S1C33 Family Application Note

S1C33L27 Software Reference Manual

SEIKO EPSON CORPORATION

Rev.1.0

Evaluation board/kit and Development tool important notice

1. This evaluation board/kit or development tool is designed for use for engineering evaluation, demonstration, or development purposes only. Do not use it for other purpose. It is not intended to meet the requirement of design for finished product.
2. This evaluation board/kit or development tool is intended for use by an electronics engineer, and it is not the product for consumer. The user should use this goods properly and safely. Seiko Epson dose not assume any responsibility and liability of any kind of damage and/or fire coursed by usage of it. User should cease to use it when any abnormal issue occurs even during proper and safe use.
3. The part used for this evaluation board/kit or development tool is changed without any notice.

NOTICE

Table of Contents

1. Preface

1.1 Purpose

This manual is the reference manual of S1C33L27 software package. Users can refer to this manual and the source code of this software package to develop software for real applications with S1C33L27.

1.2 Related Information

- For S1C33L27 specification, refer to the "S1C33L27 Technical Manual."
- For PRT33L27 board specification, refer to the "PRT33L27 Hardware Specification."
- For details on the compiler specifications and how to use GNU33 IDE, refer to the "S5U1C33001C Manual."

2. Overview

This software package is a dedicated board support package (BSP) for PRT33L27. It contains the chip specific code (register definition header files and drivers for S1C33L27 hardware modules) and the board specific code (drivers for on-board devices), as well as typical sample programs utilizing the above codes to realize certain functions for users' reference.

The following table describes the overall software organization:

Folder				Content
C33L27sample_r100				root folder of S1C33L27 software package
	include¥			register definition header files (for all modules)
	driver¥			device drivers for corresponding hardware modules and on-board devices
	sample¥			individual sample programs to show the usage of certain hardware module(s), with each sub-folder containing one sample program
		xxx¥		one specific sample
			include¥	needed header files (copied from the upper "include¥" folder)
			driver¥	needed drivers (copied from the upper "driver¥" folder)
			system¥	sample implementation, including: * boot.c (boot & system initialization) * interrupt.c (interrupt handlers) * vector.c (vector table) * main.c (main loop) * ... (other needed system level codes)
			res¥ (optional)	resource files (picture, voice data, etc.) used in this demo
			xxx¥ (optional)	specific function or middleware used in this demo, such as "usb"
		......		other samples

Programs with full source code provided in other folders are described in subsequent chapters.

3. Register Definition Header File

3.1 File Configuration

The following table lists the register definition header files of all S1C33L27 hardware modules provided in the "prt33l27_sw¥include" folder of the software package.

File Name	Content
33l27_reg.h	**S1C33L27 register definition (includes all xxx_reg.h)**
cmu_reg.h	CMU module register definition
psc_reg.h	PSC module register definition
rtc_reg.h	RTC module register definition
sramc_reg.h	SRAMC module registers
sdramc_reg.h	SDRAMC module registers
hif_master_reg.h	HIF module register definition (as master)
hif_slave_reg.h	HIF module register definition (as slave)
ccu_reg.h	CCU module register definition
itc_reg.h	ITC module register definition
dmac_reg.h	DMAC module register definition
t8_reg.h	T8 module register definition
t16a_reg.h	T16A module register definition
t16p_reg.h	T16P module register definition
wdt_reg.h	WDT module register definition
uart_reg.h	UART module register definition
usi_reg.h	USI module register definition
usil_reg.h	USIL module register definition
i2s_reg.h	I2S module register definition
remc_reg.h	REMC module register definition
card_reg.h	CARD module register definition
sd_mmc_reg.h	SD_MMC module register definition
gpio_reg.h	GPIO module register definition
adc_reg.h	ADC module register definition
lcdc_reg.h	LCDC module register definition
misc_reg.h	MISC register definition

3.2 Application Notes

1. Each register definition file defines the macros to access registers of the corresponding hardware module. Using register definition files is convenient for driver and application program development as they provide flexible ways to access registers and function bits of each register.

   Taking the register definition file for CMU module (cmu_reg.h) as an example, the union is defined as follows for the "Oscillation Control Register (CMU_OSCCTL)":

```
union CMU_CLKCNTL_tag {
    volatile struct {
            unsigned int SOSC3:1;       //Low-speed OSC3 On/Off
            unsigned int SOSC1:1;       //High-speed OSC1 On/Off
            unsigned int DUMMY:2;
            unsigned int OSCTM:4;       //OSC3 stabilization wait time
    } bCTL;
    volatile unsigned char ucCTL;
};
```

   The macros to access this register and its function bits are defined as follows:

```
#define CMU_BASE            0x00300100
......
#define CMU_OSCCTL          (*(union CMU_CLKCNTL_tag *)(CMU_BASE+1)).ucCTL
#define CMU_OSC3EN          (*(union CMU_CLKCNTL_tag *)(CMU_BASE+1)).bCTL.SOSC3
#define CMU_OSC1EN          (*(union CMU_CLKCNTL_tag *)(CMU_BASE+1)).bCTL.SOSC1
#define CMU_OSC3WT          (*(union CMU_CLKCNTL_tag *)(CMU_BASE+1)).bCTL.OSCTM
```

   In other words, users can read/write the value of the register using the CMU_OSCCTL macro, or read/write the value of "OSC3 wait cycle" bits using the CMU_OSC3WT macro, depending on the needs.

2. The macros in the register definition header files are basically named by following the rules below:
   - The name of the macro to access the register itself is the same as the "register name" defined in the S1C33L27 Technical Manual.
   - The name of the macro to access the function bits is defined as the [module name prefix] ("CMU_" in above example) + the "bit name" defined in the S1C33L27 Technical manual.

3. In the drivers and sample programs, the macros defined in the register definition header files are used to read/write registers. Users can use these register definition header files directly, or refer to the sample driver to create their own drivers according to the actual needs (for instance, add wrapper APIs to read/write registers in the driver layer).

4. "33l27_reg.h" is the overall register definition header file, which includes all needed register definition header files for the hardware modules. Therefore, it is enough to just include this file in the driver and application layer program.

4. Driver

4.1 File Configuration

The following table lists the sample drivers for S1C33L27 H/W modules and PRT33L27 on-board devices (NAND Flash, SPI Flash, LCD, audio codec, key, latch, etc.) provided in the "prt33l27_sw¥driver" folder of the software package.

File Name	Content
driver.h	**overall driver header (includes all xxx_drv.h)**
cmu_drv.h & .c	CMU driver
rtc_drv.h & .c	RTC driver
sramc_drv.h & .c	SRAMC driver
sdramc_drv.h & .c	SDRAMC driver
hif_master_drv.h & .c	HIF driver (as master)
hif_slave_drv.h & .c	HIF driver (as slave)
ccu_drv.h & .c	CCU driver
dmac_drv.h & .c	DMAC driver
t16a_drv.h & .c	T16A driver
t16p_drv.h & .c	T16P driver
uart_drv.h & .c	UART driver
spi_drv.h & .c	SPI driver (using USI module)
i2s_drv.h & .c	I2S driver
nand_drv.h & .c	NAND driver
sd_mmc_drv.h & .c	SD/MMC driver
spi_flash_drv.h & .c	SPI Flash driver
adc_drv.h & .c	ADC driver
lcd_drv.h & .c	LCD driver
codec_drv.h & .c	audio codec driver
key_drv.h & .c	key driver
latch_drv.h & .c	latch (373) driver (used by LCD and SPI drivers)
gpc_drv.h & .c	GPC (Graphic library) driver (for graphic drawing and text output)
calc.h	CALC API table
eecc.h	Enhanced ECC API table

4.2 Application Notes

1. All drivers are created according to the usage of the sample program or demonstration program implemented on PRT33L27, and may not be suitable for other applications.

2. "driver.h" is the overall driver header file which includes all needed driver header files. Therefore, it is enough to just include this file in the application layer program.

3. Users can check the driver program source codes to understand the implementation in detail, as well as refer to them to develop their own drivers according to the actual usage.

5. Sample Program

The following sample programs are provided to show how to use certain H/W modules and the corresponding drivers to realize specific functions, as typical sample codes for users' reference.

- RTC
- T16A
- T16P
- UART
- USI(SPI)
- I2S
- SD
- ADC
- LCDC
- Key
- USB
- CARD
- HIF

Each sample program is built as an individual GNU33 project, so that users can open the project and build and execute it using the GNU33 IDE.

All sample programs (except HIF sample program) are executed in the GNU33 Debugger (gdb) since they are built to run in the internal RAM or the on-board SDRAM. Some of them use the [Simulated I/O] window of the GNU33 Debugger (gdb) to show results or indication messages even without connecting the LCD sub-board.

The following chapters describe the function and operation of the sample programs, with user operation marked in **bold**. Users can check the source code to understand the implementation in detail.

5.1 RTC

This is a sample program that mainly uses the RTC module and its driver. It shows how to initialize the RTC, set time information and provide interrupt handler to make the RTC count.

Switch Setting & Sub-Board Connection:

1) Set the function switch to select RTC (set W5 to "left") before executing this sample program.
2) Make sure the LCD sub-board is connected before executing this sample program.

Process:

1) Initializes the RTC with a certain start time (2010-01-01, 08:00:00).
2) Displays the time information (year, month, date, hour, minute, second) on the LCD panel and refreshes the panel every 1 second, just like an electronic clock.

5.2 T16A

This is a sample program that mainly uses the T16A module and its driver.

Process:

1) Turns on/off the on-board LED every 1 second (by 1-second interrupt).

5.3 T16P

This is a sample program that mainly uses the T16P module and its driver. It shows how to replay PCM sound using the PWM timer.

Switch Setting & Sub-Board Connection:

1) Make sure the PWM audio sub-board is connected before executing this sample program.

Process:

1) Replays the built-in PCM sound data "Welcome to use C33! This is a demo program to replay PCM sound." once.

5.4 UART

This is a sample program that mainly uses the UART module and its driver. It shows how to receive data from a PC and transfer data back to the PC through RS-232. The UART console must be running on the PC for command input and result output.

Switch Setting & Sub-Board Connection:

1) Set the function switch to select RS-232 (set W1 to "left") before executing this sample program.

Process:

1) [User operation] **Connect the PRT board with the PC through RS-232.**
2) [User operation] **Launch the UART console on the PC and configure (1-bit start, 8-bit data, 1-bit stop, 115200bps).**
3) [User operation] **After the above operation (PC side is ready), run this sample program in the GNU33 Debugger (gdb).**
4) Shows the menu below (lists the available commands) in the UART console.

```
<< COMMAND LIST>>
draw heart
draw earth
exit
```

5) [User operation] **Input a command in the UART console on the PC to execute and check the results.**
   - A "heart" figure (consisting of many characters) will be displayed in the UART console if the "draw heart" command is input.
   - An "earth" figure (consisting of many characters) will be displayed in the UART console if the "draw earth" command is input.
   - The "****** Sample End *******" message will be displayed in the UART console and this sample program exited if the "exit" command is input.

Note:

1) Always follow the operation sequence and make ready the PC side first, then run this sample program in GDB33.

5.5 USI (SPI)

This is a sample program that mainly uses the USI module (SPI mode) and its driver. It shows how to control the on-board SPI Flash. The operation results will be displayed in the [Simulated I/O] window of GDB33.

Switch Setting & Sub-Board Connection:

1) Set the function switch to select SPI Flash (set W1 to "middle" & W2 to "middle") before executing this sample program.

Process:

1) Sequentially performs erase, read, and write operation on the on-board M45PE80 SPI Flash and displays the operation results accordingly.

5.6 I2S

This is a sample program that mainly uses the I2S module and its driver. It shows how to control the audio codec (on the I2S audio sub-board) via I2S to record and replay sound (as PCM data). The indication message (that indicates the user to speak, etc.) and processing information will be displayed in the [Simulated I/O] window of GDB33.

Switch Setting & Sub-Board Connection:

1) Set the function switch to select I2S (set W6 to "left") before executing this sample program.
2) Make sure the I2S audio sub-board is connected before executing this sample program.

Process:

1) Replays the built-in PCM sound data "Welcome to use C33! This is a demo program to replay PCM sound."
2) When replay of the sound is finished, shows the indication message to record a sound.
3) [User operation] **Press the [Continue] button in GDB33 to start recording a sound for about 7 seconds.**
4) When recording of the sound is finished, shows the indication message to replay the recorded sound.
5) [User operation] **Press the [Continue] button in GDB33 to replay the recorded sound.**

5.7 SD

This is a sample program that mainly uses the SD_MMC module and its driver. It shows how to control the SD/MMC card. The operation results will be displayed in the [Simulated I/O] window of GDB33.

Process:

1) [User operation] **Insert a SD/MMC card into the on-board slot.**
2) Sequentially performs card identification, erase, read, and write operations, and displays the operation results accordingly.

Note:

1) Remove the card from the slot after the "/****** Sample End ******/" message is shown.

5.8 ADC

This is a sample program that mainly uses the ADC module and its driver. It shows how to perform analog to digital conversion with the ADC. The AD conversion result will be displayed in the [Simulated I/O] window of GDB33.

Process:

- Converts 20 analog samples and displays the converted values.

5.9 LCDC

This is a sample program that mainly uses the LCDC module and its driver to support the LCD panel on the PRT board. It shows how to initialize LCDC and display information on the LCD panel.

Switch Setting & Sub-Board Connection:

1) Make sure the LCD sub-board is connected before executing this sample program.

Process:

1) Displays color bars on the LCD panel.

5.10 Key

This is a sample program that mainly uses the GPIO module and the key driver to support the key matrix in the PRT board. It shows how to perform key scanning and judge the input key.

Switch Setting & Sub-Board Connection:

1) Make sure the LCD sub-board is connected before executing this sample program.

Process:

1) Waits for a key input.
2) When a key on the PRT board is pressed, displays the key name (Left, Right, Up, Down, Enter, ESC) of the pressed key on the LCD panel.

5.11 USB

This is a sample program that mainly uses the USB module. It shows how to realize the USB disk function.

Process:

1) [User operation] **Connect the PRT board with a PC using a USB cable.**
2) The PC will recognize the on-board SDRAM as a mass storage USB device.
3) [User operation] **From the PC (Explorer), copy/open/delete a file to/from the removable disk (on-board SDRAM).**
4) [User operation] **Disconnect the PC.**

Note:

1) There is no problem to first remove the disk from the PC, and then disconnect the USB cable. After that, you can connect the USB cable again to resume the USB disk operation.

5.12 CARD

This is a sample program that mainly uses the CARD module and the NAND driver to support the on-board NAND Flash. It shows how to control the on-board NAND Flash. The operation results will be displayed in the [Simulated I/O] window of GDB33.

Switch Setting & Sub-Board Connection:

1) Set the function switch to select NAND Flash (set W4 to "right" and W5 to "left") before executing this sample program.

Process:

1) Sequentially performs erase, read, and write operation on the on-board NAND Flash (K9G8G08U) and displays the operation results accordingly.

5.13 HIF

This is a sample program that mainly uses the HIF module of 33L27 and its driver. This sample program needs 2 sets of PRT33L27 boards, one to build and download the program to each of the on-board NOR Flash of the master & slave boards, and the other to connect them to run. It shows how to control the LCD display on the slave board from master side via HIF.

Switch Setting & Sub-Board Connection:

1) Prepare 2 sets of PRT33L27 boards, one as the master board, the other as the slave board. Set the function switch on the master board to select Ext-Bus (set W3 to "left" and W4 to "left"), and set the function switch on the slave board to select Host I/F (set all W1 to W6 to "right") before executing this sample program.
2) Make sure the LCD sub-board is connected with the slave PRT33L27 board before executing this sample program.

Process:

1) [User operation] **Power on both PRT33L27 boards, then connect them via the HIF interface.**
2) [User operation] **Press [Reset] switch on the master board to reset it.**
3) Sends a picture data from the master board to the slave board.
4) Displays the picture on the LCD panel of the slave board.

Notes:

1) This sample program should be re-built for the master & slave PRT33L27 boards with the corresponding compiler options first.
   - Re-build the program for the master board (by setting "MASTER" macro in the compiler option) and download the program to the on-board NOR Flash of the master board by executing "hif_gnu33IDE.cmd".
   - Re-build the program for the slave board (by setting "SLAVE" macro in the compiler option) and download the program to the on-board NOR Flash of the slave board by executing "hif_gnu33IDE.cmd".
2) After the program building and downloading are completed, you can execute this sample program as described in the above "Process" part.

Revision History

Attachment-1

Rev. No.	Date	Page	Category	Contents
Rev 1.0	2012/08/03	All	New	New creation

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: 412385100
First Issue August 2012 in JAPAN Ⓥ