ECE271: Microcomputer Architecture and Applications — University of Maine

Prelab for Lab #1: Joystick Button Input and LED Output

Week of 24 January 2022

Pre-lab

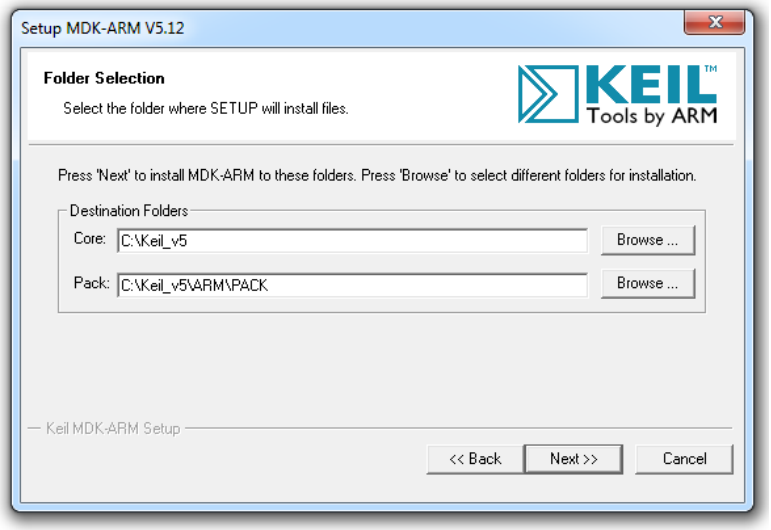
This assumes you will be doing the lab using Keil on Windows (the default). If you are feeling adventuresome and want to develop under Linux instead, see the directions in Part A.2. You should probably install the Keil tools anyway just to have it in case you change your mind later.

Part A – Setting up Keil on Windows

Caution: do not connect the STM32L4 board to your laptop until you have all the software installed. If you connect before it is installed it can confuse the Windows USB drivers and you may have to manually go in to devices under the control panel and force it to use the right driver.

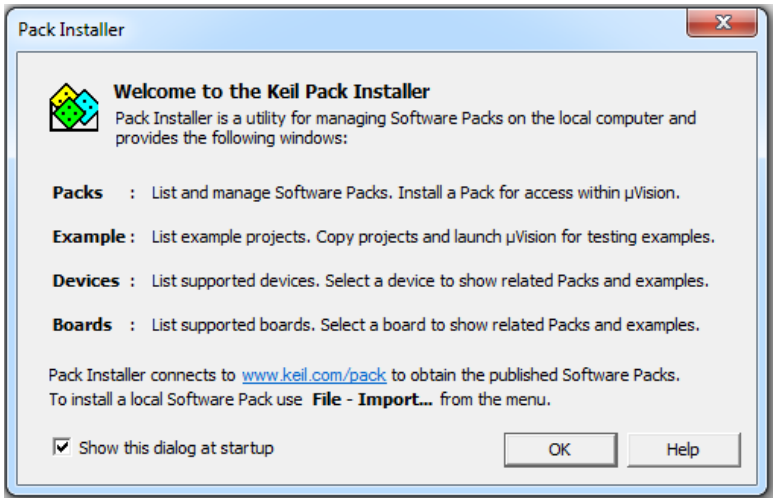
1. Download the latest "free" evaluation version of Keil MDK-ARM software from their website. Unfortunately you'll have to enter your contact info before they let you download it.
   `https://www.keil.com/demo/eval/arm.htm`
   Note, this is a fairly large download (almost 1GB)
   This includes the μVisision IDE, debugger, flash programmer, and ARM toolchain.
   This is a proprietary product and has some limitations (such as a limit on size of the project) but it is unlikely we will hit those limits with this class.

2. Run the downloaded `MDK5xx.exe` and install to the default path.

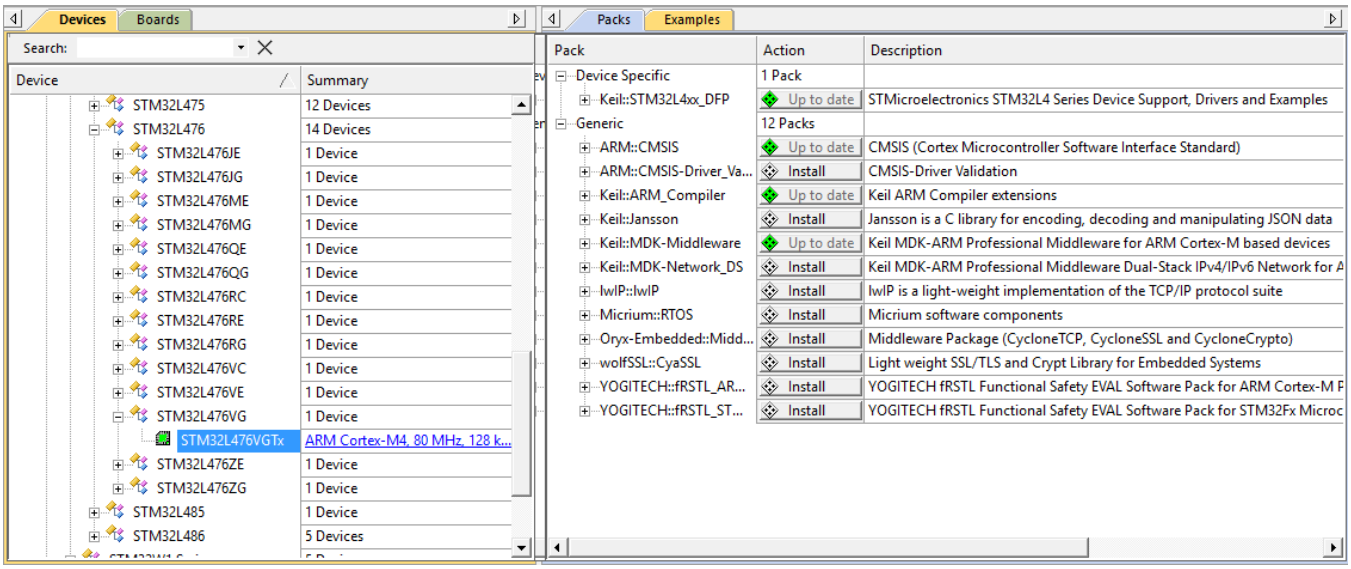


3. After the software installed, a dialog will pop up to install the Keil Pack.

4. Click OK and then another window shows up.



5. We'll be using the STM32L4 discovery kit, which has a STM32L476VG MCU in it. So select the STM32L476VG and then STM32L476VGTx, then click the corresponding "install" button by STM32L4xx_DFP on the right half of the window.



6. Once that is done, close the window.

7. Now install the ST-Linux USB driver

8. Go to the directory `C:\Keil_v5\ARM\STLink\USBDriver` and run the `stlink_winusb_install.bat` script in administrator mode (you can do this by right clicking on the batch file)

9. Once this is done, it should be safe to connect your discovery board via USB (we'll be handing them out in lab).

10. OPTIONAL STEP: you can also install the STM32 ST-Link utility from ST. This is not strictly needed, but this can be a useful program that can re-flash the memory if somehow a bad program gets uploaded

to the board and the Keil tools are unable to do anything about it.
`http://www.st.com/web/en/catalog/tools/PF258168`

Part A.2 – Setting up Linux (optional)

Only do this if you want to develop your lab under Linux. Skip this section if you are using Windows (which is probably most people).

These instructions assume you are running Debian Linux. The steps should be similar for other Linux distributions. If you are running non-Debian and have trouble, let me know and I can help set things up.

1. You will need to install the ARM cross-compiler tools, the openocd flash upload utility, and maybe st-link for the first program.

   ```
   sudo apt-get install openocd
   sudo apt-get install gcc-arm-none-eabi
   sudo apt-get install stlink-gui
   ```

2. Also install gdb. On newer systems you do this with

   ```
   sudo apt-get install gdb-multiarch
   ```

   on older systems you would instead install this

   ```
   sudo apt-get install gdb-arm-none-eabi
   ```

3. On Fedora if you get an error about stdint.h being missing you should try installing the `arm-none-eabi-newlib` package as well.

4. Also, it might be helpful if you have other development tools installed on your machine.

   ```
   sudo apt-get install gcc make git
   ```

5. You will want to find a text editor to use for development. (Alternatively you can set up a full IDE, but that's more complex). `nano` is a nice, simple editor. Other popular command-line editors are `vim` or `emacs`. There are also GUI editors you can install.

Part B – Setup Gitlab

1. We will eventually be using `git` source-control to work on and submit the assignments. Further info will be provided on that, but it's unlikely we will start using it until after the first lab.

Part C – Textbook Readings / Videos

The following might be helpful in preparing for the prelab.

1. Textbook `Chapter 4.6` to review bit-wise operations
2. Textbook `Chapter 14` to review GPIO operations
3. The classnotes are also posted to the course website.
4. The textbook has some related Youtube Tutorials here:
   `http://web.eece.maine.edu/~zhu/book/tutorials.php`
   Lectures 5, 6, and 7 there might be useful. I have to admit I haven't watched them myself so I can't vouch for how useful they are.

Part D – Prelab assignment

Before doing the lab it will be helpful to calculate what values or masks you need to write to the various registers. So go through the following and fill in the values you will need.

1. Initialize the GPIO Clocks

Before you can use the GPIO (general-purpose input/output) pins you have to enable the clock to them. By default the clock is turned off, as this saves power.

For the STM32L4 board, the GPIO clock enable pins can be found in the AHB2ENR register. You can read the official documentation for this register in section 6.4.17 of the manual "RM0351 Reference manual: STM32L4x5 and STM32L4x6 advanced Arm-based 32-bit MCUs" which I've posted a link to on the course website.

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AHB2ENR														RNGEN		AESEN				OTGFSEN				GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
Value																																

Value in hex:_____

We want to enable the two LEDs, which are hooked to GPIO-PB2 and GPIO-PE8. GPIO-PB2 is the second GPIO in bank "B", and GPIO-PE8 is the eighth GPIO in bank "E". This info can be found in section 10.5 of the STM32L4 datasheet: "UM1879 User manual Discovery kit with STM32L476VG MCU" that I've also linked to on the course website. We also want to enable the joystick buttons, which are found on GPIO-PA0 through GPIO-PA5.

This means we need to set GPIOAEN (GPIO-A enable), GPIOBEN (GPIO-B enable) and GPIOEEN (GPIO-E enable) all to 1 while leaving all the other bits alone.

Fill in the "Value" table the bits we want to set in binary, and then convert it to hexadecimal (base 16).

2. Initialize the LED GPIOs to be Outputs

Next we will need to enable PB-2 as an output. This is done in the GPIOB MODER register, which can be found described in section 8.4.1 of the manual. This register has 16 fields, each two bits wide. To set to output, the corresponding field must be set to `01` for MODER #2. To do this you will first need to clear the two bits to zero by ANDing with the proper mask, then ORing in the value. So calculate the mask below, then the value to OR in, first in binary then in hex.

Register	31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
MODER	MODER15	MODER14	MODER13	MODER12	MODER11	MODER10	MODER9	MODER8	MODER7	MODER6	MODER5	MODER4	MODER3	MODER2	MODER1	MODER0
Mask																
Value																

Mask in hex:_____ Value in hex:_____

You will need to do the same for PE8. GPIO-E also has its own MODER register (we'll worry about which register is where in the actual lab, for now just calculate the value for GPIO#8).

Register	31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
MODER	MODER15	MODER14	MODER13	MODER12	MODER11	MODER10	MODER9	MODER8	MODER7	MODER6	MODER5	MODER4	MODER3	MODER2	MODER1	MODER0
Mask																
Value																

Mask in hex:_____ Value in hex:_____

3. Initialize LED GPIOS to be Push-Pull

We want the output to be push-pull (instead of open-drain). This is set in the OTYPER register, as described in section 8.4.2 of the manual. Push-pull is 0, open-drain is 1, so we want to set PB2 (so OT2) to zero.

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OTYPER																	OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
Mask																																
Value																																

Mask in hex:_____ Value in hex:_____

Also calculate the same for the offset #8 for PE8.

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OTYPER																	OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
Mask																																
Value																																

Mask in hex:_____ Value in hex:_____

4. Initialize the LED GPIOs for No PullUp/PullDown

We will also need to set the Pull-Up / Pull-Down registers. This is done with the PUPDR register as described in section 8.4.4 of the manual. There are two-bit fields. For GPIO PB2 we want to set this to 00 which means no pull-up, no pull-down.

Register	31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
PUPDR	PUPDR15	PUPDR14	PUPDR13	PUPDR12	PUPDR11	PUPDR10	PUPDR9	PUPDR8	PUPDR7	PUPDR6	PUPDR5	PUPDR4	PUPDR3	PUPDR2	PUPDR1	PUPDR0
Mask																
Value																

Mask in hex:_____ Value in hex:_____

Now calculate the same for setting PE8 to 00.

Register	31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
PUPDR	PUPDR15	PUPDR14	PUPDR13	PUPDR12	PUPDR11	PUPDR10	PUPDR9	PUPDR8	PUPDR7	PUPDR6	PUPDR5	PUPDR4	PUPDR3	PUPDR2	PUPDR1	PUPDR0
Mask																
Value																

Mask in hex:_____ Value in hex:_____

5. Initialize the Joystick GPIOs to be Inputs

Next we will need to enable PA0, PA1, PA2, PA3, and PA5 to be Inputs. This is done in the GPIOA MODER register, which can be found described in section 8.4.1 of the manual. We want to set the corresponding MODER fields to 00 which means input.

Register	31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
MODER	MODER15	MODER14	MODER13	MODER12	MODER11	MODER10	MODER9	MODER8	MODER7	MODER6	MODER5	MODER4	MODER3	MODER2	MODER1	MODER0
Mask																
Value																

Mask in hex:_____ Value in hex:_____

6. Initialize the Joystick GPIOs to PullDown

We will also need to set the Pull-Down register. Again, this is done with the PUPDR register as described in section 8.4.4 of the manual. There are two-bit fields. We want to set PA0, PA1, PA2, PA3, and PA5 to have the pulldown enabled, with is `10`.

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPDR	PUPDR15		PUPDR14		PUPDR13		PUPDR12		PUPDR11		PUPDR10		PUPDR9		PUPDR8		PUPDR7		PUPDR6		PUPDR5		PUPDR4		PUPDR3		PUPDR2		PUPDR1		PUPDR0	
Mask																																
Value																																

Mask in hex:_____ Value in hex:_____