

Lab 7

Basic I/O and Finite State Machines

Objectives:

- Use basic I/O on the Keil development board
- Implement a basic FSM using a foreground/background system

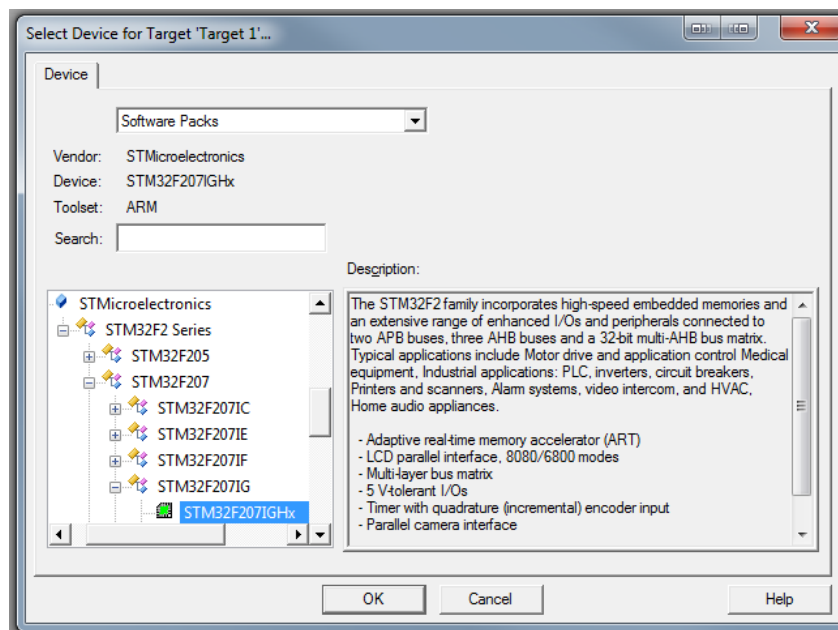
Files Needed:

- lab7.c
- Serial.c, Serial.h
- I2C_STM32F2xx.c, I2C.h
- JOY.c, JOY.h

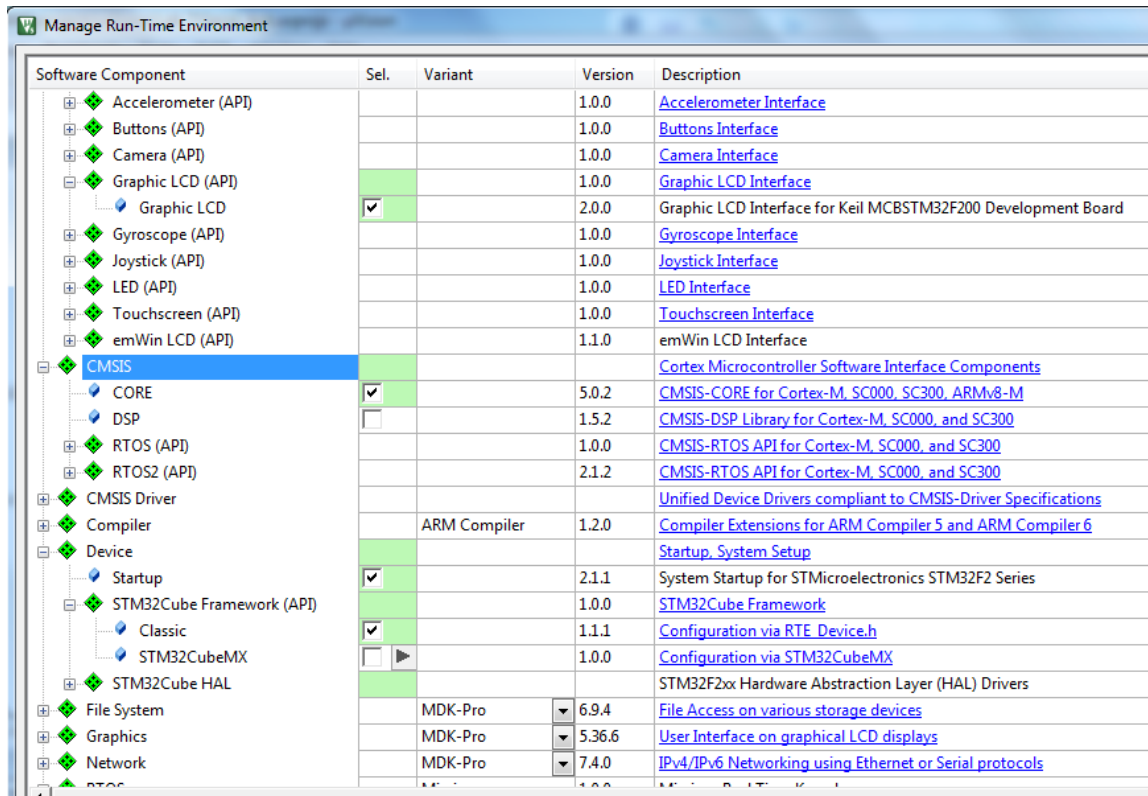
Assignment:

Create a new project using the following steps. **There are differences from Lab 6!**

1. Open the Keil uVision5 development environment.
2. Click “Project”, then “New μ Vision Project...”
3. In the “Create New Project” dialog window, change to an appropriate directory and enter a name for the project file.
4. In the hierarchy, select “STMicroelectronics”, “STM32F2 Series”, “STM32F207IG”, “STM32F207IGHx”.



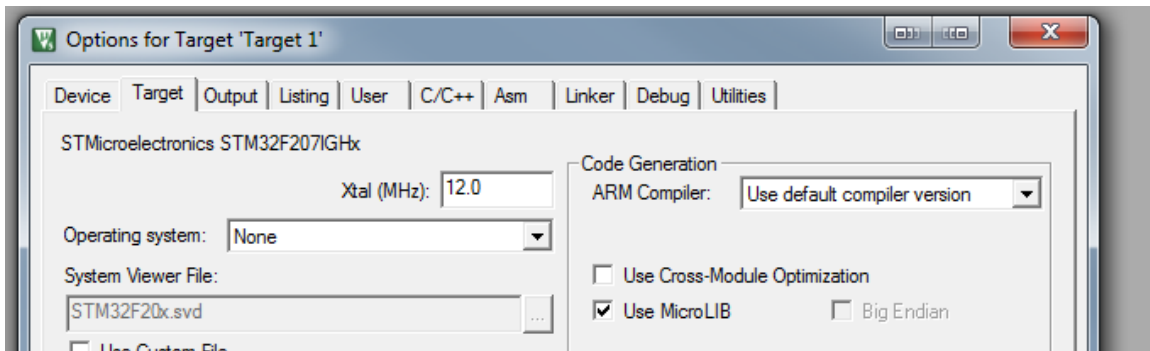
5. In the Manage Run-Time Environment dialog window, there are three components that need to be added. Click on *CMSIS*->*CORE*, and click on *Device*->*Startup*, then OK. Generally, these are always added. Items under “Board Support” access provided code for using the peripherals specifically on the MCBSTM32F200 board. Enable the *Graphic LED* as shown below, as well as *Device*->*STM32Cube Framework (API)*->*Classic*. Finally, click on the *Resolve* button at the bottom left when everything has been selected.



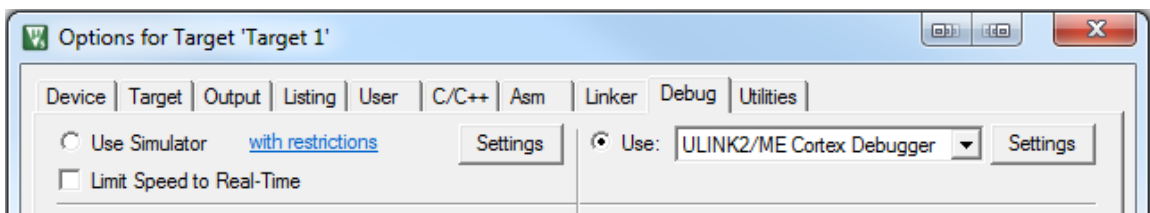
6. Copy all files listed under **Files Needed** above from Canvas to your project directory. This may be the same directory as the project file (*.uvprojx), or any subdirectory.
7. Add the four .c files from step 6 to the project by right-clicking on “Source Group 1” and selecting “Add Existing Files to Group ‘Source Group 1’...”. You do not add the three .h header files.

The three sets of .c and .h files provide drivers to access the joystick (which needs I2C) and the USART serial port. Although these components have drivers in the uVision Run-Time Environment window shown above, the latest versions require significantly more additional code. For now, the earlier, stand-alone versions will be easier to use.

8. Right click on “*Target1*” in the Project tab, click “*Options for Target ‘Target1’...*”, and click the “*Use MicroLIB*” checkbox in the “*Code Generation*” box of the “*Target 1*” tab.



9. Switch to the “*Debug*” tab in the Options window. This lab requires the physical board, so make sure that the “*ULINK2/ME Cortex Debugger*” is selected.





10. Build the project by clicking on the “*Build*” icon  in the upper left of the IDE window. The file should build without any errors or warnings.
11. Enter the debug mode by clicking the “*Start/Stop Debug Session*” icon . The IDE will warn you that only 32k of code is supported, which is fine. Then, run the program.
12. The program will send and receive characters using RS-232, which can be controlled with a computer using the USB-to-RS-232 cable provided. Figure 1 shows which port the cable should use. Note also that the three jumpers **MUST** be in the positions shown.



Figure 1 - Jumper Positions for USART 3

13. A fairly common program to use on the computer's side is PuTTY, though you may use any terminal program. Assuming that you installed PuTTY, open the program. The first window is the *PuTTY Configuration* window. The only thing you need to do on this window is click the Serial radial button in the upper right, then click on Serial in the hierarchy in the bottom left.

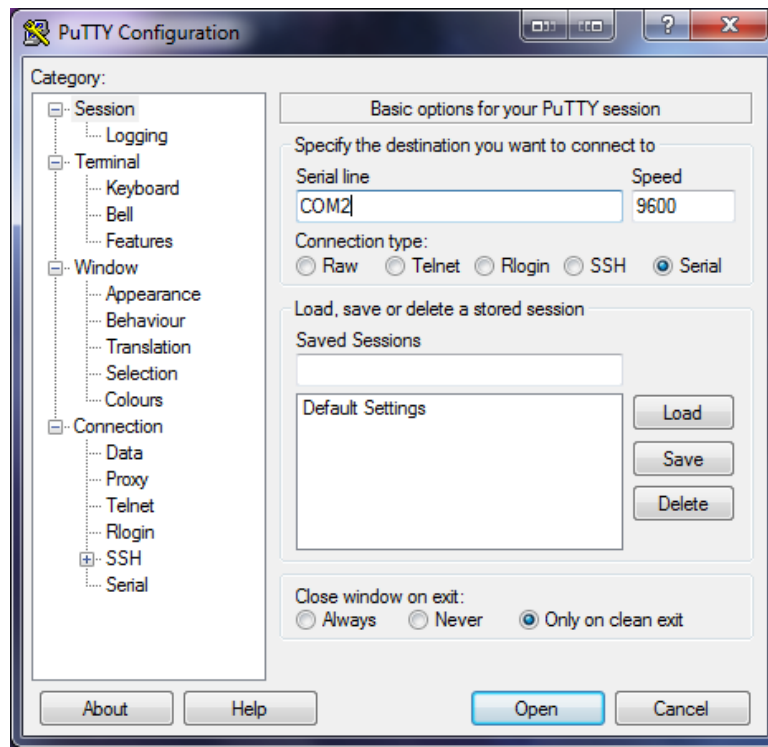


Figure 2 - Initial PuTTY Screen

14. The window should change to the configuration below. Enter/select the values shown, with the exception that COM2 needs to be replaced with whatever COM port your cable was assigned. PuTTY does not know/detect this for you. Then click *Open*. If everything is working correctly, the board should slowly send a stream of characters.

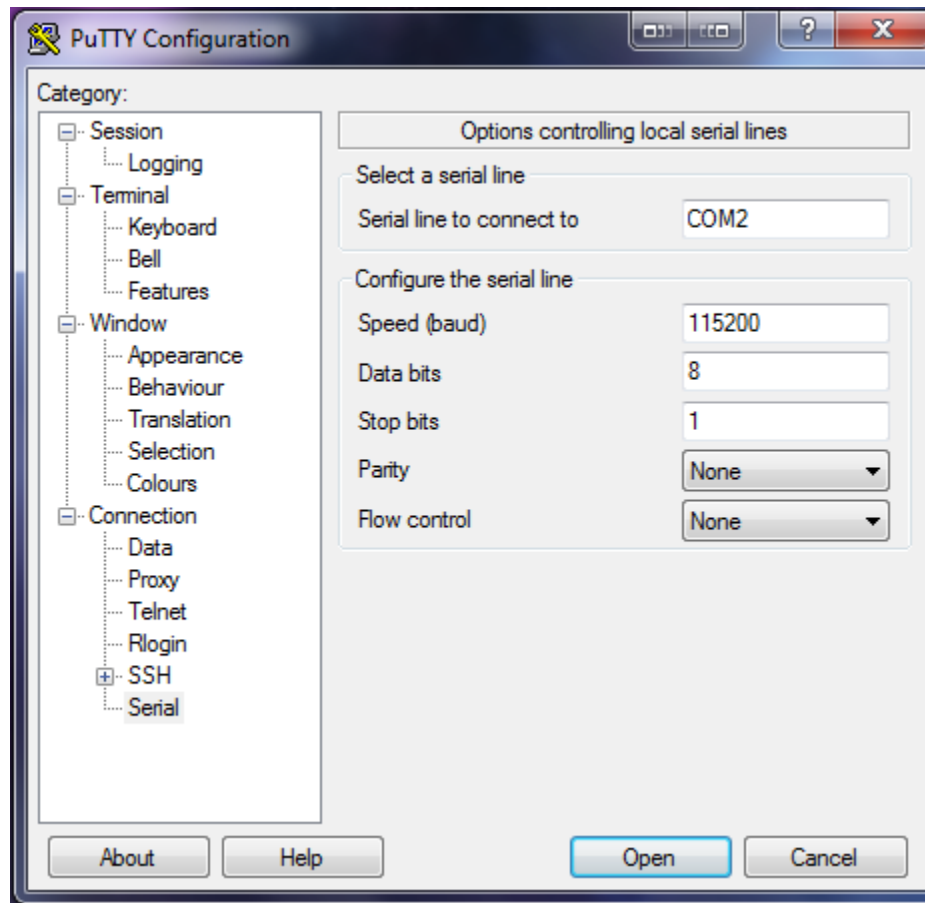


Figure 3 - Second PuTTY Screen

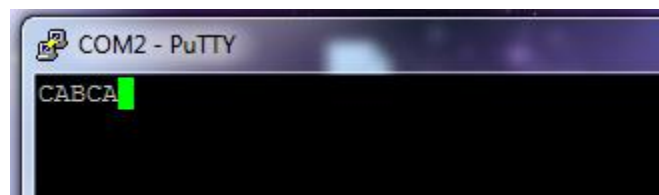
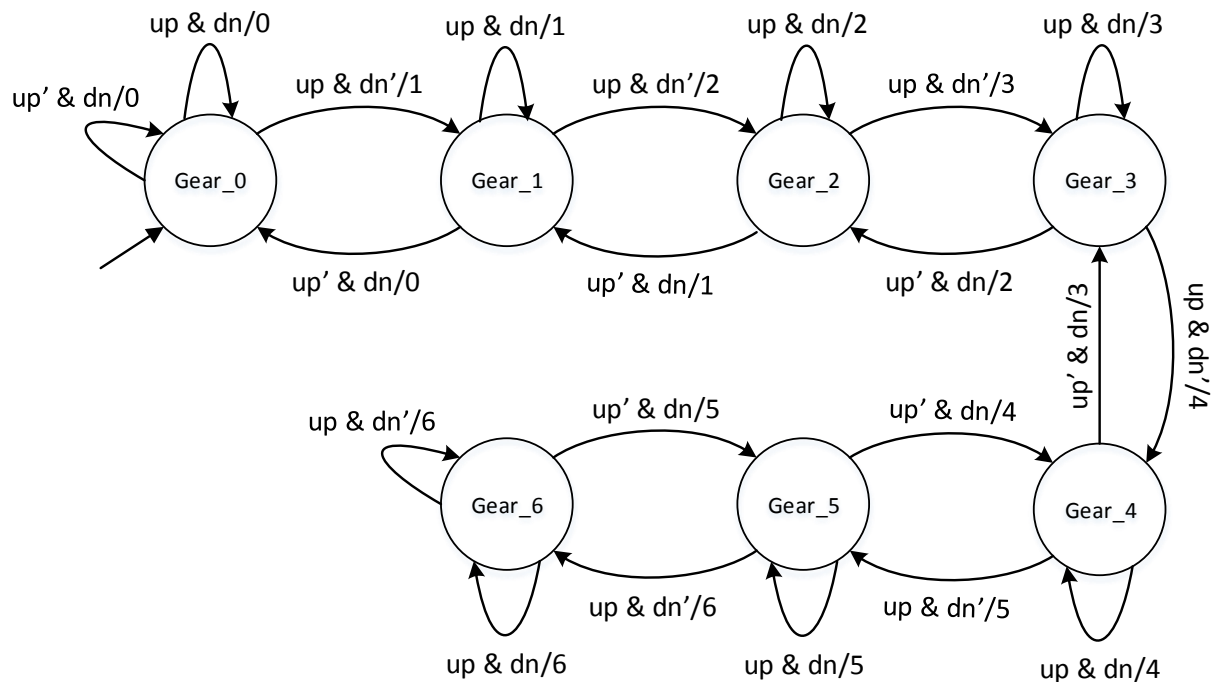


Figure 4 - Sample of Output from Board after ~25 seconds

15. Alter the program to implement the finite state machine below. The upshift input (*up*) to the FSM must come from receiving a 'U' or 'u' on the serial port, or detecting that the joystick was pressed downward relative to the screen. The downshift input (*dn*) to the FSM must come from receiving a 'D' or 'd' on the serial port, or detecting that the joystick was pressed upward relative to the screen. The output *gear* must write the corresponding character to both the middle of the graphic LCD screen and to the serial port.

Inputs: *up*, *dn* : pure

Output: *gear* : character



Deliverables:

Upload an electronic copy of the deliverables to Canvas per the lab submission guidelines. Please note: **ALL** deliverables must be present in the single PDF report.

- Lab7.c file with additional code.