S1C17 Family Application Note

S1C17589
Peripheral Circuit Sample
Software

Evaluation board/kit and Development tool important notice

1. This evaluation board/kit or development tool is designed for use for engineering evaluation, demonstration, or development purposes only. Do not use it for other purposes. It is not intended to meet the requirements of design for finished products.

2. This evaluation board/kit or development tool is intended for use by an electronics engineer and is not a consumer product. The user should use it properly and in a safe manner. Seiko Epson dose not assume any responsibility or liability of any kind of damage and/or fire coursed by the use of it. The user should cease to use it when any abnormal issue occurs even during proper and safe use.

3. The part used for this evaluation board/kit or development tool may be changed without any notice.

NOTICE

Table of Contents

1. Overview

This manual describes the usage and behavior of the sample software for S1C17589.

The S1C17589 sample software is intended to show usage examples of peripheral circuits embedded in the S1C17589 microcontroller.

Refer to this document along with various model information, each technical manual, and S5U1C17001C Manual.

1.1 Operating Environment

Before running the S1C17589 sample software, prepare the following components:

・ Board equipped with an S1C17589
・ S5U1C17001H (hereinafter referred to as "ICDmini")
・ S5U1C17001C (hereinafter referred to as "GNU17")

Notes: • The operational check of this sample software is carried out in GNU17 v2.3.0/v3.0.5.

• There are multiple versions of GNU17 and ICDmini. For the ICDmini supported by the GNU17 to be used, refer to the S5U1C17001C Manual.

2. Sample Software Configuration

This chapter describes the file configuration of the S1C17589 sample software.

The S1C17589 sample software is composed of "Sample software" for checking the behavior of each peripheral circuit and "Sample driver" for controlling each peripheral circuit.

2.1 Directory Structure and File Composition

The following shows the directory structure of the S1C17589 sample software:

```
[s1c17589_xxx_sample]
   │
   ├─ [s1c17589_xxx_gnu17v2] : Sample software (GNU17 version 2 project)
   │     ├─ GNU17 project files, etc.
   │     ├─ ...
   │     ├─ [inc] : s1c17589 header files
   │     │     ├─ c17589_reg.h
   │     │     └─ [reg]
   │     │           ├─ ad10a_reg.h
   │     │           ├─ ...
   │     │           └─ wdt_reg.h
   │     │
   │     └─ [src] : Source codes and header files
   │           ├─ boot.c
   │           ├─ init.c
   │           ├─ init.h
   │           ├─ ...
   │           └─ main.c
   │
   └─ [s1c17589_xxx_gnu17v3] : Sample software (GNU17 version 3 project)
         ├─ GNU17 project files, etc.
         ├─ ...
         ├─ [inc] : s1c17589 header files
         │     ├─ c17589_reg.h
         │     └─ [reg]
         │           ├─ ad10a_reg.h
         │           ├─ ...
         │           └─ wdt_reg.h
         │
         └─ [src] : Source codes and header files
               ├─ boot.c
               ├─ init.c
               ├─ init.h
               ├─ ...
               └─ main.c
```

Figure 2.1.1   Directory structure of S1C17589 sample software

(1) s1c17589_xxx_sample directories

These directories contain the sample software.

(2) s1c17589_xxx_gnu17vx directories

These directories contain the files which pertain the GNU17 project.

"_gnu17v2" and "_gnu17v3," the last part of the directory names, represent the GNU17 version.

(3) inc directory

This directory contains the header files in which the register address for S1C17589 are defined.

・ Header files which define the register address for S1C17589 (c17589_reg.h, etc.)

(4) reg directory

This directory contains the header files in which the control bit assignments for each peripheral circuit are defined.

・ Header files which define the control bit assignments for each peripheral circuit (clg_589_reg.h, etc.)

(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 items used in the sample programs and drivers are defined.

・ File that contains vector table and initialization processing (boot.c)
・ Sample software file for each peripheral circuit (main.c)
・ Sample driver files for each peripheral circuit (init.c, clg.c, etc.)
・ Sample driver header files for each peripheral circuit (init.h, clg.h, etc.)

3.   Execution Method

3.1   Software Development Environment

This chapter describes the software development environment using the following tools. For more information on each tool, refer to each tool manual.

- ・ S5U1C17001C (GNU17) ver. 2.3.0/ver. 3.05
- ・ Model information files (S1C17589)
- ・ S5U1C17001H (ICDmini) ver. 2.0/ver. 3.0

3.2   Installing each Tool

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

- ・ Download the compressed GNU17 file from the Epson MCU users site.
  https://www.epsondevice.com/support_e/mcu/product/c17.html
- ・ Decompress the compressed GNU17 file into any folder.
- ・ Run Setup.exe to install GNU17.

(2) Install the model information files

- ・ Confirm and download the latest model information files from the Epson MCU users site.
  https://www.epsondevice.com/support_e/mcu/product/page04.html#02
- ・ Decompress the downloaded model information files into the [mcu_model] subfolder of the folder where GNU17 is installed.
  Example:

```
[GNU17]
  ├── [mcu_model] : Model information folder
  │         ├── [17xxx]
  │         │
  │         ├── [17589]
  │         │     ├── S1C17589.ini
  │         │     ├── S1C17589.properties
  │         │     ├── ...
  │         │     └── [fls]
  │         │           ├── fls17589_readme_e.txt
  │         │           ├── ...
  │         │           └── fls17589.elf
  ├── ...
```

Added model information files
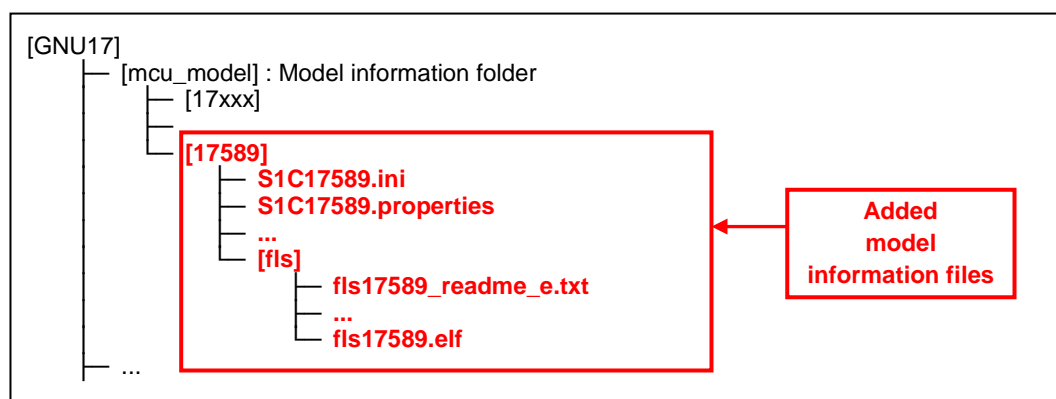
Figure 3.2.1   Adding model information files for S1C17589

(3) ICDmini ver. 2.0 (S5U1C17001H)

- ・ After installing GNU17 version 2, connect between the ICDmini and the PC with the USB cable.
- ・ Install the USB driver, that exists in the folder where the GNU17 was installed, to the PC. When the GNU17 was installed in the C:\EPSON\GNU17 directory, the USB driver was placed into the directories shown below.
  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 between the ICDmini and the PC with the USB cable.
- ・ Install the USB driver, that exists in the folder where the GNU17 was installed, to the PC. When the GNU17 was installed in the C:\EPSON\GNU17V3 directory, the USB driver was placed into the C:\EPSON\GNU17V3\utility\drv_usb\Mini3Driver directory.

3.3 Importing Project

This section describes how to import the sample program into GNU17.

Notes: • For more information on importing programs, refer to the S5U1C17001C Manual.

  • The program import procedure between GNU17 version 2 and version 3 is different.

3.3.1 Importing Sample Program (Common to GNU17 version 2 and version 3)

(1) Launch GNU17 IDE

  Double-click the eclipse.exe icon in the eclipse sub-directory of the directory where GNU17 was installed to launch IDE. IDE is also launched by selecting [EPSON MCU] > [GNU17] > [GNU17 IDE] ([EPSON MCU] >[GNU17V3] > [GNU17V3 IDE] in GNU17 version 3) from the Windows startup menu.

(2) Decompress sample program

  Decompress the sample program downloaded from the Epson MCU users site.

3. Execution Method

3.3.2 Importing Sample Program (GNU17 version 2)

(1) Import sample program

Select [Import...] from the GNU17 [File] menu. In the [Import] wizard, select [General] > [Existing Project to Workspace].
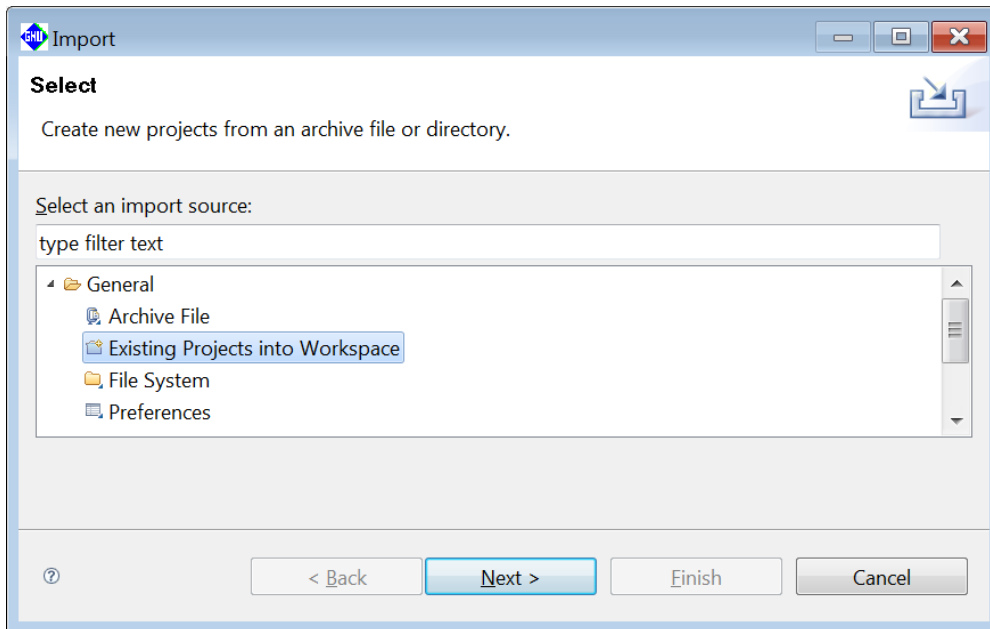
Click the [Browse...] button at the right of [Select root directory:] and select the [xxx_gnu17v2] project in the sample project folder decompressed in step (2) of Section 3.3.1.
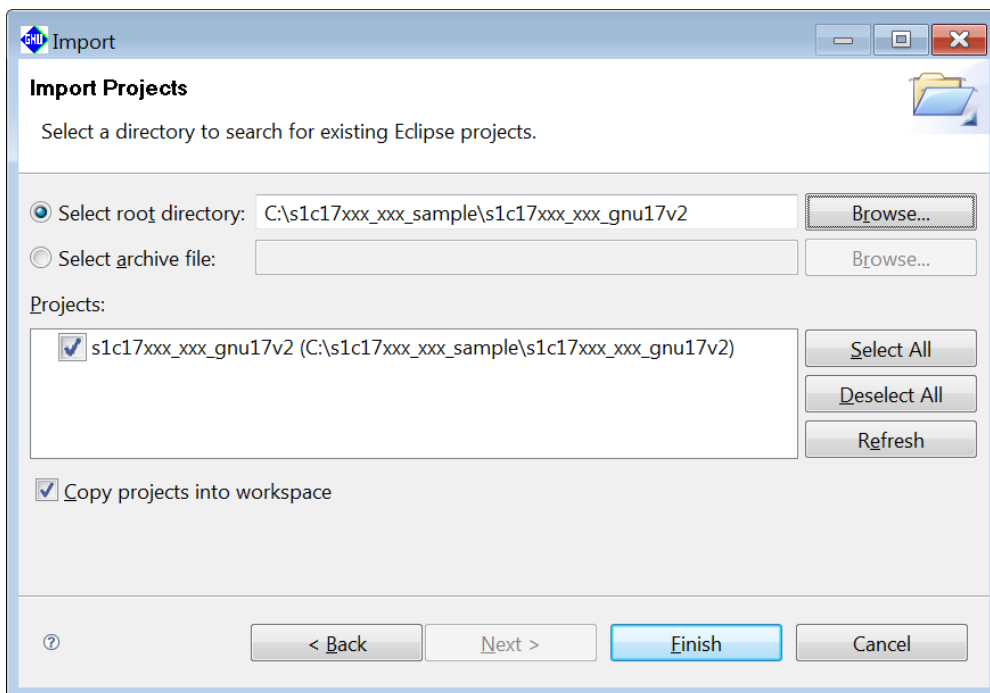
Select the [Copy projects into workspace] checkbox. This enables to create the project's copy in the workspace directory, thereby not changing the original files. Finally, click the [Finish] button. Check to see in GNU17 that the sample project has been imported and then remove the folder decompressed in step (2) of Section 3.3.1.

(2) Change the target

Change the target for the sample program. Select the sample program's project in the "C/C++ project" view and select [Properties] in the [Project] menu.

In the [Properties for ...] dialog, select [GNU17 General].
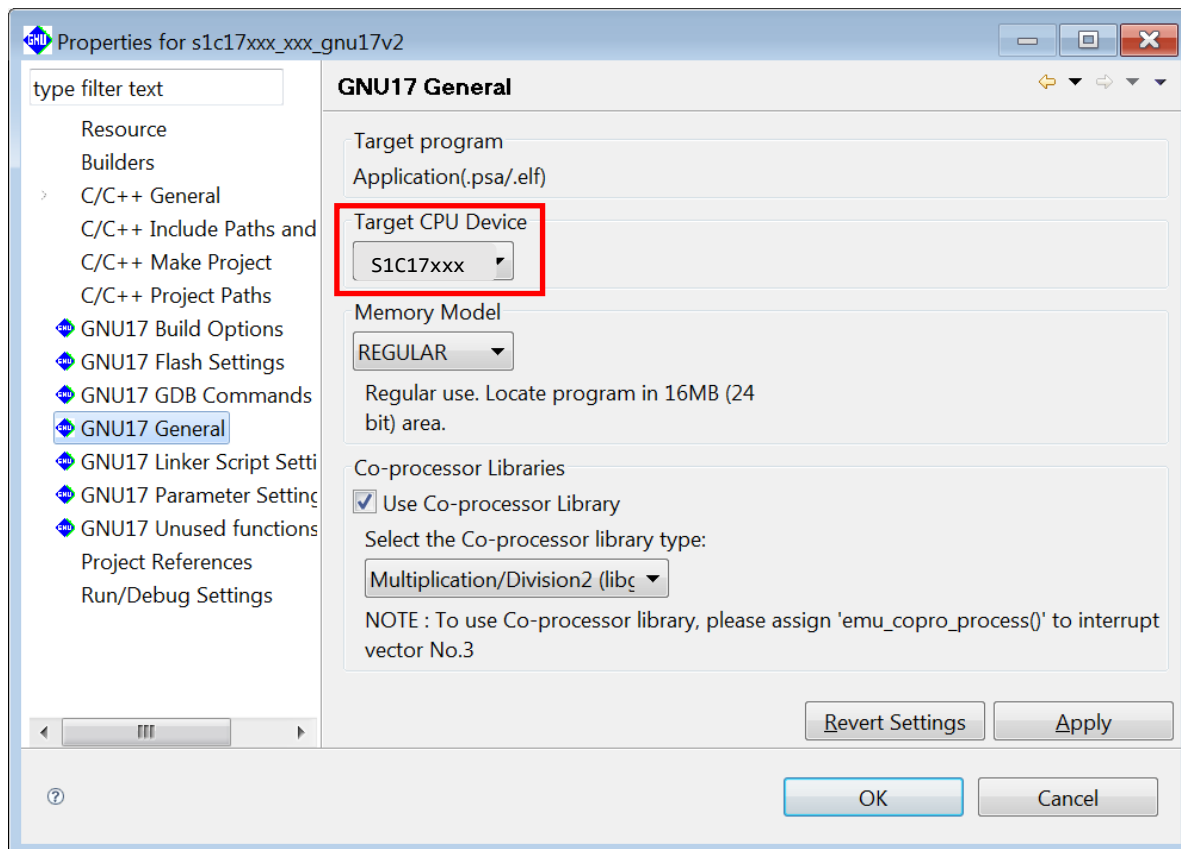
In the [Target CPU Device], select a target CPU and click the [OK] button.

For example, if the target CPU is S1C17589, change the Target CPU Device to "S1C17589."

3. Execution Method

(3) Set the debugger's startup options

Set the Startuo options for the debugger. Select the sample program's project in the "C/C++ project" view and select [Properties] in the [Project] menu.

In the [Properties for ...] dialog, select [GNU17 GDB Commands].
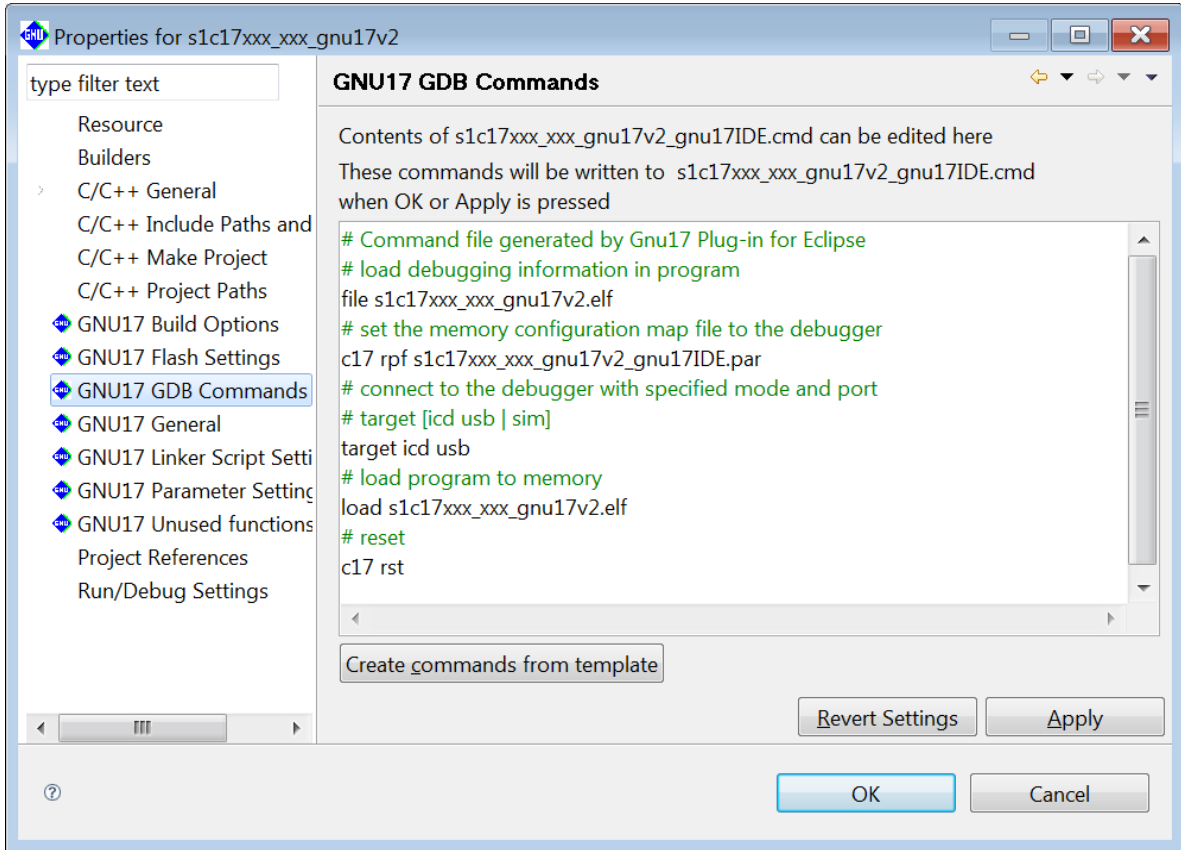
This page displays the contents of the debugger startup command file that will be generated by the IDE.

The debugger must be set to the appropriate mode that suits the ICD used, etc. before it can be operated.

Seiko Epson Corporation

The debugger startup command file should be created with a procedure shown below.

Click the [Create commands from template] button to display the [Create a simple startup command] dialog.

Select "ICD Mini" from the [Debugger:] combo box.
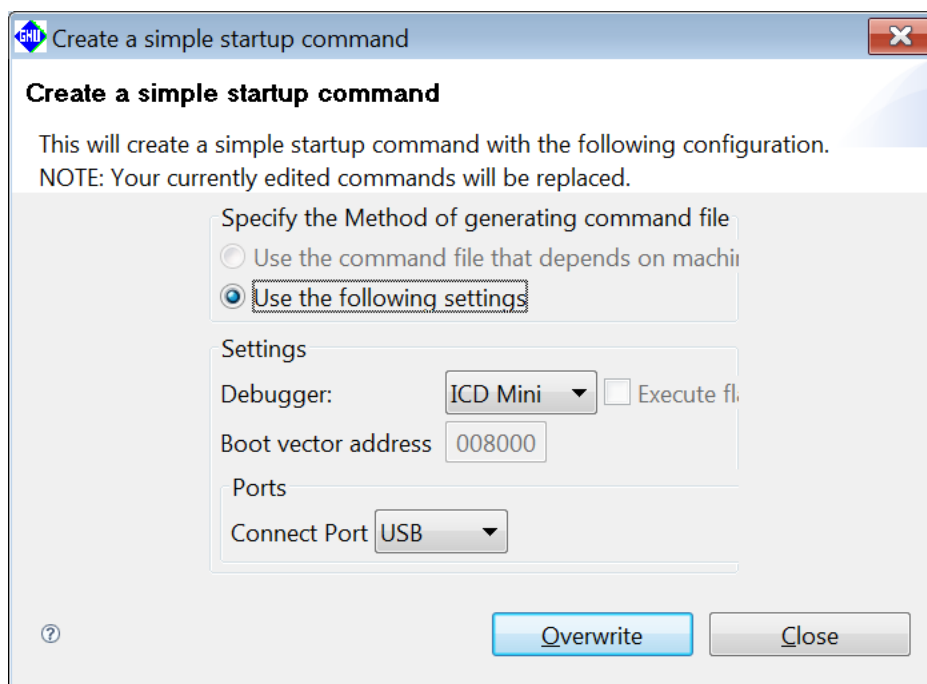
Click the [Overwrite] button, then click the [OK] button in the [Overwrite commands] dialog displayed next.

The commands may be added and edited directly in this page as necessary.

* Precautions on using a simulated I/O

When using the debugger's simulated I/O (to evaluate the external input/output functions such as a serial interface using a standard input/output (stdin, stdout) or by inputting/outputting a file), append the following lines to the GDB command file.

After last line "c17 rst", add following three lines

u main

c17 stdout 1 WRITE_FLASH WRITE_BUF

c17 stdin 1 READ_FLASH READ_BUF

The sample programs using simulated I/O are as follows.
・ Clock generator (CLG)
・ 16 bit PWM timer (T16B)

3. Execution Method

### 3.3.3 Importing Sample Program (GNU17 version 3)

(1) Importing the sample program

Select [Import...] from the [File] menu of GNU17 to start the [Import] wizard, and then select [General] > [Existing Projects into Workspace].
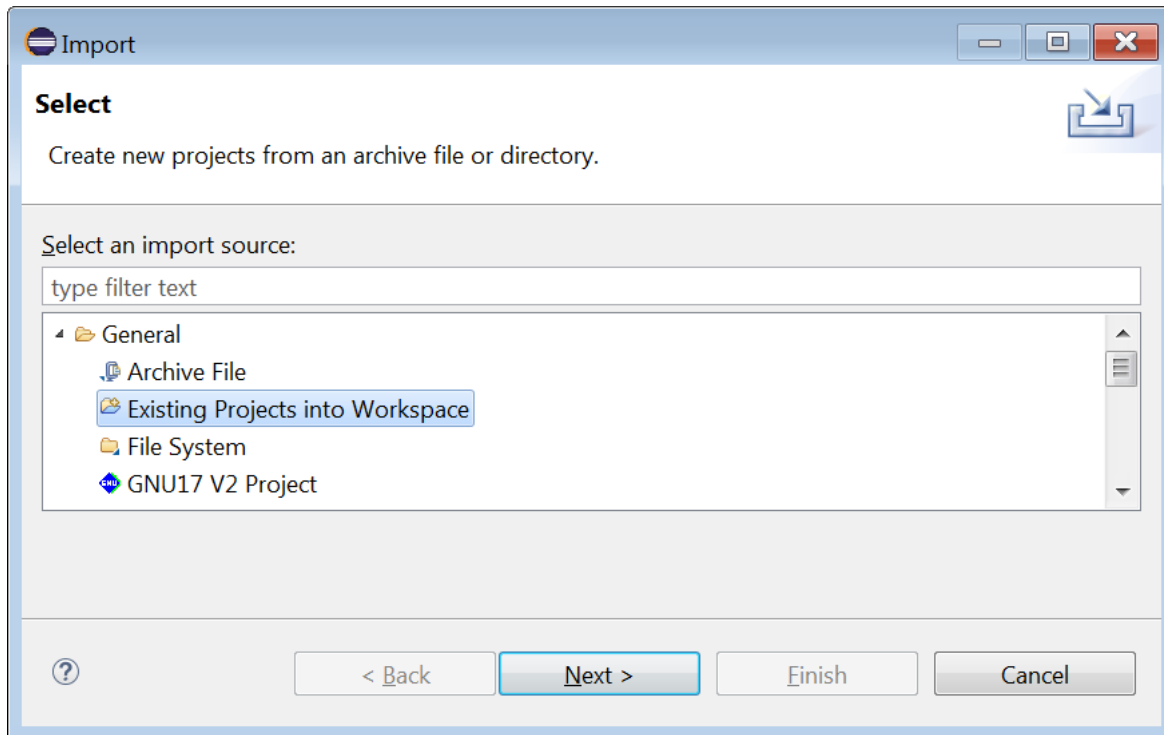
Click the [Browse...] button at the right of [Select root directory:] and select the [xxx_gnu17v3] project in the sample project folder decompressed in step (2) of Section 3.3.1.
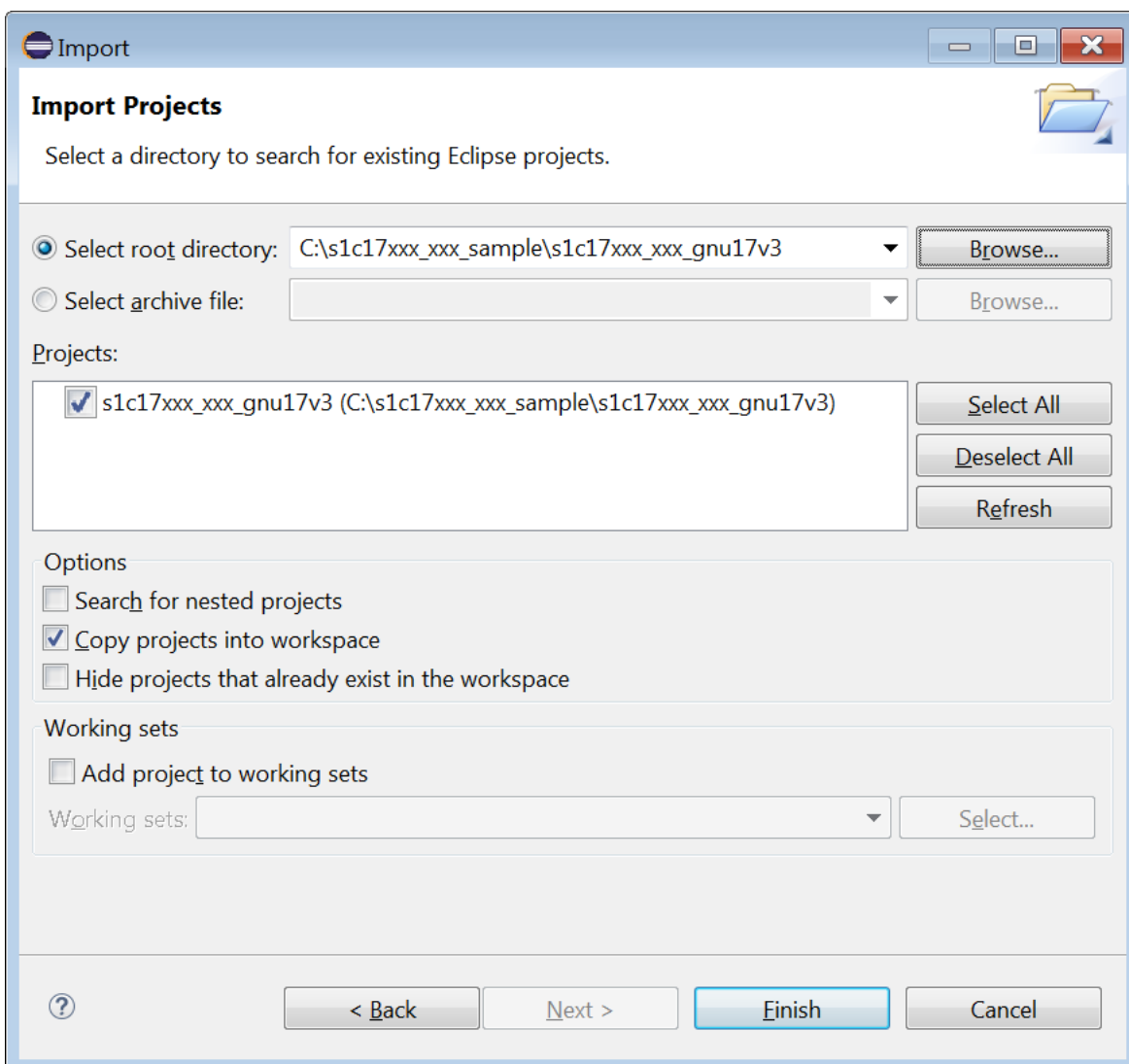

Select the [Copy projects into workspace] checkbox. This enables copying of the project into the workspace directory so that the original files will not be modified. Finally, click the [Finish] button. After making sure that the sample project has been imported to GNU17, the folder decompressed in the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

(2) Select the target CPU

The target CPU to run the sample program should be selected. Select the sample project in the [Project Explorer] view and select [Properties] from the [Project] menu to open the [Properties] dialog. Select [GNU17 Settings] from the property list on the dialog.
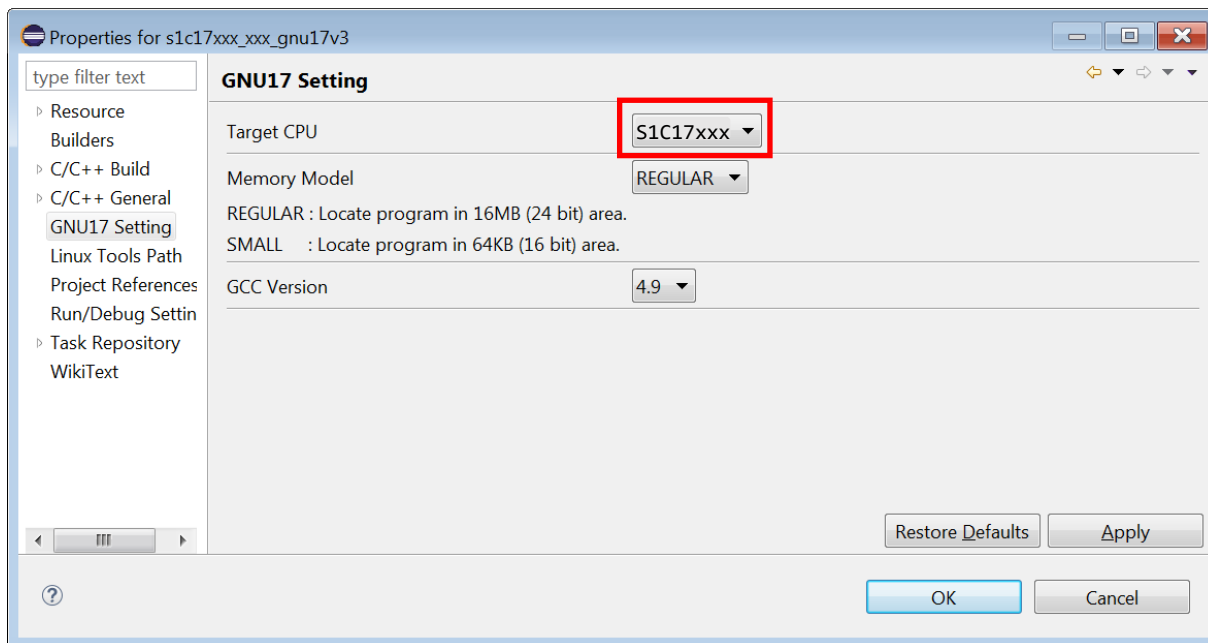
Select a target CPU from the [Target CPU] dropdown list, for example, select "S1C17589" if it is the target CPU, and click the [Apply] button. Then click the {OK} button to close the dialog.

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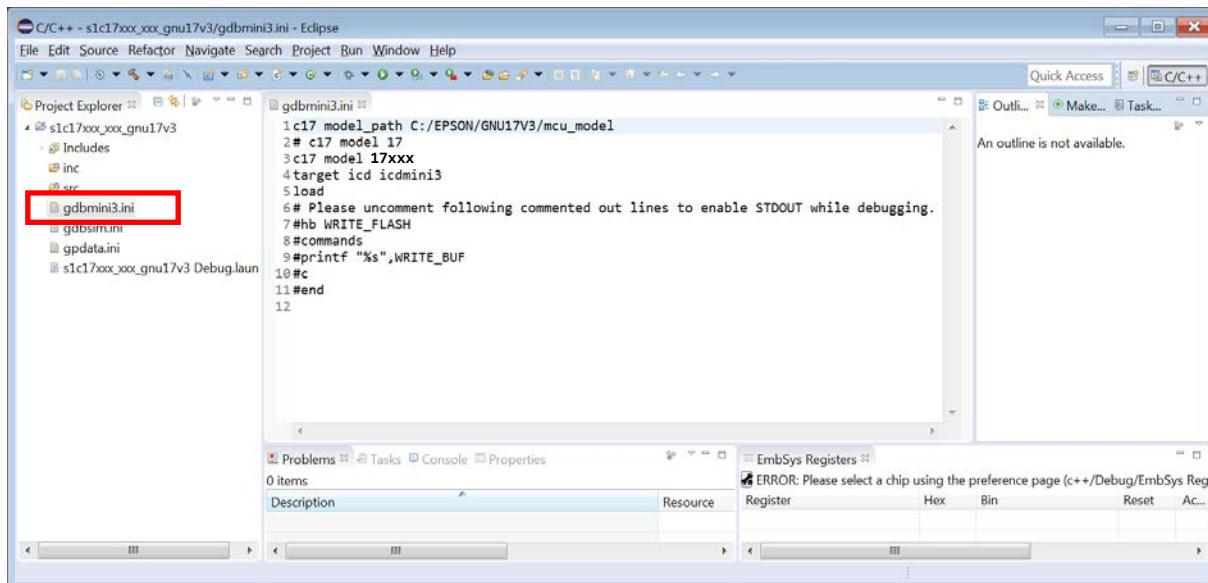
The editor displays the contents of the debugger startup command file. The debugger must be set to the appropriate mode that suits the microcontroller and ICD used.

* 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

When a standard output is used, it will take some time for displaying character strings.

The following sample programs use the standard output:

・ Clock generator (CLG)

・ 16-bit PWM timer (T16B)

3.4 Connecting with Target

Connect the PC where GNU17 was installed and the ICDmini with the USB cable, and connect the board equipped with an S1C17589 and ICDmini.

Note: When using ICDmini ver. 3.0 with GNU17 version 3, use the ICDmini Ver. 1.0, 1.1, and 2.0 compatible cable (S5U1C1700 1 W7200) to connect between the ICDmini and the target system.
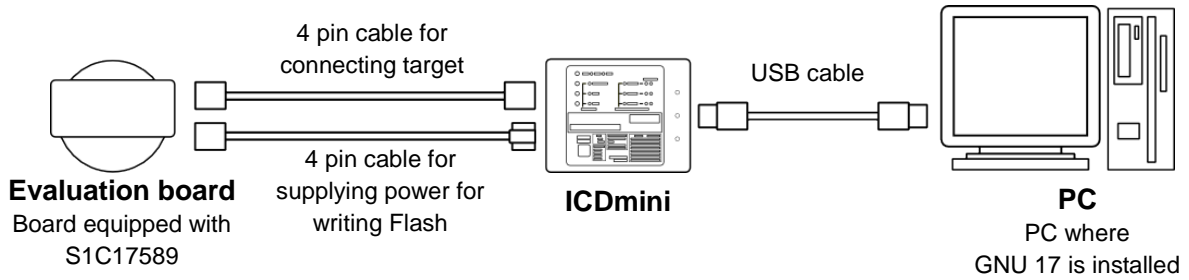
Figure 3.4.1 Connection diagram among the target, ICDmini, and PC

(1) Connection example between ICDmini and S5U1C17589T21

The figure below shows the connection example between ICDmini and S5U1C17589T21. For more information, refer to the S5U1C17589T21 Manual.
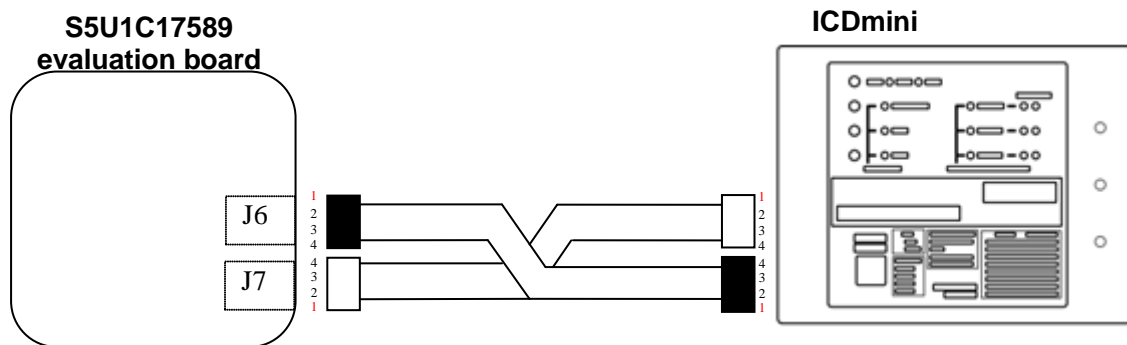
Figure 3.4.2 Connection diagram between ICDmini and S5U1C17589T21

3.5 Operating ICDmini and Target

3.5.1 When ICDmini ver. 2.0 is Used

Connect the target, ICDmini, and PC, reset them in the order below, and then confirm that the connection between the target and ICDmini is established. In addition, the setting of DIP Switch of ICDmini is as follows.

Table 3.5.1.1 DIP switch of ICDmini

DIP Switch	Description	When S5U1C17589T21 is connected
1	Target CPU (C17 / C33)	OPEN↑
2	FLASH WRITER (OFF / WRITE)	OPEN↑
3	FLASH WRITER (OFF / VERIFY)	OPEN↑
4	DSIO LVL (ON / TRGET VCC)	ON↓
5	DSIO LVL (3.3V / 1.8V)	OPEN↑
6	CONNECTION DIAG (OFF / ON)	OPEN↑
7	FW UPDATE MODE (OFF / ON)	OPEN↑
8	FLASH VCC OUT (OFF / ON)	ON↓
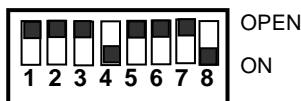
Figure 3.5.1.1 DIP switch of ICDmini

(1) Reset ICDmini

Set the DIP switch of ICDmini as the figure above and click the [RESET/START] button in ICDmini.

(2) Reset the target

Reset the target. For information on how to reset each target, refer to the target's manual.

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

Confirm that the LEDs light up as follows in ICDmini.
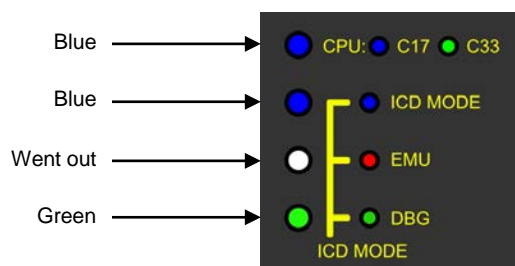
Figure 3.5.1.2 LED display when ICDmini connection is established

3.5.2   When ICDmini ver. 3.0 is Used

Connect the target, ICDmini, and PC, and then confirm that the connection between the target and ICDmini is established.

(1) Confirm the established connection between the target and ICDmini

   Confirm that the ICDmini LEDs light up as follows.
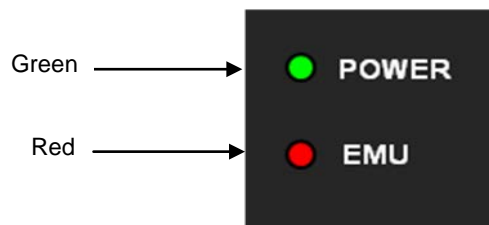
Figure 3.5.2.1   LED display when ICDmini connection is established

3.6   Building Project and Starting Debugger

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

Notes: • For more information on how to build 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) Build programs

In the GNU17's "C/C++ project" view, select the desired project to build. In the GNU17's [Project] menu, select [Build Project] to build the project. Select the desired project in the GNU17's "C/C++ project" view and select [Project's Build] in the context menu displayed by right-clicking the project to also build the project.

(2) Display debugger launch configuration dialog

Select [Debug Configuration...] in the GNU17's [Run] menu to display the [Debug Configurations] dialog. This dialog can be also displayed by clicking the [Debug] button in the toolbar.

(3) Select sample program to be launched

Select [GDB17 Debugger for <project name>] in the tree list.
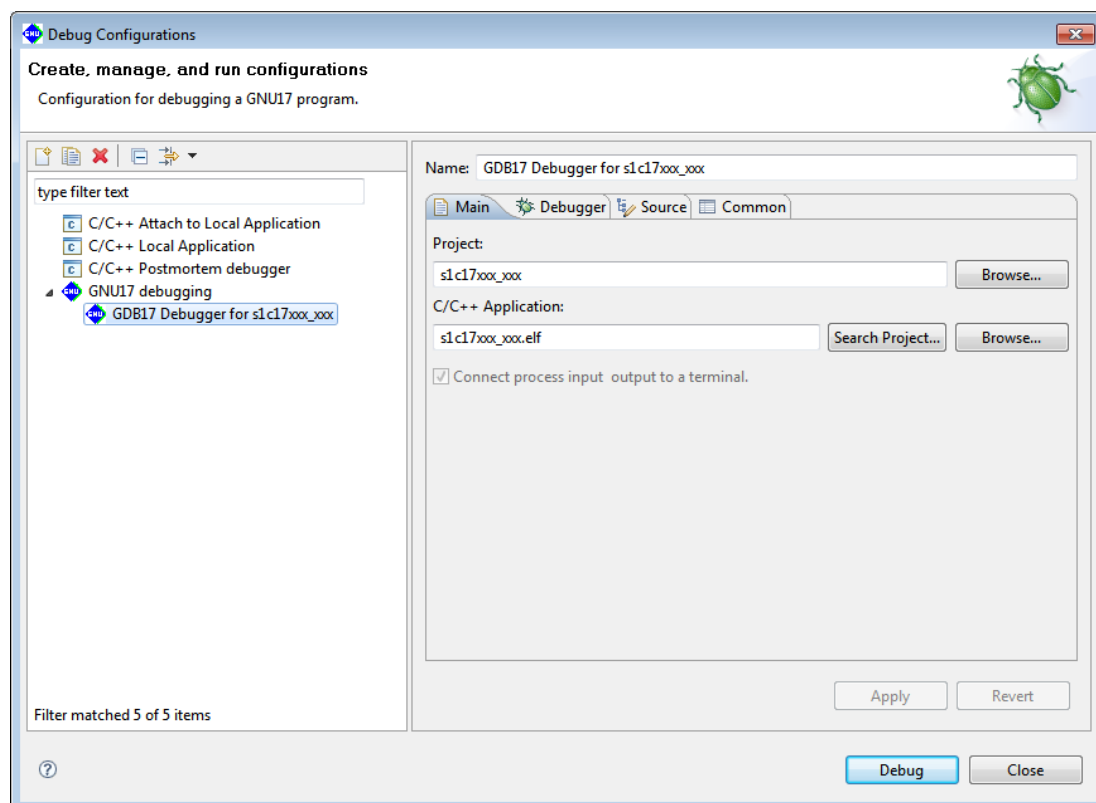
(4) Launch debugger

Click the [Debug] button. The debugger gdb starts and executes prescribed command file.

3. Execution Method

3.6.2 When GNU17 version 3 is Used

(1) Building

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

(2) Displaying [Debug Configuration] dialog

Select [Debug Configurations...] from the GNU17 [Run] menu or the [Debug] dropdown list on the toolbar to display the [Debug Configurations] dialog.

(3) 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.
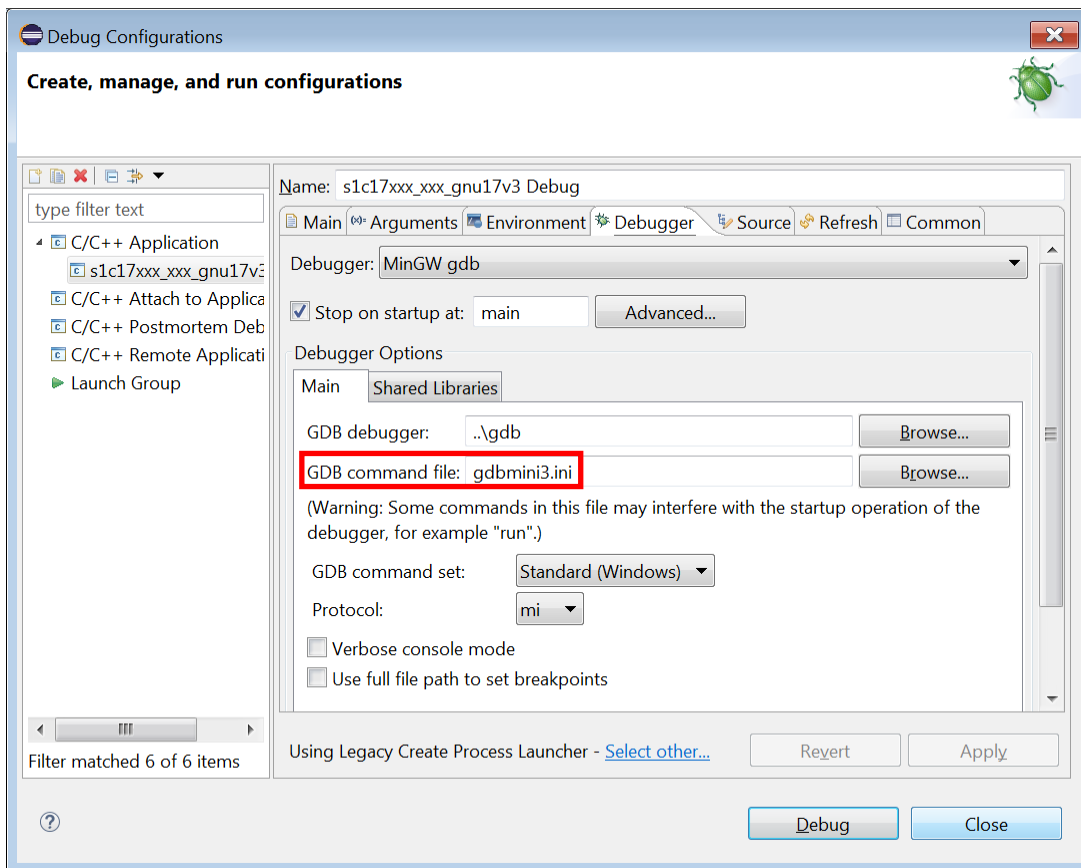
4. Details of Sample Software Function

This chapter describes the functions of the S1C17589 sample software in detail.

4.1 I/O Port (PPORT)

4.1.1 Sample Software Specification

This sample software works as follows by using I/O ports.

• When a port is set to input and the input signal becomes High or Low level, an interrupt occurs.
• The port is set to output and a High or Low level signal is output.

The table below lists the ports to be used and port names.

Table 4.1.1 I/O port setting list

Setting	Port name
Input interrupt port	P01
Output port	P02

4.1.2 Hardware Conditions

This sample program runs in the OSC1 built-in oscillating circuit (32.768kHz) in the microcontroller.

For this sample program, connect the ports as follows in the microcontroller.
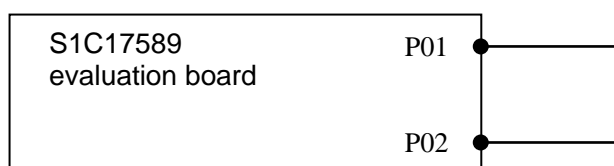
Figure 4.1.1 I/O port sample software/hardware connection diagram

4. Details of Sample Software Function

4.1.3 Operations Overview

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 P01 port to output and Output a Low level signal.

4. Set the P02 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 P01 port.

6. Confirm that the P02 port is at High level after an interrupt in the P02 port.

7. Set the P02 port so that an interrupt occurs when the level changes from High to Low.

8. Output a Low level signal from the P01 port.

9. Confirm that the P02 port is at Low level after an interrupt in the P02 port.

4.2 Clock Generator (CLG)

4.2.1 Sample Software Specification

This sample program works as follows by using OSC.

- Oscillate and stop OSC1.
- Oscillate and stop OSC3. The setting of the oscillatory frequency is possible, too.
- Oscillate and stop IOSC.
- Change the system clock to OSC1.
- Change the system clock to OSC3.
- Change the system clock to IOSC.
- In the case of system clock is IOSC/ OSC3, do the transition control between SLEEP mode and RUN mode.
- Do the transition control between OSC1 HALT, IOSC RUN , and OSC3 RUN .
- Change the system clock of OSC3.

4.2.2 Hardware Conditions

This sample program works by using the built-in oscillation (IOSC) in the microcontroller. Then change the system clock to OSC1 or OSC3 . Therefore it needs transducer. For details, please refer to "Clock Generator (CLG)" in S1C17589 technical manual.

4.2.3 Operations Overview

This sample program starts to work while IOSC is working.

This sample program starts to work by using IOSC as a system clock. Note that the count value displayed below is estimated to be a little bigger than 327, the 10 millisecond count value of OSC1's oscillating frequency = 32.768kHz.

0. Start to oscillate OSC1, change the system clock to OSC1, and then stop the IOSC oscillation.
1. Stop CPU (HALT) and repeat to interrupt the timer three times per second. Each time a timer interrupt is received, start the IOSC oscillation, execute 10 milliseconds of loop, display a difference between the T16 timer count values before and after the loop on Simulated I/O, and stop IOSC.
2. Trim IOSC's oscillating frequency automatically based on the OSC1's oscillating clock.
3. Since this operation is the same as step 1, the automatic trimming result can be confirmed.
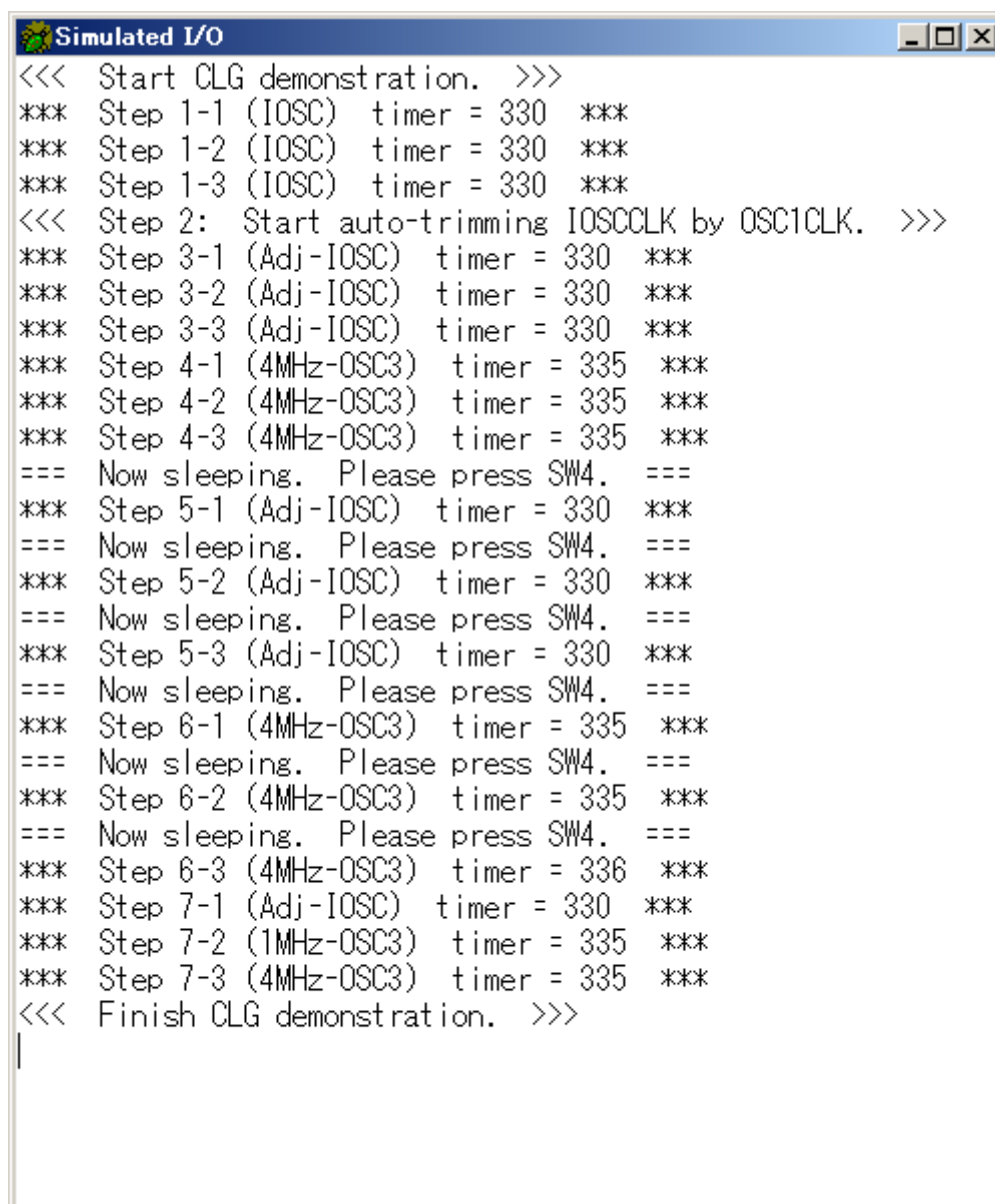4. Stop CPU (HALT) and repeat to interrupt the timer three times per second. Each time a timer interrupt is received, start the OSC3 oscillation (4MHz), execute 10 millisecond of loop, display a difference between the T16 timer count values before and after the loop on Simulated I/O, and stop OSC3.
5. After IOSC started to oscillate, stop CPU (SLEEP). In order to wake up CPU from the sleep mode, it is necessary to put the level "L" signal into port P04. For S5U1C17589 (SVT17mini589), it is possible to execute the action by pushing the push button SW4. When CPU wakes up and IOSC automatically starts to oscillate, execute 10 milliseconds of loop, display a difference between the T16 timer count values before and after the loop on Simulated I/O, and stop IOSC. This operation is repeated three times.
6. After OSC3 started to oscillate (4MHz), stop CPU (SLEEP). In order to wake up CPU from the sleep mode, it is necessary to put the level "L" signal into port P04. For S5U1C17589 (SVT17mini589), it is possible to execute the action by pushing the push button SW4. When CPU wakes up and OSC3 automatically starts to oscillate, execute 10 millisecond of loop, display a difference between the T16 timer count values before and after the loop on Simulated I/O, and stop OSC3. This operation is repeated three times.
7. At first, start the IOSC oscillation, execute 10 milliseconds of loop, display a difference between the T16 timer count values before and after the loop on Simulated I/O. Next, start the OSC3 oscillation (1MHz), stop the IOSC oscillation, execute 10 milliseconds of loop, display a difference between the T16 timer count values before and after the loop on Simulated I/O. Then, start the IOSC oscillation, change the clock, and stop the OSC3 oscillation in order to start the OSC3 oscillation at 4MHz. Then, start the OSC3 oscillation (4MHz), stop the IOSC oscillation, execute 10 milliseconds of loop, display a difference between the T16 timer count values before and after the loop on Simulated I/O.

8. Finally, the program starts the IOSC oscillation, stops the OSC1 and OSC3 oscillations to finish the series of operations.


```
Simulated I/O                                    _ □ X
<<<  Start CLG demonstration.  >>>
***  Step 1-1 (IOSC)  timer = 330   ***
***  Step 1-2 (IOSC)  timer = 330   ***
***  Step 1-3 (IOSC)  timer = 330   ***
<<<  Step 2:  Start auto-trimming IOSCCLK by OSC1CLK.  >>>
***  Step 3-1 (Adj-IOSC)  timer = 330   ***
***  Step 3-2 (Adj-IOSC)  timer = 330   ***
***  Step 3-3 (Adj-IOSC)  timer = 330   ***
***  Step 4-1 (4MHz-OSC3)  timer = 335   ***
***  Step 4-2 (4MHz-OSC3)  timer = 335   ***
***  Step 4-3 (4MHz-OSC3)  timer = 335   ***
===  Now sleeping.  Please press SW4.   ===
***  Step 5-1 (Adj-IOSC)  timer = 330   ***
===  Now sleeping.  Please press SW4.   ===
***  Step 5-2 (Adj-IOSC)  timer = 330   ***
===  Now sleeping.  Please press SW4.   ===
***  Step 5-3 (Adj-IOSC)  timer = 330   ***
===  Now sleeping.  Please press SW4.   ===
***  Step 6-1 (4MHz-OSC3)  timer = 335   ***
===  Now sleeping.  Please press SW4.   ===
***  Step 6-2 (4MHz-OSC3)  timer = 335   ***
===  Now sleeping.  Please press SW4.   ===
***  Step 6-3 (4MHz-OSC3)  timer = 336   ***
***  Step 7-1 (Adj-IOSC)  timer = 330   ***
***  Step 7-2 (1MHz-OSC3)  timer = 335   ***
***  Step 7-3 (4MHz-OSC3)  timer = 335   ***
<<<  Finish CLG demonstration.  >>>
|
```

Figure 4.2.2   Display example of CLG_OSC sample program screen

4.3 16-Bit Timer (T16)

4.3.1 Sample Software Specification

The 16-bit timer sample program works as follows by using 16-bit timer.

• Cause an interrupt in the 16-bit timer to get a counter value of the timer.
• When waiting for an interrupt, put CPU in a halt mode to reduce the power consumption.

4.3.2 Hardware Conditions

This sample program runs in the IOSC built-in oscillating circuit (8MHz) in the microcontroller.

4.3.3 Operations Overview

1. Set the counter of the 16-bit timer.

2. Start the 16-bit timer and put CPU in a Halt state.

3. When an interrupt occurs in the 16-bit timer, release CPU from the Halt state.

4. Get the number of interrupts of the 16-bit timer.

5. When the number of interrupts of the 16-bit timer is less than 1000, put CPU in a Halt state again.

6. When the number of interrupts of the 16-bit timer is not less than 1000, stop the 16-bit timer and exit the sample program.

4.4 16-Bit PWM Timer (T16B)

4.4.1 Sample Software Specification

The 16-bit PWM timer sample program works as follows by using 16-bit PWM timer.
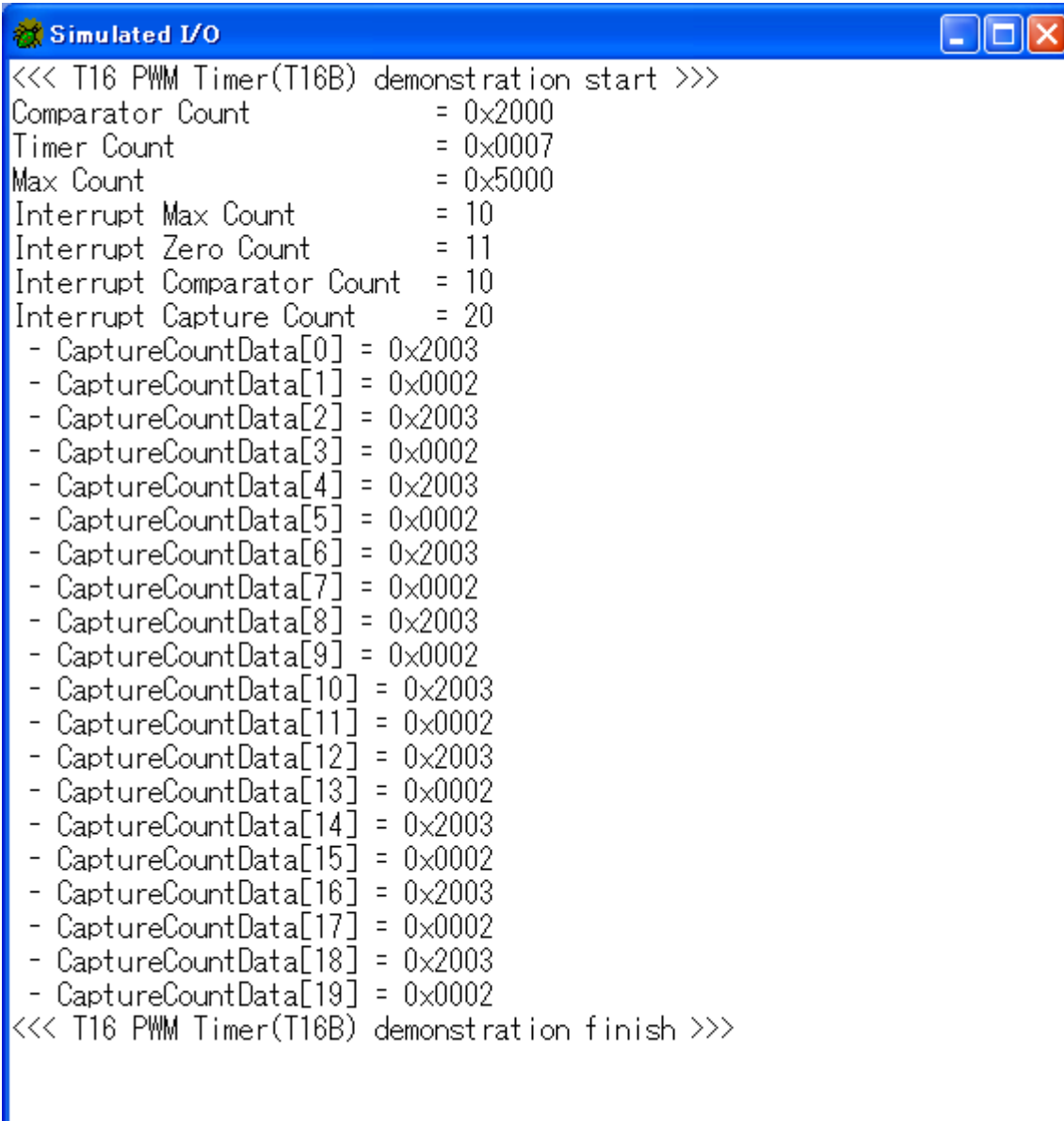
• Cause a comparator interrupt in the 16-bit PWM timer.
• Cause a capture interrupt in the 16-bit PWM timer to get a counter value of the timer.
• When waiting for an interrupt, put CPU in a halt mode to reduce the power consumption.

4.4.2 Hardware Conditions

This sample program runs in the IOSC built-in oscillating circuit (8MHz) in the microcontroller.

4.4.3 Operations Overview

1. Set the 16bit PWM timer as follows and start the T16-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.

7. When the number of maximum-count interrupts is less than 10, put CPU in a Halt state again.

8. When the number of maximum-count interrupts is not less than 10, stop the 16-bit timer, shows a count value when an interrupt occurred for Simulated I/O and exit the sample program.

```
Simulated I/O                                          _ □ ×
<<< T16 PWM Timer(T16B) demonstration start >>>
Comparator Count               = 0x2000
Timer Count                    = 0x0007
Max Count                      = 0x5000
Interrupt Max Count            = 10
Interrupt Zero Count           = 11
Interrupt Comparator Count     = 10
Interrupt Capture Count        = 20
 - CaptureCountData[0] = 0x2003
 - CaptureCountData[1] = 0x0002
 - CaptureCountData[2] = 0x2003
 - CaptureCountData[3] = 0x0002
 - CaptureCountData[4] = 0x2003
 - CaptureCountData[5] = 0x0002
 - CaptureCountData[6] = 0x2003
 - CaptureCountData[7] = 0x0002
 - CaptureCountData[8] = 0x2003
 - CaptureCountData[9] = 0x0002
 - CaptureCountData[10] = 0x2003
 - CaptureCountData[11] = 0x0002
 - CaptureCountData[12] = 0x2003
 - CaptureCountData[13] = 0x0002
 - CaptureCountData[14] = 0x2003
 - CaptureCountData[15] = 0x0002
 - CaptureCountData[16] = 0x2003
 - CaptureCountData[17] = 0x0002
 - CaptureCountData[18] = 0x2003
 - CaptureCountData[19] = 0x0002
<<< T16 PWM Timer(T16B) demonstration finish >>>
```

Figure 4.4.1 T16B sample program screen

4.5 Real Time Clock (RTCA)

4.5.1 Sample Software Specification

The real time clock sample program works as follows by using real time clock.

• Set the time of real time clock.
• Get the time of real time clock.
• Get the number of interrupts of real time clock.
• Use theoretical regulation function.
• Use the stopwatch.

4.5.2 Hardware Conditions

This sample program runs in the OSC1 built-in oscillating circuit (32.768 kHz) in the microcontroller.

4.5.3 Operations Overview

1. Initialize the Real Time Clock.

2. Enable one- and a half-second interrupts of real time clock.

3. Set the theoretical regulation function of real time clock (theoretical regulation interval: 4,096sec, correction factor: -58.2ppm).

4. Set the date and time of real time clock to "Sat 99-12-31 23:59:51" and set the clock to 24-hour mode.

5. Start real time clock and put CPU in a Halt state.

6. When an interrupt occurs in real time clock, release CPU from the Halt state.

7. Get the number of one-second interrupts of real time clock.

8. When the number of one-second interrupts is less than 10, put CPU in a Halt state again.

9. When the number of one-second interrupts is not less than 10, set the clock to 12-hour mode and get the date and time of real time clock (the value is "Sun 00-01-01 12:00:01 AM").

10. Disable the one-second interrupt of real time clock.

11. Execute theoretical regulation at 4,096sec interval after real time clock starts.

12. Initialize the stopwatch.

13. Enable an interrupts of stopwatch.

14. Start stopwath and put CPU in a Halt state.

15. When an interrupt occurs in stopwatch, release CPU from the Halt state.

16. Get the number of 1 Hz interrupts of stopwatch.

17. When the number of 1 Hz interrupts is less than 10, put CPU in a Halt state again.

18. When the number of 1 Hz interrupts is not less than 10, read a stopwatch count (10 Hz digit, 100 Hz digit)

19. Stop the stopwatch and disable an interrupt of stopwatch.

4.6 Watchdog Timer (WDT)

4.6.1 Sample Software Specification

The watchdog timer sample program works as follows by using the watchdog timer.

• Confirm that the watchdog timer is cleared.
• Cause a reset interrupt with the watchdog timer.

4.6.2 Hardware Conditions

This sample program runs in the OSC1 built-in oscillating circuit (32.768 kHz) in the microcontroller.

4.6.3 Operations Overview

1. Start the watchdog timer and real time clock.

2. If one-second interrupt occurs in real time clock, reset the watchdog timer.

3. If one-second interrupt occurs 10 times in real time clock, stop real time clock.

4. If a reset interrupt occurs with the watchdog timer, the processing returns to the top (boot.c) of this sample program.

4. Details of Sample Software Function

4.7 UART (UART)

4.7.1 Sample Software Specification

The UART sample program works as follows by using UART.

- Send data by using UART.
- Receive data by using UART.

4.7.2 Hardware Conditions

This sample program runs in the IOSC built-in oscillating circuit (8MHz) in the microcontroller.

Connect the evaluation boards where the UART master/slave sample programs are installed. Launch the slave sample program first. Connect each port as follows:

Figure 4.7.1 Hardware connection diagram for UART sample program

4.7.3 Operations Overview

4.7.3.1 Operations Overview of Master Sample

1. Initialize the UART ports with the values below:
   - Communication speed   : 500kbps
   - Data length   : 8bit
   - Stop bit   : 1bit
   - Parity   : None

2. Send the data from "A (0x41)" to "Z (0x5A)" and "¥0" to the slave.

3. Receive the 27byte of data from the slave.

4. Compare whether the received data is the same as the sent data and exit the sample program.

4.7.3.2 Operations Overview of Slave Sample

1. Initialize the UART ports with the values below:
   - Communication speed   : 500kbps
   - Data length   : 8bit
   - Stop bit   : 1bit
   - Parity   : None

2. Receive the 27byte of data from the master.

3. Send the data from "A (0x41)" to "Z (0x5A)" and "¥0" to the master.

4. Compare whether the received data is the same as the sent data and exit the sample program.

4.8   SPI (SPI)

4.8.1   Sample Software Specification

The SPI sample program works as follows by using SPI.

- Send data by using SPI.
- Receive data by using SPI.

4.8.2   Hardware Conditions

This sample program runs in the IOSC built-in oscillating circuit (8MHz) in the microcontroller.

Connect the evaluation boards where the SPI master/slave sample programs are installed. Launch the slave sample program first. Connect each port as follows:
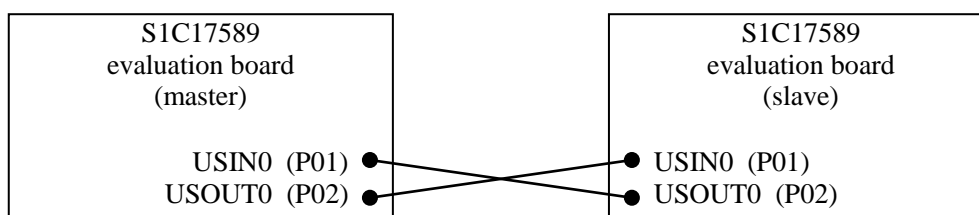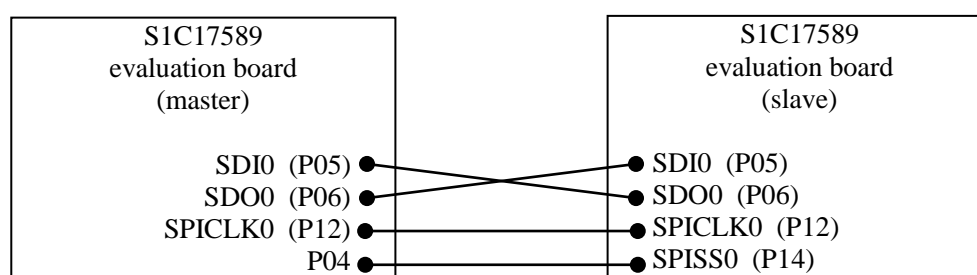

Figure 4.8.1 Hardware connection diagram for SPI master/slave sample programs

4.8.3   Operations Overview

4.8.3.1   Operations Overview of Master Sample

1.   Initialize the SPI ports with the values below:
- Communication speed   : 2Mbps
- Data length   : 8bit
- Data format   : MSB first

2.   Send the data from "A (0x41)" to "Z (0x5A)" and "¥0" to the slave.

3.   Receive the 27byte of data from the slave.

4.   Compare whether the received data is the same as the sent data and exit the sample program.

4.8.3.2   Operations Overview of Slave Sample

1.   Initialize the SPI ports with the values below:
- Communication speed   : 2Mbps
- Data length   : 8bit
- Data format   : MSB first

2.   Receive the 27byte of data from the master.

3.   Send the data from "A (0x41)" to "Z (0x5A)" and "¥0" to the master.

4.   Compare whether the received data is the same as the sent data and exit the sample program.

4.9 I2C (I2C)

4.9.1 Sample Software Specification

The I2C sample program works as follows by using I2C.

- Send data by using I2C.
- Receive data by using I2C.

4.9.2 Hardware Conditions

This sample program runs in the IOSC built-in oscillating circuit (8MHz) in the microcontroller.

Connect the evaluation boards where the I2C master/slave sample programs are installed. Launch the slave sample program first. Connect each port as follows:
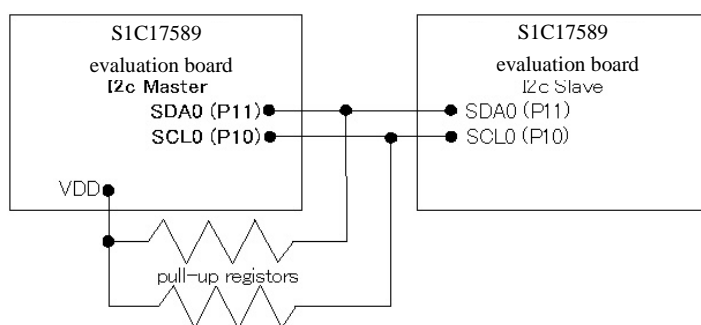

Figure 4.9.1 Hardware connection diagram for I2C master/slave sample programs

4.9.3 Operations Overview

4.9.3.1 Operations Overview of Master Sample

1. Initialize the I2C port with the value below:
     - Communication speed : 400kbps

2. Send the data from "A (0x41)" to "Z (0x5A)" and "¥0" to the slave.

3. Receive the 27byte of data from the slave.

4. Compare whether the received data is the same as the sent data and exit the sample program.

4.9.3.2 Operations Overview of Slave Sample

1. Initialize the I2C ports with the values below:
     - Address mode : 7bit
     - Slave address : 0x2A
     - General call : Invalid

2. Receive the 27byte of data from the master.

3. Send the data from "A (0x41)" to "Z (0x5A)" and "¥0" to the master.

4. Compare whether the received data is the same as the sent data and exit the sample program.

4.10 Power Supply Voltage Detection Circuit (SVD)

4.10.1 Sample Software Specification

The power supply voltage detection circuit sample program works as follows by using the power supply voltage detection circuit.

• In the power supply voltage detection circuit, compare the power supply voltage and the comparison voltage.

4.10.2 Hardware Conditions

This sample program runs in the OSC1 built-in oscillating circuit (32.768 kHz) in the microcontroller. Before the use of this sample program, connect the board as follows:
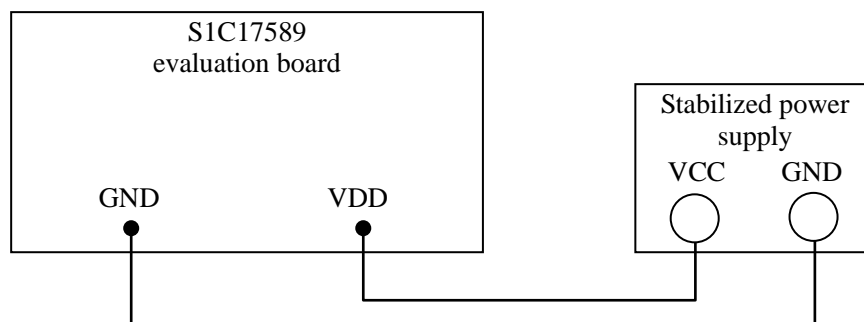
Figure 4.10.1 Hardware connection diagram for power supply voltage detection circuit sample program

4.10.3 Operations Overview

1. Set and start real time clock so that an interrupt occurs every second.

2. Set and start the power supply voltage detection circuit so that the voltage drop can be detected at 2.7V or less voltage.

3. Check every second whether the voltage drop occurred.

4. Measure the number of voltage drops for 10seconds and exit the sample program.

4.11 10-bit A/D Convertor (ADC10A)

4.11.1 Sample Software Specification

The 10-bit A/D convertor sample program works as follows by using the A/D convertor.

- Executes A/D conversion with 16-bit timer underflow trigger.
- Executes A/D conversion with software trigger.

4.11.2 Hardware Conditions

This sample program runs in the IOSC built-in oscillating circuit (8MHz) in the microcontroller. Before the use of this sample program, connect the board as follows:
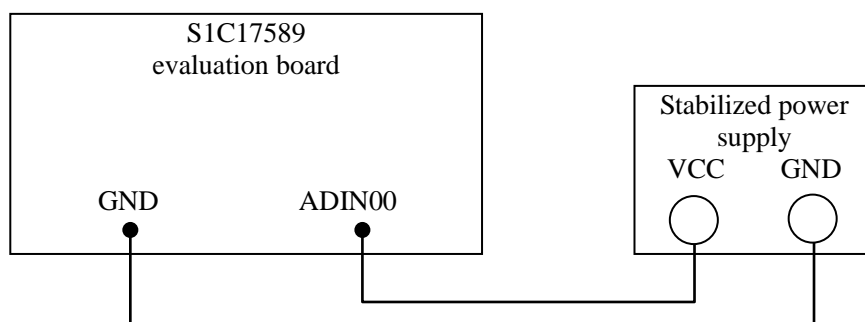

Fig. 4.11.1 10-bit A/D Convertor sample program exeternal connection circuit

4.11.3 Operations Overview

1   The A/D converter uses T16 Ch. 3 for a conversion clock. The converter repeats the behavior that the A/D conversion is triggered by an underflow of T16 Ch. 3 occurring every second and an interrupt occurs upon termination of the conversion. This operation is repeated.

2   When an overwrite error interrupt occurs in the A/D converter, the A/D conversion is retried. When an A/D conversion completion interrupt occurs, the program reads and saves the ADIN00 value. This operation is repeated ten times.

3   The converter repeats the behavior that the A/D conversion is triggered by an software with continuous mode occurring every second and an interrupt occurs upon termination of the conversion. This operation is repeated.

4   When an overwrite error interrupt occurs in the A/D converter, the A/D conversion is retried. When an A/D conversion completion interrupt occurs, the program reads and saves the ADIN00 value. This operation is repeated ten times.

4.12   IR Remote Controller (REMC2)

4.12.1 Sample Software Specification

The REMC2 sample program generates signals for infrared ray remote-controller by using REMC2. This sample program performs the following operations by using REMC2.

- Move to a sleep mode after initialization and wait for interrupts triggered by inputing a Low level signal to ports P04 to P07.
- When an interrupt is caused by P04 to P07, analyze which port was inputed and accordingly generate a signal (frame signal) for infrared ray remote-controller on REMC2. Then, activate the T16 Ch.0 timer and check the switch status after about 108ms.
- If a Low level signal was inputed to the same port consecutively when an interrupt occurred at the T16 Ch.0 timer, generate a repeat signal and set the T16 Ch.0 timer so that the port status is checked again after about 108ms. Otherwise, stop the T16 Ch.0 timer and REMC2, enter into a sleep mode, and wait for interrupts triggered by ports P04 to P07.

4.12.2 Hardware Conditions

This sample program is for the S1C17589 evaluation board (SVT17mini17589). For information on a connection to the circuit as shown in Figure 4.12.1, refer to the SVT17mini589's manual.
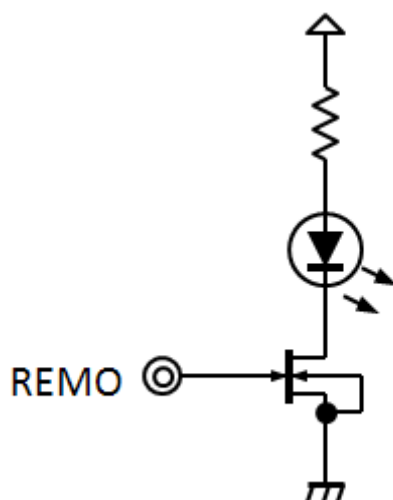
Figure 4.12.1 Infrared-emitting diode connection circuit example for REMC2 sample program

4.12.3 Operations Overview

1   As communication formats for infrared ray remote-controller, there are NEC, Association for Electric Home Appliances and SONY formats, and this sample software uses the NEC format.
2   The NEC format outline is as follows and Figure 4.12.2 also shows the NEC format.

Carrier: Infrared ray ($\lambda p = 940nm$)
Sub carrier: fsc = 38kHz, 1/3duty
$T = 562\mu s$
Fixed length frame (32bit)
16bit customer code
8bit data + 8bit reverse data

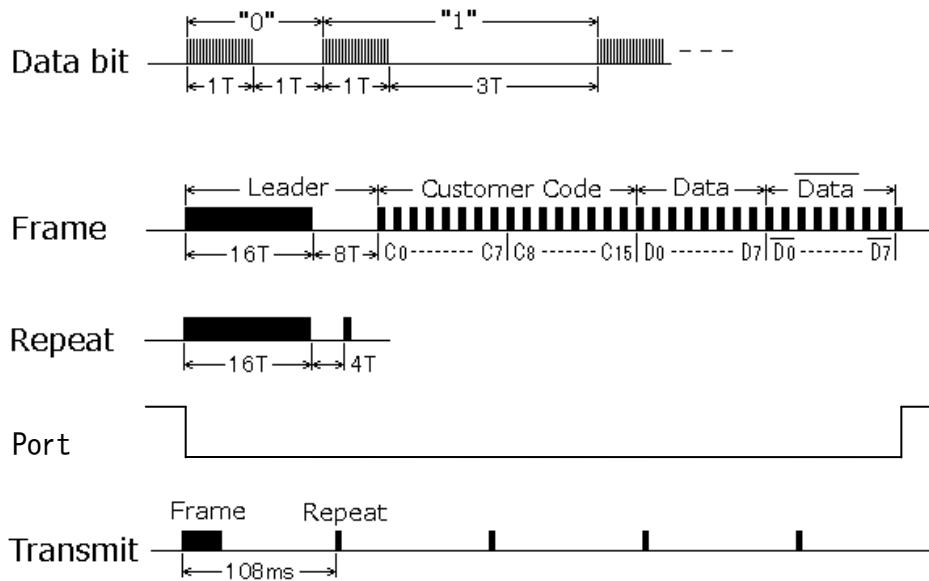Figure 4.12.2   Signal waveform of NEC format

3   REMC2 determines the data waveform shape by two parameters: APLEN (data signal duty) and DBLEN (data signal cycle). Each of APLEN and DBLEN defines the waveform shape with 34 frame signals and 2 repeat signals and allocates the leader, customer code, and repeat signals common to push switches SW1 to SW4 at initialization. Since the 16 values at data part may vary according to P04 to P07, the data values are allocated when an interrupt occurs by inputing a Low level signal to P04 to P07.

4   This sample program uses the IOSC behavior at 8MHz. Set a GPIO mode for ports P04 to P07, allocate ports for REMO2, initialize T16 Ch.0, and enter into a sleep mode to wait for interrupts caused by inputing a Low level signal to P04 to P07.

5   When an interrupt occurs, analyze which port was inputed and allocate APLEN and DBLEN values in data part according to the port inputed. Since REMC2 uses a buffer mode, activate REMC2 after initialization. Also, activate T16 Ch.0 and generate a T16 Ch.0 interrupt after about 108ms.

6   When a compare DB interrupt occurs at REMC2, write the following APLEN and DBLEN data into register. When writing last but one piece of data, change REMC2 to a one-shot mode to prevent the whisker-like signal output. At the last data, stop the REMC2 operation.

7   When a T16 Ch.0 interrupt occurred, check whether a Low level signal was inputed to the same port P04 to P07. If the same port was inputed, set the register values of REMC2 so that a repeat waveform is generated, initialize REMC2 before activation, and activate REMC2. Otherwise, stop the T16 Ch.0 operation.

Revision History

<div align="right">Attachment-1</div>

Rev. No.	Date	Page	Category	Contents
Rev. 1.2	2016/1/25	All	new	

EPSON

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.

7F, Jinbao Bldg., No.89 Jinbao St.,
Dongcheng District,
Beijing 100005, CHINA
Phone: +86-10-8522-1199   FAX: +86-10-8522-1125

SHANGHAI BRANCH

7F, Block B, Hi-Tech Bldg., 900 Yishan Road,
Shanghai 200233, CHINA
Phone: +86-21-5423-5577   FAX: +86-21-5423-4677

SHENZHEN BRANCH

12F, Dawning Mansion, Keji South 12th Road,
Hi-Tech Park, Shenzhen 518057, CHINA
Phone: +86-755-2699-3828   FAX: +86-755-2699-3838

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.
MICRODEVICES OPERATIONS DIVISION

Device Sales & Marketing Department
421-8, Hino, Hino-shi, Tokyo 191-8501, JAPAN
Phone: +81-42-587-5816   FAX: +81-42-587-5117