Instructions

You have from now until Friday (1 April) morning 8:00 to write code to perform the tasks specified. At 8:00 am you will meet in the Electronics lab where we have been having lectures and will be required to answer some questions on your code.

You should bring your development board (along with all other hardware, including cables, handed out during the course) programmed with your code. You should also bring a printed version of your code (including the LCD file used) with your name on it as well as those you collaborated with. You may also bring your printed notes.

It is beneficial to get some of the tasks working properly rather than trying all of them and having them not work properly. Please remember to comment your code. Please also email a .zip file containg all your source code or submit via RUConnected (including any include files used beyond the m16def.inc) as well as the .hex and .eep files to A.Sullivan@ru.ac.za Your code will be verified against the .hex provided as well as what is actually running on the micro-controller.

If there are any problems with the hardware you may contact me on 0722746534. (not after 6pm and not before 8am)

You may not get assistance from any member of staff or any student that has done the course before.

Hardware Layout

- PORTD4-7 will be used to drive the stepper motor via DRV0-3
- PD0 connected to TX on the USB-serial converter
- PD1 connected to RX on the USB-serial converter
- PD2 connected to Button 1
- PD3 Connected to Button 2
- PA0 connected to VAR
- PA1-7 connected to LEDs 0-6
- PC0-7 connected to LCD_Data0-7
- PB0 connected to LCD_E
- PB1 connected to LCD_RW
- PB2 connected to LCD_RS

Specifications

- On startup, your code should read 3 messages from EEPROM to SRAM. Each variable will contain up to 19 characters and be terminated by a NULL (the 20th character in the longest string). The variables in EEPROM will not necessarily be the same length, or the maximum length, they will be packed in to take up the least needed space. All three variables should be read to RAM and retained, even if they are not used.
- The LCD should be initialised to 2 lines with a non-blinking cursor.
- The stepper should be initialised to hold.
- At startup the following menu of options should be sent to the PC via the UART (and USB/serial converter) operating at 9600,8,n,2

```
Project tasks:
--------------
1) Rotate clockwise for 5 seconds at a low speed - half steps
2) Rotate anti-clockwise for 5 seconds at a low speed - half speed.
3) Rotate clockwise for 10 seconds at a high speed - single steps
4) Rotate anti-clockwise for 10 seconds at a high speed - single steps
5) Disable stepper.
6) Enable Stepper.
7) Start ADC PWM Task.
8) Stop ADC PWM Task.
9) Print 3rd stored message to LCD.
10) Clear LCD.
11) Reset Microcontroller.
```

- The micro controller should listen on the UART for commands (the numbers sent via the terminal). The commands should only be parsed when the "." is received after the number.
- Explanation of tasks.
    1. The stepper motor must rotate clockwise for 5 seconds, using half steps. The speed should be such that the half stepping is evident.
    2. The stepper motor must rotate anti-clockwise for 5 seconds, using half steps. The speed should be such that the half stepping is evident.
    3. The stepper motor must rotate clockwise for 10 seconds, using single steps. The speed should be such that the single stepping is evident.
    4. The stepper motor must rotate anti-clockwise for 10 seconds, using single steps. The speed should be such that the single stepping is evident.
    5. All outputs to the stepper should be switched off after receiving this command.
    6. The stepper should hold again (any position).
    7. The ADC should be initialised to perform a conversion on ADC0. This result should be displayed as a bar graph on LED0-6. The brightness of the leds should be controlled as well. The higher the voltage the brighter the LEDs. A single conversion is needed, so that every time the command is received the LED output is updated.
    8. The ADC should be disabled and all LEDs put off. (This is the default state of the ADC task.)
    9. The 3rd message read into RAM should be printed to the LCD, with text wrapping.
    10. The LCD should be cleared.
    11. The micro controller should be reset.
- On beginning the tasks (1-4) the menu should be cleard and further reception of characters via the UART disabled.
- On completion of those tasks the menu should be reprinted and reception of characters on the UART re-enabled.
- While a task is being executed, input from the UART should be ignored and discarded.
- While Tasks 1-4 are in operation, pressing button 1 should pause the stepper motor.
- When button 2 is pressed the stepper function should be unpaused and continue with its' operation.

Please build and check your hardware **before** leaving the lab today (monday)
Your hardware can be checked by getting it programmed by me up front.
A working version is available for viewing before you leave the lab.

Hints

- Use the template (main.asm) provided on RUConnected - just to give you a hint and so you don't have to type the menu from scratch.
- Get the transmission of the menu done first. Make sure it is working properly.
- Get the reception and parsing of the task selection working. (at this stage you can use the LEDs as debug LEDs.
- If all of that is working correctly then you can start doing each actual task.