EPSON
EXCEED YOUR VISION

S1C17 Family Application Note

S1C17651/653 Sample Software for Peripherals

SEIKO EPSON CORPORATION

Rev.1.0

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 purpose. It is not intended to meet the requirement of design for finished product.
2. This evaluation board/kit or development tool is intended for use by an electronics engineer, and it is not the product for consumer. The user should use this goods properly and safely. Seiko Epson dose not assume any responsibility and liability of any kind of damage and/or fire coursed by usage of it. 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 is changed without any notice.

NOTICE

Table of Contents

1. Overview

This manual describes how the sample software for the S1C17651 and S1C17653 is used and works.

The purpose of the sample software for the S1C17651 and S1C17653 is to show examples of how to use each internal peripheral of the S1C 17651/653 microcontroller.

The sample software for the S1C17651 and S1C17653 is offered separately for each model for easy installation. But how each function of the software works is basically the same for both models.

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

1.1 Working Environment

Prepare the following equipment when you use the sample software for the S1C17651 and S1C17653.

- A printed circuit board with S1C17651/653 mounted on
- S5U1C17001H (hereafter referred to as ICDmini)
- S5U1C17001C (hereafter referred to as GNU17)

Note The functionality of this sample software is tested with GNU17v2.2.0.

2. Details Of The Sample Software

This chapter describes the file structure of the sample software for the S1C17651 and S1C17653 and how to execute the sample software.

The sample software for the S1C17651 and S1C17653 consists of two parts: the “sample software” part intended for checking the function of each peripheral and the “sample driver” part, which is a collection of a sample driver for each peripheral.

2.1 Directory and File Structure

The following shows the directory structure of the sample software for the S1C17651 and S1C17653.

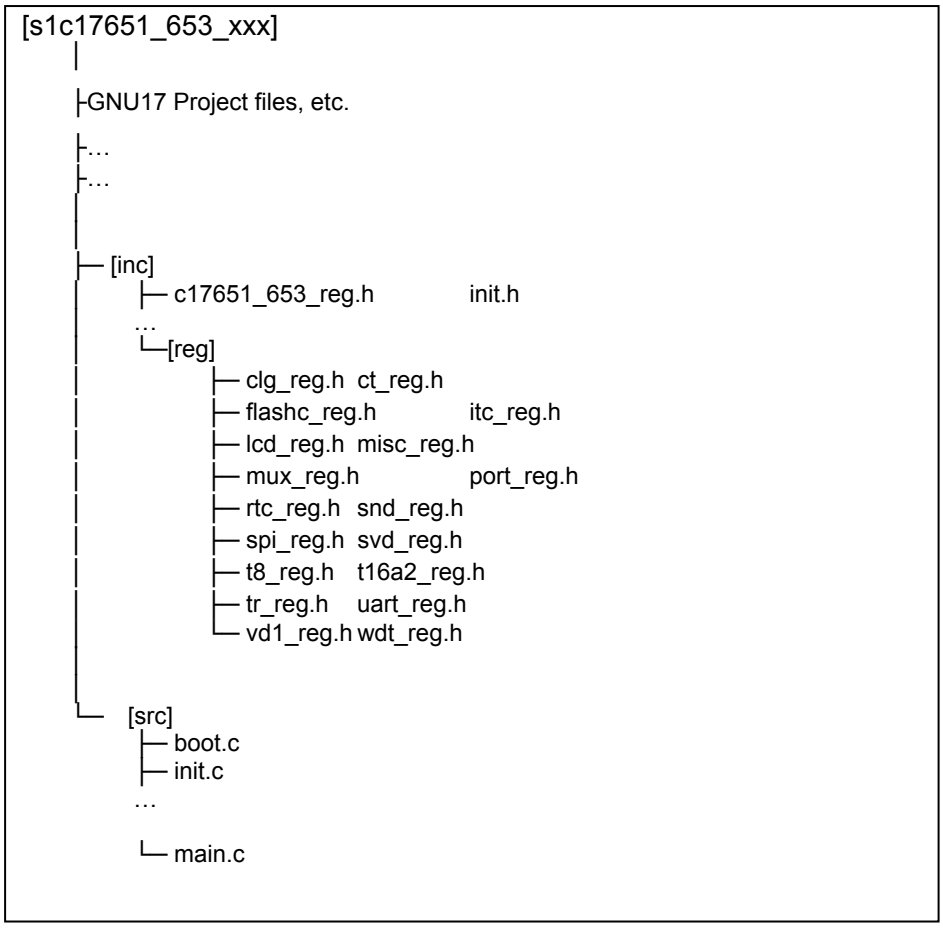

Figure 2.1 Directory Structure of the Sample Software for the S1C17651 and S1C17653

(1) s1c17651_653_xxx directory

This directory contains directories for storing files related to GNU17 projects.

(2) inc directory

This directory contains a file for defining model-dependent data and a header file for defining the constants and the like used in the sample software for each peripheral.

- The header file that defines the register addresses of the target models (c17651_653_reg.h)
- A header file for each peripheral (init.h and so on)

(3) reg directory

This directory contains a header file for defining the bit assignment and the like for each peripheral.

- A header file that defines the bit assignment and the like for each peripheral (clg_reg.h and so on)

(4) src directory

This directory contains initialization file, sample software and a sample driver for each peripheral.

- An initialization file (boot.c)
- A sample software file for each peripheral (main.c)
- A sample driver file for each peripheral (init.c and others)

2.2 Executing the Sample Software

Follow the procedure below to execute the sample software for the S1C17651 and S1C17653.

(1) Importing a project

Start GNU17, and import a project in the sample software for the S1C17651 and S1C17653

For details about how to import a project, refer to “Software Development Procedures” in the S5U1C17001C Manual.

(2) Building the project

Build the S1C176xx project using GNU17.

For details about how to build a project, refer to “GNU17 IDE” in the S5U1C17001C Manual.

(3) Connecting the ICDmini

Connect the ICDmini to the PC and the development board, and then turn on the power for the development board.

(4) Loading and executing the program using the debugger

Start debugging by pressing the [Debug] button on the GNU17.

The program will be loaded to the S1C17651/653 and then run.

For details about how to use the debugger, refer to “Debugger” in the S5U1C17001C Manual.

3. Details Of The Functions Of The Sample Software

This chapter describes the details of the functions of the sample software for the S1C17651 and S1C17653.

3.1 I/O Ports (P)

3.1.1 Specifications of the Sample Software

This sample software uses the I/O ports and executes the following items.

- Some ports are set as input interrupt ports so that you can detect when one of the input signals gets Low.
- Some ports are set as output ports to output High-level and Low-level signals.

The port settings and port names in use are as follows.

Table 3.1.1 List of I/O Port Settings

Setting	Port name
Input interrupt port	P00 P01 P02 P03
Output port	P04 P05 P06 P07

3.1.2 Hardware Conditions

This sample program uses the OSC3B(2MHz) internal oscillator of the microcontroller.

When you use this sample program, connect the ports of the microcontroller as shown in the following figure.

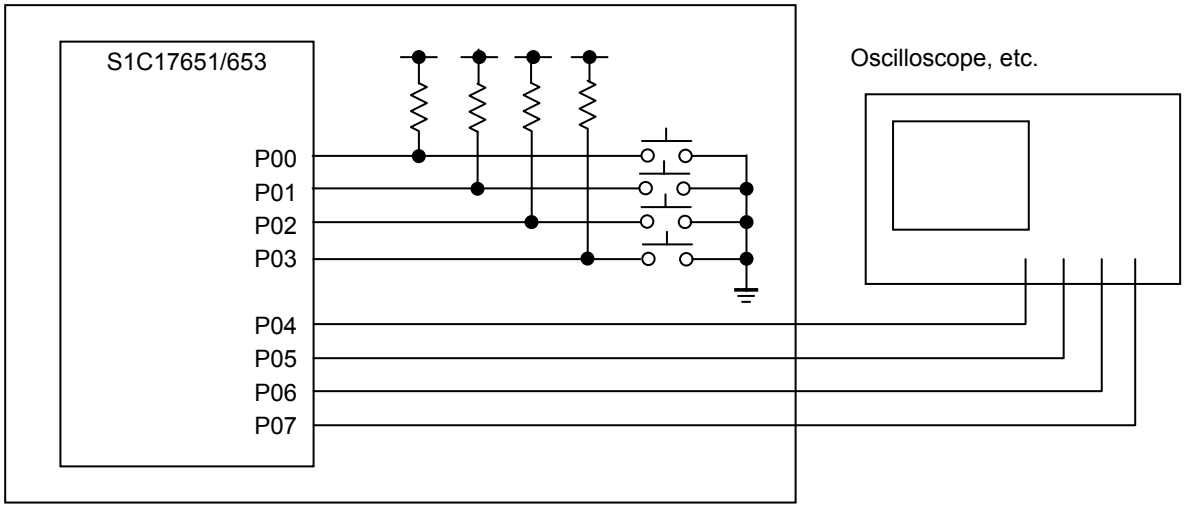

Figure 3.1.1 Hardware Connection Diagram for the I/O Ports (P) Sample Software

3.1.3 Summary of the Operation

1. When the input signal for the P00 port gets Low, “P00 Interrupt” is displayed on the Simulated I/O output window, and the output of the P04 port is inverted. (If the output is High, it is set to Low, If it is Low, it is set to High.)
2. When the input signal for the P01 port gets Low, “P01 Interrupt” is displayed on the Simulated I/O output window, and the output of the P05 port is inverted. (If the output is High, it is set to Low, If it is Low, it is set to High.)
3. When the input signal for the P02 port gets Low, “P02 Interrupt” is displayed on the Simulated I/O output window, and the output of the P06 port is inverted. (If the output is High, it is set to Low, If it is Low, it is set to High.)
4. When the input signal for the P03 port gets Low, “P03 Interrupt” is displayed on the Simulated I/O output window, and the output of the P07 port is inverted. (If the output is High, it is set to Low, If it is Low, it is set to High.)

```
<<< Port demonstration start >>>
*** P00 Interrupt ***
*** P01 Interrupt ***
*** P02 Interrupt ***
*** P03 Interrupt ***
<<< Port demonstration finish >>>
```

Figure 3.1.2 I/O Ports Sample Program Screen Display Example

3.2 Clock Generator (CLG)

3.2.1 Specifications of the Sample Software

The oscillator sample program uses the OSCs and executes the following items.

- Oscillation of the OSC1 starts and then stops. (This sample program selects the OSC1A. Hereafter, it is denoted as “OSC1”.)
- Oscillation of the OSC3A starts and then stops.
- Oscillation of the OSC3B starts and then stops.
- The system clock is switched from the OSC3B to the OSC3A.
- The system clock is switched from the OSC3A to the OSC1.
- The system clock is switched from the OSC1 to the OSC3B.

3.2.2 Hardware Conditions

This sample program requires a crystal or ceramic resonator to be connected to both the OSC1A and the OSC3A. For information about how to connect a resonator, refer to “Clock Generator (CLG)” in the S1C17651/S1C17653 technical manual.

3.2.3 Summary of the Operation

When this sample program starts, the OSC3B is used.

1. At a fixed interval, on the Simulated I/O output window, “1”, “2”, “3” ..., “9” are displayed. Then the OSC3A starts oscillating, the system clock is switched from the OSC3B to the OSC3A, and the OSC3B stops.
2. At a fixed interval, on the Simulated I/O output window, “1”, “2”, “3” ..., “9” are displayed. Then the OSC1 starts oscillating, the system clock is switched from the OSC3A to the OSC1, and the OSC3A stops.
3. At a fixed interval, on the Simulated I/O output window, “1”, “2”, “3” ..., “9” are displayed. Then the OSC3B starts oscillating, the system clock is switched from the OSC1 to the OSC3B, and the OSC1 stops.
4. At a fixed interval, on the Simulated I/O output window, “1”, “2”, “3” ..., “9” are displayed. Then the sample program terminates.

```
<<< CLG(OSC) demonstration start >>>
OSC3B *** 1 ***
OSC3B *** 2 ***
…
OSC3B *** 9 ***
*** Change from OSC3B to OSC3A ***
OSC3A *** 1 ***
OSC3 A*** 2 ***
…
OSC3A*** 9 ***
*** Change from OSC3A to OSC1 ***
OSC1 *** 1 ***
OSC1*** 2 ***
…
OSC1*** 9 ***
*** Change from OSC1 to OSC3B ***
OSC3B *** 1 ***
OSC3B *** 2 ***
…
OSC3B *** 9 ***
<<< CLG(OSC) demonstration finish >>>
```

Figure 3.2.1 CLG_OSC Sample Program Screen Display Example

3.3 8-bit Timer (T8)

3.3.1 Specifications Of The Sample Software

The 8-bit timer sample program uses the 8-bit timer and executes the following items.

- An 8-bit timer interrupt is generated and the counter value of the timer is read.
- While waiting for an interrupt, the CPU is in the halt mode to reduce power consumption.

3.3.2 Hardware Conditions

This sample program uses the OSC3B(2MHz) internal oscillator of the microcontroller.

3.3.3 Summary of the Operation

1. The 8-bit timer interrupt is set up and the CPU enters the Halt mode.
2. When an 8-bit timer interrupt is generated, the Halt mode of the CPU is cleared.
3. The counter data of the 8-bit timer is stored to an internal variable and the CPU enters the Halt mode again.
4. When an 8-bit timer interrupt is generated ten times, the 8-bit timer stops.
5. The counter data when each interrupt is generated is displayed on the Simulated I/O output window. Then the sample program terminates.

```
<<< 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 3.3.1 8-bit Timer Sample Program Screen Display Example

3.4 T16-bit PWM Timer (T16A2)

3.4.1 Specifications of the Sample Software

The T16-bit PWM timer sample program uses the T16-bit PWM timer and executes the following items.

- An T16-bit PWM timer compare A interrupt is generated, and the counter value of the timer is read.
- An T16-bit PWM timer compare B interrupt is generated, and the counter value of the timer is read.
- A waveform is output to both the TOUTA0 and TOUTB0 pins.
- While waiting for an interrupt, the CPU is in the halt mode to reduce power consumption.

3.4.2 Hardware Conditions

This sample program uses the OSC3B(2MHz) internal oscillator of the microcontroller.

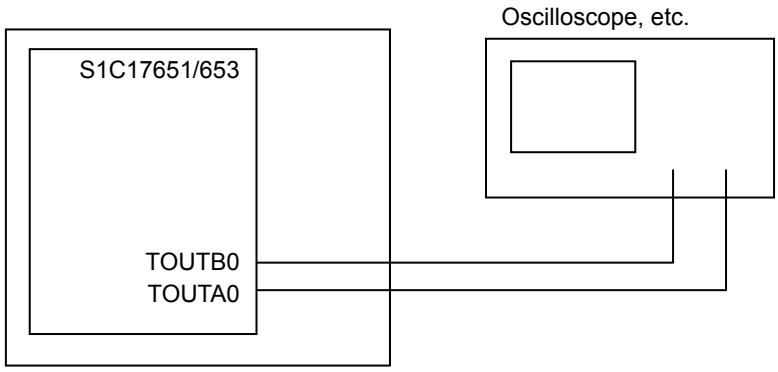

Figure 3.4.1 Hardware Connection Diagram for the T16-bit PWM Timer Sample Program

3.4.3 Summary of the Operation

1. The compare A interrupt and the compare B interrupt are enabled and the T16-bit PWM timer starts.
2. When a compare A interrupt or compare B interrupt is generated, the counter value of the up counter is read.
3. When a compare B interrupt is generated five times, the T16-bit PWM timer stops and the type and the counter value of each interrupt are displayed on the Simulated I/O output window. Then the sample program terminates.

```
<<< T16 PWM timer demonstration start >>>
*** T16 PWM timer compare A interrupt :30 ***
*** T16 PWM timer compare B interrupt :61    ***
*** T16 PWM timer compare A interrupt :30 ***
…
*** T16 PWM timer compare B interrupt: 61 ***
<<< T16 PWM timer demonstration finish >>>
```

Figure 3.4.2 T16-bit PWM Timer Sample Program Screen Display Example

3.5 Clock Timer (CT)

3.5.1 Specifications of the Sample Software

The clock timer sample program uses the clock timer and executes the following items.

- A clock timer interrupt is generated, and the elapsed time is calculated.
- While waiting for an interrupt, the CPU is in the halt mode to reduce power consumption.

3.5.2 Hardware Conditions

This sample program uses the OSC3B(2MHz) internal oscillator and the OSC1B(32kHz) of the microcontroller.

3.5.3 Summary of the Operation

1. The clock timer starts.
2. When a clock timer interrupt is generated, the elapsed time from when the program starts is calculated, and the elapsed time is displayed on the Simulated I/O output window.
3. When a clock timer interrupt is generated ten times, the sample program terminates.

```
<<< Clock timer demonstration start >>>
*** 1.0 sec ***
*** 2.0 sec ***
*** 3.0 sec ***
…
*** 10.0 sec ***
<<< Clock timer demonstration finish >>>
```

Figure 3.5.1 Clock Timer Sample Program Screen Display Example

3.6 Real-Time Clock (RTC)

3.6.1 Specifications of the Sample Software

The real-time clock sample program uses the real-time clock and executes the following items.

- The sample program menu of the real-time clock is displayed.
- The time is read from the real-time clock.
- The time of the real-time clock is set.
- The number of real-time clock interrupts is displayed.

3.6.2 Hardware Conditions

This sample program uses the OSC3B(2MHz) internal oscillator and the OSC1B(32kHz) of the microcontroller.

3.6.3 Summary of the Operation

1. The menu of the RTC sample program is displayed on the Simulated I/O output window.
2. A user sets the time using the Simulated I/O.
3. After the time is set, an RTC interrupt is generated every one second. The number of RTC interrupts is displayed on the Simulated I/O output window.
4. When an RTC interrupt is generated ten times, the sample program terminates.

```
<<< Real Time Clock demonstration start >>>
> Input BCD format.
> 24H/12H (0/1):
0
> Hour (00 - 23) :
01
> Minute (00 - 59) :
23
> Second (00 - 59) :
45
interrupt count value = 1
interrupt count value = 2
.
.
.
interrupt count value = 10
<<< Real Time Clock demonstration finish >>>
```

Figure 3.6.1 Real-Time Clock Sample Program Screen Display Example

3.7 Watchdog Timer (WDT)

3.7.1 Specifications of the Sample Software

The watchdog timer sample program uses the watchdog timer and executes the following items.

- Clearance of the watchdog timer is confirmed.
- An NMI interrupt is generated by the watchdog timer.

3.7.2 Hardware Conditions

This sample program uses the OSC3B(2MHz) internal oscillator and the OSC1B(32kHz) of the microcontroller.

3.7.3 Summary of the Operation

1. The watchdog timer and the 8-bit timer start.
2. When an 8-bit timer interrupt is generated, the watchdog timer is cleared.
3. When an 8-bit timer interrupt is generated ten times, the 8-bit timer stops.
4. When an NMI interrupt is generated by the watchdog time, a message is displayed on the Simulated I/O output window. Then the sample program terminates.

```
<<< Watchdog timer demonstration start >>>
*** T8 timer : reset watchdog timer ***
*** T8 timer : reset watchdog timer ***
*** T8 timer : reset watchdog timer ***
…
*** T8 timer : reset watchdog timer ***
*** stop T8 timer ***
*** NMI occurred ***
<<< Watchdog timer demonstration finish >>>
```

Figure 3.7.1 Watchdog Timer Sample Program Screen Display Example

3.8 UART

3.8.1 Specifications of the Sample Software

The UART sample program that uses the clock from the OSC3A uses the UART and executes the following items.

- Data is transmitted using the UART.
- Data is received using the UART.

3.8.2 Hardware Conditions

This sample program uses the OSC3A internal oscillator of the microcontroller.
When you use this sample program, connect the ports of the CPU as shown in the following figure.

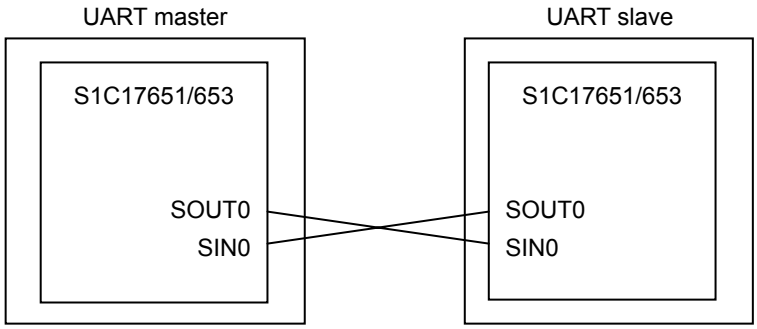

Figure 3.8.1 Hardware Connection Diagram for the UART Sample Program

3.8.3 Summary of the Operation

3.8.3.1 Summary of the Master Sample Operation

1. The UART port is initialized as follows.
   - Communication speed : maximum baud rate
   - Data length : 8bit
   - Stop bit : 1bit
   - Parity : none
2. The clock timer is configured to generate an interrupt every one second.
3. Until the connection confirmation flag "0x7F" is received, "0x7F" is sent every one second.
4. When the connection confirmation flag is received, sending "0x7F" stops and the data in ASCII code from 0x21 through 0x7E is transmitted from the UART port.
5. After the transmission, the UART port receives data, and the data is displayed on the Simulated I/O output window.

```
<<< UART OSC3A demonstration start >>>
waiting connection.
connected
*** receive data ***
ABCDEFG…
<<< UART OSC3A demonstration finish >>>
```

Figure 3.8.2 UART (OSC3A) Master Sample Program Screen Display Example

3.8.3.2 Summary of the Slave Sample Operation

1. The UART port is initialized as follows.
   - Communication speed : maximum baud rate
   - Data length : 8bit
   - Stop bit : 1bit
   - Parity : none
2. The program waits until the connection confirmation flag "0x7F" is received.
3. When the connection confirmation flag is received, "0x7F" is sent, and then data is received from the master.
4. After the reception, the UART port transmits the data in ASCII code from 0x21 through 0x7E, and the received data is displayed on the Simulated I/O output window.

```
<<< UART OSC3A demonstration start >>>
waiting connection.
connected
*** receive data ***
ABCDEFG…
<<< UART OSC3A demonstration finish >>>
```

Figure 3.8.3 UART (OSC3A) Slave Sample Program Screen Display Example

3.9 SPI

3.9.1 Specifications of the Sample Software

3.9.1.1 Specifications of the Master Sample Software

The SPI master sample program uses the SPI master and executes the following items.

- 8 bytes of data are sent to the SPI slave.
- 8 bytes of data are received from the SPI slave.
- While waiting for an interrupt, the CPU is in the halt mode to reduce power consumption.

3.9.1.2 Specifications of the Slave Sample Software

The SPI slave sample program uses the SPI slave and executes the following items.

- 8 bytes of data are received from the SPI master.
- 8 bytes of data are sent to the SPI master.
- While waiting for an interrupt, the CPU is in the halt mode to reduce power consumption.

3.9.2 Hardware Conditions

This sample program uses the OSC3B(2MHz) internal oscillator of the microcontroller.
When you use this sample program, connect the ports of the microcontroller as shown in the following figure.
When you use this sample program, connect an S1C176xx board as the SPI slave while the SPI slave sample program is running on the board.
When you use this sample program, connect the ports as shown in the following figure.

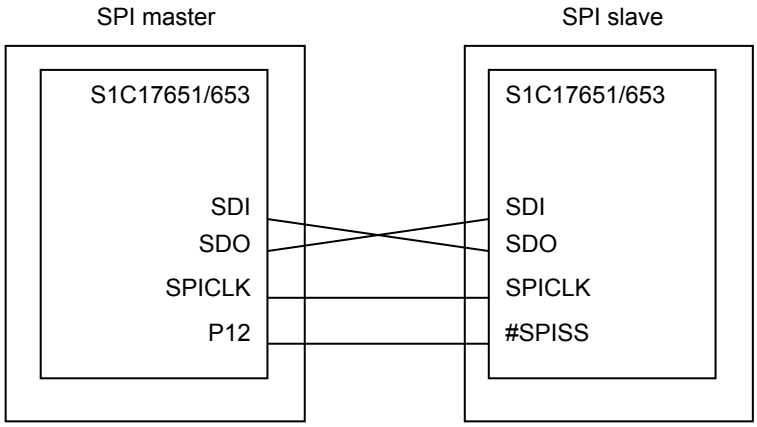

Figure 3.9.1 Hardware Connection Diagram for the SPI Master and Slave Sample Programs

3.9.3 Summary of the Operation

3.9.3.1 Summary of the Master Sample Operation

1. The SPI master is initialized.
2. The 8 bytes of ASCII data “FROM MST” is sent to the SPI slave.
3. When the data transmission to the SPI slave is finished, the SPI clock is output to the SPI slave to receive data.
4. The data received from the SPI slave is displayed on the Simulated I/O output window. Then the sample program terminates.

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

Figure 3.9.2 SPI Master Sample Program Screen Display Example

3.9.3.2 Summary of the Slave Sample Operation

1. The SPI slave is initialized.
2. The program waits for data reception from the SPI master.
3. When data is received form the SPI master, the received data is displayed on the Simulated I/O output window.
4. The 8 bytes of ASCII data “FROM SLV” is sent to the SPI master. Then the sample program terminates.

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

Figure 3.9.3 SPI Slave Sample Program Screen Display Example

3.10 LCD Driver (LCD)

3.10.1 Specifications of the Sample Software

The LCD driver sample program uses the LCD driver and executes the following items.

- All segments are turned on and off in the normal display mode.
- All segments are turned on and off in the all on/off mode.

3.10.2 Hardware Conditions

This sample program uses the OSC3B(2MHz) internal oscillator of the microcontroller.
When you use this sample program, connect the ports of the microcontroller as shown in the following figure.

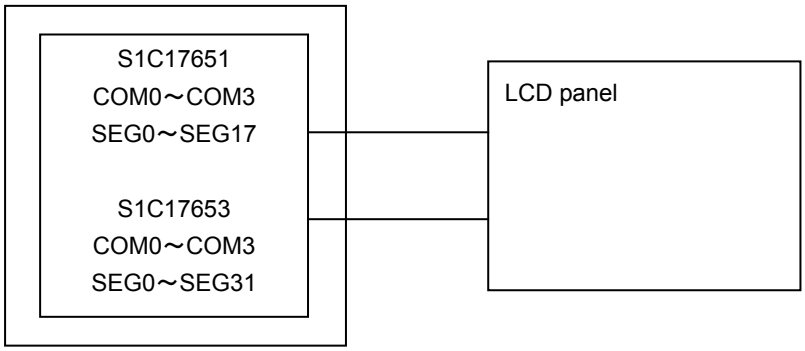

Figure 3.10.1 Hardware Connection Diagram for the LCD Driver Sample Program

3.10.3 Summary of the Operation

1. When “on0” is entered and the ENTER key is pressed on the Simulated I/O, all segments of the LCD are turned on in the normal display mode.
2. When “on1” is entered and the ENTER key is pressed on the Simulated I/O, all segments of the LCD are turned on in the all on mode.
3. When “off0” is entered and the ENTER key is pressed on the Simulated I/O, all segments of the LCD are turned off in the normal display mode.
4. When “off1” is entered and the ENTER key is pressed on the Simulated I/O, all segments of the LCD are turned off in the all off mode.

```
<<< LCD driver demonstration start >>>
on0
on1
off0
off1
exit
<<< LCD driver demonstration finish >>>
```

Figure 3.10.2 LCD Driver Sample Program Screen Display Example

3.11 Sleep/Halt Mode Switching

3.11.1 Specifications of the Sample Software

The Sleep/Halt mode switching sample program executes the following items.

- The CPU enters the halt mode by executing the halt instruction.
- The halt mode of the CPU is cleared by an 8-bit timer interrupt.
- The CPU enters the sleep mode by executing the sleep instruction.
- The sleep mode of the CPU is cleared by a port interrupt.
- The CPU enters the sleep mode by executing the sleep instruction.
- The sleep mode of the CPU is cleared by an RTC interrupt.

3.11.2 Hardware Conditions

This sample program uses the OSC3B(2MHz) internal oscillator of the microcontroller.
When you use this sample program, connect the ports of the microcontroller as shown in the following figure.

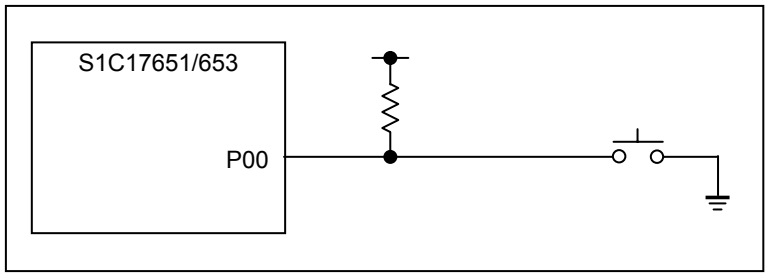

Figure 3.11.1 Sleep/Halt Sample Program Screen Display Example

3.11.3 Summary of the Operation

1. The 8-bit timer is set, and the CPU enters the halt mode.
2. When an 8-bit timer interrupt is generated, the halt mode is cleared, and a message is displayed on the Simulated I/O output window.
3. When an 8-bit timer interrupt is generated five times, the CPU enters the sleep mode.
4. When the P00 port gets Low, the sleep mode is cleared.
5. When an RTC interrupt is generated, the sleep mode is cleared.

```
<<< Sleep/halt demonstration start >>>
go to halt mode
return from halt mode
…
go to sleep mode(SW P00)
return from sleep mode
go to sleep mode(RTC)
return from sleep mode
<<< Sleep/halt demonstration finish >>>
```

Figure 3.11.2 Sleep/Halt Mode Switching Sample Program Screen Display Example

3.12 Sound Generator (SND)

3.12.1 Specifications of the Sample Software

The SND sample program uses the SND and executes the following items.

- Using the envelope mode of the SND, waveforms are output to the BZOUT pin at decreasing frequency.

3.12.2 Hardware Conditions

This sample program uses the OSC3B(2MHz) internal oscillator and the OSC1A(32.768kHz) of the microcontroller.

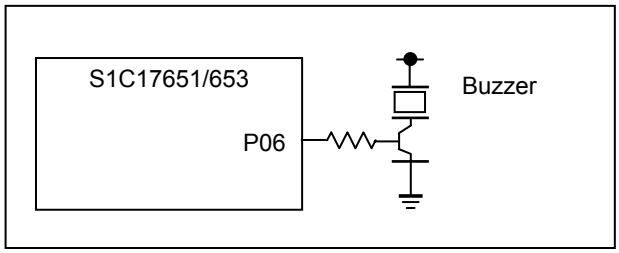

Figure 3.12.1 Hardware Connection Diagram for the Sound Generator Sample Programs

3.12.3 Summary of the Operation

1. The buzzer frequency is set to “4096.0Hz”. The newly set buzzer frequency is displayed on the Simulated I/O output window.
2. The buzzer frequency is displayed on the Simulated I/O output window.
3. The SND is set to the envelope mode and outputs a buzzer.
4. Due to the envelope mode, the duty ratio is changed from Level 1 (maximum) to Level 8 (minimum) automatically, and the buzzer stops at Level 8.
5. The buzzer frequency is decremented. The newly set buzzer frequency is displayed on the Simulated I/O output window.
6. A buzzer is output for one second. The procedure goes back to the step 2. This procedure is repeated until the buzzer frequency reaches “1170.3Hz”.
7. When the buzzer frequency is set to “1170.3Hz” and the buzzer output is completed, then the sample program terminates.

```
<<<Sound Generator demonstration start >>>
***Buzzer Frequncy : 4096.0Hz***
***Buzzer Frequncy : 3276.8Hz***
…
***Buzzer Frequncy : 1170.3Hz***
<<< Sound Generator demonstration finish >>>
```

Figure 3.12.2 Sound Generator Sample Programs Screen Display Example

3.13 Current Consumption

3.13.1 Specifications of the Sample Software

The current measurement sample program is used to measure the current consumption in the following cases.

- The microcontroller is in the Halt mode while only the OSC1A is oscillating and both the RTC and the PCLK are stopped.
- The microcontroller is in the Halt mode while only the OSC1A is oscillating, the RTC is running, and the PCLK is stopped.
- The microcontroller is in the Sleep mode while the RTC is stopped.
- The microcontroller is in the Sleep mode while the RTC is running.

3.13.2 Hardware Conditions

This sample program uses the OSC1A(32.768kHz) in the microcontroller.

3.13.3 Summary of the Operation

This sample program can change its operation by using the definition described in the table 3.13.1. By default, HALT_OSC1A is selected in the symbol definition for the project. If you want to change the operation of the program, change the symbol definition. For information about how to change the symbol definition, refer to the S5U1C17001C Manual.

Table 3.13.1 List of Settings for the Current Consumption Evaluation Program

Setting	Operation
HALT_OSC1A	The microcontroller enters the Halt mode while only the OSC1A is oscillating and both the RTC and the PCLK are stopped.
HALT_OSC1A_RTC	The microcontroller enters the Halt mode while only the OSC1A is oscillating, the RTC is running, and the PCLK is stopped.
SLEEP_ONLY	The microcontroller enters the Sleep mode while the RTC is stopped.
SLEEP_RTC	The microcontroller enters the Sleep mode while the RTC is running.

1. Write the sample program used for the measurement to the Flash memory beforehand.
2. Turn on the microcontroller and execute the sample program. Then measure the current.

3.14 Supply Voltage Detection Circuit (SVD)

3.14.1 Specifications of the Sample Software

The real-time clock sample program uses the real-time clock and executes the following items.

- The microcontroller is in the Halt mode while only the OSC1A is oscillating and both the RTC and the PCLK are stopped.
- The sample program menu of the real-time clock is displayed.
- The time is read from the real-time clock.
- The time of the real-time clock is set.
- The number of real-time clock interrupts is displayed.

3.14.2 Hardware Conditions

This sample program uses the OSC3B(2MHz) internal oscillator of the microcontroller.
When you use this sample program, set up the connection as shown in the following figure.

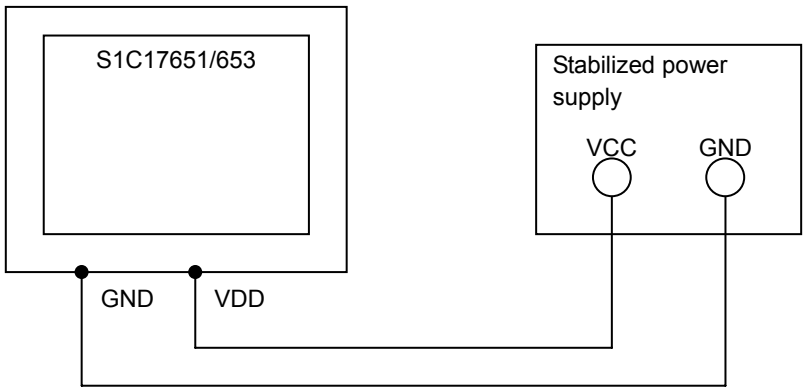

Figure 3.14.1 Hardware Connection Diagram for the Supply Voltage Detection Circuit Port Sample Programs

3.14.3 Summary of the Operation

1. The comparison voltage is set to “3.20V”.
2. A message that the comparison voltage is set to “3.20V” is displayed on the Simulated I/O output window.
3. The supply voltage is compared to the comparison voltage, and the result is shown on the Simulated I/O output window. If the supply voltage is lower than the comparison voltage, “VDD is smaller than comparison voltage.” is displayed. If the supply voltage is higher than the comparison voltage, “VDD is more than comparison voltage.” is displayed.

```
<<< SVD demonstration start >>>
Comparison Voltage:3.20V
VDD is smaller than comparison voltage.
Or
VDD is more than comparison voltage.
<<< SVD demonstration finish >>>
```

Figure 3.14.2 Supply Voltage Detection Circuit Sample Program Screen Display Example

3.15 MISC

3.15.1 Summary of the Operation

During debug, add the following settings at the beginning of the sample program.
1. Configure the wait cycles for the Flash memory.
2. Select the status of the peripherals running on the PCLK during the debug mode.
3. Select the status of the peripherals not running on the PCLK during the debug mode.
4. Configure the gear ratio when dividing the system clock.
5. Configure the clock supply for the internal peripheral modules.
6. Initialize the I/O pin functions.

When the target runs as a stand-alone system, the DCLK/DSIO/DST2 pins are configured as I/O ports (P11/P12/P13) at the step 6.

3.16 Theoretical Regulation (TR)

3.16.1 Specifications of the Sample Software

The theoretical regulation sample program executes the following items.
The clock timer starts, and the regulated clock (F256) is output to the REGMON pin.

3.16.2 Hardware Conditions

This sample program uses the OSC3B(2MHz) internal oscillator and the OSC1A(32.768kHz) of the microcontroller.
When you use this sample program, set up the connection as shown in the following figure.

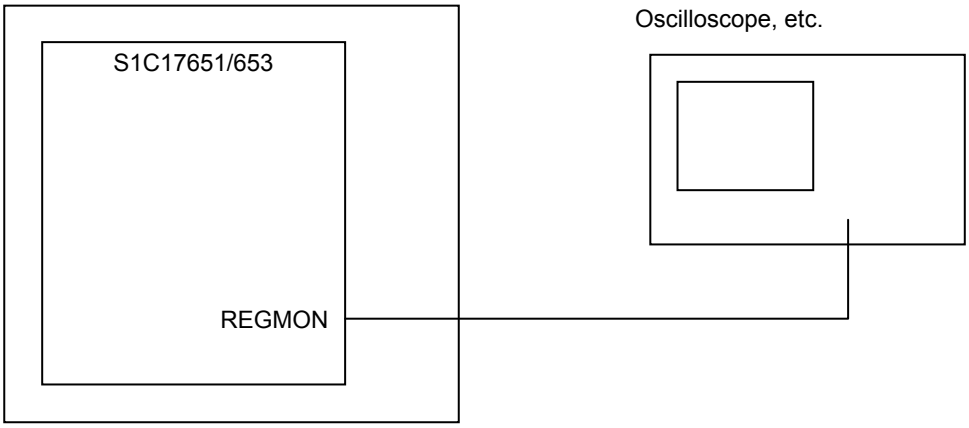

Figure 3.16.1 Hardware Connection Diagram for the Theoretical Regulation Sample Program

3.16.3 Summary of the Operation

1. The menu of the sample program is displayed on the Simulated I/O output window.
2. When "1" is entered and the ENTER key is pressed on the Simulated I/O while the menu is displayed, "-15/32768" is specified for the theoretical regulation value. You can specify the value between "1" and "32" (-15/32768 through +16/32768) for the theoretical regulation value.
3. After the theoretical regulation value is configured, theoretical regulation is performed every one second.
4. Theoretical regulation is performed for 10 seconds. After 10 seconds, the theoretical regulation value can be specified again.

```
<<< TR demonstration start >>>
1. -15/32768    2. -14/32768    3. -13/32768    4. -12/32768
5. -11/32768    6. -10/32768    7. -9/32768     8. -8/32768
9.  -7/32768    10. -6/32768    11. -5/32768    12. -4/32768
13. -3/32768    14. -2/32768    15. -1/32768    16. 0/32768
17.  +1/32768   18. +2/32768    19. +3/32768    20. +4/32768
21.  +5/32768   22. +6/32768    23. +7/32768    24. +8/32768
25.  +9/32768   26. +10/32768   27. +11/32768   28. +12/32768
29.  +13/32768  30. +14/32768   31. +15/32768   32. +16/32768
Please input number
<<< TR demonstration finish >>>
```

Figure 3.16.2 SVD Sample Program Screen Display Example

4. List Of Sample Driver Functions

This chapter lists the functions of each sample driver included in the sample software.

4.1 I/O Ports (P)

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

Table 4.1 List of the Functions of the I/O Port (P) Sample Driver

Function name	Purpose
SetPortInput	Configures the pull-up settings of the Px port
SetPortOutput	Configures the output settings of the Px port
SetPortOutputData	Configures the data output settings of the Px port
SetP0Chat	Configures the chattering filter settings of the P0 port
InitP0Int	Configures the interrupt initialization settings of the P0 port
EnableP0Int	Configures the interrupt enable settings of the P0 port
DisableP0Int	Configures the interrupt disable settings of the P0 port
isP0Int	Checks the interrupt causes of the P0 port
ClrP0IntFlg	Configures the flag clear settings for the interrupt causes of the P0 port

This sample driver is included in port.c and port.h.

If a program uses this sample driver, you need to include port.h.

4.2 Oscillator (OSC)

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

Table 4.2 List of the Functions of the Oscillator (OSC) Sample Driver

Function name	Purpose
StopOSC1	Stops the oscillation of the OSC1
StartOSC1	Starts the oscillation of the OSC1
StopOSC3A	Stops the oscillation of the OSC3A
StartOSC3A	Starts the oscillation of the OSC3A
StopOSC3B	Stops the oscillation of the OSC3B
StartOSC3B	Starts the oscillation of the OSC3B
SetWaitCycleClg	Configures the wait time when the clock oscillation starts
ChgOSC	Switches the system clock

This sample driver is included in clg.c and clg.h.

If a program uses this sample driver, you need to include clg.h.

4.3 8-bit Timer (T8)

Table 4.3 shows a list of the functions in this sample driver. For details about those functions, refer to the source code t8.c.

Table 4.3 List of the Functions of the 8-bit Timer (T8) Sample Driver

Function name	Purpose
InitT8	Initializes the 8-bit timer
GetT8Count	Reads the counter data
StartT8	Configures the 8-bit timer start settings
StopT8	Configures the 8-bit timer stop settings
InitT8Int	Configures the interrupt initialization settings of the 8-bit timer
EnableT8Int	Configures the interrupt enable settings of the 8-bit timer
DisableT8Int	Configures the interrupt disable settings of the 8-bit timer
isT8Int	Checks the 8-bit timer interrupt
ClrT8IntFlg	Configures the flag clear settings for the interrupt causes of the 8-bit timer

This sample driver is included in t8.c and t8.h.

If a program uses this sample driver, you need to include t8.h.

4.4 T16 PWM Timer (T16A2)

Table 4.4 shows a list of the functions in this sample driver. For details about those functions, refer to the source code t16a2.c

Table 4.4 List of the Functions of the T16 PWM Timer (T16A2) Sample Driver

Function name	Purpose
InitT16A2	Initializes the T16-bit PWM timer
DataInitT16A2	Configures the counter value of the T16-bit PWM timer
GetT16A2Count	Reads the counter data
StartT16A2	Configures the T16-bit PWM timer start settings
StopT16A2	Configures the T16-bit PWM timer stop settings
InitT16A2Int	Configures the interrupt initialization settings of the T16-bit PWM timer
EnableT16A2Int	Configures the interrupt enable settings of the T16-bit PWM timer
DisableT16A2Int	Configures the interrupt disable settings of the T16-bit PWM timer
isT16A2Int	Checks the T16-bit PWM timer interrupt
ClrT16A2IntFlg	Configures the flag clear settings for the interrupt causes of the T16-bit PWM timer

This sample driver is included in t16a2.c and t16a2.h.

If a program uses this sample driver, you need to include t16a2.h.

4.5 Clock Timer (CT)

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

Table 4.5 List of the Functions of the Clock Timer (CT) Sample Driver

Function name	Purpose
ResetCTCount	Initializes the clock timer
StartCT	Configures the clock timer start settings
StopCT	Configures the clock timer stop settings
InitCTInt	Configures the interrupt initialization settings of the clock timer
EnableCTInt	Configures the interrupt enable settings of the clock timer
DisableCTInt	Configures the interrupt disable settings of the clock timer
isCTInt	Checks the clock timer interrupt
ClrCTIntFlg	Configures the flag clear settings for the interrupt causes of the clock timer

This sample driver is included in ct.c and ct.h.

If a program uses this sample driver, you need to include ct.h.

4.6 Real-Time Clock (RTC)

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

Table 4.6 List of the Functions of the Real-Time Clock (RTC) Sample Driver

Function name	Purpose
InitRTC	Initializes the RTC
StartRTC	Configures the RTC start settings
StopRTC	Configures the RTC stop settings
InitRTCInt	Configures the interrupt initialization settings of the RTC
EnableRTCInt	Configures the interrupt enable settings of the RTC
DisableRTCInt	Configures the interrupt disable settings of the RTC
isRTCInt	Checks the RTC interrupt
ClrRTCIntFlg	Configures the flag clear settings for the interrupt causes of the RTC

This sample driver is included in rtc.c and rtc.h.

If a program uses this sample driver, you need to include rtc.h.

4.7 Watchdog Timer (WDT)

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

Table 4.7 List of the Functions of the Watchdog Timer (WDT) Sample Driver

Function name	Purpose
InitWDT	Initializes the WDT
StartWDT	Configures the WDT start settings
StopWDT	Configures the WDT stop settings
ResetWDT	Configures the WDT reset settings
isWDTnmi	Checks the NMI generated by the WDT

This sample driver is included in wdt.c and wdt.h.

If a program uses this sample driver, you need to include wdt.h.

4.8 UART

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

Table 4.8 List of the Functions of the UART Sample Driver

Function name	Purpose
InitUART	Initializes the UART
SendDataUART	Configures the transmission data
ReceiveDataUART	Reads the received data
StartUART	Configures the start settings for UART transmission/reception
StopUART	Configures the stop settings for UART transmission/reception
EnableUARTInt	Configures the interrupt enable settings of the UART
DisableUARTInt	Configures the interrupt disable settings of the UART
InitUARTInt	Configures the interrupt initialization settings of the UART
isUARTInt	Checks the UART interrupt
ClrUARTIntFlg	Configures the flag clear settings for the interrupt causes of the UART

This sample driver is included in uart.c and uart.h.

If a program uses this sample driver, you need to include uart.h.

4.9 SPI

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

Table 4.9 List of the Functions of the SPI Sample Driver

Function name	Purpose
InitSPI	Initializes the SPI
SendDataSPI	Configures the transmission data
ReceiveDataSPI	Reads the received data
StartSPI	Configures the start settings for SPI transmission/reception
StopSPI	Configures the stop settings for SPI transmission/reception
EnableSPIInt	Configures the interrupt enable settings of the SPI
DisableSPIInt	Configures the interrupt disable settings of the SPI
InitSPIInt	Configures the interrupt initialization settings of the SPI
isSPIInt	Checks the SPI interrupt
ClrSPIIntFlg	Configures the flag clear settings for the interrupt causes of the SPI

This sample driver is included in spi.c and spi.h.

If a program uses this sample driver, you need to include spi.h.

4.10 LCD Driver (LCD)

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

Table 4.10 List of the Functions of the LCD Sample Driver

Function name	Purpose
InitLCDPower	Initializes the LCD power
InitLCD	Initializes the LCD
SetLCDDisplay1Seg	Turns on one segment
StartLCDClock	Configures the LCD clock supply start settings
StopLCDClock	Configures the LCD clock supply stop settings
InitLCDInt	Configures the interrupt initialization settings of the LCD
EnableLCDInt	Configures the interrupt enable settings of the LCD
DisableLCDInt	Configures the interrupt disable settings of the LCD
isLCDInt	Checks the LCD interrupt
ClrLCDIntFlg	Configures the flag clear settings for the interrupt causes of the LCD

This sample driver is included in lcd.c and lcd.h.

If a program uses this sample driver, you need to include lcd.h.

4.11 Sound Generator (SND)

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

Table 4.11 List of the Functions of the SND Sample Driver

Function name	Purpose
InitSND	Initializes the SND
StartSND	Configures the SND buzzer start settings
StopSND	Configures the SND buzzer stop settings

This sample driver is included in snd.c and snd.h.

If a program uses this sample driver, you need to include snd.h.

4.12 Supply Voltage Detection Circuit (SVD)

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

Table 4.12 List of the Functions of the SVD Sample Driver

Function name	Purpose
SetSVDCompVolt	Configures the SVD comparison voltage
StartSVD	Configures the SVD detection start settings
StopSVD	Configures the SVD detection stop settings
GetSVDResult	Reads the SVD detection results

This sample driver is included in svd.c and svd.h.

If a program uses this sample driver, you need to include svd.h.

4.13 MISC

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

Table 4.13 List of the Functions of the MISC Sample Driver

Function name	Purpose
DebugModePsc	Configures the status of the peripherals running on the PCLK during the debug mode
ControlClg	Configures the clock supply for the internal peripheral modules
SetClockGear	Configures the gear ratio when dividing the system clock
SetFlashcAccessCycle	Configures the wait cycles for the Flash memory.
DebugModeMisc	Configures the status of the peripherals not running on the PCLK during the debug mode
ProtectMisc	Configures the enable/disable of the MISC register write protection
SetMiscIramSize	Configures the internal RAM size
SetMiscVecAddress	Configures the vector table address
SetMUXTargetOnly	Configures the MUX settings when the target runs as a stand-alone system.

This sample driver is included in init.c and init.h.

If a program uses this sample driver, you need to include init.h.

4.14 Theoretical Regulation (TR)

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

Table 4.14 List of the Functions of the TR Sample Driver

Function name	Purpose
InitTR	Initializes the TR
StartTR	Configures the TR clock monitor output start settings
StopTR	Configures the TR clock monitor output stop settings
SetTRLogicAdjustVal	Configures the theoretical regulation value
SetTRtrig	Configures the execution of the theoretical regulation

This sample driver is included in tr.c and tr.h.

If a program uses this sample driver, you need to include tr.h.

Appendix A Multiplier/Divider

This chapter explains how to use the multiplier/divider.

A.1 Multiplication and Division Using the Multiplier/Divider

GNU17 has a coprocessor library to calculate multiplication and division using the multiplier/divider.

For information about how to use the coprocessor library, refer to the S5U1C17001C Manual.

A.2 MAC Operation Using the Multiplier/Divider

The following is a program to calculate a MAC operation using the multiplier/divider.

This program calculates a MAC operation (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

Rev. No.	Date	Page	Category	Contents
Rev 1.0	2012/07/06	All pages	new	New document

EPSON

International Sales Operations

AMERICA

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

EUROPE

EPSON EUROPE ELECTRONICS GmbH
Riesstrasse 15, 80992 Munich,
GERMANY
Phone: +49-89-14005-0 FAX: +49-89-14005-110

ASIA

EPSON (CHINA) CO., LTD.
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.
Unit 715-723, 7/F Trade Square, 681 Cheung Sha Wan Road,
Kowloon, Hong Kong.
Phone: +852-2585-4600 FAX: +852-2827-4346

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
5F, 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

IC Sales & Marketing Department
421-8, Hino, Hino-shi, Tokyo 191-8501, JAPAN
Phone: +81-42-587-5814 FAX: +81-42-587-5117

Document Code: 412394200
First Issue August 2012 in JAPAN Ⓗ