EPSON
EXCEED YOUR VISION

S1C17 Family Application Note

S1C17F00 シリーズ
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.1.0

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. 本資料に掲載されている製品のうち「外国為替及び外国貿易法」に定める戦略物資に該当するものについては、輸出する場合、同法に基づく輸出許可が必要です。

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S1C17F00シリーズ向け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。

S1C17F00 シリーズ向けサンプルソフトウェアは S1C17F00 シリーズ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。

S1C17F00 シリーズ向けサンプルソフトウェアはインストールの簡便さなどから、機種毎に提供していますが、各機能の基本的な動作は同じです。

テクニカルマニュアル、開発ツールマニュアル（S5U1C17001C マニュアル、S5U1C17001H User Manual）と合わせてご覧下さい。

1.1. 動作環境

S1C17F00 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。

- S1C17F00 が実装されたボード
- 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、GNU17v2.0.0 で動作確認を行っています。

2. サンプルソフトウェア説明

本章では S1C17F00 シリーズサンプルソフトウェアのファイル構成と実行方法を記載します。

S1C17F00 シリーズサンプルソフトウェアは、 “サンプルソフトウェア”と、“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 S1C17F00 シリーズサンプルソフトウェアのディレクトリ構成を示します。

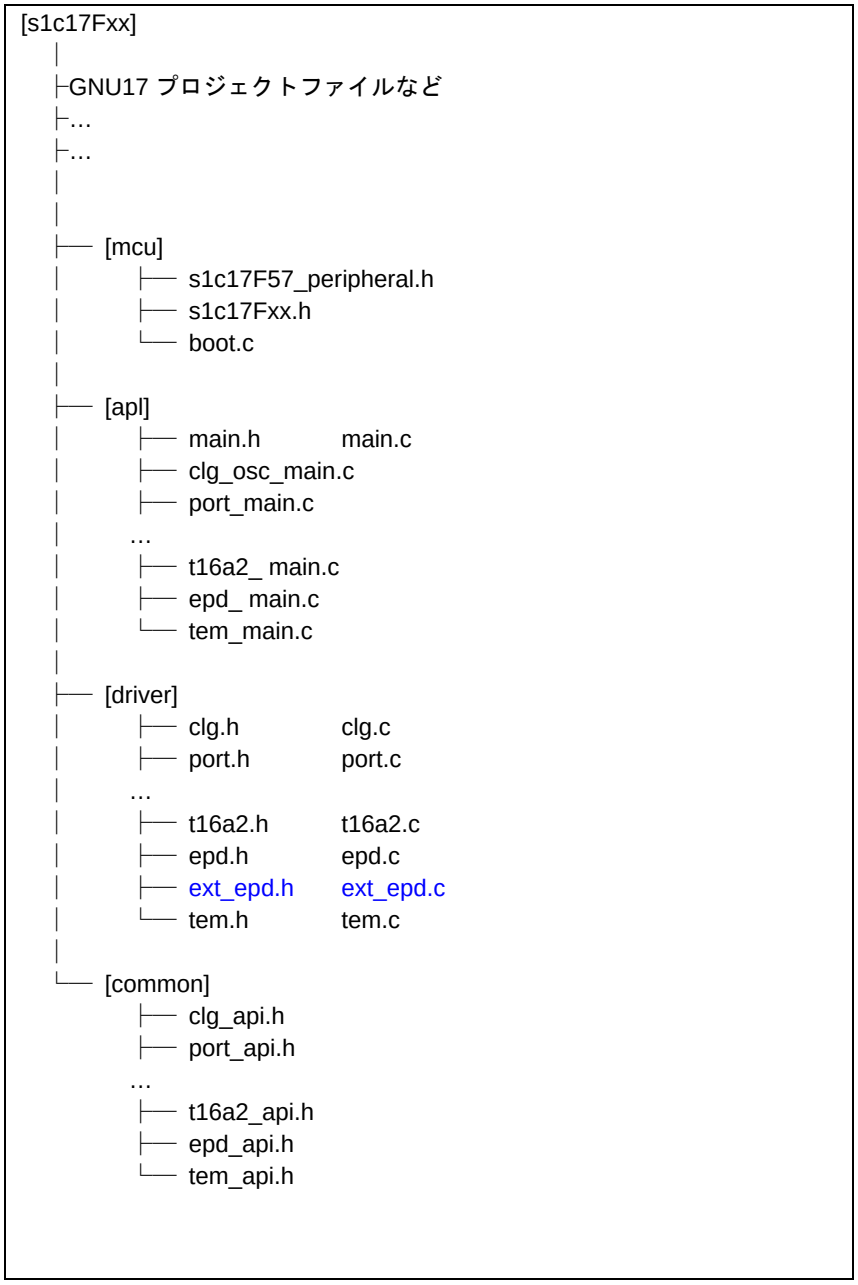

図 2.1 S1C17F00 シリーズサンプルソフトウェアディレクトリ構成図

（1）s1c17Fxx ディレクトリ

本ディレクトリには GNU17 のプロジェクトに関係するファイルと、サンプルソフトウェアのソースコードが格納されているディレクトリが配置されています。

（2）mcu ディレクトリ

マイコンの初期化処理と各機種に依存する情報を定義したファイルが配置してあります。

- 対象機種のレジスタアドレスなどが定義されたヘッダファイル（s1c17F57_peripheral.h など）
- 機種共通のヘッダファイル（s1c17Fxx.h）
- 初期化処理のファイル（boot.c）

（3）apl ディレクトリ

周辺回路毎のサンプルソフトウェアとサンプルソフトウェア内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。

- 周辺回路毎のヘッダファイル（xxx.h）
- 周辺回路毎のサンプルソフトウェア（xxx.c）

（4）driver ディレクトリ

周辺回路毎のサンプルドライバが配置してあります。

- 周辺回路毎のレジスタアドレスやビットアサインを定義したヘッダファイル（xxx.h）
- 周辺回路毎のプログラム（xxx.c）

（5）common ディレクトリ

周辺回路毎のサンプルドライバが外部に提供する関数のプロトタイプを定義したヘッダファイルが配置してあります。

- 周辺回路毎のサンプルドライバが外部に提供する関数プロトタイプと引数の定数定義をしたヘッダファイル（xxx.h）

サンプルドライバを使用するソフトウェアは common ディレクトリに配置してあるヘッダファイルをインクルードしてサンプルドライバの関数を呼び出します。

2.2 実行方法

以下の手順で S1C17F00 シリーズサンプルソフトウェアを実行して下さい。

（1）プロジェクトをインポート

GNU17 を起動して S1C17F00 シリーズサンプルソフトウェアのプロジェクトをインポートして下さい。

プロジェクトインポート方法の詳細につきましては S5U1C17001C Manual の“3 ソフトウェア開発手順”を参照して下さい。

（2）プロジェクトをビルド

GNU17 で S1C17Fxx プロジェクトをビルドして下さい。

ビルド方法の詳細につきましては S5U1C17001C Manual の“5 GNU17 IDE”を参照して下さい。

（3）ICDmini を接続

ICDmini を PC、開発ボードに接続して開発ボードの電源を投入して下さい。

（4）デバッガによるプログラムのロードと実行

GNU17 のデバッガを起動、実行してください。プログラムが S1C17F00 にロードされプログラムが起動します。

デバッガ使用方法の詳細につきましては S5U1C17001C Manual の“10 デバッガ”を参照して下さい。

2.3 サンプルソフトウェアメニュー

サンプルソフトウェアを起動すると、GNU17 のシミュレーテッド I/O(以下 SimI/O)にメニュー画面が表示されます(サンプルソフトウェアの起動前に、新規コンソールビューを表示してください)。

プログラム番号を入力して Enter キーを押下すると選択したサンプルソフトウェアが起動します。

各サンプルソフトウェアの詳細は 3 章を参照して下さい。

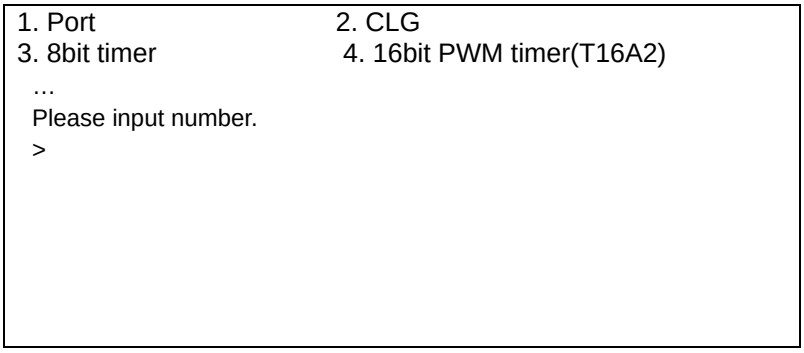

図 2.2 メニュー画面表示例

2.4 特定モジュールのビルド方法

S1C17Fxx サンプルソフトウェアは複数のプログラムがビルドされる状態で配布しております。

サンプルソフトウェアのソースコードを修正することで必要な周辺モジュールのサンプルソフトウェアだけをビルドすることができます。

以下に手順を示します。

（1）修正対象ファイル

機種別の定義ヘッダを修正します。

S1C17F57 サンプルソフトウェアの場合は s1c17F57_periperal.h を修正します。

（2）修正箇所

ファイル下部にある以下の箇所を修正します。

```
//#undef  PE_MISC
//#undef  PE_UART
//#undef  PE_UART_OSC3A
//#undef  PE_UART_OSC3B
//#undef  PE_T8
//#undef  PE_SPI
//#undef  PE_SPI_MASTER
//#undef  PE_SPI_SLAVE
//#undef  PE_I2CM
//#undef  PE_I2CS
//#undef  PE_CT
//#undef  PE_SWT
#undef    PE_WDT
#undef    PE_CLG_OSC
#undef    PE_CLG
#undef    PE_SVD
#undef    PE_VD1
#undef    PE_SND
#undef    PE_TEM
#undef    PE_PORT
#undef    PE_MUX
#undef    PE_MISC2
#undef    PE_RFC
#undef    PE_T16A2
#undef    PE_EPD
#undef    PE_EXT_EPD
#undef    PE_RTC
#undef    PE_TR
#undef    PE_SLEEP_HALT
```

図 2.3 特定モジュールの定義変更例

例えば入出力ポートサンプルソフトウェアのみビルドする場合、“#undef PE_PORT”の定義を無効にし、それ以外の“#undef PE_XXX”定義を有効にします。

ビルドする周辺モジュールのサンプルソフトウェアが他の周辺モジュールを使用する場合、使用する周辺モジュールサンプルソフトウェアもビルドする必要があります。

例えば I2CM サンプルソフトウェアは 8 ビットタイマを使用しますので、I2CM サンプルソフトウェアをビルドする際は、“#undef PE_I2CM”と“#undef PE_T8”の定義を無効にする必要があります。

3　サンプルソフトウェア機能詳細

本章では S1C17F00 シリーズサンプルソフトウェアの機能詳細について記載します。

3.1　入出力ポート（P）

3.1.1　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。

- ポートを入力割り込みに設定し、入力信号が Low レベルになったことを検出する。
- ポートを出力に設定し、High レベルや Low レベルの信号を出力する。

使用するポート設定とポート名は以下の通りです。

表 3.1　入出力ポート設定一覧

設定	ポート名
入力割り込みポート	P01 P02 P03
出力ポート	P12 P11

注 ポート設定は機種によって変更がある場合があります。各機種のソースを参照してご確認下さい。

3.1.2　ハードウェア条件

本サンプルソフトウェアは OSC1 と OSC3A が発振可能な状態で動作します。

振動子の接続方法につきましては、各テクニカルマニュアル「クロックジェネレータ（CLG）発振回路（OSC)」を参照して下さい。

本サンプルソフトウェアは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。

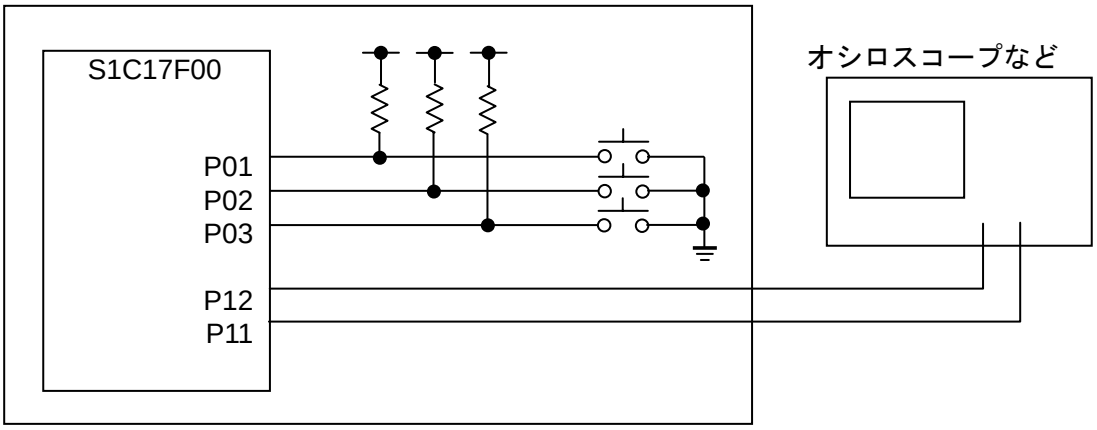

図 3.1　入出力ポート（P）サンプルソフトウェアハードウェア接続図

3.1.3　動作概要

（1）サンプルソフトウェア動作概要

- P01 ポートの入力信号を Low レベルにすると SimI/O に「P01 Interrupt」と表示し、P11 ポートの出力を反転させます。(High レベルであれば Low レベル、Low レベルであれば High レベルにします。)
- P02 ポートの入力信号を Low レベルにすると SimI/O に「P02Interrupt」と表示し、P12 ポートの出力を反転させます。(High レベルであれば Low レベル、Low レベルであれば High レベルにします。)

```
<<< Port demonstration start >>>
*** P01 Interrupt ***
*** P02 Interrupt ***

<<< Port demonstration finish >>>
```

図 3.2　入出力ポート（P）サンプルソフトウェア画面表示例

（2）サンプルソフトウェアの停止方法

P03 ポートの入力信号を Low レベルにすると、サンプルソフトウェアを終了してメニュー画面に戻ります。

3.2　クロックジェネレータ（CLG）

3.2.1　サンプルソフトウェア仕様

本サンプルソフトウェアは発振回路を使用して以下の動作を行います。

- OSC3B の発振と停止を行う。
- OSC1 の発振と停止を行う。
- OSC3A の発振と停止を行う。
- システムクロックを OSC3B から OSC3A へ切り替える。
- システムクロックを OSC3A から OSC1 へ切り替える。
- システムクロックを OSC1 から OSC3B へ切り替える。

3.2.2　ハードウェア条件

本サンプルソフトウェアは OSC1 と OSC3A が発振可能な状態で動作します。

3.2.3　動作概要

（1）サンプルソフトウェア動作概要

- 本サンプルソフトウェアは OSC3B を使用した状態で動作を開始します。
- 一定間隔で SimI/O に「1」、「2」、「3」…、「9」と表示した後、OSC3A の発振を開始してシステムクロックを OSC3B から OSC3A に切り替え、OSC3B を停止します。
- 次に一定間隔で SimI/O に「1」、「2」、「3」…、「9」と表示した後、OSC1 の発振を開始してシステムクロックを OSC3A から OSC1 に切り替え、OSC3A を停止します。
- 次に一定間隔で SimI/O に「1」、「2」、「3」…、「9」と表示した後、OSC3B の発振を開始してシステムクロックを OSC1 から OSC3B に切り替え、OSC1 を停止します。
- 次に一定間隔で SimI/O に「1」、「2」、「3」…、「9」と表示します。

```
<<< CLGdemonstration start >>>
OSC3B *** 1 ***
OSC3B*** 2 ***
…
OSC3B *** 9 ***
*** Change from OSC3B to OSC3A ***
OSC3A *** 1 ***
OSC3A *** 2 ***
…
OSC3A *** 9 ***
<<< CLGdemonstration finish >>>
```

図 3.3　クロックジェネレータ（CLG）サンプルソフトウェア画面表示例

（2）サンプルソフトウェアの停止方法

上記“サンプルソフトウェア動作概要”に記載された全ての動作を終了すると、サンプルソフトウェアを終了してメニュー画面に戻ります。

3.3　8 ビットタイマ（T8）

3.3.1　サンプルソフトウェア仕様

本サンプルソフトウェアは 8 ビットタイマを使用して以下の動作を行います。

- 8 ビットタイマ割り込みを発生させ、タイマのカウンタ値を取得する。
- 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。

3.3.2　ハードウェア条件

本サンプルソフトウェアは OSC1 と OSC3A が発振可能な状態で動作します。

3.3.3　動作概要

（1）サンプルソフトウェア動作概要

- 8 ビットタイマ割り込みを開始し、CPU を halt モードにします。
- 8 ビットタイマ割り込みが発生すると CPU の halt モードが解除され、8 ビットタイマのカウンタ値を内部変数に保存して再び CPU を halt モードにします。
- 8 ビットタイマ割り込みが 10 回発生したところで 8 ビットタイマを停止し、各割り込みが発生したときのカウンタ値を SimI/O に表示します。

```
<<< T8 timer demonstration start >>>
*** T8 interrupt 1 time, count data at this time : 32 ***
*** T8 interrupt 2 time, count data at this time : 32 ***
*** T8 interrupt 3 time, count data at this time : 32 ***
*** T8 interrupt 4 time, count data at this time : 32 ***
…
*** T8 interrupt 10 time, count data at this time : 32 ***
<<< T8 timer demonstration finish >>>
```

図 3.4　16 ビットタイマ（T8）サンプルソフトウェア画面表示例

（2）サンプルソフトウェアの停止方法

上記“サンプルソフトウェア動作概要”に記載された全ての動作を終了すると、サンプルソフトウェアを終了してメニュー画面に戻ります。

3.4　PWM タイマ（T16A2）

3.4.1　サンプルソフトウェア仕様

本サンプルソフトウェアは PWM タイマを使用して以下の動作を行います。

- PWM タイマコンペア A マッチ割り込みをノーマルモードで 5 回発生させ、タイマのカウンタ値を取得する。
- PWM タイマコンペア B マッチ割り込みをノーマル/ハーフクロックモードでそれぞれ 5 回発生させ、タイマのカウンタ値を取得する。
- TOUTA1 端子に PWM 波形をノーマル/ハーフクロックモードで出力する。
- 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。

3.4.2　ハードウェア条件

本サンプルソフトウェアは OSC3A に水晶振動子またはセラミック振動子が接続された状態で動作します。

振動子の接続方法につきましては、S1C177xx シリーズテクニカルマニュアル「クロックジェネレータ（CLG）」をご参照下さい。

本サンプルソフトウェアは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。

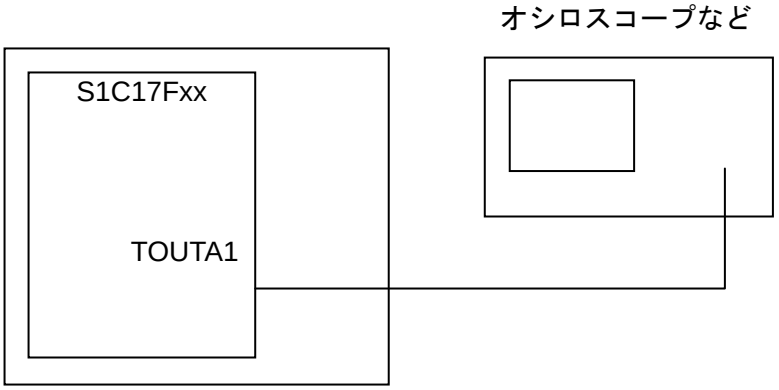

図 3.5　PWM タイマ（T16A2）サンプルソフトウェアハードウェア接続図

3.4.3　動作概要

（1）サンプルソフトウェア動作概要

- ノーマルクロックモードに設定し、コンペア A マッチ割り込みとコンペア B マッチ割り込みを有効にして PWM タイマーを開始します。
- コンペア A マッチ割り込み、およびコンペア B マッチ割り込みが発生したときにアップカウンタのカウンタ値を取得します。
- コンペア B マッチ割り込みが 5 回発生したところで PWM タイマを停止し、割り込みの種類とカウンタ値を取得し SimI/O に表示します。
- ハーフクロックモードに設定し、コンペア A マッチ割り込みを無効にして PWM タイマーを開始します。
- コンペア B マッチ割り込みが 5 回発生したところで PWM タイマを停止し、割り込みの種類とカウンタ値を取得し SimI/O に表示します。

```
<<< PWM timer(T16A2) demonstration start >>>
Normal clock mode start
*** PWM compare A interrupt :633 ***
*** PWM compare B interrupt :    0 ***
*** PWM compare A interrupt :633 ***
…
*** PWM Interrupt B interrupt:     0 ***

Half clock mode start
*** PWM Interrupt B interrupt:     0 ***
・・・
<<< PWM timer demonstration finish >>>
```

図 3.6　PWM タイマ（T16A2）サンプルソフトウェア画面表示例

（2）サンプルソフトウェアの停止方法

上記“サンプルソフトウェア動作概要”に記載された全ての動作を終了すると、サンプルソフトウェアを終了してメニュー画面に戻ります。

3.5　計時タイマ（CT）

3.5.1　サンプルソフトウェア仕様

本サンプルソフトウェアは計時タイマを使用して以下の動作を行います。

- 計時タイマ割り込みを発生させ、経過時間を算出する。
- 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。

3.5.2　ハードウェア条件

本サンプルソフトウェアは OSC1 と OSC3A が発振可能な状態で動作します。

3.5.3　動作概要

（1）サンプルソフトウェア動作概要

- 計時タイマを開始し、CPU を halt モードにします。
- 計時タイマ割り込みが発生すると CPU の halt モードが解除され、プログラム開始時からの経過時間を算出し SimI/O に経過時間を表示して再び CPU を halt モードにします。
- 計時タイマ割り込みが 10 回発生したところで計時タイマを停止します。

```
<<< Clock timer demonstration start >>>
*** 0.5 sec ***
*** 1.0 sec ***
*** 1.5 sec ***
…
*** 5.0 sec ***
<<< Clock timer demonstration finish >>>
```

図 3.7　計時タイマ（CT）サンプルソフトウェア画面表示例

（2）サンプルソフトウェアの停止方法

上記“サンプルソフトウェア動作概要”に記載された全ての動作を終了すると、サンプルソフトウェアを終了してメニュー画面に戻ります。

3.6　ストップウォッチタイマ（SWT）

3.6.1　サンプルソフトウェア仕様

本サンプルソフトウェアはストップウォッチタイマを使用して以下の動作を行います。

- ストップウォッチタイマ割り込みを発生させ、経過時間を算出する。

3.6.2　ハードウェア条件

本サンプルソフトウェアは OSC1 と OSC3A が発振可能な状態で動作します。

3.6.3　動作概要

（1）サンプルソフトウェア動作概要

- 1 から 9 の数字を入力して ENTER キーを押下することで割り込みの回数を指定することができます。
- ストップウォッチタイマを開始し、1Hz のストップウォッチタイマ割り込みが指定回数発生したところで経過時間を SimI/O に表示してストップウォッチタイマを停止します。

```
<<< Stop watch timer demonstration start >>>
Please input time 1-9[sec]
4
Start stopwatch timer…
4 sec passed
<<< Stop watch timer demonstration finish >>
```

図 3.8　ストップウォッチタイマ（SWT）サンプルソフトウェア画面表示例

（2）サンプルソフトウェアの停止方法

上記“サンプルソフトウェア動作概要”に記載された全ての動作を終了すると、サンプルソフトウェアを終了してメニュー画面に戻ります。

3.7　ウォッチドッグタイマ（WDT）

3.7.1　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による NMI 割り込みを発生させる。

3.7.2　ハードウェア条件

本サンプルソフトウェアは OSC1 と OSC3A が発振可能な状態で動作します。

3.7.3　動作概要

（1）サンプルソフトウェア動作概要

- ウォッチドッグタイマと 8 ビットタイマを開始します。
- 8 ビットタイマ割り込みが発生するとウォッチドッグタイマをクリアします。
- 8 ビットタイマ割り込みが 10 回発生すると 8 ビットタイマを停止しします。
- ウォッチドッグタイマによる NMI 割り込みが発生すると SimI/O にメッセージを表示します。

```
<<< Watchdog timer demonstration start >>>
*** T8 timer : reset watchdog timer ***
*** T8 timer : reset watchdog timer ***
*** T8 timer : reset watchdog timer ***
…
*** T8 timer : reset watchdog timer ***
*** stop T8 timer ***
*** NMI occurred ***
<<< Watchdog timer demonstration finish >>>
```

図 3.9　ウォッチドッグタイマ（WDT）サンプルソフトウェア画面表示例

（2）サンプルソフトウェアの停止方法

上記“サンプルソフトウェア動作概要”に記載された全ての動作を終了すると、サンプルソフトウェアを終了してメニュー画面に戻ります。

3.8　OSC3A を使用した UART

3.8.1　サンプルソフトウェア仕様

本サンプルソフトウェアは UART を使用して以下の動作を行います。

- UART を使用してデータを送信する。
- UART を使用してデータを受信する。

3.8.2　ハードウェア条件

本サンプルソフトウェアは OSC1 と OSC3A が発振可能な状態で動作します。

本サンプルソフトウェアは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。

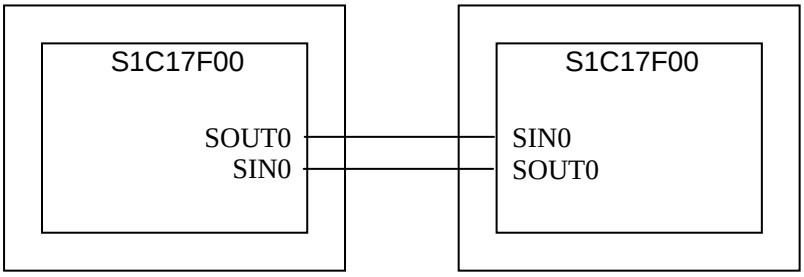

図 3.10　OSC3A を使用した UART サンプルソフトウェアハードウェア接続図

3.8.3　動作概要

（1）サンプルソフトウェア動作概要

- UART ポートを通信速度 115200bps/データ長 8bit/ストップビット 1bit/パリティ無しに初期化します。
- 接続確認フラグ“0x7F”を受信するまで“0x7F”を送信し続けます。
- 接続確認フラグを受信すると“0x7F”の送信を停止し、ASCII コード 0x21～0x7E のデータを UART ポートに送信します。
- 全てのデータを送信し、かつ 34 バイトのデータを受信すると受信データを SimI/O に表示します。

```
<<< UART OSC3A demonstration start >>>
waiting connection.
connected.
*** sent data ***
*** received data ***
ABCDEFG…
<<< UART OSC3A demonstration finish >>>
```

図 3.11　OSC3A を使用した UART サンプルソフトウェア画面表示例

（2）サンプルソフトウェアの停止方法

上記“サンプルソフトウェア動作概要”に記載された全ての動作を終了すると、サンプルソフトウェアを終了してメニュー画面に戻ります。

3.9　OSC3B を使用した UART

3.9.1　サンプルソフトウェア仕様

本サンプルソフトウェアは UART を使用して以下の動作を行います。

- OSC3B クロックと OSC1 のカウンタ値を比較し、OSC3B の周波数を算出する。
- OSC3B を UART クロックとして設定する。
- UART を使用してデータを送信する。
- UART を使用してデータを受信する。

3.9.2　ハードウェア条件

ハードウェア条件は OSC3A を使用した UART サンプルソフトウェアと同一です。

3.9.3　動作概要

（1）サンプルソフトウェア動作概要

- OSC3B を使った 8 ビットタイマと OSC1 を使ったアドバンスドタイマを動作させ、OSC3B の発振周波数を計算します。
- 計算した OSC3B 発振周波数を元に UART ポートを通信速度 57600bps/データ長 8bit/ストップビット 1bit/パリティ無しに初期化します。
- 接続確認フラグ“0x7F”を受信するまで“0x7F”を送信し続けます。
- 接続確認フラグを受信すると“0x7F”の送信を停止し、ASCII コード 0x21～0x7E のデータを UART ポートに送信します。
- 全てのデータを送信し、かつ 34 バイトのデータを受信すると受信データを SimI/O に表示します。

```
<<< UART OSC3B demonstration start >>>
waiting connection.
connected.
*** sent data ***
*** received data ***
ABCDEFG…
<<< UART OSC3B demonstration finish >>>
```

図 3.12　OSC3B を使用した UART サンプルソフトウェア画面表示例

（2）サンプルソフトウェアの停止方法

上記“サンプルソフトウェア動作概要”に記載された全ての動作を終了すると、サンプルソフトウェアを終了してメニュー画面に戻ります。

3.9.4　OSC3B 発振周波数計算方法

OSC3B を使って UART を使用する際のタイマカウンタ設定値を算出する手順を以下に示します。

- T16A2 のコンペアデータを 4 カウントに設定し、OSC3B を使った T8 と OSC1 を使った T16A2 をスタートさせます。
- T16A2 のコンペアマッチ割り込みが発生したときに T8 を停止します。
- T8 停止後のカウンタ値を読み出し OSC3B の周波数を求めます。
- n×8192÷(div×bps)の商が UART_BR+1、剰余が UART_FMD になります。（n=OSC3B のカウンタ値、div=カウントクロック分周比の逆数、bps＝UART のビットレート）

3.10　SPI マスタ

3.10.1　サンプルソフトウェア仕様

本サンプルソフトウェアは SPI マスタを使用して以下の動作を行います。

- 8byte のデータを SPI スレーブに送信する。
- 8byte のデータを SPI スレーブから受信する。
- 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。

3.10.2　ハードウェア条件

本サンプルソフトウェアは OSC1 と OSC3A が発振可能な状態で動作します。

本サンプルソフトウェアは SPI スレーブサンプルソフトウェアが動作している S1C17F00 を SPI スレーブとして接続し、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。

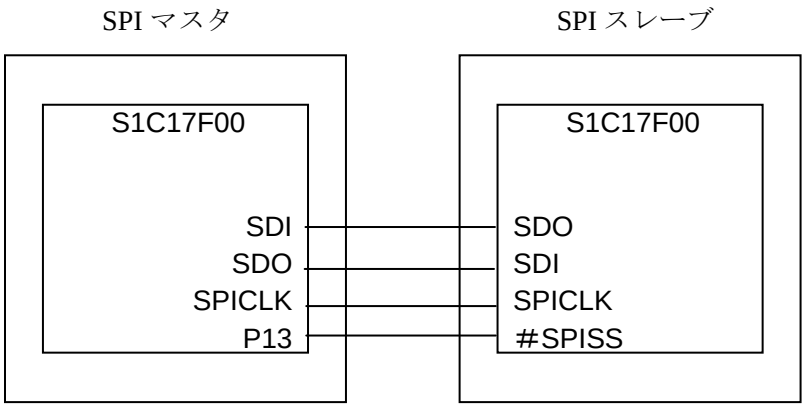

図 3.13　SPI マスタ、スレーブサンプルソフトウェアハードウェア接続図

注 各端子は機種により変更される場合があります。ソースコードをご確認下さい。

3.10.3　動作概要

（1）サンプルソフトウェア動作概要

- SPI マスタの初期設定を行い、SPI スレーブへ 8byte の ASCII データ「FROM MST」を送信します。
- SPI スレーブへのデータ送信を終了すると ENTER キーの入力待ちになります。
- ENTER キーを押下すると SPI スレーブへ SPI クロックを出力してデータの受信を待ちます。
- SPI スレーブからデータを受信すると受信データを SimI/O に表示します。

```
<<< SPI master demonstration start >>>
Transmitted data : FROM MST
please press enter key

Received data : FROM SLV
<<< SPI master demonstration finish >>>
```

図 3.14　SPI マスタサンプルソフトウェア画面表示例

（2）サンプルソフトウェアの停止方法

上記“サンプルソフトウェア動作概要”に記載された全ての動作を終了すると、サンプルソフトウェアを終了してメニュー画面に戻ります。

3.11　SPI スレーブ

3.11.1　サンプルソフトウェア仕様

本サンプルソフトウェアは SPI スレーブを使用して以下の動作を行います。

- 8byte のデータを SPI マスタから受信する。
- 8byte のデータを SPI マスタに送信する。
- 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。

3.11.2　ハードウェア条件

ハードウェア条件は SPI マスタサンプルソフトウェアと同一です。

本サンプルソフトウェアは SPI マスタサンプルソフトウェアが動作している S1C17F00 を SPI マスタとして接続してご使用下さい。

3.11.3　動作概要

（1）サンプルソフトウェア動作概要

- SPI スレーブの初期設定を行い、SPI マスタからのデータ受信を待ちます。
- SPI マスタからデータを受信すると受信したデータを SimI/O に表示し、SPI マスタへ 8byte の ASCII データ「FROM SLV」を送信します。

```
<<< SPI slave demonstration start >>>
Received data : FROM MST
Transmitted data : FROM SLV
<<< SPI slave demonstration finish >>>
```

図 3.15　SPI スレーブサンプルソフトウェア画面表示例

（2）サンプルソフトウェアの停止方法

上記“サンプルソフトウェア動作概要”に記載された全ての動作を終了すると、サンプルソフトウェアを終了してメニュー画面に戻ります。

3.12　I2C マスタ（I2CM）

3.12.1　サンプルソフトウェア仕様

本サンプルソフトウェアは I2C マスタを使用して以下の動作を行います。

- I2C スレーブへデータを送信する。
- I2C スレーブからデータを受信する。

3.12.2　ハードウェア条件

本サンプルソフトウェアは OSC1 と OSC3A が発振可能な状態で動作します。

本サンプルソフトウェアは I2C スレーブサンプルソフトウェアが動作している S1C17F00 マイコンを I2C スレーブとして接続し、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。

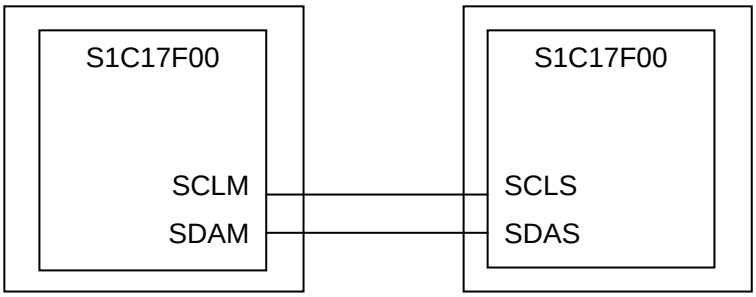

図 3.16　I2C マスタ（I2CM）、スレーブ（I2CS）サンプルソフトウェアハードウェア接続図

3.12.3　動作概要

（1）サンプルソフトウェア動作概要

- I2C マスタの初期設定を行い、I2C スレーブへ 8byte の ASCII データ「FROM MST」を送信します。
- I2C スレーブへのデータ送信を終了すると ENTER キーの入力待ちになります。
- ENTER キーを押下すると I2C スレーブからのデータ受信を待ちます。
- I2C スレーブからデータを受信すると受信したデータを SimI/O に表示します。

```
<<< I2C master demonstration start >>>
Transmitted data : FROM MST
please press enter key

Received data : FROM SLV
<<< I2C master demonstration finish >>>
```

図 3.17　I2C マスタ（I2CM）サンプルソフトウェア画面表示例

（2）サンプルソフトウェアの停止方法

上記“サンプルソフトウェア動作概要”に記載された全ての動作を終了すると、サンプルソフトウェアを終了してメニュー画面に戻ります。

3.13　I2C スレーブ（I2CS）

3.13.1　サンプルソフトウェア仕様

本サンプルソフトウェアは I2C スレーブを使用して以下の動作を行います。

- I2C マスタからデータを受信する。
- I2C マスタへデータを送信する。

3.13.2　ハードウェア条件

ハードウェア条件は I2C マスタ（I2CM）サンプルソフトウェアと同一です。

本サンプルソフトウェアは I2C マスタ（I2CM）サンプルソフトウェアが動作している S1C17F00 マイコンを I2C マスタとして接続してご使用下さい。

3.13.3　動作概要

（1）サンプルソフトウェア動作概要

- I2C スレーブの初期設定を行い、I2C マスタからのデータ受信を待ちます。
- I2C マスタからデータを受信すると受信したデータを SimI/O に表示し、I2C マスタへ 8byte の ASCII データ「FROM SLV」を送信します。

```
<<< I2C slave demonstration start >>>
Received data : FROM MST
Transmitted data : FROM SLV
<<< I2C slave demonstration finish >>>
```

図 3.18　I2C スレーブ（I2CS）サンプルソフトウェア画面表示例

（2）サンプルソフトウェアの停止方法

上記“サンプルソフトウェア動作概要”に記載された全ての動作を終了すると、サンプルソフトウェアを終了してメニュー画面に戻ります。

3.14　EPD コントローラ/ドライバ（EPD）

3.14.1　サンプルソフトウェア仕様

本サンプルソフトウェアは EPD コントローラ/ドライバを使用して以下の動作を行います。

- EPD コントローラ/ドライバ/拡張 EPD コントローラ/ドライバ(S1D14F51)それぞれの電源の設定を行う。
- EPD コントローラ/ドライバ/拡張 EPD コントローラ/ドライバ(S1D14F51)それぞれの動作モードの設定を行う。
- EPD コントローラが生成する駆動波形を EPD /拡張 EPD それぞれに設定を行う。
- 表示データの設定を行う。
- SEG15 端子または SEG47 端子/SEG7 端子より波形を出力する（表示：白→黒→黒→白）。

3.14.2　ハードウェア条件

本サンプルソフトウェアは OSC1 と OSC3A が発振可能な状態で動作します。

本サンプルソフトウェアは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。

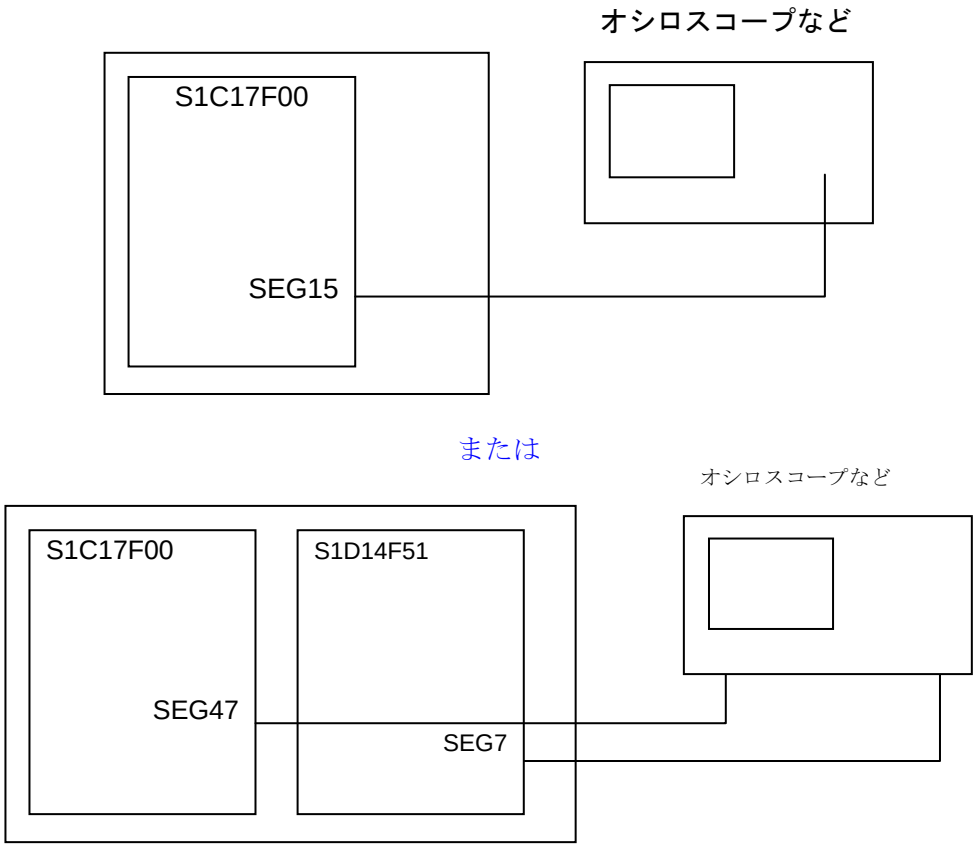

図 3.19.　EPD コントローラ/ドライバサンプルプログラムハードウェア接続図

3.14.3　動作概要

（1）サンプルソフトウェア動作概要

- メニューから「1」を入力して ENTER キーを押下すると EPD の SEG15 端子より波形が出力（白→黒→黒→白の順に表示します。）されます。
- メニューから「2」を入力して ENTER キーを押下すると EPD の SEG47 端子、拡張 EPD の SEG7 端子より波形が出力（白→黒→黒→白の順に表示します。）されます。
- メニューから「3」を入力して ENTER キーを押下するとサンプルソフトウェアを終了します。

```
<<< EPD Controller demonstration start >>>
1. EPD                    2. EPD(using S1D14F51)
3.exit
>2
SEG47 White,    SEG7 White
SEG47 Black,    SEG7 Black
SEG47 Black,    SEG7 Black
SEG47 White,    SEG7 White
<<< EPD Controller r demonstration finish >>>
```

図 3.20　EPD コントローラ/ドライバサンプルプログラム画面表示例

（2）サンプルソフトウェアの停止方法

メニューから「3」を入力して ENTER キーを押下することで、サンプルプログラムを終了してメニュー画面に戻ります。

3.15　電源電圧検出回路（SVD）

3.15.1　サンプルソフトウェア仕様

本サンプルソフトウェアは電源電圧検出回路（以下、SVD 回路）を使用して以下の動作を行います。

- SVD 回路を用いて電源電圧の検出を行う。

3.15.2　ハードウェア条件

本サンプルソフトウェアは OSC1 と OSC3A が発振可能な状態で動作します。

任意の電源電圧を設定して動作させてください。

3.15.3　動作概要

（1）サンプルソフトウェア動作概要

- SVD 回路を用いて電源電圧（VDD）の検出を行い、現在の VDD 電圧を SimI/O に表示します。比較電圧は 1.2V～3.2V です。
- 電源電圧値 1.2V 未満、3.2V 以上の場合、SimI/O に「SVD interrupt did not occurred」と表示されます。

```
<<< SVD demonstration start >>>
Vdd=2.5V
<<< SVD demonstration finish >>>
```

図 3.21　電源電圧検出回路（SVD）サンプルソフトウェア画面表示例

注 検出電圧は機種によって変更がある場合があります。各機種のソースコードを参照してご確認下さい。

（2）サンプルソフトウェアの停止方法

上記“サンプルソフトウェア動作概要”に記載された全ての動作を終了すると、サンプルソフトウェアを終了してメニュー画面に戻ります。

3.16　R/F 変換器（RFC）

3.16.1　サンプルソフトウェア仕様

本サンプルソフトウェアは R/F 変換器を使用して以下の動作を行います。

- 抵抗性センサー測定用 DC 発振モードで発振させカウンタ値の取得を行う。
- 抵抗性センサー測定用 AC 発振モードで発振させカウンタ値の取得を行う。

3.16.2　ハードウェア条件

本サンプルソフトウェアは OSC1 と OSC3A が発振可能な状態で動作します。

本サンプルソフトウェアは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。

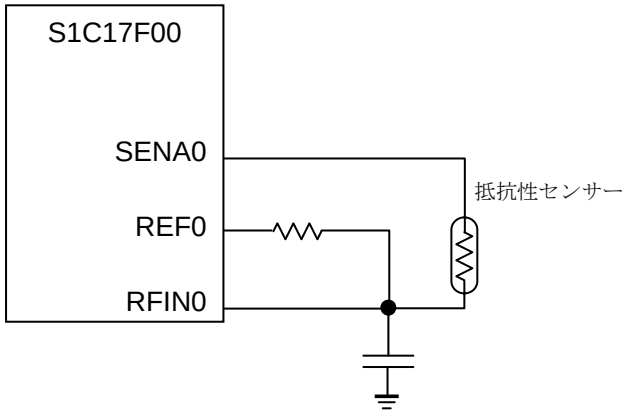

図 3.22　R/F 変換器（RFC DC）サンプルソフトウェアハードウェア接続図

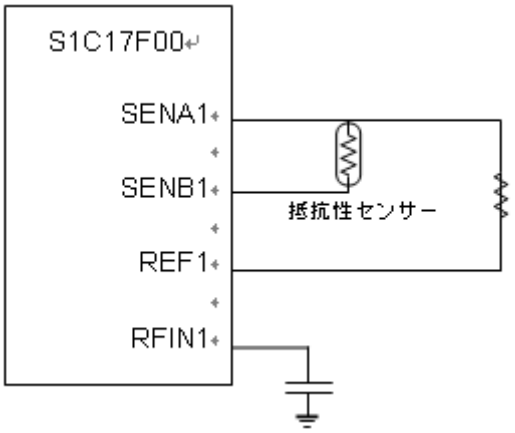

図 3.23　R/F 変換器（RFC AC）サンプルソフトウェアハードウェア接続図

3.16.3　動作概要

（1）サンプルソフトウェア動作概要

- 抵抗性センサー測定用 DC 発振モードを設定します。
- 基準発振を開始し、発振終了でカウンタ値を取得し SimI/O に表示します。
- センサーA 発振を開始し、発振終了でカウンタ値を取得し SimI/O に表示します。

```
<<< RFC demonstration start >>>
1.DC mode   2.AC mode
Please input number.
>1
Reference
measurement counter : 0000
time base counter counter : 0000
Sensor A
measurement counter : 0000
time base counter counter : 0000
<<< RFC demonstration finish >>>
```

図 3.24　R/F 変換器（RFC）サンプルソフトウェア画面表示例

（2）サンプルソフトウェアの停止方法

上記“サンプルソフトウェア動作概要”に記載された全ての動作を終了すると、サンプルソフトウェアを終了してメニュー画面に戻ります。

3.17　リアルタイムクロック（RTC）

3.17.1　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取得する。
- 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割り込み発生回数を表示する。

3.17.2　ハードウェア条件

本サンプルソフトウェアは OSC3A に水晶振動子またはセラミック振動子が接続された状態で動作します。

振動子の接続方法につきましては、S1C177xx シリーズテクニカルマニュアル「クロックジェネレータ（CLG）」をご参照下さい。

3.17.3　動作概要

（1）サンプルソフトウェア動作概要

- プログラム開始後、RTC サンプルプログラムのメニューを表示します。
- メニューから「1」を入力して ENTER キーを押下することで RTC の時刻を取得し、24 時間モード又は 12 時間モードで表示します。
- メニューから「2」を入力して ENTER キーを押下することで、24 時間モード又は 12 時間モードの設定と RTC の時刻を設定します。
- メニューから「3」を入力して ENTER キーを押下することで、RTC 割り込みの回数を表示します。

```
<<< Real Time Clock demonstration start >>>
1.get RTC                          2.set RTC
3.indicate the count of interrupt   4.exit
Please input number.
>1
10:00:00

1.get RTC                          2.set RTC
3.indicate the count of interrupt   4.exit
Please input number.
>2
> Input 24H mode.
>   24H :1 or 12H :2
1
> Input BCD format.
> Hour (00 –23)
10
> Minute (0 - 59)
30
> Second (0 - 59)
15
1.get RTC                          2.set RTC
3.indicate the count of interrupt   4.exit
Please input number.
>3
> interrupt count value = xx
```

```
1.get RTC                          2.set RTC
3.indicate the count of interrupt  4.exit
>4
<<< Real Time Clock demonstration finish >>>
```

図 3.25　リアルタイムクロック サンプルソフトウェア画面表示例

（2）サンプルソフトウェアの停止方法

メニューから「4」を入力して ENTER キーを押下することで、サンプルプログラムを終了してメニュー画面に戻ります。

3.18　サウンドジェネレータ（SND）

3.18.1　サンプルソフトウェア仕様

本サンプルソフトウェアは以下の状態を評価するためのプログラムです。

- サウンドジェネレータを使い、周波数を変えながら BZ 端子より出力する。

3.18.2　ハードウェア条件

本サンプルソフトウェアは OSC1 と OSC3A が発振可能な状態で動作します。
本サンプルソフトウェアはマイコンのポートを以下のように接続してご使用下さい。

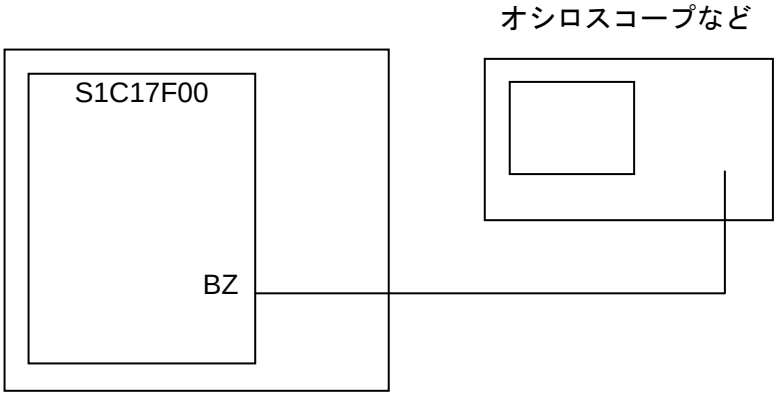

図 3.26. サウンドジェネレータ（SND）サンプルソフトウェアハードウェア接続図

3.18.3　動作概要

（1）サンプルソフトウェア動作概要

- メニューが表示された状態で「1」を入力して ENTER キーを押下すると、BZ 端子からノーマルモードで周波数を変えながら出力します。
- メニューが表示された状態で「2」を入力して ENTER キーを押下すると、BZ 端子からワンショットモードで周波数を変えながら出力します。
- メニューが表示された状態で「3」を入力して ENTER キーを押下すると、BZ 端子からエンベロープモードで周波数を変えながら出力します。

```
<<< SND demonstration start >>>
1.Normal                    2.One shot
3.Envelop                   4.exit
Please input number.
>1
1070.3Hz
1365.6Hz
…
<<< SND demonstration finish >>>
```

図 3.27　サウンドジェネレータサンプルソフトウェア画面表示例

（2）サンプルソフトウェアの停止方法

メニューから「4」を入力して ENTER キーを押下することで、サンプルプログラムを終了してメニュー画面に戻ります。

3.19　温度検出回路（TEM）

3.19.1　サンプルソフトウェア仕様

本サンプルソフトウェアは以下の状態を評価するためのプログラムです。

- 温度検出の比較時間を設定する。
- 温度を計測し、その結果を表示する。

3.19.2　ハードウェア条件

本サンプルソフトウェアは OSC1 と OSC3A が発振可能な状態で動作します。

3.19.3　動作概要

（1）サンプルソフトウェア動作概要

- プログラム開始後、温度検出の比較時間を入力します。
- ENTERキーを押下することで温度検出を開始しその結果を表示します。

```
<<< TEM demonstration start >>>
Please input comparison time 0-255
16
Start a TEM. . .
Result 25
<<< TEM demonstration finish >>>
```

図 3.28　温度検出回路サンプルソフトウェア画面表示例

（2）サンプルソフトウェアの停止方法

上記“サンプルソフトウェア動作概要”に記載された全ての動作を終了すると、サンプルソフトウェアを終了してメニュー画面に戻ります。

3.20　論理緩急（TR）

3.20.1　サンプルソフトウェア仕様

本サンプルソフトウェアは以下の状態を評価するためのプログラムです。

- 論理緩急の調整値の設定を行う。
- 論理緩急を実行させ、その結果を REGMON 端子から出力する。

3.20.2　ハードウェア条件

本サンプルソフトウェアは OSC1 と OSC3A が発振可能な状態で動作します。

本サンプルソフトウェアはマイコンのポートを以下のように接続してご使用下さい。

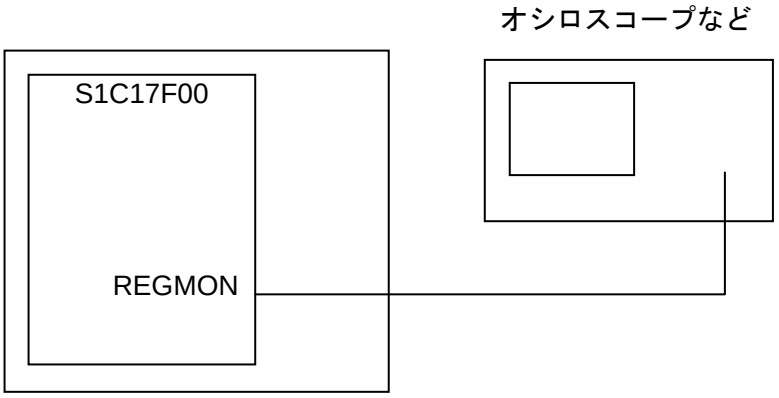

図 3.29　論理緩急サンプルプログラム実行時のハードウェア接続図

3.20.3　動作概要

（1）サンプルソフトウェア動作概要

- プログラム開始後、論理緩急の調整値を入力します。
- RTC1Hz を使い、論理緩急を実行させます。
- 論理緩急のモニターは REGNMON 端子にて行います。
- RTC を使っての論理緩急によるクロックの調整を１０回行います。

```
<<< TR demonstration start >>>
Please input TRIM 0-31
>16
Start a TR…
<<< TR demonstration finish >>>
```

図 3.30　論理緩急サンプルソフトウェア画面表示例

（2）サンプルソフトウェアの停止方法

上記“サンプルソフトウェア動作概要”に記載された全ての動作を終了すると、サンプルソフトウェアを終了してメニュー画面に戻ります。

3.21　Sleep/Halt モード切替

3.21.1　サンプルソフトウェア仕様

本サンプルソフトウェアは以下の動作を行います。

- halt 命令を実行し CPU を halt モードにする。
- 8 ビットタイマ割り込みを使用し CPU の halt モードを解除する。
- sleep 命令を実行し CPU を sleep モードにする。
- ポート割り込みを使用し CPU の sleep モードを解除する。

3.21.2　ハードウェア条件

本サンプルソフトウェアは OSC1 と OSC3A が発振可能な状態で動作します。

本サンプルソフトウェアは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。

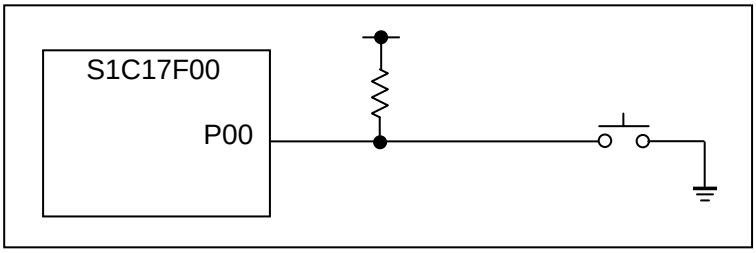

図 3.31　Sleep/Halt モード切替サンプルソフトウェア画面表示例

注 ポート設定は機種によって変更がある場合があります。各機種のソースを参照してご確認下さい。

3.21.3　動作概要

（1）サンプルソフトウェア動作概要

- 8 ビットタイマを開始し CPU を halt モードにします。
- 8 ビットタイマ割り込みが発生すると halt モードを解除し、SimI/O にメッセージを表示します。
- 8 ビットタイマ割り込みが 5 回発生すると 8 ビットタイマを停止し、CPU を sleep モードにします。
- P00 ポートが Low レベルになると sleep モードを解除します。

```
<<< Sleep/halt demonstration start >>>
go to halt mode
return from halt mode
…
go to sleep mode
return from sleep mode
<<< Sleep/halt demonstration finish >>>
```

図 3.32　Sleep/Halt モード切替サンプルソフトウェア画面表示例

（2）サンプルソフトウェアの停止方法

上記“サンプルソフトウェア動作概要”に記載された全ての動作を終了すると、サンプルソフトウェアを終了してメニュー画面に戻ります。

4. サンプルドライバ関数一覧

ここでは各周辺回路のサンプルドライバの一覧を記述します。

4.1 入出力ポート（P）

表 4.1 に本サンプルドライバの関数一覧を示します。関数の詳細はソースコード port.c を参照して下さい。

表 4.1 入出力ポート（P）サンプルドライバ関数一覧

関数名	機能名
PORT_init	Px ポート初期化
PORT_getInputData	Px ポートデータ入力
PORT_setOutputData	Px ポートデータ出力
PORT_controlInput	Px ポート入力許可/禁止設定
PORT_controlOutput	Px ポート出力許可/禁止設定
PORT_controlPullup	Px ポートプルアップ抵抗設定
PORT_controlSchmittTrigger	Px ポートシュミットトリガ設定
PORT_initInt	Px ポート割り込み初期化
PORT_controlInt	Px ポート割り込み許可/禁止設定
PORT_setIntEdge	Px ポート割り込みエッジ設定
PORT_resetIntFlag	Px ポート割り込み要因フラグリセット
PORT_checkIntFlag	Px ポート割り込み要因フラグチェック
PORT_setChatteringFilter	Px ポートチャタリング除去設定

本サンプルドライバは port.c と port.h、port_api.h に記述しています。
本サンプルドライバを使用するプログラムは port_api.h をインクルードして下さい。

4.2 クロックジェネレータ（CLG_OSC）

表 4.2 に本サンプルドライバの関数一覧を示します。関数の詳細はソースコード clg.c、osc.c を参照して下さい。

表 4.2 クロックジェネレータ（CLG_OSC）サンプルドライバ関数一覧

関数名	機能名
CLG_OSC_setClockSource	クロック源設定
CLG_OSC_setOSC3Bfrequency	OSC3B のクロック設定
CLG_OSC_setWaitCycle	発信安定待ち時間設定
CLG_OSC_controlOscillation	OSC 発振開始/停止設定
CLG_OSC_setFOUTDivision	FOUT クロック分周比設定
CLG_OSC_setFOUTClockSource	FOUT クロック源設定
CLG_OSC_controlFOUT	FOUT クロック出力許可/禁止設定
CLG_OSC_setRFCClock	RFC クロック設定
CLG_OSC_controlRFC	RFC クロック供給許可/禁止設定
CLG_OSC_setT16A2Clock	T16A2 クロック設定
CLG_OSC_controlT16A2	T16A2 クロック供給許可/禁止設定
CLG_OSC_setUARTClock	UART クロック設定
CLG_OSC_controlUART	UART クロック供給許可/禁止設定
CLG_OSC_setEPDClock	EPD タイミングクロック設定
CLG_OSC_controlEPD	EPD タイミングクロック供給許可/禁止設定
CLG_OSC_setEPDDClock	EPD Doubler クロック設定
CLG_OSC_controlDEPD	EPD Doubler クロック供給許可/禁止設定
CLG_OSC_setEPDBClock	EPD Booster クロック設定
CLG_OSC_controlEPDB	EPD Booster クロック供給許可/禁止設定
CLG_OSC_controlSND	SND クロック供給許可/禁止設定
CLG_OSC_setTEMClock	TEM クロック設定
CLG_OSC_controlTEM	TEM クロック供給許可/禁止設定
CLG_setPCLKEnable	PCLK 供給許可/禁止設定
CLG_setCCLKGearRatio	システムクロックギア比設定

本サンプルドライバは clg.c と osc.c、clg.h と osc.h、clg_api.h と osc_api.h に記述しています。
本サンプルドライバを使用するプログラムは clg_api.h と osc_api.h をインクルードして下さい。

4.3 8 ビットタイマ（T8）

表 4.3 に本サンプルドライバの関数一覧を示します。関数の詳細はソースコード t8.c を参照して下さい。

表 4.3 8 ビットタイマ（T8）サンプルドライバ関数一覧

関数名	機能名
T8_init	8 ビットタイマ初期化
T8_setInputClock	プリスケーラ出力クロック設定
T8_setReloadData	リロードデータ設定
T8_getCounterData	カウンタデータ取得
T8_setTimerMode	8 ビットタイマモード設定
T8_resetTimer	8 ビットタイマリセット
T8_setTimerRun	8 ビットタイマ開始/停止設定
T8_initInt	8 ビットタイマ割り込み初期化
T8_controlInt	8 ビットタイマ割り込み許可/禁止設定
T8_resetIntFlag	8 ビットタイマ割り込み要因フラグリセット
T8_checkIntFlag	8 ビットタイマ割り込み要因フラグチェック

本サンプルドライバは t8.c と t8.h、t8_api.h に記述しています。
本サンプルドライバを使用するプログラムは t8_api.h をインクルードして下さい。

4.4 PWM タイマ（T16A2）

表 4.4 に本サンプルドライバの関数一覧を示します。関数の詳細はソースコード t16a2.c を参照して下さい。

表 4.4 PWM タイマ（T16A2）サンプルドライバ関数一覧

関数名	機能名
T16A2_init	PWM タイマ（T16A2）初期化
T16A2_setTimerMode	PWM タイマ（T16A2）モード設定
T16A2_setComparatorCapture	コンパレータ/キャプチャ設定
T16A2_getCounterData	カウントデータ取得
T16A2_setCompareData	コンペアデータ設定
T16A2_getCaptureData	キャプチャデータ取得
T16A2_resetTimer	PWM タイマ（T16A2）リセット
T16A2_setTimerRun	PWM タイマ（T16A2）開始/停止設定
T16A2_initInt	PWM タイマ（T16A2）割り込み初期化
T16A2_controlInt	PWM タイマ（T16A2）割り込み許可/禁止設定
T16A2_resetIntFlag	PWM タイマ（T16A2）割り込み要因フラグリセット
T16A2_checkIntFlag	PWM タイマ（T16A2）割り込み要因フラグチェック

本サンプルドライバは t16a2.c と t16a2.h、t16a2_api.h に記述しています。
本サンプルドライバを使用するプログラムは t16a2_api.h をインクルードして下さい。

4.5 計時タイマ（CT）

表 4.5 に本サンプルドライバの関数一覧を示します。関数の詳細はソースコード ct.c を参照して下さい。

表 4.5 計時タイマ（CT）サンプルドライバ関数一覧

関数名	機能名
CT_resetTimer	計時タイマリセット
CT_setTimerRun	計時タイマ開始/停止設定
CT_getCounterData	カウンタデータ取得
CT_initInt	計時タイマ割り込み初期化
CT_controlInt	計時タイマ割り込み許可/禁止設定
CT_resetIntFlag	計時タイマ割り込み要因フラグリセット
CT_checkIntFlag	計時タイマ割り込み要因フラグチェック

本サンプルドライバは ct.c と ct.h、ct_api.h に記述しています。
本サンプルドライバを使用するプログラムは ct_api.h をインクルードして下さい。

4.6 ストップウォッチタイマ（SWT）

表 4.6 に本サンプルドライバの関数一覧を示します。関数の詳細はソースコード swt.c を参照して下さい。

表 4.6 ストップウォッチタイマ（SWT）サンプルドライバ関数一覧

関数名	機能名
SWT_resetTimer	ストップウォッチタイマリセット
SWT_setTimerRun	ストップウォッチタイマ開始/停止設定
SWT_getCounterDataBCD	BCD カウンタデータ取得
SWT_initInt	ストップウォッチタイマ割り込み初期化
SWT_controlInt	ストップウォッチタイマ割り込み許可/禁止設定
SWT_resetIntFlag	ストップウォッチタイマ割り込み要因フラグリセット
SWT_checkIntFlag	ストップウォッチタイマ割り込み要因フラグチェック

本サンプルドライバは swt.c と swt.h、swt_api.h に記述しています。
本サンプルドライバを使用するプログラムは swt_api.h をインクルードして下さい。

4.7 ウォッチドッグタイマ（WDT）

表 4.7 に本サンプルドライバの関数一覧を示します。関数の詳細はソースコード wdt.c を参照して下さい。

表 4.7 ウォッチドッグタイマ（WDT）サンプルドライバ関数一覧

関数名	機能名
WDT_resetTimer	ウォッチドッグタイマリセット
WDT_setTimerRun	ウォッチドッグタイマ開始/停止設定
WDT_setTimerMode	ウォッチドッグチタイマモード設定
WDT_checkNMI	ウォッチドッグタイマ NMI 発生チェック

本サンプルドライバは wdt.c と wdt.h、wdt_api.h に記述しています。
本サンプルドライバを使用するプログラムは wdt_api.h をインクルードして下さい。

4.8 UART

表 4.8 に本サンプルドライバの関数一覧を示します。関数の詳細はソースコード uart.c を参照して下さい。

表 4.8 UART サンプルドライバ関数一覧

関数名	機能名
UART_init	UART 初期化
UART_setTransmitData	送信データ設定
UART_getReceiveData	受信データ取得
UART_setComEnable	UART 送受信許可/禁止設定
UART_initInt	UART 割り込み初期化
UART_controlInt	UART 割り込み許可/禁止設定
UART_resetIntFlag	UART 割り込み要因フラグリセット
UART_checkReceiveFlag	UART 割り込み要因フラグチェック
UART_setIrDAmode	IrDA モード設定
UART_setBaudRate	ボーレート設定

本サンプルドライバは uart.c と uart.h、uart_api.h に記述しています。
本サンプルドライバを使用するプログラムは uart_api.h をインクルードして下さい。

4.9 SPI

表 4.9 に本サンプルドライバの関数一覧を示します。関数の詳細はソースコード spi.c を参照して下さい。

表 4.9 SPI サンプルドライバ関数一覧

関数名	機能名
SPI_init	SPI 初期化
SPI_setTransmitData	送信データ設定
SPI_getReceiveData	受信データ取得
SPI_setComEnable	SPI 送受信許可/禁止設定
SPI_initInt	SPI 割り込み初期化
SPI_controlInt	SPI 割り込み許可/禁止設定
SPI_checkIntFlag	SPI 割り込み要因フラグチェック
SPI_checkBusyFlag	送受信 BUSY フラグチェック

本サンプルドライバは spi.c と spi.h、spi_api.h に記述しています。
本サンプルドライバを使用するプログラムは spi_api.h をインクルードして下さい。

4.10 I2C マスタ（I2CM）

表 4.10 に本サンプルドライバの関数一覧を示します。関数の詳細はソースコード i2cm.c を参照して下さい。

表 4.10 I2C マスタ（I2CM）サンプルドライバ関数一覧

関数名	機能名
I2CM_init	I2C マスタ初期化
I2CM_setComEnable	I2C マスタ送受信許可/禁止設定
I2CM_genCondition	スタート/ストップコンディション生成
I2CM_checkTransmitReg	送信データレジスタチェック
I2CM_setTransmitData	送信データ設定
I2CM_checkTransmitBusy	送信動作状態チェック
I2CM_getSlaveResponse	スレーブ応答取得
I2CM_setReceiveStart	データ受信開始設定
I2CM_checkReceiveBusy	受信動作状態チェック
I2CM_getReceiveData	受信データ取得
I2CM_checkReceiveReg	受信データレジスタチェック
I2CM_initInt	I2C マスタ割り込み初期化
I2CM_controlInt	I2C マスタ割り込み許可/禁止設定
I2CM_transmitSlaveAddress	スレーブアドレス送信データ作成

本サンプルドライバは i2cm.c と i2cm.h、i2cm_api.h に記述しています。

本サンプルドライバを使用するプログラムは i2cm_api.h をインクルードして下さい。

4.11 I2C スレーブ（I2CS）

表 4.11 に本サンプルドライバの関数一覧を示します。関数の詳細はソースコード i2cs.c を参照して下さい。

表 4.11 I2C スレーブ（I2CS）サンプルドライバ関数一覧

関数名	機能名
I2CS_reset	I2C スレーブソフトウェアリセット
I2CS_setAddress	I2C スレーブアドレス設定
I2CS_setClockStretch	クロックストレッチ機能設定
I2CS_setAsyncDetection	非同期アドレス検出機能設定
I2CS_setNoiseRemove	ノイズ除去機能選択
I2CS_setBusFreeReq	バス解放要求許可/禁止設定
I2CS_setReceiveResponse	データ受信応答設定
I2CS_init	I2C スレーブ初期化
I2CS_setEnable	I2C スレーブモジュール動作許可/禁止設定
I2CS_setComEnable	データ送受信許可/禁止設定
I2CS_setTransmitData	送信データ設定
I2CS_getReceiveData	受信データ取得
I2CS_initInt	I2C スレーブ割り込み初期化
I2CS_controlInt	I2C スレーブ割り込み許可/禁止設定
I2CS_resetIntFlag	I2C スレーブバスステータス割り込み要因フラグリセット
I2CS_checkBusStatusIntFlag	I2C スレーブバスステータス割り込み要因フラグチェック
I2CS_checkIntFlag	I2C スレーブ割り込み要因フラグチェック
I2CS_checkAccessStatus	I2C スレーブアクセスステータスチェック

本サンプルドライバは i2cs.c と i2cs.h、i2cs_api.h に記述しています。

本サンプルドライバを使用するプログラムは i2cs_api.h をインクルードして下さい。

4.12 EPD コントローラ/ドライバ（EPD）

表 4.12 に EPD サンプルドライバの関数一覧を示します。関数の詳細はソースコード epd.c を参照して下さい。

表 4.12 EPD コントローラ/ドライバ（EPD）サンプルドライバ関数一覧

関数名	機能名
EPD_init	EPD 初期化
EPD_initPower	EPD 電源初期化
EPD_setDbsrt	DBSRT 設定
EPD_setDoubler	DOUBLER 設定
EPD_setVecon	VECON 設定
EPD_setVesel	VESEL 設定
EPD_setHvldve	HVLDVE 設定
EPD_setVeon	VEON 設定
EPD_setVhcon	VHCON 設定
EPD_setVhsel	VHSEL 設定
EPD_setHvldvh	HVLDVH 設定
EPD_setVhon	VHON 設定
EPD_setBstpld	BSTPLD 設定
EPD_setBooster	BOOSTER 設定
EPD_getVesel	VESEL 取得
EPD_setMode	EPD コントローラ波形/ダイレクトモード設定
EPD_setDisplayMode	EPD モード表示設定
EPD_getDisplayStatus	EPD アップデート表示ステータス取得
EPD_setDisplayTrigger	EPD 表示開始
EPD_initInt	EPD 割り込み初期化
EPD_controlInt	EPD 割り込み許可/禁止設定
EPD_resetintFlag	EPD 割り込み要因フラグリセット
EPD_checkIntFlag	EPD 割り込み要因フラグチェック
EPD_setSeghz	Segment/BackPlane 出力 Hiz 設定
EPD_setTphz	TopPlane 出力 Hiz 設定
EPD_setTp	TopPlane 出力レベル設定
EPD_setBp	backPlane の表示データ/出力レベル設定
EPD_set_TopBackPlaneData	EPD Top/Back Plane データ設定
EPD_setSegmentData	EPD セグメントデータ設定
EPD_setWaveformTiming	EPD 波形タイミング設定
EPD_setWaveformEOW	EPD 波形終了位置設定
EPD_setWaveformHIZ	セグメント/バックプレーン端子のハイインピーダンス設定
EPD_setWaveformTP	トッププレーン端子の出力波形設定
EPD_setWaveformBB	セグメント/バックプレーン端子の出力波形を設定（表示更新時に表示が黒から黒になる場合）
EPD_setWaveformBW	セグメント/バックプレーン端子の出力波形を設定（表示更新時に表示が黒から白になる場合）
EPD_setWaveformWB	セグメント/バックプレーン端子の出力波形を設定（表示更新時に表示が白から黒になる場合）
EPD_setWaveformWW	セグメント/バックプレーン端子の出力波形を設定（表示更新時に表示が白から白になる場合）
EPD_setWaveformINTV	タイミングセットの期間（TCLK クロック数）設定

本サンプルドライバは epd.c と epd.h、epd_api.h に記述しています。

本サンプルドライバを使用するプログラムは epd_api.h をインクルードして下さい。

表 4.13 に拡張 EPD サンプルドライバの関数一覧を示します。関数の詳細はソースコード ext_epd.c を参照して下さい。

表 4.13 拡張 EPD コントローラ/ドライバ（ExtEPD）サンプルドライバ関数一覧

関数名	機能名
ExtEPD_init	拡張 EPD 初期化
ExtEPD_initPower	拡張 EPD 電源初期化
ExtEPD_setWaveformTiming	拡張 EPD 波形タイミング設定
ExtEPD_setFlashMemoryDisable	拡張 EPD フラッシュメモリ設定
ExtEPD_setSegmentData	拡張 EPD セグメントデータ設定
ExtEPD_setPowerOn	拡張 EPD 電源立ち上げ
ExtEPD_setPowerOff	拡張 EPD 電源立ち下げ
ExtEPD_setBp	拡張 EPD backPlane の表示データ/出力レベル設定
ExtEPD_setDisplayMode	拡張 EPD モード表示設定
ExtEPD_setPowerControl	拡張 EPD 電源設定
ExtEPD_setSlaveMode	拡張 EPD モード設定
ExtEPD_setSoftwareReset	拡張 EPD ソフトウェアリセット
ExtEPD_getWaveformTiming	拡張 EPD 波形タイミング設定情報取得
ExtEPD_getFlashMemoryDisable	拡張 EPD フラッシュメモリ設定情報取得
ExtEPD_getSegmentData	拡張 EPD セグメントデータ設定情報取得
ExtEPD_getBp	拡張 EPD backPlane の表示データ/出力レベル設定情報取得
ExtEPD_getDisplayMode	拡張 EPD モード表示設定情報取得
ExtEPD_getPowerControl	拡張 EPD 電源設定情報取得
ExtEPD_getSlaveMode	拡張 EPD モード設定情報取得
ExtEPD_getState	拡張 EPD ステータス取得
ExtEPD_spiStart	SPI の開始
ExtEPD_spiStop	SPI の停止
spiIntHandler	SPI 割り込みハンドラ

本サンプルドライバは ext_epd.c と ext_epd.h、epd_api.h に記述しています。

本サンプルドライバを使用するプログラムは epd_api.h をインクルードして下さい。

4.13 電源電圧検出回路（SVD）

表 4.14 に本サンプルドライバの関数一覧を示します。関数の詳細はソースコード svd.c を参照して下さい。

表 4.14 電源電圧検出回路（SVD）サンプルドライバ関数一覧

関数名	機能名
SVD_setCompareVoltage	SVD 比較電圧設定
SVD_controlDetection	SVD 検出開始/停止設定
SVD_getDetectionResult	SVD 検出結果取得

本サンプルドライバは svd.c と svd.h、svd_api.h に記述しています。

本サンプルドライバを使用するプログラムは svd_api.h をインクルードして下さい。

4.14 R/F 変換器（RFC）

表 4.15 に本サンプルドライバの関数一覧を示します。関数の詳細はソースコード rfc.c を参照して下さい。

表 4.15 R/F 変換器（RFC）サンプルドライバ関数一覧

関数名	機能名
RFC_setRFC	R/F 変換器許可/禁止設定
RFC_setRFCChannel	変換チャネル設定
RFC_setRFCMode	発振モード設定
RFC_setReferenceOscillation	基準発振開始/停止設定
RFC_setSensorAOscillation	センサーA 発振開始/停止設定
RFC_setSensorBOscillation	センサーB 発振開始/停止設定
RFC_getReferenceOscillation	基準発振状態取得
RFC_getSensorAOscillation	センサーA 発振状態取得
RFC_getSensorBOscillation	センサーB 発振状態取得
RFC_setEventMode	イベントカウンタモード許可/禁止設定
RFC_setContinuous	連続発振許可/禁止設定
RFC_setMeasurementCounter	計測カウンタ値設定
RFC_setTimeBaseCounter	タイムベースカウンタ値設定
RFC_getMeasurementCounter	計測カウンタ値取得
RFC_getTimeBaseCounter	タイムベースカウンタ値取得
RFC_initInt	RFC 割り込み初期化
RFC_controlInt	RFC 割り込み許可/禁止設定
RFC_resetIntFlag	RFC 割り込み要因フラグリセット
RFC_checkIntFlag	RFC 割り込み要因フラグチェック

本サンプルドライバは rfc.c と rfc.h、rfc_api.h に記述しています。

本サンプルドライバを使用するプログラムは rfc_api.h をインクルードして下さい。

4.15 リアルタイムクロック（RTC）

表 4.16 に本サンプルドライバの関数一覧を示します。関数の詳細はソースコード rtc.c を参照して下さい。

表 4.16 リアルタイムクロック（RTC）サンプルドライバ関数一覧

関数名	機能名
RTC_init	リアルタイムクロック初期化
RTC_setTimerRun	リアルタイムクロック開始/停止設定
RTC_setTimerMode	24H/12H モード切替設定
RTC_setAMPM	AM/PM 切替設定
RTC_setTime	リアルタイムクロック時刻設定
RTC_getTime	リアルタイムクロック時刻取得
RTC_checkStatus	ステータスチェック
RTC_initInt	リアルタイムクロック割り込み設定
RTC_controlInt	リアルタイムクロック割込み許可/禁止設定
RTC_resetIntFlag	リアルタイムクロック割り込み要因フラグリセット
RTC_checkIntFlag	リアルタイムクロック割り込み要因フラグチェック

本サンプルドライバは rtc.c と rtc.h、rtc_api.h に記述しています。

本サンプルドライバを使用するプログラムは rtc_api.h をインクルードして下さい。

4.16 サウンドジェネレータ（SND）

表 4.17 に本サンプルドライバの関数一覧を示します。関数の詳細はソースコード snd.c を参照して下さい。

表 4.17 サウンドジェネレータ（SND）サンプルドライバ関数一覧

関数名	機能名
SND_init	SND 初期化
SND_setBuzzerTime	ブザーエンベロープ、ワンショット時間設定
SND_setBuzzerMode	ブザーモード設定
SND_setTrigger	ブザー出力開始
SND_setBuzzerfrequency	ブザー信号発振周波数設定
SND_setBuzzerDutyRatio	ブザー信号の音量設定

本サンプルドライバは snd.c と snd.h、snd_api.h に記述しています。

本サンプルドライバを使用するプログラムは snd_api.h をインクルードして下さい。

4.17 温度検出回路（TEM）

表 4.18 に本サンプルドライバの関数一覧を示します。関数の詳細はソースコード tem.c を参照して下さい。

表 4.18 温度検出回路（TEM）サンプルドライバ関数一覧

関数名	機能名
TEM_init	TEM 初期化
TEM_setConversionRun	温度変換開始/停止設定
TEM_setTEMEnable	温度センサー変換許可/禁止設定
TEM_setConversionTime	センサー出力と比較電圧の比較時間設定
TEM_getConversionResult	温度変換結果取得
TEM_checkStatus	TEM のステータスチェック
TEM_initInt	温度検出回路割り込み設定
TEM_controlInt	温度検出回路割込み許可/禁止設定
TEM_resetIntFlag	温度検出回路割り込み要因フラグリセット
TEM_checkIntFlag	温度検出回路割り込み要因フラグチェック

本サンプルドライバは tem.c と tem.h、tem_api.h に記述しています。
本サンプルドライバを使用するプログラムは tem_api.h をインクルードして下さい。

4.18 論理緩急（TR）

表 4.19 に本サンプルドライバの関数一覧を示します。関数の詳細はソースコード tr.c を参照して下さい。

表 4.19 論理緩急（TR）サンプルドライバ関数一覧

関数名	機能名
TR_setRclockfrequency	TR（REGMON の出力）周波数設定
TR_setEnable	TR 許可/禁止設定
TR_setTrigger	TR 開始設定
TR_setRegulationValue	TR 1 回の補正量設定

本サンプルドライバは tr.c と tr.h、tr_api.h に記述しています。
本サンプルドライバを使用するプログラムは tr_api.h をインクルードして下さい。

4.19 MISC

表 4.20 に本サンプルドライバの関数一覧を示します。関数の詳細はソースコード misc.c を参照して下さい。

表 4.20 MISC サンプルドライバ関数一覧

関数名	機能名
MISC_setDebugModeControl	デバッグモード設定
MISC_controlWriteProtect	MISC レジスター書き込み保護制御
MISC_setIRAMSize	IRAM サイズ設定
MISC_setTTBR	ベクターテーブルアドレス設定
MISC_getPSR	PSR 取得
MISC_getIRAMSize	IRAM サイズ取得

本サンプルドライバは misc.c と misc.h、misc_api.h に記述しています。

本サンプルドライバを使用するプログラムは misc_api.h をインクルードして下さい。

4.20 マルチプレクサ（MUX）

表 4.21 に本サンプルドライバの関数一覧を示します。関数の詳細はソースコード mux.c を参照して下さい。

表 4.21 マルチプレクサ（MUX）サンプルドライバ関数一覧

関数名	機能名
MUX_init	MUX 初期化
MUX_setSPIport	SPI ポート設定
MUX_setUARTport	UART ポート設定
MUX_setRFCport	RFC ポート設定
MUX_setI2CMport	I2C マスタポート設定
MUX_setI2CSport	I2C スレーブポート設定
MUX_setOSCport	OSC ポート設定
MUX_setSNDport	SND ポート設定
MUX_setDBGport	デバッグポート設定
MUX_setT16A2port	PWM タイマ（T16A2）ポート設定

本サンプルドライバは mux.c と mux.h、mux_api.h に記述しています。

本サンプルドライバを使用するプログラムは mux_api.h をインクルードして下さい。

4.21 電源制御回路（VD1）

表 4.22 に本サンプルドライバの関数一覧を示します。関数の詳細はソースコード vd1.c を参照して下さい。

表 4.22 電源制御回路（VD1）サンプルドライバ関数一覧

関数名	機能名
VD1_setMode	重不可モードの設定

本サンプルドライバは vd1.c と vd1.h、vd1_api.h に記述しています。

本サンプルドライバを使用するプログラムは vd1_api.h をインクルードして下さい。

Appendix A 乗除算器

ここでは乗除算器の使い方について説明します。

A.1　乗除算器を使った乗算と除算

乗除算器を使った乗算と除算を行うために、GNU17 にはコプロセッサ用ライブラリが用意されています。

コプロセッサ用ライブラリの使い方につきましては S5U1C17001C Manual を参照して下さい。

A.2　乗除算器を使った積和演算

乗除算器を使って積和演算を行うためのプログラムを以下に示します。

本プログラムは“0x1204×0x1080＋0x28A00”の積和演算を行います。

```
asm ( "ld.cw %r0, 0x0" );       /* clear */
asm ( "ld.cw %r0, 0x2" );       /* setup mode */
asm ( "xld %r0, 0x0002" );      /* set 0x28A00 */
asm ( "xld %r1, 0x8A00" );

asm ( "ld.cf %r0, %r1" );

asm ( "ld.cw %r0, 0x7" );       /* setup mode */
asm ( "xld %r0, 0x1204" );      /* 0x1204 */
asm ( "xld %r1, 0x1080" );      /* 0x1080 */
asm ( "ld.ca %r0, %r1" );

asm ( "ld.cw %r0, 0x13" );      /* read */
asm ( "ld.ca %r1, %r0" );
asm ( "ld.cw %r0, 0x03" );      /* read */
asm ( "ld.ca %r2, %r0" );

/* result = 0x12BCC00 */
```

改訂履歴表

付－1

コードNo.	ページ	改訂内容（旧内容を含む） および改訂理由
	全ページ	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本部　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：412154300
2011 年 1 月 作成