S1C31D5x
周辺回路サンプル
ソフトウェアマニュアル

arm

評価ボード・キット、開発ツールご使用上の注意事項

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

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

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

本資料のご使用につきましては、次の点にご留意願います。

本資料の内容については、予告無く変更することがあります。

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

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

3. 特性値の数値の大小は、数直線上の大小関係で表しています。

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

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

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

目　次

1. 概要 1

1.1 動作環境 1

2. サンプルソフトウェアの操作 2

2.1 サンプルソフトウェアの構成 2

2.2 プログラムダウンロード及びプログラム実行のための準備 4

2.2.1 ハードウェアの接続 4

2.2.2 UART 用 USB アダプタとの接続 4

2.3 IAR EWARM サンプルソフトウェアの実行手順 6

2.3.1 ソフトウェアのセットアップ 6

2.3.2 ワークスペースのオープン 7

2.3.3 アクティブプロジェクトの選択 8

2.3.4 デバッグプローブの設定 9

2.3.5 フラッシュローダの設定 10

2.3.6 プロジェクトのビルド 11

2.3.7 プロジェクトのダウンロードおよびデバッグ 13

2.4 MDK-ARM（µVision）サンプルソフトウェアの実行手順 14

2.4.1 ソフトウェアのセットアップ 14

2.4.2 ワークスペースのオープン 15

2.4.3 アクティブプロジェクトの選択 16

2.4.4 デバッグプローブの設定 17

2.4.5 フラッシュローダの設定 19

2.4.6 プロジェクトのビルド 21

2.4.7 プロジェクトのダウンロードおよびデバッグ 22

3. サンプルソフトウェアの詳細 23

3.1 12 ビット A/D コンバータ(ADC12A) 23

3.2 クロックジェネレータ(CLG) 24

3.3 DMA コントローラ(DMAC) 25

3.4 I2C (I2C_S5U1C31D5xT1) 27

3.5 入出力ポート(PPORT) 29

3.6 同期式クワッドシリアルインタフェース(QSPI) 31

3.7 同期式クワッドシリアルインタフェース with DMA (QSPI_DMA) 32

3.8 同期式クワッドシリアルインタフェース マスタ(QSPI_MASTER) 33

3.9 同期式クワッドシリアルインタフェース スレーブ(QSPI_SLAVE) 38

3.10 IR リモートコントローラ(REMC3) 39

3.11 R/F 変換器(RFC) 40

3.12 リアルタイムクロック(RTCA) 42

3.13 同期式シリアルインタフェース マスタ(SPIA_MASTER) 43

3.14 同期式シリアルインタフェース スレーブ(SPIA_SLAVE) 45

3.15 電源電圧検出回路(SVD3) 46

3.16 16 ビットタイマ(T16) 47

3.17 16 ビット PWM タイマ(T16B) 48

3.18 UART(UART3) 49

3.19 ウォッチドッグタイマ(WDT2) 50

3.20 音声再生 (SOUNDPLAY) 51
3.21 メモリチェック (MEMCHECK) 58
3.22 フラッシュプログラミング(FLASH) 59
3.23 EEPROM ライブラリ (EEPROM) 60
3.24 ブートローダ (BootLoader) 61
4. デモソフトウェア 62
4.1 ハードウェア設定 62
4.2 音声再生デモモードの実行方法 64
4.3 音声 ROM アップデートモードの実行方法 66
5. 自己診断サンプルソフトウェア(IEC60730 規格対応) 68
5.1 概要 68
5.2 サンプル 69
付録.A. HW Processor ライブラリ仕様 71
付録.A.1 SOUNDPLAY 71
付録.A.2 MEMCHECK 74
付録.B. プレイリストファイル 76
付録.C. サンプルプロジェクトのコードサイズ概要 77
付録.D. 自己診断サンプルソフトウェア仕様 78
改訂履歴表 81

1. 概要

本マニュアルは、S1C31D5xマイクロコントローラ用サンプルソフトウェアの使用方法について記載しています。また、サンプルソフトウェア実行時の出力例も記載しています。

サンプルソフトウェアは、S1C31D5xマイクロコントローラに搭載されている各種周辺回路の使い方の説明用に作成されており、S1C31D5x周辺回路サンプルソフトウェアパッケージに含まれています。

S1C31D5xマイクロコントローラの詳細については、“S1C31D5xテクニカルマニュアル”を参照してください。

S1C31D5x周辺回路サンプルソフトウェアパッケージの構成を以下に示します。

- 周辺回路ライブラリ
- 周辺回路の使用例を記述したサンプルソフトウェア
- S1C31D5x 機種定義ファイルなどの機種別情報ファイル
- システムビューアデスクリプションファイル(svd ファイル)
- フラッシュローダ

周辺回路サンプルソフトウェアは、S1C31D5x周辺回路を制御するための周辺回路ライブラリの使い方を示すことを目的としています。各サンプルソフトウェアは、個々の周辺回路の機能の実例を示しており、周辺回路の各種モードを実行します。

S1C31D5x機種定義ファイルは、周辺回路制御レジスタのアドレス、アクセスサイズ、リード/ライトアクセスタイプを定義しています。個々のレジスタやそのビットフィールドは、S1C31D5x機種定義ファイルに記述されたハードウェア情報を基にアクセスされます。周辺回路ライブラリは、周辺回路の初期化や周辺回路機能の管理など、周辺回路の機能を簡単に制御するための関数を提供します。

※1 本マニュアルに加えて、弊社Webサイトから入手できる “S1C31D50/D51テクニカルマニュアル” および “S5U1C31D5xT1マニュアル”、”S5U1C31D51T2マニュアル”も併せて参照してください。

※2 本マニュアルは、ver.3.00以降のサンプルソフトウェアパッケージを対象としています。

1.1 動作環境

S1C31D5xサンプルソフトウェアを実行するためには、以下の機器が必要です。

- 評価ボード
  - S1C31D5x 搭載 S5U1C31D5xT1 評価ボードおよび S5U1C31D51T2(S1C31D51 を使用の場合のみ)
- デバッグプローブ *1
  - IAR Systems 社製 I-jet または SEGGER 社製 J-Link
- 統合開発環境
  - IAR Embedded Workbench for ARM®（IAR EWARM）または MDK-ARM®
- その他のデバイス（オプション）
  - UART 用 USB アダプタ

*1: I-jet は IAR EWARM でのみ使用可能です。J-Link は IAR EWARM と MDK-ARM の両方で使用可能です。

2. サンプルソフトウェアの操作

2.1 サンプルソフトウェアの構成

S1C31D5xサンプルソフトウェアは、S1C31D5xの各周辺回路ハードウェアとのインタフェースを担う周辺回路ライブラリ関数のコール方法を実例で示しています。各周辺回路に対応するライブラリが、ソースファイル（“xxx.c”）とインクルードファイル（“xxx.h”）の構成で用意されています。インクルードファイルには、それぞれのライブラリで使用する定数、型、関数が定義されています。

S1C31D5xサンプルソフトウェアは、ワークスペース“Examples”にまとめられています。ほとんどの場合、各周辺回路のサンプルソフトウェアは1つのサンプルプロジェクトとして提供されます。例えば、CLGプロジェクトは、CLG（クロックジェネレータ）を制御対象の周辺回路として作成されています。より複雑な周辺回路の場合は、その周辺回路の異なる側面を扱う複数のサンプルプロジェクトが用意されています。例えば、QSPIは複数のプロジェクト（接頭辞がQSPIのプロジェクト）が用意されています。図2.1.1にソフトウェアレイヤ間の関係を示します。

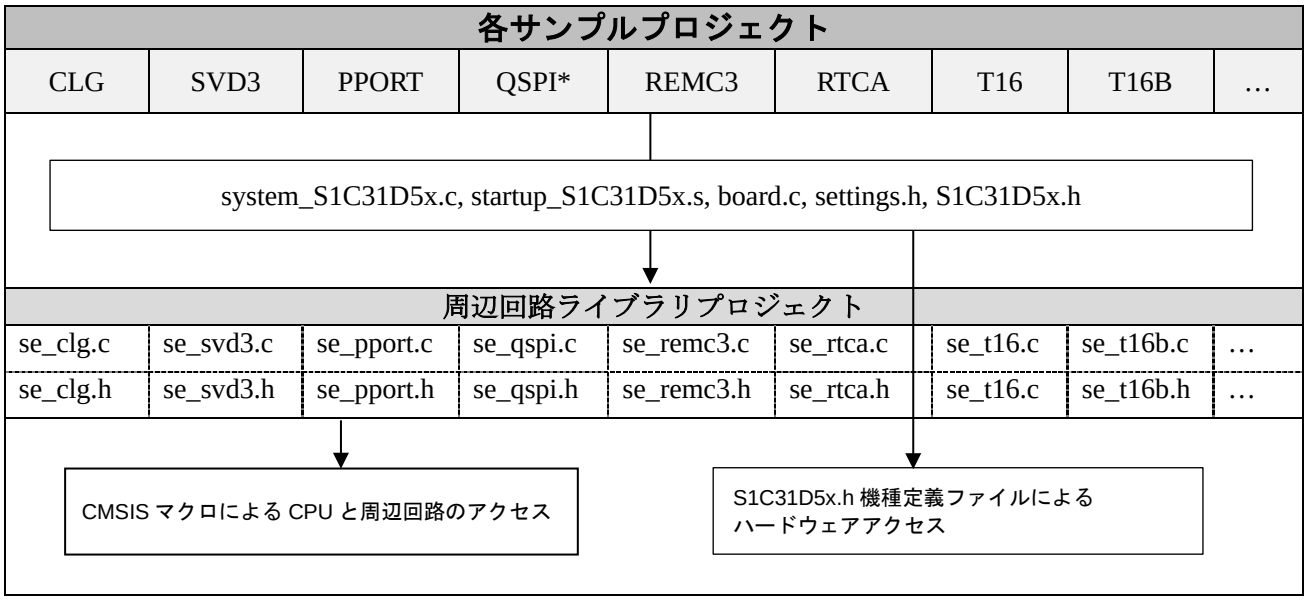


図 2.1.1 S1C31D5x サンプルソフトウェアワークスペース

注意：

- すべてのサンプルプロジェクトは、“CMSIS¥Device”フォルダにある IAR/Keil スタートアップファイルと IAR/Keil システムファイルを共有します。
- すべてのサンプルプロジェクトは、“board”フォルダにあるボード関連ファイルを共有します。
- board.cファイルのBoardInit関数に、起動時のシステム初期化処理が実装されています。
- BoardInit関数は、SystemInit関数からコールされます。この関数で、CPU動作周波数、Sys Tick値、優先順位などを設定します。
- SystemInit関数は、IAR/Keilスタートアップファイルからコールされます。

settings.hファイルには、デフォルト設定の起動動作を変更するための定数定義がコメントアウトして記述されています。表2.1.1に、ボード機能設定用定数と設定内容を示します。

表2.1.1　S1C31D5xボード定義

設定用定数	定義時	未定義時
UART_PRINTF	標準ライブラリのprintf関数が、UARTコンソールに対して出力を行います。ただし、ハードウェアの追加が必要です。 MDK-ARM(µVision)の場合、この設定機能を定義する必要があります。	セミホスティングライブラリのprintf関数が、IAR EWARMターミナルウィンドウに対して出力を行います。
EXECUTE_ON_OSC3	BoardInit関数で、CPUの動作クロックを高精度のクロックに切り換えます。	CPUがデフォルトクロックであるIOSCで動作します。
OSC3_AUTOTRIMMING_ON	CPU動作クロックの高い精度を維持するために、BoardInit関数で、OSC3トリミングが実行されます。	OSC3トリミングは実行されません。起動時間を短縮できます。
OSC3_SRC_XTAL	OSC3のクロックソースに水晶/セラミック発振回路が選択されます。	OSC3のクロックソースに内蔵発振回路が選択されます。
CACHE_ENABLED	Flashアクセス時の命令キャッシュが有効になります。デバッグターゲットでは無効です。	Flashアクセス時の命令キャッシュは無効です。
TICKLESS_ENABLED	SYSTICK割り込みが無効になります。	最後の起動時からのCPU動作時間の記録にSYSTICK割り込みが使用されます。
BOOT_LOADER	ブートローダーを実行します。	アプリケーションを実行します。
QSPI_MODE_SINGLE	QSPIの転送モードとしてシングルが選択されます。	QSPIの転送モードとしてデュアルまたはクワッドが選択されます。
SPIA_DMA	SPIA_MASTERおよびSPIA_SLAVE プロジェクトでDMAを使用します。	SPIA_MASTERおよびSPIA_SLAVE プロジェクトでDMAを使用しません。

注意：

- デフォルト設定ではすべてが未定義の状態です。
- 入出力にUARTコンソールを使用するには、settings.hファイル内のUART_PRINTFを定義している行をコメント解除してください。
- CPUのディープスリープ機能を使用する場合、settings.hファイル内のTICKLESS_ENABLEDを定義している行をコメント解除してください。

2.2 プログラムダウンロード及びプログラム実行のための準備

2.2.1 ハードウェアの接続

サンプルソフトウェアの実行およびデバッグには、以下のハードウェアの使用を推奨します。

- S5U1C31D5xT1 評価ボード
- デバッグプローブ（IAR Systems I-jet または SEGGER J-Link）

ハードウェア接続の詳細については、“S5U1C31D5xT1 マニュアル”を参照してください。

2.2.2 UART用USBアダプタとの接続

UART 用 USB アダプタは、UART を使用するサンプルプログラムで使用します。UART 用 USB アダプタを使用して S5U1C31D5xT1 評価ボートと PC を接続することで、PC との UART 通信が可能になります。図 2.2.2.1（写真）、図 2.2.2.2（結線図）に UART 用 USB アダプタと S5U1C31D5xT1 評価ボードの接続を示します。

UART 通信を行うためには、settings.h ファイルの UART_PRINTF の定義を有効にしてサンプルプログラムをビルドする必要があります（2.1 節および表 2.1.1 を参照）。また、PC 上でシリアル通信用のターミナルソフトウェアを起動し、シリアルポートを設定する必要があります。表 2.2.2.1 にシリアルポートの設定値を示します。

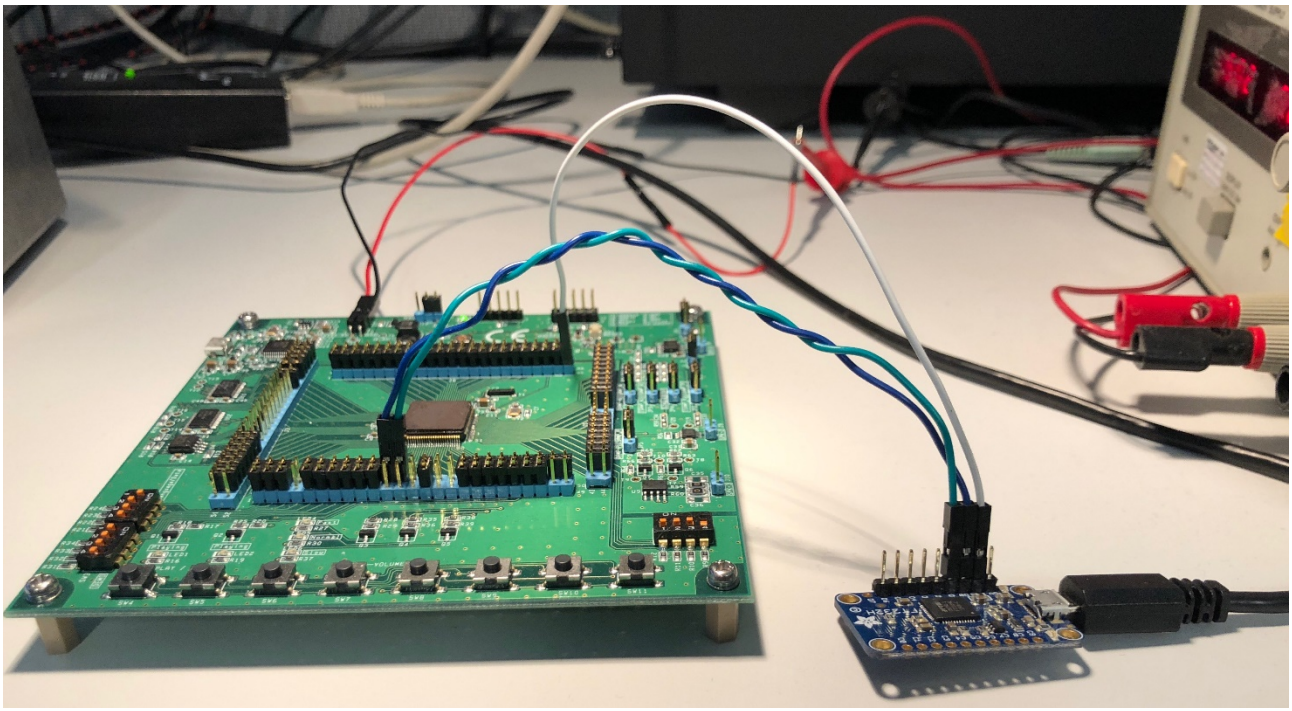


図2.2.2.1　S5U1C31D5xT1評価ボードとUART用USBアダプタの接続例

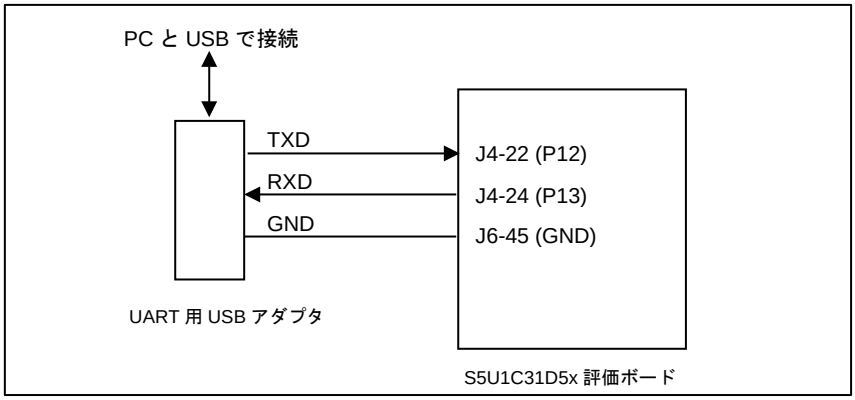


図2.2.2.2　S5U1C31D5xT1評価ボードとUART用USBアダプタとの結線

表2.2.2.1 ターミナルソフトウェアのシリアルポート設定

パラメータ	設定値
ボーレート	115200 bps
データ	8 ビット
ストップビット	1 ビット
パリティ	None

S5U1C31D5xT1 評価ボードの詳細については、“S5U1C31D5x マニュアル”を参照してください。

注： 図 2.2.2.1 で使用している UART 用 USB アダプタは市販の製品であり、弊社より提供されません。必要に応じて用意して下さい。

2.3 IAR EWARM サンプルソフトウェアの実行手順

2.3.1 ソフトウェアのセットアップ

IAR EWARM上でサンプルソフトウェアを実行する場合、事前にサンプルソフトウェアのセットアップが必要です。サンプルソフトウェアのセットアップは、以下の手順で行います。

(1) サンプルソフトウェアのダウンロード

S1C31D5x 周辺回路サンプルソフトウェアパッケージ（.exe）を弊社マイクロコントローラ Web サイトからダウンロードします。

(2) サンプルソフトウェアパッケージのインストール

起動中の他のすべてのプログラムを終了します。ダウンロードした実行形式のサンプルソフトウェアパッケージ（.exe）を管理者権限で実行してインストーラを起動します。インストーラ起動後に表示される「SOFTWARE LICENSE AGREEMENT」のライセンス条項を確認して、ライセンス条項に同意する場合は[I Agree]ボタンを選択します。続いて、所望のインストールフォルダを指定してインストールを行います。インストール終了後、セットアップユーティリティ「ToolchainSetup.exe」が自動的に起動するので [Next]ボタンを押します。次に、所望のバージョンの IAR EWARM をチェックボックスで選択して [Next]ボタンを押します。“Done”メッセージ表示後、[Finish]ボタンを押してインストールを完了します。

注: セットアップユーティリティは、機種別情報ファイル、svd ファイル、フラッシュローダを IAR EWARM のインストールフォルダへコピーします。

(3) IAR EWARM の起動

セットアップが正常に完了したら IAR EWARM IDE を起動します。

本サンプルソフトウェアの評価に使用した IAR EWARM のバージョン、及び、セットアップユーティリティの詳細については、サンプルソフトウェアパッケージ内の“README_IAR.txt”を参照して下さい。

2.3.2 ワークスペースのオープン

サンプルソフトウェアパッケージに含まれる、すべてのサンプルソフトウェアプロジェクトは、“Examples”フォルダに1つに纏められて格納されています。これらのプロジェクトは、ビルド時にsePeripheralLibraryを必要とします。

すべてのサンプルソフトウェアプロジェクトを含むワークスペースを開くには、IAR EWARMメニューの［ファイル］＞［開く］＞［ワークスペース］をクリックし、“Examples¥WORKSPACE¥S5U1C31D5xT1¥IAR”フォルダに移動して“Examples.eww”ファイルを選択します。

オプションとして、各サンプルソフトウェアプロジェクトのサブフォルダに、IARプロジェクトファイル（.ewp）とIARワークスペースファイル（.eww）を含む“board¥S5U1C31D5xT1¥IAR”サブフォルダが用意されています。個々のサンプルソフトウェアプロジェクトのワークスペースを開くには、IAR EWARMメニューの［ファイル］＞［開く］＞［ワークスペース］をクリックし、そのプロジェクトの“board¥S5U1C31D5xT1¥IAR”フォルダに移動してワークスペースファイル（.eww）を選択します。

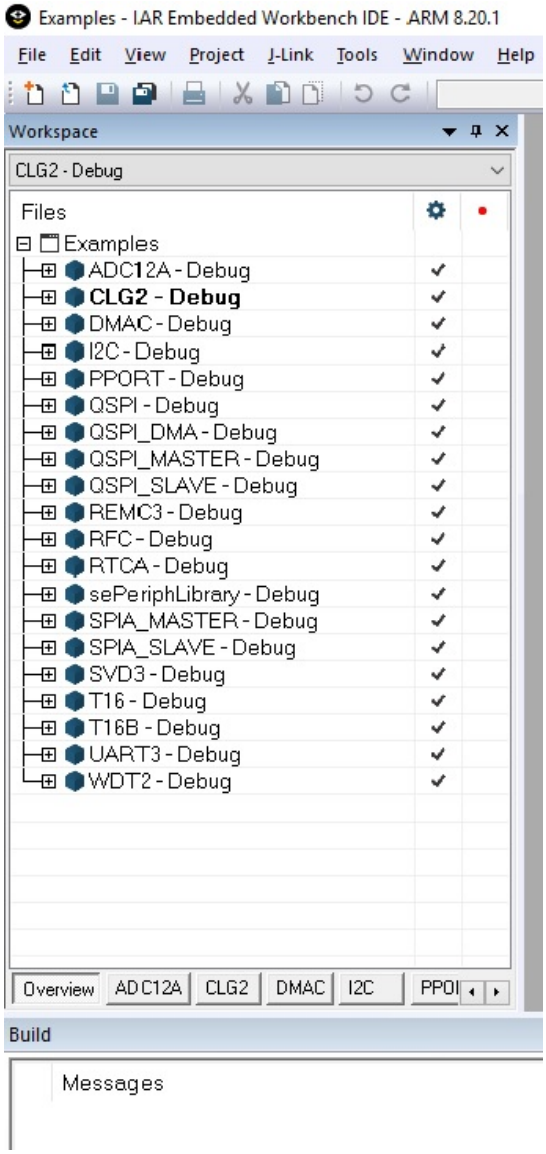


図 2.3.2.1 複数プロジェクトのワークスペースの例

2.3.3 アクティブプロジェクトの選択

ビルドおよびデバッグのターゲットとなるプロジェクト(アクティブプロジェクト)を選択します。IAR EWARMの［ワークスペース］ウィンドウからターゲットプロジェクトを選択し、右クリックで表示されるメニューから［アクティブに設定］を選択します。また、［ワークスペース］ウィンドウ上部のドロップダウンリストを使用することで、アクティブプロジェクトとビルド構成を同時に選択することができます（図2.3.3.1）。

サンプルソフトウェアの各プロジェクトには、以下に示す3種類のビルド構成が用意されています。

- Debug - 内蔵 RAM 上で実行されるプログラム ROM を生成します。
  最適化レベルは“低”に設定されています。
- DebugFlash - 内蔵 Flash メモリ上で実行されるプログラム ROM を生成します。
  最適化レベルは“低”に設定されています。
- ReleaseFlash - 内蔵 Flash メモリ上で実行されるプログラム ROM を生成します。
  最適化レベルは“高”に設定されています。

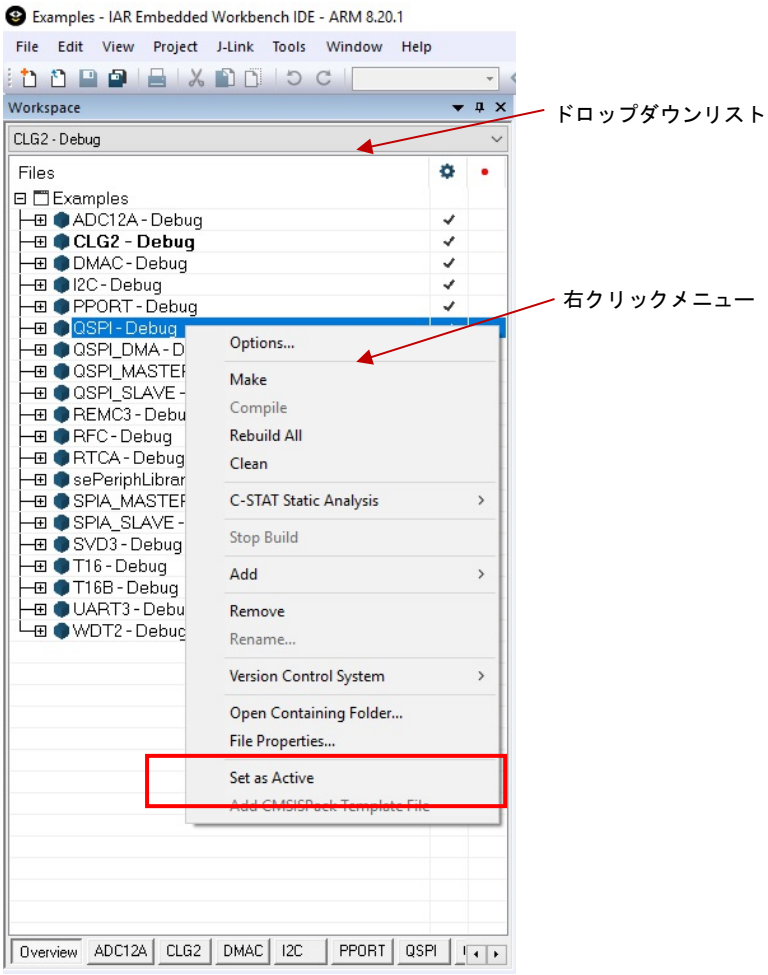


図 2.3.3.1 アクティブプロジェクトの設定

2.3.4 デバッグプローブの設定

ターゲットボードにデバッグプローブを接続してデバッグする場合、使用するデバッグプローブの種類に応じて、IAR EWARM 上でドライバの設定が必要です。

デバッグプローブの設定は、以下の手順で行います。

(1) IAR EWARM メニューの［プロジェクト］＞［オプション］を選択します。

(2) 表示されたダイアログの［カテゴリ］リストから［デバッガ］を選択します（図 2.3.4.1）。

(3) ［設定]タブを選択し、[ドライバ］のドロップダウンリストから使用するデバッグプローブの種類を選択します。I-jet を使用する場合は“I-jet/JTAGjet”、J-Link を使用する場合は“J-Link/J-Trace”を選択してください（図 2.3.4.1）。

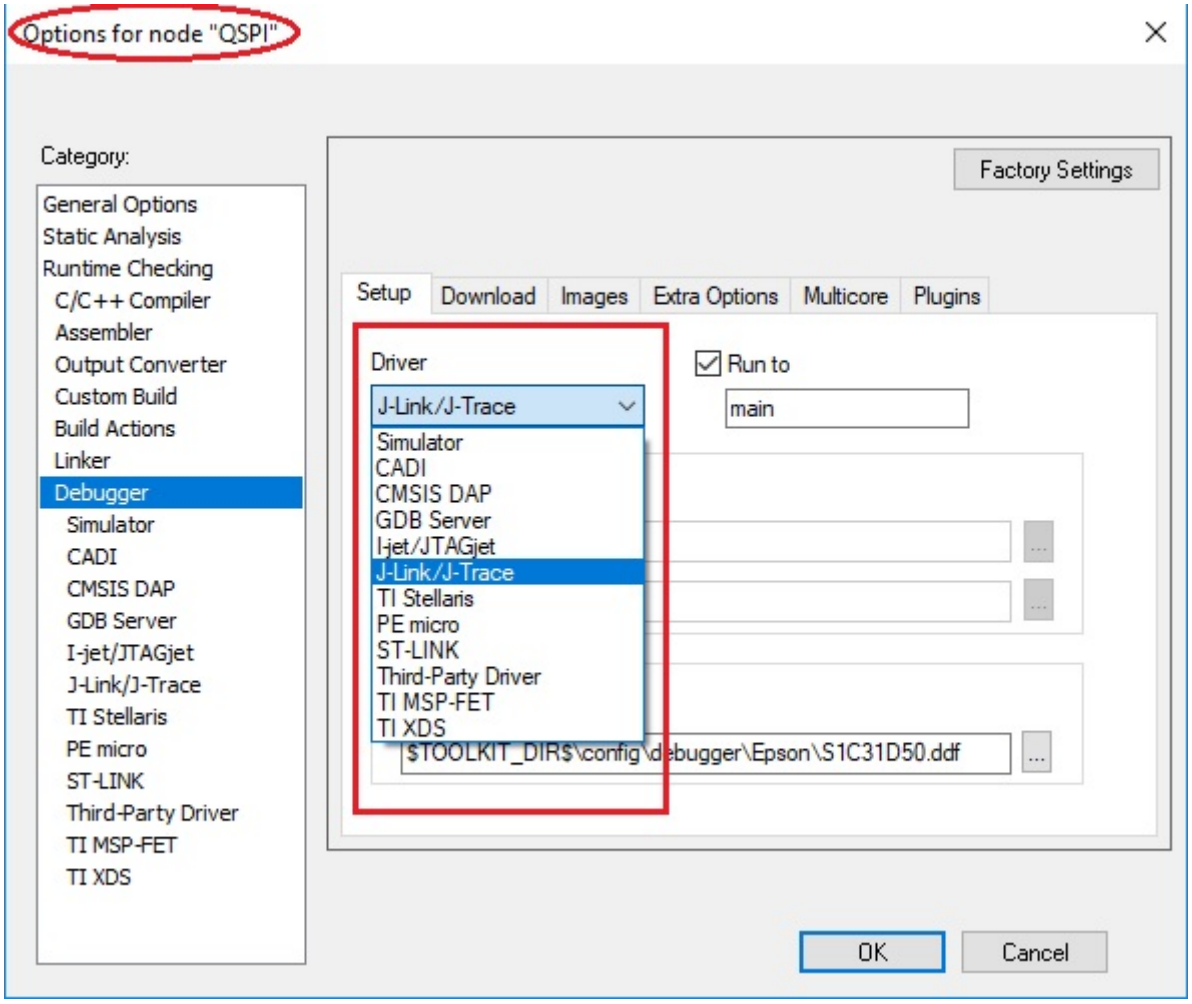


図 2.3.4.1 デバッグプローブの設定

2.3.5 フラッシュローダの設定

アクティブプロジェクトのビルド構成が DebugFlash または ReleaseFlash の場合、内蔵 Flash メモリにプログラムをロード（Flash プログラミング）するためのフラッシュローダを設定する必要があります。ビルド構成が Debug の場合は、内蔵 Flash メモリではなく内蔵 RAM にプログラムをロードするため、フラッシュローダの設定を解除する必要があります。

フラッシュローダの設定は、以下の手順で行います。

- ビルド構成が Debug の場合（RAM 上のプログラムを実行）

(1) IAR EWARM メニューの［プロジェクト］＞［オプション］を選択します。

(2) 表示されたダイアログの［カテゴリ］リストから［デバッガ］を選択します（図 2.3.5.1）。

(3) ［ダウンロード］タブを選択します（図 2.3.5.1）。

(4) ［フラッシュローダを使用する(U)］のチェックを無効にします（図 2.3.5.1）。

- ビルド構成が DebugFlash または ReleaseFlash の場合（内蔵 Flash メモリのプログラムを実行）

(1) IAR EWARM メニューの［プロジェクト］＞［オプション］を選択します。

(2) 表示されたダイアログの［カテゴリ］リストから［デバッガ］を選択します（図 2.3.5.1）。

(3) ［ダウンロード］タブを選択します（図 2.3.5.1）。

(4) ［フラッシュローダを使用する(U)］のチェックを有効にします（図 2.3.5.1）。

(5) ［デフォルトの.board ファイルのオーバライド］のチェックを有効にします（図 2.3.5.1）。

(6) ［…］ボタンをクリックし、.board ファイルとして“S1C31D5x_int.board”を選択します（図 2.3.5.1）。

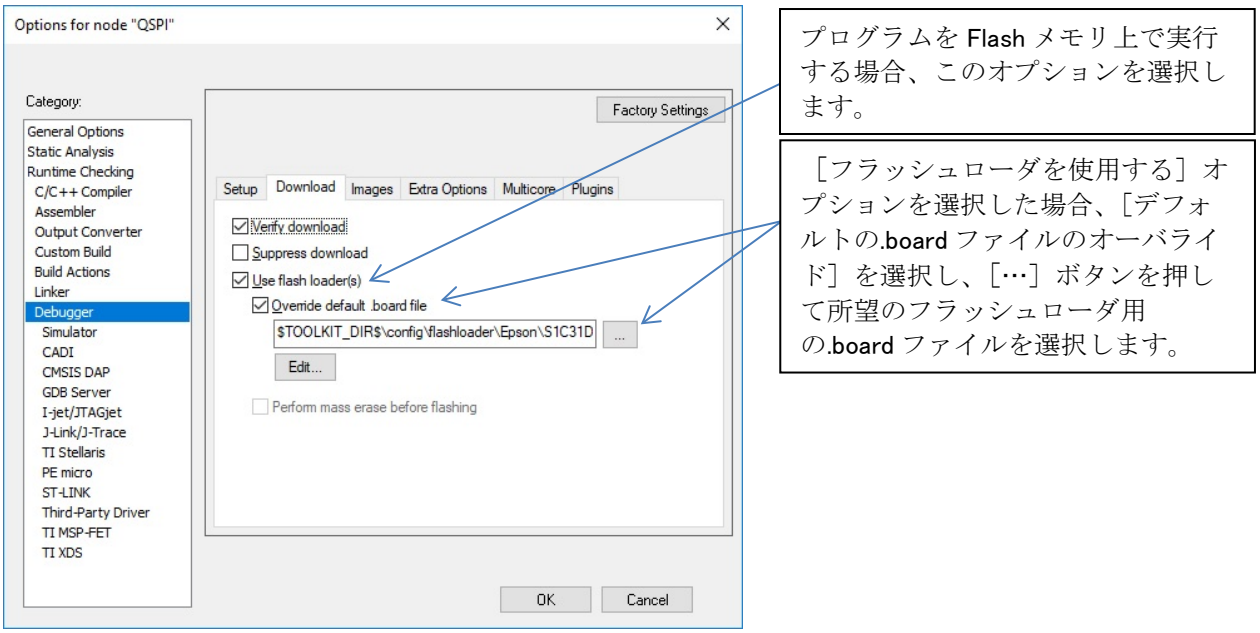


図 2.3.5.1 フラッシュローダの設定

2.3.6 プロジェクトのビルド

プロジェクトをビルドするには、IAR EWARM メニューの［プロジェクト］から［メイク］、［すべてを再ビルド］の 2 つのビルドコマンドのいずれかを選択します（図 2.3.6.1）。

注意： ビルド中にリンカエラーが発生する場合、ライブラリプロジェクト sePeriphLibrary がビルドされていない可能性があります。このライブラリプロジェクトをビルドしてから、再度、ターゲットプロジェクトをビルドしてください。このとき、ライブラリプロジェクトはターゲットプロジェクトと同じビルド構成を選択してください。例えば、“CLG-Debug”をビルドする場合、“sePeriphLibrary-Debug”を選択してビルドしてください。

また、IAR EWARM メニューの［プロジェクト］＞［バッチビルド］を選択することで、サンプルソフトウェアに含まれる、すべてのプロジェクトをまとめてビルドすることができます（図 2.3.6.1）。［バッチビルド］ダイアログの［Make］または［Rebuild All］ボタンのいずれかをクリックすると、指定されたビルド構成のプロジェクトがまとめてビルドされます（図 2.3.6.2）。

バッチビルドは、以下に示すビルド構成から選択して実行できます。

- all_Debug - ビルド構成が Debug のプロジェクトをまとめてビルド
- all_DebugFlash - ビルド構成が DebugFlash のプロジェクトをまとめてビルド
- all_ReleaseFlash - ビルド構成が ReleaseFlash のプロジェクトをまとめてビルド

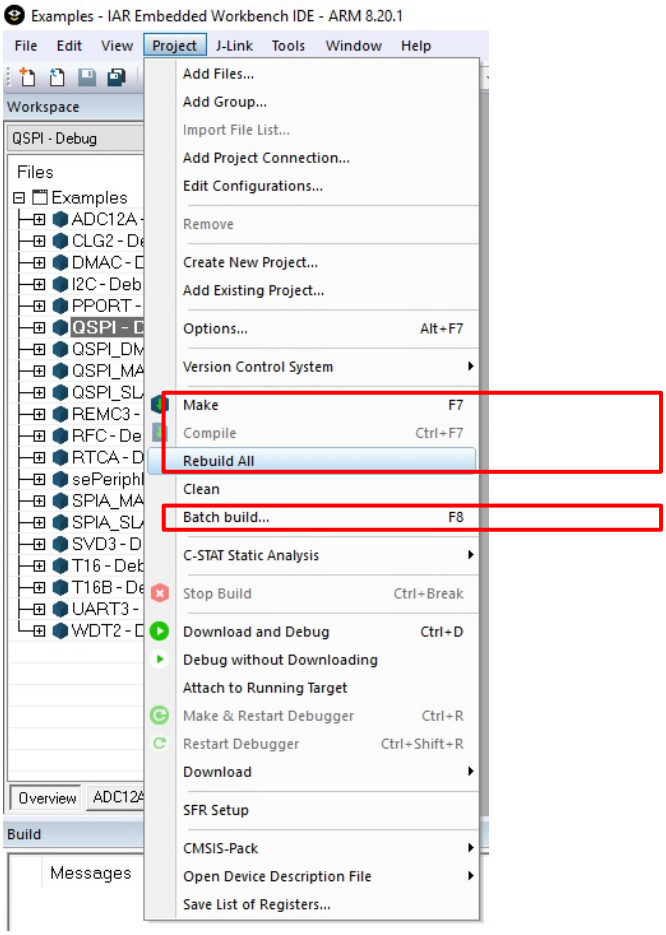


図 2.3.6.1 ビルドコマンド

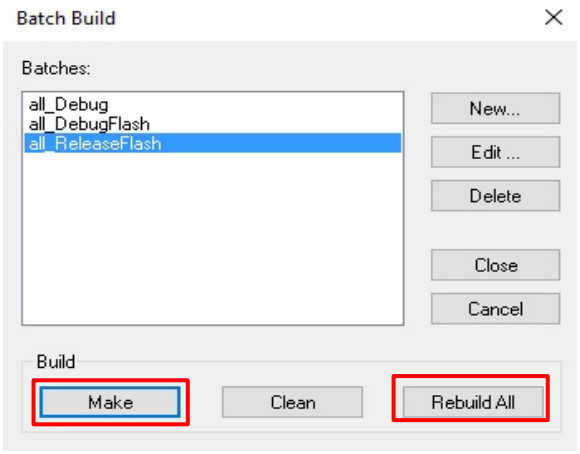


図 2.3.6.2 バッチビルド

2.3.7 プロジェクトのダウンロードおよびデバッグ

ビルドに成功したら、アクティブプロジェクトをターゲットボードにダウンロードします。アクティブプロジェクトをダウンロードするには、IAR EWARMメニューの［プロジェクト］＞［ダウンロードしてデバッグ］を選択します（図2.3.7.1）。

プロジェクトがDebugビルドの場合、プログラムイメージは内蔵RAMにロードされてデバッグを開始します。プロジェクトがDebugFlashビルドまたはReleaseFlashビルドの場合、プログラムイメージは内蔵Flashメモリにロードされてデバッグを開始します。

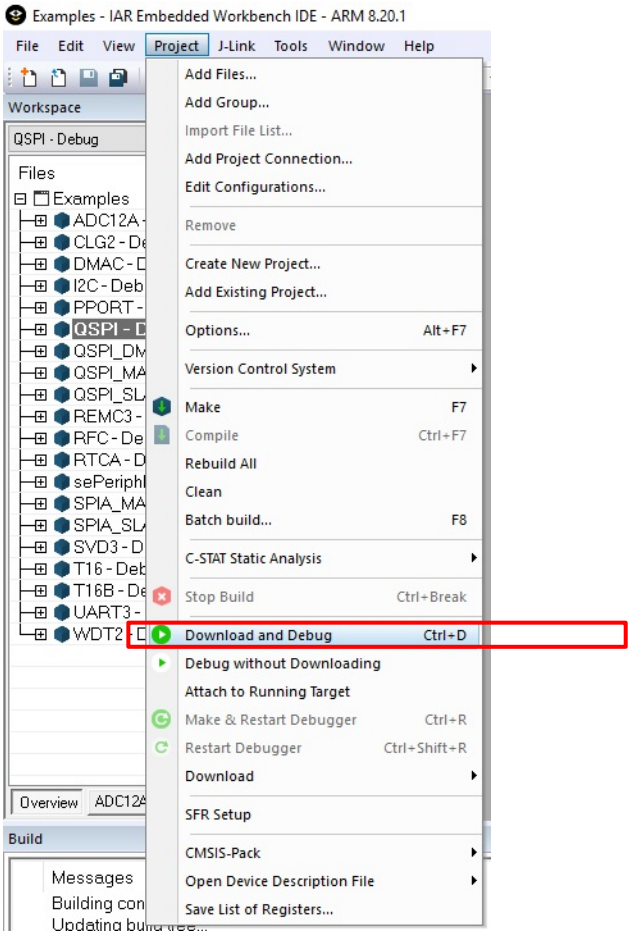


図 2.3.7.1 プロジェクトのダウンロードおよびデバッグ

2.4 MDK-ARM（µVision）サンプルソフトウェアの実行手順

2.4.1 ソフトウェアのセットアップ

MDK-ARM（µVision）上でサンプルソフトウェアを実行する場合、事前にサンプルソフトウェアのセットアップが必要です。サンプルソフトウェアのセットアップは、以下の手順で行います。

(1) サンプルソフトウェアのダウンロード

S1C31D5x 周辺回路サンプルソフトウェアパッケージ（.exe）を弊社マイクロコントローラ Web サイトからダウンロードします。

(2) サンプルソフトウェアパッケージのインストール

起動中の他のすべてのプログラムを終了します。ダウンロードした実行形式のサンプルソフトウェアパッケージ（.exe）を管理者権限で実行してインストーラを起動します。インストーラ起動後に表示される「SOFTWARE LICENSE AGREEMENT」のライセンス条項を確認して、ライセンス条項に同意する場合は[I Agree]ボタンを選択します。続いて、所望のインストールフォルダを指定してインストールを行います。インストール終了後、セットアップユーティリティ「ToolchainSetup.exe」が自動的に起動するので [Next]ボタンを押します。次に、所望のバージョンの µVision とデバッグプローブをチェックボックスで選択して[Next]ボタンを押します。“Done”メッセージ表示後、[Finish]ボタンを押してインストールを完了します。

注: セットアップユーティリティは、フラッシュローダを µVision のインストールフォルダへコピーし、デバッグ設定ファイルを各サンプルソフトウェアプロジェクトのワークスペースフォルダへコピーします。

(3) µVision の起動

セットアップが正常に完了したら µVision を起動します。

本サンプルソフトウェアの評価に使用した MDK-ARM(µVision)のバージョン、及び、セットアップユーティリティの詳細については、サンプルソフトウェアパッケージ内の“README_KEIL.txt”を参照して下さい。

2.4.2 ワークスペースのオープン

サンプルソフトウェアパッケージに含まれる、すべてのサンプルソフトウェアプロジェクトは、“Examples”フォルダに1つに纏められて格納されています。これらのプロジェクトは、ビルド時にsePeripheralLibraryを必要とします。

すべてのサンプルソフトウェアプロジェクトを含むワークスペースを開くには、µVisionメニューの［Project］＞［Open Project…］をクリックし、“Examples¥WORKSPACE¥S5U1C31D5xT1¥ARM”フォルダに移動して“Examples.uvmpw”ファイルを選択します。

オプションとして、各サンプルソフトウェアプロジェクトのサブフォルダに、µVision プロジェクトファイル（.uvprojx）と µVision マルチプロジェクトワークスペースファイル（.uvmpw）を含む“board¥S5U1C31D5xT1¥ARM”サブフォルダが用意されています。個々のサンプルソフトウェアプロジェクトのワークスペースを開くには、µVision メニューの［Project］＞［Open Project…］をクリックし、そのプロジェクトの“board¥S5U1C31D5xT1¥ARM”サブフォルダに移動してマルチプロジェクトワークスペースファイル（.uvmpw）を選択します。

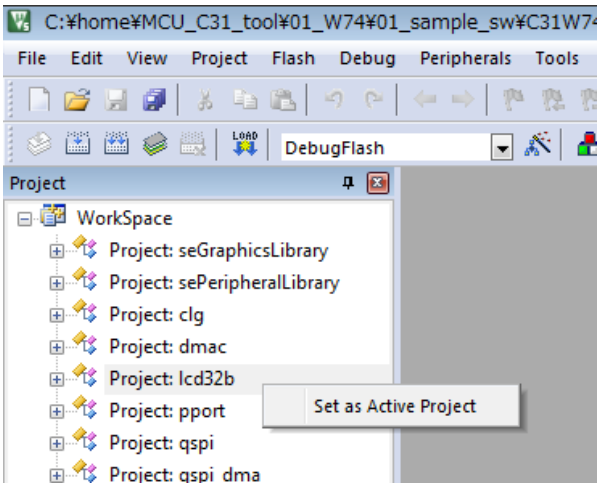


図 2.4.2.1 複数プロジェクトのワークスペースの例

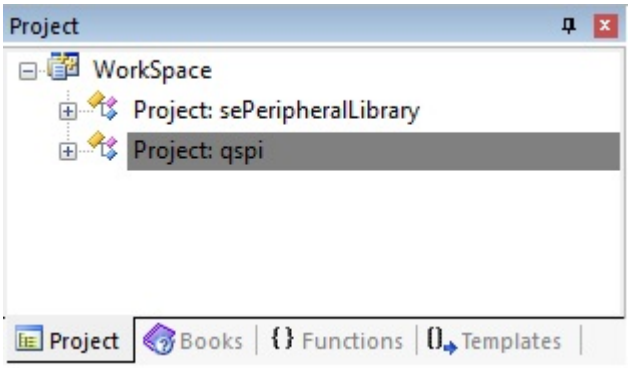


図 2.4.2.2 単体プロジェクトのワークスペースの例

2.4.3 アクティブプロジェクトの選択

ビルドおよびデバッグのターゲットとなるプロジェクト（アクティブプロジェクト）を選択します。µVisionの［Project］ウィンドウ上からターゲットプロジェクトを選択し、右クリックで表示されるメニューから［Set as Active Project］を選択します（図2.4.3.1）。次に、ツールバー上のドロップダウンリストからビルド構成を選択します（図2.4.3.2）。

サンプルソフトウェアの各プロジェクトには、以下に示す 2 種類のビルド構成が用意されています。

- Debug - 内蔵 RAM 上で実行されるプログラム ROM を生成します。
- DebugFlash - 内蔵 Flash メモリ上で実行されるプログラム ROM を生成します。

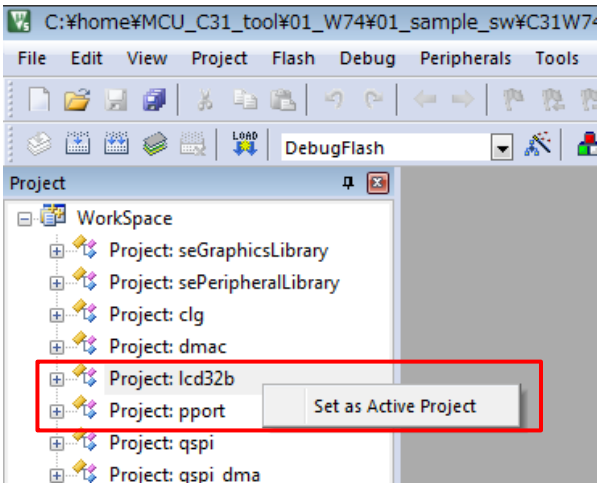


図 2.4.3.1 アクティブプロジェクトの設定

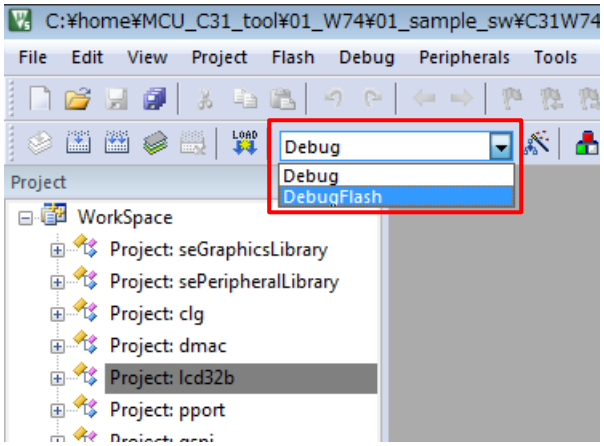


図 2.4.3.2 ビルド構成の選択

2.4.4 デバッグプローブの設定

ターゲットボードにデバッグプローブを接続してデバッグする場合、使用するデバッグプローブの種類に応じて、µVision 上でドライバの設定が必要です。

デバッグプローブの設定は、以下の手順で行います。

(1) µVision メニューの［Project］>［Options for {project} - Target ‘{build configuration}’］を選択します。

(2) ［Options for Target ‘{build configuration}’］ダイアログの［Debug］タブを選択します（図 2.4.4.1）。

(3) ［Use:］チェックボックスの右横にあるドロップダウンリストから［J-Link/J-TRACE Cortex］を選択します（図 2.4.4.1）。

(4) 上記ドロップダウンリストの右横にある［Settings］ボタンを押下します（図 2.4.4.1）。

(5) ［Cortex JLink/JTrace Target Driver Setup］ダイアログの［Port］ドロップダウンリストから［SW］を選択します（図 2.4.4.1）。

(6) 最後に[OK]ボタンを押下して、すべてのダイアログを閉じます。

注意：デバックプローブの設定は、デバッグプローブ（J-Link）を PC に接続した状態で行ってください。

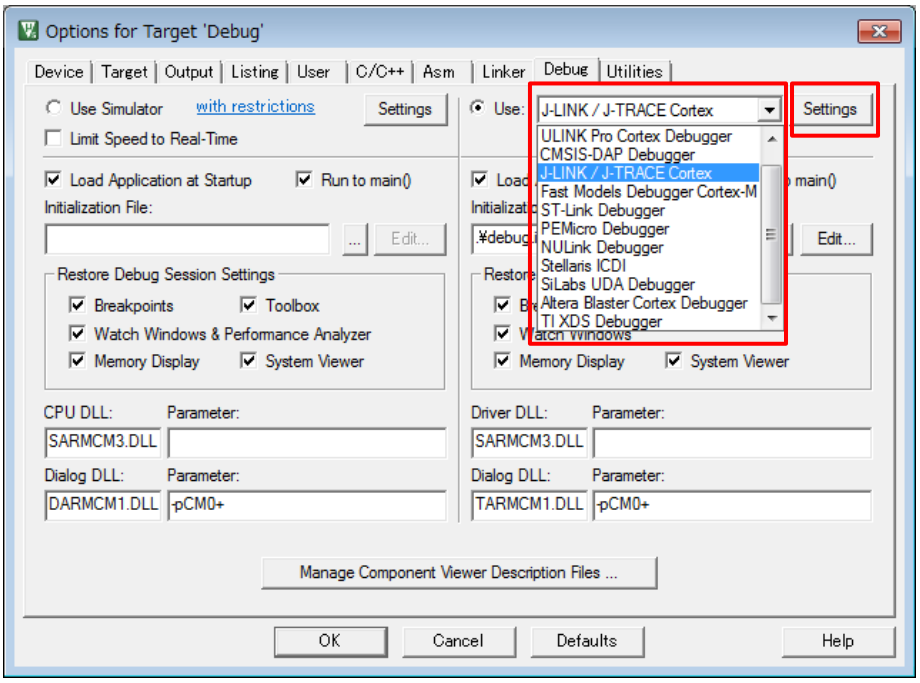


図 2.4.4.1 デバッグプローブの選択

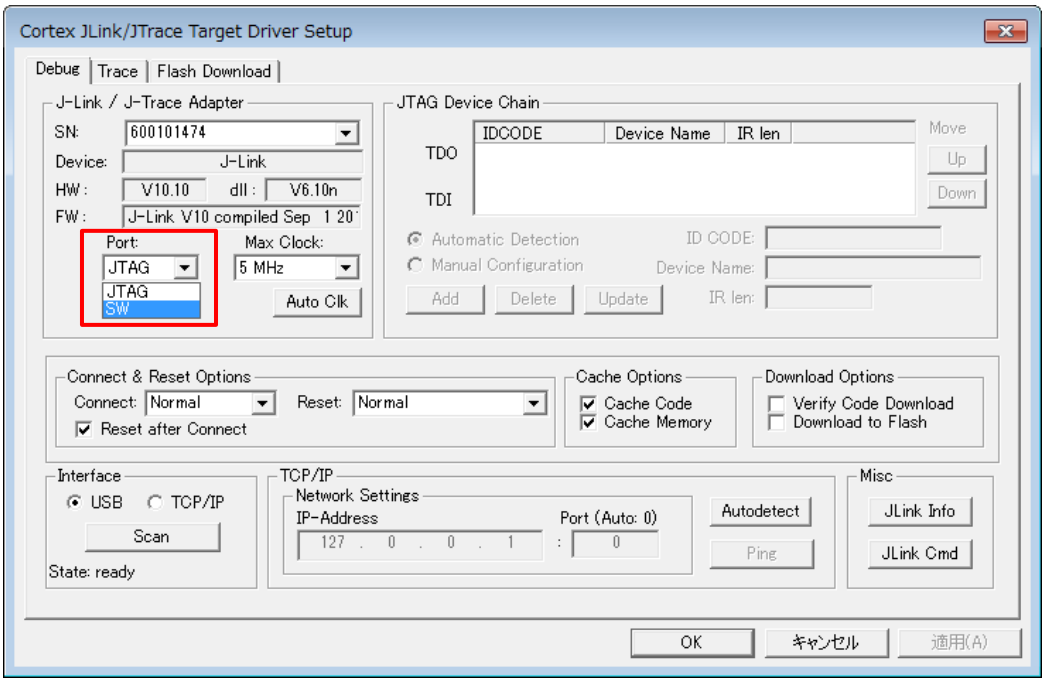


図 2.4.4.2 J-Link ドライバの設定

2.4.5 フラッシュローダの設定

アクティブプロジェクトのビルド構成が DebugFlash の場合、内蔵 Flash メモリにプログラムをロード（Flash プログラミング）するためにフラッシュローダを設定する必要があります。ビルド構成が Debug の場合は、内蔵 RAM にプログラムをロードして実行するためフラッシュローダの設定を解除する必要があります。

フラッシュローダの設定は、以下の手順で行います。

- ビルド構成が Debug の場合（RAM 上のプログラムを実行）

(1) µVision メニューの［Project］>［Options for {project} - Target ‘{build configuration}’］を選択します。

(2) ［Options for Target ‘{build configuration}’］ダイアログの［Debug］タブを選択します。

(3) ［Initialize File］エディットボックス右横の［…］ボタンを押下し、debug.ini ファイルを選択します（図 2.4.5.1）。

(4) ［Options for Target ‘{build configuration}’］ダイアログの［Utilities］タブを選択し、［Configure Flash Menu Command］枠内の［Settings］ボタンを押下します。

(5) ［Cortex JLink/JTrace Target Driver Setup]ダイアログの[Flash Download]タブを選択します。

(6) ［Remove］ボタンを押下して、[Programming Algorithm]リスト上のフラッシュローダを全て削除します。

(7) ［Download Function］枠内の［Do not Erase］、［Reset and Run］のチェックを有効にし、それ以外のチェックを無効にします（図 2.4.5.2）。

- ビルド構成が DebugFlash の場合（内蔵 Flash メモリのプログラムを実行）

(1) µVision メニューの［Project］>［Options for {project} - Target ‘{build configuration}’］を選択します。

(2) ［Options for Target ‘{build configuration}’］ダイアログの［Debug］タブを選択します。

(3) ［Initialize File］エディットボックスに何も設定せずに空白にします。

(4) ［Options for Target ‘{build configuration}’］ダイアログの［Utilities］タブを選択し、［Configure Flash Menu Command］枠内の［Settings］ボタンを押下します。

(5) ［Cortex JLink/JTrace Target Driver Setup]ダイアログの[Flash Download]タブを選択します。

(6) ［Remove］ボタンを押下して、［Programming Algorithm］リスト上のフラッシュローダを全て削除します。

(7) ［Add］ボタンを押下して、［Add Flash Programming Algorithm］ダイアログを開きます。

(8)［Add Flash Programming Algorithm]ダイアログ上のリストからフラッシュローダ“S1C31D5xint 192kB Flash”を選択し、[Add]ボタンを押下します。

(9) ［Download Function］枠内の［Erase Sectors］、［Program］、［Verify］のチェックを有効にし、それ以外のチェックを無効にします（図 2.4.5.3）。

注意： Debug ビルドを選択して実行する場合、プログラムは RAM にロードされるため内蔵 Flash メモリは更新されません。したがって、通常は Vector テーブルからロードされるプログラムカウンタおよびスタックレジスタの設定を、[Initialization File]オプションを使用して追加設定する必要があります。Debug ビルドを使用しているサンプルソフトウェアプロジェクトでは、debug.ini ファイルを使用して、これらの値を設定しています。DebugFlash ビルドのサンプルソフトウェアでは、［Initialization File］フィールドは何も設定せず空白のままにしてください。

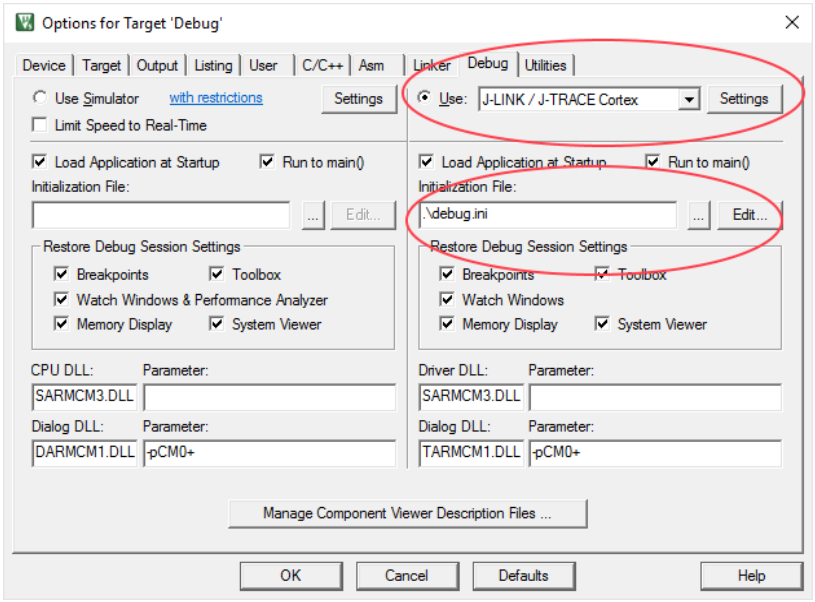


図 2.4.5.1 debug.ini ファイルの設定

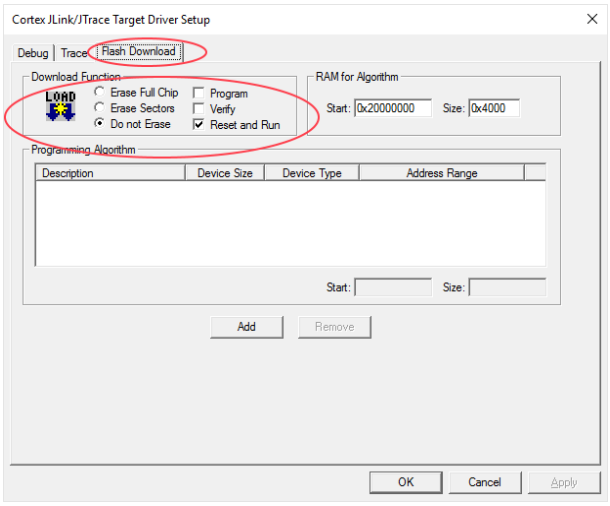


図 2.4.5.2 フラッシュローダの解除

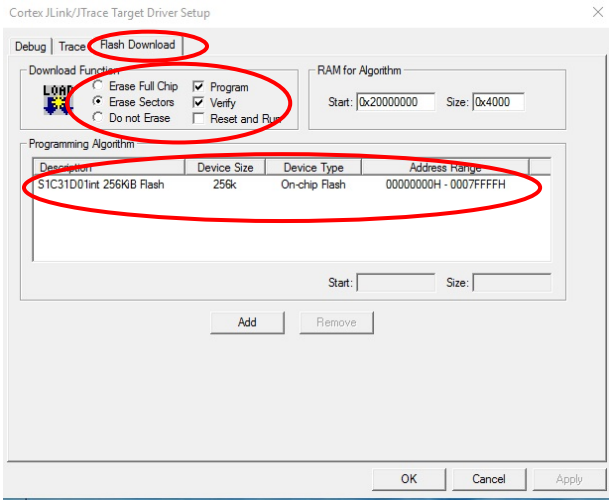


図 2.4.5.3 フラッシュローダの設定

2.4.6 プロジェクトのビルド

アクティブプロジェクトをビルドするには、µVision メニューの［Project］から［Build］、［Rebuild］の 2 つのビルドコマンドのいずれかを選択します（図 2.4.6.1）。

注意： ビルド中にリンカエラーが発生する場合は、ライブラリプロジェクト sePeriphLibrary がビルドされていない可能性があります。このライブラリプロジェクトをビルドしてから、再度、アクティブプロジェクトをビルドしてください。

また、µVision メニューの［Project］＞［Batch Build］を選択することで、すべてのサンプルソフトウェアプロジェクトをまとめてビルドすることが可能です（図 2.4.6.1）。［Batch Build］ダイアログの［Build］、［Rebuild］ボタンのいずれかをクリックすると、選択されたすべてのプロジェクトがビルドされます（図 2.4.6.2）。

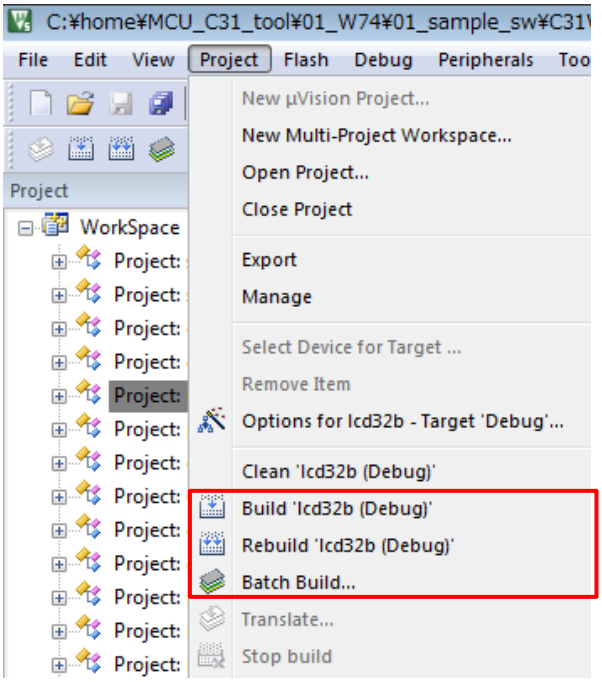


図 2.4.6.1 ビルドコマンド

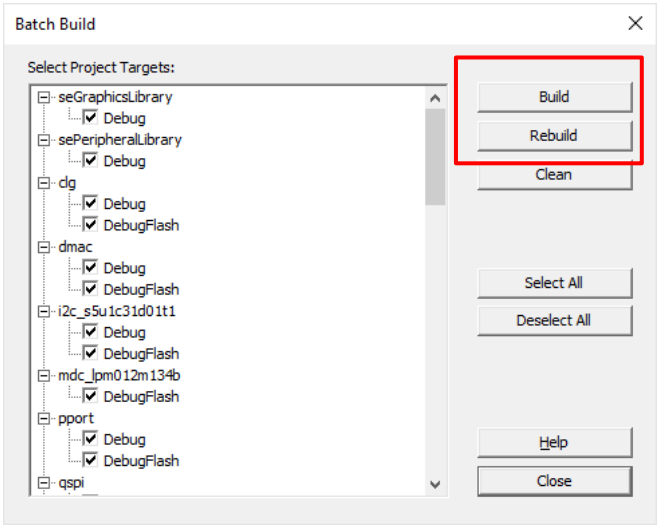


図 2.4.6.2 バッチビルド

2.4.7 プロジェクトのダウンロードおよびデバッグ

ビルドに成功したら、アクティブプロジェクトをターゲットボードにダウンロードします。アクティブプロジェクトをダウンロードするには、µVisionメニューの［Flash］＞［Download］を選択します（図2.4.7.1）。プロジェクトがDebugビルドの場合、プログラムイメージは内蔵RAMにロードされます。DebugFlashビルドの場合、プログラムイメージは内蔵Flashメモリにロードされます。

ターゲットボードにダウンロードしたプログラムイメージのデバッグを開始するには、µVisionメニューの［Debug］＞［Start/Stop Debug Session］を選択します（図2.4.7.2）。

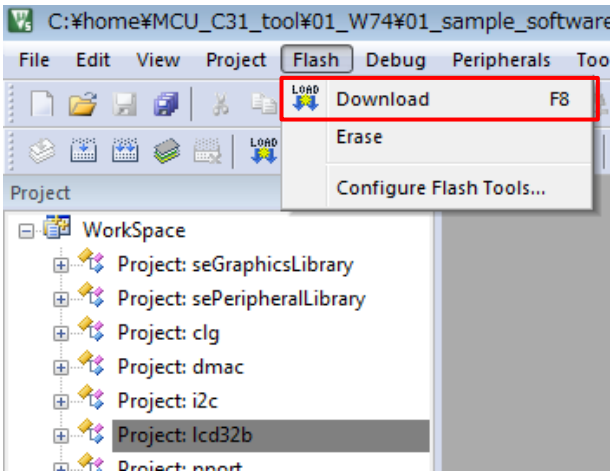


図 2.4.7.1 ダウンロード

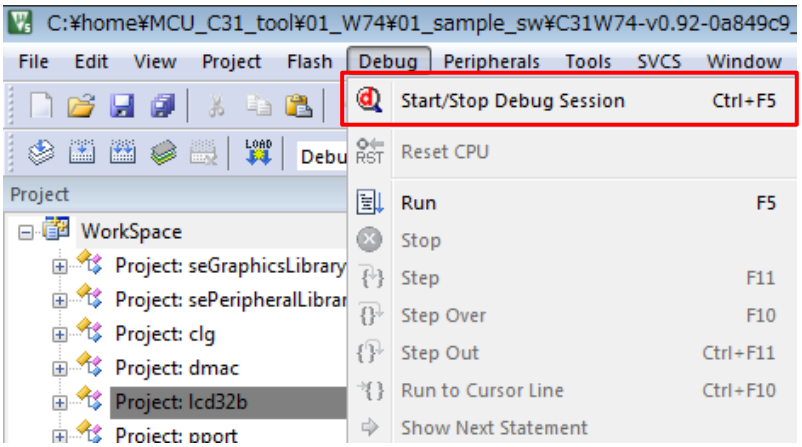


図 2.4.7.2 デバッグ

3. サンプルソフトウェアの詳細

いくつかのサンプルプロジェクトでは、ハードウェアのセットアップが必要です。ハードウェアセットアップに関しては S5U1C31D5xT1 のマニュアルも併せて参照してください。

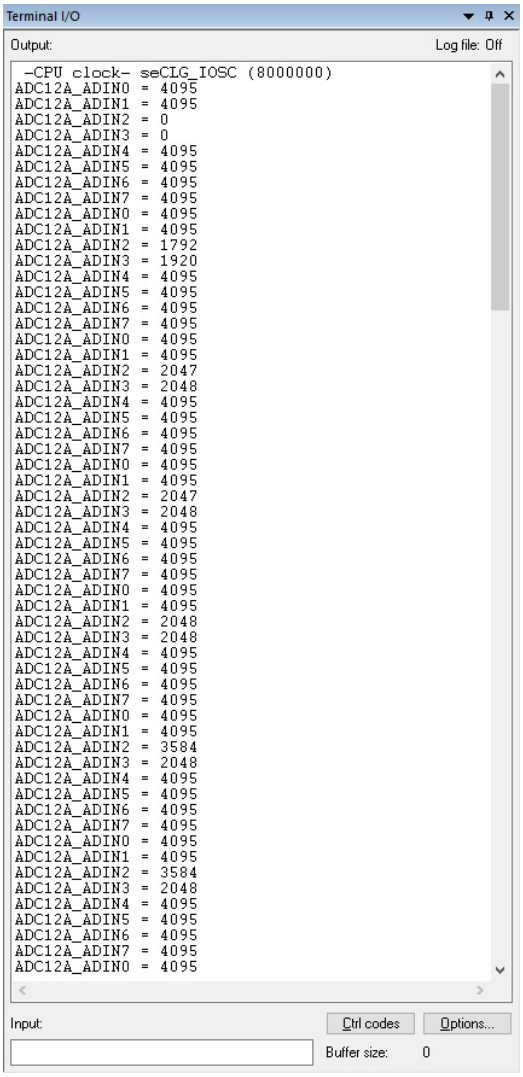
3.1 12ビットA/Dコンバータ(ADC12A)

このサンプルプロジェクトは、12 ビット A/D コンバータの各チャンネルからアナログデータを取得し、アナログデータの値を画面に出力します。

動作

1. ADC12A を初期化します。
2. ADC12A の実行を開始します。
3. ADC12A で取得したアナログデータをターミナルウィンドウに出力します。
4. ADC12A チャンネルへの入力電圧を変化させると、ターミナルウィンドウへの出力値が変化します。

出力例



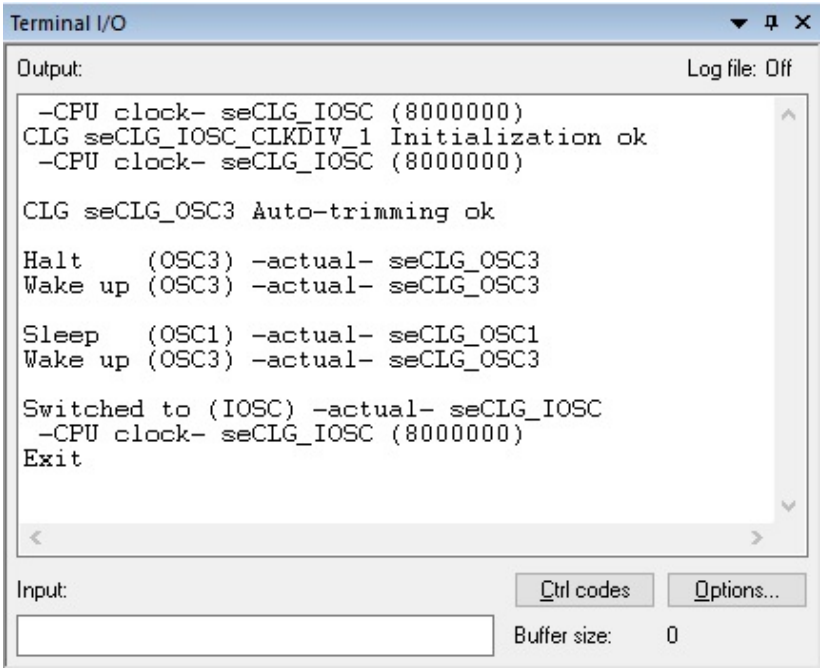
3.2 クロックジェネレータ(CLG)

このサンプルプロジェクトは、OSC の起動やウェイクアップクロック、スリープモードの設定など、様々な CLG 機能を実行します。

動作

1. CLG を初期化します。
2. IOSC のオートトリミングを実行します。
3. 様々なクロックで実行して、SLEEP または HALT の状態を確認します。
4. 様々なクロックを実行して、ウェイクアップの状態を確認します。

出力例



注意

OSC3 の実行には、水晶/セラミック発振ではなく内蔵発振を選択してください。水晶/セラミック発振ではトリミングができないため、オートトリミングが失敗します。そのため、settings.h ファイルの OSC3_SRC_XTAL をコメントアウトしてください。

3.3 DMAコントローラ(DMAC)

このサンプルプロジェクトは、以下に示すような DMAC を使用した様々なデータ転送を行います。

- DMAC を使用したメモリからメモリへのデータ転送
- DMAC を使用した周辺回路からメモリへのデータ転送
- DMAC を使用した周辺回路からメモリへのデータ転送およびメモリから周辺回路へのデータ転送の同時発生

DMAC を使用するデータ転送の他の例については、QSPI_DMA サンプルプログラムを参照してください。

ハードウェアセットアップ

UART-メモリ間の DMA 転送を実行する場合は、UART 用 USB アダプタを使用して S5U1C31D5xT1 評価ボードの UART と PC を接続してください。（図 2.2.2.1、図 2.2.2.2 参照）

動作

例 1: メモリ間 DMA 転送

- RAM 内のデータバッファの内容を、RAM 内の別のバッファにバイト単位で転送するように、DMAC Ch.0 を設定します。
- RAM 内のデータバッファの内容を、RAM 内の別のバッファにハーフワード単位で転送するように、DMAC Ch.1 を設定します。
- RAM 内のデータバッファの内容を、RAM 内の別のバッファにワード単位で転送するように、DMAC Ch.2 を設定します。
- ソフトウェアトリガにより転送を開始します。

この例では、以下の処理を行います。

1. NVIC の DMAC 割り込みを無効にします。
2. 使用する DMAC チャネルの転送を有効にします。
3. 転送元/転送先アドレスのインクリメントを有効にます。
4. 転送は DMACSWREQ レジスタビットのセットにより開始します。
5. 転送終了時に DMA 転送完了割り込み要因が発生します。
6. 割り込み要因発生後、転送回数の読み出し値は 0 になります。
7. 割り込み要因発生後、DMA 転送完了割り込みフラグをクリアします。
8. 転送元と転送先のバッファの内容を比較し、全データが正常に転送されたことをチェックします。

例 2: 周辺回路からメモリへの DMA 転送

- UART データレジスタの内容をメモリに転送するように、DMAC Ch.0 を設定します。
- UART のボーレートを 115,200bps に設定します。また、受信バッファフル要因によって DMA 転送を開始するように設定します。
- PC 上のターミナルプログラムから 8 文字を入力したことをトリガとして、データ転送を開始します。

この例では、以下の処理を行います。

1. NVIC の DMAC 割り込みを無効にします。
2. 使用する DMAC チャネルの転送を有効にします。
3. 使用する DMAC チャネルに対する周辺回路からの DMA 転送要求マスクを解除します。
4. 転送元アドレスのインクリメントを無効にします。
5. 転送先アドレスのインクリメントを有効にします。

6. 転送は UART2 の受信バッファが満杯になることにより開始します。
7. 転送終了時に DMA 転送完了割り込み要因が発生します。
8. 割り込み要因発生後、DMA 転送モードの読み出し値は 0x0(停止)になります。
9. 割り込み要因発生後、DMA 転送完了割り込みフラグをクリアします。
10. メモリから周辺回路への DMA 転送を使用して、ターミナルウィンドウに受信文字列を返送します。
11. ターミナルウィンドウに表示された文字列を見て、正常に転送されたことを確認します。

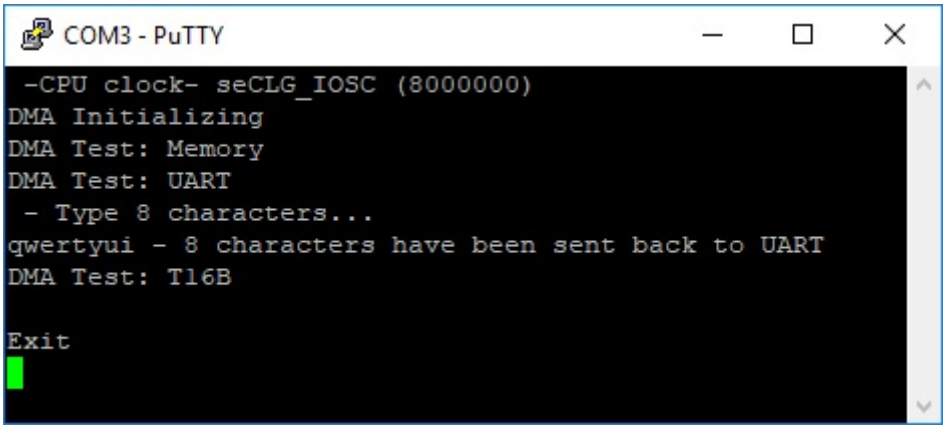
例 3: 周辺回路からメモリへの DMA 転送およびメモリから周辺回路への DMA 転送の同時発生

- T16B Ch.0 のコンパレータ/キャプチャ回路 0 をコンパレータモードに設定します。
- T16B Ch.0 のコンパレータ/キャプチャ回路 1 をキャプチャモードに設定します。
- メモリデータを T16B Ch.0 のコンパレータ/キャプチャ回路 0 データレジスタに転送するように、DMAC Ch.0 を設定します。
- T16B Ch.0 のコンパレータ/キャプチャ回路 1 データレジスタの内容をメモリに転送するように、DMAC Ch.1 を設定します。

この例では、以下の処理を行います。

1. NVIC の DMAC 割り込みを有効にします。
2. 2 つの DMAC チャネルの転送を有効にします。
3. 使用する DMAC チャネルに対する周辺回路からの DMA 転送要求マスクを解除します。
4. メモリから周辺回路への転送は、T16B Ch.0 コンパレータ回路 0 からのコンペア割り込み要因により開始します。
5. 転送終了時に DMAC Ch.0 から DMA 転送完了割り込みが発生します。
6. 割り込み発生後、DMAC 割り込み処理ルーチンの中で、ソフトウェア完了フラグをセットします。
7. DMA 転送完了割り込みフラグをクリアします。
8. 周辺回路からメモリへの転送は、T16B Ch.0 キャプチャ回路 1 からのキャプチャ割り込み要因により開始します。
9. 転送終了時に DMAC Ch.1 から DMA 転送完了割り込みが発生します。
10. 割り込み発生後、DMAC 割り込み処理ルーチンの中で、ソフトウェア完了フラグをセットします。
11. DMA 転送完了割り込みフラグをクリアします。
12. 両方の転送が完了したことをソフトウェア完了フラグで確認します。

出力例



3.4 I2C (I2C_S5U1C31D5xT1)

このサンプルプロジェクトは、I2Cをマスタモードに設定し、I2CシリアルEEPROM（24FC512）に対してリード/ライトを行います。このサンプルプログラムの実行には、別途、24FC512 EEPROMメモリチップが必要です。

ハードウェアセットアップ

以下に示すように、S5U1C31D5xT1 評価ボードと 24FC512 EEPROM メモリチップを接続してください。

表 3.4.1 マスタ-スレーブ間の結線

S1C31D5x [master]			24FC512 [slave]
P17	BOARD_I2C_SDA_PORT	----------><----------	SDA
P16	BOARD_I2C_SCL_PORT	----------->>----------	SCL

BOARD_I2C_SDA_PORT と BOARD_I2C_SCL_PORT は、board.h ファイルに P17（sePPORT_P17）と P16（sePPORT_P16）として定義されています。

図 3.4.1 および図 3.4.2 に、I2C シリアル EEPROM（24FC512）と S5U1C31D5xT1 評価ボードの接続の写真および結線図を示します。

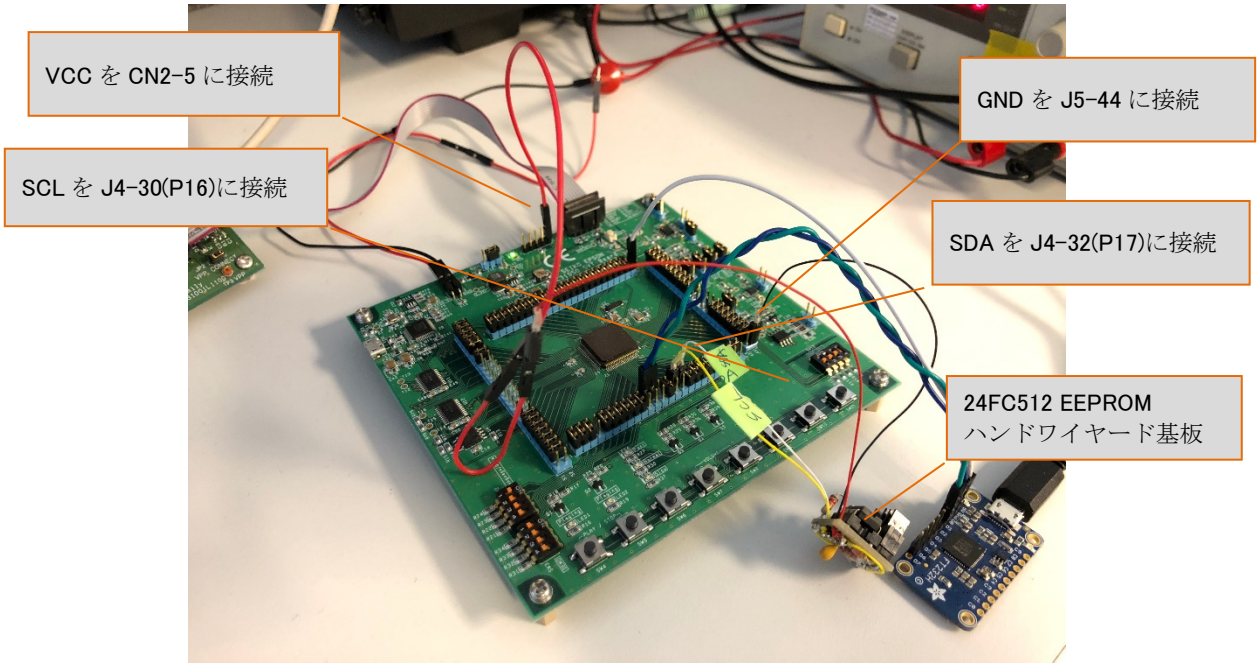


図3.4.1 I2CシリアルEEPROMの評価ボードへの接続

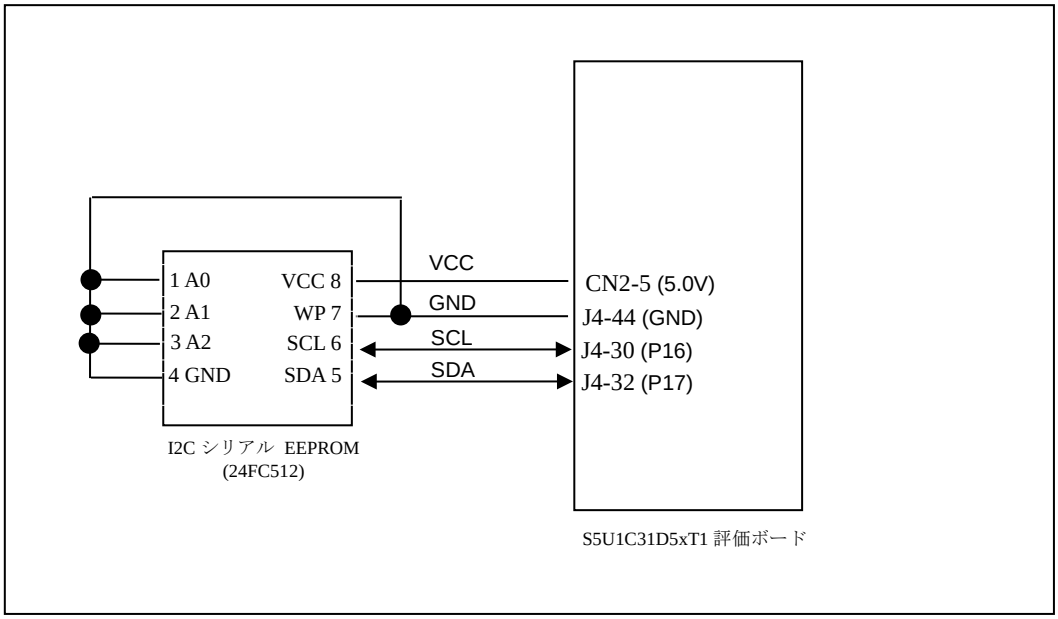


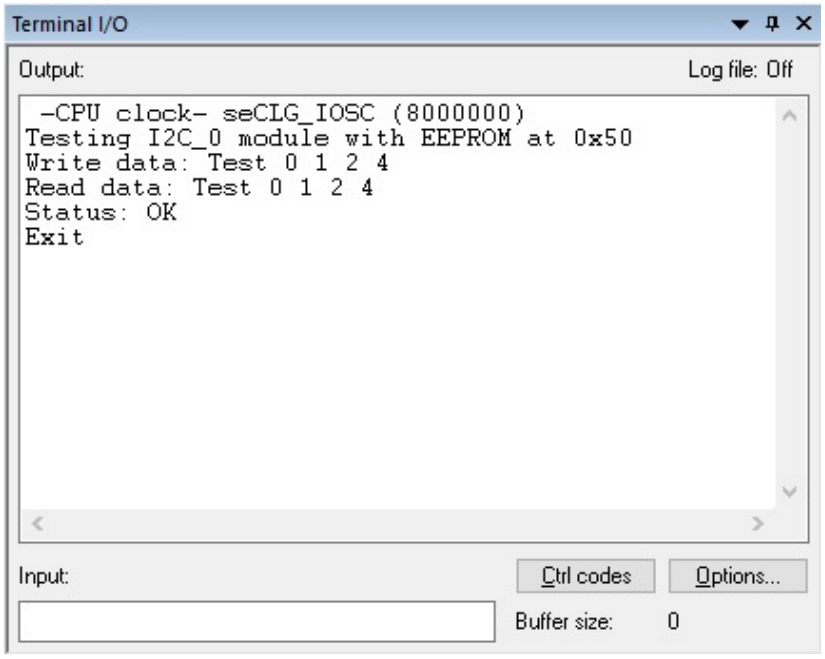
図3.4.2　S5U1C31D5xT1評価ボードとI2CシリアルEEPROM間結線図

動作

このサンプルプロジェクトでは、スレーブアドレス 0x50 で EEPROM を使用して I2C Ch.0 の通信を行います。

1. I2C Ch.0 をマスタモードに設定します。
2. EEPROM に ASCII 文字列を書き込みます。
3. EEPROM からデータを読み出し、書き込んだデータと比較します。

出力例



3.5 入出力ポート(PPORT)

このサンプルプロジェクトは、出力ポートとして設定した BOARD_OUTPORT 端子と、入力ポートとして設定した BOARD_INPORT 端子を接続し、BOARD_OUTPORT 端子と BOARD_INPORT 端子の接続をチェックするための様々な動作を行います。

BOARD_OUTPORT ---------->>--------- BOARD_INPORT

ハードウェアセットアップ

PPORT は TEST_PPORT_01、TEST_PPORT_23、TEST_PPORT_OTHER の 3 つの構成があります。詳細については、borad.h ファイルを参照してください。

デフォルトの TEST_PPORT_OTHER の構成では、以下に示すように J4-48 と J4-50 ジャンパで接続してください。

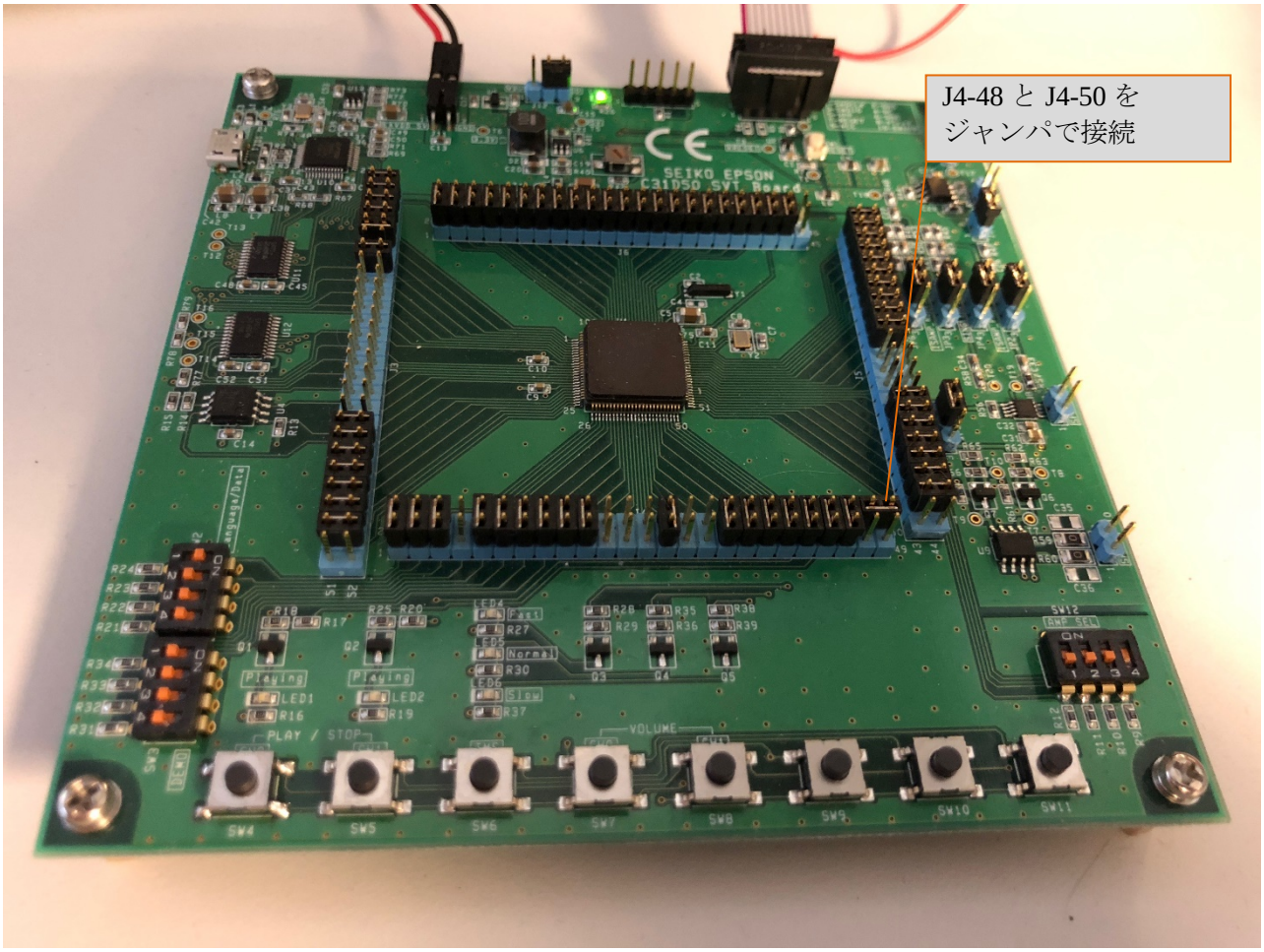


図3.5.1 TEST_PORT_OTHER構成のジャンパ設定

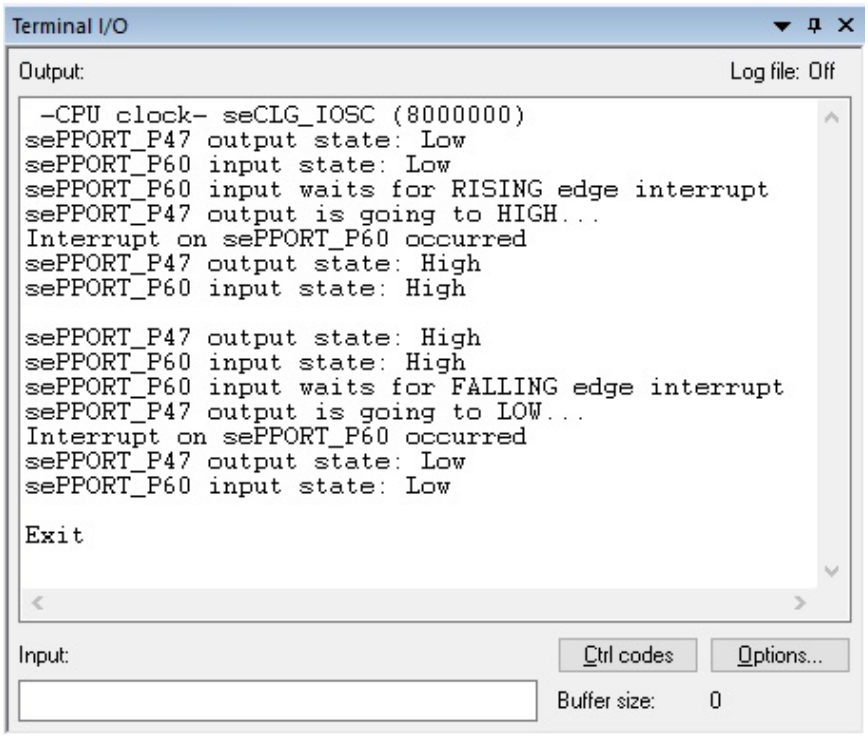
sePPORT_P47 は BOARD_OUTPORT として、sePPORT_P60 は BOARD_INPORT として使用します。他の設定については、board.h を参照して下さい。

動作

1. OSC1 の発振を開始させ、システムクロックを IOSC から OSC1 に切り換えます。その後、IOSC を停止します。
2. 入出力ポートを初期化します。

3. P35 ポートを出力に設定し、LOW レベルを出力させます。
4. P36 ポートを入力に設定すると共に、LOW から HIGH への入力の変化により割り込みが発生するように設定します。
5. P35 ポートから HIGH レベルを出力させます。
6. P36 ポートの割り込みが発生後、P36 ポートの入力が HIGH レベルであることを確認します。
7. P36 ポートを HIGH から LOW への入力の変化により割り込みが発生するように設定します。
8. P35 ポートから LOW レベルを出力させます。
9. P36 ポートの割り込みが発生後、P36 ポートの入力が LOW レベルであることを確認します。

出力例



3.6 同期式クワッドシリアルインタフェース(QSPI)

このサンプルプロジェクトは、QSPI をマスタモードに設定し、外部 QSPI Flash メモリとの通信を行います。

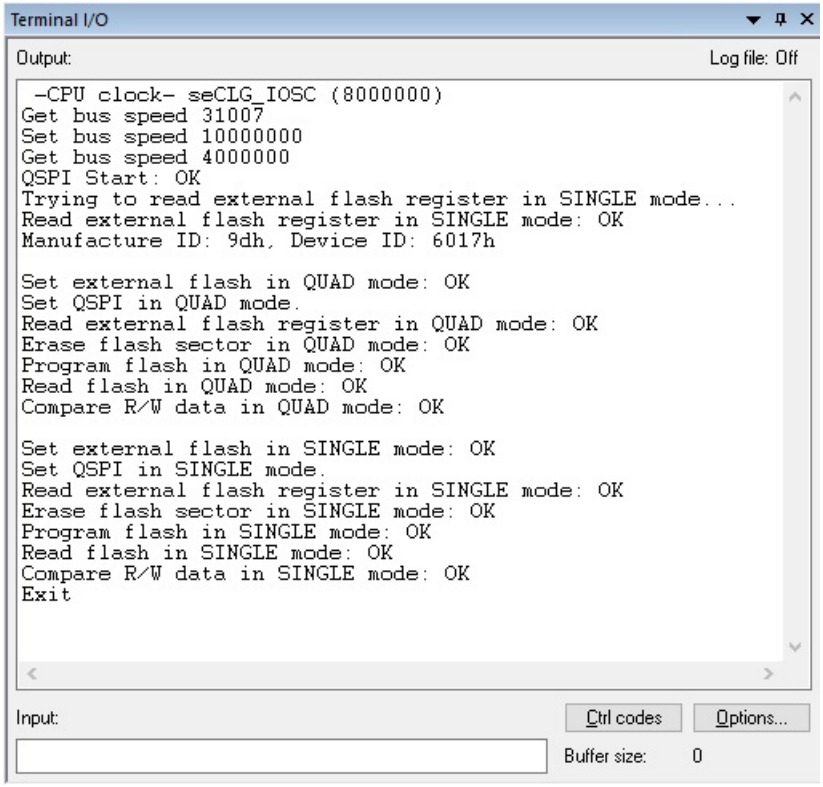
ハードウェアセットアップ

このサンプルプロジェクトでは、S5U1C31D5xT1 評価ボード搭載の ISSI 社製 QSPI Flash メモリを使用します。

動作

1. QSPI をマスタモードおよび以下の設定に初期化します。
   - データ長: 8 ビット
   - データフォーマット: MSB 先頭
   - クロックジェネレータとして 16 ビットタイマ(T16) Ch.2 を使用
2. クロック周波数を 10,000,000bps に設定します。
3. QSPI をシングルモードで起動します。
4. 外部 QSPI Flash メモリのレジスタが正しく読み出せるかチェックします。
5. フラッシュ ID を読み出します。
6. ここまでのフラッシュ動作が正常に終了した場合は、2 つのモードを順次変更し以下の動作を行います。
   - クワッドモードで、セクタ消去、セクタ書き込み、セクタ読み出しと書込みデータとの比較
   - シングルモードで、セクタ消去、セクタ書き込み、セクタ読み出しと書込みデータとの比較

出力例



3.7 同期式クワッドシリアルインタフェース with DMA (QSPI_DMA)

このサンプルプロジェクトは、QSPI をマスタモードに設定し、DMA を使用して外部 QSPI Flash メモリとの通信を行います。

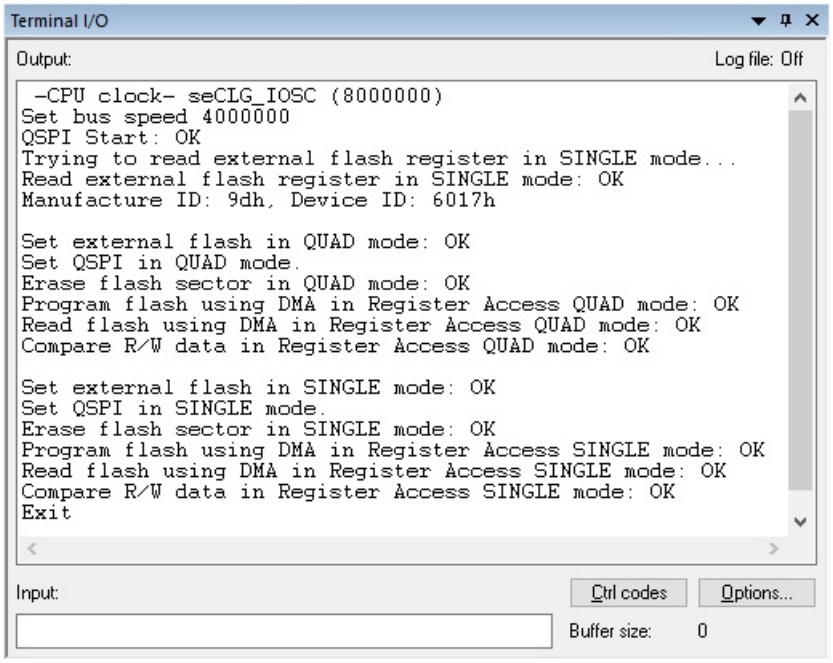
ハードウェアセットアップ

このサンプルプロジェクトでは、S5U1C31D5xT1 評価ボード搭載の ISSI 社製 QSPI Flash メモリを使用します。

動作

1. QSPI をマスタモードおよび以下の設定に初期化します。
   - データ長: 8 ビット
   - データフォーマット: MSB 先頭
   - クロックジェネレータとして 16 ビットタイマ(T16) Ch.2 を使用
2. クロック周波数を 4,000,000bps に設定します。
3. DMA 転送用のデータストラクチャを作成し、DMAC を初期化します。
4. QSPI をレジスタアクセスシングルモードで起動します。
5. 外部 QSPI メモリのレジスタが正しく読み出せるかチェックします。
6. フラッシュ ID を読み出します。
7. ここまでのフラッシュ動作が正常に終了した場合は、クワッドモードおよびシングルモードで以下の動作を行います。
   - セクタ消去
   - レジスタアクセスによる DMA 転送を使用したセクタ書き込み
   - レジスタアクセスによる DMA 転送を使用したセクタ読み出しと書込みデータとの比較

出力例



3.8 同期式クワッドシリアルインタフェース マスタ(QSPI_MASTER)

このサンプルプロジェクトは、QSPI をマスタモードに設定して通信を行います。

ハードウェアセットアップ

1. S5U1C31D5xT1 評価ボードに搭載されている QSPI Flash メモリは、S1C31D5x の QSPI インタフェースに接続されています。そのため、QSPI マスタ/スレーブのサンプルプログラムを実行するには、評価ボード上の J3-24, J3-26, J3-28, J3-30, J3-32, J3-34, J3-36 のジャンパピンを外して、QSPI Flash メモリを QSPI インタフェースから分離します。
2. QSPI マスタと QSPI スレーブのサンプルプログラムをそれぞれインストールした 2 枚の S5U1C31D5xT1 評価ボードを用意し、表 3.8.1 または表 3.8.2 に示す端子間を接続します。
   注： シングルモードで実行する場合は、settings.h の QSPI_MODE_SINGLE のコメントアウトを解除してください。
3. 最初にスレーブ用サンプルプログラムを起動し、その後にマスタ用サンプルプログラムを起動してください。

表 3.8.1 マスタ-スレーブ間の結線（デュアル／クワッドモード）

デュアルモード／クワッドモード					
[マスタ]				[スレーブ]	
J3-34	QSDIOn3	----------><----------	QSDIOn3		J3-34
J3-32	QSDIOn2	----------><----------	QSDIOn2		J3-32
J3-30	QSDIOn1	----------><----------	QSDIOn1		J3-30
J3-28	QSDIOn0	----------><----------	QSDIOn0		J3-28
J3-26	QSPICLKn	----------->>----------	QSPICLKn		J3-26
J3-36	#QSPISSn	----------->>----------	#QSPISSn		J3-36
J3-24	BOARD_QSPI_HANDSHAKE_PORT	----------->>----------	BOARD_QSPI_HANDSHAKE_PORT		J3-24
J3-52	GND	----------><----------	GND		J3-52

* J3の詳細については、S5U1C31D5xT1マニュアル記載の回路図を参照してください。

表 3.8.2 マスタ-スレーブ間の結線（シングルモード）

シングルモード				
[マスタ]			[スレーブ]	
J3-28	QSDIOn0	----------->>----------	QSDIOn1	J3-30
J3-30	QSDIOn1	-----------<<----------	QSDIOn0	J3-28
J3-26	QSPICLKn	----------->>----------	QSPICLKn	J3-26
J3-36	#QSPISSn	----------->>----------	#QSPISSn	J3-36
J3-24	BOARD_QSPI_HANDSHAKE_PORT	----------->>----------	BOARD_QSPI_HANDSHAKE_PORT	J3-24
J3-52	GND	-----------><----------	GND	J3-52

* J3の詳細については、S5U1C31D5xT1マニュアル記載の回路図を参照してください。

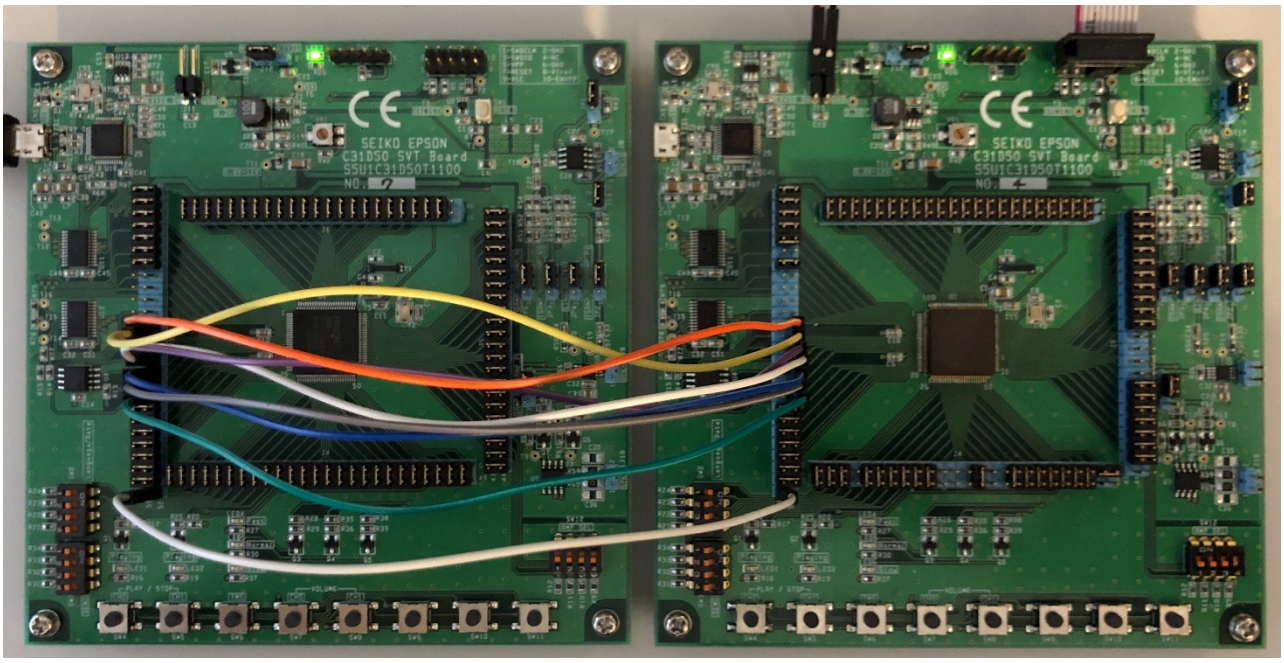


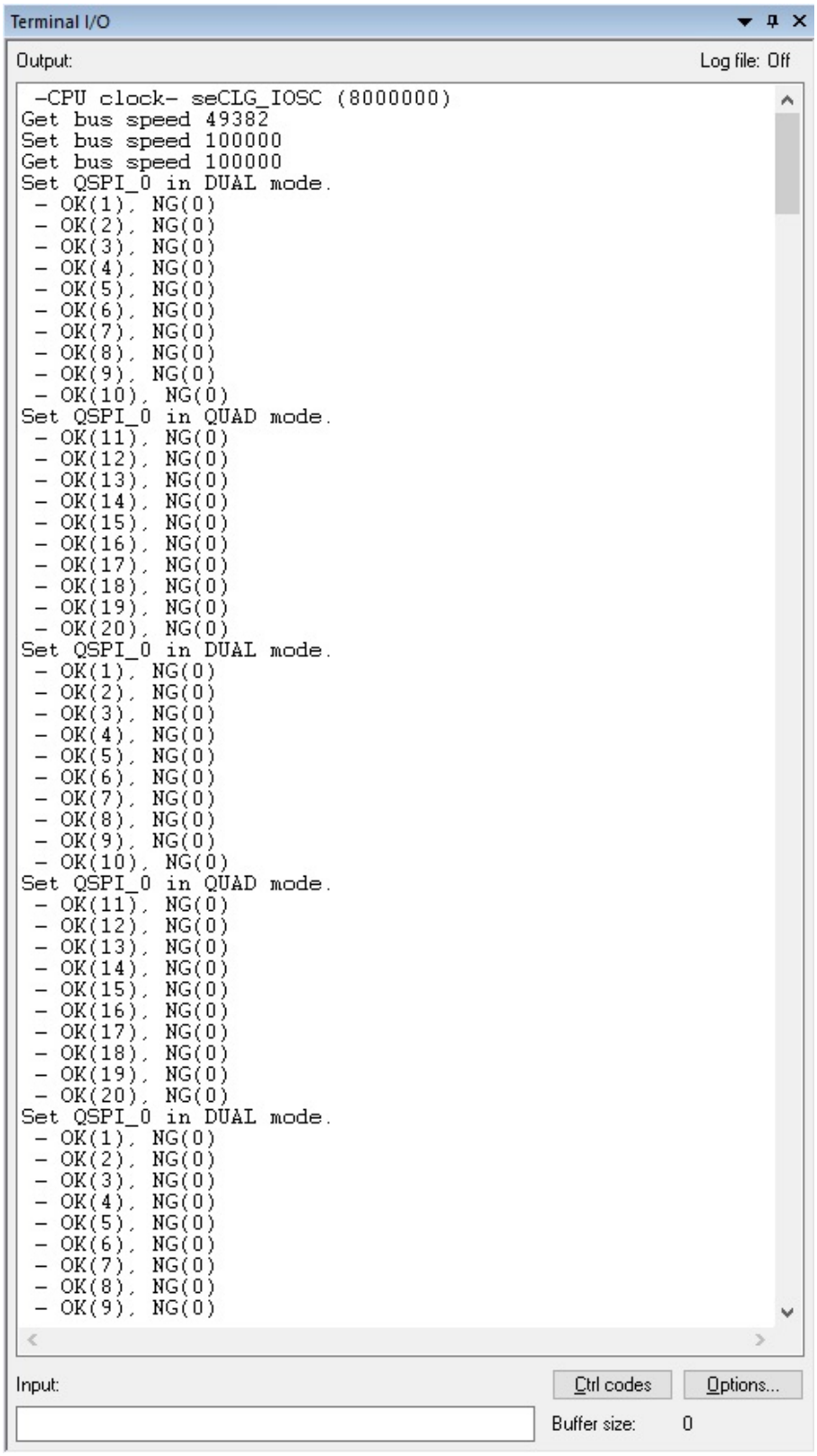
Figure 3.8.1 評価ボード間の接続（デュアル／クワッドモード）

動作

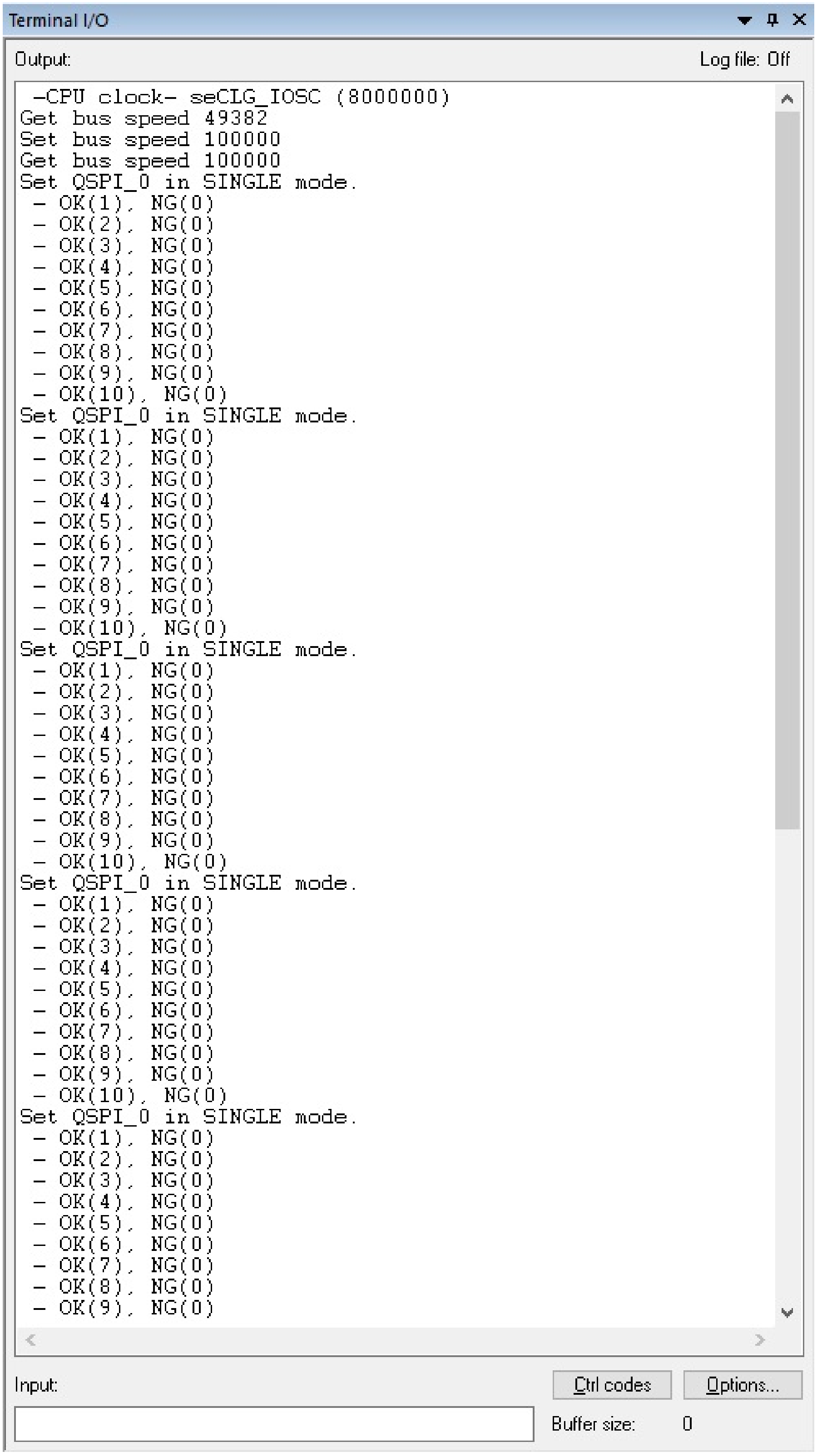
1. QSPI をマスタモードで以下の設定に初期化します。
   - データ長: 8 ビット
   - データフォーマット: MSB 先頭
   - クロックジェネレータとして 16 ビットタイマ(T16) Ch.2 を使用
2. クロック周波数を 300,000bps に設定します。
3. スレーブへの Read/Write コマンドを設定するために BOARD_QSPI_HANDSHAKE_PORT を汎用出力に設定します。
4. “BUF_SIZE”バイトのランダムデータをスレーブに送信します。
5. スレーブから送られる“BUF_SIZE”バイトのデータを受信します。
6. 受信データと送信データを比較して終了します。

出力例

デュアル/クワッドモード：



シングルモード：


Terminal I/O
Output:
Log file: Off
-CPU clock- seCLG_IOSC (8000000)
Get bus speed 49382
Set bus speed 100000
Get bus speed 100000
Set QSPI_0 in SINGLE mode.
- OK(1), NG(0)
- OK(2), NG(0)
- OK(3), NG(0)
- OK(4), NG(0)
- OK(5), NG(0)
- OK(6), NG(0)
- OK(7), NG(0)
- OK(8), NG(0)
- OK(9), NG(0)
- OK(10), NG(0)
Set QSPI_0 in SINGLE mode.
- OK(1), NG(0)
- OK(2), NG(0)
- OK(3), NG(0)
- OK(4), NG(0)
- OK(5), NG(0)
- OK(6), NG(0)
- OK(7), NG(0)
- OK(8), NG(0)
- OK(9), NG(0)
- OK(10), NG(0)
Set QSPI_0 in SINGLE mode.
- OK(1), NG(0)
- OK(2), NG(0)
- OK(3), NG(0)
- OK(4), NG(0)
- OK(5), NG(0)
- OK(6), NG(0)
- OK(7), NG(0)
- OK(8), NG(0)
- OK(9), NG(0)
- OK(10), NG(0)
Set QSPI_0 in SINGLE mode.
- OK(1), NG(0)
- OK(2), NG(0)
- OK(3), NG(0)
- OK(4), NG(0)
- OK(5), NG(0)
- OK(6), NG(0)
- OK(7), NG(0)
- OK(8), NG(0)
- OK(9), NG(0)
- OK(10), NG(0)
Set QSPI_0 in SINGLE mode.
- OK(1), NG(0)
- OK(2), NG(0)
- OK(3), NG(0)
- OK(4), NG(0)
- OK(5), NG(0)
- OK(6), NG(0)
- OK(7), NG(0)
- OK(8), NG(0)
- OK(9), NG(0)
Input:
Ctrl codes
Options...
Buffer size:
0

3.9 同期式クワッドシリアルインタフェース スレーブ(QSPI_SLAVE)

このサンプルプログラムは、QSPI をスレーブモードに設定して通信を行います。

ハードウェアセットアップ

「3.8 同期式クワッドシリアルインタフェース マスタ(QSPI_MASTER) 」のハードウェアセットアップを参照ください。

動作

1. QSPI をスレーブモードおよび以下の設定に初期化します。
   - データ長: 8 ビット
   - データフォーマット: MSB 先頭
2. BOARD_QSPI_HANDSHAKE_PORT を汎用入力に設定し、通信開始検出用に立ち上がりエッジでの割り込みを許可します。
3. マスタから送信される“BUF_SIZE”バイトのランダムデータを受信します。
4. “BUF_SIZE”バイトのデータをマスタに送信します。

出力例

なし

3.10 IRリモートコントローラ(REMC3)

このサンプルプログラムは、REMC3 から様々な IR パルスを出力します。

ハードウェアセットアップ

このサンプルプログラムは、別途、赤外線 LED が必要です。赤外線 LED の接続については、S5U1C31D5xT1 のマニュアルを参照してください。

動作

このサンプルプログラムは、NEC フォーマットのリモコン信号を出力します。

1. REMC3 は、APLEN(データ信号のデューティ)と DBLEN(データ信号の周期)の 2 つのパラメータを使用してデータの波形を決定します。APLEN と DBLEN はそれぞれ 34 個のフレーム信号と 2 個のリピート信号を定義します。初期化の際に、この APLEN と DBLEN の 2 つによりリーダー、カスタムコード、リピート信号が設定されます。これらの信号は SW100～SW103 の押下により選択される全ての動作に共通です。データは SW100～SW103 の選択により変化し、スイッチ押下による割り込み発生時に設定されます。
2. 初期化処理では、更に SW100～SW103 の設定、REMO 出力のポートの割り当て、T16 Ch.0 の初期設定が行われます。この後、CPU は HALT モードになり、SW100～SW103 の押下による割り込みの発生を待ちます。
3. 割り込みが発生すると、データ信号を生成するための APLEN と DBLEN の値が設定されます。REMC2 はコンペアバッファを使用するため、初期化終了後に起動させます。また、T16 Ch.0 も起動させます。T16 Ch.0 は約 108ms 後に割り込みを発生します。
4. REMC2 からコンペア DB 割り込みが発生した時点で、続く APLEN と DBLEN の値をレジスタに書き込みます。最後の 1 つ前のデータを書き込む際は、REMC2 をワンショットモードに変更し、ひげ状の信号が出力されることを防ぎます。最後のデータ出力の後、REMC2 の動作を停止します。
5. T16 Ch.0 割り込みが発生した時点で、SW100～SW103 の同じスイッチが押されていないかチェックします。同じスイッチが押されていた場合は、REMC2 のレジスタ値をリピート波形が生成されるように設定し、REMC2 を初期化の後、再度起動します。同じスイッチが押されていなかった場合は、T16 Ch.0 の動作を停止します。

3.11 R/F変換器(RFC)

このサンプルプログラムは、RFC を動作させて測定を行います。

ハードウェアセットアップ

このサンプルプログラムを実行する前に、J5 コネクタに以下の部品が接続されていることを確認してください。

J5-25	SENA0	----------抵抗温度センサ(DC バイアス) ----------	RFIN0	J5-21
J5-23	REF0	----------基準抵抗----------	RFIN0	J5-21
J5-21	RFIN0	----------基準抵抗および基準キャパシタ----------	GND	J5-43

RFC は以下に示すポートを使用します。

```
#define sePPORT_RFC_RFCLKO0     sePPORT_P30
#define sePPORT_RFC_SENB0       sePPORT_P20
#define sePPORT_RFC_SENA0       sePPORT_P21
#define sePPORT_RFC_REF0        sePPORT_P22
#define sePPORT_RFC_RFIN0       sePPORT_P23
```

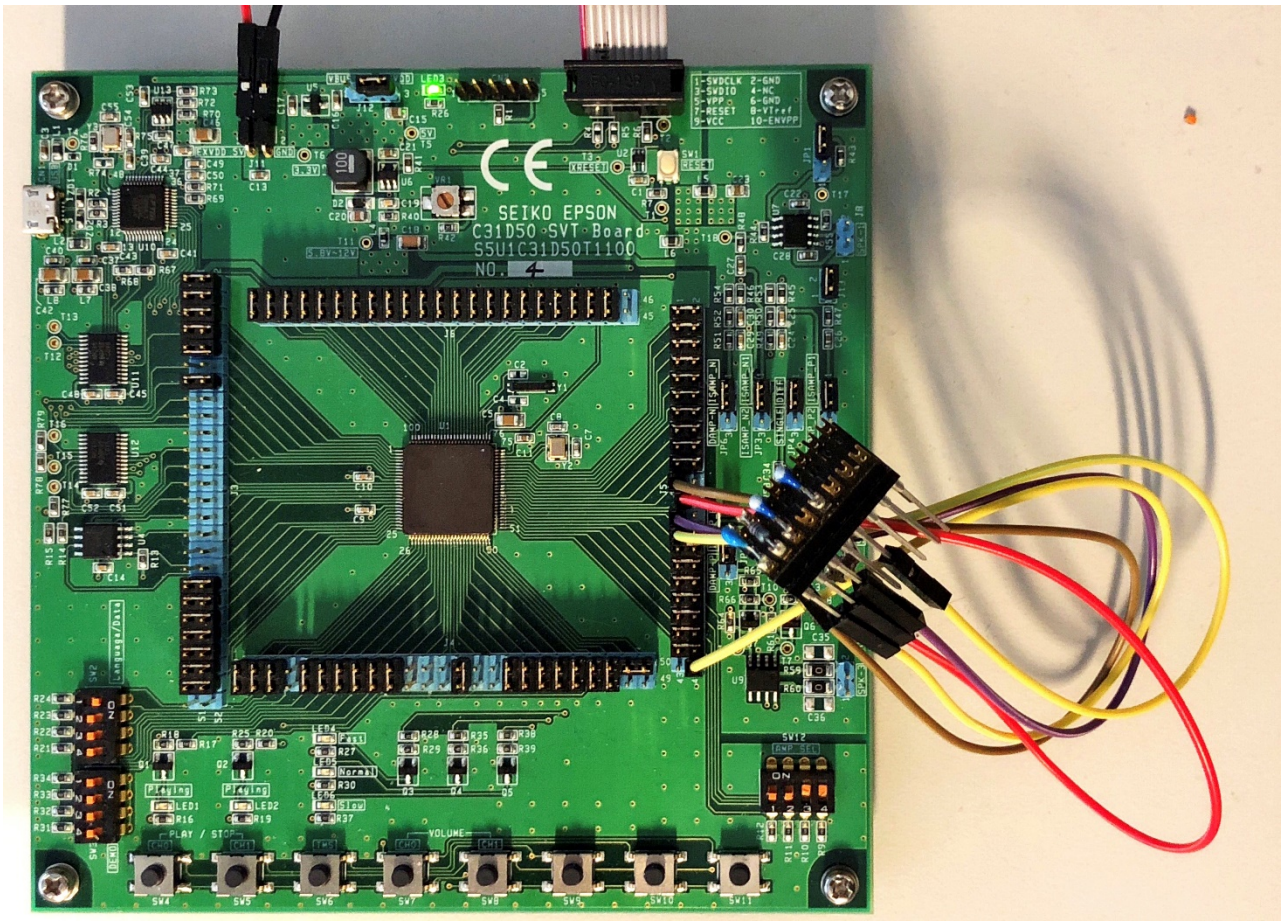
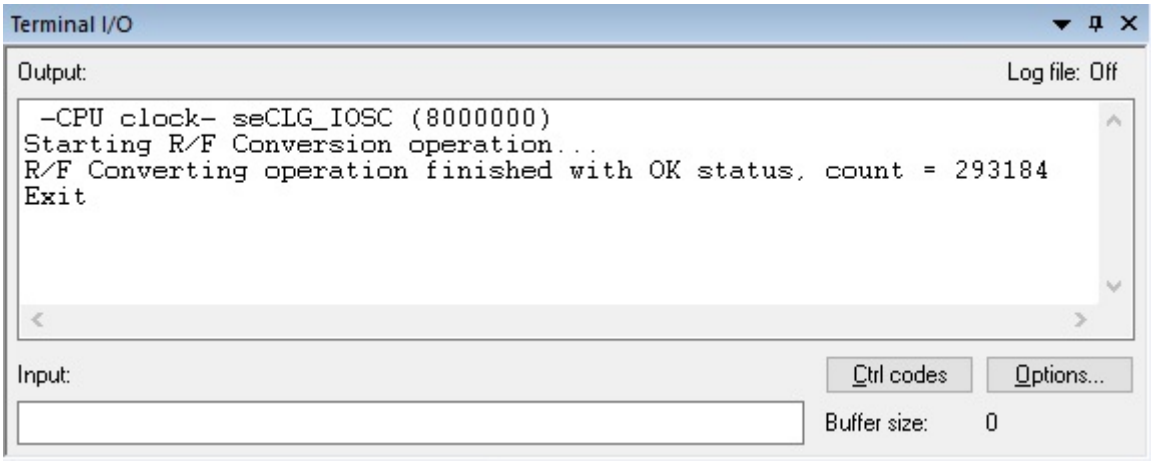


Figure 3.11.1 RFC 接続部品の評価ボードへの接続

動作

1. RFC を初期化します。
2. RFC を動作させ、計測カウンタの値を取得します。
3. 変換の状況とカウンタ値を表示します。

出力例



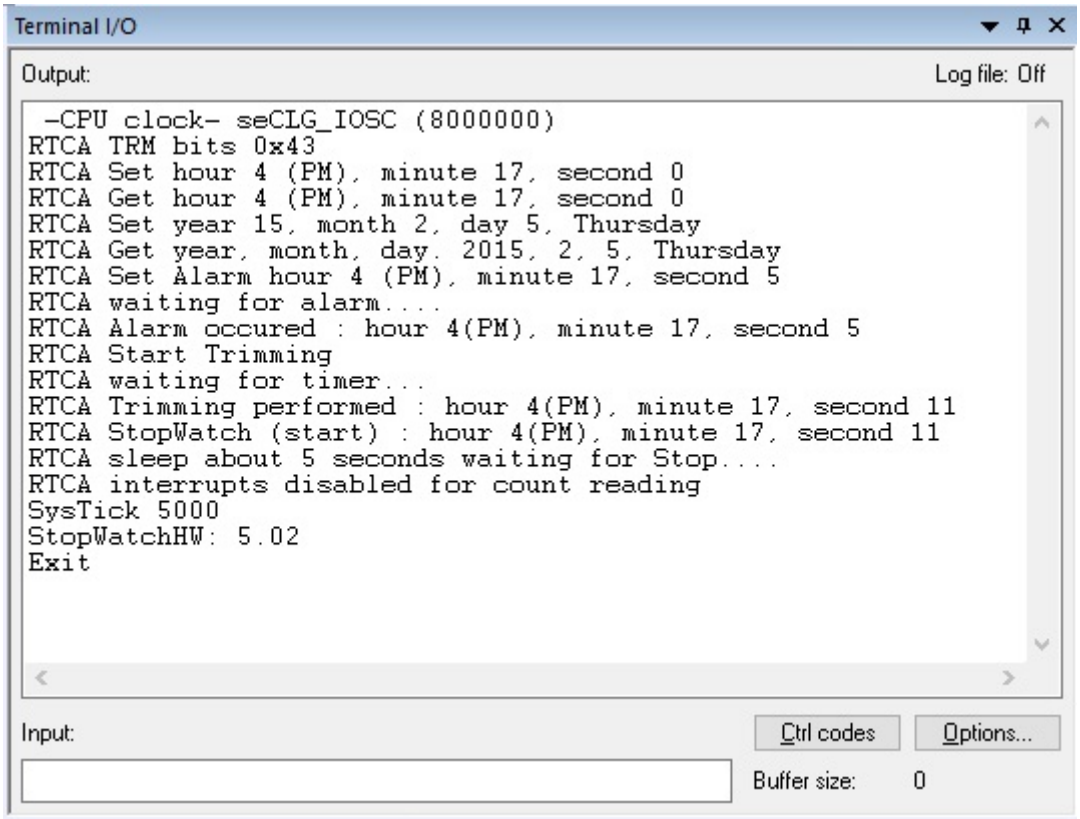
3.12 リアルタイムクロック(RTCA)

このサンプルプログラムは、時刻/日付設定、ストップウォッチ、アラーム、トリミングなどのリアルタイムクロックの様々な機能を実行します。

動作

1. RTCA を初期化します。
2. RTCA をスタートします。
3. 論理緩急補正値を算出します。
4. 日時を設定し、その値を読み出します。
5. アラームを設定し、CPU を SLEEP 状態にします。アラーム割り込みにより SLEEP 状態が解除されます。
6. 1 秒タイマをスタートさせ、論理緩急を実行します。
7. ストップウォッチの動作を確認します。

出力例



3.13 同期式シリアルインタフェース マスタ(SPIA_MASTER)

このサンプルプログラムは、SPIA をマスタモードに設定してデータ通信を行います。

ハードウェアセットアップ

1. SPIA マスタと SPIA スレーブのサンプルプログラムをそれぞれインストールした 2 枚の S5U1C31D5xT1 評価ボードを用意して、表 3.13.1 に示す端子間を接続します。
2. 最初にスレーブ用サンプルプログラムを起動し、その後にマスタ用サンプルプログラムを起動してください。

表 3.13.1 マスタ-スレーブ間の結線

[マスタ]				[スレーブ]		
J3-18	BOARD_SPI_SDI_PORT	SDIn	----------<<----------	SDOn	BOARD_SPI_SDO_PORT	J3-20
J3-20	BOARD_SPI_SDO_PORT	SDOn	---------->>----------	SDIn	BOARD_SPI_SDI_PORT	J3-18
J3-22	BOARD_SPI_CLK_PORT	SPICLKn	---------->>----------	SPICLKn	BOARD_SPI_CLK_PORT	J3-22
J3-16	BOARD_MASTER_CS_PORT		---------->>----------	#SPISSn	BOARD_SPI_SS_PORT	J3-16
J3-24	BOARD_SPI_HANDSHAKE_PORT		---------->>----------	BOARD_SPI_HANDSHAKE_PORT		J3-24
J3-52		GND	----------><----------	GND		J3-52

SPIA は UPMUX を介して設定された GPIO を使用します。

```
#define BOARD_SPI_SS_PORT             sePPORT_P33
#define BOARD_SPI_SDI_PORT            sePPORT_P34
#define BOARD_SPI_SDO_PORT            sePPORT_P35
#define BOARD_SPI_CLK_PORT            sePPORT_P36
#define BOARD_SPI_HANDSHAKE_PORT      sePPORT_P37
#define BOARD_MASTER_CS_PORT          BOARD_SPI_SS_PORT
```

SPIA_MASTER, SPIA_SLAVE サンプルプログラムで使用する特定のポートについては、*board.h* ファイルを参照ください。

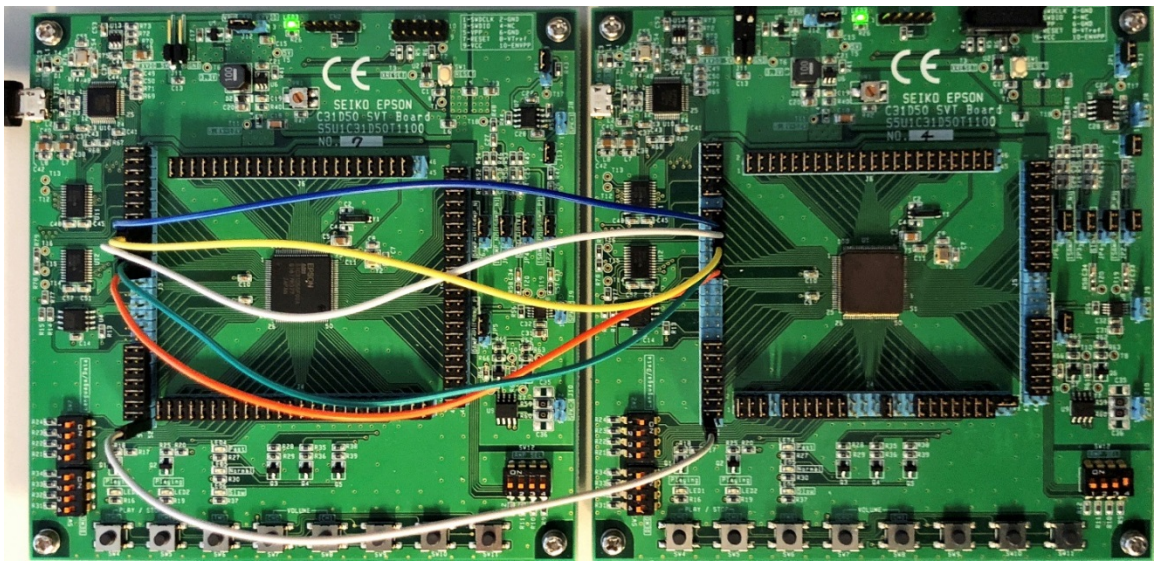


図 3.13.1 評価ボード間の接続

動作

1. SPIA をマスタモードおよび以下の設定に初期化します。
   - データ長: 8 ビット
   - データフォーマット: MSB 先頭
   - クロックジェネレータとして 16 ビットタイマ(T16) Ch.1 を使用
2. クロックを 1,000,000 に設定します。
3. スレーブへの Read/Write コマンドを設定するために BOARD_SPI_HANDSHAKE_PORT を汎用出力に設定します。
4. “BUF_SIZE”バイトのランダムデータをスレーブに送信します。
5. スレーブから送信される“BUF_SIZE”バイトのデータを受信します。
6. 受信データと送信データを比較して終了します。

出力例

3.14 同期式シリアルインタフェース スレーブ(SPIA_SLAVE)

このサンプルプログラムは、SPIA をスレーブモードに設定してデータ通信を行います。

ハードウェアセットアップ

「3.14 同期式シリアルインタフェース マスタ(SPI_MASTER) 」のハードウェアセットアップを参照ください。

動作

1. SPIA をスレーブモードおよび以下の設定に初期化します。
   - データ長: 8 ビット
   - データフォーマット: MSB 先頭
2. マスタからの Read/Write コマンドを認識するために BOARD_SPI_HANDSHAKE_PORT を汎用入力に設定します。
3. マスタからの“BUF_SIZE”バイトのランダムデータを受信します。
4. “BUF_SIZE”バイトのデータをマスタに送信します。

出力例

なし

3.15 電源電圧検出回路(SVD3)

このサンプルプログラムは、電源電圧検出回路（SVD3）を使用して VDD 電圧低下の検出を行います。

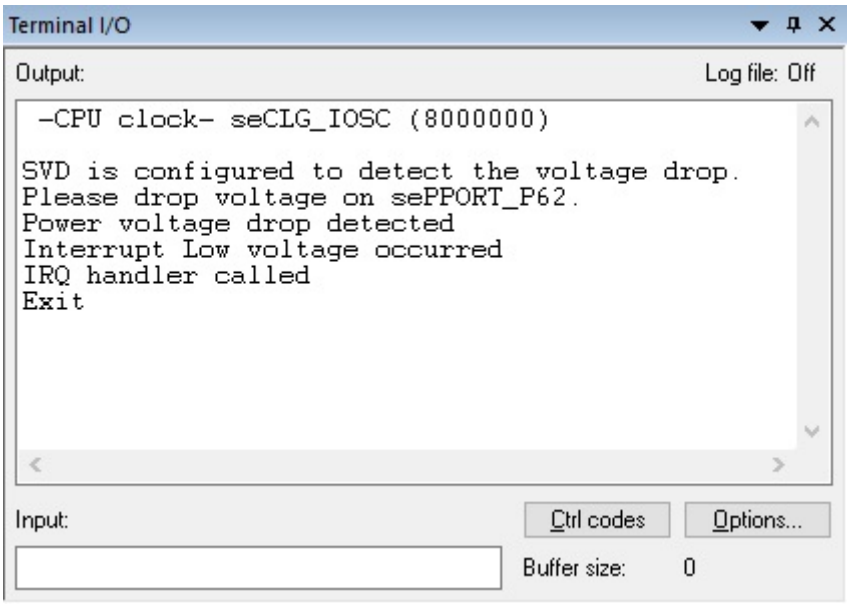
ハードウェアセットアップ

このサンプルプログラムを実行するには、デバッグプローブから S5U1C31D5xT1 評価ボードに電源供給する必要があります。また、S5U1C31D5xT1 評価ボード上の J12 の EXVDD 側にジャンパを設定してください。

動作

1. プログラムが V_{DD} の電圧低下を監視するための設定を行います。
2. J12 からジャンパを外します。
3. プログラムが電圧低下を検出し、その結果が表示されます。

出力例



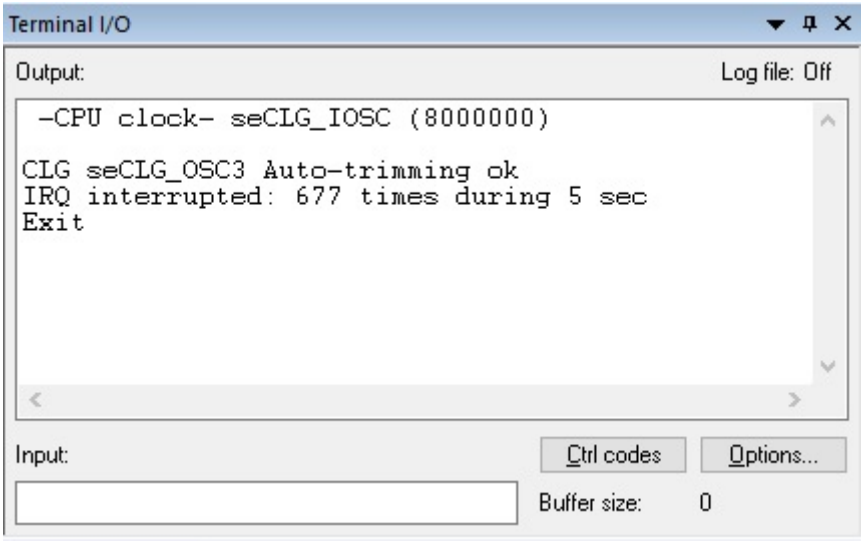
3.16 16 ビットタイマ(T16)

このサンプルプログラムは、16 ビットタイマ（T16）を定期的に中断して 5 秒間隔で発生した割り込みの数をカウントします。

動作

1. OSC3 をタイマのクロックソースとして選択します。このサンプルプログラムでは、タイマクロックの精度を確保するため、OSC3 のオートトリミングを有効にしています。
2. T16 Ch.0 を初期化します。
3. T16 Ch.0 割り込みを設定します。
4. T16 割り込みをイネーブルにします。
5. 5 秒間に発生する割り込みの回数をカウントします。

出力例



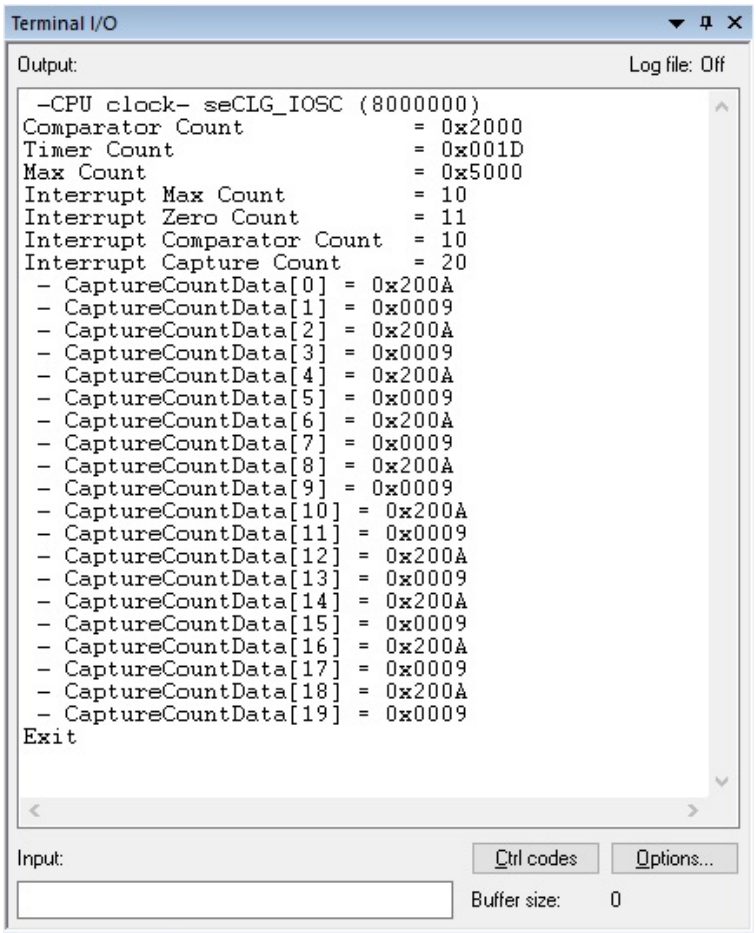
3.17 16ビットPWMタイマ(T16B)

このサンプルプログラムは、16 ビット PWM タイマ（T16B）の様々なコンパレータ/キャプチャ機能を実行します。

動作

1. T16B を以下のとおり設定し、タイマをスタートさせます。
   - モード: リピートアップカウントモード
   - MAX カウンタ値: 0x5000
   - コンパレータ/キャプチャ回路 0: コンパレータモード(コンペアバッファ: 0x2000)
   - コンパレータ/キャプチャ回路 1: キャプチャモード、割り込み回数設定(トリガ信号: LOW)
2. T16B をスタートさせ、CPU を HALT 状態にします。
3. T16B 割り込みにより CPU の HALT 状態が解除されます。
4. T16B 割り込みが発生するごとに、以下の割り込み回数をカウントします。
   - コンペア割り込み回数
   - キャプチャ割り込み回数
   - MAXカウンタ割り込み回数
   - カウンタゼロ割り込み回数
5. コンペア割り込みが発生した時点でキャプチャトリガ信号を HIGH に設定し直します。
6. MAX カウンタ割り込みが発生した時点でキャプチャトリガ信号を LOW に設定し直します。

出力例



3.18 UART(UART3)

このサンプルプログラムは、PC と S5U1C31D5xT1 評価ボードの間で UART 通信を行います。

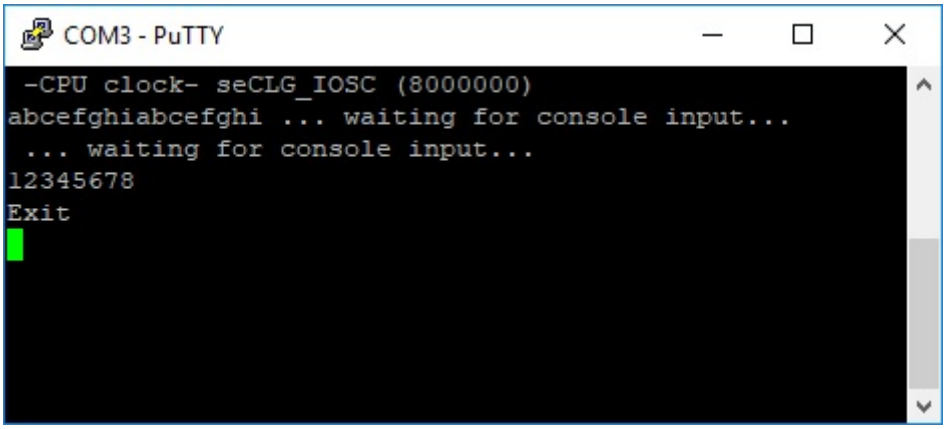
ハードウェアセットアップ

1. UART 用 USB アダプタを介して S5U1C31D5xT1 評価ボードを PC に接続します（詳細は 2.2.2 節、図 2.2.2.1、図 2.2.2.2 を参照）。
2. PC 上で UART コンソールとして動作するターミナルプログラムを起動します。
3. ターミナルプログラム上でシリアルポートの設定を行います（表 2.2.2.1）。

動作

1. UART3 を以下の設定で初期化します。
   - ボーレート: 115200 bps
   - データ: 8 ビット
   - ストップビット: 1 ビット
   - パリティ: なし
2. UART3 から ASCII 文字列を送信します。送信された文字列はターミナルプログラムのウィンドウに表示されます。
3. ターミナルプログラムからの受信に待機します。ターミナルプログラムで 8 文字を入力してください。
4. 入力された文字列をターミナルプログラムに送り返します。

出力例



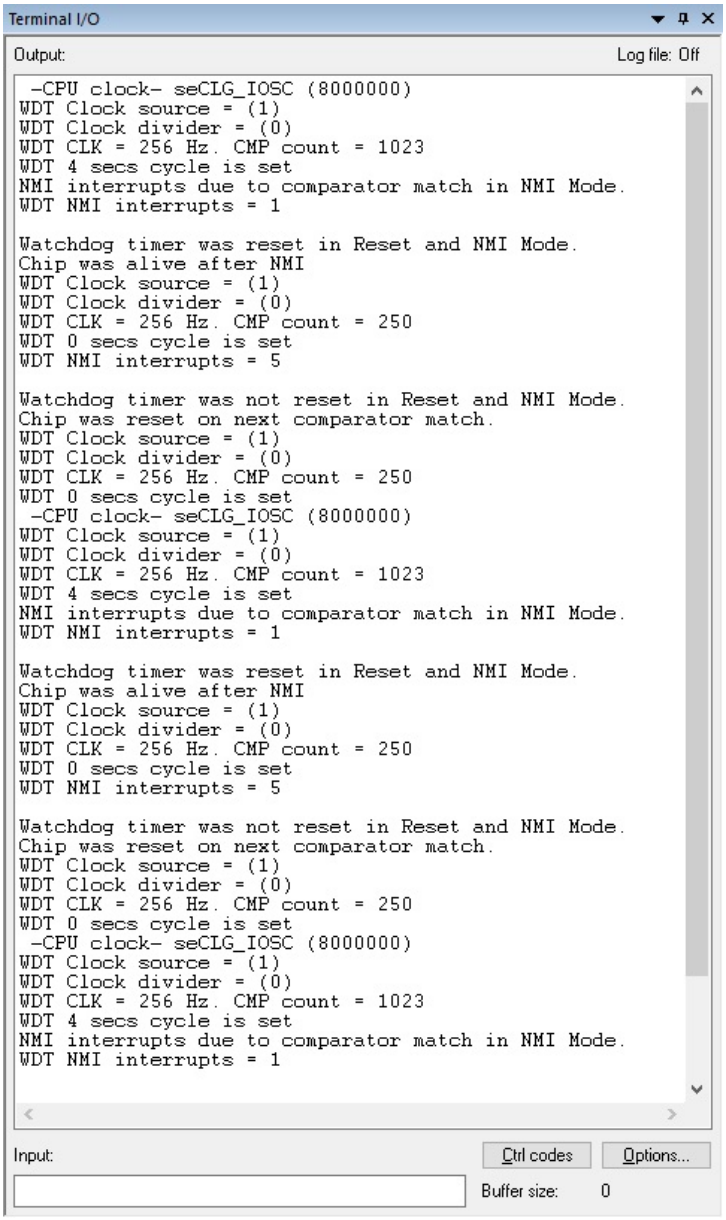
3.19 ウォッチドッグタイマ(WDT2)

このサンプルプログラムは、WDT2 を使用して NMI 割り込みとリセットの生成を行います。

動作

1. WDT2 を初期化します。
2. OSC1 を WDT2 のクロックソースとして設定し、OSC を起動してスリープモードで動作するように設定します。
3. NMI モードに設定し、NMI/リセット発生周期の経過による NMI を発生させます。
4. RESET after NMI モードに設定し、WDT をリセットして WDT リセットが発生しないことを確認します。
5. RESET after NMI モードに設定し、WDT をリセットせずに WDT リセットを発生させます。

出力例



3.20 音声再生 (SOUNDPLAY)

このサンプルプログラムは、HW Processor で提供される音声再生機能を使用して音声の再生を行います。以下の 3 通りの音声再生機能が実現できます。

・オーディオアンプ IC を使用したスピーカーによる音声再生

・ディスクリート回路を使用した電磁ブザーによる音声再生 (S1C31D51 のみ)

・ディスクリート回路を使用した圧電ブザーによる音声再生 (S1C31D51 のみ)

表 3.20.1 に示すボード定義により、本サンプルプログラムの機能を切り替えることができます。

表3.20.1 音声再生(SOUNDPLAY)用ボード定義

設定用定数	定義時
PLAY_AMPSPEAKER	オーディオアンプICを使用したスピーカーによる音声再生を実現します。
PLAY_ELBUZZER	ディスクリート回路を使用した電磁ブザーによる音声再生 (S1C31D51のみ)
PLAY_PIBUZZER	ディスクリート回路を使用した圧電ブザーによる音声再生 (S1C31D51のみ)

(1) オーディオアンプ IC を使用したスピーカーによる音声再生

サンプルソフトウエアセットアップ

1. SOUNDPLAY プロジェクトのシンボル定義に”PLAY_AMPSEAKER”を定義する。(設定方法は図 3.20.1、表 3.20.1 参照)
2. SOUNDPLAY プロジェクトをビルドする。

ハードウェアセットアップ

1. 評価ボード(S5U1C31D5xT1)のオーディオアンプのジャンパ設定を確認します(表 3.20.2、図 3.20.2 参照)。
2. 付属のスピーカーケーブルを介して、スピーカー接続コネクタにスピーカーを接続します（図 3.20.2 参照)。

注 : 電源供給中にオーディオアンプの設定を行うと、オーディオアンプなどの実装部品が破損する可能性があります。電源 OFF の状態で設定を行ってください。

表 3.20.2 オーディオアンプのジャンパ設定

コネクタ: JP1/JP2/JP3/J13/JP4/J5						
アンプ種類	JP1	JP2	JP3	J13	JP4	J5
AB 級	2-3 ショート	1-2 ショート	1-2 ショート	1-2 ショート	-	9-10 ショート 11-12 ショート
D 級	-	2-3 ショート	2-3 ショート	-	1-2 ショート	9-10 ショート 11-12 ショート

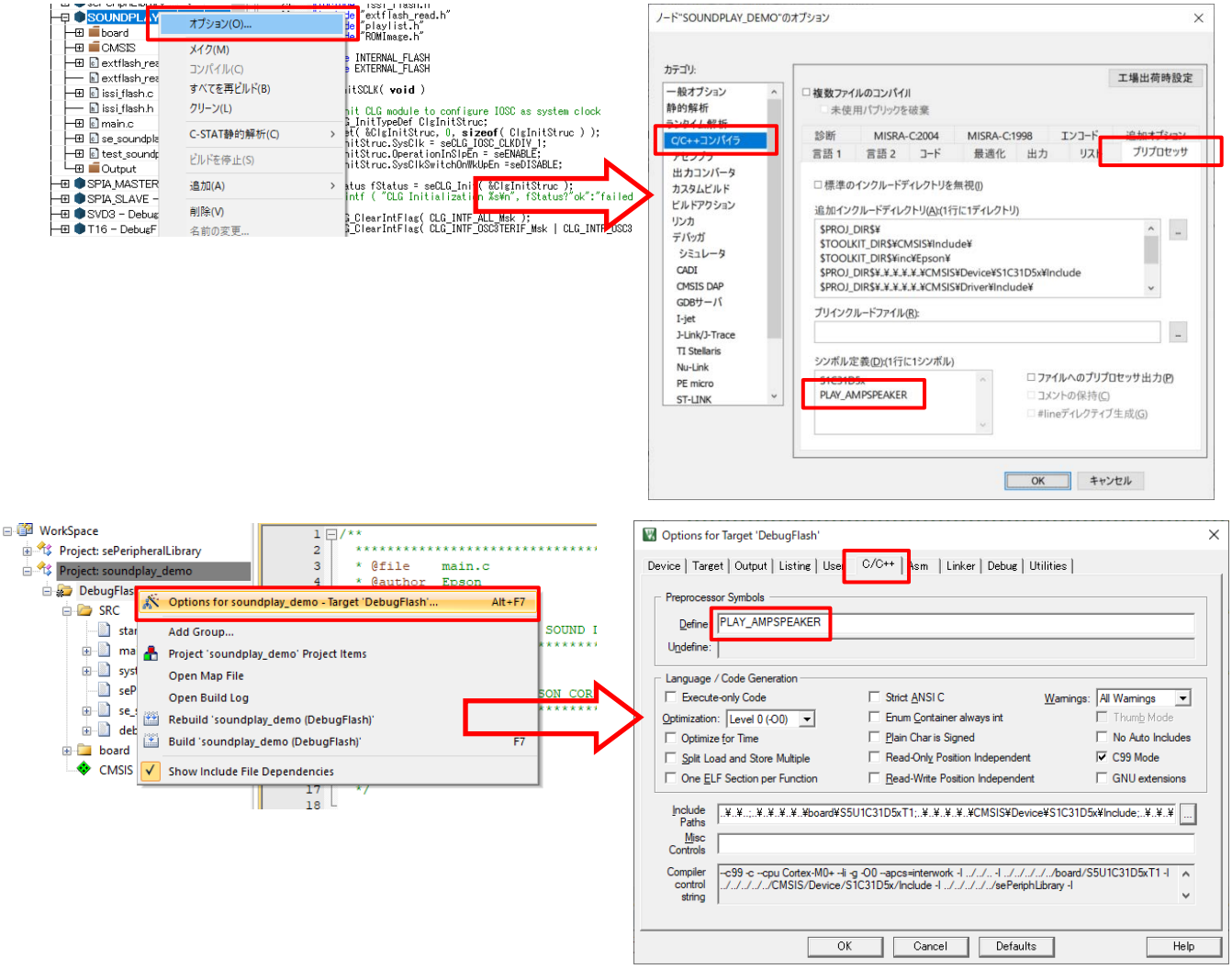


図 3.20.1　シンボル定義方法(上図は IAR EWARM、下図は Keil MDKARM)

動作

1. SOUNDPLAY 用にシステムクロックを設定します。（詳細は main.c の InitSCLK 関数を参照）
2. オーディオアンプを有効にします。(詳細は main.c の InitAMP 関数を参照)
3. 内蔵 Flash メモリ中の音声ファイルで構成されるセンテンスの再生を行います。
   - 3.1 seSoundPlayInit 関数を呼出し、SOUNDPLAY を初期化します。
   - 3.2 seSoundPlaySetParamteter 関数を呼出し、センテンス番号、再生チャンネル、ボリューム、リピート数、音声速度を設定します。
   - 3.3 seSoundPlayCommand 関数を呼出し、指定されたセンテンスの再生を開始します。
   - 3.4 音声再生が終了したことを示す割り込みを待ちます。
   - 3.5 必要に応じて 3.2~3.4 を繰り返します。
   - 3.6 seSoundPlayFinish 関数を呼出して、SOUNDPLAY を終了します。
4. 外部 QSPI Flash メモリにアクセスするために QSPI モジュールを初期化します。（詳細は extflash_read.c の InitExtFlash 関数を参照）

5. 外部 QSPI Flash メモリ中の音声ファイルで構成されるセンテンスの再生を行います。
   - 5.1 seSoundPlayInit 関数を呼出し、SOUNDPLAY を初期化します。

5.2 seSoundPlaySetParamteter 関数を呼出し、センテンス番号、再生チャンネル、ボリューム、リピート数、音声速度を設定します。
5.3 seSoundPlayCommand 関数を呼出し、指定されたセンテンスの再生を開始します。
5.4 音声再生が終了したことを示す割り込みを待ちます。
5.5 必要に応じて 5.2~5.4 を繰り返します。
5.6 seSoundPlayFinish 関数を呼出して、SOUNDPLAY を終了します。

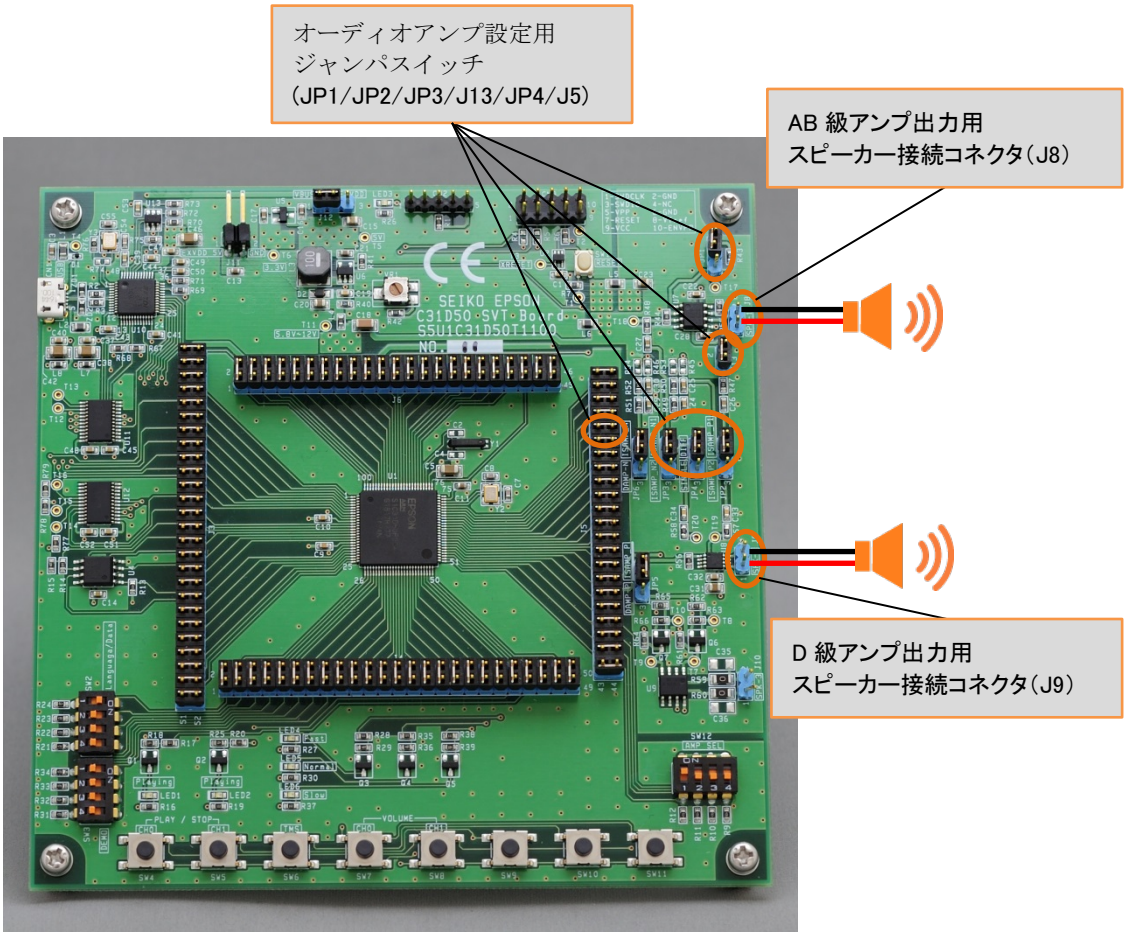


図 3.20.2 音声再生機能に関連するジャンパとコネクタの配置

(2) ディスクリート回路を使用した電磁ブザーによる音声再生 (S1C31D51のみ)

サンプルソフトウエアセットアップ

1. SOUNDPLAY プロジェクトのシンボル定義に"PLAY_ELBUZZER"を定義する。(設定方法は図 3.20.1 参照)
2. SOUNDPLAY プロジェクトをビルドする。

ハードウェアセットアップ

1. 評価ボード(S5U1C31D51T1)とブザーボード(S5U1C31D51T2)を、シルクの向きを合わせて勘合します。(表 3.20.3、図 3.20.3、図 3.20.4 参照)
2. ブザーボードのジャンパおよび抵抗値を確認します(表 3.20.4、図 3.20.4 参照)。

3. ブザーボード(S5U1C31D51T2)に同梱の電磁ブザー(TDK 製 SD160709)を、ブザー接続コネクタに接続します（図 3.20.4 参照）。

表 3.20.3 評価ボード(S5U1C31D51T1)の設定（ブザーボード(S5U1C31D51T2)との勘合時）

ジャンパ: J3/J4/J5/J6			
J3	**J4**	**J5**	**J6**
51-52 オープン	45-46 オープン	1-2 オープン	1-2 オープン
-	-	21-22 オープン	43-44 オープン
-	-	23-24 オープン	45-46 オープン
-	-	43-44 オープン	-

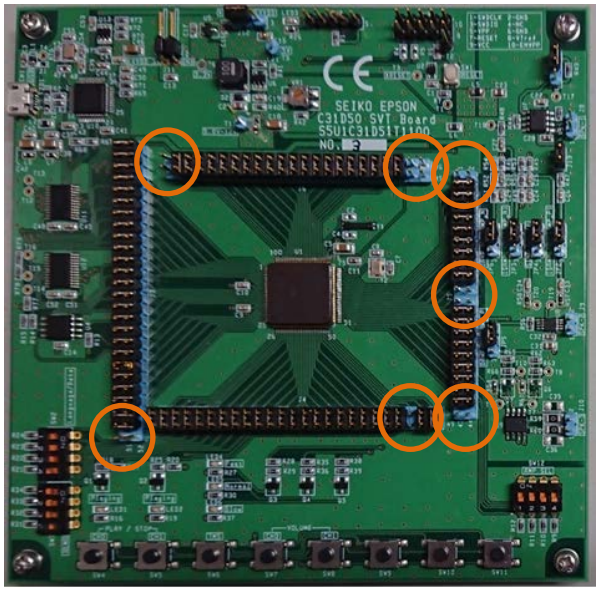


図 3.20.3 評価ボード(S5U1C31D51T1)のジャンパ設定（ブザーボード(S5U1C31D51T2)との勘合時）

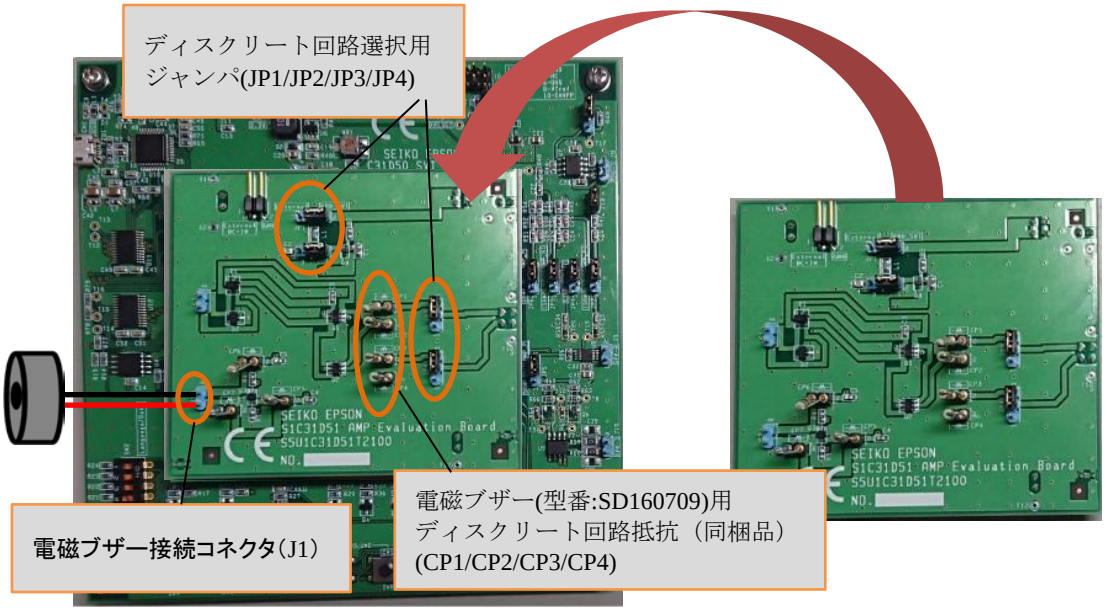


図 3.20.4 電磁ブザー使用時のブザーボード(S5U1C31D51T2)のジャンパ設定および抵抗値

表 3.20.4 電磁ブザー使用時のブザーボード(S5U1C31D51T2)のジャンパ設定および抵抗値

ジャンパ: JP1/JP2/JP3/JP4			
JP1	JP2	JP3	JP4
1-2 ショート	1-2 ショート	1-2 ショート	1-2 ショート

抵抗: CP1/CP2/CP3/CP4			
CP1	CP2	CP3	CP4
2.2kΩ	2.2kΩ	2.2kΩ	2.2kΩ

動作

1. SOUNDPLAY 用にシステムクロックを設定します。(詳細は main.c の InitSCLK 関数を参照)
2. 内蔵 Flash メモリ中の音声ファイルで構成されるセンテンスの再生を行います。
   2.1 seSoundPlayInit_ELBuzzer 関数を呼出し、SOUNDPLAY を初期化します。
   2.2 seSoundPlaySetParamteter 関数を呼出し、センテンス番号、再生チャンネル、ボリューム、リピート数、音声速度を設定します。
   2.3 seSoundPlayCommand 関数を呼出し、指定されたセンテンスの再生を開始します。
   2.4 音声再生が終了したことを示す割り込みを待ちます。
   2.5 必要に応じて 3.2~3.4 を繰り返します。
   2.6 seSoundPlayFinish_ELBuzzer 関数を呼出して、SOUNDPLAY を終了します。
3. 外部 QSPI Flash メモリにアクセスするために QSPI モジュールを初期化します。(詳細は extflash_read.c の InitExtFlash 関数を参照)
4. 外部 QSPI Flash メモリ中の音声ファイルで構成されるセンテンスの再生を行います。
   4.1 seSoundPlayInit_ELBuzzer 関数を呼出し、SOUNDPLAY を初期化します。
   4.2 seSoundPlaySetParamteter 関数を呼出し、センテンス番号、再生チャンネル、ボリューム、リピート数、音声速度を設定します。
   4.3 seSoundPlayCommand 関数を呼出し、指定されたセンテンスの再生を開始します。
   4.4 音声再生が終了したことを示す割り込みを待ちます。
   4.5 必要に応じて 5.2~5.4 を繰り返します。
   4.6 seSoundPlayFinish_ELBuzzer 関数を呼出して、SOUNDPLAY を終了します。

※ ブザーボード(S5U1C31D51T21)に搭載されている電磁ブザー用ディスクリート回路には、**回路の入力信号によっては、大電流が流れる状態があります。**そのため、本ディスクリート回路の電源は、S1C31D51 のポート(P46)で ON/OFF が制御可能となっており、本サンプルソフトウェアで用意している初期化関数(seSoundPlayInit_ELBuzzer)内で電源 ON (P46 を制御)、再生終了後に呼び出すための終了関数(seSoundPlayFinish_ELBuzzer)内で、電源 OFF とし、再生中のみ電源を ON になるように制御しています。

(3) ディスクリート回路を使用した圧電ブザーによる音声再生 (S1C31D51のみ)

サンプルソフトウエアセットアップ

1. SOUNDPLAY プロジェクトのシンボル定義に"PLAY_PIBUZZER"を定義する。(設定方法は図 3.20.1 参照)
2. SOUNDPLAY プロジェクトをビルドする。

ハードウェアセットアップ

1. 評価ボード(S5U1C31D51T1)とブザーボード(S5U1C31D51T2)を、シルクの向きを合わせて勘合します。(表 3.20.3、図 3.20.3、図 3.20.4 参照)
2. ブザーボードのジャンパおよび抵抗値を確認します(表 3.20.5)。
3. ブザーボード(S5U1C31D51T2)に同梱の圧電ブザー(TDK 製 PS1740P02CE)を、ブザー接続コネクタに接続します(図 3.20.5 参照)。

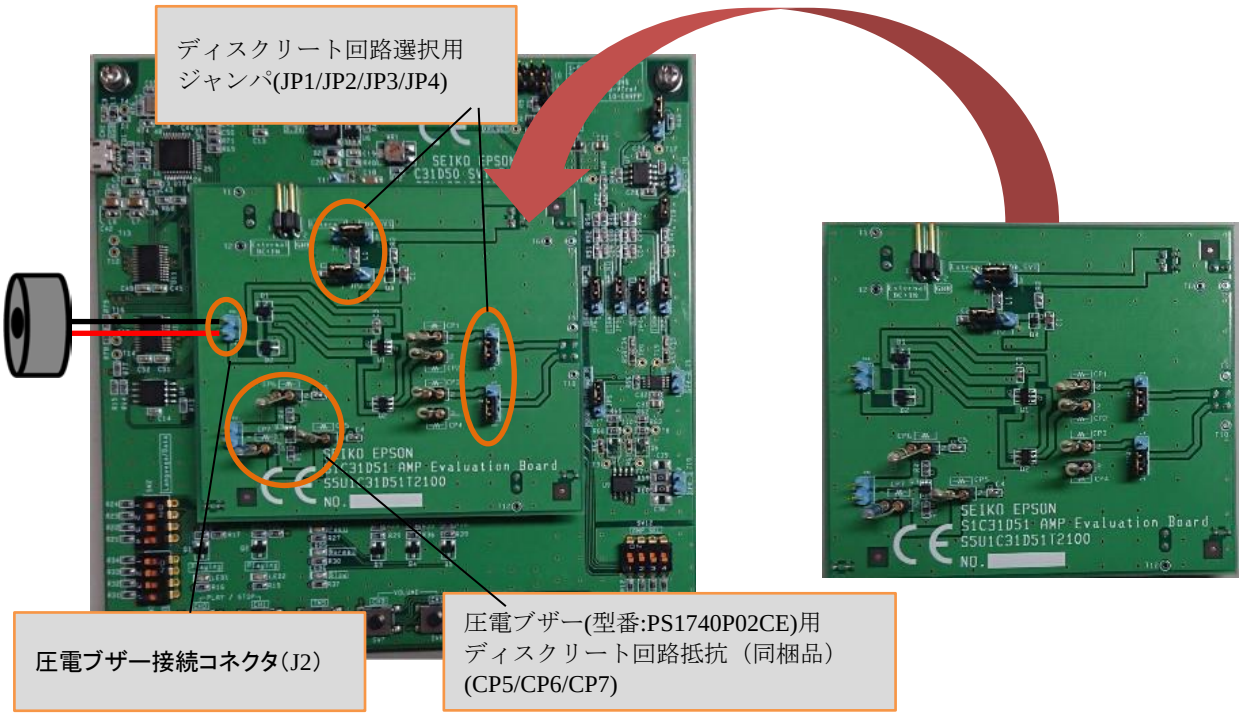


図 3.20.5 圧電ブザー使用時のブザーボード(S5U1C31D51T2)のジャンパ設定および抵抗値

表 3.20.5 圧電ブザー使用時のブザーボード(S5U1C31D51T2)のジャンパ設定および抵抗値

ジャンパ: JP1/JP2/JP3/JP4			
JP1	JP2	JP3	JP4
1-2 ショート	2-3 ショート	2-3 ショート	2-3 ショート

抵抗: CP5/CP6/CP7		
CP5	CP6	CP7
180Ω	180Ω	100Ω

動作

1. SOUNDPLAY 用にシステムクロックを設定します。(詳細は main.c の InitSCLK 関数を参照)
2. 内蔵 Flash メモリ中の音声ファイルで構成されるセンテンスの再生を行います。
   2.1 seSoundPlayInit_PIBuzzer 関数を呼出し、SOUNDPLAY を初期化します。
   2.2 seSoundPlaySetParamteter 関数を呼出し、センテンス番号、再生チャンネル、ボリューム、リピート数、音声速度を設定します。
   2.3 seSoundPlayCommand 関数を呼出し、指定されたセンテンスの再生を開始します。

2.4 音声再生が終了したことを示す割り込みを待ちます。
2.5 必要に応じて 3.2~3.4 を繰り返します。
2.6 seSoundPlayFinish_PIBuzzer 関数を呼出して、SOUNDPLAY を終了します。

3. 外部 QSPI Flash メモリにアクセスするために QSPI モジュールを初期化します。（詳細は extflash_read.c の InitExtFlash 関数を参照）

4. 外部 QSPI Flash メモリ中の音声ファイルで構成されるセンテンスの再生を行います。
4.1 seSoundPlayInit_PIBuzzer 関数を呼出し、SOUNDPLAY を初期化します。
4.2 seSoundPlaySetParamteter 関数を呼出し、センテンス番号、再生チャンネル、ボリューム、リピート数、音声速度を設定します。
4.3 seSoundPlayCommand 関数を呼出し、指定されたセンテンスの再生を開始します。
4.4 音声再生が終了したことを示す割り込みを待ちます。
4.5 必要に応じて 5.2~5.4 を繰り返します。
4.6 seSoundPlayFinish_PIBuzzer 関数を呼出して、SOUNDPLAY を終了します。

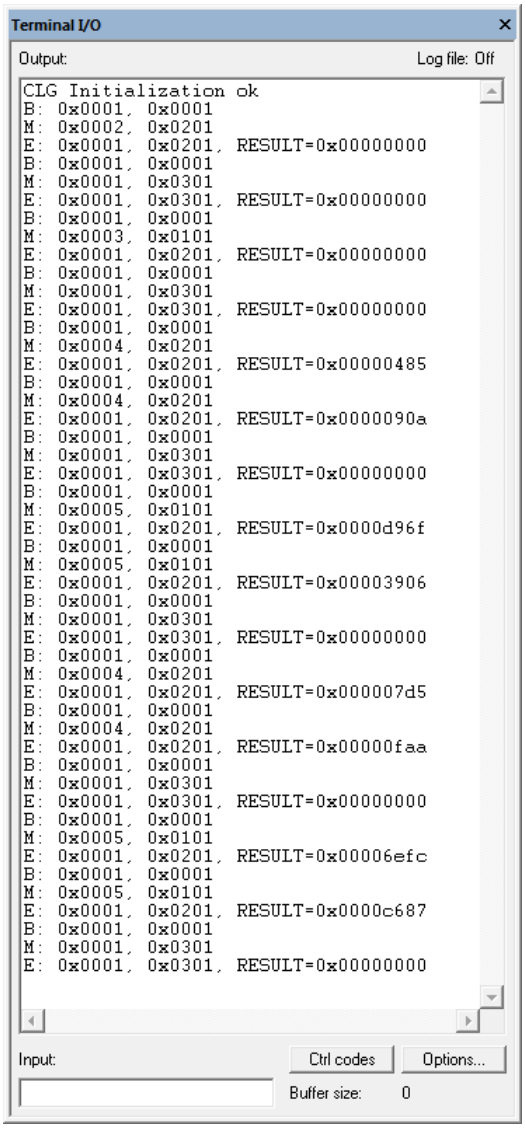
3.21 メモリチェック (MEMCHECK)

このサンプルプログラムは、HW Processor で提供されるメモリチェック機能を使用して、内蔵 RAM、内蔵 Flash メモリ、外部 QSPI-Flash メモリの自己診断を行います。

動作

1. MEMCHECK 用にシステムクロックを設定します。(詳細は main.c の InitSCLK 関数を参照)
2. 内蔵 RAM に対して Read/Write による RAM チェックを実行します。
3. 内蔵 RAM に対して March-C アルゴリズムによる RAM チェックを実行します。
4. 内蔵 Flash メモリに対してチェックサムを実行します。
5. 内蔵 Flash メモリに対して CRC-16 を実行します。
6. 外部 QSPI Flash メモリにアクセスするために QSPI モジュールを初期化します。(詳細は extflash_read.c の InitExtFlash 関数を参照)
7. 外部 QSPI Flash メモリに対してチェックサムを実行します。
8. 外部 QSPI Flash メモリに対して CRC-16 を実行します。

出力例



3.22 フラッシュプログラミング(FLASH)

このサンプルプログラムは、seFlashLibrary を使用して内蔵 Flash メモリの消去／書き込みを行います。

seFlashLibrary は、CMSIS-Driver Flash Interface の仕様に基づいて実装しています。

動作

1. GetInfo 関数をコールして、内蔵 Flash メモリの情報を取得します。
2. GetVersion 関数をコールして、フラッシュドライバのバージョンを取得します。
3. Initialize 関数をコールして、フラッシュドライバを初期化します。
4. EraseSector 関数をコールして、引数指定されたセクタを消去します。
5. ProgramData 関数をコールして、ステップ 4 で指定されたセクタにデータを書き込みます。
6. ReadData 関数をコールして、ステップ 4 で指定されたセクタからデータを読み込み、その読み込みデータとステップ 5 の書き込みデータを比較します。
7. Uninitialize 関数をコールして、フラッシュドライバの初期化状態を解除します。

seFlashLibrary の使用上の注意

本ライブラリを使用する場合、以下の点に注意してください。

- 本ライブラリで提供される関数をコールする前に必ず割込み禁止に設定してください。
- 本ライブラリ実行時は、ライブラリを配置した領域を壊さないようにしてください。
- 本ライブラリを使用する際は、Flash メモリの書き換え可能回数に注意してください。Flash メモリの仕様については“S1C31D5x テクニカルマニュアル”を参照してください。
- 本ライブラリは 16 ビットタイマ（T16） ch.0 を使用しています。そのため、本ライブラリ実行後は、16 ビットタイマ（T16） ch.0 のレジスタ設定が変更されています。16 ビットタイマ(T16)ch.0 を使用しているプログラムで、本ライブラリを使用する場合は注意して下さい。
- 本ライブラリはシステムクロックを IOSC に切り換えて実行します。そのため、ライブラリ実行後は、クロックジェネレータ（CLG）のレジスタ設定が変更されています。必要に応じて、ライブラリ実行前に CLG のレジスタ設定をバックアップし、ライブラリ実行後に CLG のレジスタ設定を復元するなどの対策を取ってください。

3.23 EEPROMライブラリ (EEPROM)

このサンプルプログラムは、内蔵 Flash メモリで EEPROM 機能を実現するエミュレーションライブラリ（seEepromLibrary）を使用して、データの書き込み／読み込みを行います。seEepromLibrary は、EEPROM の各バイトに 256 バイトの内蔵 Flash メモリを使用します。0x20000 から 0x2FFFF の領域を EEPROM のエミュレーションに使用した場合、256 バイトの EEPROM に 100,000 回の読み書きを提供します。

seEepromLibrary は、CMSIS-Driver Flash Interface の仕様に基づいて実装しています。EEPROM ライブラリのヘッダーファイルは、CMSIS¥Driver¥Include¥Driver_EEPROM.h にあります。seEepromLibrary を使用して初めてデータを書く場合は、0x20000 から 0x2FFFF の領域を一度だけ消去する必要があります。

パラメータ設定

EEPROM のエミュレーション領域を変更するにはリンカ設定ファイルを編集し、EEPROM を配置するセクションのメモリ領域を再定義してください。

IAR EWARM の場合は、"Examples/EEPROM/board/S5U1C31D5xT1/IAR/config/S1C31D5x_flash.icf" の "EEPROM1"のメモリ領域を再定義してください。

MDK-ARM（μVision）の場合は、"Examples /EEPROM/board/S5U1C31D5xT1/ARM/eeprom_flash.sct"の "LR_IROM2"と"ER_IROM2"のメモリ領域を再定義してください。

EEPROM のサイズとリトライ回数を変更するには"Driver_EEPROM.h"を編集し、以下の定数を再定義してください。

#define CONFIG_EEPROM_SIZE_MAX（256）
#define CONFIG_RETRY_COUNT（4）

「CONFIG_EEPROM_SIZE_MAX」は EEPROM のサイズを示します。このサイズは 32/64/128/256 から選択できます。「CONFIG_RETRY_COUNT」は書き込みが失敗した時の書き込みリトライ回数を示します。書き込みリトライ回数を増やすと、書き込みルーチンの処理時間が長くなり、パフォーマンスが低下するおそれがありますので、数回程度で設定することを推奨します。

動作

1. GetInfo 関数をコールして、EEPROM の情報を取得します。
2. GetVersion 関数をコールして、EEPROM ドライバのバージョンを取得します。
3. ProgramData 関数をコールして、EEPROM にデータを書き込みます。
4. ReadData 関数をコールして、EEPROM からデータを読み込みます。

seFlashLibrary の使用上の注意

本ライブラリを使用する場合、以下の点に注意してください。

- 本ライブラリで提供される関数をコールする前に必ず割込み禁止に設定してください。
- 本ライブラリ実行時は、ライブラリを配置した領域を壊さないようにしてください。
- 本ライブラリは 16 ビットタイマ（T16） ch.0 を使用しています。そのため、本ライブラリ実行後は、16 ビットタイマ（T16） ch.0 のレジスタ設定が変更されています。16 ビットタイマ(T16)ch.0 を使用しているプログラムで、本ライブラリを使用する場合は注意して下さい。
- 本ライブラリはシステムクロックを IOSC に切り換えて実行します。そのため、ライブラリ実行後は、クロックジェネレータ（CLG）のレジスタ設定が変更されています。必要に応じて、ライブラリ実行前に CLG のレジスタ設定をバックアップし、ライブラリ実行後に CLG のレジスタ設定を復元するなどの対策を取ってください。

3.24 ブートローダ (BootLoader)

このサンプルプログラムは、UART 通信により外部から送信されるプログラムのロードを行います。

本サンプルプログラムの詳細については、別紙「S1C31D5xブートローダマニュアル」を参照ください。

4. デモソフトウェア

デモソフトウェアは、S5U1C31D5xT1 評価ボード上の S1C31D5x の内蔵 Flash メモリにプリインストールされています。このデモソフトウェアのソースコードは、サンプルソフトウェアパッケージの“Demos ¥SOUDNPLAY_DEMO”に含まれています。

デモソフトウェアには、

- オーディオアンプとスピーカーによる音声再生
- ディスクリート回路と電磁ブザーによる音声再生 (S1C31D51 のみ)
- ディスクリート回路と圧電ブザーによる音声再生 (S1C31D51 のみ)

の 3 つの再生モードが用意されております。

また、デモソフトウェアには、以下に示す 2 つのモードがあります。

- 音声再生デモモード

  プッシュスイッチを操作して音声再生を行うモード

- 音声 ROM アップデートモード

  外部 QSPI Flash メモリ中の音声 ROM データをアップデートするモード

4.1 ハードウェア設定

図 4.1.1 は、デモソフトウェアを実行するための S5U1C31D5xT1 の主要部品配置を示しています。また、表 4.1.1~4.1.4 は、S5U1C31D5xT1 のジャンパや DIP スイッチの設定を示しています。

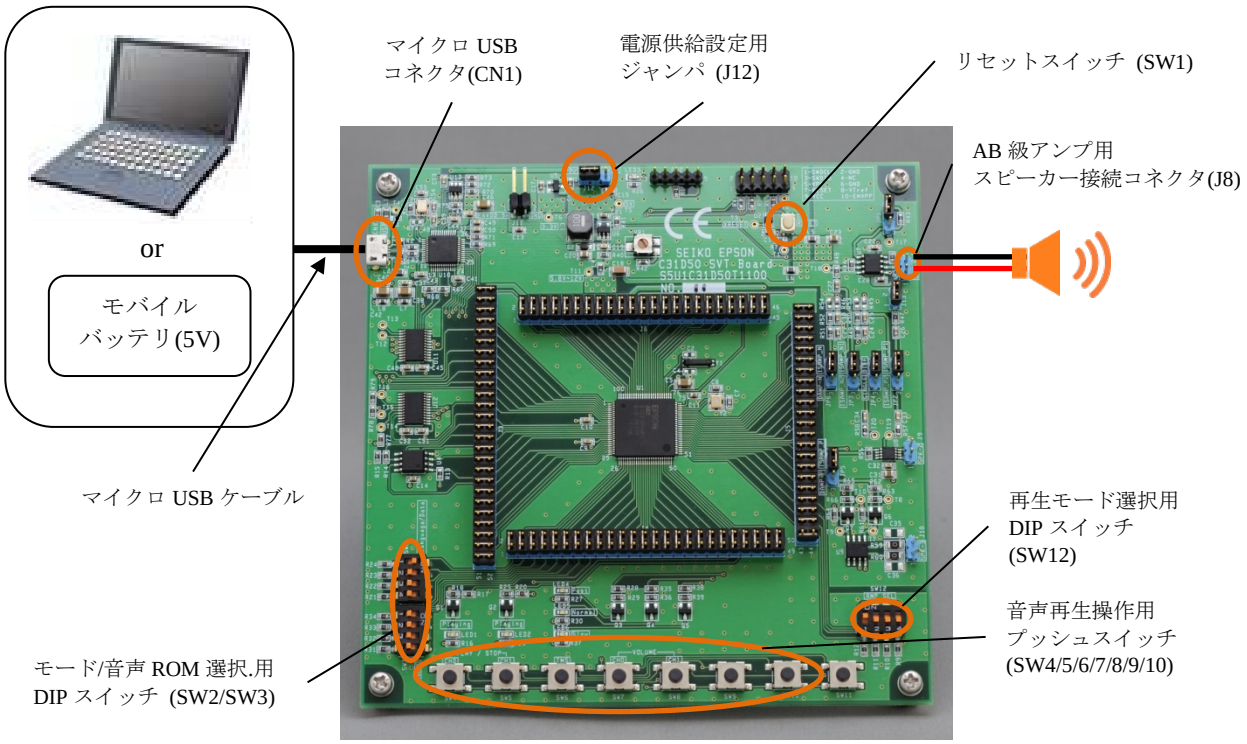


図 4.1.1 評価ボード(S5U1C31D5xT1)主要部品の配置

表 4.1.1 電源供給設定用ジャンパ

J12	備考
1 -2 ショート	デフォルト

表 4.1.2 AB 級アンプ設定用ジャンパ

JP1	JP2	JP3	J13	JP4	備考
2-3 ショート	1-2 ショート	1-2 ショート	1-2 ショート	1-2 ショート	デフォルト

表 4.1.3 ストレージ選択用 DIP スイッチ(SW2-1)

SW2-1	説明	備考
OFF	MCU 内蔵 Flash メモリ	
ON	外付け QSPI Flash メモリ	デフォルト

表 4.1.4 モード選択用 DIP スイッチ(SW3-1)

SW3-1	説明	備考
OFF	音声再生デモモード	デフォルト
ON	音声 ROM アップデートモード	

4.2 音声再生デモモードの実行方法

音声再生デモモードでは、プッシュスイッチ(SW4, SW5, …, SW10)を操作して 2 チャンネルミキシング再生や話速変換再生等の音声再生を行います。

本モードの実行手順は以下のとおりです。

1) ジャンパ J12 を“VBUS 5V”側に設定する。(J12-1 と J12-2 をショート)
2) マイクロ USB ケーブルを CN1 に接続する。
3) マイクロ USB ケーブルをモバイルバッテリーまたは PC に接続する。
4) SW3-1 を OFF に設定し、音声再生デモモードを選択する。
5) S5U1C31D51T1 で、電磁ブザー/圧電ブザーのデモモードを選択した場合は、ブザーボード(S5U1C31D51T2)を勘合接続し、JP とディスクリート回路の抵抗(CP)を適切に設定する。(3.20 項参照)) 内蔵 Flash メモリ中の音声 ROM データにアクセスする場合は SW2-1 を OFF に、外部 QSPI Flash メモリ中の音声 ROM データにアクセスする場合は SW2-1 を ON に設定する。
6) 手順 5 で外部 QSPI Flash メモリアクセスを選択した場合、SW12 で再生モード、SW2, SW3 で言語タイプを選択する。(SW2, SW3, SW12 の詳細は、S1C31D50/D51 評価ボードスタートガイドを参照。)
7) SW1(RESET)を押して、評価ボードをリセットする。リセット後、LED5 が点灯していることを確認する。
8) SW4 または SW5(PLAY/STOP)を押して音声再生を開始する。
(音声再生操作の詳細については、図 4.2.1 および表 4.2.1 を参照ください)。

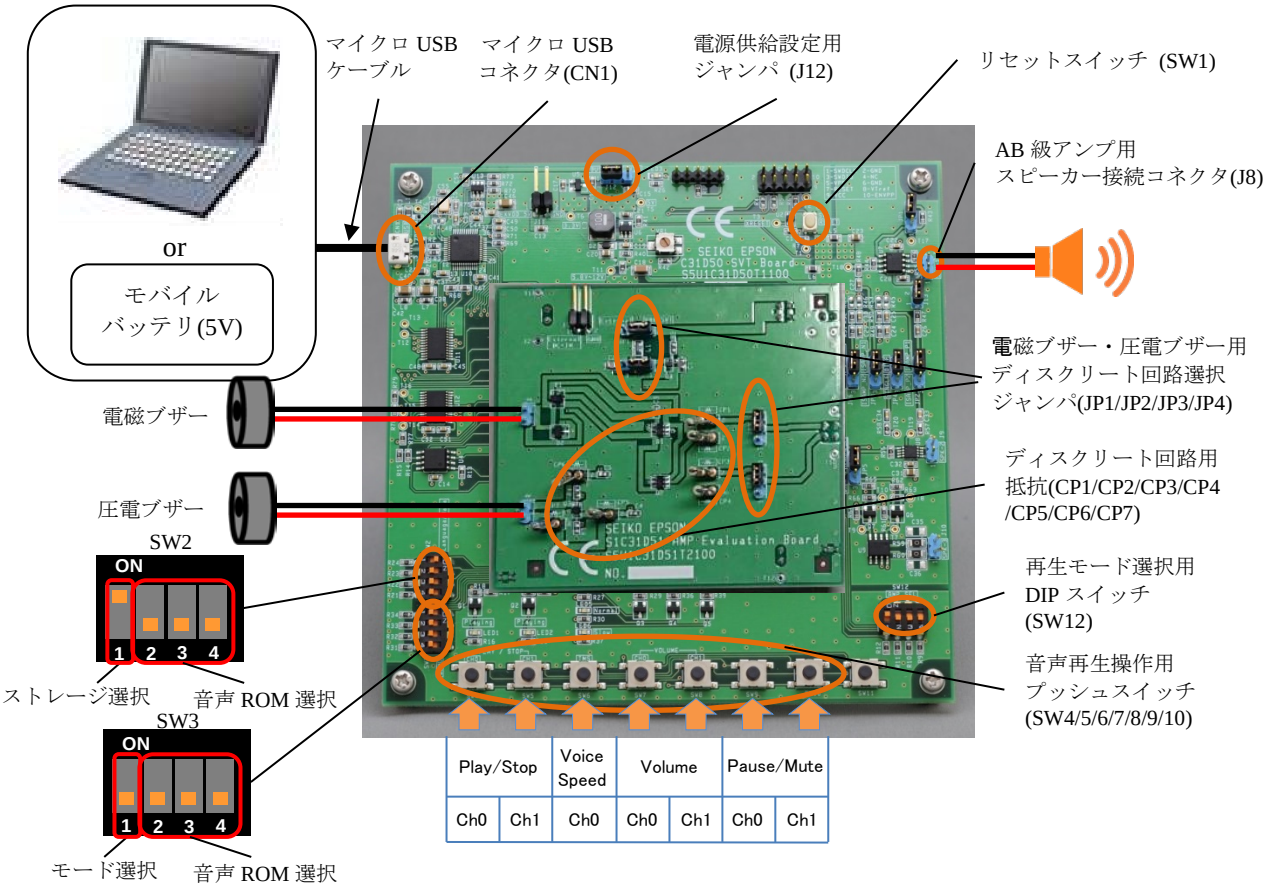


図 4.2.1 デモ設定

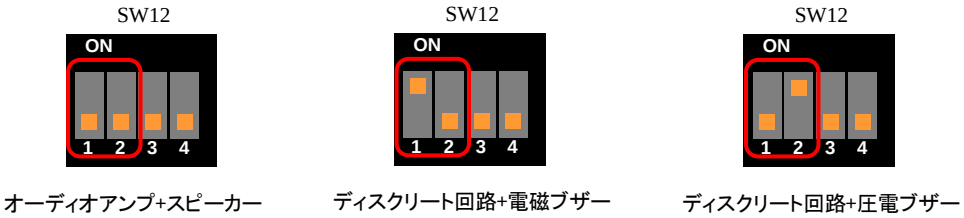


図 4.2.2　　SW12 による再生モードの設定

表 4.2.1 プッシュスイッチの詳細

Push SW	Operation	Description
SW4/SW5 (Play/Stop for Ch.0,Ch.1)	1 回押し	センテンスの再生を開始します。 この操作を行う度に、次のセンテンスを再生します。 この操作を再生中に行った場合、現在のセンテンスの再生を終了して、次のセンテンスを再生します。
	2 回押し	センテンスの再生を開始します。 この操作を行う度に、前のセンテンスを再生します。 この操作を再生中に行った場合、現在のセンテンスの再生を終了して、前のセンテンスを再生します。
	長押し	センテンスの再生を終了します。 この操作をポーズ中またはミュート中に行った場合、ポーズまたはミュートを解除して、現在のセンテンスの再生を終了します。この操作を停止中に行った場合、再生対象となるセンテンスを初期状態に戻します。
SW6 (Voice Speed for Ch.0)	1 回押し	音声速度を 5%ステップで下げます。 この操作を再生中に行った場合、現在のセンテンスの再生を終了して、速度を下げて同じセンテンスを再生します。
	2 回押し	音声速度を 5%ステップで上げます。 この操作を再生中に行った場合、現在のセンテンスの再生を終了して、速度を上げて同じセンテンスを再生します。
	長押し	音声速度を初期値に戻します。
SW7/SW8 (Volume for Ch.0,Ch.1)	1 回押し	音量を 3dB ステップで下げます。
	2 回押し	音量を 3dB ステップで上げます。
	長押し	音量を初期値に戻します。
SW9/SW10 (Pause/Mute for Ch.0,Ch.1)	1 回押し	再生中のセンテンスをポーズします。 この操作をポーズ中またはミュート中に行った場合、ポーズまたはミュートが解除されます。
	2 回押し	再生中のセンテンスをミュートします。

4.3 音声 ROM アップデートモードの実行方法

音声 ROM アップデートモードでは、外部 QSPI Flash メモリ中の音声 ROM データのアップデートを行います。

本モードの実行手順は以下のとおりです。

1) ESPER2（EPSON 製音声データ作成ツール）を使用して、S1C31D5x 用の音声 ROM データ(ROMImage_C31D5x_*.bin)を作成する。
2) ジャンパ J12 を“VBUS 5V”側に設定する。(J12-1 と J12-2 をショート)
3) マイクロ USB ケーブルを CN1 に接続する。
4) マイクロ USB ケーブルを PC に接続する。
5) SW3-1 を ON に設定し、音声 ROM アップデートモードを選択する。
6) SW1(RESET)を押して、評価ボードをリセットする。リセット後、LED1,LED2,LED6 が点灯していることを確認する。
7) ESPER2 で作成した音声 ROM データ(ROMImage_C31D5x_*.bin)と、音声 ROM データと同時に生成されるプレイリストファイル（PlayList_C31D5x_*.txt）を、以下に示すサンプルソフトウェアパッケージ内のフォルダにコピーする。
   - FlashTools/S5U1C31D5xT1
8) 以下に示すバッチファイルを実行し、外部 QSPI Flash メモリ中の音声 ROM データとプレイリストファイルを書き換えてアップデートを行う。
   - FlashTools¥S5U1C31D5xT1/run_write_flash_romdata.bat
   - FlashTools¥S5U1C31D5xT1/run_write_flash_playlist.bat

プレイリストファイル（PlayList_C31D5x_*.txt）を編集することで、再生するセンテンスの選択や順番を変更できます。プレイリストファイルの詳細については、「付録.B. プレイリストファイル」を参照ください。

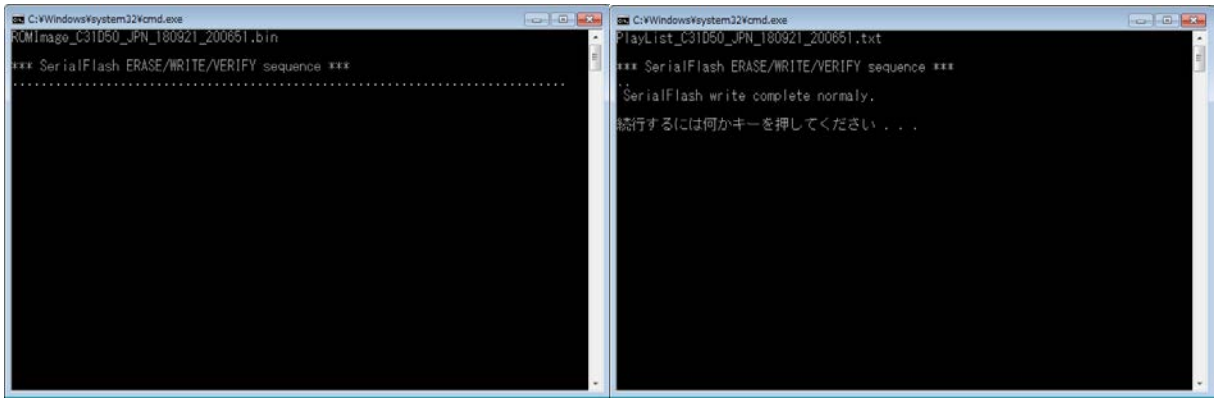


図 4.3.1 バッチファイル実行時の画面表示

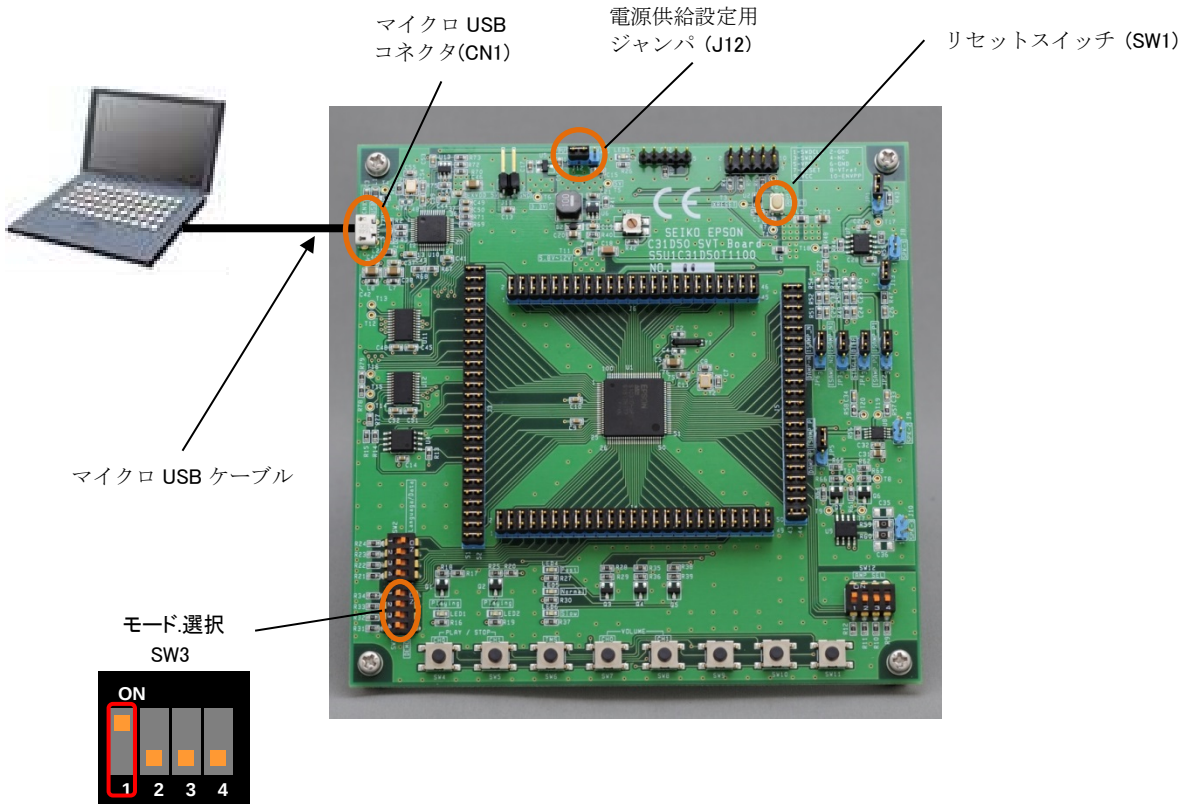


図 4.3.2 主要部品の配置（音声 ROM アップデートモード時）

5. 自己診断サンプルソフトウェア(IEC60730規格対応)

5.1 概要

国際電気標準会議（IEC: International Electrotechnical Commission）は家電機器開発のための IEC60730 指令を発表しています。この安全規格は、欧州で販売・使用される家電機器に対して準拠が義務付けられています。この規格は、最終製品に対して定期的な自己診断を実施することで、故障や不具合を早期に発見し、これによって生じる危険から消費者を守ることを目的としています。

マイクロコントローラ関係のソフトウェア制御は以下の規格によって分類されます。

Class A: 設備の安全性の維持を目的としない制御（照明器具等の制御）

Class B: 制御する設備の危険な操作を防ぐことを目的とする制御（洗濯機、冷蔵庫、冷凍庫、食器洗い機等の制御）

Class C: 特別な危険を防ぐことを目的とする制御（燃焼器具等の制御）

これらの Class のうち、家電製品を制御するソフトウェアのほとんどは Class B に分類され、最終製品に対し以下の自己診断テストを実施することが推奨対策とされています。

・マイコンやプログラムカウンタのスタック故障診断

・割り込み周期異常診断

・マイコンの動作クロック周波数の異常診断

・ROM/RAM メモリの異常診断

・外部インタフェースによる通信異常診断

より詳しい情報は、IEC60730 の Annex H を参照してください。

このドキュメントは、S1C31D5x 対してこれらの自己診断テストを行うサンプルソフトウェアおよび参考情報について記載しています。

自己診断の内容は、

・メモリの故障診断テスト（Read Write テスト、March-C テスト）

・メモリに置かれたデータの確からしさテスト（チェックサム生成、CRC 生成）

・割り込みテスト（割り込み周期、割り込み回数確認）

・メインクロック安定性確認テスト（動作周波数確認）

です。

Read Write テストおよび March-C テストでは、指定範囲のメモリ、レジスタ、スタックポインタ、そしてステータスレジスタについて書き込み、読み出しテストを実施します。

チェックサム、CRC 生成では指定範囲のメモリに置かれたデータについて、それぞれ誤り検出符号の値を求め、返します。

割り込みテストは、一定周期間に何回割り込みが発生したかをカウントし、その値を返します。

メインクロック安定性確認テストでは、サブクロック（32KHz）を用いてメインクロックが正常な動作周波数で動作しているかを確認します。

5.2 サンプル

この自己診断サンプルソフトウェアのソースコードは、サンプルソフトウェアパッケージの“IEC 60730 compliant”に含まれています。自己診断サンプルソフトウェアの内容は表 5.1.1 のとおりです。各サンプルの詳細は、付録 D を参照ください。

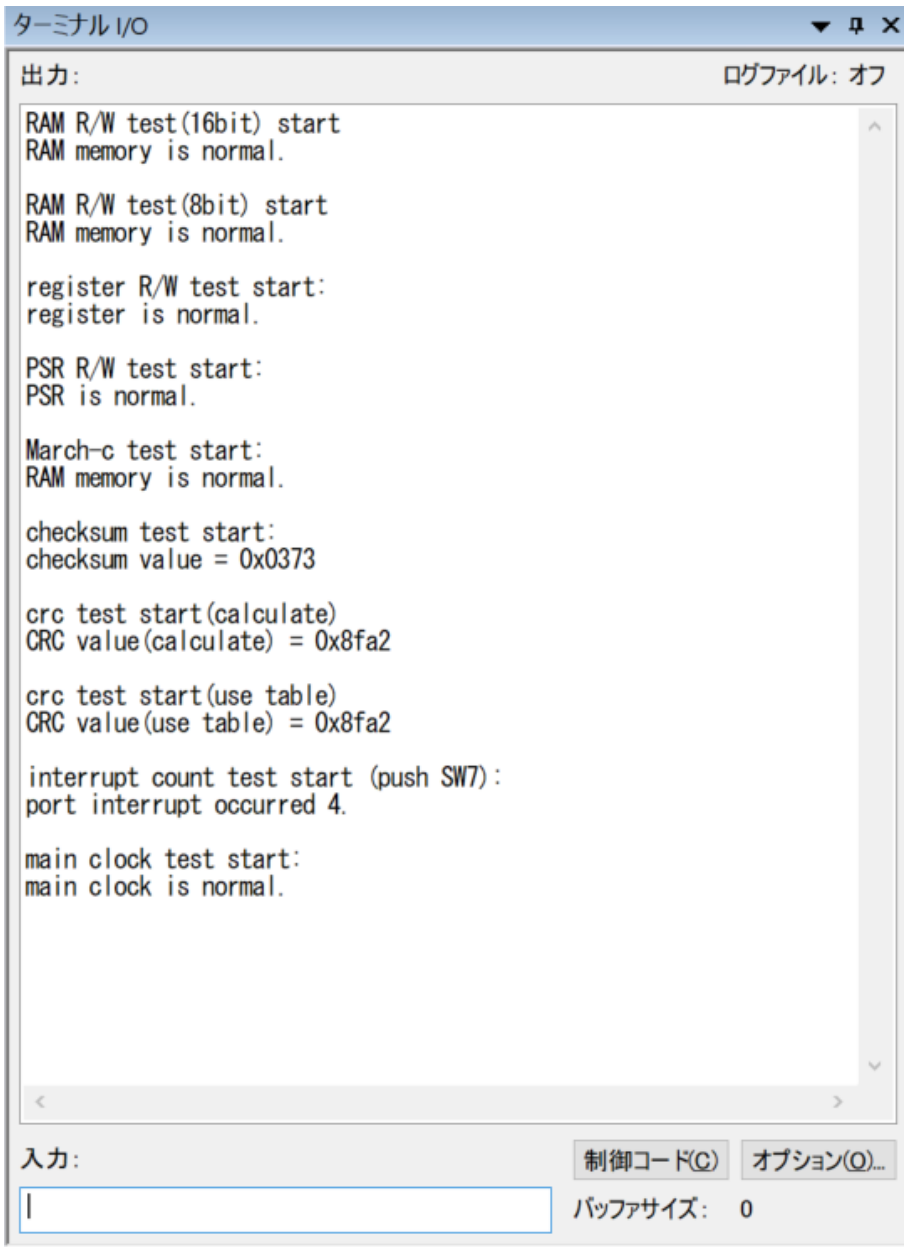
表 5.1.1 自己診断サンプルソフトウェアファイル

ファイル名	内容
main.c	テスト関数をコール
s1c31TestRam16.c	RAM の R/W テスト(16 ビットデバイス用)
s1c31TestRam8.c	RAM の R/W テスト(8 ビットデバイス用)
s1c31TestRegister.c	汎用レジスタ、スタックポインタの R/W テスト
s1c31TestRegister.s	汎用レジスタ、スタックポインタの R/W テスト
s1c31TestPsr.c	ステータスレジスタの R/W テスト
s1c31RwPsr.s	ステータスレジスタの R/W テストを行うために発生させるソフトウェア例外
s1c31TestRamMarchc.c	RAM の March-C テスト
s1c31TestChksum.c	チェックサムの算出
s1c31TestCrc.c	CRC の算出（計算）
s1c31TestCrcTbl.c	CRC の算出（テーブルを参照）
s1c31d5xTestInterupt.c	割り込みのテスト
s1c31d5xTestClk.c	メインクロック安定性のテスト
s1c31SelfTest.h	自己診断サンプルプログラムで用いられるヘッダーファイル

動作

1. RAM R/W テスト(16 ビット)を実施します。
2. RAM R/W テスト(8 ビット)を実施します。
3. レジスタ R/W テストを実施します。
4. PSR R/W テストを実施します。
5. March-C によるメモリ R/W テストを実施します。
6. チェックサムを計算します。
7. CRC を計算します。
8. 割り込みテストを実施します。(SW7 による押下回数によるポート割り込みテスト)
9. クロックテストを実施します。

出力例


ターミナル I/O
出力:
ログファイル: オフ
RAM R/W test(16bit) start
RAM memory is normal.
RAM R/W test(8bit) start
RAM memory is normal.
register R/W test start:
register is normal.
PSR R/W test start:
PSR is normal.
March-c test start:
RAM memory is normal.
checksum test start:
checksum value = 0x0373
crc test start(calculate)
CRC value(calculate) = 0x8fa2
crc test start(use table)
CRC value(use table) = 0x8fa2
interrupt count test start (push SW7):
port interrupt occurred 4.
main clock test start:
main clock is normal.
入力:
制御コード(C)
オプション(O)...
バッファサイズ: 0

付録.A. HW Processorライブラリ仕様

S1C31D5x HW Processorライブラリは、HW Processor Functionを使用するためのドライバセットです。ドライバは、デバイス定義ファイルに記述されるレジスタマップを使用して、モジュールの初期化や機能設定、モジュールへのアクセスなどを行います。

付録.A.1 SOUNDPLAY

seSoundPlayInit

構文	void seSoundPlayInit(unsigned long romStartAddress, unsigned long romSize, unsigned long keycode)
引数	romStartAddress 音声 ROM データの先頭アドレス 内蔵 Flash の場合: 0x00 0000, …, 0x02 FFF0 (16 バイトアライメント) 外部 QSPI-Flash の場合: 0x00 0000 + OFFSET 0x10 0000 + OFFSET 0x20 0000 + OFFSET ... 0xE0 0000 + OFFSET 0xF0 0000 + OFFSET * OFFSET は 0x04 0000 です。この値は、「S1C31D50/D51 テクニカルマニュアル」の「図 4.1.1 メモリマップ」の外部 Flash メモリ用メモリマップドアクセスエリアの先頭アドレスを示します。 romSize 音声 ROM データのサイズ 内蔵 Flash の場合: 0x03 0000 Byte (192 KByte) 以下 外部 QSPI-Flash の場合: 0x10 00000 Byte(16 MByte) 以下 keycode 音声 ROM データ中の eov ファイルをデコードするために必要な 32 ビット値 0x**** **** **** ****
戻り値	なし
説明	SOUNDPLAY の初期化を行います。オーディオアンプとスピーカーデモソフトウェアによる SOUNDPLAY を使用する場合、最初にコールしてください。

seSoundPlayInit_ELBuzzer

構文	**seSoundPlayInit** と同様
引数	**seSoundPlayInit** と同様
戻り値	なし
説明	SOUNDPLAY の初期化を行います。電磁ブザーデモソフトウェアによる SOUNDPLAY を使用する場合、最初にコールしてください。

seSoundPlayInit_PIBuzzer

構文	**seSoundPlayInit** と同様
引数	**seSoundPlayInit** と同様
戻り値	なし
説明	SOUNDPLAY の初期化を行います。圧電ブザーデモソフトウェアによる SOUNDPLAY を使用する場合、最初にコールしてください。

seSoundPlaySetParameter

構文	void seSoundPlaySetParameter unsigned char ch, unsigned short sentenceNo, unsigned char volume, unsigned char repeat unsigned char speed)
引数	ch SOUNDPLAY のチャンネル 0: チャンネル 0 1: チャンネル 1 sentenceNo 再生対象となるセンテンス番号 1(0x0001)～65535(0xFFFF) volume 音声ボリューム 0x7F: 0dB 0x7E: -0.5dB 0x7D: -1.0dB … 0x02: -63dB 0x01: -63.5dB 0x00: 無音 repeat センテンス再生のリピート回数 0x01～0xFE: リピート回数 0xFF: 無限ループ speed 音声再生速度(チャンネル 0 のみ) 75: 75% 遅い 80: 80% 85: 85% 90: 90% 95: 95% 100: 100% 通常速度 105: 105% 110: 110% 115: 115% 120: 120% 125: 125% 速い (5%刻みで指定)
戻り値	なし
説明	SOUNDPLAY 実行のためのパラメータを設定します。 seSoundPlayRunCommand()関数をコールする前にコールしてください。

seSoundPlayRunCommand

構文	void seSoundPlayRunCommand unsigned char ch, unsigned short command)
引数	ch SOUNDPLAY のチャンネル 0: チャンネル 0 1: チャンネル 1 2: 全チャンネル command SOUNDPLAY の制御コマンド 0x01 (sp_command_start): 再生開始 0x02 (sp_command_stop): 即時停止 0x03 (sp_command_stop_after_phrase): フレーズ再生後に停止 0x04 (sp_command_pause): 即時ポーズ 0x05 (sp_command_pause_after_phrase): フレーズ再生終了後にポーズ 0x06 (sp_command_unpause): ポーズ解除 0x07 (sp_command_mute): 即時ミュート 0x08 (sp_command_mute_after_phrase): フレーズ再生終了後にミュート 0x09 (sp_command_unmute): ミュート解除
戻り値	なし
説明	制御コマンドを実行します。 seSoundPlaySetParameters()関数をコールした後にコールしてください。

seSoundPlayGetState

構文	unsigned short seSoundPlayGetState unsigned char ch)
引数	ch SOUNDPLAY のチャンネル 0: チャンネル 0 1: チャンネル 1
戻り値	SOUNDPLAY の現在の状態 0x0000 (sp_state_init): 初期化中 0x0001 (sp_state_idle): アイドル中 0x0002 (sp_state_play): 再生中 0x0003 (sp_state_pause): ポーズ中 0x0004 (sp_state_mute): ミュート中
説明	引数で指定されたチャンネルの SOUNDPLAY の現在の状態を取得します。

seSoundPlayFinish

構文	void seSoundPlayFinish(void)
引数	なし
戻り値	なし
説明	SOUNDPLAY を終了します。

seSoundPlayFinish_ELBuzzer

構文	void seSoundPlayFinish_ELBuzzer(void)
引数	なし
戻り値	なし
説明	電磁ブザーデモソフトウェアによる SOUNDPLAY を終了します。

seSoundPlayFinish_PIBuzzer

構文	void seSoundPlayFinish_PIBuzzer(void)
引数	なし
戻り値	なし
説明	圧電ブザーデモソフトウェアによる SOUNDPLAY を終了します。

付録.A.2　MEMCHECK

seMemCheckInit

構文	void seMemCheckInit(void)
引数	なし
戻り値	なし
説明	MEMCHECK の初期化を行います。 MEMCHECK を使用する場合、最初にコールします。

seMemCheckSetParameter

構文	void seMemCheckSetParameter unsigned long memStartAddress, unsigned long memSize, unsigned long initValue)
引数	memStartAddress チェック対象となるメモリエリアの先頭アドレス RAM の場合: 0x15 0000, ..., 0x15 1FFF 0x15 3000, ..., 0x15 67FF 内蔵 Flash の場合: 0x00 0000, …, 0x02 FFFF 外部 QSPI-Flash の場合: 0x00 0000 + OFFSET, ・・・, 0x0F FFFF + OFFSET * OFFSET は 0x04 0000 です。この値は、「S1C31D50/D51 テクニカルマニュアル」の「図 4.1.1 メモリマップ」の外部 Flash メモリ用メモリマップドアクセスエリアの先頭アドレスを示します。 memSize チェック対象となるメモリエリアのサイズ initValue チェックサムまたは CRC16 の計算で使用される初期値 0x0000, …, 0xFFFF
戻り値	なし
説明	MEMCHECK 実行のためのパラメータを設定します。 seMemCheckRunCommand()関数をコールする前にコールしてください。

seMemCheckRunCommand

構文	void seSoundPlayRunCommand unsigned short command)
引数	command MEMCHECK の制御コマンド 0x02 (mc_command_ram_rw): RAM チェック(R/W)実行開始 0x03 (mc_command_ram_marach_c): RAM チェック(March-C)実行開始 0x04 (mc_command_checksum): チェックサム実行開始 0x05 (mc_command_crc): CRC 実行開始 0xFF (mc_command_stop): メモリチェック停止
戻り値	なし
説明	制御コマンドを実行します。 seMemCheckSetParameters()関数をコールした後でコールしてください。

seMemCheckGetState

構文	unsigned short seMemCheckGetState(void)
引数	なし
戻り値	MEMCHECK の現在の状態 0x0000 (mc_state_init): 初期化中 0x0001 (mc_state_idle): アイドル中 0x0002 (mc_state_ram_rw): RAM チェック(R/W)実行中 0x0003 (mc_state_ram_march_c): RAM チェック(March-C)実行中 0x0004 (mc_state_checksum): チェックサム実行中 0x0005 (mc_state_crc): CRC 実行中
説明	MEMCHECK の現在の状態を取得します。

seMemCheckFinish

構文	void seMemCheckFinish()
引数	なし
戻り値	なし
説明	MEMCHECK を終了します。

付録.B. プレイリストファイル

表 B.1 に、ESPER2 (EPSON 製音声作成ツール) で音声 ROM データと共に生成されるプレイリストファイルで指定できるパラメータのリストを示します。

表 B.1 プレイリストファイルで指定可能なパラメータリスト

パラメータ	説明
DIPSW_DEMO	デモモードを指定します。 デモモードは評価ボードの SW2-2/3/4 に割り当てられます。 SW2-2 SW2-3 SW2-4 0: OFF OFF OFF 1: OFF OFF ON … 7: ON ON ON
DIPSW_LANG	言語タイプを指定します。 言語タイプは評価ボードの SW3-2/3/4 に割り当てられます。 SW3-2 SW3-3 SW3-4 0: OFF OFF OFF 1: OFF OFF ON … 7: ON ON ON
ROM_ADRS	外部 QSPI Flash メモリに格納した音声 ROM データの開始アドレスを指定します。
ROM_SIZE	外部 QSPI Flash メモリに格納した音声 ROM データのサイズを指定します。
KEYCODE	ESPER2 で登録したキーコードを指定します。
PLAY_LIST_CH0	チャンネル 0 で再生するセンテンスのプレイリストを指定します。
PLAY_LIST_CH1	チャンネル 1 で再生するセンテンスのリストを指定します。
VOLUME_CH0	チャンネル 0 の音量の初期値を指定します。
VOLUME_CH1	チャンネル 1 の音量の初期値を指定します。
REPEAT_CH0	チャンネル 0 のリピート回数を指定します。
REPEAT_CH1	チャンネル 1 のリピート回数を指定します。
SPEED	音声速度の初期値を指定します。
STOP_TYPE_CH0	チャンネル 0 で実行する STOP コマンドのタイプを指定します。 0: 即時停止, 1:フレーズ再生終了後停止
STOP_TYPE_CH1	チャンネル 1 で実行する STOP コマンドのタイプを指定します。 0: 即時停止, 1:フレーズ再生終了後停止
PAUSE_TYPE_CH0	チャンネル 0 で実行する PAUSE コマンドのタイプを指定します。 0: 即時ポーズ, 1:フレーズ再生終了後ポーズ
PAUSE_TYPE_CH1	チャンネル 1 で実行する PAUSE コマンドのタイプを指定します。 0: 即時ポーズ, 1:フレーズ再生終了後ポーズ
MUTE_TYPE_CH0	チャンネル 0 で実行する MUTE コマンドのタイプを指定します。 0: 即時ミュート, 1:フレーズ再生終了後ミュート
MUTE_TYPE_CH1	チャンネル 0 で実行する MUTE コマンドのタイプを指定します。 0: 即時ミュート, 1:フレーズ再生終了後ミュート
AUTO_PLAY_CH0	チャンネル 0 のオートプレイフラグを指定します。 0: オートプレイなし 1:オートプレイあり
AUTO_PLAY_CH1	チャンネル 1 のオートプレイフラグを指定します。 0: オートプレイなし 1:オートプレイあり

付録.C. サンプルプロジェクトのコードサイズ概要

表 C.1 に、本サンプルソフトウェアに含まれる各サンプルプロジェクトを IAR EWARM または MDK-ARM でビルドした場合のコードサイズの一覧を示します。

表C.1 各サンプルプロジェクトのコードサイズ

サンプルプロジェクト名	IAR EWARM (コードサイズ16 KB以下)	MDK-ARM (コードサイズ32 KB.)
ADC12A	✔	✔
CLG	✔	✔
DMAC	✔	✔
Flash	✔	✔
EEPROM	✔	✔
I2C_S5U1C31D5xT1	✔	✔
PPORT	✔	✔
QSPI	✔	✔
QSPI_DMA	✔	✔
QSPI_MASTER	✔	✔
QSPI_SLAVE	✔	✔
REMC3	✔	✔
RFC	✔	✔
RTCA	✔	✔
SPIA_MASTER	✔	✔
SPIA_SLAVE	✔	✔
SVD3	✔	✔
T16	✔	✔
T16B	✔	✔
UART3	✔	✔
WDT2	✔	✔
SOUNDPLAY	✔	✔
MEMCHECK	✔	✔
SOUNDPLAY_DEMO	-	-
IEC 60730 compliant	✔	✔

注: 統合開発環境のバージョンやビルドオプションの設定によって、コードサイズが表中のサイズを超える可能性があります。

付録.D. 自己診断サンプルソフトウェア仕様

s1c31TestRam16

構文	short s1c31TestRam16 (　unsigned short　*chkAddr, 　unsigned short　chkNum)
引数	*chkAddr 　テスト対象 RAM の先頭アドレス chkNum 　テスト対象データサイズ
戻り値	RAM の R/W テストの結果 　0x0000 (E_OK):　書き込んだ値と差異なし 　0x0001 (E_MEMORY):　書き込んだ値と差異あり
説明	chkAddr が指すアドレスから chkNum の数だけ、メモリに 0xaa55（0x55aa）を書き込みます。その後、読み込みを行い、書き込んだ値と比較し、差異がなければ E_OK、差異があれば E_MEMORY を返します。
注意事項	chkAddr の最下位ビットは常に 0 として扱われます。指定されたメモリ領域がスタック領域と重複したとき、動作は保障されません。s1c31TestRam16 は chkNum ×2 バイト分のメモリチェックを行います。

s1c31TestRam8

構文	short s1c31TestRam8 (　unsigned char　*chkAddr, 　unsigned short　chkNum)
引数	*chkAddr 　テスト対象 RAM の先頭アドレス chkNum 　テスト対象データサイズ
戻り値	RAM の R/W テストの結果 　0x0000 (E_OK):　書き込んだ値と差異なし 　0x0001 (E_MEMORY):　書き込んだ値と差異あり
説明	chkAddr が指すアドレスから chkNum の数だけ、メモリに 0xa5（0x5a）を書き込みます。その後、読み込みを行い、書き込んだ値と比較し、差異がなければ E_OK、差異があれば E_MEMORY を返します。
注意事項	指定されたメモリ領域がスタック領域と重複したとき、動作は保障されません。s1c31TestRam8 は chkNum×1 バイト分のメモリチェックを行います。

s1c31TestRegister

構文	short s1c31TestRegister (void)
引数	void
戻り値	Register の R/W テストの結果 　0x0000 (E_OK):　書き込んだ値と差異なし 　0x0002 (E_REGISTER):　書き込んだ値と差異あり
説明	CPU 内の汎用レジスタに 0x555555 と 0xaaaaaa を書き込みます。その後、読み込みを行い、書き込んだ値と比較します。同様にスタックポインタに 0x555554 と 0xaaaaa8 を書き込みテストし、差異がなければ E_OK、差異があれば E_REGISTER を返します。

s1c31TestPsr

構文	short s1c31TestPsr (void)
引数	void
戻り値	Register の R/W テストの結果 0x0000 (E_OK): 書き込んだ値と差異なし 0x0002 (E_REGISTER): 書き込んだ値と差異あり
説明	CPU 内のステータスレジスタに 0x55 と 0xaa を書き込みます。その後、読み込みを行い、書き込んだ値と比較します。差異がなければ E_OK、差異があれば E_REGISTER を返します。

s1c31TestRamMarchc

構文	short s1c31TestRamMarchc (unsigned char *chkAddr, unsigned short chkNum)
引数	*chkAddr テスト対象 RAM の先頭アドレス chkNum テスト対象データサイズ
戻り値	RAM の R/W テストの結果 0x0000 (E_OK): 書き込んだ値と差異なし 0x0001 (E_MEMORY): 書き込んだ値と差異あり
説明	chkAddr が指すアドレスから chkNum の数だけ、March-c テストを実施します。問題がなければ E_OK、問題があれば E_MEMORY を返します。
注意事項	指定されたメモリ領域がスタック領域と重複したとき、動作は保障されません。テスト範囲となったメモリは、0x00 に書き換えられます。

s1c31TestChksum

構文	short s1c31TestChksum (unsigned char *chkAddr, unsigned short chkNum)
引数	*chkAddr テスト対象 RAM の先頭アドレス chkNum テスト対象データサイズ
戻り値	チェックサム算出結果
説明	chkAddr が指すアドレスから chkNum の数だけメモリの値を読み、その総和を返します。

s1c31TestCrc

構文	short s1c31TestCrc (unsigned char *chkAddr, unsigned short chkNum)
引数	*chkAddr テスト対象 RAM の先頭アドレス chkNum テスト対象データサイズ
戻り値	CRC 算出結果
説明	chkAddr が指すアドレスから chkNum の数だけメモリの値を読み、CRC の計算を行い、その結果を返します。

s1c31TestCrcTbl

構文	short s1c31TestCrcTbl (unsigned char *chkAddr, unsigned short chkNum)
引数	*chkAddr テスト対象 RAM の先頭アドレス chkNum テスト対象データサイズ
戻り値	CRC 算出結果
説明	chkAddr が指すアドレスから chkNum の数だけメモリの値を読み込みます。その後、テーブル(CRC-CCITT テーブル)を参照して CRC の計算を行い、その結果を返します。

s1c31d5xTestInterrupt

構文	short s1c31d5xTestInterrupt (unsigned short numInt)
引数	numInt CLG の 16 ビットタイマで発生する割り込み指定回数
戻り値	発生した割り込み回数
説明	numInt の回数分だけ割り込み(CLG の 16 ビットタイマを使用)が発生するまでに、割り込み(P13 (SW7)割り込みを使用)が何回発生するかをカウントする。

s1c31d5xTestClk

構文	int s1c31d5xTestClk (unsigned long baseFreq unsigned short range)
引数	baseFreq 使用している理想的なメインクロック(Hz) Range 許容誤差(%)
戻り値	クロックテストの結果 0x0000 (E_OK): テスト結果が許容誤差範囲内 0x0003 (E_CLOCK): テスト結果が許容誤差範囲外
説明	使用しているメインクロックが許容誤差範囲内で動作しているかを確認する。 baseFreq で理想的な周波数の値を指定し、range で許容誤差（%）を指定する。 テスト結果が許容誤差範囲内ならば、E_OK を返し、範囲外ならば E_CLOCK を返す。

改訂履歴表

Rev. No.	日付	ページ	種別	改訂内容（旧内容を含む） および改訂理由
Rev.1.00	2018/11/10	全ページ	新規	新規制定
Rev.1.10	2018/01/18	-	削除	2.2.3節を削除
		4	修正	2.2.1節を修正
		8,16	修正	セットアップユーティリティ「ToolchainSetup.exe」のUI変更に伴い修正
		12,21	修正	2.3.5節と2.4.5節を修正
		66.67	修正	付録.A.を修正
Rev.2.00	2018/02/26	4,6,14	修正	1.1節, 2.2.1節, 2.3.1節, 2.3.5節, 2.4.1節, 2.4.5節を修正
		56	追加	3.24節を追加
Rev.3.00	2020/06/30	全ページ	修正	D50をD5xに修正
		52-57	修正	3.20節(SOUNDPLAY)を更新
		62-67	修正	4章(デモソフトウェア)を更新
		68-70	追加	5章(自己診断サンプルソフトウェア)を追加
		71-74	追加	付録.A(HW Processorライブラリ仕様)に関数を追加
		78-80	追加	付録.D(自己診断サンプルソフトウェア仕様)を追加

セイコーエプソン株式会社

営業本部　デバイス営業部

東京　〒160-8801 東京都新宿区新宿 4-1-6 JR 新宿ミライナタワー29F

大阪　〒530-6122　大阪市北区中之島 3-3-23　中之島ダイビル 22F

ドキュメントコード：413732403

2018 年 11 月 作成

2019 年 2 月 改訂

2020 年 6 月 改訂