EPSON
EXCEED YOUR VISION

CMOS 32-BIT SINGLE CHIP MICROCONTROLLER

S1C31D01
Peripheral Circuit
Sample Software Manual

ARM®

SEIKO EPSON CORPORATION

Rev.3.00

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 caused 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 S1C31D01 Microcontroller and shows the expected output when running the example software.

The example software is included in the S1C31D01 Sample Software and Peripheral Drivers software package. It is intended to demonstrate how to use the various peripheral circuits in the S1C31D01 microcontroller.

For detailed information on the S1C31D01 Microcontroller, refer to the “S1C31D01 Technical Manual”.

The S1C31D01 Peripheral Circuit Sample Software Package includes:

- Peripheral Library modules
- Example Software
- Product-specific files such as the S1C31D01 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 S1C31D01 peripheral. Each Example demonstrates features of the selected peripheral and exercises various peripheral modes.

The S1C31D01 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 hardware 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 “S1C31D01 Technical Manual” and the “S5U1C31D01T1 manual” available from Seiko Epson's website.

*2 This manual covers the sample software package of ver.3.00 or later.

1.1 Operating Environment

Before running the S1C31D01 sample software, prepare the following components:

- Evaluation Board
  - S5U1C31D01T1 evaluation board equipped with S1C31D01
  - S5U1C31001L1200 adapter board (Bridge Board Ver.2)
- 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

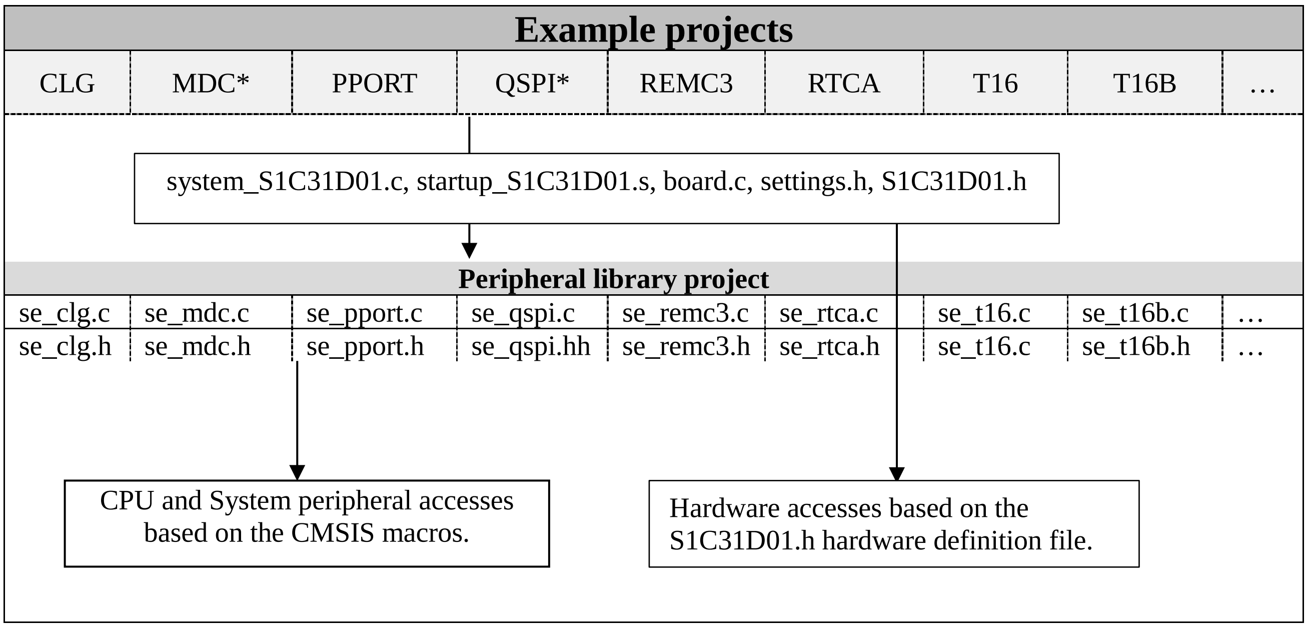
*1: I-jet is available only with IAR EWARM. J-Link is available for both IAR EWARM and MDK-ARM.

2 Sample Software Operations

2.1 Sample Software Components

The S1C31D01 Sample Software demonstrates how to call the Peripheral Library modules which interface hardware components on the microcontroller. 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 S1C31D01 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. The application layer can access peripheral hardware through the Peripheral library routines or directly using the structures defined in the “S1C31D01.h” hardware definition file.



Example projects
CLG
MDC*
PPORT
QSPI*
REMC3
RTCA
T16
T16B
…
system_S1C31D01.c, startup_S1C31D01.s, board.c, settings.h, S1C31D01.h
Peripheral library project
se_clg.c
se_mdc.c
se_pport.c
se_qspi.c
se_remc3.c
se_rtca.c
se_t16.c
se_t16b.c
…
se_clg.h
se_mdc.h
se_pport.h
se_qspi.hh
se_remc3.h
se_rtca.h
se_t16.c
se_t16b.h
…
CPU and System peripheral accesses based on the CMSIS macros.
Hardware accesses based on the S1C31D01.h hardware definition file.

Figure 2.1.1 S1C31D01 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 S1C31D01 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 higher accuracy clock in the BoardInit function.	The CPU uses the default clock.
IOSC_AUTOTRIMMING_ON	IOSC trimming is performed in the BoardInit() function to achieve higher CPU clock accuracy.	IOSC trimming is not performed. This results in a shorter boot time.
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.

- S5U1C31D01T1 evaluation board
- S5U1C31001L1200 adapter board (Bridge Board Ver.2)
- Debug probes (IAR Systems I-jet or SEGGER J-Link)

For details of the hardware connection, refer to the "S5U1C31D01T1 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 S5U1C31D01T1 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 S5U1C31D01T1 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.

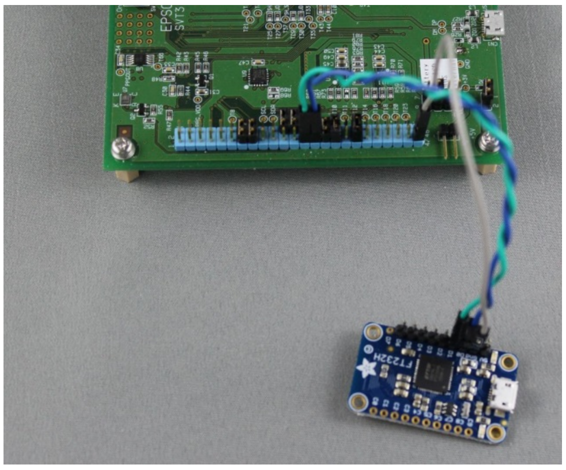


Figure 2.2.2.1 USB Adapter for UART Connection Example to S5U1C31D01T1 Evaluation Board

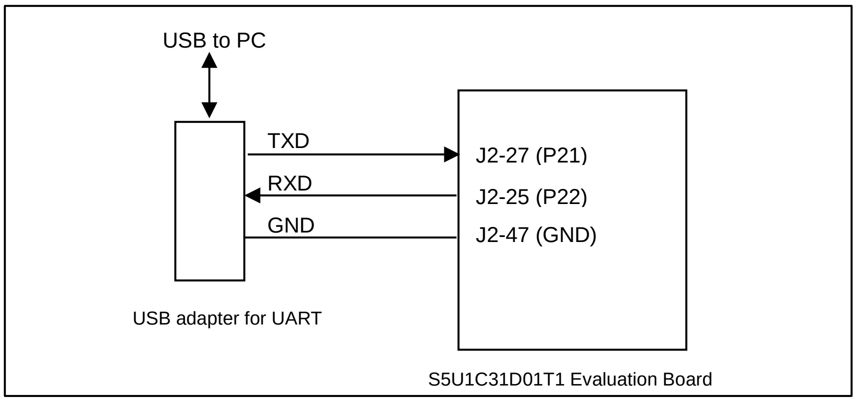


Figure 2.2.2.2 USB Adapter for UART Wiring to S5U1C31D01T1 Evaluation Board

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 S5U1C31D01T1 evaluation board, refer to the “S5U1C31D01T1 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 S1C31D01 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.

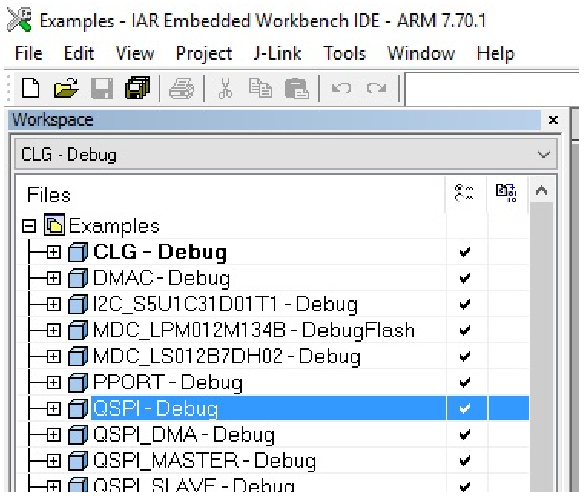
For the version of IAR EWARM used to evaluate this sample software and more information on the setup utility, refer to the “README_IAR.txt” found in this sample software package.

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 and some also need the seGraphicsLibrary.

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\S5U1C31D01T1\IAR” folder, and open the “Examples.eww” file.

Optionally, each sample software project subfolder has a “board\S5U1C31D01T1\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\S5U1C31D01T1\IAR” folder, and open the workspace file (.eww).



Examples - IAR Embedded Workbench IDE - ARM 7.70.1
File Edit View Project J-Link Tools Window Help
Workspace
CLG - Debug
Files

Examples
- CLG - Debug ✓
- DMAC - Debug ✓
- I2C_S5U1C31D01T1 - Debug ✓
- MDC_LPM012M134B - DebugFlash ✓
- MDC_LS012B7DH02 - Debug ✓
- PPORT - Debug ✓
- QSPI - Debug ✓
- QSPI_DMA - Debug ✓
- QSPI_MASTER - Debug ✓
- QSPI_SLAVE - Debug ✓

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".

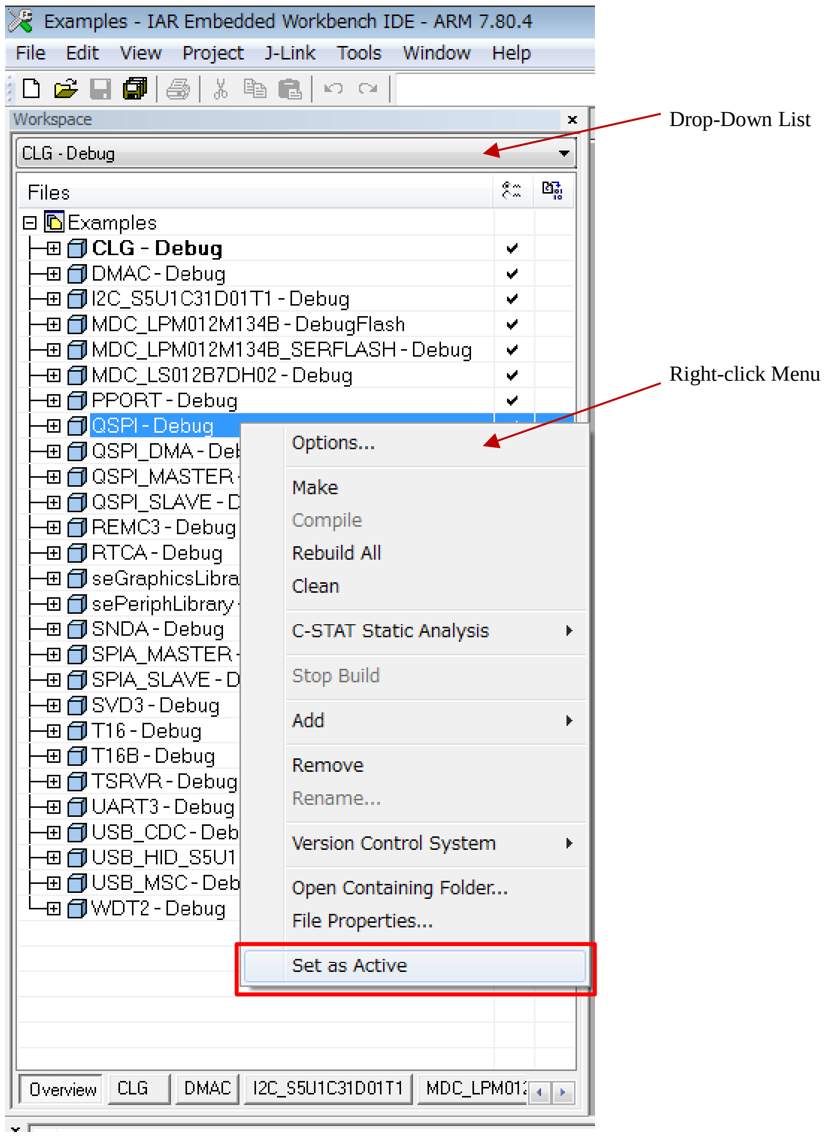


Figure 2.3.3.1 Active Project Setting

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".

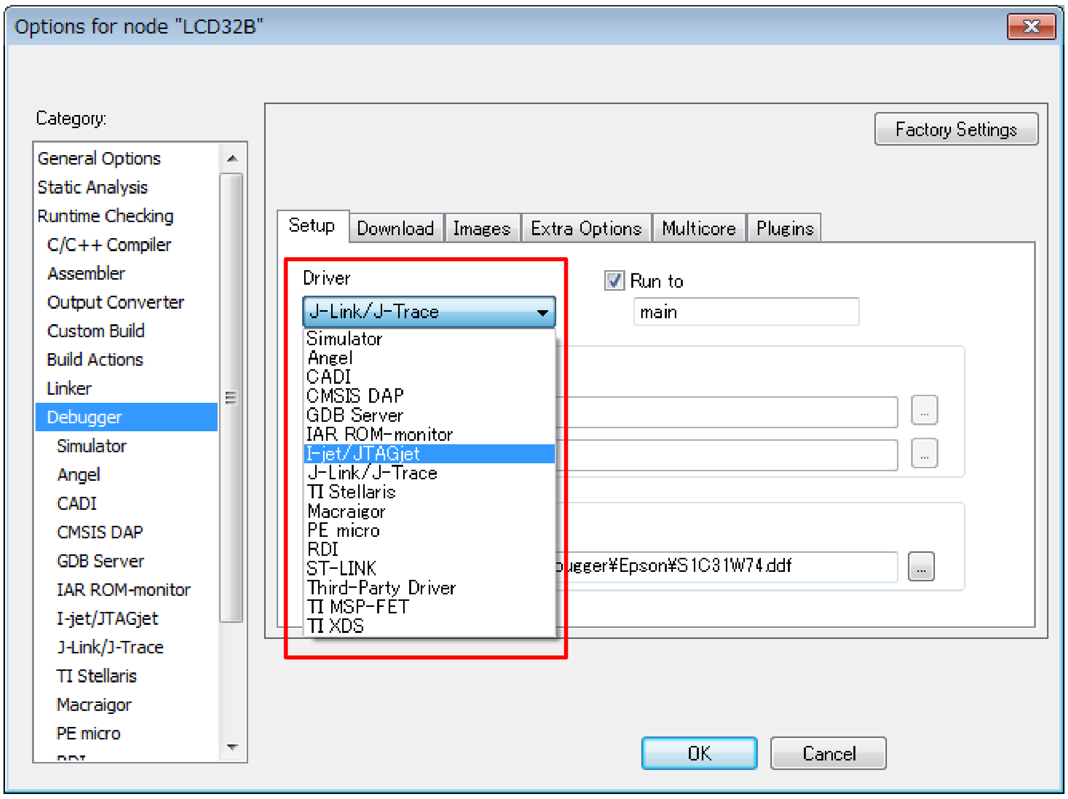


Figure 2.3.4.1 Debug Probe Setting

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.

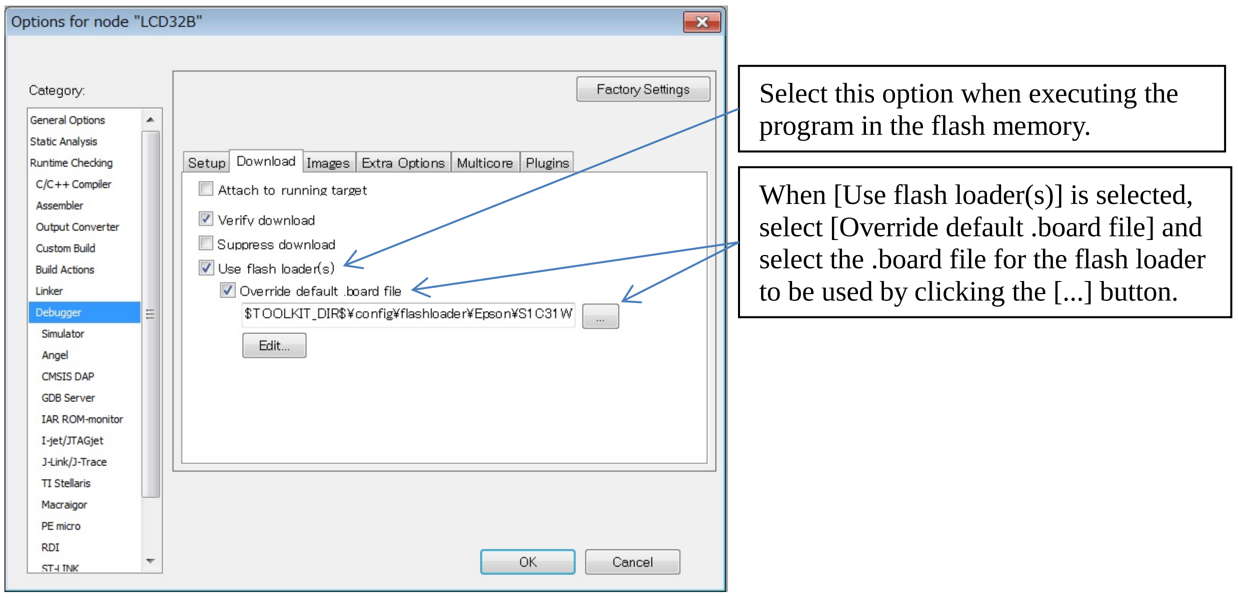
(2) Select the [Debugger] in the [Category] list on the [Options for node “{project}”] dialog (Figure 2.3.5.1).

(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 “S1C31D01_int.board” as a board file (Figure 2.3.5.1).



Attach to running target [ ]
Verify download [✓]
Suppress download [ ]
Use flash loader(s) [✓]
Override default .board file [✓]
$TOOLKIT_DIR$¥config¥flashloader¥Epson¥S1C31W

Select this option when executing the program in the flash memory.

When [Use flash loader(s)] is selected, select [Override default .board file] and select the .board file for the flash loader to be used by clicking the [...] button.

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 projects “sePeriphLibrary” and “seGraphicsLibrary” are not built. Build these library projects 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 "seGraphicsLibrary-Debug" and “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

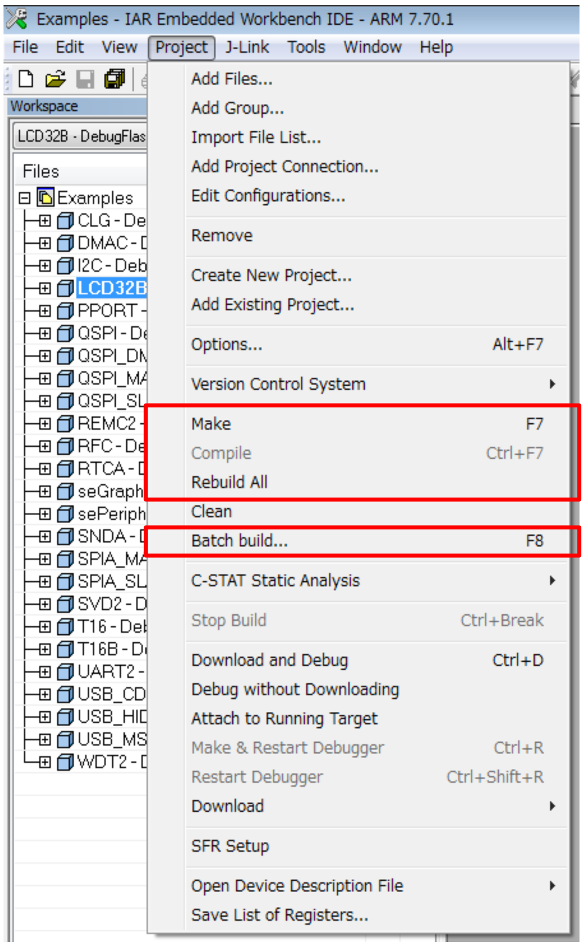


Figure 2.3.6.1 Build Commands

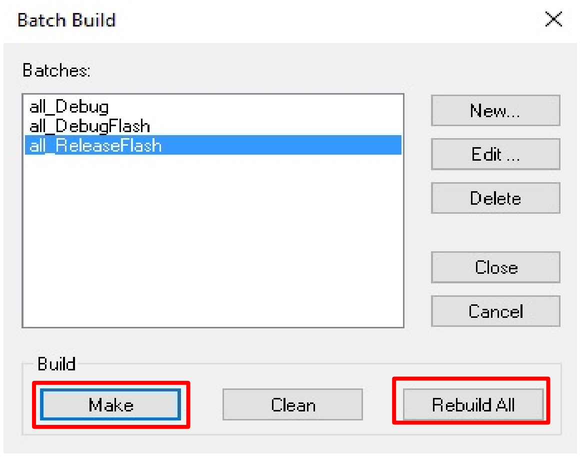


Figure 2.3.6.2 Batch Build

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.

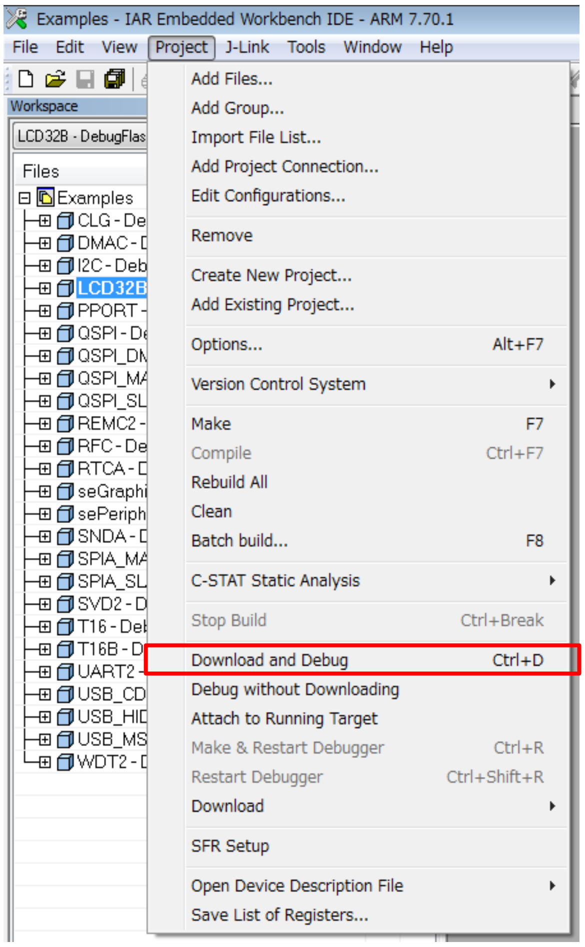


Figure 2.3.7.1 Download and Debug

2.4 KEIL MDK-ARM (uVision) Sample Software Procedures

2.4.1 Software Setup

Before executing the sample software using the MDK-ARM (uVision), 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 S1C31D01 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 uVision identified by the checkboxes. [Note: The setup utility will copy some flashing binaries into your uVision 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 uVision

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

For the version of IAR EWARM 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 and some also need the seGraphicsLibrary.

To open a workspace which contains all the sample software projects, select the [Project] > [Open Project…] in the uVision menu, navigate to “Examples\WORKSPACE\S5U1C31D01T1\ARM” folder, and open the “Examples.eww” file.

Optionally, each sample software project subfolder has a “board\S5U1C31D01T1\ARM” subfolder which contains a uVision project file (.uvprojx) and a uVision multi-project workspace file (.uvmpw). To open the workspace for an individual sample software project, select the [Project] > [Open Project…] in uVision menu, navigate to the project’s “board\S5U1C31D01T1\ARM” folder, and open the workspace file (.uvmpw).

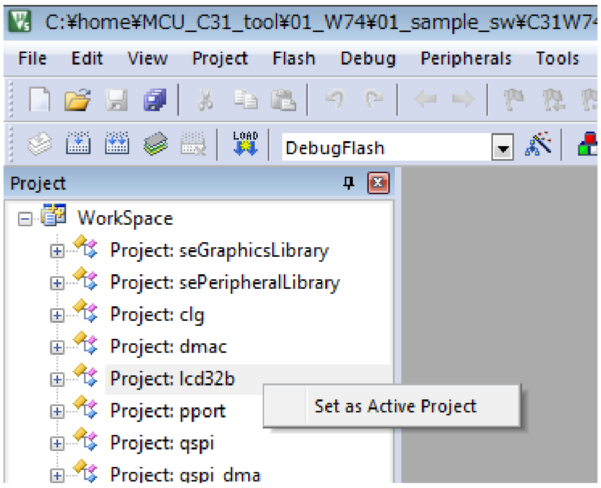


Figure 2.4.2.1 Example of Multi-Project Workspace

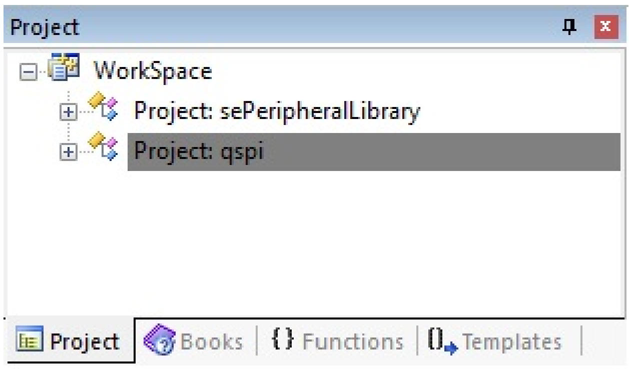


Figure 2.4.2.2 Example of Individual Project Workspace

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 uVision 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 uVision (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

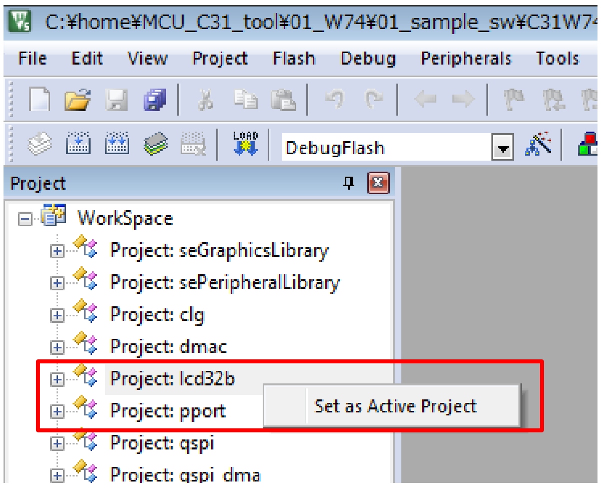

C:¥home¥MCU_C31_tool¥01_W74¥01_sample_sw¥C31W74
File Edit View Project Flash Debug Peripherals Tools
DebugFlash
Project
WorkSpace
Project: seGraphicsLibrary
Project: sePeripheralLibrary
Project: clg
Project: dmac
Project: lcd32b
Project: pport
Set as Active Project
Project: qspi
Project: qspi_dma

Figure 2.4.3.1 Active Project Setting

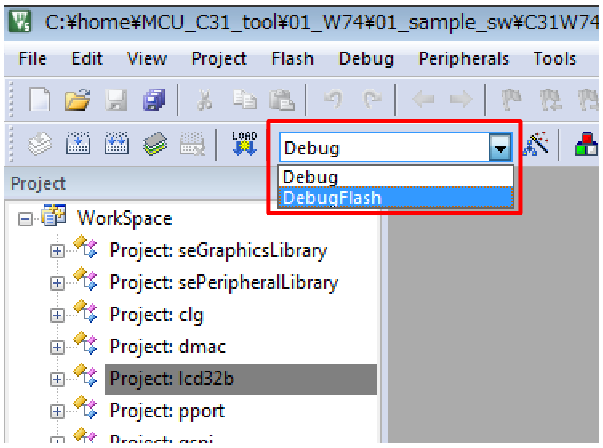

C:¥home¥MCU_C31_tool¥01_W74¥01_sample_sw¥C31W74
File Edit View Project Flash Debug Peripherals Tools
Debug
Debug
DebugFlash
Project
WorkSpace
Project: seGraphicsLibrary
Project: sePeripheralLibrary
Project: clg
Project: dmac
Project: lcd32b
Project: pport

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 uVision 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).

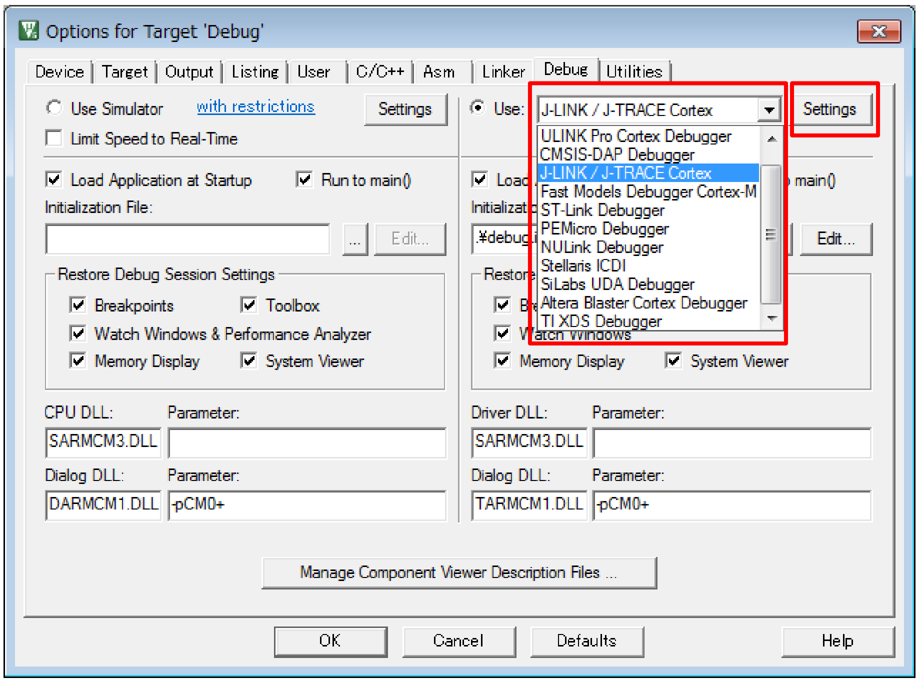
(4) Click the [Settings] button at the right side of the above drop-down list (Figure 2.4.4.1).

(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.

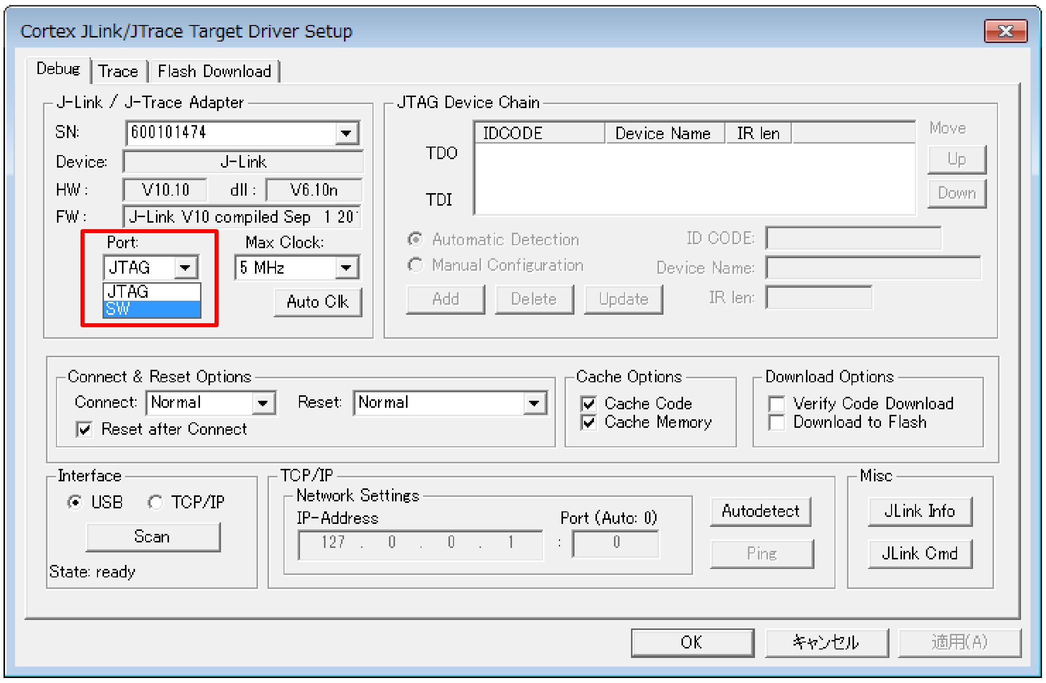
Notes:

- This setting needs to be done with J-Link connected to the PC.
- If "setup-debugger-jlink.bat" described in "2.4.1 Software Setup" has already been executed, the debug probe (J - Link) setting is applied to all projects in the sample software. However, if the debug probe does not recognize normally, follow the above procedure to check the settings.



CPU DLL: SARMCM3.DLL Parameter:
Dialog DLL: DARMCM1.DLL Parameter: -pCM0+
Driver DLL: SARMCM3.DLL Parameter:
Dialog DLL: TARMCM1.DLL Parameter: -pCM0+

Figure 2.4.4.1 Debug Probe Selecting


Cortex JLink/JTrace Target Driver Setup

Debug | Trace | Flash Download

J-Link / J-Trace Adapter
SN: 600101474
Device: J-Link
HW : V10.10 dll : V6.10n
FW : J-Link V10 compiled Sep 1 20
Port: JTAG (JTAG / SW)
Max Clock: 5 MHz
Auto Clk

JTAG Device Chain
TDO
TDI
IDCODE | Device Name | IR len
Move: Up, Down
Automatic Detection
Manual Configuration
ID CODE:
Device Name:
IR len:
Add | Delete | Update

Connect & Reset Options
Connect: Normal
Reset: Normal
☑ Reset after Connect

Cache Options
☑ Cache Code
☑ Cache Memory

Download Options
☐ Verify Code Download
☐ Download to Flash

Interface
● USB ○ TCP/IP
Scan
State: ready

TCP/IP
Network Settings
IP-Address: 127 . 0 . 0 . 1
Port (Auto: 0): 0
Autodetect
Ping

Misc
JLink Info
JLink Cmd

OK | キャンセル | 適用(A)

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 uVision 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 uVision 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.

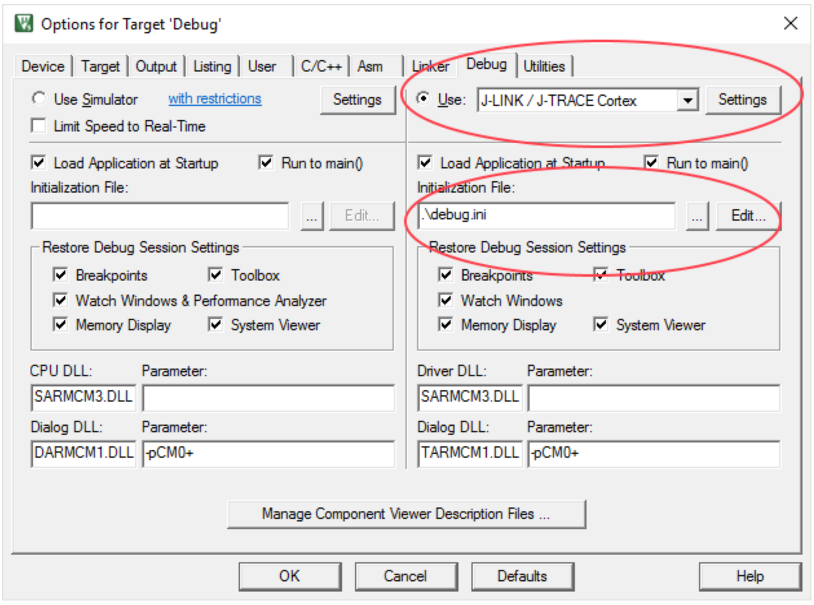
(6) Delete all the flash loaders displayed in the [Programming Algorithm] list by clicking the [Remove] button.

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

(8) Select a flash loader “S1C31D01int 256kB” 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.


Options for Target 'Debug'

Device | Target | Output | Listing | User | C/C++ | Asm | Linker | Debug | Utilities

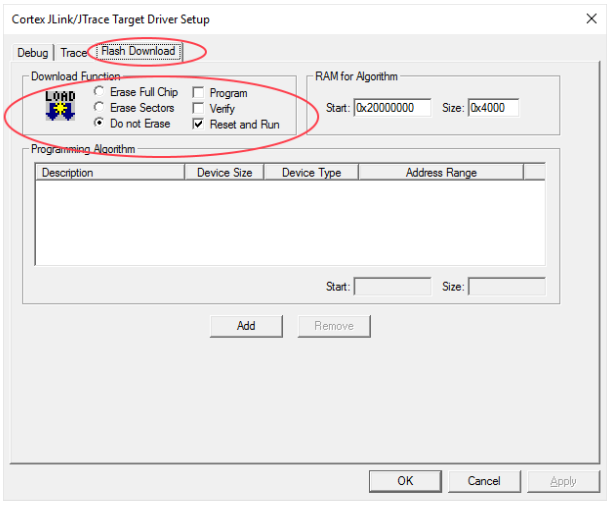
[ ] Use Simulator with restrictions Settings
[ ] Limit Speed to Real-Time
[✓] Load Application at Startup [✓] Run to main()
Initialization File:
Restore Debug Session Settings
[✓] Breakpoints [✓] Toolbox
[✓] Watch Windows & Performance Analyzer
[✓] Memory Display [✓] System Viewer
CPU DLL: SARMCM3.DLL Parameter:
Dialog DLL: DARMCM1.DLL Parameter: -pCM0+

[✓ Use:] J-LINK / J-TRACE Cortex Settings
[✓] Load Application at Startup [✓] Run to main()
Initialization File: .\debug.ini Edit...
Restore Debug Session Settings
[✓] Breakpoints [✓] Toolbox
[✓] Watch Windows
[✓] Memory Display [✓] System Viewer
Driver DLL: SARMCM3.DLL Parameter:
Dialog DLL: TARMCM1.DLL Parameter: -pCM0+

Manage Component Viewer Description Files ...

OK Cancel Defaults Help

Figure 2.4.5.1 “Initialize File” Option


Cortex JLink/JTrace Target Driver Setup

Debug | Trace | Flash Download

Download Function
Erase Full Chip [ ] Program
Erase Sectors [ ] Verify
[✓ Do not Erase] [✓] Reset and Run

RAM for Algorithm
Start: 0x20000000 Size: 0x4000

Programming Algorithm

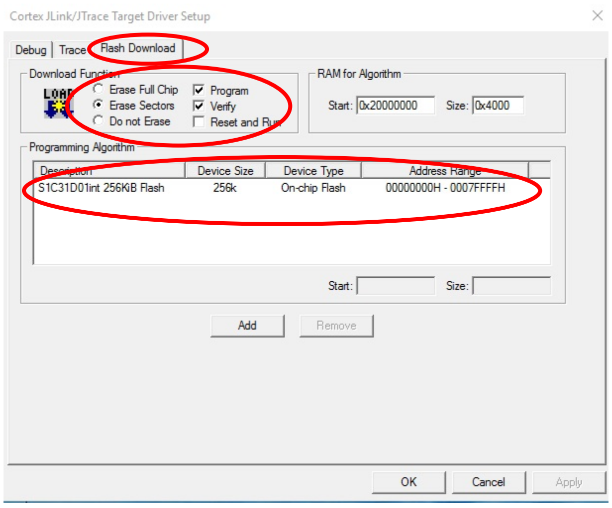
Description	Device Size	Device Type	Address Range

Start: Size:

Add Remove

OK Cancel Apply

Figure 2.4.5.2 Flash Loader Unsetting


Cortex JLink/JTrace Target Driver Setup

Debug | Trace | Flash Download

Download Function
Erase Full Chip [✓] Program
[✓ Erase Sectors] [✓] Verify
Do not Erase [ ] Reset and Run

RAM for Algorithm
Start: 0x20000000 Size: 0x4000

Programming Algorithm

Description	Device Size	Device Type	Address Range
S1C31D01int 256KiB Flash	256k	On-chip Flash	00000000H - 0007FFFFH

Start: Size:

Add Remove

OK Cancel Apply

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 uVision menu (Figure 2.4.6.1).

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

The Batch Build option, found under [Project] > [Batch Build] in the uVision 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).

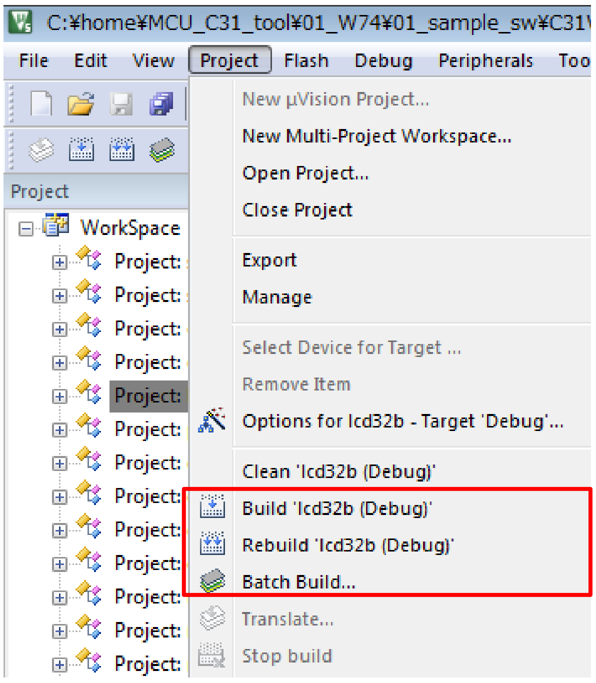


Figure 2.4.6.1 Build Commands

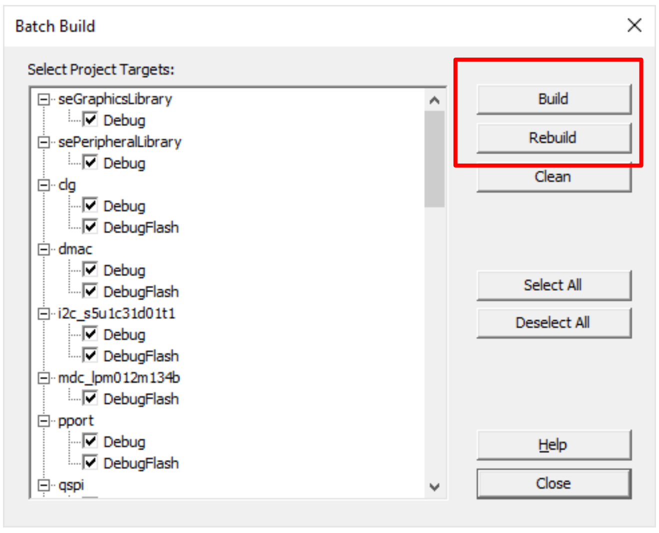


Figure 2.4.6.2 Batch Build

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 uVision 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 uVision menu (Figure 2.4.7.2).

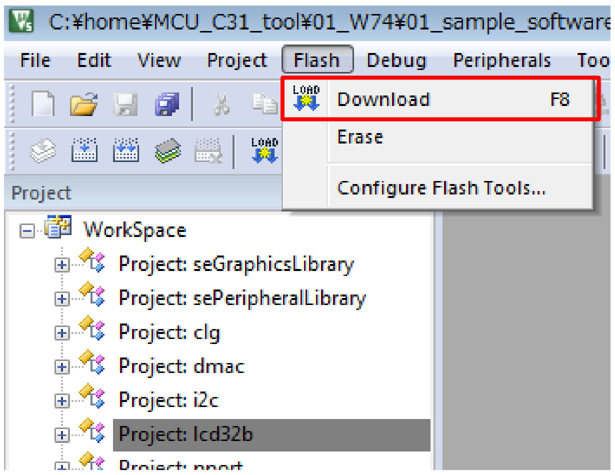

C:¥home¥MCU_C31_tool¥01_W74¥01_sample_software
File Edit View Project Flash Debug Peripherals Too
Download F8
Erase
Configure Flash Tools...
Project
WorkSpace
Project: seGraphicsLibrary
Project: sePeripheralLibrary
Project: clg
Project: dmac
Project: i2c
Project: lcd32b
Project: pport

Figure 2.4.7.1 Download

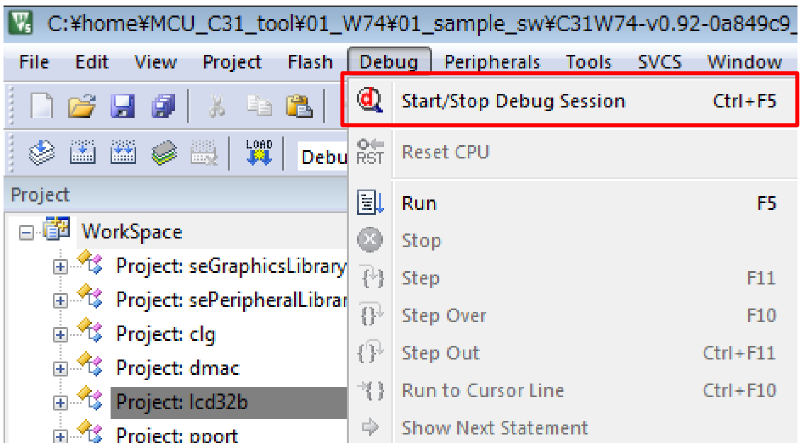

C:¥home¥MCU_C31_tool¥01_W74¥01_sample_sw¥C31W74-v0.92-0a849c9_
File Edit View Project Flash Debug Peripherals Tools SVCS Window
Start/Stop Debug Session Ctrl+F5
Reset CPU
Run F5
Stop
Step F11
Step Over F10
Step Out Ctrl+F11
Run to Cursor Line Ctrl+F10
Show Next Statement
Project
WorkSpace
Project: seGraphicsLibrary
Project: sePeripheralLibrar
Project: clg
Project: dmac
Project: lcd32b
Project: pport

Figure 2.4.7.2 Debug

2.5 MDC Tools

2.5.1 Image Scaling Calculation imgcpy_calcscaling.exe

The MDC image copying hardware has a round-off error limitation in which certain combinations of source image size, destination image size, and scaling values will produce a result in which the full width and height of the source image pixels are not copied.

The “imgcpy_calcscaling.exe” program is intended to provide a pre-calculation tool to generate the properly compensated size, centers of transformation, and scaling values to program the MDC with given the original source and destination dimensions. This utility tool takes into account the method of scaling calculations done in the hardware, and provides values that should produce the desired results.

Example of use:

If the source image is 208x208 and destination image is also 208x208 (1:1 ratio), the calculated left/right/top/bottom scaling value is 256. However, if these scaling values are used, the result is that only 207x207 pixels of the source image are copied to the destination image, and the right and bottom edge of the source image are not copied. In order to compensate for this hardware limitation and copy 208x208 pixels to the destination image, please use the compensated values calculated by this utility tool.

The usage of this utility tool is as follows.

1. Run the imgcpy_calcscaling.exe. Then, the console screen will open.
2. Input the original source image size and the original destination image size on the console screen(see Table 2.5.1.1). Then, the console screen will be output the compensated values(see Table 2.5.1.2).

Table 2.5.1.1 Input of the original image size

```
Enter source image width:  208
Enter source image height:  208                Original source image size
Enter destination image width:  208
Enter destination image height:  208           Original destination image size
```

Table 2.5.1.2 Output of the compensated values

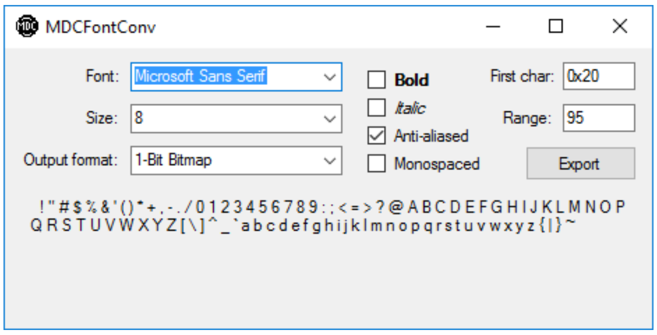
```
Original source width =  208
Original source height = 208

Source width =          210                    Compensated size for the source image
Source height =         210
Source Center X =        104                   Compensated center coordinate of transformation for
Source Center Y =        104                   the source image

Destination width =     208                    Compensated size for the destination image
Destination height =    208
Destination Center X =   104                   Compensated center coordinate of transformation for
Destination Center Y =   104                   the destination image

XLSCALE =                257
XRSCALE =                252                   Compensated scaling values
YTSCALE =                257
YBSCALE =                252
```

By using the compensated values as shown in Table 2.5.1.2, the source image are correctly copied to destination image. For example, a source image size of 210x210, left/top scaling value of 257, and right/bottom scaling value of 252 results in all pixels of the 210x210 source image copied to the 208x208 destination image.

Note: This utility tool has a limitation. Not all input and output dimension pairs are possible to correctly calculate and the utility will warn the user of that if it is unable to determine a good result. If the utility will gives warning such as "Cannot get a good fit", please try adjusting the input and output parameters.

2.5.2 Font Conversion MDCFontConv.exe

MDCFontConv.exe is a tool for generating font bitmaps header (.h) or binary files (.mdcfont) from fonts which are in the user’s Windows system.



MDCFontConv

Font: Microsoft Sans Serif
Size: 8
Output format: 1-Bit Bitmap

[ ] **Bold**
[ ] *Italic*
[✓] Anti-aliased
[ ] Monospaced

First char: 0x20
Range: 95

```
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOP
QRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
```

Tool allows user to:
- Select range of characters to be included for the font character set
- Select size of the font
- Select between 1 or 2 bits per pixel format
- Have font bold, Italic, or anti-aliased
- Make any font to be proportional or non-proportional (monospaced)

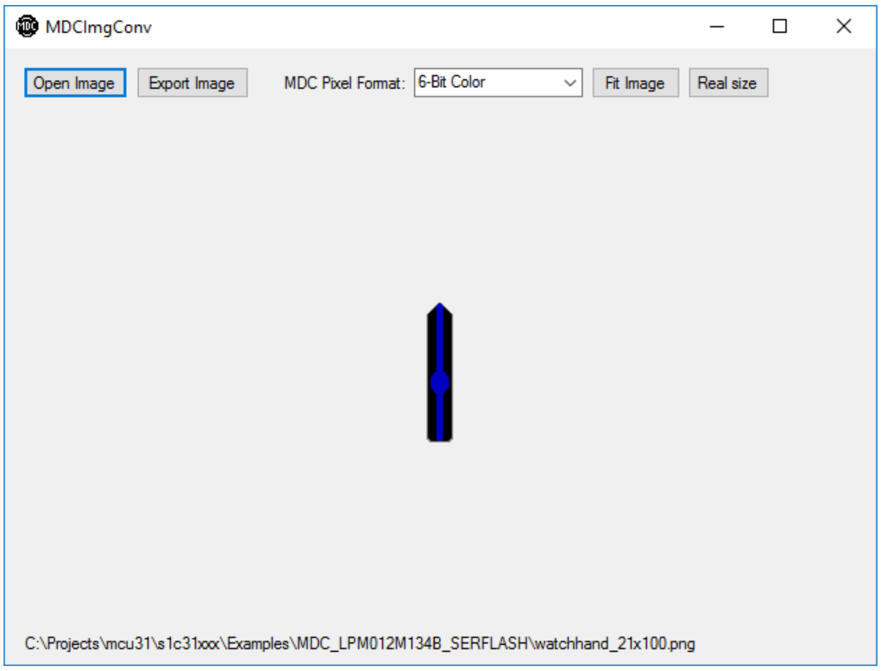
The output file (.h and .mdcfont) contains a seMDC_GFX_FontStruct structure which has the following fields:
- bitmapfmt – specifies the bitmap format: 0=1-bit, 1=2-bit
- height– specifies the height of each and every character bitmap (in pixels)
- numchars – number of characters in the font set
- asciioffset – ASCII offset of first character in the font set
- *charstbl – pointer to a table of (width, offsetloc) pairs for the characters in the font set.
  “width” is the width of the character bitmap and “offsetloc” is the offset location in the “pix_data” array of the start of the bitmap for the character
- *pxdata – pointer to byte array of the pixel data for the character bitmaps of the font set

The C header output file (.h) can be used in C code to include the font set in ROM data.

The binary format output file (.mdcfont) can be used by MDCSerFlashImg.exe tool for downloading to N25Q128 Serial NOR Flash memory chip on the S5U1C31D01T1 evaluation board.

2.5.3 Image Conversion MDCImgConv.exe

MDCImgConv.exe is a tool for converting any common type of graphic image (BMP, PNG, JPG, ICO, TIF, GIF) to any pixel formats supported by the MDC. The tool can generate header files (.h), binary files (.mdcimg) or HEX files (.hex).



MDC Pixel Format: 6-Bit Color

C:\Projects\mcu31\s1c31xxx\Examples\MDC_LPM012M134B_SERFLASH\watchhand_21x100.png

The tool allows user to:

- Select the MDC pixel format to output
- Preview look of the image according to selected pixel format
- Have image scaled to size of the tool window or actual size without affecting the image

The output file (.h, .hex, and .mdcimg) contains a seMDC_ImgStruct structure which has the following fields:

- width – width of the image (in pixels)
- height – height of the image (in pixels)
- stride – stride of the image (in pixels, same value as width)
- imgtype – specifies the MDC image format
- pxdata – pointer to byte array of the image pixels

The C header output file (.h) can be used in C code to include the image in ROM data.

The binary format output file (.mdcimg) can be used by MDCSerFlashImg.exe tool for downloading to N25Q128 NOR Flash memory chip on the S5U1C31D01T1 evaluation board.

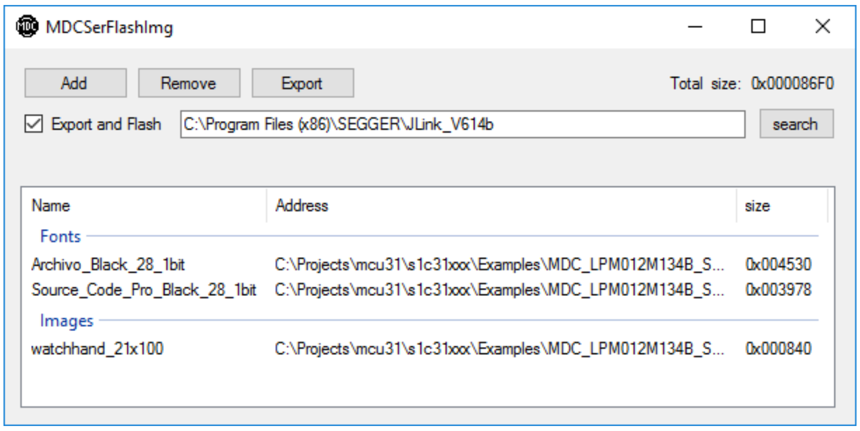
The hex format output file (.hex) can be used by an external serial flash programming tool to preprogram the N25Q128 Serial NOR Flash memory chip.

2.5.4 Serial Flash Programming MDCSerFlashImg.exe

MDCSerFlashImg.exe is a tool which helps to create and download a binary image for the N25Q128 Serial NOR Flash memory chip on the S5U1C31D01T1 evaluation board. The tool accepts binary files generated by MDCImgConv.exe and MDCFontConv.exe. The output of the tool consists of two files: a header file with addresses of all included items (to be used by the application) and a single binary image which is a collection of all the image and font binaries (.mdcfont and .mdcimg files) specified to be included in the single binary image.

The tool allows user to:

- Add/Remove font (.mdcfont) and image (.mdcimg) files to include
- Export (generate) a C header file (.h) and a single binary image (.bin)
- Optionally execute the N25Q128 flash loader tool to program the .bin image to the evaluation board



Total size: 0x000086F0

☑ Export and Flash: C:\Program Files (x86)\SEGGER\JLink_V614b

Name	Address	size
Fonts		
Archivo_Black_28_1bit	C:\Projects\mcu31\s1c31xxx\Examples\MDC_LPM012M134B_S...	0x004530
Source_Code_Pro_Black_28_1bit	C:\Projects\mcu31\s1c31xxx\Examples\MDC_LPM012M134B_S...	0x003978
Images		
watchhand_21x100	C:\Projects\mcu31\s1c31xxx\Examples\MDC_LPM012M134B_S...	0x000840

Enabling the checkbox “Export and Flash” directs the tool to download the binary image into N25Q128 Serial NOR Flash memory using the SEGGER J-Link Commander software. Refer to Section 0 for instructions on installing the SEGGER Flash programming tools for the S5U1C31D01T1 evaluation board.

The output C header file (.h) contains seMDC_GFX_SerFlashFontStruct and seMDC_SerFlashImgStruct structures for all the fonts and images included in the output binary image.

The seMDC_GFX_SerFlashFontStruct has the following fields:

- rmadrh – Value to write to the RMADRH register of the QSPI module for accessing external serial flash in Memory-Mapped Access (MMA) mode
- *Font – pointer to the seMDC_GFX_FontStruct font structure in the external serial flash image

The seMDC_SerFlashImgStruct has the following fields:

- rmadrh – Value to write to the RMADRH register of the QSPI module for accessing external serial flash in Memory-Mapped Access (MMA) mode
- *img – pointer to the seMDC_ImgStruct image structure in the external serial flash image

The tool will not accept files larger than 1 MBytes and total image size cannot be larger than 16 MBytes. Before downloading the binary image into the flash memory, connect the J-Link debugger to PC and the evaluation board.

2.6 Flash Programming Tools

The SEGGER J-Link family of debug probes can be used to program the internal Flash memory of the S1C31D01 MCU or the N25Q128 Serial NOR Flash memory on the S5U1C31D01T1 evaluation board without running an IDE tool such as IAR’s EWARM or Keil’s uVision.

Download and install the J-Link Software and Documentation Pack from the following SEGGER link: https://www.segger.com/downloads/jlink.

After the SEGGER J-Link software has been installed, follow the instructions provide in the “FlashTools\S5U1C31D01T1\README.txt” file of the S1C31D01 Sample Software package to install the flash loader binaries for the S1C31D01 internal Flash and the N25Q128 Serial NOR Flash into the SEGGER installation folders and supported devices list. The “README.txt” file also provides instructions on using the J-Link Commander software to program binary files into the S1C31D01 internal Flash or N25Q128 Serial NOR Flash.

3 Details of Sample Software

Hardware setup is required for some examples. Refer to “S5U1C31D01T1 manual” for additional information.

3.1 Clock Generator (CLG)

CLG module is the clock generator that controls the clock sources and manages clock supply to the CPU and the peripheral circuits.

The example executes various 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

```
 -CPU clock- seCLG_IOSC (20000000)
CLG Initialization ok

CLG IOSC Auto-trimming ok

Halt (OSC3)     -actual- seCLG_OSC3
Wake up (OSC3)  -actual- seCLG_OSC3

Sleep (OSC1)    -actual- seCLG_OSC1
Wake up (OSC3)  -actual- seCLG_OSC3
Exit
```

3.2 DMA Controller (DMAC)

The DMAC module is used for data transfers between memory and peripheral devices.

This example shows:

- Using DMAC in a basic memory-to-memory data transfer
- Using DMAC from memory to a peripheral 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 DMACC for transfers see QSPI_DMA Example code.

Hardware Setup

To demonstrate UART DMA transfer, connect UART on the S5U1C31D01T1 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: Memory to Peripheral DMA transfer

- DMAC Channel 0 is configured to transfer data from a buffer stored in RAM memory to SNDA data register.
- SNDA is configured to play melody.
- The start of transfer is triggered by SNDA.

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 enabled.
5. Destination address incrementing is disabled.

6. The transfer is started by sound buffer empty.
7. At the end of the transfer, a Transfer Completion interrupt flag is generated.
8. Once interrupt flag is generated, the "number of transfers" is read which must be equal to 0.
9. The Transfer Complete Interrupt flag is then cleared.
10. Correctness of the transfer is verified by sound of the playing melody.

Example 3: 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 (ex2) 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 4: 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

```
 -CPU clock- seCLG_IOSC (20000000)
DMA Initializing
DMA Test: Memory
DMA Test: SNDA
DMA Test: UART
-  Type 8 characters...
12345678 – 8 characters have been sent back to UART
DMA Test: T16B

Exit
```

3.3 I2C (I2C_S5U1C31D01T1)

This example provides a description of how to use a I2C in the master mode to read and write registers of onboard sensor devices.

Hardware Setup

Connect the following pins on the S5U1C31D01T1 evaluation.
J2-connect 17 and 18 (for SCL connection)
J2-connect 19 and 20 (for SDA connection)

Operations

This example tests the I2C_0 module by performing the following read/write operations:

1. Read the WHO_AM_I register (0x0F) of the LIS2DH accelerometer.
2. Write and read 0xAA to the CTRL_REG1 register (0x20) of the LIS2DH accelerometer.
3. Write and read 0x55 to the CTRL_REG1 register (0x20) of the LIS2DH accelerometer.
4. Write and read 0x0F to the CTRL_REG2 register (0x21) of the LIS2DH accelerometer.
5. Write and read 0xF0 to the CTRL_REG2 register (0x21) of the LIS2DH accelerometer.

6. Read the WHO_AM_I register (0x0F) of the L3GD20 gyroscope.
7. Write and read 0x5A to the CTRL_REG3 register (0x22) of the L3GD20 gyroscope.
8. Write and read 0xA5 to the CTRL_REG3 register (0x22) of the L3GD20 gyroscope.
9. Write and read 0x3C to the CTRL_REG5 register (0x24) of the L3GD20 gyroscope.
10. Write and read 0xC3 to the CTRL_REG5 register (0x24) of the L3GD20 gyroscope.

11. Read the WIA1 register (0x00) of the AK09911C magnetometer.
12. Read the WIA2 register (0x01) of the AK09911C magnetometer.
13. Read the INFO1 register (0x02) of the AK09911C magnetometer.
14. Read the INFO2 register (0x03) of the AK09911C magnetometer.
15. Write and read 0x12 to the CNTL1 register (0x30) of the AK09911C magnetometer.
16. Write and read 0xED to the CNTL1 register (0x30) of the AK09911C magnetometer.

Example of Output

```
 -CPU clock- seCLG_IOSC (20000000)
Testing I2C_0 module with register reads and writes of onboard sensor devices

LIS2DH accelerometer registers read/write tests:
   Read WHO_AM_I register (0x0F):
     0x33  Pass

   Write CTRL_REG1 register (0x20):
     0xAA  Pass
   Read CTRL_REG1 register (0x20):
     0xAA  Pass
   Write CTRL_REG1 register (0x20):
       0x55  Pass
   Read CTRL_REG1 register (0x20):
     0x55  Pass

   Write CTRL_REG2 register (0x21):
     0x0F  Pass
   Read CTRL_REG2 register (0x21):
     0x0F  Pass
   Write CTRL_REG2 register (0x21):
     0xF0  Pass
   Read CTRL_REG2 register (0x21):
     0xF0  Pass

L3GD20 gyroscope registers read/write tests:
   Read WHO_AM_I register (0x0F):
     0xD4  Pass

   Write CTRL_REG3 register (0x22):
     0x5A  Pass
   Read CTRL_REG3 register (0x22):
     0x5A  Pass
   Write CTRL_REG3 register (0x22):
     0xA5  Pass
   Read CTRL_REG3 register (0x22):
     0xA5  Pass

   Write CTRL_REG5 register (0x24):
     0x3C  Pass
   Read CTRL_REG5 register (0x24):
     0x3C  Pass
   Write CTRL_REG5 register (0x24):
     0xC3  Pass
   Read CTRL_REG5 register (0x24):
     0xC3  Pass

AK09911C magnetometer registers read/write tests:
   Read WIA1 register (0x00):
     0x48  Pass

   Read WIA2 register (0x01):
     0x05  Pass

   Read INFO1 register (0x02):
     0x20  Pass

   Read INFO2 register (0x03):
     0x00  Pass

   Write CNTL1 register (0x30):
     0x12  Pass
   Read CNTL1 register (0x30):
     0x12  Pass

   Write CNTL1 register (0x30):
     0xED  Pass
   Read CNTL1 register (0x30):
     0xED  Pass

Status: OK
Exit
```

3.4 MDC Example for LPM012M134B Panel (MDC_LPM012M134B)

MDC is a Controller for Memory Displays with a Drawing and Copying engine.

This software shows examples of using the routines in "se_mdc.c" peripheral library for initializing a panel, using drawing functions, and using image/bitmap copying functions. An analog clock, digital time, and weekday are displayed on the panel. The display is updated every second by the RTC interrupt. The digital time and weekday text are animated for each display update.

Hardware Setup

Example requires an LPM012M134B panel connected to connector CN3 of the S5U1C31D01T1 evaluation board.

Operations

1. Start OSC1 and change IOSC frequency to 20MHz.
2. Initialize LPM012M134B panel.
3. Set output destination window for MDC graphics engine
4. Fill display buffer with white background.
5. Draw line "ticks" (1-minute intervals) for analog clock.
6. Initialize and start RTC.
7. Infinite "while" loop:
    - Read latest time from RTC.
    - Draw circles in center of analog clock.
    - Animation loop (3 times)
        - Render bigger "weekday" text as needed.
        - Draw line for "second" hand.
        - Render "hour" hand.
        - Render "minute" hand.
        - Render bigger digital time text as needed.
        - Update display (copy image from frame buffer to panel).

        - Erase bigger "weekday" text as needed.
        - Erase bigger digital time text as needed.
        - Render normal "weekday" text.
        - Draw line for "second" hand.
        - Render "hour" hand.
        - Render "minute" hand.
        - Render normal digital time text.
        - Update display (copy image from frame buffer to panel).

    - Go to deep sleep waiting for 1-second interrupt from RTC.

    - (wake up from sleep)
    - Erase the "second" hand.
    - Erase the "hour" hand.
    - Erase the "minute" hand.
    - Fill "weekday" area with white.
    - Erase digital time text

3.5 MDC Example for Serial Flash (MDC_LPM012M134B_SERFLASH)

MDC is a Controller for Memory Displays with a Drawing and Copying engine.

This software shows examples of using the routines in "se_mdc.c" peripheral library for initializing a panel, using drawing functions, and using image/bitmap copying functions, using micron flash on the evaluation board. An analog clock, digital time, and weekday are displayed on the panel. The display is updated every second by the RTC interrupt. The digital time and weekday text are animated for each display update.

Hardware Setup

Example requires an LPM012M134B panel connected to connector CN3 of the S5U1C31D01T1 evaluation board. The micron flash on the evaluation board is used in this example. Prior running example code, following files have to be downloaded to micron flash using MDCSerFlashImg.exe tool:

Archivo_Black_28_1bit.mdcfont

Source_Code_Pro_Black_28_1bit.mdcfont

watchhand_21x100.mdcimg

Operations

1. Start OSC1 and change IOSC frequency to 20MHz.
2. Initialize LPM012M134B panel.
3. Set output destination window for MDC graphics engine
4. Fill display buffer with white background.
5. Draw line "ticks" (1-minute intervals) for analog clock.
6. Initialize and start RTC.
7. Infinite "while" loop:
   - Read latest time from RTC.
   - Draw circles in center of analog clock.
   - Animation loop (3 times)
     - Render bigger "weekday" text as needed.
     - Draw line for "second" hand.
     - Render "hour" hand.
     - Render "minute" hand.
     - Render bigger digital time text as needed.
     - Update display (copy image from frame buffer to panel).

     - Erase bigger "weekday" text as needed.
     - Erase bigger digital time text as needed.
     - Render normal "weekday" text.
     - Draw line for "second" hand.
     - Render "hour" hand.
     - Render "minute" hand.
     - Render normal digital time text.
     - Update display (copy image from frame buffer to panel).

   - Go to deep sleep waiting for 1-second interrupt from RTC.

   - (wake up from sleep)
   - Erase the "second" hand.
   - Erase the "hour" hand.
   - Erase the "minute" hand.
   - Fill "weekday" area with white.
   - Erase digital time text.

3.6 MDC Example for LS012B7DH02 Panel (MDC_LS012B7DH02)

MDC is a Controller for Memory Displays with a Drawing and Copying engine.

This software shows examples of using the routines in "se_mdc.c" peripheral library for initializing a panel, using drawing functions, and using image/bitmap copying functions. An analog clock is displayed on the panel. The clock is updated every second by the RTC interrupt.

Hardware Setup

1. Connect an LS012B7DH02 panel to connector CN4 of the S5U1C31D01T1 evaluation board.
2. Connect the following pins on the S5U1C31D01T1 evaluation.
   JP4 - connect pin 2 and 3
   JP5 - connect pin 2 and 3
   JP6 - connect pin 1 and 2
   JP7 - connect pin 1 and 2
   JP8 - connect pin 5 and 6

Operations

1. Start OSC1 and change IOSC frequency to 20MHz.
2. Initialize LS012B7DH02 panel.
3. Set output destination window for MDC graphics engine
4. Fill display buffer with white background.
5. Draw line "ticks" (1-minute intervals) for analog clock.
6. Initialize and start RTC.
7. Infinite "while" loop:
   - Read latest time from RTC.
   - Draw circles in center of analog clock.
   - Draw rectangle border for "weekday" string.
   - Render digital time string.
   - Render "weekday" 3-character string.
   - Draw line for "second" hand.
   - Render "hour" hand.
   - Render "minute" hand.
   - Update display (copy image from frame buffer to panel).
   - Go to deep sleep waiting for 1-second interrupt from RTC.
   - (wake up from sleep)
   - Erase digital time string.
   - Erase the "second" hand.
   - Erase the "hour" hand.
   - Erase the "minute" hand.
   - Fill "weekday" area with white.Start OSC1 and change IOSC frequency to 20MHz.

3.7 I/O Ports (PPORT)

The PPORT module controls the general-purpose input/output (GPIO) pins of the S1C31D01.

This example configures pin P35 as an output connected to pin P36 configured as an input. It performs various tests to check the connectivity of P35 and P36.

Hardware Setup

For this example program, connect P35 to P36.

(J2.3) P35--------->>---------P36 (J2.1)

Operations

1. Start the OSC1 oscillation and switch the system clock from IOSC to OSC1. Then sotp the IOSC.
2. Initialize the ports.
3. Set the P35 port to output and Output a Low level signal.
4. Set the P36 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 P35 port.
6. Confirm that the P36 port is at High level after an interrupt in the P36 port.
7. Set the P36 port so that an interrupt occurs when the level changes from High to Low.
8. Output a Low level signal from the P35 port.
9. Confirm that the P36 port is at Low level after an interrupt in the P36 port. Start the OSC1 oscillation and switch the system clock from IOSC to OSC1. Then stop the IOSC.

Example of Output

```
 -CPU clock- seCLG_IOSC (20000000)
P35 output state: Low
P36 input state: Low
P36 input waits for RISING edge interrupt
P35 output is going to HIGH...
Interrupt on P36 occured
P35 output state: High
P36 input state: High

P35 output state: High
P36 input state: High
P36 input waits for FALLING edge interrupt
P35 output is going to LOW...
Interrupt on P36 occured
P35 output state: Low
P36 input state: Low

Exit
```

3.8 Quad Synchronous Serial Interface (QSPI)

QSPI module is a Quad Synchronous Serial Interface supporting both master and slave modes.

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

Hardware Setup

The micron flash on the 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.
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 three QSPI modes:
   - Quad mode Erase sector, Program Sector, Read and compare sector.
   - Dual mode Erase sector, Program Sector, Read and compare sector.
   - Single mode Erase sector, Program Sector, Read and Compare sector.

Example of Output

```
 -CPU clock- seCLG_IOSC (20000000)
Get bus speed 77519
Set bus speed 10000000
Get bus speed 10000000
QSPI Start: OK
Read external flash register in SINGLE mode: OK
Manufacture ID: 20h, Device ID: ba18h

Set external flash in QUAD mode: OK
Set QSPI in QUAD mode.
Read external flash register in QUAD mode: OK
Erase flash sector in QUAD mode: OK
Program flash in QUAD mode: OK
Read flash in QUAD mode: OK
Compare R/W data in QUAD mode: OK

Set external flash in DUAL mode: OK
Set QSPI in DUAL mode.
Read external flash register in DUAL mode: OK
Erase flash sector in DUAL mode: OK
Program flash in DUAL mode: OK
Read flash in DUAL mode: OK
Compare R/W data in DUAL mode: OK

Set external flash in SINGLE mode: OK
Set QSPI in SINGLE mode.
Read external flash register in SINGLE mode: OK
```

```
Erase flash sector in SINGLE mode: OK
Program flash in SINGLE mode: OK
Read flash in SINGLE mode: OK
Compare R/W data in SINGLE mode: OK
Exit
```

3.9 Quad Synchronous Serial Interface with DMA (QSPI_DMA)

QSPI module is a Quad Synchronous Serial Interface supporting both master and slave modes.

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

Hardware Setup

The micron flash on the 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 10000000.
3. Then it configures DMA Controller descriptors and init 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:
   - Erase Sector
   - Program Sector using register access DMA transfers.
   - Read Sector using register access DMA transfers and compare with the programmed pattern.
8. The example software configures QSPI controller to Memory mapped mode.
9. At the end the example software reads a flash sector using memory mapped DMA transfers and compares it with the previously programmed pattern.

Example of Output

```
-CPU clock- seCLG_IOSC (20000000)
Set bus speed 10000000
QSPI Start: OK
Set external flash in SINGLE mode: OK
Erase flash sector in SINGLE mode: OK
Program flash using DMA in Register Access SINGLE mode: OK
Read flash using DMA in Register Access SINGLE mode: OK
Compare R/W data in Register Access SINGLE mode: OK
Set external flash in QUAD mode: OK
QSPI Start: OK
Read flash using DMA in Memory Mapped QUAD mode: OK
Compare R/W data using DMA in Memory Mapped QUAD mode: OK
Exit
```

3.10 Quad Synchronous Serial Interface Master (QSPI_MASTER)

QSPI module is a Quad Synchronous Serial Interface supporting both master and slave modes.

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

Hardware Setup

1. The S5U1C31D01T1 evaluation board has an onboard serial flash device attached to the QSPI interface. 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 0-ohm resistors R30 to R35.

   Note: Ensure that the 0-ohm connections for these resistors are restored after evaluating the QSPI master/slave sample programs.

2. Connect the evaluation boards where the QSPI master/slave sample programs are installed. Connect each port as follows:

```
              DUAL or QUAD mode
      [master]                    [slave]

 TH34   QSDI0n3------------><-----------QSDIOn3   TH34

 TH36   QSDI0n2------------><-----------QSDIOn2   TH36

 TH38   QSDI0n1------------><-----------QSDIOn1   TH38

 TH39   QSDI0n0------------><-----------QSDIOn0   TH39

 TH40  QSPICLKn------------>>-----------QSPICLKn  TH40

 TH32  #QSPISSn------------>>-----------#QSPISSn  TH32

 J2.1      P36------------>>-----------P36      J2.1

 J2.47     GND------------><-----------GND      J2.47
```

For definition of THx see S5U1C31D01T1 schematic in S5U1C31D01T1 manual.

Please note that the connections are different for single mode.

Uncomment out QSPI_MODE_SINGLE in settings.h for the single mode setup.

```
                  SINGLE mode

      [master]                    [slave]

 TH39   QSDI0n0------------->>----------QSDIOn1  TH38

 TH38   QSDI0n1------------><-----------QSDIOn0  TH39

 TH40  QSPICLKn------------>>-----------QSPICLKn TH40

 TH32  #QSPISSn------------>>-----------#QSPISSn TH32

 J2.1      P36------------>>-----------P36     J2.1

 J2.47     GND------------><-----------GND     J2.47
```

For definition of THx see S5U1C31D01T1 schematic in S5U1C31D01T1 manual.

3. Launch the slave example program first, then the master program

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_CH2 for baud rate generator.
2. Set bus speed to 100000.
3. Assign GPIO P36 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:

```
 -CPU clock- seCLG_IOSC (20000000)
Get bus speed 123456
Set bus speed 100000
Get bus speed 100000
Set QSPI_0 in DUAL 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 QUAD mode.
 - OK(11), NG(0)
 - OK(12), NG(0)
 - OK(13), NG(0)
 - OK(14), NG(0)
 - OK(15), NG(0)
 - OK(16), NG(0)
 - OK(17), NG(0)
 - OK(18), NG(0)
 - OK(19), NG(0)
 - OK(20), NG(0)
```

For Single Mode:

```
 -CPU clock- seCLG_IOSC (20000000)
Get bus speed 123456
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)
```

3.11 Quad Synchronous Serial Interface Slave (QSPI_SLAVE)

QSPI module is a Quad Synchronous Serial Interface supporting both master and slave modes.

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

Hardware Setup

1. The S5U1C31D01T1 evaluation board has an onboard serial flash device attached to the QSPI interface. 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 0-ohm resistors R30 - R35.

   Note: Ensure that the 0-ohm connections for these resistors are restored after evaluating the QSPI master/slave sample programs.

2. Connect the evaluation boards where the QSPI master/slave sample programs are installed.
   Connect each port as follows:

```
               DUAL or QUAD mode
      [master]                    [slave]

TH34   QSDI0n3------------><-----------QSDIOn3   TH34

TH36   QSDI0n2------------><-----------QSDIOn2   TH36

TH38   QSDI0n1------------><-----------QSDIOn1   TH38

TH39   QSDI0n0------------><-----------QSDIOn0   TH39

TH40  QSPICLKn------------>>-----------QSPICLKn  TH40

TH32  #QSPISSn------------>>-----------#QSPISSn  TH32

J2.1      P36------------>>-----------P36      J2.1

J2.47     GND------------><-----------GND      J2.47
```

For definition of THx see S5U1C31D01T1 schematic in S5U1C31D01T1 manual.

Please note that the connections are different for single mode.

```
                 SINGLE mode

      [master]                    [slave]

TH39   QSDI0n0------------->>----------QSDIOn1  TH38

TH38   QSDI0n1------------><-----------QSDIOn0  TH39

TH40  QSPICLKn------------>>-----------QSPICLKn TH40

TH32  #QSPISSn------------>>-----------#QSPISSn TH32

J2.1      P36------------>>-----------P36     J2.1

J2.47     GND------------><-----------GND     J2.47
```

3. Launch the slave example program first, then the master program

Operations

1. Initialize the QSPI module in slave mode as below:
   - Data length is 8bit
   - Data format is MSB first
2. Assign GPIO P36 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.

3.12 IR Remote Controller (REMC3)

The REMC3 circuit generates infrared remote control output signals.

This example programs the REMC3 to generate various IR pulses.

Hardware Setup

This sample program is for the S5U1C31D01T1 evaluation board. For information on a connection refer to the "S5U1C31D01T1 manual".

Operations

This example software 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.

3.13 Real Time Clock (RTCA)

RTCA module is a real-time clock with a perpetual calendar function.

This example shows how to program and read the various functions of the real-time clock 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

```
 -CPU clock- seCLG_IOSC (20000000)
RTCA TRM bits 0x43
RTCA Set hour 4 (PM), minute 17, second 0
RTCA Get hour 4 (PM), minute 17, second 0
RTCA Set year 15, month 2, day 5, Thursday
RTCA Get year, month, day. 2015, 2, 5, Thursday
RTCA Set Alarm hour 4 (PM), minute 17, second 5
RTCA waiting for alarm....
RTCA Alarm occurred : hour 4(PM), minute 17, second 5
RTCA Start Trimming
RTCA waiting for timer....
RTCA Trimming performed : hour 4(PM), minute 17, second 11
RTCA StopWatch (start) : hour 4(PM), minute 17, second 11
RTCA sleep 5 seconds waiting for Stop....
RTCA interrupts disabled for count reading
SysTick 5000
StopWatchHW: 5.14
Exit
```

3.14 Sound Generator (SNDA)

SNDA is a sound generator that generates melodies and buzzer signals.

This example programs the SNDA to generate various tones and music.

Operations

1. Example starts OSC1 and initializes SNDA.
2. SNDA runs in normal buzzer mode for 5 sec.
3. SNDA runs in one-shot buzzer mode for 250msecs.
4. SNDA runs in melody mode.

Example of Output

```
 -CPU clock- seCLG_IOSC (20000000)
Start SNDA in normal buzzer mode for 5 sec
Start SNDA in one-shot buzzer mode, 250ms
Start SNDA in melody mode
Start SNDA playing of 'Postoj Parovoz' music
Exit
```

3.15 Synchronous Serial Interface Master (SPIA_MASTER)

SPIA module is a synchronous serial interface supporting both master and slave modes. Only one SPIA channel is available in S1C31D01 microcontroller.

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

Hardware Setup

1. Connect the evaluation boards where the SPIA master/slave sample programs are installed.
   Connect each port as follows:

```
        [master]                     [slave]
    P32   SDIn------------<<-----------SDOn    P33

    P33   SDOn------------>>-----------SDIn    P32

    P34 SPICLKn------------>>-----------SPICLKn P34

    P35       ------------>>-----------#SPISSn  P35

    P36       ------------>>-----------        P36

    J2.47   GND------------><-----------GND    J2.47
```

2. Launch the slave example program first, then the master program

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_1 for baud rate generator.
2. Set bus speed to 1000000.
3. Assign GPIO P36 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

```
-CPU clock- seCLG_IOSC (20000000)
Set bus speed 1000000
Get bus speed 1000000
OK(1), NG(0)
OK(2), NG(0)
OK(3), NG(0)
OK(4), NG(0)
OK(5), NG(0)
OK(6), NG(0)
```

3.16 Synchronous Serial Interface Slave (SPIA_SLAVE)

SPIA module is a synchronous serial interface supporting both master and slave modes. Only one SPIA channel is available in S1C31D01 microcontroller.

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

Hardware Setup

1. Connect the evaluation boards where the SPIA master/slave sample programs are installed.
   Connect each port as follows:

```
     [master]                    [slave]
P32   SDIn------------<<-----------SDOn    P33

P33   SDOn------------>>-----------SDIn    P32

P34 SPICLKn------------>>-----------SPICLKn P34

P35       ------------>>-----------#SPISSn  P35

P36       ------------>>-----------        P36

J2.47   GND------------><-----------GND    J2.47
```

2. Launch the slave example program first, then the master program

Operations

1. Initialize the SPIA module in slave mode as below:
   - Data length is 8bit
   - Data format is MSB first
2. Assign GPIO P36 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.

3.17 Power Supply Voltage Detection Circuit (SVD3)

SVD3 is a supply voltage detector to monitor the power supply voltage on the VDD pin or the voltage applied to an external pin.

This example shows VDD voltage drop detection.

Hardware Setup

Example requires evaluation board configured to power from the debugger adapter.

A jumper across J2-35 and J2-36 must be installed.

Connect evaluation board to PC by USB cable.

Operations

1. Software prepares to watch the VDD pin dropping the voltage below some limit.
2. USB cable is disconnected.
3. Software detects voltage drop and reports it on the terminal.

Example of Output

```
 - CPU clock- seCLG_IOSC (20000000)
SVD is configured to detect the voltage drop.
Please Disconnect USB cable.
Power voltage drop detected
Interrupt Low voltage occurred
IRQ handler called
Exit
```

3.18 16-Bit Timer (T16)

T16 module is a 16-bit timer.

This example programs the T16 channel 0 timer to interrupt periodically and counts the number of interrupts that occurred within a 5-second interval.

Operations

1. IOSC is selected as timer clock source. Example code does IOSC 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

```
 -CPU clock- seCLG_IOSC (20000000)
CLG IOSC Auto-trimming ok
IRQ interrupted: 1695 times during 5 sec
Exit
```

3.19 16-bit PWM Timer (T16B)

T16B module is a 16-bit PWM timer with comparator/capture functions.

This example exercises the various comparator/capture functions of the 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

```
 -CPU clock- seCLG_IOSC (20000000)
Comparator Count = 0x2000
Timer Count = 0x0010
Max Count = 0x5000
Interrupt Max Count = 10
Interrupt Zero Count = 11
Interrupt Comparator Count = 10
Interrupt Capture Count = 20
 - CaptureCountData[0] = 0x2007
- CaptureCountData[1] = 0x0005
- CaptureCountData[2] = 0x2007
- CaptureCountData[3] = 0x0005
- CaptureCountData[4] = 0x2007
- CaptureCountData[5] = 0x0005
- CaptureCountData[6] = 0x2007
- CaptureCountData[7] = 0x0005
- CaptureCountData[8] = 0x2007
- CaptureCountData[9] = 0x0005
- CaptureCountData[10] = 0x2007
- CaptureCountData[11] = 0x0005
- CaptureCountData[12] = 0x2007
- CaptureCountData[13] = 0x0005
- CaptureCountData[14] = 0x2007
- CaptureCountData[15] = 0x0005
- CaptureCountData[16] = 0x2007
- CaptureCountData[17] = 0x0005
- CaptureCountData[18] = 0x2007
- CaptureCountData[19] = 0x0005
Exit
```

3.20 Temperature Sensor and Voltage Reference (TSRVR)

MDC is a Controller for Memory Displays with a Drawing and Copying engine.

This software shows examples of using the routines in "se_mdc.c" peripheral library for initializing a panel, using drawing functions, and using image/bitmap copying functions. It also shows examples for using the ADC12A ("se_adc12a.c")and TSRVR ("se_tsrvr.c") peripherals. The "tsrvr_temperature.*" files contain the routines for reading temperature using ADC12A and TSRVR. An analog clock and temperature value is displayed on the panel. The clock and temperature value are updated every second by the RTC interrupt.

Hardware Setup

Example requires an LPM012M134B panel connected to connector CN3 of the S5U1C31D01T1 evaluation board.

Operations

1. Start OSC1 and change IOSC frequency to 20MHz.
2. Initialize temperature reading routines.
3. Initialize LPM012M134B panel.
4. Set output destination window for MDC graphics engine
5. Fill display buffer with white background.
6. Draw line "ticks" (1-minute intervals) for analog clock.
7. Initialize and start RTC.
8. Infinite "while" loop:
    - Read temperature.
    - Read latest time from RTC.
    - Draw circles in center of analog clock.
    - Draw rectangle border for "weekday" string.
    - Render "weekday" 3-character string.
    - Draw two ellipses logo.
    - Render temperature value at bottom part of display.
    - Draw line for "second" hand.
    - Render "hour" hand.
    - Render "minute" hand.
    - Update display (copy image from frame buffer to panel).
    - Go to deep sleep waiting for 1-second interrupt from RTC.
    - (wake up from sleep)
    - Fill temperature display area with white.
    - Erase the "second" hand.
    - Erase the "hour" hand.
    - Erase the "minute" hand.
    - Fill "weekday" area with white.

3.21 UART (UART3)

The UART3 module provides an asynchronous serial interface.

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

Hardware Setup

1. Connect S5U1C31D01T1 evaluation board to PC via USB Adapter for UART as it is explained in Section 2.2.1 and shown on the Figure 2.2.1.3 and on Figure 2.2.1.4.
2. Run a terminal program.
3. Configure terminal program for baud rate 115200, 8 data bits, 1 stop bit, no 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

```
 -CPU clock- seCLG_IOSC (20000000)
abcefghiabcefghi ... waiting for console input...
1234567812345678
Exit
```

3.22 USB CDC Device (USB_CDC)

USB module is a USB target device controller that supports FS mode based on the USB 2.0 standard. The Example code for USB module is a CDC ACM (COM PORT).

The example application echoes back the character received from the COM PORT.

Hardware Setup

Jumper on J2 pins 35 and 36 must be installed.

Connect evaluation board to PC by USB cable.

3.23 USB HID (USB_HID_S5U1C31D01T1)

USB module is a USB target device controller that supports FS mode based on the USB 2.0 standard. The Example code for USB module is a Human Interface Device (HID) device. The open-source CMSIS USB stack is used for the HID example software.

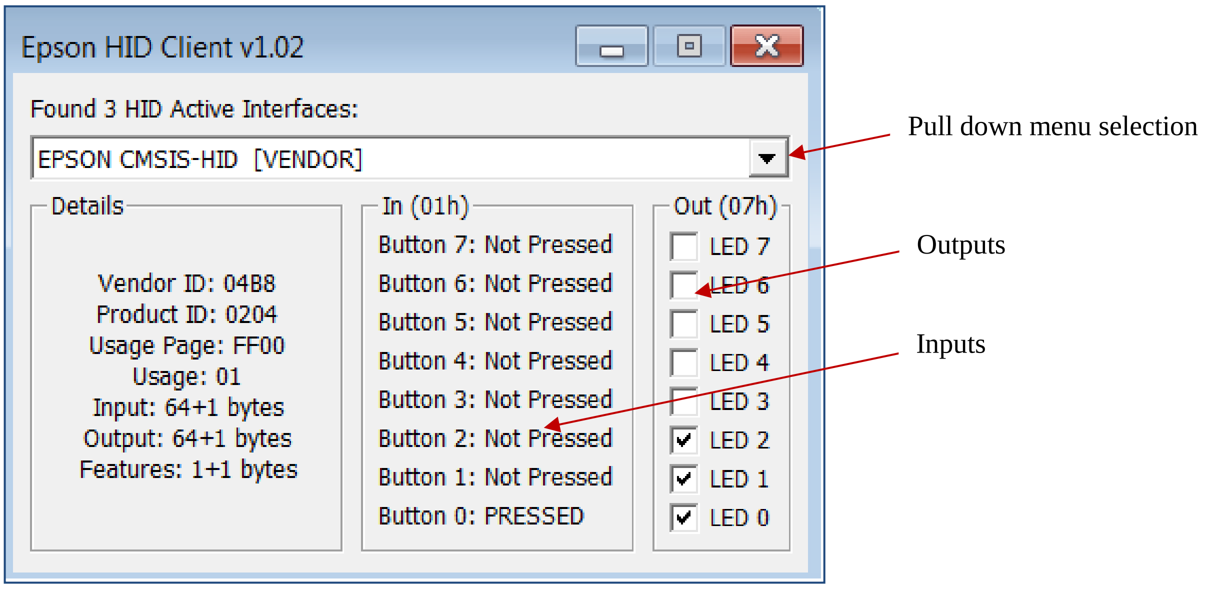
HID devices must meet a few general requirements that are imposed to keep the HID interface standardized and efficient. All HID devices must have a control endpoint (Endpoint 0), and an interrupt IN endpoint. Many devices also use an interrupt OUT endpoint. In most cases, HID devices are not allowed to have more than one OUT and one IN endpoint. All data transferred must be formatted as reports whose structure is defined in the report descriptor. HID devices must respond to standard HID requests in addition to all standard USB requests.

Hardware Setup

1. Jumper on J2 pins 35 and 36 must be installed.
2. Example requires an LPM012M134B panel connected to connector CN3 of the S5U1C31D01T1 evaluation board.
3. Connect the evaluation board to PC by USB cable.

Operations

1. On the PC side, start the Tools\HidClient.exe utility.
2. In the utilities window select EPSON CMSIS-HID
3. Click on outputs to observe bits on the S5U1C31D01T1 evaluation board display.
4. Push SW2, SW3 buttons on the board to observe inputs in the HidClient.exe utility window.



Found 3 HID Active Interfaces:

EPSON CMSIS-HID [VENDOR]

Details

Vendor ID: 04B8
Product ID: 0204
Usage Page: FF00
Usage: 01
Input: 64+1 bytes
Output: 64+1 bytes
Features: 1+1 bytes

In (01h)

Button 7: Not Pressed
Button 6: Not Pressed
Button 5: Not Pressed
Button 4: Not Pressed
Button 3: Not Pressed
Button 2: Not Pressed
Button 1: Not Pressed
Button 0: PRESSED

Out (07h)

☐ LED 7
☐ LED 6
☐ LED 5
☐ LED 4
☐ LED 3
☑ LED 2
☑ LED 1
☑ LED 0

Figure 3.21 Epson HID Client Utility

- Click LED0-LED7 outputs and observe bits set on the S5U1C31D01T1 evaluation board display.
- Push SW2, SW3 buttons on the board to observe the “PRESSED” status in the utility’s window.

3.24 USB MSC Device (USB_MSC)

USB module is a USB target device controller that supports FS mode based on the USB 2.0 standard. The Example code for USB module is an MSC device.

The example application presents a RAM disk drive with a README.txt file to the PC.

Hardware Setup

1. Jumper on J2 pins 35 and 36 must be installed.
2. Connect the evaluation board to PC by USB cable.

Example of Output

```
 -CPU clock- seCLG_IOSC (20000000)
Reset
Suspend
Resume
Reset
Reset
Configure
```

3.25 WatchDog Timer (WDT2)

WDT2 module restarts the system if a problem occurs.

This example shows how to program the 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

```
 -CPU clock- seCLG_IOSC (20000000)
WDT Clock source = (1)
WDT Clock divider = (0)
WDT CLK = 250 Hz. CMP count = 1023
WDT 4 secs cycle is set
NMI interrupts due to comparator match in NMI Mode.
WDT NMI interrupts = 1

Watchdog timer was reset in Reset and NMI Mode.
Chip was alive after NMI
WDT Clock source = (1)
WDT Clock divider = (0)
WDT CLK = 250 Hz. CMP count = 250
WDT 1 secs cycle is set
WDT NMI interrupts = 5

Watchdog timer was not reset in Reset and NMI Mode.
Chip was reset on next comparator match.
WDT Clock source = (1)
WDT Clock divider = (0)
WDT CLK = 250 Hz. CMP count = 250
WDT 1 secs cycle is set
```

3.26 Flash Programming (Flash)

This example erases and programs the internal flash memory in the S1C31D01 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 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 "S1C31D01 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.

3.27 EEPROM Emulation Library (EEPROM)

This example writes data to the emulated EEPROM which uses the internal flash memory of the S1C31D01 using the seEepromLibrary, and then reads back and compares 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 0x3FFFF is used for EEPROM emulation and provides 512 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/S5U1C31D01T1/IAR/config/S1C31D01_flash.icf”.

For MDK-ARM(uVision), 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/S5U1C31D01T1/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.

```
#defineCONFIG_EEPROM_SIZE_MAX(512)
#defineCONFIG_RETRY_COUNT(4)
```

The “CONFIG_EEPROM_SIZE_MAX” indicaates the size of the emulated EEPROM. This size can be selected one of 32/64/128/256/512.

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.

3.28 Boot Loader (BootLoader)

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

For details of this example, refer to "S1C31D01 Boot Loader Manual" on the separate document.

Appendix. Use of Free Trial Version IDE

You can build the example projects in this sample software using the free trial version of IDEs, for example, IAR EWARM or MDK-ARM(uVision). For the free trial version, you can select the version of “time-limited with fully functional” or the version of “size-limited without any time limit”. However, when selecting the version of “size-limited without any time limit”, some example projects may not be built due to the code-size limitation. Below is the list of example projects which can be built using the size-limited version.

Table A.1 Available Example Projects

Example Project Name	**IAR EWARM**	**MDK-ARM（uVision）**
CLG	Available	Available
DMAC	Available	Available
Flash	Available	Available
EEPROM	Available	Available
I2C_S5U1C31D01T1	Available	Available
MDC_LPM012M134B	N/A	N/A
MDC_LPM012M134B_SERFLASH	N/A	N/A
MDC_LS012B7DH02	N/A	N/A
PPORT	Available	Available
QSPI	Available	Available
QSPI_DMA	Available	Available
QSPI_MASTER	Available	Available
QSPI_SLAVE	Available	Available
REMC3	Available	Available
RTCA	Available	Available
SNDA	Available	Available
SPIA_MASTER	Available	Available
SPIA_SLAVE	Available	Available
SVD3	Available	Available
T16	Available	Available
T16B	Available	Available
TSRVR	N/A	N/A
UART3	Available	Available
USB_CDC	Available	Available
USB_HID_S5U1C31D01T1	N/A	N/A
USB_MSC	N/A	N/A
WDT2	Available	Available

Note: Depending on the version of the free trial version IDE, example projects may not be build. Also, the above table is not intended to guarantee the operation of example projects to be built using the free trial version IDE.

Revision History

Attachment-1

Rev. No.	Date	Page	Category	Contents
Rev.1.00	05/29/2017	All	New	New establishment.
Rev.2.00	08/03/2017	1	Modify	Modified section 1.1 to support ENVPP pin and Bridge Board Ver.2.
		4-24	Change	Changed sections 2.2, 2.3 and 2.4 significantly to clarify procedures regarding the sample software operations and support ENVPP pin and Bridge Board Ver.2.
		63	Add	Added section 3.26 to support the sample software package Ver.2.0.
Rev.2.10	06/06/2018	8,16	Change	Changed to Sample Software installer and setup utility executables
		65	Add	Added “Appendix”.
		Last	Change	Updated contact information.
Rev.3.00	02/26/2019	4,6,10, 14,19	Change	Modified section 2.2.1, 2.3.1, 2.3.5, 2.4.1 and 2.4.5.
		62,63	Add	Added section 3.27 and 3.28.
		-	Delete	Deleted section 2.2.3.

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

Seiko Epson Corp.
Korea Office
19F, KLI 63 Bldg, 60 Yoido-dong,
Youngdeungpo-Ku, Seoul 150-763, Korea
Phone: +82-2-784-6027 FAX: +82-2-767-3677

Seiko Epson Corp.
Sales & Marketing Division

Device Sales & Marketing Department
421-8, Hino, Hino-shi, Tokyo 191-8501, Japan
Phone: +81-42-587-5816 FAX: +81-42-587-5116

Document Code: 413471503
First Issue May 2017
Revised February 2019 in Japan