Explanation of Program

The program begins with the normal programmer comments on author, date, hardware and outputs. The include files are next, which are the essential header files needed to control the PIC. The Local Function Prototypes declare functions that timer functions for the PIC to keep time and use delays, but declaring functions are optional. Global Variable Declarations declare the variables that will be used throughout the program. Most of the variables are constants, meaning that they are not changed anywhere in the program. There is a variable for the pin number of each output to the LEDs and input for the buttons. The only variables that are change are the variables that tell if a button has been pressed. When the pedestrian button or advance green button is pressed, the corresponding variable (pedOn or advOn) is set to 1 and after the signal has been executed, the variable is reset to zero. An array is used to contain the output pin numbers for the stepper motor. They are ordered so that the program cycle through the numbers forward or backward and make the stepper motor spin counterclockwise and clockwise.

The main function is what the program executes. First, the setup is performed. In the setup, the peripherals are initialized (timer started), the PORTA is set to all digital inputs, and PORTB is set to all digital outputs. The rest of the code which controls the traffic cycles are inside a while(1) loop. This repeats the code forever. The traffic cycle code controls green light cycles on the north-south corridor (NS) and east-west corridor (EW). It can execute a pedestrian signal on the EW corridor, an advance green signal on the NS corridor, and a level crossing at any time. The green lights last for 7 seconds, the yellow lights last for 2 seconds, and all intersection red lights are on for 1 second before switching to the next green cycle.

First, all intersection red lights are turned on. The program waits 1 second, while checking all button inputs. Instead of using a delay only, which stops the entire program and prevents the detection of inputs, a for loop repeating 100 times is used. Inside the for loop, a delay of 1/100 of the time that the program waits for used. For example, if the program, is supposed to wait 1000 milliseconds, delay of 10 is used. In the loop, the program checks 100 times if the pedestrian, advance green or level crossing buttons are pressed. If the pedestrian or advance green button is pressed, the program sets its corresponding variable to 1. If the level crossing variable is pressed, the program immediately executes the level crossing function. Next, the NS red light is turned off. The program checks if advance green has been pressed. If so, the advance green function is executed (flashing green light on NS for 3 seconds) and advOn is reset to 0. The NS green cycle starts. The NS green light is turned on, and the program waits 7 seconds using the for loop mentioned above, checking if the pedestrian button is pressed. The NS green light is turned off, and the NS yellow light is turned on. The program waits 2 seconds while checking if the pedestrian button is pressed. The NS yellow light is turned off and the NS red light is turned on. The EW red light is turned off. The program checks if the pedestrian button has been pressed. If so, it executes the pedestrian signal, which is flashing yellow EW lights for 3 seconds and resets pedOn to 0. Then, the EW green cycle starts. It is the same as the NS green cycle, except that the program constantly checks if the advance green button has been pressed.

After the main functions, there are custom made functions to make the code easier to write. The first function is digitalWrite, which assigns a state to a PORTB output 6 to 15. This function is convenient because the output can be expressed as a single number, instead of a long variable such as LATBbits.LATB6. It uses the switch-case structure, which does something based on what number it is given. The next function, digitalRead, returns a value from PORTA inputs 0 to 4. It also uses the switch-case structure. The pedestrian functions flashes EW yellow lights 3 times in 3 seconds and the advance function flashes NS green lights 3 times in 3 seconds. The railroad function is used to activate the level crossing procedure. It accepts a parameter (lightOn) which indicates which light is active at the time the function is called. The active light is turned off, and all red lights are turned on. All red lights flash 2 times in 2 seconds. The stepper motor rotates 90 degrees counterclockwise to lower the boom gate. All red lights flash 3 times in 3 seconds. The stepper motor rotates 90 degrees clockwise to raise the boom gate. All red lights flash 2 more times in 2 seconds, ending with all red lights off. Then, the program figures out which red light it is supposed to turn back on. If the active light number is less than redNS, then redEW is turned back on. Otherwise, redNS is turned back on. Next, the original active light is turned back on, and normal traffic cycle resumes. The last function is rotate. It accepts two parameters. The first one is the direction of rotation (0 for counterclockwise and 1 for clockwise). The second parameter is number of cycles. The stepper motor has four outputs, so one cycle is 30 degrees (4 steps of 7.5 degrees). The program cycles through the D array of stepper motor outputs forward for counterclockwise, and backward for clockwise. Each output is turned on and off with a delay of 40 milliseconds.