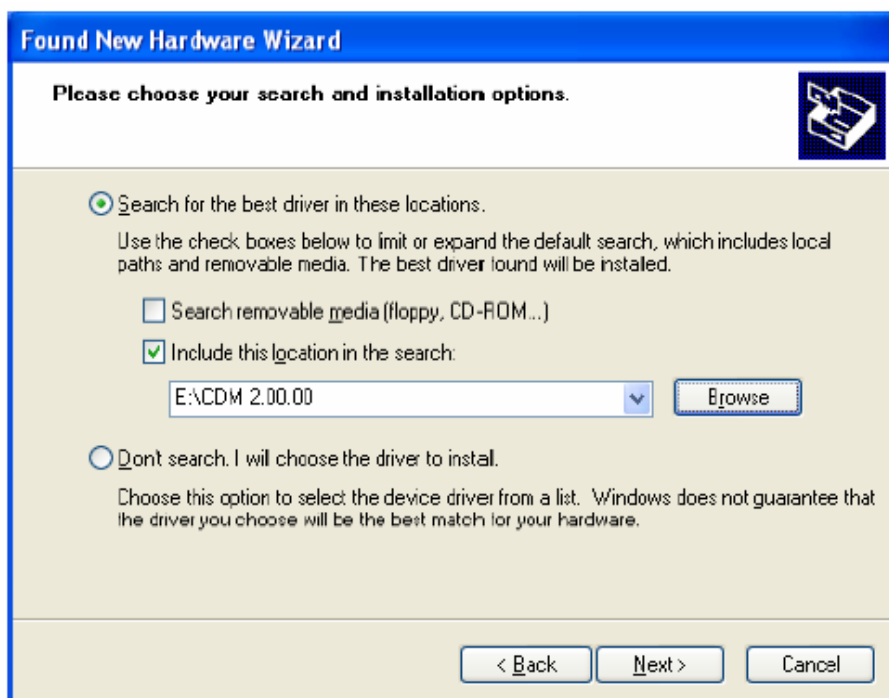


In this box, select installation from a specific location and select next



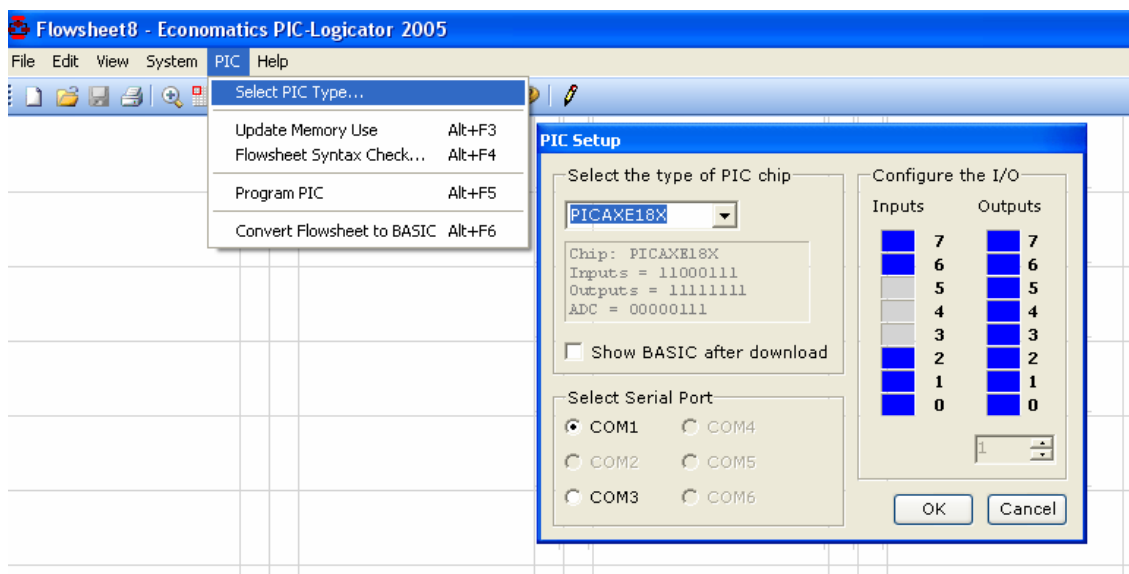


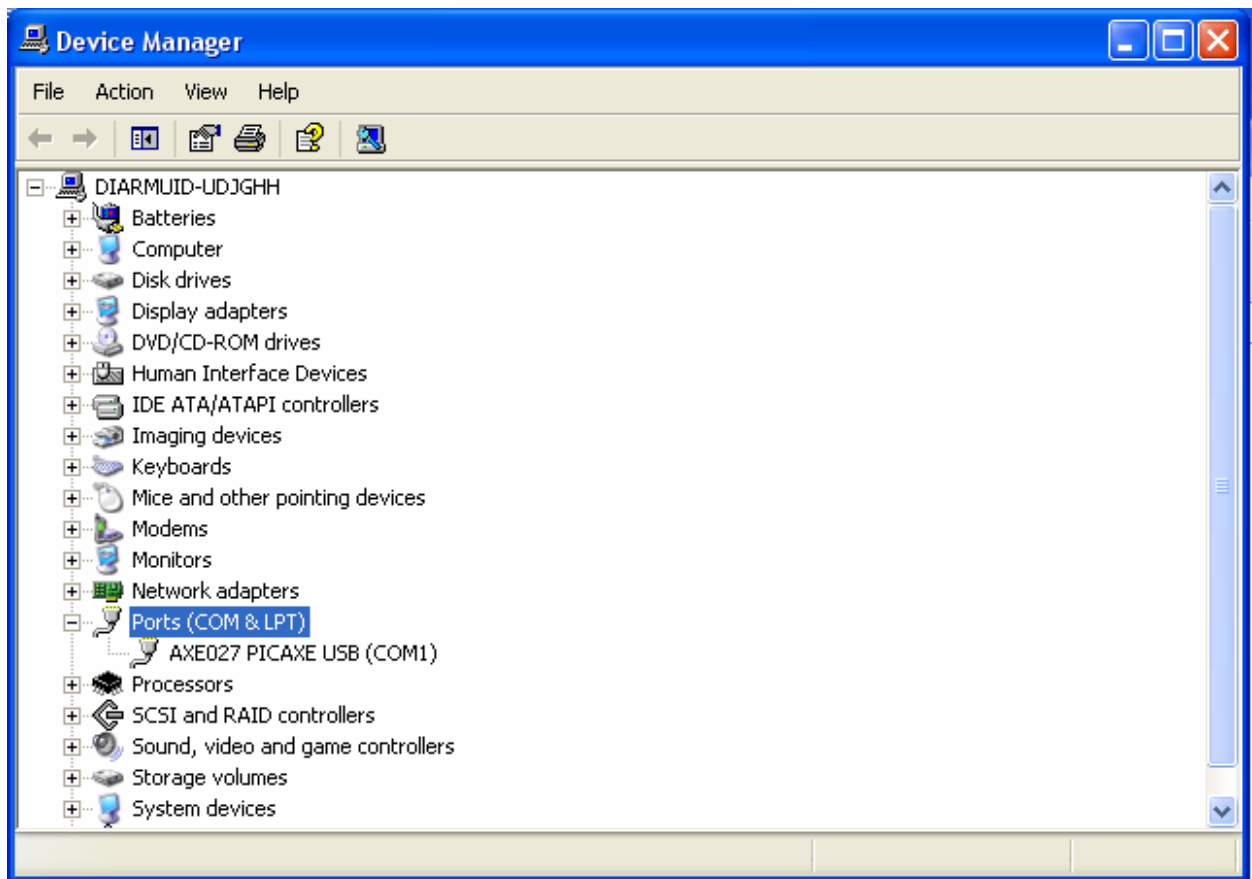
This process has just finished installing USB serial converter.

A few seconds after clicking on finish another Found New Hardware Box will appear, do not cancel this.

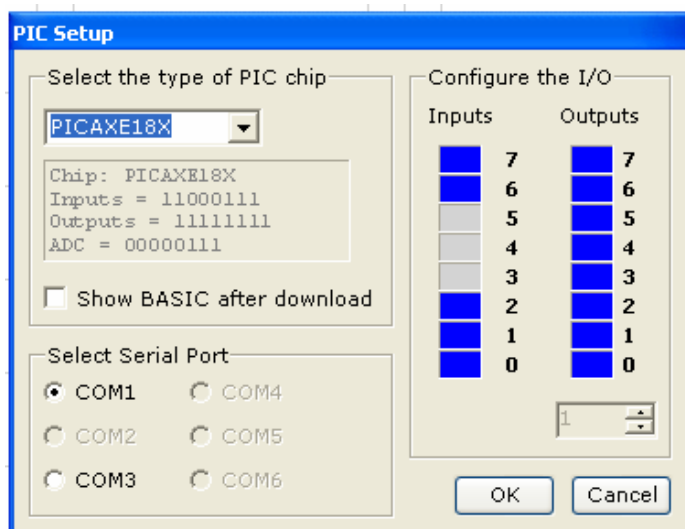
This set up is to install a USB serial port. To install this, repeat the exact same process as before.

Once these two pieces of software have been installed it is necessary to make sure that Logicator is using the correct Com Port.



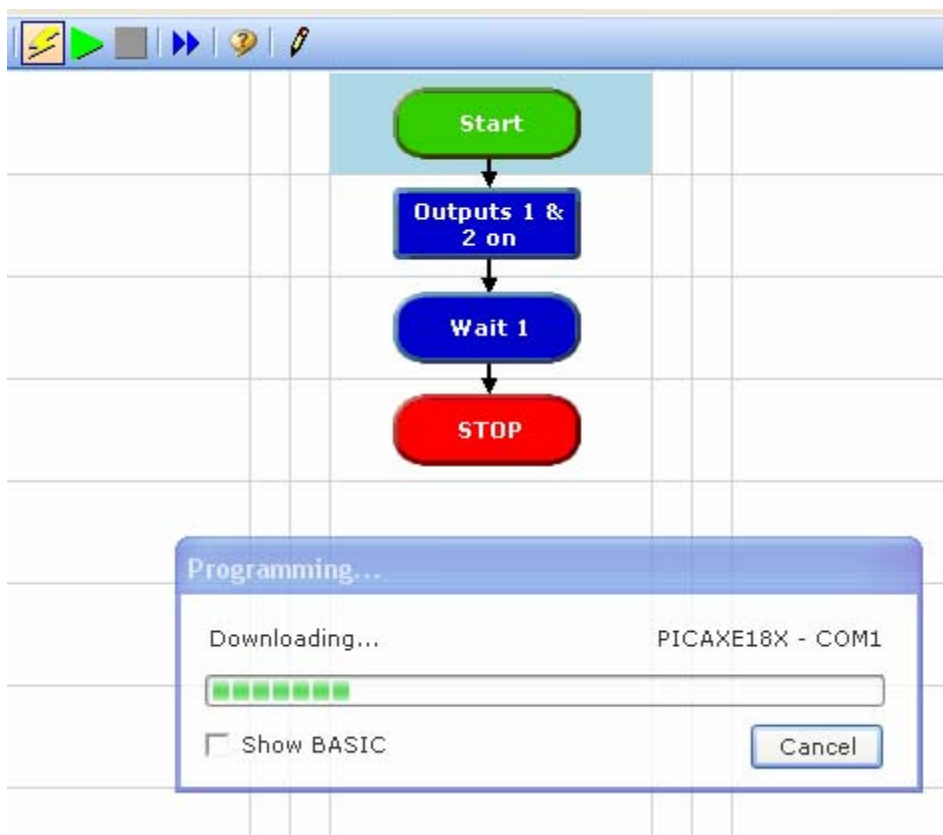


Click on the Ports (Com & LPT), this will show you what com the Picaxe cable has been assigned. If the com is between 1 and 6 simply return to Logicator and in the 'Pic set up box' tick the correct com.



Finally select the 'Advanced' button. Under 'Advanced' it is possible to assign the PICAXE to a com port manually. If a com port between 1 and 6 is free simply select it. If none of them are free, click on any one between 1 and 6. Do not worry about selecting an already assigned com, the computer will reallocate this com. Once you have selected the com click ok. In Logicator again go to Pic setup and tick the com port that has been just assigned.

To test to see if the device is operating correctly write a simple program in Logicator and upload it to the Pic chip.



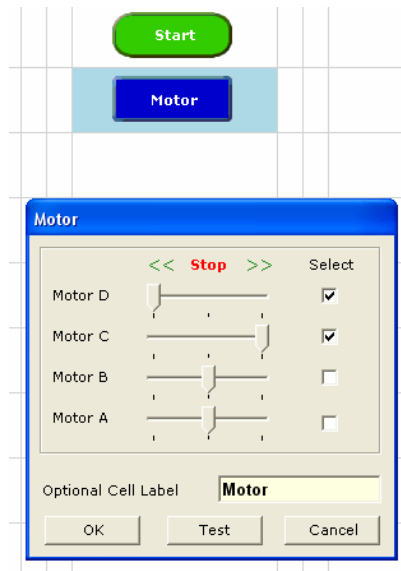


Figure 2- The motor block

The direction of the motor can be selected by moving the slider on the programming window. In Fig. 2 above Motor D is reversing, Motor C is moving forward.

It is necessary to tick the box to select which motors are to be activated. To stop the motor, move the slider to the centre.

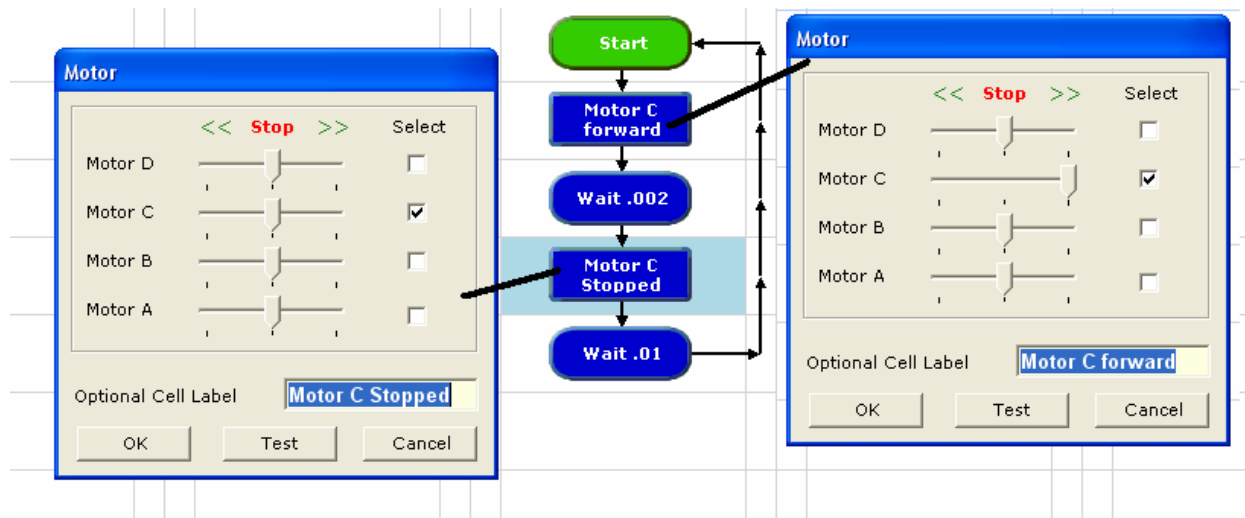


Figure 3 DC Motor speed control

It is not possible to control the speed of the motor by regulating the current flowing through it. The most convenient method is to literally start the motor and stop it before it can build up full speed. In Fig. 3 the motor is started for .002 sec and turned off for .01sec. In real time this means starting and stopping the motor 100 times per second.

Stepper Motor Control

The stepper motor is far superior to the dc motor in terms of control, resulting in a more expensive motor. The stepper motor gives continuous controlled movement with good torque. The degrees of rotation per step are determined by the motor manufacture. The stepper motors typically used in projects have step angles of 1.8° or 3.6°, meaning that it is possible to create 200 or 100 positions per revolution.

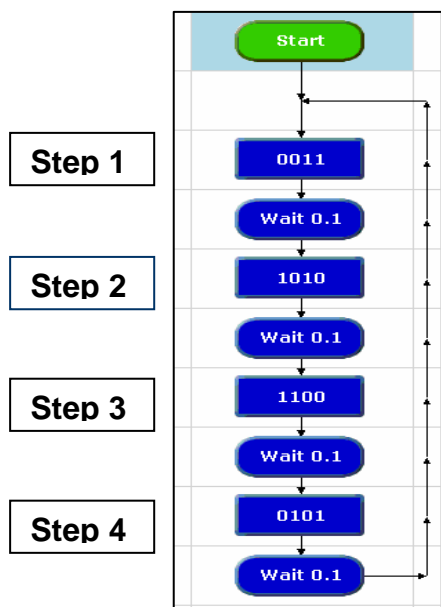


	Output0	Output1	Output2	Output3
Step 1	0	0	1	1
Step 2	1	0	1	0
Step 3	1	1	0	0
Step 4	0	1	0	1

Step sequence

Figure 5 Stepper Motor

Each motor comes with a table listing the required sequence for correct motor operation. A flowchart for continuous stepper movement is shown below. The table gives the sequence for 4 steps of 1.8°, it is necessary to repeat this sequence repeatedly to maintain rotation.



Wait value will determine motor speed.

Smallest wait value achievable is .005

Reversing the sequence will reverse the direction of rotation of the stepper motor.

i.e.

Step 1: 0101

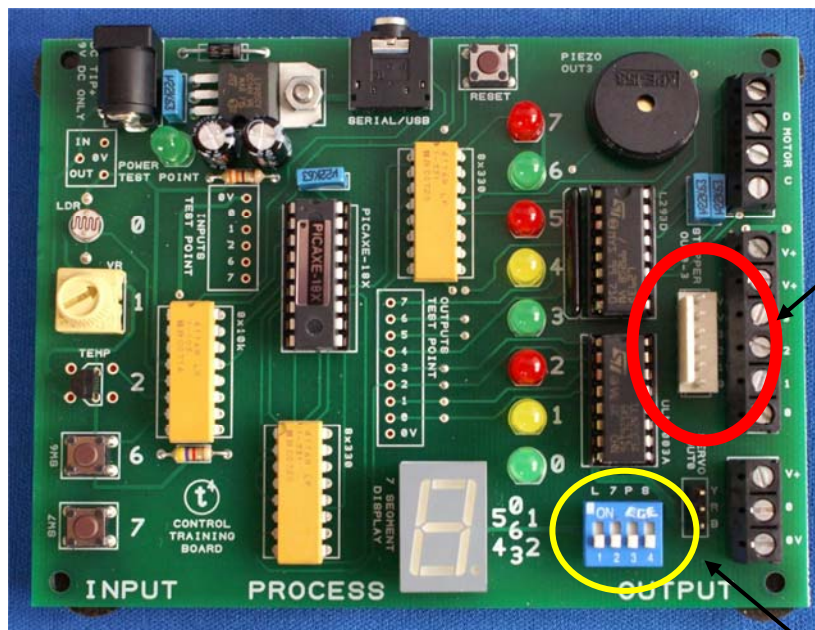
Step 2: 1100

Step 3: 1010

Step 4: 0011

To reverse the direction of the motor, the sequence of the steps is simply reversed. Instead of inputting Step 1, Step 2 etc, it becomes Step 1, Step 4, Step 3, Step 2, Step 1, Step 4the wait times remain the same.

The stepper motor can be controlled directly from the T4 Boards. The outputs that control the stepper are 0,1,2,3. The stepper also requires constant 0v and 5v. The white 6 pin plug will connect directly to the T4 Board.



Stepper motor output connection

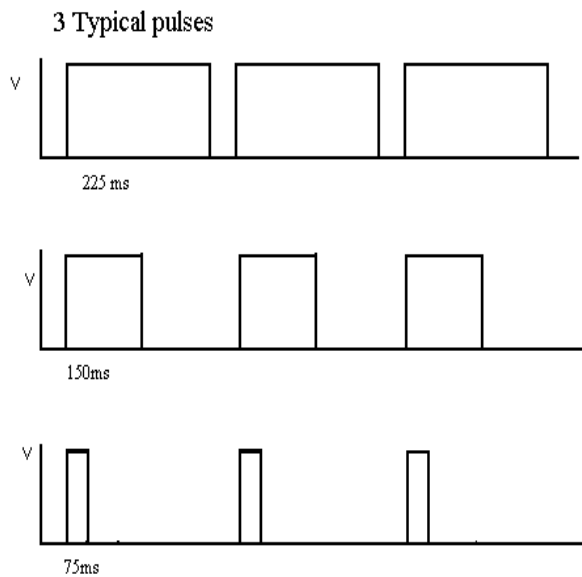
Ensure switch no. 4 (S) is turned on !

T4 training board

Remember for a 1.8° step:-

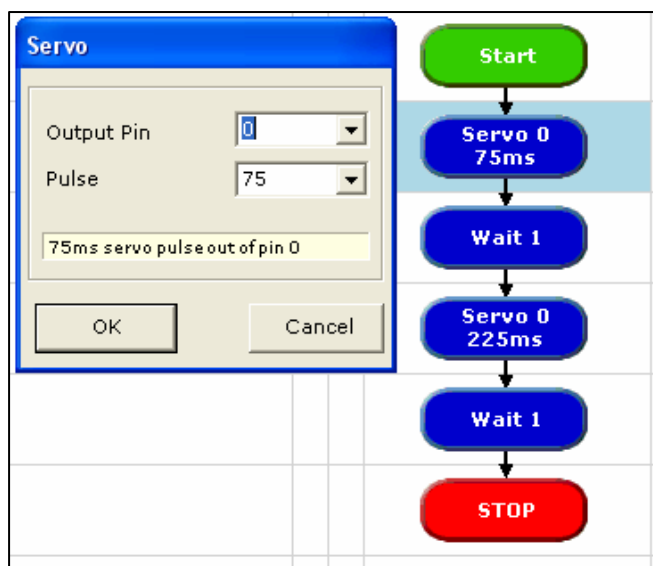
1 revolution = 360°

$$\frac{360^\circ}{1.8^\circ} = 200 \text{ steps} \quad \frac{200}{4} = 50 \text{ loops for 1 complete revolution.}$$



Servos work on a frequency sent from the PIC, this frequency is a voltage sent for a number of milli-seconds depending on what position the servo is required to be. In Fig. 8 three pulses are shown, 75ms, 150ms & 225ms. These are the starting, middle and ending frequencies of a servo. The range between 75ms and 225ms is 150, giving 1ms per degree of movement. In Logicator setting the *Servo Block* to 75ms will move the servo to 0° (home) and 225ms will take it to 150°.

Figure 8 Servo pulse signals



A simple exercise to move the servo between 0° and 150° can be seen in Fig. 9. The program starts with a servo block that moves the servo to 0°, this corresponds to a pulse frequency of 75ms. It is also necessary to set the output pin that the servo is connected to. On the T4 Board this is output 0.

Figure 9- A simple servo program

It is important to always place a Wait block after the servo block. This is to give the servo time to move into the desired position. In Fig. 9 the wait is 1 sec and this will give the servo time to complete its full rotation.

L.D.R. Input:

This program will use the LDR as the trigger switch which a car activates when it drives up to the barrier. When it drives over the LDR it should open the barrier. The barrier however should not close until it knows the car has pulled out from under it. As with most barriers it should open from a horizontal to a vertical position, effectively a 90° servo motion.

When the program (Fig. 11 below) is activated it makes sure the barrier is closed by moving the servo home with a 1 sec wait. It will keep the barrier closed until the compare block sees that the light level from the LDR falls below 80, signifying that a car has driven over the sensor.

Note the analogue inputs have a range from 0 – 255.

Complete darkness being 0 and total brightness being 255 in the case of the LDR.

Once the Compare block notes the change in the light level it opens the barrier 90°. The servo will hold this position until the compare block sees that the car has driven off the LDR, raising the light level. The program returns to the beginning closing the barrier.

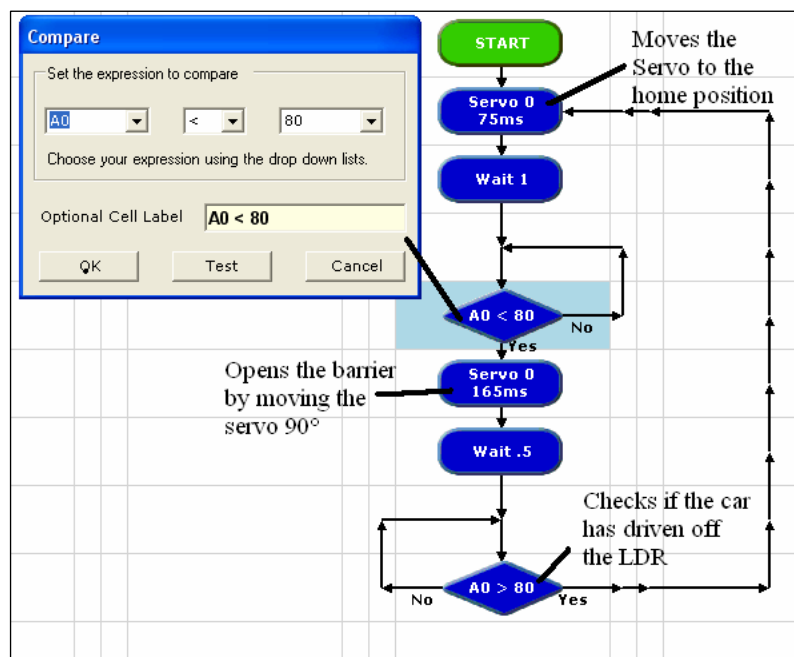


Figure 11 Car park

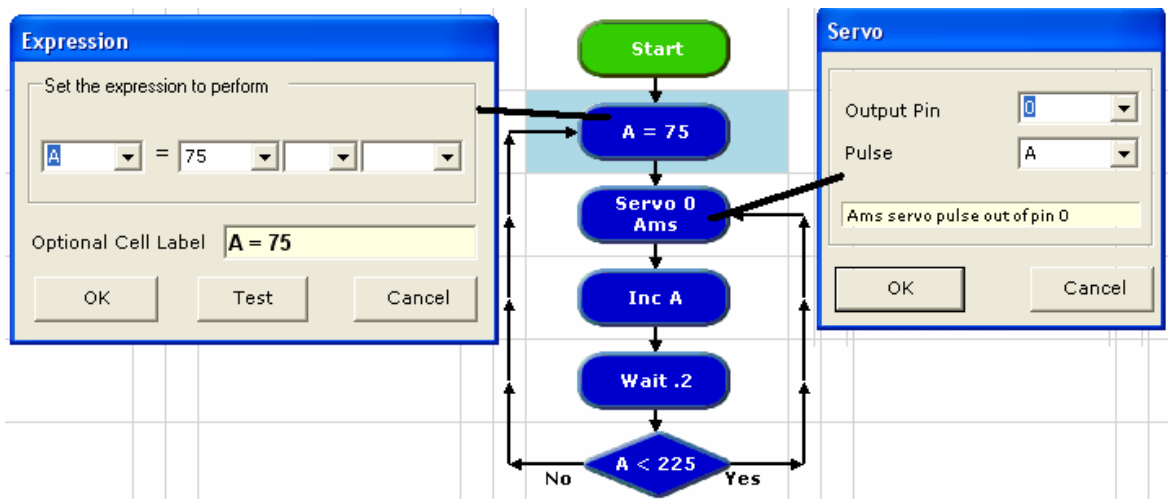


Figure 13 Servo- variable control

In the above program (Fig.13) the first block used is *Express*. This is used to tell the variable A it's value, 75 which is the start frequency of the servo. Note how the servo block next is not set to a pulse number but to the variable A. For this program the servo command is going to look in the variable A and move the servo to whatever value this is set to. It will then add one to value of A with the Inc command. The wait block is set to .2sec; this is the speed at which the servo will move each degree. The compare block at the end will keep the loop circling until the value of A is 225 the max value of the servo, it will then send the loop to the start where A becomes 75 again and the process starts again.

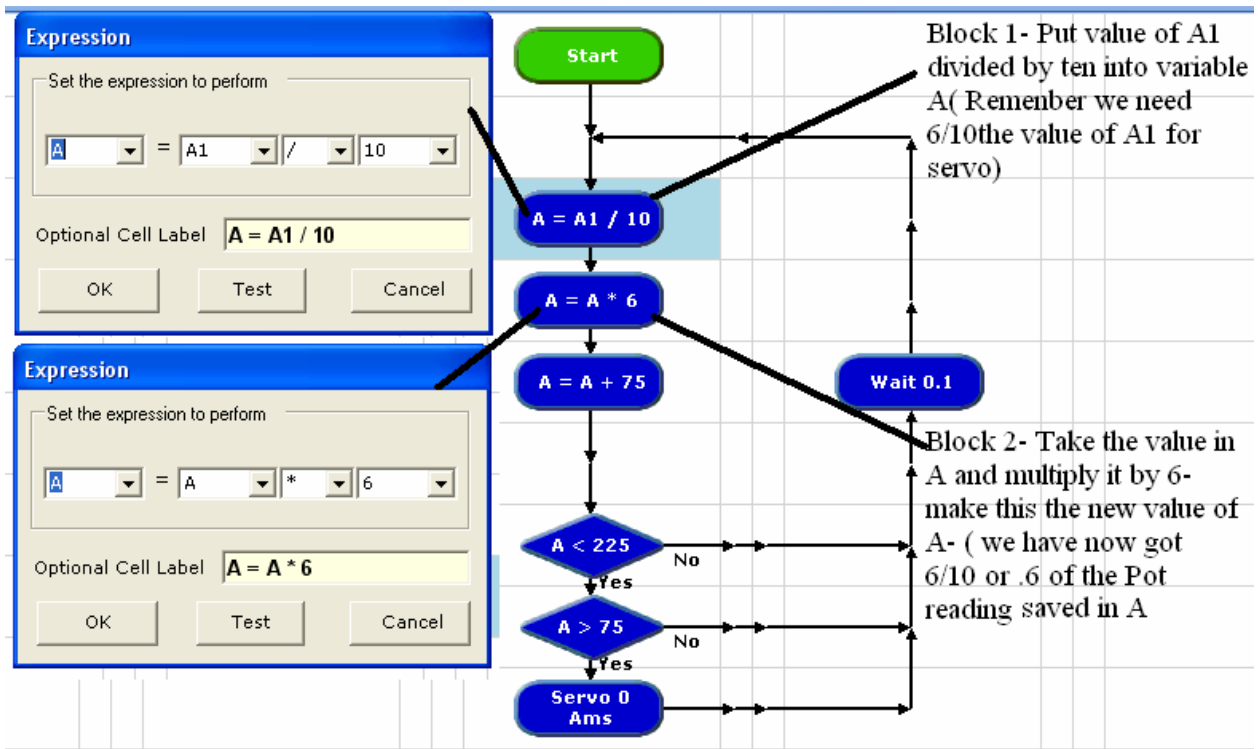


Figure 14- Potentiometer to Servo (1)

The first two blocks in the program are used to find a 6th of the potentiometer value. It is not possible to carry this out in one block by simply dividing the value of A1 by .6 as Logicator only deals in real numbers from 0- 255. The easiest way is to divide by 10 and multiply by 6.

The third block makes the final adjustment so the potentiometer and servo are synchronised. As discussed previously the potentiometer range starts at 0 and the servo starts at 75. This third block adds 75 onto the new potentiometer value in A and saves this to A. The value in A is now the value of the pulse frequency we send to the servo. The two compare blocks act as safety blocks in case our math calculations go slightly above or below the range of the servo.

Finally the servo block sends a pulse frequency to the desired output of the Pic (Output 0) with a wait of .1secs to allow the servo to move into position.

To create a Light Meter simple change the input A1 in the first block to A0.

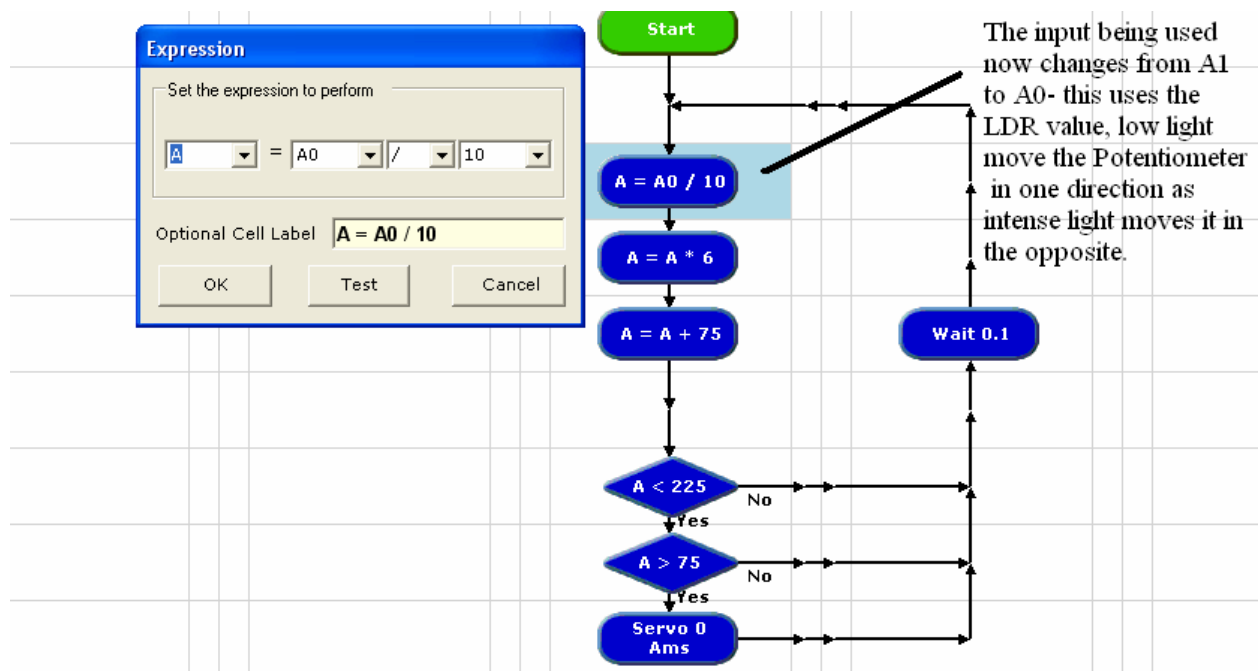


Figure 15- Light Meter

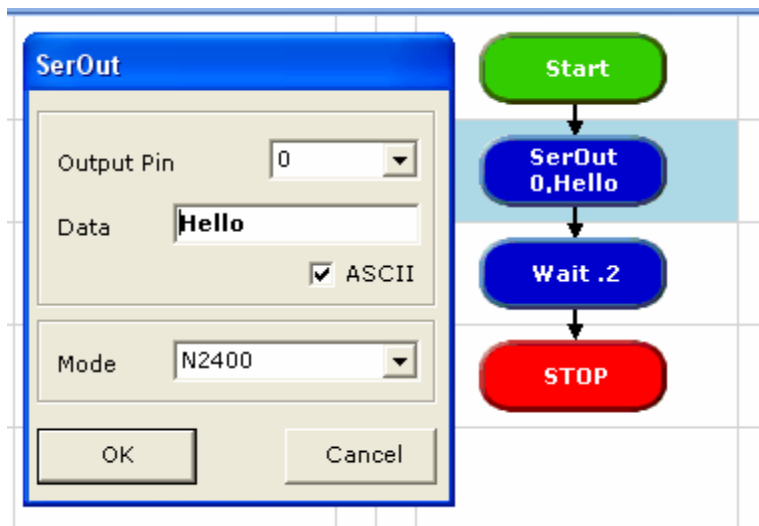


Figure 17- Basic display

The program in Fig 17 will display “Hello” on the display screen. It is important to note the settings on the Serout block. The Output pin is familiar at this stage and is set to pin 0. Once the ASCII box is ticked whatever is written in the data box will be displayed on the screen. For this specific display board N2400 has to be selected. This is the speed at which the board communicates with the computer. A small wait time must be added to allow time to send the signal. In the edition with three data lines simply ignore the two unused lines. If the Stop block is deleted and the program looped, the display will become a screen full of “Hello”. The screen has not been told to clear itself, just keep writing hello.

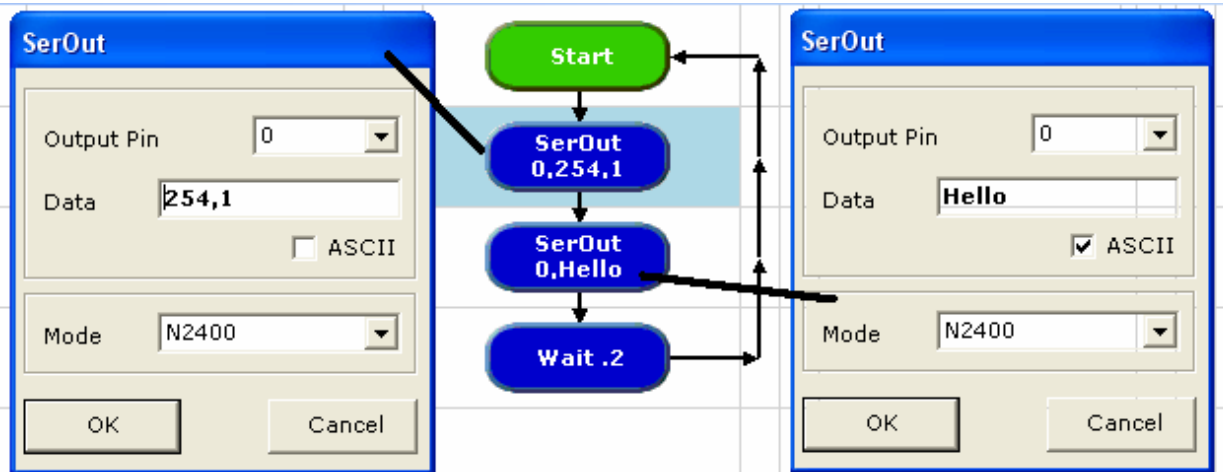


Figure 19 Using Ascii code

This program has an extra block that is placed at the start of the program. Note that the ascii box is not ticked when sending code. The next block then sends the text message. In the Logicator edition with three data line this program could be written with a single Serout block, the ASCII box would not be ticked in the first data line and ticked in the second.

The LCD display can be also used as a counter or to display the values of the analogue inputs which can be helpful to correctly set values in compare blocks. The 'Debug' command, previously covered, will also provide the same result.

