EPSON
EXCEED YOUR VISION

S1C17 Family Application Note

S1C17700 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S1C17700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。

S1C17700 シリーズ向けサンプルソフトウェアは S1C17700 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。

S1C17700 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、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

S1C17700 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。

- S1C17700 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、GNU17v1.5.0 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 S1C17700 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。

S1C17700 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 S1C17700 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。

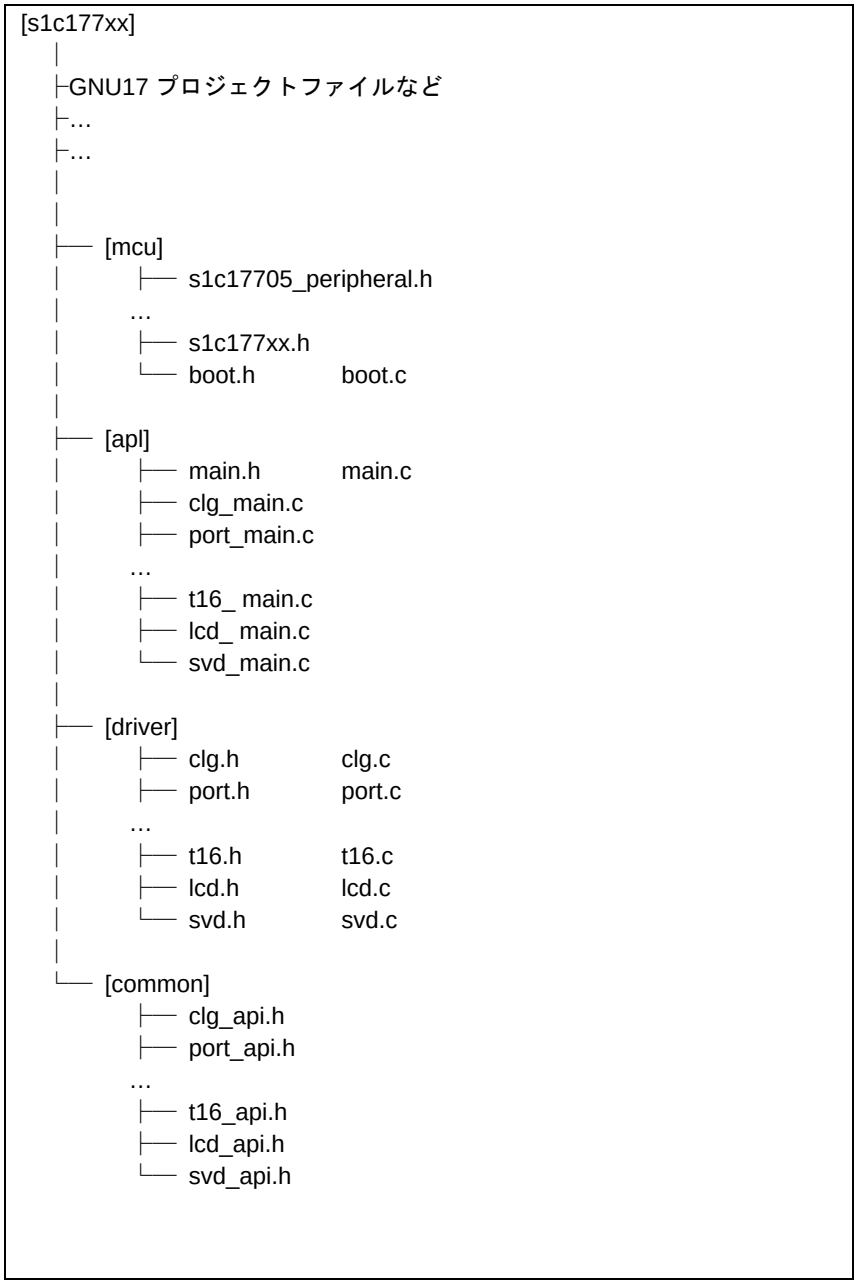

図 2.1 S1C17700 シリーズサンプルソフトウェアディレクトリ構成図

（1）s1c177xx 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（s1c17705_peripheral.h など）
- 機種共通のヘッダファイル（s1c177xx.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 S1C17700 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 S1C17700 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 S1C177xx 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“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 S1C17700 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 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。

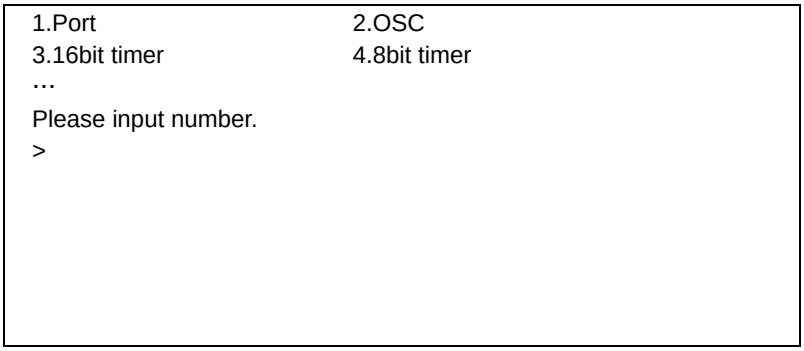

図 2.2 メニュー画面表示例

2.4 特定モジュールのビルド方法

S1C177xx 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。

S1C17705 サンプルソフトウェアの場合は s1c17705_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
//#undef PE_T16
//#undef PE_T16A
//#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_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 S1C17700 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	P02 P15 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 OSC3 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。

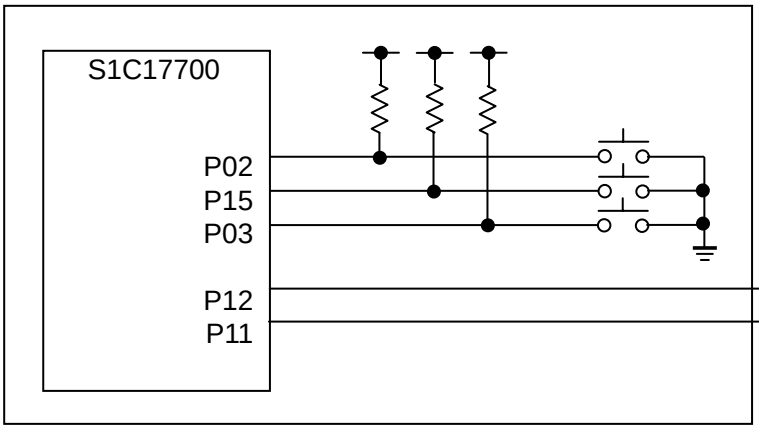

図 3.1　入出力ポート（P）サンプルソフトウェアハードウェア接続図

3.1.3　動作概要

（1）サンプルソフトウェア動作概要

- P02 ポートの入力信号を Low レベルにすると SimI/O に「P02 Interrupt」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。)
- P15 ポートの入力信号を Low レベルにすると SimI/O に「P15 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。)

```
<<< Port demonstration start >>>
*** P02 Interrupt ***
*** P15 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。

- IOSC の発振と停止を行う。
- OSC1 の発振と停止を行う。
- OSC3 の発振と停止を行う。
- EXOSC3 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。
- 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 EXOSC3 へ切り替える。
- システムクロックを EXOSC3 から IOSC 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 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

- 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、外部 OSC3 クロックを設定して OSC3 を発振開始しシステムクロックを OSC1 から EXOSC3 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」と表示した後、IOSC を発振開始しシステムクロックを EXOSC3 から IOSC に切り替え、OSC3 を停止し内部 OSC3 クロックを設定します。
- 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 3.3　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　16 ビットタイマ（T16）

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 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

- 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。
- 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、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。
- 16 ビットタイマ割り込みが 10 回発生したところで 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。

```
<<< T16 timer demonstration start >>>
*** T16 interrupt 1 time, count data at this time : 32 ***
*** T16 interrupt 2 time, count data at this time : 32 ***
*** T16 interrupt 3 time, count data at this time : 32 ***
*** T16 interrupt 4 time, count data at this time : 32 ***
…
*** T16 interrupt 10 time, count data at this time : 32 ***
<<< T16 timer demonstration finish >>>
```

図 3.4　16 ビットタイマ（T16）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　アドバンスドタイマ（T16A）

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アドバンスドタイマを使用して以下の動作を行います。

- アドバンスドタイマコンペア 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。
- アドバンスドタイマコンペア 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 TOUTN0 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。

3.4.2　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。

3.4.3　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アドバンスドタイマを開始し、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、アドバンスド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アドバンスドタイマを停止し、割り込みの種類とカウンタ値を SimI/O に表示します。

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

図 3.5　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。

3.5　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 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

- 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 3.6　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 OSC3 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 3.7　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 OSC3 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 16 ビットタイマを開始します。
- 16 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。
- 16 ビットタイマ割り込みが 10 回発生すると 16 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 >>>
*** T16 timer : reset watchdog timer ***
*** T16 timer : reset watchdog timer ***
*** T16 timer : reset watchdog timer ***
…
*** T16 timer : reset watchdog timer ***
*** stop T16 timer ***
*** NMI occurred ***
<<< Watchdog timer demonstration finish >>>
```

図 3.8　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　OSC3 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 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。

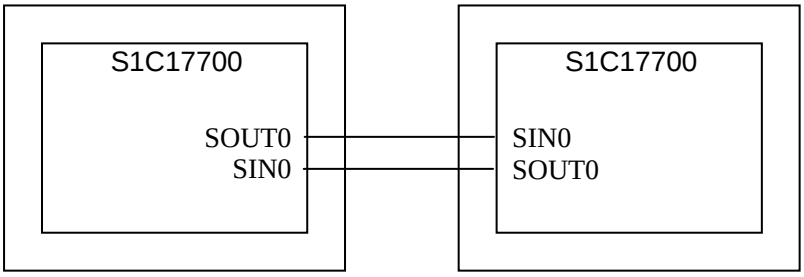

図 3.9　OSC3 を使用した UART サンプルソフトウェアハードウェア接続図

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 OSC3 demonstration start >>>
waiting connection.
connected.
*** sent data ***
*** received data ***
ABCDEFG…
<<< UART OSC3 demonstration finish >>>
```

図 3.10　OSC3 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　IOSC 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。

- IOSC クロックと OSC1 のカウンタ値を比較し、IOSC の周波数を算出する。
- IOSC 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 OSC3 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

- IOSC を使った 16 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、IOSC の発振周波数を計算します。
- 計算した IOSC 発振周波数を元に 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 IOSC demonstration start >>>
waiting connection.
connected.
*** sent data ***
*** received data ***
ABCDEFG…
<<< UART IOSC demonstration finish >>>
```

図 3.11　IOSC 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　IOSC 発振周波数計算方法

IOSC 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。

- T16A のコンペアデータを 8 カウントに設定し、IOSC を使った T16 と OSC1 を使った T16A をスタートさせます。
- T16A のコンペアマッチ割り込みが発生したときに T16 を停止します。
- T16 停止後のカウンタ値を読み出し IOSC の周波数を求めます。
- n×4096÷(div×bps)の商が UART_BR+1、剰余が UART_FMD になります。(n=IOSC 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 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 SPI スレーブサンプルソフトウェアが動作している S1C17700 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。

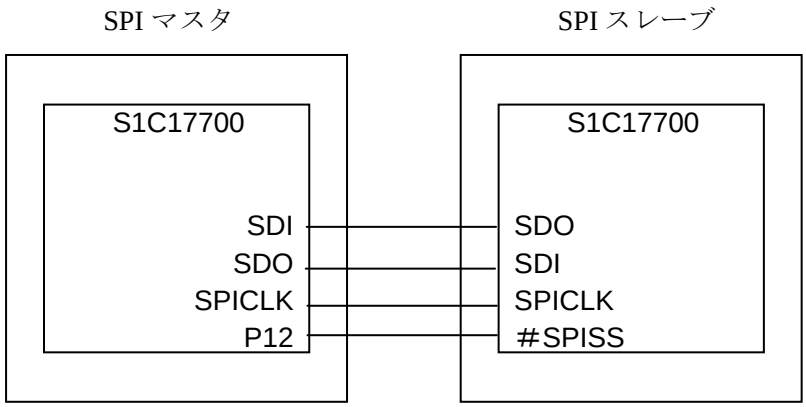

図 3.12　SPI マスタ、スレーブサンプルソフトウェアハードウェア接続図

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 3.13　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 S1C17700 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 3.14　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 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 I2C スレーブサンプルソフトウェアが動作している S1C17700 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。

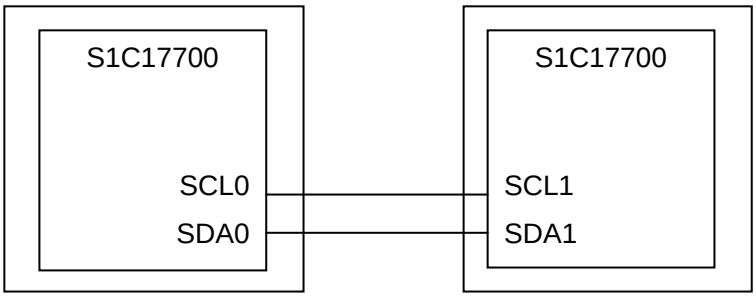

図 3.15　I2C マスタ（I2CM）、スレーブ（I2CS）サンプルソフトウェアハードウェア接続図

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 3.16　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 S1C17700 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 3.17　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　LCD ドライバ（LCD）

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 LCD ドライバを使用して以下の動作を行います。

- 全点灯と全消灯を行う。
- 指定されたセグメントの点灯、消灯を行う。
- LCD 表示の白黒反転を行う。
- LCD の 4 階調表示を行う。

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 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。

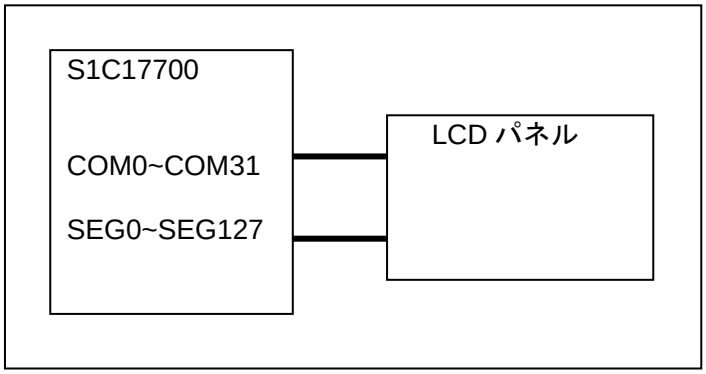

図 3.18　LCD ドライバ（LCD）サンプルソフトウェアハードウェア接続図

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 LCD を全点灯します。
- メニューから「2」を入力して ENTER キーを押下すると LCD を全消灯します。
- メニューから「3」を入力して ENTER キーを押下すると SEG/COM 番号入力待ちになり、「(SEG 番号)、(COM 番号)」を入力して ENTER キーを押下すると指定されたセグメントを点灯します。
- メニューから「4」を入力して ENTER キーを押下すると SEG/COM 番号入力待ちになり、「(SEG 番号)、(COM 番号)」を入力して ENTER キーを押下すると指定されたセグメントを消灯します。
- メニューから「5」を入力して ENTER キーを押下すると市松模様を表示し、一定時間間隔で LCD の白黒反転を 2 回実施します。
- メニューから「6」を入力して ENTER キーを押下すると LCD 割り込みを利用して LCD の 4 階調表示を行います。(1,5 行目は黒、2,6 行目は 75%グレー、3,7 行目は 50%グレー、4,8 行目は 25%グレーにします。)
- メニューから「7」を入力して ENTER キーを押下すると漢字の表示を行います。
- メニューから「8」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。

```
<<< LCD driver demonstration start >>>
1.All on                2.All off
3.Turn dot on           4.Turn dot off
5.Reverse               6.Grayscale
7.Kanji                 8.exit

<<< LCD driver demonstration finish >>>
```

図 3.19　LCD ドライバ（LCD）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「8」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。

3.15　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 OSC3 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 1.8V～3.2V です。
- 電源電圧値 1.8V 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 3.20　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。

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 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。

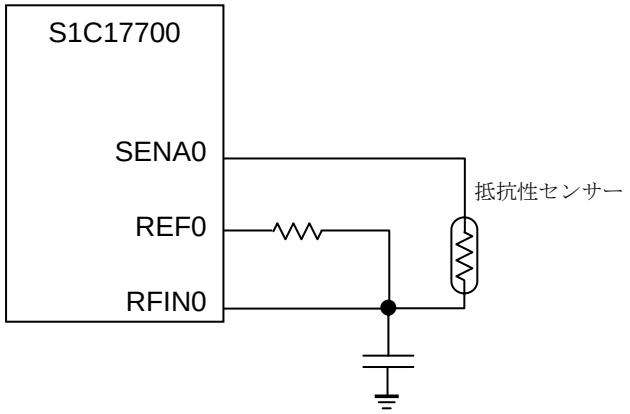

図 3.21　R/F 変換器（RFC）サンプルソフトウェアハードウェア接続図

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 >>>
Reference
measurement counter : 0000
time base counter : 0000
Sensor A
measurement counter : 0000
time base counter : 0000
<<< RFC demonstration finish >>>
```

図 3.22　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　A/D 変換器（ADC10）

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 A/D 変換器を使用して以下の動作を行います。

- A/D 変換終了割り込みを検出し A/D 変換結果を取得する。
- 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。

3.17.2　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。

AIN0 端子に、アナログ入力可能電圧範囲で任意の電圧を与え使用ください。

3.17.3　動作概要

（1）サンプルソフトウェア動作概要

- A/D 変換器を設定します。
- A/D 変換終了割り込みが発生すると A/D 変換結果を読み出します。
- A/D 変換終了割り込みが 10 回発生すると SimI/O に結果を表示します。

```
<<< AD converter demonstration start >>>
AD conversion result : 0000
AD conversion result : 0001
AD conversion result : 0003
…
AD conversion result : 0003
<<< AD converter demonstration finish >>>
```

図 3.23　A/D 変換器（ADC10）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。

3.18　リモートコントローラ送信（REMC）

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、リモートコントローラ（以下、REMC）周辺回路を使用して以下の動作をします。

- REMC 周辺回路を使用しデータを送信する。

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 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リモートコントローラ受信（REMC）サンプルソフトウェアが動作している S1C17700 を通信相手とし、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。

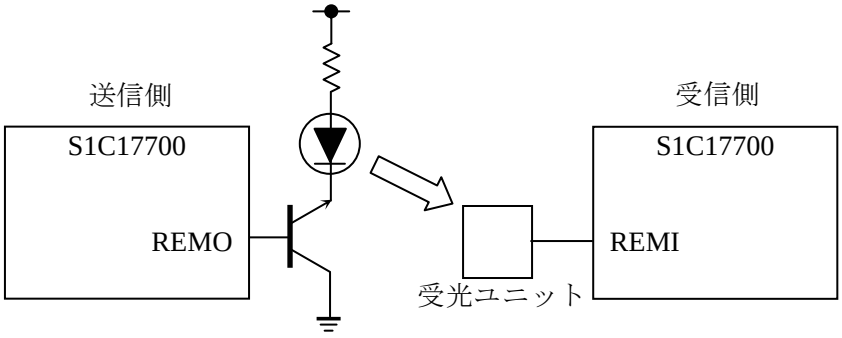

図 3.24　リモートコントローラ（REMC）サンプルソフトウェアハードウェア接続図

3.18.3　動作概要

（1）サンプルソフトウェア動作概要

- REMC 周辺回路をデータ送信用に設定します。
- REMC 周辺回路を使用して 10 個のデータを送信します。
- SimI/O に送信データを表示し、データを送信します。

```
<<< REMC transmit demonstration start >>>
Transmitted data : 05
Transmitted data : 10
Transmitted data : 15
…
<<< REMC transmit demonstration finish >>>
```

図 3.25　リモートコントローラ送信（REMC）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。

3.19　リモートコントローラ受信（REMC）

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、リモートコントローラ（以下、REMC）周辺回路を使用して以下の動作をします。

- REMC 周辺回路を使用しデータを受信する。

3.19.2　ハードウェア条件

ハードウェア条件はリモートコントローラ送信（REMC）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リモートコントローラ送信（REMC）サンプルソフトウェアが動作している S1C17700 マイコンを通信相手としてご使用下さい。

3.19.3　動作概要

（1）サンプルソフトウェア動作概要

- REMC 周辺回路をデー受信用に設定します。
- REMC 周辺回路から 10 個のデータを受信します。
- REMC 周辺回路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。

```
<<< REMC receive demonstration start >>>
Received data : 05
Received data : 10
Received data : 15
…
<<< REMC receive demonstration finish >>>
```

図 3.26　リモートコントローラ受信（REMC）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　電流測定

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。

- 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。

3.20.2　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。

3.20.3　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 キーを押下することで CPU を sleep モードにします。
- 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。
- 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。
- メニューが表示された状態で「4」を入力して ENTER キーを押下することで OSC1 と OSC3 と IOSC 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
3.Halt with OSC1/3 4.Halt with OSC1/3/IOSC
Please input number.
>1
sleeping
```

図 3.27　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。

注　P クロックは常に ON 状態なので測定値が期待値と異なる場合があります。任意の電流測定条件に合わせてソースコードを修正してご使用下さい。

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。
- 16 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 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。

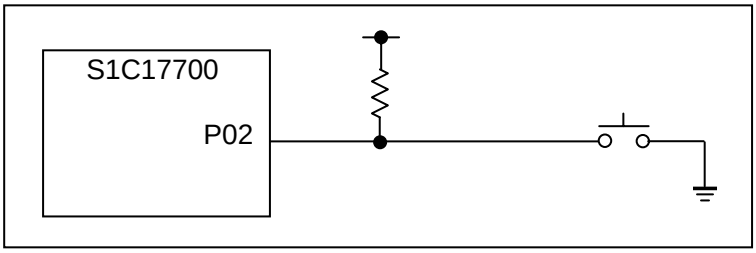

図 3.28　Sleep/Halt モード切替サンプルソフトウェア画面表示例

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

- 16 ビットタイマを開始し CPU を halt モードにします。
- 16 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。
- 16 ビットタイマ割り込みが 5 回発生すると 16 ビットタイマを停止し、CPU を sleep モードにします。
- P02 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 3.29　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）

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_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 アドバンスドタイマ（T16A）

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 t16a.c を参照して下さい。

表 4.4 アドバンスドタイマ（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	アドバンスドタイマ初期化
T16A_setTimerMode	アドバンスドタイマモード設定
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	アドバンスドタイマリセット
T16A_setTimerRun	アドバンスドタイマ開始/停止設定
T16A_initInt	アドバンスドタイマ割り込み初期化
T16A_controlInt	アドバンスドタイマ割り込み許可/禁止設定
T16A_resetIntFlag	アドバンスドタイマ割り込み要因フラグリセット
T16A_checkIntFlag	アドバンスド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。

4.5 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
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。

4.9 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 LCD ドライバ（LCD）

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 lcd.c を参照して下さい。

表 4.12 LCD ドライバ（LCD）サンプルドライバ関数一覧

関数名	機能名
LCD_initPower	LCD 電源初期化
LCD_init	LCD 初期化
LCD_setSEGAssignment	SEG 端子メモリ割り当て設定
LCD_setCOMAssignment	COM 端子メモリ割り当て設定
LCD_setDisplayArea	LCD 表示領域設定
LCD_setDisplayReverse	LCD 表示白黒反転設定
LCD_controlDisplay	LCD 表示制御
LCD_setContrast	LCD コントラスト設定
LCD_display1Seg	1 セグメント表示
LCD_initInt	LCD 割り込み初期化
LCD_controlInt	LCD 割り込み許可/禁止設定
LCD_resetIntFlag	LCD 割り込み要因フラグリセット
LCD_checkIntFlag	LCD 割り込み要因フラグチェック
LCD_setCOMPinAssignment	COM 端子ピン割り付け設定
LCD_displayDraw	矩形表示
LCD_setLCDClock	LCD クロック設定
LCD_controlLCDClock	LCD クロック供給許可/禁止設定
LCD_allClear	LCD 表示エリア全クリア

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。

4.13 電源電圧検出回路（SVD）

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 svd.c を参照して下さい。

表 4.13 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
SVD_initInt	SVD 割り込み初期化
SVD_controlInt	SVD 割り込み許可/禁止設定
SVD_resetIntFlag	SVD 割り込み要因フラグリセット
SVD_checkIntFlag	SVD 割り込み要因フラグチェック
SVD_setSVDClock	SVD クロック設定
SVD_controlSVDClock	SVD クロック供給許可/禁止設定

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 4.14 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 rfc.c を参照して下さい。

表 4.14 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
RFC_setLCDClock	RFC クロック設定
RFC_controlLCDClock	RFC クロック供給許可/禁止設定

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 A/D 変換器（ADC10）

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 adc.c を参照して下さい。

表 4.15 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_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 リモートコントローラ（REMC）

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 remc.c を参照して下さい。

表 4.16 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。

4.17 MISC

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 misc.c を参照して下さい。

表 4.17 MISC サンプルドライバ関数一覧

関数名	機能名
MISC_setFlashReadAccessCycle	Flash コントローラリードアクセスサイクル設定
MISC_setOSC1PeripheralControl	デバッグ時 OSC1 動作周辺機能設定
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_setPrescalerControl	プリスケーラ開始/停止設定
MISC_getActualIRAMSize	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。

4.18 マルチプレクサ（MUX）

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 mux.c を参照して下さい。

表 4.18 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_setLCDport	LCD 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	アドバンスドタイマポート設定

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。

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
411883200	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：411883200
2010 年 2 月 作成