

**S1D13C00 Memory Display Controller**

# **ET011TT2 Demo Software Manual**

**Document Number: XB8A-B-002-01.0**

## 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

<b>1</b>	<b>Overview</b>	<b>3</b>
<b>2</b>	<b>Hardware Preparation for Target Download and Execution</b>	<b>4</b>
2.1	Hardware Connections for TI EK-TM4C1294XL Host MCU Board	4
2.2	Hardware Connections for ST STM32 Nucleo-144 development board	5
<b>3</b>	<b>Demo Firmware Binary Setup</b>	<b>6</b>
3.1	<b>Demo Firmware Binary Setup for TI EK-TM4C1294XL Board</b>	<b>6</b>
3.1.1	TI Stellaris ICDI USB Driver	6
3.1.2	TI LM Flash Programmer Software	7
3.2	<b>Demo Firmware Binary Setup for ST Nucleo Boards</b>	<b>9</b>
3.2.1	ST-Link USB Driver	9
3.2.2	STM32CubeProgrammer Software	9
<b>4</b>	<b>Terminal Emulation Software and Serial Flash Custom Images</b>	<b>11</b>
<b>5</b>	<b>Tools for Custom Image Display</b>	<b>14</b>
5.1.1	Font Conversion MDCFontConv.exe	14
5.2	Image Conversion MDClmgConv.exe	15
5.3	Binary File for Serial Flash MDCSerFlashlmg.exe	18
<b>6</b>	<b>Demo Firmware Source Code Setup</b>	<b>20</b>
6.1	<b>Code Composer Studio and TivaWare Installation</b>	<b>20</b>
6.1.1	Code Composer Studio	20
6.1.2	TivaWare	20
6.1.3	Demo Software Source Code Package	23
6.1.4	Software Build and Target Download	23
6.1.5	DEBUG_PRINT Macro Symbol	24
6.2	<b>System Workbench for STM32 and STM32CubeF7 Installation</b>	<b>25</b>
6.2.1	System Workbench for STM32	25
6.2.2	STM32CubeF7 Software Package	25
6.2.3	Demo Software Source Code Package	25
6.2.4	Software Build and Target Download	27
6.2.5	DEBUG_PRINT Macro Symbol	27
6.3	<b>Demo Software Components</b>	<b>28</b>
6.3.1	Host Hardware-Dependent Layer (HHDL)	28
6.3.2	Host Interface Configuration	31
6.3.3	Peripherals Library (sePeriphLibrary)	32
6.3.4	Graphics Libray (seGraphicsLibrary)	32
6.3.5	External Peripherals Library (seSerflashLib)	32
6.4	eink2C.exe Conversion Program	32
<b>7</b>	<b>Details of Demo Software</b>	<b>33</b>
7.1	GC4 Pictures	34
7.2	GU4 Graphics	35
7.3	A2 Scroll	36
7.4	GU4 White	37

---

7.5 GC4 User .....	38
8 Revision History.....	39

## 1 Overview

This manual describes how to install and use the demo software for the S1D13C00 memory display controller (MDC) driving an E Ink ET011TT2 round 240x240 EPD panel on the S5U13C00P01C100 board.

The **S1D13C00 ET011TT2 Demo Software** package includes:

- Demo firmware binary files for the supported Host MCU boards
- Serial flash binary file with 4 user images to display in the demo

Before running the demo 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 STM32F746ZG)
  - ST Nucleo-F767ZI Board (STM32 Nucleo-144 development board with STM32F767ZI)
- S5U13C00P01C100 Board (direct connect for TI EK-TM4C1294X1)
- 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 S5U13C00P01C100 Board, refer to the *S5U13C00P01C100 Board User Manual*, document number XB8A-G-003-xx. For detailed information on the S5U13C00M00C100 Adapter Board, refer to the *S5U13C00M00C100 Adapter Board User Manual*, document number XB8A-G-002-xx.

## 2. Hardware Preparation for Target Download and Execution

---

## 2 Hardware Preparation for Target Download and Execution

The S5U13C00P01C100 Board is designed to connect directly to the TI EK-TM4C1294XL Launchpad host MCU board. For the ST STM32 Nucleo-144 development boards, the S5U13C00M00C100 Adapter Board to connect the S5U13C00P01C100 board to the STM32 Nucleo-144 development board.

### 2.1 Hardware Connections for TI EK-TM4C1294XL Host MCU Board

The following figure shows the S5U13C00P01C100 evaluation board connected to a TI EK-TM4C1294XL Launchpad board.

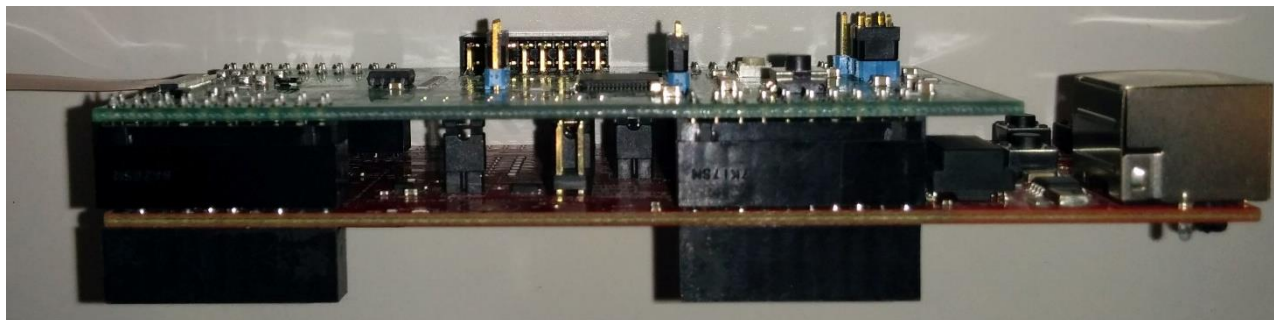


Figure 2.1 Physical Connection of S5U13C00P01C100 Board to EK-TM4C1294XL

For specific evaluation board configuration refer to the *S5U13C00P01C100 Board User Manual*, document number XB8A-G-003-xx.

The EK-TM4C1294XL host MCU board is connected and powered through a Micro-USB cable to a PC. Before connecting the hardware, the Stellaris ICDI USB driver (needed for Virtual COM port and SWD debugger interface) should be installed on the PC. See Section 3.

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.

### 2.2 Hardware Connections for ST STM32 Nucleo-144 development board

The following figure shows the S5U13C00P01C100 Board connected to a STM32 Nucleo-144 development board.

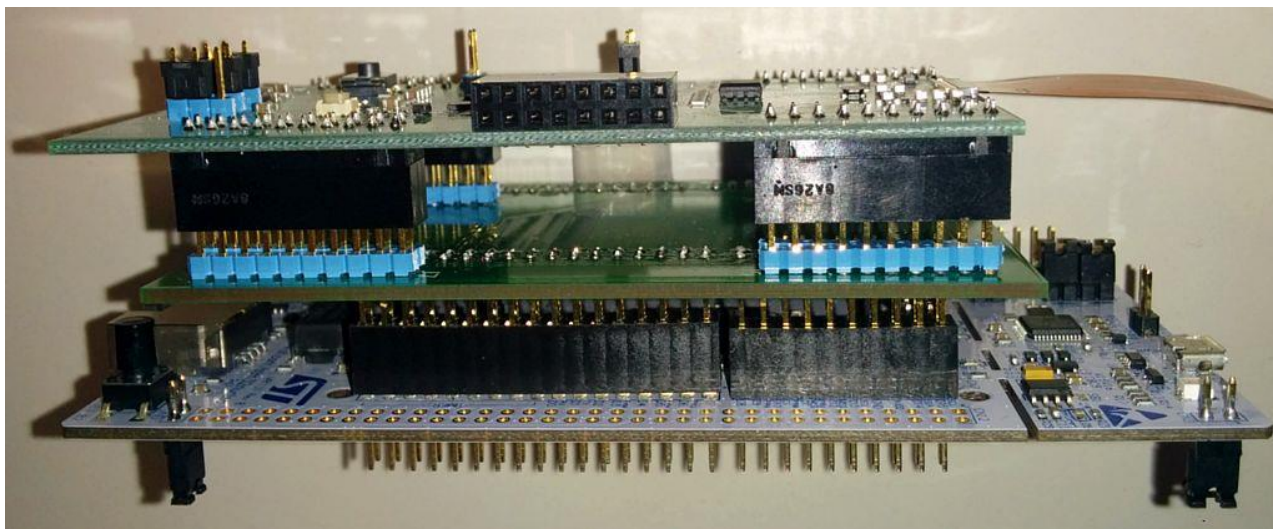


Figure 2.2 Physical Connection of S5U13C00P01C100 Board to a STM32 Nucleo-144 development board

The top board is the S5U13C00P01C100 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, the STSW-LINK009 driver (needed for Virtual COM port and SWD debugger interface) should be installed on the PC. See Section 3.2.

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.

### 3. Demo Firmware Binary Setup

---

## 3 Demo Firmware Binary Setup

This section describes installations and setup needed if the user only receives a binary file (not source code package) for the demo firmware.

### 3.1 Demo Firmware Binary Setup for TI EK-TM4C1294XL Board

The following software and firmware are needed for installing and running the demo:

- TI Stellaris ICDI USB driver
- TI LM Flash Programmer software for writing the demo firmware to the EK-TM4C1294XL board
- A terminal emulation software such as TeraTerm
- The “demo2\_ET011TT2\_EK-TM4C1294XL.bin” firmware binary file

#### 3.1.1 TI Stellaris ICDI USB Driver

The TI Stellaris ICDI USB driver is available for download from [http://www.ti.com/tool/STELLARIS\\_ICDI\\_DRIVERS](http://www.ti.com/tool/STELLARIS_ICDI_DRIVERS).

Before connecting the board, install this driver first. This driver provides the debugger interface needed to write the demo firmware binary file to the TM4C1294XL MCU's internal flash memory. It also provides a Virtual COM (serial) port which is needed to write a binary file to the onboard serial flash on the S5U13C00P01C100 board.

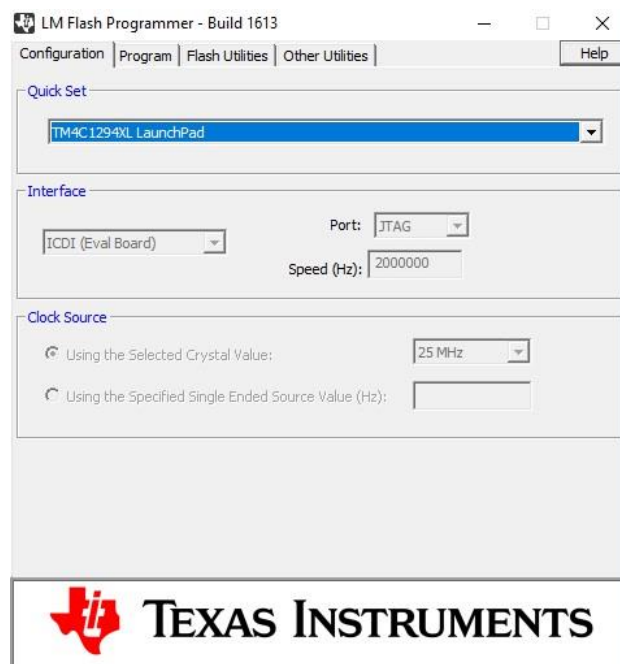


#### 3.1.2 TI LM Flash Programmer Software

The TI LM Flash Programmer is available for download from <http://www.ti.com/tool/LMFLASHPROGRAMMER>.

Download and install this program.

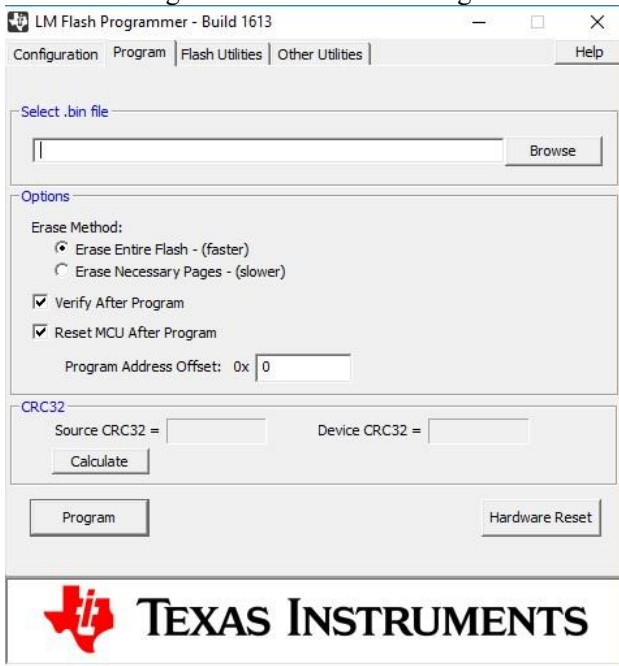
After installation, connect the EK-TM4C1294XL board to the PC and run the LM Flash Programmer software. The following window is displayed:



### 3. Demo Firmware Binary Setup

---

1. In the “Quick Set” list, select “TM4C1294XL Launchpad”.
2. Click the “Program” tab. The following window is displayed:



3. Click “Browse”, then find and select the “demo2\_ET011TT2\_EK-TM4C1294XL.bin” demo firmware file.
4. Make sure the “Verify After Program” and “Reset MCU After Program” boxes have check mark beside them.
5. Click “Program” to write the binary file to the EK-TM4C1294XL board.

After the LM Flash Programmer has finished programming, it restarts the board and the demo should be running. Section 6 describes the details of the demo items.

## 3.2 Demo Firmware Binary Setup for ST Nucleo Boards

The following software and firmware are needed for installing and running the demo:

- ST-Link USB driver
- STM32CubeProgrammer software
- A terminal emulation software such as TeraTerm
- The “demo2\_ET011TT2\_NUCLEO-F746ZG.bin” or “demo2\_ET011TT2\_NUCLEO-F767ZI” firmware binary file

### 3.2.1 ST-Link USB Driver

The ST-Link USB driver is available for download from <https://www.st.com/en/development-tools/stsw-link009.html>.

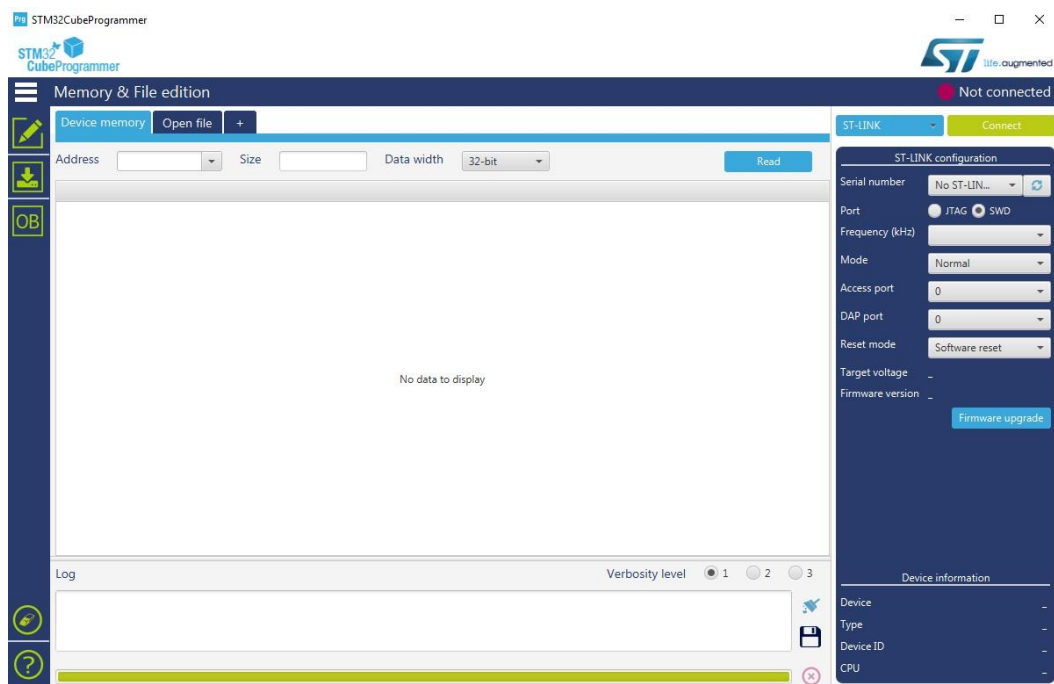
Before connecting the board, install this driver first. This driver provides the debugger interface needed to write the demo firmware binary file to the STM32F7xx MCU’s internal flash memory. It also provides a Virtual COM (serial) port which is needed to write a binary file to the onboard serial flash on the S5U13C00P01C100 board.

### 3.2.2 STM32CubeProgrammer Software

The STM32CubeProgrammer software is available for download from <https://www.st.com/en/development-tools/stm32cubeprog.html>.

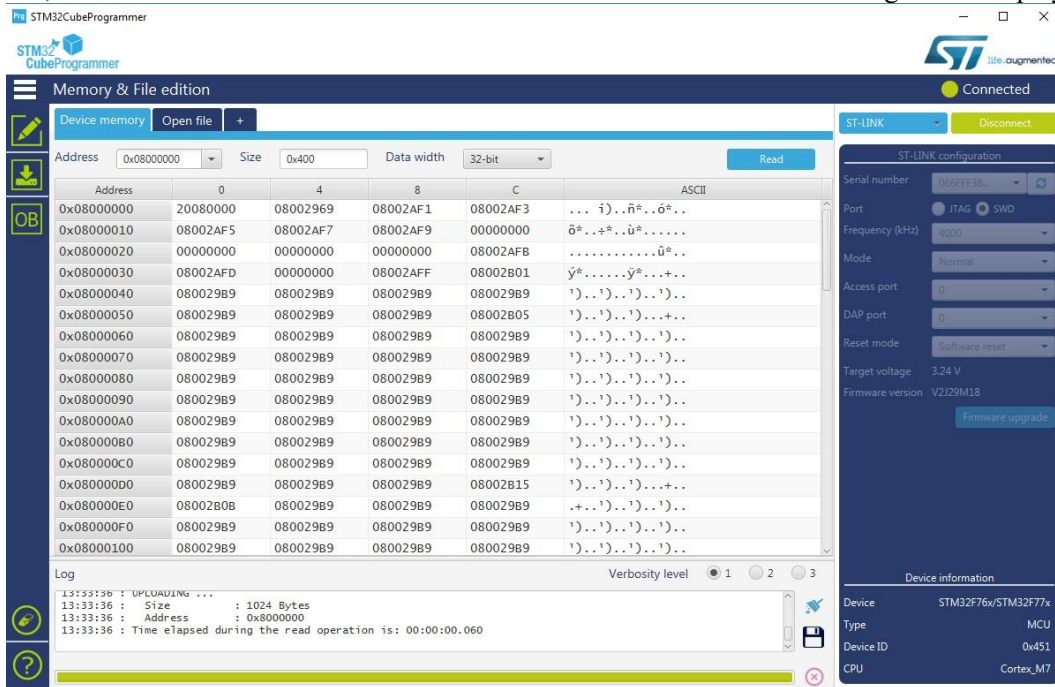
Download and install this program.

After installation, connect the Nucleo board to the PC and run the STM32Cube Programmer software. The following window is displayed:

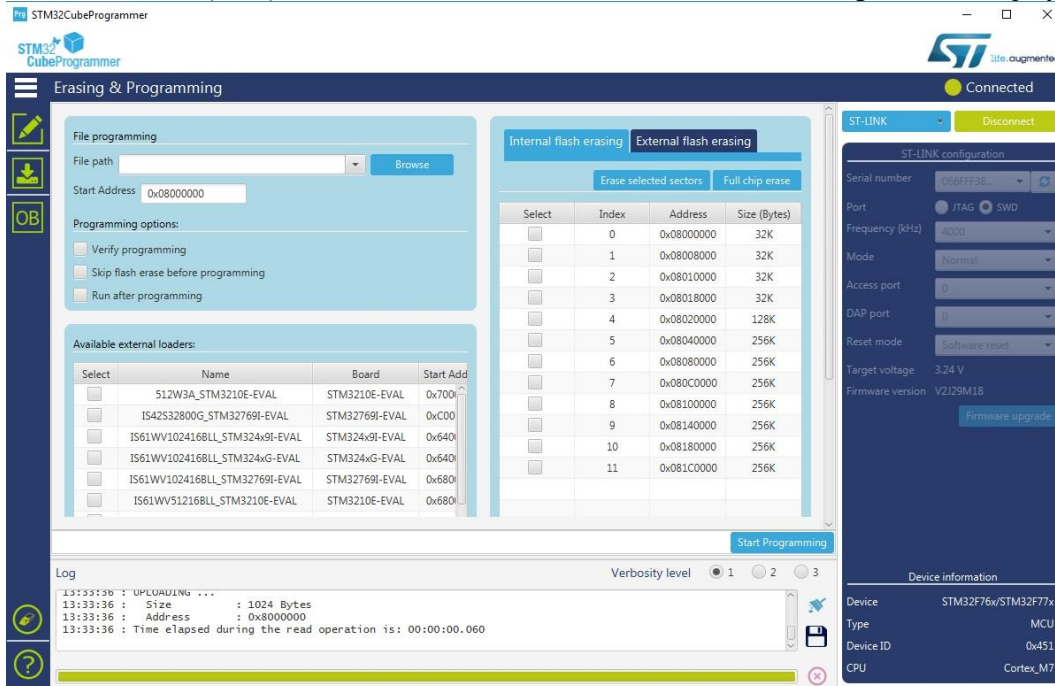


### 3. Demo Firmware Binary Setup

1. Click the “Connect” button to connect through the debugger of the Nucleo board. After successful connection, the “Connect” button icon should turn into “Disconnect” and the following screen displayed:



2. Click the download (  ) icon on the left side of the window. The following screen is displayed:



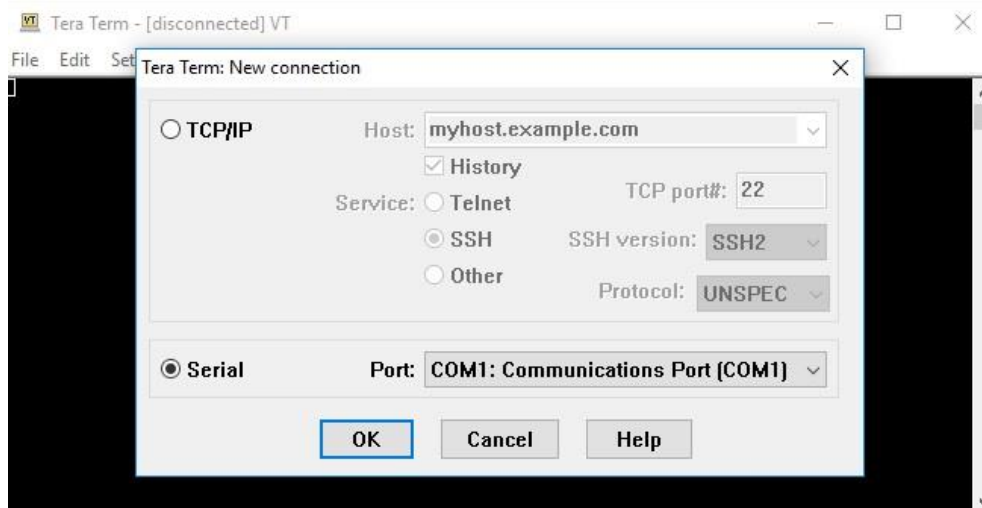
3. Click “Browse”, then find and select either the “demo2\_ET011TT2\_NUCLEO-F746ZG.bin” or “demo2\_ET011TT2\_NUCLEO-F767ZI.bin” demo firmware file depending on the Nucleo board used.
4. Make sure the “Verify programming” and “Run after programming” boxes have check mark beside them.
5. Click “Start Programming” write the binary file to the Nucleo board.

After the STM32CubeProgrammer software has finished programming, it restarts the board and the demo should be running. Section 6 describes the details of the demo items.

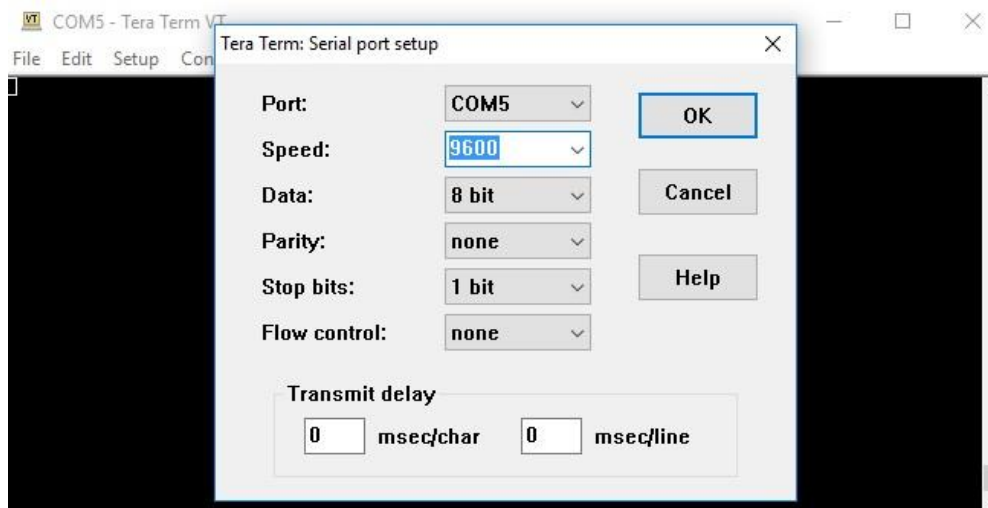
## 4 Terminal Emulation Software and Serial Flash Custom Images

A terminal emulation software, such as TeraTerm, is needed to be able to reprogram the onboard serial flash on the S5U13C00P01C100 board with user images to display in the demo. TeraTerm is available for download from <https://osdn.net/projects/tssh2/releases/>.

Download and install TeraTerm. After installation, run TeraTerm and the following screen is displayed:



1. Click on the “Serial” circle and select a “Port”. The Virtual COM port of the TI or ST board should be one of the “Port” selections if the board is connected.
2. After selecting the Virtual COM port, click “OK”.
3. Click “Setup->Serial port...”. The following screen is displayed:

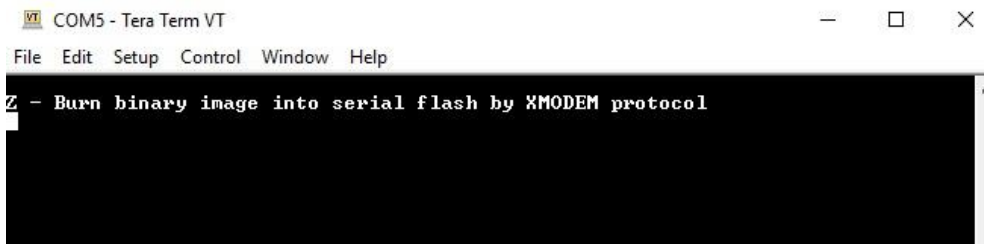


4. For the “Speed”, select “115200” and then click “OK”.

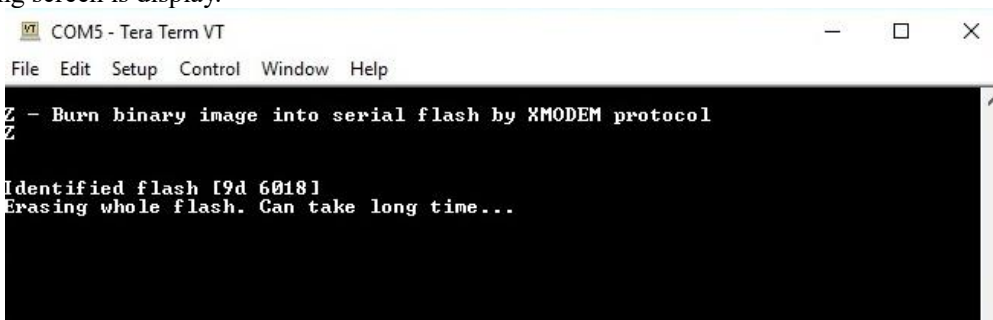
## 4. Terminal Emulation Software and Serial Flash Custom Images

---

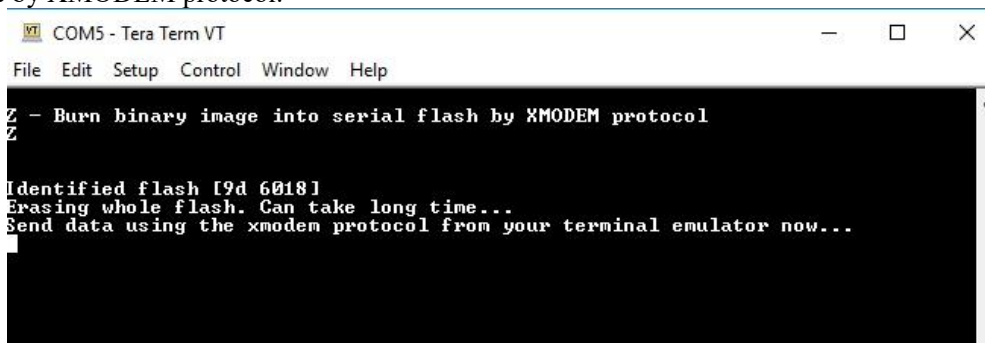
5. Press the reset button on the board to restart the demo firmware. The demo firmware should display the following text on the TeraTerm screen:



6. In the TeraTerm window, press the 'Z' key to initiate writing of a binary image file to the serial flash. The following screen is display.

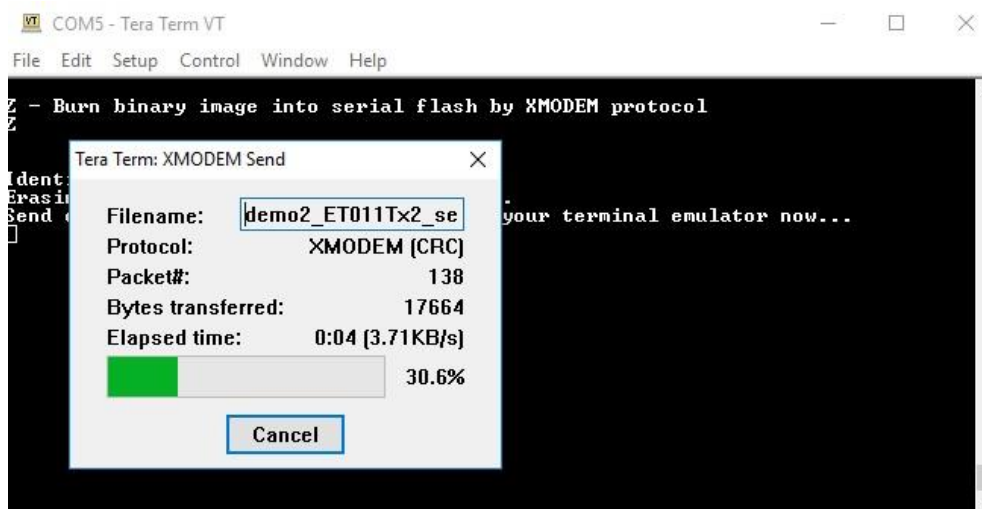


The serial flash is erased first. After erasing, the following message is displayed to ask the user to send a binary file by XMODEM protocol:

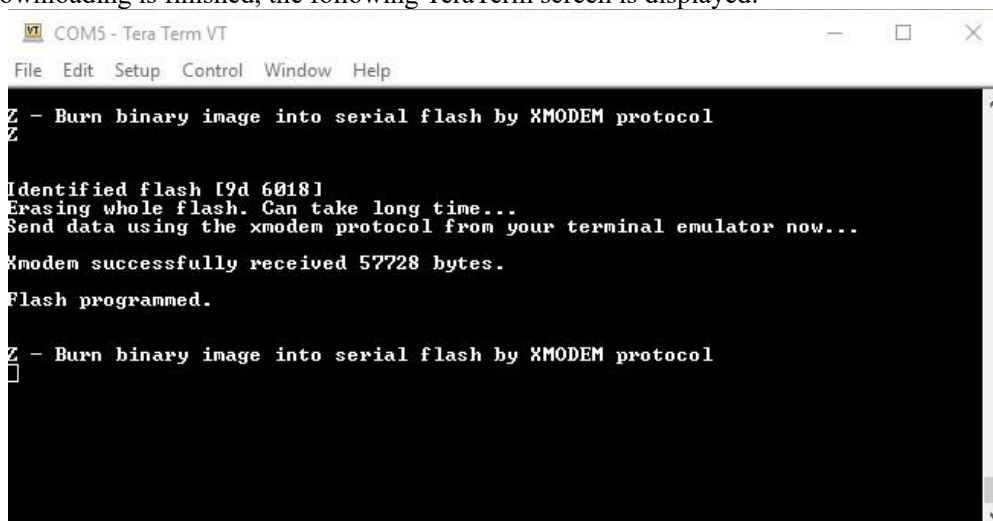


## 4. Terminal Emulation Software and Serial Flash Custom Images

- Click “File->Transfer->XMODEM->Send...”, then find and select the “demo2\_ET011TT2\_serflash.bin” file. When TeraTerm starts downloading the binary file, a pop-up window is displayed showing the progress of the download:



- After downloading is finished, the following TeraTerm screen is displayed:

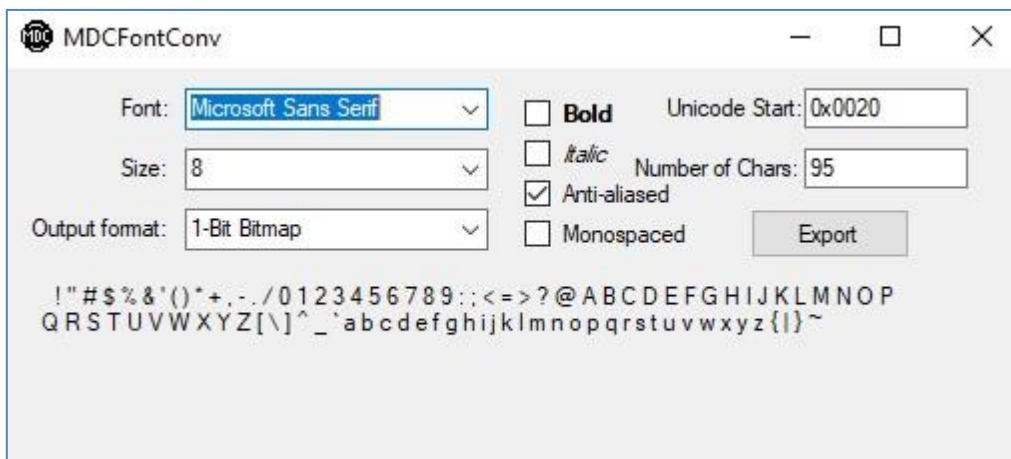


The “demo2\_ET011TT2\_serflash.bin” file contains 4 default images with each image showing the letters A, B, C, and D, respectively. A binary file with 4 custom images can be generated by following the steps in Section 5 and then the new binary file can be downloaded to the serial flash by following steps 6 to 8 above.

## 5 Tools for Custom Image Display

### 5.1.1 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
- 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
- `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.



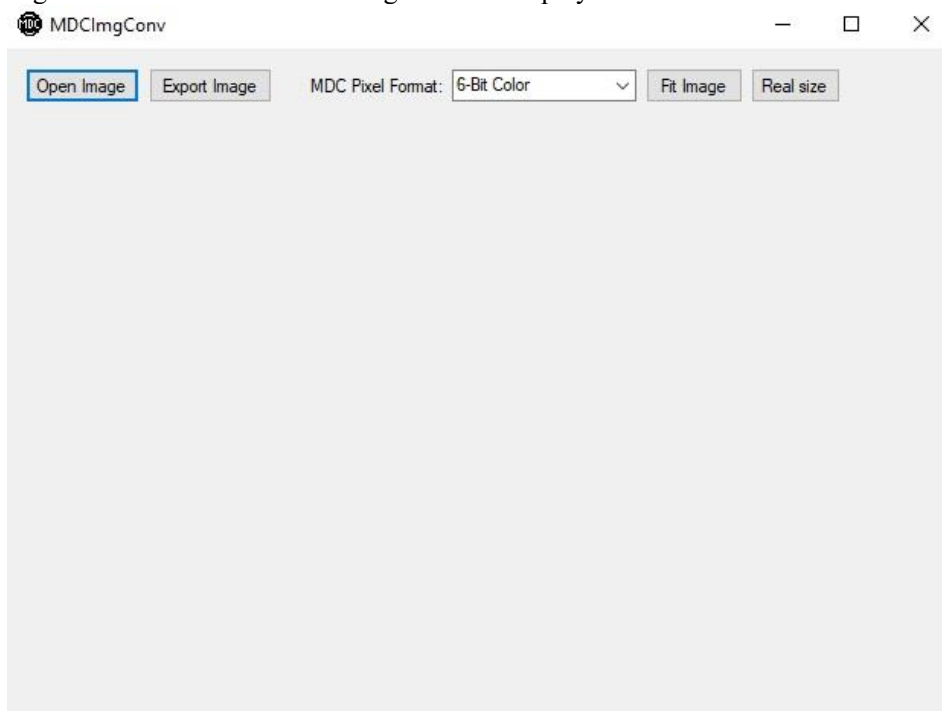
When the board is reset and run with the user button pressed, the demo software goes into a mode which displays 4 user images (240x240, 2-bit grayscale) in a slideshow loop. These 4 images are read from the serial flash. The serial flash can be reprogrammed with a binary file which contains new set of 4 images by following the instruction in Section 4.

There two PC software tools which are used to generate the 4 images and combined them into one binary file for download to the serial flash: MDCImgConv.exe and MDCSerFlashImg.exe.

### 5.2 Image Conversion MDCImgConv.exe

MDCImgConv.exe is a tool for converting any common type of graphic image (BMP, PNG, JPG, ICO, TIF, GIF) to the 2 bits-per-pixel (bpp) grayscale format of the MDC.

Run the MDCImgConv.exe tool. The following screen is displayed:



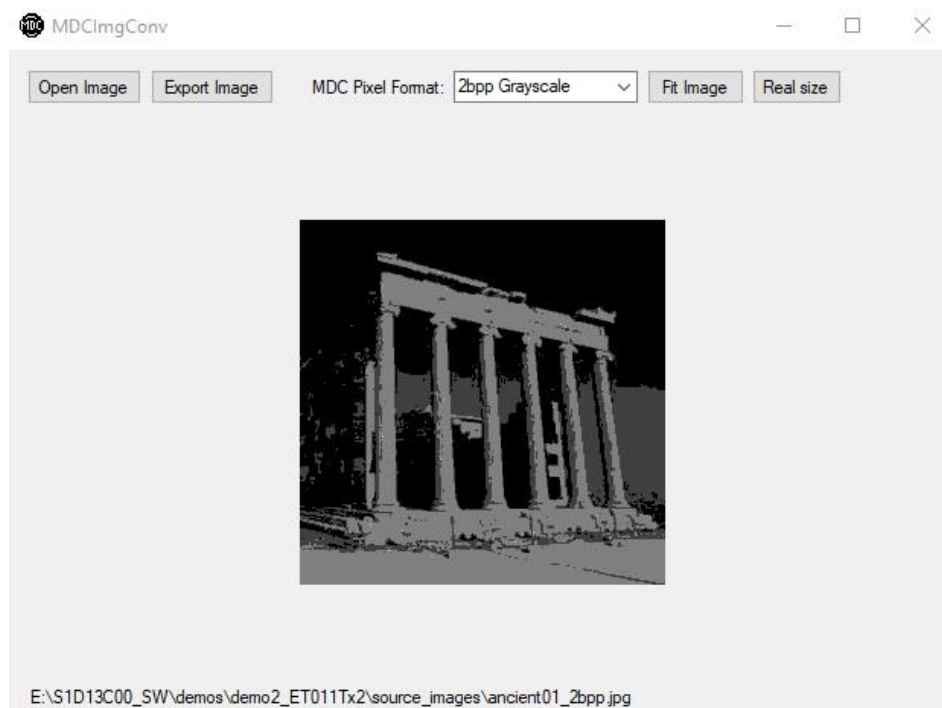
## 5. Tools for Custom Image Display

---

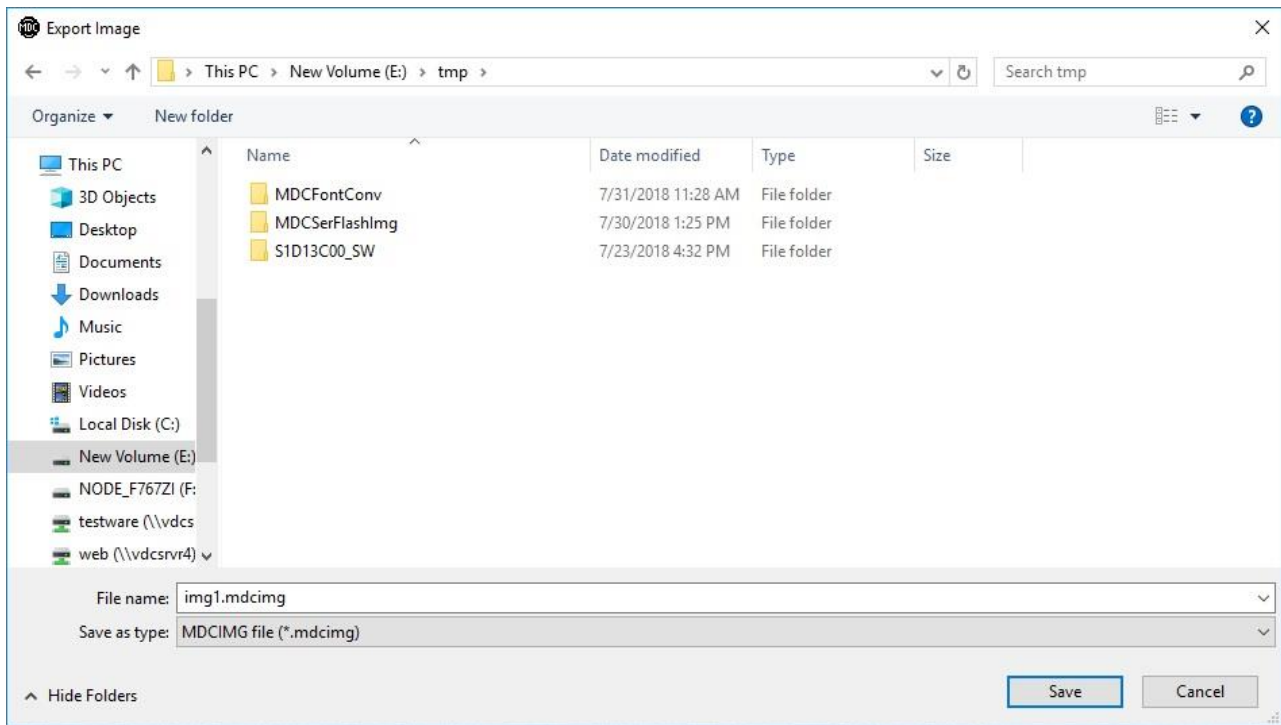
1. Click “Open Image” and select a 240x240 image to convert. After opening the image, the following screen is displayed with an example image:



2. In the “MDC Pixel Format” drop-down, select the “2bpp Grayscale” format. The following screen is displayed:



3. To generate a binary file for the image, click “Export Image” and the following pop-up window is displayed:



4. For the “Save as type”, select “MDCIMG file (\*.mdcimg)”. Enter the desired filename (for example, “img1.mdcimg”).
5. Click “Save” to save the image to a binary file format suitable for the demo software to display.
6. For each of the 4 custom images, repeat steps 1 to 5 to generate a .mdcimg file. For example, generate “img1.mdcimg”, “img2.mdcimg”, “img3.mdcimg”, and “img4.mdcimg”.

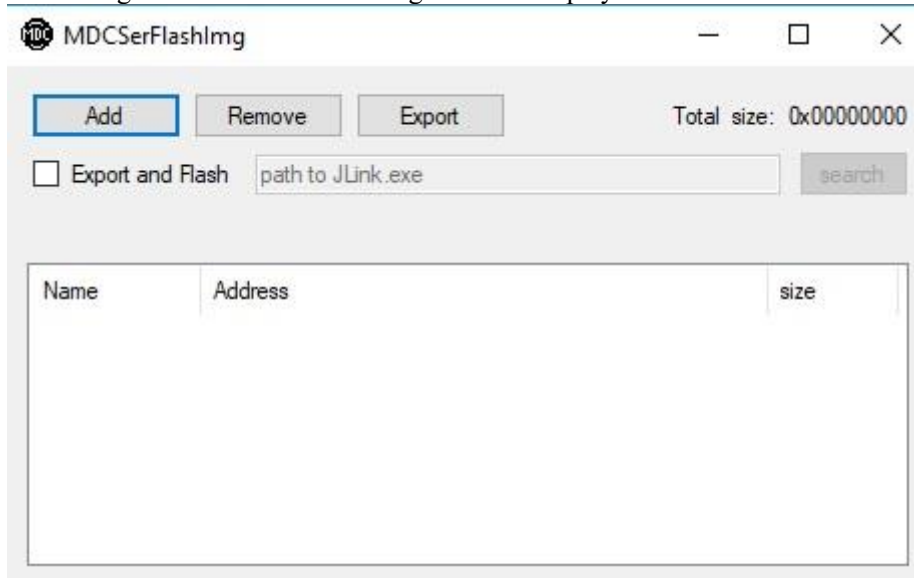
The “MDCSerFlashImg.exe” tool described in the next section is used to combine the .mdcimg images into one binary file for downloading to the serial flash.

## 5. Tools for Custom Image Display

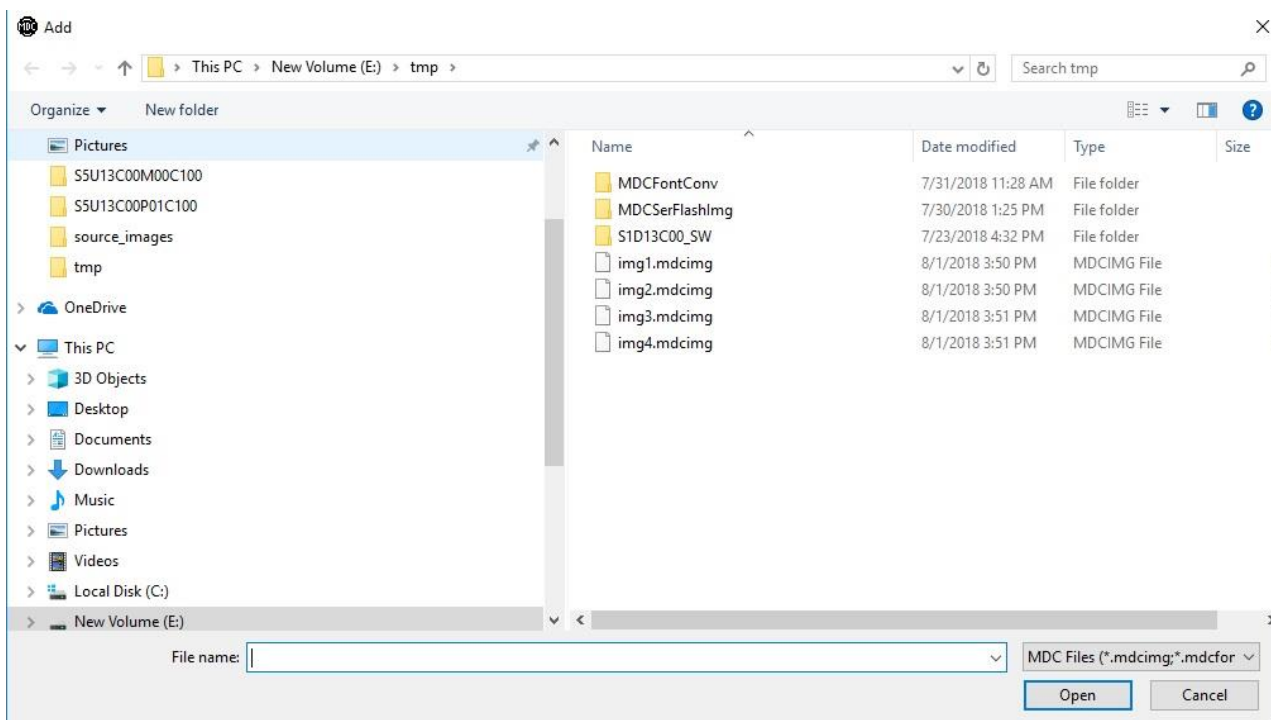
### 5.3 Binary File for Serial Flash MDCSerFlashImg.exe

MDCSerFlashImg.exe is a tool which helps to create a binary image for downloading to the serial flash. The tool accepts binary files generated by MDCImgConv.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 images binaries (.mdcimg files) specified to be included in the single binary image.

Run the MDCSerFlashImg.exe tool. The following screen is displayed:

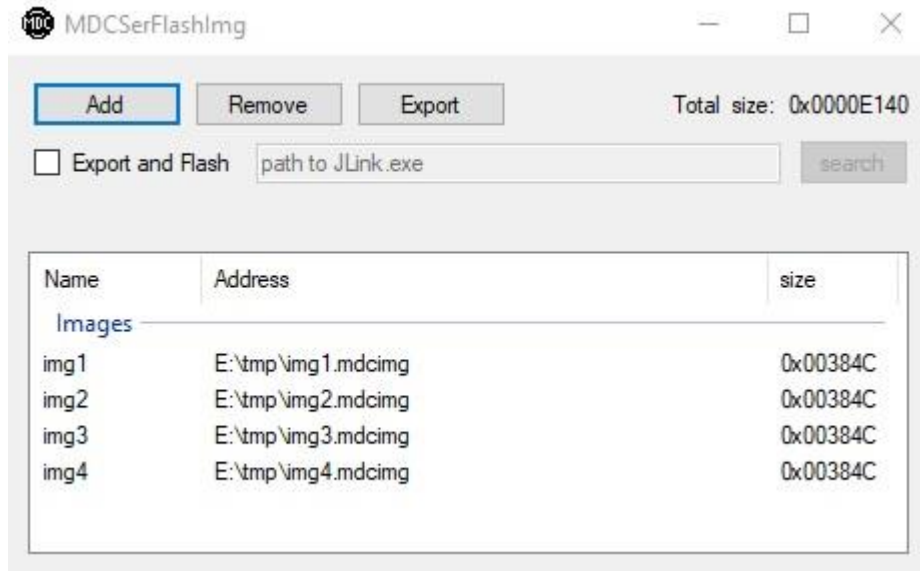


1. Click “Add” to select the .mdcimg files to include in the output binary file. The following pop-up window is displayed.



2. Select the 4 .mdcimg files (for example, “img1.mdcimg” to “img4.mdcimg”) and click “Open”.

3. After selecting the .mdcimg files, the following screen is displayed:



4. Click "Export", enter the desired output filename, and then click "Save" to save generate the binary file for downloading to the serial flash.

## 6. Demo Firmware Source Code Setup

---

### 6 Demo Firmware Source Code Setup

This section describes installations and setup needed if the user receives a source code package and would like to modify and rebuild the demo software.

#### 6.1 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.

##### 6.1.1 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.

##### 6.1.2 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.

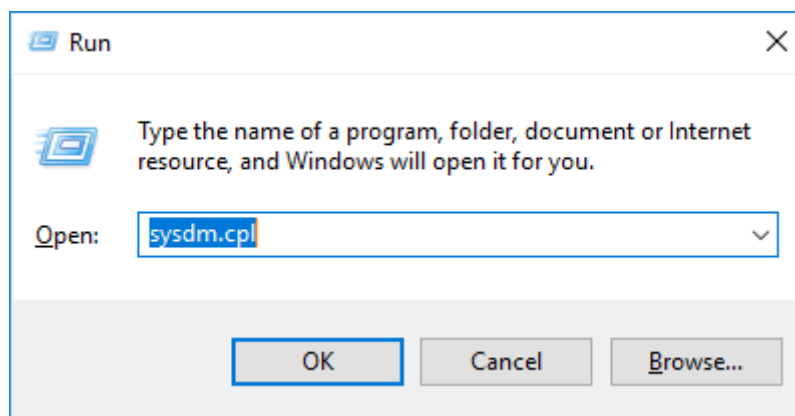
Information and downloads for the TivaWare for C series can be found here: <http://www.ti.com/tool/SW-TM4C>

**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.

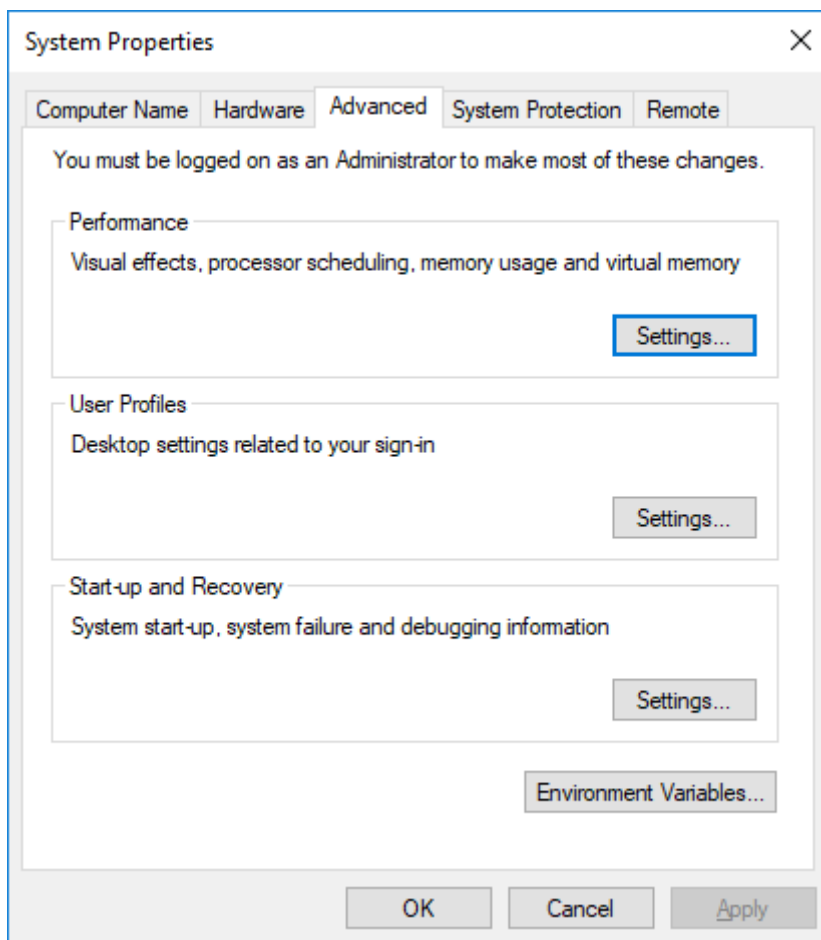
The procedure to create the environment variable under Windows 10 is as follows;

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.

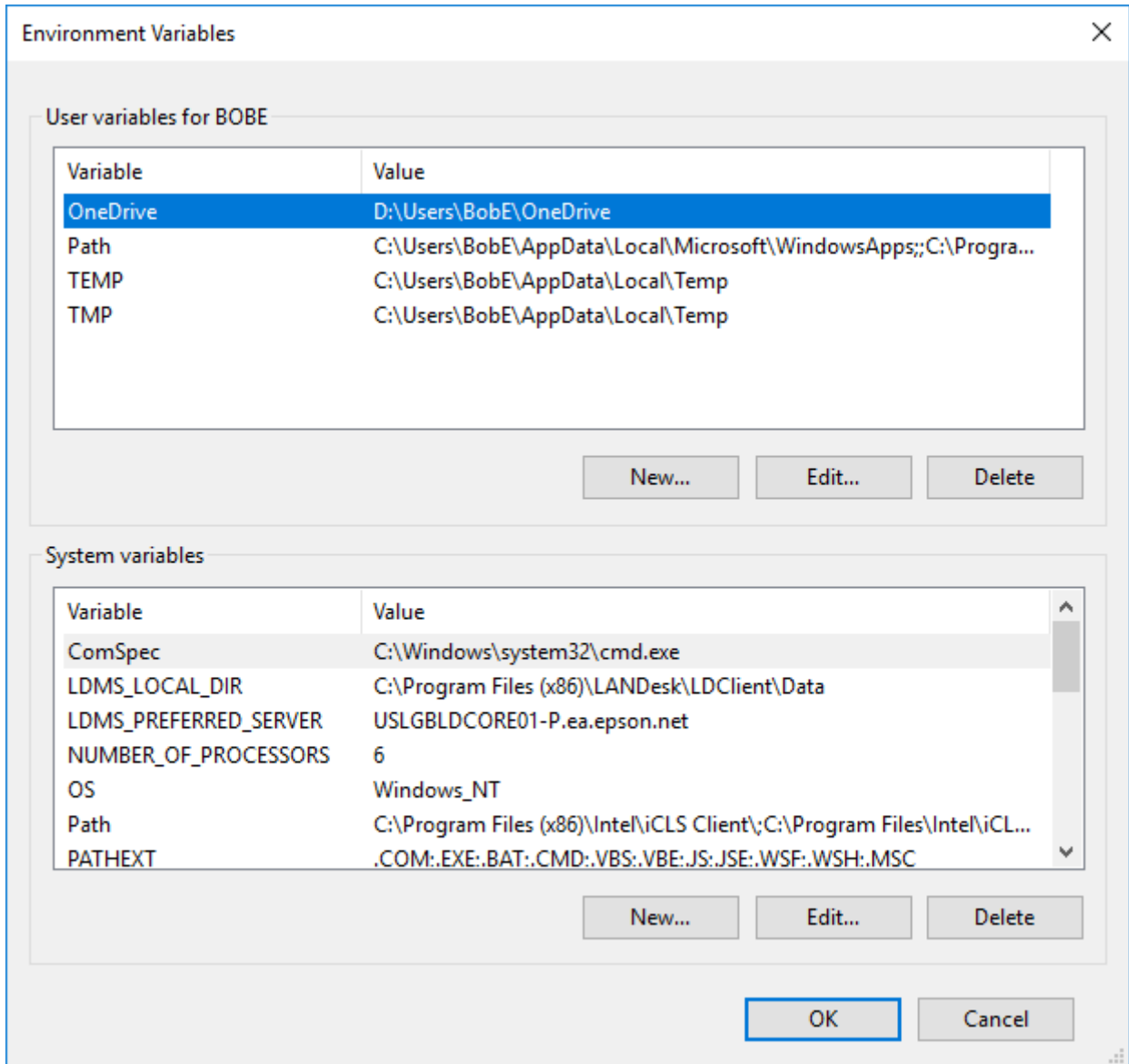


Select the Advanced tab and then click the “Environmental Variables...” button.



## 6. Demo Firmware Source Code Setup

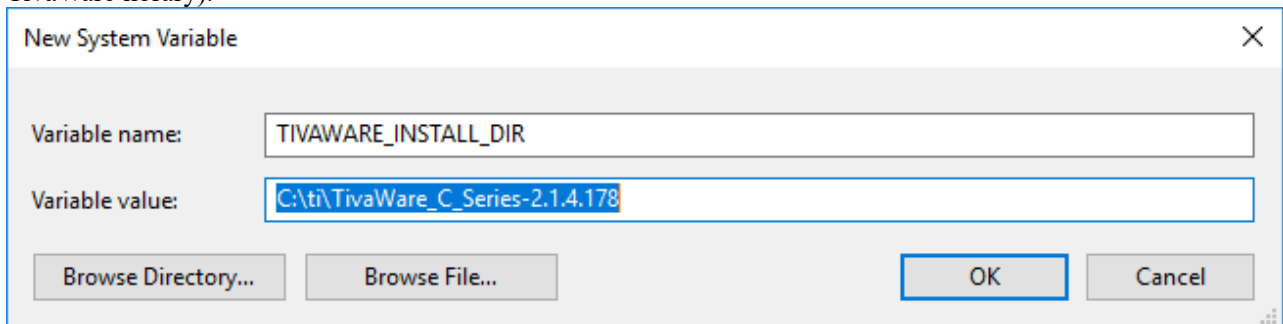
Click the “New...” button in the System variables section to add a new environment variable.



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).





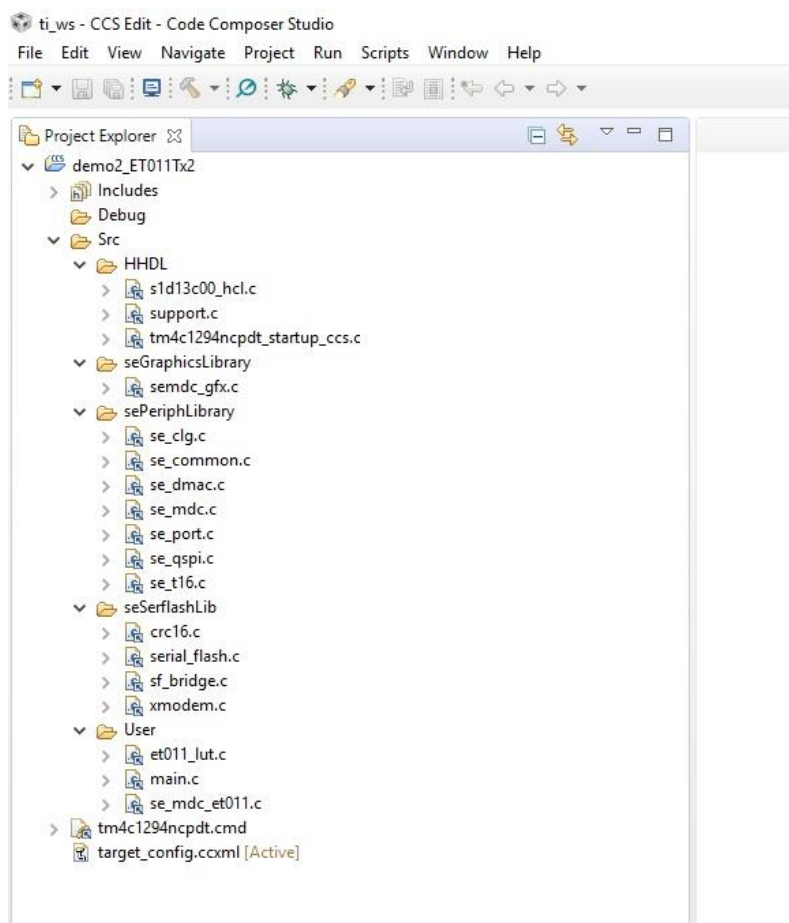
### 6.1.3 Demo Software Source Code Package

The demo software source code package is a ZIP file: demo2\_ET011TT2.zip.

Extract the ZIP file into a folder. After extraction, 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 demo software project is in the “S1D13C00\_SW\demos” subfolder of the folder where the ZIP file package was extracted. To open the demo project, 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\demos\demo2\_ET011TT2\HOSTS\EK-TM4C1294XL\ccs” folder.

After selecting the project folder, click “Finish” in the “Import Projects from File System or Archive” window to open the project. The project name 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:



### 6.1.4 Software Build and Target Download

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

After a successful build, click the 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.



## 6. Demo Firmware Source Code Setup

---

### 6.1.5 DEBUG\_PRINT Macro Symbol

The demo project provided has 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.

## 6.2 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.

### 6.2.1 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.

### 6.2.2 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](#)

### 6.2.3 Demo Software Source Code Package

The demo software source code package is a ZIP file: demo2\_ET011TT2.zip.

For the sample software projects to build correctly in the System Workbench and STM32CubeF7 environment, the demo software ZIP package must be extracted into “<STM32CubeF7 location>\Projects”, where <STM32CubeF7 location> is the root folder where the STM32CubeF7 was installed (for example, “C:\STM32Cube\_FW\_F7\_V1.11.0”).

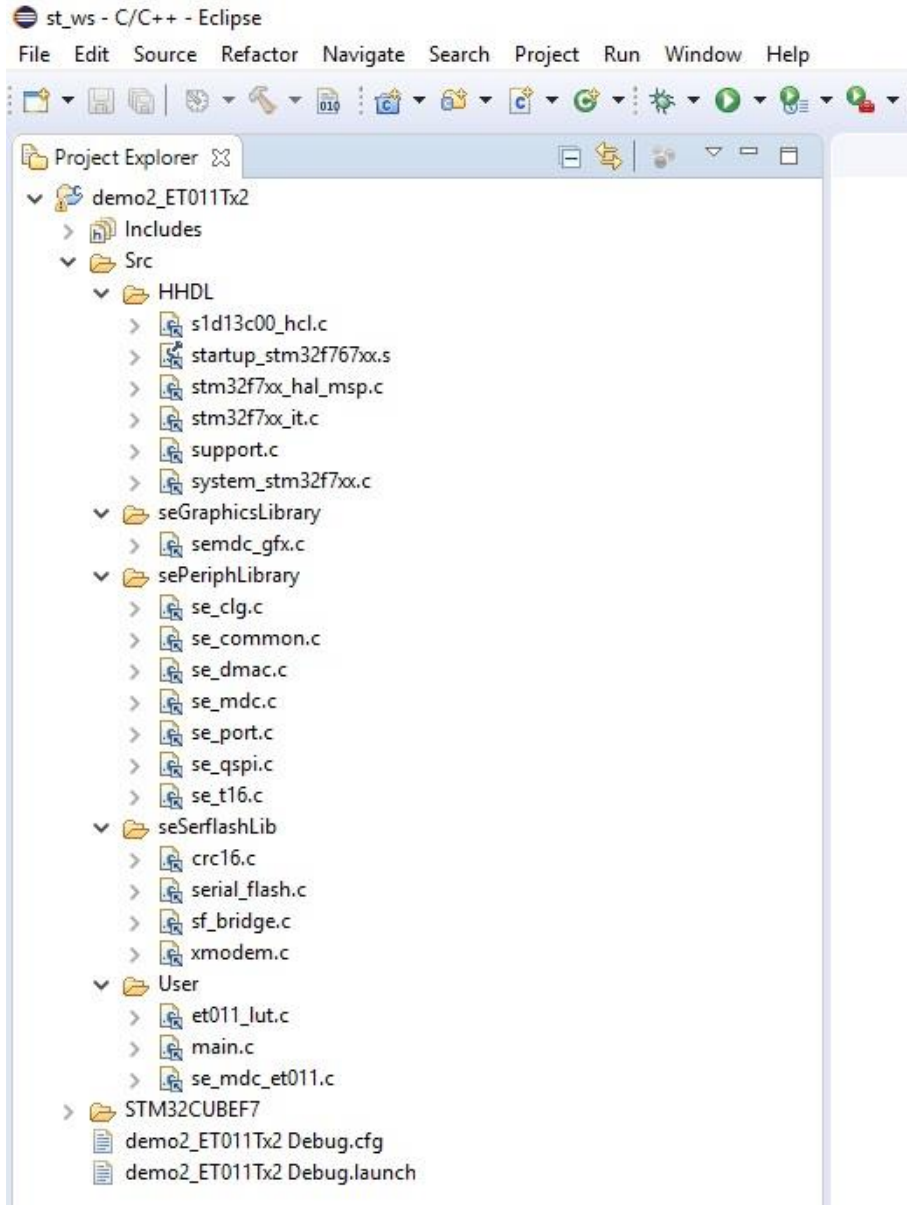
After extraction, run the System Workbench for STM32 IDE. When the Eclipse Launcher prompts for a workspace directory, browse to an existing workspace or enter a new workspace directory location, then click OK to open the workspace.

The demo project is in the “S1D13C00\_SW\demos” subfolder of the folder where the demo software package was extracted. To open the demo project, 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\demos\demo2\_ET011TT2\HOSTS\<NUCLEOBoard>\SW4STM32” folder, where <NUCLEOBoard> is either NUCLEO-F746ZG or NUCLEO-F767ZI.

## 6. Demo Firmware Source Code Setup

---

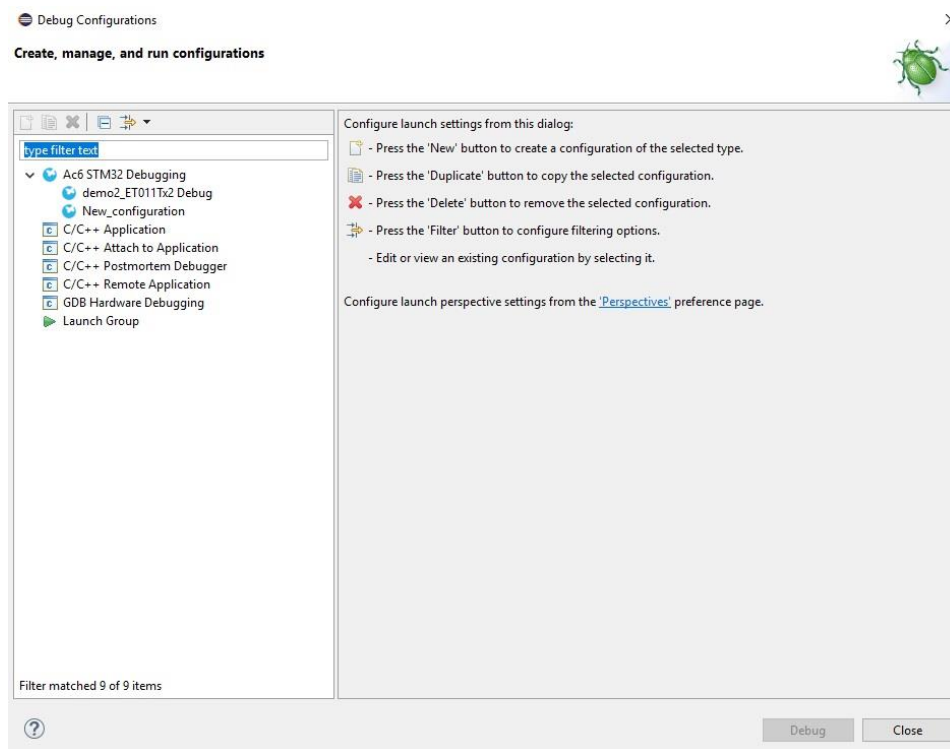
After selecting the project folder, click “Finish” in the “Import Projects from File System or Archive” window to open the project. The project name will be displayed in the Project Explorer window of SW4STM32 (Eclipse). Expand the project by clicking on the ‘>’ arrow beside it. Sub-items can also be 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:



## 6.2.4 Software Build and Target Download

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

After a successful build, click the 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 “demo2\_ET011TT2 Debug”. Click “Debug” to download, run, and debug the firmware on the target board.

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

## 6.2.5 DEBUG\_PRINT Macro Symbol

The demo project provided has 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”.)

Click “OK” and then rebuild the project.

## 6. Demo Firmware Source Code Setup

### 6.3 Demo Software Components

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

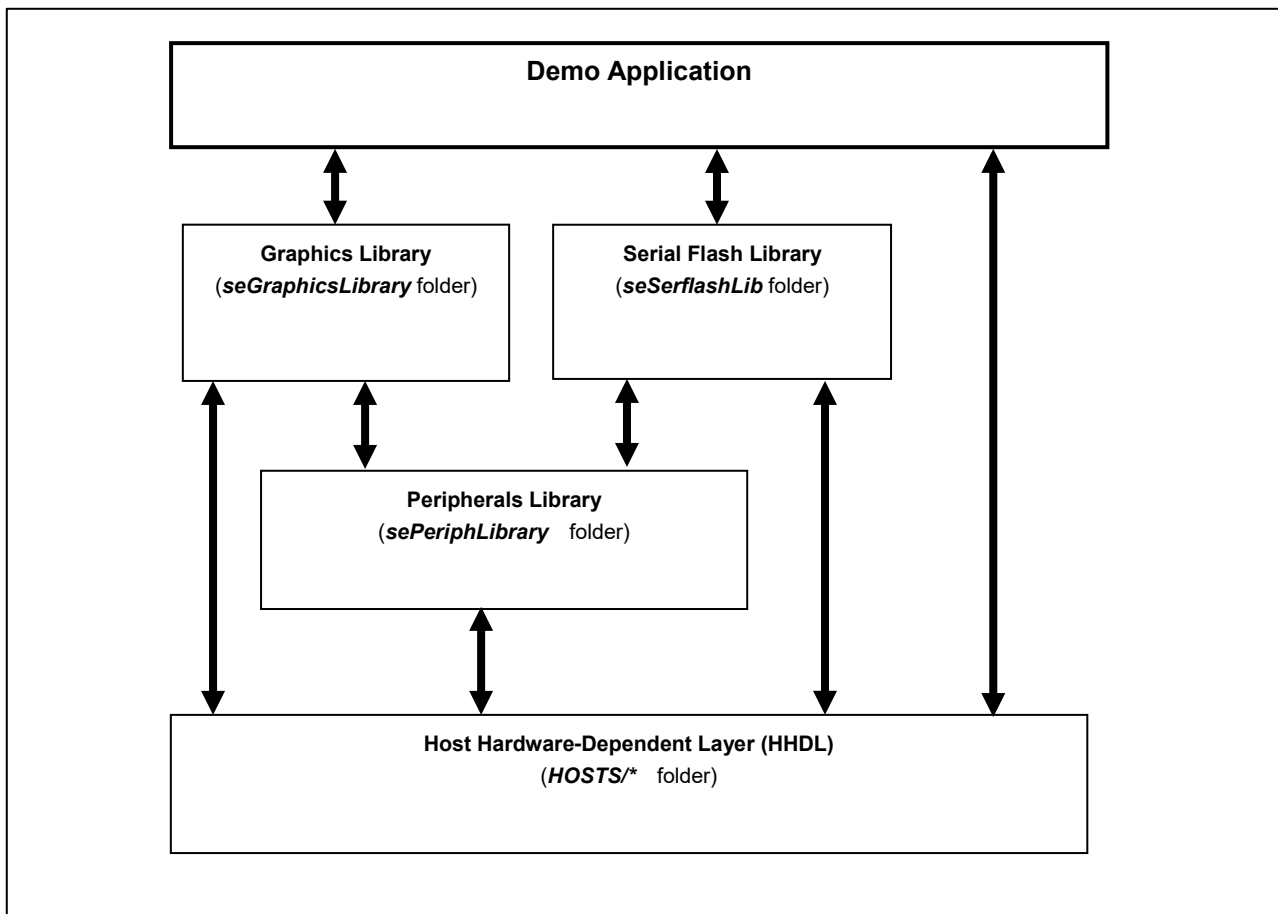


Figure 6.1 S1D13C00 Demo Software Structure

#### 6.3.1 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.\*”.



## 6. Demo Firmware Source Code Setup

---

### 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.



- **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.

#### 6.3.2 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).

## 6. Demo Firmware Source Code Setup

---

### 6.3.3 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”.

### 6.3.4 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).

### 6.3.5 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.

## 6.4 eink2C.exe Conversion Program

The “eink2C” subfolder in the demo software project folder contains a file called “et011\_lutdata.h” which contains the waveforms for the EPD panel. The waveform data is proprietary and so the default values of the tables in “et011\_lutdata.h” default to 0x00. The user can request and obtain a waveform file (.eink suffix) from E Ink and generate “et011\_lutdata.h” with proper waveform values.

The “eink2C.exe” program in the “eink2C” folder is used to convert a .eink file into “et011\_lutdata.h”. After conversion, the demo software can be built and loaded into the target MCU board.

## 7 Details of Demo Software

The demo software is controlled by a user button. On the TI EK-TM4C1294XL board, the user button is USR\_SW1. On the ST STM32 Nucleo-144 development board, the user button is the blue-colored button.

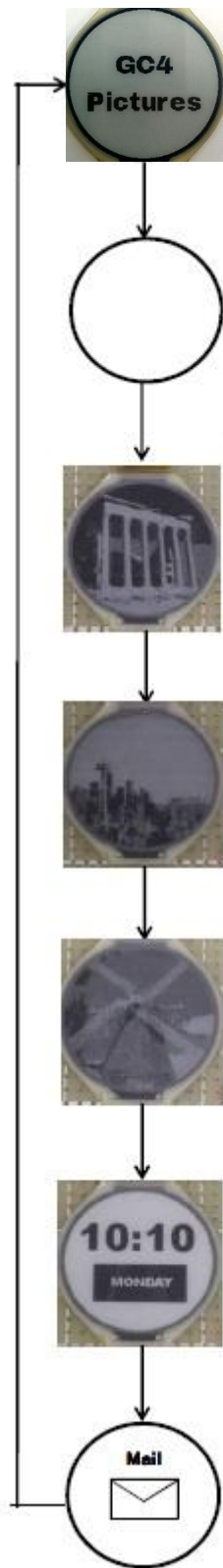
There are 5 demo loops, each showing a different set of images.

- GC4 Pictures – display images using GC4 waveform
- GU4 Graphics – display graphic (not photo) images using GU4 waveform,
- A2 Scroll – display scrolling text images using A2 waveform
- GU4 White – display analog and digital clocks by Event Processor using GU4 waveform
- GC4 User – display 4 images from serial flash

The 5 demo loops are grouped into a set of the first 4 demo loops and the “GC4 User” demo loop. On reset or power-up, if the user button is held down (pressed), the software will run the “GC4 User” demo loop showing 4 pictures from serial flash. If the user button is not pressed during power-up or reset, the software runs the first demo loop (GC4 Pictures) of the group of 4.

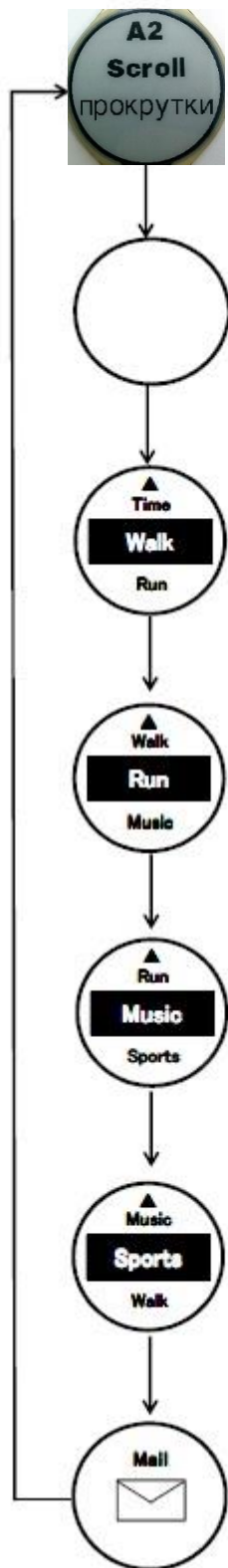
When the “GC4 User” demo loop is running, pressing the user button will cause the software to switch to the group of 4 demo loops. When a demo loop in the group of 4 is running, pressing the user button will cause the software to switch to the next demo loop in the group in a round-robin fashion.

7.1 GC4 Pictures

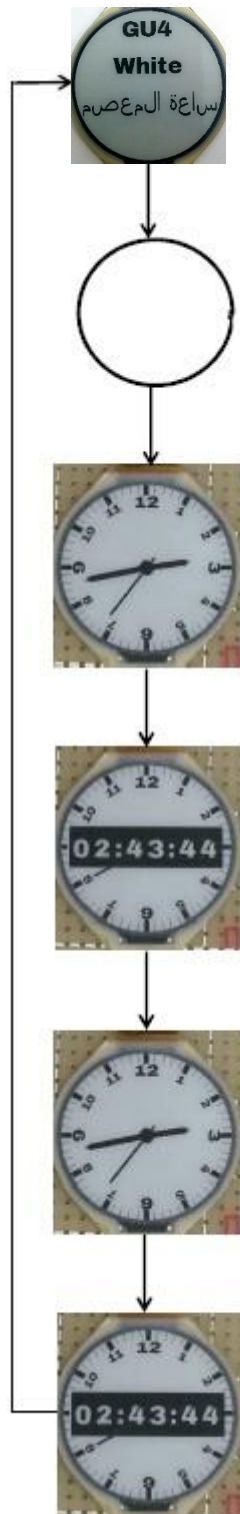




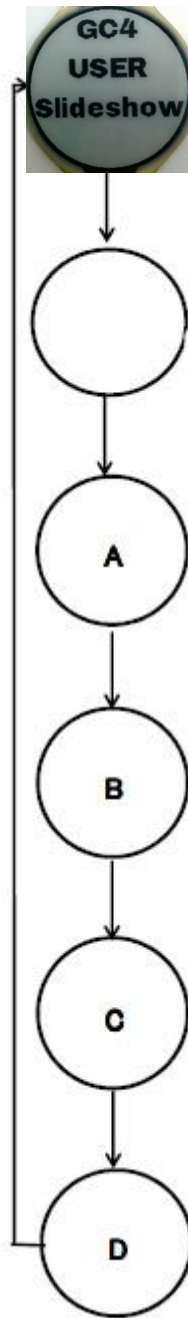
7.3 A2 Scroll



7.4 GU4 White



7.5 GC4 User





---

## 8 Revision History

Attachment-1

<b>Rev. No.</b>	<b>Date</b>	<b>Page</b>	<b>Category</b>	<b>Contents</b>
1.0	23/01/2019			Released as Rev 1.0

---

For more information on the S1C13C00 and other Epson Display Controllers, visit the Epson Global website.

[https://global.epson.com/products\\_and\\_drivers/semicon/products/display\\_controllers/](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](https://global.epson.com/products_and_drivers/semicon/information/support.html)



Document Code: XB8A-G-001-01.0  
First Issue August 2018  
Revised January 2019