーネント・カタログ ウィンドウ

コンポーネント・カタログ

Catalog Folders

Analog

- ADC
- Amplifier
- DAC

Digital

- Registers
- Functions
- Logic

Communication

Display

System

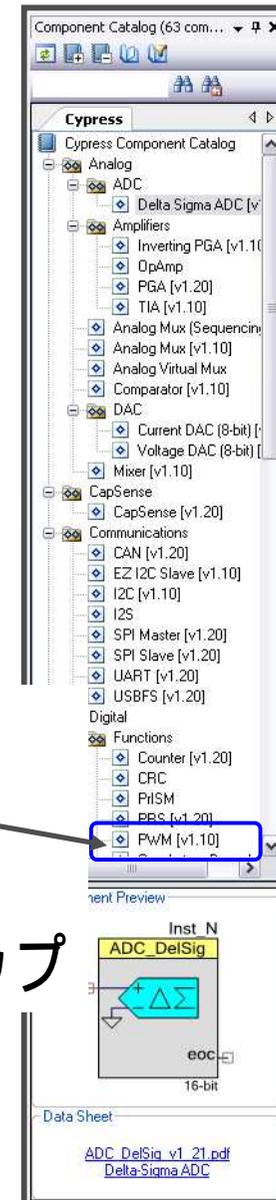
Catalog Preview

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)

拡大縮小アイコンの動作を確認する

PWM_1
PWM
tc
pwm1
pwm2
clock
reset interrupt
8-bit (UDB)

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を選ぶとコンポーネントの詳細が見れる

Component Catalog (63 com...)

- Cypress
- Analogue
- ADC
- Delta Sigma ADC [v1.10]
- Amplifiers
- Inverting PGA [v1.10]
- OpAmp
- PGA [v1.20]
- TIA [v1.10]
- Analog Mux (Sequencing)
- Analog Mux [v1.10]
- Analog Virtual Mux
- Comparator [v1.10]
- DAC
- Current DAC (8-bit)
- Voltage DAC (8-bit)
- Mixer [v1.10]
- CapSense
- CapSense [v1.20]
- Communications
- CAN [v1.20]
- E2 I2C Slave [v1.10]
- I2C [v1.10]
- I2S
- SPI Master [v1.20]
- SPI Slave [v1.20]
- UART [v1.20]
- USBFS [v1.20]
- Digital
- Functions
- Counter [v1.20]
- CRC
- PRSM
- PRS [v1.20]
- PWM [v1.10]

Component Preview

Inst N
PWM
tc
pwm1
pwm2
clock
reset interrupt
8-bit

Data Sheet

Pwm1 v1.10.pdf
8, 16, 24 or 32-bit Pulse Width Modulation

Pins, Logic and Clock コンポーネントの配置

コンポーネントを探して配置してみよう

System > Clock

Digital > Logic > Logic High '1'

Digital > Logic > Logic Low '0'

Port and Pins > Digital Output Pin

Component Catalog (63 com...)

- Logic High '1'
- Logic Low '0'
- Lookup Table
- Multiplexer
- Nand
- Nor
- Not
- Or
- Virtual Mux
- Xnor
- Xor
- Registers
- Control Register [v1.10]
- Status Register [v1.10]
- Display
- Character LCD [v1.20]
- Segment LCD - Static [v1.10]
- Segment LCD [v1.20]
- Filters
- Filter [v1.20]
- Ports and Pins
- Analog Pin
- Digital Bidirectional Pin
- Digital Input Pin
- Digital Output Pin
- System
- Boost Converter [v1.10]
- Clock
- Die Temperature [v1.10]
- DMA
- EEPROM [v1.10]
- Interrupt [v1.10]
- RTC [v1.20]
- SleepTimer
- VRef

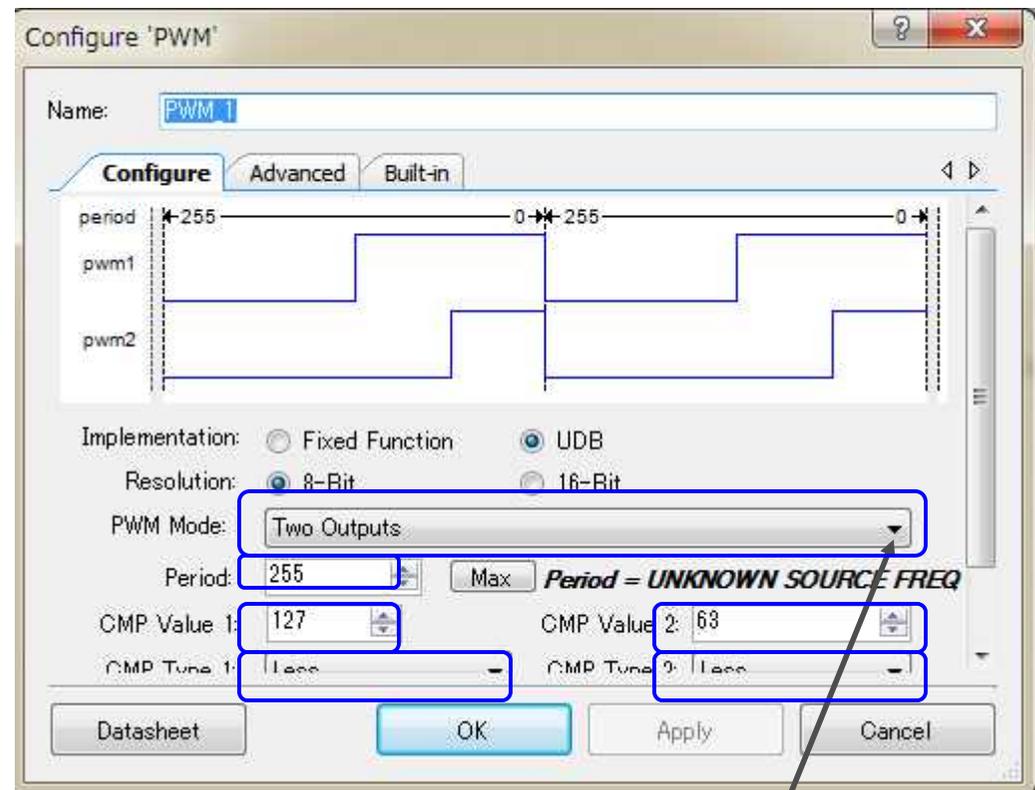
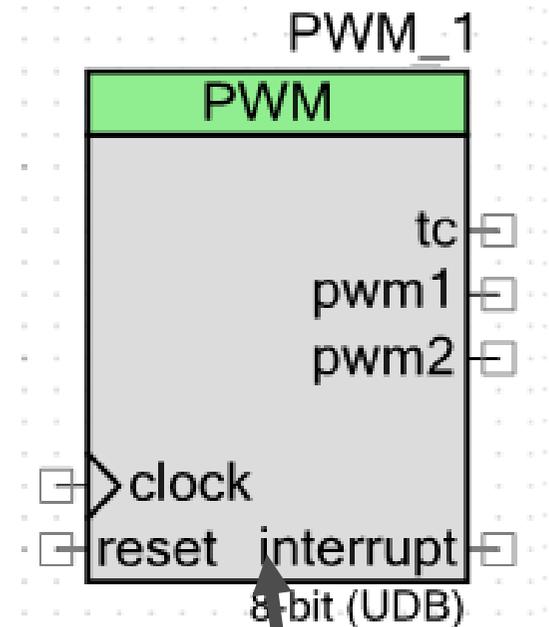
追補：回路図エディタの操作

下記のコマンドを使用して、
回路図の拡大縮小を試して下さい

| 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

Configure 'PWM'

Name: PWM_1

Configure Advanced Built-in

period 199 0 199 0

pwm

Implementation: Fixed Function UDB

Resolution: 8-Bit 16-Bit

PWM Mode: One Output

Period: 255 Max

CMP Value 1: 127

CMP Type 1: Less

Dead Band: Disabled

Data Sheet OK Apply

Configure 'PWM'

Name: PWM_1

Configure Advanced Built-in

Enable Mode: Hardware Only

Run Mode: Continuous

Trigger Mode: None

Terminal Count Event

Compare 1 Event

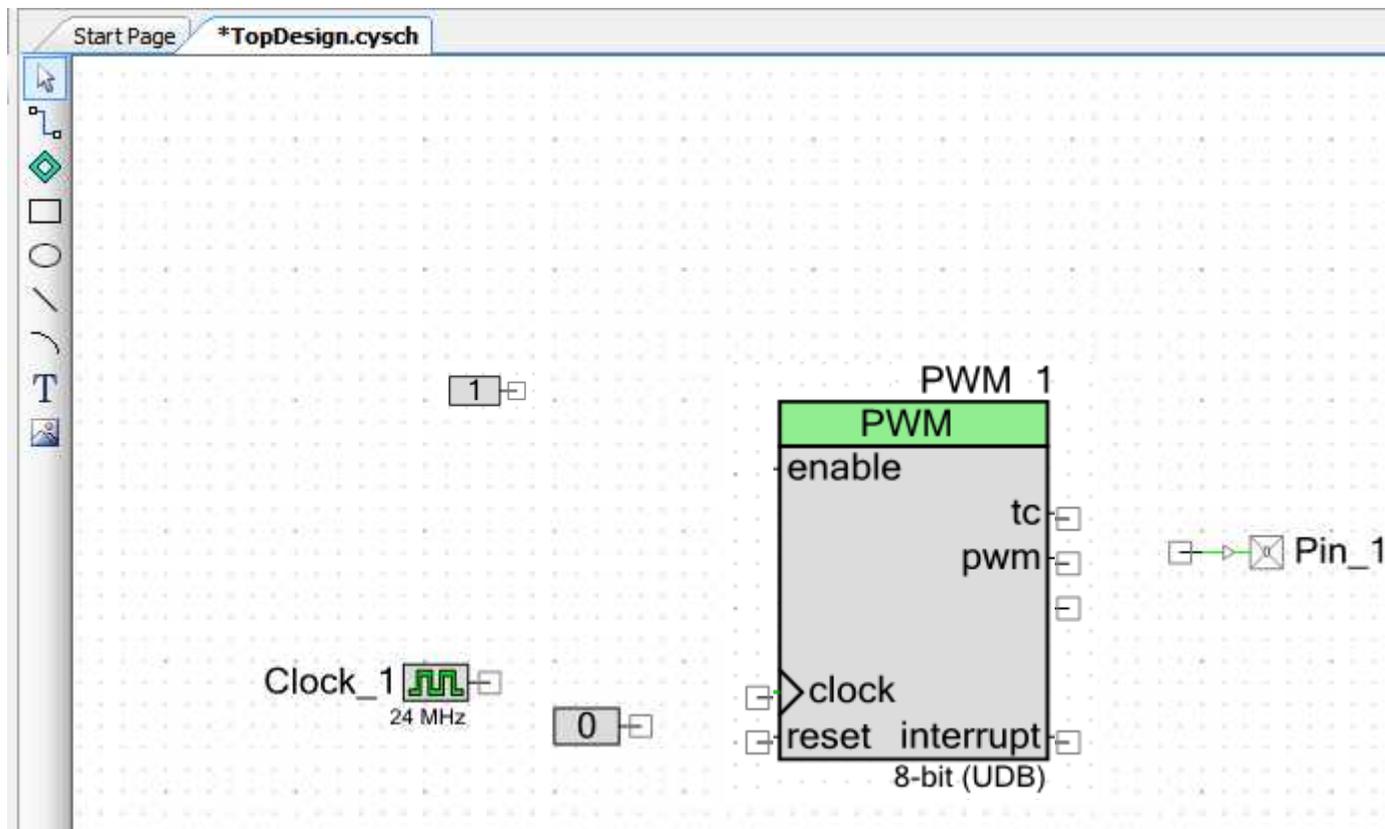
Compare 2 Event

Fill Event

Data Sheet OK Apply Cancel

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

The screenshot shows a schematic editor window titled "Start Page *TopDesign.cysch". On the left, a toolbar contains a wire tool icon, which is highlighted with a blue box. A tooltip for the wire tool is displayed, stating: "Wire Tool (Hot Key: W) Single-click this button to draw one wire at a time. Double-click on the schematic to start or end a wire drawing process. Double-click this button to draw multiple wires at once. Esc to quit drawing wires." The schematic contains a block labeled "PWM_1" with a green "PWM" header and terminals "pwm", "tc", and "pwm". Below it are terminals "clock", "reset", and "interrupt" with a note "8-bit (UDB)". To the left is a "Clock_1" block with a 24 MHz square wave symbol and a terminal "0". A green wire is being drawn from the "clock" terminal of the "Clock_1" block to the "clock" terminal of the "PWM_1" block. The wire is currently in a dashed state, indicating it is being actively drawn.

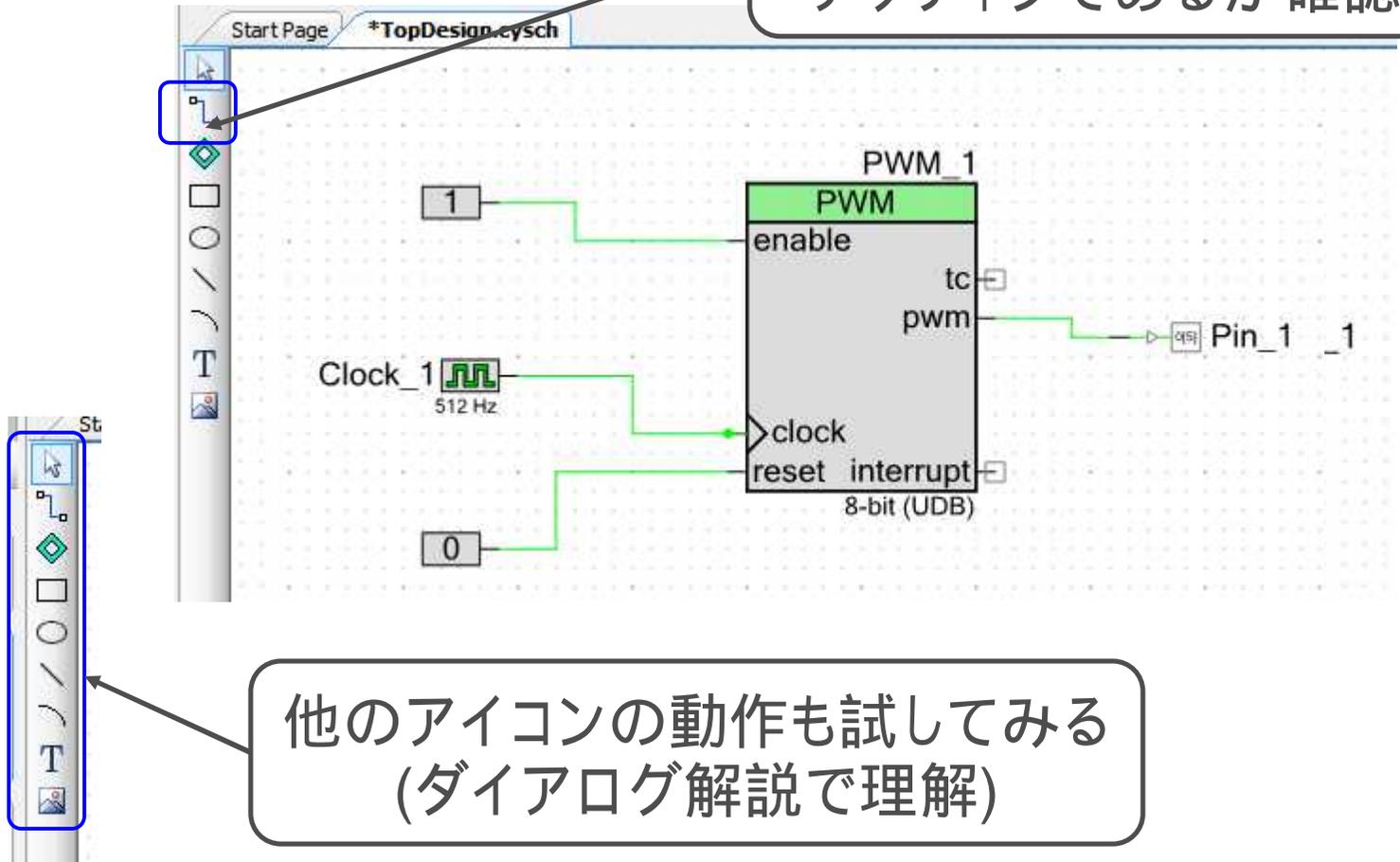
1.配線アイコンをクリック

2.Clock端子の右端に移動し
でクロスカーソル (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

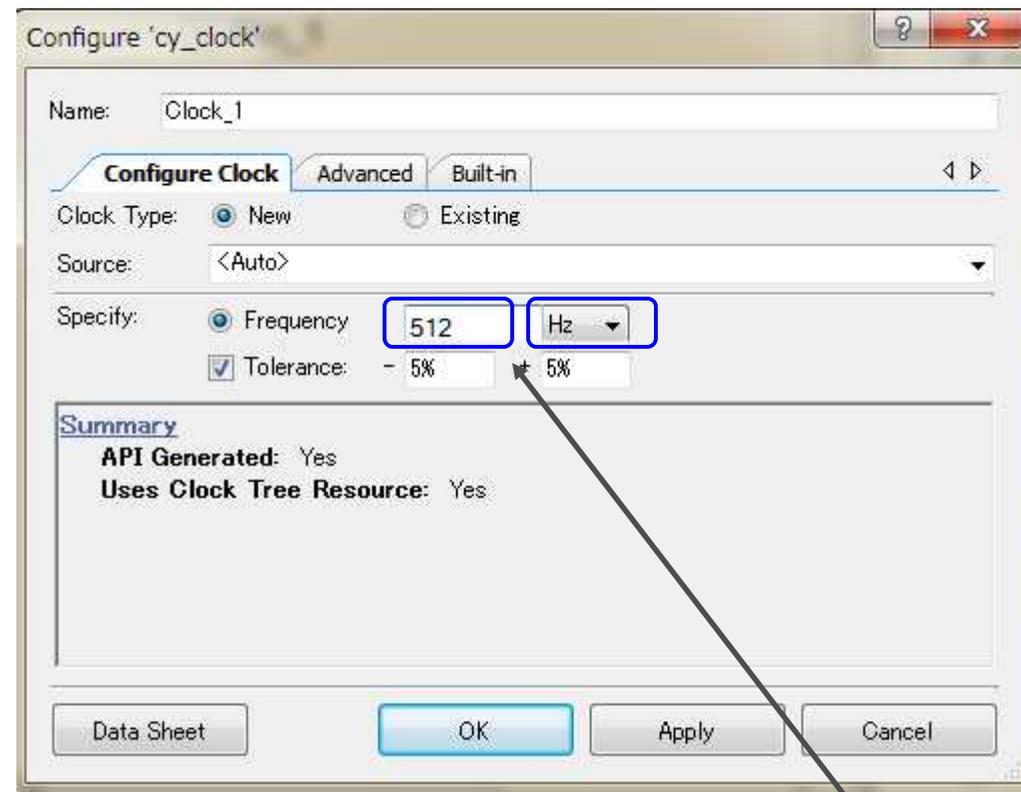
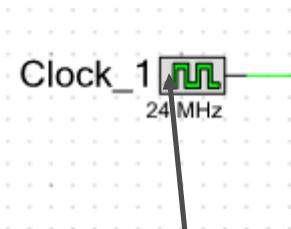
配線中は、配線アイコンがアクティブであるか確認



他のアイコンの動作も試してみる
(ダイアログ解説で理解)

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

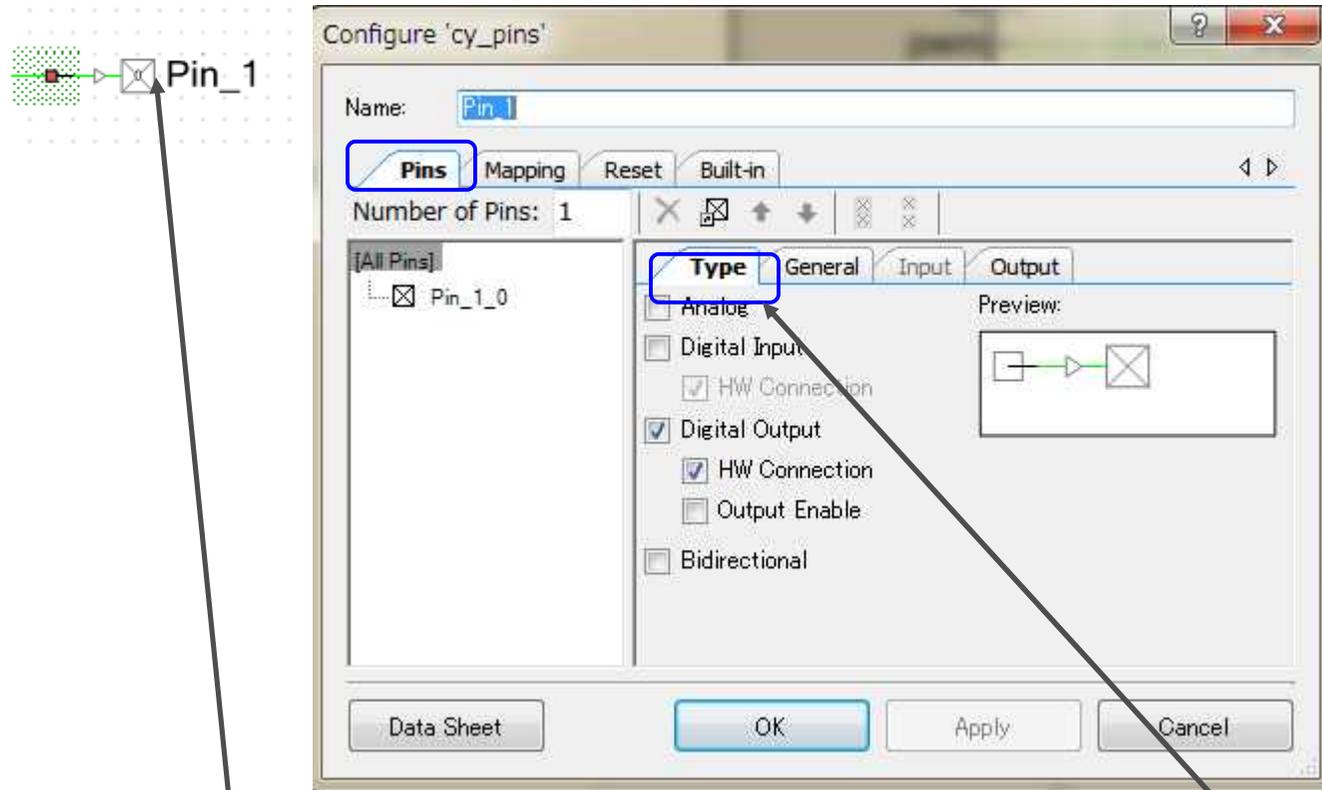


1.シンボルの上にカーソルを置いて
右クリックしてConfigureをクリック

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

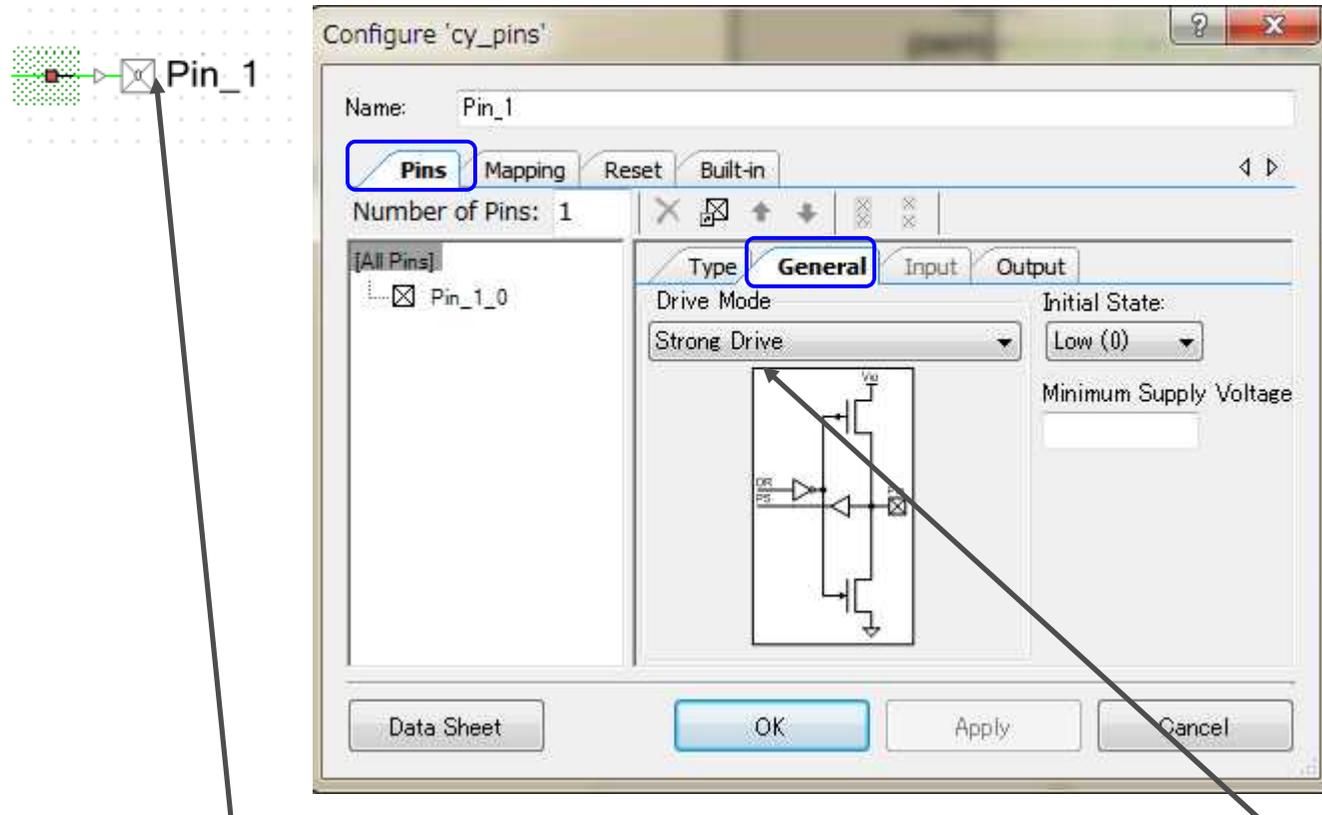


1. シンボルの上にカーソルを置いて
右クリックしてConfigureをクリック

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

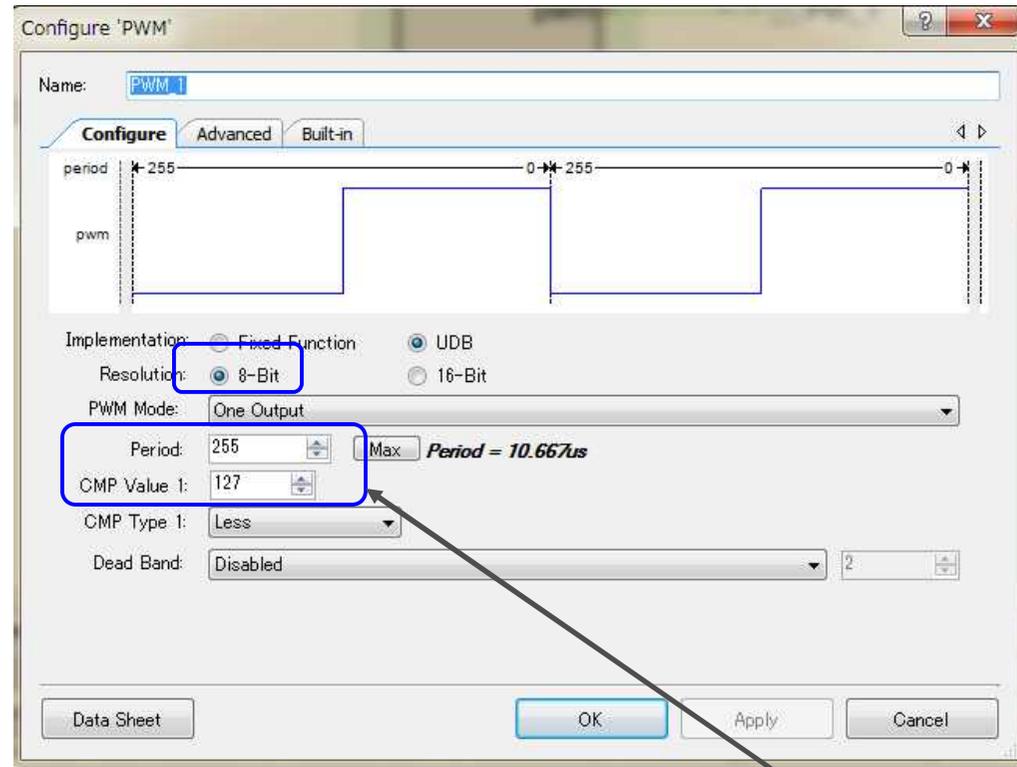
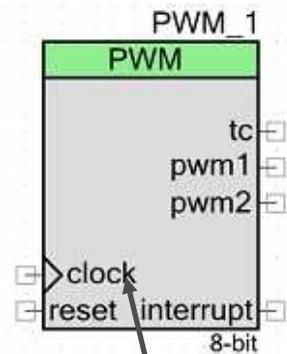


1. シンボルの上にカーソルを置いて
右クリックしてConfigureをクリック

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

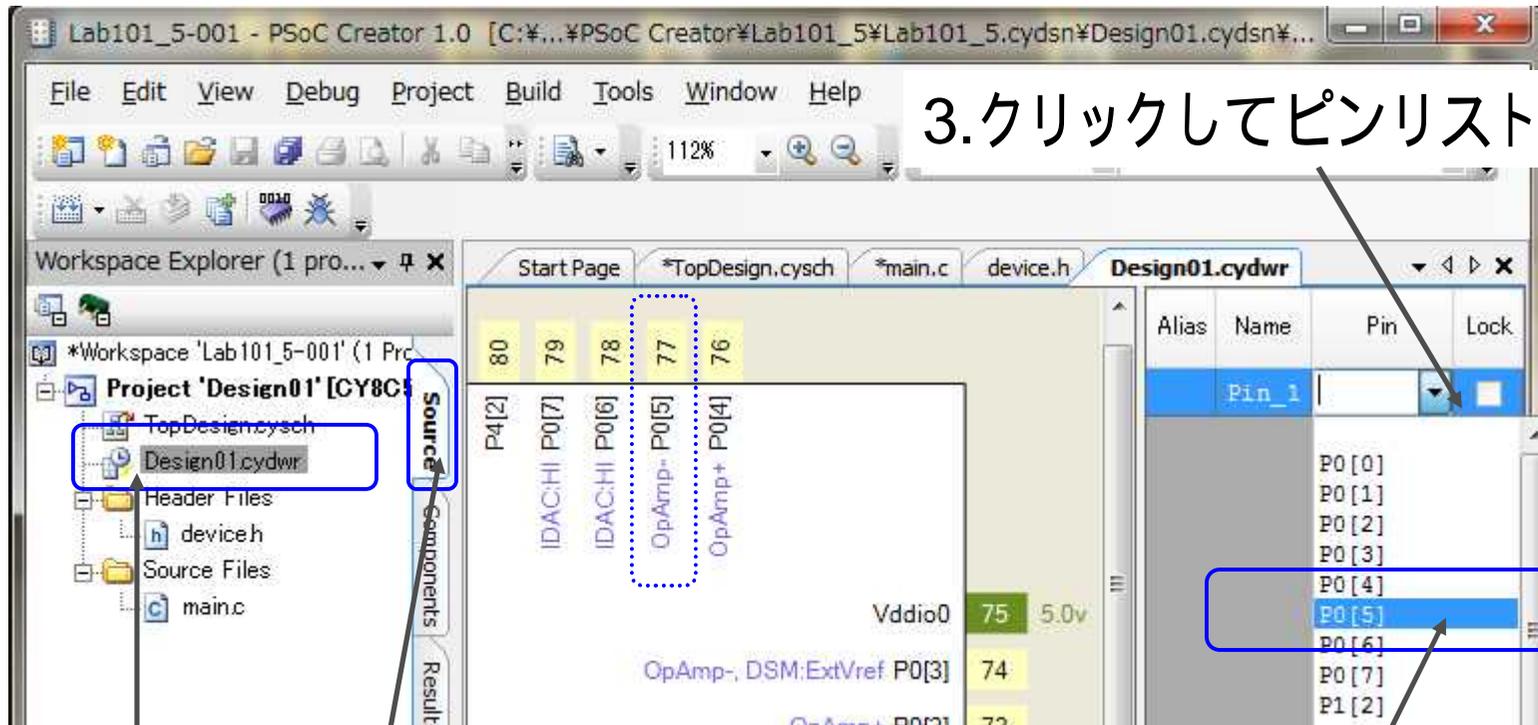
コンポーネントのシンボルをダブルクリックする



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



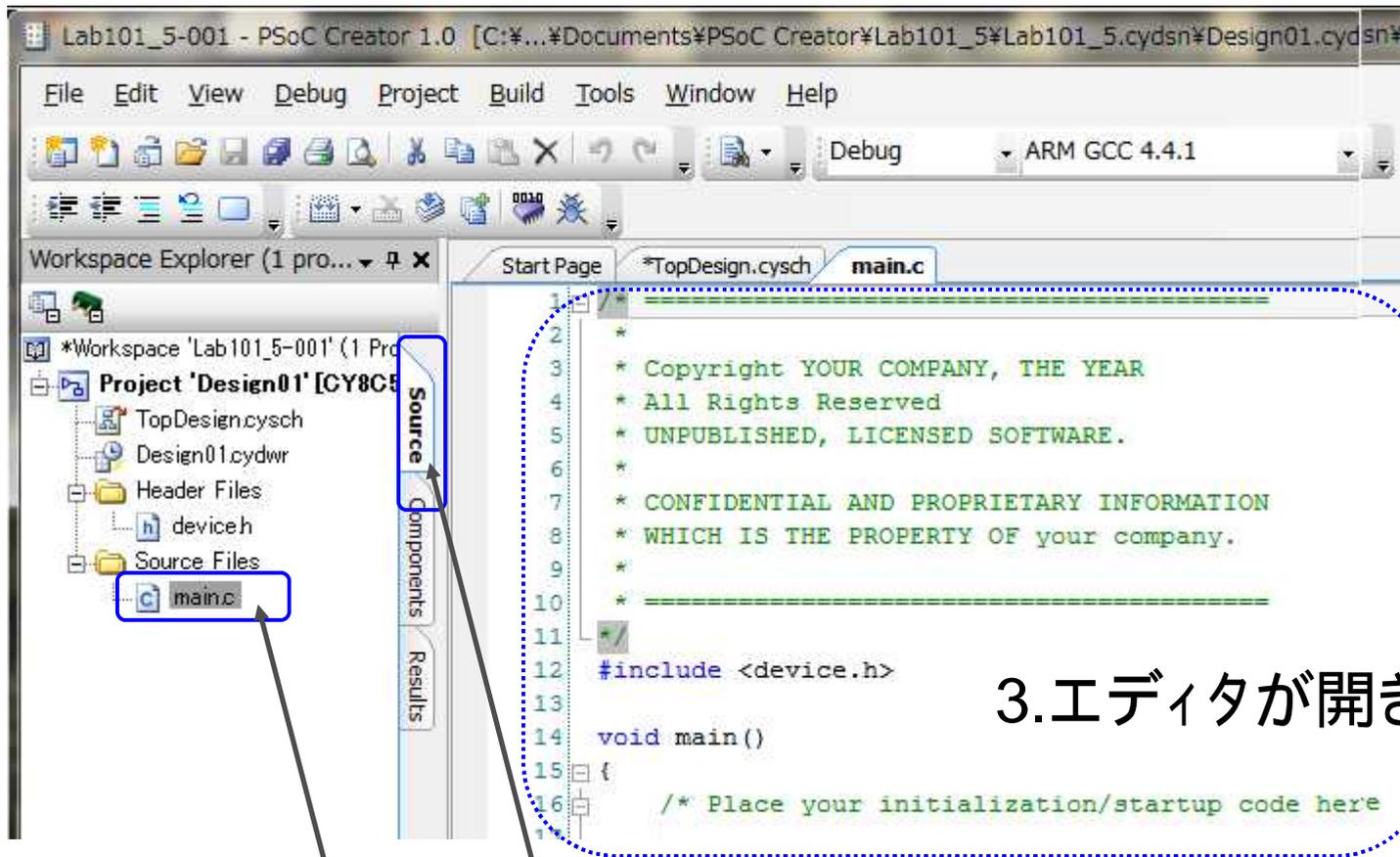
3. クリックしてピンリストを開く

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

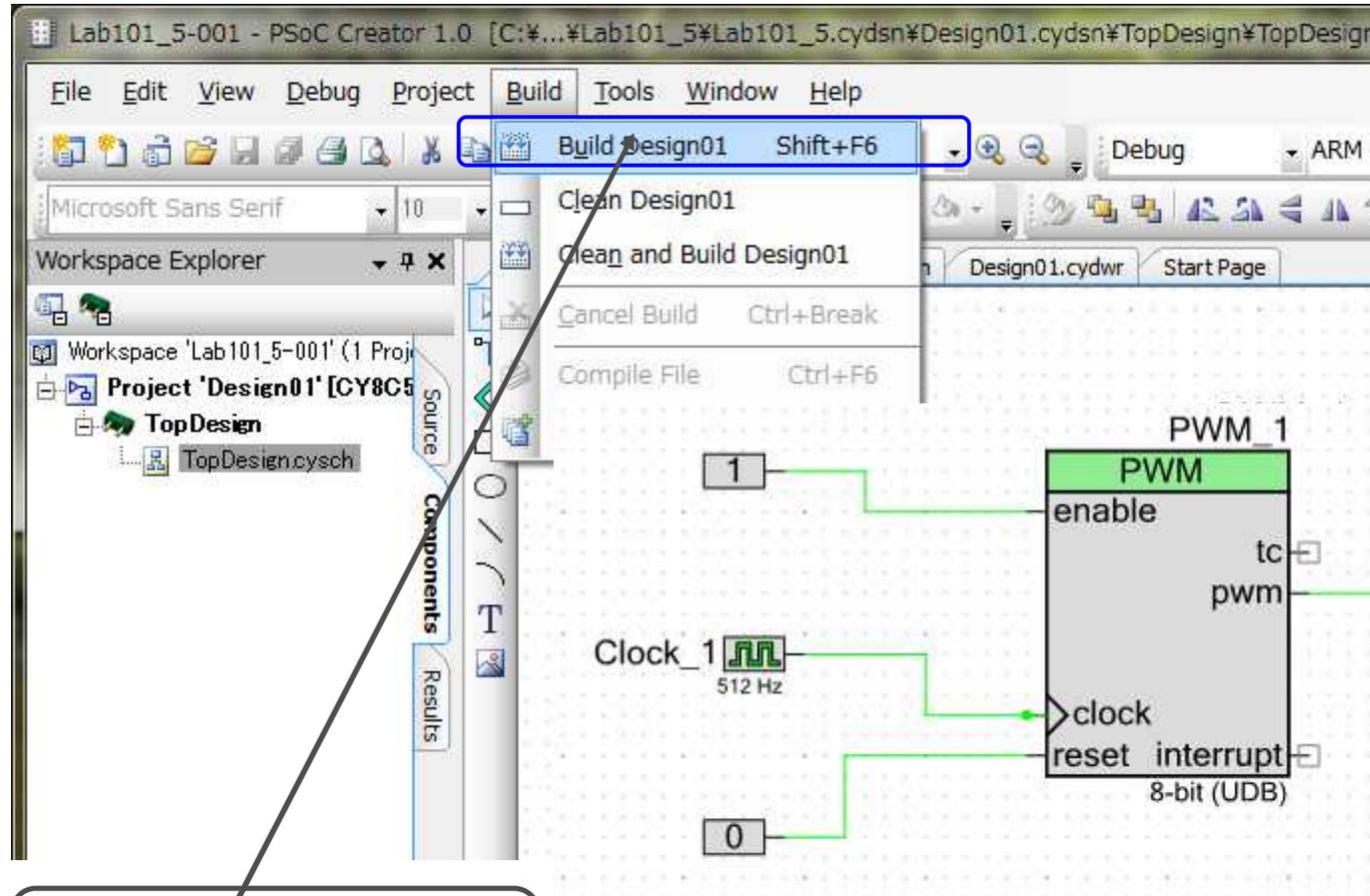
Step9.ソースコードの記述

```
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
```

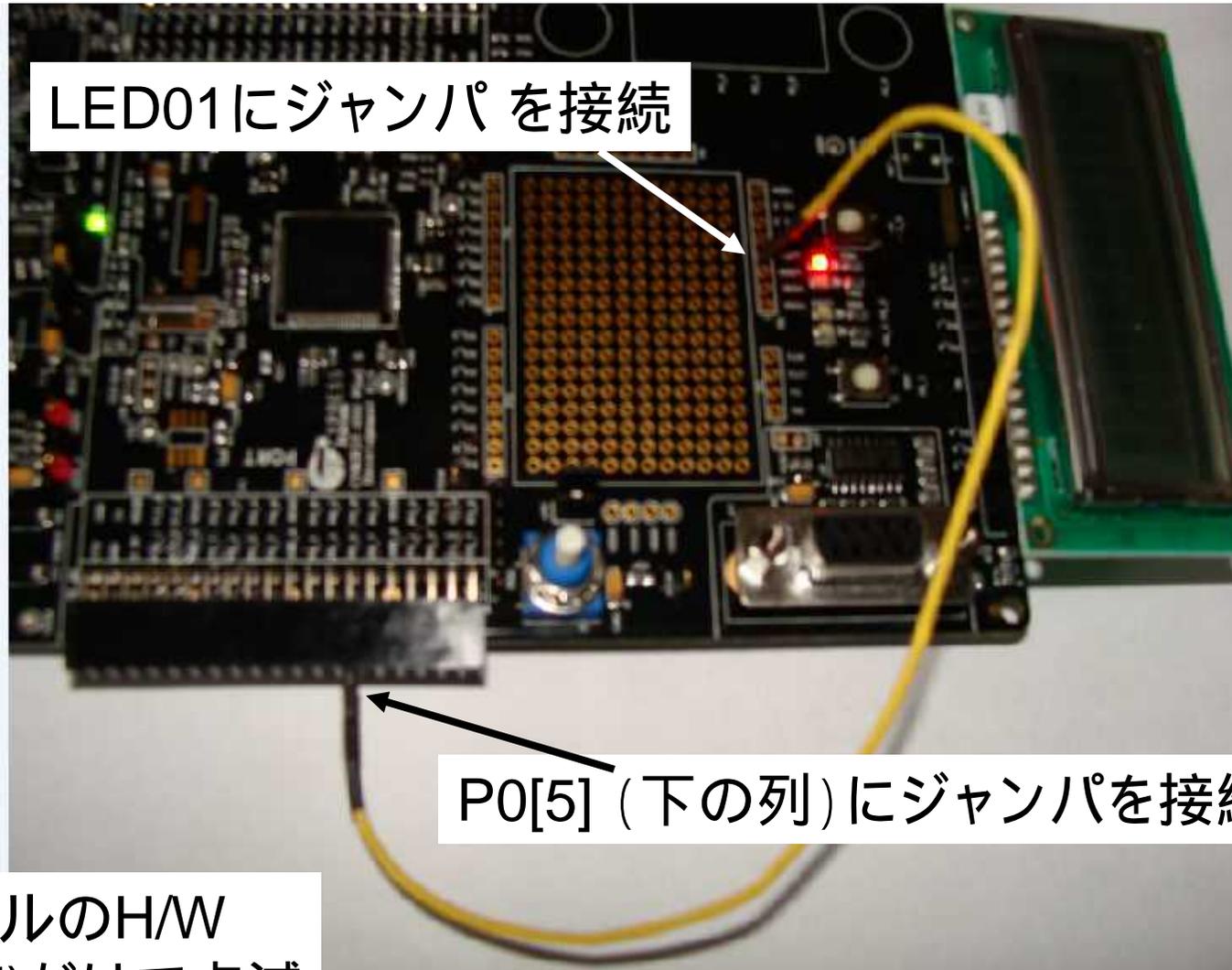
PWM_1_Start(); の記述

Step10.ビルドの実行



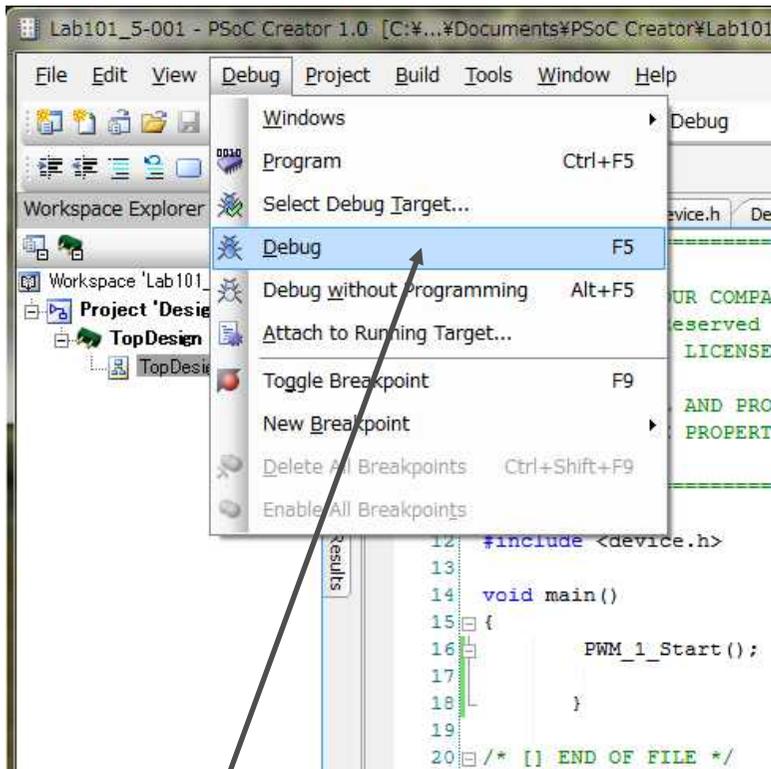
Buildの実行

Step11.動作確認



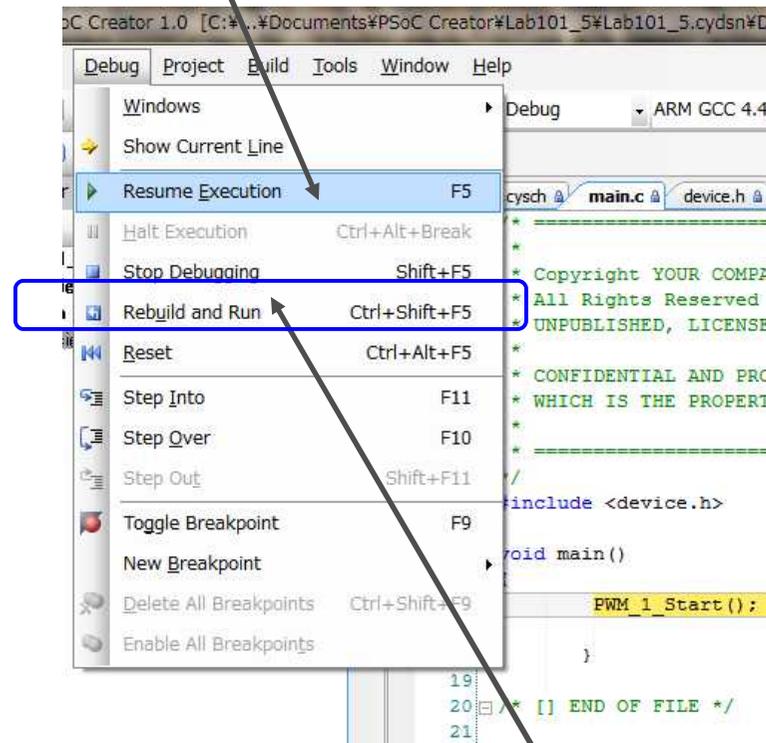
デジタルのH/W
(PWM)だけで点滅

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)

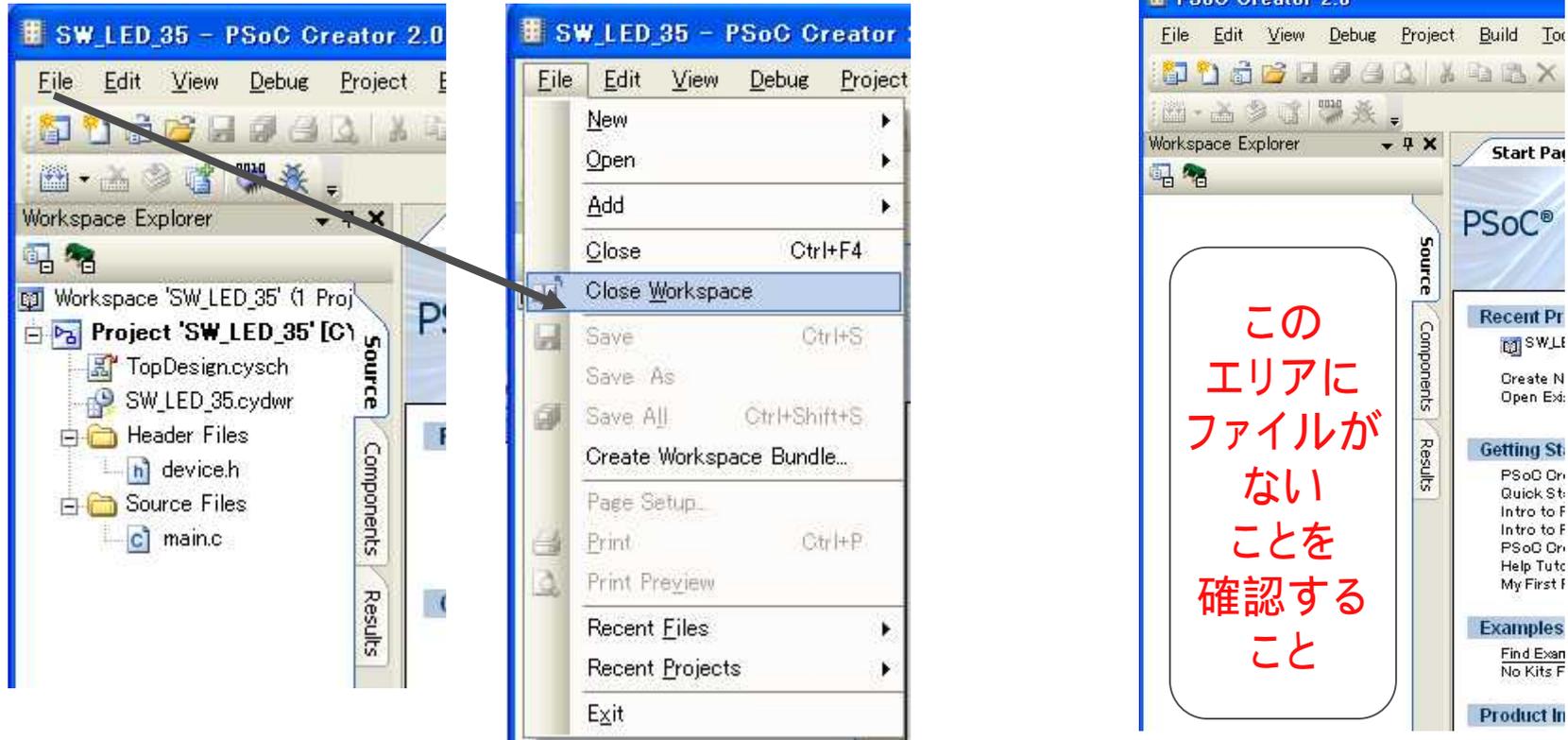
2.実デバッグの開始
(PSoC基板への書込開始)



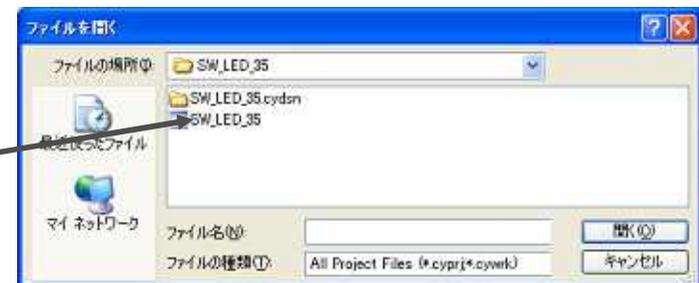
3.デバッグの終了は
Stop Debuggingをクリック

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...>



解説

ビルドのプロセス

API Generation (API生成)

Compilation

Configuration Generation

Configuration Verification

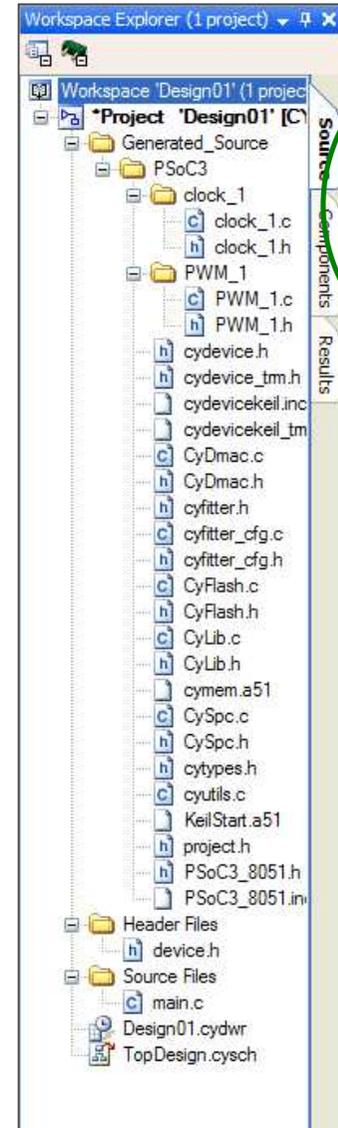
API:Application Program Interface

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



統合されているデバッガ

JTAG and SWD connection

- All devices support debug
- MiniProg3 programmer / debugger



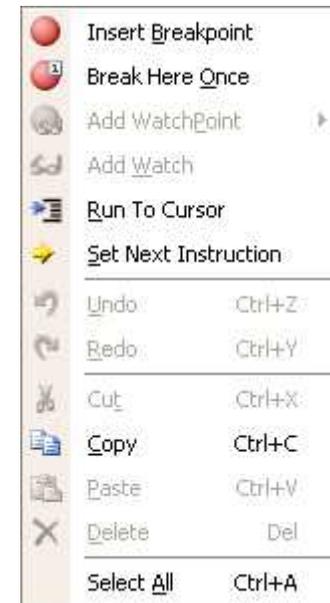
解説

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



| Name | Value | Address | T... | Radix |
|--------|-------|------------------|------|---------|
| period | 0x0 | 000000F4 (XData) | int | Default |
| duty | 0x4 | 000000F6 (XData) | int | Default |

Debugger Windows

The screenshot shows a debugger window titled "Debugging - PSoC Creator 1.0 [C:\CYDev\hello\hello_sw.cydsn\main.c]". The main window displays C code for a delay function and a main function. The code is as follows:

```
17 //-----  
18 // software delay  
19 //-----  
20 void delay(uint8 d)  
21 {  
22     unsigned int i;  
23     if( d > 0 )  
24     {  
25         while( d-- )  
26         {  
27             for( i=0; i < 8000; i++ )  
28             {  
29             }  
30         }  
31     }  
32 }  
33  
34  
35 void main()  
36 {  
37  
38     for(;;)  
39     {  
40         CY_SET_REGS(dPort_1_DR, (CY_GET_REGS(dPort_1_PS) ^ dPort_1_led_MASK));  
41         delay(20);  
42     }  
43 }  
44  
45 /* [] END OF FILE */  
46
```

The debugger interface includes a menu bar (File, Edit, View, Debug, Project, Build, Tools, Window, Help), a toolbar, and a status bar at the bottom showing "Debugging - Halted", "Ln 23 Col 1 INS 0 Errors 0 Warnings 3 Notes".

At the bottom of the debugger window, there are two panels:

- Registers:** A table showing the current state of various registers. The PC register is highlighted with a value of 0x05BD.
- Call Stack:** A table showing the current call stack. The top entry is for the `_delay()` function at line 23 of `main.c`, with address 000005BD. The bottom entry is for the `main()` function at line 41 of `main.c`, with address 000006CB.

A green circle on the right side of the image contains the Japanese characters "解説" (Explanation).

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

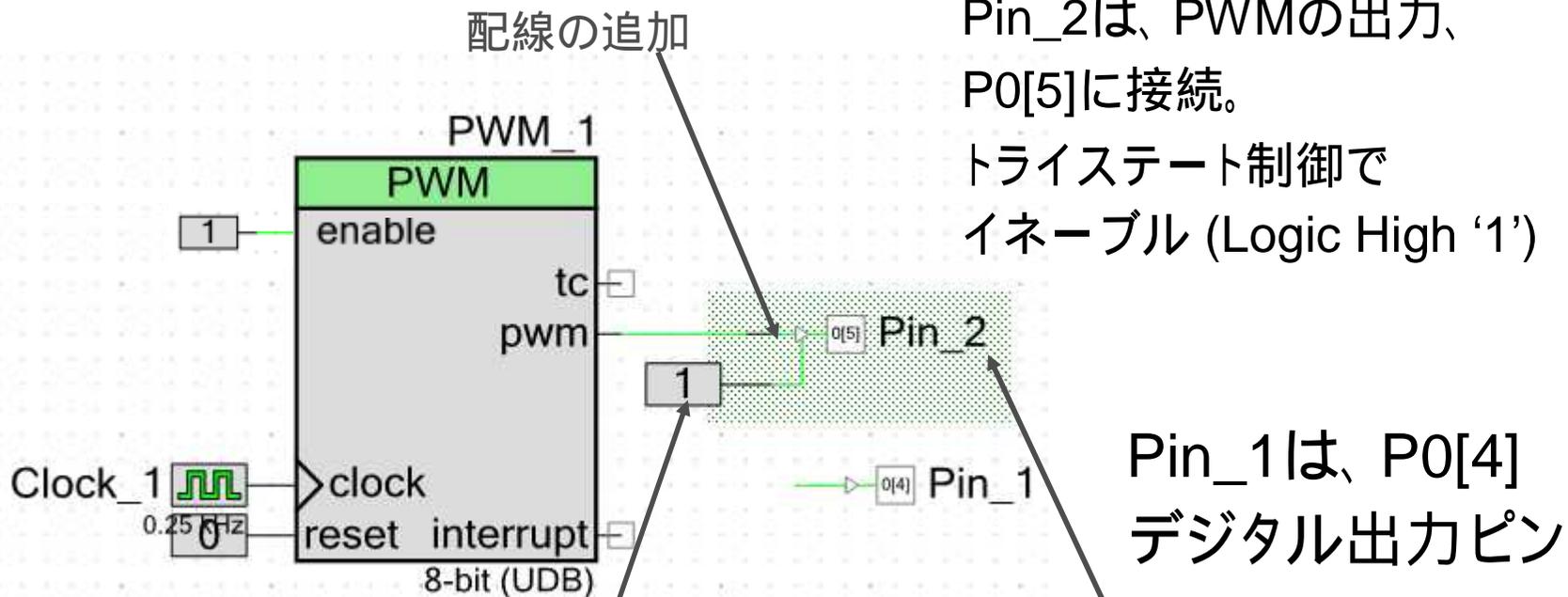
ソフトウェアによるピン制御(Pin_1)と
ハードウェアによる
(Pin_2)トリステート制御の追加で
二通りの方法でLEDを点滅させる

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。
トリステート制御で
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

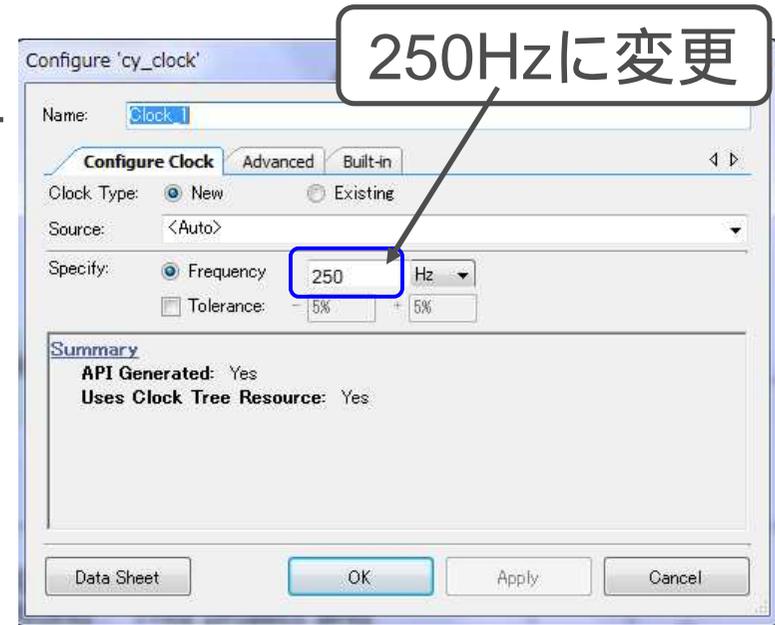
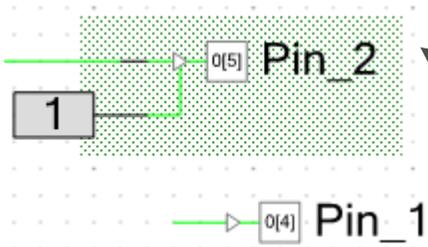


Digital > Logic > Logic High '1'

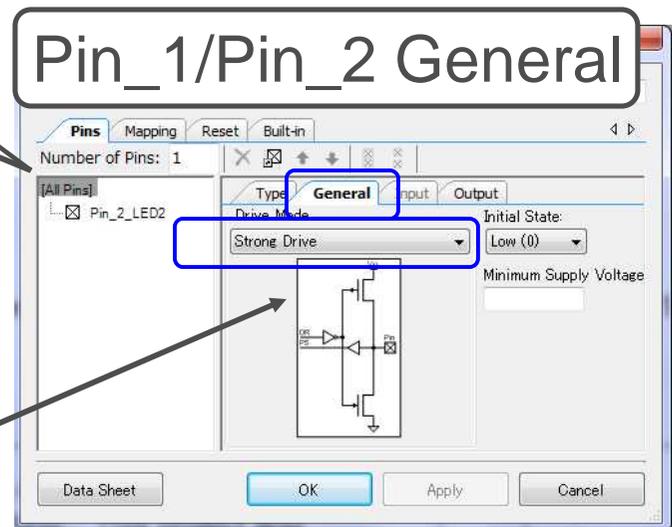
Port and Pins > Digital Output Pin
自動的にピン番号がインクリメント(+1)されます

クロックの変更, Pin_1/_2共通設定

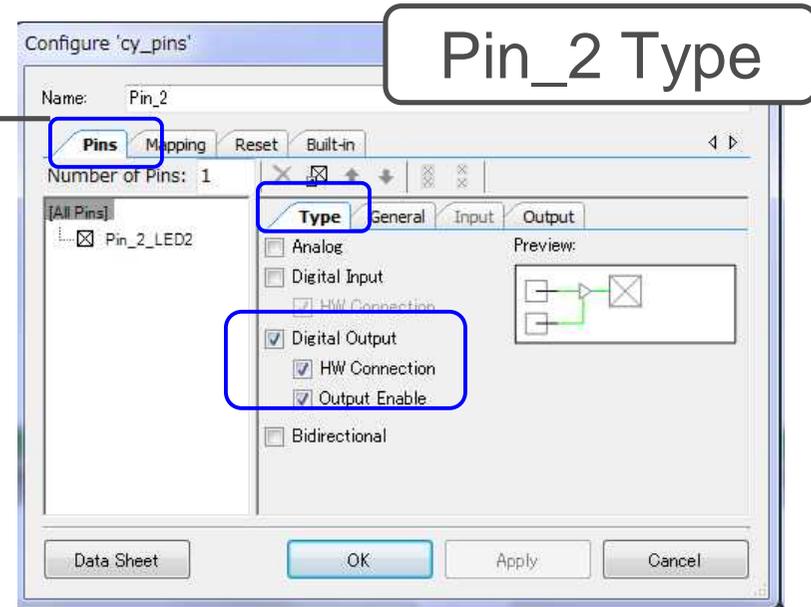
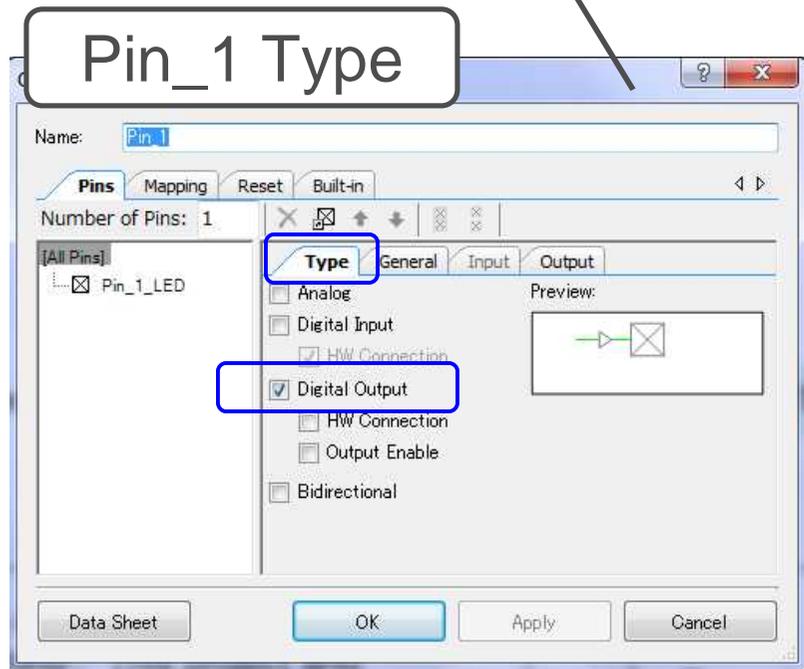
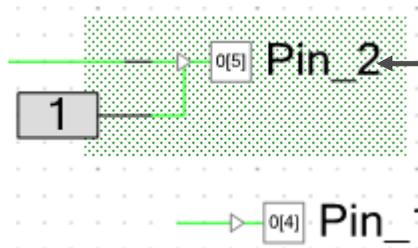
Clock_1 



Pin_1/Pin_2 General
共通の設定



Pin_1/Pin_2のコンフィギュレーション

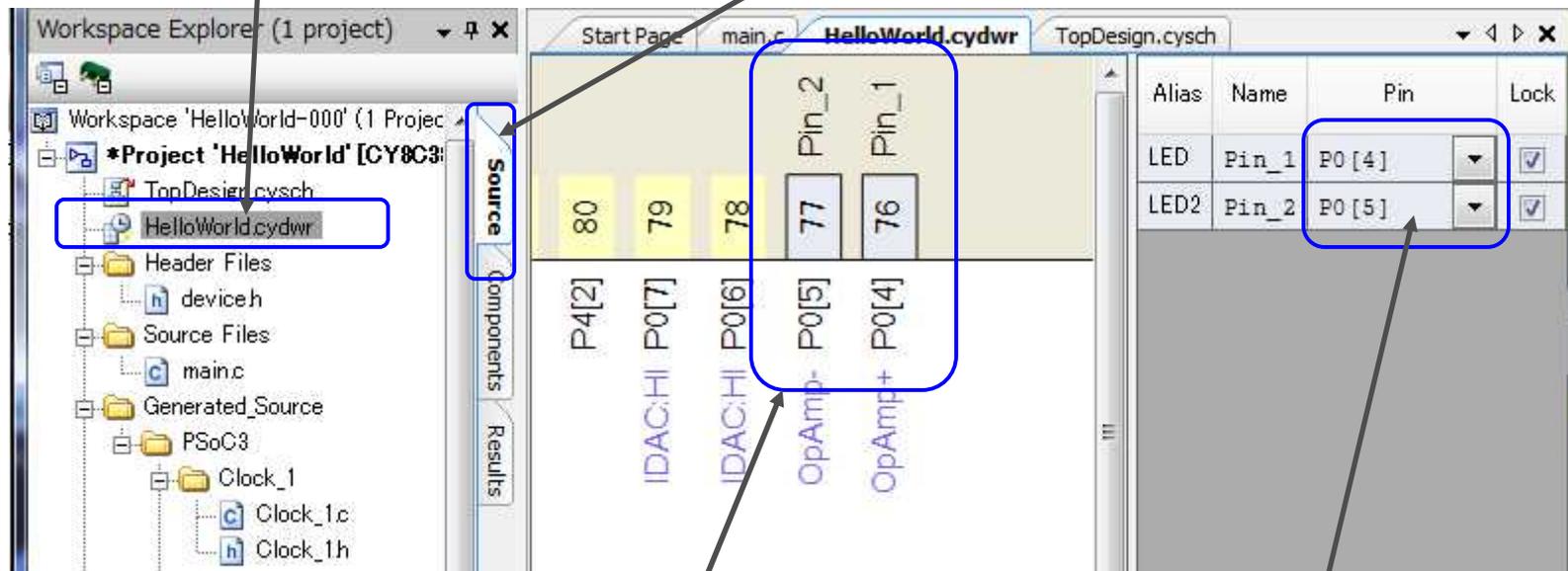


Pin_2は、PWMの出力、
P0[5]に接続。
トリステート制御で
イネーブル (Logic High '1')

ピンの設定

2. .cydwrファイルを
ダブルクリック

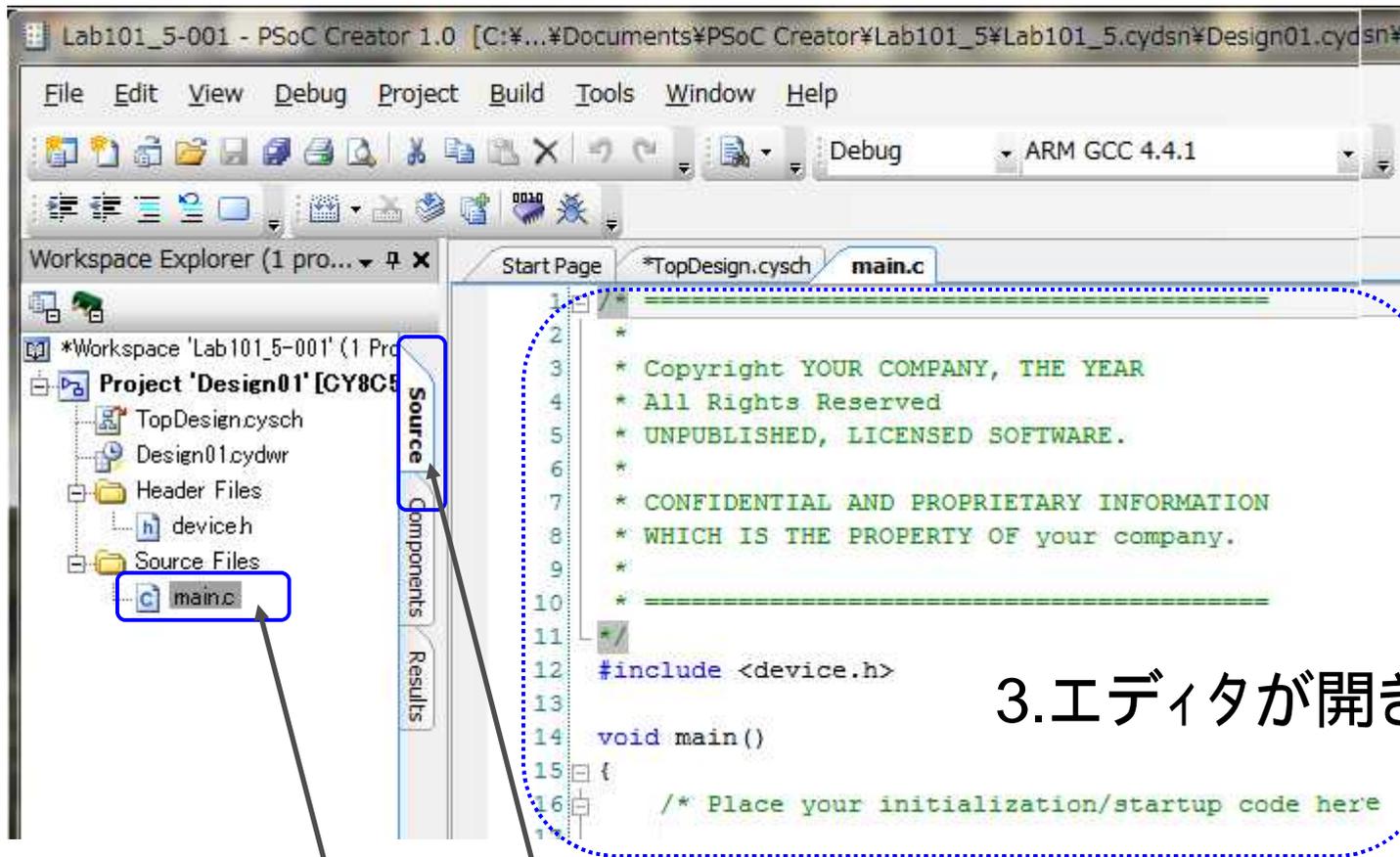
1. Source ウィンドウをクリック



4. Pinアサインの確認

3. Pinのアサイン

ソースコードエディタを開く



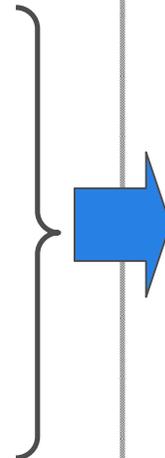
3.エディタが開きます

1.Sourceタブをクリック

2.main.cをダブルクリック

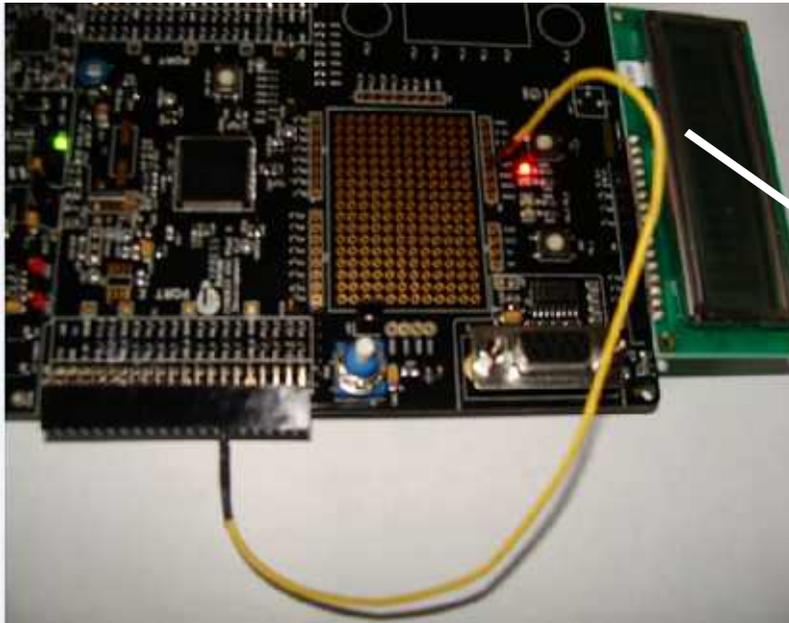
ソースコードの変更

```
Start Page *TopDesign.cysch main.c
1  /**
2  *
3  * Copyright YOUR COMPANY, THE Y
4  * All Rights Reserved
5  * UNPUBLISHED, LICENSED SOFTWARE
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     PWM_1_Start(); /
17
18 }
19
20 /* [] END OF FILE */
21
```



```
32 void delay(uint8 d)
33 {
34     unsigned int i;
35     if( d > 0 )
36     {
37         while( d-- )
38         {
39             for( i=0; i < 8000; i++ )
40             {
41             }
42         }
43     }
44 }
45
46 void main()
47 {
48     PWM_1_Start(); /* Default PWM name */
49
50     for(;;)
51     {
52         if( CyPins_ReadPin( Pin_1_LED ) )
53         {
54             CyPins_ClearPin( Pin_1_LED );
55         }
56         else
57         {
58             CyPins_SetPin( Pin_1_LED );
59         }
60         delay(20);
61     }
62 }
63
64 /* [] 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。
トリステート制御で
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 Creator 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ており、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)