

S1D13C00 Memory Display Controller

Peripheral Circuit Sample Software Manual

Document Number: XB8A-B-001-01.12

NOTICE

No part of this material may be reproduced or duplicated in any form or by any means without the written permission of Seiko Epson. Seiko Epson reserves the right to make changes to this material without notice. Seiko Epson does not assume any liability of any kind arising out of any inaccuracies contained in this material or due to its application or use in any product or circuit and, further, there is no representation that this material is applicable to products requiring high level reliability, such as, medical products. Moreover, no license to any intellectual property rights is granted by implication or otherwise, and there is no representation or warranty that anything made in accordance with this material will be free from any patent or copyright infringement of a third party. When exporting the products or technology described in this material, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You are requested not to use, to resell, to export and/or to otherwise dispose of the products (and any technical information furnished, if any) for the development and/or manufacture of weapon of mass destruction or for other military purposes.

All brands or product names mentioned herein are trademarks and/or registered trademarks of their respective companies.

©SEIKO EPSON CORPORATION 2018-2019, All rights reserved.

Table of Contents

1	Overview	3
2	Setup and Details of Sample Software Package	4
2.1	Hardware Preparation for Target Download and Execution	4
2.1.1	Hardware Connections for TI EK-TM4C1294XL Host MCU Board	4
2.1.2	Hardware Connections for ST STM32 Nucleo-144 development board	5
2.2	Code Composer Studio and TivaWare Installation	6
2.2.1	Code Composer Studio	6
2.2.2	TivaWare	6
2.2.3	S1D13C00 Sample Software Installation	9
2.2.4	Software Build and Target Download	10
2.2.5	DEBUG_PRINT Macro Symbol	10
2.3	System Workbench for STM32 and STM32CubeF7 Installation	11
2.3.1	System Workbench for STM32	11
2.3.2	STM32CubeF7 Software Package	11
2.3.3	S1D13C00 Sample Software Installation	11
2.3.4	Software Build and Target Download	13
2.3.5	DEBUG_PRINT Macro Symbol	13
2.4	Sample Software Components	14
2.4.1	Host Hardware-Dependent Layer (HHDL)	14
2.4.2	Host Interface Configuration	16
2.4.3	Peripherals Library (sePeriphLibrary)	17
2.4.4	Graphics Library (seGraphicsLibrary)	17
2.4.5	External Peripherals Library (seSerflashLib)	17
2.4.6	Example Applications/Projects	17
2.5	MDC Tools	17
2.5.1	Font Conversion MDCFontConv.exe	18
2.5.2	Image Conversion MDCImgConv.exe	19
2.5.3	Binary File for Serial Flash MDCSerFlashImg.exe	20
3	Details of Sample Software	21
3.1	Clock Generator (CLG)	21
3.2	DMA Controller (DMAC)	22
3.3	I2C	23
3.4	MDC Example for LPM012M134B Panel (MDC_LPM012M134B)	24
3.5	I/O Ports (PORT)	25
3.6	Quad Synchronous Serial Interface Master (QSPI_MASTER)	26
3.7	Quad Synchronous Serial Interface Slave (QSPI_SLAVE)	27
3.8	IR Remote Controller (REMC)	28
3.9	Real Time Clock (RTC)	29
3.10	Serial Flash Download (SERFLASH_DOWNLOAD)	30
3.11	Sound Generator (SND)	31
3.12	Synchronous Serial Interface Master (SPI_MASTER)	32
3.13	Synchronous Serial Interface Slave (SPI_SLAVE)	33

3.14 16-Bit Timer (T16)	34
4 Revision History.....	35
5 Sales and Technical Support	36

1 Overview

This manual describes how to use the example software and libraries provided for the S1D13C00 memory display controller (MDC) and shows the expected output when running the example software.

The example software is included in the S1D13C00 Peripheral Circuit Sample Software package. It is intended to demonstrate how to use the various peripheral circuits in the S1D13C00 memory display controller. Each Example demonstrates features of the selected peripheral and exercises various peripheral modes.

The **S1D13C00 Peripheral Circuit Sample Software** package includes:

- Peripheral Library
- Graphics Library
- External Peripheral Library
- Example Software

The Peripheral Library provides easy to use methods which perform complex peripheral functions. Those methods include functions such as peripheral initialization and peripheral feature management.

The Graphics Library provides routines for drawing and rendering objects to the display buffer. It includes hardware drawing for lines/rectangles/ellipses, software drawing of arcs, and rendering scaled/rotated text strings using a bitmap font set.

The External Peripheral Library provides routines to interface to the onboard serial flash device on the S5U13C00P00CX00 development board. It includes routines to erase and program the serial flash as well as routines for downloading a binary file to the serial flash through the UART interface of the host microcontroller using the XMODEM transfer protocol.

Before running the S1D13C00 sample software, prepare the following hardware components:

- Host Microcontroller (MCU) board (one of the following):
 - TI EK-TM4C1294XL Launchpad Board
 - ST Nucleo-F746ZG Board (STM32 Nucleo-144 development board with STM32F756ZG)
 - ST Nucleo-F767ZI Board (STM32 Nucleo-144 development board with STM32F767ZI)
- S5U13C00P00CX00 Customer Development Board (direct connect for TI EK-TM4C1294XL)
- S5U13C00M00C100 Adapter Board (if using a STM32 Nucleo-144 development board)

For detailed information on the S1D13C00 Microcontroller, refer to the S1D13C00 Hardware Functional Specification, document number XB8A-A-001-xx. For detailed information on the S5U13C00P00CX00 Customer Development Board, refer to the S5U13C00P00CX00 Customer Development Board User Manual, document number XB8A-G-001-xx. For detailed information on the S5U13C00M00C100 Adapter Board, refer to the S5U13C00M00C100 Adapter Board User Manual, document number XB8A-G-002-xx.

2 Setup and Details of Sample Software Package

As mentioned in the previous section, the S1D13C00 Peripheral Circuit Sample Software package can be built and run on the host MCU board; TI EK-TM4C1294XL Launchpad or a ST STM32 Nucleo-144 development board (Nucleo-F746ZG or Nucleo-F767ZI). For the EK-TM4C1294XL host MCU board, TI's Code Composer Studio IDE and TivaWare software package are needed. For a STM32 Nucleo-144 development board, System Workbench for STM32 (SW4STM32) IDE and the STM32CubeF7 software package are needed.

2.1 Hardware Preparation for Target Download and Execution

The S5U13C00P00CX00 Customer Development Board is designed to connect directly to the TI EK-TM4C1294XL Launchpad host MCU board. For the STM32 Nucleo-144 development boards, the S5U13C00M00C100 Adapter Board to connect the S5U13C00P00CX00 board to the STM32 Nucleo-144 development board.

2.1.1 Hardware Connections for TI EK-TM4C1294XL Host MCU Board

The following figure shows the S5U13C00P00CX00 evaluation board connected to a TI EK-TM4C1294XL Launchpad board.

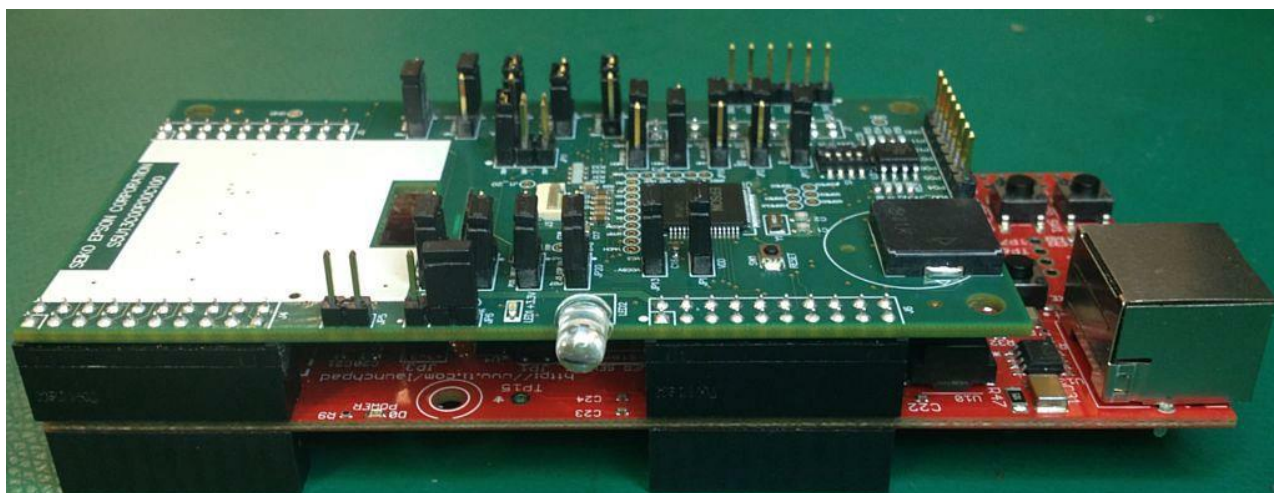


Figure 2.1 Physical Connection of S5U13C00P00CX00 Board to EK-TM4C1294XL

For specific evaluation board configuration refer to the S5U13C00P00CX00 Customer Development Board User Manual, document number XB8A-G-001-xx.

The EK-TM4C1294XL host MCU board is connected and powered through a Micro-USB cable to a PC. Before connecting the hardware, Code Composer Studio should be installed on the PC (see Section 2.2.1) which will install the USB drivers (Virtual COM port and SWD debugger interface) needed for the board.

There are two Micro-USB connectors on the EK-TM4C1294XL board. One is beside the Ethernet (network) connector on one end of the board and the other is on the opposite side of the board. The USB cable should be connected to the one opposite to the Ethernet connector.

2.1.2 Hardware Connections for ST STM32 Nucleo-144 development board

The following figure shows the S5U13C00P00CX00 evaluation board connected to a STM32 Nucleo-144 development board.

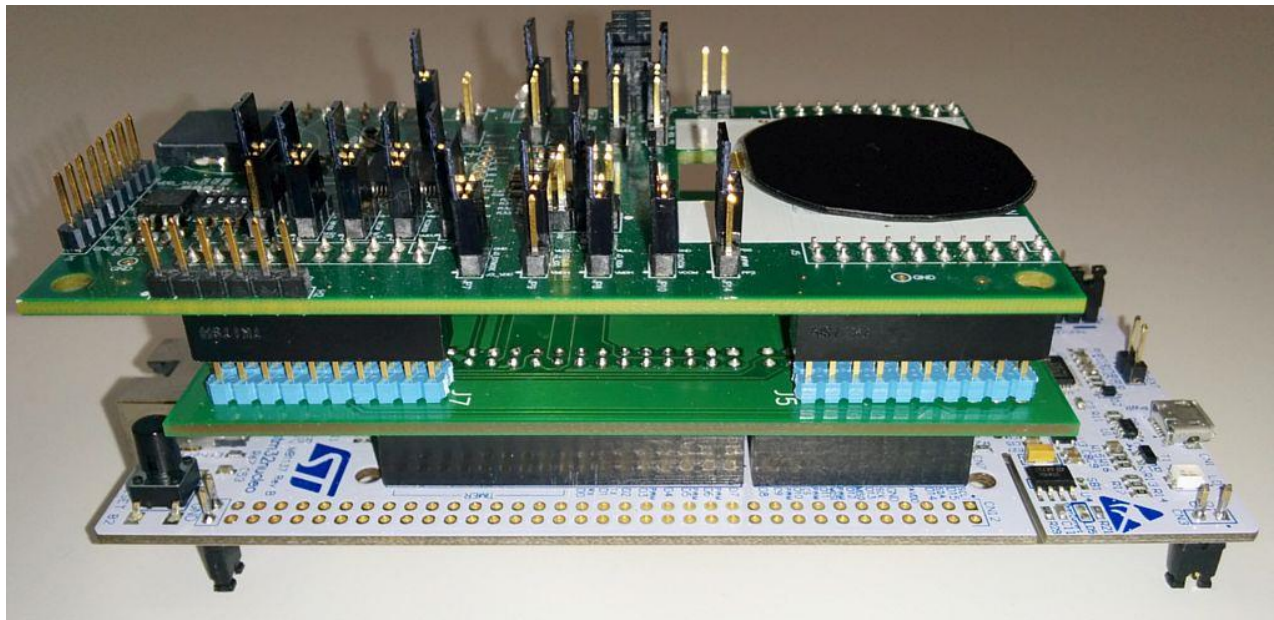


Figure 2.2 Physical Connection of S5U13C00P00CX00 Board to a STM32 Nucleo-144 development board

The top board is the S5U13C00P00CX00 Customer Development Board, the board in the middle is the S5U13C00M00C100 Adapter Board for the STM32 Nucleo-144 development board, and the board at the bottom is the STM32 Nucleo-144 development board.

The STM32 Nucleo-144 development board (either Nucleo-F746ZG or Nucleo-F767ZI) is connected and powered through a Micro-USB cable to a PC. Before connecting the hardware, System Workbench for STM32 (SW4STM32) on the PC (see Section 2.3.1) which will install the USB drivers (ST-Link Virtual COM port and debugger interface) needed for the board.

There are two Micro-USB connectors on the STM32 Nucleo-144 development board. One is beside the Ethernet (network) connector on one end of the board and the other is on the opposite side of the board. The USB cable should be connected to the one opposite to the Ethernet connector.

2. Setup and Details of Sample Software Package

2.2 Code Composer Studio and TivaWare Installation

Code Composer Studio, or CCS, is the selected development environment for use with the Texas Instruments EK-TM4C1294XL Launchpad board. This section describes how to install and build projects using CCS.

2.2.1 Code Composer Studio

CCS is a free development environment offered by Texas Instrument for use with their evaluation board products. Downloads and information about CCS can be found here: <http://www.ti.com/tool/CCSTUDIO>

The S1D13C00 software release has been built and tested with Code Composer Studio version 7.4.00015 and built (but not tested) with version 8.0.0.00016. It has not been built or tested with other versions of CCS.

Download the CCS package and install according to the CCS instructions. For the most compatible installation, it is recommended to use the default install directory of c:\ti.

2.2.2 TivaWare

The TivaWare package is a royalty free collection of library routines to control the TM4C1294 features and peripherals and is required by the S1D13C00 sample software. It is recommended to download and install the latest TivaWare C package. When this document was written the current TivaWare C series package was 2.1.4.178.

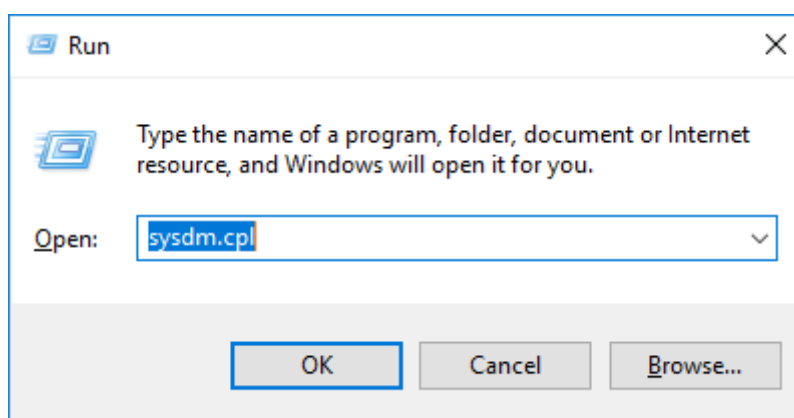
Information and downloads for the TivaWare for C series can be found here: <http://www.ti.com/tool/SW-TM4C>

IMPORTANT: Different versions of Code Composer Studio manage the TivaWare libraries differently and incompatibly. In an effort to keep setup and use of this package as simple as possible across different CCS version, an alternate method was developed. This method requires creating a system environment variable that points to the TivaWare install directory.

The procedure to create the environment variable under Windows 10 is as follows;

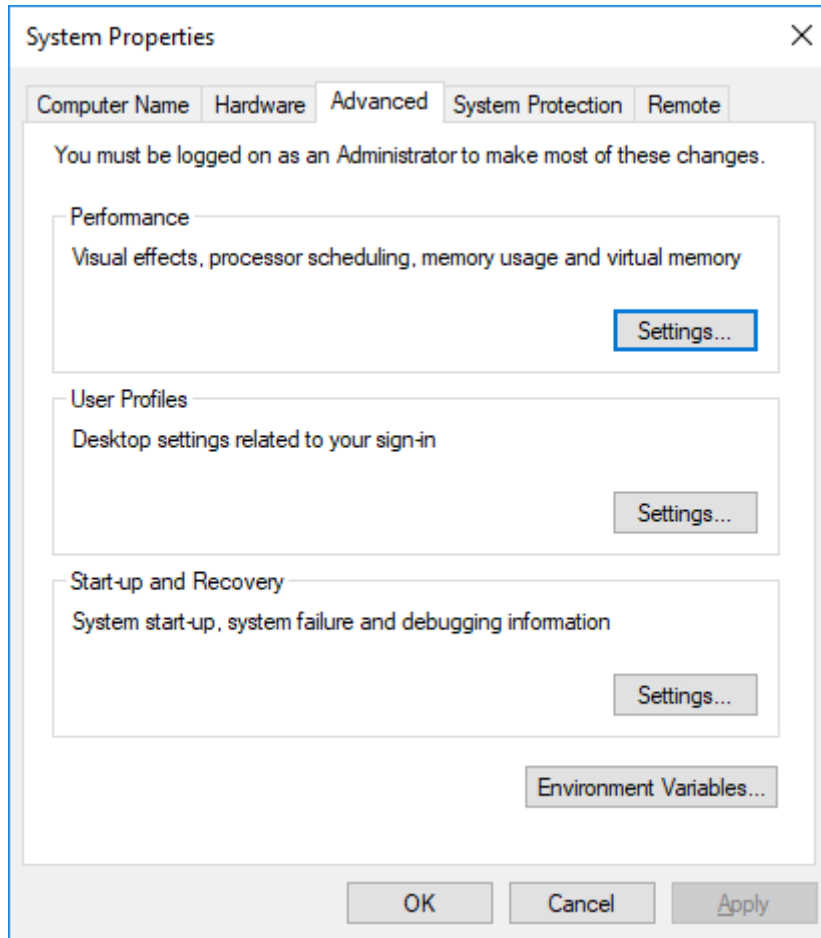
Press **Win**+**R** at the same time to get run dialog prompt.

In the run dialog enter sysdm.cpl to start the system control dialog.



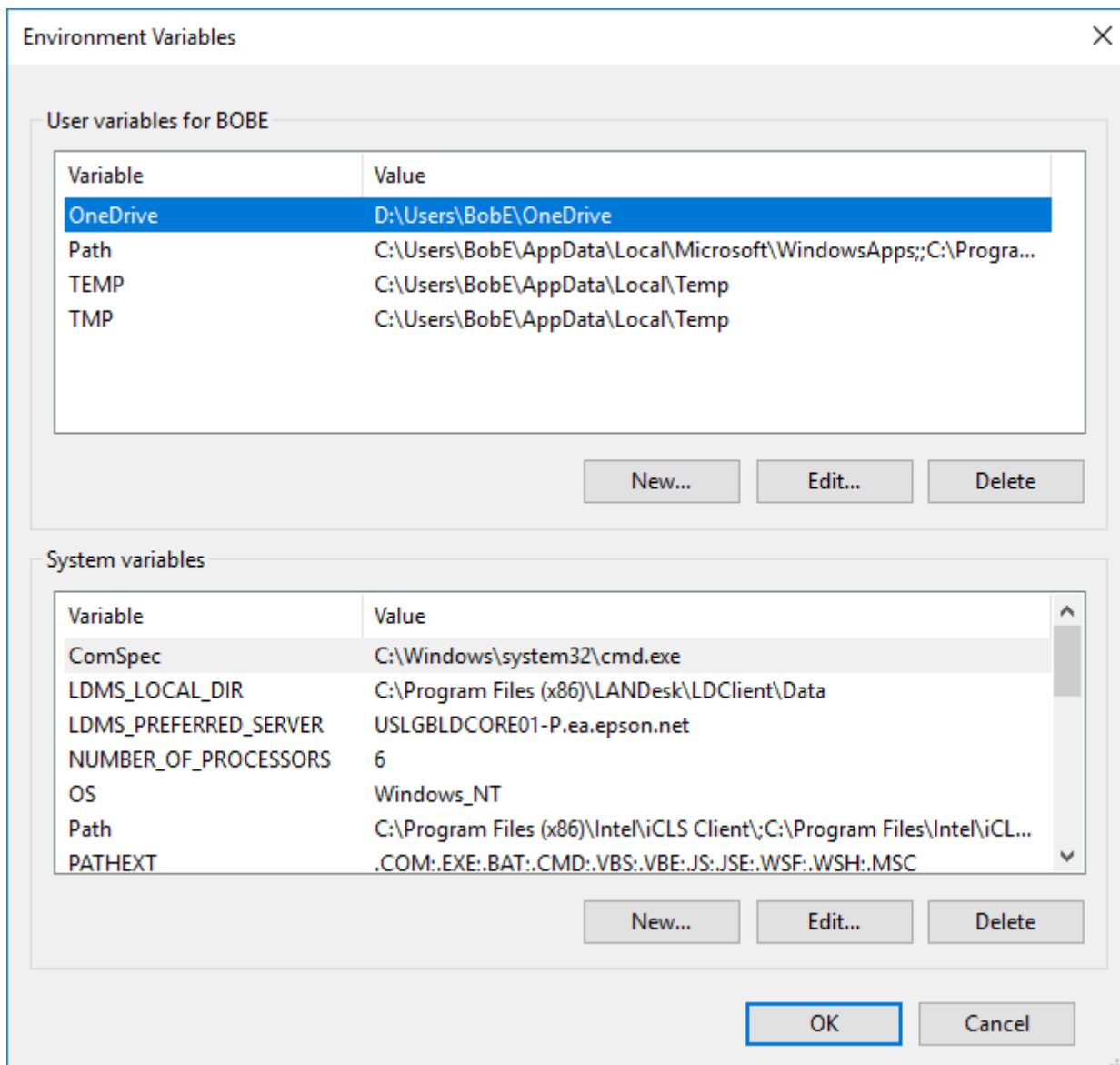
2. Setup and Details of Sample Software Package

Select the Advanced tab and then click the “Environmental Variables...” button.



2. Setup and Details of Sample Software Package

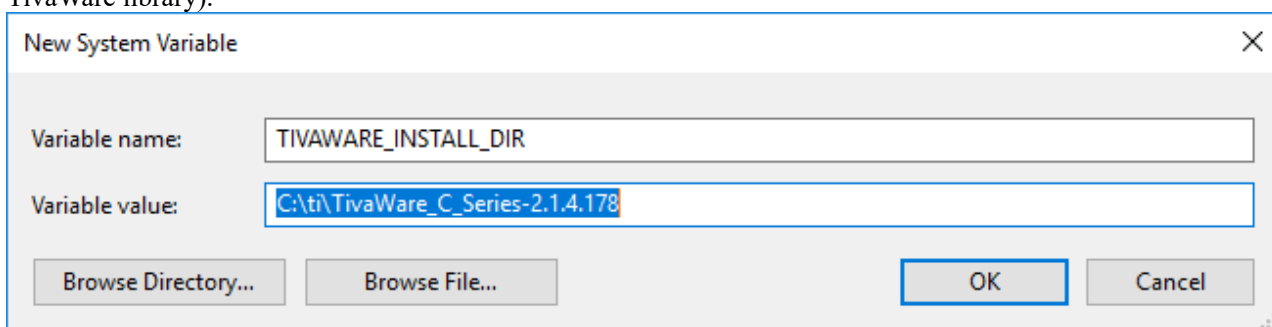
Click the “New...” button in the System variables section to add a new environment variable.



Add the following variable.

Variable name: TIVAWARE_INSTALL_DIR

Variable value: (use the browse directory button to locate and select the directory where you installed the TivaWare library).



2.2.3 S1D13C00 Sample Software Installation

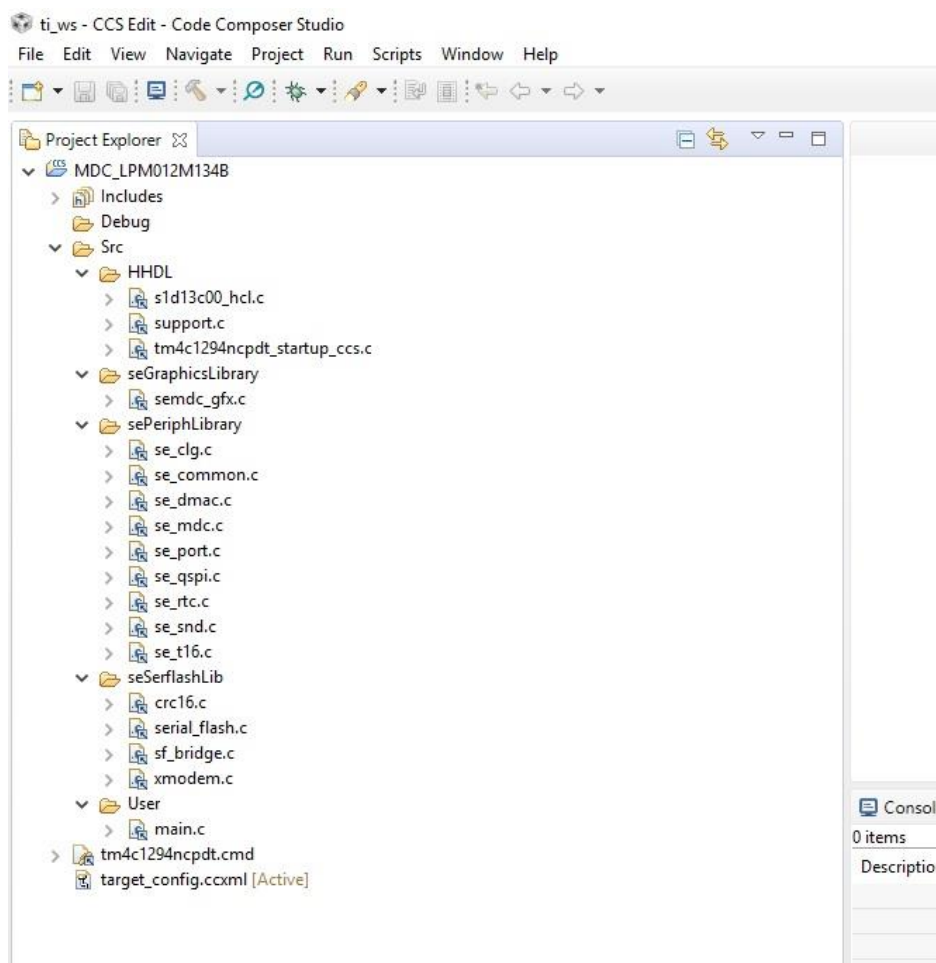
The S1D13C00 Sample Software and Peripheral Drivers software package is available for download from [Epson's website](#). Please ensure that you have the latest version before installing.

For distribution the installation package is encapsulated in an executable ZIP file, S1D13C00_SW.exe. Double click the executable file to begin installation. The default destination location for the sample software is c:\Epson, but it can be changed to any folder location.

Once setup has completed successfully, run the CCS IDE. When the Eclipse Launcher prompts for a workspace directory, browse to an existing workspace or enter a new workspace directory location, then click OK to open the workspace.

The example projects are in the “S1D13C00_SW\Examples” subfolder of the folder where the sample software package was installed. To open one of the example projects, click the menu item “File->Open Projects from File System”. In the “Import Projects from File System or Archive” window, click the “Directory” button to browse to the “S1D13C00_SW\Examples\<projectname>\HOSTS\EK-TM4C1294XL\ccs” folder, where <projectname> is one of the example projects. For example, an example project for displaying two types of watch faces on the LPM012M134B panel is “MDC_LPM012M134B”.


After selecting the project folder, click “Finish” in the “Import Projects from File System or Archive” window to open the project. The project name, for example “MDC_LPM012M134B”, will be displayed in the Project Explorer window of CCS. Expand the project by clicking on the ‘>’ arrow beside it. Sub-items can also be expanded by clicking on the ‘>’ arrow beside the item. For example, after expanding the “Src” sub-item and its sub-items, the following screen will be displayed:



2. Setup and Details of Sample Software Package

2.2.4 Software Build and Target Download

To rebuild all the projects, select Project ->Build All or press the key combination <CTRL>


After a successful build, click a project to select it for debugging. Click the debug symbol  to download, run, and debug the firmware on the target board.

For actual debugging procedures, please refer to the CCS documentation.

2.2.5 DEBUG_PRINT Macro Symbol

The example projects provided have a macro symbol “DEBUG_PRINT” defined, and the “printf()” statements in all the sample code are surrounded by “#ifdef DEBUG_PRINT” and “#endif”. The output of the “printf()” routine goes to the CCS “Console” window through the semi-hosting feature of the debugger.

When building for release (where debugger is not connected), the “printf()” statements can be excluded by removing the “DEBUG_PRINT” macro definition. Perform the following steps to remove (or add back) the “DEBUG_PRINT” macro symbol:

1. Select Project->Properties.
2. In the left area of the Properties window, expand Build->ARM Compiler and select “Predefined Symbols”.
3. The list of “Pre-define NAME (--define, -D)” should include “DEBUG_PRINT”.
4. Select “DEBUG_PRINT” and click the  icon to delete it.

(To add back the “DEBUG_PRINT” later, click the  icon and enter “DEBUG_PRINT”.)

5. Click “OK” and then rebuild the project.

2.3 System Workbench for STM32 and STM32CubeF7 Installation

System Workbench for STM32 or SW4STM32, is the selected development environment for use with the ST STM32 Nucleo-144 development boards (Nucleo-F746ZG or Nucleo-F767ZI). This section describes how to install and build projects using SW4STM32.

2.3.1 System Workbench for STM32

SW4STM32 is a free development environment offered by STMicroelectronics for use with their evaluation board products through the OpenSTM32 Community. Downloads and information about SW4STM32 can be found here: [SW4STM32](#) and [OpenSTM32](#).

The S1D13C00 software release has been built with SW4STM32 version 2.4.

Download the latest SW4STM32 package and install according to the instructions provided.

2.3.2 STM32CubeF7 Software Package

The STM32CubeF7 software package is a royalty free collection of library routines to control the STM32F7 MCU series features and peripherals and is required by the S1D13C00 sample software. It is recommended to download and install the latest STM32CubeF7 software package. When this document was written the current STM32CubeF7 package was 1.11.0.

Information and downloads for the STM32CubeF7 can be found here: [STM32CubeF7](#)

2.3.3 S1D13C00 Sample Software Installation

The S1D13C00 Sample Software and Peripheral Drivers software package is available for download from [Epson's website](#). Please ensure that you have the latest version before installing.

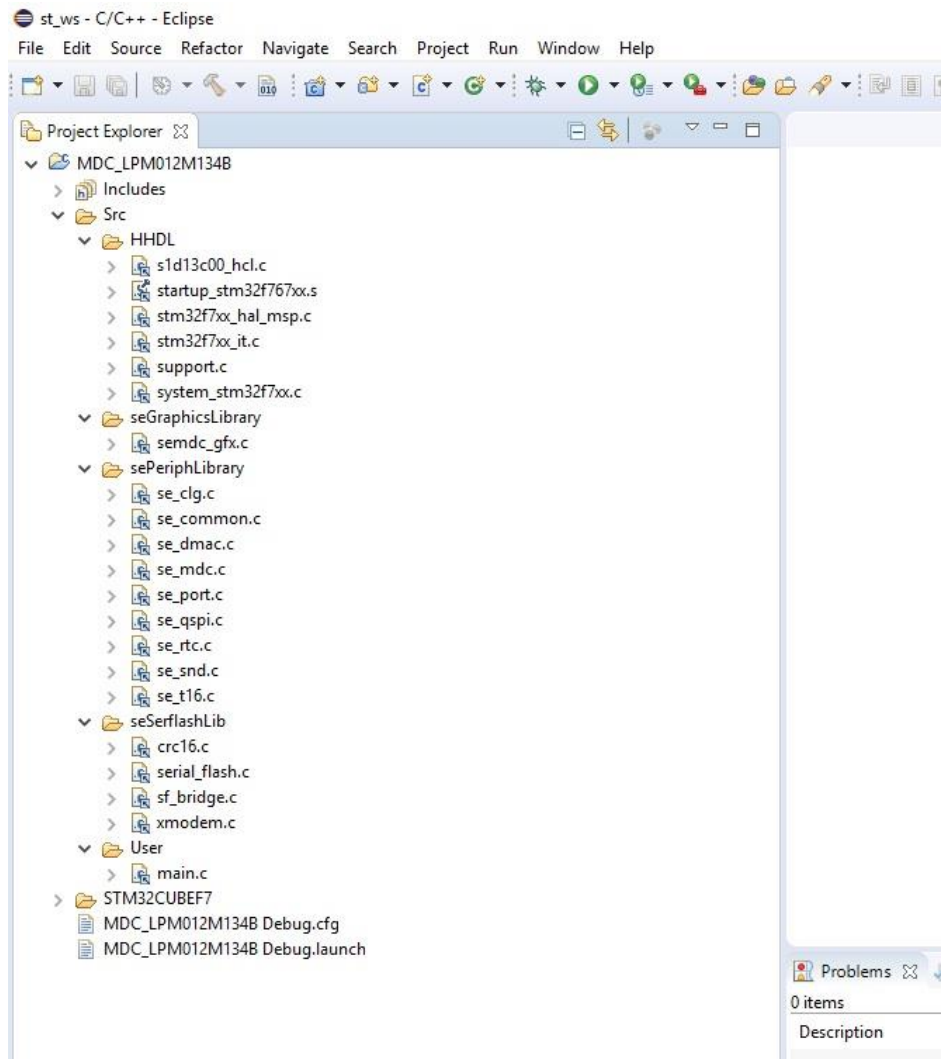
For distribution the installation package is encapsulated in an executable ZIP file, S1D13C00_SW.exe. Double click the executable file to begin installation. The default destination location for the sample software is c:\Epson. For the sample software projects to build correctly in the System Workbench and STM32CubeF7 environment, the destination location must be "<STM32CubeF7 location>\Projects", where <STM32CubeF7 location> is the root folder where the STM32CubeF7 was installed (for example, "C:\STM32Cube_FW_F7_V1.11.0").

Once setup has completed successfully, run the System Workbench for STM32 IDE. When the Eclipse Launcher prompts for a workspace directory, browse to an existing workspace or enter a new workspace directory location, then click OK to open the workspace.

The example projects are in the "S1D13C00_SW\Examples" subfolder of the folder where the sample software package was installed. To open one of the example projects, click the menu item "File->Open Projects from File System". In the "Import Projects from File System or Archive" window, click the "Directory" button to browse to the "S1D13C00_SW\Examples\<projectname>\HOSTS\<NUCLEOBoard>\SW4STM32" folder, where <projectname> is one of the example projects and <NUCLEOBoard> is either NUCLEO-F746ZG or NUCLEO-F767ZI. For example, an example project for displaying two types of watch faces on the LPM012M134B panel is "MDC_LPM012M134B".

2. Setup and Details of Sample Software Package

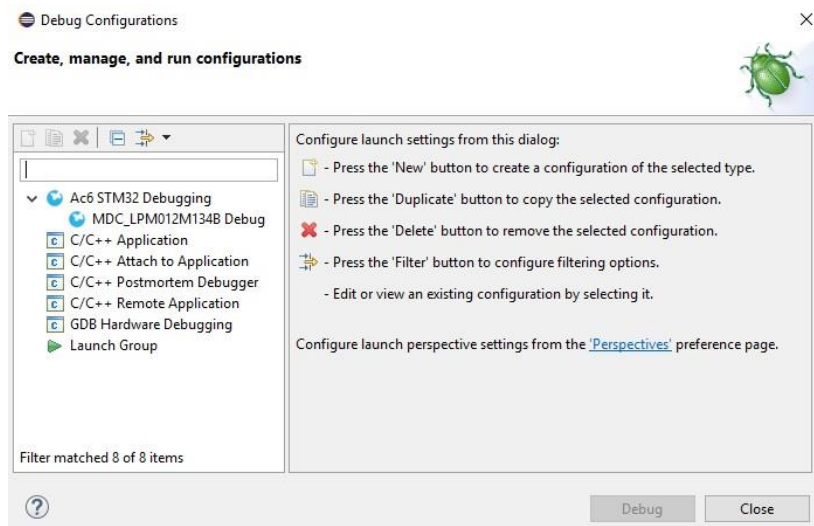
After selecting the project folder, click “Finish” in the “Import Projects from File System or Archive” window to open the project. The project name, for example “MDC_LPM012M134B”, will be displayed in the Project Explorer window of SW4STM32 (Eclipse). Expand the project by clicking on the ‘>’ arrow beside it. Sub-items can also be expanded by clicking on the ‘>’ arrow beside the item. For example, after expanding the “Src” sub-item and its sub-items, the following screen will be displayed:



2.3.4 Software Build and Target Download

To rebuild all the projects, select Project ->Build All or press the key combination <CTRL>

After a successful build, click a project to select it for debugging. Click Run->Debug Configurations to select the debug configuration for the project. The “Debug Configurations” window will be displayed.





Expand the “Ac6 STM32 Debugging” and select “<project> Debug”, where <project> is the project name. Click “Debug” to download, run, and debug the firmware on the target board.

For actual debugging procedures, please refer to the SW4STM32 documentation.

2.3.5 DEBUG_PRINT Macro Symbol

The example projects provided have a macro symbol “DEBUG_PRINT” defined, and the “printf()” statements in all the sample code are surrounded by “#ifdef DEBUG_PRINT” and “#endif”. The output of the “printf()” routine goes to the SW4STM32 “Console” window through the semi-hosting feature of the debugger.

When building for release (where debugger is not connected), the “printf()” statements can be excluded by removing the “DEBUG_PRINT” macro definition. Perform the following steps to remove (or add back) the “DEBUG_PRINT” macro symbol:

1. Select Project->Properties.
2. In the left area of the Properties window, expand “C/C++ Build” and click on “Settings”.
The “Tool Settings” tab will be displayed.
3. Expand “MCU GCC Compiler” and click “Preprocessor”.
4. The list of “Defined symbols (-D)” should include “DEBUG_PRINT”.
5. Select “DEBUG_PRINT” and click the  icon to delete it.
(To add back the “DEBUG_PRINT” later, click the  icon and enter “DEBUG_PRINT”.)
6. Click “OK” and then rebuild the project.

2. Setup and Details of Sample Software Package

2.4 Sample Software Components

The following diagram shows the structure of the S1D13C00 sample software package.

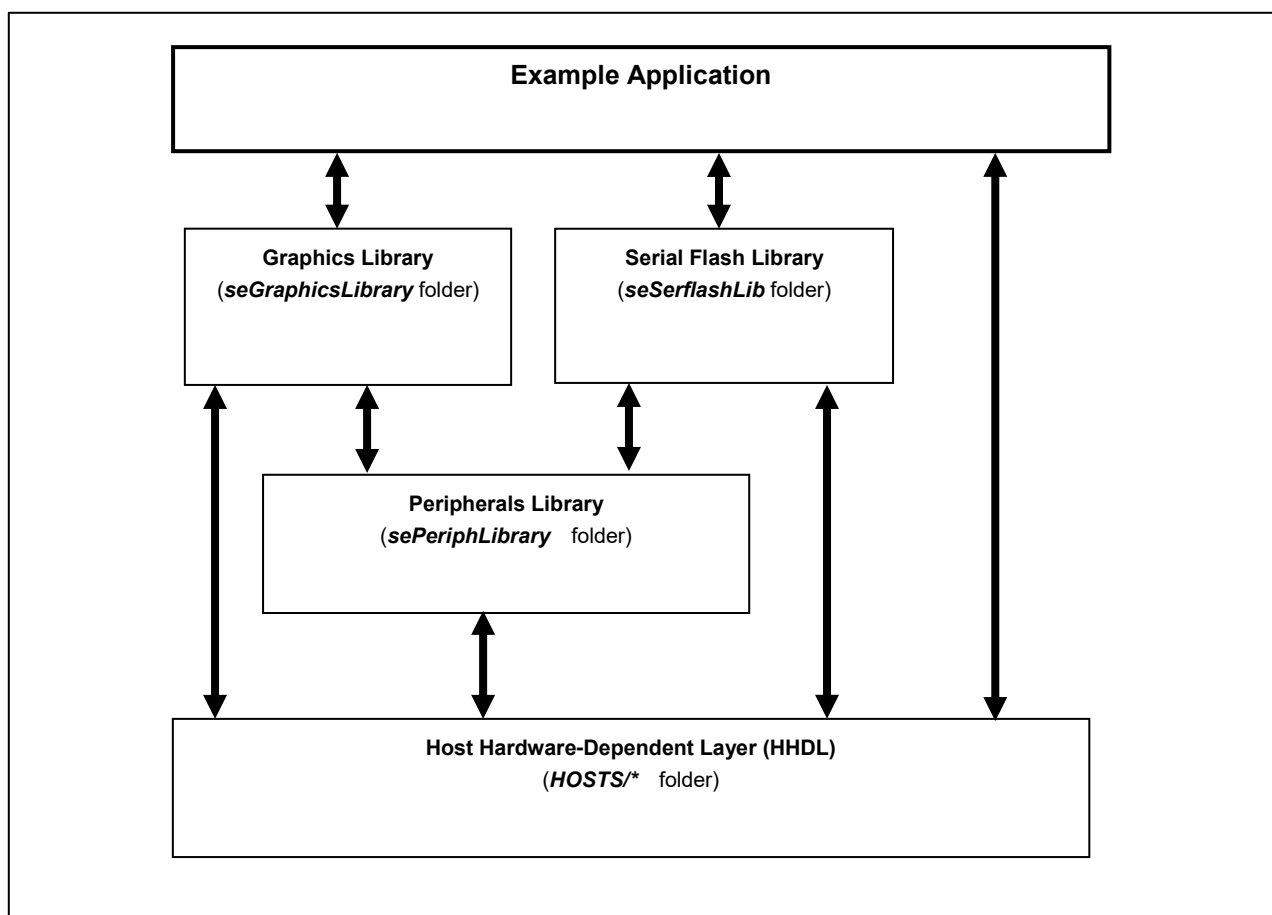


Figure 2.3 S1D13C00 Sample Software Structure

2.4.1 Host Hardware-Dependent Layer (HHDL)

Each host MCU board (EK-TM4C1294XL, STM32 Nucleo-144 development board) has its own set of hardware-dependent layer files:

- to provide read and write routines for accessing the S1D13C00 registers and RAM over the host interface (SPI/QSPI or Indirect 8-bit Parallel),
- to provide a common set of startup and vector table files for the specific MCU,
- to provide a common set of routines for the UART interface of the MCU for communicating over a Virtual COM port on the PC through the USB debug interface,
- to register a callback function for a push-button on the MCU board.

The HHDL files for the EK-TM4C1294XL board are located in the “S1D13C00_SW\HOSTS\EK-TM4C1294XL” folder and the HHDL files for the STM32 Nucleo-144 development boards are located in the “S1D13C00_SW\HOSTS\NUCLEO-144” folder.

The upper layer libraries use the hardware-abstracted routines provide by the HHDL.

There are two sets of common hardware-dependent files whose routines need to be implemented for each board: “support.*” and “s1d13c00_hcl.*”.

support.c

The “support.c” file contains the following routines:

- **InitializeHost()** – initialize the host MCU hardware, such as enabling and configuring clocks, configuring GPIO ports, and enabling interrupts. This should be the first function called in the “main.c” application program.
- **ConfigureHostInterrupt()** – configures the GPIO input which is connected to the HIFIRQ interrupt output signal of the S1D13C00 and registers the callback handler for the HIFIRQ interrupt.
- **EnableHostInterrupt()** – enables the GPIO input interrupt for the HIFIRQ signal.
- **DisableHostInterrupt()** – disables the GPIO input interrupt for the HIFIRQ signal.
- **InitializeMDC()** – initialize the host interface between the host MCU and the S1D13C00. This routine calls the “seS1D13C00InitializeController()” function in “s1d13c00_hcl.c” and needs to be called before the register/memory read and write functions in “s1d13c00_hcl.c” can be used. The parameter passed to this routine is the host interface type to use (“hostmcu_config” type) which are defined in “s1d13c00_hcl.h”. Configuration jumpers on the S5U13C00P00Cx00 board need to be set up according to the host interface used. See the S5U13C00P00CX00 Customer Development Board User Manual, document number XB8A-G-001-xx Host Interface settings section for more details. The default host interface used by the example program is “HOSTMCU_SPI_QUAD_ALL”, which selects the full QSPI interface (quad-data for command, address, and data).
- **InitializeButton1()** – initialize the GPIO input port used for the push-button and register the callback handler for the button interrupt. This only needs to be called if the button is used by the application.
- **InitializeUART** – initialize the UART interface which is connected to the Virtual COM port provide through the USB debugger interface. The UART is initialized to 115,200bps, 8 data bits, 1 stop bit, no parity, no handshake. UART transmit is blocking and uses pollng. UART receive is set up to be interrupt driven with a queue/buffer of received characters.
- **UARTSend()** – routine to send a string to the UART.
- **UARTWriteChar()** – routine to send a character to the UART.
- **UARTCharAvail()** – routine to check if characters are available in the UART RX buffer/queue.
- **UARTReadChar()** – routine to read a character from the UART RX buffer/queue.

s1d13c00_hcl.c

The “s1d13c00_hcl.c” file contains the following routines:

- **seS1D13C00InitializeController()** – initialize the host interface between the host MCU and the S1D13C00. It is called by the “InitializeMDC()” function in “support.c”.
- **seS1D13C00SoftReset()** – performs a soft reset of the S1D13C00 chip.
- **seS1D13C00Write()** – routine for writing a byte sequence to a memory location in the S1D13C00.
- **seS1D13C00Write8()** – routine for writing a byte to a memory location in the S1D13C00.
- **seS1D13C00Write16()** – routine for writing a 16-bit word (2 bytes) to a memory location in the S1D13C00 (little-endian, lower byte sent first).
- **seS1D13C00Write32()** – routine for writing a 32-bit word (4 bytes) to a memory location in the S1D13C00 (little-endian, lower byte sent first).
- **seS1D13C00Read()** – routine for reading a byte sequence from a memory location in the S1D13C00 to a buffer.
- **seS1D13C00Read8()** – routine for reading a byte from a memory location in the S1D13C00.

2. Setup and Details of Sample Software Package

- **seS1D13C00Read16()** – routine for reading a 16-bit word (2 bytes) from a memory location in the S1D13C00 (little-endian, lower byte read first).
- **seS1D13C00Read32()** – routine for reading a 32-bit word (4 bytes) from a memory location in the S1D13C00 (little-endian, lower byte read first).
- **seS1D13C00InitDispEn()** – routine to configure the GPIO output which is connected to the DISPEN input of the S5U13C00P00CX00 board for enabling/disable a 1-bit/3-bit SPI panel.
- **seS1D13C00DispEnable()** – routine to assert high the DISPEN input of the S5U13C00P00CX00 board to enable the 1-bit/3-bit SPI panel.
- **seS1D13C00DispDisable()** – routine to assert low the DISPEN input of the S5U13C00P00CX00 board to disable the 1-bit/3-bit SPI panel.

EK-TM4C1294XL Related Files

The following are files in the “S1D13C00_SW\HOSTS\EK-TM4C1294XL” folder which are specific to the EK-TM4C1294XL board:

- **tm4c1294ncpdt.cmd** – this file contains the linker settings for the project.
- **tm4c1294ncpdt_startup_ccs.c** – this file contains the interrupt vector table and default handlers.

STM32 NUCLEO-144 development board Related Files

The following are files in the “S1D13C00_SW\HOSTS\NUCLEO-144” folder which are specific to the STM32 NUCLEO-144 development board:

- **startup_stm32f746xx.s** – this file contains the interrupt vector table for the NUCLEO-F746ZG board.
- **startup_stm32f767xx.s** – this file contains the interrupt vector table for the NUCLEO-F767ZI board.
- **STM32F746ZGTx_FLASH.ld** – this file contains the linker settings for the NUCLEO-F746ZG project.
- **STM32F767ZITx_FLASH.ld** – this file contains the linker settings for the NUCLEO-F767ZG project.
- **Src/system_stm32f7xx.c** – this file contains routines for initializing the MCU system.
- **Src/stm32f7xx_it.c** – this file contains default and specific interrupt handler routines.
- **Src/stm32f7xx_hal_msp.c** – this file contains STM32Cube F7 HAL routines for the UART.

2.4.2 Host Interface Configuration

On the S5U13C00P00CX00 board, jumper JP19 selects the Host Interface between INDIRECT 8-Bit (JP19 pins 2 and 3 connected) and SPI/QSPI (JP19 pins 1 and 2 connected). Depending on the selection of JP19, jumpers JP14 to JP18 also need to be configured appropriately. See the S5U13C00P00CX00 Customer Development Board User Manual, document number XB8A-G-001-xx Host Interface settings section for more details. The default JP14 to JP19 configuration selects SPI/QSPI. For the SPI/QSPI selection, the HIFD[5:4] pins select the SPI/QSPI protocol type (Single/Extended, Dual, or Quad). HIFD[5:4] are driven from GPIO outputs on the Host MCU board and controlled by software.

The “InitializeMDC()” function initializes the Host Interface. The parameter passed to this routine is the host interface type to use (“hostmcu_config” type) which are defined in “s1d13c00_hcl.h”, and the selected type must match the JP14 to JP19 configuration. The host interface type selected for all sample projects in the “Examples” folder is “HOSTMCU_SPI_QUAD_ALL” (Quad protocol, 4-bit for Command, Address, and Data).

2.4.3 Peripherals Library (sePeriphLibrary)

The “S1D13C00_SW\sePeriphLibrary” folder contains routines for using the S1D13C00’s peripherals (MDC, SPI, I2C, SND, RTC, T16). The filenames are “se_<peripheral>.c”, where <peripheral> is the peripheral name (for example, “se_rtc.c” for RTC and “se_mdc.c” for MDC).

The S1D13C00’s registers and RAM address locations and bits are defined in “s1d13c00_memregs.h”.

2.4.4 Graphics Libray (seGraphicsLibrary)

The “S1D13C00_SW\seGraphicsLibrary” folder contains the “semdc_gfx.c/h” files. The “semdc_gfx.c” file has routines for drawing objects, images, and text bitmaps to the S1D13C00’s RAM (display frame buffer).

2.4.5 External Peripherals Library (seSerflashLib)

The “S1D13C00_SW\seSerflashLib” folder contains files for accessing the onboard serial flash on the S5U13C00P00CX00 board and for downloading a binary file to the serial flash over the UART interface using the XMODEM protocol.

2.4.6 Example Applications/Projects

The “S1D13C00_SW\Examples” folder contains subfolders of example application projects which use the library routines. The name of each sample project is related to the S1D13C00 peripheral which is used. For example, the “SND” example project demonstrates how to use the “se_snd.c” routines.

More details about each sample project is provided in Section 3.

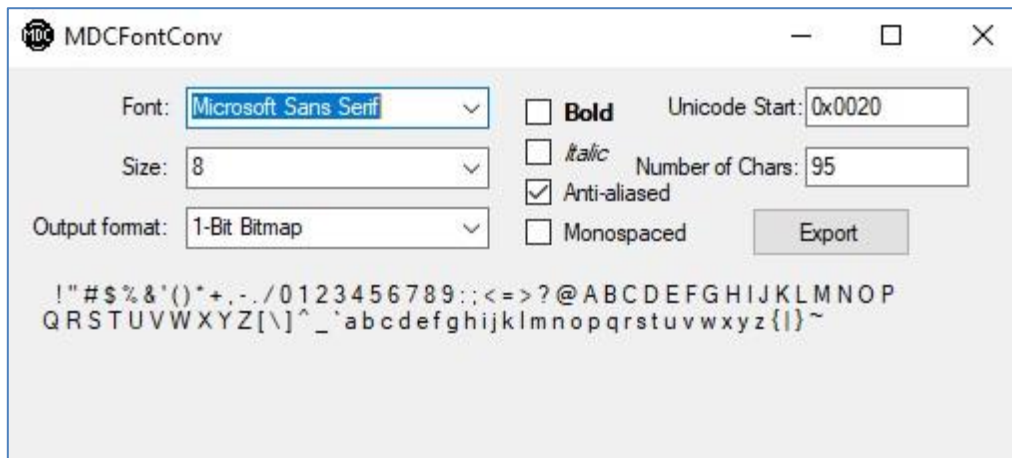
2.5 MDC Tools

The “S1D13C00_SW\Tools” folder contains software tools for converting images and fonts on the PC to formats suitable for inclusion in the embedded software system.

2. Setup and Details of Sample Software Package

2.5.1 Font Conversion MDCFontConv.exe

MDCFontConv.exe is a tool for generating font bitmaps header (.h) or binary files (.mdcfont) from fonts which are in the user's Windows system.



The tool allows the user to:

- Select the range of Unicode characters to be included for the font character set
- Select the size of the font
- Select between 1 or 2 bits per pixel format
- Choose type of font: bold, Italic, or anti-aliased
- Make any font be proportional or non-proportional (monospaced)

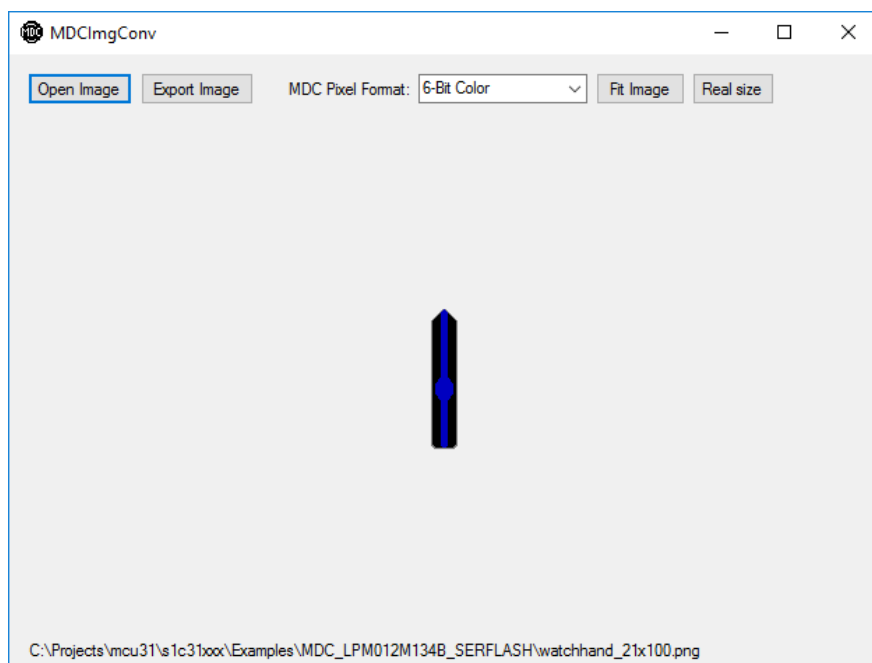
The output file (.h) contains a `seMDC_GFX_FontStruct` structure which has the following fields:

- `bitmapfmt` – specifies the bitmap format: 0=1-bit, 1=2-bit
- `height`– specifies the height of each and every character bitmap (in pixels)
- `numchars` – number of characters in the font set
- `unicode_base` – first Unicode character in the font set
- `*charstbl` – pointer to a table of (width, offsetloc) pairs for the characters in the font set. “width” is the width of the character bitmap and “offsetloc” is the offset location in the “pix_data” array of the start of the bitmap for the character
- `*pxdata` – pointer to byte array of the pixel data for the character bitmaps of the font set

The C header output file (.h) can be used in C code to include the font set in ROM data.

2.5.2 Image Conversion MDClmgConv.exe

MDClmgConv.exe is a tool for converting any common type of graphic image (BMP, PNG, JPG, ICO, TIF, GIF) to any pixel formats supported by the MDC. The tool can generate header files (.h), binary files (.mdcimg) or HEX files (.hex).



The tool allows user to:

- Select the MDC pixel format to output
- Preview look of the image according to selected pixel format
- Have image scaled to size of the tool window or actual size without affecting the image

The output file (.h, .hex, and .mdcimg) contains a seMDC_ImgStruct structure which has the following fields:

- width – width of the image (in pixels)
- height – height of the image (in pixels)
- stride – stride of the image (in pixels, same value as width)
- imgtype – specifies the MDC image format
- pxdata – pointer to byte array of the image pixels

The C header output file (.h) can be used in C code to include the image in ROM data.

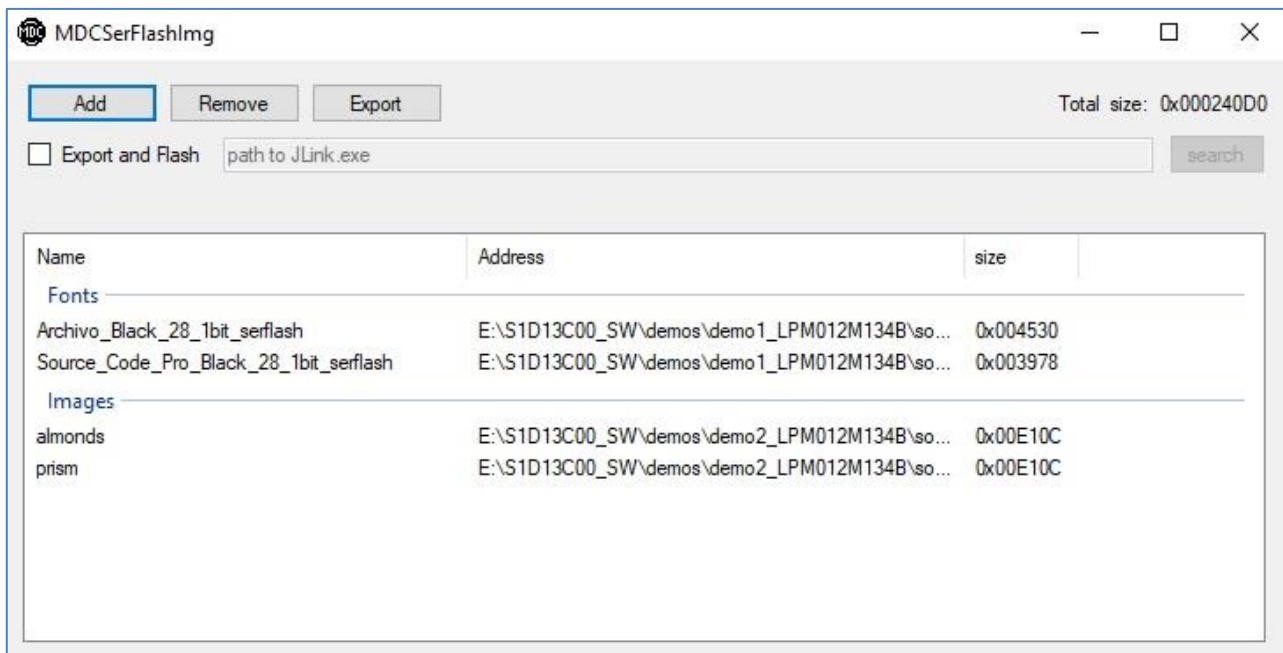
2. Setup and Details of Sample Software Package

2.5.3 Binary File for Serial Flash MDCSerFlashImg.exe

MDCSerFlashImg.exe is a tool which helps to create a binary image for downloading to the serial flash which is connected to the S1D13C00. The tool accepts binary files generated by MDCImgConv.exe and MDCFontConv.exe. The output of the tool consists of two files: a header file with addresses of all included items (to be used by the application) and a single binary image which is a collection of all the image and font binaries (.mdcfont and .mdcimg files) specified to be included in the single binary image.

The tool allows user to:

- Add/Remove font (.mdcfont) and image (.mdcimg) files to include
- Export (generate) a C header file (.h) and a single binary image (.bin)



The output C header file (.h) contains definitions for the locations of fonts and images in the binary file. It can be included in an application to access the fonts and images after the binary image is downloaded into the serial flash.

The tool will not accept files larger than 1 MBytes and total image size cannot be larger than 16 MBytes.

3 Details of Sample Software

This section provides some detail about the sample software projects in the “Examples” folder. There are several projects in the “Examples\MDC_*” folders which demonstrate programming for different memory display panels. Only the “Examples\MDC_LPM012M134B” is described in this manual. The LPM012M134B is the display panel installed with the S5U13C00P00Cx00 board.

Hardware setup is required for some examples.

3.1 Clock Generator (CLG)

CLG module is the clock generator that controls the clock sources and manages clock supply to the S1D13C00.

Operations

1. Initializes CLG.
2. Select IOSC as T16 source.
3. Trim adjust IOSC to 0 and count T16 interrupts detected in 5 seconds.
4. Trim adjust IOSC to 63 and count T16 interrupts detected in 5 seconds.
5. Select internal 32kHz OSC1 as T16 source.
6. Trim adjust OSC1 to 0 and count T16 interrupts detected in 5 seconds.
7. Trim adjust OSC1 to 63 and count T16 interrupts detected in 5 seconds.

Example of Output

```
Host MCU configured...
S1D13C00 Interface configured...
HIFIRQ (Host Interrupt port) configured...
```

```
Selecting IOSC as T16 source
Setting IOSC frequency to 20 MHz

Setting IOSC trim adjust to 0
Starting measurement ...
Interrupted: 157 times during 5 sec
```

```
Setting IOSC trim adjust to 63
Starting measurement ...
Interrupted: 257 times during 5 sec
```

```
Selecting OSC1

Setting OSC1 trim adjust to 0
Starting measurement ...
Interrupted: 40 times during 5 sec
```

3. Details of Sample Software

```
Setting OSC1 trim adjust to 63
Starting measurement ...
Interrupted: 40 times during 5 sec
```

```
Exit
```

3.2 DMA Controller (DMAC)

The DMAC module is used for data transfers between memory and peripheral devices.

This example shows:

- Using DMAC in a basic memory-to-memory data transfer
- Using DMAC from memory to a peripheral data transfer

Operations

Example 1: Memory to Memory DMA transfer

- DMAC Channel 0 is configured to transfer the contents of a data buffer stored in RAM to the reception buffer also in RAM. Access size is Byte.
- DMAC Channel 1 is configured to transfer the contents of a data buffer stored in RAM to the reception buffer in RAM. Access size is Half Word.
- DMAC Channel 2 is configured to transfer the contents of a data buffer stored in RAM to the reception buffer in RAM. Access size is Word.
- The start of transfer is triggered by software.

In this example:

1. The DMAC interrupts are not enabled.
2. DMAC Channel transfer is enabled.
3. Source and destination addresses incrementing is enabled.
4. The transfer is started by setting the Software Request register bits.
5. At the end of the transfer, a Transfer Completion interrupt flag is generated.
6. Once interrupt flag is generated, the "number of transfers" is read which must be equal to 0.
7. The Transfer Complete Interrupt flag is then cleared.
8. A comparison between the source and destination buffers is done to check that all data have been correctly transferred.

Example 2: Memory to Peripheral DMA transfer

- DMAC Channel 0 is configured to transfer data from a buffer stored in RAM memory to SND data register.
- SND is configured to play melody.
- The start of transfer is triggered by SND.

In this example:

1. The DMAC interrupts are not enabled.
2. DMAC Channel transfer is enabled.
3. DMAC Channel filtering is disabled for the selected DMAC Channel.
4. Source address incrementing is enabled.
5. Destination address incrementing is disabled.
6. The transfer is started by sound buffer empty.

7. At the end of the transfer, a Transfer Completion interrupt flag is generated.
8. Once interrupt flag is generated, the "number of transfers" is read which must be equal to 0.
9. The Transfer Complete Interrupt flag is then cleared.
10. Correctness of the transfer is verified by sound of the playing melody.

Example of Output

```
Host MCU configured...
S1D13C00 Interface configured...
HIFIRQ (Host Interrupt port) configured...

DMA Initializing
DMA Test: Memory
DMA Test: SND

Exit
```

3.3 I2C

This example provides a description of how to use a I2C in the master mode with EEPROM 24AA014 EEPROM chip.

Hardware Setup

The S5U13C00P00Cx00 evaluation board has a 24AA014 EEPROM chip which is connected to ports P04 and P05 of the S1D13C00.

```
S1D13C00          24AA014
[master]         [slave]
P04 SCL-----><-----SCL
P05 SDA-----><-----SDA
```

Operations

This example tests the I2C module with 24AA014 EEPROM at slave address 0x50.

1. Configure I2C for master mode and set up port pins for I2C interface.
2. Write "Test 0 1 2 4" string to EEPROM.
3. Read back from EEPROM and check for "Test 0 1 2 4" string.

Example of Output

```
Host MCU configured...
S1D13C00 Interface configured...
Testing I2C module with EEPROM at 0x50
Write data: Test 0 1 2 4
Read data: Test 0 1 2 4
Status: OK
Exit
```

3. Details of Sample Software

3.4 MDC Example for LPM012M134B Panel (MDC_LPM012M134B)

MDC is a Controller for Memory Displays with a Drawing and Copying engine.

This software shows examples of using the routines in "se_mdc.c" peripheral library and "se_gfx.c" graphics library for initializing a panel, using drawing functions, and using image/bitmap copying functions. There are two clock faces displayed at a time, selectable by the SW1 button on the EK-TM4C1294XL board.

Hardware Setup

Example requires an LPM012M134B panel connected to connector CN3 of the S1D13C00 evaluation board.

Operations

1. Initialize PJ0 port of MCU as input with port interrupt handler for SW1 button.
2. Start OSC1 and change IOSC frequency to 20MHz.
3. Initialize LPM012M134B panel.
4. Set output destination window for MDC graphics engine
5. Initialize and start RTC.
6. Read latest time from RTC.
7. Display time/date using the selected clock face routine.
8. Wait for 1-second interrupt from RTC.
9. Repeat 6 to 8.

3.5 I/O Ports (PORT)

The PORT module controls the general-purpose input/output (GPIO) pins of the S1D13C00.

This example configures pin P06 as an output connected to pin P07 configured as an input. It performs various tests to check the connectivity of P06 and P07.

Hardware Setup

For this example program, connect P06 to P07.

```
P06<----->P07
```

Operations

1. Start IOSC and OSC1. Select OSC1 as the clock source for the PORT module.
2. Set the P06 port to output and output a Low level signal.
3. Set the P07 port to input and set it so that an interrupt occurs when the level changes from Low to High.
4. Output a High level signal from the P06 port.
5. Confirm that the P07 port is at High level after an interrupt in the P07 port.
6. Set the P07 port so that an interrupt occurs when the level changes from High to Low.
7. Output a Low level signal from the P06 port.
8. Confirm that the P07 port is at Low level after an interrupt in the P07 port.

Example of Output

```
Host MCU configured...
S1D13C00 Interface configured...
HIFIRQ (Host Interrupt port) configured...

P06 output state: Low
P07 input state: Low
P07 input waiting for RISING edge interrupt
P06 output is going HIGH...
Interrupt occurred on P07
P06 output state: High
P07 input state: High

P06 output state: High
P07 input state: High
P07 input waiting for FALLING edge interrupt
P06 output is going LOW...
Interrupt occurred on P07
P06 output state: Low
P07 input state: Low

Exit
```

3. Details of Sample Software

3.6 Quad Synchronous Serial Interface Master (QSPI_MASTER)

QSPI module is a Quad Synchronous Serial Interface supporting both master and slave modes.

This example provides a description of how to use a QSPI in the SINGLE data master mode and the SPI in slave mode.

Hardware Setup

1. All 4 of the S1 DIP switch sliders on the S5U13C00P00Cx00 board should be in the ON position.

The 4 DIP switch sliders in the ON position makes the following connections:

```
[master]                [slave]
QSDIO0----->>-----SDI (P02)
QSDIO1-----><-----SDO (P03)
QSPICLK----->>-----SPICLK (P01)
#QSPISS----->>-----#SPISS (P00)
```

Operations

1. Initialize the QSPI module in master mode as below:
 - Data length is 8bit
 - Data format is MSB first
 - Use 16-bit timer T16_CH2 for baud rate generator.
2. Initialize the SPI module in slave mode.
3. Set bus speed to 5000000.
4. Send the 'BUF_SIZE' bytes of random data to the slave.
5. Receive the 'BUF_SIZE' bytes of data from the slave.
6. Compare whether the received data is the same as the sent data.
7. Repeat steps 4 to 6.

Example of Output

```
Host MCU configured...
S1D13C00 Interface configured...

Set bus speed 5000000
Get bus speed 5000000
- OK(1), NG(0)
- OK(2), NG(0)
- OK(3), NG(0)
- OK(4), NG(0)
- OK(5), NG(0)
- OK(6), NG(0)
```

3.7 Quad Synchronous Serial Interface Slave (QSPI_SLAVE)

QSPI module is a Quad Synchronous Serial Interface supporting both master and slave modes.

This example provides a description of how to use a QSPI in the SINGLE data master mode and the SPI in slave mode.

Hardware Setup

1. All 4 of the S1 DIP switch sliders on the S5U13C00P00Cx00 board should be in the ON position.

The 4 DIP switch sliders in the ON position makes the following connections:

```

[master]                [slave]
QSDIO0----->>-----SDI (P02)
QSDIO1-----><-----SDO (P03)
QSPICLK----->>-----SPICLK (P01)
#QSPISS----->>-----#SPISS (P00)

```

Operations

1. Initialize the SPI module in master mode as below:
 - Data length is 8bit
 - Data format is MSB first
 - Use 16-bit timer T16_CH1 for baud rate generator.
2. Initialize the QSPI module in slave mode.
3. Set bus speed to 5000000.
4. Send the 'BUF_SIZE' bytes of random data to the slave.
5. Receive the 'BUF_SIZE' bytes of data from the slave.
6. Compare whether the received data is the same as the sent data.
7. Repeat steps 4 to 6.

Example of Output

```

Host MCU configured...
S1D13C00 Interface configured...

Set bus speed 5000000
Get bus speed 5000000
- OK(1), NG(0)
- OK(2), NG(0)
- OK(3), NG(0)
- OK(4), NG(0)
- OK(5), NG(0)
- OK(6), NG(0)

```

3. Details of Sample Software

3.8 IR Remote Controller (REMC)

The REMC circuit generates infrared remote control output signals.

This example programs the REMC to generate various IR pulses.

Hardware Setup

Two jumper wires are needed to connect P00 (H2-5) and P01 (H2-4) to ground to emulate the SW2 and SW3 push-buttons described below.

Operations

This example software uses the NEC format for output.

(Two virtual push-button switches SW2 and SW3 are defined. SW2 "push" occurs when port P00 is connected to GND and disconnected. SW3 "push" occurs when port P01 is connected to GND and disconnected.)

1. REMC determines the data waveform using two parameters: APLEN (data signal duty) and DBLEN (data signal cycle). Each of APLEN and DBLEN defines the waveform with 34 frame signals and 2 repeat signals. APLEN and DBLEN pair allocates the leader, customer code, and repeat signals at initialization time. Those signals are common to different scenarios when switches SW2 to SW3 are pushed. Data may vary according to selection of the SW2 to SW3. Data is allocated when a switch interrupt occurs.
2. Further initialization consists of configuration of switches SW2 to SW3, allocation ports for REMC, and initialization of T16. Then program enters into an endless loop to wait for interrupts caused by pushing switches SW2 to SW3.
3. When an interrupt occurs allocate data's APLEN and DBLEN values. Since REMC uses a buffer mode, REMC should be activated after initialization. Also, activate T16 and generate a T16 interrupt after about 108ms.
4. When a compare DB interrupt occurs in REMC, write the following APLEN and DBLEN data into register. When writing last but one data element, change REMC to a one-shot mode to prevent the whisker-like signal output. After last data element, stop the REMC operation.
5. When a T16 interrupt occurs, check whether the same switch, SW2 to SW3, was pushed. If the same switch was pushed, set the register values of REMC so that a repeat waveform is generated, initialize and activate REMC. Otherwise, stop the T16 operation.

3.9 Real Time Clock (RTC)

RTC module is a real-time clock with a perpetual calendar function.

This example shows how to program and read the various functions of the real-time clock such as the time/date, stopwatch, alarm, and trimming.

Operations

1. Initializes RTC.
2. Starts RTC.
3. Calculates TRM.
4. Sets time and date, and reads it back.
5. Sets an alarm and CPU goes to sleep. Expects interrupt while sleeping.
6. Starts 1 second timer to perform trimming.
7. Checks stop watch operations.

Example of Output

```
Host MCU configured...
S1D13C00 Interface configured...
HIFIRQ (Host Interrupt port) configured...

RTC TRM bits 0x43

RTC Set hour 4(PM), minute 17, second 0
RTC Get hour 4(PM), minute 17, second 0

RTC Set year 15, month 2, day 5, Thursday
RTC Get year 15, month 2, day 5, Thursday

RTC Set Alarm hour 4(PM), minute 17, second 10
RTC waiting for alarm....
RTC Alarm occurred : hour 4(PM), minute 17, second 10

RTC Start Trimming
RTC waiting for timer...
RTC Trimming performed : hour 4(PM), minute 17, second 16

RTC StopWatch (start) : hour 4(PM), minute 17, second 17
RTC sleep about 5 seconds waiting for Stop....
StopWatchHW: 4.99
Exit
```

3. Details of Sample Software

3.10 Serial Flash Download (SERFLASH_DOWNLOAD)

The S5U13C00P00Cx00 board has an onboard serial flash device connected to the QSPI interface of the S1D13C00 chip.

This example program utilizes the use of library routines to download a binary file to the serial flash over the UART interface enumerated as a Virtual COM port over the USB connection to the PC.

Hardware Setup

The Virtual COM port connection to the MCU board should be configured for 115,200bps, 8 bits, 1 stop bit, no parity, no handshake. A PC serial port terminal emulation program such as TeraTerm can be used to interact with the firmware to download/burn a binary image to the serial flash.

Operations

When the program is run, the following message is sent to the Virtual COM port:

```
-----  
Xmodem transfer of binary file to serial flash  
-----  
D - Download binary file to serial flash starting from zero offset.  
  
Select Test:
```

When user presses 'D' key, the following message is displayed:

```
Identified flash [01 2018]  
Erasing whole flash. Can take long time...  
Send data using the xmodem protocol from your terminal emulator now...
```

Send the binary file by XMODEM protocol using the terminal emulator program (such as TeraTerm).

Example of Output

```
Host MCU configured...  
S1D13C00 Interface configured...
```


3.11 Sound Generator (SND)

SND is a sound generator that generates melodies and buzzer signals.

This example programs the SND to generate various tones and music.

Operations

1. Example starts OSC1 and initializes SND.
2. SND runs in normal buzzer mode for 5 sec.
3. SND runs in one-shot buzzer mode for 250msecs.
4. SND runs in melody mode.

Example of Output

```
Host MCU configured...
```

```
S1D13C00 Interface configured...
```

```
Start SND in normal buzzer mode for 5 sec
```

```
Start SND in one-shot buzzer mode, 250ms
```

```
Start SND in melody mode
```

```
Start SND playing of 'Postoj Parovoz' music
```

```
SND Exit
```

3. Details of Sample Software

3.12 Synchronous Serial Interface Master (SPI_MASTER)

SPI module is a synchronous serial interface supporting both master and slave modes. Only one SPI channel is available in S1D13C00.

This example provides a description of how to use the SPI channel in master mode to transfer data buffer.

Hardware Setup

1. Connect the evaluation boards where the SPI master/slave sample programs are installed.
Connect each port as follows:

	[master]		[slave]	
H2-3	SDIn-----<<-----	SDOn	H2-2	
H2-2	SDOn----->>-----	SDIn	H2-3	
H2-4	SPICLKn----->>-----	SPICLKn	H2-4	
H2-5	#SPISS----->>-----	#SPISS	H2-5	
H1-3	P05----->>-----	P05	H1-3	
H1-8	GND-----><-----	GND	H1-8	

2. Ensure all 4 of the S1 DIP switch positions are in the OFF position.
3. Launch the slave example program first, then the master program

Operations

1. Initialize the SPI module in master mode as below:
 - Data length is 8bit
 - Data format is MSB first
 - Use 16-bit timer T16_1 for baud rate generator.
2. Set bus speed to 1000000.
3. Assign GPIO P05 as handshake signal to slave.
4. Send the 'BUF_SIZE' bytes of random data to the slave.
5. Receive the 'BUF_SIZE' bytes of data from the slave.
6. Compare whether the received data is the same as the sent data and exit.
7. Repeat

Example of Output

```
Host MCU configured...
S1D13C00 Interface configured...
Set bus speed 1000000
Get bus speed 1000000
- OK(1), NG(0)
- OK(2), NG(0)
- OK(3), NG(0)
- OK(4), NG(0)
- OK(5), NG(0)
```

3.13 Synchronous Serial Interface Slave (SPI_SLAVE)

SPI module is a synchronous serial interface supporting both master and slave modes. Only one SPI channel is available in S1D13C00.

This example provides a description of how to use a SPI channel in the slave mode to transfer data buffer.

Hardware Setup

1. Connect the evaluation boards where the SPI master/slave sample programs are installed.
Connect each port as follows:

	[master]		[slave]	
H2-3	SDIn-----<<-----		SDOn	H2-2
H2-2	SDOn----->>-----		SDIn	H2-3
H2-4	SPICLKn----->>-----		SPICLKn	H2-4
H2-5	#SPISS----->>-----		#SPISS	H2-5
H1-3	P05----->>-----		P05	H1-3
H1-8	GND-----><-----		GND	H1-8

2. Ensure all 4 of the S1 DIP switch positions are in the OFF position.
3. Launch the slave example program first, then the master program

Operations

1. Initialize the SPI module in slave mode as below:
 - Data length is 8bit
 - Data format is MSB first
2. Assign GPIO P05 as input handshake signal from master which generates an interrupt.
3. In the handler to service the rising edge interrupt of P05, perform the following:
 - a) Receive the 'BUF_SIZE' bytes of data from the master.
 - b) Send the 'BUF_SIZE' bytes of random data to the master.

Example of Output

Slave has no output.

3. Details of Sample Software

3.14 16-Bit Timer (T16)

T16 module is a 16-bit timer.

This example programs the T16 channel 0 timer to interrupt periodically and counts the number of interrupts that occurred within a 5-second interval.

Operations

1. IOSC is selected as timer clock source. Example code does IOSC auto-trimming to provide clock accuracy for the timer clock.
2. Initializes T16 channel 0.
3. Configures T16 channel 0 interrupts.
4. Enables T16 interrupts.
5. Counts a number of interrupts happened during 5 second interval.

Example of Output

```
Host MCU configured...
S1D13C00 Interface configured...
HIFIRQ (Host Interrupt port) configured...
IRQ interrupted: 202 times during 5 sec
Exit
```

4 Revision History

Attachment-1

Rev. No.	Date	Page	Category	Contents
1.12	23/01/2019			Minor edits to clarify some references to STM32 products Minor formatting
1.11	12/09/2018			Minor edit to remove some text.
1.1	07/10/2018			- Major update to Section 2 to add support for ST Nucleo-F746ZG and Nucleo-F767ZI board development environments. - Updated output messages to match latest SW changes. - Added "Serial Flash Download" example program section. - Updated Section 2.5.1 "MDCFontConv.exe" description.
1.0	05/09/2018			Released as Rev 1.0

5. Sales and Technical Support

5 Sales and Technical Support

For more information on Epson Display Controllers, visit the Epson Global website.

https://global.epson.com/products_and_drivers/semicon/products/display_controllers/



For Sales and Technical Support, contact the Epson representative for your region.

https://global.epson.com/products_and_drivers/semicon/information/support.html

