EPSON
EXCEED YOUR VISION

S1C17 Family Application Note

S1C17 Series Peripheral Circuit Sample Software

SEIKO EPSON CORPORATION

Rev.1.1

Evaluation board/kit and Development tool important notice

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

NOTICE

Table of Contents

1. Overview 1

1.1 Operating Environment 1

2. Sample Software Configuration 2

2.1 Configuration of Directories and Files 2

3. How to Run Sample Software 4

3.1 Software Development Environment 4

3.2 Installing Tools 4

3.3 Importing Project 5

3.3.1 Importing Sample Software (Common to GNU17 Version 2 and Version 3) 5

3.3.2 Importing Sample Software (GNU17 Version 2) 6

3.3.3 Importing Sample Software (GNU17 Version 3) 9

3.4 Connecting with Target 12

3.5 Operation of ICDmini and Target 12

3.5.1 When ICDmini Ver. 2.0 is Used 12

3.5.2 When ICDmini Ver. 3.0 is Used 13

3.6 Building Project and Starting Debugger 14

3.6.1 When GNU17 Version 2 is Used 14

3.6.2 When GNU17 Version 3 is Used 15

4. Details of Sample Software Functions 16

4.1 Models and Supported Functions 16

4.2 Initializing Peripheral Circuits 17

4.2.1 “init_config.h” Configuration Examples 17

4.2.2 Precautions on Editing “init_config.h” 18

4.3 I/O Ports (PPORT) 19

4.3.1 Sample Program Specifications 19

4.3.2 Hardware Conditions 19

4.3.3 Operations Overview 19

4.4 Clock Generator (CLG) 20

4.4.1 Sample Program Specifications 20

4.4.2 Hardware Conditions 20

4.4.3 Operations Overview 20

4.5 16-bit Timer (T16) 21

4.5.1 Sample Program Specifications 21

4.5.2 Hardware Conditions 21

4.5.3 Operations Overview 21

4.6 16-bit PWM Timer (T16B) 22

4.6.1 Sample Program Specifications 22

4.6.2 Hardware Conditions 22

4.6.3 Operations Overview 22

4.7 Real-Time Clock (RTCA, RTCA2) 23

4.7.1 Sample Program Specifications 23

4.7.2 Hardware Conditions 23

4.7.3 Operations Overview 23

4.8 Watchdog Timer (WDT, WDT2) 24
4.8.1 Sample Program Specifications 24
4.8.2 Hardware Conditions 24
4.8.3 Operations Overview 24
4.9 UART (UART, UART2, UART3) 25
4.9.1 Sample Program Specifications 25
4.9.2 Hardware Conditions 25
4.9.3 Operations Overview 25
4.9.3.1 Master Sample Program Operations Overview 25
4.9.3.2 Slave Sample Program Operations Overview 25
4.9.4 Calibration Function 26
4.10 SPI (SPIA) 27
4.10.1 Sample Program Specifications 27
4.10.2 Hardware Conditions 27
4.10.3 Operations Overview 27
4.10.3.1 Master Sample Program Operations Overview 27
4.10.3.2 Slave Sample Program Operations Overview 27
4.11 I^2C（I2C） 28
4.11.1 Sample Program Specifications 28
4.11.2 Hardware Conditions 28
4.11.3 Operations Overview 28
4.11.3.1 Master Sample Program Operations Overview 28
4.11.3.2 Slave Sample Program Operations Overview 28
4.12 LCD Driver (LCD4A) 29
4.12.1 Sample Program Specifications 29
4.12.2 Hardware Conditions 29
4.12.3 Operations Overview 29
4.13 LCD Driver (LCD8A) 30
4.13.1 Sample Program Specifications 30
4.13.2 Hardware Conditions 30
4.13.3 Operations Overview 30
4.14 LCD Driver (LCD8B) 31
4.14.1 Sample Program Specifications 31
4.14.2 Hardware Conditions 31
4.14.3 Operations Overview 31
4.15 LCD Driver (LCD16A) 32
4.15.1 Sample Program Specifications 32
4.15.2 Hardware Conditions 32
4.15.3 Operations Overview 32
4.16 LCD Driver (LCD24A) 33
4.16.1 Sample Program Specifications 33
4.16.2 Hardware Conditions 33
4.16.3 Operations Overview 33
4.17 LCD Driver (LCD32B) 34
4.17.1 Sample Program Specifications 34
4.17.2 Hardware Conditions 34
4.17.3 Operations Overview 34
4.18 Seven-Segment LED Controller (LEDC) 35
4.18.1 Sample Program Specifications 35
4.18.2 Hardware Conditions 35
4.18.3 Operations Overview 35

4.19 R/F Converter (RFC) 36
4.19.1 Sample Program Specifications 36
4.19.2 Hardware Conditions 36
4.19.3 Operations Overview 36
4.20 Sound Generator (SNDA) 37
4.20.1 Sample Program Specifications 37
4.20.2 Hardware Conditions 37
4.20.3 Operations Overview 37
4.21 Supply Voltage Detector (SVD, SVD3) 38
4.21.1 Sample Program Specifications 38
4.21.2 Hardware Conditions 38
4.21.3 Operations Overview 38
4.22 Power Generator (PWG, PWG2) 39
4.22.1 Sample Program Specifications 39
4.22.2 Hardware Conditions 39
4.22.3 Operations Overview 39
4.23 IR Remote Controller (REMC, REMC2, REMC3) 40
4.23.1 Sample Program Specifications 40
4.23.2 Hardware Conditions 40
4.23.3 Operations Overview 41
4.24 Software IR Remote Controller (SOFT REMC) 42
4.24.1 Sample Program Specifications 42
4.24.2 Hardware Conditions 42
4.24.3 Operations Overview 42
4.25 12-bit A/D Converter (ADC12A) 43
4.25.1 Sample Program Specifications 43
4.25.2 Hardware Conditions 43
4.25.3 Operations Overview 43
4.26 Operational Amplifier/Comparator (OPCMP) 44
4.26.1 Sample Program Specifications 44
4.26.2 Hardware Conditions 44
4.26.3 Operations Overview 44
4.27 Temperature Sensor/Reference Voltage Generator (TSRVR) 45
4.27.1 Sample Program Specifications 45
4.27.2 Hardware Conditions 45
4.27.3 Operations Overview 45
Revision History 46

1. Overview

This manual describes the usage and behavior of the S1C17 Series sample software.

The S1C17 Series sample software is intended to show usage examples of the peripheral circuits embedded in the S1C17 Series microcontrollers.

In addition to this document, refer to the model information and technical manual for each model, and the S5U1C17001C Manual.

1.1 Operating Environment

To run the S1C17 Series sample software, the following tools are required:

- A board with an S1C17 Series microcontroller mounted
- S5U1C17001H (hereinafter referred to as ICDmini)
- S5U1C17001C (hereinafter referred to as GNU17)

Notes: • GNU17 v2.4.0 and v3.1.0 were used for checking the operations of this sample software.
• There is more than one version of GNU17 and ICDmini available. For the ICDmini supported by the GNU17 to be used, refer to the S5U1C17001C Manual.

2. Sample Software Configuration

This chapter shows the file configuration of the S1C17 Series sample software.
The S1C17 Series sample software consists of "Sample software" for checking the behavior of each peripheral circuit and "Sample driver" for controlling each peripheral circuit.

2.1 Configuration of Directories and Files

The following shows the directory configuration of the S1C17 Series sample software:

Note: xxx = model name

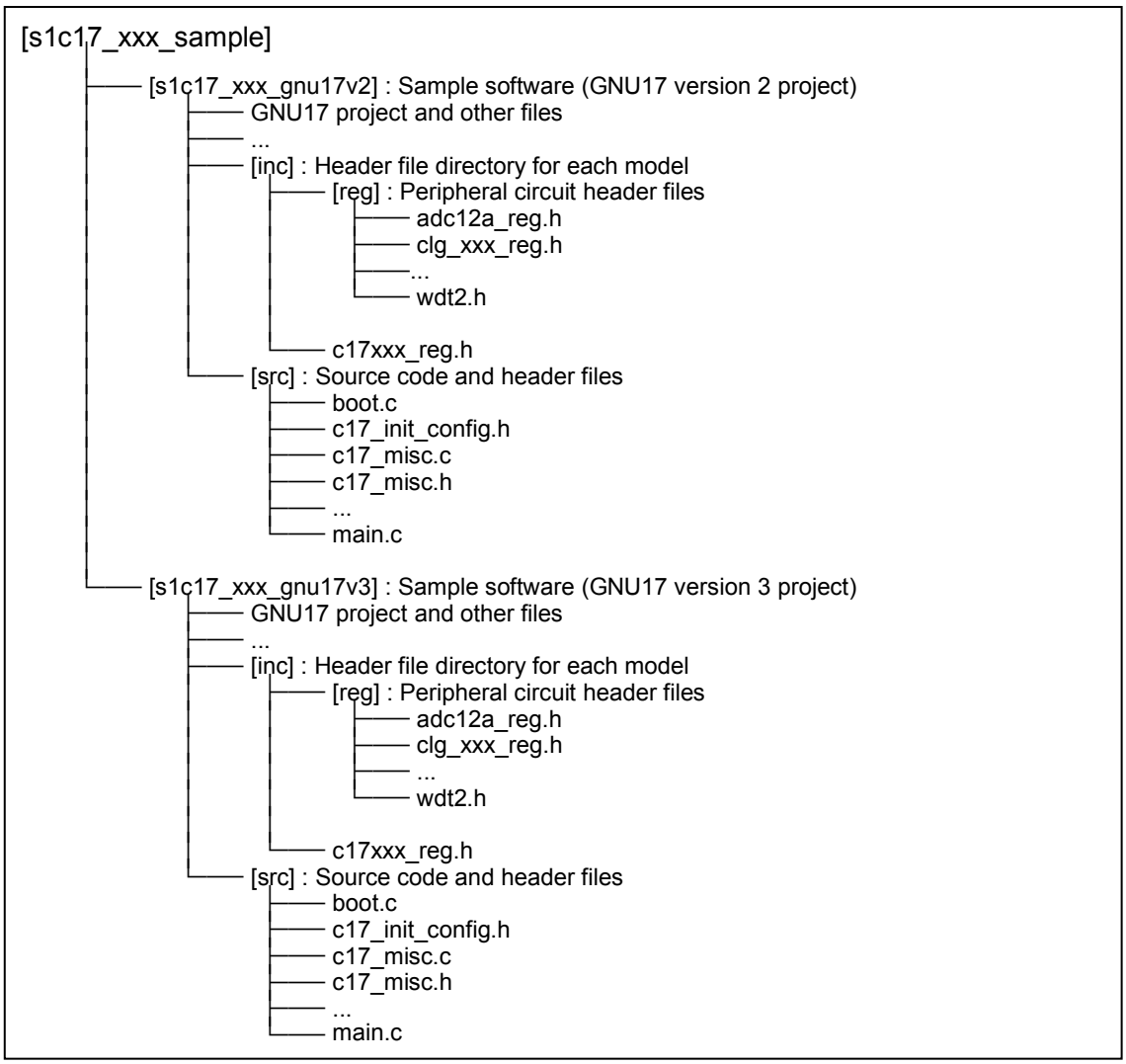


Figure 2.1.1 Directory Configuration of S1C17 Series Sample Software

(1) "s1c17_xxx_sample" directory

This directory contains the S1C17xxx sample software.

(2) "s1c17_xxx_gnu17vx" directory

This directory contains the files that constitute a GNU17 project. "_gnu17v2" or "_gnu17v3", the last part of the directory name, represents the GNU17 version.

(3) "inc" directory

This directory contains the header file in which the model-specific information, such as registers, is defined. It also contains the header file for selecting an S1C17 Series microcontroller.

- c17xxx_reg.h - Header file in which the S1C17xxx register addresses and other information are defined
- c17_mcu_select.h - Header file for selecting the S1C17 Series microcontroller to be used

(4) "reg" directory

This directory contains the header files in which the information for each peripheral circuit, such as the control bit assignments, is defined.

Example: clg_xxx_reg.h - Header file in which the control bit assignments and other information for a peripheral circuit (CLG in this case) are defined.

(5) "src" directory

This directory contains microcontroller initialization programs, sample programs and drivers for controlling the peripheral circuits, and the header files in which the constants and other values used in the sample software and drivers are defined.

Examples:

boot.c	- File that contains a vector table and initialization processing routines
main_clg.c, main_port.c	- Sample program files for each peripheral circuit
c17_misc.c, c17_clg.c	- Sample driver files for each peripheral circuit
c17_misc.h, c17_clg.h	- Sample driver header files for each peripheral circuit
c17_init_config.h	- Header file for configuring the peripheral circuit

3. How to Run Sample Software

This chapter describes the software development environment and how to run the sample software.

3.1 Software Development Environment

This manual assumes that the tools listed below are used for software development. For detailed information on each tool, refer to each tool manual.

- S5U1C17001C (GNU17) ver. 2.4.0 or ver. 3.1.0
- Model-specific information files (S1C17 Series)
- S5U1C17001H (ICDmini) ver. 2.0 or ver. 3.0

3.2 Installing Tools

(1) GNU17 version 2/version 3 (S5U1C17001C)

- Download the compressed GNU17 file from the Epson Microcontrollers Site.
  http://global.epson.com/products_and_drivers/semicon/products/micro_controller/16bit/sw_tool.html
- Unzip the downloaded GNU17 file into any folder.
- Run "Setup.exe" to install the GNU17 tools.

(2) Model-specific information files

- Download the model-specific information tool from the Epson Microcontrollers Site.
  http://global.epson.com/products_and_drivers/semicon/products/micro_controller/16bit/sw_tool.html
- Unzip the downloaded file and copy over the model-specific information files to the [mcu_model] folder under the folder where GNU17 was installed.

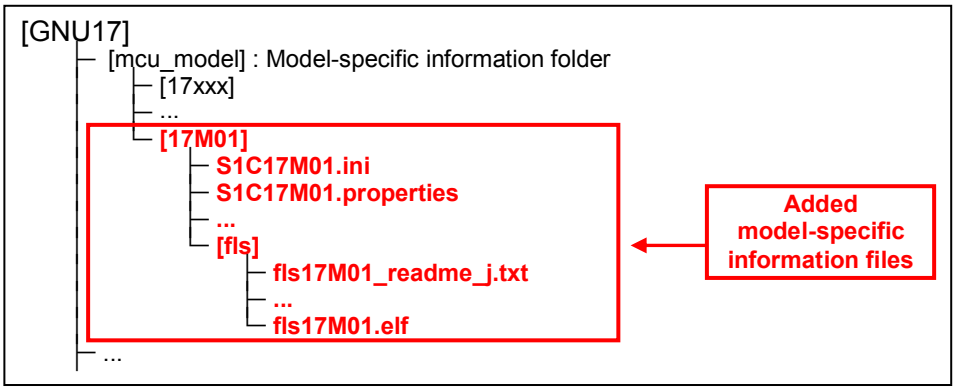


Figure 3.2.1 Adding Model-Specific Information Files (example for S1C17M01)

(3) ICDmini ver. 2.0 (S5U1C17001H)

- After installing GNU17 version 2, connect the ICDmini to the PC using a USB cable.
- Install the USB driver to the PC. It exists in the folder where GNU17 was installed. The following is the default USB driver directory when GNU17 was installed in the "C:\EPSON\GNU17" directory.
  For 32-bit OS: C:\EPSON\GNU17\utility\drv_usb\OS_32bit
  For 64-bit OS: C:\EPSON\GNU17\utility\drv_usb\OS_64bit

(4) ICDmini ver. 3.0 (S5U1C17001H)

- After installing GNU17 version 3, connect the ICDmini to the PC using a USB cable.
- Install the USB driver to the PC. It exists in the folder where GNU17 was installed. The following is the default USB driver directory when GNU17 was installed in the "C:\EPSON\GNU17V3" directory.
  C:\EPSON\GNU17V3\utility\drv_usb\Mini3Driver

3.3 Importing Project

This section describes how to import a sample software project into GNU17.

Notes: • For more information on importing a project, refer to the S5U1C17001C Manual.
• The project import procedure is different between GNU17 version 2 and version 3.

3.3.1 Importing Sample Software (Common to GNU17 Version 2 and Version 3)

(1) Launching GNU17

Launch IDE by double-clicking the "eclipse.exe" icon that exists in the "eclipse" sub-directory under the directory where GNU17 was installed, or by selecting [EPSON MCU] > [GNU17] > [GNU17 IDE] (for GNU17 version 2) or [EPSON MCU] >[GNU17V3] > [GNU17V3 IDE] (for GNU17 version 3) from the Windows startup menu.

(2) Unzipping the sample software

Unzip the sample software downloaded from the Epson Microcontrollers Site.

(3) Editing the header file

Edit the "reg\c17_mcu_select.h" header file in the sample software according to the target model. Uncomment the macro definition line for the target model as shown below (other target model definitions must be commented out).

Examples: • When using an S1C17M01 sample software

```
// Please activate macro of used MCU.
#define C17_MCUSEL_C17M01
//#define C17_MCUSEL_C17W22
//#define C17_MCUSEL_C17W23
```

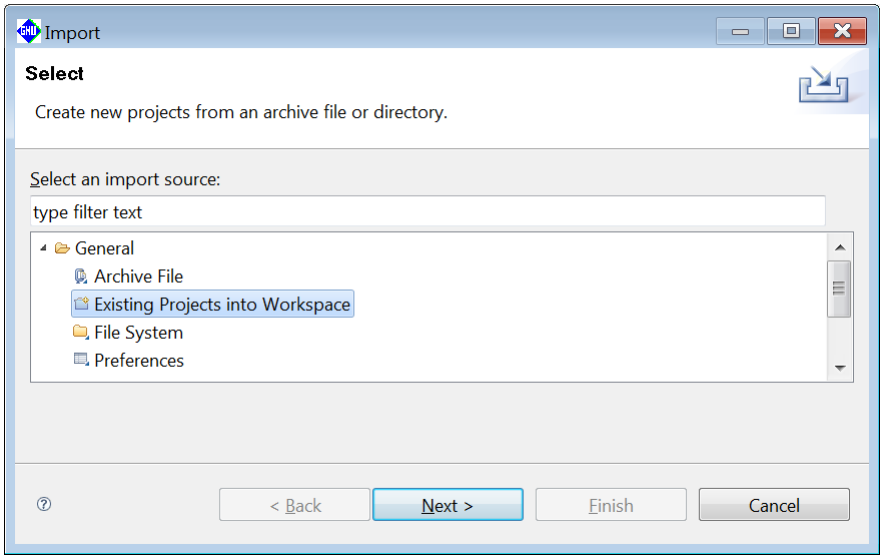
• When using an S1C17W22 sample software

```
// Please activate macro of used MCU.
//#define C17_MCUSEL_C17M01
#define C17_MCUSEL_C17W22
//#define C17_MCUSEL_C17W23
```

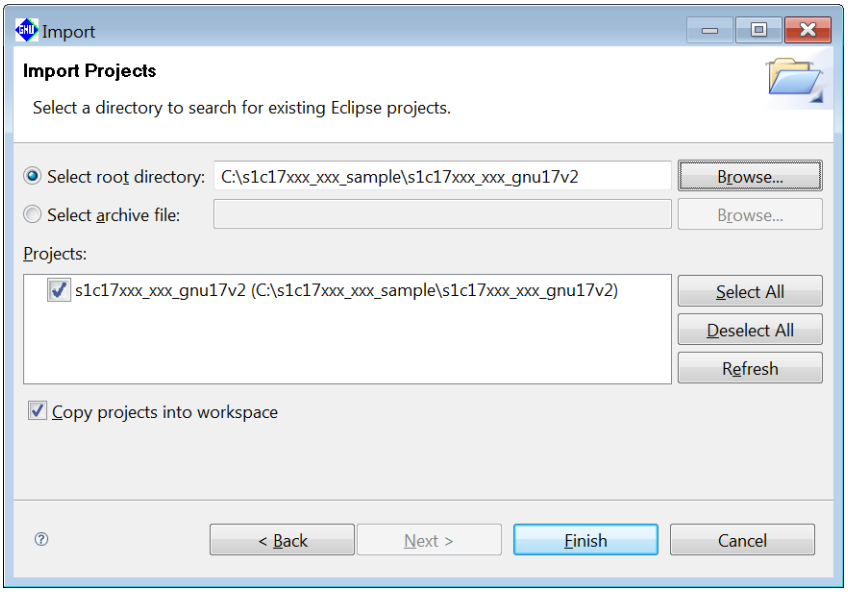
3.3.2 Importing Sample Software (GNU17 Version 2)

(1) Importing a sample software project

Select [Import...] from the [File] menu of GNU17 to start the [Import] wizard. Select [General] > [Existing Projects into Workspace] and then click the [Next >] button.



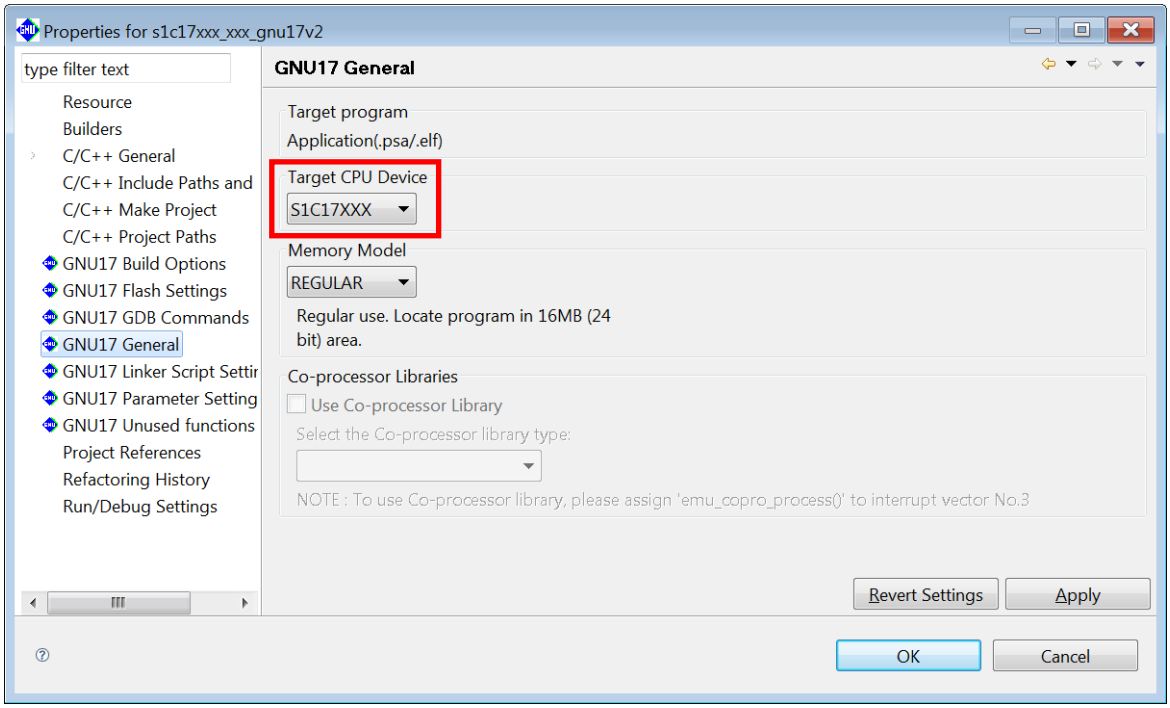
Click the [Browse...] button at [Select root directory:] and select the "xxx_gnu17v2" project that exists in the sample software folder extracted in Step (2) of Section 3.3.1.



Select the [Copy projects into workspace] checkbox. This enables copying of the project into the workspace directory so that the original files will not be modified. Finally, click the [Finish] button. After making sure that the sample project has been imported to GNU17, the folder extracted in Step (2) of Section 3.3.1 can be removed.

(2) Selecting a target CPU

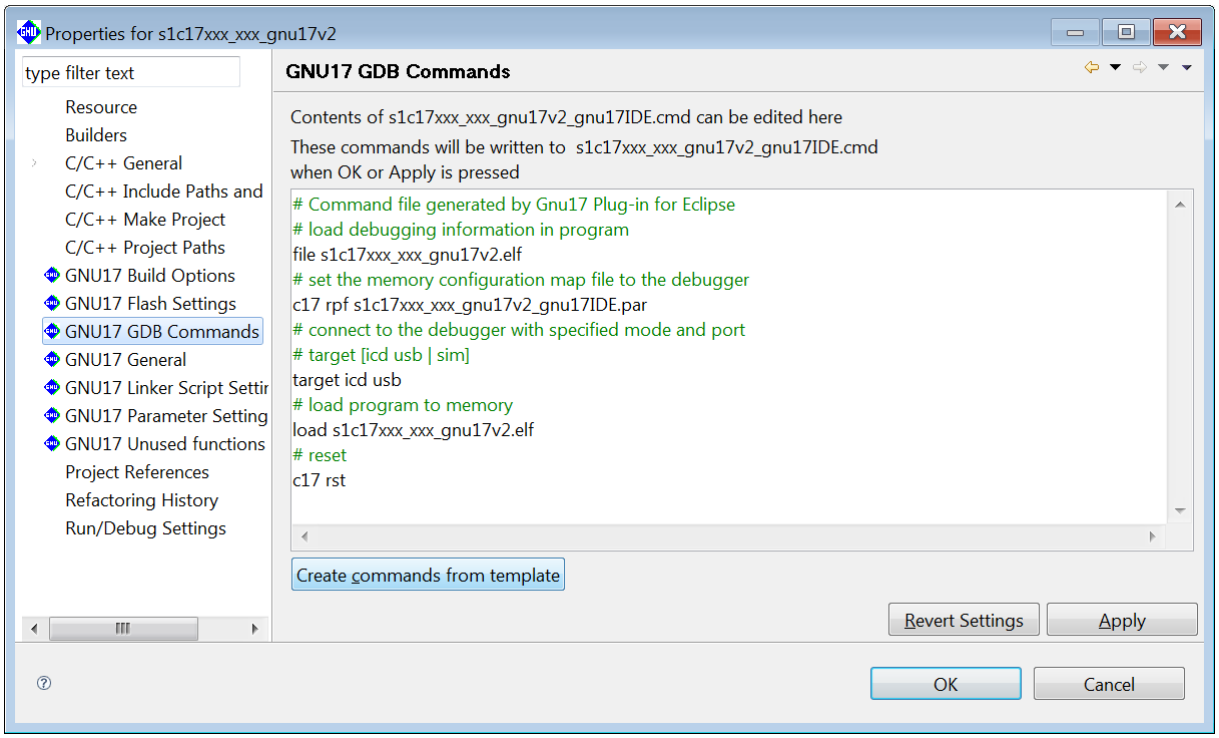
The target CPU to run the sample software should be selected. Select the sample software project in the [C/C++ project] view and select [Properties] from the [Project] menu to open the [Properties] dialog. Select [GNU17 General] from the property list in the dialog.



Select a target CPU from the [Target CPU Device] dropdown list, for example, select “S1C17W22” if it is the target CPU. Then, click the [Apply] button.

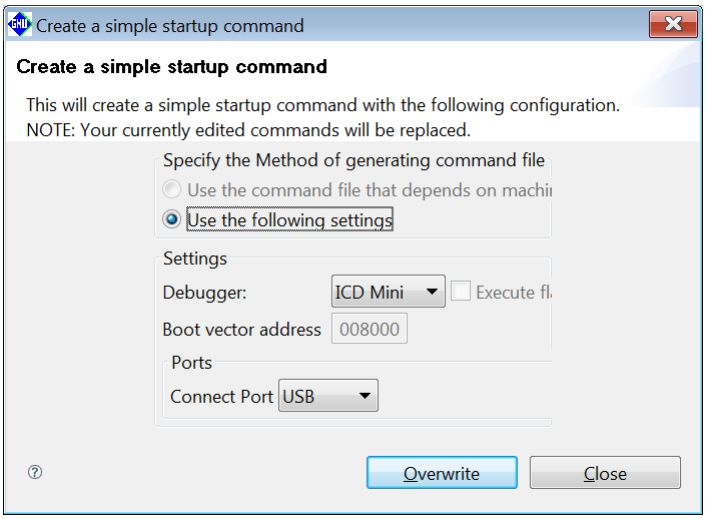
(3) Setting the debugger startup options

The debugger startup options should be selected. Select [GNU17 GDB Commands] from the property list on the [Properties] dialog.



This page displays the contents of the debugger startup command file that will be generated by the IDE. The debugger must be set to the appropriate mode that suits the microcontroller and ICD used.

Create the debugger startup command file with the procedure shown below.
Click the [Create commands from template] button to open the [Create a simple startup command] dialog.
Select "ICD Mini" from the [Debugger:] dropdown list.



Click the [Overwrite] button, then click the [OK] button in the [Overwrite commands] dialog that appears.
The commands may be added and edited directly in the [GNU17 GDB Commands] page as necessary.

* Precautions on using simulated I/O

When using the simulated I/O function, which allows emulation of the external input/output function such as a serial interface by inputting/outputting from/to a standard input/output (stdin, stdout) device or a file, append the command lines shown below to the GDB command file.

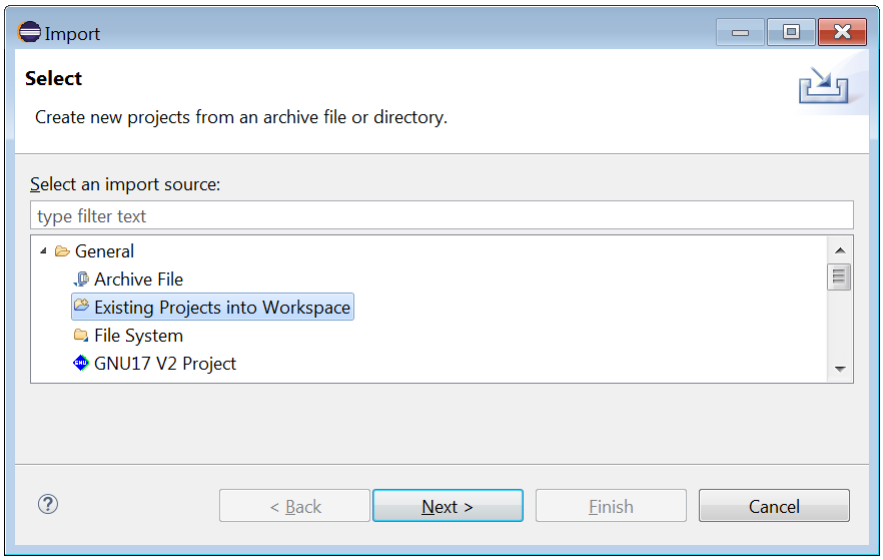
```
# Append the following three lines after the last line “c17 rst”.

u main
c17 stdout 1 WRITE_FLASH WRITE_BUF
c17 stdin 1 READ_FLASH READ_BUF
```

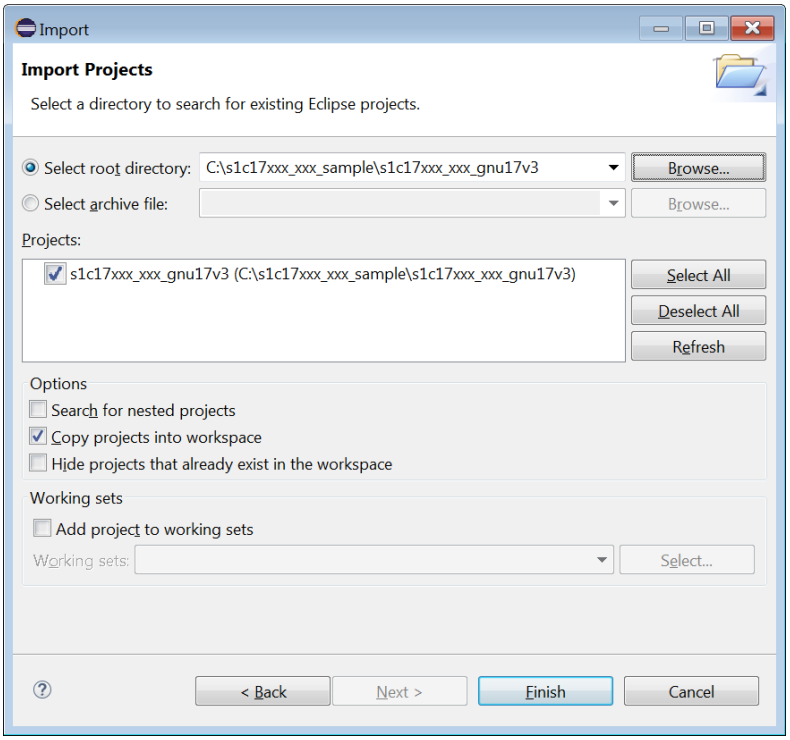
3.3.3 Importing Sample Software (GNU17 Version 3)

(1) Importing a sample software project

Select [Import...] from the [File] menu of GNU17 to start the [Import] wizard. Select [General] > [Existing Projects into Workspace] and then click the [Next >] button.



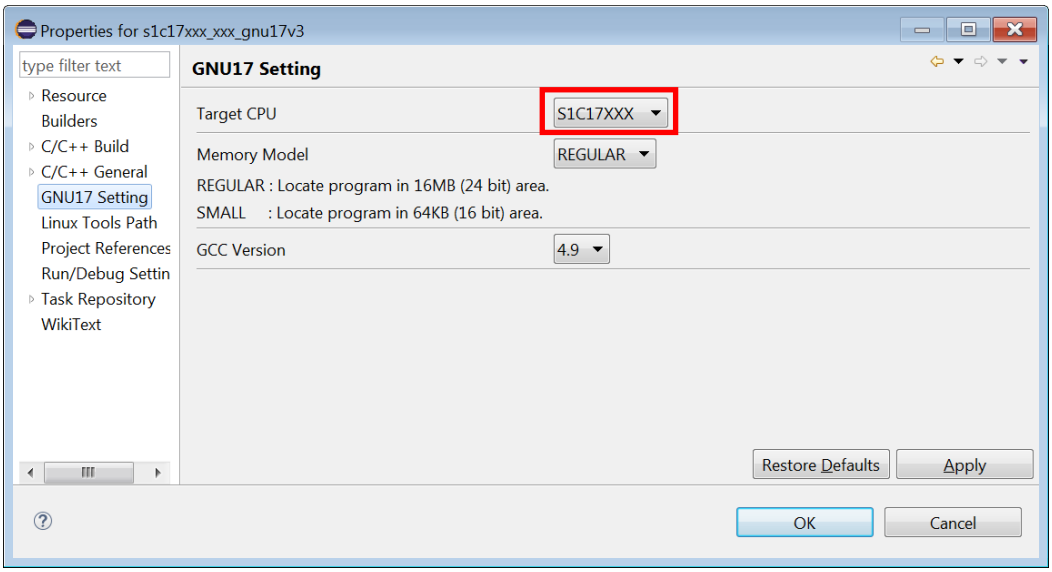
Click the [Browse...] button at [Select root directory:] and select the "xxx_gnu17v3" project that exists in the sample software folder extracted in Step (2) of Section 3.3.1.



Select the [Copy projects into workspace] checkbox. This enables copying of the project into the workspace directory so that the original files will not be modified. Finally, click the [Finish] button. After making sure that the sample software project has been imported to GNU17, the folder extracted in Step (2) of Section 3.3.1 can be removed.

(2) Selecting a target CPU

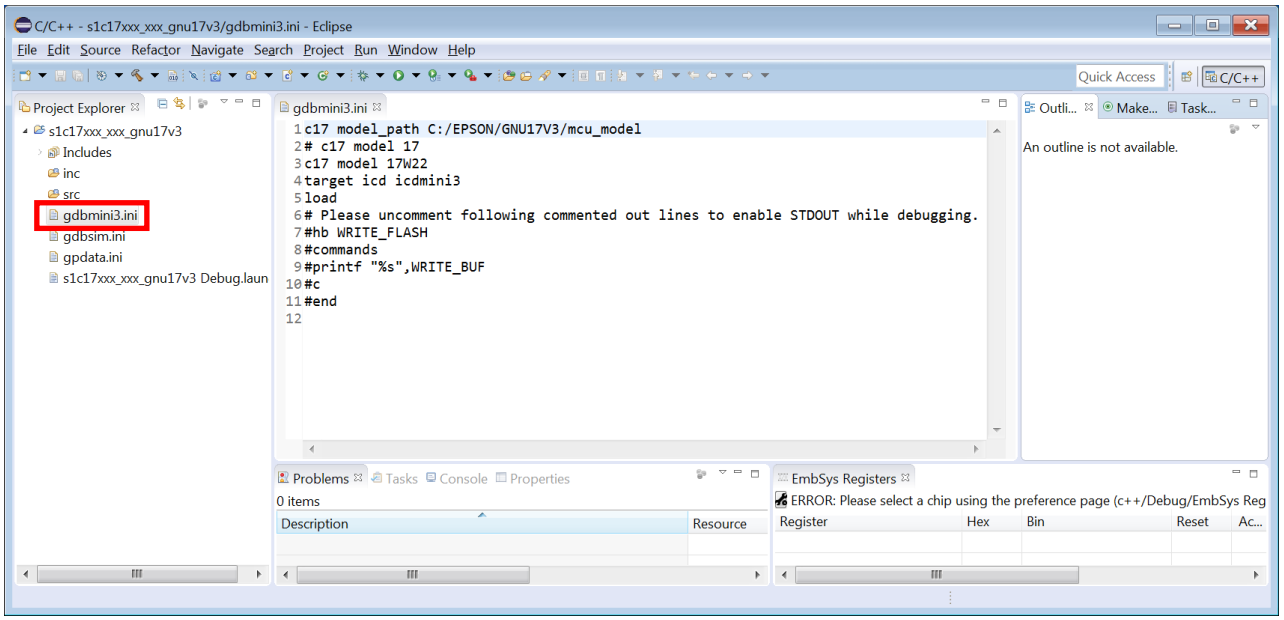
The target CPU to run the sample software should be selected. Select the sample software project in the [Project Explorer] view and select [Properties] from the [Project] menu to open the [Properties] dialog. Select [GNU17 Setting] from the property list in the dialog.



Select a target CPU from the [Target CPU] dropdown list, for example, select "S1C17W22" if it is the target CPU, and click the [Apply] button. Then click the [OK] button to close the dialog.

(3) Setting the debugger startup options

The debugger startup options should be selected. Expand the sample project in the [Project Explorer] view and open [gdbmini3.ini] with the editor.



The editor displays the contents of the debugger startup command file. The debugger must be set to the appropriate mode that suits the microcontroller and ICD used.

* Precautions on using the standard output

When using the standard output, uncomment (remove #) the command lines of the GDB command file as shown below (the lines highlighted in red) to enable them to be executed.

GDB command file [gdbmini3.ini]

```
# Please uncomment following commented out lines to enable STDOUT while debugging.
hb WRITE_FLASH
commands
printf "%s",WRITE_BUF
c
end
```

3.4 Connecting with Target

Connect a PC with GNU17 installed and ICDmini using a USB cable, and connect a target board with an S1C17 Series microcontroller mounted and the ICDmini.

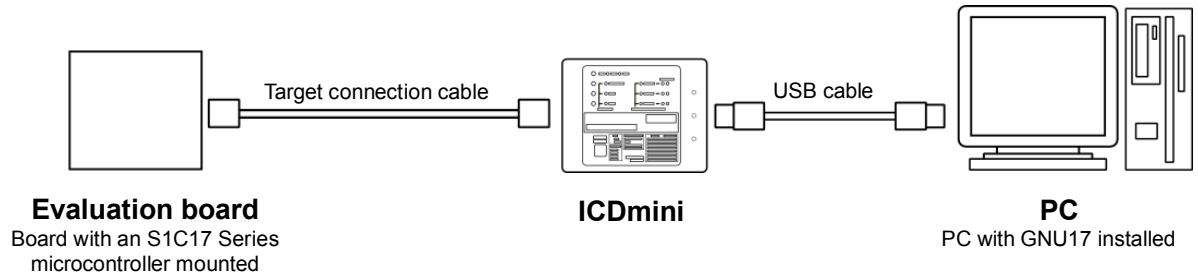


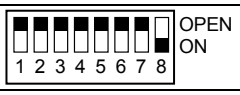
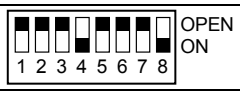
Figure 3.4.1 Connection Diagram Among Target, ICDmini, and PC

3.5 Operation of ICDmini and Target

3.5.1 When ICDmini Ver. 2.0 is Used

After connecting the system as shown in Section 3.4, reset the ICDmini and the target, and then confirm that the connection between them is established. The table below lists the ICDmini DIP switch settings.

Table 3.5.1.1 ICDmini DIP Switch Settings

Switch No.	Setting item	When S5U1C17M01T is connected	When S5U1C17W23T is connected
1	Target CPU (C17 / C33)	OPEN↑	OPEN↑
2	FLASH WRITER (OFF / WRITE)	OPEN↑	OPEN↑
3	FLASH WRITER (OFF / VERIFY)	OPEN↑	OPEN↑
4	DSIO LVL (ON / TRGET VCC)	OPEN↑	ON↓
5	DSIO LVL (3.3V / 1.8V)	OPEN↑	OPEN↑
6	CONNECTION DIAG (OFF / ON)	OPEN↑	OPEN↑
7	FW UPDATE MODE (OFF / ON)	OPEN↑	OPEN↑
8	FLASH VCC OUT (OFF / ON)	ON↓	ON↓
			

(1) Resetting ICDmini

Set the DIP switch on the ICDmini as the table above and then press the “RESET/START” button on the ICDmini.

(2) Resetting the target

Reset the target system. For how to reset the target system, refer to the manual provided for the evaluation board.

(3) Confirming the established connection between the target and ICDmini

Confirm that the ICDmini LEDs light up as follows:

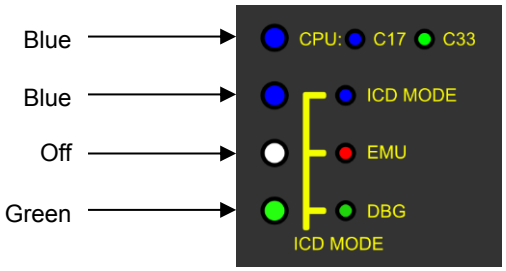


Figure 3.5.1.2 ICDmini Ver. 2.0 LED Lighting Status at Connection Establishment

3.5.2 When ICDmini Ver. 3.0 is Used

After connecting the system as shown in Section 3.4, confirm that the connection between the target and ICDmini is established by checking if the ICDmini LEDs light up as follows:

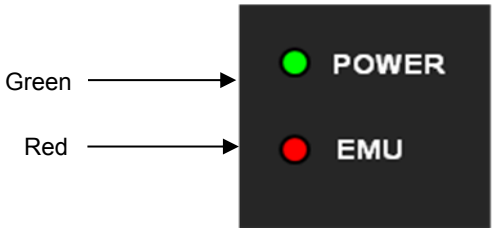


Figure 3.5.2.1 ICDmini Ver. 3.0 LED Lighting Status at Connection Establishment

3.6 Building Project and Starting Debugger

This section describes how to build the sample software project and start the debugger.

Notes:
- For more information on how to build a project and start the debugger, refer to the S5U1C17001C Manual.
- Be aware that the IC may not operate normally if you terminate the program that is controlling the clocks or the related functions. For debugging the codes concerned, refer to the DCLK change mode command described in the S5U1C17001C Manual.

3.6.1 When GNU17 Version 2 is Used

(1) Building the project

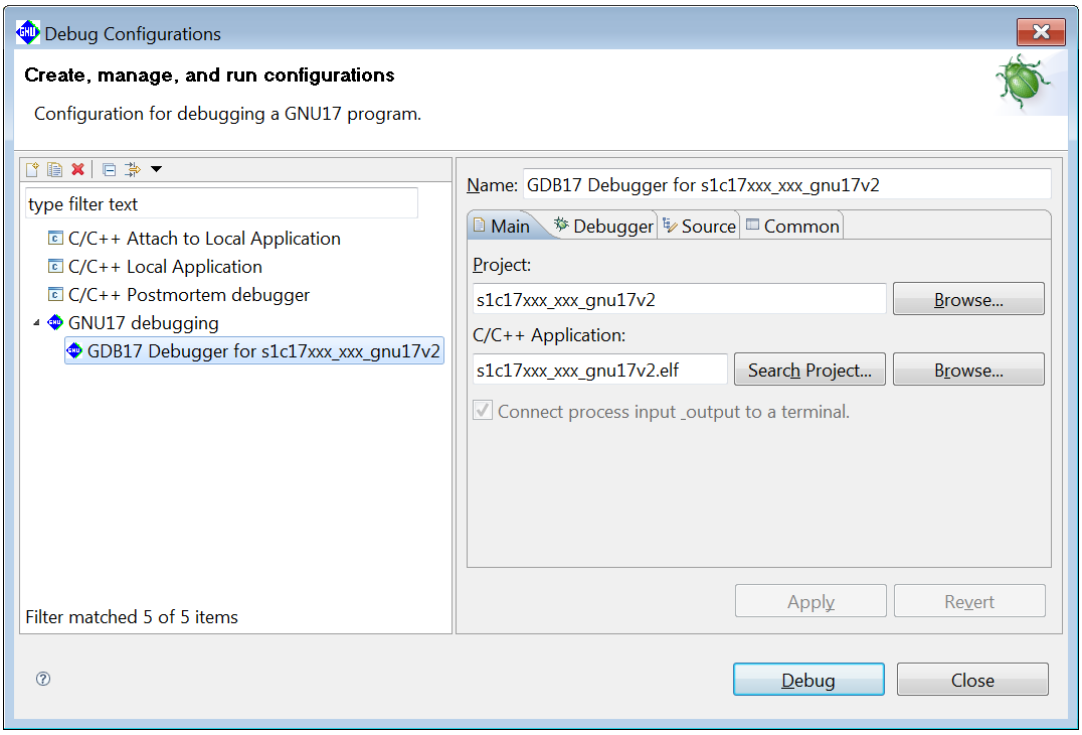
In the GNU17 [C/C++ project] view, select the project to be built. Select [Build Project] from the GNU17 [Project] menu (or the context menu that appears by right-clicking the project) to execute building.

(2) Displaying [Debug Configurations] dialog

Select [Debug Configurations...] from the GNU17 [Run] menu or the [Debug] pull-down menu on the toolbar to display the [Debug Configurations] dialog.

(3) Selecting the sample software project to be run (debugged)

Select [GDB17 Debugger for <project name>] from the list in the dialog.



(4) Starting the debugger

Click the [Debug] button. The debugger gdb starts up and executes the command file that has been specified.

3.6.2 When GNU17 Version 3 is Used

(1) Building the project

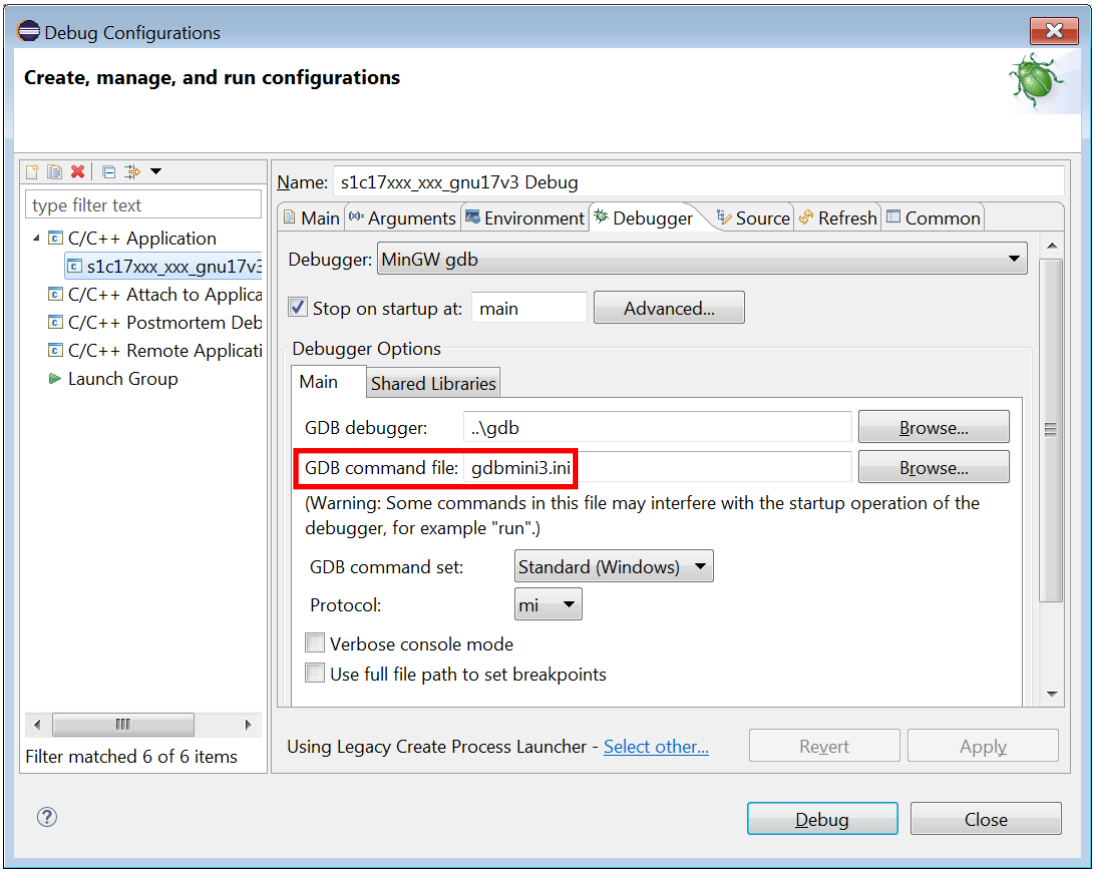
In the GNU17 [Project Explorer] view, select the project to be built. Select [Build Project] from the GNU17 [Project] menu (or the context menu that appears by right-clicking the project) to execute building.

(2) Displaying [Debug Configurations] dialog

Select [Debug Configurations...] from the GNU17 [Run] menu or the [Debug] pull-down menu on the toolbar to display the [Debug Configurations] dialog.

(3) Selecting the sample software project to be run (debugged) and configuring the debugger

Open the [Debugger] tab page and specify "gdbmini3.ini" for [GDB command file:]. ("gdbmini3.ini" has been specified by default in the sample project.)
Click the [Apply] button and then the [Debug] button to execute the specified command file.



4. Details of Sample Software Functions

This chapter gives a detailed description of the S1C17 Series sample software functions.

4.1 Models and Supported Functions

The tables below list the correspondence between the S1C17 Series models and the built-in peripheral circuits. Clicking "○" in the table jumps to the operation explanation page for the peripheral circuit sample software.

Table 4.1.1 Correspondence between S1C17W00 Series Model and Functions

Model name	Peripheral circuit																		
	CLG	PPORT	WDT	RTC	T16	T16B	SVD	UART	SPI	I2C	SNDA	REMC	SOFT REMC	LCD	RFC	PWG	ADC	TSRVR	OPCMP
S1C17W03	○	○	○	○	○	○	○	○	○	○	○	○	–	–	○	○	○	–	–
S1C17W04	○	○	○	○	○	○	○	○	○	○	○	○	–	–	○	○	○	–	–
S1C17W12	○	○	○	○	○	○	○	○	○	○	○	○	–	○	○	○	–	–	–
S1C17W13	○	○	○	○	○	○	○	○	○	○	○	○	–	○	○	○	–	–	–
S1C17W14	○	○	○	○	○	○	○	○	○	○	○	○	–	○	○	○	–	–	–
S1C17W15	○	○	○	○	○	○	○	○	○	○	○	–	○	○	○	○	–	–	–
S1C17W16	○	○	○	○	○	○	○	○	○	○	○	○	–	○	○	○	○	–	–
S1C17W18	○	○	○	○	○	○	○	○	○	○	○	○	–	○	○	○	○	○	–
S1C17W22	○	○	○	○	○	○	○	○	○	○	○	○	–	○	○	○	○	–	–
S1C17W23	○	○	○	○	○	○	○	○	○	○	○	○	–	○	○	○	○	–	○
S1C17W34	○	○	○	○	○	○	○	○	○	○	○	○	–	○	○	○	○	○	–
S1C17W35	○	○	○	○	○	○	○	○	○	○	○	○	–	○	○	○	○	○	–
S1C17W36	○	○	○	○	○	○	○	○	○	○	○	○	–	○	○	○	○	○	–

Table 4.1.2 Correspondence between S1C17M00 Series Model and Functions

Model name	Peripheral circuit																		
	CLG	PPORT	WDT	RTC	T16	T16B	SVD	UART	SPI	I2C	SNDA	REMC	SOFT REMC	LCD	RFC	PWG	ADC	TSRVR	OPCMP
S1C17M01	○	○	○	○	○	–	○	○	○	○	–	–	○	○	○	○	–	–	–
S1C17M10	○	○	○	○	○	○	○	○	○	○	–	–	○	○	–	○	–	–	–
S1C17M12	○	○	○	–	○	○	○	○	○	○	–	○	–	○	–	○	–	–	–
S1C17M13	○	○	○	–	○	○	○	○	○	○	–	○	–	○	–	○	○	–	–
S1C17M20	○	○	○	○	○	○	○	○	○	○	○	○	–	–	–	○	○	–	–
S1C17M21	○	○	○	○	○	○	○	○	○	○	○	○	–	–	–	○	○	–	–
S1C17M22	○	○	○	○	○	○	○	○	○	○	○	○	–	–	○	○	○	–	–
S1C17M23	○	○	○	○	○	○	○	○	○	○	○	○	–	–	–	○	○	–	–
S1C17M24	○	○	○	○	○	○	○	○	○	○	○	○	–	–	–	○	○	–	–
S1C17M25	○	○	○	○	○	○	○	○	○	○	○	○	–	–	○	○	○	–	–
S1C17M30	○	○	○	○	○	○	○	○	○	○	○	○	–	○	○	○	○	○	–
S1C17M31	○	○	○	○	○	○	○	○	○	○	○	○	–	○	○	○	○	○	–
S1C17M32	○	○	○	○	○	○	○	○	○	○	○	○	–	○	○	○	○	○	–
S1C17M33	○	○	○	○	○	○	○	○	○	○	○	○	–	○	○	○	○	○	–
S1C17M34	○	○	○	○	○	○	○	○	○	○	○	○	–	○	○	○	○	○	–

4.2 Initializing Peripheral Circuits

The sample software uses the macros defined in the "src\c17_init_config.h" file, which can be edited by the user, to initialize the peripheral circuits.

- A peripheral circuit is initialized when the c17_init_xxx() function is called after being configured using the macro.
- Edit "src\c17_init_config.h" to change the peripheral circuit configuration and the interrupt level.
- Uncomment (enable) the "#define C17_PE_xxx" for the peripheral circuits to be used before building the sample software.

4.2.1 "init_config.h" Configuration Examples

- **When executing the I/O port sample program**

  Edit the "c17_init_config.h" in the "src" folder under the folder where "s1c17xxx_sample" was extracted.

  (1) Enabling the I/O port module

  Uncomment the "#define C17_PE_PORT" to enable the I/O port sample program.

  Examples: • When enabling the I/O port sample program

  ```
  #define C17_PE_PORT
  ```

  • When disabling the I/O port sample program

  ```
  // #define C17_PE_PORT
  ```

  (2) Changing the port input interrupt level

  To change the port input interrupt level, rewrite the value of "#define INTERRUPT_LEVEL_PORT" written in the "src\c17_init_config.h" file.

  Example: When changing the interrupt level to 4

  ```
  #define INTERRUPT_LEVEL_PORT (4)        /// Set PORT interrupt level  0 to 7
  ```

  (3) Configuring the P00 port

  To configure the P00 port, edit the arguments (disable, input, chatta_disable, none) of "#define PORT00_INIT PORT_INIT" written in the "src\c17_init_config.h" file.

  First argument: Specifies whether to configure the P00 port or not. (enable or disable)
  Second argument: Specifies the P00 input/output direction. (input or output)
  Third argument: Enables or disables the P00 chattering filter function. (chatta_enable or chatta_disable)
  Fourth argument: Specifies the P00 pull-up/down function. (pullup, pulldown, or none)

  Example: When configuring the P00 port as I/O = output, chattering filter = disable, pull-up/down = none

  ```
  #define PORT00_INIT PORT_INIT(enable, output, chatta_disable, none)
  ```

4.2.2 Precautions on Editing “init_config.h”

(1) Precautions in the case of a sample software that supports two or more models

“init_config.h” contains definitions for configuring all the peripheral circuits of the default model (maximum configuration). However, other supported models may not have some peripheral circuits. Do not enable configuration of a peripheral circuit that is not included in the model to be used. For example, the “init_config.h” included in the “s1c17m01_w22_w23_sample_gnu17v2” sample software contains “#define C17_PE_LCD8A,” note, however that LCD8A cannot be used in the S1C17W23 even if the definition is enabled.

(2) Master and slave circuits/modes of the communication interfaces are exclusive functions.

Master and slave circuits/modes cannot be used simultaneously. If one of them is enabled, the other must be disabled.

4.3 I/O Ports (PPORT)

4.3.1 Sample Program Specifications

The I/O port sample program performs the following processing using I/O ports:

- Configures a port for input with interrupt enabled to detect that the input signal level changes to high or low.
- Configures a port for output and toggles the port output signal level high and low.

The table below shows the port configuration used in this sample program.

Table4.3.1.1 Port Configuration

Configuration	Port name
Output port	P01
Input port with interrupt enabled	P02

When executing this sample program, set T16 (#define C17_PE_T16) and PORT (#define C17_PE_PORT) to be active.

4.3.2 Hardware Conditions

To run this sample program, the ports should be connected as shown in the figure below.

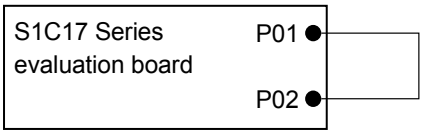


Figure 4.3.2.1 Hardware Connection Diagram for I/O Port Sample Program

4.3.3 Operations Overview

1. The sample program initializes the I/O ports.
2. Configures P01 to an output port and sets the output level to low.
3. Configures P02 to an input port that generates an interrupt when the input signal level changes from low to high.
4. Sets the P01 output level to high.
5. When a P02 port interrupt occurs, the sample program checks if the P02 port input level is high.
6. Configures P02 to generate an interrupt when the input signal level changes from high to low.
7. Sets the P01 output level to low.
8. When a P02 port interrupt occurs, the sample program checks if the P02 port input level is low.

4.4 Clock Generator (CLG)

4.4.1 Sample Program Specifications

The clock generator sample program performs the following processing using the clock generator:

- Initiates the OSC1 oscillation (only for the model with the OSC1 oscillator).
- Initiates the OSC3 oscillation (only for the model with the OSC3 oscillator). The oscillation frequency may be specified.
- Initiates the IOSC oscillation (only for the model with the IOSC oscillator). The oscillation frequency may be specified.
- Switches the system clock to OSC1 (only for the model with the OSC1 oscillator).
- Switches the system clock to OSC3 (only for the model with the OSC3 oscillator).
- Switches the system clock to IOSC (only for the model with the IOSC oscillator).
- Outputs the system clock from the FOUT pin (the clock switching status is reflected on the FOUT output).
- Performs transition of operating mode to HALT and SLEEP modes.

When executing this sample program, set T16 (#define C17_PE_T16) to be active.

4.4.2 Hardware Conditions

This sample program starts running with an internal oscillator clock. After that it switches the system clock to OSC1, OSC3, and IOSC sequentially. Therefore, external resonators may be required for their oscillation depending on the model. For detailed information on the oscillator circuits, refer to "Clock Generator (CLG)" in the technical manual of each S1C17 Series model.

Table 4.4.2.1 List of Ports used for FOUT Output

Model name	FOUT port
S1C17M01	P32
S1C17M10	P04
S1C17M12/M13	P16
S1C17M20/M21/M22/M23/M24/M25	P13
S1C17M30/M31/M32/M33/M34	P10
S1C17W03/W04	P06
S1C17W12/W13	P10
S1C17W14/W16	P34
S1C17W15	P16
S1C17W18	P16
S1C17W22/W23	P15
S1C17W34/W35/W36	P12

4.4.3 Operations Overview

1. The sample program initiates an oscillator available in the target model and configures a 16-bit timer to generate an interrupt in one-second intervals.
2. Output the current system clock as FOUT for monitoring.
3. Suspends the CPU operation (puts the CPU into HALT mode) until the 16-bit timer generates an interrupt. Repeats this operation twice.
4. Suspends the CPU operation (puts the CPU into SLEEP mode) until the 16-bit timer generates an interrupt. Repeats this operation twice.
5. Repeats Steps 1 to 4 using IOSC, OSC1, and OSC3 (except for the oscillator that does not exist in the target model).
6. Stops the FOUT output. After that the sample program stops running.

4.5 16-bit Timer (T16)

4.5.1 Sample Program Specifications

The 16-bit timer sample program performs the following processing using a 16-bit timer:

- Configures the 16-bit timer to generate interrupts to obtain the counter values.
- Puts the CPU into HALT mode to reduce power consumption when waiting for an interrupt.

When executing this sample program, set T16 (#define C17_PE_T16) to be active.

4.5.2 Hardware Conditions

This sample program runs using an internal oscillator.

4.5.3 Operations Overview

1. The sample program configures a 16-bit timer to generate an interrupt in one-second intervals.
2. Starts the 16-bit timer and puts the CPU into HALT mode.
3. When a 16-bit timer interrupt occurs, the CPU exits from HALT mode.
4. The sample program counts the number of 16-bit timer interrupts that have occurred.
5. If the interrupt count is less than 10, the sample program puts the CPU into HALT mode again. (To Step 3)
6. When the interrupt count reaches 10, the sample program stops the 16-bit timer and stops running.

4.6 16-bit PWM Timer (T16B)

4.6.1 Sample Program Specifications

The 16-bit PWM timer sample program performs the following processing using a 16-bit PWM timer:

- Configures a 16-bit PWM timer to generate compare interrupts.
- Configures the 16-bit PWM timer to generate capture interrupts for the timer counter values to be obtained.
- Puts the CPU into HALT mode to reduce power consumption when waiting for an interrupt.
- Outputs the PWM waveforms generated by the 16-bit PWM timer from the TOUT pin.

When executing this sample program, set T16 (#define C17_PE_T16) and T16B (#define C17_PE_T16B) to be active.

4.6.2 Hardware Conditions

This sample program runs using an internal oscillator.

Table 4.6.2.1 Port used for TOUT Output

Model name	TOUT port
S1C17 Series	P00

4.6.3 Operations Overview

1. The sample program configures a 16-bit PWM timer as follows and then starts it.
   - Mode: Repeat up count mode
   - MAX counter value: 0x5000
   - Comparator/capture circuit 0: Comparator mode (compare buffer: 0x2000)
   - Comparator/capture circuit 1: Capture mode (trigger: falling edge)
   - TOUT output: Enabled
2. Starts the 16-bit PWM timer and puts the CPU into HALT mode.
3. When a 16-bit PWM timer interrupt occurs, the CPU exits from HALT mode.
4. The sample program counts the number of the following 16-bit timer interrupts that have occurred.
   - Number of compare interrupts
   - Number of capture interrupts
   - Number of counter MAX interrupts
   - Number of counter zero interrupts
5. When a compare interrupt occurs, the sample program sets the capture trigger condition to rising edge.
6. When a counter MAX interrupt occurs, the sample program sets the capture trigger condition to falling edge.
7. If the counter MAX interrupt count is less than 10, the sample program puts the CPU into HALT mode again. (To Step 3)
8. When the counter MAX interrupt count reaches 10, the sample program stops the 16-bit PWM timer, saves the interrupt count values, and then stops running.

4.7 Real-Time Clock (RTCA, RTCA2)

4.7.1 Sample Program Specifications

The real-time clock sample program performs the following processing using the real-time clock:

- Sets a time to the real-time clock.
- Obtains the time from the real-time clock.
- Obtains the number of real-time clock interrupts.
- Activates the theoretical regulation function.
- Activates the stopwatch.

When executing this sample program, set T16 (#define C17_PE_T16) and RTCA (#define C17_PE_RTCA) to be active.

4.7.2 Hardware Conditions

This sample program runs using the OSC1 oscillator (32.768 kHz).

4.7.3 Operations Overview

1. The sample program initializes the real-time clock.
2. Enables one- and half-second interrupts of the real-time clock.
3. Configures the theoretical regulation function of the real-time clock (theoretical regulation execution cycle time: 4,096 seconds, correction rate: -58.2 ppm).
4. Sets the real-time clock to 24-hour mode and the date and time to “Saturday, December 31, 2016, 23:59:51.”
5. Starts the real-time clock and puts the CPU into HALT mode.
6. When a real-time clock interrupt occurs, the CPU exits from HALT mode.
7. The sample program counts the number of real-time clock one-second interrupts that have occurred.
8. If the one-second interrupt count is less than 10, the sample program puts the CPU into HALT mode again. (To Step 6)
9. When the one-second interrupt count reaches 10, the sample program sets the real-time clock to 12-hour mode and reads the date and time from the real-time clock (the read data is “Sunday, January 1, 2017, 12:00:01 a.m.”).
10. The sample program disables one-second interrupts of the real-time clock.
11. Executes theoretical regulation at 4,096-second intervals after the real-time clock starts.
12. Initializes the stopwatch.
13. Enables stopwatch interrupts.
14. Starts the stopwatch and puts the CPU into HALT mode.
15. When a stopwatch interrupt occurs, the CPU exits from HALT mode.
16. The sample program counts the number of stopwatch 1-Hz interrupts that have occurred.
17. If the stopwatch 1-Hz interrupt count is less than 10, the sample program puts the CPU into HALT mode again. (To Step 15)
18. When the stopwatch 1-Hz interrupt count reaches 10, the sample program reads the stopwatch counter (10-Hz and 100-Hz digits).
19. The sample program stops the stopwatch and disables stopwatch interrupts.

4.8 Watchdog Timer (WDT, WDT2)

4.8.1 Sample Program Specifications

The watchdog timer sample program performs the following processing using the watchdog timer:

- Resets the watchdog timer and confirms the behavior.
- Causes the watchdog timer to issue a reset.

When executing this sample program, set T16 (#define C17_PE_T16) and WDT (#define C17_PE_WDT) to be active.

4.8.2 Hardware Conditions

This sample program runs using the OSC1 oscillator (32.768 kHz) or an internal oscillator.

4.8.3 Operations Overview

1. The sample program starts the watchdog timer and a 16-bit timer.
2. When a 16-bit timer interrupt occurs (one-second intervals), the sample program resets the watchdog timer.
3. When the 16-bit timer (one-second) interrupt count reaches 10, the sample program stops the 16-bit timer.
4. The watchdog timer issues a reset and the processing jumps to the top of the sample program (boot.c).

4.9 UART (UART, UART2, UART3)

4.9.1 Sample Program Specifications

The UART sample program performs the following processing using the UART:

- Sends data via the UART.
- Receives data via the UART.

When executing this sample program, set T16 (#define C17_PE_T16) and UART (#define C17_PE_UART_MASTER or #define C17_PE_UART_SLAVE) to be active.

4.9.2 Hardware Conditions

This sample program runs using an internal oscillator.
To run the sample program, connect two evaluation boards, one that has loaded the UART master sample program and another that has loaded the UART slave sample program, as in the figure below. Note that the evaluation boards must be turned on from the slave first.

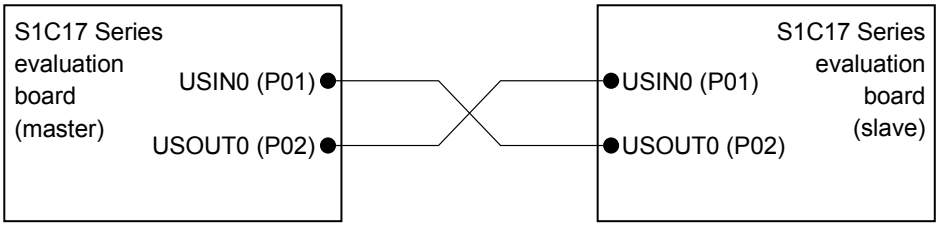


Figure 4.9.2.1 Hardware Connection Diagram for UART Sample Program (default port assignment)

4.9.3 Operations Overview

4.9.3.1 Master Sample Program Operations Overview

1. The sample program initializes the transfer data format as follows:
   - Data length: 8 bits
   - Stop bit: 1 bit
   - Parity: None
2. Sends data from 'A' (0x41) to 'Z' (0x5a) and '\0' to the slave.
3. Receives a 27-byte data from the slave.
4. Verifies if the received data is the same as the data sent and then stops running.

4.9.3.2 Slave Sample Program Operations Overview

1. The sample program initializes the transfer data format as follows:
   - Data length: 8 bits
   - Stop bit: 1 bit
   - Parity: None
2. Receives a 27-byte data from the master.
3. Sends data from 'A' (0x41) to 'Z' (0x5a) and '\0' to the master.
4. Verifies if the received data is the same as the data sent and then stops running.

4.9.4 Calibration Function

When using IOC as the UART operating clock, the sample program allows use of the calibration function using the 16-bit timer Ch.0.
Configure the 16-bit timer operating clock to OSC1 or OSC3.
When using OSC3 as the 16-bit timer operating clock, determine the clock division ratio so that the operating clock frequency will be set as close as possible to 32 kHz.

1. Edit the following definitions in "c17_init_config.h":
   - #define IOSC_CALIBRATION_ENABLE : enable = 1
   - #define IOSC_CALIBRATION_CLOCK : OSC1 or OSC3 frequency [Hz]
   - #define T16CH0_INIT T16_INIT : enable = 1
     : OSC1 or OSC3
     : OSC1 = CLKDIV_1, OSC3 = value close to 32 kHz
     : mode = oneshot
2. Edit the following functions in "c17_uartX.c":
   - baudrateCalibration function Start oscillation : OSC1 = c17startClgOSC1();
     : OSC3 = c17startClgExtOSC3();
   - baudrateCalibration function Stop oscillation : OSC1 = c17stopClgOSC1();
     : OSC3 = c17stopClgOSC3();

4.10 SPI (SPIA)

4.10.1 Sample Program Specifications

The SPI sample program performs the following processing using the SPIA:

- Sends data via the SPIA.
- Receives data via the SPIA.

When executing this sample program, set T16 (#define C17_PE_T16) and SPIA (#define C17_PE_SPIA_MASTER or #define C17_PE_SPIA_SLAVE) to be active.

4.10.2 Hardware Conditions

This sample program runs using an internal oscillator.
To run the sample program, connect two evaluation boards, one that has loaded the SPI master sample program and another that has loaded the SPI slave sample program, as in the figure below. Note that the evaluation boards must be turned on from the slave first.

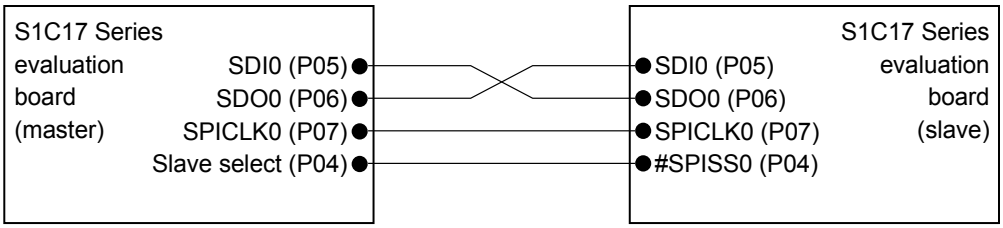


Figure 4.10.2.1 Hardware Connection Diagram for SPI Master and Slave Sample Program (default port assignment)

Table 4.10.2.1 Exceptional Model Having Other Port Assignment

Model name	SDI0 port	SDO0 port	SPICLK0 port	Slave select or #SPISS0 port
S1C17W15	P15	P16	P17	P14
S1C17M20/M21/M22/M23/M24/M25	P01	P02	P03	P00

4.10.3 Operations Overview

4.10.3.1 Master Sample Program Operations Overview

1. The sample program initializes the transfer data format as follows:
   - Data length: 8 bits
   - Data format: MSB first
2. Sends data from 'A' (0x41) to 'Z' (0x5a) and '\0' to the slave.
3. Receives a 27-byte data from the slave.
4. Verifies if the received data is the same as the data sent and then stops running.

4.10.3.2 Slave Sample Program Operations Overview

1. The sample program initializes the transfer data format as follows:
   - Data length: 8 bits
   - Data format: MSB first
2. Receives a 27-byte data from the master.
3. Sends data from 'A' (0x41) to 'Z' (0x5a) and '\0' to the master.
4. Verifies if the received data is the same as the data sent and then stops running.

4.11 I²C (I2C)

4.11.1 Sample Program Specifications

The I²C sample program performs the following processing using the I2C:

- Sends data via the I2C.
- Receives data via the I2C.

When executing this sample program, set T16 (#define C17_PE_T16) and I2C (#define C17_PE_I2C_MASTER or #define C17_PE_I2C_SLAVE) to be active.

4.11.2 Hardware Conditions

This sample program runs using the IOSC oscillator.
To run the sample program, connect two evaluation boards, one that has loaded the I²C master sample program and another that has loaded the I²C slave sample program, as in the figure below. Use a common GND line for both evaluation boards. Note that the evaluation boards must be turned on from the slave first. Also the sample program must be run from the slave first when starting debugging.

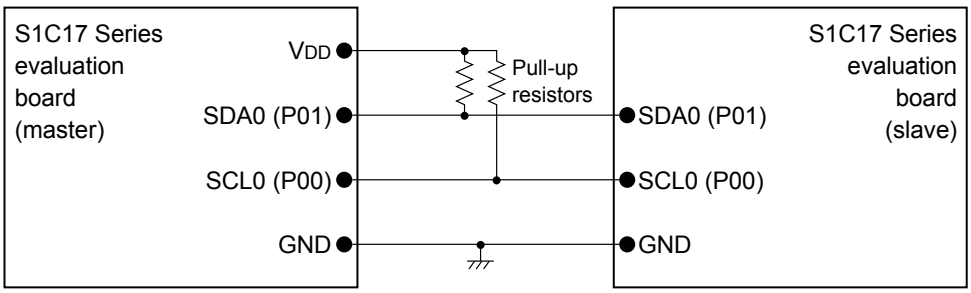


Figure 4.11.2.1 Hardware Connection Diagram for I²C Master and Slave Sample Program (default port assignment)

Table 4.11.2.1 Exceptional Models Having Other Port Assignment

Model name	SCL0 port	SDA0 port
S1C17M01	P03	P04
S1C17W22/W23	P03	P04

4.11.3 Operations Overview

4.11.3.1 Master Sample Program Operations Overview

1. The sample program initializes the I2C.
2. Sends data from 'A' (0x41) to 'Z' (0x5a) and '\0' to the slave.
3. Receives a 27-byte data from the slave.
4. Verifies if the received data is the same as the data sent and then stops running.

4.11.3.2 Slave Sample Program Operations Overview

1. The sample program configures the I2C as follows:
   - Address mode: 7 bits
   - Slave address: 0x2a
   - General call: Disabled
2. Receives a 27-byte data from the master.
3. Sends data from 'A' (0x41) to 'Z' (0x5a) and '\0' to the master.
4. Verifies if the received data is the same as the data sent and then stops running.

4.12 LCD Driver (LCD4A)

4.12.1 Sample Program Specifications

This LCD driver sample program is intended for use with a model that includes LCD4A. It performs the following processing using the LCD4A:

- Turns all dots (segments) on and off in normal display mode.
- Sets the LCD driver into all on mode and all off mode to turn all dots (segments) on and off.

When executing this sample program, set T16 (#define C17_PE_T16), RTCA (#define C17_PE_RTCA), and LCD4A (#define C17_PE_LCD4A) to be active.

4.12.2 Hardware Conditions

This sample program requires a target model equipped with LCD4A to run. For connection with an LCD panel and other information on the LCD driver, refer to the technical manual of each S1C17 Series model.
The sample program runs using the OSC1 oscillator (32.768 kHz).

4.12.3 Operations Overview

1. The sample program turns all the dots (segments) on for two seconds.
2. Turns all the dots (segments) off for two seconds.
3. Sets a checkered pattern in display area 0.
4. Selects display area 0 and performs normal display for two seconds.
5. Performs inverted display for two seconds.
6. Turns the LCD off and stops running.

4.13 LCD Driver (LCD8A)

4.13.1 Sample Program Specifications

This LCD driver sample program is intended for use with a model that includes LCD8A. It performs the following processing using the LCD8A:

- Turns all dots (segments) on and off in normal display mode.
- Sets the LCD driver into all on mode and all off mode to turn all dots (segments) on and off.

When executing this sample program, set T16 (#define C17_PE_T16), RTCA (#define C17_PE_RTCA), and LCD8A (#define C17_PE_LCD8A) to be active.

4.13.2 Hardware Conditions

This sample program requires a target model equipped with LCD8A to run. For connection with an LCD panel and other information on the LCD driver, refer to the technical manual of each S1C17 Series model.
The sample program runs using the OSC1 oscillator (32.768 kHz).

4.13.3 Operations Overview

Steps 3 to 18 are performed only when the dedicated LCD panel for the S5U1C17M01T1 board is connected.

1. The sample program turns all the dots (segments) on for two seconds.
2. Turns all the dots (segments) off for two seconds.
3. Displays “00000000” on the segment display for two seconds.
4. Displays “12345678” on the segment display for two seconds.
5. Turns the “battery warning” indicator on for two seconds.
6. Turns the “overflow warning” indicator on for two seconds.
7. Turns the “water-leak warning (up)” indicator on for two seconds.
8. Turns the “water-leak warning (down)” indicator on for two seconds.
9. Turns the “water-leak warning (up-down)” indicator on for two seconds.
10. Turns the “reverse flow warning (inside)” indicator on for two seconds.
11. Turns the “reverse flow warning (outside)” indicator on for two seconds.
12. Turns the “reverse flow warning (inside-outside)” indicator on for two seconds.
13. Turns the “unit ("m3/h")” indicator on for two seconds.
14. Turns the “communication” indicator on for two seconds.
15. Turns the “pilot” indicator on for two seconds.
16. Turn the segments off for two seconds.
17. Displays “Error” on the segment display for two seconds.
18. Displays “87654321” on the segment display for two seconds.
19. Turns the LCD off and stops running.

4.14 LCD Driver (LCD8B)

4.14.1 Sample Program Specifications

This LCD driver sample program is intended for use with a model that includes LCD8B. It performs the following processing using the LCD8B:

- Turns all dots (segments) on and off in normal display mode.
- Sets the LCD driver into all on mode and all off mode to turn all dots (segments) on and off.

When executing this sample program, set T16 (#define C17_PE_T16), RTCA (#define C17_PE_RTCA), and LCD8B (#define C17_PE_LCD8B) to be active.

4.14.2 Hardware Conditions

This sample program requires a target model equipped with LCD8B to run. For connection with an LCD panel and other information on the LCD driver, refer to the technical manual of each S1C17 Series model.
The sample program runs using the OSC1 oscillator (32.768 kHz).

4.14.3 Operations Overview

1. The sample program turns all the dots (segments) on for two seconds.
2. Turns all the dots (segments) off for two seconds.
3. Sets a checkered pattern in display area 0.
4. Selects display area 0 and performs normal display for two seconds.
5. Performs inverted display for two seconds.
6. Turns the LCD off and stops running.

4.15 LCD Driver (LCD16A)

4.15.1 Sample Program Specifications

This LCD driver sample program is intended for use with a model that includes LCD16A. It performs the following processing using the LCD16A:

- Turns all dots (segments) on and off in normal display mode.
- Sets the LCD driver into all on mode and all off mode to turn all dots (segments) on and off.

When executing this sample program, set T16 (#define C17_PE_T16), RTCA (#define C17_PE_RTCA), and LCD16A (#define C17_PE_LCD16A) to be active.

4.15.2 Hardware Conditions

This sample program requires a target model equipped with LCD16A to run. For connection with an LCD panel and other information on the LCD driver, refer to the technical manual of each S1C17 Series model.
The sample program runs using the OSC1 oscillator (32.768 kHz).

4.15.3 Operations Overview

1. The sample program turns all the dots (segments) on for two seconds.
2. Turns all the dots (segments) off for two seconds.
3. Sets a checkered pattern in display area 0.
4. Selects display area 0 and performs normal display for two seconds.
5. Performs inverted display for two seconds.
6. Turns the LCD off and stops running.

4.16 LCD Driver (LCD24A)

4.16.1 Sample Program Specifications

This LCD driver sample program is intended for use with a model that includes LCD24A. It performs the following processing using the LCD24A:

- Turns all dots (segments) on and off in normal display mode.
- Sets the LCD driver into all on mode and all off mode to turn all dots (segments) on and off.

When executing this sample program, set T16 (#define C17_PE_T16), RTCA (#define C17_PE_RTCA), and LCD24A (#define C17_PE_LCD24A) to be active.

4.16.2 Hardware Conditions

This sample program requires a target model equipped with LCD24A to run. For connection with an LCD panel and other information on the LCD driver, refer to the technical manual of each S1C17 Series model.
The sample program runs using the OSC1 oscillator (32.768 kHz).

4.16.3 Operations Overview

1. The sample program turns all the dots (segments) on for two seconds.
2. Turns all the dots (segments) off for two seconds.
3. Sets a checkered pattern (small) in display area 0.
4. Selects display area 0 and performs normal display for two seconds.
5. Performs inverted display for two seconds.
6. Sets a checkered pattern (medium) in display area 1.
7. Selects display area 1 and performs normal display for two seconds.
8. Performs inverted display for two seconds.
9. Sets a checkered pattern (large) in display area 0.
10. Selects display area 0 and performs normal display for two seconds.
11. Performs inverted display for two seconds.
12. Turns the LCD off and stops running.

4.17 LCD Driver (LCD32B)

4.17.1 Sample Program Specifications

This LCD driver sample program is intended for use with a model that includes LCD32B. It performs the following processing using the LCD32B:

- Turns all dots (segments) on and off in normal display mode.
- Sets the LCD driver into all on mode and all off mode to turn all dots (segments) on and off.

When executing this sample program, set T16 (#define C17_PE_T16), RTCA (#define C17_PE_RTCA), and LCD32B (#define C17_PE_LCD32B) to be active.

4.17.2 Hardware Conditions

This sample program requires a target model equipped with LCD32B to run. For connection with an LCD panel and other information on the LCD driver, refer to the technical manual of each S1C17 Series model.
The sample program runs using the OSC1 oscillator (32.768 kHz).

4.17.3 Operations Overview

1. The sample program turns all the dots (segments) on for two seconds.
2. Turns all the dots (segments) off for two seconds.
3. Sets a checkered pattern (small) in display area 0.
4. Selects display area 0 and performs normal display for two seconds.
5. Performs inverted display for two seconds.
6. Sets a checkered pattern (medium) in display area 1.
7. Selects display area 1 and performs normal display for two seconds.
8. Performs inverted display for two seconds.
9. Sets a checkered pattern (large) in display area 0.
10. Selects display area 0 and performs normal display for two seconds.
11. Performs inverted display for two seconds.
12. Turns the LCD off and stops running.

4.18 Seven-Segment LED Controller (LEDC)

4.18.1 Sample Program Specifications

The seven-segment LED controller sample program performs the following processing using the seven-segment LED controller:

- Controls the display on the seven-segment LED modules connected.

When executing this sample program, set T16 (#define C17_PE_T16), RTCA (#define C17_PE_RTCA), and LEDC (#define C17_PE_LEDC) to be active.

4.18.2 Hardware Conditions

This sample program requires a target model equipped with LEDC to run. For connection with seven-segment LED modules and other information on the seven-segment LED controller, refer to the technical manual of each S1C17 Series model.
The sample program runs using the OSC3 oscillator.

4.18.3 Operations Overview

1. The sample program displays '0' on the seven-segment LED module connected to COM0.
2. Displays '1' on the seven-segment LED module connected to COM1.
3. Displays '2' on the seven-segment LED module connected to COM2.
4. Displays '3' on the seven-segment LED module connected to COM3.
5. Displays '4' on the seven-segment LED module connected to COM4.
6. Stops the LEDC and stops running.

4.19 R/F Converter (RFC)

4.19.1 Sample Program Specifications

The R/F converter sample program performs the following processing using the R/F converter:

- Convert the resistance value of a sensor into a digital value from the oscillation clock count value obtained by CR oscillating the sensor.

When executing this sample program, set T16 (#define C17_PE_T16) and RFC (#define C17_PE_RFC) to be active.

4.19.2 Hardware Conditions

This sample program runs using the OSC1 oscillator (32.768 kHz). To run the sample program, external components must be connected as in the figure below. The SENB0 pin should be left open.

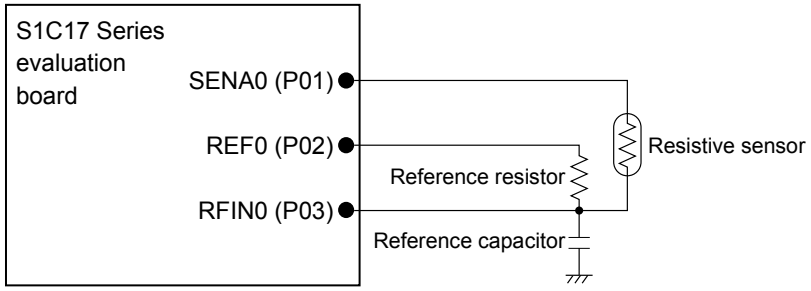


Figure 4.19.2.1 Hardware Connection Diagram for R/F Converter Sample Program (default port assignment)

Table 4.19.2.1 Exceptional Model Having Other Port Assignment

Model name	SENA0 port	REF0 port	RFIN0 port
S1C17M01	P04	P05	P06
S1C17M22/M25	P34	P35	P36
S1C17W03/W04	P16	P15	P14
S1C17W12/W13	P05	P06	P07

4.19.3 Operations Overview

1. The sample program sets the clock source and oscillation mode of the RFC.
2. Sets 0x00fc0000 to the measurement counter as the initial value and initiates measurement using the RFC (measurement time: about eight seconds).
3. Obtains the measured value after reference oscillation and sensor oscillation have completed. Then the sample program stops running.

4.20 Sound Generator (SNDA)

4.20.1 Sample Program Specifications

The sound generator sample program performs the following processing using the sound generator:

- Sounds a buzzer in normal buzzer mode.
- Sounds a buzzer in one-shot buzzer mode.
- Sounds a buzzer in melody mode.

When executing this sample program, set T16 (#define C17_PE_T16), RTCA (#define C17_PE_RTCA), and SNDA (#define C17_PE_SNDA) to be active.

4.20.2 Hardware Conditions

This sample program runs using the OSC1 oscillator (32.768 kHz) or an internal oscillator. Before using this sample program, connect a buzzer to the evaluation board as follows:

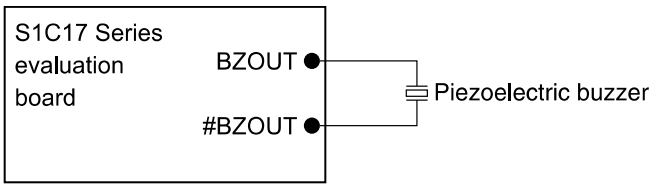


Figure 4.20.2.1 Hardware Connection Diagram for Sound Generator Sample Program

Table 4.20.2.1 List of Ports used for Buzzer Output

Model name	BZOUT port	#BZOUT port
S1C17M20/M21/M22/M23/M24/M25	P02	P03
S1C17M30/M31/M32/M33/M34	P33	P34
S1C17W03/W04	P04	P05
S1C17W12/W13	P02	P03
S1C17W14/W16	P06	P07
S1C17W15	P15	P14
S1C17W18	P15	P14
S1C17W22/W23	P14	P13
S1C17W34/W35/W36	P23	P22

4.20.3 Operations Overview

1. The sample program sounds the buzzer in normal buzzer mode.
2. Sounds the buzzer in one-shot buzzer mode.
3. Sounds the buzzer in melody mode.

4.21 Supply Voltage Detector (SVD, SVD3)

4.21.1 Sample Program Specifications

The supply voltage detector sample program performs the following processing using the supply voltage detector:

- Compares the power supply voltage with the SVD detection voltage set in the supply voltage detector.

When executing this sample program, set T16 (#define C17_PE_T16) and SVD (#define C17_PE_SVD) to be active.

4.21.2 Hardware Conditions

This sample program runs using an internal oscillator. To evaluate the SVD function using the sample program, a stabilized (variable) power supply should be connected as in the figure below.

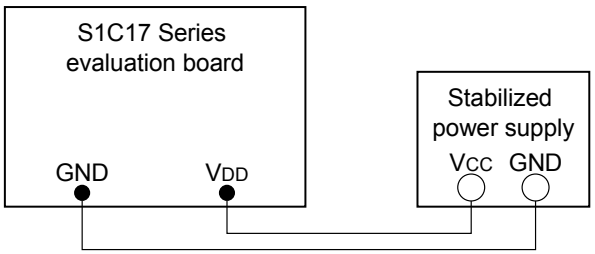


Figure 4.21.2.1 Hardware Connection Diagram for Supply Voltage Detector Sample Program

4.21.3 Operations Overview

1. The sample program configures a 16-bit timer to generate an interrupt in one-second intervals and starts it.
2. Configures the supply voltage detector to detect if the power supply voltage drops less than 2.7 V, and starts it.
3. When a 16-bit timer interrupt occurs, the sample program checks whether the power supply voltage drops or not.
4. The sample program performs this voltage drop detection for 10 seconds and then stops running.

4.22 Power Generator (PWG, PWG2)

4.22.1 Sample Program Specifications

The power generator sample program performs the following processing using the power generator:

- Switches the power generator operation mode (between normal mode and economy mode for power saving).

When executing this sample program, set T16 (#define C17_PE_T16) and PWG (#define C17_PE_PWG) to be active.

4.22.2 Hardware Conditions

This sample program runs using the OSC1 oscillator (32.768 kHz) or an internal oscillator.

4.22.3 Operations Overview

1. The sample program initiates the OSC1 oscillator.
2. Sets the power generator operating mode.

4.23 IR Remote Controller (REMC, REMC2, REMC3)

4.23.1 Sample Program Specifications

The IR remote controller sample program performs the following processing using the IR remote controller:

- Generates a data signal in NEC format.
- Generates a data signal in AEHA format.
- Generates a data signal in SONY format.

When executing this sample program, set T16 (#define C17_PE_T16) and REMC (#define C17_PE_REMC) to be active.

4.23.2 Hardware Conditions

This sample program runs using an internal oscillator.
For information on connection with the circuit example shown in Figure 4.23.2.1, refer to the SVT17 Series manual.

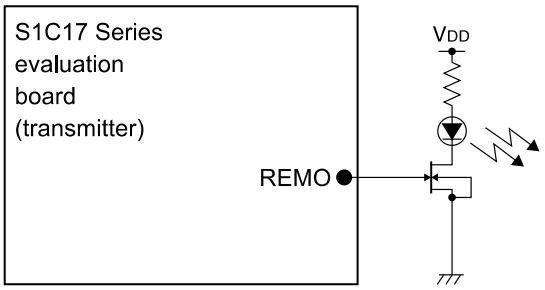


Figure 4.23.2.1 Infrared Light Emitting Diode Connection Circuit Example for IR Remote Controller Sample Program

Table 4.23.2.1 List of Ports used for REMO Output

Model name	REMO port
S1C17M12/M13	P50
S1C17M20/M21/M22/M23/M24/M25	P12
S1C17M30/M31/M32/M33/M34	P07
S1C17W03/W04	P00
S1C17W12/W13	P00
S1C17W14/W16	P16
S1C17W18	P62
S1C17W22/W23	P16
S1C17W34/W35/W36	P20

4.23.3 Operations Overview

1. The following, for example, shows the outline of the NEC format specifications and Figure 4.23.3.1 shows the signal waveforms.
   - Carrier: Infrared ray (λp = 940 nm)
   - Subcarrier: fsc = 38 kHz, 1/3 duty
   - T = 562 µs
   - Fixed-length frame (32 bits)
   - 16-bit customer code
   - 8-bit data + 8-bit inverted data

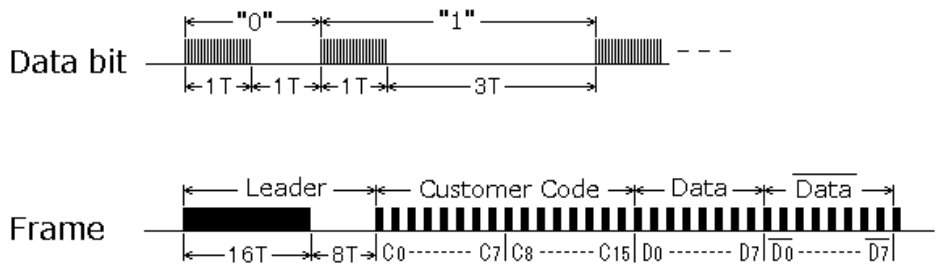


Figure 4.23.3.1 Signal Waveform in NEC Format

2. The sample program generates a data signal in NEC format.
3. Generates a data signal in AEHA format.
4. Generates a data signal in SONY format.

4.24 Software IR Remote Controller (SOFT REMC)

4.24.1 Sample Program Specifications

The software IR remote controller sample program simulatively realizes IR remote controller operations using T16 and T16B to perform the following processing:

- Generates a data signal in NEC format.
- Generates a data signal in AEHA format.
- Generates a data signal in SONY format.

In an MCU that includes T16B, the sample program uses two channels of T16B, or one channel of T16B and two channels of T16; in an MCU that does not include T16B, the sample program uses four channels of T16. The timers to be used can be specified in "c17_init_config.h." However, do not use these timers for other purposes while this program is running. Also be aware that running this sample software changes the register values for these timers.

When executing this sample program, set T16 (#define C17_PE_T16) and SOFTREMC (#define C17_PE_SOFTREMC) to be active.

4.24.2 Hardware Conditions

This sample program runs using an internal oscillator.

Table 4.24.2.1 List of Ports used for SOFT REMC Output

Model name	SOFT REMC output port
S1C17 Series	P00

4.24.3 Operations Overview

1. The following, for example, shows the outline of the NEC format specifications and Figure 4.24.3.1 shows the signal waveforms.
   - Carrier: Infrared ray (λp = 940 nm)
   - Subcarrier: fsc = 38 kHz, 1/3 duty
   - T = 562 µs
   - Fixed-length frame (32 bits)
   - 16-bit customer code
   - 8-bit data + 8-bit inverted data

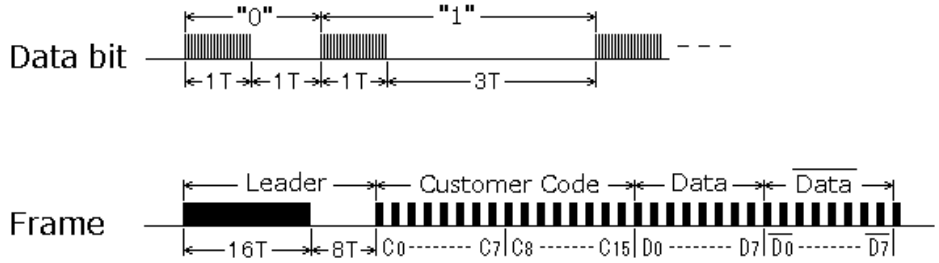


Figure 4.24.3.1 Signal Waveform in NEC Format

2. The sample program generates a data signal in NEC format.
3. Generates a data signal in AEHA format.
4. Generates a data signal in SONY format.

4.25 12-bit A/D Converter (ADC12A)

4.25.1 Sample Program Specifications

The A/D converter sample program performs the following processing using the 12-bit A/D converter:

- Performs an A/D conversion every second and obtains the A/D conversion results.
- Puts the CPU into HALT mode to reduce power consumption when waiting for an interrupt.

When executing this sample program, set T16 (#define C17_PE_T16) and ADC (#define C17_PE_ADC) to be active.

4.25.2 Hardware Conditions

This sample program runs using an internal oscillator.

4.25.3 Operations Overview

1. The sample program configures the 12-bit A/D converter.
2. Sets the 16-bit timer counter.
3. Starts the 16-bit timer.
4. Sets the 12-bit A/D converter to wait for a trigger and puts the CPU into HALT mode.
5. A 16-bit timer interrupt occurs every second to trigger an A/D conversion.
6. When a 16-bit timer interrupt occurs, the CPU exits from HALT mode.
7. When a 12-bit A/D converter interrupt occurs, the sample program reads the conversion results from the 12-bit A/D converter and counts the number of interrupts occurred.
8. If the 12-bit A/D converter interrupt count is less than 10, the sample program puts the CPU into HALT mode again. (To Step 5)
9. When the 12-bit A/D converter interrupt count reaches 10, the sample program stops the 12-bit A/D converter and 16-bit timer, and then stops running.

4.26 Operational Amplifier/Comparator (OPCMP)

4.26.1 Sample Program Specifications

The operational amplifier/comparator sample software performs the following processing using the operational amplifier/comparator:

- Obtains the comparison results from the comparator in OPCMP.

When executing this sample program, set T16 (#define C17_PE_T16) and OPCMP (#define C17_PE_OPCMP) to be active.

4.26.2 Hardware Conditions

This sample program runs using an internal oscillator.

Table 4.26.2.1 List of Ports used for SOFT REMC Output

Model name	OPIN0P port	OPIN0N port	OPOUT0 port
S1C17W23	P06	P07	P10

4.26.3 Operations Overview

1. The sample program configures the operational amplifier/comparator.
2. Loads the comparison results from the operational amplifier/comparator.
3. Stops the operational amplifier/comparator and then stops running.

4.27 Temperature Sensor/Reference Voltage Generator (TSRVR)

4.27.1 Sample Program Specifications

The temperature sensor/reference voltage generator sample program performs the following processing using the temperature sensor/reference voltage generator and the 12-bit A/D converter:

- Performs an A/D conversion of the internal temperature sensor every second and obtains the A/D conversion results.
- Puts the CPU into HALT mode to reduce power consumption when waiting for an interrupt.

When executing this sample program, set T16 (#define C17_PE_T16), ADC (#define C17_PE_ADC), and TSRVR (#define C17_PE_TSRVR) to be active.

4.27.2 Hardware Conditions

The sample program runs using an internal oscillator.

4.27.3 Operations Overview

1. The sample program configures the temperature sensor/reference voltage generator.
2. The sample program configures the 12-bit A/D converter.
3. Sets the 16-bit timer counter.
4. Starts the 16-bit timer.
5. Sets the 12-bit A/D converter to wait for a trigger and puts the CPU into HALT mode.
6. A 16-bit timer interrupt occurs every second to trigger an A/D conversion.
7. When a 16-bit timer interrupt occurs, the CPU exits from HALT mode.
8. When a 12-bit A/D converter interrupt occurs, the sample program reads the conversion results from the 12-bit A/D converter and counts the number of interrupts occurred.
9. If the 12-bit A/D converter interrupt count is less than 10, the sample program puts the CPU into HALT mode again. (To Step 6)
10. When the 12-bit A/D converter interrupt count reaches 10, the sample program calculates the temperature from the 12-bit A/D converter conversion results.
11. The sample program stops the temperature sensor/reference voltage generator, 12-bit A/D converter, and 16-bit timer, and then stops running.

Revision History

Attachment-1

Rev. No.	Date	Page	Category	Contents
Rev 1.0	2017/02/28	All	New	New establishment
Rev 1.1	2018/01/10	All	Correction	Unified the words as follows: • Sample software (means the whole program.) • Sample program (means an individual program module) Corrected Figure/table numbers.
		2, 3	Correction	Modified the directory and file names.
		3	Modification	Modified the order of items.
		4	Modification	Changed the linked web addresses. Corrected the model-specific information files copy procedure.
		5	Modification	Deleted the “c17_mcu_select.h” file name from Step (3) in Section 3.3.1.
		16	Addition	Added models (S1C17W12, S1C17M10, S1C17M20–M25) and a peripheral circuit (SOFT REMC) to Tables 4.1.1 and 4.1.2.
		17	Correction	Replaced the contents of Section 4.2 (before Section 4.2.1).
		18	Deletion	Deleted item (3) in Section 4.2.2.
		19–45	Addition	Added explanation on the modules to be activated when executing the sample program to Section 4.x.1.
		20	Addition	Added models (S1C17M10, S1C17M20–M25, S1C17W12) to Table 4.4.2.1.
		27	Addition	Added models (S1C17M20–M25) to Table 4.10.2.1.
		32	Addition	Added a new section, 4.15 LCD Driver (LCD16A). (Changed the subsequent section numbers.)
		36	Addition	Added models (S1C17M22/M25, S1C17W03/W04, S1C17W12/W13) to Table 4.19.2.1.
		37	Addition	Added models (S1C17M20–M25, S1C17W12) to Table 4.20.2.1.
		39	Modification	Modified the contents in Section 4.22.2.
		40	Addition	Added models (S1C17M20–M25, S1C17W12) to Table 4.23.2.1.
		41	Modification	Deleted signals (Repeat, Button, Transmit) from Figure 4.23.3.1.
		42	Addition	Added a new section, 4.24 Software IR Remote Controller (SOFT REMC).
		44	Correction	Corrected the contents in Section 4.26.
		45	Correction	Corrected the contents in Section 4.27.

EPSON

International Sales Operations

AMERICA

EPSON ELECTRONICS AMERICA, INC.
214 Devcon Drive,
San Jose, CA 95112, USA
Phone: +1-800-228-3964 FAX: +1-408-922-0238

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.
14F, No. 7, Song Ren Road,
Taipei 110, TAIWAN
Phone: +886-2-8786-6688 FAX: +886-2-8786-6660

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: 413445801
First Issue March 2016
Revised January 2018 in JAPAN Ⓛ