S1C17 Family Application Note

S1C17600 Series Peripheral Circuit Sample Software 2

SEIKO EPSON CORPORATION

Rev.1.0

Table of Contents

1. Overview

This manual describes how to use the sample software for peripheral devices newly added to the S1C17604, S1C17622, and S1C17624 microcontrollers and its operations. Refer to the following manual for information on peripheral circuits common among the S1C17600 series.

- S1C17600 Series Peripheral Circuit Sample Software

The S1C17600 series peripheral circuit sample software 2 is aimed at presenting examples of how to use peripheral circuits newly added to the S1C17604, S1C17622, and S1C17624 microcontrollers.

The S1C17600 series peripheral circuit sample software 2 is provided for each model to simplify the installation; however the basic operations of each function are the same.

Use this manual along with respective model information, technical manuals, and the "S5U1C17001C Manual."

1.1 Operating Environment

In order to run the S1C17600 series peripheral circuit sample software, prepare the following items.

- A board on where S1C17604, S1C17622, or S1C17624 is mounted
- S5U1C17001H (hereinafter ICDmini)
- S5U1C17001C (hereinafter GNU17)

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

2. Explanation of Sample Software

This chapter describes the file configuration and execution method of the S1C17600 series peripheral circuit sample software 2.

The S1C17600 series peripheral circuit sample software 2 consists of "sample software" that checks operations of each peripheral circuit, and "sample drivers" that are sample drivers of each peripheral circuit.

2.1 Included Sample Software

The following lists the included sample software items.

Table 2.1 List of included sample software items

Peripheral Circuit	Sample Software
I/O port (P)	○
Clock generator (CLG_OSC)	●
16-bit timer (T16)	○
8-bit timer (T8F)	○
PWM timer (T16E)	○
8-bit OSC1 timer (T8OSC1)	○
Clock timer (CT)	○
Stopwatch timer (SWT)	○
Watchdog timer (WDT)	○
UART that uses OSC3	○
UART that uses IOSC	○
SPI master	○
SPI slave	○
I2C master (I2CM)	○
I2C slave (I2CS)	○
LCD driver (LCD8)	○
Power supply voltage detection circuit (SVD)	○
RF converter (RFC)	○
A/D converter (ADC10SA)	○
Remote controller sending (REMC)	○
Remote controller receiving (REMC)	○
Sleep/Halt	○
PWM timer (T16A2)	●
Real time clock (RTC)	●
Electric current measurement	●

●: New or modified, ○: Common among the series

2.2 Included Sample Drivers

The following lists the included sample drivers.

Table 2.2 List of included sample drivers

Peripheral Circuit	Sample Driver
I/O port (P)	○
Clock generator (CLG_OSC)	●
16-bit timer (T16)	○
8-bit timer (T8F)	○
PWM timer (T16E)	○
8-bit OSC1 timer (T8OSC1)	○
Clock timer (CT)	○
Stopwatch timer (SWT)	○
Watchdog timer (WDT)	○
UART	○
SPI	○
I2C master (I2CM)	○
I2C slave (I2CS)	○
LCD driver (LCD8)	●
Power supply voltage detection circuit (SVD)	○
RF converter (RFC)	○
A/D converter (ADC10SA)	○
Remote controller (REMC)	○
Prescaller (PSC)	○
MISC	○
Multiplexer (MUX)	●
PWM timer (T16A2)	●
Real time clock (RTC)	●
Power supply control circuit (VD1)	●

●: New or modified, ○: Common among the series

2.3 Directory Structure and File Configuration

The following indicates the directory structure of the S1C17600 series peripheral circuit sample software 2.

```
[s1c176xx]
   │
   ├ GNU17 project files, etc
   ├…
   ├…
   │
   │
   ├─ [mcu]
   │      ├─ s1c17604_peripheral.h
   │      …
   │      ├─ s1c176xx.h
   │      └─ boot.h  boot.c
   │
   ├─ [apl]
   │      ├─ main.h main.c
   │      ├─ clg_osc_main.c
   │      ├─ port_main.c
   │      …
   │      ├─ t16_ main.c
   │      ├─ lcd_ main.c
   │      └─ svd_main.c
   │
   ├─ [driver]
   │      ├─ osc.h    osc.c
   │      ├─ clg.h    clg.c
   │      ├─ psc.h    psc.c
   │      ├─ port.h   port.c
   │      …
   │      ├─ lcd.h    lcd.c
   │      └─ svd.h    svd.c
   │
   └─ [common]
           ├─ osc_api.h
           ├─ clg_api.h
           ├─ psc_api.h
           ├─ port_api.h
           …
           ├─ lcd_api.h
           └─ svd_api.h
```

Figure 2.1 Directory structure of the S1C17600 series peripheral circuit sample software 2

(1) "s1c176xx" directory

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

(2) "mcu" directory

This directory contains files for microcontroller initialization and files which define model-dependent information.

- Header files that define the target model's register addresses, etc (s1c17604_peripheral.h, etc)
- Header file common to the models (s1c176xx.h)
- Initialization file (boot.c)

(3) "apl" directory

This directory contains sample software for each peripheral circuit and header files which define constants and others used in the sample software.

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

(4) "driver" directory

This directory contains sample drivers for each peripheral circuit.

- Header files which define register addresses and bit assignments for each peripheral circuit (xxx.h)
- Program for each peripheral circuit (xxx.c)

(5) "common" directory

This directory contains header files which define prototypes of externally accessible functions provided by sample drivers for each peripheral circuit.

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

Software should include the header file in the "common" directory before calling the sample driver function.

2.4 Execution Method

Execute the S1C17600 series peripheral circuit sample software 2 in the following sequence.

(1) Import the project

Start up GNU17 and import a project of the S1C17600 series peripheral circuit sample software 2.

Refer to "3. Software Development Steps" in the "S5U1C17001C Manual" for how to import a project.

(2) Build the project

Build the S1C176xx project with GNU17.

Refer to "5. GNU17 IDE" in the "S5U1C17001C Manual" for how to build a project.

(3) Connect ICDmini

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

Refer to the "S5U1C17001H User Manual" for how to use ICDmini.

(4) Load and execute program using debugger

Press the [Debug Configurations] button of GNU17 to start the debugger, and press the [Resume] button in the debug view.

The program is loaded to S1C176xx and starts.

Refer to "10. Debugger" in the "S5U1C17001C Manual" for how to use the debugger.

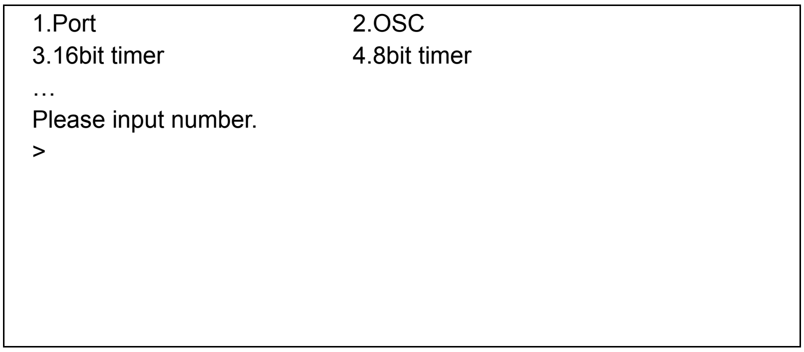
2.5 Sample Software Menu

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

Refer to "10.4.11 [Simulated I/O] view" in the "S5U1C17001C Manual" for information on Simulated I/O.

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

See Chapter 3 for the details of each sample software item.



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

Figure 2.2 Menu screen display example

2.6 Build Method of Specific Modules

The S1C176xx sample software is distributed in the condition where multiple programs will be built.

However, you can build only the sample software for the required peripheral module by modifying the source code of the sample software.

The steps are shown below.

(1) File to be modified

Modify the model-specific definition header.

For the case of the S1C17624 sample software, modify the s1c17624_peripheral.h file.

(2) Locations to be modified

Modify the following locations at the bottom of the file.

```
//#undef PE_PORT
//#undef PE_CLG_OSC
//#undef PE_T16
//#undef PE_T8F
//#undef PE_T16E
//#undef PE_T8OSC1
//#undef PE_CT
//#undef PE_SWT
//#undef PE_WDT
#undef   PE_UART
#undef   PE_UART_OSC3
#undef   PE_UART_IOSC
#undef   PE_SPI
#undef   PE_SPI_MASTER
#undef   PE_SPI_SLAVE
#undef   PE_I2CM
#undef   PE_I2CS
#undef   PE_LCD
#undef   PE_SVD
#undef   PE_RFC
#undef   PE_ADC
#undef   PE_REMC
#undef   PE_REMC_TX
#undef   PE_REMC_RX
#undef   PE_SLEEP_HALT
//#undef PE_T16A2
//#undef PE_RTC
//#undef PE_VD1
#undef   PE_CURRENT_MEASURE
```

Figure 2.3 Definition modification example for a 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 for the peripheral module you are building uses other peripheral modules, you also need to build sample software for these peripheral modules.

For example, the UART (OSC3) sample software uses 8-bit timer; therefore, you need to disable "#undef PE_UART_OSC3" and "#undef PE_T8F" definitions when building the UART (OSC3) sample software.

3. Sample Software Function Details

This chapter describes the function details of the S1C17600 series peripheral circuit sample software 2.

3.1 Clock Generator (CLG_OSC)

3.1.1 Sample software specifications

This sample software performs the following operations using the clock generator (CLG_OSC).

- The software starts and stops oscillation of IOSC.
- The software starts and stops oscillation of OSC1.
- The software starts and stops oscillation of OSC3.
- The software switches the system clock from IOSC to OSC3.
- The software switches the system clock from OSC3 to OSC1.
- The software switches the system clock from OSC1 to IOSC.

3.1.2 Hardware conditions

This sample software operates in the condition where the crystal oscillator or ceramic oscillator is connected to OSC1 and OSC3.

Refer to the section describing "Clock generator (CLG)" in the "S1C176xx Series Technical Manual" for information on how to connect the oscillator.

3.1.3 Operations overview

(1) Overview of sample software operations

- When this sample software initially starts, IOSC is used.
- After displaying "1", "2", "3"..., "9" on SimI/O at a fixed interval, the sample software starts oscillation of OSC3, switches the system clock from IOSC to OSC3, and then stops IOSC.
- Next, after displaying "1", "2", "3"..., "9" on SimI/O at a fixed interval, the sample software starts oscillation of OSC1, switches the system clock from OSC3 to OSC1, and then stops OSC3.
- Then, after displaying "1", "2", "3"..., "9" on SimI/O at a fixed interval, the sample software starts oscillation of IOSC and OSC3, switches the system clock from OSC1 to IOSC, and then stops OSC1 and OSC3.
- Then the sample software displays "1", "2", "3"..., and "9" on SimI/O at a fixed interval.

```
<<< CLG(OSC) demonstration start >>>
IOSC *** 1 ***
IOSC *** 2 ***
…
IOSC *** 9 ***
*** Change from IOSC to OSC3 ***
OSC3 *** 1 ***
OSC3 *** 2 ***
…
OSC3 *** 9 ***
<<< CLG(OSC) demonstration finish >>>
```

Figure 3.1 Display example of the clock generator (CLG_OSC) on the sample software screen

(2) Stopping the sample software

When all the operations described in the above "Overview of sample software operations" are completed, the sample software is terminated and the display returns to the menu screen.

3.2 PWM Timer (T16A2)

3.2.1 Sample software specifications

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

- The software causes PWM timer compare A match interrupts five times in normal mode and acquires counter values of the timer.
- The software causes PWM timer compare B match interrupts five times in normal and half clock modes, and acquires counter values of their timer.
- The software outputs the PWM waveform to the TOUTA5 terminal in normal and half clock modes.
- The software reduces the power consumption while waiting for the interrupt by putting the CPU into HALT mode.

3.2.2 Hardware conditions

This sample software operates in the condition where the crystal oscillator or ceramic oscillator is connected to OSC3.

Refer to the section describing "Clock generator (CLG)" in the "S1C176xx Series Technical Manual" for information on how to connect the oscillator.

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

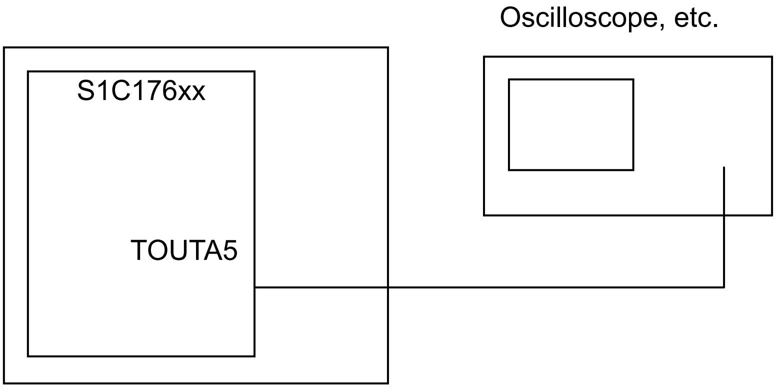
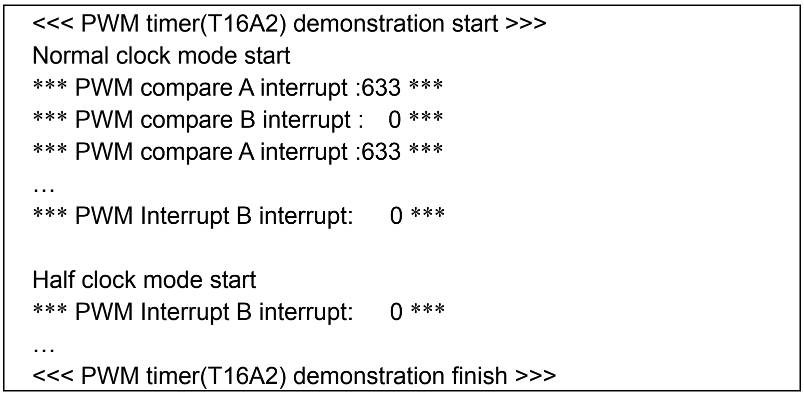


Figure 3.2 Hardware connection diagram for the PWM timer (T16A2) in the sample software

3.2.3 Operations overview

(1) Overview of sample software operations

- The sample software sets to normal clock mode, enables compare A match interrupts and compare B match interrupts, and then starts the PWM timer.
- The sample software acquires the counter value of the up counter when a compare A match interrupt or compare B match interrupt occurs.
- The sample software stops the PWM timer after the 5th compare B match interrupts, acquires the interrupt type and counter value, and displays them on SimI/O.
- Then the sample software sets to half clock mode, disables compare A match interrupts, and starts the PWM timer.
- The sample software stops the PWM timer after the 5th compare B match interrupts, acquires the interrupt type and counter value, and displays them on SimI/O.



```
<<< PWM timer(T16A2) demonstration start >>>
Normal clock mode start
*** PWM compare A interrupt :633 ***
*** PWM compare B interrupt :   0 ***
*** PWM compare A interrupt :633 ***
…
*** PWM Interrupt B interrupt:     0 ***

Half clock mode start
*** PWM Interrupt B interrupt:     0 ***
…
<<< PWM timer(T16A2) demonstration finish >>>
```

Figure 3.3 Display example of the PWM timer (T16A2) on the sample software screen

(2) Stopping the sample software

When all the operations described in the above "Overview of sample software operations" are completed, the sample software is terminated and the display returns to the menu screen.

3.3 Real time Clock (RTC)

3.3.1 Sample software specifications

This sample software performs the following operations using the real time clock.

- The sample software acquires the time from the real time clock.
- The sample software sets the time on the real time clock.
- The sample software displays the number of real time clock interrupts.

3.3.2 Hardware conditions

This sample software operates in the condition where the crystal oscillator or ceramic oscillator is connected to OSC3.

Refer to the section describing "Clock generator (CLG)" in the "S1C176xx Series Technical Manual" for information on how to connect the oscillator.

3.3.3 Operations overview

(1) Overview of sample software operations

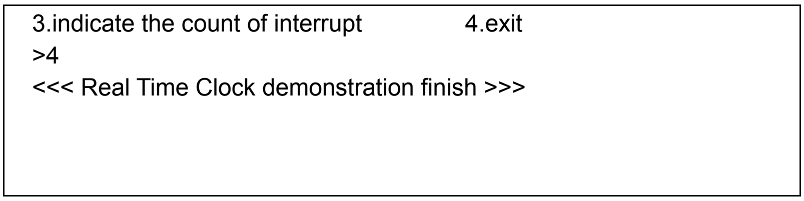
- The sample software displays the RTC sample program menu after the program has started.
- When you enter "1" and press the Enter key from the menu, the sample software obtains the time from RTC and displays in the 24-hour notation.
- When you enter "2" and press the Enter key from the menu, the sample software sets the time to RTC.
- When you enter "3" and press the Enter key from the menu, the sample software displays the number of RTC interrupts.

```
<<< Real Time Clock demonstration start >>>
1.get RTC                              2.set RTC
3.indicate the count of interrupt      4.exit
Please input number.
>1
10/03/01(Mon) 10:00:00

1.get RTC                              2.set RTC
3.indicate the count of interrupt      4.exit
Please input number.
>2
> Input BCD format.
> Year (0x00 - 0x99) :0x**
> Month (0x01 - 0x12) :0x**
> Day (0x01 - 0x31) :0x**
> Hour (0x00 - 0x23) :0x**
> Minute (0x00 - 0x59) :0x**
> Second (0x00 - 0x59) :0x**
> Week (0x0 - 0x6) :0x*

1.get RTC                              2.set RTC
3.indicate the count of interrupt      4.exit
Please input number.
>3
> interrupt count value = xx

1.get RTC                              2.set RTC
```



```
3.indicate the count of interrupt              4.exit
>4
<<< Real Time Clock demonstration finish >>>
```

Figure 3.4 Display example of the real time clock on the sample software screen

(2) Stopping the sample software

Entering "4" and pressing the Enter key from the menu terminates the sample software and the display returns to the menu screen.

3.4 Electric Current Measurement

3.4.1 Sample software specifications

This sample software drives the CPU in the following status in order to measure the electric current.

- The sample software puts the CPU into SLEEP mode.
- The sample software puts the CPU into HALT mode while oscillating only OSC1.
- The sample software puts the CPU into HALT mode while oscillating OSC1 and OSC3.
- The sample software puts the CPU into HALT mode while oscillating OSC1, OSC3, and IOSC.

3.4.2 Hardware conditions

This sample software operates in the condition where the crystal oscillator or ceramic oscillator is connected to OSC3.

Refer to the section describing "Clock generator (CLG)" in the "S1C176xx Series Technical Manual" for information on how to connect the oscillator.

3.4.3 Execution method

Build this sample software as described below to make it executable.

(1) Program to be modified

Modify the model-specific definition header.

For the case of the S1C17624 sample software, modify the s1c17624_peripheral.h file.

(2) Locations to be modified

Modify the following locations at the bottom of the file.

```
//#undef PE_T16A2
//#undef PE_RTC
//#undef PE_VD1
//#undef PE_CURRENT_MEASURE


/*************************************************************************/
/* Current measurement mode selecting                                    */
/*************************************************************************/
// Comment out "#undef      PE_CURRENT_MEASURE", when using current measure program.
#ifdef PE_CURRENT_MEASURE
#define CURRENT_MEASURE_SLEEP                              (0x00)
#define CURRENT_MEASURE_HALT_OSC1                          (0x01)
#define CURRENT_MEASURE_HALT_OSC1_OSC3                     (0x02)
#define CURRENT_MEASURE_HALT_OSC1_OSC3_IOSC   (0x03)
// Remove the comment on the selected measurement mode.
//#define CURRENT_MEASURE_MODE CURRENT_MEASURE_SLEEP
//#define CURRENT_MEASURE_MODE CURRENT_MEASURE_HALT_OSC1
#define CURRENT_MEASURE_MODE CURRENT_MEASURE_HALT_OSC1_OSC3
//#define CURRENT_MEASURE_MODE CURRENT_MEASURE_HALT_OSC1_OSC3_IOSC
#endif
```

Figure 3.5 Modification example for electric current measurement sample software definitions

For example, when putting the CPU into HALT mode while oscillating OSC1 and OSC3, disable the "#undef PE_CURRENT_MEASURE" definition and enable the "#define CURRENT_MEASURE_MODE CURRENT_MEASURE_HALT_OSC1_OSC3" definition before building.

(3) Loading and executing program by debugger

Start the debugger and load the sample software to S1C176xx.

(4) Executing the program

Remove ICDmini from the development board and reset S1C176xx in order to eliminate the influence from the electric current from ICDmini.

(5) Operation overview

- The electric current measurement sample software is executed after the reset.

Carry out step (2) to (4) if you want to measure the electric current in a different condition.

(6) Stopping the sample software

There is no procedure for stopping. Simply stop the power supply to the development board.

4. List of Sample Driver Functions

This chapter lists the sample drivers for the sample software.

4.1 Clock Generator (CLG_OSC)

Table 4.1 shows the list of functions of this sample driver. Refer to source code clg.c and osc.c for details of the function.

Table 4.1 List of functions of the clock generator (CLG_OSC) sample driver

Function Name	Description Name
CLG_OSC_setClockSource	Clock source setting
CLG_OSC_setWaitCycle	Oscillation stability wait time setting
CLG_OSC_controlOscillation	OSC oscillation start/stop setting
CLG_OSC_setNoiseFilter	Noise filter enable/disable setting
CLG_OSC_setLCDClock	LCD clock setting
CLG_OSC_controlLCDClock	LCD clock supply allow/disallow setting
CLG_OSC_setFOUTHDivision	FOUTH clock frequency division setting
CLG_OSC_controlFOUT	FOUT clock output allow/disallow setting
CLG_OSC_setT8OSC1Division	T8OSC1 clock frequency division setting
CLG_OSC_controlT8OSC1	T8OSC1 clock supply allow/disallow setting
CLG_OSC_setSVDClock	SVD clock setting
CLG_OSC_controlSVD	SVD clock supply allow/disallow setting
CLG_OSC_setRFCClock	RFC clock setting
CLG_OSC_controlRFC	RFC clock supply allow/disallow setting
CLG_OSC_setT16A2Clock	T16A2 clock setting
CLG_OSC_controlT16A2	T16A2 clock supply allow/disallow setting
CLG_OSC_controlRTC	RTC clock supply allow/disallow setting
CLG_setPCLKEnable	PCLK supply allow/disallow setting
CLG_setCCLKGearRatio	System clock gear ratio setting

You can find descriptions of this sample driver in clg.c, osc.c, clg.h, osc.h, clg_api.h, and osc_api.h.

Include clg_api.h and osc_api.h in programs which use this sample driver.

4.2 LCD Driver (LCD8)

Table 4.2 shows the list of functions of this sample driver. Refer to source code lcd.c for details of the function.

Table 4.2 List of functions of the LCD driver (LCD8) sample driver

Function Name	Description Name
LCD8_initPower	LCD power supply initialization
LCD8_init	LCD initialization
LCD8_setSEGAssignment	SEG terminal memory assignment setting
LCD8_setCOMAssignment	COM terminal memory assignment setting
LCD8_setDisplayArea	LCD display area setting
LCD8_setDisplayReverse	LCD reverse video display setting
LCD8_controlDisplay	LCD display control
LCD8_setContrast	LCD contrast setting
LCD8_display1Seg	1 segment display
LCD8_initInt	LCD interrupt initialization
LCD8_controlInt	LCD interrupt allow/disallow setting
LCD8_resetIntFlag	LCD interrupt factor flag reset
LCD8_checkIntFlag	LCD interrupt factor flag check

You can find descriptions of this sample driver in lcd.c, lcd.h, and lcd_api.h.

Include lcd_api.h in programs which use this sample driver.

4.3 Power Supply Control Circuit (VD1)

Table 4.3 shows the list of functions of this sample driver. Refer to source code vd1.c for details of the function.

Table 4.3 List of functions of the power supply control circuit (VD1) sample driver

Function Name	Description Name
VD1_setMode	Heavy load mode setting

You can find descriptions of this sample driver in vd1.c, vd1.h, and vd1_api.h.

Include vd1_api.h in programs which use this sample driver.

4.4 PWM Timer (T16A2)

Table 4.4 shows the list of functions of this sample driver. Refer to source code t16a2.c for details of the function.

Table 4.4 List of functions of the PWM timer (T16A2) sample driver

Function Name	Description Name
T16A2_init	PWM timer (T16A2) initialization
T16A2_setTimerMode	PWM timer (T16A2) mode setting
T16A2_setComparatorCapture	Comparator/capture setting
T16A2_getCounterData	Count data acquisition
T16A2_setCompareData	Compare data setting
T16A2_getCaptureData	Capture data acquisition
T16A2_resetTimer	PWM timer (T16A2) reset
T16A2_setTimerRun	PWM timer (T16A2) start/stop setting
T16A2_initInt	PWM timer (T16A2) interrupt initialization
T16A2_controlInt	PWM timer (T16A2) interrupt allow/disallow setting
T16A2_resetIntFlag	PWM timer (T16A2) interrupt factor flag reset
T16A2_checkIntFlag	PWM timer (T16A2) interrupt factor flag check

You can find descriptions of this sample driver in t16a2.c, t16a2.h, and t16a2_api.h.

Include t16a2_api.h in programs which use this sample driver.

4.5 Real time Clock (RTC)

Table 4.5 shows the list of functions of this sample driver. Refer to source code rtc.c for details of the function.

Table 4.5 List of functions of the real time clock (RTC) sample driver

Function Name	Description Name
RTC_init	Real time clock initialization
RTC_setTimerRun	Real time clock start/stop setting
RTC_setTimerMode	24H/12H mode switch setting
RTC_setTime	Real time clock time setting
RTC_setAdj	30-second adjustment
RTC_getTime	Real time clock time acquisition
RTC_checkBusy	Busy check
RTC_initInt	Real time clock interrupt setting
RTC_controlInt	Real time clock interrupt allow/disallow setting
RTC_resetIntFlag	Real time clock interrupt factor flag reset
RTC_checkIntFlag	Real time clock interrupt factor flag check

You can find descriptions of this sample driver in rtc.c, rtc.h, and rtc_api.h.

Include rtc_api.h in programs which use this sample driver.

4.6 Multiplexer (MUX)

Table 4.6 shows the list of functions of this sample driver. Refer to source code mux.c for details of the function.

Table 4.6 List of functions of the multiplexer (MUX) sample driver

Function Name	Description Name
MUX_init	MUX initialization
MUX_setREMCport	REMC port setting
MUX_setADCport	ADC port setting
MUX_setSPIport	SPI port setting
MUX_setUARTport	UART port setting
MUX_setRFCport	RFC port setting
MUX_setI2CMport	I2C master port setting
MUX_setI2CSport	I2C slave port setting
MUX_setOSCport	OSC port setting
MUX_setT16Eport	PWM timer (T16E) port setting
MUX_setLCDport	LCD port setting
MUX_setT8OSC1port	8-bit OSC1 timer port setting
MUX_setDBGport	Debug port setting
MUX_setT16port	16-bit timer port setting
MUX_setT16A2port	PWM timer (T16A2) port setting

You can find descriptions of this sample driver in mux.c, mux.h, and mux_api.h.

Include mux_api.h in programs which use this sample driver.

Revision History

Attachment-1

Rev. No.	Date	Page	Category	Contents
Rev 1.0	2010/xx/xx	All	new	

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
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

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

Document Code: 412017600
First Issue October 2010 in JAPAN Ⓥ