EPSON
EXCEED YOUR VISION

S1C17 Family Application Note

S1C17600 シリーズ周辺回路 サンプルソフトウェア２

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 S1C17604/622/624 マイコンで新たに追加された周辺回路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。S1C17600 シリーズ共通の周辺回路については、以下のマニュアルを参照してください。

・S1C17600 シリーズ周辺回路サンプルソフトウェア

S1C17600 シリーズ周辺回路サンプルソフトウェア２は S1C17604/622/624 マイコンで新たに追加された周辺回路の使用例を示すことを目的としています。

S1C17600 シリーズ周辺回路サンプルソフトウェア２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。

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

S1C17600 シリーズ周辺回路サンプルソフトウェア２を動作させるにあたり、以下の機材をご用意下さい。

- S1C17604/622/624 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S1C17600シリーズ周辺回路サンプルソフトウェア２のファイル構成と実行方法を記載します。

S1C17600 シリーズ周辺回路サンプルソフトウェア２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 収録されているサンプルソフトウェア

以下に収録されているサンプルソフトウェアの一覧を示します。

表 2.1 収録されているサンプルソフトウェア一覧

周辺回路	サンプルソフトウェア
入出力ポート(P)	○
クロックジェネレータ(CLG_OSC)	◎
16 ビットタイマ(T16)	○
8 ビットタイマ(T8F)	○
PWM タイマ(T16E)	○
8 ビット OSC1 タイマ(T8OSC1)	○
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	○
SPI マスタ	○
SPI スレーブ	○
I2C マスタ(I2CM)	○
I2C スレーブ(I2CS)	○
LCD ドライバ(LCD8)	○
電源電圧検出回路(SVD)	○
R/F 変換器(RFC)	○
A/D 変換器(ADC10SA)	○
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)	○
Sleep/Halt	○
PWM タイマ(T16A2)	◎
リアルタイムクロック(RTC)	◎
電流測定	◎

◎:新規、変更 ○:シリーズ共通

2.2 収録されているサンプルドライバ

以下に収録されているサンプルドライバの一覧を示します。

表 2.2 収録されているサンプルドライバー覧

周辺回路	サンプルドライバ
入出力ポート(P)	○
クロックジェネレータ(CLG_OSC)	◎
16 ビットタイマ(T16)	○
8 ビットタイマ(T8F)	○
PWM タイマ(T16E)	○
8 ビット OSC1 タイマ(T8OSC1)	○
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)	○
UART	○
SPI	○
I2C マスタ(I2CM)	○
I2C スレーブ(I2CS)	○
LCD ドライバ(LCD8)	◎
電源電圧検出回路(SVD)	○
R/F 変換器(RFC)	○
A/D 変換器(ADC10SA)	○
リモートコントローラ(REMC)	○
プリスケーラ(PSC)	○
MISC	○
マルチプレクサ(MUX)	◎
PWM タイマ(T16A2)	◎
リアルタイムクロック(RTC)	◎
電源制御回路(VD1)	◎

◎:新規、変更 ○:シリーズ共通

2.3 ディレクトリ構成及びファイル構成

以下に S1C17600 シリーズ周辺回路サンプルソフトウェア２のディレクトリ構成を示します。

```
[s1c176xx]
   │
   ├GNU17 プロジェクトファイルなど
   ├…
   ├…
   │
   │
   ├── [mcu]
   │       ├── s1c17604_peripheral.h
   │       …
   │       ├── s1c176xx.h
   │       └── boot.h          boot.c
   │
   ├── [apl]
   │       ├── main.h          main.c
   │       ├── clg_osc_main.c
   │       ├── port_main.c
   │       …
   │       ├── t16_ main.c
   │       ├── lcd_ main.c
   │       └── svd_main.c
   │
   ├── [driver]
   │       ├── osc.h           osc.c
   │       ├── clg.h           clg.c
   │       ├── psc.h           psc.c
   │       ├── port.h          port.c
   │       …
   │       ├── lcd.h           lcd.c
   │       └── svd.h           svd.c
   │
   └── [common]
           ├── osc_api.h
           ├── clg_api.h
           ├── psc_api.h
           ├── port_api.h
           …
           ├── lcd_api.h
           └── svd_api.h
```

図 2.1　S1C17600 シリーズ周辺回路サンプルソフトウェア２ディレクトリ構成図

（1）s1c176xx 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ガ配置されています。

（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（s1c17604_peripheral.h など）
- 機種共通のヘッダファイル（s1c176xx.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。

2.4 実行方法

以下の手順で S1C17600 シリーズ周辺回路サンプルソフトウェア 2 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 S1C17600 シリーズ周辺回路サンプルソフトウェア 2 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 S1C176xx 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。

ICDmini 使用方法の詳細については S5U1C17001H User Manual を参照して下さい。

（4）デバッガによるプログラムのロードと実行

GNU17 の“デバッグの構成”ボタンを押下してデバッガを起動し、デバッグビューの“再開”ボタンを押下して下さい。

プログラムが S1C176xx 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いては S5U1C17001C Manual の“10 デバッガ”を参照して下さい。

2.5 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。

Simulated I/O については S5U1C17001C Manual の“10.4.11 [シミュレーテッド I/O]ビュー”を参照して下さい。

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。

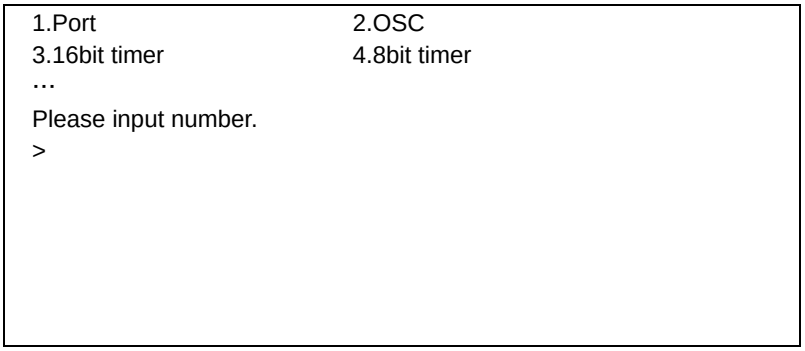

図 2.2 メニュー画面表示例

2.6 特定モジュールのビルド方法

S1C176xx 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。

例として S1C17624 サンプルソフトウェアの場合は s1c17624_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_CLG_OSC
//#undef PE_T16
//#undef PE_T8F
//#undef PE_T16E
//#undef PE_T8OSC1
//#undef PE_CT
//#undef PE_SWT
//#undef PE_WDT
#undef  PE_UART
#undef  PE_UART_OSC3
#undef  PE_UART_IOSC
#undef  PE_SPI
#undef  PE_SPI_MASTER
#undef  PE_SPI_SLAVE
#undef  PE_I2CM
#undef  PE_I2CS
#undef  PE_LCD
#undef  PE_SVD
#undef  PE_RFC
#undef  PE_ADC
#undef  PE_REMC
#undef  PE_REMC_TX
#undef  PE_REMC_RX
#undef  PE_SLEEP_HALT
//#undef PE_T16A2
//#undef PE_RTC
//#undef PE_VD1
#undef  PE_CURRENT_MEASURE
```

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 UART（OSC3）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、UART（OSC3）サンプルソフトウェアをビルドする際は、“#undef PE_UART_OSC3”と“#undef PE_T8F”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 S1C17600 シリーズ周辺回路サンプルソフトウェア２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_OSC）

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クロックジェネレータ（CLG_OSC）を使用して以下の動作を行います。

- IOSC の発振と停止を行います。
- OSC1 の発振と停止を行います。
- OSC3 の発振と停止を行います。
- システムクロックを IOSC から OSC3 へ切り替えます。
- システムクロックを OSC3 から OSC1 へ切り替えます。
- システムクロックを OSC1 から IOSC へ切り替えます。

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 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、S1C176xx 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。

3.1.3 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 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。
- 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 と OSC3 を発振開始しシステムクロックを OSC1 から IOSC に切り替え、OSC1 と OSC3 を停止します。
- 次に一定間隔で SimI/O に「1」、「2」、「3」…、「9」と表示します。

```
<<< CLG(OSC) demonstration start >>>
IOSC *** 1 ***
IOSC *** 2 ***
…
IOSC *** 9 ***
*** Change from IOSC to OSC3 ***
OSC3 *** 1 ***
OSC3 *** 2 ***
…
OSC3 *** 9 ***
<<< CLG(OSC) demonstration finish >>>
```

図 3.1 クロックジェネレータ（CLG_OSC）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 PWM タイマ（T16A2）

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 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します。
- 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します。
- TOUTA5 端子に PWM 波形をノーマル/ハーフクロックモードで出力します。
- 割り込み待機時は CPU を halt モードにして消費電力を低減します。

3.2.2 ハードウェア条件

本サンプルソフトウェアは OSC3 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、S1C176xx 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。

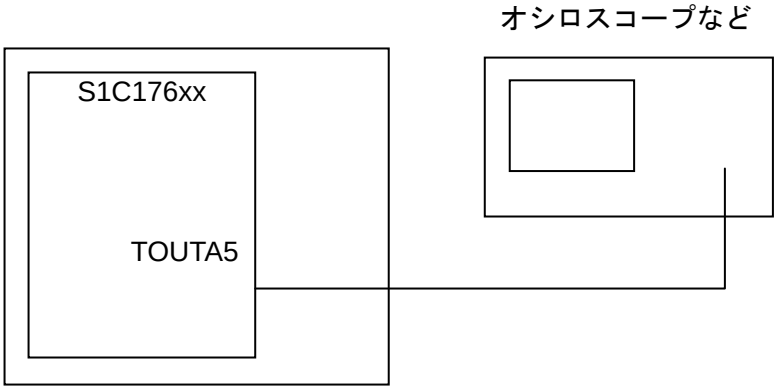

図 3.2 PWM タイマ（T16A2）サンプルソフトウェアハードウェア接続図

3.2.3 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(T16A2) demonstration finish >>>
```

図 3.3　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。

3.3 リアルタイムクロック（RTC）

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します。
- リアルタイムクロックの時刻を設定します。
- リアルタイムクロックの割り込み発生回数を表示します。

3.3.2 ハードウェア条件

本サンプルソフトウェアは OSC3 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、S1C176xx 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。

3.3.3 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、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
10/03/01(Mon) 10:00:00

1.get RTC                          2.set RTC
3.indicate the count of interrupt  4.exit
Please input number.
>2
> Input BCD format.
> Year (0x00 - 0x99) :0x**
> Month (0x01 - 0x12) :0x**
> Day (0x01 - 0x31) :0x**
> Hour (0x00 - 0x23) :0x**
> Minute (0x00 - 0x59) :0x**
> Second (0x00 - 0x59) :0x**
> Week (0x0 - 0x6) :0x*

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 3.4　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。

3.4 電流計測

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電流を計測するために以下の状態で CPU を実行するためのプログラムです。

- CPU を sleep 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/OSC3/IOSC を発振させた状態で CPU を halt モードにする

3.4.2 ハードウェア条件

本サンプルソフトウェアは OSC3 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、S1C176xx 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。

3.4.3 実行方法

本サンプルソフトウェアは以下の手順でビルドすることで使用可能となります。

（1）プログラムの変更

機種別の定義ヘッダを変更します。

例として S1C17624 の場合は s1c17624_peripheral.h を変更します。

（2）変更箇所

ファイル下部にある以下の箇所を変更します。

```
//#undef PE_T16A2
//#undef PE_RTC
//#undef PE_VD1
//#undef PE_CURRENT_MEASURE


/*****************************************************************************/
/* Current measurement mode selecting                                        */
/*****************************************************************************/
// Comment out "#undef         PE_CURRENT_MEASURE", when using current measure program.
#ifdef PE_CURRENT_MEASURE
#define CURRENT_MEASURE_SLEEP                                   (0x00)
#define CURRENT_MEASURE_HALT_OSC1                               (0x01)
#define CURRENT_MEASURE_HALT_OSC1_OSC3                          (0x02)
#define CURRENT_MEASURE_HALT_OSC1_OSC3_IOSC   (0x03)
// Remove the comment on the selected mesurement mode.
//#define CURRENT_MEASURE_MODE CURRENT_MEASURE_SLEEP
//#define CURRENT_MEASURE_MODE CURRENT_MEASURE_HALT_OSC1
#define CURRENT_MEASURE_MODE CURRENT_MEASURE_HALT_OSC1_OSC3
//#define CURRENT_MEASURE_MODE CURRENT_MEASURE_HALT_OSC1_OSC3_IOSC
#endif
```

図 3.5　電流測定サンプルソフトウェアのための定義変更例

例えば OSC1/OSC3 を発振させた状態で CPU を halt モードにする場合、

“#undef PE_CURRENT_MEASURE”の定義を無効にし、

“#define CURRENT_MEASURE_MODE CURRENT_MEASURE_HALT_OSC1_OSC3”の定義を有効にしてビルドを行います。

（3）デバッガによるプログラムのロード

デバッガを起動し、サンプルソフトウェアを S1C176xx へロードします。

（4）プログラムの実行

ICDmini からの電流の影響を除外するため、ICDmini を開発ボードから外し S1C176xx をリセットしてください。

（5）動作概要

- リセット後、電流計測サンプルソフトウェアが実行されます。

別の状態で電流計測を行いたい場合は、（2）～（4）の手順を行ってください。

（6）サンプルソフトウェアの停止方法

停止方法はありません。開発ボードへの電源供給を止めてください。

4　サンプルドライバ関数一覧

ここではサンプルソフトウェアの各サンプルドライバの一覧を記述します。

4.1　クロックジェネレータ（CLG_OSC）

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 clg.c、osc.c を参照して下さい。

表 4.1　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_setWaitCycle	発振安定待ち時間設定
CLG_OSC_controlOscillation	OSC 発振開始/停止設定
CLG_OSC_setNoiseFilter	ノイズフィルタ有効/無効設定
CLG_OSC_setLCDClock	LCD クロック設定
CLG_OSC_controlLCDClock	LCD クロック供給許可/禁止設定
CLG_OSC_setFOUTHDivision	FOUTH クロック分周比設定
CLG_OSC_controlFOUT	FOUT クロック出力許可/禁止設定
CLG_OSC_setT8OSC1Division	T8OSC1 クロック分周比設定
CLG_OSC_controlT8OSC1	T8OSC1 クロック供給許可/禁止設定
CLG_OSC_setSVDClock	SVD クロック設定
CLG_OSC_controlSVD	SVD クロック供給許可/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_controlRTC	RTC 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。

4.2　LCD ドライバ（LCD8）

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 lcd.c を参照して下さい。

表 4.2　LCD ドライバ（LCD8）サンプルドライバ関数一覧

関数名	機能名
LCD8_initPower	LCD 電源初期化
LCD8_init	LCD 初期化
LCD8_setSEGAssignment	SEG 端子メモリ割り当て設定
LCD8_setCOMAssignment	COM 端子メモリ割り当て設定
LCD8_setDisplayArea	LCD 表示領域設定
LCD8_setDisplayReverse	LCD 表示白黒反転設定
LCD8_controlDisplay	LCD 表示制御
LCD8_setContrast	LCD コントラスト設定
LCD8_display1Seg	1 セグメント表示
LCD8_initInt	LCD 割り込み初期化
LCD8_controlInt	LCD 割り込み許可/禁止設定
LCD8_resetIntFlag	LCD 割り込み要因フラグリセット
LCD8_checkIntFlag	LCD 割り込み要因フラグチェック

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

4.3　電源制御回路（VD1）

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 vd1.c を参照して下さい。

表 4.3　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	重負荷モードの設定

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。

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　リアルタイムクロック（RTC）

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 rtc.c を参照して下さい。

表 4.5　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_setTime	リアルタイムクロック時刻設定
RTC_setAdj	30 秒補正
RTC_getTime	リアルタイムクロック時刻取得
RTC_checkBusy	ビジーチェック
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。

4.6　マルチプレクサ（MUX）

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 mux.c を参照して下さい。

表 4.6　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_setOSCport	OSC ポート設定
MUX_setT16Eport	PWM タイマ（T16E）ポート設定
MUX_setLCDport	LCD ポート設定
MUX_setT8OSC1port	8 ビット OSC1 タイマポート設定
MUX_setDBGport	デバッグポート設定
MUX_setT16port	16 ビットタイマポート設定
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。

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
412017500	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：412017500
2010 年 10 月 作成