Department of ACES, SHU

Embedded Systems Design

Semester 2 academic year 2017/18

**Lab A: Familiarisation with STM32CubeMX for firmware development**

Lab A consists of three assignments  
- development of custom firmware from a slightly modified code template generated by the STM32CubeMX;

- custom modification of a given STM32CubeMX project, generating the code template and running the developed firmware on it;

- creating a new STM32CubeMX project from scratch.

***Assignment 1 (Lab A1)*** - develop firmware utilising GPIO HAL drivers

Your firmware will use the four LEDs and user pushbutton (blue) available on the STM32F4Discovery board.

The firmware will switch on and off two LEDs, toggle another and control the last one by using the pushbutton (the LED is ON when the pushbutton is pressed).

First you will need to determine your personal variant by using your 8 character student ID number like bXXABCDE; last five numbers denoted ABCDE will be used.

Please use the following table to select your LEDs   
(their names in the table are stated as they are shown on the board)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Last digit of the student ID number  (E from bXXABCDE) | LED to toggle | LED to control by the pushbutton | LED1B | LED2B |
| 0 | LD3 | LD4 | LD5 | LD6 |
| 1 | LD4 | LD5 | LD3 | LD6 |
| 2 | LD5 | LD3 | LD4 | LD6 |
| 3 | LD3 | LD5 | LD4 | LD6 |
| 4 | LD5 | LD4 | LD3 | LD6 |
| 5 | LD4 | LD3 | LD5 | LD6 |
| 6 | LD3 | LD6 | LD4 | LD5 |
| 7 | LD6 | LD4 | LD3 | LD5 |
| 8 | LD4 | LD3 | LD6 | LD5 |
| 9 | LD3 | LD4 | LD6 | LD5 |

The startup code added to the CubeMX generated template shows how to use the Delay function and prints the clock frequency (consult appendix C of the reference manual on how to view the printf output in the keil MDK).

All your code will be executed inside the superloop ( indefinite **while(1){…}** loop ) that is very common to most embedded systems that operate 24/7. (E.g., in Arduino programming environment such the code is laced inside the ***loop*** subroutine.)

The firmware needs to toggle the required LED every time the superloop is executed.

The firmware also needs to turn ON/OFF the LED1B and LED2B in the superloop in some order after some delays that are specified below.

Additionally, check whether the pushbutton is pressed and, if yes, switch on the appropriate LED.

The pseudo code structure is as follows (***fill in blanks***):

while (1) {

toggle the required LED (LD\_\_\_);  
 check the pushbutton and control the required LED;

// t1=(D+3)\*0.1s (D is from bXXABCDE) =\_\_\_\_\_;

switch LD\_\_\_ on, LD\_\_\_ off; provide delay of t1=\_\_\_\_\_ ms;

// t2=(C+5)\*0.1s (C from bXXABCDE) =\_\_\_\_\_\_\_;

switch LD\_\_\_ off, LD\_\_\_ on; provide delay of t2=\_\_\_\_\_ ms;

// t3=B\*0.1s (B from bXXABCDE) =\_\_\_\_\_

switch LD\_\_\_ off, LD\_\_\_ off; provide delay of t3= \_\_\_\_\_ ms;

}

In your code use HAL functions (L1 / slides 19-21). You will need to find the MCU pins which the button and LEDs are connected to from the schematic diagram available from the Discovery board manual (BB / Semester2 / Reference materials).

***Demonstrate your working code to the lecturer in the lab as soon as it is developed.***

=============================================================

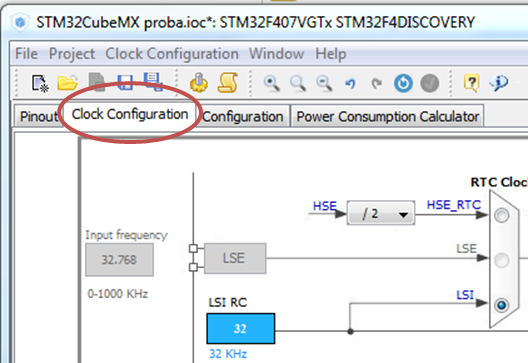
To undertake the coursework developments from home install STM32CubeMX to your home computer then install the STM32CubeF4 firmware package:   
- select Help / InstallNewLibraries…   
- scroll down and select the top checkbox under the heading STM32CubeF4 Releases - it is 1.18.0 in January 2018  
- click Install Now at the bottom of the window.

***Assignment 2 (Lab A2)*** - custom modification of a given STM32CubeMX project, generating the code template and running the developed A1 firmware on it

You will need to amend the given STM32CubeMX project A2.ioc available on the BlackBoard.

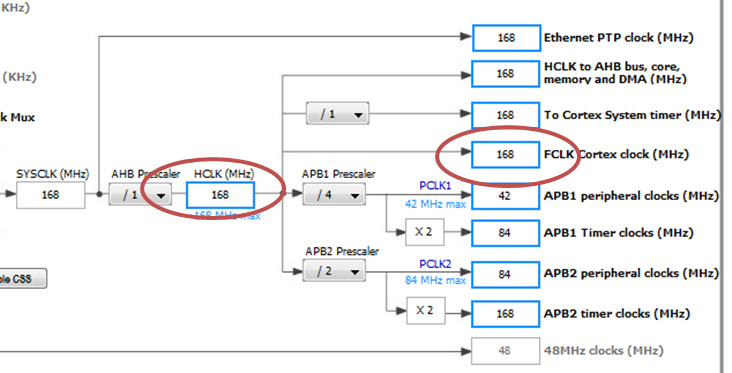
The amendments include:

a) changing the clock frequency FCLK/HCLK (the clock for the MCU Cortex M4F core) to the value determined by the last three digits of your student ID number. Please use the tab "Clock configuration" in the STM32CubeMX:



If the number is bXXABCDE, the frequency to be set is 10\*B+C+0.1\*D + 20 (MHz). For example, the student with ID number b1234567 will need to achieve the said clock frequency of 10\*5 + 6 + 0.1\*7 + 20 = 76.7 MHz. This is the minimal frequency you design should run at (any higher frequency is fine).

Some students will be lucky to get the required frequency by typing it straight into the "FCLK (MHz)" box (or into the "HCLK (MHz)" box that defines the same frequency).

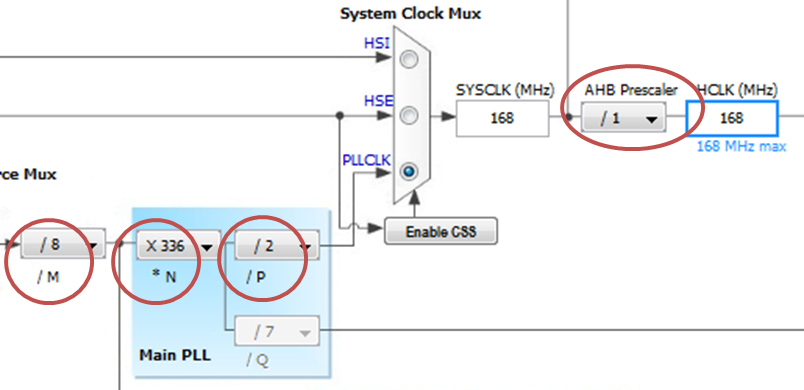


Most students will be not, and they will require to type increasing integer frequencies into some box and hit Enter until the CubeMX finds an appropriate arrangement for available PLL/divider settings.   
***Please take a screenshot of the clock window for the report.***

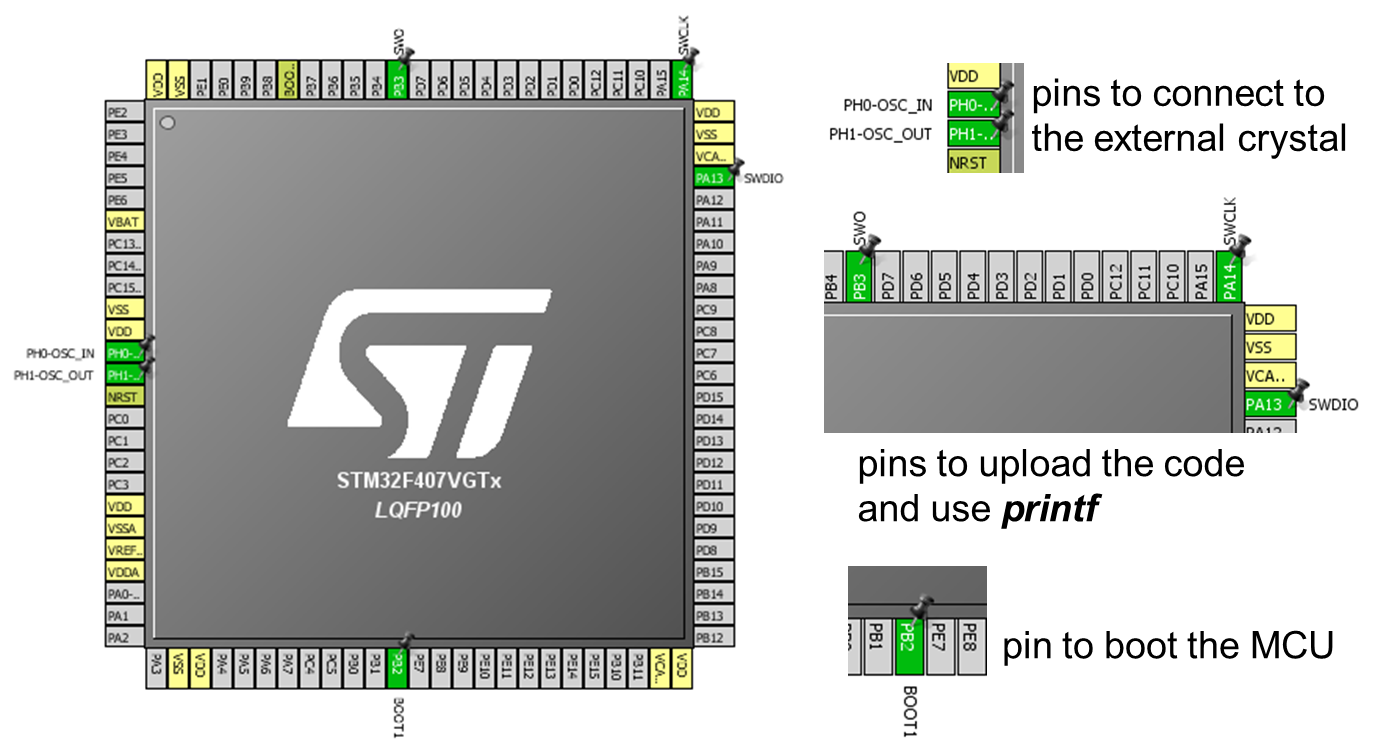
Mathematically

HCLK = 8 MHz (frequency of the on board crystal) /M \*N /P /AHBprescaler

where all the above parameters are shown in the screenshot below



b) initialising the pins connected to the LEDs and pushbutton appropriately.   
Please do not alter functions of the already initialised (and pinned down) pins because this will prevent your project from working:



You need to find the pins connected to the LD3..LD6 and set these as ***GPIO\_Output***; the pin connected to the pushbutton and set it as ***GPIO\_Input*** (left click and choose from the list).

Generate the code template (RefMan, appendix A).

Proceed with the code development using Keil ARM-MDK (RefMan, appendix B).

PLEASE DO REMEMBER THAT YOU HAVE TO PUT ANY NEW CODES ONLY WITHIN THE SPACEHOLDERS GENERATED BY THE STM32CubeMX.   
This will enable you to re-generate the MDK-ARM project from the STMCubeMX project later without losing any of your code.

Copy all the files you have got for the Lab A1in the ***Src*** folder, and paste these in the ***Src*** folder of the new project, overwriting all the existing files. Generate the code from CubeMX one more time to overwrite the old settings in the code. Compile and run the codes from Keil IDE, observing the value for the new clock frequency.

In order to enable operation of **printf** , please consult RefMan, appendix C.

***Assignment 3*** ***(Lab A3)*** - creating a new STM32CubeMX project from scratch

Motivation for this assignment comes from a typical development scenario - you have developed firmware for a particular application using the Discovery board and want to manufacture the designed device in some sizeable quantity. From your completed design you know that the code takes that much flash and RAM and needs that many MCU pins. Then you need to select an appropriate (but hopefully cheaper and/or more convenient to use) MCU and design a PCB for it.

I only printed some key points for this assignment but additional information is available from the electronic version of this lab sheet.

High-level descriptions of actions to be taken for this assignment:

- using your student ID number, find the minimal MCU requirements;

- start a new STMCubeMX project (File…/New project) and use filters for the requirements to narrow down your options to only 5 MCUs out of 1100+ MCUs available as of Jan 2018;

- use the principal design goal to select the only one MCU for the project;

- set up the pins for the external crystal oscillator (for this you need to enable the oscillator in the RCC peripheral) and the same clock frequency as in the Lab A2;

- set up the pin for the boot;

- set up the pins for uploading the code and using ***printf*** (the names of these pins could be different from these you have used, please read the relevant note);

- set four pins to output to control the required four LEDs (ideally pins from the same port with the same pin numbers; if this is possible then the code will require no changes);

- set one pin to input to read a push button ((ideally pins from the same port with the same pin number as discussed above).

***Selecting the MCU requirements out of your student ID number (bXXABCDE):***

number of input/output pins : 10 \* A + 10 \* C + E = \_\_\_\_\_\_\_\_\_\_\_\_\_\_

size of the flash memory, kB : 100 \* B + 10 \* D + 20 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_

size of RAM, kB : 10 \* C + 10 \* D + 10 \* E = \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Use the sliders to set the ***minimum*** values that can be set and are acceptable according to you variant.

***Selecting the most important additional selection requirement*** ***out of your student ID number (bXXABCDE):***

using the numbers D and E from your student ID number as follows (D+E) % 5, where % stands for the remainder operator in C. For example, student b1234567 will select line (6+7) % 5 = 3 from the table below:

|  |  |
| --- | --- |
| (D+E) % 5 | The most important additional selection requirement for the MCU |
| 0 | lowest cost on Digikey for 1 pcs |
| 1 | lowest cost on Farnell for 1,000 pcs |
| 2 | lowest cost on RS for 100 pcs |
| 3 | lowest cost on Mouser for 10 pcs |
| 4 | smallest physical size of the package |

Please take a screenshot(s) comparing the five appropriate MCUs (obtained, e.g., by using Windows Snipping Tool), and highlight the one that you selected using the above criterion.

Please do read through the Q&As as these were asked by students and address various real world issues.

***Q&A for the Lab A3***

***Q1: why I cannot use the Discovery board for production?***

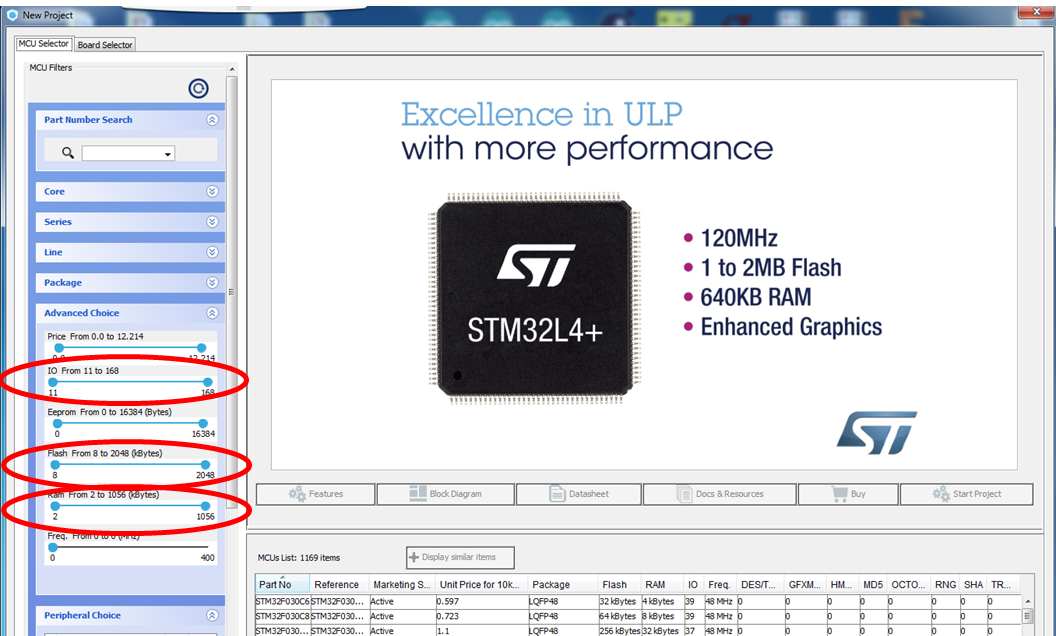
For two reasons. First, the cost of the MCU alone is the same as the cost of the board - STMicroelectronics make loss on the boards to get people used to their product lines (which makes the board perfect for educational purposes indeed - plenty of features at low cost). However they explicitly excluded any use of board for production; when you buy a board you agree to this. If you try to replicate the board yourself the cost will quickly mount up, maybe making the whole device too expensive for the purpose.   
The second reason is that you may want using another (hopefully substantially cheaper) MCU to lower costs of your custom device as the CortexM4F MCU is quite advanced and you can get away without many of its high performance features (like DSP capabilities and built in floating point hardware) without the need to re-design the firmware. (Yes, you will still need to design a custom PCB but it was the case if you try to replicate the Discovery board anyway.) Therefore you need to select a suitable MCU, develop a new project for the selected MCU using STM32CubeMX, generate C code for MDK and copy your developed firmware. Assignment 3 gives you an opportunity to develop an STM32CubeMX project from scratch.

***Q2: Is it possible to use the same CubeMX project for another MCU?***

In short NO. STM32CubeMX projects can be re-used for the same microcontrollers only. That is because the hardware setup code to be generated for different microcontrollers is different. However you will use the same STM32CubeMX workflow as before; and your generated project will have similar placeholders where you can copy/add appropriate code. Thus strating a new project is necessary.

***Q3: How to enable the required selection criteria?***

When starting a new STMCubeMX project, first you will be presented with the MCU selector pane where you need to select the microcontroller for the project



Most likely there will be several microcontrollers that are appropriate for the design, and you will need some further considerations to select the only one. From the obtained MCUs list (like the one shown above) note down only five microcontrollers (that ideally belong to different series if it is possible) for further considerations.

***Q4: were there any complications with the MCU selection in the past? What to do if there are less than 5 MCUs are available?***

Quite a few. As it is a real world exercise, we have got a lot of real world problems that were resolved in the following way:

- all the package sizes of the five microcontrollers, selected in STM32CubeMX, were the same - please use any other criterion to select only one (e.g., pick the cheapest MCU out of these five);

- some of the selected MCU could not be found at the required supplier or became deprecated/no longer stocked/are not available in the required quantity etc - just use the ones that are available at the required quantity and pick one.

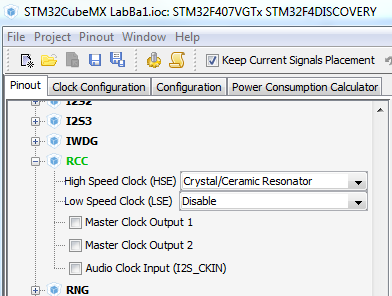
(The reason for this is that in real life every company deals with one preferred components supplier - some discounts/return/shipping terms are negotiated in exchange for the loyalty. Thus you are going to order from this supplier only. STM makes 800+ microcontrollers, and no supplier can stock all of these in any quantity. If you order is really big, in real life you may call STMicroelectronics directly though.)

The bottom line: get down to ***random*** ***five*** MCUs in the STM32CubeMX, and get to a single one using your criterion if it is possible. If not (e.g., several MCUs have the same price), make your pick at random.

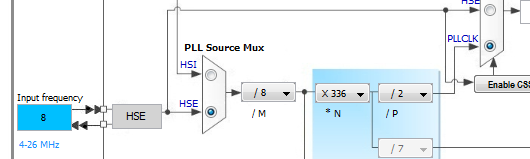
***Q5: Any more information on how to configure the clock and set the crystal pins?***

You need to enable the external oscillator 8 MHz crystal and relevant crystal pins to easily set the same core clock frequency. This can be done by tweaking the peripheral RCC (Reset and Clock Config) in the left part of the "Pins" tab of the STM32CubeMX.

You need to unfold RCC and enable external crystal as shown below



After this, the crystal pins will be configured automatically. Proceed to the clock configuration tab and enter the same 8 MHz frequency for the external crystal (input frequency) as we have for the crystal on the Discovery board:



Please check that the two multiplexers shown above are set as shown (click at the bottom radio button(s) on the multiplexers if it is required).

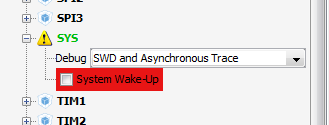
Then set the required FCLK Cortex clock frequency according to your variant.

***Q5: Any more information on how to set the pins?***

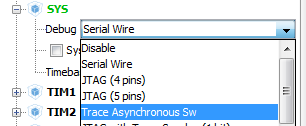
When you are presented with 100+ pins it is easy to get lost. They do have "Find" facility but unfortunately it does not take partial names for alternative pin functions.

When you type "PD" in the "Find" box, all the pins of port D will start changing their colour. Pick up PD12..PD15 outputs if possible not to have to change any user code. If these pins are not available, use any other pins to set as outputs for driving the LEDs.

The easiest way to set the ***Debug*** pins, which are required for debugging the code at the run time, is to enable the debug option into the SYS peripheral (tab Pins of the STM32CubeMX) like it is shown below



or (depending on the MCU)



(Do not be afraid of the System Wake-Up painted red - this option is no longer available if you use pin PA0 for input. Therefore you simply cannot enable it which is shown by the colour. More through discussion of this issue is included into description of the assignment 2 for Lab B.)

The first option is available when your chosen MCU is equipped with the instrumentation trace macrocell (ITM) hardware like the MCU on board. For quite a few MCU this hardware is not available to save on the cost. In this case use the option "Serial wire" instead. You will be able to run, stop, set breakpoints but no longer ***printf*** to the Keil MDK's console).