EPSON
EXCEED YOUR VISION

S1C17 Family Application Note

# S1C17500 Series Peripheral Circuit Sample Software

SEIKO EPSON CORPORATION

Rev.1.0

## NOTICE

# Table of Contents

# 1. Overview

This manual describes the usage method and operations of the sample software for the S1C17500 series.

The purpose of the S1C17500 series sample software is to demonstrate the usage example of each peripheral circuit built into the S1C17500 series microcontroller.

The S1C17500 series sample software is provided on a per model basis for installation convenience, but the basic operations of each function are the same.

Refer to the model information (S1C175xx sample program ReadMe), respective technical manuals, S5U1C17001C manual, and S1C17700 peripheral circuit sample software together.

## 1.1. Operating Environment

In order to run the S1C17500 sample software, prepare the following items.

- Board mounted with S1C17500
- S5U1C17001H (hereinafter ICDmini)
- S5U1C17001C (hereinafter GNU17)

Note: This sample software has been confirmed to operate on GNU17v2.0.0.

## 1.2. Applicable Models

The applicable models in this manual are as shown below.

Table 1.1 List of S1C17500 series applicable models

| Applicable Model Name | Non-Applicable Model |
|---|---|
| S1C17554 | S1C17501 |
| S1C17564 | S1C17511 |
| - | S1C17512 |

# 2. Explanation on Sample Software

This chapter describes the S1C17500 series sample software's file configuration and execution method.

The S1C17500 series sample software is made up of "sample software" that checks the operations of each peripheral circuit, and "sample drivers" that are the sample drivers of the respective peripheral circuits.

## 2.1 Directory Structure and File Structure

The S1C17500 series sample software's directory structure is shown below.

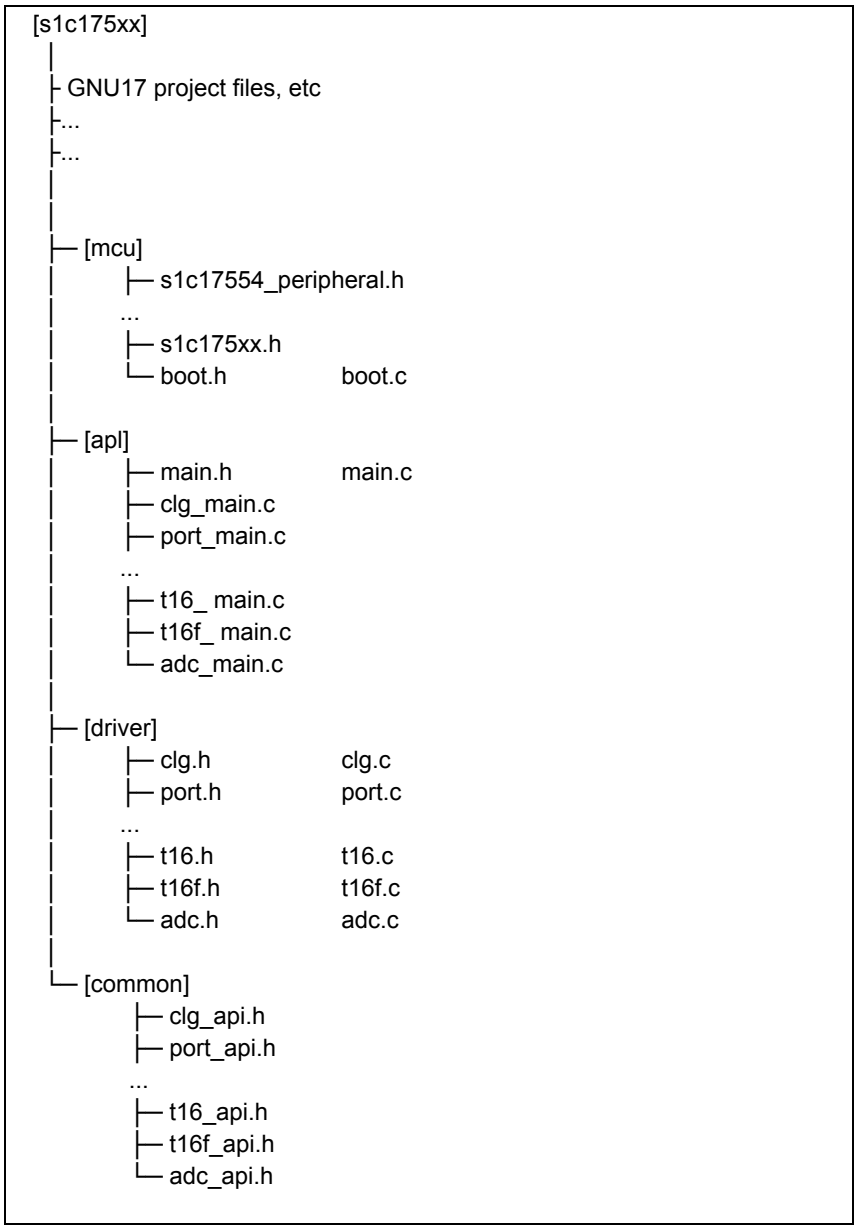


Figure 2.1 S1C17500 series sample software's directory structure diagram

(1) "s1c175xx" directory

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

(2) "mcu" directory

It contains files for microcontroller initialization and for defining model-dependent information.

- Header files that define the target model's register addresses, etc (s1c17554_peripheral.h, etc)
- Header file common to the models (s1c175xx.h)
- Initialization file (boot.c)

(3) "apl" directory

It contains sample software for the respective peripheral circuits and the header file that defines the definitions 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 sample drivers for the respective peripheral circuits.

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

(5) "common" directory

It contains header files that define the prototypes of functions provided externally by the sample drivers of respective peripheral circuits.

- Header file that defines the constants of arguments and prototypes of functions provided externally by the sample driver of each peripheral circuit (xxx.h)

For software using a sample driver, include the header files residing in the "common" directory and call the functions of the sample driver.

## 2.2 Execution Method

Execute the S1C17500 series sample software in the following sequence.

(1) Import the project

Start up GNU17 and import the S1C17500 series sample software project.

For details of project import method, refer to "software development steps" in the S5U1C17001C manual.

(2) Build the project

Build the S1C175xx project with GNU17.

For details of build method, refer to "GNU17 IDE" in the S5U1C17001C manual.

(3) Connect ICDmini

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

(4) Load and execute program using debugger

Press the [External Tools] button of GNU17 to start up the debugger, and press the [Continue] button of the debugger.

The program is loaded to S1C17500 and the program starts.

For details of debugger usage method, refer to "Debugger" in the S5U1C1001C manual.

## 2.3 Sample Software Menu

When the sample software starts up, a menu screen is displayed on the GNU17 Simulated I/O (hereinafter SimI/O).

Entering the program number and pressing the [ENTER] key starts the selected sample software.

Refer to Chapter 3 for the details of each sample software.

```
1.Port                    2.CLG
3.16bit timer             4.fine mode 16bit timer
...
Please input number.
>
```

Figure 2.2 Menu screen display example

## 2.4 Specific Module's Build Method

The S1C175xx sample software is distributed in the condition for building multiple programs.

It is possible to build only the sample software of needed peripheral module by modifying the source code of the sample software.

The steps are shown below.

(1) Files to be modified

Modify the per-model definition header.

For the case of the S1C17554 sample software, modify the s1c17554_peripheral.h file.

(2) Correction locations

Modify the following location at the bottom of the file.

```
//#undef PE_PORT
//#undef PE_T16
//#undef PE_T16F
//#undef PE_T16A
//#undef PE_CT
//#undef PE_SWT
//#undef PE_WDT
#undef   PE_UART
#undef   PE_UART_OSC3
#undef   PE_SPI
#undef   PE_SPI_MASTER
#undef   PE_SPI_SLAVE
#undef   PE_I2CM
#undef   PE_I2CS
#undef   PE_ADC
#undef   PE_REMC
#undef   PE_REMC_TX
#undef   PE_REMC_RX
#undef   PE_CURRENT_MEASURE
#undef   PE_SLEEP_HALT
```

Figure 2.3 Definition modification example of specific module

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

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

As an example, the I2CM sample software uses 16-bit timer, and when building the I2CM sample software, it is necessary to disable "#undef PE_I2CM" and "#undef PE_T16."

# 3. Details of Sample Software Functions

This chapter describes the function details of the S1C17500 series sample software.

## 3.1 Clock Generator (CLG)

### 3.1.1 Sample software specifications

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

- Do IOSC oscillation and stopping. (Only models with IOSC)
- Do OSC1 oscillation and stopping.
- Do OSC3 oscillation and stopping.
- Switch system clock from IOSC to OSC3. (Only models with IOSC)
- Switch system clock from OSC3 to OSC1.
- Switch system clock from OSC1 to IOSC (OSC3 for models without IOSC).

### 3.1.2 Hardware conditions

This sample software operates in conditions in which OSC1 and OSC3 can oscillate.

### 3.1.3 Operations overview

(1) Sample software operations overview

- This sample software starts operations in the condition of using IOSC (OSC3 for models without IOSC).
- After displaying "1," "2," "3"..., "9" to SimI/O at a fixed interval, it starts OSC3 oscillation, switches system clock from IOSC to OSC3, and stops IOSC. (Only models with IOSC)
- Next, after displaying "1," "2," "3"..., "9" to SimI/O at a fixed interval, it starts OSC1 oscillation, switches system clock from OSC3 to OSC1, and stops OSC3.
- Next, after displaying "1," "2," "3"..., "9" to SimI/O at a fixed interval, it starts IOSC (OSC3 for models without IOSC) oscillation, switches system clock from OSC1 to IOSC (OSC3 for models without IOSC), and stops OSC1.
- Next, it displays "1," "2," "3"..., and "9" to SimI/O at a fixed interval.

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

Figure 3.1 Clock generator (CLG) sample software 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.

## 3.2 Fine Mode 16-bit Timers (T16F)

### 3.2.1 Sample software specifications

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

- Fine mode 16-bit timer interrupt is generated, and the timer's counter value is acquired.
- While waiting for interrupt, the power consumption is reduced by placing the CPU into HALT mode.

### 3.2.2 Hardware conditions

This sample software operates in conditions in which OSC1 and OSC3 can oscillate.

### 3.2.3 Operations overview

(1) Sample software operations overview

- It starts fine mode 16-bit timer interrupt, and places CPU into HALT mode.
- Upon occurrence of fine mode 16-bit timer interrupt, it cancels CPU's HALT mode, stores the fine mode 16-bit timer's counter value to internal variable, and again places the CPU into HALT mode.
- Upon the $10^{th}$ occurrence of fine mode 16-bit timer interrupt, it stops the fine mode 16-bit timer interrupt, and displays the counter value at each interrupt occurrence to SimI/O.

```
<<< T16F timer demonstration start >>>
*** T16F interrupt 1 time, count data at this time: 32 ***
*** T16F interrupt 2 time, count data at this time: 32 ***
*** T16F interrupt 3 time, count data at this time: 32 ***
*** T16F interrupt 4 time, count data at this time: 32 ***
...
*** T16F interrupt 10 time, count data at this time: 32 ***
<<< T16F timer demonstration finish >>>
```

Figure 3.2 Fine mode 16-bit timer (T16F) sample software 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 PWM Timers (T16A)

### 3.3.1 Sample software specifications

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

- 16-bit PWM timer compare A match interrupt is generated, and the timer's counter value is acquired.
- 16-bit PWM timer compare B match interrupt is generated, and the timer's counter value is acquired.
- Waveform is outputted to TOUT0 terminal.
- While waiting for interrupt, the power consumption is reduced by placing the CPU into HALT mode.

Note: The TOUT0 terminal name may change depending on models. Check by referring to the source code of each model.

### 3.3.2 Hardware conditions

This sample software operates in conditions in which OSC1 and OSC3 can oscillate.

### 3.3.3 Operations overview

(1) Sample software operations overview

- It enables compare A match interrupt and compare B match interrupt, starts 16-bit PWM timer, and places CPU into HALT mode.
- Upon occurrence of compare A match interrupt and compare B match interrupt, it cancels CPU's HALT mode, stores the counter value of 16-bit PWM timer to internal variable, and again places the CPU into HALT mode.
- Upon the 5th occurrence of compare B match interrupt, it stops 16-bit PWM timer, and displays the interrupt type and counter value to SimI/O.

```
<<< 16 bit PWM timer demonstration start >>>
*** 16 bit PWM compare A interrupt: 633 ***
*** 16 bit PWM compare B interrupt: 126 ***
*** 16 bit PWM compare A interrupt: 633 ***
...
*** 16 bit PWM Interrupt B interrupt: 126 ***
<<< 16 bit PWM timer demonstration finish >>>
```

Figure 3.3 Advanced timer (T16A) sample software 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 Universal Serial Interface (USI)

### 3.4.1 Sample software specifications

This sample software performs the following operations using the universal serial interface.

- Use universal serial interface's UART to send data.
- Use universal serial interface's UART to receive data.
- Use universal serial interface's SPI master to send data to SPI slave.
- Use universal serial interface's SPI master to receive data from SPI slave.
- Use universal serial interface's I2C master to send data to I2C slave.
- Use universal serial interface's I2C master to receive data from I2C slave.
- Use universal serial interface's I2C slave to receive data from I2C master.
- Use universal serial interface's I2C slave to send data to I2C master.

### 3.4.2 Hardware conditions

This sample software operates in conditions in which OSC1 and OSC3 can oscillate.

Use this sample software by connecting the respective ports of the microcontroller as shown below.

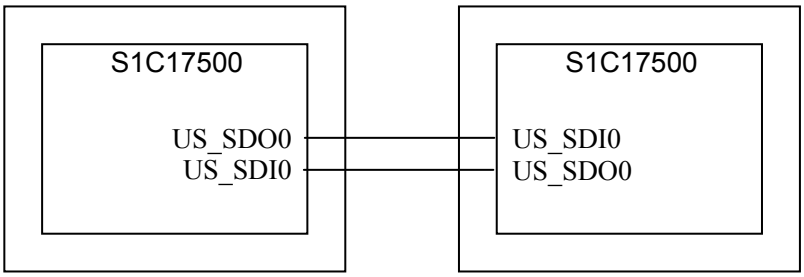


Figure 3.4 Sample software and hardware connection diagram when UART is selected

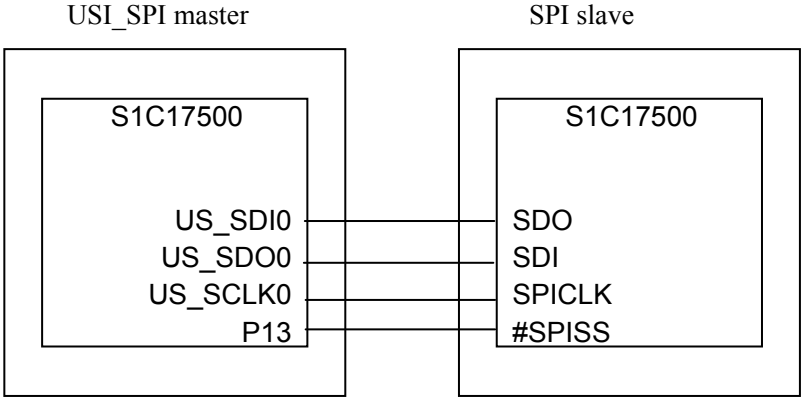


Figure 3.5 Sample software and hardware connection diagram when SPI master is selected

Note: Each terminal may change depending on models. Check the source code.

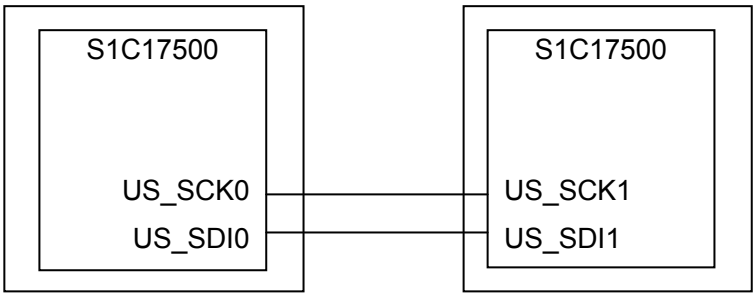


Figure 3.6 Sample software and hardware connection diagram when I2C master/slave is selected

### 3.4.3 Operations overview

(1) Sample software operations overview

- Inputting the UART's menu number from the menu and pressing the [ENTER] key performs sending and receiving using universal serial interface's UART.
- Inputting the SPI master's menu number from the menu and pressing the [ENTER] key performs sending and receiving using universal serial interface's SPI master.
- Inputting the I2C master's menu number from the menu and pressing the [ENTER] key performs sending and receiving using universal serial interface's I2C master.
- Inputting the I2C slave's menu number from the menu and pressing the [ENTER] key performs sending and receiving using universal serial interface's I2C slave.
- Inputting exit's menu number from the menu and pressing the [ENTER] key ends the sample software.

```
<<< USI demonstration start >>>
1.UART                  2.SPI master
3.I2C master            4.I2C slave
5.exit

<<< USI demonstration finish >>>
```

Figure 3.7 Universal serial interface (USI) sample software screen display example

(2) Stop method for sample Software

Inputting exit's menu number from this sample software's menu and pressing the [ENTER] key ends the sample software and it returns to the menu screen.

## 3.5 Electric Current Measurement

### 3.5.1 Sample software specifications

This sample software program is for evaluating the following conditions.

For models with built-in regulator, this is done by setting the built-in regulator to automatic control mode.

- Place CPU into SLEEP mode.
- Place CPU into HALT mode in the condition where only OSC1 is made to oscillate and PCLK is set to OFF.
- Place CPU into HALT mode in the condition where only OSC1 is made to oscillate.
- Place CPU into HALT mode in the condition where only OSC1/OSC3 are made to oscillate.
- Place CPU into HALT mode in the condition where only OSC1/IOSC are made to oscillate. (Only models with IOSC)

### 3.5.2 Hardware conditions

This sample software operates in conditions in which OSC1 and OSC3 can oscillate.

### 3.5.3 Operations overview

(1) Sample software operations overview

- Inputting [1] and pressing the [ENTER] key places CPU into SLEEP mode.
- Inputting [2] and pressing the [ENTER] key places CPU into HALT mode in the condition where only OSC1 is made to oscillate and PCLK is set to OFF.
- Inputting [3] and pressing the [ENTER] key places CPU into HALT mode in the condition where only OSC1 is made to oscillate.
- Inputting [4] and pressing the [ENTER] key places CPU into HALT mode in the condition where only OSC1 and OSC3 are made to oscillate.
- Inputting [5] and pressing the [ENTER] key places CPU into HALT mode in the condition where only OSC1 and IOSC are made to oscillate.

```
<<< Current measurement demonstration start >>>
1. Sleep                    2.Halt with OSC1, PCLK off
3.Halt with OSC1            4.Halt with OSC1/3
5.Halt with OSC1/IOSC

Please input number.
>1
sleeping
```

Figure 3.8 Electric current measurement sample software 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.

## 3.6 Other Functions

The following functions are included in this sample software. For the explanation of each function, refer to S1C17000 peripheral circuit sample software's document.

- I/O port (P)
- 16-bit timer (T16)
- Clock timer (CT)
- Stopwatch timer (SWT)
- Watchdog timer (WDT)
- UART that uses OSC3
- UART that uses IOSC (only models with IOSC)
- SPI master
- SPI slave
- I2C master (I2CM)
- I2C slave (I2CS)
- Remote controller sending (REMC)
- Remote controller receiving (REMC)
- A/D converter (ADC10)
- Sleep/Halt mode switching

# 4. List of Sample Driver Functions

Here, each peripheral circuit's sample driver listing is set forth.

## 4.1 I/O Port (P)

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

Table 4.1 List of I/O port (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 pullup resistance 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 set forth in port.c/port.h and port_api.h.

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

## 4.2 Clock Generator (CLG)

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

Table 4.2 List of clock generator (CLG) sample driver functions

| Function Name | Description Name |
|---|---|
| CLG_setClockSource | Clock source setting |
| CLG_setWaitCycle | Oscillation stability wait time setting |
| CLG_controlOscillation | OSC oscillation start/stop setting |
| CLG_setFOUTDivision | FOUTx clock setting |
| CLG_controlFOUT | FOUTx clock output allow/disallow setting |
| CLG_setIOSCfrequency | IOSC oscillation frequency setting |
| CLG_setPCLKEnable | PCLK supply allow/disallow setting |
| CLG_setCCLKGearRatio | System clock gear ratio setting |

This sample driver is set forth in clg.c/clg.h and clg_api.h.

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

## 4.3 16-bit Timers (T16)

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

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 set forth in t16.c/t16.h and t16_api.h.

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

## 4.4 Fine Mode 16-bit Timers (T16F)

Table 4.4 shows a list of this sample driver's functions. For details of the functions, refer to source code t16f.c.

Table 4.4 List of fine mode 16-bit timer (T16F) sample driver functions

| Function Name | Description Name |
|---|---|
| T16F_init | Fine mode 16-bit timer initialization |
| T16F_setInputClock | Prescaler output clock setting |
| T16F_setReloadData | Reload data setting |
| T16F_getCounterData | Counter data acquisition |
| T16F_setTimerMode | Fine mode 16-bit timer mode setting |
| T16F_resetTimer | Fine mode 16-bit timer reset |
| T16F_setTimerRun | Fine mode 16-bit timer start/stop setting |
| T16F_initInt | Fine mode 16-bit timer interrupt initialization |
| T16F_controlInt | Fine mode 16-bit timer interrupt allow/disallow setting |
| T16F_resetIntFlag | Fine mode 16-bit timer interrupt factor flag reset |
| T16F_checkIntFlag | Fine mode 16-bit timer interrupt factor flag check |

This sample driver is set forth in t16f.c/t16f.h and t16f_api.h.

For programs using this sample driver, include t16f_api.h.

## 4.5 16-bit PWM Timers (T16A)

Table 4.5 shows a list of this sample driver's functions. For details of the functions, refer to source code t16a.c.

Table 4.5 List of 16-bit PWM timer (T16A) sample driver functions

| Function Name | Description Name |
|---|---|
| T16A_setInputClock | Input clock setting |
| T16A_controlInputClock | Input clock supply allow/disallow setting |
| T16A_init | 16-bit PWM timer initialization |
| T16A_setTimerMode | 16-bit PWM timer mode setting |
| T16A_setComparatorCapture | Comparator/capture setting |
| T16A_getCounterData | Count data acquisition |
| T16A_setCompareData | Compare data setting |
| T16A_getCaptureData | Capture data acquisition |
| T16A_resetTimer | 16-bit PWM timer reset |
| T16A_setTimerRun | 16-bit PWM timer start/stop setting |
| T16A_initInt | 16-bit PWM timer interrupt initialization |
| T16A_controlInt | 16-bit PWM timer interrupt allow/disallow setting |
| T16A_resetIntFlag | 16-bit PWM timer interrupt factor flag reset |
| T16A_checkIntFlag | 16-bit PWM timer interrupt factor flag check |

This sample driver is set forth in t16a.c/t16a.h and t16a_api.h.

For programs using this sample driver, include t16a_api.h.

## 4.6 Clock timer (CT)

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

Table 4.6 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 set forth in ct.c/ct.h and ct_api.h.

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

## 4.7 Stopwatch Timer (SWT)

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

Table 4.7 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 set forth in swt.c/swt.h and swt_api.h.

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

## 4.8 Watchdog timer (WDT)

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

Table 4.8 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 set forth in wdt.c/wdt.h and wdt_api.h.

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

## 4.9 UART

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

Table 4.9 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 send/receive 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 |
| UART_setBaudRate | Baud rate setting |
| UART_setCountClock | UART count clock setting |
| UART_controlCountClock | UART count clock allow/disallow setting |

This sample driver is set forth in uart.c/uart.h and uart_api.h.

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

## 4.10 SPI

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

Table 4.10 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 send/receive allow/disallow setting |
| SPI_initInt | SPI interrupt initialization |
| SPI_controlInt | SPI interrupt allow/disallow setting |
| SPI_checkIntFlag | SPI interrupt factor flag check |
| SPI_checkBusyFlag | Send/receive BUSY flag check |

This sample driver is set forth in spi.c/spi.h and spi_api.h.

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

## 4.11 I2C Master (I2CM)

Table 4.11 shows a list of this sample driver's functions. For details of the functions, refer to source code i2cm.c.

Table 4.11 List of I2C master (I2CM) sample driver functions

| Function Name | Description Name |
|---|---|
| I2CM_init | I2C master initialization |
| I2CM_setComEnable | I2C master send/receive allow/disallow setting |
| I2CM_genCondition | Generation of start/stop condition |
| 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 set forth in i2cm.c/i2cm.h and i2cm_api.h.

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

## 4.12 I2C Slave (I2CS)

Table 4.12 shows a list of this sample driver's functions. For details of the functions, refer to source code i2cs.c.

Table 4.12 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 stretch function setting |
| I2CS_setAsyncDetection | Asynchronous address detection function setting |
| I2CS_setNoiseRemove | Noise elimination function 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 send/receive 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 set forth in i2cs.c/i2cs.h and i2cs_api.h.

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

## 4.13 Universal Serial Interface (USI)

### 4.13.1 Common to each mode

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

Table 4.13 List of universal serial interface (USI) common sample driver functions

| Function Name | Description Name |
|---|---|
| USI_init | USI initialization |
| USI_setTransmitData | Send data setting |
| USI_getReceiveData | Receive data acquisition |
| USI_initInt | USI interrupt initialization |

This sample driver is set forth in usi.c/usi.h and usi_api.h.

For programs using this sample driver, include usi_api.h.

### 4.13.2 UART mode

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

Table 4.14 List of universal serial interface (USI) UART mode sample driver functions

| Function Name | Description Name |
|---|---|
| USI_UART_init | UART initialization |
| USI_UART_controlInt | UART interrupt allow/disallow setting |
| USI_UART_resetIntFlag | UART interrupt factor flag reset |
| USI_UART_checkReceiveFlag | UART interrupt factor flag check |

This sample driver is set forth in usi_uart.c/usi_uart.h and usi_uart_api.h.

For programs using this sample driver, include usi_api.h.

### 4.13.3 SPI mode

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

Table 4.15 List of universal serial interface (USI) SPI mode sample driver functions

| Function Name | Description Name |
|---|---|
| USI_SPI_init | SPI initialization |
| USI_SPI_controlInt | SPI interrupt allow/disallow setting |
| USI_SPI_resetIntFlag | SPI interrupt factor flag reset |
| USI_SPI_checkIntFlag | SPI interrupt factor flag check |
| USI_SPI_checkBusyFlag | Send/receive BUSY flag check |
| USI_SPI_setReceiveMaskData | Receive master data setting |

This sample driver is set forth in usi_spi.c/usi_spi.h and usi_spi_api.h.

For programs using this sample driver, include usi_api.h.

### 4.13.4 I2C mode

Table 4.16 shows a list of this sample driver's functions. For details of the functions, refer to source code usi_i2c.c.

Table 4.16 List of universal serial interface (USI) I2C mode sample driver functions

| Function Name | Description Name |
|---|---|
| USI_I2C_setTrigger | I2C trigger setting |
| USI_I2C_checkStatus | I2C status check |
| USI_I2C_checkBusy | I2C operation status check |
| USI_I2C_controlInt | I2C interrupt allow/disallow setting |
| USI_I2C_resetIntFlag | I2C interrupt factor flag reset |
| USI_I2C_checkIntFlag | I2C interrupt factor flag check |

This sample driver is set forth in usi_i2c.c/usi_i2c.h and usi_i2c_api.h.

For programs using this sample driver, include usi_api.h.

## 4.14 Remote Controller (REMC)

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

Table 4.17 List of remote controller (REMC) sample driver functions

| Function Name | Description Name |
|---|---|
| REMC_init | REMC initialization |
| REMC_setEnable | REMC send/receive start/stop setting |
| REMC_setTransmitData | Send data setting |
| REMC_setReceiveLength | Receive data length setting |
| REMC_getReceiveData | Receive data acquisition |
| REMC_calcReceiveDataPulse | Receive 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 set forth in remc.c/remc.h and remc_api.h.

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

## 4.15 A/D Converter (ADC10)

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

Table 4.18 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 busy channel number acquisition |
| ADC_checkBusyStatus | A/D conversion busy check |
| ADC_controlTrigger | Software trigger control |
| ADC_adjustmentComparator | Comparator adjustment |
| 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 set forth in adc.c/adc.h and adc_api.h.

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

## 4.16 MISC

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

Table 4.19 List of MISC sample driver functions

| Function Name | Description Name |
|---|---|
| MISC_setDebugModeControl | Debug mode peripheral circuit operations setting |
| MISC_controlWriteProtect | MISC register write protect control |
| MISC_setIRAMSize | IRAM size setting |
| MISC_getIRAMSize | IRAM size acquisition |
| MISC_setTTBR | Vector table address setting |
| MISC_getPSR | PSR acquisition |

This sample driver is set forth in misc.c/misc.h and misc_api.h.

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

## 4.17 Multiplexer (MUX)

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

Table 4.20 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_setDBGport | Debug port setting |
| MUX_setCLGport | CLG port setting |
| MUX_setT16Aport | 16-bit PWM timer port setting |
| MUX_setUSIport | USI port setting |

This sample driver is set forth in mux.c/mux.h and mux_api.h.

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

## 4.18 Built-in Regulator (PWG)

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

Table 4.21 List of built-in regulator (PWG) sample driver functions

| Function Name | Description Name |
|---|---|
| PWG_setOperationMode | Operation mode setting |

This sample driver is set forth in pwg.c/pwg.h and pwg_api.h.

For programs using this sample driver, include pwg_api.h.

## 4.19 Flash Control (FLASHC)

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

Table 4.22 List of flash control (FLASHC) sample driver functions

| Function Name | Description Name |
|---|---|
| FLASHC_setReadWaitCycle | Flash memory read wait count setting |

This sample driver is set forth in flashc.c/flashc.h and flashc_api.h.

For programs using this sample driver, include flashc_api.h.

# Appendix A Multiplier/Divider

Here, the usage method of multiplier/divider is explained.

## A.1 Multiplication and Division Using Multiplier/Divider

In order to perform multiplication and division using multiplier/divider, GNU17 is provided with coprocessor library.

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

## A.2 Sum-of-Products Calculation Using Multiplier/Divider

The program for calculating the sum of products using multiplier/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 | Revision Details (includes old content) and Revision Reason |
|---|---|---|
| 411916600 | All pages | New document |
| | | |
| | | |
| | | |

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: 411916600
First Issue May 2010 in JAPAN Ⓥ