ECE 153B – Winter 2022
Sensor and Peripheral Interface Design

Getting Started

Contents

Chapter 1

Installation of Keil MDK

Warning: Do not connect the Nucleo board into your PC/laptop before all software and driver installation is complete. If you connect your kit to your PC/laptop before the software installation is complete, Windows often mistakenly associates a wrong USB driver to the kit. As a result, you will not be able to program the kit. If this happens, the solution is to change the USB driver to the ST-Link USB driver by going into the Control Panel.

Go through each of the following steps to complete installation of the required software. Note that the lab computers already have Keil installed. In addition, Keil works on Windows only. For MAC users, you can use the lab computers to do the labs or use Parallels to run Windows or use NoMachine (Chapter 4).

1.1 Install Keil MDK-ARM

Download the latest free evaluation version of Keil MDK-ARM using this link. Keil MDK-ARM contains μVision 4 Integrated Development Environment (IDE) with a debugger, a flash programmer, and the ARM compiler toolchain.

Run the downloaded file `MDK5xx.exe`. Note that the software takes up 2 GB of disk space, so you can install the software in a different drive if you do not have enough space in the default (C:) drive.

After the core software is installed you will see a window prompting you to install packs. Click **OK** to proceed with installing the packs. You should now see a window identical to the one shown in Figure 1. Under the **Devices** tab, select device `STM32L476RGTx` for Nucleo board (you can use the search bar to find it quickly). Under the **Packs** tab, find pack `Keil::STM32L4xx_DFP` and click **Install**. The installation status can be seen at the bottom of the window.

1.2 Install ST-Link USB Driver

Navigate to the directory `<Installation Location>\ARM\STLink\USBDriver` and run the batch file `stlink_winusb_install.bat` as administrator. In the window that pops up (see Figure 2), click **Next** to install the drivers.

Now, you can connect the Nucleo board to your PC/laptop using a USB Type A to Mini B cable. Upon connection, the Nucleo board should be recognized as “STMicroelectronics STLink dongle".

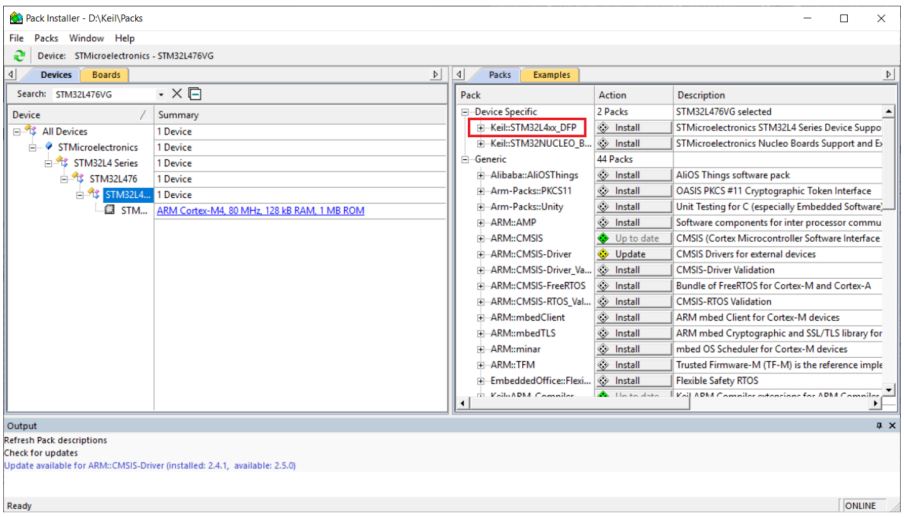


Figure 1: Pack Installation Window

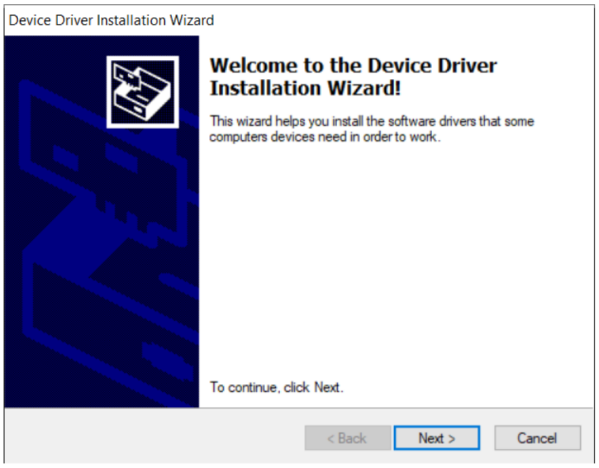


Figure 2: Device Driver Installation Wizard

Figure 3: USB Type A to Mini B

1.3 Install STM32 ST-Link Utility

Download the ST-Link Utility using this link. Unzip the downloaded file and run the executable to install the ST-Link Utility to your PC/laptop.

Typically, we will use Keil to program the Nucleo board. However, the ST-Link Utility can be used to reprogram the flash memory should you make mistakes in programming the debug/program pins of the STM32 processor.

1.4 Common Issues

- **Firmware Upgrade**

  When you download the binary code to the Nucleo board, sometimes a window pops up asking “Old ST-Link firmware detected. Do you want to upgrade it?". If you see this message, select **Yes** and click **Device Connect**. If you get an error “ST-Link is not in the DFU mode", unplug the USB cable from the board, reconnect the cable, and try clicking **Device Connect** again. If the connection is successful, click **Yes** to upgrade the firmware.

  Another way to update the firmware is to use the ST-Link Utility. Under **ST-LINK** in the menu bar, click **Firmware update** (see Figure 4).

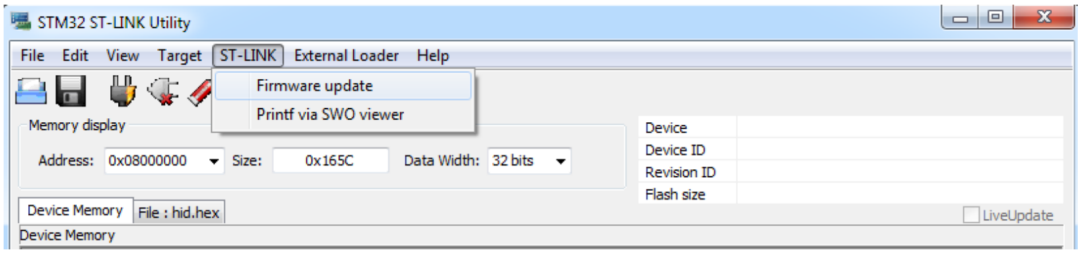


Figure 4: Firm Update Using ST-Link Utility

- **“Target Not Found"**

  When you use the Nucleo board for the very first time, you might not be able to program it in Keil and might receive an error saying “Target not found" when downloading the code to your board. This is because the demo program quickly puts the microprocessor into a very low power mode after a reset. There are several ways to solve this issue – below are the two simplest ways. The error should go away permanently afterwards.

  - In Keil, click the **Options for Target** button (see Figure 5), move to the **Debug** tab, and press the **Settings** button at the top-right of the window (see Figure 6) to pop up another window with additional setting options. For the **Connect** option under the **Debug** → **Connect & Reset Options** section, select **with Pre-reset** (see Figure 7).

Figure 5: Options for Target Button

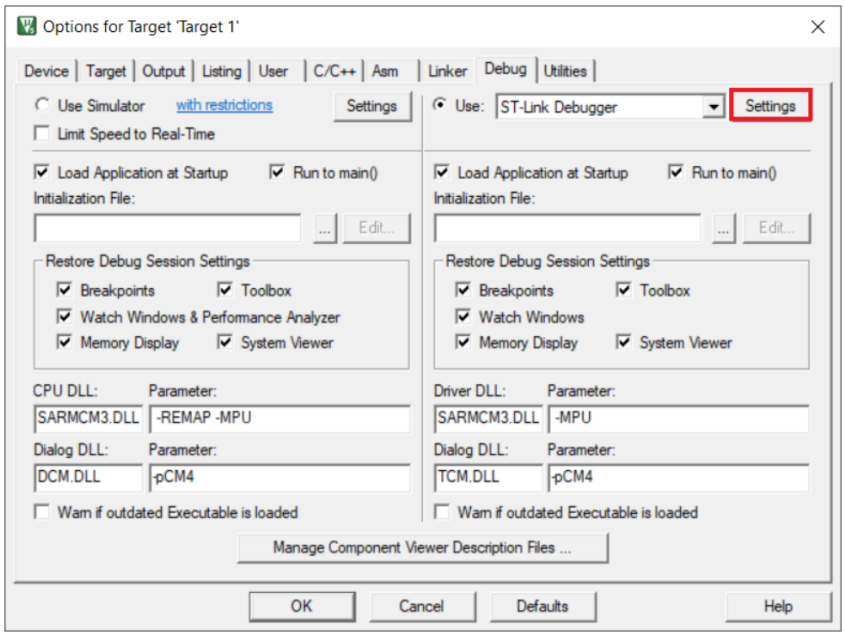


Figure 6

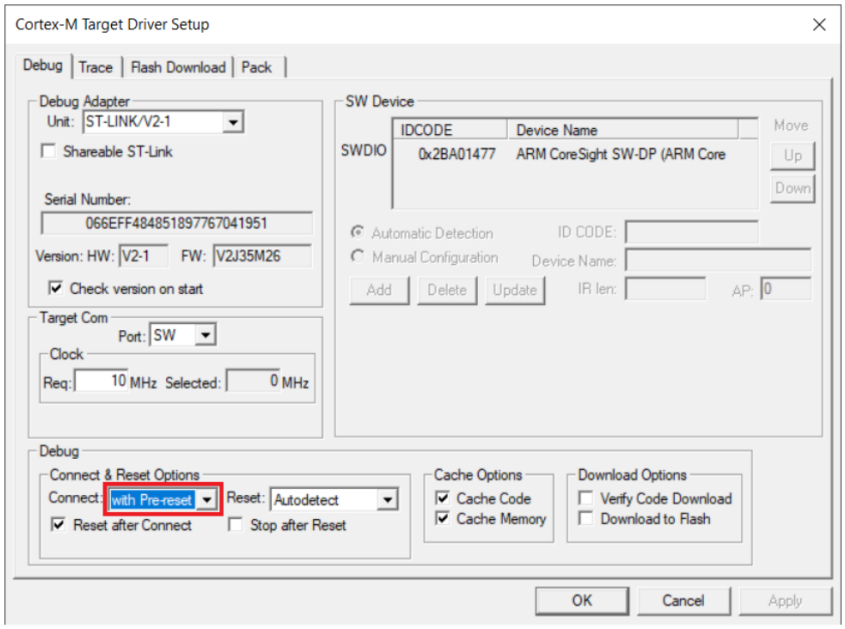


Figure 7

If the problem returns, try some of the following different methods to solve the issue.

- Make sure all jumpers are correctly connected.
- Use the ST-Link Utility to erase the chip. Hold down the reset button before the USB cable is plugged in. When the reset button is released, immediately select **Target** → **Erase Chip** in ST-Link Utility. Several attempts might be needed to get the timing correct.
- Reinstall the USB drive.

- **No ST-Link Detected**

  First, make sure that the device driver is correctly installed. If you are not sure, reinstall the USB device driver: go to directory `<Installation Location>\ARM\STLink\USBDriver` and run the batch file `stlink_winusb_install.bat` as administrator.

  On the board, the alternative function of **PA13** and **PA14** should be ST-Link `SWDIO` and `SWCLK`, respectively. If your code accidentally changed the mode or the alternative function of these two pins, you can no longer program the Nucleo board. To fix this issue, erase the bad code stored on the board using ST-Link Utility.

  1. Open ST-Link Utility. Click **Target** on the menu bar and click **Settings**.
  2. Select the **Connect Under Reset** option from the drop-down menu under **Connection settings** → **Mode** (see Figure 8).

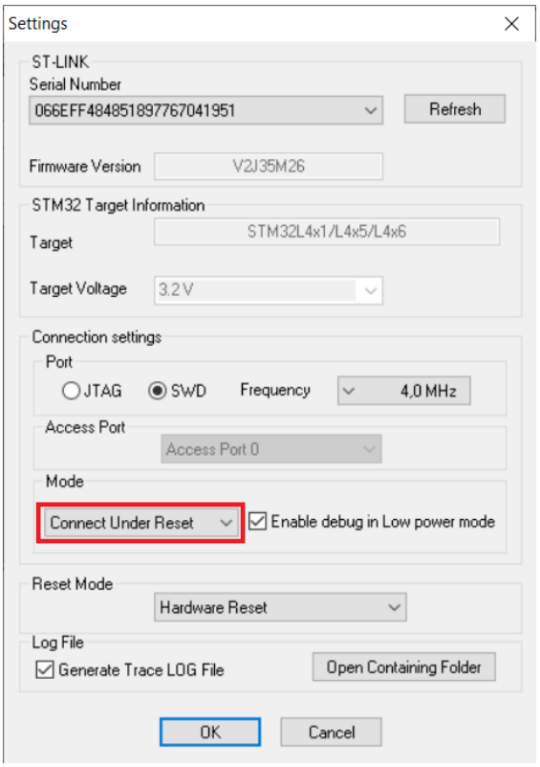


Figure 8

  3. From the home screen of ST-Link Utility, click **Target** → **Connect**.
  4. Click **Target** → **Erase Chip**.
  5. Click **Target** → **Disconnect**.

- **Flash Download Failed**

  When you program the Nucleo board, you may get an error saying “Flash download failed". Go into **Options for Target**, click the **Debug** tab, and click the **Settings** button on the top-right of the window (see Figure 6). In the new window that pops up, click the **Flash Download** tab and ensure that under the **Programming Algorithm** section, the option `STM32L4xx 1MB Flash` is selected (see Figure 14). If the option is not present, click the **Add** button to add the option.

- **ST-Link Connection Error**

  When you program the board, you might receive an error saying “ST-Link connection error". To fix this issue, make sure that your board is not connected to another software. For example, your link might be connected to the ST-Link Utility, which means that your board will not be able to connect to Keil.

- **No Target Connected**

  When you program the board, you might receive an error saying “No target connected".Use the following steps to fix this issue.

  1. Make sure that the USB cable is well connected.
  2. Make sure that all jumpers on the board are on the correct positions. Refer to the *User Manual* for the correct jumper locations.

Chapter 2

Keil Project Creation

Go through each of the following steps to create a project in Keil IDE.
Note: If you already have created a project, you can quickly create a new project by moving the project file (`*.uvprojx`) to a new directory, renaming the file, and replacing the necessary files. However, you may need to reconfigure the options for **Debug** (see the third bullet in Step 5 below).

1. From the menu bar, select **Project** → **New µVision Project**. Select a folder for your project.

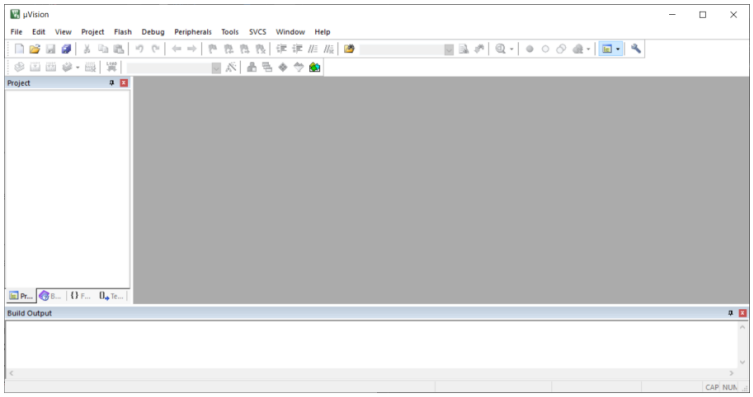


Figure 1: Keil IDE

2. In the **Select Device** window, use the search bar and select the device `STM32L476RGTx` for Nucleo board (see Figure 2). If you do not see the target processor in the list, make sure that you have the `Keil::STM32L4xx_DFP` pack installed. You can do this by clicking the **Pack Installer** button on the main window of the IDE (see Figure 3 for the button image).

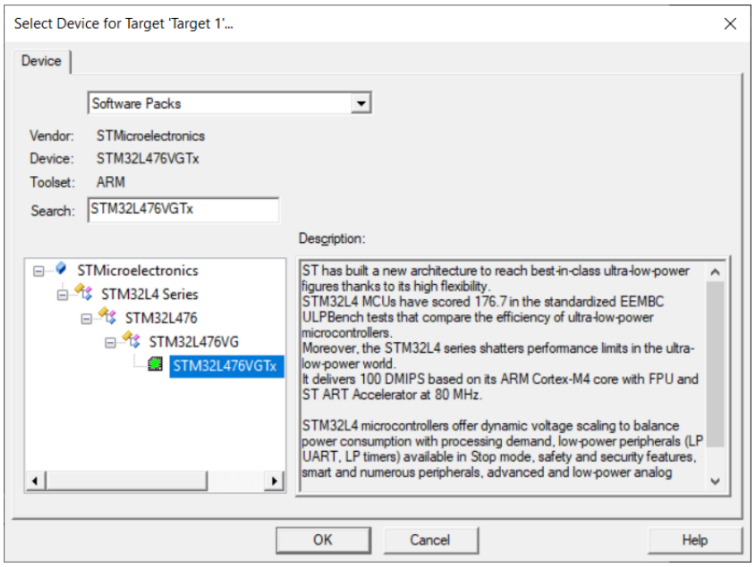


Figure 2: Select Device

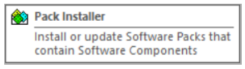


Figure 3: Pack Installer

3. In the **Manage Run-Time Environment** window, ensure that

   - **CMSIS** → **CORE** is selected and
   - **Device** → **Startup** is *not* selected

   (see Figure 4).

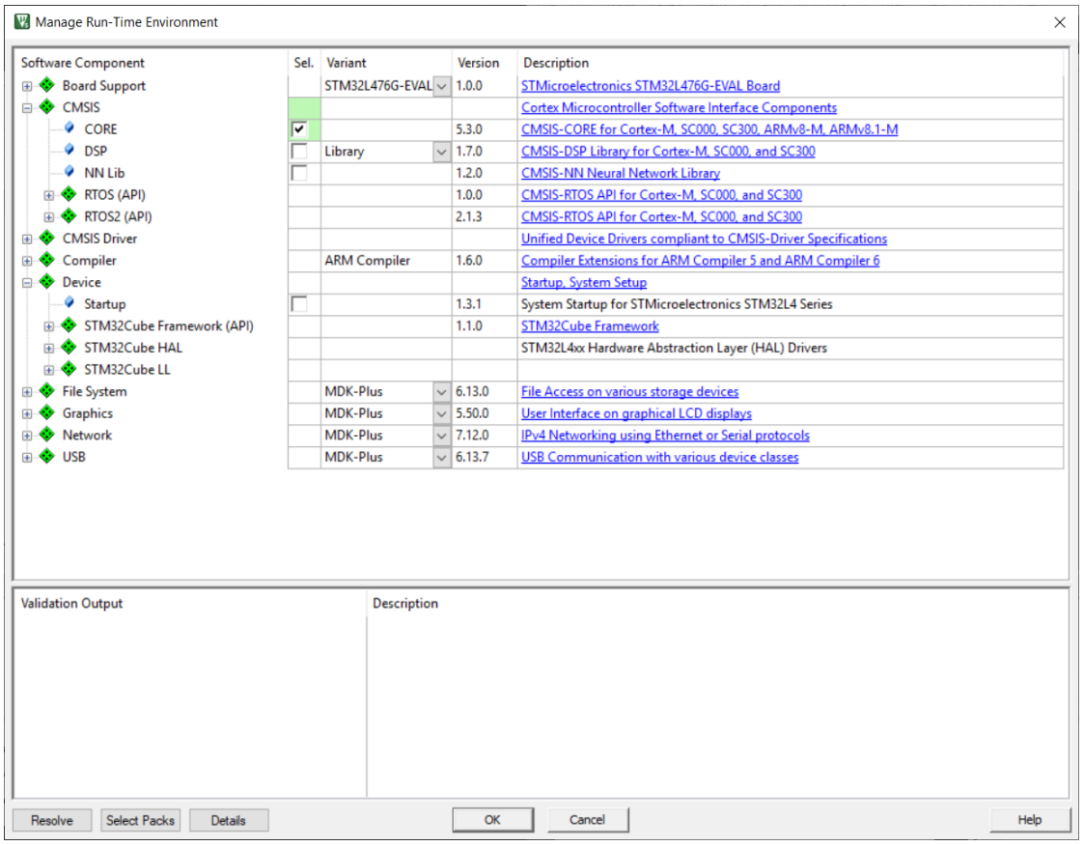


Figure 4: Manage Run-Time Environment

4. In the **Project** window, add files `main.c`, `stm32l476xx.h`, and `startup_stm32l476xx.s` (see Figure 5).

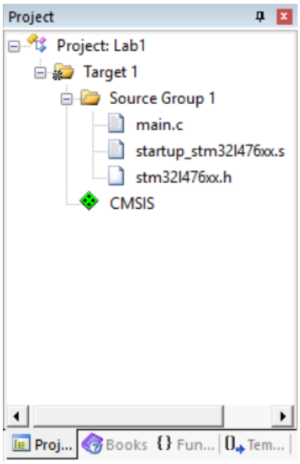


Figure 5: Add Files to Project

5. Set options for the target by right-clicking **Target 1** and selecting **Options for Target**.

   - Under the **Target** tab, select **Use default compiler version 5** under **Code Generation → ARM Compiler** (see Figure 6).

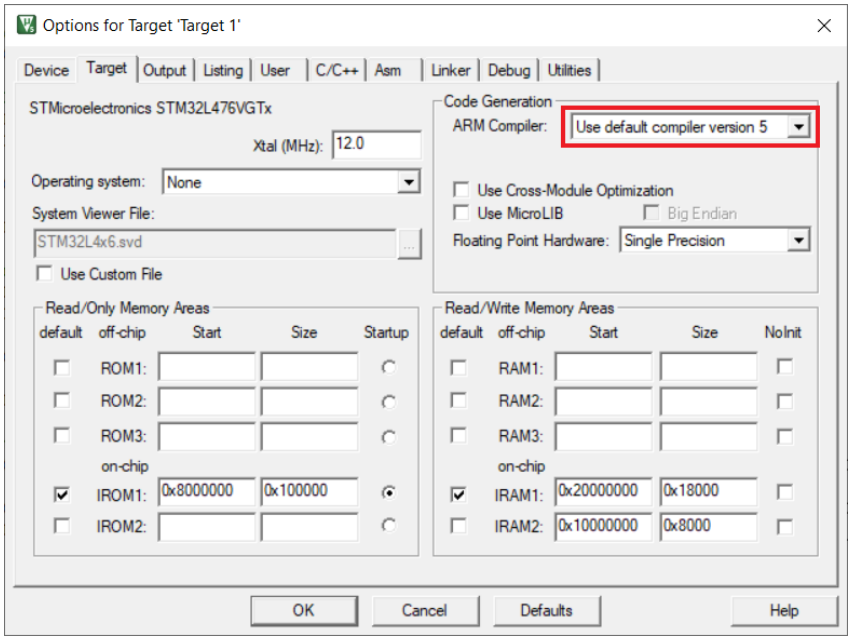


Figure 6

   - Under the **Output** tab, select **Create HEX File** (see Figure 7).

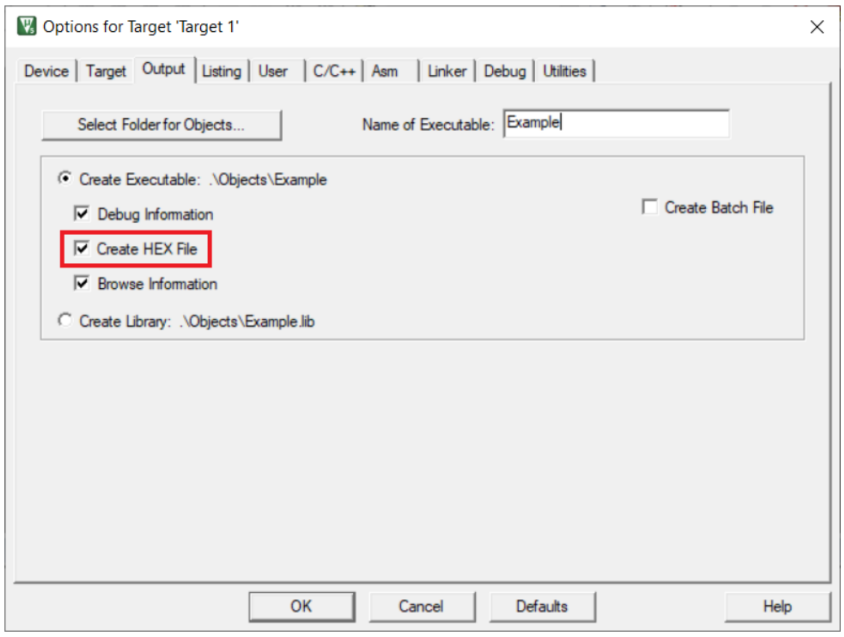


Figure 7

- Under the **C/C++** tab, ensure that the **C99** option is checked (see Figure 8).

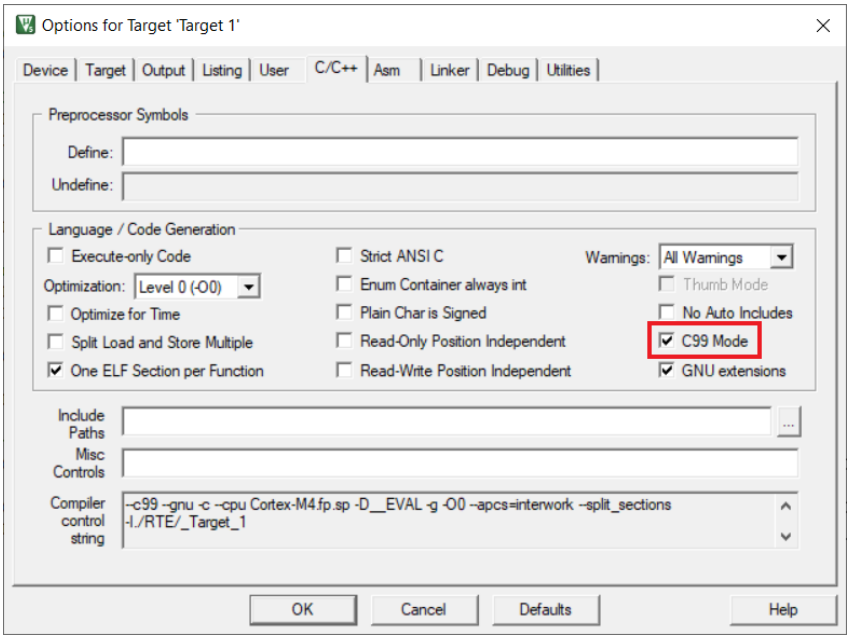


Figure 8

- Under the **Linker** tab, select **Use Memory Layout from Target Dialog** (see Figure 9).

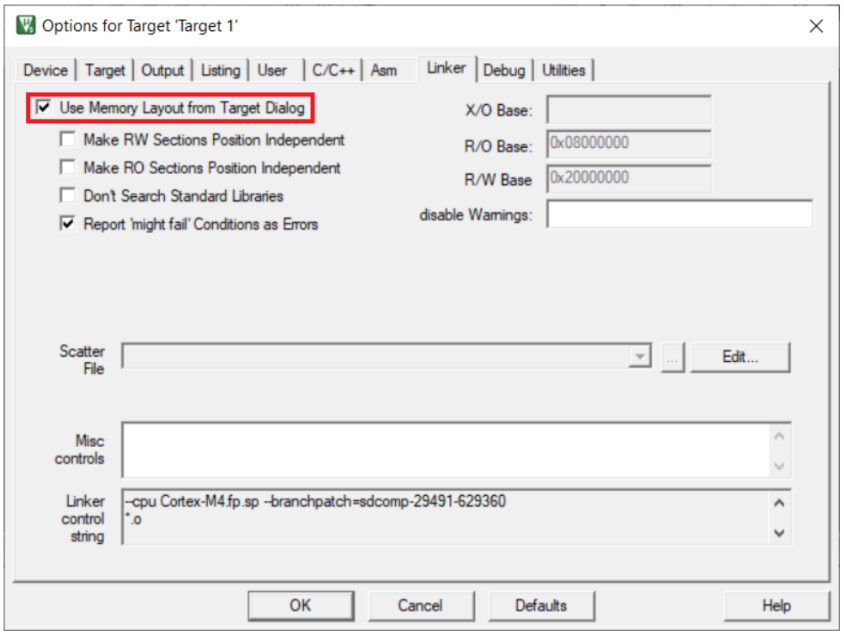


Figure 9

- Under the **Debug** tab, find the **Use** option at the top-right of the window. Select `ST-Link Debugger` from the drop-down menu (see Figure 10).

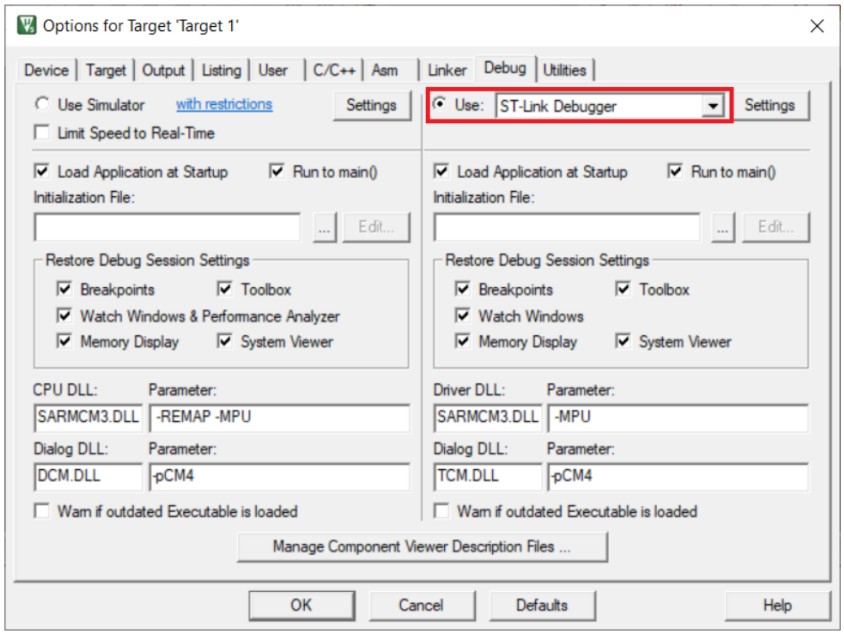


Figure 10

Note: If you don't have the board, you can still simulate your program. Select **Use Simulator** option under the **Debug** tab and skip to Step 6. (see Figure 11).

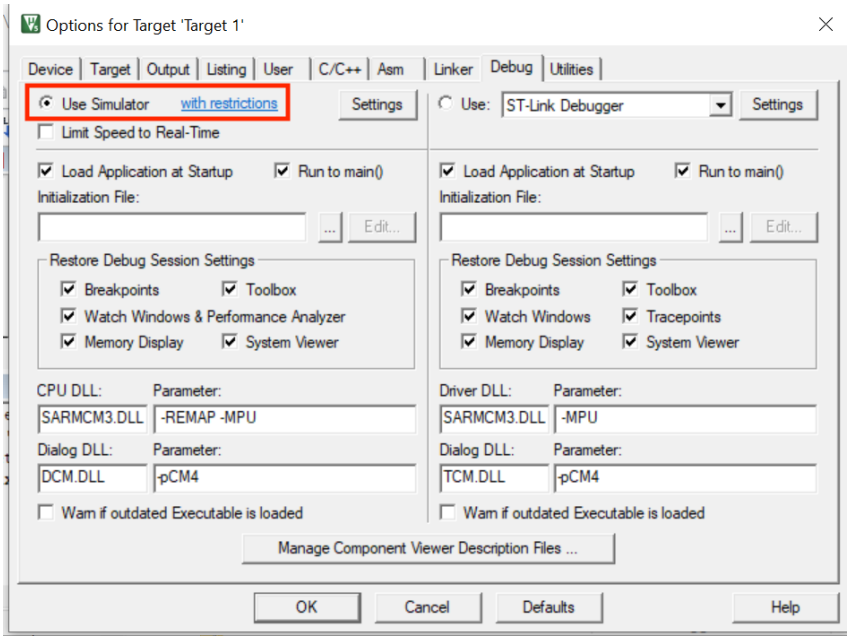


Figure 11

Click the **Settings** button next to the drop-down menu.

- Under the **Debug** tab, select `SW` from the drop-down menu for the **Target Com → Port** option (see Figure 12).

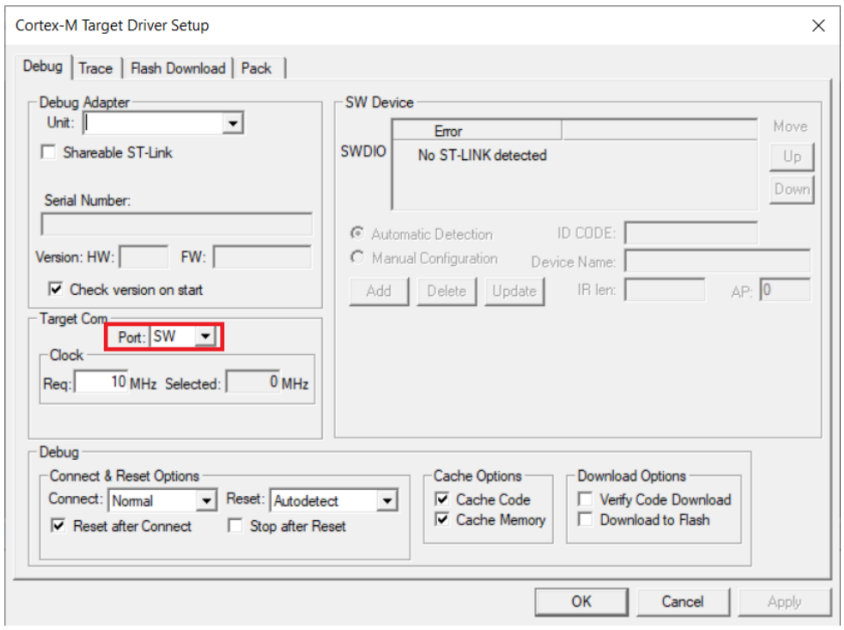


Figure 12

- Ensure that the board is plugged in. For the **Debug Adapter → Unit** option, select `ST-Link/V2-1` from the drop-down menu (see Figure 13).

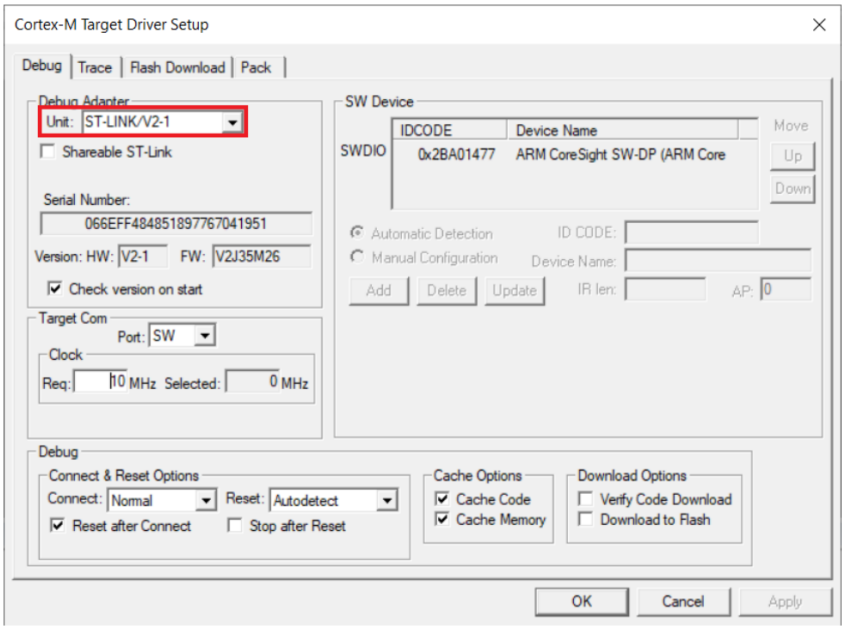


Figure 13

If that option is not there, select `Under Reset` for the **Connect** option and check the **Reset after Connect** option. Now, retry to see if the `ST-LINK/V2-1` option shows up.

- Under the **Flash Download** tab, select the `STM32L4xx 1MB Flash` option for the **Programming Algorithm** (see Figure 14). If this option is not present, manually add and select it by clicking the **Add** button.

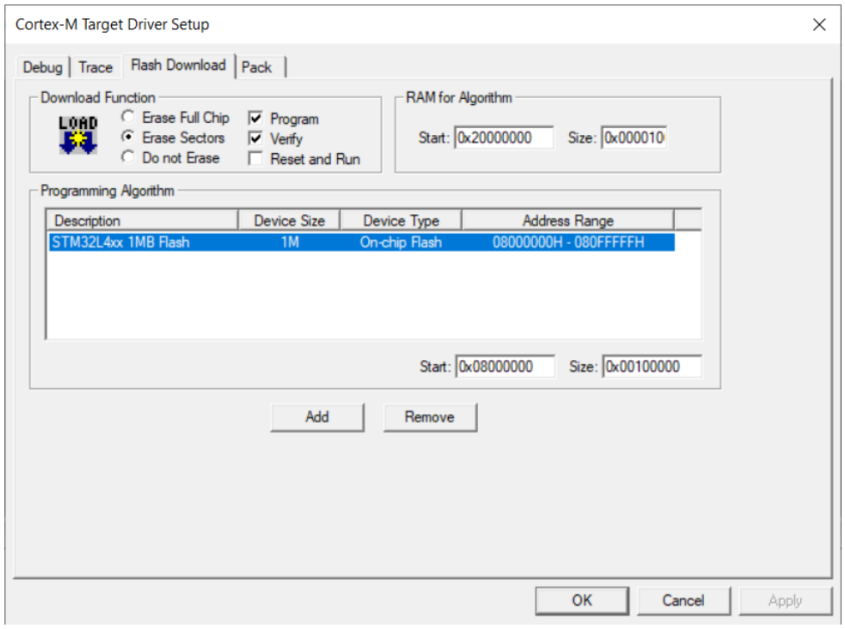


Figure 14

6. Click **OK** to save all settings and go back to the main screen of the IDE.
7. Click **Build** to build your target files. After your project finishes building, click **Download** to program the STM32 flash memory.

Figure 15: Build Button

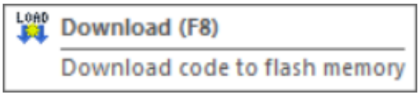


Figure 16: Download Button

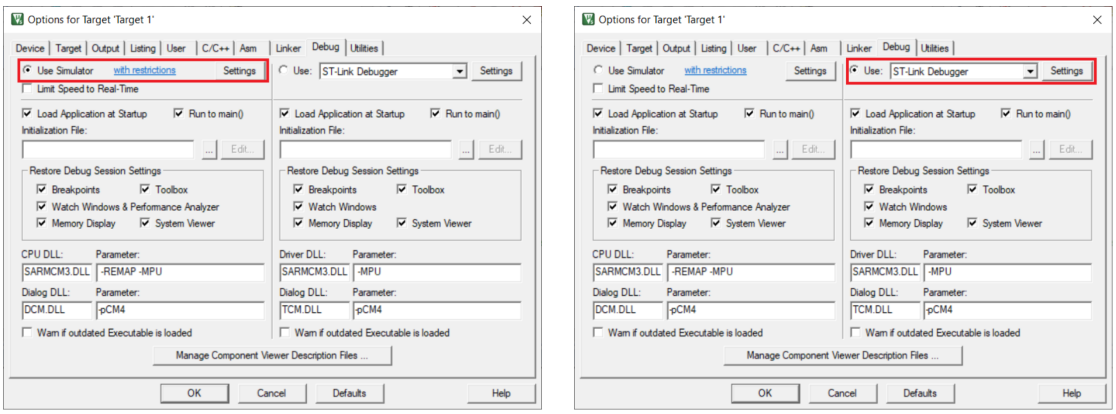
8. Pressing the reset button on the board will run your program.

Chapter 3

Keil Debugger

3.1 Software *vs* Hardware Debugging

There are two methods for debugging your program: software debugging and hardware debugging. You do not need to have the hardware board to use the software debug, but you do need the hardware board to use hardware debug. Figure 1 shows how to select software/hardware debugging in Keil through the window that pops up by clicking **Options for Target**.



(a) Software Debug (b) Hardware Debug

Figure 1: Selecting Debug Mode

3.2 Debug Control

The following buttons are commonly used debug controls.

Download - Program the STM32 flash memory.

Start/Stop Debug - Start/Stop the debugging session. Note that if the disassembly window is in focus, the debugger executes assembly instructions step by step. If the source window is focused, the debugger steps through the program source lines instead (see Figure 2).

Insert/Remove Breakpoint - Insert/Remove a breakpoint in either the disassembly or the source window. Note that STM32 allows up to six breakpoints at a time during hardware debugging . When the program stops at a breakpoint, it stops right before executing the instruction for which a breakpoint was created.

Run - Continues the execution from the current line (that the program is paused at, either because of a breakpoint or because **Stop** was clicked).

Stop - Halts program execution.

Step - Steps through and into function calls.

Step Over - Steps through and over function calls.

Step Out - Steps out of the current function.

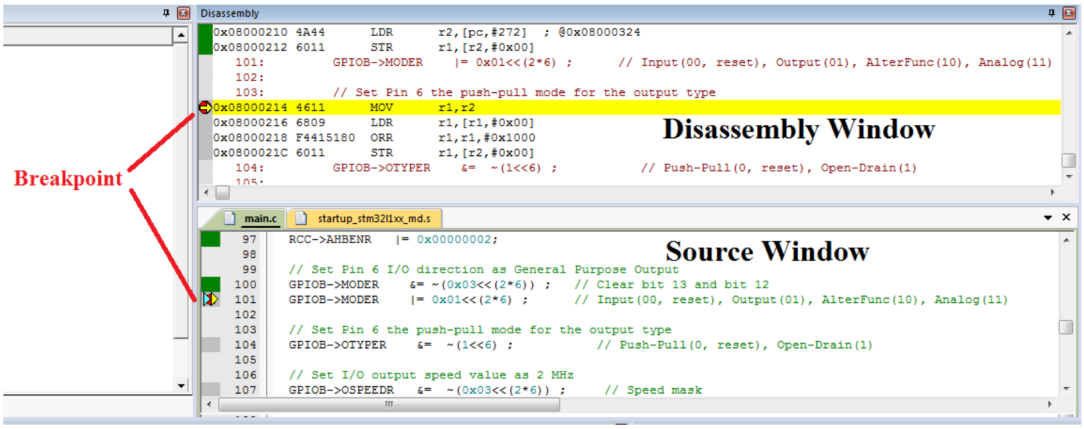


Figure 2: Disassembly and Source Windows

3.3 Memory Window

The memory window is used to view the memory content in real time. By default, the address of the data memory (RAM) begins at `0x20000000`. This is specified in the scatter-loading file (`*.sct`).

For example, the following assembly program defines and allocates an array of four words. Each word consists of four bytes.

```
        AREA    myData, DATA, READWRITE
        ALIGN
array DCD     1, 2, 3, 4
```

When we enter address `0x20000000` into the memory window, we can see the contents of the array (Figure 3).
Note that the memory format is little-endian. The format of the memory content can be changed by right-clicking the memory window and selecting a different display format.

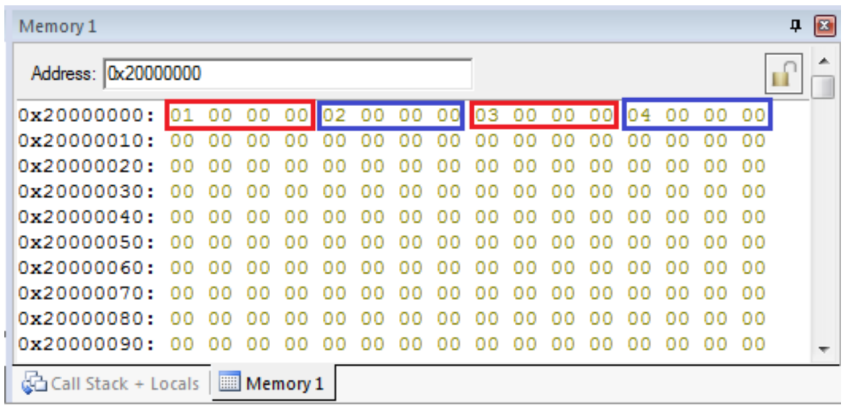


Figure 3

3.4 Saving Memory Content to a File

In the debug environment, running the command

```
SAVE <filename> <start address>, <end address>
```

allows you to save the contents of the memory (specified by the start and end addresses) into a file. The output is saved in Intel HEX format. Figure 4 shows an example of saving the contents of memory from addresses `0x20000000` to `0x20000888` into a file named `memory.dat`.

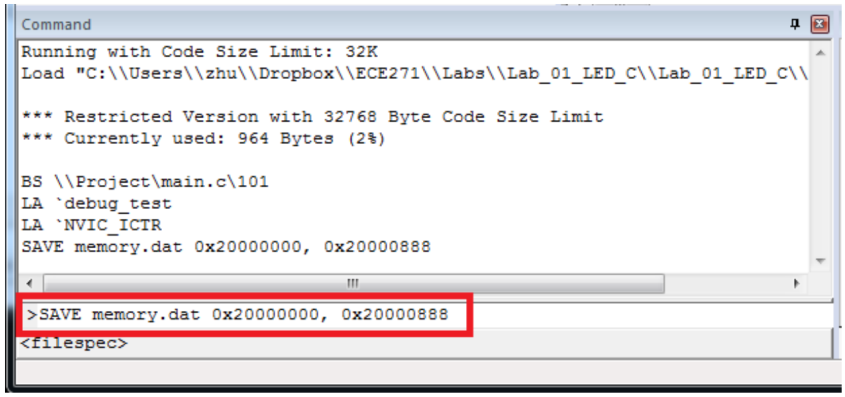


Figure 4

3.5 Processor Registers

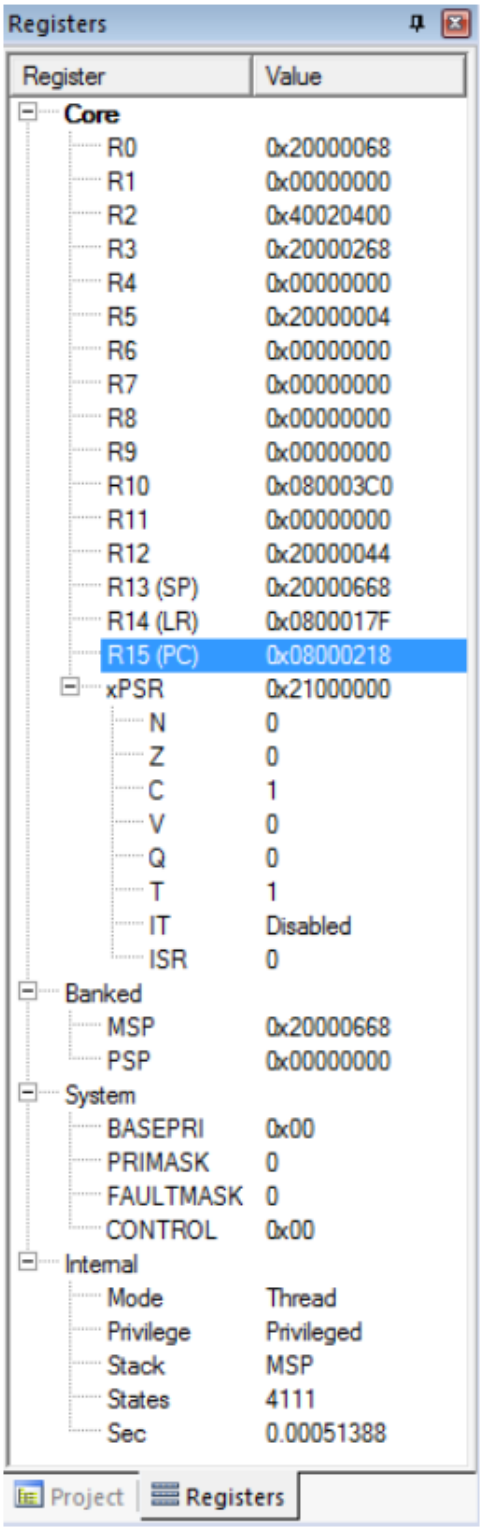


Figure 5

Core Registers

- **Program Counter** (PC/R15) stores the address of the next instruction to be fetched.
- **Stack Pointer** (SP/R13) stores the address of the top of the stack.
- **Special-Purpose Program Status Register** (xPSR) is a combination of the Application, Interrupt, and Execution PSR's. Table 1 gives details about the bits in the xPSR.

N	Negative Flag (1 = negative result)
Z	Zero Flag (1 = result is 0)
C	Carry Flag (1 = carry out)
V	Overflow Flag (1 = overflow occurred)
Q	Sticky Saturation Flag
T	Thumb State Bit
IT	If-Then Bits
ISR	Interrupt Status Register Bits

Table 1: xPSR Bits

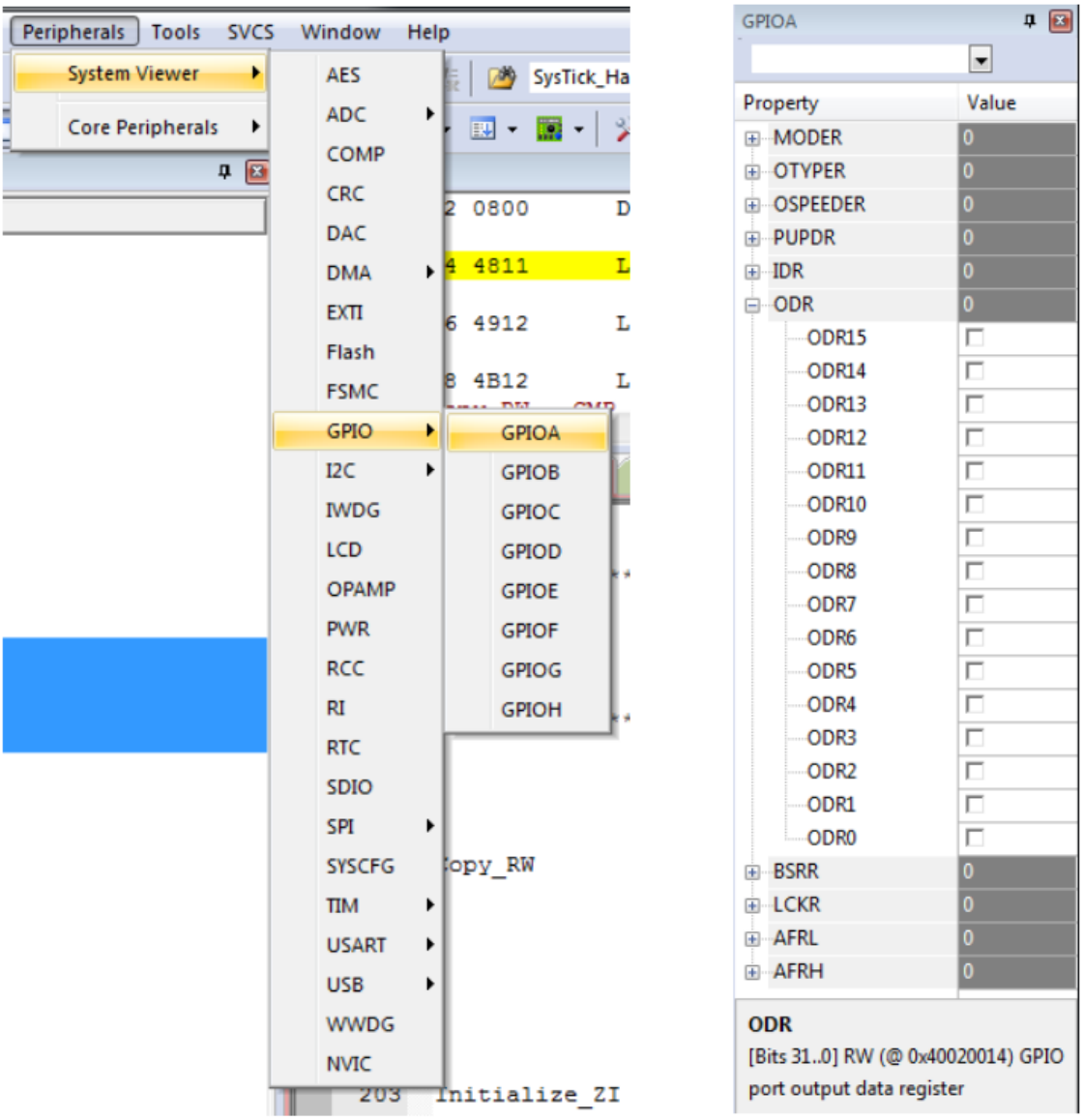
System

- **Base Priority Mask Register** (BASEPRI) defines the minimum priority for exception processing.
- **Priority Mask Register** (PRIMASK) is used to disable all interrupts excluding hard faults and non-maskable interrupts (NMI). If an interrupt is masked, it is ignored (i.e. disabled) by the processor.
- **Fault Mask Register** (FAULTMASK) is used to disable all interrupts excluding non-maskable interrupts.
- **Control Register** (CONTROL) determines which stack (main or process) is used and whether to operate in privileged or unprivileged mode.

3.6 Peripheral Registers

From the menu bar, selecting **Peripherals** → **System Viewer** allows you to view/update the control/data registers of all available peripherals.

Figure 6a show how to access the registers for GPIO Port A. Figure 6b shows each specific register for GPIOA such as **Mode Register** (MODER), **Output Type Register** (OTYPER), **Output Speed Register** (OSPEEDR), **Input Data Register** (IDR), and **Output Data Register** (ODR) among others.



(a) (b)

Figure 6

3.7 Logic Analyzer

From the menu bar, selecting **View** → **Analysis Windows** → **Logic Analyzer** opens up a separate window for the Logic Analyzer. The Logic Analyzer can display a trace of static/global variables over the time that the program is run for. Note that local variables cannot be displayed and that register values cannot be analyzed.

Let's say that we want to use the Logic Analyzer to monitor the variable `output` that is defined in the data area. Note that the `EXPORT output` instruction makes the variable `output` a global variable, which means that it can be analyzed using Logic Analyzer. First, we need to set up the Logic Analyzer – to do this, click **Setup**. Add the signal `(signed int) output` that should be observed and ensure that the data display range is adjusted such that the entire curve can be seen (see Figure 7). Figure 8 shows the signal (in this example, a sine wave) that the Logic Analyzer displays.

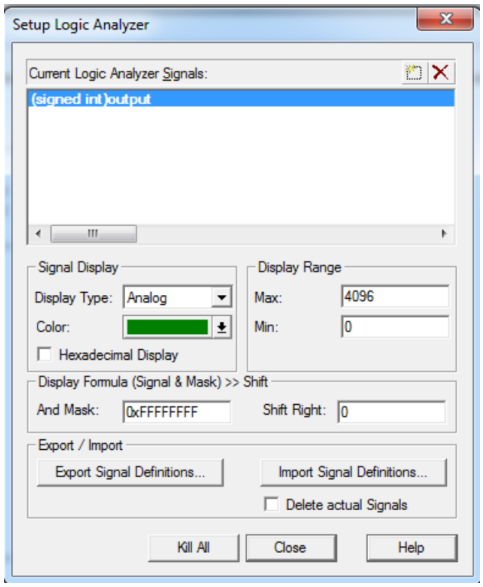


Figure 7: Logic Analyzer Setup

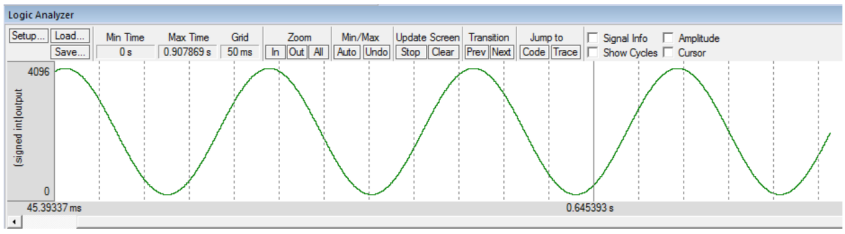


Figure 8: Logic Analyzer Display

References

[1] ARM Keil Microcontroller Tools, "Getting Started with MDK"

[2] Yifeng Zhu, "Embedded Systems with ARM Cortex-M Microcontrollers in Assembly Language and C", ISBN: 0982692633

[3] Reference Manual: STM32L4x6 Advanced ARM-based 32-bit MCU