EPSON
EXCEED YOUR VISION

S1C17 Family Application Note

S1C17600 Series Peripheral Circuit Sample Software

SEIKO EPSON CORPORATION

Rev.1.0

Table of Contents

1. Overview

This manual describes the usage method of the S1C17600 series sample software and the operations of the sample software.

The purpose of the S1C17600 series sample software is to demonstrate the usage example of each peripheral circuit built into the S1C17600 series microcontroller.

The S1C17600 series sample software is offered on a per model basis for ease of installation but the basic operations of each model are the same.

Each model information, each technical manual, and the S5U1C17001C manual should be viewed together.

1.1 Operating Environment

When running the S1C17600 sample software, prepare the following materials.

- Board with S1C17600 mounted
- S5U1C17001H (hereinafter referred to as ICD mini)
- S5U1C17001C (hereinafter referred to as GNU17)

Note: This sample software has been checked for operations on GNU17 v1.5.0.

2. Explanation on Sample Software

This chapter describes the file configuration and execution method for the S1C17600 series sample software.

The S1C17600 series sample software comprises "sample software" that checks the operations of each peripheral circuit and "sample drivers" which are the sample drivers for the respective peripheral circuits.

2.1 Directory Structure and File Structure

The directory structure of the S1C17600 series sample software is shown below.

```
[s1c176xx]
   │
   ├GNU17 project files, etc
   ├...
   ├...
   │
   │
   ├─ [mcu]
   │      ├─ s1c17602_peripheral.h
   │      ...
   │      ├─ s1c176xx.h
   │      └─ boot.h            boot.c
   │
   ├─ [apl]
   │      ├─ main.h            main.c
   │      ├─ osc_main.c
   │      ├─ port_main.c
   │      ...
   │      ├─ t16_ main.c
   │      ├─ lcd_ main.c
   │      └─ svd_main.c
   │
   ├─ [driver]
   │      ├─ osc.h             osc.c
   │      ├─ clg.h             clg.c
   │      ├─ psc.h             psc.c
   │      ├─ port.h            port.c
   │      ...
   │      ├─ lcd.h             lcd.c
   │      └─ svd.h             svd.c
   │
   └─ [common]
          ├─ osc_api.h
          ├─ clg_api.h
          ├─ psc_api.h
          ├─ port_api.h
          ...
          ├─ lcd_api.h
          └─ svd_api.h
```

Figure 2.1 Block diagram for S1C17600 series sample software directories

(1) "s1c176xx" directory

This directory contains the files related to the GNU17 project and the directory where the source code for the sample software is stored.

(2) "mcu" directory

It contains the microcontroller's initialization process and the files that define the information that is model dependent.

- Header file that defines the register addresses and others of the target model (s1c17602_peripheral.h, etc)
- Header file common to all models (s1c176xx.h)
- Initialization file (boot.c)

(3) "apl" directory

It contains the sample software for each peripheral circuit as well as the header files that define the constants used in the sample software.

- Header file for each peripheral circuit (xxx.h)
- Sample software for each peripheral circuit (xxx.c)

(4) "driver" directory

It contains the sample driver for each peripheral circuit.

- Header file that defines the bit addresses and the register addresses of each peripheral circuit (xxx.h)
- Program for each peripheral circuit (xxx.c)

(5) "common" directory

It contains the header files that define the prototypes of the functions offered to external parts by the sample drivers for the respective peripheral circuits.

- Header file that defines the argument constants and function prototypes offered to external parts by the sample drivers for the respective peripheral circuits (xxx.h)

The software that uses the sample drivers includes the header files found in the "common" directory and calls the sample drivers' functions.

2.2 Execution Method

Execute the S1C17600 series sample software through the following steps.

(1) Import the project

Start up GNU17, and import the S1C17600 series sample software's project.

For details of project import method, refer to S5U1C17001C manual's "3. Software Development Steps."

(2) Build the project

Build the S1C176xx project on GNU17.

For details of build method, refer to S5U1C17001C manual's "5. GNU17 IDE."

(3) Connect ICD mini

Connect ICD mini to PC and development board, and turn on the development board's power supply.

(4) Load and execute the program using the debugger

Start up the debugger by pressing the GNU17's [External Tools] button, and press the debugger's [Continue] button.

The program is loaded onto S1C17600, and the program starts.

For details of debugger usage method, refer to S5U1C17001C manual's "10. Debugger."

2.3 Sample Software Menu

When the sample software is started up, the menu screen is displayed to Simulated I/O (hereinafter, SimI/O) of GNU17.

When the program number is entered and the [Enter] key is pressed, the selected sample software starts up.

For the details of each sample software, refer to Chapter 3.

```
1.Port                  2.OSC
3.16bit timer           4.8bit timer
...
Please input number.
>
```

Figure 2.2 Menu screen display example

2.4 Specific Module's Build Method

S1C176xx sample software's multiple programs are distributed in the built condition.

By modifying the sample software's source code, it is possible to build only the sample software for the required peripheral module.

The steps are shown below.

(1) File to be modified

Modify the definition header by model.

For example, modify the s1c17602_peripheral.h file in the case of S1C17602 sample software.

(2) Correction locations

Correct the following places at the bottom of the file.

```
//#undef PE_PORT
//#undef PE_OSC
//#undef PE_T16
//#undef PE_T8F
//#undef PE_T16E
//#undef PE_T8OSC1
//#undef PE_CT
//#undef PE_SWT
//#undef PE_WDT
#undef   PE_UART
#undef   PE_UART_OSC3
#undef   PE_UART_IOSC
#undef   PE_SPI
#undef   PE_SPI_MASTER
#undef   PE_SPI_SLAVE
#undef   PE_I2CM
#undef   PE_I2CS
#undef   PE_LCD
#undef   PE_SVD
#undef   PE_RFC
#undef   PE_ADC
#undef   PE_CURRENT_MEASURE
#undef   PE_SLEEP_HALT
```

Figure 2.3 Example of modifying specific module's definition

For example, if only building the I/O port sample software, disable the "#undef PE_PORT" definition and enable definitions other than "#undef PE_XXX."

If the peripheral module's sample software to be built uses another peripheral module, it is also necessary to build the sample software of the peripheral module that is used.

For example, the UART (OSC3) sample software uses 8-bit timer, and when building the UART (OSC3) sample software, it is necessary to disable the definitions "#undef PE_UART_OSC3" and "#undef PE_T8F."

3. Details of Sample Software Functions

This chapter describes the details of the S1C17600 series sample software's functions.

3.1 I/O Ports (P)

3.1.1 Sample software specifications

This sample software performs the following operations using the I/O ports.

- Sets the ports to input interrupt, and detects for input signal level becoming LOW.
- Sets the ports to output, and sends out the HIGH or LOW signal.

The port settings and port names are shown below.

Table 3.1 List of I/O port settings

Setting	Port Name
Input interrupt port	P02 P15 P03
Output port	P10 P11

Note: Port settings may be different depending on model. Refer to the source code of each model.

3.1.2 Hardware conditions

This software sample operates under the oscillatable conditions of OSC1 and OSC3.

For the connection method of oscillator, refer to "Oscillation Circuit (OSC)" of each technical manual.

Use this sample software with each port of the microcontroller connected as shown below.

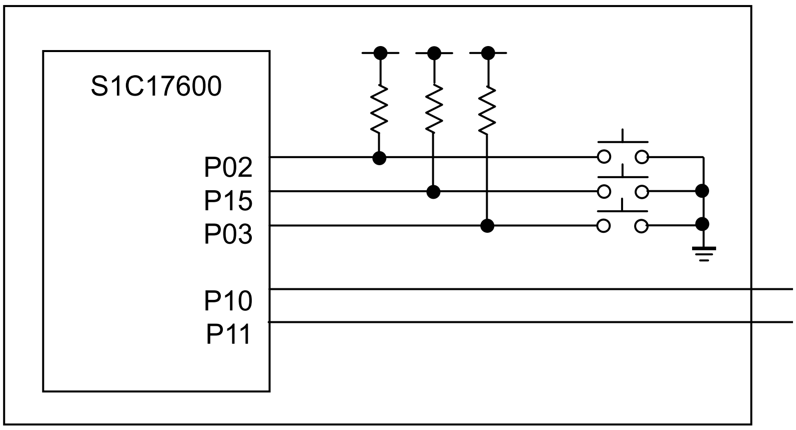

Figure 3.1 I/O ports (P) sample software's hardware connection diagram

3.1.3 Operations overview

(1) Sample software operations overview

- When the input signal of port P02 is made LOW, "P02 Interrupt" is displayed to SimI/O and the output of port P10 is reversed. (Set to LOW if HIGH, and set to HIGH if LOW.)
- When the input signal of port P15 is made LOW, "P15 Interrupt" is displayed to SimI/O and the output of port P11 is reversed. (Set to LOW if HIGH, and set to HIGH if LOW.)

```
<<< Port demonstration start >>>
*** P02 Interrupt ***
*** P15 Interrupt ***

<<< Port demonstration finish >>>
```

Figure 3.2 I/O ports (P) sample software's screen display example

(2) Stop method for sample software

When the input signal of port P03 is made LOW, the sample software ends and it returns to the menu screen.

3.2 Oscillation Circuit (OSC)

3.2.1 Sample software specifications

This sample software performs the following operations using the oscillation circuit.

- Performs IOSC oscillation and stopping.
- Performs OSC1 oscillation and stopping.
- Performs OSC3 oscillation and stopping.
- Switches system clock from IOSC to OSC3.
- Switches system clock from OSC3 to OSC1.
- Switches system clock from OSC1 to IOSC.

3.2.2 Hardware conditions

This software sample operates under the oscillatable conditions of OSC1 and OSC3.

3.2.3 Operations overview

(1) Sample software operations overview

- This sample software starts operating under the condition of using IOSC.
- After displaying '1', '2', '3', ..., '9' to SimI/O at a fixed interval, starts OSC3 oscillation, switches system clock from IOSC to OSC3, and stops IOSC.
- After displaying '1', '2', '3', ..., '9' to SimI/O at a fixed interval, starts OSC1 oscillation, switches system clock from OSC3 to OSC1, and stops OSC3.
- After displaying '1', '2', '3', ..., '9' to SimI/O at a fixed interval, starts IOSC and OSC3 oscillation, switches system clock from OSC1 to IOSC, and stops OSC1 and OSC3.
- Next, displays '1', '2', '3', ..., '9' to SimI/O at a fixed interval.

```
<<< OSC demonstration start >>>
IOSC *** 1 ***
IOSC *** 2 ***
...
IOSC *** 9 ***
*** Change from IOSC to OSC3 ***
OSC3 *** 1 ***
OSC3 *** 2 ***
...
OSC3 *** 9 ***
<<< OSC demonstration finish >>>
```

Figure 3.3 Oscillation circuit (OSC) sample software's screen display example

(2) Stop method for sample software

When all the operations described in the above "Sample software operations overview" are completed, the sample software ends and it returns to the menu screen.

3.3 16-Bit Timer (T16)

3.3.1 Sample software specifications

This sample software performs the following operations using the 16-bit timer.

- 16-bit timer interrupt is made to occur, and the counter value of the timer is acquired.
- While waiting for interrupt, the power consumption is reduced by making the CPU go into the halt mode.

3.3.2 Hardware conditions

This software sample operates under the oscillatable conditions of OSC1 and OSC3.

3.3.3 Operations overview

(1) Sample software operations overview

- Starts 16-bit timer interrupt, and makes CPU go into halt mode.
- When 16-bit timer interrupt occurs, the CPU's halt mode is canceled, the 16-bit timer's counter value is stored to internal variable, and the CPU is made to go into halt mode again.
- Upon the 10th occurrence of 16-bit timer interrupt, stops the 16-bit timer, and displays the counter value of each interrupt occurrence to SimI/O.

```
<<< T16 timer demonstration start >>>
*** T16 interrupt 1 time, count data at this time : 32 ***
*** T16 interrupt 2 time, count data at this time : 32 ***
*** T16 interrupt 3 time, count data at this time : 32 ***
*** T16 interrupt 4 time, count data at this time : 32 ***
...
*** T16 interrupt 10 time, count data at this time : 32 ***
<<< T16 timer demonstration finish >>>
```

Figure 3.4 16-bit timer (T16) sample software's screen display example

(2) Stop method for sample software

When all the operations described in the above "Sample software operations overview" are completed, the sample software ends and it returns to the menu screen.

3.4 8-Bit Timer (T8F)

3.4.1 Sample software specifications

This sample software performs the following operations using the 8-bit timer.

- 8-bit timer interrupt is made to occur, and the counter value of the timer is acquired.
- While waiting for interrupt, the power consumption is reduced by making the CPU go into the halt mode.

3.4.2 Hardware conditions

This software sample operates under the oscillatable conditions of OSC1 and OSC3.

3.4.3 Operations overview

(1) Sample software operations overview

- Starts 8-bit timer interrupt, and makes CPU go into halt mode.
- When 8-bit timer interrupt occurs, the CPU's halt mode is canceled, the 8-bit timer's counter value is stored to internal variable, and the CPU is made to go into halt mode again.
- Upon the 10th occurrence of 8-bit timer interrupt, stops the 8-bit timer, and displays the counter value of each interrupt occurrence to SimI/O.

```
<<< T8 timer demonstration start >>>
*** T8 interrupt 1 time, count data at this time : 128 ***
*** T8 interrupt 2 time, count data at this time : 128 ***
*** T8 interrupt 3 time, count data at this time : 128 ***
*** T8 interrupt 4 time, count data at this time : 128 ***
...
*** T8 interrupt 10 time, count data at this time : 128 ***
<<< T8 timer demonstration finish >>>
```

Figure 3.5 8-bit timer (T8F) sample software's screen display example

(2) Stop method for sample software

When all the operations described in the above "Sample software operations overview" are completed, the sample software ends and it returns to the menu screen.

3.5 PWM Timer (T16E)

3.5.1 Sample software specifications

This sample software performs the following operations using the PWM timer.

- PWM timer compare A match interrupt is made to occur, and the counter value of the timer is acquired.
- PWM timer compare B match interrupt is made to occur, and the counter value of the timer is acquired.
- PWM waveform is outputted to TOUT3 terminal and TOUTN3 terminal.
- While waiting for interrupt, the power consumption is reduced by making the CPU go into the halt mode.

Note: Terminal name TOUT3/TOUTN3 may be different depending on model. Refer to the source code of each model.

3.5.2 Hardware conditions

This software sample operates under the oscillatable conditions of OSC1 and OSC3.

3.5.3 Operations overview

(1) Sample software operations overview

- Enables compare A match interrupt and compare B match interrupt, starts PWM timer, and makes CPU go into halt mode.
- When compare A match interrupt and compare B match interrupt occur, the CPU's halt mode is canceled, the PWM timer's counter value is stored to internal variable, and the CPU is made to go into halt mode again.
- Upon the 5th occurrence of compare B match interrupt, stops the PWM timer, and displays the interrupt type and counter value to SimI/O.

```
<<< PWM timer demonstration start >>>
*** PWM compare A interrupt : 633 ***
*** PWM compare B interrupt : 126 ***
*** PWM compare A interrupt : 633 ***
...
*** PWM Interrupt B interrupt: 126 ***
<<< PWM timer demonstration finish >>>
```

Figure 3.6 PWM timer (T16E) sample software's screen display example

(2) Stop method for sample software

When all the operations described in the above "Sample software operations overview" are completed, the sample software ends and it returns to the menu screen.

3.6 8-Bit OSC1 Timer (T8OSC1)

3.6.1 Sample software specifications

This sample software performs the following operations using the 8-bit OSC1 timer.

- 8-bit OSC1 timer interrupt is made to occur, and the counter value of the timer is acquired.
- While waiting for interrupt, the power consumption is reduced by making the CPU go into the halt mode.

3.6.2 Hardware conditions

This software sample operates under the oscillatable conditions of OSC1 and OSC3.

3.6.3 Operations overview

(1) Sample software operations overview

- Starts 8-bit OSC1 timer interrupt, and makes CPU go into halt mode.
- When 8-bit OSC1 timer interrupt occurs, the CPU's halt mode is canceled, the 8-bit OSC1 timer's counter value is stored to internal variable, and the CPU is made to go into halt mode again.
- Upon the 10th occurrence of 8-bit OSC1 timer interrupt, stops the 8-bit OSC1 timer, and displays the counter value of each interrupt occurrence to SimI/O.

```
<<< T8OSC1 timer demonstration start >>>
*** T8OSC1 Interrupt 1 time, count data at this time : 10 ***
*** T8OSC1 Interrupt 2 time, count data at this time : 11 ***
*** T8OSC1 Interrupt 3 time, count data at this time : 10 ***
*** T8OSC1 Interrupt 4 time, count data at this time : 10 ***
...
*** T8OSC1 Interrupt 10 time, count data at this time : 10 ***
<<< T8OSC1 timer demonstration finish >>>
```

Figure 3.7 8-bit OSC1 timer (T8OSC1) sample software's screen display example

(2) Stop method for sample software

When all the operations described in the above "Sample software operations overview" are completed, the sample software ends and it returns to the menu screen.

3.7 Clock Timer (CT)

3.7.1 Sample software specifications

This sample software performs the following operations using the clock timer.

- Clock timer interrupt is made to occur, and the elapsed time is computed.
- While waiting for interrupt, the power consumption is reduced by making the CPU go into the halt mode.

3.7.2 Hardware conditions

This software sample operates under the oscillatable conditions of OSC1 and OSC3.

3.7.3 Operations overview

(1) Sample software operations overview

- Starts the clock timer, and makes CPU go into halt mode.
- When clock timer interrupt occurs, the CPU's halt mode is canceled, the elapsed time since the start of program is computed and displayed to SimI/O, and the CPU is made to go into halt mode again.
- Upon the 10th occurrence of clock timer interrupt, stops the clock timer.

```
<<< Clock timer demonstration start >>>
*** 0.5 sec ***
*** 1.0 sec ***
*** 1.5 sec ***
...
*** 5.0 sec ***
<<< Clock timer demonstration finish >>>
```

Figure 3.8 Clock timer (CT) sample software's screen display example

(2) Stop method for sample software

When all the operations described in the above "Sample software operations overview" are completed, the sample software ends and it returns to the menu screen.

3.8 Stopwatch Timer (SWT)

3.8.1 Sample software specifications

This sample software performs the following operations using the stopwatch timer.

- Stopwatch timer interrupt is made to occur, and the elapsed time is computed.

3.8.2 Hardware conditions

This software sample operates under the oscillatable conditions of OSC1 and OSC3.

3.8.3 Operations overview

(1) Sample software operations overview

- By inputting the numerals 1 to 9 and pressing the [ENTER] key, it is possible to specify the number of times for the interrupt.
- Starts the stopwatch timer and upon the 1Hz stopwatch timer interrupt occurrence reaching the specified number of times, displays the elapsed time to SimI/O and stops the stopwatch timer.

```
<<< Stop watch timer demonstration start >>>
Please input time 1-9[sec]
4
Start stopwatch timer...
4 sec passed
<<< Stop watch timer demonstration finish >>
```

Figure 3.9 Stopwatch timer (SWT) sample software's screen display example

(2) Stop method for sample software

When all the operations described in the above "Sample software operations overview" are completed, the sample software ends and it returns to the menu screen.

3.9 Watchdog Timer (WDT)

3.9.1 Sample software specifications

This sample software performs the following operations using the watchdog timer.

- NMI interrupt is made to occur through the watchdog timer.

3.9.2 Hardware conditions

This software sample operates under the oscillatable conditions of OSC1 and OSC3.

3.9.3 Operations overview

(1) Sample software operations overview

- Starts the watchdog timer and 16-bit timer.
- When the 16-bit timer interrupt occurs, clears the watchdog timer.
- Upon the 10th occurrence of 16-bit timer interrupt, stops the 16-bit timer.
- When the NMI interrupt from watchdog timer occurs, displays message to SimI/O.

```
<<< Watchdog timer demonstration start >>>
*** T16 timer : reset watchdog timer ***
*** T16 timer : reset watchdog timer ***
*** T16 timer : reset watchdog timer ***
...
*** T16 timer : reset watchdog timer ***
*** stop T16 timer ***
*** NMI occurred ***
<<< Watchdog timer demonstration finish >>>
```

Figure 3.10 Watchdog timer (WDT) sample software's screen display example

(2) Stop method for sample software

When all the operations described in the above "Sample software operations overview" are completed, the sample software ends and it returns to the menu screen.

3.10 UART Using OSC3

3.10.1 Sample software specifications

This sample software performs the following operations using the UART.

- Sends data using the UART.
- Receives data using the UART.

3.10.2 Hardware conditions

This software sample operates under the oscillatable conditions of OSC1 and OSC3.

Use this sample software with each port of the microcontroller connected as shown below.

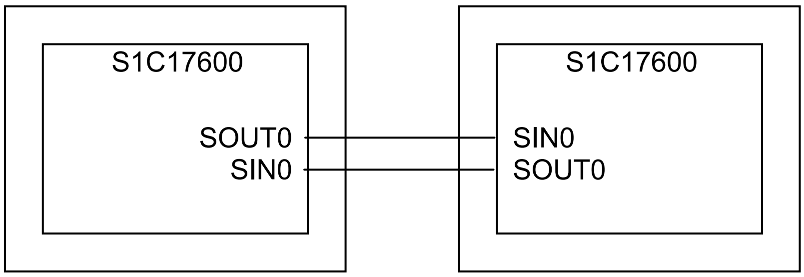

Figure 3.11 Sample software's hardware connection diagram for UART using OSC3

3.10.3 Operations overview

(1) Sample software operations overview

- Initializes the UART port to 115200bps communication speed, 8-bit data length, 1-bit stop bit, and no parity.
- Continues to send "0x7F" until the connection confirmation flag "0x7F" is received.
- When the connection confirmation flag is received, stops sending "0x7F," and sends data using ASCII codes 0x21~0x7E to the UART port.
- Sends all data, and when 34-byte data is received, displays the received data to SimI/O.

```
<<< UART OSC3 demonstration start >>>
waiting connection.
connected.
*** sent data ***
*** received data ***
ABCDEFG...
<<< UART OSC3 demonstration finish >>>
```

Figure 3.12 Sample software's screen display example for UART using OSC3

(2) Stop method for sample software

When all the operations described in the above "Sample software operations overview" are completed, the sample software ends and it returns to the menu screen.

3.11 UART Using IOSC

3.11.1 Sample software specifications

This sample software performs the following operations using the UART.

- Compares the IOSC clock and OSC1 counter values, and computes the IOSC frequency.
- Sets IOSC as UART clock.
- Sends data using the UART.
- Receives data using the UART.

3.11.2 Hardware conditions

The hardware conditions are the same as that of UART sample software using OSC3.

3.11.3 Operations overview

(1) Sample software operations overview

- 16-bit timer using IOSC and 8-bit timer using OSC1 are made to operate, and the IOSC oscillation frequency is computed.
- Based on the computed IOSC oscillation frequency, initializes the UART port to 115200bps communication speed, 8-bit data length, 1-bit stop bit, and no parity.
- Continues to send "0x7F" until the connection confirmation flag "0x7F" is received.
- When the connection confirmation flag is received, stops sending "0x7F," and sends data using ASCII codes 0x21~0x7E to the UART port.
- Sends all data, and when 34-byte data is received, displays the received data to SimI/O.

```
<<< UART IOSC demonstration start >>>
waiting connection.
connected.
*** sent data ***
*** received data ***
ABCDEFG...
<<< UART IOSC demonstration finish >>>
```

Figure 3.13 Sample software's screen display example for UART using IOSC

(2) Stop method for sample software

When all the operations described in the above "Sample software operations overview" are completed, the sample software ends and it returns to the menu screen.

3.11.4 IOSC oscillation frequency calculation method

The steps for calculating the timer counter setting value when using UART with IOSC are shown below.

- Sets the T8OSC1 compare data to 8counts, and starts the T8OSC1 and T16 that use IOSC.
- When the T8OSC1 compare match interrupt occurs, stops T16.
- After stopping T16, reads the counter value and seeks the IOSC frequency.
- The quotient of (n×4096÷(div×bps)) is T8F_TR+1, and the remainder is TFMD. (n=IOSC counter value, div=T8F prescaler frequency division ratio's reciprocal, bps=UART bit rate)

3.12 SPI Master

3.12.1 Sample software specifications

This sample software performs the following operations using the SPI master.

- Sends 8-byte data to SPI slave.
- Receives 8-byte data from SPI slave.
- While waiting for interrupt, the power consumption is reduced by making the CPU go into the halt mode.

3.12.2 Hardware conditions

This software sample operates under the oscillatable conditions of OSC1 and OSC3.

For this sample software, use by connecting S1C17600 running SPI slave sample software as the SPI slave and connect each port of the microcontroller as shown below.

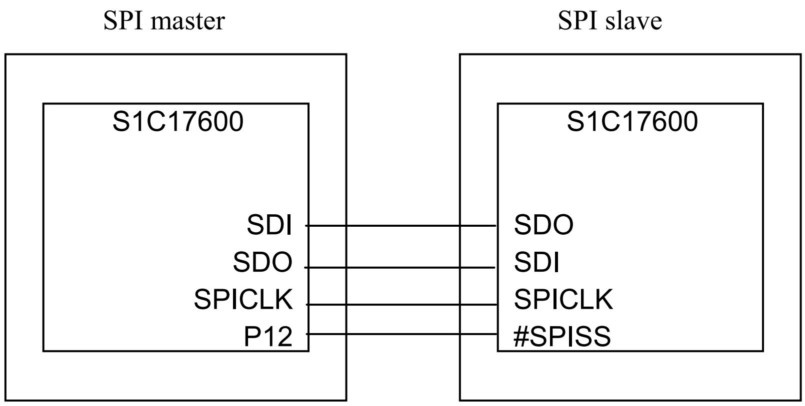

Figure 3.14 SPI master and slave sample software's hardware connection diagram

Note: Each terminal may be different depending on the model. Check with the source code.

3.12.3 Operations overview

(1) Sample software operations overview

- Performs initial setting of SPI master, and sends 8-byte ASCII data "FROM MST" to SPI slave.
- When sending of data to SPI slave ends, it waits for the [ENTER] key input.
- When [ENTER] key is pressed, SPI clock is outputted to SPI slave and it waits to receive data.
- When data is received from SPI slave, the received data is displayed to SimI/O.

```
<<< SPI master demonstration start >>>
Transmitted data : FROM MST
please press enter key

Received data : FROM SLV
<<< SPI master demonstration finish >>>
```

Figure 3.15 SPI master sample software's screen display example

(2) Stop method for sample software

When all the operations described in the above "Sample software operations overview" are completed, the sample software ends and it returns to the menu screen.

3.13 SPI Slave

3.13.1 Sample software specifications

This sample software performs the following operations using the SPI slave.

- Receives 8-byte data from SPI master.
- Sends 8-byte data to SPI master.
- While waiting for interrupt, the power consumption is reduced by making the CPU go into the halt mode.

3.13.2 Hardware conditions

The hardware conditions are the same as that of SPI master sample software.

For the sample software, use by connecting S1C17600 running SPI master sample software as the SPI master.

3.13.3 Operations overview

(1) Sample software operations overview

- Performs initial setting of SPI slave, and waits to receive data from SPI master.
- When data is received from SPI master, displays the received data to SimI/O and sends 8-byte ASCII data "FROM SLV" to SPI master.

```
<<< SPI slave demonstration start >>>
Received data : FROM MST
Transmitted data : FROM SLV
<<< SPI slave demonstration finish >>>
```

Figure 3.16 SPI slave sample software's screen display example

(2) Stop method for sample software

When all the operations described in the above "Sample software operations overview" are completed, the sample software ends and it returns to the menu screen.

3.14 I2C Master (I2CM)

3.14.1 Sample software specifications

This sample software performs the following operations using the I2C master.

- Sends data to I2C slave.
- Receives data from I2C slave.

3.14.2 Hardware conditions

This software sample operates under the oscillatable conditions of OSC1 and OSC3.

For this sample software, use by connecting the S1C17600 microcontroller running I2C slave sample software as the I2C slave and connect each port of the microcontroller as shown below.

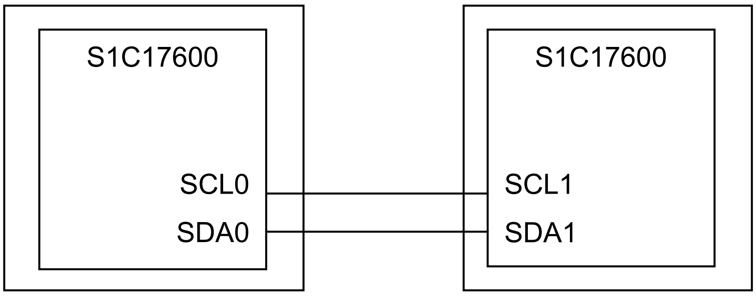

Figure 3.17 I2C master (I2CM) and slave (I2CS) sample software's hardware connection diagram

3.14.3 Operations overview

(1) Sample software operations overview

- Performs initial setting of I2C master, and sends 8-byte ASCII data "FROM MST" to I2C slave.
- When sending of data to I2C slave ends, it waits for the [ENTER] key input.
- When [ENTER] key is pressed, it waits to receive data from I2C slave.
- When data is received from I2C slave, the received data is displayed to SimI/O.

```
<<< I2C master demonstration start >>>
Transmitted data : FROM MST
please press enter key

Received data : FROM SLV
<<< I2C master demonstration finish >>>
```

Figure 3.18 I2C master (I2CM) sample software's screen display example

(2) Stop method for sample software

When all the operations described in the above "Sample software operations overview" are completed, the sample software ends and it returns to the menu screen.

3.15 I2C Slave (I2CS)

3.15.1 Sample software specifications

This sample software performs the following operations using the I2C slave.

- Receives data from I2C master.
- Sends data to I2C master.

3.15.2 Hardware conditions

The hardware conditions are the same as that of I2C master (I2CM) sample software.

For this sample software, use by connecting the S1C17600 microcontroller running I2C master (I2CM) sample software as the I2C master.

3.15.3 Operations overview

(1) Sample software operations overview

- Performs initial setting of I2C slave, and waits to receive data from I2C master.
- When data is received from I2C master, displays the received data to SimI/O and sends 8-byte ASCII data "FROM SLV" to I2C master.

```
<<< I2C slave demonstration start >>>
Received data : FROM MST
Transmitted data : FROM SLV
<<< I2C slave demonstration finish >>>
```

Figure 3.19 I2C slave (I2CS) sample software's screen display example

(2) Stop method for sample software

When all the operations described in the above "Sample software operations overview" are completed, the sample software ends and it returns to the menu screen.

3.16 LCD Driver (LCD)

3.16.1 Sample software specifications

This sample software performs the following operations using the LCD driver.

- Turns on all lamps and turns off all lamps.
- Turns on and turns off specified segments.

3.16.2 Hardware conditions

This software sample operates under the oscillatable conditions of OSC1 and OSC3.

Use this sample software with each port of the microcontroller connected as shown below.

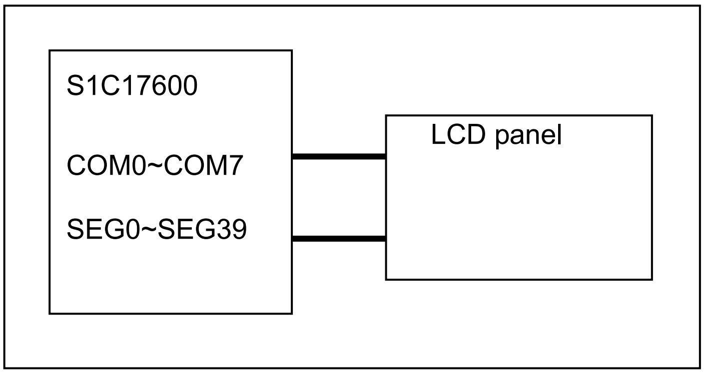

Figure 3.20 LCD driver (LCD) sample software's hardware connection diagram

3.16.3 Operations overview

(1) Sample software operations overview

- When [on] is inputted to SimI/O and [ENTER] key is pressed, turns on entire LCD.
- When [off] is inputted to SimI/O and [ENTER] key is pressed, turns off entire LCD.
- When [segon (SEG No.), (COM No.)] is inputted and [ENTER] key is pressed, turns on the specified segment.
- When [segoff (SEG No.), (COM No.)] is inputted and [ENTER] key is pressed, turns off the specified segment.

```
<<< LCD driver demonstration start >>>
on
off
segon 0,0
segoff 10,5
exit
<<< LCD driver demonstration finish >>>
```

Figure 3.21 LCD driver (LCD) sample software's screen display example

(2) Stop method for sample software

When [exit] is inputted and [ENTER] key is pressed, the sample software ends and it returns to the menu screen.

3.17 Power Source Voltage Detection Circuit (SVD)

3.17.1 Sample software specifications

This sample software performs the following operations using the power source voltage detection circuit (hereinafter, SVD circuit).

- Detects power source voltage using SVD circuit.

3.17.2 Hardware conditions

This software sample operates under the oscillatable conditions of OSC1 and OSC3.

Operate by setting the desired power source voltage.

3.17.3 Operations overview

(1) Sample software operations overview

- Detects power source voltage (VDD) using the SVD circuit, and displays the current VDD voltage to SimI/O. The comparative voltage is 1.8V to 3.2V.
- If the power source voltage is less than 1.8V or above 3.2V, "SVD interrupt did not occurred" is displayed to SimI/O.

```
<<< SVD demonstration start >>>
Vdd=2.5V
<<< SVD demonstration finish >>>
```

Figure 3.22 Power source voltage detection circuit (SVD) sample software's screen display example

Note: The detection voltage may be different depending on model. Refer to the source code of each model.

(2) Stop method for sample software

When all the operations described in the above "Sample software operations overview" are completed, the sample software ends and it returns to the menu screen.

3.18 R/F Converter (RFC)

3.18.1 Sample software specifications

This sample software performs the following operations using the R/F converter.

- Makes to oscillate in the DC oscillation mode for resistive sensor measurement, and acquires the counter value.

3.18.2 Hardware conditions

This software sample operates under the oscillatable conditions of OSC1 and OSC3.

Use this sample software with each port of the microcontroller connected as shown below.

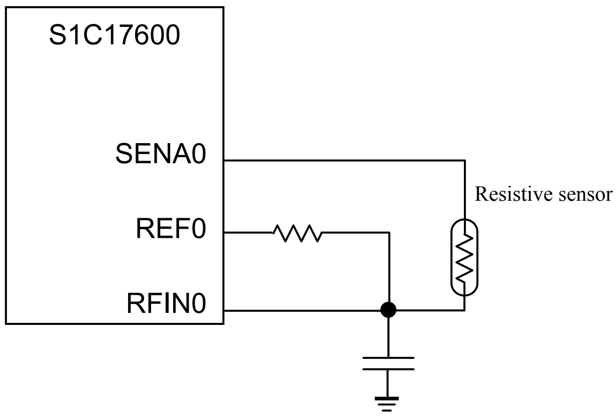

Figure 3.23 R/F converter (RFC) sample software's hardware connection diagram

3.18.3 Operations overview

(1) Sample software operations overview

- Sets the DC oscillation mode for resistive sensor measurement.
- Starts the reference oscillation, and upon end of oscillation, acquires the counter value and displays to SimI/O.
- Starts sensor A oscillation, and upon end of oscillation, acquires the counter value and displays to SimI/O.

```
<<< RFC demonstration start >>>
Reference
measurement counter : 0000
time base counter : 0000
Sensor A
measurement counter : 0000
time base counter : 0000
<<< RFC demonstration finish >>>
```

Figure 3.24 R/F converter (RFC) sample software's screen display example

(2) Stop method for sample software

When all the operations described in the above "Sample software operations overview" are completed, the sample software ends and it returns to the menu screen.

3.19 A/D Converter (ADC10)

3.19.1 Sample software specifications

This sample software performs the following operations using the A/D converter.

- Detects the end of A/D conversion interrupt, and acquires the A/D conversion result.
- While waiting for interrupt, the power consumption is reduced by making the CPU go into the halt mode.

3.19.2 Hardware conditions

This software sample operates under the oscillatable conditions of OSC1 and OSC3.

At the AIN0 terminal, use by applying the desired voltage within the possible voltage range for analog input.

3.19.3 Operations overview

(1) Sample software operations overview

- Sets the A/D converter.
- When the end of A/D conversion interrupt occurs, reads the A/D conversion result.
- Upon the 10th occurrence of the end of A/D conversion interrupt, displays the result to SimI/O.

```
<<< AD converter demonstration start >>>
AD conversion result : 0000
AD conversion result : 0001
AD conversion result : 0003
...
AD conversion result : 0003
<<< AD converter demonstration finish >>>
```

Figure 3.25 A/D converter (ADC10) sample software's screen display example

(2) Stop method for sample software

When all the operations described in the above "Sample software operations overview" are completed, the sample software ends and it returns to the menu screen.

3.20 Remote Controller Sending (REMC)

3.20.1 Sample software specifications

This sample software performs the following operations using the remote controller (hereinafter, REMC) peripheral circuit.

- Sends data using the REMC peripheral circuit.

3.20.2 Hardware conditions

This software sample operates under the oscillatable conditions of OSC1 and OSC3.

For this sample software, use with the S1C17600 running the remote controller receiving (REMC) sample software as the communications counterpart, and connect each port of the microcontroller as shown below.

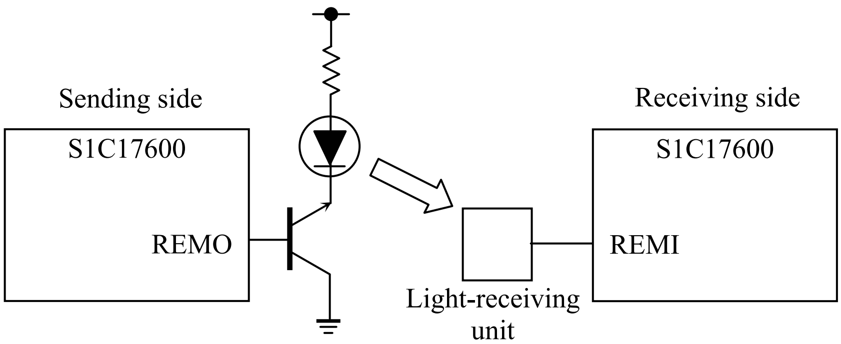

Figure 3.26 Remote controller (REMC) sample software's hardware connection diagram

3.20.3 Operations overview

(1) Sample software operations overview

- Sets the REMC peripheral circuit for data sending.
- Sends 10 items of data using the REMC peripheral circuit.
- Displays the send data to SimI/O, and sends the data.

```
<<< REMC transmit demonstration start >>>
Transmitted data : 05
Transmitted data : 10
Transmitted data : 15
...
<<< REMC transmit demonstration finish >>>
```

Figure 3.27 Remote controller sending (REMC) sample software's screen display example

(2) Stop method for sample software

When all the operations described in the above "Sample software operations overview" are completed, the sample software ends and it returns to the menu screen.

3.21 Remote Controller Receiving (REMC)

3.21.1 Sample software specifications

This sample software performs the following operations using the remote controller (hereinafter, REMC) peripheral circuit.

- Receives data using the REMC peripheral circuit.

3.21.2 Hardware conditions

The hardware conditions are the same as that of remote controller sending (REMC) sample software.

For this sample software, use with the S1C17600 microcontroller running the remote controller sending (REMC) sample software as the communications counterpart.

3.21.3 Operations overview

(1) Sample software operations overview

- Sets the REMC peripheral circuit for data receiving.
- Receives 10 items of data from the REMC peripheral circuit.
- When data is received from the REMC peripheral circuit, displays the received data to SimI/O.

```
<<< REMC receive demonstration start >>>
Received data : 05
Received data : 10
Received data : 15
...
<<< REMC receive demonstration finish >>>
```

Figure 3.28 Remote controller receiving (REMC) sample software's screen display example

(2) Stop method for sample software

When all the operations described in the above "Sample software operations overview" are completed, the sample software ends and it returns to the menu screen.

3.22 Electric Current Measurement

3.22.1 Sample software specifications

This sample software is a program for evaluating the following conditions.

- Sets CPU to sleep mode.
- Sets CPU to halt mode under the condition of only OSC1 being made to oscillate.
- Sets CPU to halt mode under the condition of OSC1/OSC3 being made to oscillate.
- Sets CPU to halt mode under the condition of OSC1/OSC3/IOSC being made to oscillate.

3.22.2 Hardware conditions

This software sample operates under the oscillatable conditions of OSC1 and OSC3.

3.22.3 Operations overview

(1) Sample software operations overview

- Under the condition of menu being displayed, when [1] is inputted and [ENTER] key is pressed, puts the CPU into sleep mode.
- Under the condition of menu being displayed, when [2] is inputted and [ENTER] key is pressed, puts the CPU into halt mode under the condition of only OSC1 being made to oscillate.
- Under the condition of menu being displayed, when [3] is inputted and [ENTER] key is pressed, puts the CPU into halt mode under the condition of only OSC1 and OSC3 being made to oscillate.
- Under the condition of menu being displayed, when [4] is inputted and [ENTER] key is pressed, puts the CPU into halt mode under the condition of OSC1, OSC3 and IOSC being made to oscillate.

```
<<< Current measurement demonstration start >>>
1. Sleep              2.Halt with OSC1
3.Halt with OSC1/3 4.Halt with OSC1/3/IOSC
Please input number.
>1
sleeping
```

Figure 3.29 Electric current measurement sample software's screen display example

(2) Stop method for sample software

When ending this sample software, reset the software by asserting the microcontroller's external reset terminal.

Note: The P clock is normally ON, and the measurement value may differ from the expected value. Use by correcting the source code to adjust to the desired electric current measurement conditions.

3.23 Sleep/Halt

3.23.1 Sample software specifications

This sample software performs the following operations.

- Makes the CPU go into halt mode by executing the halt command.
- Cancels CPU's halt mode by using 16-bit timer interrupt.
- Makes the CPU go into sleep mode by executing the sleep command.
- Cancels CPU's sleep mode by using port interrupt.

3.23.2 Hardware conditions

This software sample operates under the oscillatable conditions of OSC1 and OSC3.

Use this sample software with each port of the microcontroller connected as shown below.

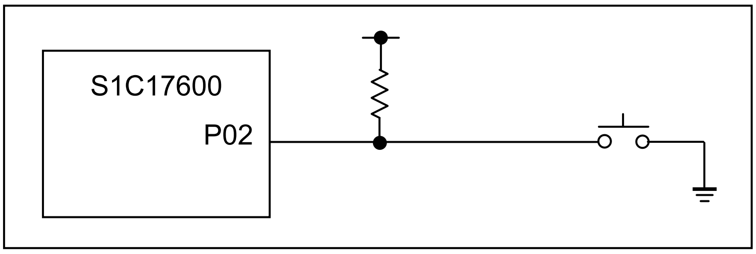

Figure 3.30 Sleep/halt mode switching sample software's screen display example

Note: Port settings may be different depending on model. Refer to the source code of each model.

3.23.3 Operations overview

(1) Sample software operations overview

- Starts the 16-bit timer, and makes CPU go into halt mode.
- When 16-bit timer interrupt occurs, cancels the halt mode and displays message to SimI/O.
- Upon the 5th occurrence of 16-bit timer interrupt, stops the 16-bit timer and makes CPU go into sleep mode.
- When P02 port becomes LOW, cancels the sleep mode.

```
<<< Sleep/halt demonstration start >>>
go to halt mode
return from halt mode
...
go to sleep mode
return from sleep mode
<<< Sleep/halt demonstration finish >>>
```

Figure 3.31 Sleep/halt mode switching sample software's screen display example

(2) Stop method for sample software

When all the operations described in the above "Sample software operations overview" are completed, the sample software ends and it returns to the menu screen.

4. List of Sample Driver Functions

Here, each sample driver of the sample software is listed.

4.1 I/O Ports (P)

Table 4.1 shows a list of the functions of this sample driver. For the details of the functions, refer to the port.c source code.

Table 4.1 List of I/O ports (P) sample driver functions

Function Name	Description Name
PORT_init	Px port initialization
PORT_getInputData	Px port data input
PORT_setOutputData	Px port data output
PORT_controlInput	Px port input allow/disallow setting
PORT_controlOutput	Px port output allow/disallow setting
PORT_controlPullup	Px port pull-up resistance setting
PORT_controlSchmittTrigger	Px port Schmitt trigger setting
PORT_initInt	Px port interrupt initialization
PORT_controlInt	Px port interrupt allow/disallow setting
PORT_setIntEdge	Px port interrupt edge setting
PORT_resetIntFlag	Px port interrupt factor flag reset
PORT_checkIntFlag	Px port interrupt factor flag check
PORT_setChatteringFilter	Px port chattering elimination setting

This sample driver is described in port.c and port.h as well as port_api.h.

For programs using this sample driver, include the port_api.h file.

4.2 Oscillation Circuit (OSC)

Table 4.2 shows a list of the functions of this sample driver. For the details of the functions, refer to the osc.c source code.

Table 4.2 List of oscillation circuit (OSC) sample driver functions

Function Name	Description Name
OSC_setClockSource	Clock source setting
OSC_setWaitCycle	Oscillation stability wait time setting
OSC_controlOscillation	OSC oscillation start/stop setting
OSC_setNoiseFilter	Noise filter enable/disable setting
OSC_setLCDClock	LCD clock setting
OSC_controlLCDClock	LCD clock supply allow/disallow setting
OSC_setFOUTHDivision	FOUTH clock frequency division ratio setting
OSC_controlFOUT	FOUT clock output allow/disallow setting
OSC_setT8OSC1Division	T8OSC1 clock frequency division ration setting
OSC_controlT8OSC1	T8OSC1 clock output allow/disallow setting
OSC_setSVDClock	SVD clock setting
OSC_controlSVD	SVD clock output allow/disallow setting
OSC_setRFCClock	RFC clock setting
OSC_controlRFC	RFC clock supply allow/disallow setting

This sample driver is described in osc.c and osc.h as well as osc_api.h.

For programs using this sample driver, include the osc_api.h file.

4.3 16-Bit Timer (T16)

Table 4.3 shows a list of the functions of this sample driver. For the details of the functions, refer to the t16.c source code.

Table 4.3 List of 16-bit timer (T16) sample driver functions

Function Name	Description Name
T16_init	16-bit timer initialization
T16_setInputClock	Prescaler output clock setting
T16_setReloadData	Reload data setting
T16_getCounterData	Counter data acquisition
T16_setTimerMode	16-bit timer mode setting
T16_resetTimer	16-bit timer reset
T16_setTimerRun	16-bit timer start/stop setting
T16_initInt	16-bit timer interrupt initialization
T16_controlInt	16-bit timer interrupt allow/disallow setting
T16_resetIntFlag	16-bit timer interrupt factor flag reset
T16_checkIntFlag	16-bit timer interrupt factor flag check

This sample driver is described in t16.c and t16.h as well as t16_api.h.

For programs using this sample driver, include the t16_api.h file.

4.4 8-Bit Timer (T8F)

Table 4.4 shows a list of the functions of this sample driver. For the details of the functions, refer to the t8f.c source code.

Table 4.4 List of 8-bit timer (T8F) sample driver functions

Function Name	Description Name
T8F_init	8-bit timer initialization
T8F_setInputClock	Prescaler output clock setting
T8F_setReloadData	Reload data setting
T8F_getCounterData	Counter data acquisition
T8F_setTimerMode	8-bit timer mode setting
T8F_resetTimer	8-bit timer reset
T8F_setTimerRun	8-bit timer start/stop setting
T8F_initInt	8-bit timer interrupt initialization
T8F_controlInt	8-bit timer interrupt allow/disallow setting
T8F_resetIntFlag	8-bit timer interrupt factor flag reset
T8F_checkIntFlag	8-bit timer interrupt factor flag check

This sample driver is described in t8f.c and t8f.h as well as t8f_api.h.

For programs using this sample driver, include the t8f_api.h file.

4.5 PWM Timer (T16E)

Table 4.5 shows a list of the functions of this sample driver. For the details of the functions, refer to the t16e.c source code.

Table 4.5 List of PWM timer (T16E) sample driver functions

Function Name	Description Name
T16E_init	PWM timer initialization
T16E_setCompareData	Compare-data setting
T16E_getCounterData	Counter data acquisition
T16E_setTimerMode	PWM timer mode setting
T16E_resetTimer	PWM timer reset
T16E_setTimerRun	PWM timer start/stop setting
T16E_setInputClock	Prescaler output clock setting
T16E_initInt	PWM timer interrupt initialization
T16E_controlInt	PWM timer interrupt allow/disallow setting
T16E_resetIntFlag	PWM timer interrupt factor flag reset
T16E_checkIntFlag	PWM timer interrupt factor flag check

This sample driver is described in t16e.c and t16e.h as well as t16e_api.h.

For programs using this sample driver, include the t16e_api.h file.

4.6 8-Bit OSC1 Timer (T8OSC1)

Table 4.6 shows a list of the functions of this sample driver. For the details of the functions, refer to the t8osc1.c source code.

Table 4.6 List of 8-bit OSC1 timer (T8OSC1) sample driver functions

Function Name	Description Name
T8O_init	8-bit OSC1 timer initialization
T8O_resetTimer	8-bit OSC1 timer reset
T8O_setTimerMode	8-bit OSC1 timer mode setting
T8O_setTimerRun	8-bit OSC1 timer start/stop setting
T8O_getCounterData	Counter data acquisition
T8O_setCompareData	Compare-data setting
T8O_initInt	8-bit OSC1 timer interrupt initialization
T8O_controlInt	8-bit OSC1 timer interrupt allow/disallow setting
T8O_resetIntFlag	8-bit OSC1 timer interrupt factor flag reset
T8O_checkIntFlag	8-bit OSC1 timer interrupt factor flag check
T8O_setPWMOut	PWM output setting

This sample driver is described in t8osc1.c and t8osc1.h as well as t8osc1_api.h.

For programs using this sample driver, include the t8osc1_api.h file.

4.7 Clock Timer (CT)

Table 4.7 shows a list of the functions of this sample driver. For the details of the functions, refer to the ct.c source code.

Table 4.7 List of clock timer (CT) sample driver functions

Function Name	Description Name
CT_resetTimer	Clock timer reset
CT_setTimerRun	Clock timer start/stop setting
CT_getCounterData	Counter data acquisition
CT_initInt	Clock timer interrupt initialization
CT_controlInt	Clock timer interrupt allow/disallow setting
CT_resetIntFlag	Clock timer interrupt factor flag reset
CT_checkIntFlag	Clock timer interrupt factor flag check

This sample driver is described in ct.c and ct.h as well as ct_api.h.

For programs using this sample driver, include the ct_api.h file.

4.8 Stopwatch Timer (SWT)

Table 4.8 shows a list of the functions of this sample driver. For the details of the functions, refer to the swt.c source code.

Table 4.8 List of stopwatch timer (SWT) sample driver functions

Function Name	Description Name
SWT_resetTimer	Stopwatch timer reset
SWT_setTimerRun	Stopwatch timer start/stop setting
SWT_getCounterDataBCD	BCD counter data acquisition
SWT_initInt	Stopwatch timer interrupt initialization
SWT_controlInt	Stopwatch timer interrupt allow/disallow setting
SWT_resetIntFlag	Stopwatch timer interrupt factor flag reset
SWT_checkIntFlag	Stopwatch timer interrupt factor flag check

This sample driver is described in swt.c and swt.h as well as swt_api.h.

For programs using this sample driver, include the swt_api.h file.

4.9 Watchdog Timer (WDT)

Table 4.9 shows a list of the functions of this sample driver. For the details of the functions, refer to the wdt.c source code.

Table 4.9 List of watchdog timer (WDT) sample driver functions

Function Name	Description Name
WDT_resetTimer	Watchdog timer reset
WDT_setTimerRun	Watchdog timer start/stop setting
WDT_setTimerMode	Watchdog timer mode setting
WDT_checkNMI	Watchdog timer NMI occurrence check

This sample driver is described in wdt.c and wdt.h as well as wdt_api.h.

For programs using this sample driver, include the wdt_api.h file.

4.10 UART

Table 4.10 shows a list of the functions of this sample driver. For the details of the functions, refer to the uart.c source code.

Table 4.10 List of UART sample driver functions

Function Name	Description Name
UART_init	UART initialization
UART_setTransmitData	Send-data setting
UART_getReceiveData	Receive-data acquisition
UART_setComEnable	UART sending and receiving allow/disallow setting
UART_initInt	UART interrupt initialization
UART_controlInt	UART interrupt allow/disallow setting
UART_resetIntFlag	UART interrupt factor flag reset
UART_checkReceiveFlag	UART interrupt factor flag check
UART_setIrDAmode	IrDA mode setting

This sample driver is described in uart.c and uart.h as well as uart_api.h.

For programs using this sample driver, include the uart_api.h file.

4.11 SPI

Table 4.11 shows a list of the functions of this sample driver. For the details of the functions, refer to the spi.c source code.

Table 4.11 List of SPI sample driver functions

Function Name	Description Name
SPI_init	SPI initialization
SPI_setTransmitData	Send-data setting
SPI_getReceiveData	Receive-data acquisition
SPI_setComEnable	SPI sending and receiving allow/disallow setting
SPI_initInt	SPI interrupt initialization
SPI_controlInt	SPI interrupt allow/disallow setting
SPI_checkIntFlag	SPI interrupt factor flag check
SPI_checkBusyFlag	Sending and receiving BUSY flag check

This sample driver is described in spi.c and spi.h as well as spi_api.h.

For programs using this sample driver, include the spi_api.h file.

4.12 I2C Master (I2CM)

Table 4.12 shows a list of the functions of this sample driver. For the details of the functions, refer to the i2cm.c source code.

Table 4.12 List of I2C master (I2CM) sample driver functions

Function Name	Description Name
I2CM_init	I2C master initialization
I2CM_setComEnable	I2C master sending and receiving allow/disallow setting
I2CM_genCondition	Start/stop condition generation
I2CM_checkTransmitReg	Send-data register check
I2CM_setTransmitData	Send-data setting
I2CM_checkTransmitBusy	Send-operation status check
I2CM_getSlaveResponse	Slave response acquisition
I2CM_setReceiveStart	Data receiving start setting
I2CM_checkReceiveBusy	Receive-operation status check
I2CM_getReceiveData	Receive-data acquisition
I2CM_checkReceiveReg	Receive-data register check
I2CM_initInt	I2C master interrupt initialization
I2CM_controlInt	I2C master interrupt allow/disallow setting
I2CM_transmitSlaveAddress	Slave address send-data creation

This sample driver is described in i2cm.c and i2cm.h as well as i2cm_api.h.

For programs using this sample driver, include the i2cm_api.h file.

4.13 I2C Slave (I2CS)

Table 4.13 shows a list of the functions of this sample driver. For the details of the functions, refer to the i2cs.c source code.

Table 4.13 List of I2C slave (I2CS) sample driver functions

Function Name	Description Name
I2CS_reset	I2C slave software reset
I2CS_setAddress	I2C slave address setting
I2CS_setClockStretch	Clock stretching capability setting
I2CS_setAsyncDetection	Asynchronous address detection capability setting
I2CS_setNoiseRemove	Noise elimination capability selection
I2CS_setBusFreeReq	Bus release request allow/disallow setting
I2CS_setReceiveResponse	Data receiving response setting
I2CS_init	I2C slave initialization
I2CS_setEnable	I2C slave module operation allow/disallow setting
I2CS_setComEnable	Data sending and receiving allow/disallow setting
I2CS_setTransmitData	Send-data setting
I2CS_getReceiveData	Receive-data acquisition
I2CS_initInt	I2C slave interrupt initialization
I2CS_controlInt	I2C slave interrupt allow/disallow setting
I2CS_resetIntFlag	I2C slave bus status interrupt factor flag reset
I2CS_checkBusStatusIntFlag	I2C slave bus status interrupt factor flag check
I2CS_checkIntFlag	I2C slave interrupt factor flag check
I2CS_checkAccessStatus	I2C slave access status check

This sample driver is described in i2cs.c and i2cs.h as well as i2cs_api.h.

For programs using this sample driver, include the i2cs_api.h file.

4.14 LCD Driver (LCD)

Table 4.14 shows a list of the functions of this sample driver. For the details of the functions, refer to the lcd.c source code.

Table 4.14 List of LCD driver (LCD) sample driver functions

Function Name	Description Name
LCD_initPower	LCD power source initialization
LCD_init	LCD initialization
LCD_setSEGAssignment	SEG terminal memory allocation setting
LCD_setCOMAssignment	COM terminal memory allocation setting
LCD_setDisplayArea	LCD display area setting
LCD_setDisplayReverse	LCD display black-white reversal setting
LCD_controlDisplay	LCD display control
LCD_setContrast	LCD contrast setting
LCD_display1Seg	1 segment display
LCD_initInt	LCD interrupt initialization
LCD_controlInt	LCD interrupt allow/disallow setting
LCD_resetIntFlag	LCD interrupt factor flag reset
LCD_checkIntFlag	LCD interrupt factor flag check

This sample driver is described in lcd.c and lcd.h as well as lcd_api.h.

For programs using this sample driver, include the lcd_api.h file.

4.15 Power Source Voltage Detection Circuit (SVD)

Table 4.15 shows a list of the functions of this sample driver. For the details of the functions, refer to the svd.c source code.

Table 4.15 List of power source voltage detection circuit (SVD) sample driver functions

Function Name	Description Name
SVD_setCompareVoltage	SVD comparative voltage setting
SVD_controlDetection	SVD detection start/stop setting
SVD_getDetectionResult	SVD detection result acquisition
SVD_initInt	SVD interrupt initialization
SVD_controlInt	SVD interrupt allow/disallow setting
SVD_resetIntFlag	SVD interrupt factor flag reset
SVD_checkIntFlag	SVD interrupt factor flag check

This sample driver is described in svd.c and svd.h as well as svd_api.h.

For programs using this sample driver, include the svd_api.h file.

4.16 R/F Converter (RFC)

Table 4.16 shows a list of the functions of this sample driver. For the details of the functions, refer to the rfc.c source code.

Table 4.16 R/F converter (RFC) sample driver functions

Function Name	Description Name
RFC_setRFC	R/F converter allow/disallow setting
RFC_setRFCChannel	Conversion channel setting
RFC_setRFCMode	Oscillation mode setting
RFC_setReferenceOscillation	Reference oscillation start/stop setting
RFC_setSensorAOscillation	Sensor A oscillation start/stop setting
RFC_setSensorBOscillation	Sensor B oscillation start/stop setting
RFC_getReferenceOscillation	Reference oscillation status acquisition
RFC_getSensorAOscillation	Sensor A oscillation status acquisition
RFC_getSensorBOscillation	Sensor B oscillation status acquisition
RFC_setEventMode	Event counter mode allow/disallow setting
RFC_setContinuous	Continuous oscillation allow/disallow setting
RFC_setMeasurementCounter	Measurement counter value setting
RFC_setTimeBaseCounter	Time base counter value setting
RFC_getMeasurementCounter	Measurement counter value setting
RFC_getTimeBaseCounter	Time base counter value acquisition
RFC_initInt	RFC interrupt initialization
RFC_controlInt	RFC interrupt allow/disallow setting
RFC_resetIntFlag	RFC interrupt factor flag reset
RFC_checkIntFlag	RFC interrupt factor flag check

This sample driver is described in rfc.c and rfc.h as well as rfc_api.h.

For programs using this sample driver, include the rfc_api.h file.

4.17 A/D Converter (ADC10)

Table 4.17 shows a list of the functions of this sample driver. For the details of the functions, refer to the adc.c source code.

Table 4.17 List of A/D converter (ADC10) sample driver functions

Function Name	Description Name
ADC_init	ADC initialization
ADC_setChannel	A/D conversion channel setting
ADC_setStoreMode	Conversion result storage method setting
ADC_setConversionMode	A/D conversion mode setting
ADC_setConversionTrigger	A/D conversion start trigger method setting
ADC_setSamplingClock	Sampling time setting
ADC_setDividedFrequency	A/D conversion clock frequency division setting
ADC_setEnable	A/D conversion start/stop setting
ADC_getResult	A/D conversion result acquisition
ADC_getConversionChannel	A/D conversion in-use channel number acquisition
ADC_checkBusyStatus	A/D conversion in-use check
ADC_controlTrigger	Software trigger control
ADC_initInt	ADC interrupt initialization
ADC_controlInt	ADC interrupt allow/disallow setting
ADC_resetIntFlag	ADC interrupt factor flag reset
ADC_checkIntFlag	ADC interrupt factor flag check

This sample driver is described in adc.c and adc.h as well as adc_api.h.

For programs using this sample driver, include the adc_api.h file.

4.18 Remote Controller (REMC)

Table 4.18 shows a list of the functions of this sample driver. For the details of the functions, refer to the remc.c source code.

Table 4.18 List of remote controller (REMC) sample driver functions

Function Name	Description Name
REMC_init	REMC initialization
REMC_setEnable	REMC sending and receiving start/stop setting
REMC_setTransmitData	Send-data setting
REMC_setReceiveLength	Receive-data length setting
REMC_getReceiveData	Receive-data acquisition
REMC_calcReceiveDataPulse	Send-data pulse length calculation
REMC_initInt	REMC interrupt initialization
REMC_controlInt	REMC interrupt allow/disallow setting
REMC_resetIntFlag	REMC interrupt factor flag reset
REMC_checkIntFlag	REMC interrupt factor flag check

This sample driver is described in remc.c and remc.h as well as remc_api.h.

For programs using this sample driver, include the remc_api.h file.

4.19 Clock Generator (CLG)

Table 4.19 shows a list of the functions of this sample driver. For the details of the functions, refer to the clg.c source code.

Table 4.19 List of clock generator (CLG) sample driver functions

Function Name	Description Name
CLG_setPCLKEnable	PCLK supply allow/disallow setting
CLG_setCCLKGearRatio	System clock gear ratio setting

This sample driver is described in clg.c and clg.h as well as clg_api.h.

For programs using this sample driver, include the clg_api.h file.

4.20 Prescaler (PSC)

Table 4.20 shows a list of the functions of this sample driver. For the details of the functions, refer to the psc.c source code.

Table 4.20 List of prescaler (PSC) sample driver functions

Function Name	Description Name
PSC_setPrescalerControl	Prescaler start/stop setting

This sample driver is described in psc.c and psc.h as well as psc_api.h.

For programs using this sample driver, include the psc_api.h file.

4.21 MISC

Table 4.21 shows a list of the functions of this sample driver. For the details of the functions, refer to the misc.c source code.

Table 4.21 List of MISC sample driver functions

Function Name	Description Name
MISC_setFlashReadAccessCycle	Flash controller read access cycle setting
MISC_setOSC1PeripheralControl	Debug-time OSC1 operation peripheral capability setting
MISC_controlWriteProtect	MISC register write-protection control
MISC_setIRAMSize	IRAM size setting
MISC_setTTBR	Vector table address setting
MISC_getPSR	PSR acquisition

This sample driver is described in misc.c and misc.h as well as misc_api.h.

For programs using this sample driver, include the misc_api.h file.

4.22 Multiplexer (MUX)

Table 4.22 shows a list of the functions of this sample driver. For the details of the functions, refer to the mux.c source code.

Table 4.22 List of multiplexer (MUX) sample driver functions

Function Name	Description Name
MUX_init	MUX initialization
MUX_setREMCport	REMC port setting
MUX_setADCport	ADC port setting
MUX_setSPIport	SPI port setting
MUX_setUARTport	UART port setting
MUX_setRFCport	RFC port setting
MUX_setI2CMport	I2C master port setting
MUX_setI2CSport	I2C slave port setting
MUX_setOSCport	OSC port setting
MUX_setT16Eport	PWM timer port setting
MUX_setLCDport	LCD port setting
MUX_setT8OSC1port	8-bit OSC1 timer port setting
MUX_setDBGport	Debug-port setting
MUX_setT16port	16-bit timer port setting

This sample driver is described in mux.c and mux.h as well as mux_api.h.

For programs using this sample driver, include the mux_api.h file.

Appendix A Multiplier and Divider

Here, the usage method of multiplier and divider is described.

A.1 Multiplication and Division using Multiplier and Divider

In order to multiply and divide using the multiplier and divider, the library for the coprocessor is provided in GNU17.

For the usage method of the library for coprocessor, refer to the S5U1C17001C manual.

A.2 Sum of Products Calculation using Multiplier and Divider

The program for calculating the sum of products using the multiplier and divider is shown below.

This program calculates the sum of products for "0x1204×0x1080+0x28A00."

```
asm ( "ld.cw %r0, 0x0" );        /* clear */
asm ( "ld.cw %r0, 0x2" );        /* setup mode */
asm ( "xld %r0, 0x0002" );       /* set 0x28A00 */
asm ( "xld %r1, 0x8A00" );

asm ( "ld.cf %r0, %r1" );

asm ( "ld.cw %r0, 0x7" );        /* setup mode */
asm ( "xld %r0, 0x1204" );       /* 0x1204 */
asm ( "xld %r1, 0x1080" );       /* 0x1080 */
asm ( "ld.ca %r0, %r1" );

asm ( "ld.cw %r0, 0x13" );       /* read */
asm ( "ld.ca %r1, %r0" );
asm ( "ld.cw %r0, 0x03" );       /* read */
asm ( "ld.ca %r2, %r0" );

/* result = 0x12BCC00 */
```

Revision History

Attachment-1

Code No.	Page	Contents
411883100	All	First edition

EPSON

International Sales Operations

AMERICA

EPSON ELECTRONICS AMERICA, INC.
2580 Orchard Parkway,
San Jose, CA 95131, 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 HONG KONG LTD.
20/F, Harbour Centre, 25 Harbour Road,
Wanchai, Hong Kong
Phone: +852-2585-4600 FAX: +852-2827-4346
Telex: 65542 EPSCO HX

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
50F, 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.
SEMICONDUCTOR OPERATIONS DIVISION

IC Sales Dept.
IC International Sales Group
421-8, Hino, Hino-shi, Tokyo 191-8501, JAPAN
Phone: +81-42-587-5814 FAX: +81-42-587-5117

Document Code: 411883100
First Issue February 2010 in JAPAN Ⓥ