EPSON
EXCEED YOUR VISION

S1C17 Family Application Note

S1C17656
Peripheral Circuit
Sample Software

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

NOTICE

Table of Contents

1. Overview

This manual describes how to use the sample software and its functions. The S1C17656 sample software is provided to show usage examples of the peripheral circuits included in the S1C17656. Also refer to the device information, technical manual, and S5U1C17001C Manual.

1.1 Operating Environment

Please prepare the hardware/software tools shown below to run the S1C17656 sample software.

• A target board with an S1C17656 mounted

• S5U1C17001H (hereinafter referred to as “ICDmini”)

• S5U1C17001C (hereinafter referred to as “GNU17”)

Note: GNU17 V2.3.0 is used for operation checking of this sample software.

2. Descriptions of Sample Software

This chapter describes the file configuration and execution method of the S1C17656 sample software.

The S1C17656 sample software consists of the sample programs for checking each peripheral circuit operation and the sample driver for running each peripheral circuit.

2.1 Configuration of Directories and Files

The following shows the directory configuration of the S1C17656 sample software.

```
[s1c17656_xxx]
 │
 ├─ GNU17 project and other files
 ├─ ...
 ├─ ...
 │
 ├─ [inc]
 │   ├─ c17656_reg.h
 │   :
 │   └─ [reg]
 │       ├─ clg_656_reg.h      ct_reg.h
 │       ├─ flashc_reg.h       itc_reg.h
 │       ├─ lcd_reg.h          misc_reg.h
 │       ├─ mux_reg.h          port_656_reg.h
 │       ├─ rfc_reg.h          rtc_reg.h
 │       ├─ snd_reg.h          spi_reg.h
 │       ├─ svd_reg.h          t8_reg.h
 │       ├─ t16a2_reg.h        uart_reg.h
 │       └─ vd1_reg.h          wdt_reg.h
 │
 └─ [src]
     ├─ boot.c
     ├─ init.c
     ├─ init.h
     :
     └─ main.c
```

Figure 2.1 S1C17656 Sample Software Directory Configuration

(1) s1c17656_xxx directory

This directory contains the sub-directories in which the files related to GNU17 projects are located.

(2) inc directory

This directory contains the files in which model dependent information is defined.

- Header file that defines the S1C17656 register addresses and other information (c17656_reg.h)

(3) reg directory

This directory contains the header files in which each peripheral circuit register bit assignment is defined.

- Header files that define the S1C17656 peripheral circuit register bit assignments (e.g., clg_656_reg.h)

(4) src directory

This directory contains a microcontroller initialization program, peripheral circuit sample programs, sample drivers, and header files in which the constants used in the peripheral circuit sample programs and other conditions are defined.

- Initialization program file (boot.c)
- Peripheral circuit sample program file (main.c)
- Peripheral circuit sample driver files (e.g., init.c)
- Peripheral circuit header files (e.g., init.h)

2.2 Execution Method

Follow the procedure shown below to execute the S1C17656 sample software.

(1) Importing a project

Launch GNU17 and import the S1C17656 sample software project. For how to import a project, refer to Chapter 3, “Software Development Procedures,” in the S5U1C17001C Manual.

(2) Building the project

Build the S1C17656 sample software project using GNU17. For how to build a project, refer to Chapter 5, “GNU17 IDE,” in the S5U1C17001C Manual.

(3) Connecting ICDmini

Connect ICDmini to the PC and the target board, then turn the power supply to the target board on.

(4) Loading and executing the program using the debugger

Click on the “Debug” button of GNU17 to start debugging. The program is loaded to the S1C17656 and started. For how to use the debugger, refer to Chapter 10, “Debugger,” in the S5U1C17001C Manual.

3. Details of Sample Software Functions

This chapter describes details of the S1C17656 sample software functions.

3.1 I/O Port (P)

3.1.1 Sample Software Specifications

This sample software executes the processing shown below using the I/O ports embedded in the microcontroller.

- Configures the I/O ports as an input port with an interrupt function to detect if the input signal goes to a low level.
- Configures the I/O ports as an output port and outputs a high or low level signal.

The table below lists the I/O ports used in the sample software and their configurations.

Table 3.1.1 I/O Port Configuration List

Configuration	I/O port
Interrupt input port	P00 P01 P02 P03
Output port	P04 P05 P06 P07

3.1.2 Hardware Requirements

This sample software runs with the OSC3B (2 MHz) internal oscillator clock of the microcontroller. To confirm the I/O port operations by the sample software, connect an interrupt input circuit and equipment for monitoring output signals to the port used as shown in the figure below.

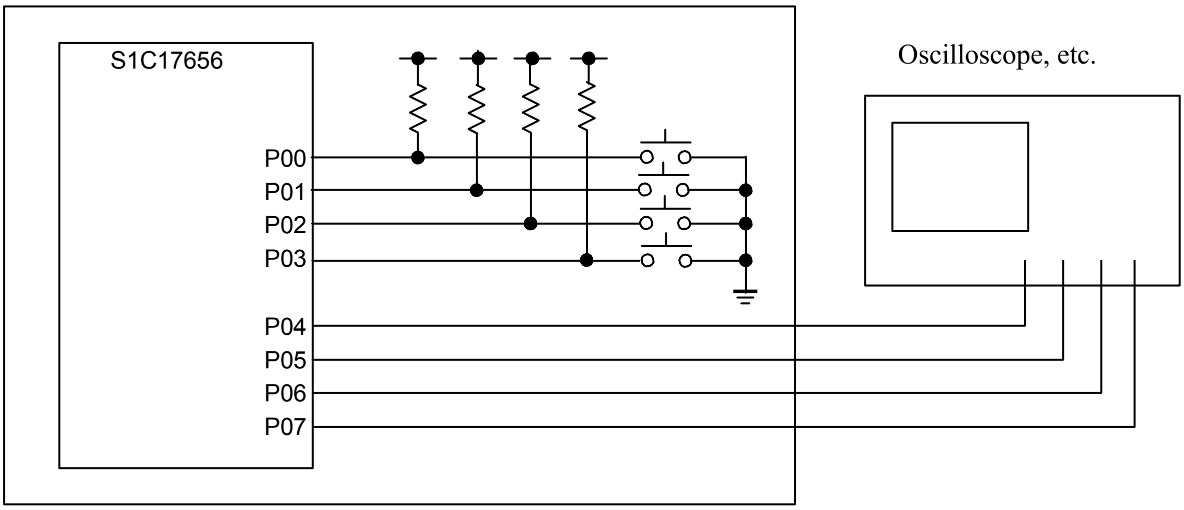

Figure 3.1.1 Hardware Connection Diagram for I/O Port Sample Software

3.1.3 Outline of Operations

1. When the P00 port input signal is set to a low level, “P00 Interrupt” appears on the simulated I/O view and the P04 port output level is inverted (port level goes low if the current level is high and vice versa).
2. When the P01 port input signal is set to a low level, “P01 Interrupt” appears on the simulated I/O view and the P05 port output level is inverted (port level goes low if the current level is high and vice versa).
3. When the P02 port input signal is set to a low level, “P02 Interrupt” appears on the simulated I/O view and the P06 port output level is inverted (port level goes low if the current level is high and vice versa).
4. When the P03 port input signal is set to a low level, “P03 Interrupt” appears on the simulated I/O view and the P07 port output level is inverted (port level goes low if the current level is high and vice versa). Then the sample software is terminated.

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

Figure 3.1.2 Display Example of I/O Port Sample Software

3.2 Clock Generator (CLG)

3.2.1 Sample Software Specifications

This sample software executes the processing shown below using the clock generator embedded in the microcontroller.

- Starts/stops the OSC1A oscillation.
- Starts/stops the OSC3B oscillation.
- Switches the system clock from OSC3B to OSC1A.
- Switches the system clock from OSC1A to OSC3B.

3.2.2 Hardware Requirements

To run this sample software, the OSC1A oscillator must be operated with an external crystal resonator connected. For the oscillator configuration including resonator connection, refer to the “Clock Generator (CLG)” chapter in the S1C17656 Technical Manual.

3.2.3 Outline of Operations

This sample software starts running from a status in which the system is running with the OSC3B clock.

1. After the simulated I/O view displays the numbers 1 to 9 at fixed intervals, OSC1A starts oscillating, the system clock is switched from OSC3B to OSC1A, and then OSC3B stops oscillating.
2. After the simulated I/O view displays the numbers 1 to 9 at fixed intervals, OSC3B starts oscillating, the system clock is switched from OSC1A to OSC3B, and then OSC1A stops oscillating.
3. After the simulated I/O view displays the numbers 1 to 9 at fixed intervals, the sample software is terminated.

```
<<< CLG(OSC) demonstration start >>>
OSC3B *** 1 ***
OSC3B *** 2 ***
…
OSC3B *** 9 ***
*** Change from OSC3B 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 Display Example of Clock Generator Sample Software

3.3 8-bit Timer (T8)

3.3.1 Sample Software Specifications

This sample software executes the processing shown below using the 8-bit timer embedded in the microcontroller.

- Generates an 8-bit timer interrupt to obtain the timer counter value.
- Puts the CPU into HALT mode to decrease power consumption while it is waiting for an interrupt.

3.3.2 Hardware Requirements

This sample software runs with the OSC3B (2 MHz) internal oscillator clock of the microcontroller.

3.3.3 Outline of Operations

1. 8-bit timer interrupts are enabled and the CPU is placed into HALT mode.
2. The CPU cancels HALT mode when an 8-bit timer interrupt has occurred.
3. The CPU is placed into HALT mode again after assigning the 8-bit timer counter data to an internal variable.
4. The 8-bit timer stops after an 8-bit timer interrupt has occurred 10 times.
5. The counter data values when the interrupts occurred are displayed on the simulated I/O view, and then the sample software is terminated.

```
<<< 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 Display Example of 8-bit Timer Sample Software

3.4 16-bit PWM Timer (T16A2)

3.4.1 Sample Software Specifications

This sample software executes the processing shown below using the 16-bit PWM timer embedded in the microcontroller.

• Generates a 16-bit PWM timer compare A interrupt to obtain the timer counter value.

• Generates a 16-bit PWM timer compare B interrupt to obtain the timer counter value.

• Puts the CPU into HALT mode to decrease power consumption while it is waiting for an interrupt.

3.4.2 Hardware Requirements

This sample software runs with the OSC3B (2 MHz) internal oscillator clock of the microcontroller.

3.4.3 Outline of Operations

1. Compare A and compare B interrupts are enabled and the 16-bit PWM timer starts counting.
2. The up counter value is read out when a compare A or compare B interrupt has occurred.
3. The 16-bit PWM timer stops after a compare B interrupt has occurred 5 times. The type of interrupts that occurred and the counter values are displayed on the simulated I/O view, and then the sample software is terminated.

```
<<< 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 3.4.1 Display Example of 16-bit PWM Timer Sample Software

3.5 Clock Timer (CT)

3.5.1 Sample Software Specifications

This sample software executes the processing shown below using the clock timer embedded in the microcontroller.

• Generates a clock timer interrupt to calculate the elapsed time.

• Puts the CPU into HALT mode to decrease power consumption while it is waiting for an interrupt.

3.5.2 Hardware Requirements

This sample software runs with the OSC3B (2 MHz) internal oscillator clock and the OSC1A (32.768 kHz) crystal oscillator clock of the microcontroller.

3.5.3 Outline of Operations

1. The clock timer starts operating.
2. When a clock timer interrupt has occurred, the elapsed time from starting of the clock timer is calculated and displayed on the simulated I/O view.
3. The sample software is terminated after a clock timer interrupt has occurred 10 times.

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

Figure 3.5.1 Display Example of Clock Timer Sample Software

3.6 Real-Time Clock (RTC)

3.6.1 Sample Software Specifications

This sample software executes the processing shown below using the real-time clock embedded in the microcontroller.

- Displays the menu of the real-time clock sample software.
- Acquires the current time data from the real-time clock.
- Sets a time to the real-time clock.
- Displays the number of real-time clock interrupts that occurred.

3.6.2 Hardware Requirements

This sample software runs with the OSC3B (2 MHz) internal oscillator clock and the OSC1A (32.768 kHz) crystal oscillator clock of the microcontroller.

3.6.3 Outline of Operations

1. The menu of the real-time clock sample software is displayed on the simulated I/O view.
2. The user sets a time via the simulated I/O.
3. After setting the above, a real-time clock interrupt occurs every second and the number of interrupts that occurred is displayed on the simulated I/O view.
4. The real-time clock stops after a real-time clock interrupt has occurred 10 times.

```
<<< 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 Display Example of Real-Time Clock Sample Software

3.7 Watchdog Timer (WDT)

3.7.1 Sample Software Specifications

This sample software executes the processing shown below using the watchdog timer embedded in the microcontroller.

• Resets the watchdog timer periodically.

• Generates an NMI using the watchdog timer.

3.7.2 Hardware Requirements

This sample software runs with the OSC3B (2 MHz) internal oscillator clock and the OSC1A (32.768 kHz) crystal oscillator clock of the microcontroller.

3.7.3 Outline of Operations

1. The watchdog timer and 8-bit timer start operating.
2. The watchdog timer is reset every time an 8-bit timer interrupt occurs.
3. The 8-bit timer stops after an 8-bit timer interrupt has occurred 10 times.
4. When the watchdog timer generates an NMI, its information is displayed on the simulated I/O view and the sample software is terminated.

```
<<< 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 Display Example of Watchdog Timer Sample Software

3.8 UART

3.8.1 Sample Software Specifications

This sample software executes the processing shown below using the UART embedded in the microcontroller.

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

3.8.2 Hardware Requirements

This sample software runs with the OSC3B (4 MHz) internal oscillator clock of the microcontroller.
The UART ports should be connected as shown in the figure below.

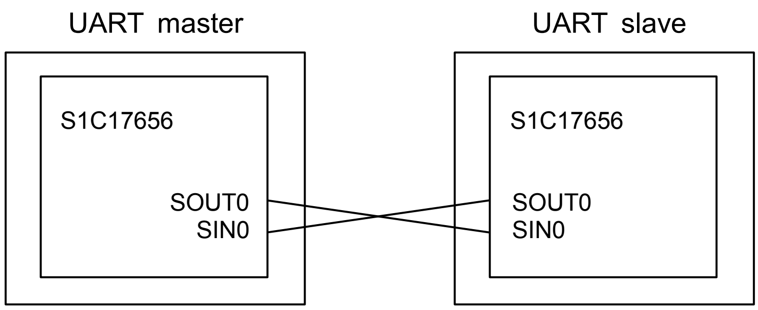

Figure 3.8.1 Hardware Connection Diagram for UART Sample Software

3.8.3 Outline of Operations

3.8.3.1 Outline of Operations by Master Sample Software

1. The UART communication conditions are initialized as follows:
   - Transfer rate: 222,222 bps
   - Data length: 8 bits
   - Stop bit: 1 bit
   - Parity: None
2. The clock timer is configured so that it will generate an interrupt in one second intervals.
3. “0x7f” is sent repeatedly in one second cycles until the connection acknowledge flag “0x7f” sent by the slave device is received.
4. Receiving the connection acknowledge flag stops sending “0x7f” and data of the ASCII codes from 0x21 to 0x7e are sent from the UART port.
5. After sending has finished, data is received via the UART port and the received data is displayed on the simulated I/O view.

```
<<< UART OSC3B demonstration start >>>
waiting connection.
connected
*** receive data ***
!"#$%&'()*+,-./0…
<<< UART OSC3B demonstration finish >>>
```

Figure 3.8.2 Display Example of UART (OSC3B) Master Sample Software

3.8.3.2 Outline of Operations by Slave Sample Software

1. The UART communication conditions are initialized as follows:
   - Transfer rate: 222,222 bps
   - Data length: 8 bits
   - Stop bit: 1 bit
   - Parity: None
2. Waits until the connection request flag “0x7f” is received.
3. After the connection request flag has been received, the connection acknowledge flag “0x7f” is sent to the master. Then data sent from the master is received.
4. After the reception above, data of the ASCII codes from 0x21 to 0x7e are sent from the UART port and the received data is displayed on the simulated I/O view.

```
<<< UART OSC3B demonstration start >>>
waiting connection.
connected
*** receive data ***
!"#$%&'()*+,-./0…
<<< UART OSC3B demonstration finish >>>
```

Figure 3.8.3 Display Example of UART (OSC3B) Slave Sample Software

3.9 SPI

3.9.1 Sample Software Specifications

3.9.1.1 SPI Master Sample Software Specifications

This sample software executes the processing shown below using the SPI master embedded in the microcontroller.

- Sends 8-byte data to an SPI slave.
- Receives 8-byte data from an SPI slave.
- Puts the CPU into HALT mode to decrease power consumption while it is waiting for an interrupt.

3.9.1.2 SPI Slave Sample Software Specifications

This sample software executes the processing shown below using the SPI slave embedded in the microcontroller.

- Receives 8-byte data from an SPI master.
- Sends 8-byte data to an SPI master.
- Puts the CPU into HALT mode to decrease power consumption while it is waiting for an interrupt.

3.9.2 Hardware Requirements

The master and slave sample software run with the OSC3B (2 MHz) internal oscillator clock of the microcontroller.

Use two S1C17656 microcontrollers, one with the SPI master sample software programmed and another with the SPI slave sample software programmed, as the SPI master and slave devices, respectively.

The SPI ports should be connected as shown in the figure below.

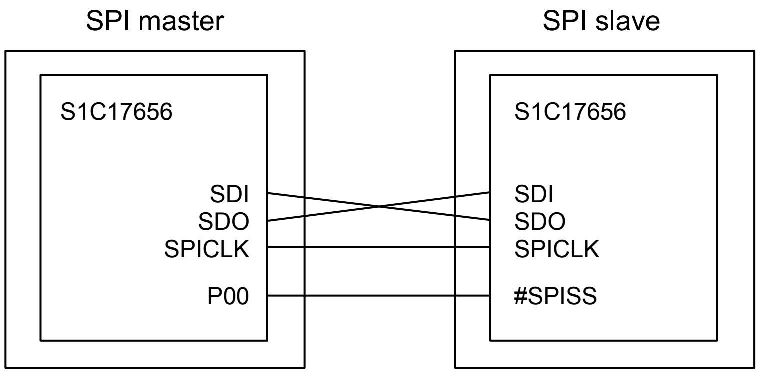

Figure 3.9.1 Hardware Connection Diagram for SPI Master/Slave Sample Software

3.9.3 Outline of Operations

3.9.3.1 Outline of Operations by SPI Master Sample Software

1. The SPI master is initialized.
2. “FROM MST” of 8-byte ASCII data is sent to the SPI slave.
3. The SPI clock is output to the SPI slave to receive data.
4. The sample software is terminated after displaying data received from the SPI slave on the simulated I/O view.

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

Figure 3.9.2 Display Example of SPI Master Sample Software

3.9.3.2 Outline of Operations by SPI Slave Sample Software

1. The SPI slave is initialized.
2. Waits until data sent from the SPI master is received.
3. After data sent from the SPI master has been received, “FROM SLV” of 8-byte ASCII data is sent to the SPI master.
4. The sample software is terminated after displaying data received from the SPI master on the simulated I/O view.

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

Figure 3.9.3 Display Example of SPI Slave Sample Software

3.10 LCD Driver (LCD)

3.10.1 Sample Software Specifications

This sample software executes the processing shown below using the LCD driver embedded in the microcontroller.

- Turns all LCD segments on and off in normal display mode.
- Turns all LCD segments on and off by setting the LCD driver into all on and all off modes.
- Turns all LCD segment off by setting the LCD driver into display off mode.

3.10.2 Hardware Requirements

This sample software runs with the OSC3B (2 MHz) internal oscillator clock of the microcontroller.

The LCD ports should be connected as shown in the figure below.

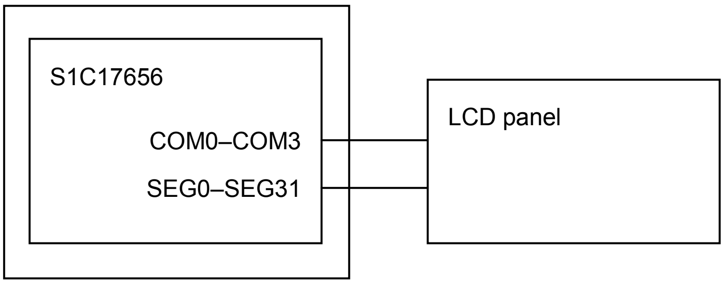

Figure 3.10.1 Hardware Connection Diagram for LCD Driver Sample Software

3.10.3 Outline of Operations

1. All the LCD segments go on in normal display mode by entering “on0” through the simulated I/O and pressing the ENTER key.
2. The LCD driver is placed into all on mode and all the LCD segments go on by entering “on1” through the simulated I/O and pressing the ENTER key.
3. All the LCD segments go off in normal display mode by entering “off0” through the simulated I/O and pressing the ENTER key.
4. The LCD driver is placed into all off mode and all the LCD segments go off by entering “off1” through the simulated I/O and pressing the ENTER key.
5. The LCD driver is placed into display off mode and all the LCD segments go off by entering “off2” through the simulated I/O and pressing the ENTER key.

```
<<< LCD driver demonstration start >>>
on0
on1
off0
off1
off2
exit
<<< LCD driver demonstration finish >>>
```

Figure 3.10.2 Display Example of LCD Driver Sample Software

3.11 Switching to SLEEP/HALT mode

3.11.1 Sample Software Specifications

This sample software executes the SLEEP/HALT mode switching processing shown below.

- Puts the CPU into HALT mode by executing the halt instruction.
- Releases the CPU from HALT mode using an 8-bit timer interrupt.
- Puts the CPU into SLEEP mode by executing the slp instruction.
- Releases the CPU from SLEEP mode using a port input interrupt.
- Releases the CPU from SLEEP mode using a real-time clock interrupt.

3.11.2 Hardware Requirements

This sample software runs with the OSC3B (2 MHz) internal oscillator clock and the OSC1A (32.768 kHz) crystal oscillator clock of the microcontroller.

A port interrupt input circuit should be connected as shown in the figure below.

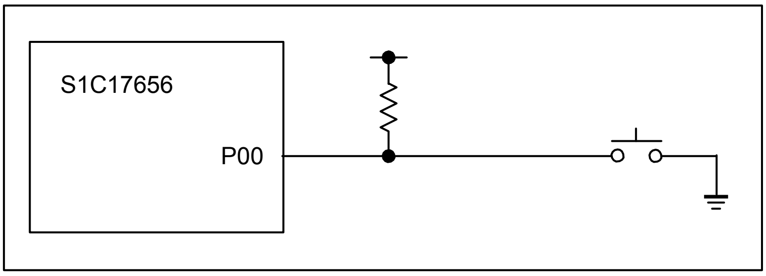

Figure 3.11.1 Hardware Connection Diagram for SLEEP/HALT Mode Switching Sample Software

3.11.3 Outline of Operations

1. The 8-bit timer is configured and the CPU is placed into HALT mode.
2. When an 8-bit timer interrupt has occurred, HALT mode is cancelled and its information is displayed on the simulated I/O view.
3. After an 8-bit timer interrupt has occurred five times, the CPU is placed into SLEEP mode.
4. SLEEP mode is cancelled by setting the P00 port to a low level.
5. The CPU is placed into SLEEP mode again.
6. SLEEP mode is cancelled when a real-time clock interrupt occurs.

```
<<< 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 Display Example of SLEEP/HALT Mode Switching Sample Software

3.12 Sound Generator (SND)

3.12.1 Sample Software Specifications

This sample software executes the processing shown below using the sound generator embedded in the microcontroller.

- Changes the buzzer frequency and outputs the buzzer signal in envelope mode to the BZ pin.

3.12.2 Hardware Requirements

This sample software runs with the OSC3B (2 MHz) internal oscillator clock and the OSC1A (32.768 kHz) crystal oscillator clock of the microcontroller.

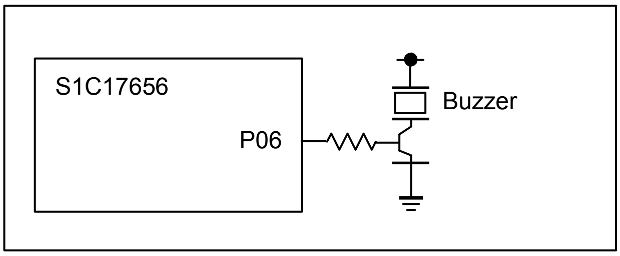

Figure 3.12.1 Hardware Connection Diagram for Sound Generator Sample Software

3.12.3 Outline of Operations

1. The buzzer frequency is set to 4,096.0 Hz.
2. The buzzer frequency is displayed on the simulated I/O view.
3. The buzzer signal is output for one second.
4. The buzzer signal is continuously output after setting envelope mode.
5. The buzzer duty ratio (volume level) is gradually changed from level 1 (maximum) to level 8 (minimum) automatically in envelop mode, and the buzzer signal output is terminated at level 8.
6. The buzzer frequency is decreased one step and the sequence returns to Step 2. This processing is repeated until the buzzer frequency reaches 1,170.3 Hz.
7. The sample software is terminated after the 1,170.3 Hz buzzer signal output is completed.

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

Figure 3.12.2 Display Example of Sound Generator Sample Software

3.13 Evaluating Current Consumption

3.13.1 Sample Software Specifications

This sample software creates the conditions shown below for evaluating current consumption.

- Puts the microcontroller into HALT mode with only the OSC1A being oscillated, RTC and PCLK deactivated.
- Puts the microcontroller into HALT mode with only the OSC1A being oscillated, RTC being operated, and PCLK deactivated.
- Puts the microcontroller into SLEEP mode with RTC deactivated.
- Puts the microcontroller into SLEEP mode with RTC being operated.

3.13.2 Hardware Requirements

This sample software runs with the OSC3B (2 MHz) internal oscillator clock and the OSC1A (32.768 kHz) crystal oscillator clock of the microcontroller.

3.13.3 Outline of Operations

The evaluation condition to be created by the sample software can be configured by defining a symbol listed in Table 3.13.1. By default, HALT_OSC1A is defined in the project. Modify this symbol definition to change the condition. For more information on symbol definition, refer to the S5U1C17001C Manual.

Table 3.13.1 Configuration of Current Consumption Evaluation Sample Software

Symbol to be defined	Condition created
HALT_OSC1A	OSC1A: oscillating (other oscillators: deactivated), RTC: deactivated, PCLK: deactivated, CPU: HALT status
HALT_OSC1A_RTC	OSC1A: oscillating (other oscillators: deactivated), RTC: operating, PCLK: deactivated, CPU: HALT status
SLEEP_ONLY	RTC: deactivated, CPU: SLEEP status
SLEEP_RTC	RTC: operating, CPU: SLEEP status

1. Program the current consumption evaluation sample software to the flash memory in advance.
2. Measure the current value after turning the microcontroller on and executing the sample software.

3.14 Supply Voltage Detection Circuit (SVD)

3.14.1 Sample Software Specifications

This sample software executes the processing shown below using the supply voltage detection circuit embedded in the microcontroller.

• Sets the comparison voltage and checks whether the V_{DD} value is lower than the comparison voltage or not.

3.14.2 Hardware Requirements

This sample software runs with the OSC3B (2 MHz) internal oscillator clock of the microcontroller. To evaluate the supply voltage detection operation, a stabilized power supply should be connected to the microcontroller as shown in the figure below.

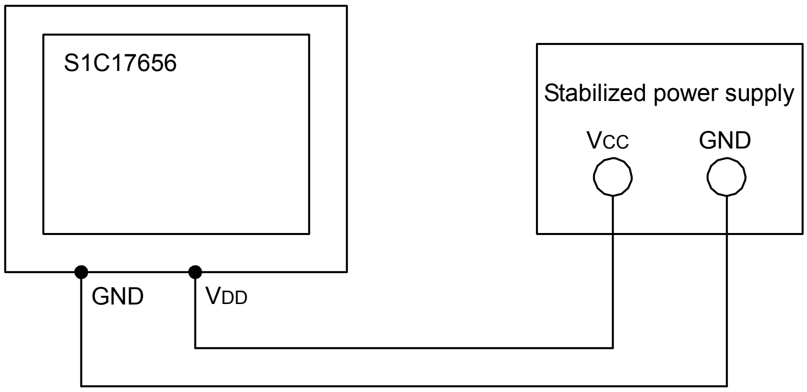

Figure 3.14.1 Hardware Connection Diagram for Supply Voltage Detection Circuit Sample Software

3.14.3 Outline of Operations

1. The comparison voltage is set to 3.20 V.
2. Information that the comparison voltage is set to 3.20 V is displayed on the simulated I/O view.
3. The detected voltage is compared with the comparison voltage and the result is displayed on the simulated I/O view.

```
<<< 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 Display Example of Supply Voltage Detection Circuit Sample Software

3.15 Misc Configuration

3.15.1 Outline of Processing

The Misc configuration routine configures the conditions shown below at the beginning of the sample software.

1. Sets the number of wait cycles for flash memory read.
2. Selects the condition of the peripheral circuits, that operate with PCLK, in debug mode.
3. Selects the condition of the peripheral circuits, that operate with a clock other than PCLK, in debug mode.
4. Sets the gear ratio for reducing system clock speed.
5. Configures clock supplies to the internal peripheral modules.

3.16 R/F Converter (RFC)

3.16.1 Sample Software Specifications

This sample software executes the processing shown below using the R/F converter embedded in the microcontroller.

- Converts the sensor resistance value into a digital value by oscillating the sensor resistance in the CR oscillation circuit and counting the oscillation clock.
- Example of external parts: 10 kΩ reference resistor, 15 kΩ resistive sensor, 1,000 pF reference capacitor

3.16.2 Hardware Requirements

This sample software runs with the OSC3B (2 MHz) internal oscillator clock and the OSC1A (32.768 kHz) crystal oscillator clock of the microcontroller.

Connect external parts as shown in the figure below to run the R/F converter sample software.

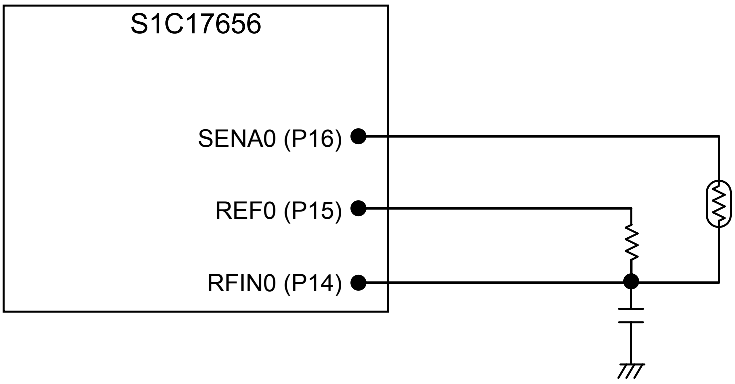

Figure 3.16.1 Hardware Connection Diagram for R/F Converter Sample Software

3.16.3 Outline of Operations

1. The R/F converter is configured to external clock input mode and the continuous oscillation function is enabled.
2. The initial value 0x00fc0000 is set to the R/F converter measurement counter and measurement starts.
3. After the reference oscillation and sensor oscillation have completed, the measurement counter value is read out as the conversion result, and then the sample software is terminated.

```
<<< R/F Converter demonstration start >>>
    *** Measurement Counter : 262166 ***
<<< R/F Converter demonstration finish >>>
```

Figure 3.16.2 Display Example of R/F Converter Sample Software

4. List of Sample Driver Functions

This chapter lists the sample drivers of each peripheral circuit.

4.1 I/O Port (P)

Table 4.1 lists the sample driver functions for the I/O port. For details of the functions, see the port.c source code.

Table 4.1 List of Sample Driver Functions for I/O Port

Function name	Function
SetPortInput	Px port pull-up setting function
SetPortOutput	Px port output setting function
SetPortOutputData	Px port output data setting function
SetP0Chat	P0 port chattering filter setting function
InitP0Int	P0 port interrupt initialization function
EnableP0Int	P0 port interrupt enabling function
DisableP0Int	P0 port interrupt disabling function
isP0Int	P0 port interrupt check function
ClrP0IntFlg	P0 port interrupt flag clear function

These sample drivers are defined in port.c and port.h. Include port.h into the program that uses these sample drivers.

4.2 Clock Generator (CLG)

Table 4.2 lists the sample driver functions for the clock generator. For details of the functions, see the clg.c source code.

Table 4.2 List of Sample Driver Functions for Clock Generator

Function name	Function
StopOSC1	OSC1 oscillation stop function
StartOSC1	OSC1 oscillation start function
StopOSC3B	OSC3B oscillation stop function
StartOSC3B	OSC3B oscillation start function
SetWaitCycleClg	Oscillation stabilization wait time setting function
ChgOSC	System clock switching function

These sample drivers are defined in clg.c and clg.h. Include clg.h into the program that uses these sample drivers.

4.3 8-bit Timer (T8)

Table 4.3 lists the sample driver functions for the 8-bit timer. For details of the functions, see the t8.c source code.

Table 4.3 List of Sample Driver Functions for 8-bit Timer

Function name	Function
InitT8	8-bit timer initialization function
GetT8Count	8-bit timer counter data acquisition function
StartT8	8-bit timer start function
StopT8	8-bit timer stop function
InitT8Int	8-bit timer interrupt initialization function
EnableT8Int	8-bit timer interrupt enabling function
DisableT8Int	8-bit timer interrupt disabling function
isT8Int	8-bit timer interrupt check function
ClrT8IntFlg	8-bit timer interrupt flag clear function

These sample drivers are defined in t8.c and t8.h. Include t8.h into the program that uses these sample drivers.

4.4 16-bit PWM Timer (T16A2)

Table 4.4 lists the sample driver functions for the 16-bit PWM timer. For details of the functions, see the t16a2.c source code.

Table 4.4 List of Sample Driver Functions for 16-bit PWM Timer

Function name	Function
InitT16A2	16-bit PWM timer initialization function
DataInitT16A2	16-bit PWM timer counter preset function
GetT16A2Count	16-bit PWM timer counter data acquisition function
StartT16A2	16-bit PWM timer start function
StopT16A2	16-bit PWM timer stop function
InitT16A2Int	16-bit PWM timer interrupt initialization function
EnableT16A2Int	16-bit PWM timer interrupt enabling function
DisableT16A2Int	16-bit PWM timer interrupt disabling function
isT16A2Int	16-bit PWM timer interrupt check function
ClrT16A2IntFlg	16-bit PWM timer interrupt flag clear function

These sample drivers are defined in t16a2.c and t16a2.h. Include t16a2.h into the program that uses these sample drivers.

4.5 Clock Timer (CT)

Table 4.5 lists the sample driver functions for the clock timer. For details of the functions, see the ct.c source code.

Table 4.5 List of Sample Driver Functions for Clock Timer

Function name	Function
ResetCTCount	Clock timer reset function
StartCT	Clock timer start function
StopCT	Clock timer stop function
InitCTInt	Clock timer interrupt initialization function
EnableCTInt	Clock timer interrupt enabling function
DisableCTInt	Clock timer interrupt disabling function
isCTInt	Clock timer interrupt check function
ClrCTIntFlg	Clock timer interrupt flag clear function

These sample drivers are defined in ct.c and ct.h. Include ct.h into the program that uses these sample drivers.

4.6 Real-Time Clock (RTC)

Table 4.6 lists the sample driver functions for the real-time clock. For details of the functions, see the rtc.c source code.

Table 4.6 List of Sample Driver Functions for Real-Time Clock

Function name	Function
InitRTC	Real-time clock initialization function
StartRTC	Real-time clock start function
StopRTC	Real-time clock stop function
InitRTCInt	Real-time clock interrupt initialization function
EnableRTCInt	Real-time clock interrupt enabling function
DisableRTCInt	Real-time clock interrupt disabling function
isRTCInt	Real-time clock interrupt check function
ClrRTCIntFlg	Real-time clock interrupt flag clear function

These sample drivers are defined in rtc.c and rtc.h. Include rtc.h into the program that uses these sample drivers.

4.7 Watchdog Timer (WDT)

Table 4.7 lists the sample driver functions for the watchdog timer. For details of the functions, see the wdt.c source code.

Table 4.7 List of Sample Driver Functions for Watchdog Timer

Function name	Function
InitWDT	Watchdog timer initialization function
StartWDT	Watchdog timer start function
StopWDT	Watchdog timer stop function
ResetWDT	Watchdog timer reset function
isWDTnmi	Watchdog timer NMI check function

These sample drivers are defined in wdt.c and wdt.h. Include wdt.h into the program that uses these sample drivers.

4.8 UART

Table 4.8 lists the sample driver functions for the UART. For details of the functions, see the uart.c source code.

Table 4.8 List of Sample Driver Functions for UART

Function name	Function
InitUART	UART initialization function
SendDataUART	UART sending data setting function
ReceiveDataUART	UART received data acquisition function
StartUART	UART transfer start function
StopUART	UART transfer stop function
EnableUARTInt	UART interrupt enabling function
DisableUARTInt	UART interrupt disabling function
InitUARTInt	UART interrupt initialization function
isUARTInt	UART interrupt check function
ClrUARTIntFlg	UART interrupt flag clear function

These sample drivers are defined in uart.c and uart.h. Include uart.h into the program that uses these sample drivers.

4.9 SPI

Table 4.9 lists the sample driver functions for the SPI. For details of the functions, see the spi.c source code.

Table 4.9 List of Sample Driver Functions for SPI

Function name	Function
InitSPI	SPI initialization function
SendDataSPI	SPI sending data setting function
ReceiveDataSPI	SPI received data acquisition function
StartSPI	SPI transfer start function
StopSPI	SPI transfer stop function
EnableSPIInt	SPI interrupt enabling function
DisableSPIInt	SPI interrupt disabling function
InitSPIInt	SPI interrupt initialization function
isSPIInt	SPI interrupt check function

These sample drivers are defined in spi.c and spi.h. Include spi.h into the program that uses these sample drivers.

4.10 LCD Driver (LCD)

Table 4.10 lists the sample driver functions for the LCD driver. For details of the functions, see the lcd.c source code.

Table 4.10 List of Sample Driver Functions for LCD Driver

Function name	Function
InitLCDPower	LCD drive power supply initialization function
InitLCD	LCD driver initialization function
SetLCDDisplay1Seg	LCD driver 1 segment display function
StartLCDClock	LCD driver clock supply start function
StopLCDClock	LCD driver clock supply stop function
InitLCDInt	LCD driver interrupt initialization function
EnableLCDInt	LCD driver interrupt enabling function
DisableLCDInt	LCD driver interrupt disabling function
isLCDInt	LCD driver interrupt check function
ClrLCDIntFlg	LCD driver interrupt flag clear function

These sample drivers are defined in lcd.c and lcd.h. Include lcd.h into the program that uses these sample drivers.

4.11 Sound Generator (SND)

Table 4.11 lists the sample driver functions for the sound generator. For details of the functions, see the snd.c source code.

Table 4.11 List of Sample Driver Functions for Sound Generator

Function name	Function
InitSND	Sound generator initialization function
StartSND	Sound generator buzzer output start function
StopSND	Sound generator buzzer output stop function

These sample drivers are defined in snd.c and snd.h. Include snd.h into the program that uses these sample drivers.

4.12 Supply Voltage Detection Circuit (SVD)

Table 4.12 lists the sample driver functions for the supply voltage detection circuit. For details of the functions, see the svd.c source code.

Table 4.12 List of Sample Driver Functions for Supply Voltage Detection Circuit

Function name	Function
SetSVDCompVolt	Supply voltage detection circuit comparison voltage setting function
StartSVD	Supply voltage detection circuit detection start function
StopSVD	Supply voltage detection circuit detection stop function
GetSVDResult	Supply voltage detection circuit detection result acquisition function

These sample drivers are defined in svd.c and svd.h. Include svd.h into the program that uses these sample drivers.

4.13 Misc Configuration

Table 4.13 lists the sample driver functions for misc configuration. For details of the functions, see the init.c source code.

Table 4.13 List of Sample Driver Functions for Misc Configuration

Function name	Function
DebugModePsc	Peripheral circuit (PCLK used) condition in debug mode setting function
ControlClg	Peripheral circuit clock supply control function
SetClockGear	System clock gear ratio setting function
SetFlashcAccessCycle	Number of wait cycles for flash memory read setting function
DebugModeMisc	Peripheral circuit (PCLK not used) condition in debug mode setting function
ProtectMisc	MISC register write protection setting/releasing function
SetMiscIramSize	Internal RAM size setting function
SetMiscVecAddress	Vector table base address setting function

These sample drivers are defined in init.c and init.h. Include init.h into the program that uses these sample drivers.

4.14 R/F Converter (RFC)

Table 4.14 lists the sample driver functions for the R/F converter. For details of the functions, see the rfc.c source code.

Table 4.14 List of Sample Driver Functions for R/F Converter

Function name	Function
initRfc	R/F converter initialization function
startRfc	R/F converter start function
stopRfc	R/F converter stop function
setRfcMeasurementCounter	R/F converter measurement counter value setting function
getRfcMeasurementCounter	R/F converter measurement counter value read function
setRfcTimeBaseCounter	R/F converter time base counter value setting function
getRfcTimeBaseCounter	R/F converter time base counter value read function
runRfcConvertingOperation	R/F converter reference/sensor oscillation converting control function

These sample drivers are defined in rfc.c and rfc.h. Include rfc.h into the program that uses these sample drivers.

Revision History

Attachment-1

Rev. No.	Date	Page	Category	Contents
Rev 1.0	2014/09/16	All	New	New establishment

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: 412867700
Issue October 2014 in JAPAN Ⓛ