****
Proj_1A_LEDdisplay THE FIRST PROGRAM: Learning to drive the led segments.

This uses the display to generate a simple pattern that repeats endlessly.

LEARNING SOMETHING ABOUT THE C PROGRAMMING LANGUAGE

1. The 'main' routine:
When power is applied or immediately after programming, program execution starts with the "main" routine. Note the '{}' symbols used to initiate and terminate this routine.

2. The "for-loop": Code within the {} brackets is executed 16 times as "m" increments from 1 to 16. In C, 'm++;' is shorthand notation for add 1 to m. When 'm' gets to 17 program execution jumps to the statement 'SW_reset;'.

3. Variables: 'PORT_1' and 'm' are names assigned to variables. They can take on numeric values which are the results of logical or arithmetic operations. In this case 'PORT_1' is initialised to 1 and then doubled every time that the 'for-loop' repeats (the shift symbol '<<' followed by 1 is equivalent to multiply by 2).

4. Comments: No text in green needs retyping to get the program working. This is what is called 'comment'.

DISCOVERING THE ATMEGA HARDWARE (HW)

1. The Central Processor Unit (CPU): This is the unit that performs the arithmetic and logical operations (plus, minus, and, or, not etc.) including '<<' already referred to.

2. Data memory also know as SRAM: This is memory space in which variables such as 'PORT_1' and 'm' are stored. In this case the variables are defined as integer, and 16 binary digits (bits) are reserved for each one. 16 bit numbers can hold 2^16 values. PORT_1 which is unsigned can hold values from 0 to 65535. 'm' which is signed (by default) can hold numbers between -32,678 and +32677.

3. Output ports: Output ports are connected to the display and used to power the segments. For each bit of PORT_1 there are corresponding segments in the display. For each bit that is set to 1 these segments are illuminated and each each bit that is set to 0 they are not.

4. I2C bus: This enables data to be shared between the Atmega devices. Here it is used to enable user code running on the UNO device to drive the display which is under the control of the PCB 111000_1 device.

5. A HW timer T0 that pauses program execution for 60ms in this case.

INTRODUCING THE PROJECT FIRM WARE (FW)

1. The 'setup_HW' macro: This is a code segment that sets up the UNO device HW (hard ware). For example it determines which pins are to be configured as outputs and which as inputs.

2. The I2C subroutine 'I2C_Tx_2_integers': This sends 32 data bits (each 0 or 1) over the I2C bus to the PCB 111000_1 device, where the mini-OS uses them to control illumination of the 32 vertical led segments.

3. The 'SW_reset' statement: This causes program execution to repeat from the beginning as it does for a Power on Reset (POR) when power is first applied.

4. The .h file: This gives details of 'setup_HW', and 'SW_reset', and the location of the 'I2C subroutine'. It also tells the compiler about certain WinAVR files that will be required.

DRIVING THE DEVELOPMENT TOOLS

To create the .hex file click on 'tools' and then select 'Make All'. Select 'Make Clean' to delete all extra files generated except the .hex one. Note: The Makefile has been slightly edited to ensure that the .hex file is not also deleted.