CMOS 32-BIT SINGLE CHIP MICROCONTROLLER

S1C31D5x
Peripheral Circuit
Sample Software Manual

arm

Evaluation board/kit and Development tool important notice

1. This evaluation board/kit or development tool is designed for use for engineering evaluation, demonstration, or development purposes only. Do not use it for other purposes. It is not intended to meet the requirements of design for finished products.
2. This evaluation board/kit or development tool is intended for use by an electronics engineer and is not a consumer product. The user should use it properly and in a safe manner. Seiko Epson dose not assume any responsibility or liability of any kind of damage and/or fire coursed by the use of it. The user should cease to use it when any abnormal issue occurs even during proper and safe use.
3. The part used for this evaluation board/kit or development tool may be changed without any notice.

NOTICE

Table of Contents

1. Overview

This manual describes how to use the example software provided for the S1C31D5x microcontroller and shows the expected output when running the example software.

The example software is included in the S1C31D5x Sample Software and Peripheral Drivers software package. It is intended to demonstrate how to use the various peripheral circuits in the S1C31D5x microcontroller.

For detailed information on the S1C31D5x microcontroller, refer to the “S1C31D5x Technical Manual”.

The S1C31D5x Peripheral Circuit Sample Software Package includes:

- Peripheral Library modules
- Example Software
- Product-specific files such as the S1C31D5x hardware definition file
- System viewer description file
- Flash loader

The goal of the Peripheral Sample Software is to show how to use the Peripheral Library modules in order to control a S1C31D5x peripheral. Each Example demonstrates features of the selected peripheral and exercises various peripheral modes.

The S1C31D5x hardware definition file defines the hardware register addresses, access sizes, and access read-write types. The individual registers and their fields are accessed based on the described S1C31D5x hardware definition file. The Peripheral Library also provides easy to use methods which perform complex peripheral functions. Those methods include functions such as peripheral initialization and peripheral feature management.

*1 Refer to this manual along with the “S1C31D50/D51 Technical Manual” and the “S5U1C31D5xT1 manual”, “S5U1C31D51T2 manual" available from Seiko Epson's website.

*2 This manual covers the sample software package of ver.2.00 or later.

1.1 Operating Environment

Before running the S1C31D5x sample software, prepare the following components:

- Evaluation Board
  - S5U1C31D5xT1 evaluation board equipped with S1C31D5x and S5U1C31D51T2 (only S1C31D51)
- Debug Probes *1
  - IAR Systems I-jet or SEGGER J-Link
- Integrated Development Environment
  - IAR Embedded Workbench for ARM® (IAR EWARM) or MDK-ARM®
- Other Devices (option)
  - An USB Adapter for UART

*1: I-jet is available only with IAR EWARM. J-Link is available for both IAR EWARM and MDK-ARM.

2 Sample Software Operations

2.1 Sample Software Components

The S1C31D5x Sample Software demonstrates how to call the Peripheral Library modules which interface hardware components on S1C31D5x. Each peripheral module has an associated library module that consists of a source file ("xxx.c") and an include file ("xxx.h"). The include file describes the constants, types, and functions for each library module.

The S1C31D5x Sample Software is organized in the "Examples" folder. In most cases, the peripheral module is presented in one corresponding example project. For example, the CLG (Clock Generator) project focuses on the CLG peripheral module. For more complex modules, several example projects may be available to feature different aspects of the module. An example of this is the QSPI module which is presented in a few projects (each project begins with the QSPI prefix). The relationship between different layers of the software is shown in Figure 2.1.1.

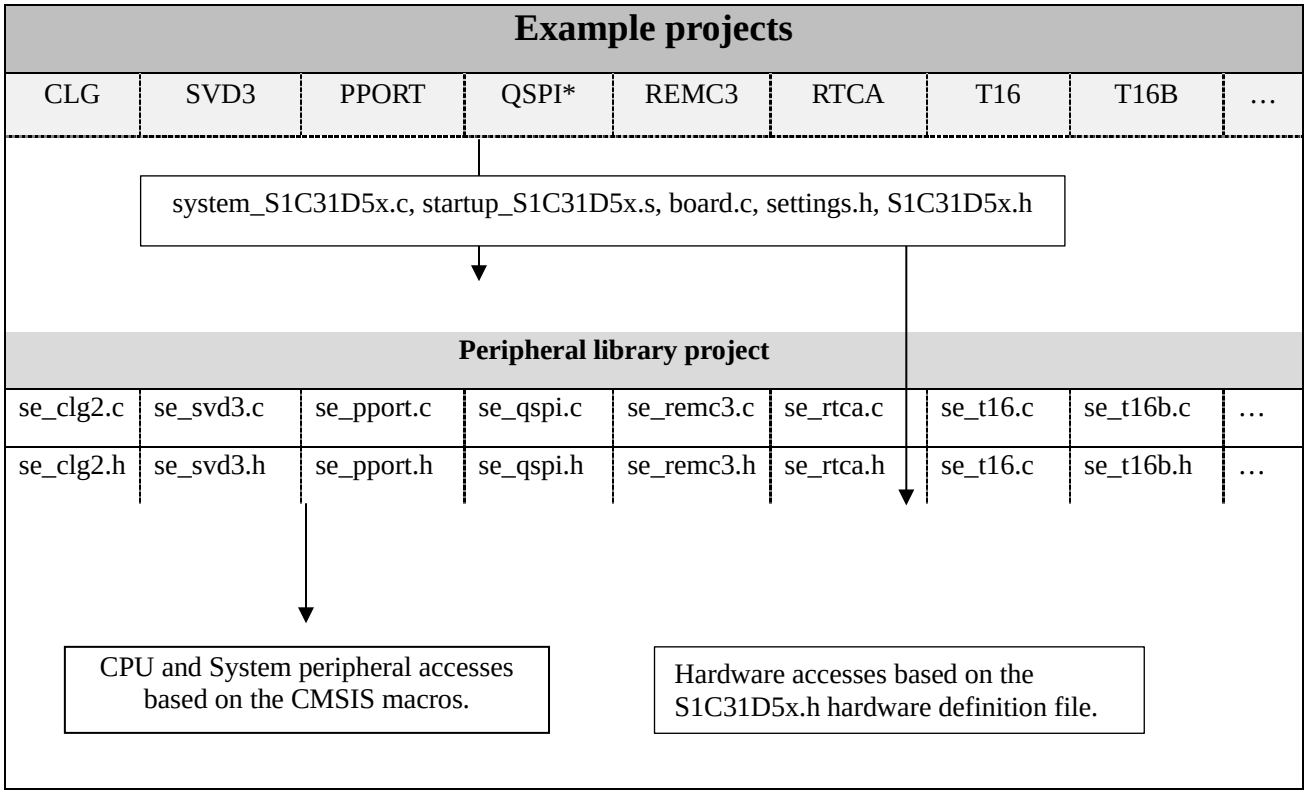


Figure 2.1.1 S1C31D5x Example Software Workspace

Notes:

- All example projects share the IAR/Keil startup files and IAR/Keil system files located in the "CMSIS¥Device" folder.
- All example projects also share the board-related files in the "board" folder.
- The board.c file implements the boot-up system initialization procedure in the BoardInit function.
- The BoardInit function is called from the SystemInit function. The BoardInit function sets the CPU frequency, Sys Tick value, and priority, etc.
- The SystemInit function is called from the IAR/Keil startup file.

The settings.h file is where the default boot-up behavior is changed by selecting the appropriate defines. The meaning of each setting is described in Table 2.1.1.

Table 2.1.1 S1C31D5x Board Defines

Configuration Feature	Defined	Un-defined
UART_PRINTF	The standard library printf function outputs to the UART console. Additional hardware is required. For Keil IDE, this configuration feature should be defined.	The semihosting library printf function outputs to the IAR IDE terminal window.
EXECUTE_ON_OSC3	The CPU switches to the clock that optionally can be run from crystal or internal trimmable oscillator in the BoardInit function.	The CPU uses the default clock, IOSC.
OSC3_AUTOTRIMMING_ON	OSC3 trimming is performed in the BoardInit() function to achieve higher CPU clock accuracy.	OSC3 trimming is not performed. This results in a shorter boot time.
OSC3_SRC_XTAL	Crystal is source of OSC3	Internal RC oscillator is source of OSC3
CACHE_ENABLED	The cache is enabled for the Flash targets. No importance for Debug targets,	The cache is not enabled for the Flash targets.
TICKLESS_ENABLED	SYSTICK interrupt is disabled.	SYSTICK interrupt is used to keep track of the elapsed CPU time since the last boot.
BOOT_LOADER	Executing code is a boot loader.	Executing code is an application.
QSPI_MODE_SINGLE	QSPI SINGLE mode is selected.	QSPI DUAL or QUAD modes are selected.
SPIA_DMA	Use DMA for SPIA_MASTER and SPIA_SLAVE projects.	Do not use DMA for SPIA_MASTER and SPIA_SLAVE projects.

Notes:

- By default, all board features are un-defined.
- To use the UART console for input/output, un-comment UART_PRINTF in the settings.h file.
- In cases where the CPU deep sleep function is used, un-comment TICKLESS_ENABLED in the settings.h file.

2.2 Preparation for Program Download and Execution

2.2.1 Hardware Connections

To use and debug the Sample Software the following hardware components are recommended.

- S5U1C31D5xT1 evaluation board
- Debug probes (IAR Systems I-jet or SEGGER J-Link)

For details of the hardware connection, refer to the “S5U1C31D5xT1 Manual”.

2.2.2 Connection with USB adapter for UART

USB adapter for UART is used in the sample program that uses UART. By connecting the S5U1C31D5xT1 evaluation boat to the PC using the USB adapter for UART, UART communication with the PC becomes possible. The picture in Figure 2.2.2.1 and the diagram in Figure 2.2.2.2 show the connection for an USB Adapter for UART to the S5U1C31D5xT1 evaluation board.

To perform UART communication, it is necessary to build the sample program with the definition of UART_PRINTF in the *settings.h* file enabled (see Section 2.1 and Table 2.1.1). In addition, it is necessary to start the serial communication terminal software on the PC and set the serial port. Table 2.2.2.1 shows the serial port setting values.

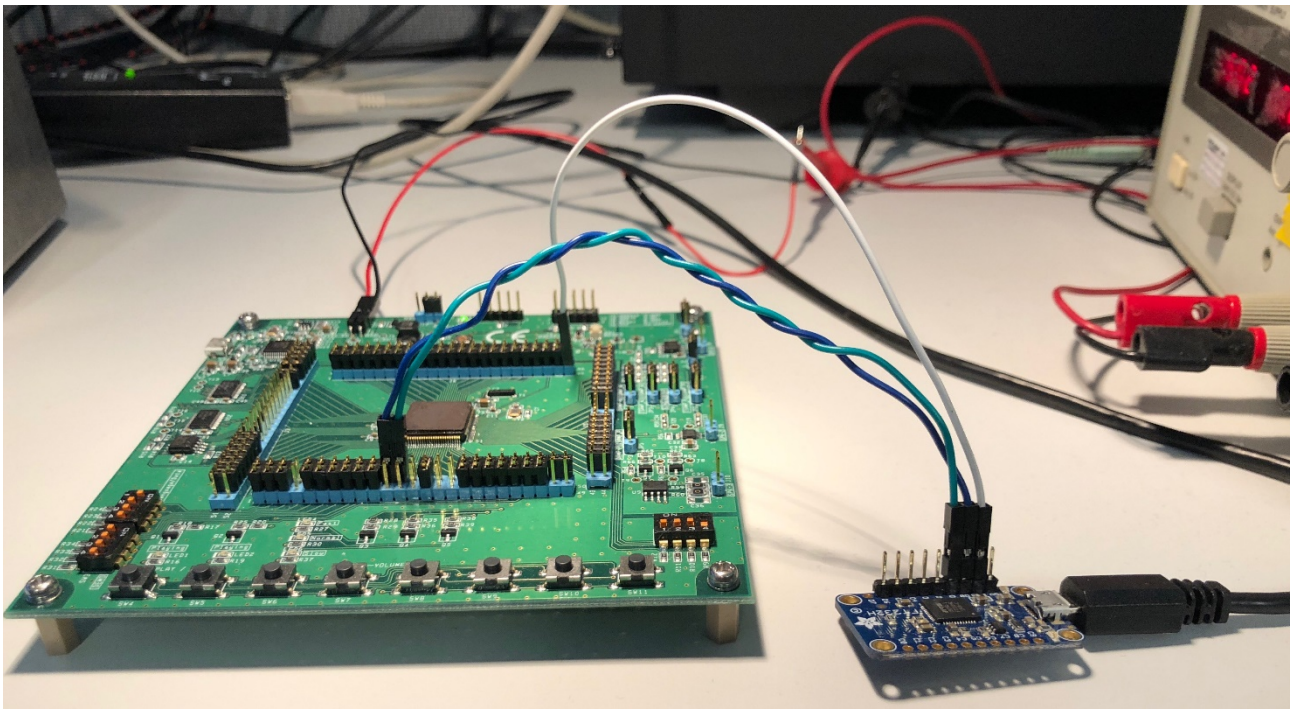


Figure 2.2.2.1 USB Adapter for UART Connection Example to S5U1C31D5xT1 Evaluation Board

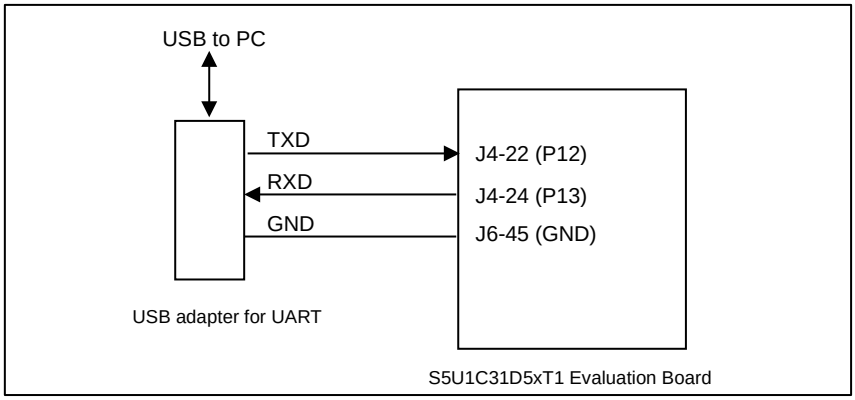


Figure 2.2.2.2 USB Adapter for UART Wiring to S5U1C31D5xT1 Evaluation Board

Table 2.2.2.1 Serial Port Settings for Terminal Software

Parameter	Setting Value
Baud rate	115200 bps
Data	8 bits
Stop bits	1 bits
Parity	None

For more information on S5U1C31D5xT1 evaluation board, refer to the “S5U1C31D5xT1 Manual”.

Notes: USB adapter for UART used in Figure 2.2.2.1 is a commercial product, not provided by Seiko Epson. Please purchase as necessary.

2.3 IAR EWARM IDE Sample Software Procedures

2.3.1 Software Setup

Before executing the sample software using the IAR EWARM, you need to set up the sample software. To set up the sample software, follow the procedure below.

(1) Download the sample software

Download the S1C31D5x Peripheral Circuit Sample Software Package (.exe) from Seiko Epson microcontroller web site.

(2) Install the Peripheral Circuit Sample Software Package

Close all other programs during this installation. Run downloaded Peripheral Circuit Sample Software Package executable file as administrator. Review the license terms before proceeding with installation. Select desired folder package to be installed into and select “Install”. After the software has been installed into the destination, a setup utility called “ToolchainSetup.exe” will automatically launch. Press “Next” on setup utility, and select your preferred versions of IAR EWARM identified by the checkboxes. [Note: The setup utility will copy some flashing binaries and device description files into your IAR folders]. Press “Next” to begin the setup, and when the “Done” message appears select “Finish” to complete.

(3) Launch the IAR EWARM

Once the software setup has completed successfully, launch the IAR EWARM.

For the version of IAR EWARM used to evaluate this sample software and more information on the setup utility, refer to the “README_IAR.txt” found in this sample software package.

2.3.2 Workspace Open

The sample software package has a collection of sample software projects in the “Examples” folder. All of the samples software projects build against the sePeripheralLibrary.

To open a workspace which contains all the sample software projects, select the [File] > [Open] > [Workspace...] in the IAR EWARM menu, navigate to “Examples¥WORKSPACE¥S5U1C31D5xT1¥IAR” folder, and open the “Examples.eww” file.

Optionally, each sample software project subfolder has a “board¥S5U1C31D5xT1¥IAR” subfolder which contains an IAR project file (.ewp) and an IAR workspace file (.eww). To open the workspace for an individual sample software project, select the [File] > [Open] > [Workspace] in the IAR EWARM menu, navigate to the project’s “board¥S5U1C31D5xT1¥IAR” folder, and open the workspace file (.eww).

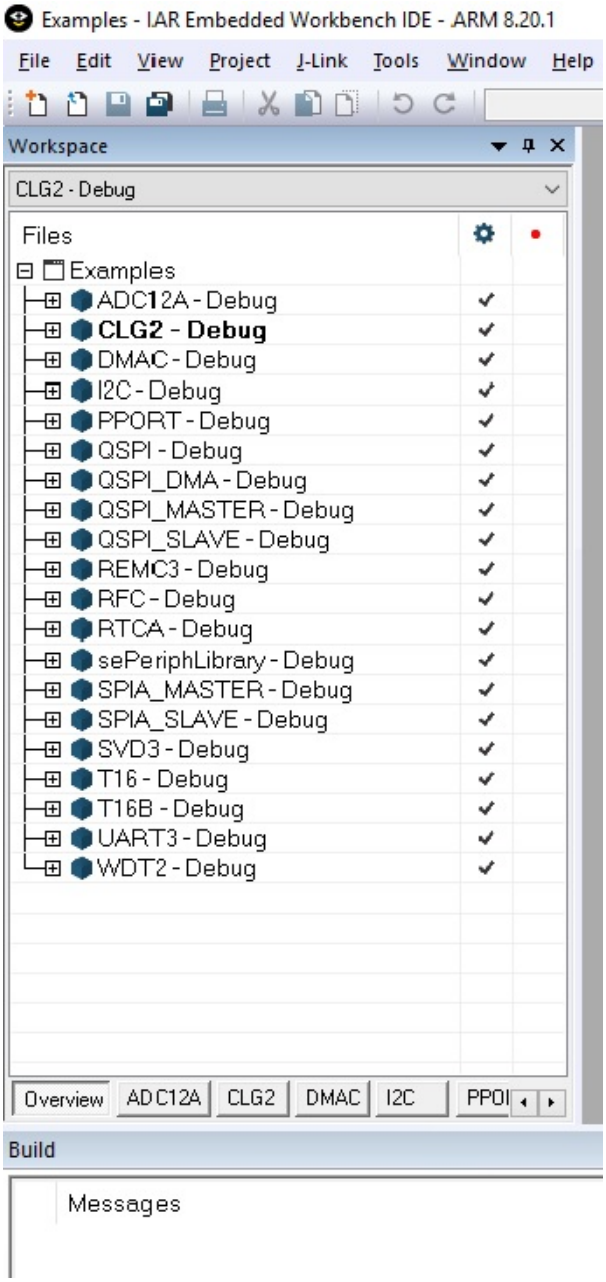


Figure 2.3.2.1 Example of Multi-Project Workspace

2.3.3 Active Project Selecting

To build the sample software project, right-click the target project to be built and executed in [Workspace] window on IAR EWARM and select the [Set as Active] in right-clicked menu (Figure 2.3.3.1). By using the drop-down list at the top of the [Workspace] window, the active project and build configuration can be selected at the same time (Figure 2.3.3.1).

As shown below, each sample software project contains three build configurations.

- Debug - Build configuration to execute code in internal RAM memory
  The optimization level is set to “low”.
- DebugFlash - Build configuration to execute code in internal flash memory
  The optimization level is set to “low”.
- ReleaseFlash - Build configuration to execute code in internal flash memory
  The optimization level is set to “high”.

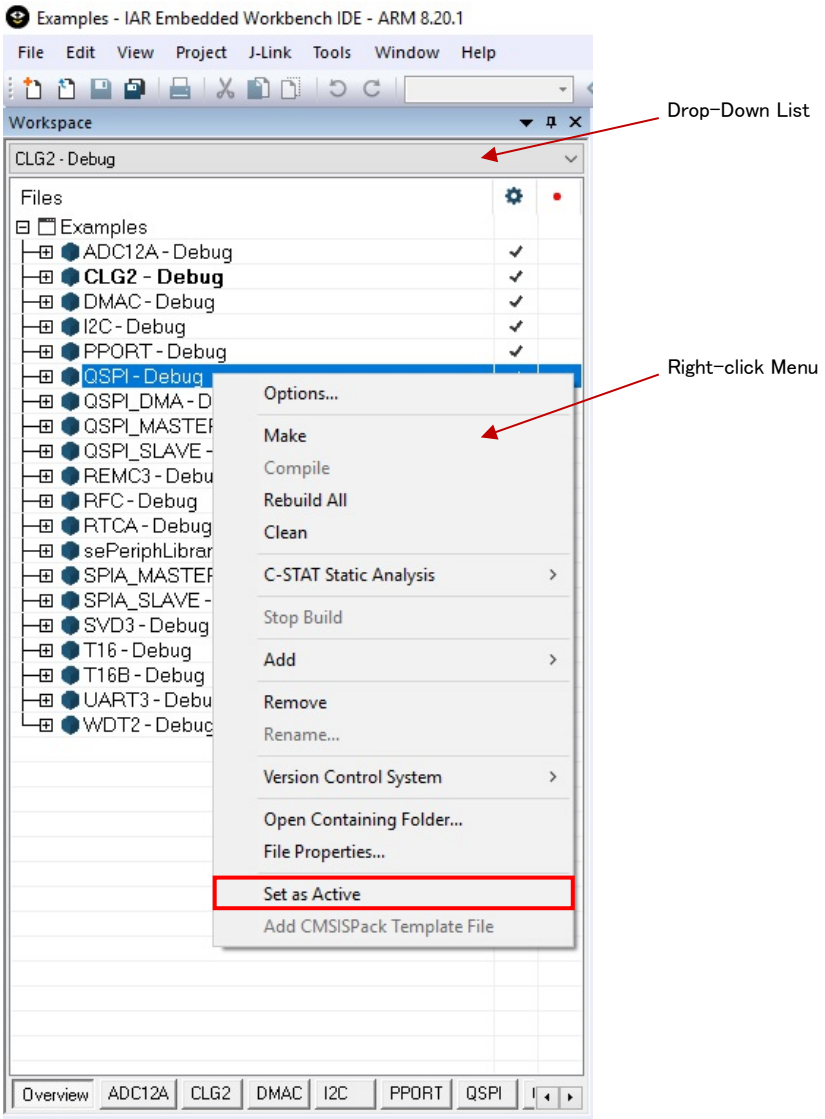


Figure 2.3.3.1 Active Project Setting

2.3.4 Debug Probe Setting

Before debugging an active project using the target board connected to the debug probe, you need to select a driver for the debug probe, if necessary.

To select the debug probe driver, follow the procedure below.

(1) Select the [Project] > [Option] in the IAR EWARM menu.

(2) Select the [Debugger] in the [Category] list on the [Options for node "{project}"] dialog (Figure 2.3.4.1).

(3) Select the [Setting] tab, and then select the debug probe in the [Driver] drop-down list as shown below (Figure 2.3.4.1).

- When using the I-jet , select the "I-jet/JTAGjet".

- When using the J-Link, select the "J-Link/J-Trace".

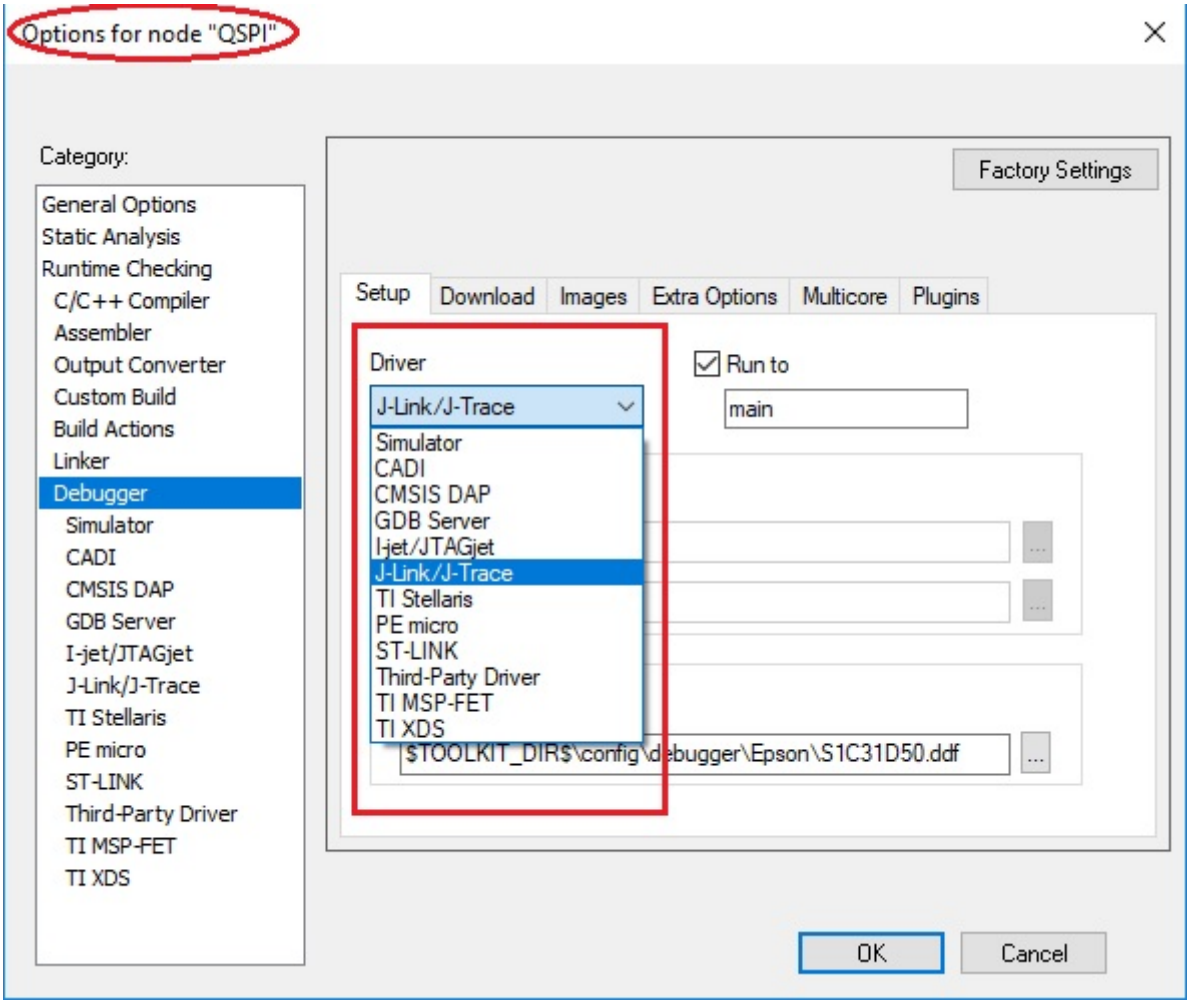


Figure 2.3.4.1 Debug Probe Setting

2.3.5 Flash Loader Setting

When the build configuration of the active project is “DebugFlash” or “ReleaseFlash”, it is necessary to set a flash loader to load the program in the internal flash memory. However, when the build configuration is “Debug”, it is necessary to unset a flash loader because the program is executed on the internal RAM.

To set the flash loader, follow the procedure below.

- When the build configuration is “Debug” (program execution in RAM)

(1) Select the [Project] > [Option] in the IAR EWARM menu.

(2) Select the [Debugger] in the [Category] list on the [Options for node “{project}”] dialog (Figure 2.3.5.1).

(3) Select the [Download] tab (Figure 2.3.5.1).

(4) Disable the [Use flash loader(s)] checkbox (Figure 2.3.5.1).

- When the build configuration is “DebugFlash” or “ReleaseFlash” (program execution in internal flash memory)

(1) Select the [Project] > [Option] in the IAR EWARM menu.

(2) Select the [Debugger] in the [Category] list on the [Options for node “{project}”] dialog (Figure 2.3.5.1).

(3) Select the [Download] tab (Figure 2.3.5.1).

(4) Enable the [Use flash loader(s)] checkbox (Figure 2.3.5.1).

(5) Enable the [Override default .board file] checkbox (Figure 2.3.5.1).

(6) Click the [...] button and select “S1C31D5x_int.board” as a board file (Figure 2.3.5.1).

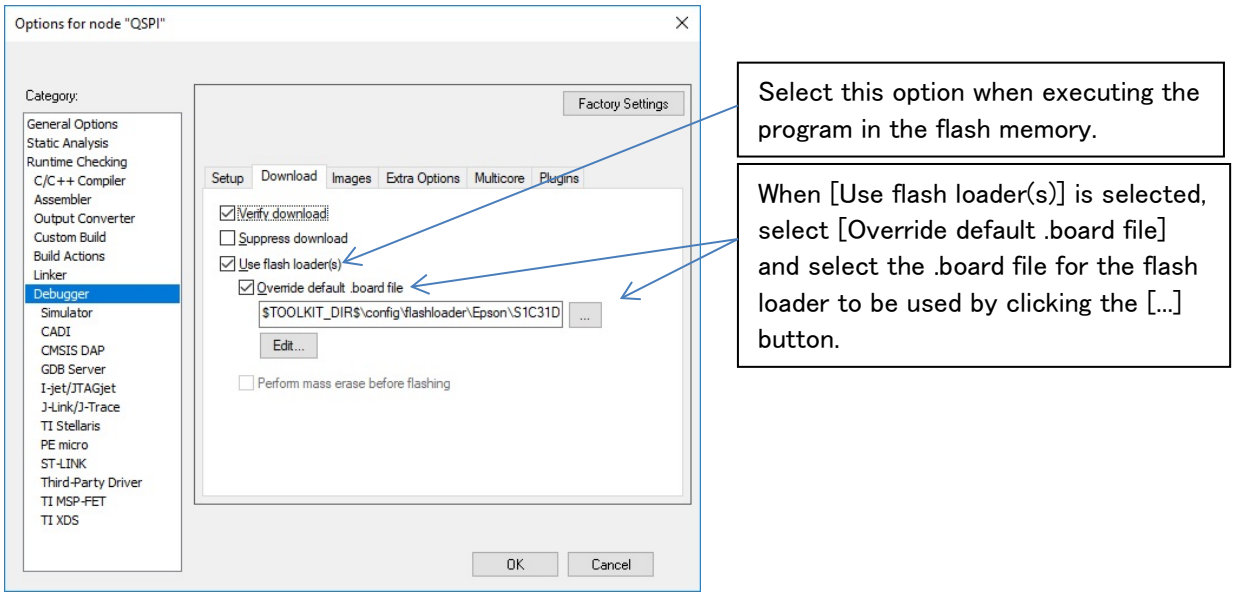


Figure 2.3.5.1 Flash Loader Setting

2.3.6 Project Build

To build an active project, select one of the build commands [Make] and [Rebuild All] from the [Project] in the IAR EWARM menu (Figure 2.3.6.1).

Note: If linker errors occur in the build, there is a possibility that the library project “sePeriphLibrary” is not built. Build this library project with the same build configuration as the active project, and then build the active project again. For example, to build the active project “QSPI-Debug”, build “sePeriphLibrary-Debug”.

Also, the batch build option to build all the projects included in the sample software at once is available. To use the batch build option, select the [Project] > [Batch build…] in the IAR EWARM menu (Figure 2.3.6.1). And then select the desired build configuration on the displayed dialog box and click the [Make] or [Rebuild All] button(Figure 2.3.6.2).

The batch build option is available to build the following build configurations.

- all_Debug - built debug targets to execute code in internal RAM memory
- all_DebugFlash - built debug targets to execute code in internal Flash memory
- all_ReleaseFlash - build release targets to execute code in internal Flash memory

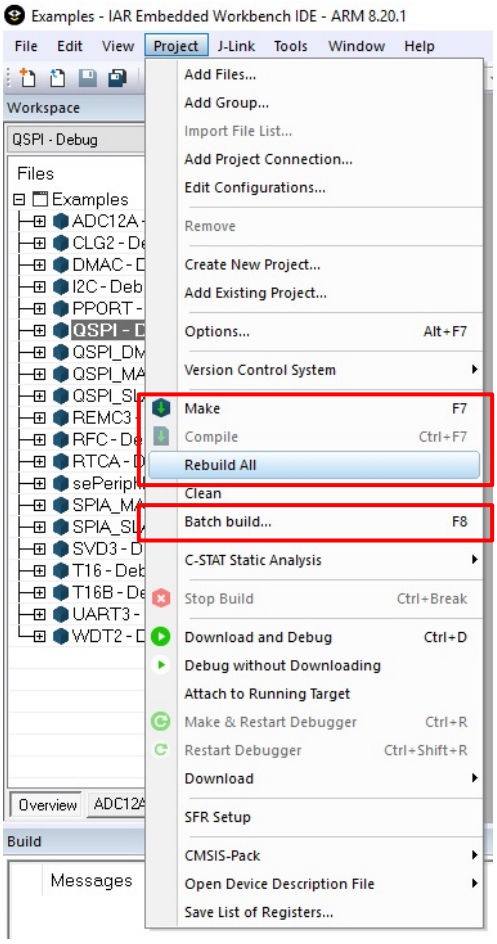


Figure 2.3.6.1 Build Commands

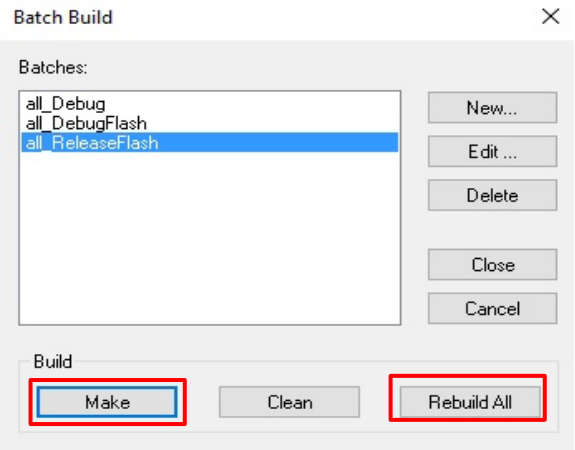


Figure 2.3.6.2 Batch Build

2.3.7 Project Download and Debug

Following a successful build, download the program image of the active project to the target board. To download the program image, select the [Project] > [Download and Debug] in IAR EWARM menu (Figure 2.3.7.1).

When the active project is “Debug” build, the program image is loaded in the internal RAM and debugging is started. When the active project is “DebugFlash” build or “ReleaseFlash” build, the program image is loaded in the internal flash memory and debugging is started.

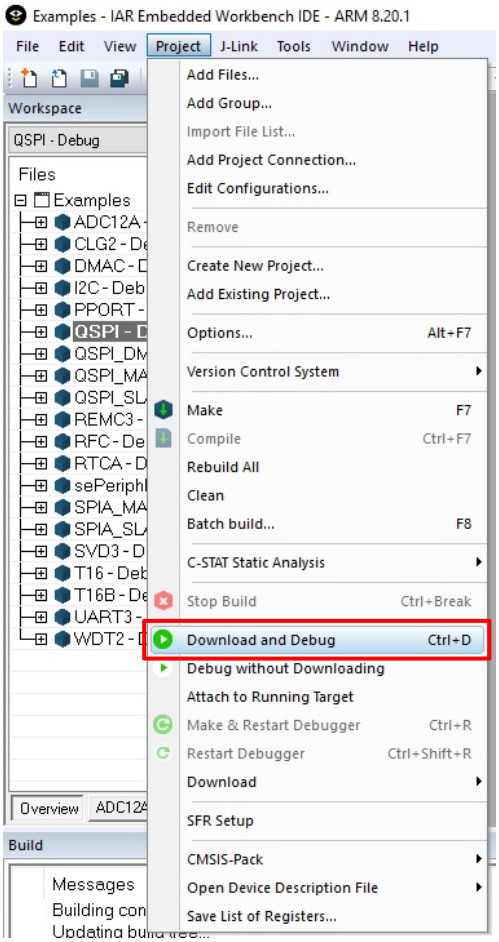


Figure 2.3.7.1 Download and Debug

2.4 KEIL MDK-ARM (µVision) Sample Software Procedures

2.4.1 Software Setup

Before executing the sample software using the MDK-ARM (µVision), you need to set up the sample software. To set up the sample software, follow the procedure below.

(1) Download the sample software

Download the S1C31D5x Peripheral Circuit Sample Software Package (.exe) from Seiko Epson microcontroller web site.

(2) Install the Peripheral Circuit Sample Software Package

Close all other programs during this installation. Run downloaded Peripheral Circuit Sample Software Package executable file as administrator. Review the license terms before proceeding with installation. Select desired folder package to be installed into and select “Install”. After the software has been installed into the destination, a setup utility called “ToolchainSetup.exe” will automatically launch. Press “Next” on setup utility and select your preferred versions of µVision identified by the checkboxes. [Note: The setup utility will copy some flashing binaries into your µVision folders] Select preferred version of a debugger from the list of supported debug probes. [Note: The setup utility will copy debug configuration files into your work space] During installation workspace directory will be set automatically. Press “Next” to begin the setup, and when the “Done” message appears select “Finish” to complete.

(3) Launch the µVision

Once the software setup has completed successfully, launch the µVision.

For the version of MDK-ARM(µVision) used to evaluate this sample software and more information on the setup utility, refer to the “README_KEIL.txt” found in this sample software package.

2.4.2 Workspace Open

The sample software package has a collection of sample software projects in the “Examples” folder. All of the samples software projects build against the sePeripheralLibrary.

To open a workspace which contains all the sample software projects, select the [Project] > [Open Project…] in the µVision menu, navigate to “Examples¥WORKSPACE¥S5U1C31D5xT1¥ARM” folder, and open the “Examples.eww” file.

Optionally, each sample software project subfolder has a “board¥S5U1C31D5xT1¥ARM” subfolder which contains a µVision project file (.uvprojx) and a µVision multi-project workspace file (.uvmpw). To open the workspace for an individual sample software project, select the [Project] > [Open Project…] in µVision menu, navigate to the project’s “board¥S5U1C31D5xT1¥ARM” folder, and open the workspace file (.uvmpw).

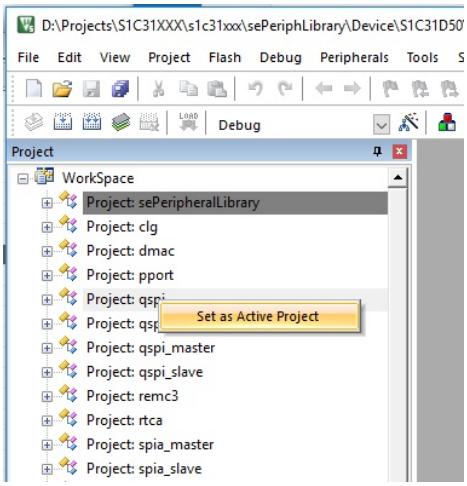


Figure 2.4.2.1 Example of Multi-Project Workspace

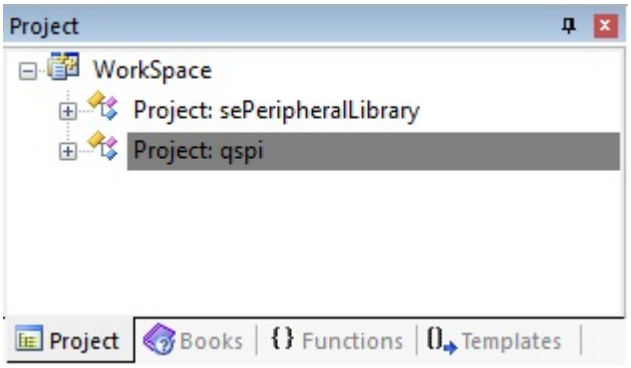


Figure 2.4.2.2 Example of Individual Project Workspace

2.4.3 Active Project Selecting

To build the sample software project, select a target project to be built and executed. Right-click the target project in [Project] window on µVision and select the [Set as Active Project] in right-clicked menu (Figure 2.4.3.1). Next, select the build configuration listed in the drop-down list on the tool bar of µVision (Figure 2.4.3.2).

As shown below, each sample software project contains two build configurations.

- Debug - Build configuration to execute code in internal RAM memory
- DebugFlash - Build configuration to execute code in internal flash memory

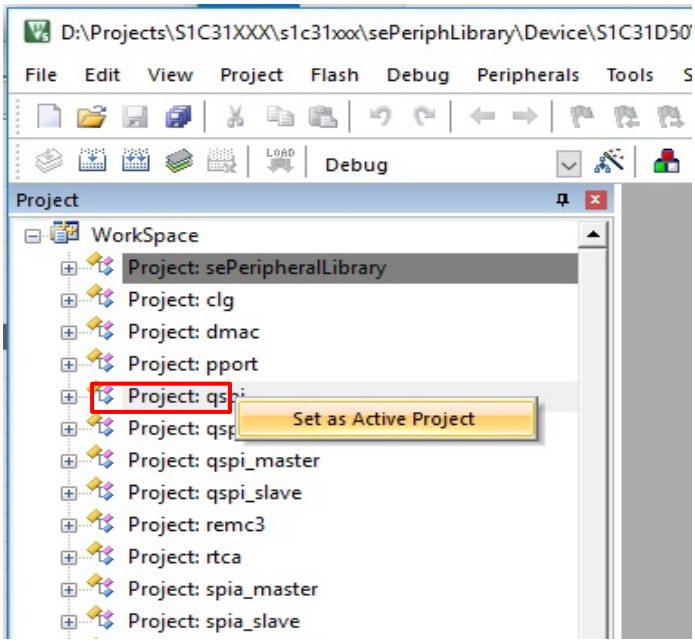


Figure 2.4.3.1 Active Project Setting

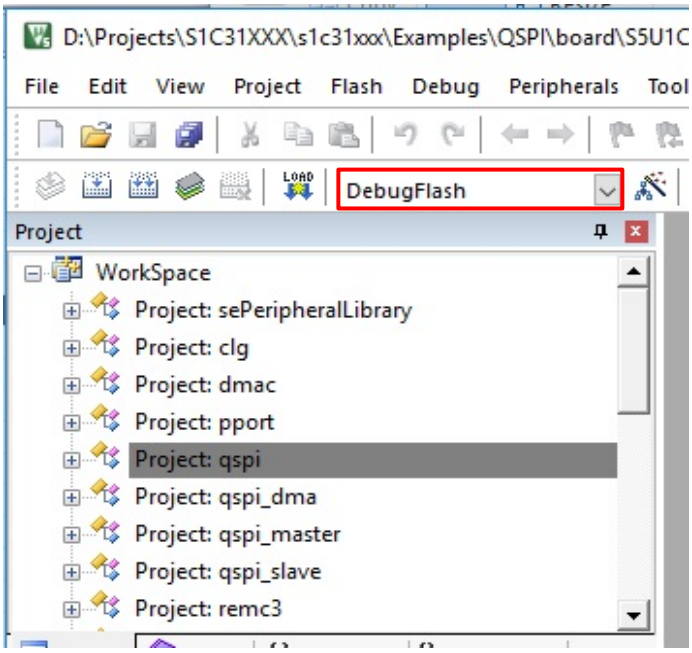


Figure 2.4.3.2 Selection of Build Configuration

2.4.4 Debug Probe Setting

Before debugging an active project using the target board connected to the debug probe, you need to select a driver for the debug probe, if necessary.

To select the debug probe driver, follow the procedure below.

(1) Select the [Project] > [Options for {project} - Target '{build configuration}'] in the µVision menu.

(2) Switch the [Debug] tab in the [Options for Target '{build configuration}'] dialog (Figure 2.4.4.1).

(3) Select the "J-Link/J-TRACE Cortex" from the drop-down list at the right side of [Use:] checkbox (Figure 2.4.4.1).

(4) Click the [Settings] button at the right side of the above drop-down list (Figure 2.4.4.1).

(5) Select the [SW] from the [Port:] drop-down list in the [Cortex JLink/JTrace Target Driver Setup] dialog box.

(6) Click the all [OK] button to close all dialogs.

Notes: This setting needs to be done with J-Link connected to the PC.

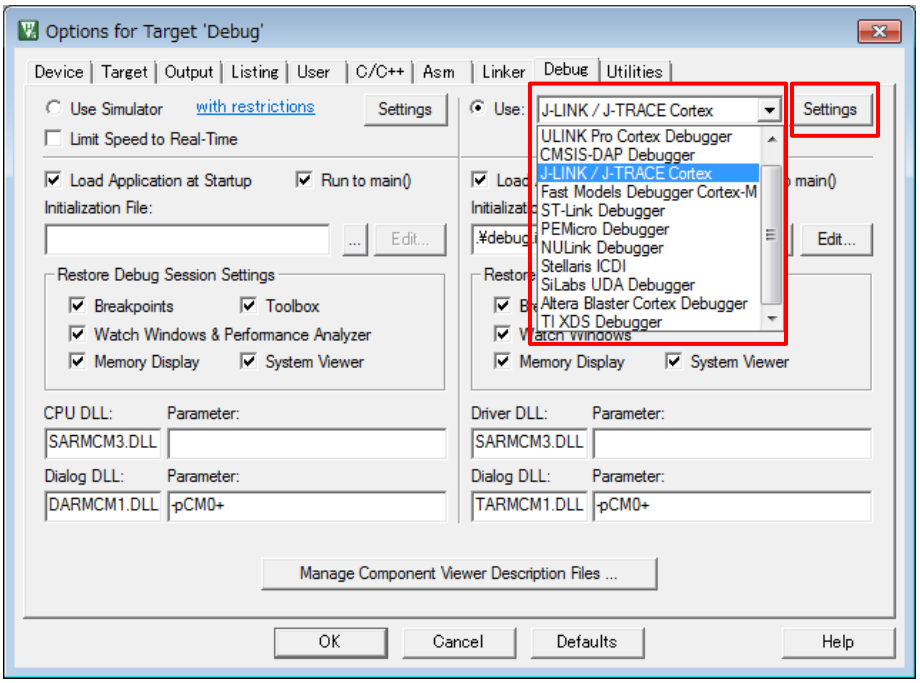


Figure 2.4.4.1 Debug Probe Selecting

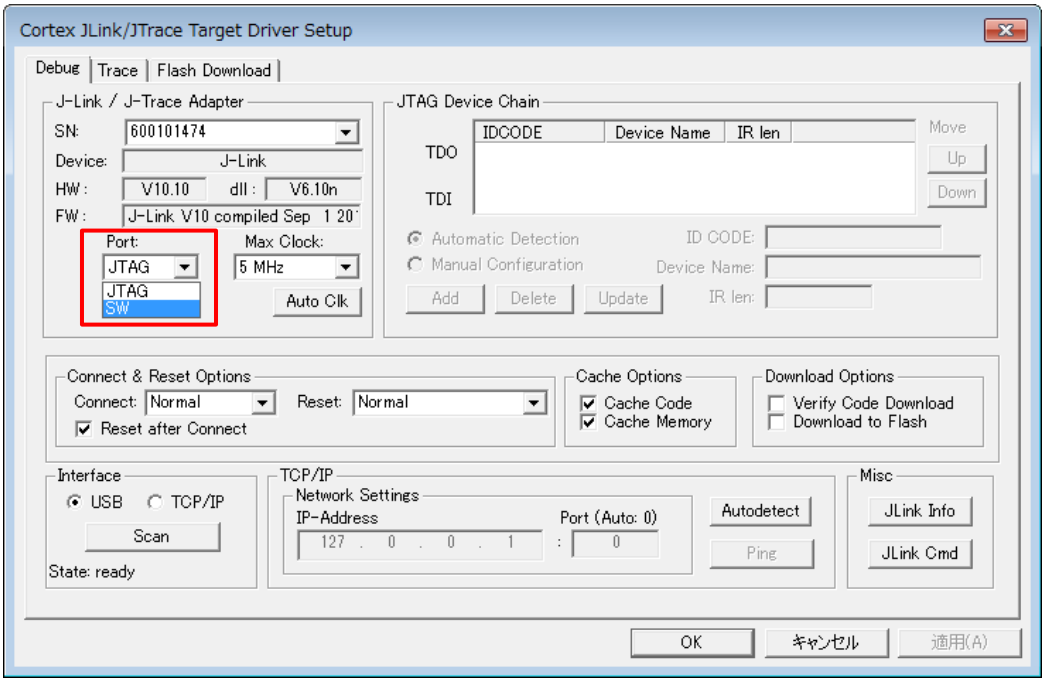


Figure 2.4.4.2 J-Link Driver Setup

2.4.5 Flash Loader Setting

When the build configuration of the active project is “DebugFlash”, it is necessary to set a flash loader to load the program in the internal flash memory. However, when the build configuration is “Debug”, it is necessary to unset a flash loader because the program is executed on the internal RAM.

To set the flash loader, follow the procedure below.

- When the build configuration is “Debug” (program execution on RAM)

(1) Select the [Project] > [Options for {project name} - Target ‘{build configuration}’] in the µVision menu.

(2) Switch the [Debug] tab in the [Options for Target ‘{build configuration}’] dialog.

(3) Click the […] button at the right side of the [Initialize File] edit box and select “debug.ini” file (Figure 2.4.5.1).

(4) Switch the [Utilities] tab in the [Options for Target ‘{build configuration}’] dialog and then click the [Settings] button in the [Configure Flash Menu Command] group box.

(5) Switch the [Flash Download] tab in the [Cortex JLink/JTrace Target Driver Setup] dialog.

(6) Delete all the flash loaders displayed in the [Programming Algorithm] list by clicking the [Remove] button.

(7) Enable the checkboxes, [Do not Erase] and [Reset and Run], in the [Download Function] group box, and disable other checkboxes (Figure 2.4.5.2).

- When the build configuration is “DebugFlash” (program execution on internal flash memory)

(1) Select the [Project] > [Options for {project name} - Target ‘{build configuration}’] in the µVision menu.

(2) Switch the [Debug] tab in the [Options for Target ‘{build configuration}’] dialog.

(3) Leave blank the [Initialize File] edit box.

(4) Switch the [Utilities] tab in the [Options for Target ‘{build configuration}’] dialog and then click the [Settings] button in the [Configure Flash Menu Command] group box.

(5) Switch the [Flash Download] tab in the [Cortex JLink/JTrace Target Driver Setup] dialog.

(6) Delete all the flash loaders displayed in the [Programming Algorithm] list by clicking the [Remove] button.

(7) Click the [Add] button to open the [Add Flash Programming Algorithm] dialog box.

(8) Select an flash loader “S1C31D5xint 192kB Flash” in the list on [Add Flash Programming Algorithm] dialog box.

(9) Enable the checkboxes, [Erase Sectors], [Program] and [Verify], in the [Download Function] group box, and disable other checkboxes (Figure 2.4.5.3).

Note: When working with the “Debug” build, the internal flash memory will be not updated as the image will be loaded into internal RAM so an additional setting must be added via the [Initialization File] option to set the Program Counter and Stack registers that would normally be loaded from the Vector table in the internal flash memory. In the examples using “Debug” build, we use the “debug.ini” file to set those values correctly. In the example using “DebugFlash” build, the [Initialization File] field should be left blank.

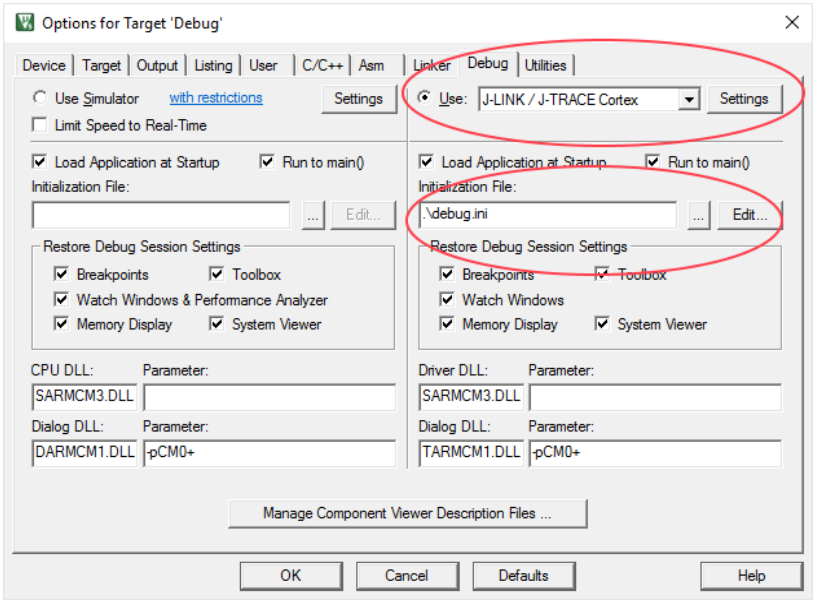


Figure 2.4.5.1 “Initialize File” Option

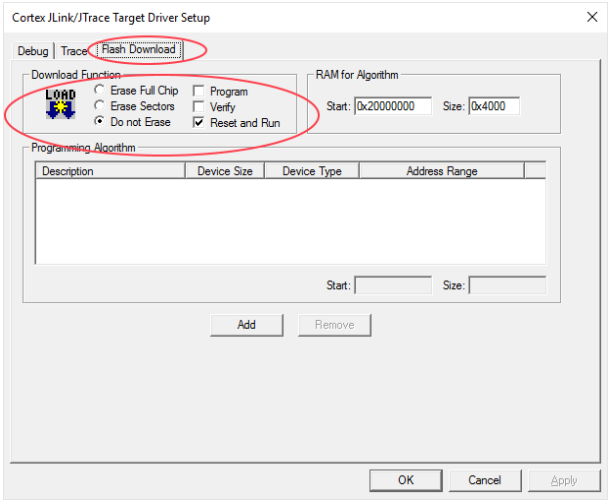


Figure 2.4.5.2 Flash Loader Unsetting

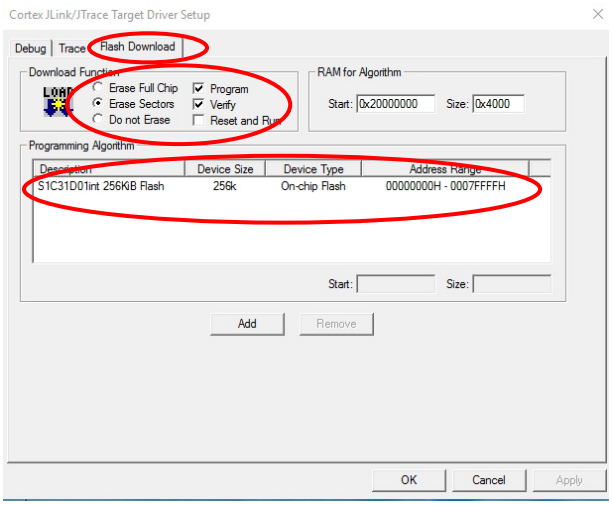


Figure 2.4.5.3 Flash Loader Setting

2.4.6 Project Build

To build an active project, select one of the build commands [Build] and [Rebuild] from the [Project] in the µVision menu (Figure 2.4.6.1).

Note: If linker errors occur in the build, there is a possibility that the library projects “sePeriphLibrary” is not built. Build this library project, and then build the active project again.

The Batch Build option, found under [Project] > [Batch Build] in the µVision menu, can be used to build all the Examples and by default constructs both the Debug and DebugFlash builds for each of the appropriate projects. The projects in the list are ordered such that the libraries are built first (Figure 2.4.6.2).

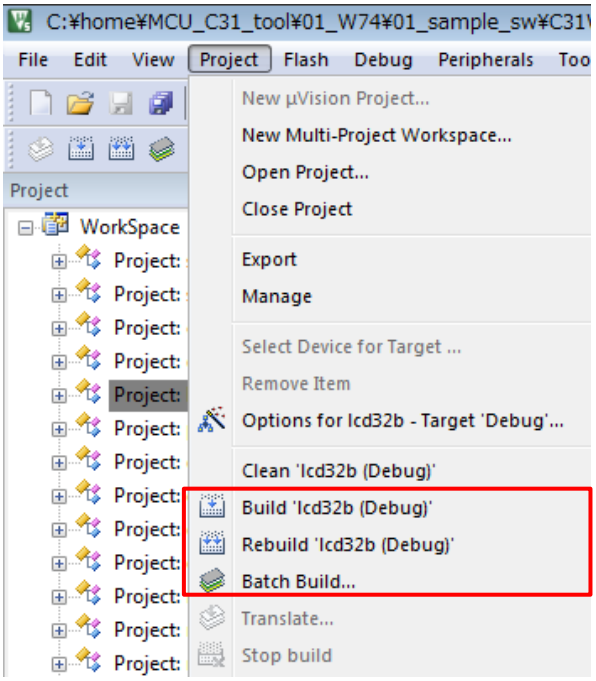


Figure 2.4.6.1 Build Commands

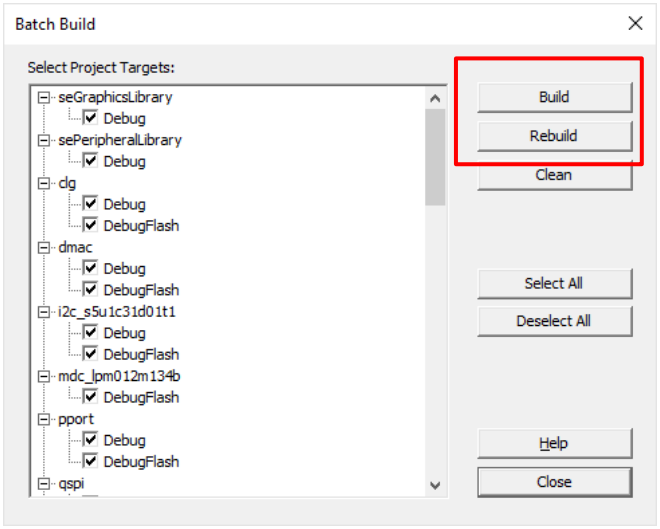


Figure 2.4.6.2 Batch Build

2.4.7 Project Download and Debug

Following a successful build, download the program image of the active project to the target board. To download the program image, select the [Flash] > [Download] in µVision menu (Figure 2.4.7.1). When the active project is "Debug" build, the program image is loaded in the internal RAM. When the active project is "DebugFlash" build, the program image is loaded in the internal flash memory.

To debug the program image downloaded to the target board, select the [Debug] > [Start/Stop Debug Session] in µVision menu (Figure 2.4.7.2).

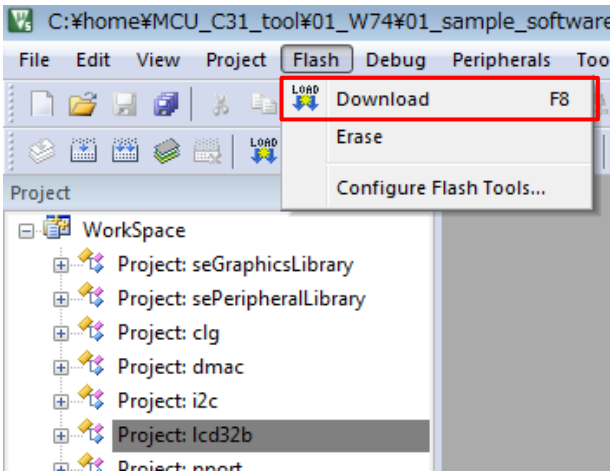


Figure 2.4.7.1 Download

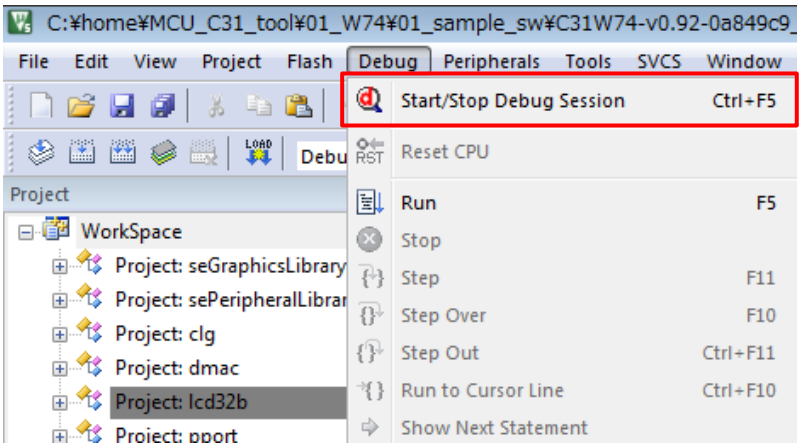


Figure 2.4.7.2 Debug

3 Details of Sample Software

Hardware setup is required for some examples. Refer to “S5U1C31D5xT1 manual” for additional information.

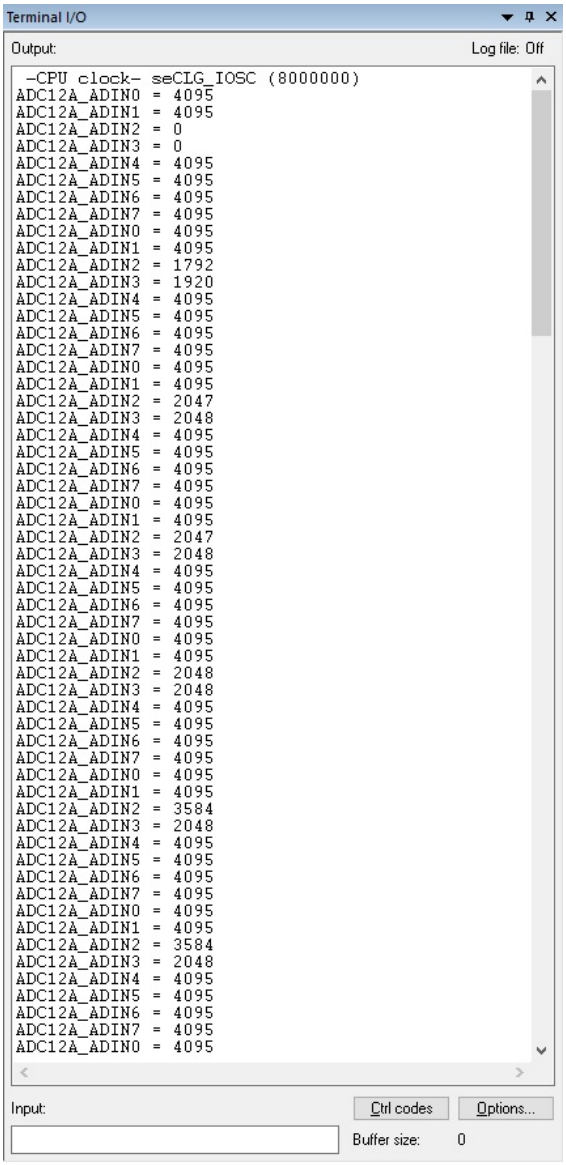
3.1 12-bit A/D Converter (ADC12A)

This example shows how to initialize the 12-bit A/D converter (ADC12A) and read data from various channels of ADC12A.

Operations

1. Initializes ADC12A.
2. Starts ADC12A.
3. Observe data output in the terminal program window.
4. You can change voltage value on ADC12A channels.

Example of Output



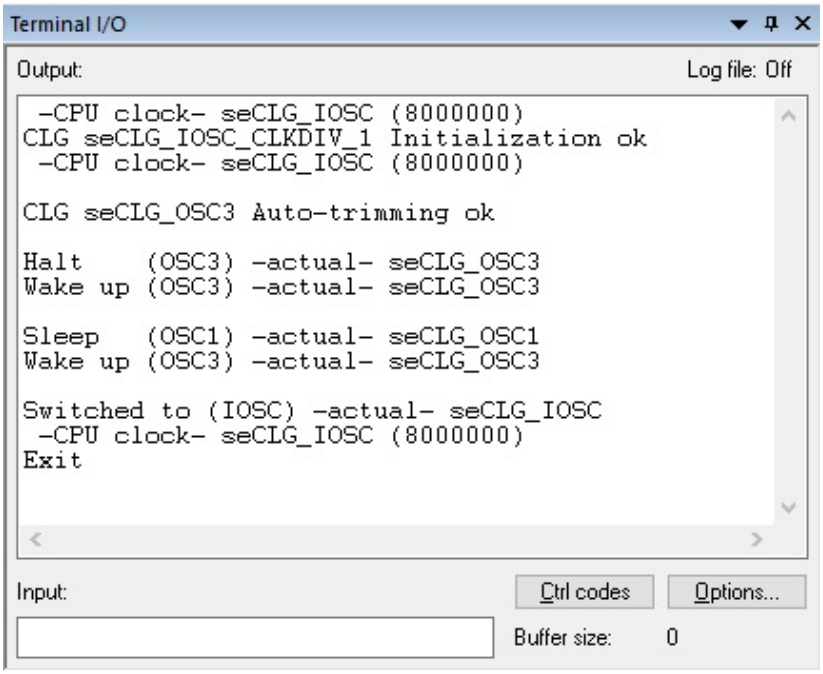
3.2 Clock Generator (CLG)

This example executes various Clock Generator (CLG) functions such as starting OSCs, setting Wake up clocks and sleep modes.

Operations

1. Initializes CLG.
2. Run auto-trimming of IOSC.
3. Verifies Sleep or Halt states running various clocks.
4. Verifies Wakeup states running various clocks.

Example of Output



Prerequisites

The OSC3 should not run from crystal in this example. Otherwise Auto-trimming will fail because crystal cannot be trimmed. The OSC3_SRC_XTAL must be commented out in *settings.h* file.

3.3 DMA Controller (DMAC)

This example provides a description of how to use DMA Controller (DMAC) to transfer data between memory and peripheral devices, as follows:

- Using DMAC in a basic memory-to-memory data transfer
- Using DMAC from a peripheral to memory data transfer
- Using DMAC concurrent peripheral to memory and memory to peripheral data transfer

For more examples of using DMAC for transfers see QSPI_DMA Example code.

Hardware Setup

To demonstrate the UART to memory DMA transfer, connect UART on the S5U1C31D5xT1 evaluation board to PC by a USB Adapter for UART. (see Figure 2.2.2.1, Figure 2.2.2.2)

Operations

Example 1: Memory to Memory DMA transfer

- DMAC Channel 0 is configured to transfer the contents of a data buffer stored in RAM to the reception buffer also in RAM. Access size is Byte.
- DMAC Channel 1 is configured to transfer the contents of a data buffer stored in RAM to the reception buffer in RAM. Access size is Half Word.
- DMAC Channel 2 is configured to transfer the contents of a data buffer stored in RAM to the reception buffer in RAM. Access size is Word.
- The start of transfer is triggered by software.

In this example:

1. The DMAC interrupts in NVIC are not enabled.
2. DMAC Channel transfer is enabled.
3. Source and destination addresses incrementing is enabled.
4. The transfer is started by setting the Software Request register bits.
5. At the end of the transfer, a Transfer Completion interrupt flag is generated.
6. Once interrupt flag is generated, the "number of transfers" is read which must be equal to 0.
7. The Transfer Complete Interrupt flag is then cleared.
8. A comparison between the source and destination buffers is done to check that all data have been correctly transferred.

Example 2: Peripheral to Memory DMA transfer.

- DMAC Channel 0 is configured to transfer data from a UART data register to memory.
- UART is configured with baud rate 115200. DMA transfers are enabled for Receive buffer full event.
- The start of transfer is triggered by typing 8 characters in the PC window running a terminal program.

In this example:

1. The DMAC interrupts in NVIC are not enabled.
2. DMAC Channel transfer is enabled.
3. DMAC Channel filtering is disabled for the selected DMAC Channel.
4. Source address incrementing is disabled.
5. Destination address incrementing is enabled.
6. The transfer is started by an UART receive buffer becoming full.
7. At the end of the transfer, a DMAC Transfer Completion interrupt flag is generated.
8. Once interrupt flag is generated, the "transfer mode" is read which must be equal to 0(STOP mode).
9. The Transfer Complete Interrupt flag is then cleared.
10. The Memory to Peripheral transfer is used to output characters back to the terminal

window.
11. Correctness of the transfer is verified by seeing correct characters displayed in the terminal window.

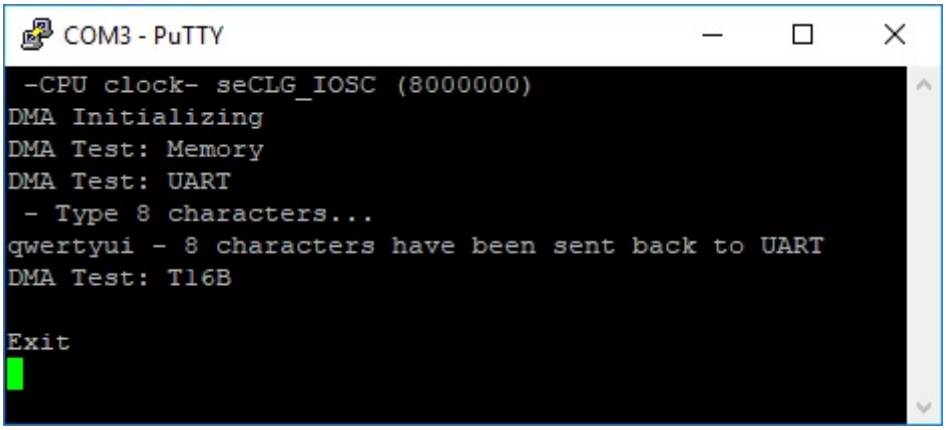
Example 3: Concurrent Peripheral to Memory and Memory to Peripheral DMA transfers.

- T16B_0 sub-channel 0 is configured for compare mode.
- T16B_0 sub-channel 1 is configured for capture mode.
- DMAC Channel 0 is configured to transfer data from memory to the T16B sub-channel 0 data register.
- DMAC Channel 1 is configured to transfer data from the T16B sub-channel 1 data register to memory.

In this example:

1. The DMAC interrupts in NVIC are enabled.
2. DMAC Channel transfers are enabled for two channels.
3. DMAC Channel filtering is disabled for the selected DMAC channels.
4. The DMA transfer is started by a compare interrupt on the T16B sub-channel 0.
5. At the end of the transfer, a DMAC Transfer Completion interrupt flag is generated on the DMAC channel 0.
6. Once the DMAC interrupt is generated it sets a software completion flag by software in the DMAC interrupt service routine.
7. The Transfer Complete Interrupt flag is then cleared.
8. The Peripheral to Memory transfer is started by a capture interrupt on T16B sub-channel 1.
9. At the end of the transfer, a DMAC Transfer Completion interrupt flag is generated on the DMAC channel 1.
10. Once interrupt flag is generated it sets a software completion flag in the DMAC interrupt service routine.
11. The Transfer Complete Interrupt flag is then cleared.
12. Software detects both transfer completion.

Example of Output



3.4 I2C (I2C_S5U1C31D5xT1)

This example shows how to use the I2C module in the master mode with I2C Serial EEPROM (24FC512). This example requires 24FC512 EEPROM memory chip.

Hardware Setup

Connect the S5U1C31D5xT1 evaluation board to the 24FC512 EEPROM memory chip as follows.

Table 3.4.1 Connection between I2C Master and I2C Slave

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

The BOARD_I2C_SDA_PORT and BOARD_I2C_SCL_PORT are defined as sePPORT_P17 and sePPORT_P16 in *board.h* file

The picture in Figure 3.4.1 and the diagram in Figure 3.4.2 show the connection of S5U1C31D5xT1 with an I2C Serial EEPROM (24FC512).

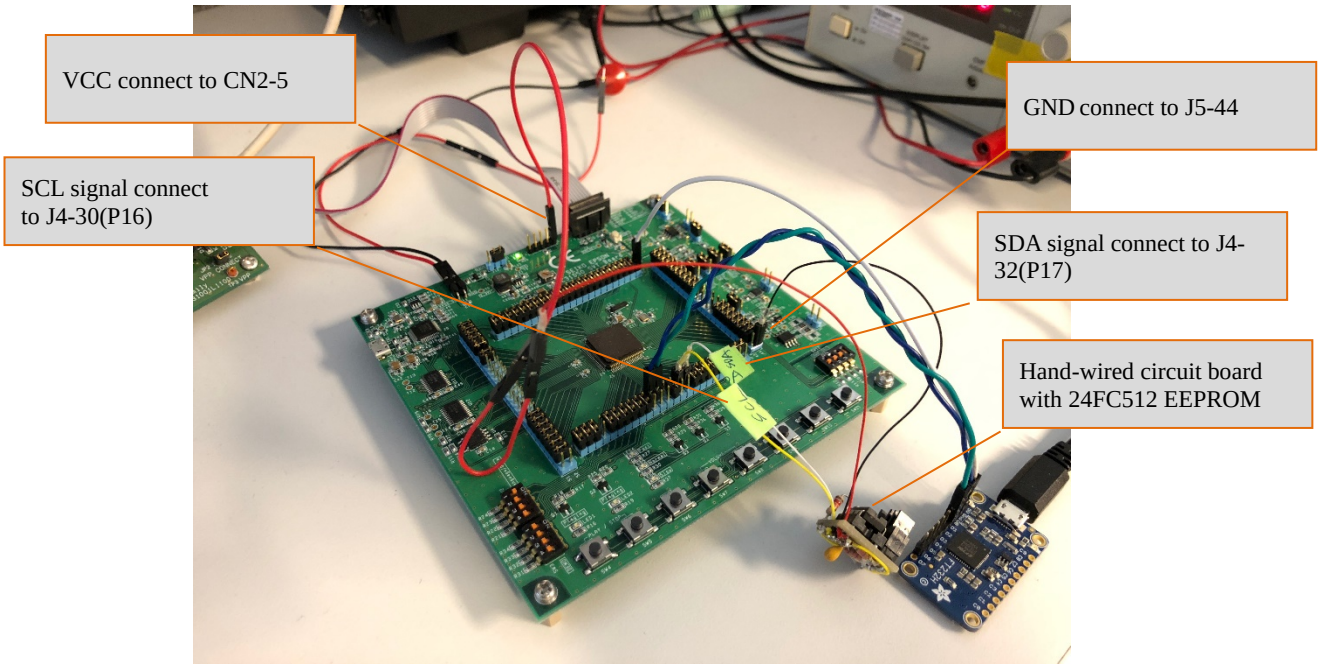


Figure 3.4.1 I2C Serial EEPROM Connection to Evaluation Board

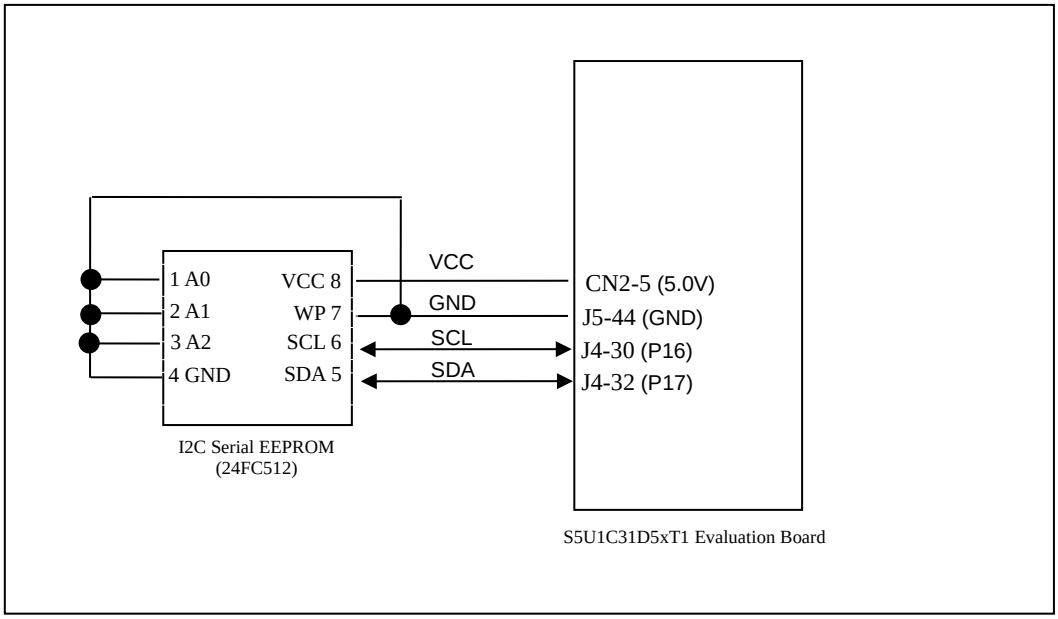


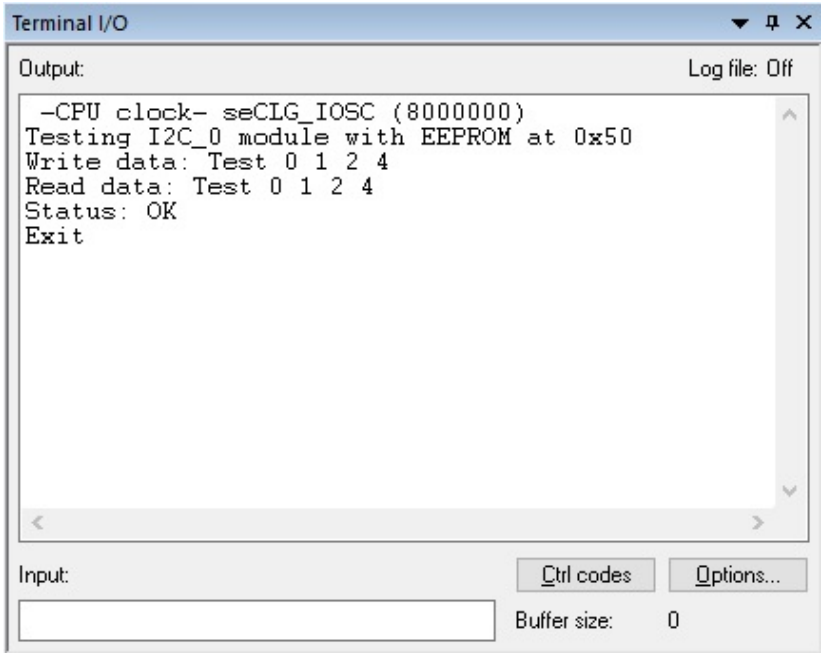
Figure 3.4.2 I2C Serial EEPROM Wiring to S5U1C31D5xT1 Evaluation Board

Operations

This example tests I2C_0 module with EEPROM at address 0x50.

1. Test configures I2C Master mode.
2. Test writes ascii string to the EEPORM memory.
3. Then it reads data back from EEPORM, compares data.

Example of Output



3.5 I/O Ports (PPORT)

This example configures pin BOARD_OUTPORT as an output connected to pin BOARD_INPORT as an input. It performs various tests to check the connectivity of BOARD_OUTPORT and BOARD_INPORT.

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

Hardware Setup

The PPORT example has three configurations, TEST_PPORT_01, TEST_PPORT_23 and TEST_PPORT_OTHER. See *board.h* file for details.

For the TEST_PPORT_OTHER configuration, which is default, set jumper between J4-48 and J4-50 pins.

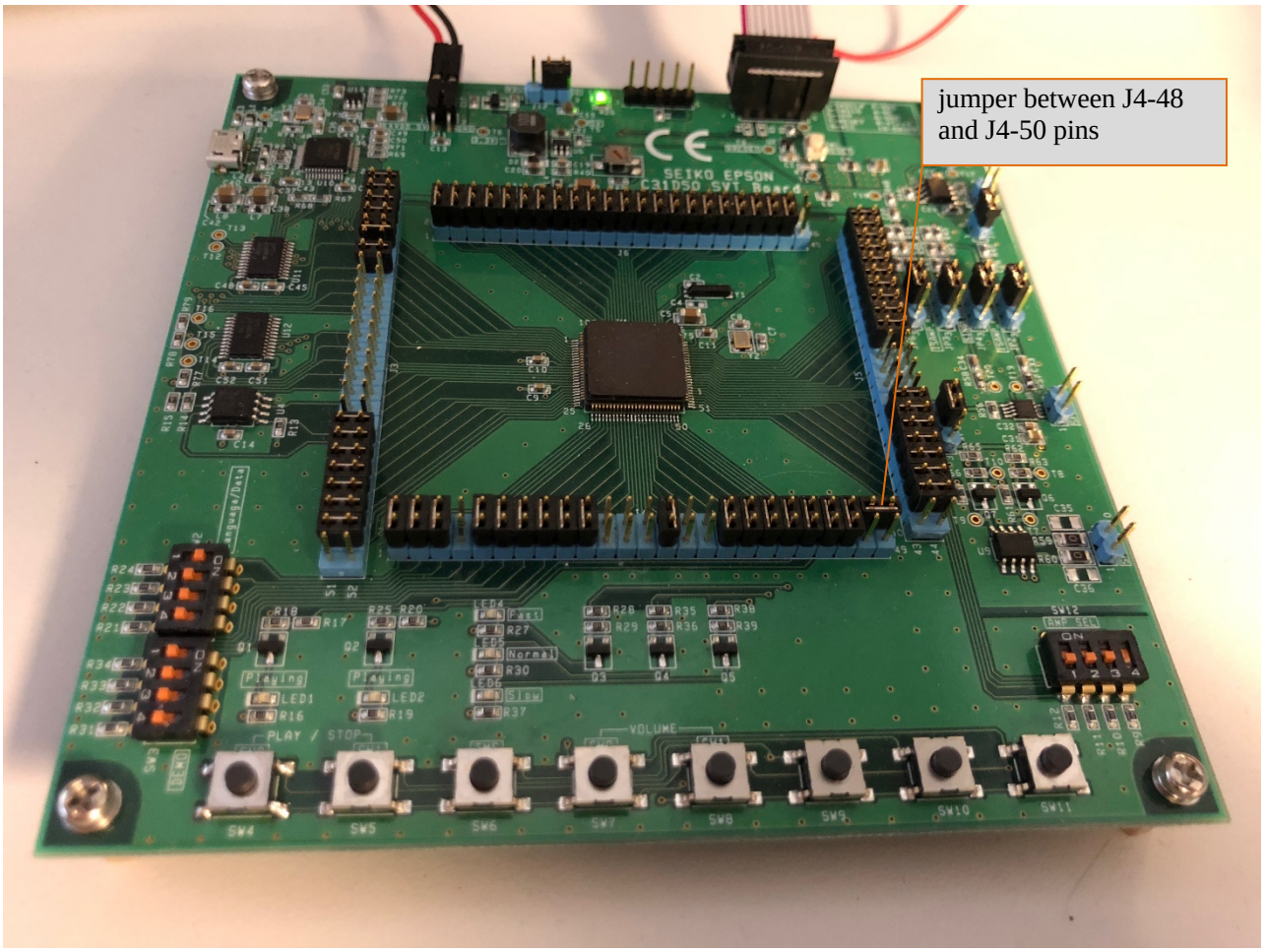


Figure 3.5.1 Jumper Setting of TEST_PPORT_OTHER Configuration

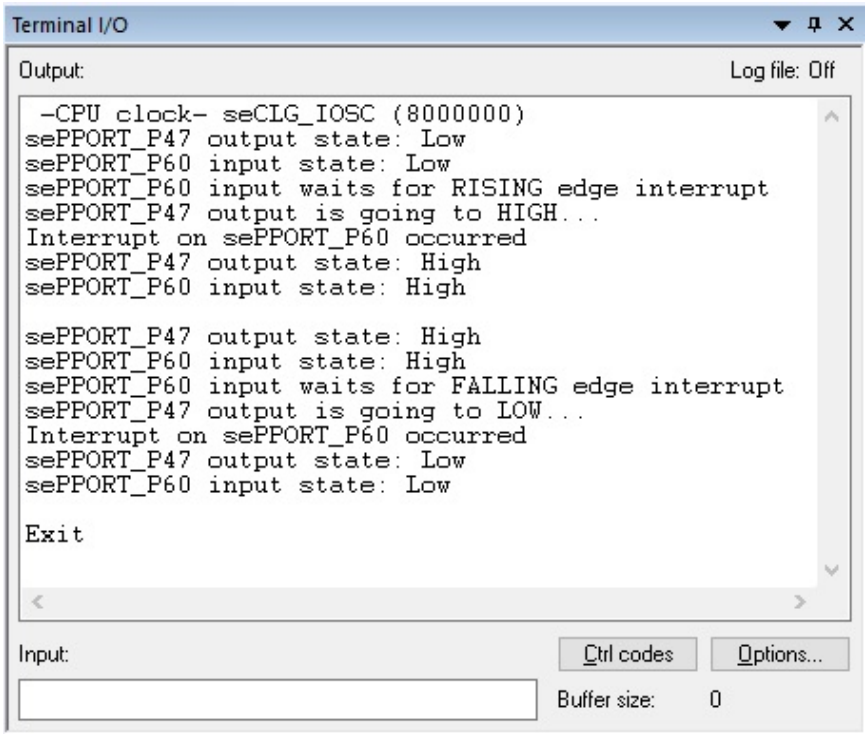
The sePPORT_P47 is used as BOARD_OUTPORT.
The sePPORT_P60 is used as BOARD_INPORT.
For other configurations see *board.h* file.

Operations

1. Start the OSC1 oscillation and switch the system clock from IOSC to OSC1. Then stop the IOSC.
2. Initialize the ports.
3. Set the P47 port to output and Output a Low level signal.
4. Set the P60 port to input and set it so that an interrupt occurs when the level changes from Low to High.
5. Output a High level signal from the P47 port.

6. Confirm that the P60 port is at High level after an interrupt in the P60 port.
7. Set the P60 port so that an interrupt occurs when the level changes from High to Low.
8. Output a Low level signal from the P47 port.
9. Confirm that the P60 port is at Low level after an interrupt in the P60 port.

Example of Output



3.6 Quad Synchronous Serial Interface (QSPI)

This example provides a description of how to use a QSPI in the master mode to communicate with an external QSPI flash memory.

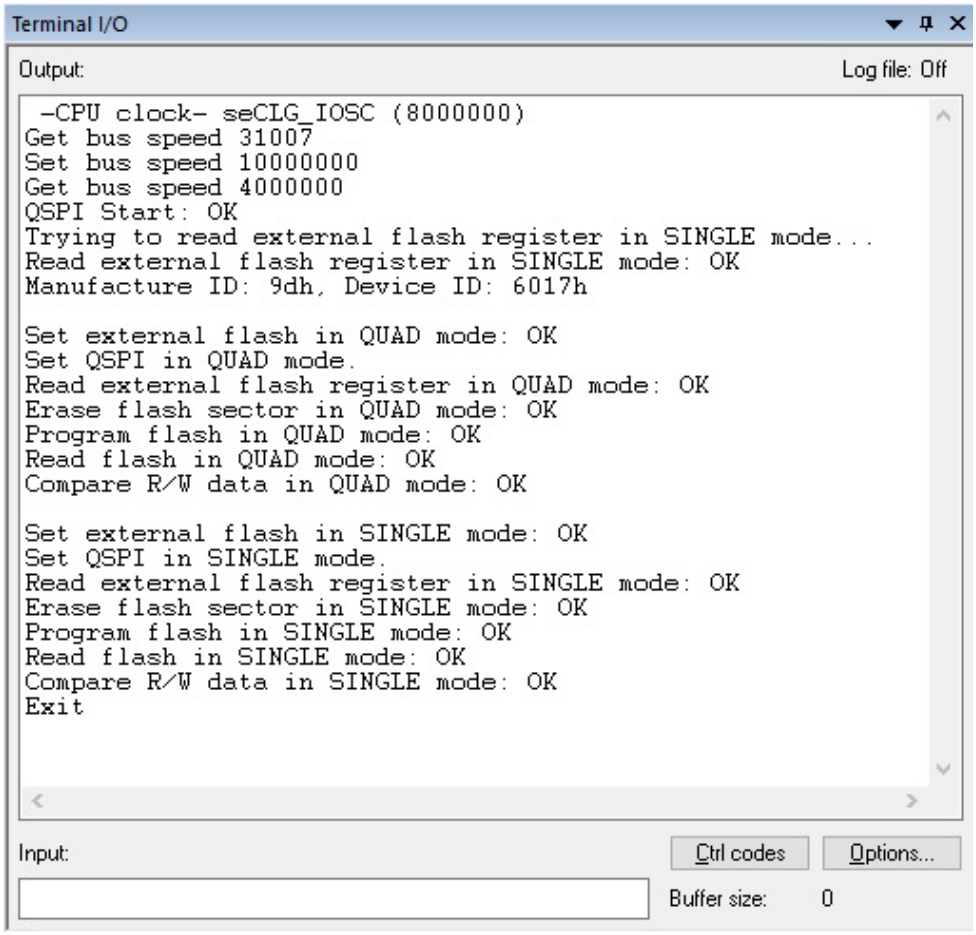
Hardware Setup

The ISSI QSPI flash memory on S5U1C31D5xT1 evaluation board is used in this example.

Operations

1. The example code initializes the QSPI module in master mode as below:
   - Data length is 8bit
   - Data format is MSB first
   - Use 16-bit timer T16_2 for baud rate generator.
2. Then software sets bus speed to 10000000 if possible (depends on selected clock speed).
3. Example software starts QSPI in Single mode.
4. Then software checks if it can read External Flash mode register correctly.
5. Then it reads Flash ID.
6. If the flash operations succeed following flash actions are taken consecutively for two QSPI modes:
   - Quad mode Erase sector, Program Sector, Read and compare sector.
   - Single mode Erase sector, Program Sector, Read and Compare sector.

Example of Output



3.7 Quad Synchronous Serial Interface with DMA (QSPI_DMA)

This example provides a description of how to use a QSPI with DMA in the master mode to communicate with an external QSPI flash memory.

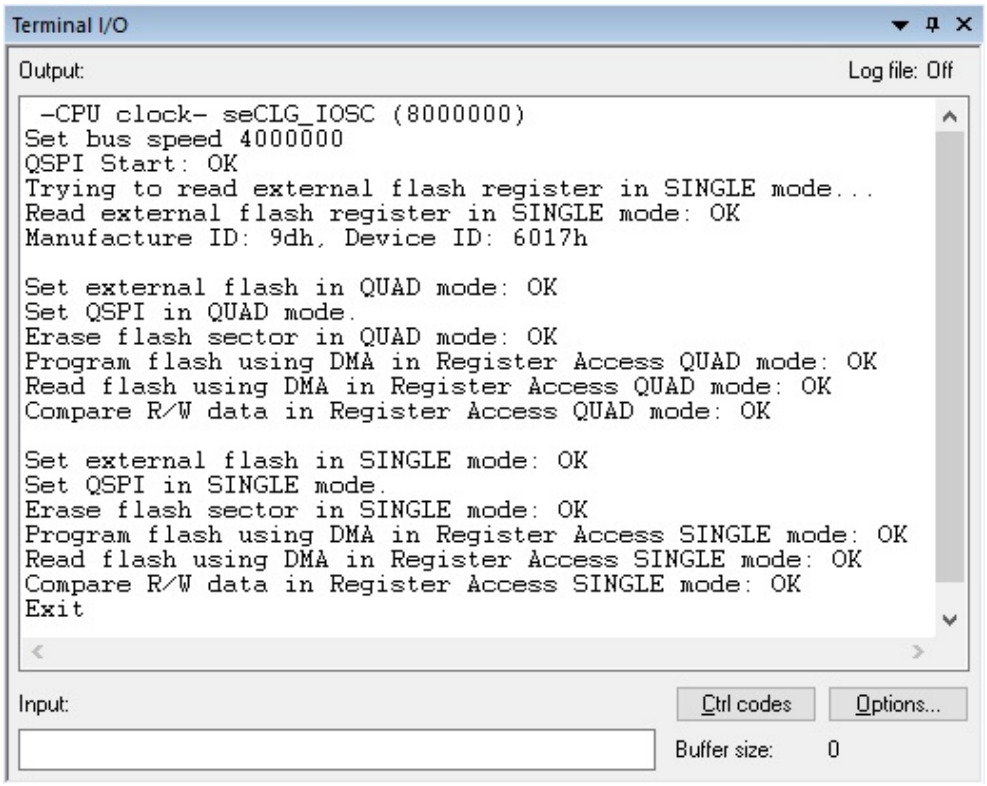
Hardware Setup

The ISSI QSPI flash memory on S5U1C31D5xT1 evaluation board is used in this example.

Operations

1. The example code initializes the QSPI module in master mode as below:
   - Data length is 8bit
   - Data format is MSB first
   - Use 16-bit timer T16_2 for baud rate generator.
2. The example code sets bus speed to 4000000.
3. Then it configures DMA Controller descriptors and initializes DMA Controller.
4. Example software starts QSPI in the register access Single mode.
5. Then software checks if it can read the External Flash mode register correctly.
6. Then it reads Flash ID.
7. If the flash operations succeed software performing following flash actions (in Quad and Single modes):
   - Erase Sector
   - Program Sector using register access DMA transfers.
   - Read Sector using register access DMA transfers and compare with the programmed pattern.

Example of Output



3.8 Quad Synchronous Serial Interface Master (QSPI_MASTER)

This example provides a description of how to use a QSPI in the master mode.

Hardware Setup

1. The S5U1C31D5xT1 evaluation board has an onboard QSPI flash memory attached to the QSPI interface on S1C31D5x. In order to run the QSPI master/slave sample programs, the serial flash on each board must be isolated from the QSPI interface by removing J3-24, J3-26, J3-28, J3-30, J3-32, J3-34 and J3-36.
2. Connect the S5U1C31D5xT1 evaluation boards where the QSPI master/slave sample programs are installed. Then, connect each port as shown Table 3.8.1 or 3.8.2.
   Note: Please uncomment out QSPI_MODE_SINGLE in *settings.h* for the Single mode setup.
3. Launch the slave example program first, then the master program

Table 3.8.1 Connection between QSPI Master and QSPI Slave (Dual or Quad)

Dual / Quad mode					
[master]				**[slave]**	
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

* For definition of J3 see the schematic in S5U1C31D5xT1 manual.

Table 3.8.2 Connection between QSPI Master and QSPI Slave (Single)

Single mode					
	[master]			**[slave]**	
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	

* For definition of J3 see the schematic in S5U1C31D5xT1 manual.

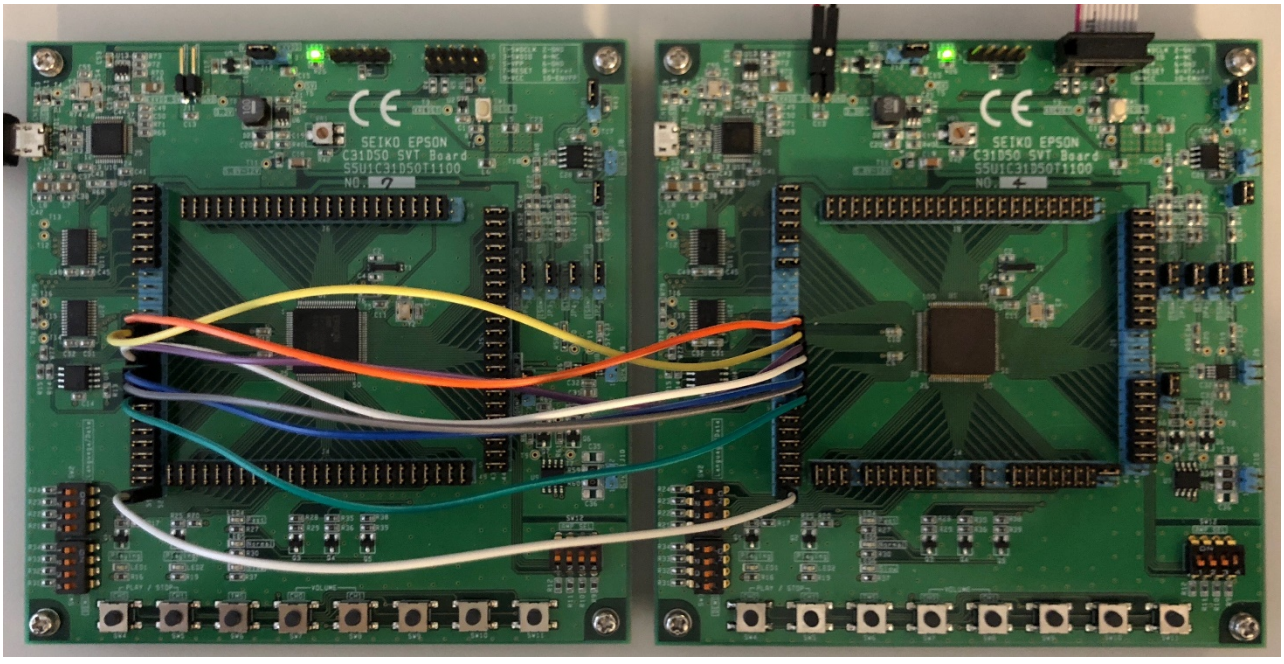


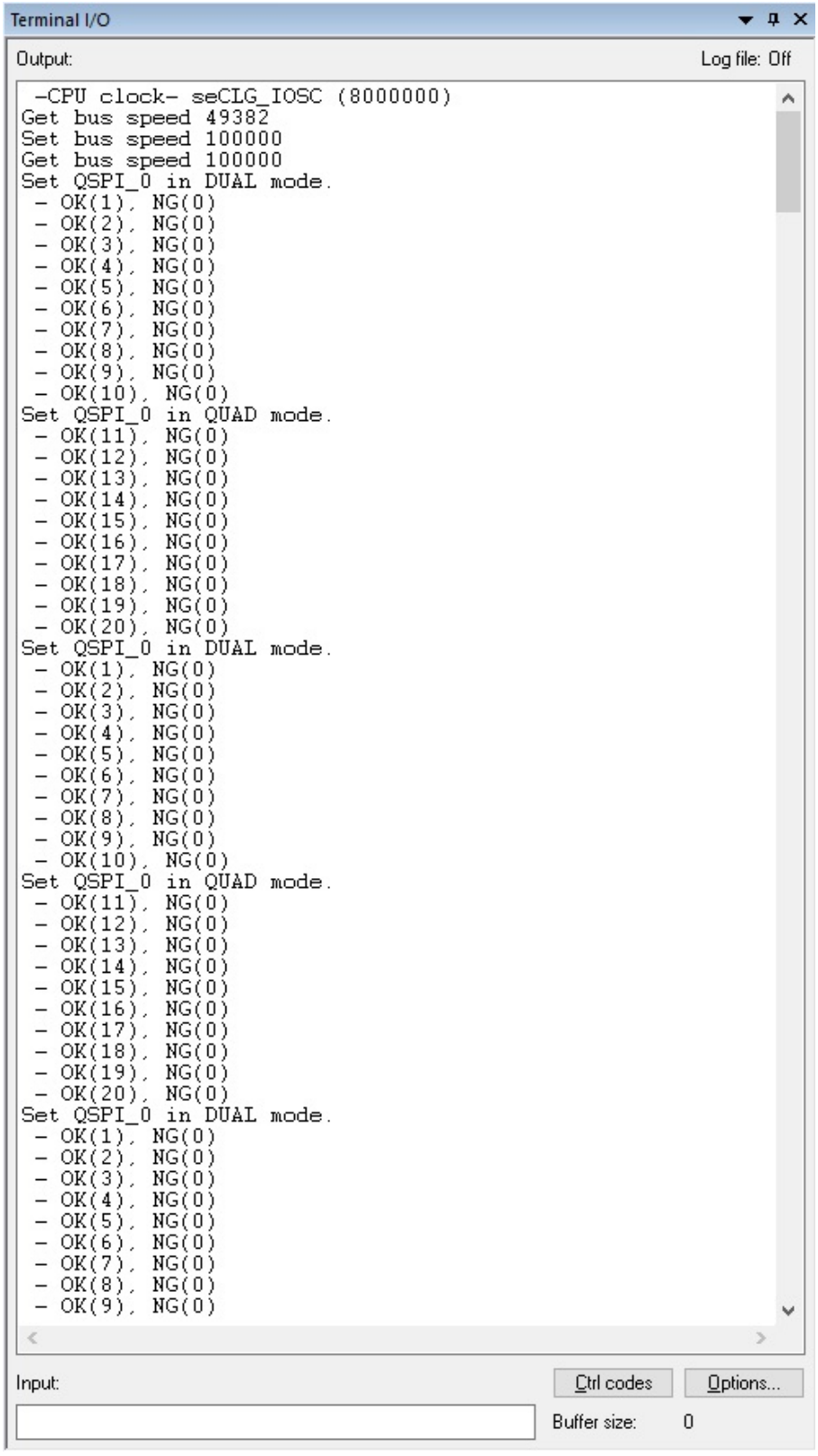
Figure 3.8.1 Connection between the Evaluation Boards (DUAL/QUAD Mode)

Operations

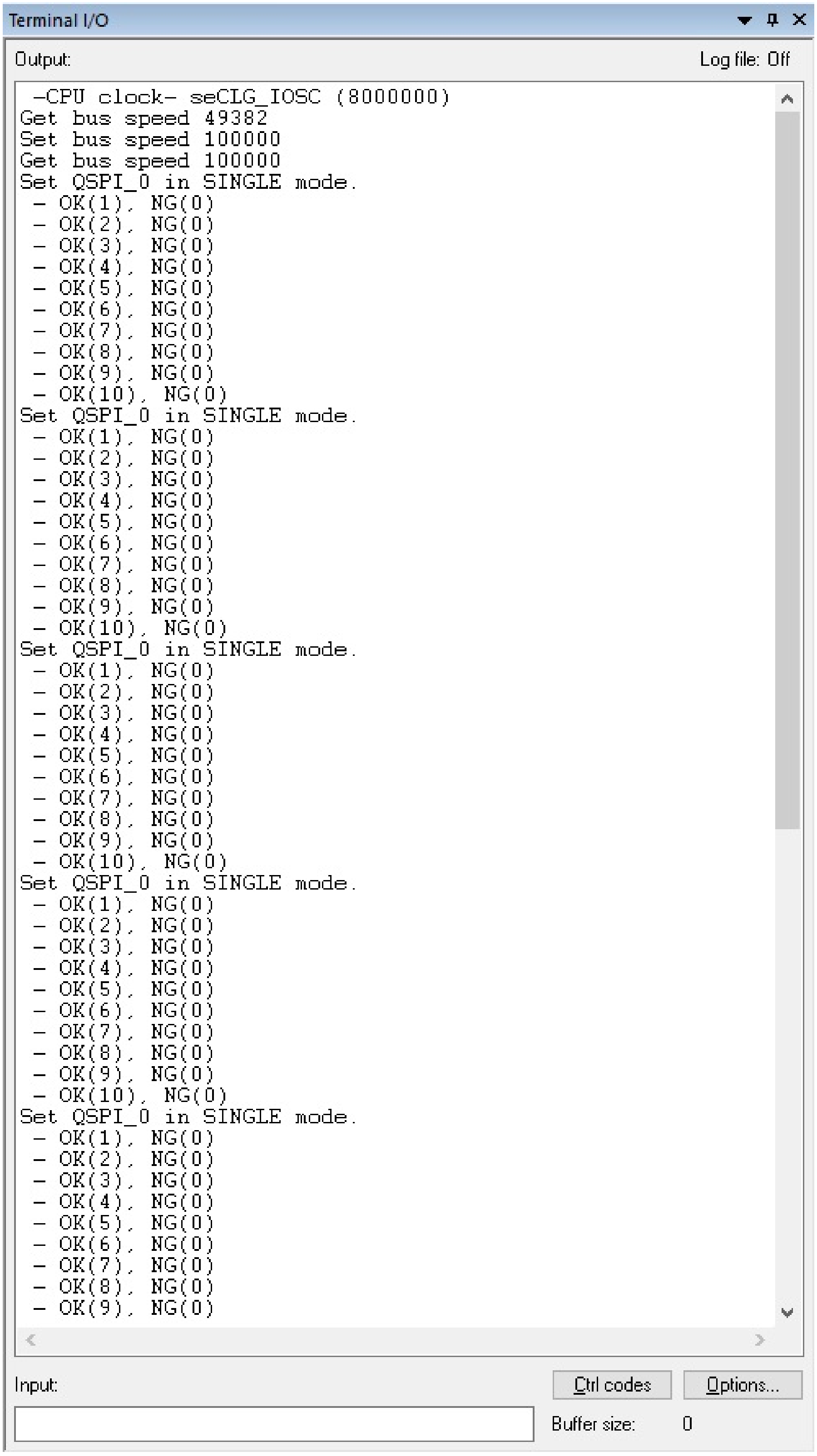
1. Initialize the QSPI module in master mode as below:
   - Data length is 8bit
   - Data format is MSB first
   - Use 16-bit timer T16_2 for baud rate generator.
2. Set bus speed to 100000.
3. Assign "**BOARD_QSPI_HANDSHAKE_PORT**" as output to set Read/Write command to the slave.
4. Send the "BUF_SIZE" bytes of random data to the slave.
5. Receive the "BUF_SIZE" bytes of data from the slave.
6. Compare whether the received data is the same as the sent data and exit.

Example of Output

For Double/Quad Mode:



For Single Mode:




```
 -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)
```

3.9 Quad Synchronous Serial Interface Slave (QSPI_SLAVE)

This example provides a description of how to use a QSPI in the slave mode.

Hardware Setup

See the description of Hardware Setup in “3.8 Quad Synchronous Serial Interface Master (QSPI_MASTER)”.

Operations

1. Initialize the QSPI module in slave mode as below:
   - Data length is 8bit
   - Data format is MSB first
2. Assign “**BOARD_QSPI_HANDSHAKE_PORT**” as input and enable interrupt on rising edge to detect a start of communication
3. Receive the “BUF_SIZE” bytes of random data from the master
4. Send the “BUF_SIZE” bytes of data back to the master

Example of Output

Slave has no output.

3.10 IR Remote Controller (REMC3)

This example provides a description of how to use the IR Remote Controller (REMC3) to generate various IR pulses.

Hardware Setup

This example requires an infrared LED device. For information on a connection of an infrared LED, refer to the "S5U1C31D5xT1 manual".

Operations

This example program uses the NEC format for output.

1. REMC3 determines the data waveform using two parameters: APLEN (data signal duty) and DBLEN (data signal cycle). Each of APLEN and DBLEN defines the waveform with 34 frame signals and 2 repeat signals. APLEN and DBLEN pair allocates the leader, customer code, and repeat signals at initialization time. Those signals are common to different scenarios when switches SW2 to SW3 are pushed. Data may vary according to selection of the SW2 to SW3. Data is allocated when a switch interrupt occurs.
2. Further initialization consists of configuration of switches SW2 to SW3, allocation ports for REMO, and initialization of T16 Ch.0. Then CPU enters into the halt mode to wait for interrupts caused by pushing switches SW2 to SW3.
3. When an interrupt occurs allocate data's APLEN and DBLEN values. Since REMC3 uses a buffer mode, REMC3 should be activated after initialization. Also, activate T16 Ch.0 and generate a T16 Ch.0 interrupt after about 108ms.
4. When a compare DB interrupt occurs in REMC3, write the following APLEN and DBLEN data into register. When writing last but one data element, change REMC3 to a one-shot mode to prevent the whisker-like signal output. After last data element, stop the REMC3 operation.
5. When a T16 Ch.0 interrupt occurred, check whether the same switch SW2 to SW3 was pushed. If the same switch was pushed, set the register values of REMC3 so that a repeat waveform is generated, initialize REMC3 before activation, and activate REMC3. Otherwise, stop the T16 Ch.0 operation.

3.11 R/F Converter (RFC)

This example provides a description of how to use the R/F Converter (RFC) to take measurements.

Hardware Setup

Connect the J5 connector to the following components.

J5-25	SENA0	----------Resistive sensor (DC bias) ----------	RFIN0	J5-21	
J5-23	REF0	---------- Reference resistor ----------	RFIN0	J5-21	
J5-21	RFIN0	---------- Reference resistor and Reference capacitor----------	GND	J5-43	

The RFC module has designated ports as follow.

```
#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
```

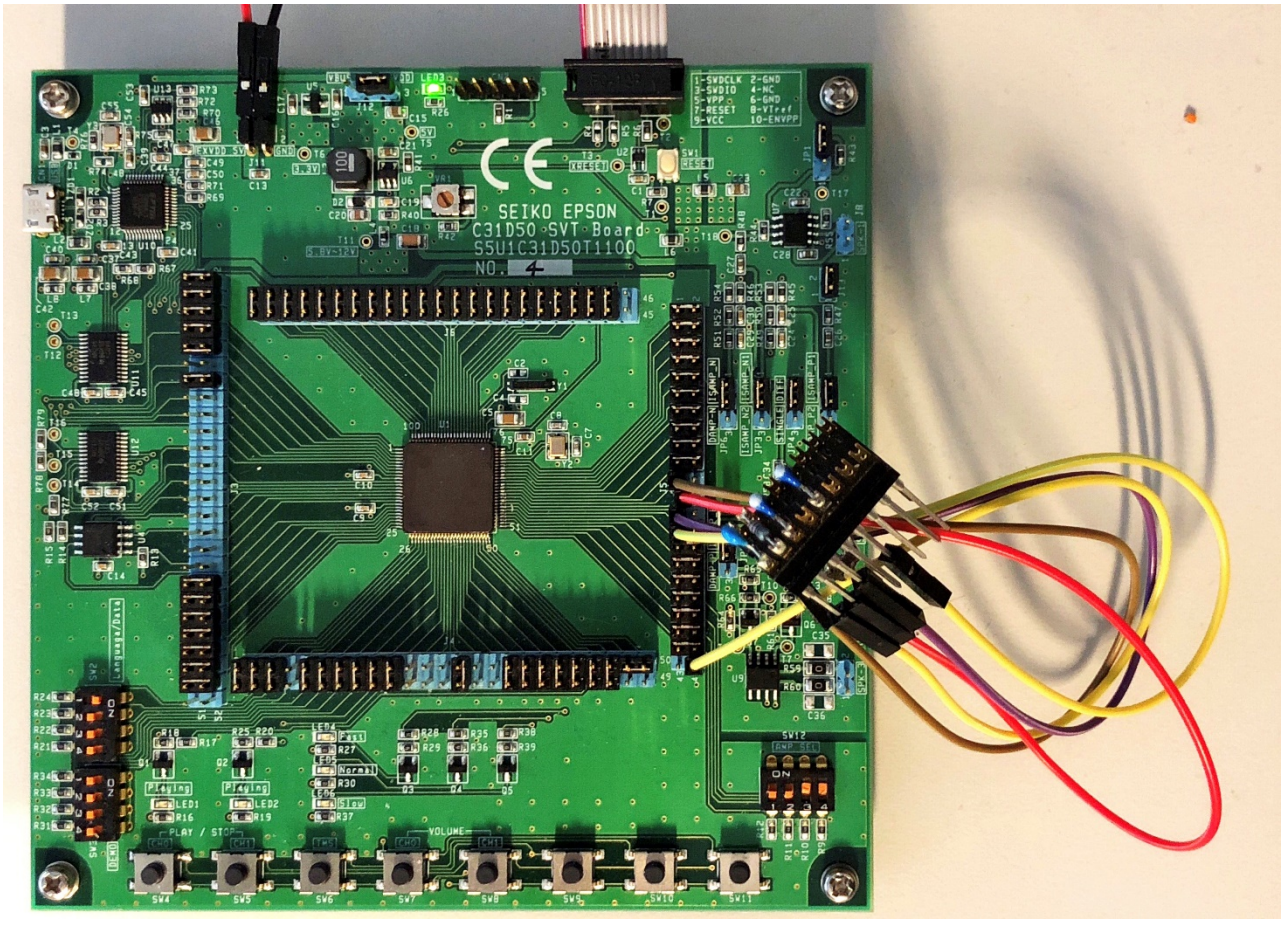
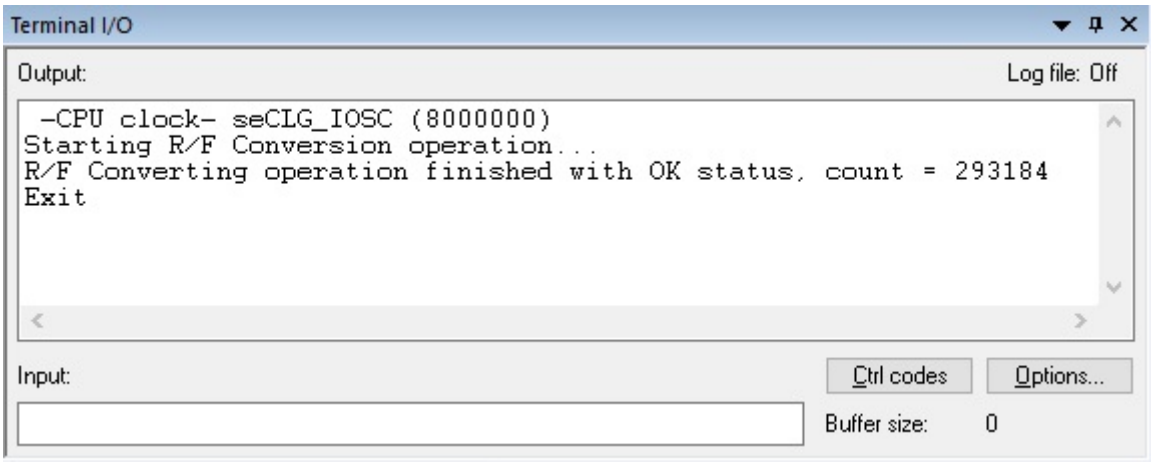


Figure 3.11.1 Connection of RFC connection components to Evaluation Board

Operations

1. Initializes RFC.
2. Runs RFC and gets measurement counter.
3. Displays conversion status and counter number.

Example of Output



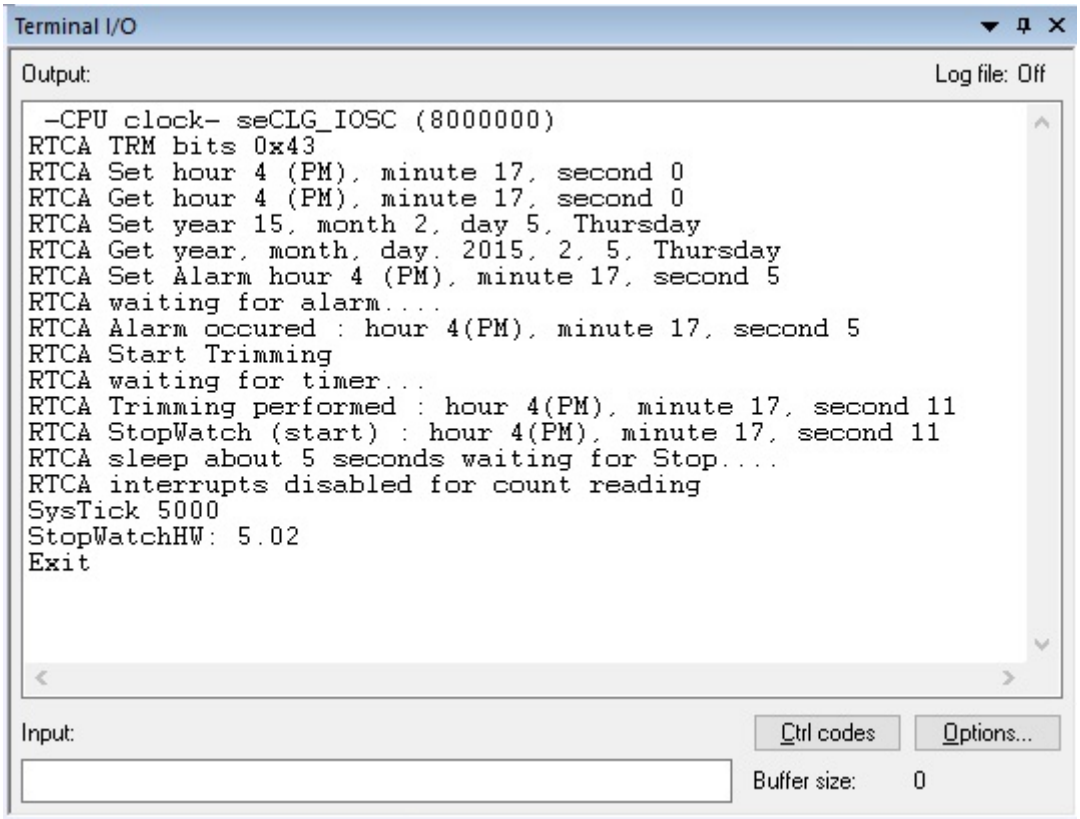
3.12 Real Time Clock (RTCA)

This example provides a description of how to program and read the various functions of the Real Time Clock (RTCA) such as the time/date, stopwatch, alarm, and trimming.

Operations

1. Initializes RTCA.
2. Starts RTCA.
3. Calculates TRM.
4. Sets time and date, and reads it back.
5. Sets an alarm and CPU goes to sleep. Expects interrupt while sleeping.
6. Starts 1 second timer to perform trimming.
7. Checks stop watch operations.

Example of Output



3.13 Synchronous Serial Interface Master (SPIA_MASTER)

This example provides a description of how to use an SPIA in the master mode to transfer data buffer.

Hardware Setup

1. Connect the S5U1C31D5xT1 evaluation boards where the SPIA master/slave sample programs are installed. Then, connect each port as shown Table 3.13.1.
2. Launch the slave example program first, then the master program

Table 3.13.1 Connection between SPIA Master and SPIA Slave

[master]				[slave]		
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

The SPIA uses GPIO for IO configured via UPMUX.

```
#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
```

See more details in *board.h* file for specific ports used in the SPIA master/slave sample programs.

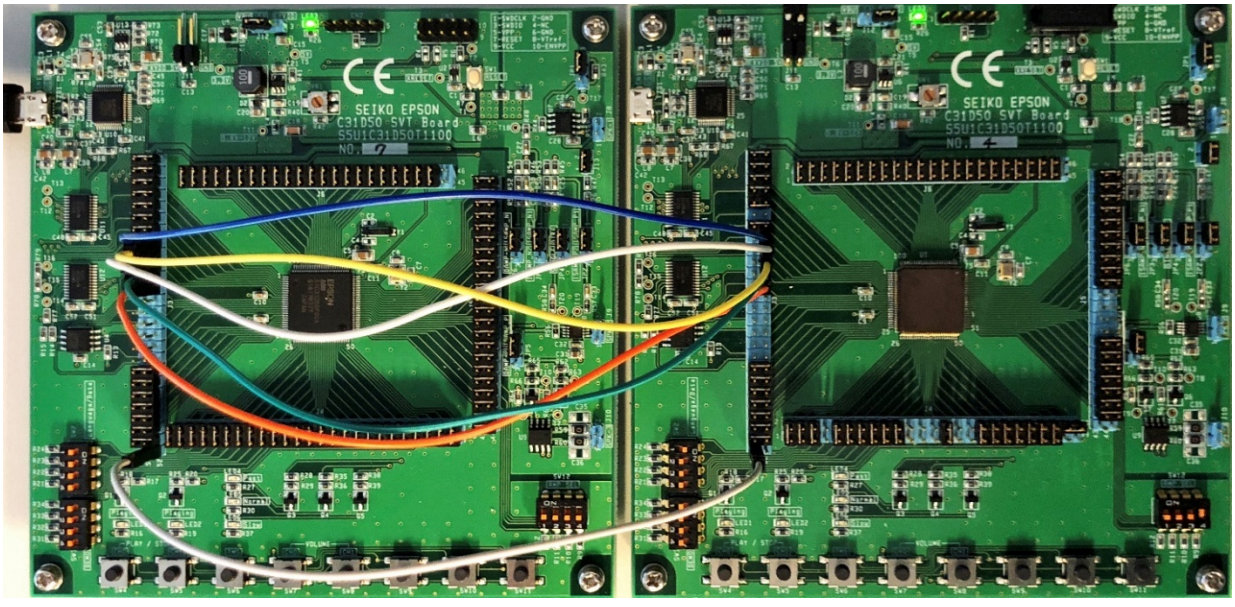
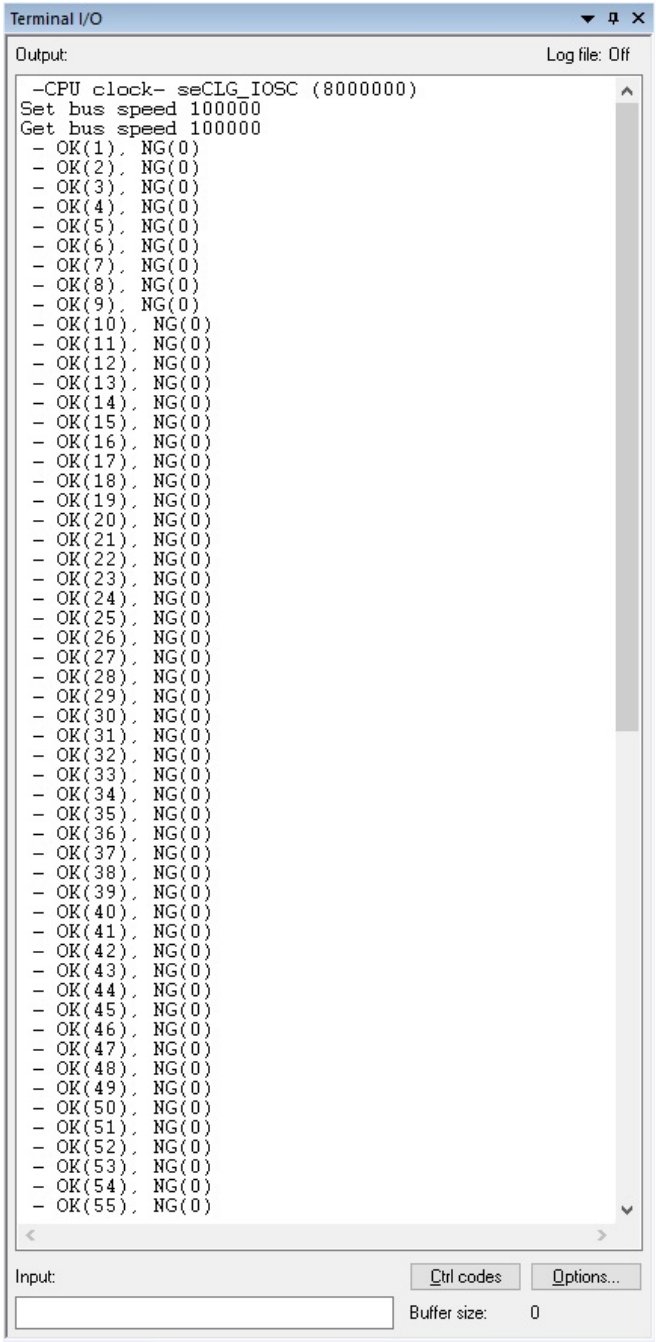


Figure 3.13.1 Connection between the Evaluation Boards

Operations

1. Initialize the SPIA module in master mode as below:
   - Data length is 8bit
   - Data format is MSB first
   - Use 16-bit timer T16 Ch.1 for baud rate generator.
2. Set bus speed to 1000000.
3. Assign "**BOARD_SPI_HANDSHAKE_PORT**" as output to set Read/Write command to the slave.
4. Send the "BUF_SIZE" bytes of random data to the slave.
5. Receive the "BUF_SIZE" bytes of data from the slave.
6. Compare whether the received data is the same as the sent data and exit.

Example of Output



3.14 Synchronous Serial Interface Slave (SPIA_SLAVE)

This example provides a description of how to use a SPIA in the slave mode to transfer data buffer.

Hardware Setup

See the description of Hardware Setup in "3.13 Synchronous Serial Interface Master (SPIA_MASTER)".

Operations

1. Initialize the SPIA module in slave mode as below:
   - Data length is 8bit
   - Data format is MSB first
2. Assign "**BOARD_SPI_HANDSHAKE_PORT**" as input to recognize Read/Write command from master.
3. Receive the "BUF_SIZE" bytes of data from the master.
4. Send the "BUF_SIZE" bytes of random data to the master.

Example of Output

Slave has no output.

3.15 Power Supply Voltage Detection Circuit (SVD3)

This example provides a description of how to use the Power Supply Voltage Detection Circuit (SVD3) to detect VDD voltage drop.

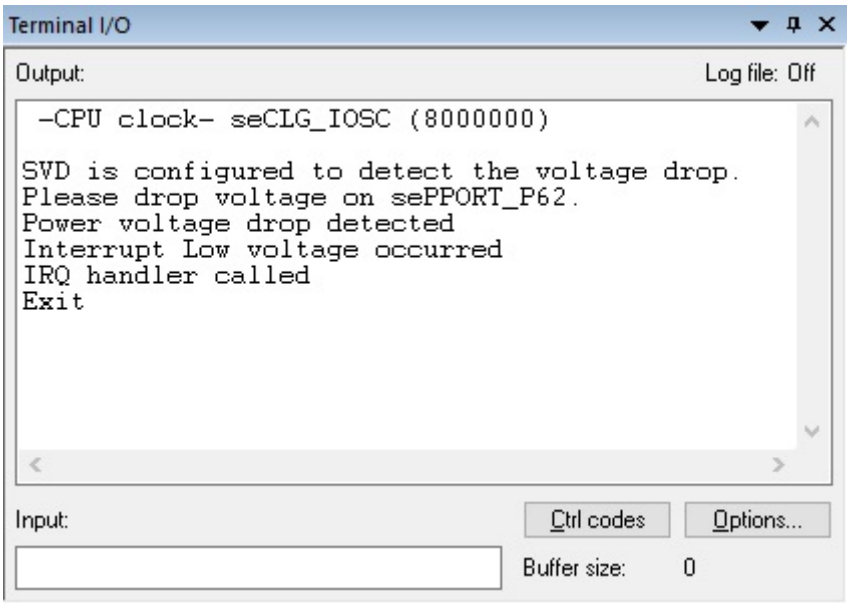
Hardware Setup

This example requires the S5U1C31D5xT1 evaluation board configured to power from the debugger probe. Also, set the jumper to the EXVDD side of J12 on the S5U1C31D5xT1 evaluation board before running example.

Operations

1. Software prepares to watch the VDD pin dropping the voltage below some limit.
2. Disconnect the jumper from J12.
3. Software detects voltage drop and reports it on the terminal.

Example of Output



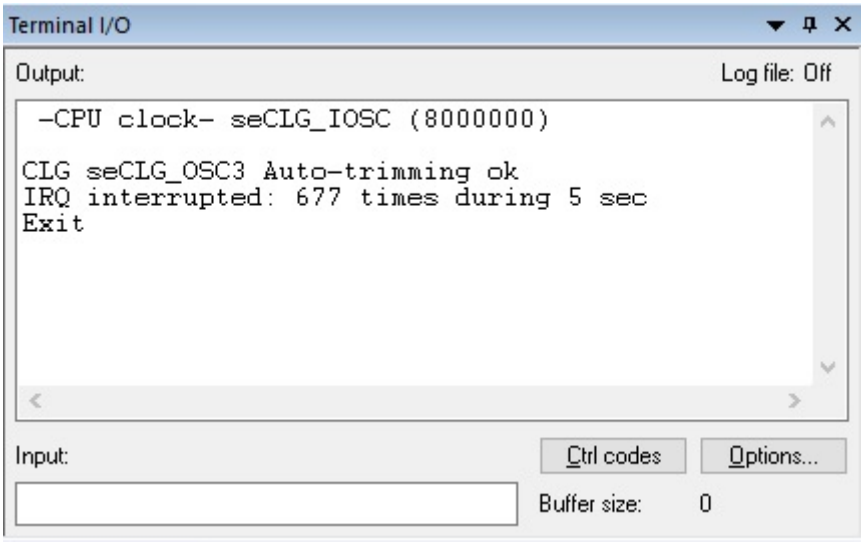
3.16 16-Bit Timer (T16)

This example programs the 16-Bit Timer (T16) to interrupt periodically and counts the number of interrupts that occurred within a 5-second interval.

Operations

1. OSC3 is selected as timer clock source. Example code does OSC3 auto-trimming to provide clock accuracy for the timer clock.
2. Initializes T16 channel 0.
3. Configures T16 channel 0 interrupts.
4. Enables T16 interrupts.
5. Counts a number of interrupts happened during 5 second interval.

Example of Output



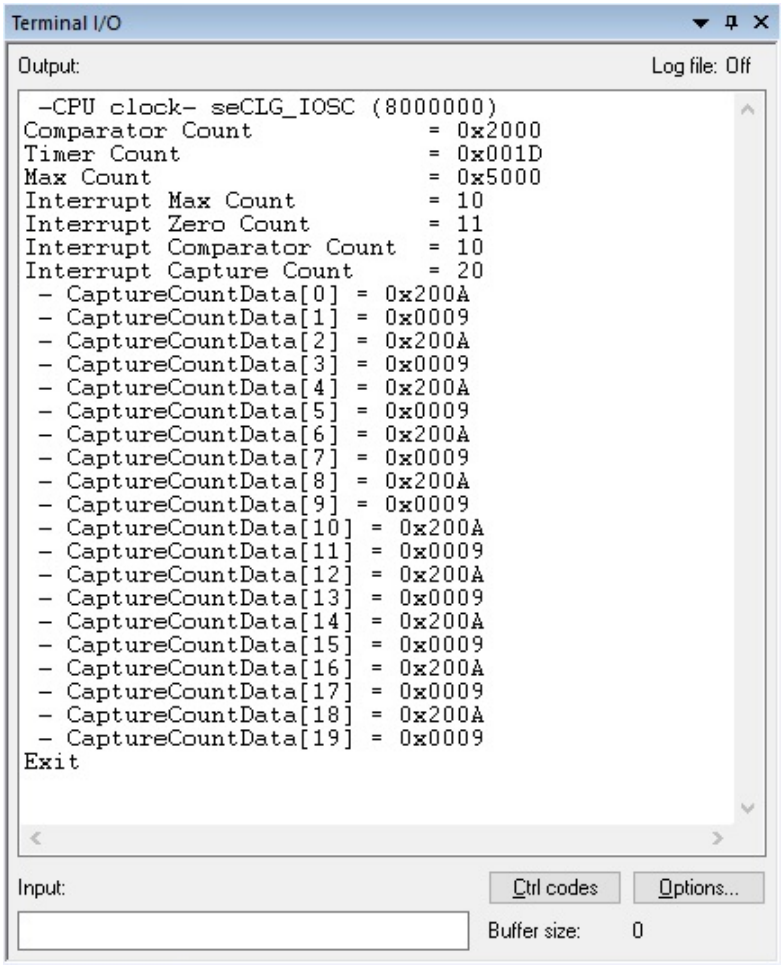
3.17 16-Bit PWM Timer (T16B)

This example exercises the various comparator/capture functions of the 16-Bit PWM Timer (T16B).

Operations

1. Set the 16-bit PWM timer as follows and start the 16-bit PWM timer.
   - Mode: Repeat up-count mode
   - Maximum counter value: 0x5000
   - Comparator/capture circuit 0: Comparator mode (compare buffer: 0x2000)
   - Comparator/capture circuit 1: Number of capture interrupts in capture mode (trigger signal: LOW)
2. Start the 16-bit PWM timer and put CPU in a Halt state.
3. When an interrupt occurs in the 16-bit PWM timer, release CPU from the Halt state.
4. Each time an interrupt occurs in the 16-bit PWM timer, get the number of interrupts below.
   - Number of compare interrupts
   - Number of capture interrupts
   - Number of maximum-count interrupts
   - Number of count-zero interrupts
5. When a compare interrupt occurs, set the trigger signal for capture to HIGH.
6. When a maximum-count interrupt occurs, set the trigger signal for capture to LOW.

Example of Output



3.18 UART (UART3)

This example shows how to program the UART3 to send/receive characters to/from S5U1C31D5xT1 when connected to a PC through a USB-to-UART adapter.

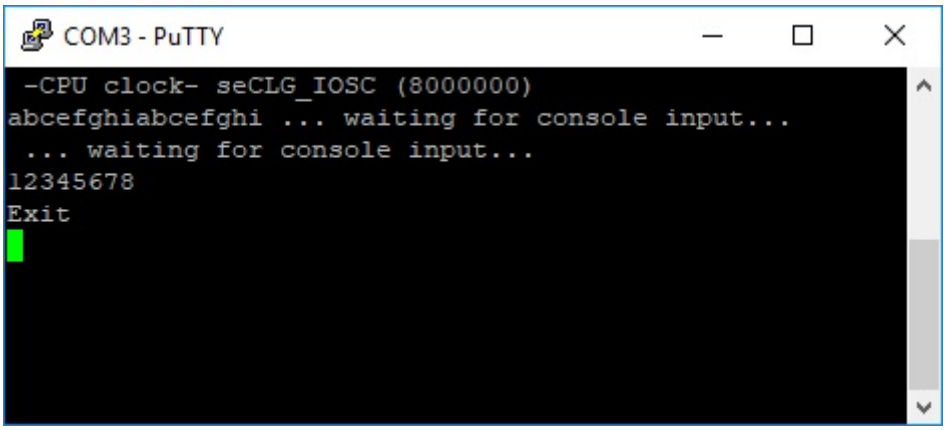
Hardware Setup

1. Connect S5U1C31D5xT1 evaluation board to PC via USB Adapter for UART. For detail, see Figure 2.2.2.1 and 2.2.2.2 in Section 2.2.2.
2. Run a terminal program for UART communication on PC.
3. Configure terminal program for baud rate 115200, 8 data bits, 1 stop bit, non parity.

Operations

1. Example initializes UART3 as below:
   - Baud rate is 115200
   - Data length is 8bit
   - Stop bit length is 1bit
   - Parity is Non parity
2. ASCII characters are transmitted from UART3 and displayed in the terminal program window.
3. Then the Example program waits for the input of the 8 characters.
4. When typing of the 8 characters completed the characters are sent back to the terminal program window.

Example of Output



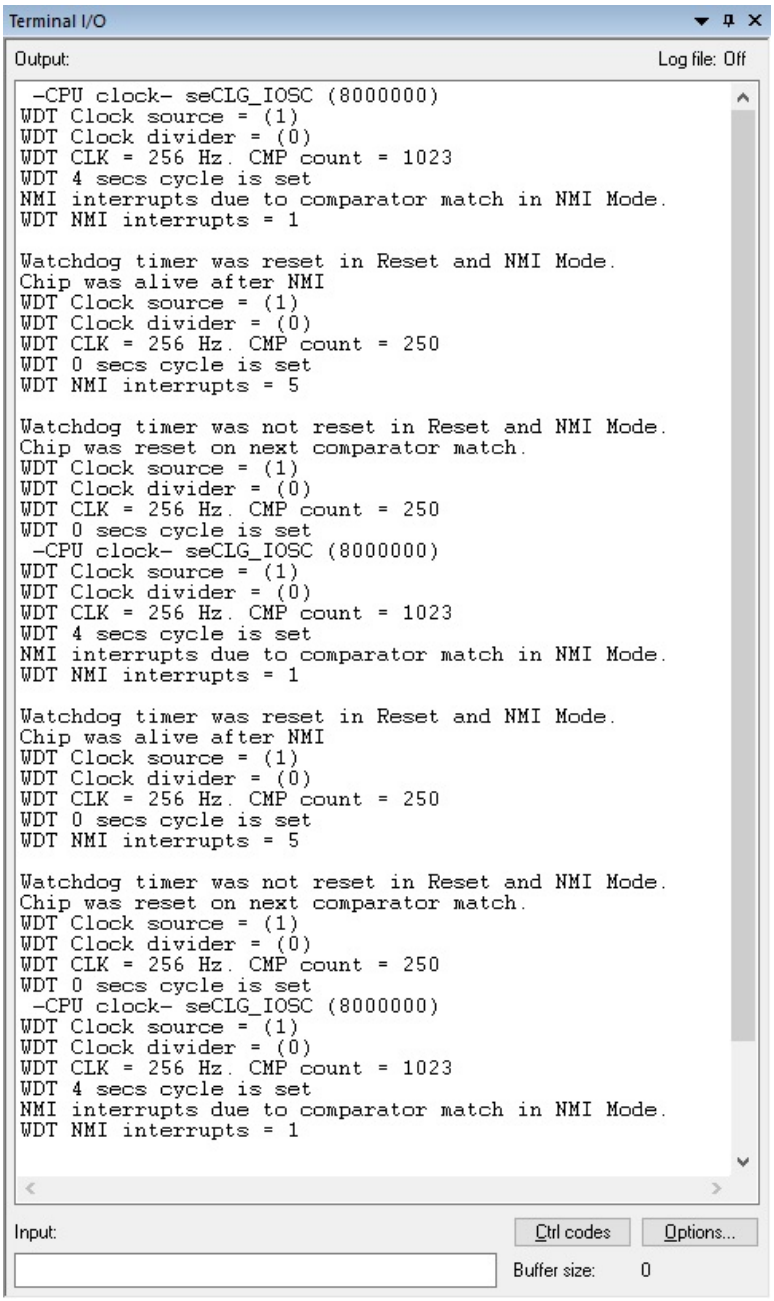
3.19 WatchDog Timer (WDT2)

This example shows how to program the WatchDog Timer (WDT2) to generate an NMI interrupt and a reset.

Operations

1. Initializes WDT2.
2. OSC1 is set as WTD2's clock source. OSC1 is started and configured to operate in sleep mode.
3. Shows NMI interrupts due to comparator match.
4. Second scenario shows Reset after NMI mode when WDT2 is reset.
5. Third scenario shows Reset after NMI mode when WDT2 is not reset.

Example of Output



3.20 Sound Play (SOUNDPLAY)

This example shows how to play the sounds such as voice and music using the sound play function provided by HW Processor. This example can realize the following three audio playback functions.

- Playback on Audio-Amp IC + Speaker
- Playback on discrete circuit + Electromagnetic Buzzer (only S1C31D51)
- Playback on discrete circuit + Piezoelectric Buzzer (only S1C31D51)

This example can be switched these functions in using the board definition shown in Table 3.20.1.

Table 3.20.1 Board Defines for Sound Play (SOUNDPLAY)

Configuration Feature	Defined
PLAY_AMPSPEAKER	Playback on Audio-Amp IC + Speaker
PLAY_ELBUZZER	Playback on discrete circuit + Electromagnetic Buzzer (only S1C31D51)
PLAY_PIBUZZER	Playback on discrete circuit + Piezoelectric Buzzer (only S1C31D51)

(1) Playback on Audio-Amp IC + Speaker

Sample Software Setup

1. Define “PLAY_AMPSPEAKER” as a symbol definition on SOUNDPLAY project. (See Figure 3.20.1, Table 3.20.1.)
2. Build SOUNDPLAY project.

Hardware Setup

1. Check the jumper settings of audio amplifiers on S5U1C31D5xT1. (see Table 3.20.2 and Figure 3.20.2.)
2. Connect the speaker output connector and the speaker via the speaker cable included with the evaluation board. (see Figure 3.20.1.)

Note: Amplifier settings should not be done during power supply. Parts mounted on the board such as amplifier may be damaged. Please switch the amplifier with the power off.

Table 3.20.2 Jumper Settings of Audio Amplifiers

Connectors: JP1/JP2/JP3/J13/JP4/JP5						
Amplifier Type	**JP1**	**JP2**	**JP3**	**J13**	**JP4**	**J5**
Class-AB	Short 2 to 3	Short 1 to 2	Short 1 to 2	Short 1 to 2	Don’t care	Short 9 to 10 Short 11 to 12
Class-D	Don’t care	Short 2 to 3	Short 2 to 3	Don’t care	Short 1 to 2	Short 9 to 10 Short 11 to 12

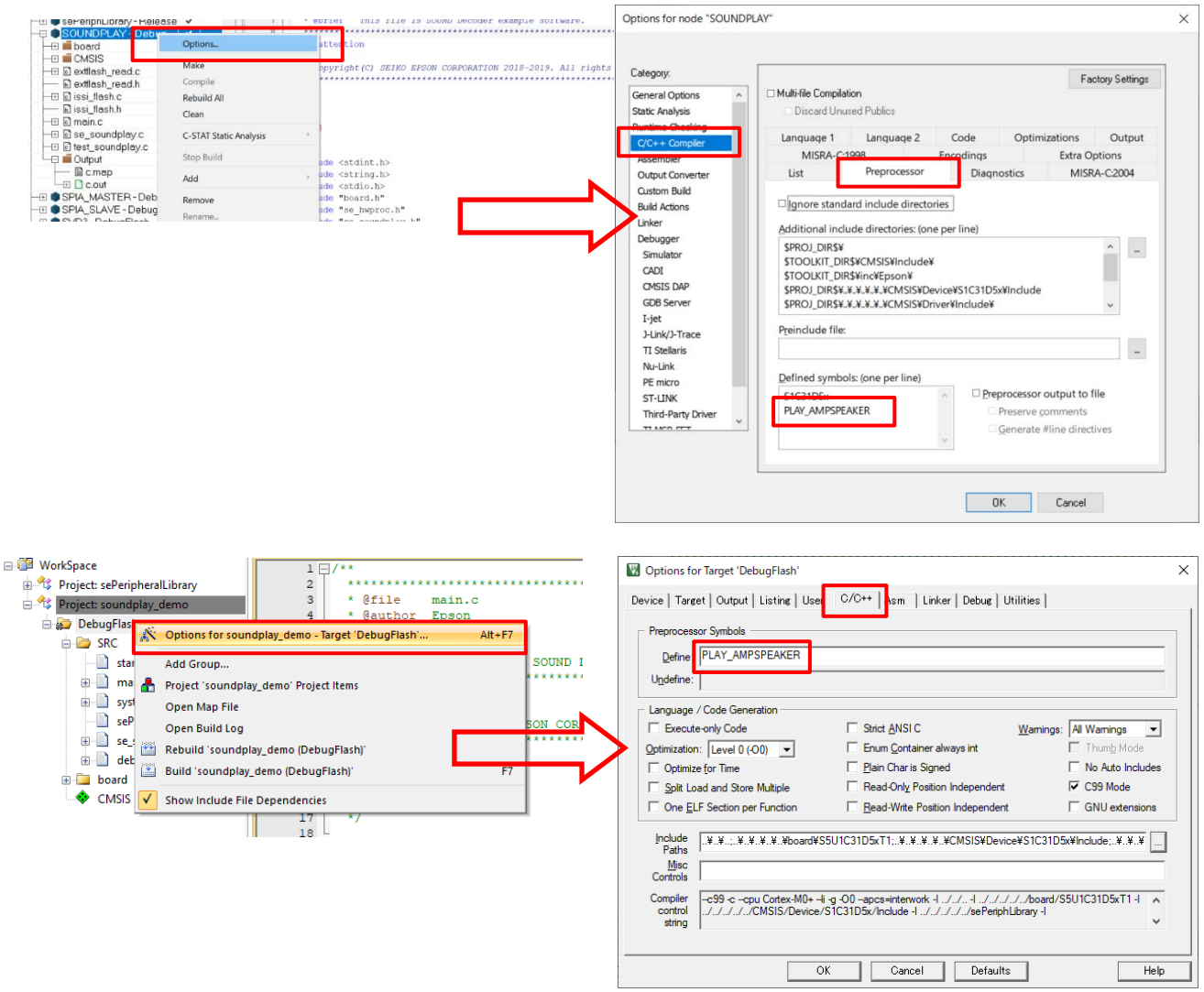


Figure 3.20.1 Symbol Definition (IAR EWARM on Upper figure, Keil MDKARM on Lower figure)

Operations

1. Set the system clock for SOUNDPLAY. (See InitSCLK() function in main.c)
2. Enable the Speaker Amplifier. (See InitAMP() function in main.c)
3. Play sentences composed of sound files in the internal flash memory.
   - 3.1 Call the seSoundPlayInit() function to initialize the SOUNDPLAY.
   - 3.2 Call the seSoundPlaySetParameter() function to set a sentence number, channel, volume, repeat times, and voice speed.
   - 3.3 Call the seSoundPlayRunCommand() function to start sound play of the specified sentence.
   - 3.4 Wait for an interrupt indicating the sound play is finished.
   - 3.5 Repeat steps 3.2 to 3.4 if necessary.
   - 3.6 Call the seSoundPlayFinish() function to finish the SOUNDPLAY.
4. Initialize the QSPI peripheral module to access an external QSPI flash memory (See InitExtFlash() function in extflash_read.c).
5. Play sentences composed of sound files in the external flash memory.
   - 5.1 Call the seSoundPlayInit() function to initialize the SOUNDPLAY.
   - 5.2 Call the seSoundPlaySetParameter() function to set a sentence number, channel, volume, repeat times, and voice speed.
   - 5.3 Call the seSoundPlayRunCommand() function to start sound play of the specified

sentence.
5.4 Wait for an interrupt indicating the sound play is finished.
5.5 Repeat steps 5.2 to 4.4 if necessary.
5.6 Call the seSoundPlayFinish() function to finish the SOUNDPLAY.

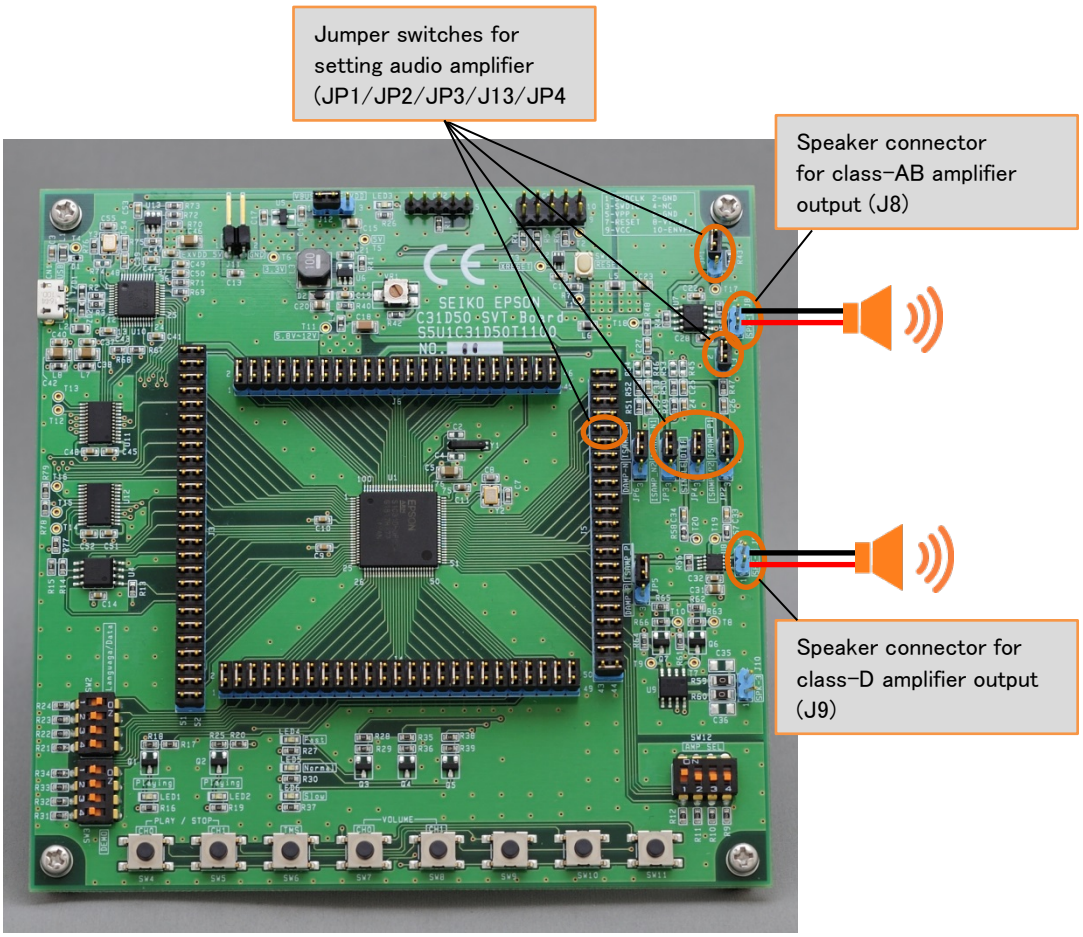


Figure 3.20.2 Jumpers and Connectors for Sound Play

(2) Playback on discrete circuit + Electromagnetic Buzzer (only S1C31D51)

Sample Software Setup

1. Define “PLAY_ELBUZZER” as a symbol definition on SOUNDPLAY project. (see Figure 3.20.1.)
2. Build SOUNDPLAY project.

Hardware Setup

1. Connect Buzzer Evaluation board (S5U1C31D51T2) to S5U1C31D51T1 as matching direction of both silks. (see Table 3.20.3, Figure 3.20.3, Figure 3.20.4)
2. Check the jumper settings and resistances of the board. (see Table 3.20.4, Figure 3.20.4)
3. Connect the Electromagnetic Buzzer (SD160709 made by TDK) to the connector on the Buzzer Evaluation board (S5U1C31D51T2). (see Table 3.20.4)

Table 3.20.3 Settings of the board(S5U1C31D51T1) when connected to S5U1C31D51T2)

Jumper: J3/J4/J5/J6			
J3	**J4**	**J5**	**J6**
51-52 Open	45-46 Open	1-2 Open	1-2 Open
-	-	21-22 Open	43-44 Open
-	-	23-24 Open	45-46 Open
-	-	43-44 Open	-

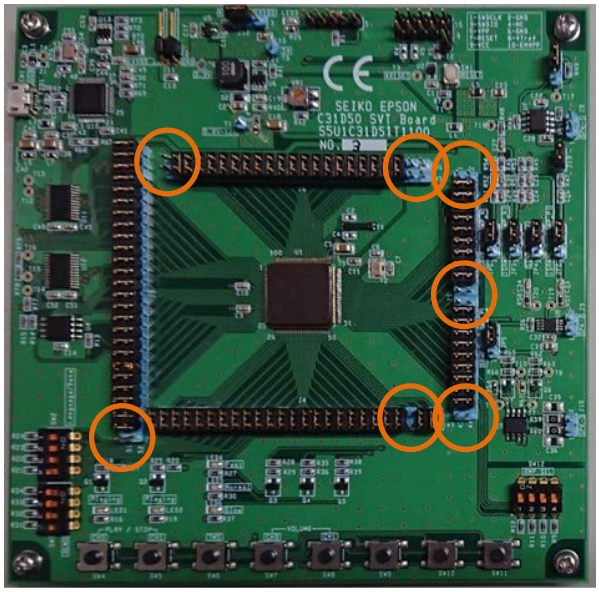


Figure 3.20.3 Settings of the board (S5U1C31D51T1) when connected to S5U1C31D51T2)

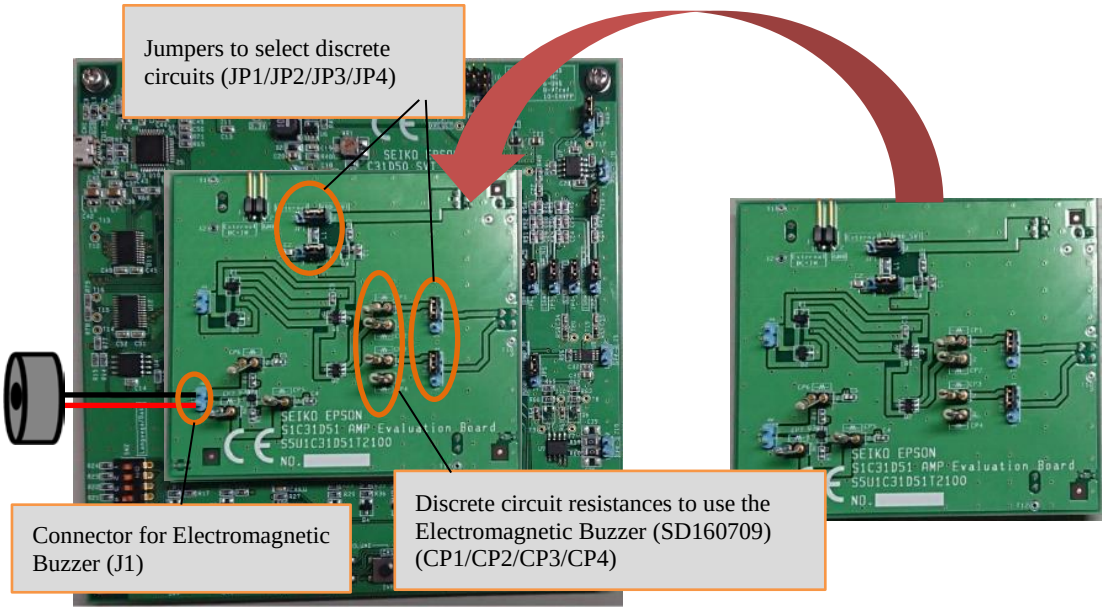


Figure 3.20.4 Settings and resistances of S5U1C31D51T2 when Electromagnetic buzzer is used

Table 3.20.4 Settings and resistances of S5U1C31D51T2 when Electromagnetic buzzer is used

Jumper: JP1/JP2/JP3/JP4			
JP1	**JP2**	**JP3**	**JP4**
1-2 Short	1-2 Short	1-2 Short	1-2 Short

Resistance: CP1/CP2/CP3/CP4			
CP1	**CP2**	**CP3**	**CP4**
2.2kohm	2.2kohm	2.2kohm	2.2kohm

Operations

1. Set the system clock for SOUNDPLAY. (See InitSCLK() function in main.c)
2. Play sentences composed of sound files in the internal flash memory.
   - 2.1 Call the seSoundPlayInit_ELBuzzer() function to initialize the SOUNDPLAY.
   - 2.2 Call the seSoundPlaySetParameter() function to set a sentence number, channel, volume, repeat times, and voice speed.
   - 2.3 Call the seSoundPlayRunCommand() function to start sound play of the specified sentence.
   - 2.4 Wait for an interrupt indicating the sound play is finished.
   - 2.5 Repeat steps 3.2 to 3.4 if necessary.
   - 2.6 Call the seSoundPlayFinish_ELBuzzer() function to finish the SOUNDPLAY.
3. Initialize the QSPI peripheral module to access an external QSPI flash memory (See InitExtFlash() function in extflash_read.c).
4. Play sentences composed of sound files in the external flash memory.
   - 4.1 Call the seSoundPlayInit_ELBuzzer() function to initialize the SOUNDPLAY.
   - 4.2 Call the seSoundPlaySetParameter() function to set a sentence number, channel, volume, repeat times, and voice speed.
   - 4.3 Call the seSoundPlayRunCommand() function to start sound play of the specified sentence.
   - 4.4 Wait for an interrupt indicating the sound play is finished.
   - 4.5 Repeat steps 5.2 to 4.4 if necessary.
   - 4.6 Call the seSoundPlayFinish_ELBuzzer() function to finish the SOUNDPLAY.

Notes:

In the discrete circuit for the electromagnetic buzzer mounted on S5U1C31D51T21, a large current may flow depend on the input signal of the circuit. Therefore, the power of this discrete circuit can be controlled ON/OFF by the port(P46) of S1C31D51, and the power is turned ON (P46 is controlled) in the initialization function (seSoundPlayInit_ELBuzzer) prepared in this sample software. And after the playback, the power is turned off in the end function (seSoundPlayFinish_ELBuzzer) that is called after playback is finished, and the power is turned on only during playback.

(3) Playback on discrete circuit + Piezoelectric Buzzer (only S1C31D51)

Sample Software Setup

1. Define “PLAY_PIBUZZER” as a symbol definition on SOUNDPLAY project. (see Figure 3.20.1.)
2. Build SOUNDPLAY project.

Hardware Setup

1. Connect Buzzer Evaluation board (S5U1C31D51T2) to S5U1C31D51T1 as matching direction of both silks. (see Table 3.20.3, Figure 3.20.3, Figure 3.20.4)
2. Check the jumper settings and resistances of the board. (see Table 3.20.5)
3. Connect the Piezoelectric Buzzer (PS1740P02CE made by TDK) to the connector on the Buzzer Evaluation board (S5U1C31D51T2). (see Table 3.20.5)

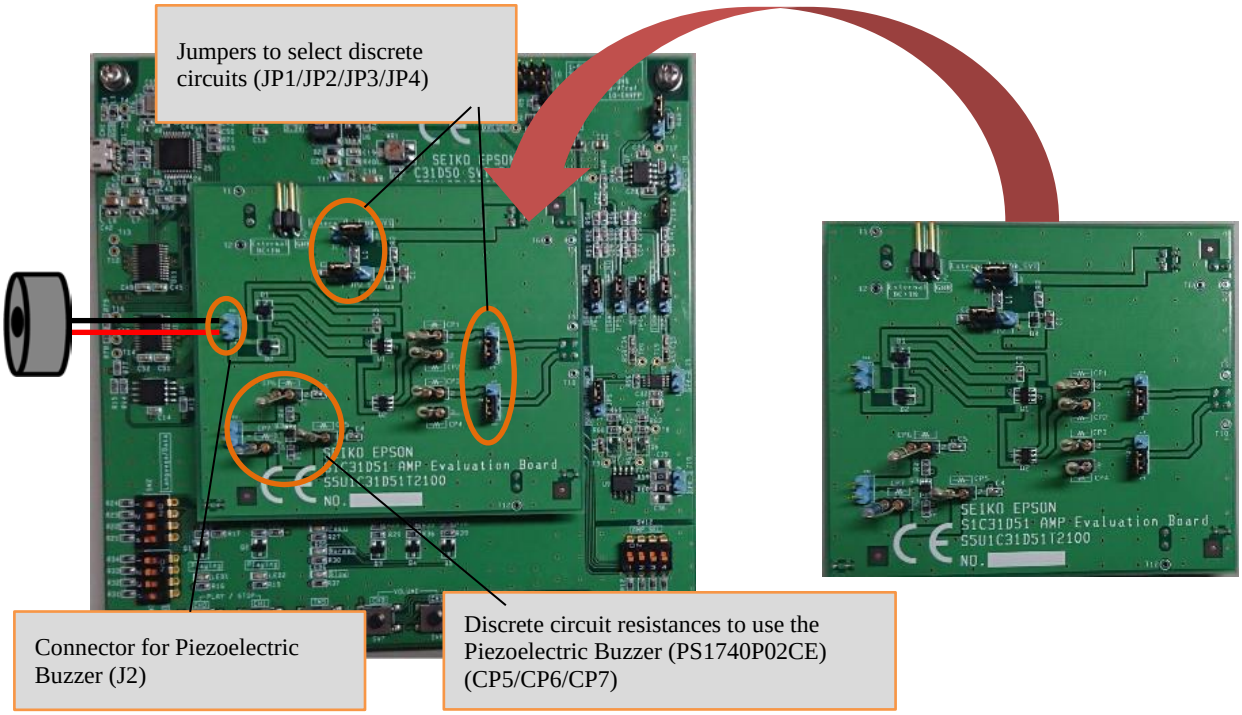


Figure 3.20.5 Settings and resistances of S5U1C31D51T2 when Piezoelectric buzzer is used

Table 3.20.5 Settings and resistances of S5U1C31D51T2 when Piezoelectric buzzer is used

Jumper: JP1/JP2/JP3/JP4			
JP1	**JP2**	**JP3**	**JP4**
1-2 Short	2-3 Short	2-3 Short	2-3 Short

Resistance: CP5/CP6/CP7		
CP5	**CP6**	**CP7**
180 ohm	180 ohm	100 ohm

Operations

1. Set the system clock for SOUNDPLAY. (See InitSCLK() function in main.c)
2. Play sentences composed of sound files in the internal flash memory.
   2.1 Call the seSoundPlayInit_PIBuzzer() function to initialize the SOUNDPLAY.
   2.2 Call the seSoundPlaySetParameter() function to set a sentence number, channel, volume, repeat times, and voice speed.
   2.3 Call the seSoundPlayRunCommand() function to start sound play of the

specified sentence.
2.4 Wait for an interrupt indicating the sound play is finished.
2.5 Repeat steps 3.2 to 3.4 if necessary.
2.6 Call the seSoundPlayFinish_PIBuzzer() function to finish the SOUNDPLAY.

3. Initialize the QSPI peripheral module to access an external QSPI flash memory (See InitExtFlash() function in extflash_read.c).
4. Play sentences composed of sound files in the external flash memory.
   4.1 Call the seSoundPlayInit_PIBuzzer() function to initialize the SOUNDPLAY.
   4.2 Call the seSoundPlaySetParameter() function to set a sentence number, channel, volume, repeat times, and voice speed.
   4.3 Call the seSoundPlayRunCommand() function to start sound play of the specified sentence.
   4.4 Wait for an interrupt indicating the sound play is finished.
   4.5 Repeat steps 5.2 to 4.4 if necessary.
   4.6 Call the seSoundPlayFinish_PIBuzzer() function to finish the SOUNDPLAY.

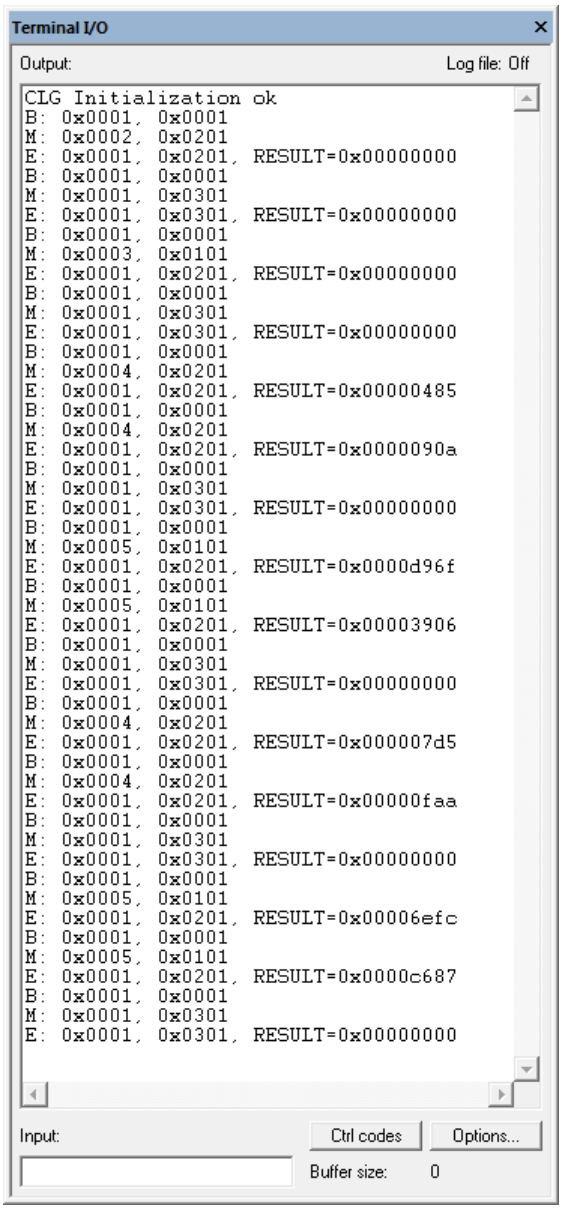
3.21 Memory Check (MEMCHECK)

This example shows how to check internal RAM, internal flash memory or external QSPI flash memory using the memory check function provided by HW Processor.

Operations

1. Set the system clock for MEMCHECK. (See InitSCLK() function in main.c)
2. Perform the RAM check by data reading/writing for internal RAM.
3. Perform the RAM check by March-C algorithm for internal RAM.
4. Perform the checksum for internal flash memory.
5. Perform the CRC-16 for internal flash memory.
6. Initialize the QSPI peripheral module to access an external QSPI flash memory (See InitExtFlash() function in extflash_read.c).
7. Perform the checksum for external QSPI flash memory.
8. Perform the CRC-16 for external QSPI flash memory.

Example of Output



3.22 Flash Programming (FLASH)

This example shows how to erase and program the internal flash memory using the seFlashLibrary.

The seFlashLibrary is implemented based on the specification of the CMSIS-Driver Flash Interface.

Operations

1. Call the GetInfo() function to get the information on the internal flash memory.
2. Call the GetVersion() function to get the version of the flash driver.
3. Call the Initialize() function to initialize the flash driver.
4. Call the EraseSector() function to erase the sector specified by the argument.
5. Call the ProgramData() function to write the data to the sector specified in step 4.
6. Call the ReadData() function to read the data from the sector specified in step 4, and compare the read data with the data written in step 5.
7. Call the Uninitialize() function to make the flash driver uninitialized.

Usage Notes of seFlashLibrary

Please note the following points when using this library.

- Make sure to disable interrupts before calling functions provided in this library.
- When executing this library, do not destroy the area where the library is placed.
- When using this library, pay attention to the number of times Flash memory can be rewritten. For the specifications of flash memory, refer to "S1C31D5x Technical Manual".
- This library uses the 16-bit timer (T16) ch.0. Therefore, after executing this library, the register setting of the 16-bit timer (T16) ch.0 has been changed. Be careful when using this library with a program using the 16-bit timer (T16) ch.0.
- This library switches the system clock to IOSC and executes it. Therefore, after executing the library, the register setting of the clock generator (CLG) has been changed. If necessary, take countermeasures such as backing up register settings of CLG before library execution and restoring register setting of CLG after library execution.

3.23 EEPROM Library (EEPROM)

This example shows how to write the data to the emulated EEPROM which uses the internal flash memory using the seEepromLibrary, and then read back and compare the data to expected values. Emulated EEPROM uses 256 bytes of internal Flash memory for each byte of EEPROM. The Flash area from 0x20000 to 0x2FFFF is used for EEPROM emulation and provides 256 bytes of EEPROM with 100,000 read/write cycles.

The seEepromLibrary is implemented based on the specification of the CMSIS-Driver Flash Interface. The header file for the emulated EEPROM routines is in CMSIS¥Driver¥Include¥Driver_EEPROM.h.

Before using the emulated EEPROM, a one-time erase of the Flash area from 0x20000 to 0x3FFFF must be performed.

Parameter Settings

To change the starting address of the internal Flash area for the emulate EEPROM, edit the linker configuration file.

For IAR EWARM, the internal Flash area of the emulated EEPROM is located in the section “EEPROM1”. To change the starting address, redefine the memory region of “EEPROM1” in the “Examples/EEPROM/board/S5U1C31D5xT1/IAR/config/S1C31D5x_flash.icf”.

For MDK-ARM(µVision), the internal Flash area of the emulated EEPROM is located in the section “ER_IROM2”. To change the starting address, redefine the memory region of “LR_IROM2” and “ER_IROM2” in “Examples/EEPROM/board/S5U1C31D5xT1/ARM/eeprom_flash.scf”.

To change the size of emulated EEPROM and the number of the retries, edit “Driver_EEPROM.h” to redefine the constants shown below.

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

The “CONFIG_EEPROM_SIZE_MAX” indicates the size of the emulated EEPROM. This size can be selected one of 32/64/128/256.

The “CONFIG_ RETRY_COUNT” indicates the number of write retries when a writing has failed. Increasing the number of write retries causes the processing time of the writing routine to increase and performance to decrease. So it should only be set to several times.

Operations

1. Call the GetInfo() function to get the information on the emulated EEPROM, if necessary.
2. Call the GetVersion() function to get the version of the emulated EEPROM driver, if necessary.
3. Call the ProgramData() function to write the data to emulated EEPROM.
4. Call the ReadData() function to read the data from emulated EEPROM.

Usage Notes of seEepromLibrary

Please note the following points when using this library.

- Make sure to disable interrupts before calling functions provided in this library.
- When executing this library, do not destroy the area where the library is placed.
- This library uses the 16-bit timer (T16) ch.0. Therefore, after executing this library, the register setting of the 16-bit timer (T16) ch.0 has been changed. Be careful when using this library with a program using the 16-bit timer (T16) ch.0.
- This library switches the system clock to IOSC and executes it. Therefore, after executing the library, the register setting of the clock generator (CLG) has been changed. If necessary, take countermeasures such as backing up register settings of CLG before library execution and restoring register setting of CLG after library execution.

3.24 Boot Loader (BootLoader)

This example loads a program transmitted from the outside by UART communication.

For details of this example, refer to "S1C31D5x Boot Loader Manual" on the separate document.

4 Demo Software

The demo software is preinstalled in the internal flash of S1C31D5x on S5U1C31D5xT1 evaluation board. The source code of this demo software is included in “Demos¥SOUNDPLAY_DEMO” in the sample software package.

The demo software has

- Playback on Audio-Amp IC + Speaker
- Playback on discrete circuit + Electromagnetic Buzzer (only S1C31D51)
- Playback on discrete circuit + Piezoelectric Buzzer (only S1C31D51)

three playback modes as above.

The demo software has two modes as follows:

- Sound Play Demo Mode (Mode to demonstrate the sound play by operating the push switches.)
- Sound ROM Update Mode (Mode to update the sound ROM data in the external QSPI flash memory.)

4.1 Hardware Setup

Figure 4.1.1 shows the layout of main parts on S5U1C31D5xT1 evaluation board to run the demo software. Table 4.1.1 to 4.1.4 show the default settings of the jumpers and the DIP switches on S5U1C31D5xT1.

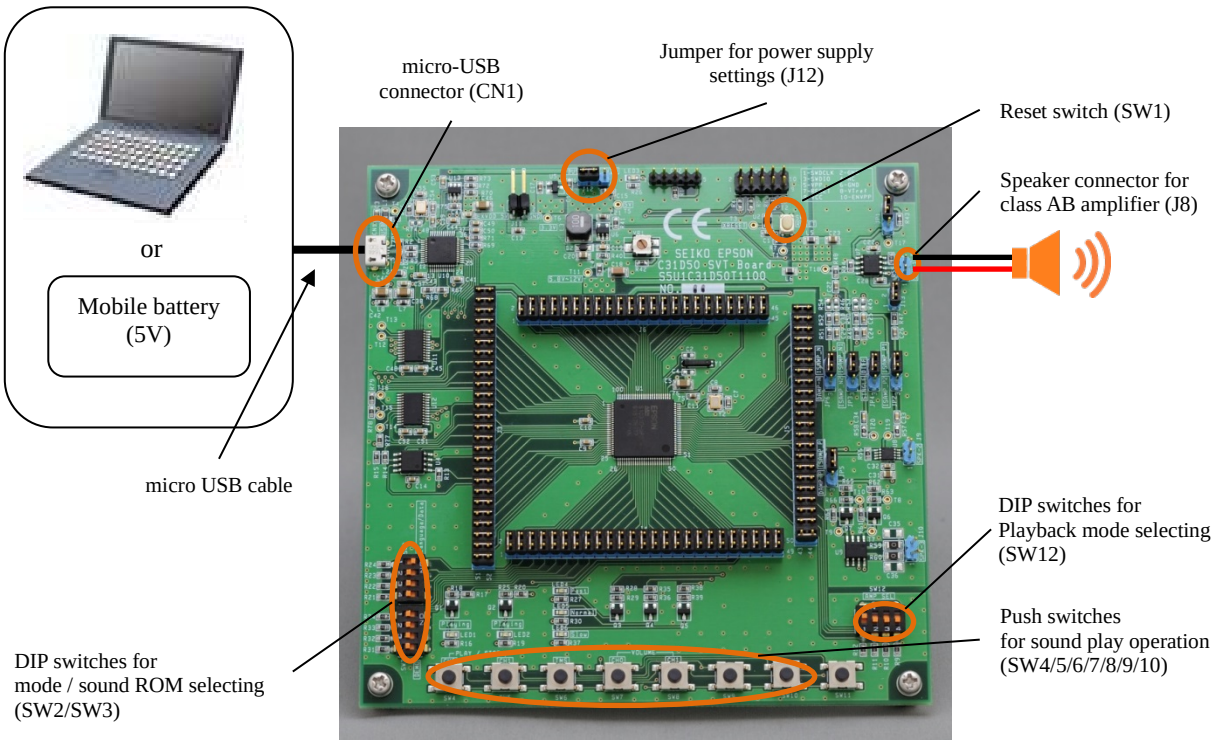


Figure 4.1.1 Layout of Main Parts on the evaluation board (S5U1C31D5xT1)

Table 4.1.1 Jumper for Power Supply Settings

J12	Remark
Short 1 to 2	default

Table 4.1.2 Jumper for Class-AB Amplifier Settings

JP1	JP2	JP3	J13	JP4	Remark
2-3 short	1-2 short	1-2 short	1-2 short	1-2 short	default

Table 4.1.3 DIP Switch for Storage Selection (SW2-1)

SW2-1	Description	Remark
OFF	Access to sound ROM in internal flash	
ON	Access to sound ROM in external QSPI flash memory.	default

Table 4.1.4 DIP Switch for Mode Selection (SW3-1)

SW3-1	Description	Remark
OFF	Sound Play Demo Mode	default
ON	Sound ROM Update Mode	

4.2 How to Run Sound Play Demo Mode

In the “Sound Play Demo Mode”, you can operate the push switches (SW4, SW5, ..., SW10) on the evaluation board to run the sound playback such as 2 channel mixing playback and voice speed conversion playback.

The procedure for running this mode is as follows.

1. Set J12 to “VBUS 5V”. (Short J12-1 and J12-2)
2. Connect CN1 to micro USB cable.
3. Connect a micro USB cable to mobile battery or PC to supply the 5V power.
4. Set SW3-1 to “OFF” to select “Sound Play Demo Mode”.
5. In case of playback mode on Electromagnetic/Piezoelectric Buzzer, connect S5U1C31D51T2 to S5U1C31D51T1 and set “JP” and resistances of “CP” properly (see 3.20). Set SW2-1 to “OFF” to access to the sound ROM data in the internal flash memory. Set SW2-1 to “ON” to access to the sound ROM data in the external flash memory.
6. When external flash memory access is selected in step 5, select the playback mode with SW12 and the sound ROM data with SW2 and SW3. (For detail of SW2/SW3/SW12 settings, refer toS1C31D50/D51 Evaluation Board Getting Started)
7. Press the SW1(RESET) to reset the evaluation board.
8. Press the SW4(PLAY/STOP-CH0) and/or SW5(PLAY/STOP-CH1) to start the sound play. (Refer to Figure 4.2.1 and Table 4.2.1 for details of the sound play operation.)

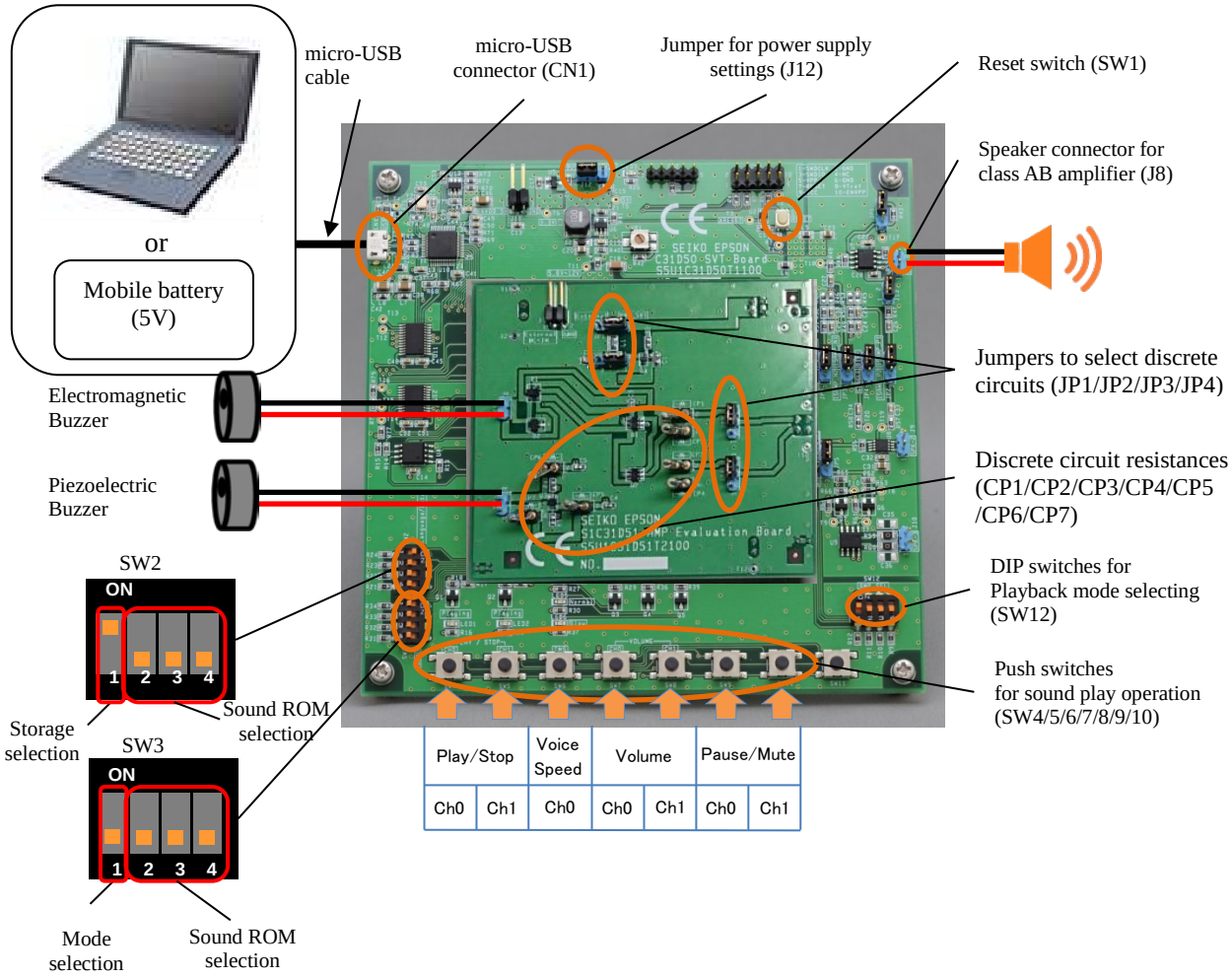


Figure 4.2.1 Layout of Main Parts (Sound Play Demo Mode)

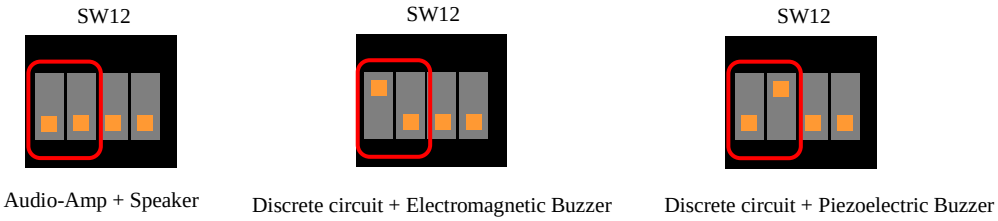


Figure 4.2.2 Setting of playback mode (SW12)

Table 4.2.1 Details of Push Switches

Push SW	Operation	Description
SW4/SW5 (Play/Stop for Ch.0,Ch.1)	Press with single-click	This operation starts playback of the sentence. Every time this operation is performed, playback of the next sentence is started. This operation during sound playing stops playback and starts playback of the next sentence.
	Press with double-click	This operation starts playback of the sentence. Every time this operation is performed, playback of the previous sentence is started. This operation during sound playing stops playback and starts playback of the previous sentence.
	Long Press	This operation stops playback. The operation during pausing cancels pause and stops playback. The operation during muting cancels mute and stops playback. This operation while stopped returns the sentence to be played back to the initial state.
SW6 (Voice Speed for Ch.0)	Press with single-click	This operation slows down the voice speed in steps of 5%. When this operation is performed during playing, playback of the current sentence is stopped and the same sentence is played back at a lower speed.
	Press with double-click	This operation speeds up the voice speed in steps of 5%. When this operation is performed during playing, playback of the current sentence is stopped and the same sentence is played back at a higher speed.
	Long Press	This operation turns the voice speed back to default value.
SW7/SW8 (Volume for Ch.0,Ch.1)	Press with single-click	This operation decreases the volume in 3dB steps.
	Press with double-click	This operation increases the volume in 3dB steps.
	Long Press	This operation turns the volume back to default value.
SW9/SW10 (Pause/Mute for Ch.0,Ch.1)	Press with single-click	This operation pauses the sound play. This operation during pausing, the pause will be released. This operation during muting, the mute will be released.
	Press with	This operation mutes the sound play.

	double-click	

4.3 How to Sound ROM Update Mode

In the “Sound ROM Update Mode”, you can rewrite the sound ROM data in the external QSPI flash memory.

The procedure for running this mode is as follows.

1. Create the sound ROM data (ROMImage_C31D5x_*.bin) for S1C31D5x using ESPER2(Speech Data Creation Tool by EPSON).
2. Set J12 to “VBUS 5V”. (Short J12-1 and J12-2)
3. Connect CN1 to micro USB cable.
4. Connect a micro USB cable to PC.
5. Set SW3-1 to “ON” to select “Sound ROM Update Mode”.
6. Press the SW1(RESET) to reset the evaluation board. After resetting, confirm that LED1, LED2 and LED6 are lit.
7. Copy the sound ROM data(ROMImage_C31D5x_*.bin) created by ESPER 2 and the playlist file(PlayList_C31D5x_*.txt) generated at the same time as the sound ROM data to the folders in the sample software package shown below.
   - FlashTools¥S5U1C31D5xT1
8. Run the following batch files to update a sound ROM data and a playlist file to external QSPI flash memory.
   - FlashTools¥S5U1C31D5xT1/run_write_flash_romdata.bat
   - FlashTools¥S5U1C31D5xT1/run_write_flash_playlist.bat

Also, by editing the playlist file (PlayList_C31D5x_*.txt), you can select and sort sentences to be played back. For details of the playlist file, refer to "Appendix.B. Playlist File".

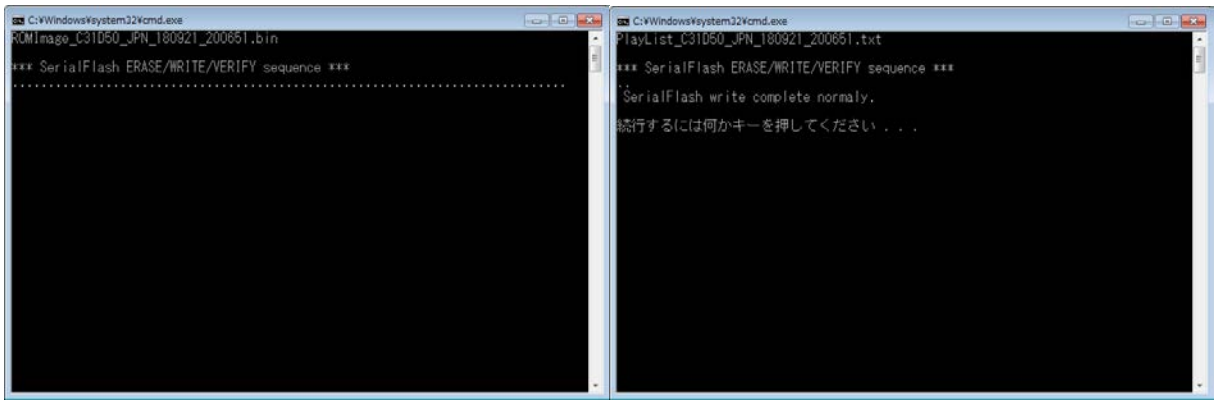


Figure 4.3.1 Screen Display when Batch Files is Executed

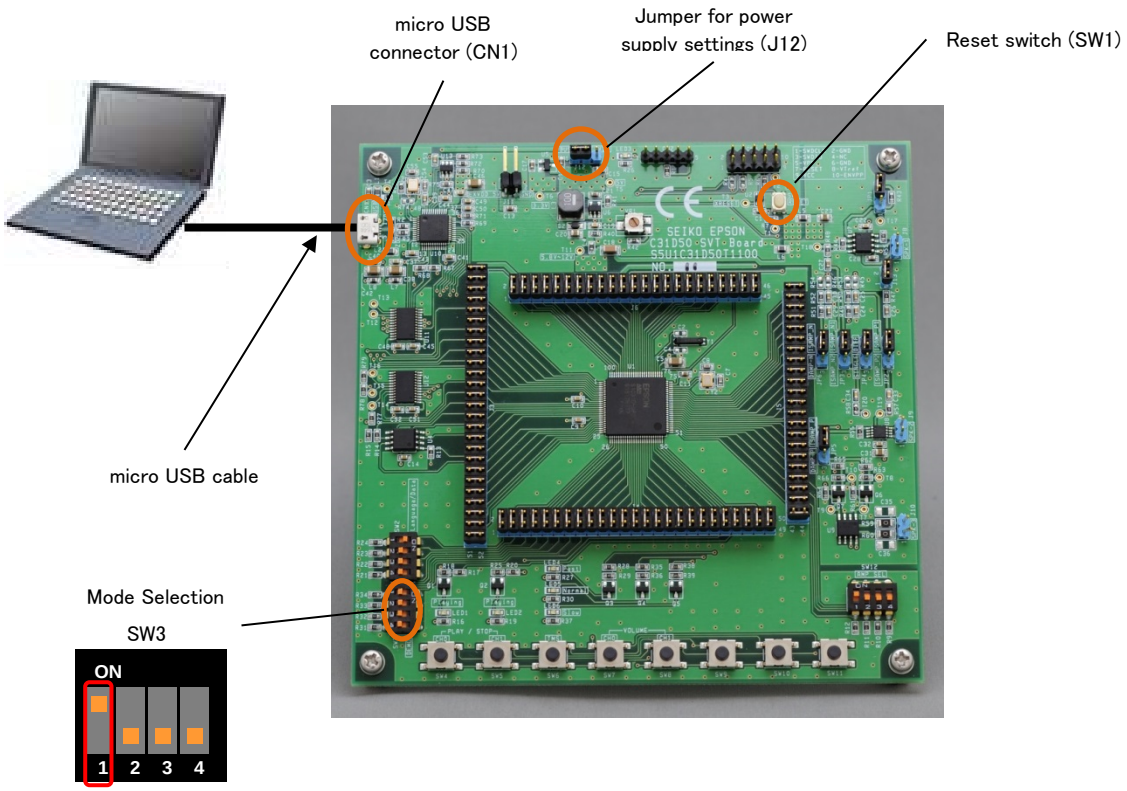


Figure 4.3.2 Layout of Main Parts (Sound ROM Update Mode)

5 Self-Testing Sample Software (Compliance with the IEC60730 Standard)

5.1 Description

The International Electrotechnical Commission (IEC) has issued the IEC 60730 standard for development of household appliances. Consumer electronics sold and used in Europe are required by law to comply with this safety standard. The purpose of this standard is to protect consumers from hazards arising from malfunctions and defects in final products by discovering them in a timely manner through periodic self-testing.

Software control for microcontrollers is categorized according to the following standards.

Class A: Control functions not intended to be relied upon for the safety of the equipment (e.g., lighting fixtures)

Class B: Control functions intended to prevent unsafe operation of the controlled equipment (e.g., washing machines, refrigerators, freezers, dishwashers)

Class C: Control functions intended to prevent special hazards (e.g., combustion appliances)

The majority of control software for household appliances falls under Class B, and the following self-testing is recommended for final products.

- Diagnosis of microcontroller and program counter stack failure
- Diagnosis of interrupt cycle abnormalities
- Diagnosis of abnormalities in the operating clock frequency of the microcontroller
- Diagnosis of abnormalities in the ROM/RAM memory
- Diagnosis of communication errors with external interfaces

For more detailed information, refer to Annex H of IEC 60730.

This document covers the sample software and reference information for self-testing S1C31D5x.

Self-testing includes:

- Memory failure test (read/write test, March-C test)
- Integrity testing of data in the memory (generates checksum and CRC)
- Interrupt test (interrupt cycle and interrupt count check)
- Main clock stability test (operating frequency check)

The read/write test and March-C test perform a read and write test in ranges specified for the memory, register, stack pointer, and status register.

For generating a checksum and CRC, an error detection code is requested and returned for data in the memory in the specified range.

The interrupt test counts how many interrupts occur in a certain period of time and returns that as a value.

The main clock stability test uses the sub-clock (32KHz) to check that the main clock is operating at a normal operating frequency.

5.2 Sample

The source code for this self-testing sample software is included in the folder of "IEC60730 compliant" sample software package Table 5.1.1. shows the list of sample files for self-testing. See Appendix D for details on each sample.

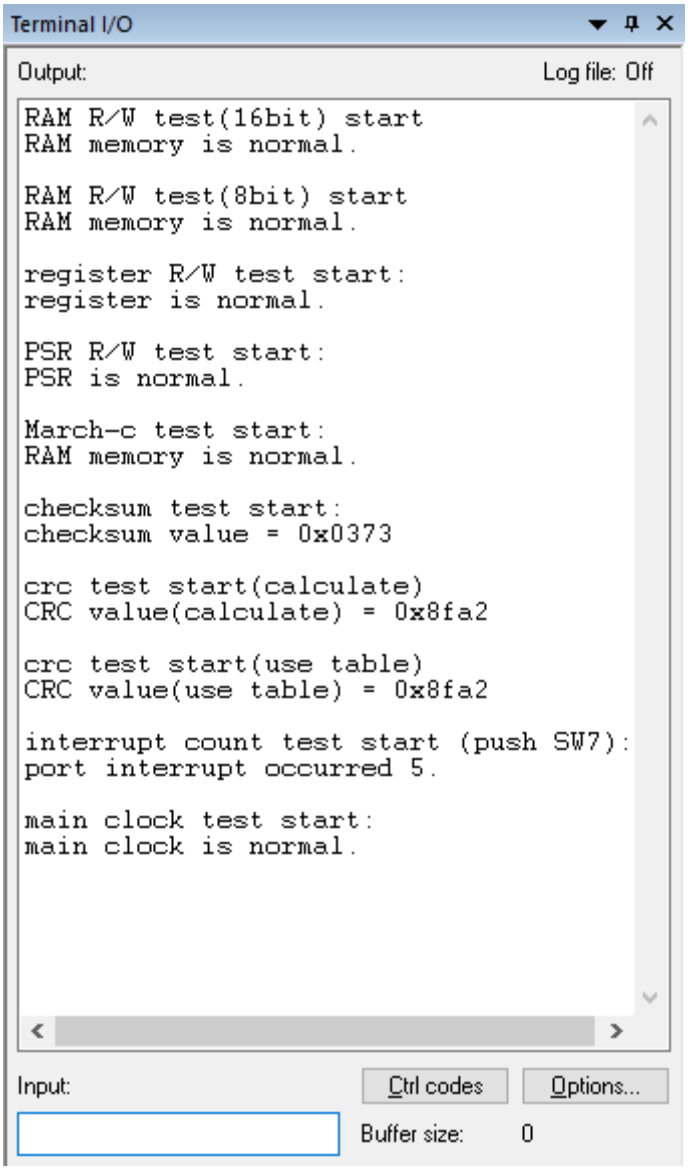
Table 5.1.1 List of sample files for Self-testing

File name	Content
main.c	Calls the test function
s1c31TestRam16.c	RAM R/W test (for 16-bit devices)
s1c31TestRam8.c	RAM R/W test (for 8-bit devices)
s1c31TestRegister.c	General purpose register, stack pointer R/W test
s1c31TestRegister.s	General purpose register, stack pointer R/W test
s1c31TestPsr.c	Status register R/W test
s1c31RwPsr.s	Software exception for causing the status register R/W test to be performed
s1c31TestRamMarchc.c	RAM March-C test
s1c31TestChksum.c	Calculates checksum
s1c31TestCrc.c	Calculates CRC (calculation)
s1c31TestCrcTbl.c	Calculates CRC (table lookup)
s1c31d5xTestInterupt.c	Interrupt test
s1c31d5xTestClk.c	Main clock stability test
s1c31SelfTest.h	Header file used by the self-testing sample program

Operations

1. Run RAM Read/Write test (for 16bit device)
2. Run RAM Read/Write test (for 8bit device)
3. Run Register Read/Write test
4. Run PSR Read/Write test
5. Run RAM Read/Write test by March-C
6. Run to calculate checksum
7. Run to calculate CRC
8. Run interrupt test (port interrupt test depending on the number if presses by SW7)
9. Run clock test

Example of Output



Appendix. A. HW Processor Library Specification

The S1C31D5x HW Processor library is a set of drivers to use HW Processor Functions. The drivers are responsible for modules initialization, features configuration, modules access. The drivers take advantage of chip peripheral addresses and register layout description in the Device Definition file.

Appendix. A.1 SOUNDPLAY

seSoundPlayInit

Syntax	void seSoundPlayInit(unsigned long romStartAddress, unsigned long romSize, unsigned long keycode)
Arguments	romStartAddress Start address of the sound ROM data. In case of internal Flash: 0x00 0000, …, 0x02 FFF0 (16 Byte alignment) In case of external QSPI-Flash: 0x00 0000 + OFFSET 0x10 0000 + OFFSET 0x20 0000 + OFFSET ... 0xE0 0000 + OFFSET 0xF0 0000 + OFFSET * The “OFFSET” is 0x04 0000, the start address of the memory mapped access area for external Flash memory (refer to “Figure 4.1.1 Memory Map” in “S1C31D50/D51 Technical Manual”). romSize Size of the sound ROM data. In case of internal Flash: 0x03 0000 Byte (192 KByte) or less In case of external QSPI-Flash: 0x10 00000 Byte(16 MByte) or less keycode 32bit value required to decode the eov files in sound ROM data. 0x**** **** **** ****
Return Value	Nothing
Explanation	This function initializes SOUNDPLAY. Call this function first to use SOUNDPLAY in using demo software for Audio Amp and Speaker.

seSoundPlayInit_ELBuzzer

Syntax	Same as seSoundPlayInit
Arguments	Same as seSoundPlayInit
Return Value	Nothing
Explanation	This function initializes SOUNDPLAY. Call this function first to use SOUNDPLAY in using demo software for Electromagnetic Buzzer.

seSoundPlayInit_PIBuzzer

Syntax	Same as seSoundPlayInit
Arguments	Same as seSoundPlayInit
Return Value	Nothing
Explanation	This function initializes SOUNDPLAY. Call this function first to use SOUNDPLAY in using demo software for Piezoelectric Buzzer.

seSoundPlaySetParameter

Syntax	void seSoundPlaySetParameter unsigned char ch, unsigned short sentenceNo, unsigned char volume, unsigned char repeat unsigned char speed)
Arguments	ch Channel of SOUNDPLAY. 0: Channel 0 1: Channel 1 sentenceNo Sentence number to play. 1(0x0001), …, 65535(0xFFFF) volume Sound volume. 0x7F: 0dB 0x7E: -0.5dB 0x7D: -1.0dB … 0x02: -63dB 0x01: -63.5dB 0x00: No sound repeat Repeat times of sentence play. 0x01, …, 0xFE: repeat times 0xFF: infinite loop speed Sound speed percentage (only channel 0). 75: 75% Slow 80: 80% 85: 85% 90: 90% 95: 95% 100: 100% Noraml Speed 105: 105% 110: 110% 115: 115% 120: 120% 125: 125% Fast (5% step)
Return Value	nothing
Explanation	This function sets parameters for SOUNDPLAY. Call this function before calling seSoundPlayRunCommand().

seSoundPlayRunCommand

Syntax	void seSoundPlayRunCommand unsigned char ch, unsigned short command)
Arguments	ch Channel of SOUNDPLAY. 0: Channel 0 1: Channel 1 2: All Channel command Command to control SOUNDPLAY. Set one of the following command definitions. 0x01 (sp_command_start): Sound Start 0x02 (sp_command_stop): Sound Stop Immediately 0x03 (sp_command_stop_after_phrase): Sound Stop after current phrase 0x04 (sp_command_pause): Pause immediately 0x05 (sp_command_pause_after_phrase): Pause after current phrase 0x06 (sp_command_unpause): Release Pause(UNPAUSE) 0x07 (sp_command_mute): Mute immediately 0x08 (sp_command_mute_after_phrase): Mute after current phrase 0x09 (sp_command_unmute): Release Mute immediately(UNMUTE)
Return Value	nothing
Explanation	This function runs the command. Call this function after calling seSoundPlaySetParameters().

seSoundPlayGetState

Syntax	unsigned short seSoundPlayGetState unsigned char ch)
Arguments	ch Channel of SOUNDPLAY. Set 0 or 1.
Return Value	Current state of SOUNDPLAY on the channel specified by the argument. 0x0000 (sp_state_init): On Initializing 0x0001 (sp_state_idle): Idle 0x0002 (sp_state_play): On playing 0x0003 (sp_state_pause): On pausing 0x0004 (sp_state_mute): On muting
Explanation	This function gets current state of SOUNDPLAY.

seSoundPlayFinish

Syntax	void seSoundPlayFinish(void)
Arguments	Nothing
Return Value	Nothing
Explanation	This function finishes SOUNDPLAY for Audio Amp and Speaker.

seSoundPlayFinish_ELBuzzer

Syntax	void seSoundPlayFinish_ELBuzzer(void)
Arguments	nothing
Return Value	nothing
Explanation	This function finishes SOUNDPLAY for electromagnetic Buzzer.

seSoundPlayFinish_PIBuzzer

Syntax	void seSoundPlayFinish_ELBuzzer(void)
Arguments	nothing
Return Value	nothing
Explanation	This function finishes SOUNDPLAY for Piezoelectric Buzzer.

Appendix. A.2 MEMCHECK

seMemCheckInit

Syntax	void seMemCheckInit(void)
Arguments	nothing
Return Value	nothing
Explanation	This function initializes MEMCHECK. Call this function first to use MEMCHECK.

seMemCheckSetParameter

Syntax	void seMemCheckSetParameter unsigned long memStartAddress, unsigned long memSize, unsigned long initValue)
Arguments	memStartAddress Start address of the memory area to be checked. In case of RAM: 0x15 0000, ..., 0x15 1FFF 0x15 3000, ..., 0x15 67FF In case of internal Flash: 0x00 0000, …, 0x02 FFFF In case of external QSPI-Flash: 0x00 0000 + OFFSET, ・・・, 0x0F FFFF + OFFSET * The “OFFSET” is 0x04 0000, the start address of the memory mapped access area for external Flash memory (refer to “Figure 4.1.1 Memory Map” in “S1C31 D50/D51 Technical Manual”). memSize Size of the memory area to be checked. initValue Initial value used for calculation of checksum or CRC16. Set the value of 16 bit width. 0x0000, …, 0xFFFF
Return Value	nothing
Explanation	This function sets parameters for MEMCHECK. Call this function before calling seMemCheckRunCommand().

seMemCheckRunCommand

Syntax	void seSoundPlayRunCommand unsigned short command)
Arguments	command Command to control MEMCHECK. Set one of the following command definitions. 0x02 (mc_command_ram_rw): RAM Check by Read/Write Start 0x03 (mc_command_ram_marach_c): March-C RAM Check Start 0x04 (mc_command_checksum): Checksum Start 0x05 (mc_command_crc): CRC Start 0xFF (mc_command_stop): Memory Check Stop
Return Value	nothing
Explanation	This function runs the command. Call this function after calling seMemCheckSetParameters().

seMemCheckGetState

Syntax	unsigned short seMemCheckGetState(void)
Arguments	nothing
Return Value	Current state of MEMCHECK. 0x0000 (mc_state_init): On initializing 0x0001 (mc_state_idle): Idle 0x0002 (mc_state_ram_rw): Running RAM Check by Read/Write 0x0003 (mc_state_ram_march_c): Running RAM Check by March-C 0x0004 (mc_state_checksum): Running Checksum 0x0005 (mc_state_crc): Running CRC
Explanation	This function gets current state of MEMCHECK.

seMemCheckFinish

Syntax	void seMemCheckFinish()
Arguments	nothing
Return Value	nothing
Explanation	This function finishes MEMCHECK.

Appendix. B. Playlist File

Table B.1 shows the parameters that can be specified in the playlist file generated with ESPER2 (Speech data creation tool by EPSON) with sound ROM data.

Table B.1 Parameters in Playlist File

Parameter Name	Description
DIPSW_DEMO	Set the demo No. Demo No. is assigned to SW3-2/3/4 on the evaluation board. SW2-2 SW2-3 SW2-4 0: OFF OFF OFF 1: OFF OFF ON … 7: ON ON ON
DIPSW_LANG	Set the language type Language type is assigned to SW2-2/3/4 on the evaluation board. SW2-2 SW2-3 SW2-4 0: OFF OFF OFF 1: OFF OFF ON … 7: ON ON ON
ROM_ADRS	Set the start address of the sound ROM data stored in external QSPI flash memory.
ROM_SIZE	Set the size of the sound ROM data stored in external QSPI flash memory.
KEYCODE	Set the KEYCODE registered into ESPER2.
PLAY_LIST_CH0	Set the playlist of the sentence to be played on channel 0.
PLAY_LIST_CH1	Set the playlist of the sentence to be played on channel 1.
VOLUME_CH0	Set the initial value of the volume for channel 0.
VOLUME_CH1	Set the initial value of the volume for channel 1.
REPEAT_CH0	Set the number of repeat for channel 0.
REPEAT_CH1	Specify the number of repeat for channel 1.
SPEED	Set the initial value of speed for channel 0.
STOP_TYPE_CH0	Set the type of STOP command to be executed on channel 0. 0: stop immediately, 1:stop after current phrase
STOP_TYPE_CH1	Set the type of STOP command to be executed on channel 1. 0: stop immediately, 1:stop after current phrase
PAUSE_TYPE_CH0	Set the type of PAUSE command to be executed on channel 0. 0: pause immediately, 1:pause after current phrase
PAUSE_TYPE_CH1	Set the type of PAUSE command to be executed on channel 1. 0: pause immediately, 1:pause after current phrase
MUTE_TYPE_CH0	Set the type of MUTE command to be executed on channel 0. 0: mute immediately, 1:mute after current phrase
MUTE_TYPE_CH1	Set the type of MUTE command to be executed on channel 1. 0: mute immediately, 1:mute after current phrase
AUTO_PLAY_CH0	Set the auto-play flag for channel 0. 0: auto-play off, 1:auto-play on
AUTO_PLAY_CH1	Set the autoplay flag for channel 1. 0: autoplay OFF, 1:autoplay ON

Appendix. C. Code Sizes of Example Projects

Table B.1 shows a list of code sizes when each example project included in this sample software is built using IAR EWARM or MDK-ARM.

Table C.1 Code Sizes of Example Projects

Example Project Name	IAR EWARM (Code size is 16 KB or less.)	MDK-ARM (Code size is 32 KB or less.)
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	✔	✔

Note: Depending on the version of the IDE or the build configuration of the IDE, the code size may exceed the size in Table B.1.

Appendix. D. Self-Testing Sample Software Specification

s1c31TestRam16

Syntax	Short s1c31TestRam16 (unsigned short *chkAddr, unsigned short chkNum)
Arguments	*chkAddr Start address of RAM to be tested. chkNum RAM data size to be tested.
Return Value	Result of RAM R/W 0x0000 (E_OK): No difference from the written value 0x0001 (E_MEMORY): There is a difference with the written value
Explanation	This function writes 0xaa55 (0x55aa) to the memory for the number of chkNum from the address pointed to by chkAddr. After that, it reads and compares it with the written value, and returns E_OK if there is no difference and E_MEMORY if there is a difference.
Caution	The least significant bit of chkAddr is always treated as 0. The operation is not guaranteed when the specified memory area overlaps the stack area. S1c31TestRam16 checks the memory for chkNum x 2 bytes.

s1c31TestRam8

Syntax	Short s1c31TestRam8 (unsigned short *chkAddr, unsigned short chkNum)
Arguments	*chkAddr Start address of RAM to be tested. chkNum RAM data size to be tested.
Return Value	Result of RAM R/W 0x0000 (E_OK): No difference from the written value 0x0001 (E_MEMORY): There is a difference with the written value
Explanation	This function writes 0xa5 (0x5a) to the memory for the number of chkNum from the address pointed to by chkAddr. After that, it reads and compares it with the written value, and returns E_OK if there is no difference and E_MEMORY if there is a difference.
Caution	The least significant bit of chkAddr is always treated as 0. The operation is not guaranteed when the specified memory area overlaps the stack area. S1c31TestRam8 checks the memory for chkNum x 1 bytes.

s1c31TestRegister

Syntax	Short s1c31TestRegister (void)
Arguments	void
Return Value	Result of Register Read/Write 0x0000 (E_OK): No difference from the written value 0x0002 (E_REGISTER): There is a difference with the written value
Explanation	This function writes 0x555555 (0xaaaaaa) to registers in Arm core. After that, it reads and compares it with the written value, and returns E_OK if there is no difference and E_REGISTER if there is a difference.

s1c31TestPsr

Syntax	Short s1c31TestPsr (void)
Arguments	void
Return Value	Result of Register Read/Write 0x0000 (E_OK): No difference from the written value 0x0002 (E_REGISTER): There is a difference with the written value
Explanation	This function writes 0x555555 (0xaaaaaa) to status registers in Arm core. After that, it reads and compares it with the written value, and returns E_OK if there is no difference and E_REGISTER if there is a difference.

s1c31TestRamMarchc

Syntax	Short s1c31TestRamMarchc (unsigned short *chkAddr, unsigned short chkNum)
Arguments	*chkAddr Start address of RAM to be tested. chkNum RAM data size to be tested.
Return Value	Result of RAM R/W 0x0000 (E_OK): No difference from the written value 0x0001 (E_MEMORY): There is a difference with the written value
Explanation	This function runs March-C tests for the number of chkNum from the address pointed by chkAddr. And returns E_OK if there is no difference and E_MEMORY if there is a difference.
Caution	The operation is not guaranteed when the specified memory area overlaps the stack area. The memory in the test range will be rewritten to 0x00.

s1c31TestChksum

Syntax	Short s1c31TestChksum (unsigned short *chkAddr, unsigned short chkNum)
Arguments	*chkAddr Start address of RAM to be tested. chkNum RAM data size to be tested.
Return Value	Result of checksum calculation.
Explanation	This function reads the value of memory for the number of chkNum from the address pointed by chkAddr and returns the result of checksum.

s1c31TestCrc

Syntax	Short s1c31TestCrc (unsigned short *chkAddr, unsigned short chkNum)
Arguments	*chkAddr Start address of RAM to be tested. chkNum RAM data size to be tested.
Return Value	Result of CRC calculation.
Explanation	This function reads the value of memory for the number of chkNum from the address pointed by chkAddr and returns the result of CRC calculation.

s1c31TestCrcTbl

Syntax	Short s1c31TestCrcTbl (unsigned short *chkAddr, unsigned short chkNum)
Arguments	*chkAddr Start address of RAM to be tested. chkNum RAM data size to be tested.
Return Value	Result of CRC calculation.
Explanation	This function reads the value of memory for the number of chkNum from the address pointed by chkAddr. After that, refer to the table (CRC-CCITT table) to calculate the CRC and return the result.

s1c31d5xTestInterupt

Syntax	Short s1c31d5xTestInterrupt (unsigned short numInt)
Arguments	numInt Number of interrupts specified by CLG 16 bit timer.
Return Value	Number of interrupts occurred
Explanation	This function counts generated interrupts (P13 (SW7) interrupt is used) before the number of interrupts specified by numInt (CLG 16 bit timer is used) are generated.

s1c31d5xTestClk

Syntax	int s1c31d5xTestClk (unsigned long baseFreq, unsigned short range)
Arguments	baseFreq The ideal main clock used (Hz) Range Allowable error (%)
Return Value	Result of Clock Test 0x0000 (E_OK): Within tolerance 0x0003 (E_CLOCK): Outside tolerance
Explanation	This function checks whether the ideal frequency of the main clock specified by baseFreq operates within the tolerance (%) range specified by range. If the result is within the specified tolerance, it returns E_OK, and if it is out of the range, it returns E_CLOCK.

Revision History

Attachment-1

Rev. No.	Date	Page	Category	Contents
Rev.1.00	11/10/2018	All	New	New establishment.
Rev.1.10	01/17/2019	-	Delete	Deleted section 2.2.3.
		4	Modify	Modified section 2.2.1.
		8,16	Modify	Modified with UI change of the setup utility "ToolchainSetup.exe".
		12,21	Modify	Modified section 2.3.5 and 2.4.5.
		66,67	Modified	Modified Appendix.A.
Rev.2.00	02/26/2019	4,6,14	Modify	Modified section 1.1, 2.2.1, 2.3.1, 2.3.5, 2.4.1 and 2.4.5.
		56	Add	Added section 3.24.
Rev.3.00	06/30/2020	All	Modify	Modified D50 to D5x
		52-57	Modify	Updated section 3.20 (SOUNDPLAY)
		62-67	Modify	Updated chapter 4 (Demo Software)
		68-70	Add	Added chapter 5 (Self-Testing Sample Software)
		71-74	Add	Added some functions to Appendix A (HW Processor Library Specification)
		78-80	Add	Added Appendix D (Self-Testing Sample Software Specification)

EPSON

International Sales Operations

America

Epson America, Inc.

Headquarter:
3840 Kilroy Airport Way
Long Beach, California 90806-2452 USA
Phone: +1-562-290-4677

San Jose Office:
214 Devcon Drive
San Jose, CA 95112 USA
Phone: +1-800-228-3964 or +1-408-922-0200

Europe

Epson Europe Electronics GmbH
Riesstrasse 15, 80992 Munich,
Germany
Phone: +49-89-14005-0 FAX: +49-89-14005-110

Asia

Epson (China) Co., Ltd.
4th Floor, Tower 1 of China Central Place, 81 Jianguo Road, Chaoyang District, Beijing 100025 China
Phone: +86-10-8522-1199 FAX: +86-10-8522-1120

Shanghai Branch
Room 1701 & 1704, 17 Floor, Greenland Center II,
562 Dong An Road, Xu Hui District, Shanghai, China
Phone: +86-21-5330-4888 FAX: +86-21-5423-4677

Shenzhen Branch
Room 804-805, 8 Floor, Tower 2, Ali Center,No.3331
Keyuan South RD(Shenzhen bay), Nanshan District, Shenzhen 518054, China
Phone: +86-10-3299-0588 FAX: +86-10-3299-0560

Epson Taiwan Technology & Trading Ltd.
15F, No.100, Songren Rd, Sinyi Dist, Taipei City 110. Taiwan
Phone: +886-2-8786-6688

Epson Singapore Pte., Ltd.
1 HarbourFront Place,
#03-02 HarbourFront Tower One, Singapore 098633
Phone: +65-6586-5500 FAX: +65-6271-3182

Epson Korea Co.,Ltd
10F Posco Tower Yeoksam, Teheranro 134 Gangnam-gu,
Seoul, 06235, Korea
Phone: +82-2-3420-6695

Seiko Epson Corp.
Sales & Marketing Division

Device Sales & Marketing Department
29th Floor, JR Shinjuku Miraina Tower, 4-1-6 Shinjuku,
Shinjuku-ku, Tokyo 160-8801, Japan

Document Code: 413732503
First Issue November 2018
Revised February 2019
Revised June 2020
in JAPAN