EPSON
EXCEED YOUR VISION

S1C17 Family Application Note

S1C17500 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

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S1C17500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。

S1C17500 シリーズ向けサンプルソフトウェアは S1C17500 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。

S1C17500 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。

機種情報(S1C175xxサンプルプログラムReadMe)、各テクニカルマニュアル、S5U1C17001C Manual、S1C17700 周辺回路サンプルソフトウェアと合わせてご覧下さい。

1.1. 動作環境

S1C17500 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。

- S1C17500 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、GNU17v2.0.0 で動作確認を行っています。

1.2. 対象機種

本マニュアルで対象となる機種は以下の通りです。

表 1.1　S1C17500 シリーズ対象機種一覧

対象機種名	非対象機種
S1C17554	S1C17501
S1C17564	S1C17511
-	S1C17512

2. サンプルソフトウェア説明

本章では S1C17500 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。

S1C17500 シリーズ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 S1C17500 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。

```
[s1c175xx]
   │
   ├GNU17 プロジェクトファイルなど
   ├…
   ├…
   │
   │
   ├── [mcu]
   │       ├── s1c17554_peripheral.h
   │       …
   │       ├── s1c175xx.h
   │       └── boot.h          boot.c
   │
   ├── [apl]
   │       ├── main.h          main.c
   │       ├── clg_main.c
   │       ├── port_main.c
   │       …
   │       ├── t16_ main.c
   │       ├── t16f_ main.c
   │       └── adc_main.c
   │
   ├── [driver]
   │       ├── clg.h           clg.c
   │       ├── port.h          port.c
   │       …
   │       ├── t16.h           t16.c
   │       ├── t16f.h          t16f.c
   │       └── adc.h           adc.c
   │
   └── [common]
           ├── clg_api.h
           ├── port_api.h
           …
           ├── t16_api.h
           ├── t16f_api.h
           └── adc_api.h
```

図 2.1　S1C17500 シリーズサンプルソフトウェアディレクトリ構成図

（1）s1c175xx 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（s1c17554_peripheral.h など）
- 機種共通のヘッダファイル（s1c175xx.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 S1C17500 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 S1C17500 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“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 S1C175xx 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“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“External Toos”ボタンを押下してデバッガを起動し、デバッガの“Continue”ボタンを押下して下さい。

プログラムが S1C17500 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 S5U1C1001C Manual の“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Simulated I/O(以下SimI/O)にメニュー画面が表示されます。

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。

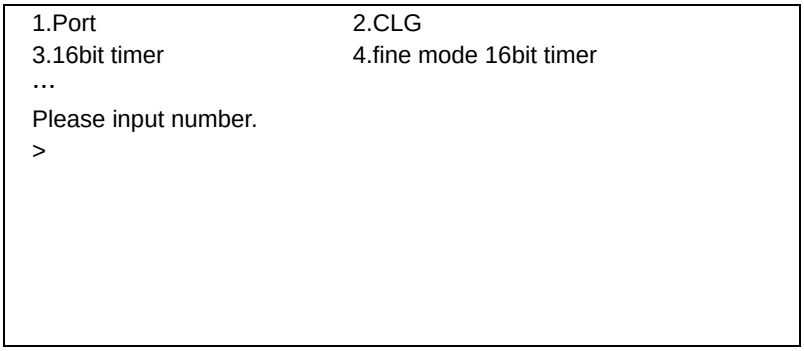

図 2.2 メニュー画面表示例

2.4 特定モジュールのビルド方法

S1C175xx 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。

S1C17554 サンプルソフトウェアの場合は s1c17554_peripheral.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_PORT
//#undef PE_T16
//#undef PE_T16F
//#undef PE_T16A
//#undef PE_CT
//#undef PE_SWT
//#undef PE_WDT
#undef   PE_UART
#undef   PE_UART_OSC3
#undef   PE_SPI
#undef   PE_SPI_MASTER
#undef   PE_SPI_SLAVE
#undef   PE_I2CM
#undef   PE_I2CS
#undef   PE_ADC
#undef   PE_REMC
#undef   PE_REMC_TX
#undef   PE_REMC_RX
#undef   PE_CURRENT_MEASURE
#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 16 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_T16”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 S1C17500 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　クロックジェネレータ（CLG）

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発振回路を使用して以下の動作を行います。

- IOSCの発振と停止を行う。(IOSC有りの機種のみ)
- OSC1 の発振と停止を行う。
- OSC3 の発振と停止を行う。
- システムクロックを IOSC から OSC3 へ切り替える。(IOSC 有りの機種のみ)
- システムクロックを OSC3 から OSC1 へ切り替える。
- システムクロックを OSC1 から IOSC（IOSC 無しの機種は OSC3）へ切り替える。

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 OSC3 が発振可能な状態で動作します。

3.1.3　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 IOSC(IOSC 無しの機種は OSC3)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、OSC3 の発振を開始してシステムクロックを IOSC から OSC3 に切り替え、IOSC を停止します。(IOSC 有りの機種のみ)
- 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 OSC3 から OSC1 に切り替え、OSC3 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、IOSC（IOSC 無しの機種は OSC3）を発振開始しシステムクロックを OSC1 から IOSC（IOSC 無しの機種は OSC3）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 >>>
IOSC *** 1 ***
IOSC *** 2 ***
…
IOSC *** 9 ***
*** Change from IOSC to OSC3 ***
OSC3 *** 1 ***
OSC3 *** 2 ***
…
OSC3 *** 9 ***
<<< CLGdemonstration finish >>>
```

図 3.1　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。

3.2　ファインモード 16 ビットタイマ（T16F）

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ファインモード 16 ビットタイマを使用して以下の動作を行います。

- ファインモード 16 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。

3.2.2　ハードウェア条件

本サンプルソフトウェアは OSC1 と OSC3 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

- ファインモード 16 ビットタイマ割り込みを開始し、CPU を halt モードにします。
- ファインモード 16 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、ファインモード 16 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。
- ファインモード 16 ビットタイマ割り込みが 10 回発生したところでファインモード 16 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。

```
<<< T16F timer demonstration start >>>
*** T16F interrupt 1 time, count data at this time : 32 ***
*** T16F interrupt 2 time, count data at this time : 32 ***
*** T16F interrupt 3 time, count data at this time : 32 ***
*** T16F interrupt 4 time, count data at this time : 32 ***
…
*** T16F interrupt 10 time, count data at this time : 32 ***
<<< T16F timer demonstration finish >>>
```

図 3.2　ファインモード 16 ビットタイマ（T16F）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　16 ビット PWM タイマ（T16A）

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 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 タイマコンペア A マッチ割り込みを発生させ、タイマのカウンタ値を取得する。
- 16 ビット PWM タイマコンペア B マッチ割り込みを発生させ、タイマのカウンタ値を取得する。
- TOUT0 端子に波形を出力する。
- 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。

注 端子名称 TOUT0 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。

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 OSC3 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

- 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 16 ビット PWM タイマを開始し、CPU を halt モードにします。
- 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すると CPU の halt モードが解除され、16 ビット PWM 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。
- コンペア B マッチ割り込みが 5 回発生したところで 16 ビット PWM タイマを停止し、割り込みの種類とカウンタ値を SimI/O に表示します。

```
<<< 16 bit PWM timer demonstration start >>>
*** 16 bit PWM compare A interrupt :633 ***
*** 16 bit PWM compare B interrupt :126 ***
*** 16 bit PWM compare A interrupt :633 ***
…
*** 16 bit PWM Interrupt B interrupt: 126 ***
<<< 16 bit PWM timer demonstration finish >>>
```

図 3.3　アドバンスドタイマ（T16A）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　ユニバーサルシリアルインタフェース（USI）

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ユニバーサルシリアルインタフェースを使用して以下の動作を行います。

- ユニバーサルシリアルインタフェースの UART を使用してデータを送信する。
- ユニバーサルシリアルインタフェースの UART を使用してデータを受信する。
- ユニバーサルシリアルインタフェースの SPI マスタを使用して SPI スレーブにデータを送信する。
- ユニバーサルシリアルインタフェースの SPI マスタを使用して SPI スレーブからデータを受信する。
- ユニバーサルシリアルインタフェースの I2C マスタを使用して I2C スレーブへデータを送信する。
- ユニバーサルシリアルインタフェースの I2C マスタを使用して I2C スレーブからデータを受信する。
- ユニバーサルシリアルインタフェースの I2C スレーブを使用して I2C マスタからデータを受信する。
- ユニバーサルシリアルインタフェースの I2C スレーブを使用して I2C マスタへデータを送信する。

3.4.2　ハードウェア条件

本サンプルソフトウェアは OSC1 と OSC3 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。

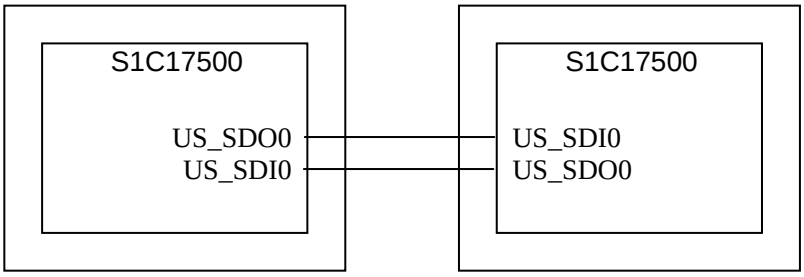

図 3.4　UART 選択時サンプルソフトウェアハードウェア接続図

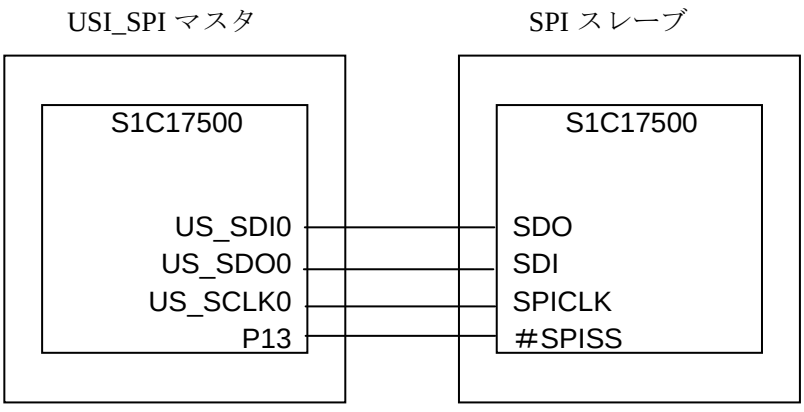

図 3.5　SPI マスタ選択時サンプルソフトウェアハードウェア接続図

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。

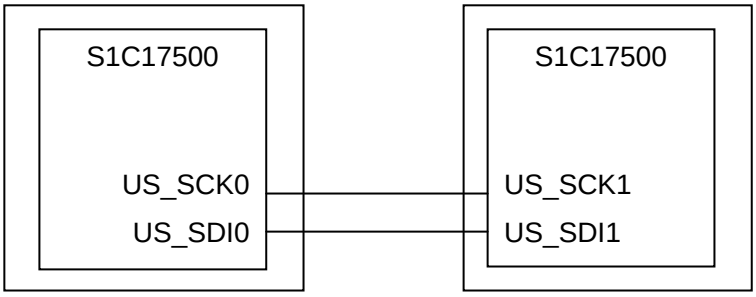

図 3.6　I2C マスタ、スレーブ選択時サンプルソフトウェアハードウェア接続図

3.4.3　動作概要

（1）サンプルソフトウェア動作概要

- メニューから UART のメニュー番号を入力して ENTER キーを押下すると、ユニバーサルシリアルインタフェースの UART を使用した送受信を行います。
- メニューから SPI master のメニュー番号を入力して ENTER キーを押下すると、ユニバーサルシリアルインタフェースの SPI マスタを使用した送受信を行います。
- メニューから I2C master のメニュー番号を入力して ENTER キーを押下すると、ユニバーサルシリアルインタフェースの I2C マスタを使用した送受信を行います。
- メニューから I2C slave のメニュー番号を入力して ENTER キーを押下すると、ユニバーサルシリアルインタフェースの I2C スレーブを使用した送受信を行います。
- メニューから exit のメニュー番号を入力して 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。

```
<<< USI demonstration start >>>
1.UART                  2.SPI master
3.I2C master            4.I2C slave
5.exit

<<< USI demonstration finish >>>
```

図 3.7　ユニバーサルシリアルインタフェース（USI）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

本サンプルソフトウェアのメニューから exit のメニュー番号を入力して ENTER キーを押下すると、サンプルソフトウェアを終了してメニュー画面に戻ります。

3.5　電流測定

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以下の状態を評価するためのプログラムです。

内蔵レギュレータがある機種は、内蔵レギュレータを自動制御モードにして行います。

- CPU を sleep モードにする。
- OSC1 のみを発振させ、PCLK をオフにした状態で CPU を halt モードにする。
- OSC1 のみを発振させた状態で CPU を halt モードにする。
- OSC1/OSC3 を発振させた状態で CPU を halt モードにする。
- OSC1/IOSC を発振させた状態で CPU を halt モードにする。（IOSC 有りの機種のみ）

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 OSC3 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

- 「1」を入力して ENTER キーを押下することで CPU を sleep モードにします。
- 「2」を入力してENTERキーを押下することでOSC1 のみを発振させ、PCLKをオフにした状態でCPUをhaltモードにします。
- 「3」を入力して ENTER キーを押下することで OSC1 のみを発振させた状態で CPU を halt モードにします。
- 「4」を入力して ENTER キーを押下することで OSC1 と OSC3 のみを発振させた状態で CPU を halt モードにします。
- 「5」を入力して ENTER キーを押下することで OSC1 と IOSC のみを発振させた状態で CPU を halt モードにします。

```
<<< Current measurement demonstration start >>>
1. Sleep                    2.Halt with OSC1, PCLK off
3.Halt with OSC1            4.Halt with OSC1/3
5.Halt with OSC1/IOSC

Please input number.
>1
sleeping
```

図 3.8　電流測定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

本サンプルソフトウェアを終了する場合は、マイコンの外部リセット端子をアサートしてソフトウェアをリセットして下さい。

3.6　その他の機能

本サンプルソフトウェアには、以下の機能が含まれます。夫々の機能説明は、S1C17000 周辺回路サンプルソフトウエアのドキュメントを参照ください。

- 入出力ポート（P）
- 16 ビットタイマ（T16）
- 計時タイマ（CT）
- ストップウォッチタイマ（SWT）
- ウォッチドッグタイマ（WDT）
- OSC3 を使用した UART
- IOSC を使用した UART（IOSC 有りの機種のみ）
- SPI マスタ
- SPI スレーブ
- I2C マスタ（I2CM）
- I2C スレーブ（I2CS）
- リモートコントローラ送信（REMC)
- リモートコントローラ受信（REMC）
- A/D 変換器（ADC10）
- Sleep/Halt モード切替

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_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）

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 を参照して下さい。

表 4.2 クロックジェネレータ（CLG）サンプルドライバ関数一覧

関数名	機能名
CLG_setClockSource	クロック源設定
CLG_setWaitCycle	発振安定待ち時間設定
CLG_controlOscillation	OSC 発振開始/停止設定
CLG_setFOUTDivision	FOUTx クロック設定
CLG_controlFOUT	FOUTx クロック出力許可/禁止設定
CLG_setIOSCfrequency	IOSC 発振周波数設定
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 clg.h、clg_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 をインクルードして下さい。

4.3 16 ビットタイマ（T16）

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 t16.c を参照して下さい。

表 4.3 16 ビットタイマ（T16）サンプルドライバ関数一覧

関数名	機能名
T16_init	16 ビットタイマ初期化
T16_setInputClock	プリスケーラ出力クロック設定
T16_setReloadData	リロードデータ設定
T16_getCounterData	カウンタデータ取得
T16_setTimerMode	16 ビットタイマモード設定
T16_resetTimer	16 ビットタイマリセット
T16_setTimerRun	16 ビットタイマ開始/停止設定
T16_initInt	16 ビットタイマ割り込み初期化
T16_controlInt	16 ビットタイマ割り込み許可/禁止設定
T16_resetIntFlag	16 ビットタイマ割り込み要因フラグリセット
T16_checkIntFlag	16 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 t16.c と t16.h、t16_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 t16_api.h をインクルードして下さい。

4.4 ファインモード 16 ビットタイマ（T16F）

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 t16f.c を参照して下さい。

表 4.4 ファインモード 16 ビットタイマ（T16F）サンプルドライバ関数一覧

関数名	機能名
T16F_init	ファインモード 16 ビットタイマ初期化
T16F_setInputClock	プリスケーラ出力クロック設定
T16F_setReloadData	リロードデータ設定
T16F_getCounterData	カウンタデータ取得
T16F_setTimerMode	ファインモード 16 ビットタイマモード設定
T16F_resetTimer	ファインモード 16 ビットタイマリセット
T16F_setTimerRun	ファインモード 16 ビットタイマ開始/停止設定
T16F_initInt	ファインモード 16 ビットタイマ割り込み初期化
T16F_controlInt	ファインモード 16 ビットタイマ割り込み許可/禁止設定
T16F_resetIntFlag	ファインモード 16 ビットタイマ割り込み要因フラグリセット
T16F_checkIntFlag	ファインモード 16 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 t16f.c と t16f.h、t16f_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 t16f_api.h をインクルードして下さい。

4.5 16 ビット PWM タイマ（T16A）

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 t16a.c を参照して下さい。

表 4.5 16 ビット PWM タイマ（T16A）サンプルドライバ関数一覧

関数名	機能名
T16A_setInputClock	入力クロック設定
T16A_controlInputClock	入力クロック供給許可/禁止設定
T16A_init	16 ビット PWM タイマ初期化
T16A_setTimerMode	16 ビット PWM タイマモード設定
T16A_setComparatorCapture	コンパレータ/キャプチャ設定
T16A_getCounterData	カウントデータ取得
T16A_setCompareData	コンペアデータ設定
T16A_getCaptureData	キャプチャデータ取得
T16A_resetTimer	16 ビット PWM タイマリセット
T16A_setTimerRun	16 ビット PWM タイマ開始/停止設定
T16A_initInt	16 ビット PWM タイマ割り込み初期化
T16A_controlInt	16 ビット PWM タイマ割り込み許可/禁止設定
T16A_resetIntFlag	16 ビット PWM タイマ割り込み要因フラグリセット
T16A_checkIntFlag	16 ビット PWM 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 t16a.c と t16a.h、t16a_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 t16a_api.h をインクルードして下さい。

4.6 計時タイマ（CT）

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 ct.c を参照して下さい。

表 4.6 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。

4.7 ストップウォッチタイマ（SWT）

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 swt.c を参照して下さい。

表 4.7 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。

4.8 ウォッチドッグタイマ（WDT）

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 wdt.c を参照して下さい。

表 4.8 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。

4.9 UART

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 uart.c を参照して下さい。

表 4.9 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
UART_setCountClock	UART カウントクロック設定
UART_controlCountClock	UART カウントクロック許可/禁止設定

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。

4.10 SPI

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 spi.c を参照して下さい。

表 4.10 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。

4.11 I2C マスタ（I2CM）

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 i2cm.c を参照して下さい。

表 4.11 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。

4.12 I2C スレーブ（I2CS）

表 4.12 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 4.12 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。

4.13 ユニバーサルシリアルインタフェース（USI）

4.13.1 各モード共通

表 4.13 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 usi.c を参照して下さい。

表 4.13 ユニバーサルシリアルインタフェース（USI）共通サンプルドライバ関数一覧

関数名	機能名
USI_init	USI 初期化
USI_setTransmitData	送信データ設定
USI_getReceiveData	受信データ取得
USI_initInt	USI 割り込み初期化

本サンプルドライバは usi.c と usi.h、usi_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 usi_api.h をインクルードして下さい。

4.13.2 UART モード

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 usi_uart.c を参照して下さい。

表 4.14 ユニバーサルシリアルインタフェース（USI）UART モードサンプルドライバ関数一覧

関数名	機能名
USI_UART_init	UART 初期化
USI_UART_controlInt	UART 割り込み許可/禁止設定
USI_UART_resetIntFlag	UART 割り込み要因フラグリセット
USI_UART_checkReceiveFlag	UART 割り込み要因フラグチェック

本サンプルドライバは usi_uart.c と usi_uart.h、usi_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 usi_api.h をインクルードして下さい。

4.13.3 SPI モード

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 usi_spi.c を参照して下さい。

表 4.15 ユニバーサルシリアルインタフェース（USI）SPI モードサンプルドライバ関数一覧

関数名	機能名
USI_SPI_init	SPI 初期化
USI_SPI_controlInt	SPI 割り込み許可/禁止設定
USI_SPI_resetIntFlag	SPI 割り込み要因フラグリセット
USI_SPI_checkIntFlag	SPI 割り込み要因フラグチェック
USI_SPI_checkBusyFlag	送受信 BUSY フラグチェック
USI_SPI_setReceiveMaskData	受信マスクデータ設定

本サンプルドライバは usi_spi.c と usi_spi.h、usi_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 usi_api.h をインクルードして下さい。

4.13.4 I2C モード

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 usi_i2c.c を参照して下さい。

表 4.16 ユニバーサルシリアルインタフェース（USI）I2C モードサンプルドライバ関数一覧

関数名	機能名
USI_I2C_setTrigger	I2C トリガ設定
USI_I2C_checkStatus	I2C 状態チェック
USI_I2C_checkBusy	I2C 動作状態チェック
USI_I2C_controlInt	I2C 割り込み許可/禁止設定
USI_I2C_resetIntFlag	I2C 割り込み要因フラグリセット
USI_I2C_checkIntFlag	I2C 割り込み要因フラグチェック

本サンプルドライバは usi_i2c.c と usi_i2c.h、usi_i2c_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 usi_api.h をインクルードして下さい。

4.14 リモートコントローラ（REMC）

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 remc.c を参照して下さい。

表 4.17 リモートコントローラ（REMC）サンプルドライバ関数一覧

関数名	機能名
REMC_init	REMC 初期化
REMC_setEnable	REMC 送受信開始/停止設定
REMC_setTransmitData	送信データ設定
REMC_setReceiveLength	受信データ長設定
REMC_getReceiveData	受信データ取得
REMC_calcReceiveDataPulse	受信データパルス長算出
REMC_initInt	REMC 割り込み初期化
REMC_controlInt	REMC 割り込み許可/禁止設定
REMC_resetIntFlag	REMC 割り込み要因フラグリセット
REMC_checkIntFlag	REMC 割り込み要因フラグチェック

本サンプルドライバは remc.c と remc.h、remc_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 remc_api.h をインクルードして下さい。

4.15 A/D 変換器（ADC10）

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 adc.c を参照して下さい。

表 4.18 A/D 変換器（ADC10）サンプルドライバ関数一覧

関数名	機能名
ADC_init	ADC 初期化
ADC_setChannel	A/D 変換チャネル設定
ADC_setStoreMode	変換結果格納方法設定
ADC_setConversionMode	A/D 変換モード設定
ADC_setConversionTrigger	A/D 変換開始トリガ方法設定
ADC_setSamplingClock	サンプリング時間設定
ADC_setDividedFrequency	A/D 変換クロック分周設定
ADC_setEnable	A/D 変換開始/停止設定
ADC_getResult	A/D 変換結果取得
ADC_getConversionChannel	A/D 変換中チャネル番号取得
ADC_checkBusyStatus	A/D 変換中チェック
ADC_controlTrigger	ソフトウェアトリガ制御
ADC_adjustmentComparator	コンパレータ調整
ADC_initInt	ADC 割り込み初期化
ADC_controlInt	ADC 割り込み許可/禁止設定
ADC_resetIntFlag	ADC 割り込み要因フラグリセット
ADC_checkIntFlag	ADC 割り込み要因フラグチェック

本サンプルドライバは adc.c と adc.h、adc_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 adc_api.h をインクルードして下さい。

4.16 MISC

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 misc.c を参照して下さい。

表 4.19 MISC サンプルドライバ関数一覧

関数名	機能名
MISC_setDebugModeControl	デバッグモード時周辺回路動作設定
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_getIRAMSize	IRAM サイズ取得
MISC_setTTBR	ベクターテーブルアドレス設定
MISC_getPSR	PSR 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。

4.17 マルチプレクサ（MUX）

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 mux.c を参照して下さい。

表 4.20 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_setREMCport	REMC ポート設定
MUX_setADCport	ADC ポート設定
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_setDBGport	デバッグポート設定
MUX_setCLGport	CLG ポート設定
MUX_setT16Aport	16 ビット PWM タイマポート設定
MUX_setUSIport	USI 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。

4.18 内蔵レギュレータ（PWG）

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 pwg.c を参照して下さい。

表 4.21 内蔵レギュレータ（PWG）サンプルドライバ関数一覧

関数名	機能名
PWG_setOperationMode	動作モード設定

本サンプルドライバは pwg.c と pwg.h、pwg_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 pwg_api.h をインクルードして下さい。

4.19 Flash 制御（FLASHC）

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 flashc.c を参照して下さい。

表 4.22 Flash 制御（FLASHC）サンプルドライバ関数一覧

関数名	機能名
FLASHC_setReadWaitCycle	Flash メモリリードウェイト数設定

本サンプルドライバは flashc.c と flashc.h、flashc_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 flashc_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
411916500	全ページ	新規制定

セイコーエプソン株式会社
半導体事業部　IC 営業部

<IC 国内営業グループ>

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：411916500
2010 年 5 月 作成