EPSON
EXCEED YOUR VISION

S1C17 Family Application Note

# S1C17589
# 周辺回路サンプルソフトウェア

セイコーエプソン株式会社

Rev.1.2

評価ボード・キット、開発ツールご使用上の注意事項

1. 本評価ボード・キット、開発ツールは、お客様での技術的評価、動作の確認および開発のみに用いられることを想定し設計されています。それらの技術評価・開発等の目的以外には使用しないで下さい。本品は、完成品に対する設計品質に適合していません。

2. 本評価ボード・キット、開発ツールは、電子エンジニア向けであり、消費者向け製品ではありません。お客様において、適切な使用と安全に配慮願います。弊社は、本品を用いることで発生する損害や火災に対し、いかなる責も負いかねます。通常の使用においても、異常がある場合は使用を中止して下さい。

3. 本評価ボード・キット、開発ツールに用いられる部品は、予告無く変更されることがあります。

本資料のご使用につきましては、次の点にご留意願います。

本資料の内容については、予告無く変更することがあります。

1. 本資料の一部、または全部を弊社に無断で転載、または、複製など他の目的に使用することは堅くお断りいたします。

2. 本資料に掲載される応用回路、プログラム、使用方法等はあくまでも参考情報であり、これらに起因する第三者の知的財産権およびその他の権利侵害あるいは損害の発生に対し、弊社はいかなる保証を行うものではありません。また、本資料によって第三者または弊社の知的財産権およびその他の権利の実施権の許諾を行うものではありません。

3. 特性値の数値の大小は、数直線上の大小関係で表しています。

4. 製品および弊社が提供する技術を輸出等するにあたっては「外国為替および外国貿易法」を遵守し、当該法令の定める手続きが必要です。大量破壊兵器の開発等およびその他の軍事用途に使用する目的をもって製品および弊社が提供する技術を費消、再販売または輸出等しないでください。

5. 本資料に掲載されている製品は、生命維持装置その他、きわめて高い信頼性が要求される用途を前提としていません。よって、弊社は本（当該）製品をこれらの用途に用いた場合のいかなる責任についても負いかねます。

6. 本資料に掲載されている会社名、商品名は、各社の商標または登録商標です。

# 目　次

# 1. 概要

本マニュアルはS1C17589向けのサンプルソフトウェアの使い方とサンプルソフトウェアの動作について記載しています。

S1C17589 サンプルソフトウェアは S1C17589 マイコンに内蔵されている各周辺回路の使用例を示すことを目的としています。

各機種情報、各テクニカルマニュアル、S5U1C17001C Manual と合わせてご覧下さい。

## 1.1 動作環境

S1C17589 サンプルソフトウェアを動作させるにあたり、以下の機材をご用意下さい。

- S1C17589 の実装されたボード
- S5U1C17001H（以下 ICDmini とします。）
- S5U1C17001C（以下 GNU17 とします。）

注　本サンプルソフトウェアは、GNU17v2.3.0/v3.0.5 で動作確認を行っています。

注　GNU17 及び ICDmini には、複数のバージョンが存在します。ご使用になる GNU17 で使用可能な ICDmini については S5U1C17001C Manual を参照して下さい。

# 2. サンプルソフトウェア構成

本章では S1C17589 サンプルソフトウェアのファイル構成を記載します。

S1C17589 サンプルソフトウェアは各周辺回路の動作を確認する“サンプルソフトウェア”と、各周辺回路のサンプルドライバである“サンプルドライバ”から成ります。

## 2.1 ディレクトリ構成及びファイル構成

以下に S1C17589 サンプルソフトウェアのディレクトリ構成を示します。

```
[s1c17589_xxx_sample]
    |
    ├── [s1c17589_xxx_gnu17v2] : サンプルソフト(GNU17 version 2 プロジェクト)
    |     ├── GNU17プロジェクトファイルなど
    |     ├── ...
    |     ├── [inc] : s1c17589ヘッダファイル
    |     |     ├── [reg] : s1c17589 周辺回路ヘッダファイル
    |     |     |     ├adc10a_reg.h
    |     |     |     ├...
    |     |     |     └wdt_reg.h
    |     |     |
    |     |     └── c17589_reg.h
    |     |
    |     └── [src] : ソースコード・ヘッダファイル
    |           ├── boot.c
    |           ├── init.c
    |           ├── init.h
    |           ├── ...
    |           └── main.c
    |
    └── [s1c17589_xxx_gnu17v3] : サンプルソフト(GNU17 version 3 プロジェクト)
          ├── GNU17プロジェクトファイルなど
          ├── ...
          ├── [inc] : s1c17589ヘッダファイル
          |     ├── [reg] : s1c17589 周辺回路ヘッダファイル
          |     |     ├adc10a_reg.h
          |     |     ├...
          |     |     └wdt_reg.h
          |     |
          |     └── c17589_reg.h
          |
          └── [src] : ソースコード・ヘッダファイル
                ├── boot.c
                ├── init.c
                ├── init.h
                ├── ...
                └── main.c
```

図 2.1.1 S1C17589 サンプルソフトウェアディレクトリ構成図

（1）s1c17589_xxx_sample ディレクトリ

サンプルソフトが格納されているディレクトリです。

（2）s1c17589_xxx_gnu17vx ディレクトリ

GNU17 のプロジェクトに関係するファイルが格納されているディレクトリです。

ディレクトリ名の最後の_gnu17v2/_gnu17v3 が GNU17 のバージョンを示しています。

（3）inc ディレクトリ

S1C17589 のレジスタなど機種に依存する情報を定義したヘッダファイルが格納されているディレクトリです。

- S1C17589 のレジスタアドレスなどを定義したヘッダファイル（c17589_reg.h）

（4）reg ディレクトリ

周辺回路毎のビットアサインなどを定義したヘッダファイが配置してあります。

- 各周辺回路のビットアサインなどを定義したヘッダファイル(clg_589_reg.h など)

（5）src ディレクトリ

マイコンの初期化処理、周辺回路毎のサンプルソフトウェア及び、サンプルドライバ、またそれらで使用する定数等を定義したヘッダファイルが配置してあります。

- ベクターテーブルや初期化処理のファイル（boot.c）
- 周辺回路毎のサンプルソフトウェアファイル（main.c）
- 周辺回路毎のサンプルドライバファイル（init.c や clg.c など）
- 周辺回路毎のサンプルドライバのヘッダファイル（init.h や clg.h など）

# 3. 実行方法

## 3.1 ソフトウェア開発環境

ソフトウェアの開発環境について記載します。本書では、以下のツールを使用した方法について記載します。各ツールの詳細については、各ツールのマニュアルを参照して下さい。

- S5U1C17001C (GNU17) ver. 2.3.0/ver.3.0.5
- 機種別情報ファイル（S1C17589）
- S5U1C17001H (ICDmini) ver. 2.0/ver 3.0

## 3.2 各種ツールのインストール

（１）GNU17 viersion 2 / version 3 (S5U1C17001C)

- GNU17 の圧縮ファイルをエプソン マイコン ユーザーズサイトよりダウンロードして下さい。
  https://www.epsondevice.com/support/mcu/product/c17.html
- 入手した GNU17 の圧縮ファイルを任意のフォルダに解凍して下さい。
- Setup.exe を実行し、インストールを行ってください。

（２）機種別情報ファイルのインストール

- 各機種の機種別情報ファイルをエプソン マイコン ユーザーズサイトよりダウンロードして下さい。最新の機種別情報ファイルは、エプソン マイコン ユーザーズサイトのご確認をお願い致します。
  https://www.epsondevice.com/support/mcu/product/page04.html#02
- 入手した機種別情報ファイルを、GNU17 をインストールしたフォルダの[mcu_model]フォルダに解凍して下さい。
  例：

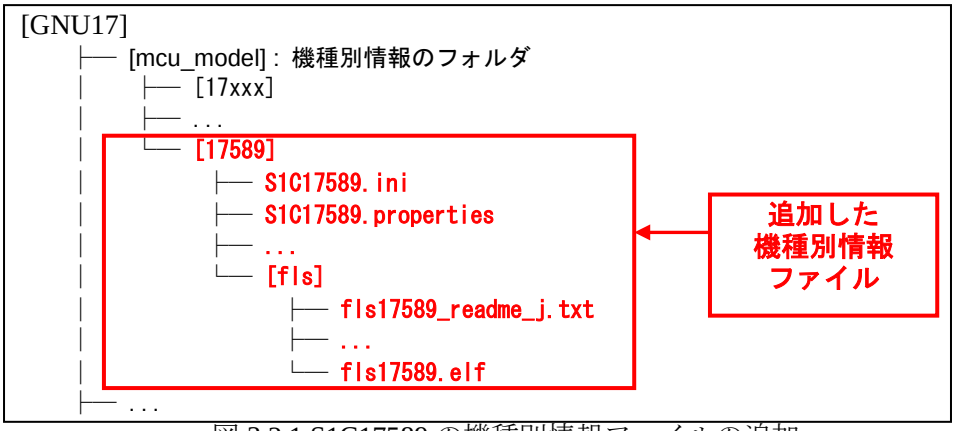



図 3.2.1 S1C17589 の機種別情報ファイルの追加

（３）ICDmini ver2.0 (S5U1C17001H)

- GNU17 version 2 をインストール後、ICDmini と PC を USB ケーブルにて接続して下さい。
- 以下の場所から、USB ドライバーをインストールして下さい。USB ドライバーは、GNU17 をインストールしたフォルダにあります。以下に GNU17 を C:¥EPSON¥GNU17 にインストールした場合のフォルダ構成を記載します。
  32bit OS の場合：C:¥EPSON¥GNU17¥utility¥drv_usb¥OS_32bit

  64bit OS の場合：C:¥EPSON¥GNU17¥utility¥drv_usb¥OS_64bit

（４）ICDmini ver3.0 (S5U1C17001H)

- GNU17 version 3 をインストール後、ICDmini と PC を USB ケーブルにて接続して下さい。
- 以下の場所から、USB ドライバーをインストールして下さい。USB ドライバーは、GNU17 をインストールしたフォルダにあります。GNU17 を C:¥EPSON¥GNU17V3 にインストールした場合には、以下のフォルダとなります。
  C:¥EPSON¥GNU17V3¥utility¥drv_usb¥Mini3Driver

## 3.3 プロジェクトのインポート

サンプルプログラムを GNU17 にインポートする方法について記載します。

注　プログラムのインポートの詳細については、S5U1C17001C Manual を参照して下さい。

注　GNU17 version 2 と GNU17 version 3 でプログラムのインポート方法が異なります。

### 3.3.1 サンプルプログラムのインポート（GNU17 version 2 / version 3 共通）

（1）GNU17 の起動

GNU17 をインストールしたディレクトリ内の eclipse ディレクトリにある、eclipse.exe のアイコンをダブルクリックし、IDE を起動させます。Windows のスタートアップメニューから[EPSON MCU] >[GNU17] > [GNU17 IDE]（GNU17 version 3 の場合、[EPSON MCU] >[GNU17V3] > [GNU17V3 IDE]）を選択しても起動します。

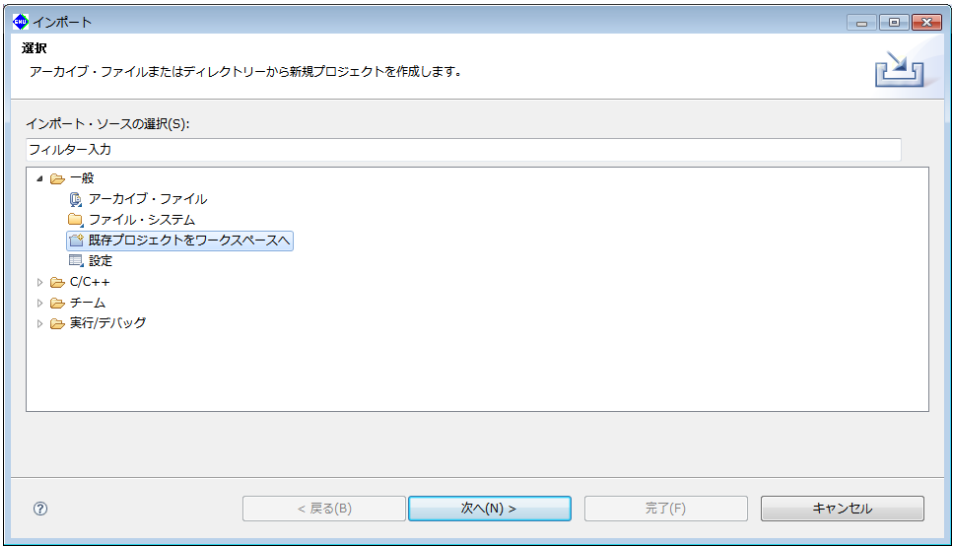
（2）サンプルプログラムの解凍

エプソン マイコン ユーザーズサイトよりダウンロードしたサンプルプログラムを解凍します。

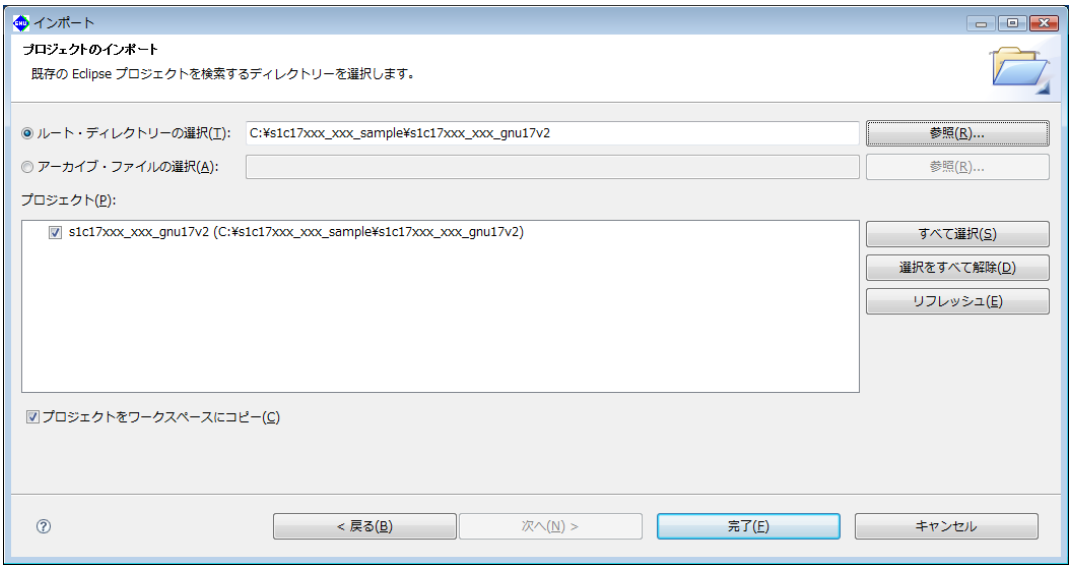
### 3.3.2 サンプルプログラムのインポート（GNU17 version 2）

（1）サンプルプログラムのインポート

GNU17 の[ファイル]メニューから[インポート...]を選択します。[インポート]ウィザードから、[一般]>[既存プロジェクトをワークスペースへ]を選択します。




[ルート・ディレクトリーの選択]の[参照...]ボタンで、3.3.1 の（2）で解凍したサンプルプロジェクトのフォルダ内の [xxx_gnu17v2] プロジェクトを選択します。

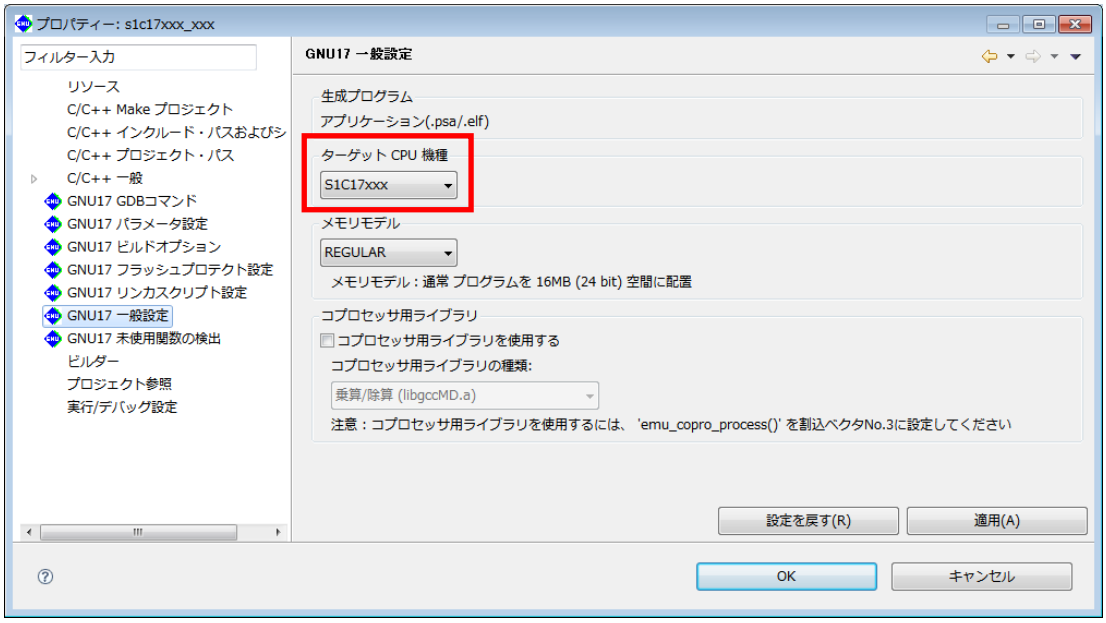



[プロジェクトをワークスペースにコピー]チェックボックスを選択します。これにより、プロジェクトのコピーがワークスペースディレクトリに作成され、オリジナルが変更されることはありません。最後に、[完了]ボタンを押します。GNU17 にて、サンプルプロジェクトがインポートされたことを確認後、3.3.1 の（2）で解凍したフォルダは削除して下さい。

（2）ターゲットの変更

サンプルプログラムのターゲットを変更します。サンプルプログラムのプロジェクトを[C/C++プロジェクト]ビューで選択し、 [プロジェクト]メニューから[プロパティ]を選択します。

[プロパティ]ダイアログから、[GNU17 一般設定]を選択します。

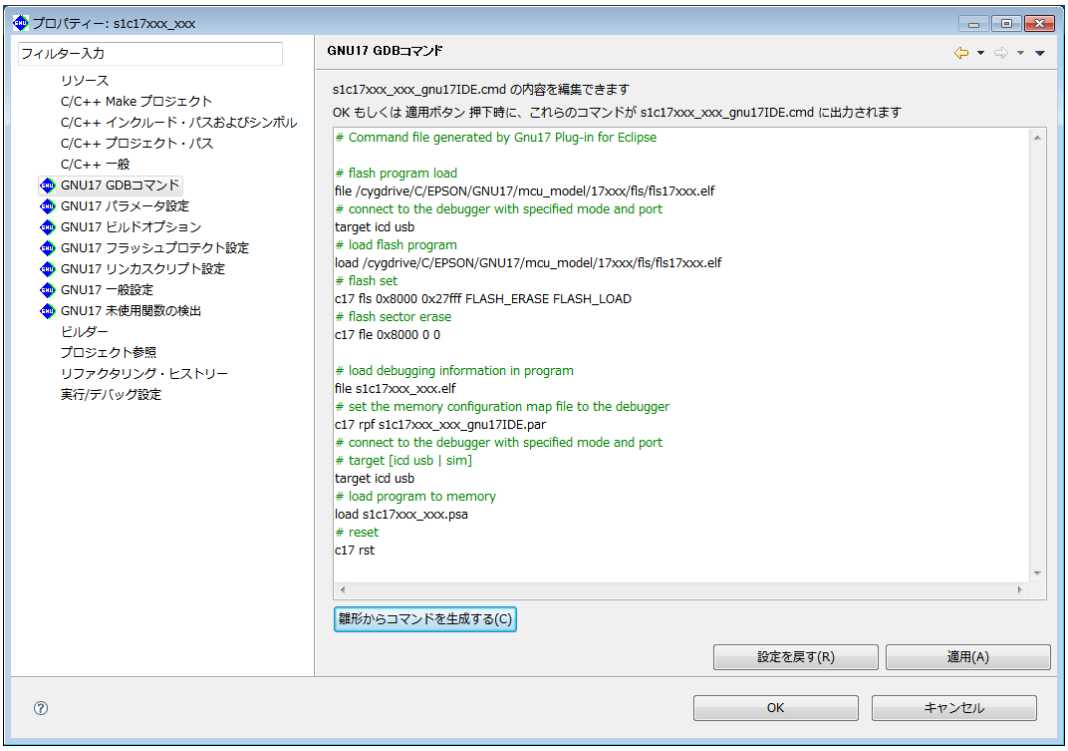



[ターゲット CPU 機種]から、ターゲットとなる CPU を選択し、[OK]ボタンを押して下さい。

ターゲットとなる CPU が S1C17589 の場合、[S1C17589]に変更して下さい。

（3）デバッガ起動オプションの変更

デバッガ起動オプションを変更します。サンプルプログラムのプロジェクトを[C/C++プロジェクト]ビューで選択し、 [プロジェクト]メニューから[プロパティ]を選択します。[プロパティ]ダイアログから、[GNU17 GDB コマンド]を選択します。

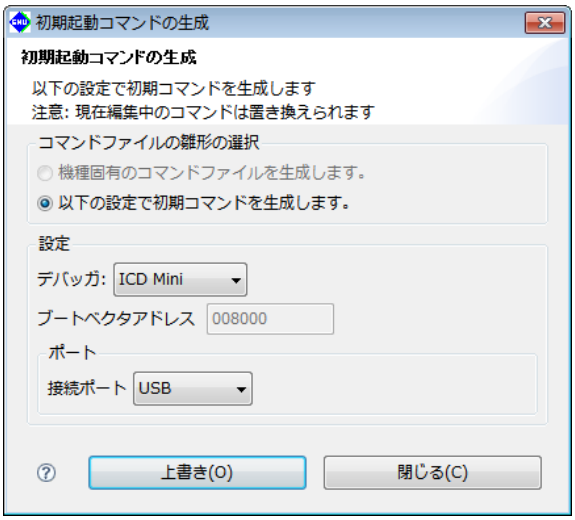



ここには、IDE が生成するデバッガ起動用コマンドファイルの内容が表示されます。デバッガは使用するマイコンや ICD などに合わせたモードに設定して動作させる必要があります。

デバッガ起動用コマンドファイルは、以下の手順により作成を行います。

[雛形からコマンドを生成する]ボタンをクリックして、[初期起動コマンドの生成]ダイアログボックスを表示させます。

[デバッガ:]コンボボックスから"ICD Mini"を選択します。




[上書き]ボタンをクリックし、その後表示される[現在のコマンドを上書き]ダイアログボックスでも[OK]ボタンをクリックします。

コマンドの修正や追加が必要な場合は、テキストボックス内で直接編集が可能です。

※：シミュレーテッド I/O を利用する場合の注意

シミュレーテッド I/O（シリアルインタフェースなどの外部入出力機能を標準入出力（stdin、stdout）あるいはファイルの入出力により評価すること）を使用する場合は、GDB コマンドの最後に以下の行を追加して下さい。

```
# 最後の行「c17 rst」の後に、以下の 3 行を追加する
u main
c17 stdout 1 WRITE_FLASH WRITE_BUF
c17 stdin 1 READ_FLASH READ_BUF
```

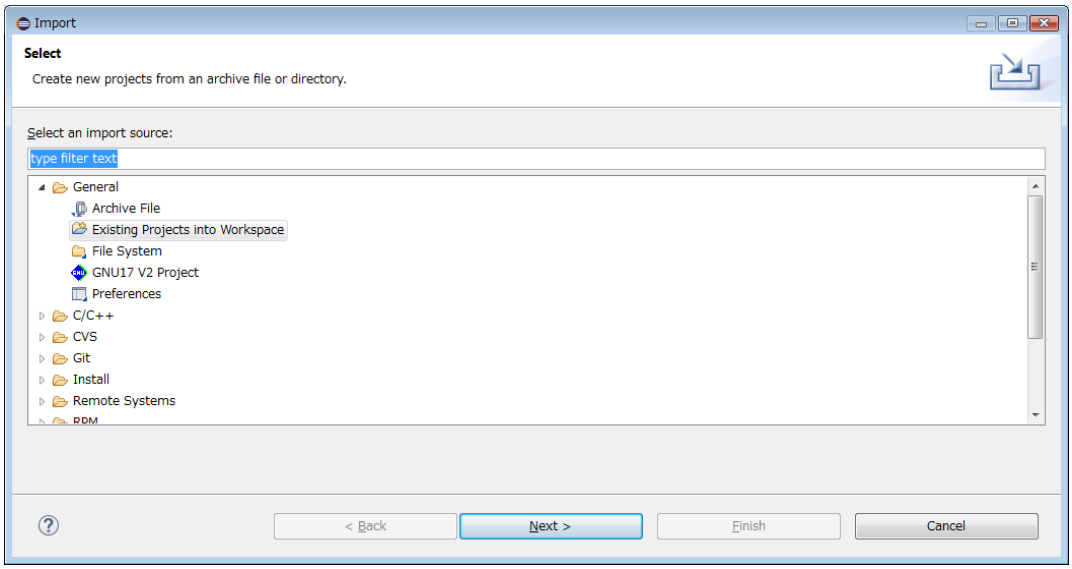
シミュレーテッド I/O を使用するサンプルプログラムは、以下となります。

- クロックジェネレータ（CLG）
- 16 ビット PWM タイマ（T16B）

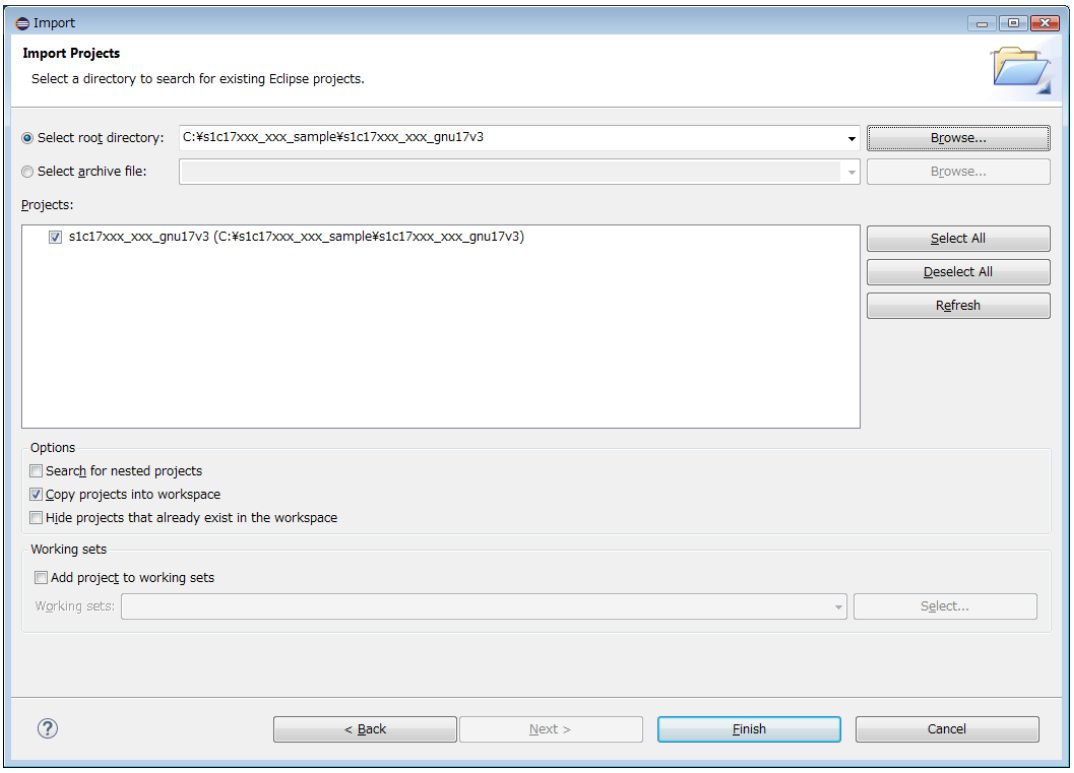
### 3.3.3 サンプルプログラムのインポート（GNU17 version 3）

（1）サンプルプログラムのインポート

GNU17 の[File]メニューから[Import...]を選択します。[Import]ウィザードから、[General]>[Existing Projects into Workspace]を選択します。



[Select root directory]の[Browse...]ボタンで、3.3.1 の（２）で解凍したサンプルプロジェクトのフォルダ内の [xxx_gnu17v3] プロジェクトを選択します。

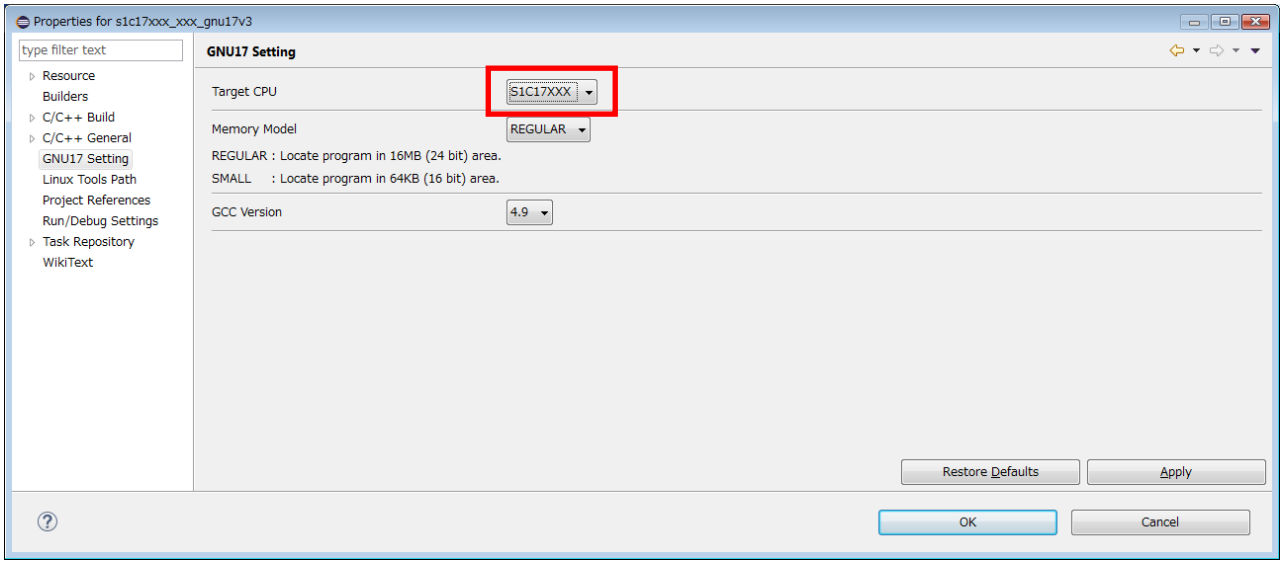



[Copy project into workspace]チェックボックスを選択します。これにより、プロジェクトのコピーがワークスペースディレクトリに作成され、オリジナルが変更されることはありません。最後に、[Finish]ボタンを押します。GNU17 にて、サンプルプロジェクトがインポートされたことを確認後、3.3.1 の（２）で解凍したフォルダは削除して下さい。

（2）ターゲットの変更

サンプルプログラムのターゲットを変更します。サンプルプログラムのプロジェクトを[Project Explorer]ビューで選択し、 [Project]メニューから[Properties]を選択します。

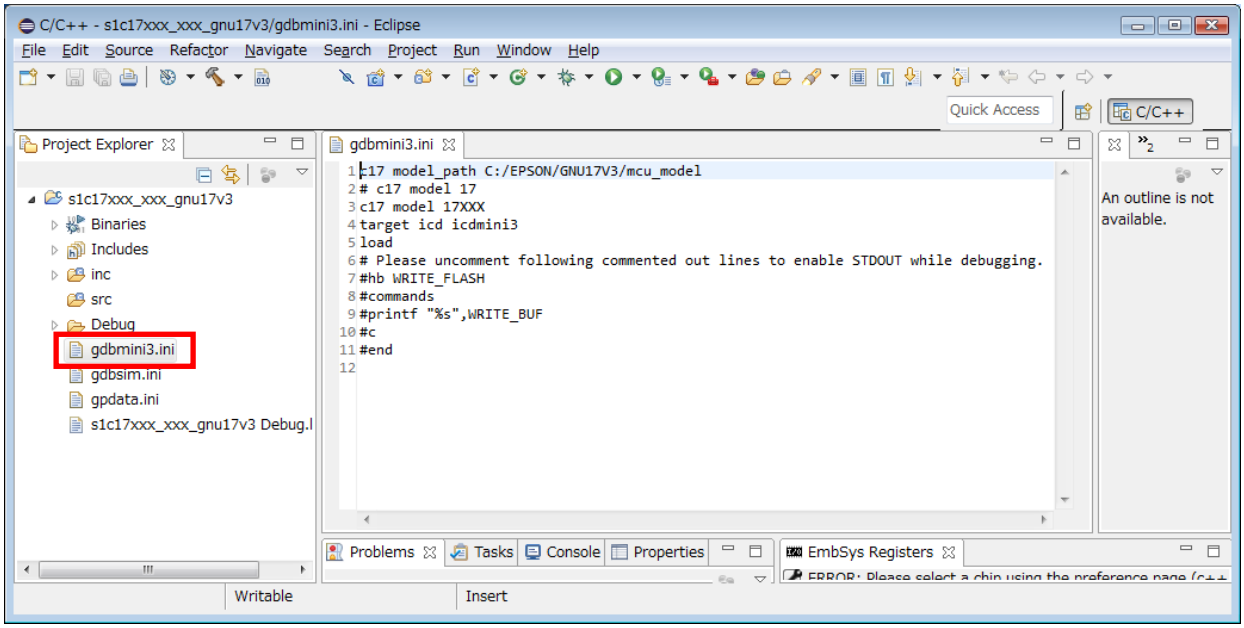
[Properties]ダイアログから、[GNU17 Settings] を選択します。




[Target CPU]から、ターゲットとなる CPU を選択し、[Apply]ボタンを押した後、[OK]ボタンを押して下さい。ターゲットとなる CPU が S1C17589 の場合、[S1C17589]に変更して下さい。

（3）デバッガ起動オプションの変更

デバッガ起動オプションを変更します。サンプルプログラムのプロジェクトを[[Project Explorer]ビューで選択し、 プルジェクト内の[gdbmini3.ini]を選択してエディタで開きます。




ここには、デバッガ起動用コマンドファイルの内容が表示されます。デバッガは使用するマイコンや ICD などに合わせたモードに設定して動作させる必要があります。

※標準出力を利用する場合の注意

標準出力を使用するには GDB コマンドファイル内の赤字の部分の＃を外して有効にしてください。

GDB コマンドファイル［gdbmini3.ini］

```
# Please uncomment following commented out lines to enable STDOUT while debugging.
hb WRITE_FLASH
commands
printf "%s",WRITE_BUF
c
end
```

標準出力を使用する場合は、文字列が表示されるまでにしばらく時間がかかります。

標準出力を使用するサンプルプログラムは、以下となります。

- クロックジェネレータ（CLG）
- 16 ビット PWM タイマ（T16B）

## 3.4 ターゲットとの接続

GNU17 をインストールした PC と ICDmini を USB ケーブルで接続し、S1C17589 を搭載したボードと ICDmini を接続して下さい。

注　GNU17 version 3 で ICDmini ver3.0 を使用する場合は、ICDminiVer1.0,1.1,2.0 互換ケーブル（S5U1C1700 1 W7200）でターゲットシステムと接続して下さい。

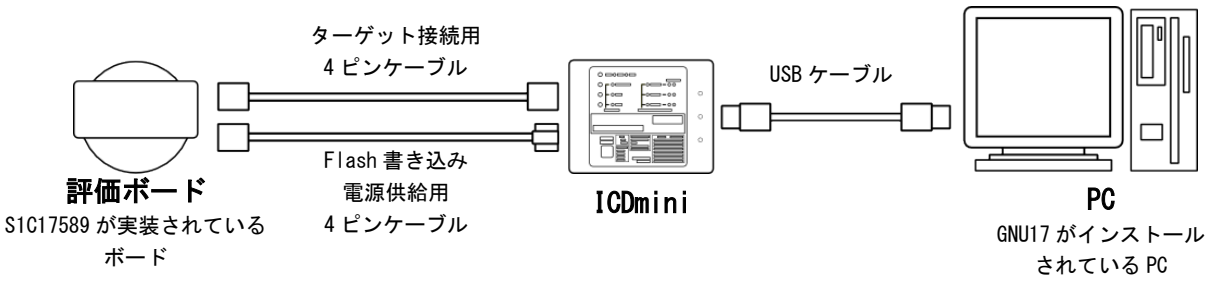



図 3.4.1 ターゲット、ICDmini、PC との接続図

ICDmini と S5U1C17589T21 との接続例

ICDmini と S5U1C17589T21 との接続例を以下に記載します。詳細については、S5U1C17589T21 のマニュアルを参照して下さい。

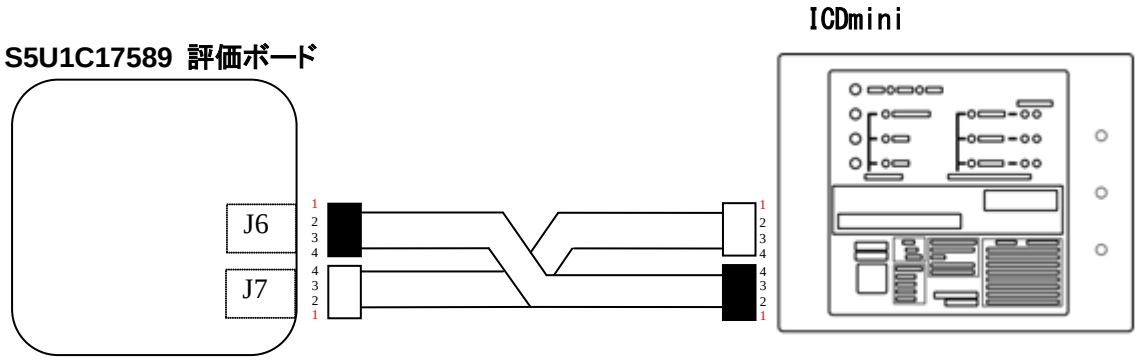



図 3.4.2 ICDmini と S5U1C17589T21 との接続図

## 3.5 ICDmini とターゲットの操作

### 3.5.1 ICDmini ver2.0 の場合

ターゲットと ICDmini、PC を接続後、以下の順序にリセットを行い、ターゲットと ICDmini の接続が確立していることを確認して下さい。なお、ICDmini の DIP Switch の設定は以下の通りです。

表 3.5.1 ICDmini の DIP Switch

| DIP Switch | 説明 | S5U1C17589T21 接続時 |
|---|---|---|
| 1 | Target CPU (C17 / C33) | OPEN ↑ |
| 2 | FLASH WRITER (OFF / WRITE) | OPEN ↑ |
| 3 | FLASH WRITER (OFF / VERIFY) | OPEN ↑ |
| 4 | DSIO LVL (OPEN / TRGET VCC) | ON ↓ |
| 5 | DSIO LVL (3.3V / 1.8V) | OPEN ↑ |
| 6 | CONNECTION DIAG (OFF / ON) | OPEN ↑ |
| 7 | FW UPDATE MODE (OFF / ON) | OPEN ↑ |
| 8 | FLASH VCC OUT (OFF / ON) | ON ↓ |

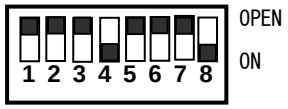



図 3.5.1.1 ICDmini の DIP Switch

（1）ICDmini のリセット

ICDmini の DIP Switch を以下のように設定し、ICDmini の「RESET/START」ボタンを押して下さい。

（2）ターゲットのリセット

ターゲットのリセットを行ってください。各ターゲットのリセット方法は、ターゲットのマニュアルを参照して下さい。

（3）ターゲットと ICDmini の接続確立の確認

ICDmini の LED が以下のように点灯していることを確認して下さい。

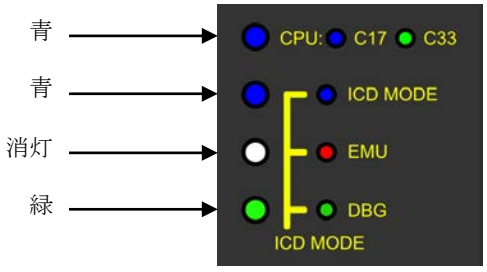



図 3.5.1.2 接続確立時の ICDmini の LED 表示

### 3.5.2 ICDmini ver3.0 の場合

ターゲットと ICDmini、PC を接続後、ターゲットと ICDmini の接続が確立していることを確認して下さい。

（1）ターゲットと ICDmini の接続確立の確認

ICDmini の LED が以下のように点灯していることを確認して下さい。

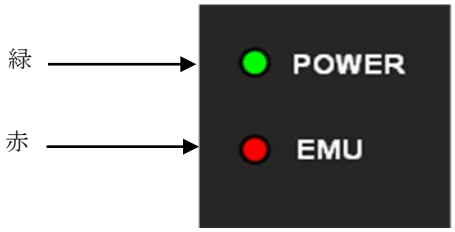



図 3.5.2.1 接続確立時の ICDmini の LED 表示

## 3.6 ビルドとデバッガの起動

サンプルプログラムのビルドを行い、デバッガを起動する方法について記載します。

注　ビルドとデバッガの起動の詳細については、S5U1C17001C Manual を参照して下さい。

注　クロック周りを操作中のプログラムを停止する場合、正常動作が期待できませんのでご注意ください。当該コードのデバッグにあたっては、S5U1C17001C Manual の DCLK 切替モードコマンドを参照してください。

### 3.6.1 GNU17 version 2 の場合

（1）ビルドの実行

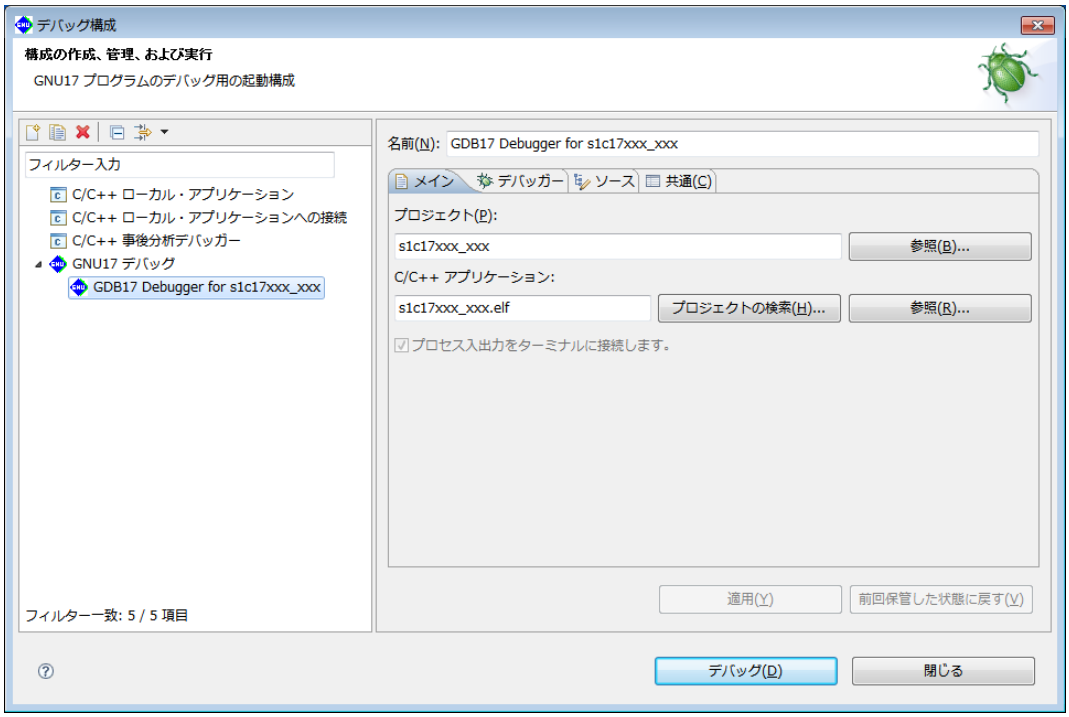
GNU17 の[C/C++ プロジェクト]ビューからビルドを実行したいプロジェクトを選択します。GNU17 の[プロジェクト]メニューから[プロジェクトのビルド]を選択し、ビルドを実行します。[C/C++ プロジェクト]ビューからビルドを実行したいプロジェクトを選択し、コンテキストメニュー（右クリックにより表示）の[プロジェクトのビルド]からも実行することができます。

（2）デバッガの起動構成ダイアログの表示

GNU17 の [実行]メニューから[デバッグの構成...]を選択し、起動構成ダイアログを表示させます。ツールバーの[デバッグ]ボタンのメニューからも表示させることができます。

(3) 起動するサンプルプログラムの選択

ツリーリストから[GDB17 Debugger for <プロジェクト名> ]を選択します。




（4）デバッガの起動

[デバッグ]ボタンをクリックします。デバッガ gdb が起動して指定のコマンドファイルを実行します。

### 3.6.2 GNU17 version 3 の場合

（1）ビルドの実行

GNU17 の[Project Explorer]ビューからビルドを実行したいプロジェクトを選択します。GNU17 の[Project]メニューから[Build Project]を選択し、ビルドを実行します。[Project Explorer]ビューからビルドを実行したいプロジェクトを選択し、コンテキストメニュー（右クリックにより表示）の[Build Project]からも実行することができます。

（2）デバッガの起動構成ダイアログの表示

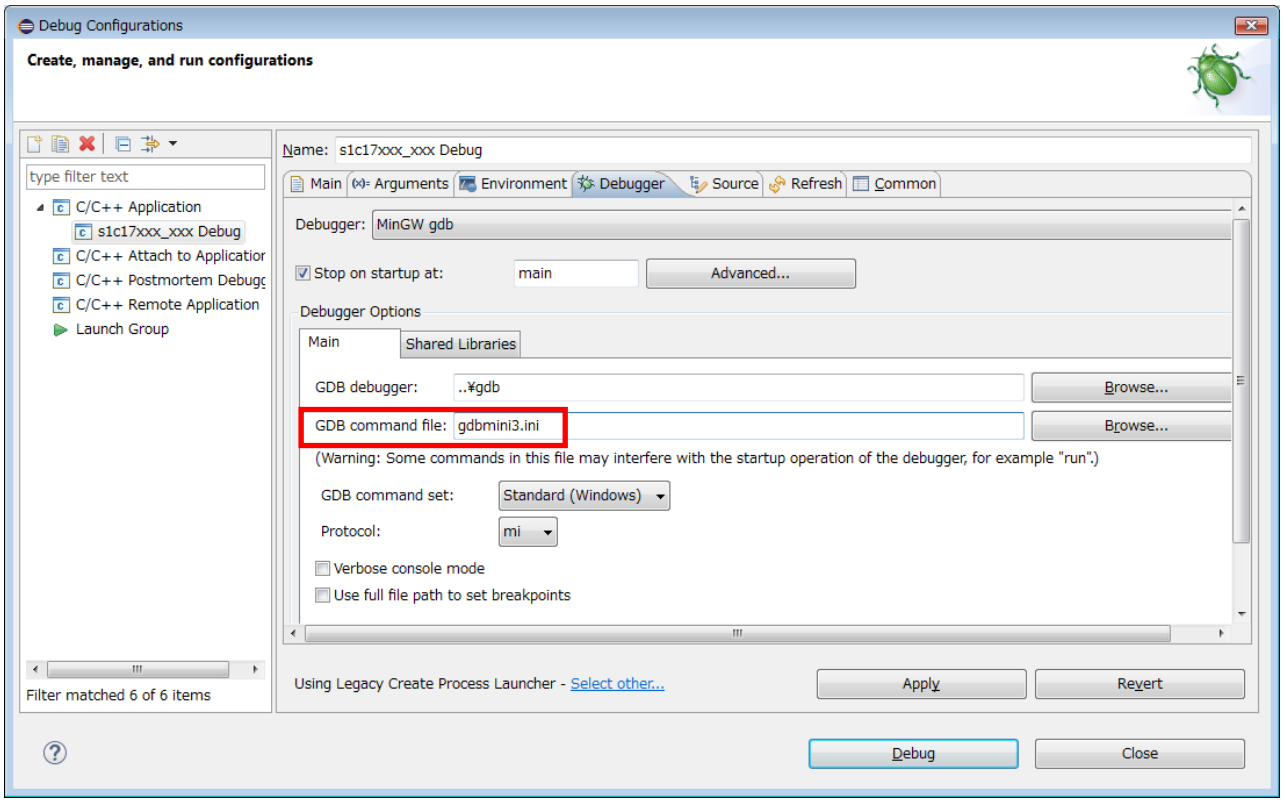
GNU17 の [Run]メニューから[Debug Configurations...]を選択し、Debug Configurations ダイアログを表示させます。ツールバーの[Debug]ボタンのメニューからも表示させることができます。

(3) 起動するサンプルプログラムの選択とデバッガの設定

[Debugger]タブの[GDB command file:]に[gdbmini3.ini]を指定します。

（サンプルでは、デフォルトで [gdbmini3.ini] が指定されています。）

設定後、[Apply]ボタンをクリックした後に[Debug]ボタンをクリックして指定のコマンドファイルを実行します。



# 4. サンプルソフトウェア機能詳細

本章では S1C17589 サンプルソフトウェアの機能詳細について記載します。

## 4.1 入出力ポート（PPORT）

### 4.1.1 サンプルソフトウェア仕様

本サンプルソフトウェアは入出力ポートを使用して以下の動作を行います。

- ポートを入力に設定し、入力信号が High レベルまたは Low レベルになったとき、割り込みが発生します。
- ポートを出力に設定し、High レベルまたは Low レベルの信号を出力します。

使用するポート設定とポート名は以下の通りです。

表 4.1.1 入出力ポート設定一覧

| 設定 | ポート名 |
|---|---|
| 出力ポート | P01 |
| 入力割り込みポート | P02 |

### 4.1.2 ハードウェア条件

本サンプルプログラムはマイコンの OSC1（32.768kHz）内蔵発振回路で動作します。
マイコンの各ポートを以下のように接続してご使用下さい。

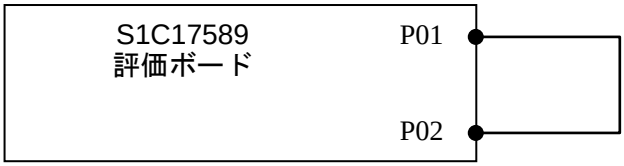



図 4.1.1 入出力ポート　サンプルソフトウェアハードウエア接続図

### 4.1.3 動作概要

1. OSC1 の発振を開始し、システムクロックを IOSC から OSC1 に切り替え、IOSC は停止します。
2. ポートを初期化します。
3. P01 ポートを出力ポートに設定して Low レベルを出力します。
4. P02 ポートを入力ポートに設定し、Low レベルから High レベルにて割り込みが発生するように設定します。
5. P01 ポートから High レベルの信号を出力します。
6. P02 ポートの割り込みにて P02 ポートが High レベルであることを確認します。
7. P02 ポートを High レベルから Low レベルにて割り込みが発生するように設定します。
8. P01 ポートから Low レベルの信号を出力します。
9. P02 ポートの割り込みにて P02 ポートが Low レベルであることを確認します。

## 4.2 クロックジェネレータ（CLG）

### 4.2.1 サンプルソフトウェア仕様

発振回路サンプルプログラムは、クロックジェネレータを使用して以下の動作を行います。

- OSC1 の発振と停止を行う。
- OSC3 の発振と停止を行う。
- IOSC の発振と停止を行う。
- システムクロックを OSC1 へ切り替える。
- システムクロックを OSC3 へ切り替える。
- システムクロックを IOSC へ切り替える。
- システムクロックが IOSC、OSC3 の時、SLEEP モード、RUN モード間の動作モードの遷移を行う。
- OSC1 HALT、IOSC RUN/ OSC3 RUN 間の動作モードの遷移を行う。
- IOSC の動作周波数を切り替える。

### 4.2.2 ハードウェア条件

本サンプルプログラムはマイコンの内蔵発振（IOSC）を使用して動作開始した後、OSC1、OSC3 に切り替える動作を行います。OSC1/OSC3 の発振のために振動子が必要ですが、詳細については、S1C17589 のテクニカルマニュアル「クロックジェネレータ(CLG)」を参照して下さい。

### 4.2.3 動作概要

本サンプルプログラムは IOSC をシステムクロックに使用した状態で動作を開始します。
なお以下で表示されるカウント値は、OSC1 の発振周波数=32.768kHz の 10 ミリ秒分の
カウント値の 327 が基準となります。
各対象 CLK での 10 ミリ秒ループが OSC1 での 10 ミリ秒ループ期待値=327 と比較することで
期待通りの発振をしているか確認します。
T16 タイマ Ch.0 = OSC1
T16 タイマ Ch.1 = 各対象 CLK

0. OSC1 を発振開始し、システムクロックを OSC1 に変更後、IOSC を発振停止します。
1. CPU を停止（HALT）し、1 秒毎にタイマ割り込みする動作を 3 回繰り返します。タイマ割り込みを受信する度、IOSC（8MHz 設定）が発振開始し、T16_ch.1 で 10 ミリ秒分のループを実行、ループ前後の T16_ch.0 タイマのカウント値の差を Simulated I/O に表示し、IOSC を停止します。。
2. OSC1 の発振クロックを基準に IOSC の発振周波数のオートトリミングを行います。
3. 先の 1 と同じ動作を行うので、オートトリミングの結果が確認できます。
4. CPU を停止（HALT）し、1 秒毎にタイマ割り込みする動作を 3 回繰り返します。タイマ割り込みを受信する度、OSC3（16MHz 設定）が発振開始し、T16_ch.1 で 10 ミリ秒分のループを実行、ループ前後の T16_ch.0 タイマのカウント値の差を Simulated I/O に表示し、OSC3 を停止します。
5. IOSC を発振開始した後に CPU を停止(SLEEP)させます。停止状態から CPU を起こすためには、P04 ポートに”L”レベルの信号を入れる必要があります。S5U1C17589T21（SVTmini17589）の場合、J3 コネクタ 35 番ピンへ”L”レベルの信号を入れることで可能です。CPU が起きると自動で IOSC が発振開始し、T16_ch.1 で 10 ミリ秒分のループを実行、ループ前後の T16_ch.0 タイマのカウント値の差を Simulated I/O に表示し、IOSC を停止します。この動作を 3 回繰り返します。
6. OSC3 を発振開始した後に CPU を停止(SLEEP)させます。停止状態から CPU を起こすためには、P04 ポートに”L”レベルの信号を入れる必要があります。S5U1C17589T21（SVTmini17589）の場合、J3 コネクタ 35 番ピンへ”L”レベルの信号を入れることで可能です。CPU が起きると自動で OSC3 が発振開始し、T16_ch.1 で 10 ミリ秒分のループを実行、ループ前後の T16_ch.0 タイマのカウント値の差を Simulated I/O に表示し、OSC3 を停止します。この動作を 3 回繰り返します。
7. 最初に IOSC(4MHz 設定) を発振開始し、T16_ch.1 で 10 ミリ秒分のループを実行、ループ前後の

T16_ch.0 タイマのカウント値の差を Simulated I/O に表示します。次に IOSC（8MHz 設定）を発振開始し、T16_ch.1 で 10 ミリ秒分のループを実行、ループ前後の T16_ch.0 タイマのカウント値の差を Simulated I/O に表示します。次に IOSC（12MHz 設定）を発振開始し、T16_ch.1 で 10 ミリ秒分のループを実行、ループ前後の T16_ch.0 タイマのカウント値の差を Simulated I/O に表示します。次に IOSC（16MHz 設定）を発振開始し、T16_ch.1 で 10 ミリ秒分のループを実行、ループ前後の T16_ch.0 タイマのカウント値の差を Simulated I/O に表示します。

8. 最後に OSC1、OSC3 を発振停止させ、一連の動作を終了します。

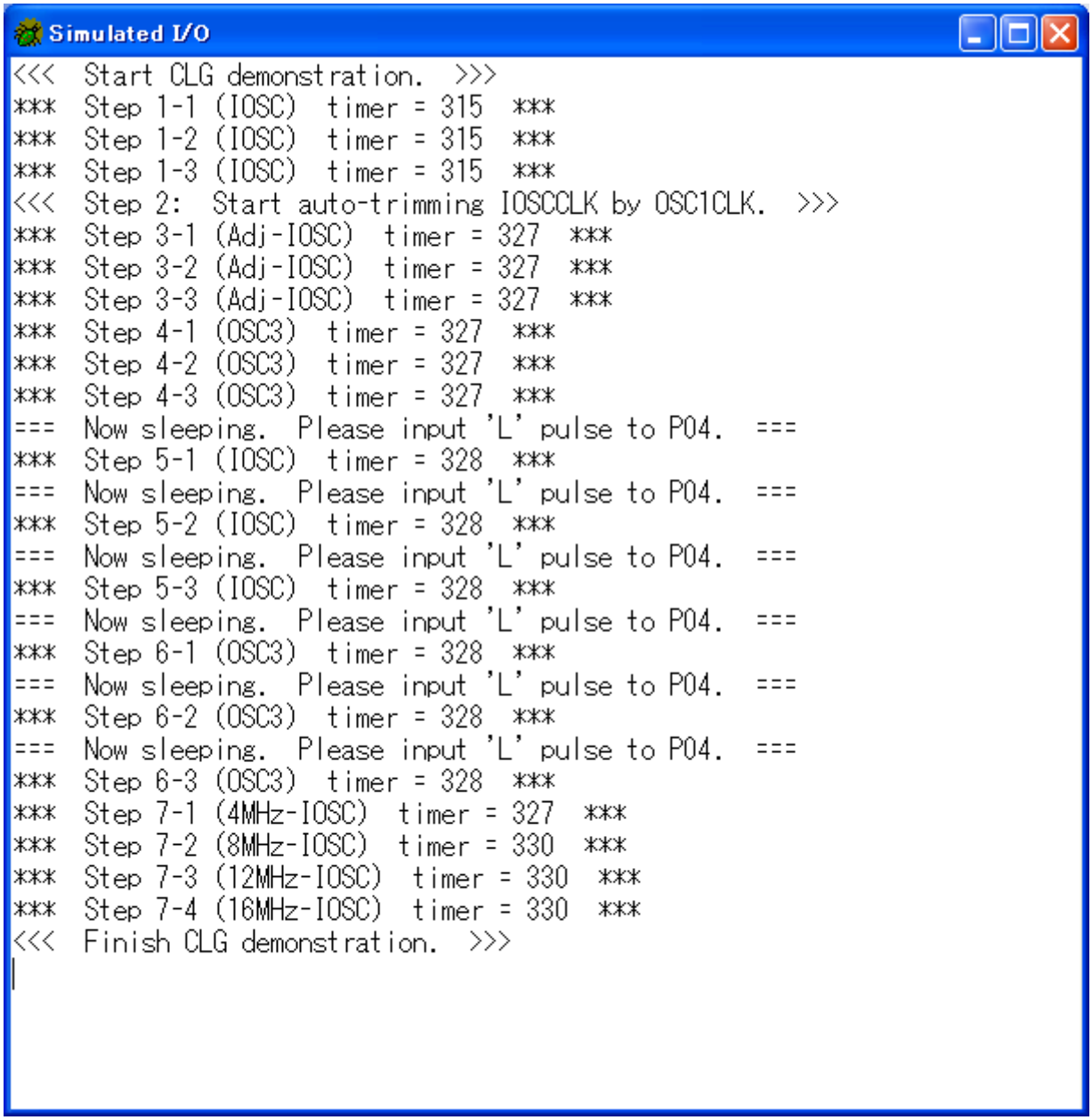



図 4.2.1　クロックジェネレータ　サンプルプログラム画面表示例

## 4.3 16 ビットタイマ（T16）

### 4.3.1 サンプルソフトウェア仕様

16 ビットタイマサンプルプログラムは 16 ビットタイマを使用して以下の動作を行います。

- 16 ビットタイマ割り込みを発生させ、タイマのカウンター値を取得する。
- 割り込み待機時は CPU を halt モードにして消費電力を低減する。

### 4.3.2 ハードウェア条件

本サンプルプログラムはマイコンの IOSC（8MHz 設定）内蔵発振回路で動作します。

### 4.3.3 動作概要

1. 16 ビットタイマのカウンターを設定します。
2. 16 ビットタイマの開始し、CPU を Halt 状態にします。
3. 16 ビットタイマの割り込みが発生すると CPU は Halt 状態を解除します。
4. 16 ビットタイマの割り込み回数を取得します。
5. 16 ビットタイマ割り込み回数が 1000 回未満のときは、再び CPU を Halt 状態にします。
6. 16 ビットタイマ割り込み回数が 1000 回以上のときは、16 ビットタイマを停止し、サンプルプログラムを終了します。

## 4.4 16 ビット PWM タイマ（T16B）

### 4.4.1 サンプルソフトウェア仕様

16 ビット PWM タイマサンプルプログラムは 16 ビット PWM タイマを使用して以下の動作を行います。

- 16 ビット PWM タイマのコンパレータ割り込みを発生させる。
- 16 ビット PWM タイマのキャプチャ割り込みを発生させ、タイマのカウンター値を取得する。
- 割り込み待機時は CPU を halt モードにして消費電力を低減する。

### 4.4.2 ハードウェア条件

本サンプルプログラムはマイコンの IOSC（8MHz 設定）内蔵発振回路で動作します。

### 4.4.3 動作概要

1. 16 ビット PWM タイマを以下のように設定し、T16 ビット PWM タイマを開始します。
   - モード：リピートアップカウントモード
   - カウンター最大値：0x5000
   - コンパレータ/キャプチャ回路 0：コンパレータモード(コンペアバッファ：0x2000)
   - コンパレータ/キャプチャ回路 1：キャプチャモード(トリガ信号：LOW)キャプチャ割り込み回数
2. 16 ビット PWM タイマを開始し、CPU を Halt 状態にします。
3. 16 ビット PWM タイマの割り込みが発生すると CPU は Halt 状態を解除します。
4. 16 ビット PWM タイマの割り込みが発生したとき、以下の割り込み回数を取得します
   - コンペア割り込み回数
   - キャプチャ割り込み回数
   - カウント MAX 割り込み回数
   - カウンタゼロ割り込み回数
5. コンペア割り込みが発生したとき、キャプチャ用トリガ信号を HIGH に設定します。
6. カウント MAX 割り込みが発生したとき、キャプチャ用トリガ信号を LOW に設定します。
7. カウント MAX 割り込み回数が 10 回未満のときは、再び CPU を Halt 状態にします。
8. カウント MAX 割り込み回数が 10 回以上のときは、16 ビットタイマを停止し、割り込みが発生したときのカウント値を Simulated I/O に表示し、サンプルプログラムを終了します。

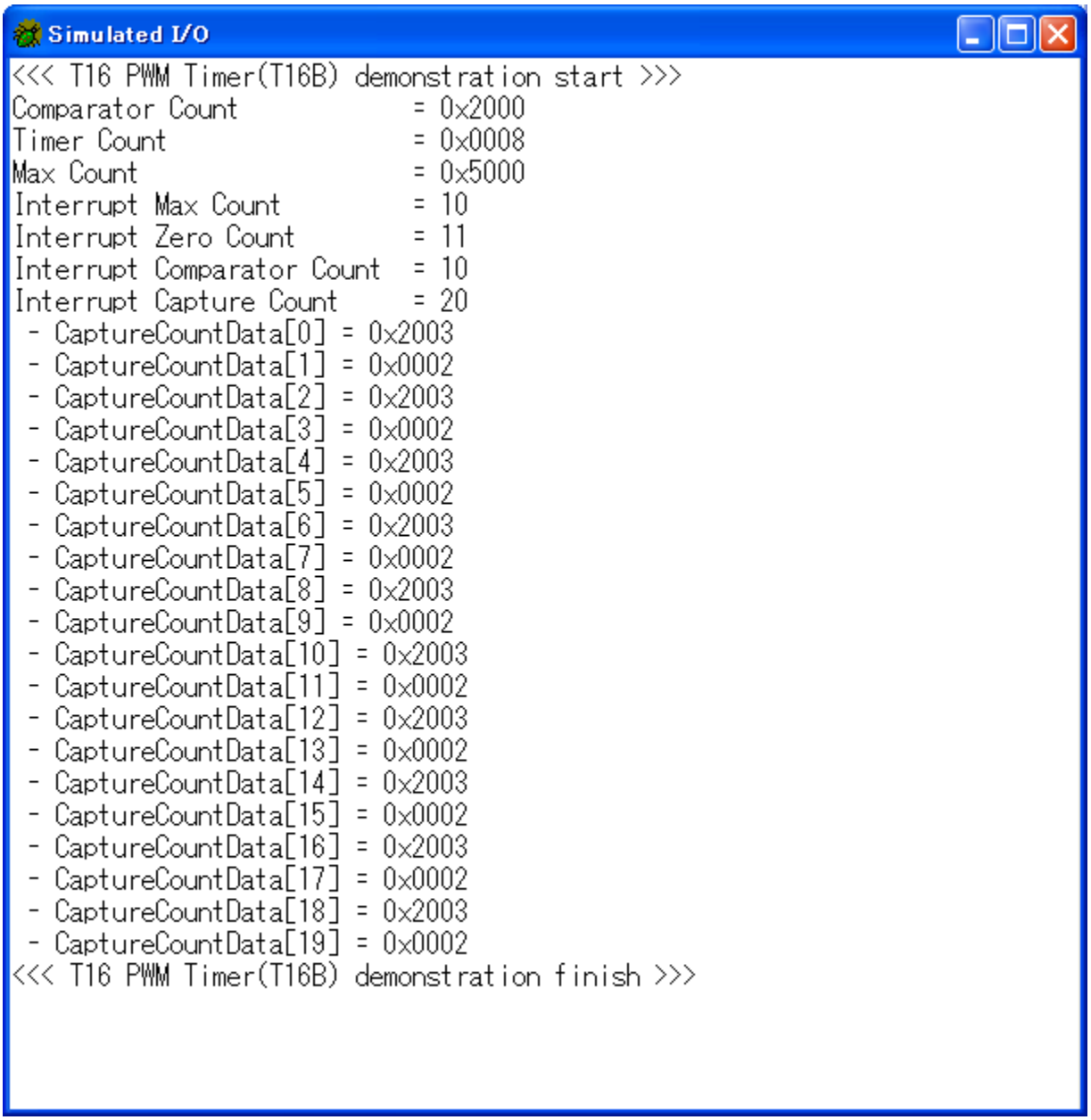



図 4.4.1　16 ビット PWM タイマ　サンプルプログラム画面表示例

## 4.5 リアルタイムクロック（RTCA）

### 4.5.1 サンプルソフトウェア仕様

リアルタイムクロックサンプルプログラムはリアルタイムクロックを使用して以下の動作を行います。

- リアルタイムクロックの時刻を設定する。
- リアルタイムクロックの時刻を取得する。
- リアルタイムクロックの割り込み回数を取得する。
- 論理緩急機能を使用する。
- ストップウォッチを使用する。

### 4.5.2 ハードウェア条件

本サンプルプログラムはマイコンの OSC1（32.768kHz）内蔵発振回路で動作します。

### 4.5.3 動作概要

1. リアルタイムクロックを初期化します。
2. リアルタイムクロックの 1 秒割り込み、1/2 秒割り込みを有効にします。
3. リアルタイムクロックの論理緩急機能（論理緩急実行間隔：4,096 秒、補正率：-58.2ppm）を設定します。
4. リアルタイムクロックの日時を「99 年 12 月 31 日(土) 23 時 59 分 51 秒」に設定し、24 時間モードに設定します。
5. リアルタイムクロックを開始し、CPU を Halt 状態にします。
6. リアルタイムクロックの割り込みが発生すると CPU は Halt 状態を解除します。
7. リアルタイムクロックの 1 秒割り込みの回数を取得します。
8. 1 秒割り込みの回数が 10 回未満のときは、再び CPU を Halt 状態にします。
9. 1 秒割り込みの回数が 10 回以上のときは、12 時間モードに設定し、リアルタイムクロックの日時を取得します（取得した値は、「00 年 1 月 1 日(日) AM 12 時 00 分 01 秒」になります）。
10. リアルタイムクロックの 1 秒割り込みを無効にします。
11. リアルタイムクロックの開始から 4,096 秒間隔で、論理緩急を実行します。
12. ストップウォッチを初期化します。
13. ストップウォッチの割り込みを有効にします。
14. ストップウォッチを開始し、CPU を Halt 状態にします。
15. ストップウォッチの割り込みが発生すると CPU は Halt 状態を解除します。
16. ストップウォッチ 1Hz 割り込みの回数を取得します。
17. ストップウォッチ 1Hz 割り込みの回数が 10 回未満のときは、再び CPU を Halt 状態にします。
18. ストップウォッチ 1Hz 割り込みの回数が 10 回以上のときは、ストップウォッチカウント(10Hz 桁、100Hz 桁)を読み出します。
19. ストップウォッチを停止し、ストップウォッチの割り込みを無効にします。

## 4.6 ウォッチドッグタイマ（WDT）

### 4.6.1 サンプルソフトウェア仕様

ウォッチドッグタイマサンプルプログラムはウォッチドッグタイマを使用して以下の動作を行います。

- ウォッチドッグタイマのクリアを確認する。
- ウォッチドッグタイマによるリセット割り込みを発生させる。

### 4.6.2 ハードウェア条件

本サンプルプログラムはマイコンの OSC1（32.768kHz）内蔵発振回路で動作します。

### 4.6.3 動作概要

1. ウォッチドッグタイマとリアルタイムクロックを開始します。
2. リアルタイムクロックの 1 秒割り込みが発生するとウォッチドッグタイマをリセットします。
3. リアルタイムクロックの 1 秒割り込みが 10 回発生するとリアルタイムクロックを停止します。
4. ウォッチドッグタイマによるリセット割り込みが発生すると本サンプルプログラムの先頭(boot.c)に処理が移ります。

## 4.7 UART（UART）

### 4.7.1 サンプルソフトウェア仕様

UART サンプルプログラムは UART を使用して以下の動作を行います。

- UART を使いデータを送信する。
- UART を使いデータを受信する。

### 4.7.2 ハードウェア条件

本サンプルプログラムはマイコンの IOSC（8MHz 設定）内蔵発振回路で動作します。
UART マスターサンプルプログラムと UART スレーブサンプルプログラムが動作している評価ボードを接続し、スレーブから起動して下さい。各ポートは以下のように接続してご使用下さい。

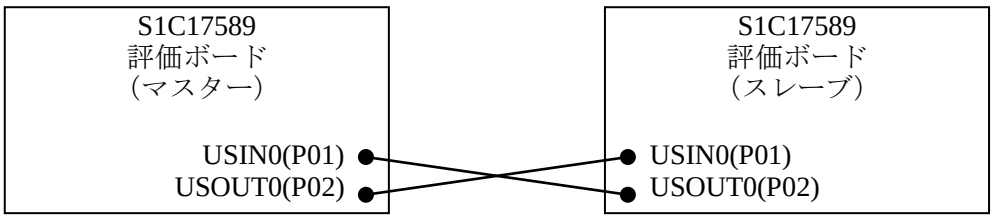



図 4.7.1 UART サンプルプログラムハードウェア接続図

### 4.7.3 動作概要

#### 4.7.3.1 マスターサンプル動作概要

1. UART ポートを以下の値に初期化します。
   - 通信速度 : 500k bps
   - データ長 : 8bit
   - ストップビット : 1bit
   - パリティ : 無し
2. 「A(0x41)」から「Z(0x5A)」までのデータと「¥0」をスレーブに送信する。
3. スレーブから 27 バイトのデータを受信する。
4. 受信したデータが、送信したデータと同じか比較し、サンプルプログラムを終了します。

#### 4.7.3.2 スレーブサンプル動作概要

1. UART ポートを以下の値に初期化します。
   - 通信速度 : 500k bps
   - データ長 : 8bit
   - ストップビット : 1bit
   - パリティ : 無し
2. マスターから 27 バイトのデータを受信する。
3. 「A(0x41)」から「Z(0x5A)」までのデータと「¥0」をマスターに送信する。
4. 受信したデータが、送信したデータと同じか比較し、サンプルプログラムを終了します。

## 4.8 同期式シリアルインタフェース（SPIA）

### 4.8.1 サンプルソフトウェア仕様

SPI サンプルプログラムは SPI を使用して以下の動作を行います。

- SPI を使いデータを送信する。
- SPI を使いデータを受信する。

### 4.8.2 ハードウェア条件

本サンプルプログラムはマイコンの IOSC（8MHz 設定）内蔵発振回路で動作します。
SPI マスターサンプルプログラムと SPI スレーブサンプルプログラムが動作している評価ボードを接続し、スレーブから起動して下さい。各ポートは以下のように接続してご使用下さい。

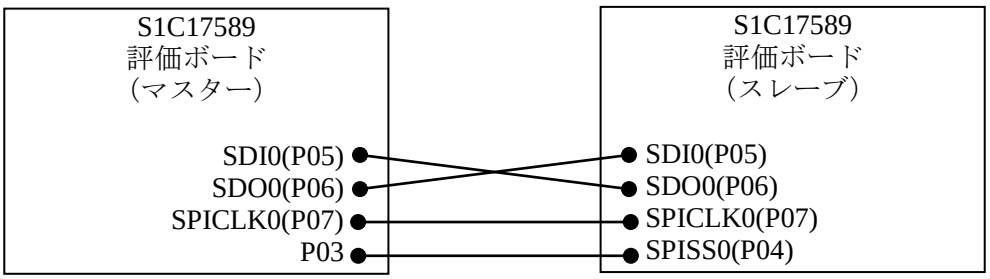



図 4.8.1 SPI マスター、スレーブサンプルプログラムハードウェア接続図

### 4.8.3 動作概要

#### 4.8.3.1 マスターサンプル動作概要

1. SPI ポートを以下の値に初期化します。
   - 通信速度 : 2Mbps
   - データ長 : 8bit
   - データフォーマット : MSB 先頭
2. 「A(0x41)」から「Z(0x5A)」までのデータと「¥0」をスレーブに送信する。
3. スレーブから 27 バイトのデータを受信する。
4. 受信したデータが、送信したデータと同じか比較し、サンプルプログラムを終了します。

#### 4.8.3.2 スレーブサンプル動作概要

1. SPI ポートを以下の値に初期化します。
   - 通信速度 : 2Mbps
   - データ長 : 8bit
   - データフォーマット : MSB 先頭
2. マスターから 27 バイトのデータを受信する。
3. 「A(0x41)」から「Z(0x5A)」までのデータと「¥0」をマスターに送信する。
4. 受信したデータが、送信したデータと同じか比較し、サンプルプログラムを終了します。

## 4.9 I2C（I2C）

### 4.9.1 サンプルソフトウェア仕様

I2C サンプルプログラムは I2C を使用して以下の動作を行います。

- I2C を使いデータを送信する。
- I2C を使いデータを受信する。

### 4.9.2 ハードウェア条件

本サンプルプログラムはマイコンの IOSC（8MHz 設定）内蔵発振回路で動作します。
I2C マスターサンプルプログラムと I2C スレーブサンプルプログラムが動作している評価ボードを接続し、スレーブから起動して下さい。デバッグ時は、スレーブ側のプログラムを先に走らせてください。
又、両評価ボードの GND は共通としてください。各ポートは以下のように接続してご使用下さい。

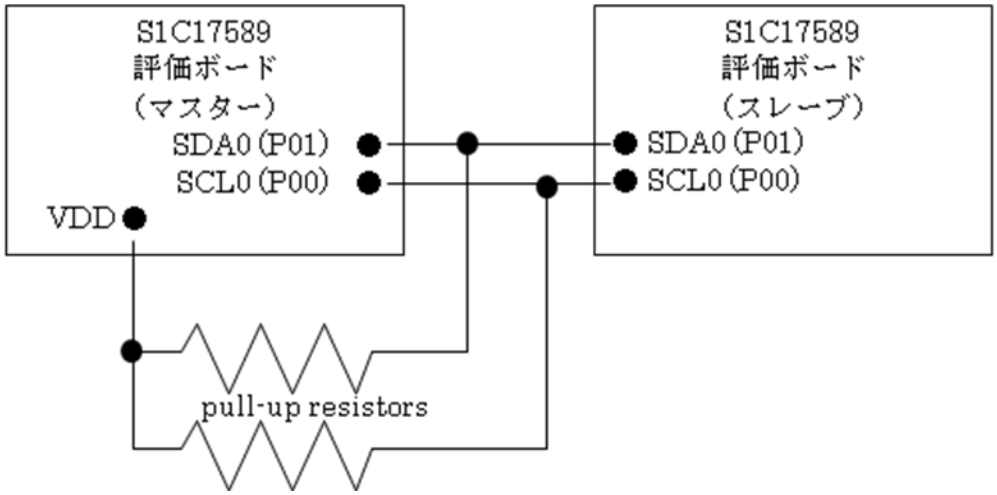



図 4.9.1　I2C マスター、スレーブサンプルプログラムハードウェア接続図

### 4.9.3 動作概要

#### 4.9.3.1 マスターサンプル動作概要

1. I2C ポートを以下の値に初期化します。
   - 通信速度　　：400kbps
2. 「A(0x41)」から「Z(0x5A)」までのデータと「¥0」をスレーブに送信する。
3. スレーブから 27 バイトのデータを受信する。
4. 受信したデータが、送信したデータと同じか比較し、サンプルプログラムを終了します。

#### 4.9.3.2 スレーブサンプル動作概要

1. I2C ポートを以下の値に初期化します。
   - アドレスモード　　：7bit
   - スレーブアドレス　　：0x2A
   - ジェネラルコール　　：無効
2. マスターから 27 バイトのデータを受信する。
3. 「A(0x41)」から「Z(0x5A)」までのデータと「¥0」をマスターに送信する。
4. 受信したデータが、送信したデータと同じか比較し、サンプルプログラムを終了します。

## 4.10 電源電圧検出回路（SVD）

### 4.10.1サンプルソフトウェア仕様

電源電圧検出回路サンプルプログラムは電源電圧検出回路を使用して以下の動作を行います。

- 電源電圧検出回路にて、電源電圧と比較電圧との比較を行う。

### 4.10.2ハードウェア条件

本サンプルプログラムはマイコンの OSC1（32.768kHz）内蔵発振回路で動作します。以下のように接続してご使用下さい。

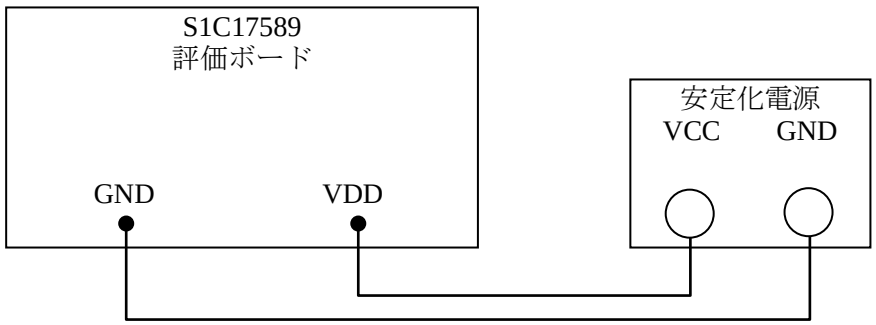



図 4.10.1 電源電圧検出回路サンプルプログラム ハードウェア接続図

### 4.10.3 動作概要

1. リアルタイムクロックを 1 秒毎に割り込みが発生するように設定し、開始します。
2. 電源電圧検出回路を 2.7v 以下の電圧で、電源電圧低下を検出するように設定し、開始します。
3. 1 秒毎に、電圧電圧の低下が発生したかどうかを確認します。
4. 10 秒間、電源電圧の低下が発生した回数を計測し、サンプルプログラムを終了します。

## 4.11 10 ビット A/D 変換器（ADC10A）

### 4.11.1サンプルソフトウェア仕様

10 ビット A/D 変換器サンプルプログラムは、A/D コンバータを使用して以下の動作を行います

- T16 ビットタイマーアンダーフロートリガ、単一変換モードで 1 秒周期で A/D 変換結果を取得する。
- ソフトウェアトリガ、連続変換モードで A/D 変換結果を取得する。

### 4.11.2 ハードウェア条件

本サンプルプログラムはマイコンの IOSC（8MHz 設定）内蔵発振回路で動作します。以下のように接続してご使用下さい。

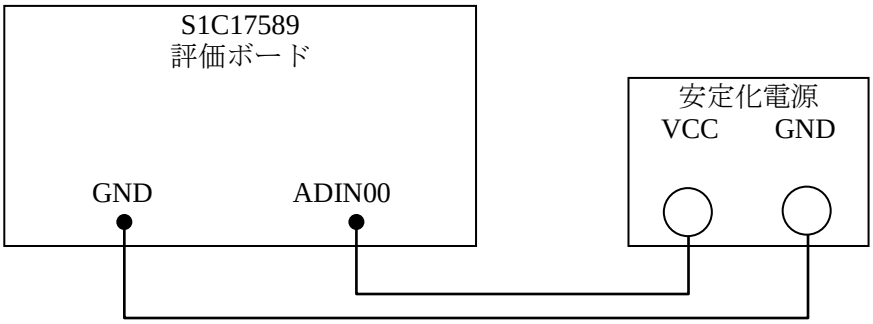



図 4.11.1　10 ビット A/D 変換器サンプルプログラム 外部接続図

### 4.11.3 動作概要

1　A/D コンバータは T16 Ch.3 を変換クロックに使用していますが、その T16 Ch.3 の 1 秒毎に発生するアンダーフローでトリガをかけて A/D 変換を開始し、終了で割り込みが発生します。この動作を繰り返します

2　A/D コンバータに割り込みが発生すると、オーバーライトエラー割り込みの場合は A/D 変換をやり直します。A/D 変換完了割り込みの場合は、ADIN00 の値を読み取り、その結果を保存します。この動作を 10 回繰り返します。

3　ソフトウェアトリガをかけて連続変換モードで A/D 変換を開始し、終了で割り込みが発生します。この動作を繰り返します。

4　A/D コンバータに割り込みが発生すると、オーバーライトエラー割り込みの場合は A/D 変換をやり直します。A/D 変換完了割り込みの場合は、ADIN00 の値を読み取り、その結果を保存します。この動作を 10 回繰り返します。

## 4.12 IR リモートコントローラ（REMC2）

### 4.12.1サンプルソフトウェア仕様

IR リモートコントローラサンプルプログラムは、REMC2 を用いて赤外線リモコン用の信号を発生させます。本サンプルプログラムは REMC2 を使用して以下の動作を行います。

- 初期化後スリープ状態に移行し、ポート入力（P04～P07）による割り込みを待ちます。
- ポート入力（P04～P07）による割り込みが発生すると、どのポート入力割り込みか解析し、それに応じて REMC2 でリモコン用の信号(フレーム信号)を発生させます。そのとき、T16Ch.0 タイマを起動し、約 108ms 後にポート入力の状況を確認します。
- T16Ch.0 タイマでの割り込み時、同じポートに“L”レベルが連続して入力されていた場合、リピート信号を発生し、約 108ms 後にポート入力の状態を再確認するよう、T16Ch.0 タイマをセットします。同じポートに“L”レベルが入力されていなかった場合、T16Ch.0、REMC2 は停止し、スリープモードに移行し、ポート入力（P04～P07）による割り込みを待ちます。

### 4.12.2 ハードウェア条件

本サンプルプログラムは S1C17589 評価ボード(SVTmini17589)を対象とするサンプルプログラムです。図 4.12.1 に示した回路例との接続などについては、SVTmini17589 のマニュアルを参照して下さい。

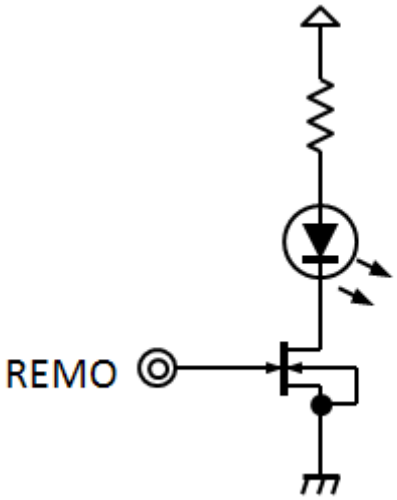



図 4.12.1 IR リモートコントローラサンプルプログラム 赤外発光ダイオード接続回路例

### 4.12.3 動作概要

1　赤外線リモコンの通信フォーマットには主に、NEC/家製協/SONY フォーマットがありますが、本サンプルソフトウェアでは NEC フォーマットを用いています。

2　NEC フォーマットの仕様の概略は、以下と図 4.12.2 に示した通りです。

キャリア: 赤外線(λp = 940nm)
サブキャリア: fsc = 38kHz, 1/3duty
T = 562μs
固定長フレーム (32bit)
16bit のカスタマーコード
8bit のデータ + 8bit の反転データ

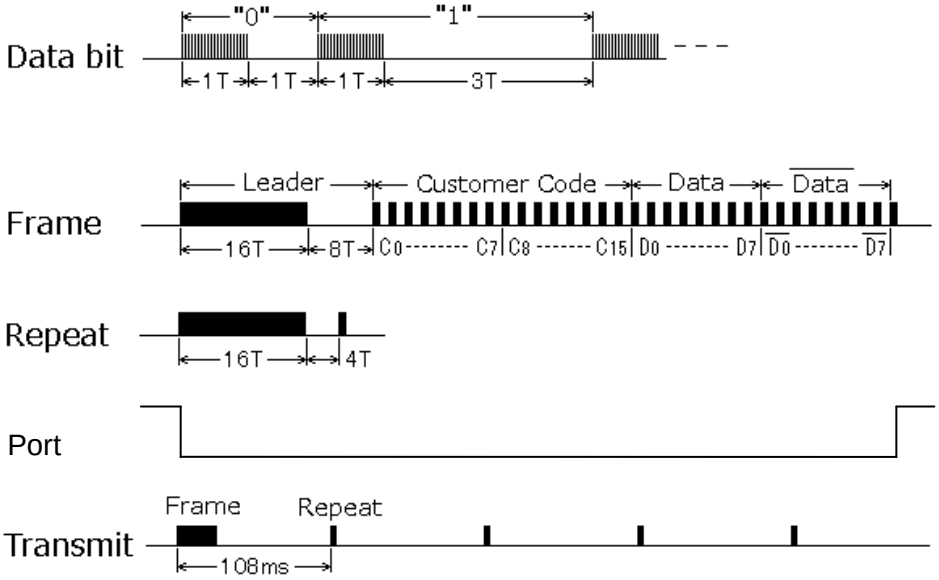



図 4.12.2 NEC フォーマットの信号波形

3 REMC2 は、APLEN（データ信号のデューティ）、DBLEN（データ信号の周期）という 2 つのパラメーターでデータ波形の形状を決めています。APLEN、DBLEN は、フレーム信号は各 34 個、リピート信号は各 2 個で波形の形状を定義していますが、P04~P07 の何れにも共通なリーダー、カスタマーコード、リピート信号は初期化時に割り当てています。データ部の 16 個はポートで異なる値になるため、“L”レベルを入力して発生した割り込みの処理のときに、値を割り当てています。

4 本サンプルプログラムは、IOSC の 8MHz 動作を用いています。P04~P07 を GPIO モードにセット、REMO 用にポートを割り当て、T16Ch.0 を初期化したのち、スリープモードに移行し、P04~P07 に“L”レベル入力されて発生する割り込みを待ちます。

5 割り込みが発生すると、どのポートに“L”レベルが入力されたか解析し、ポートに応じ、データ部の APLEN、DBLEN の値を割り当てます。REMC2 はバッファモードを用いているため、起動の前に初期化してから REMC2 を起動します。また T16Ch.0 を起動し、約 108ms 後に T16Ch.0 割り込みを発生させます。

6 REMC2 のコンペア DB 割り込みが発生すると、APLEN、DBLEN の次のデータをレジスタに書き込みます。最後のデータのひとつ前のとき、REMC2 のモードをワンショットモードに切り替え、髭状の信号出力が生じないようにします。最後のデータのときには、REMC2 の動作を停止します。

7 T16Ch.0 割り込みが発生したとき、同じポートに“L”レベル入力されているか確認し、“L”レベル入力されている場合は REMC2 をリピート波形になるようレジスタ値を設定し、起動の前に初期化してから REMC2 を起動します。“L”レベル入力されていない場合は、T16Ch.0 の動作を停止します。

# 改訂履歴表



| Rev. No. | 日付 | ページ | 種別 | 改訂内容（旧内容を含む）<br>および改訂理由 |
|---|---|---|---|---|
| Rev 1.2 | 2016/01/19 | 全ページ | 新規 | 新規制定 |
| | | | | |
| | | | | |

セイコーエプソン株式会社
マイクロデバイス事業部　デバイス営業部

東京　〒191-8501　東京都日野市日野 421-8
TEL（042）587-5313（直通）　FAX（042）587-5116

大阪　〒541-0059　大阪市中央区博労町 3-5-1　エプソン大阪ビル 15F
TEL（06）6120-6000（代表）　FAX（06）6120-6100

ドキュメントコード：413175800
2016 年 1 月作成