



TECHNOLOGY
SUBJECTS
SUPPORT
SERVICE

Leaving Certificate

Technology

Applied Control Technology

Using PIC LOGICATOR for Robotic
Control

Contents

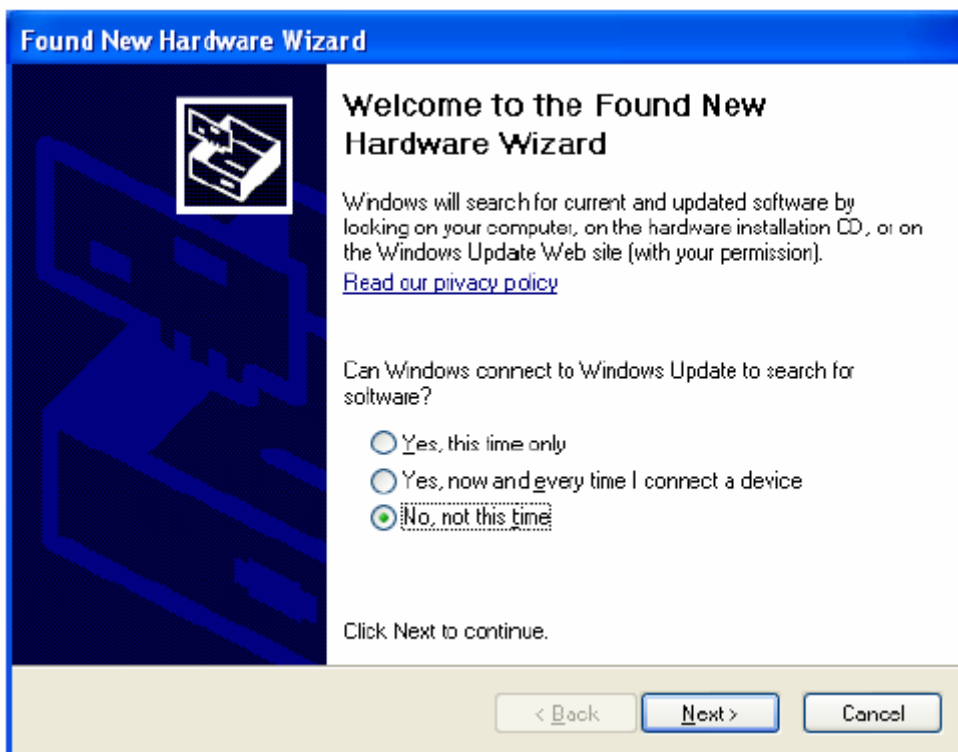
USB Cable Drivers Installation.....	3
Simple DC Motor Control	11
Stepper Motor Control.....	14
Rotation Control of Stepper Motor	14
Servo Motor Control.....	17
Simple Car Park Barrier Operation	19
LDR Control of a Servo Motor.....	20
Reading a Variable to Control a Servo.....	21
Use of Analogue Inputs to Position a Servo.....	23
Use of an LCD Display.....	27

Correct Communication Between the PC and Pic Board.

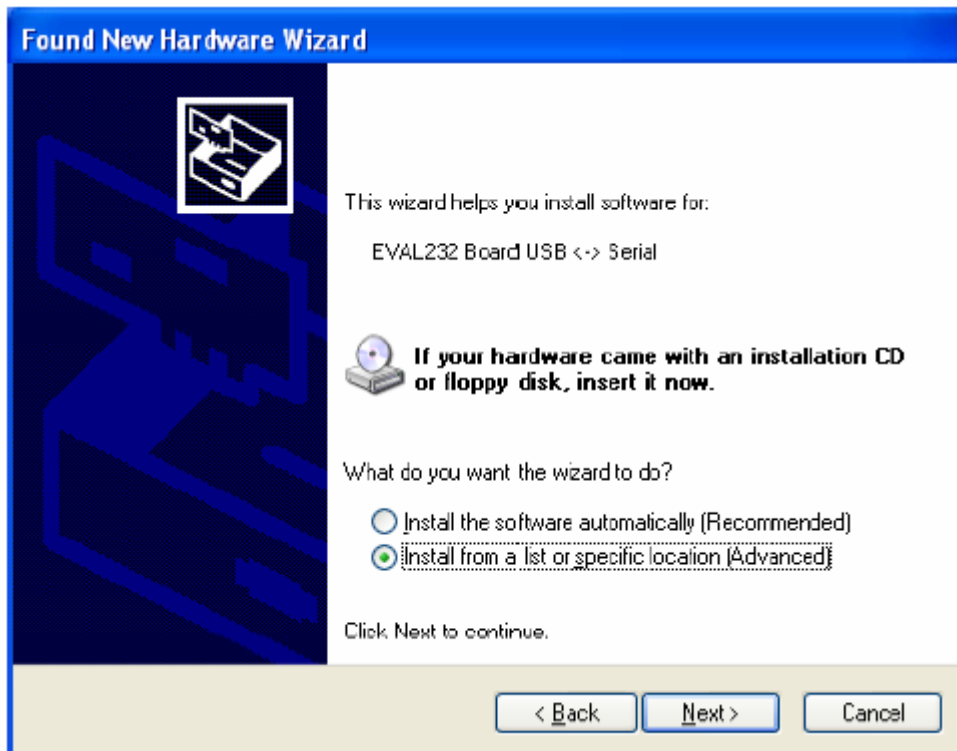
When a laptop or PC is connected to Picaxe for the first time it is necessary to install the driver software that allows the computer see the Pic board. This is standard practice, similar to setting up a new printer or other external hardware. There are some steps that should be followed to ensure trouble free operation.

Firstly the driver software for the Picaxe cable comes on a separate folder that can be obtained from the PIC Logicator CD or from the t4 website.

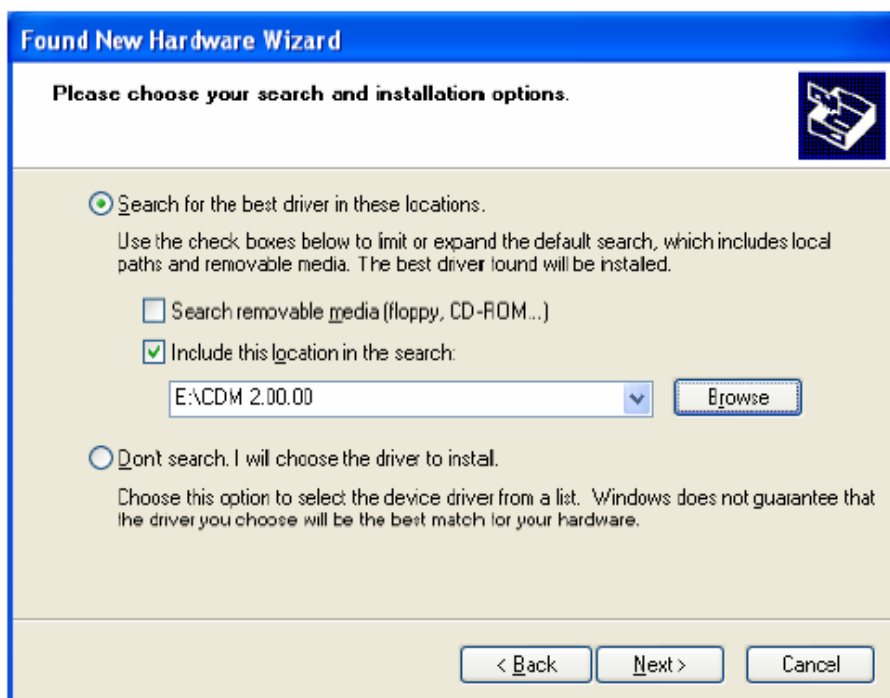
When the cable is plugged in follow the onscreen guide.



Click the “Not this time” box and select next



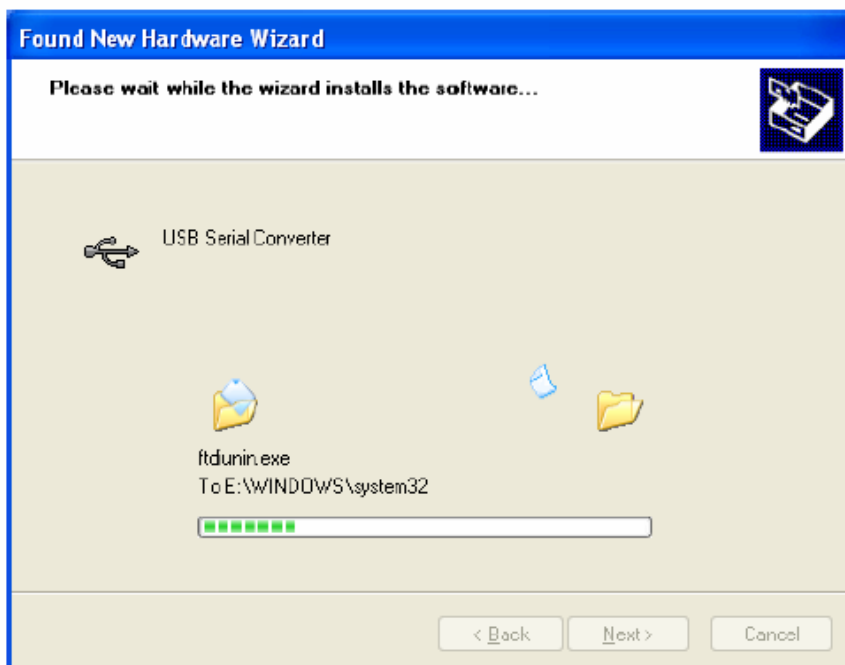
In this box, select installation from a specific location and select next

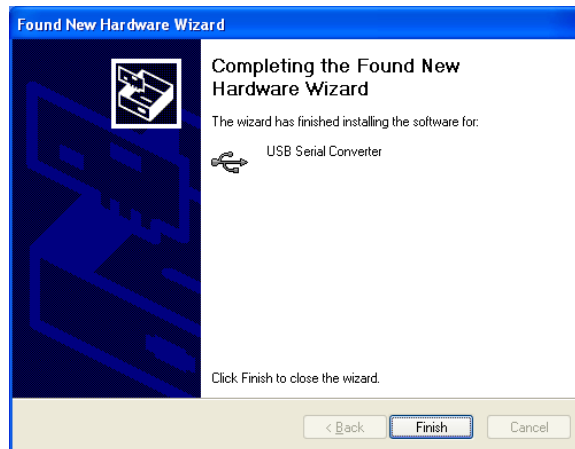


In this box check the box as shown and then press the browse to find where the driver software for the USB cable is located, select and press next. If the command box below appears click the Continue Anyway button



The following screen will appear followed by the finish screen, select 'Finish'.



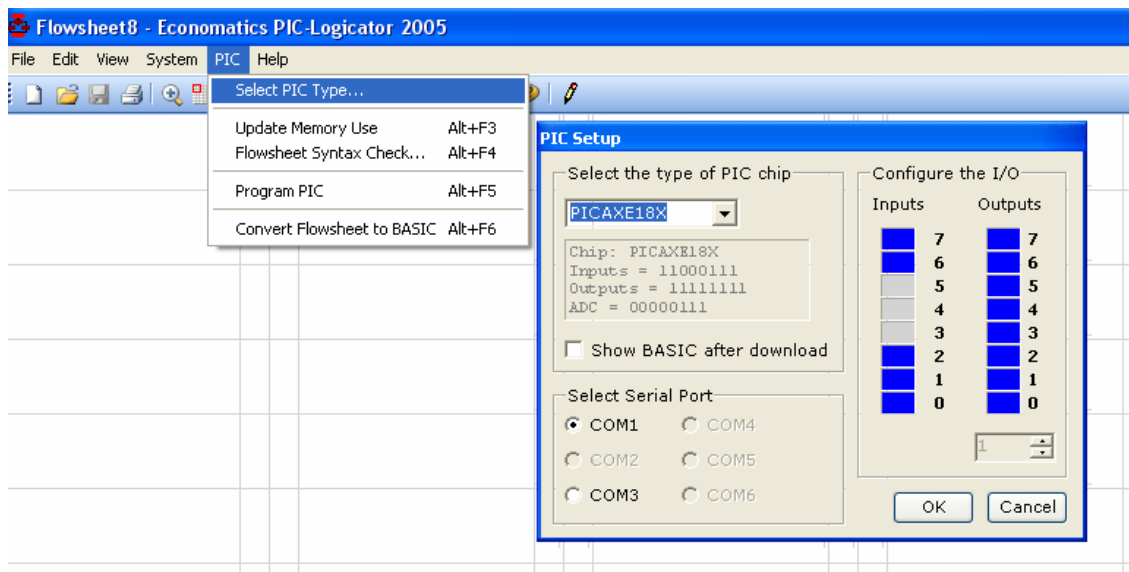


This process has just finished installing USB serial converter.

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.

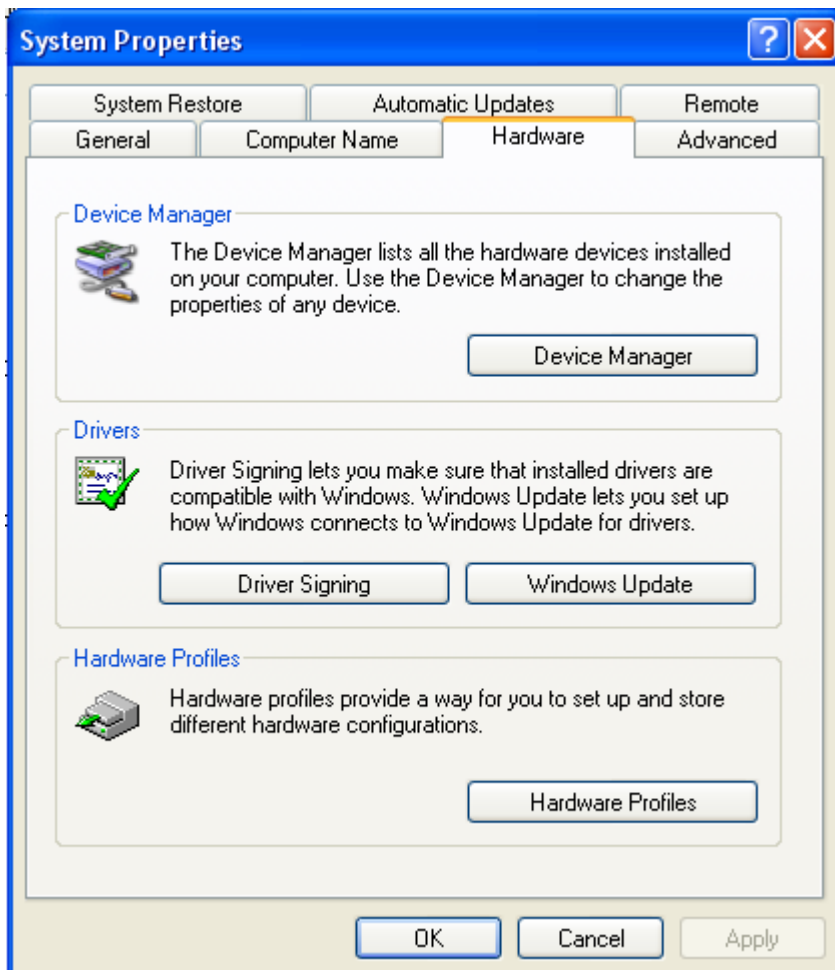
Once these two pieces of software have been installed it is necessary to make sure that Logicator is using the correct Com Port.



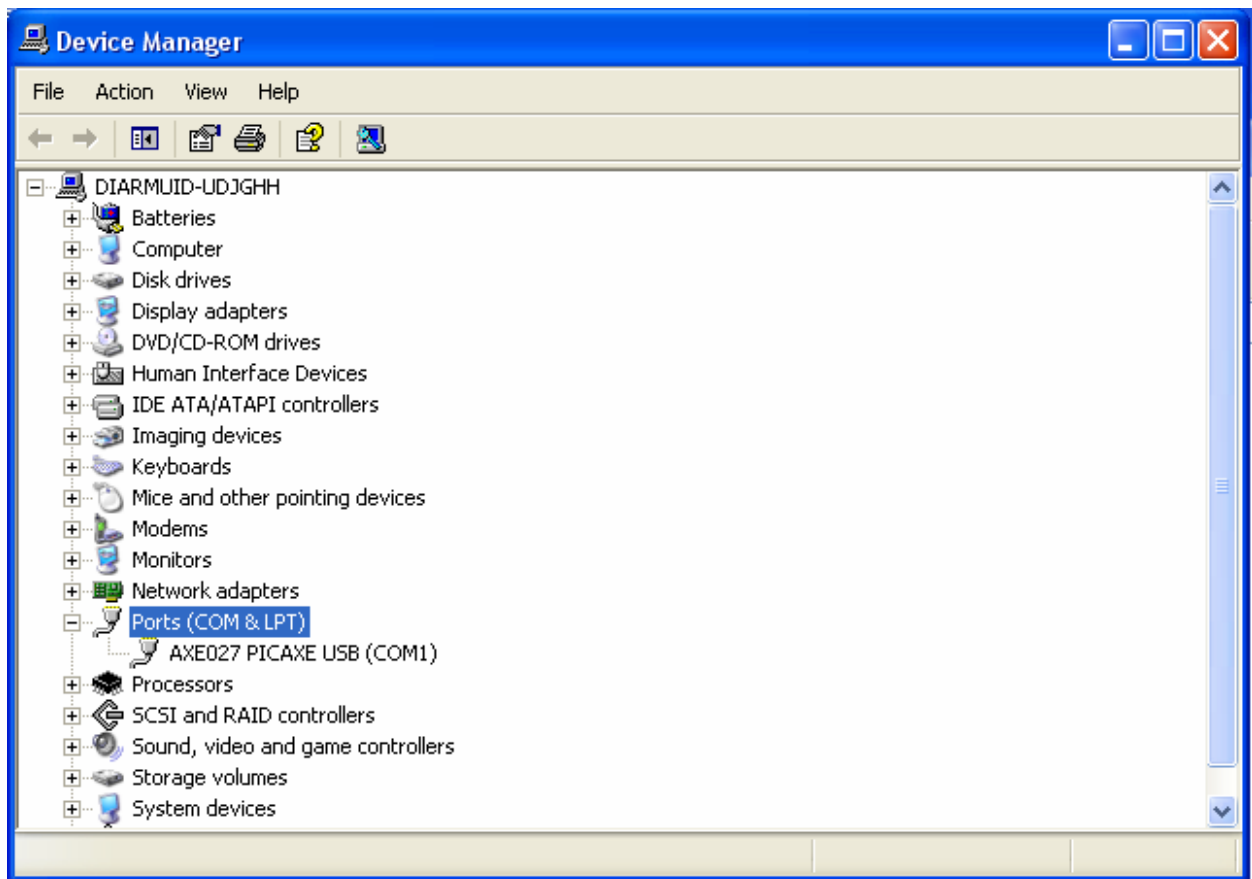
In Logicator there are six available com ports. Depending on your computer set up the software you just installed may not have been assigned to any of these six. If this is the case the user must manually select the com port by carrying out the following.

On your computer select **Start => Control Panel => System**

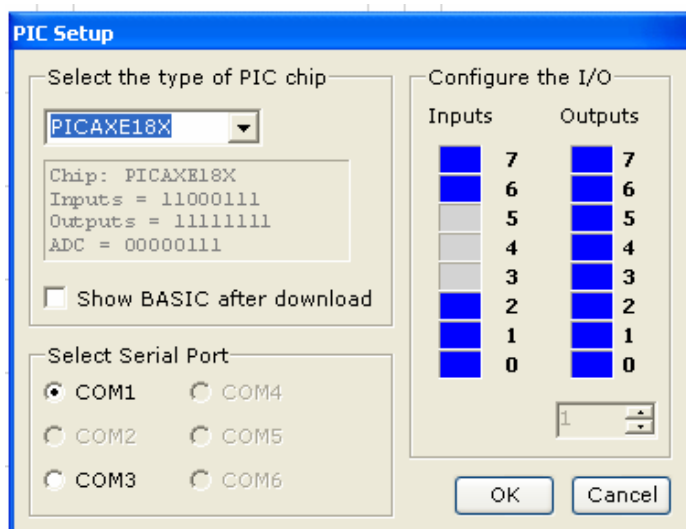
Your should now be at the following window



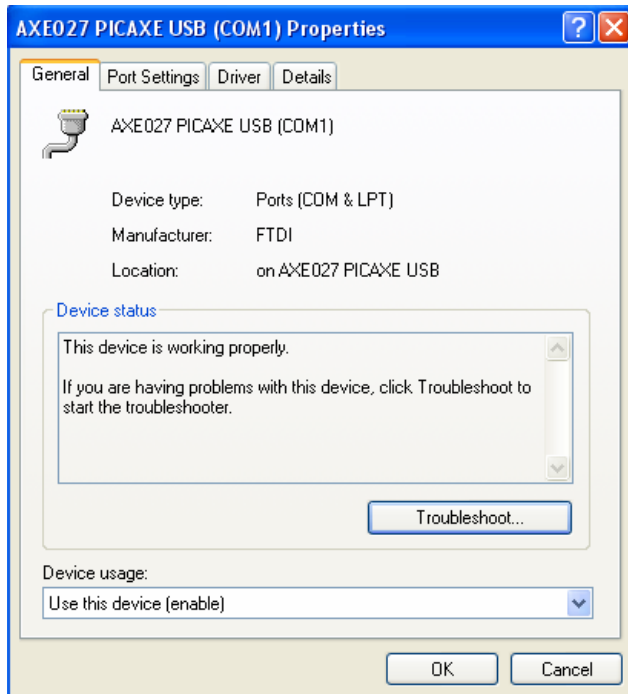
Select the **Hardware** tab, then click **Device Manager**



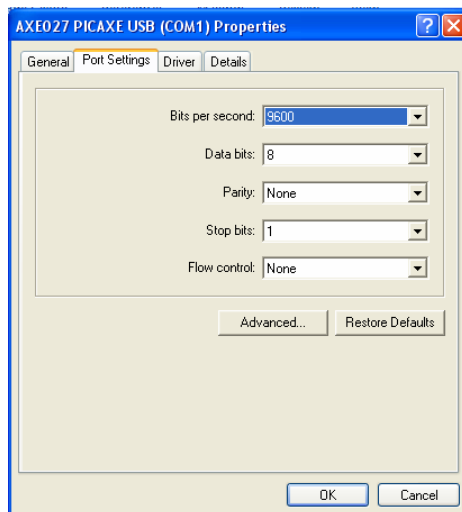
Click on the Ports (Com & LPT), this will show you what com the Picaxe cable has been assigned. If the com is between 1 and 6 simply return to Logicator and in the 'Pic set up box' tick the correct com.



If the com port number is higher than 6, there are a couple of more small steps to follow. In the Device Manager under Ports (Com & LPT) double click on Axe027 PICAXE and the following window will appear

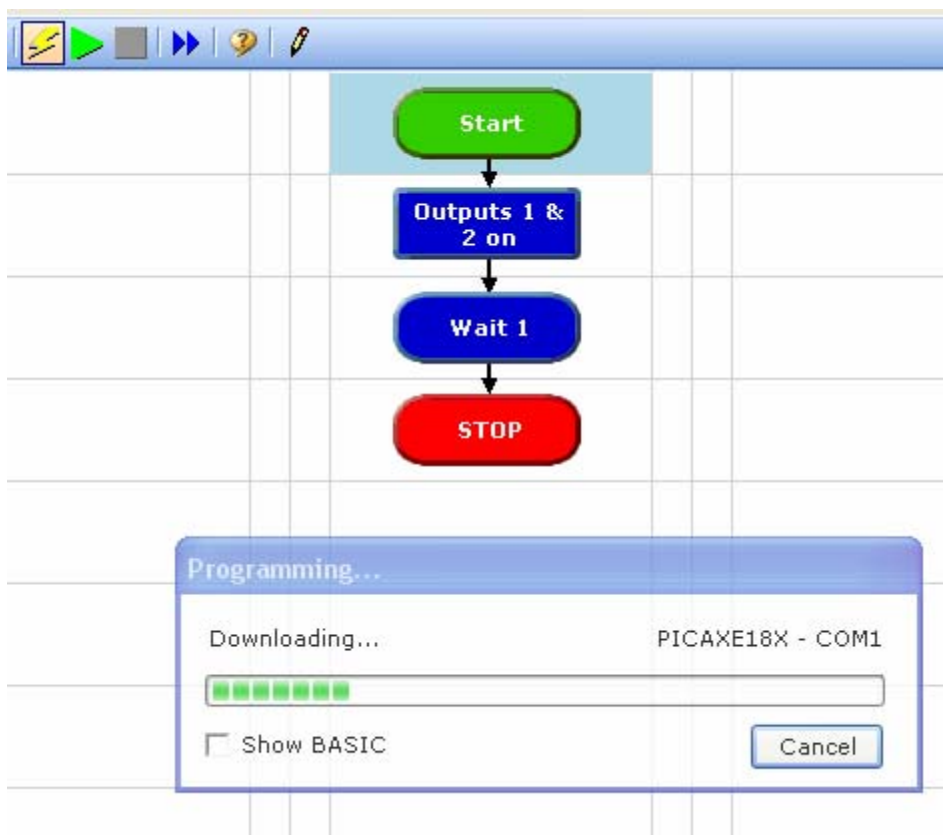


Select the Port settings tab



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.



Simple DC Motor Control



Figure 1 DC motor

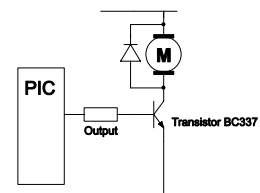
DC motors offer the cheapest option when seeking to create rotary motion. They are suitable for projects for a number of different reasons; they are compact compared to a stepper motor and offer continuous rotation unlike a servo. However DC motors require a relatively large current flow, they are high speed and low torque which can affect their choice in project work. Their speed can be altered either through gearbox reductions or the correct programming using PIC control.

It is important to note that DC motors should never be wired directly to the PIC output due to the PICs limited current flow. It is necessary to control the DC motor using transistors or an L293D transistor chip.

DC Motor – Programming

Within Logicator a motor can be switched on and off for a desired length of time. Its direction and speed can also be specified. However slowing the speed of a motor in Logicator will decrease the torque of the motor.

Fig. 2 below shows the motors programming block, giving an option for the control of up to four motors. This corresponds to the 18x chip with 8 outputs. Each motor requires 2 outputs to allow for changing the direction. The T4 training boards allow for the control of 2 motors, through outputs 4&5 (Motor C) and 6&7 (Motor D). On the T4 board it can be noted that the outputs do not come straight from the PIC chip but through the black transistor chip (L293D).



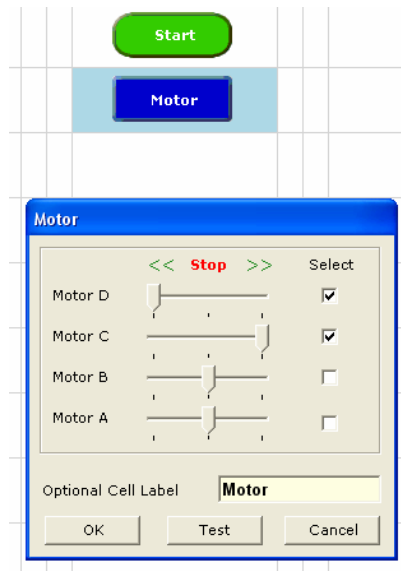


Figure 2- The motor block

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.

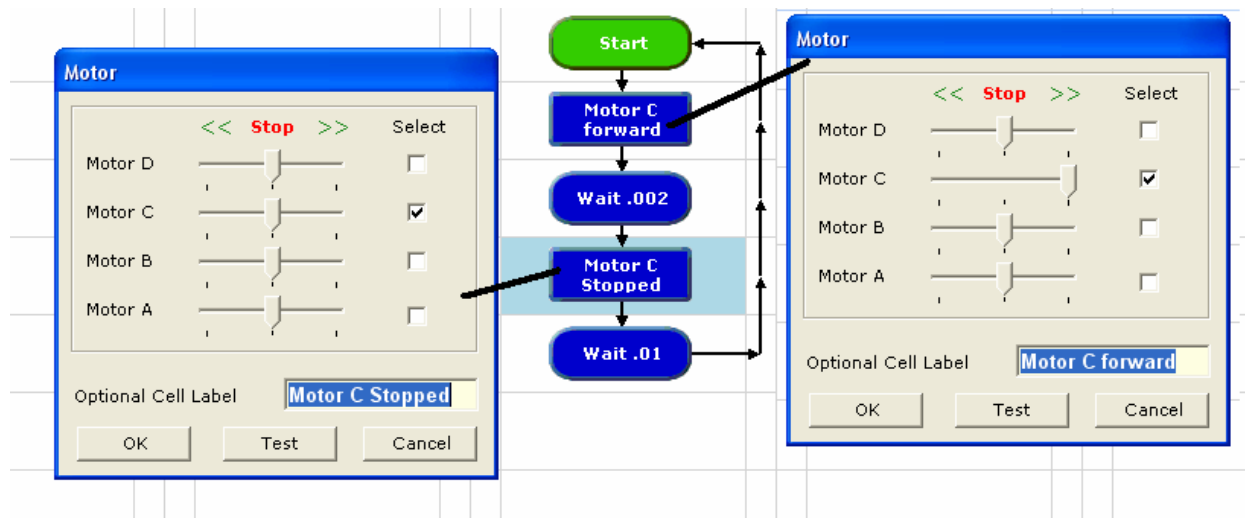


Figure 3 DC Motor speed control

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.

By increasing and decreasing the relevant wait blocks the motor will slow and speed up. This can significantly reduce the rpm of the motor; the output torque is also decreased meaning that the use of a dc motor to drive wheels etc. with varying speed is not ideal.

A suitable project that can be completed on the T4 board is a washing machine simulation.

This program uses four *Procedures*; Wash forward, Wash back, Spin and Dry. To switch between each procedure (Spin change and Dry yes/no) the digital inputs 6&7 are used. As with all modern washing machines the wash cycles alternate between spinning forward and backwards and rotate at far less RPM than the Spin cycle. In the washing machine program in Fig. 4 the speed of the motor has been slowed within the Wash forward, Wash Back and Dry cycles as seen in Fig. 3. The Spin cycle lets the motor run at full speed for 3 secs. The only difference between the Wash Fwd and Back procedures is the direction of motor C.

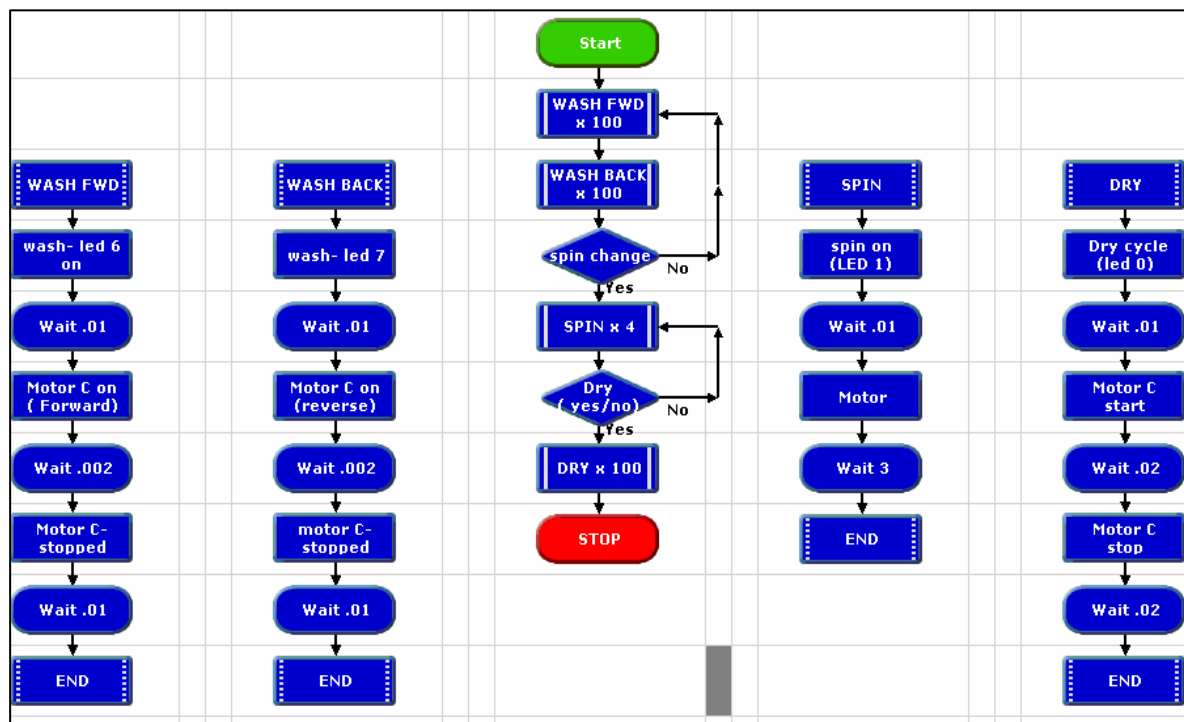


Figure 4 Washing machine cycles

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.

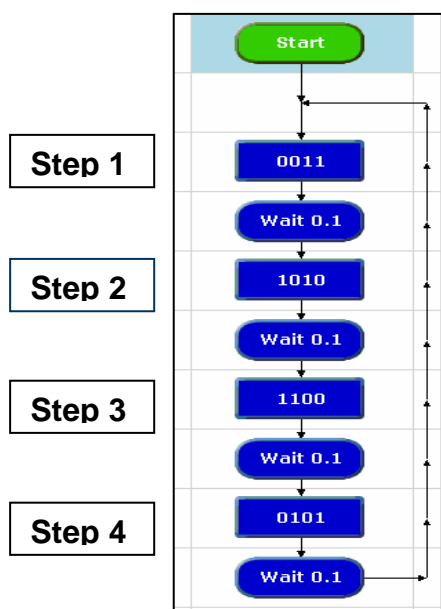


	Output0	Output1	Output2	Output3
Step 1	0	0	1	1
Step 2	1	0	1	0
Step 3	1	1	0	0
Step 4	0	1	0	1

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

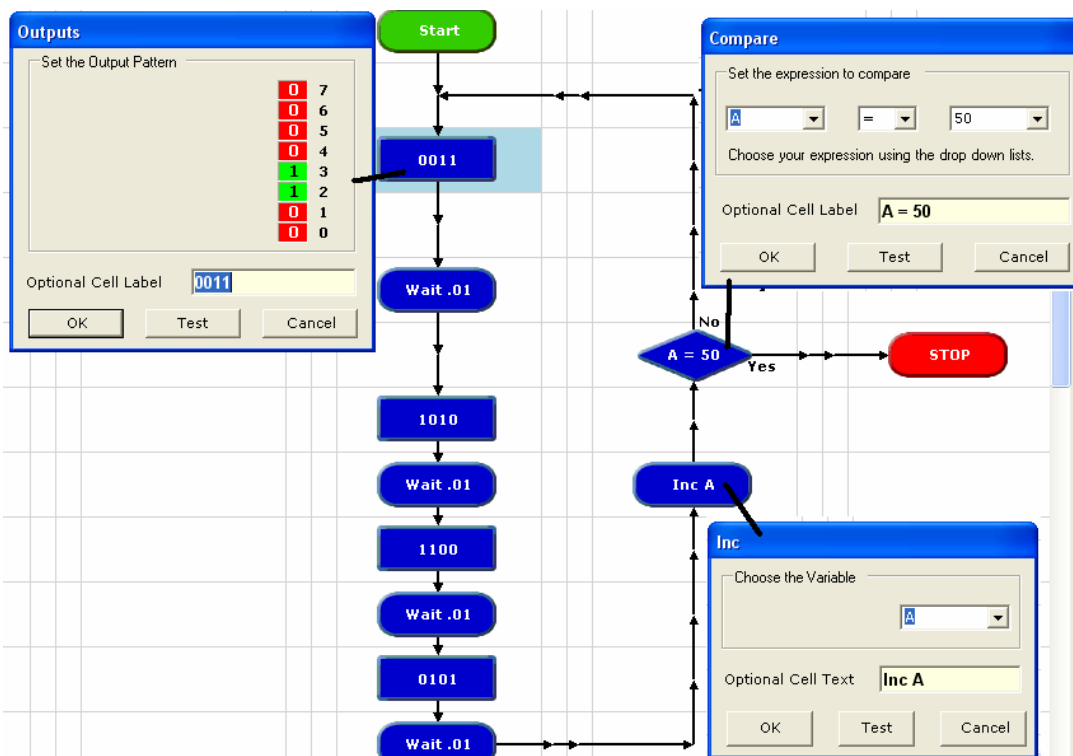
Step 2: 1100

Step 3: 1010

Step 4: 0011

To control the rotation of a stepper motor:

In Fig. 6 the stepper motor completes one revolution. Step 1 from the table is 0, 0, 1, 1. The sequence is entered into an output block as shown. This is followed by a wait block. **The length of the wait determines the speed of rotation, the shorter the wait the faster the motor.** The next output block contains the sequence for Step 2 followed by a wait and so on for steps 3&4. The final segment of the program deals with keeping account of the number of steps completed. This program wanted the motor to complete 1 rotation. If the Step angle is 1.8° then it will take 200 steps for one rotation. One complete loop of the program is only four steps; therefore it will take 50 program loops to do 1 rotation. The *Inc* block adds one number to the variable A. The variable A is like a Letter Square, each time the loop is complete the *Inc* command adds one unit to the Letter Square. The compare block then looks into the letter square and sees how many units are there. If the number is less that required it allows the loop to continue, if it is equal to 50 it sends the program the other way towards the stop block.

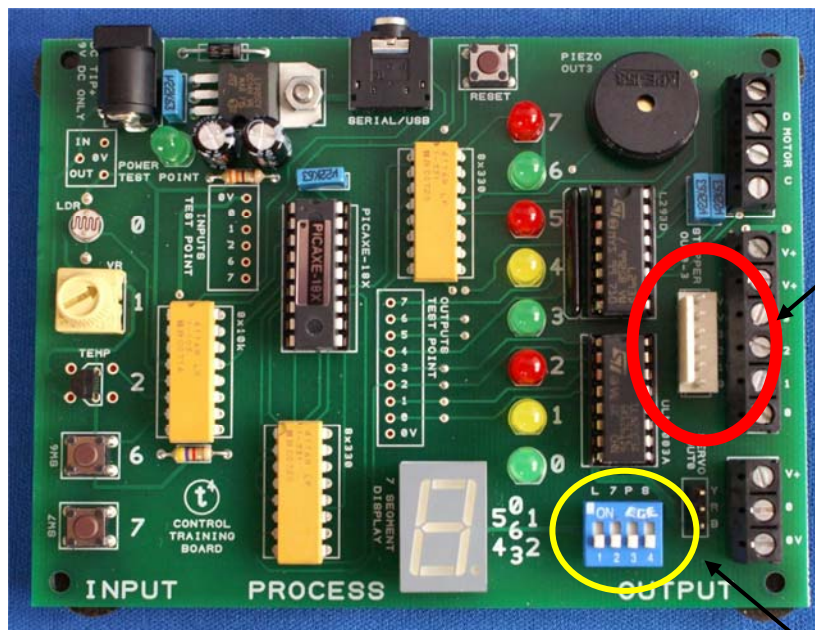


6 Stepper motor control

Figure

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°

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

Servo Motor Control

Servo motors have many advantages over the stepper motor and perhaps one major disadvantage. A servo can be positioned to specific angular positions by sending the servo a coded signal. As long as the coded signal (pulse) exists on the input line, the servo will maintain the angular position of the shaft. The servo, however, is limited to a range of motion typically between 0° and 90° to 180° and is not capable of continuous rotation due to its mechanical construction.

The servo provides an excellent holding torque; it is also a closed loop system which ensures that the servo will always be in the correct position. If it finds itself away from this position it will automatically correct this. The servo is the main actuation device in many radio controlled devices and robots.

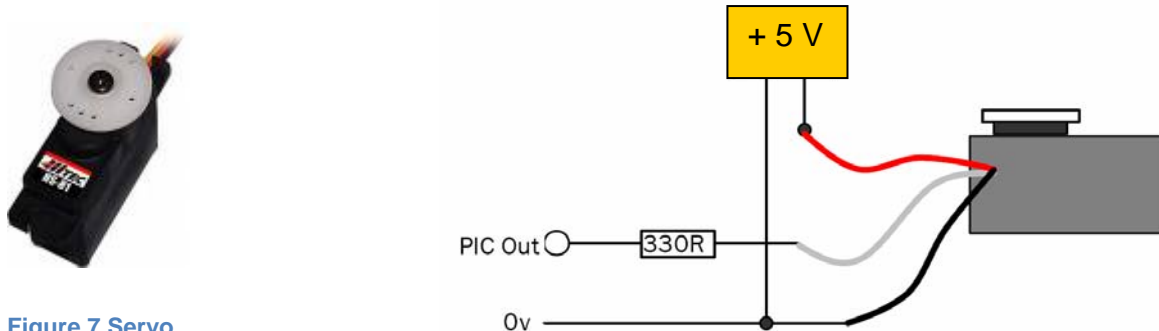
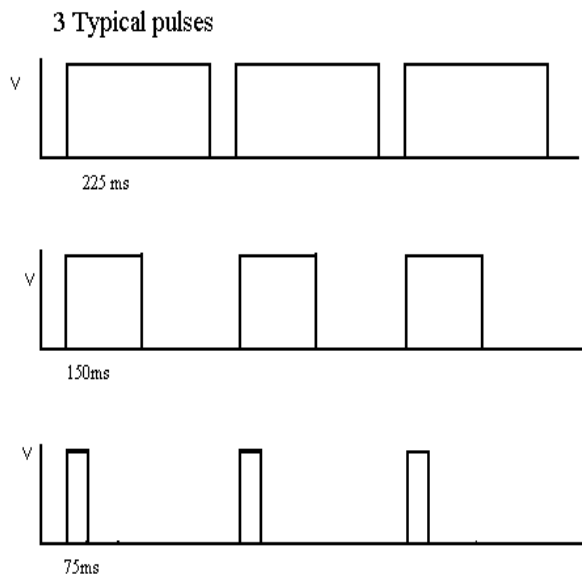


Figure 7 Servo

The general servo motor used in conjunction with PIC control has a range of motion from 0° to 150° . It is possible to control the movement degree by degree using Logicator.

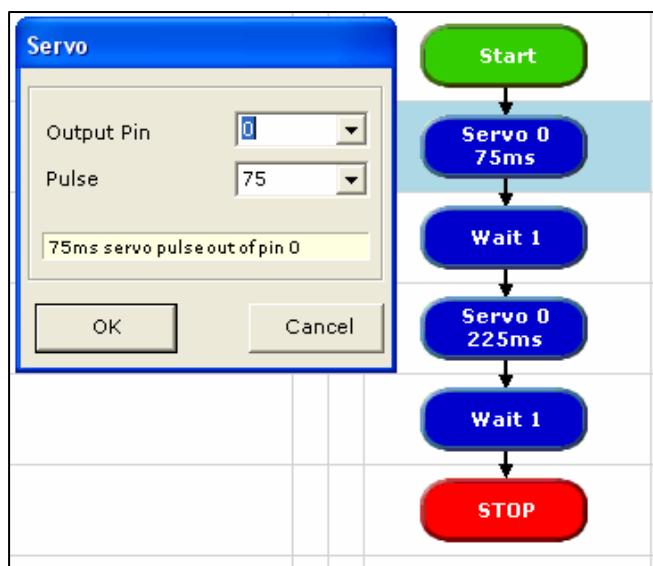
Servos have three wires connected to the control board, red (5v), black (0v) and yellow (control). Unlike the DC motor or the stepper, **the control wire of the servo must be connected directly to the output leg of the PIC**. The servo is sent a set frequency for a given position and needs direct contact with the PIC.

The T4 boards offer control of a single servo. This three pin connection can be found next to the blue switch bank on the bottom right-hand corner. As stated on the board the servo is using output 0 from the PIC. It is also necessary to have the Blue switch 4(S) in the on position. The servo wire is connected to match the labelling (YRB) on the board.



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

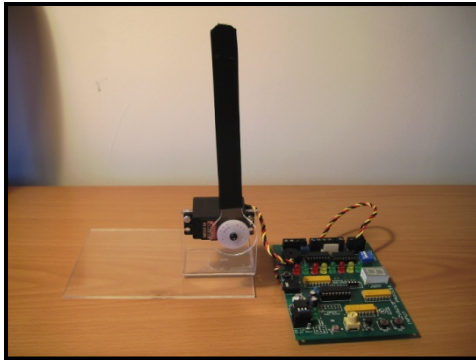


A simple exercise to move the servo between 0° and 150° can be seen in Fig. 9. The program starts with a servo block that moves the servo to 0°, this corresponds to a pulse frequency of 75ms. It is also necessary to set the output pin that the servo is connected to. On the T4 Board this is output 0.

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.

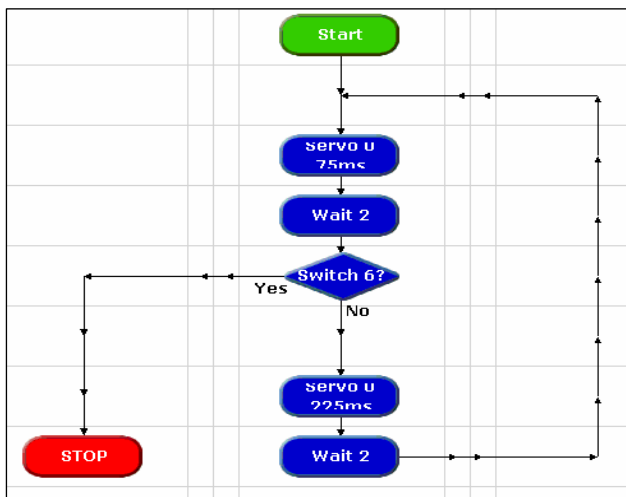
Simple car park barrier operation



The following programmes introduce the use of

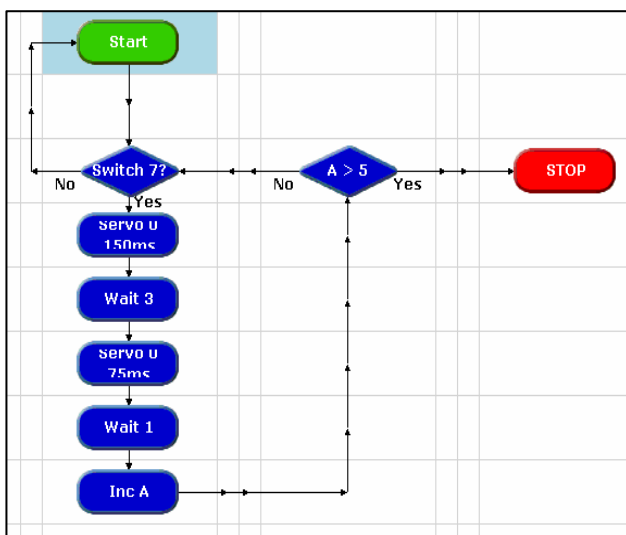
- Switch input
- Light Dependent Resistor (LDR)
- Reading a variable to control a servo motor.

Switch input:



This program responds to a digital input from switch 6 on the T4 board.

The barrier continues to operate until 'Switch 6' is pressed. The barrier should then stop in an closed position.



Can you establish how the car barrier will behave if the program shown in Fig. 10 is used?

Figure10

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.

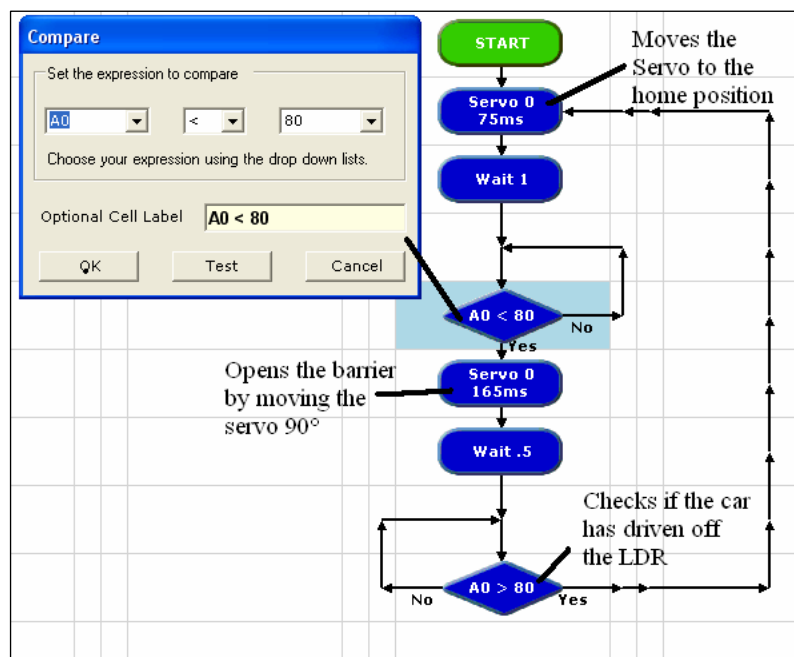


Figure 11 Car park

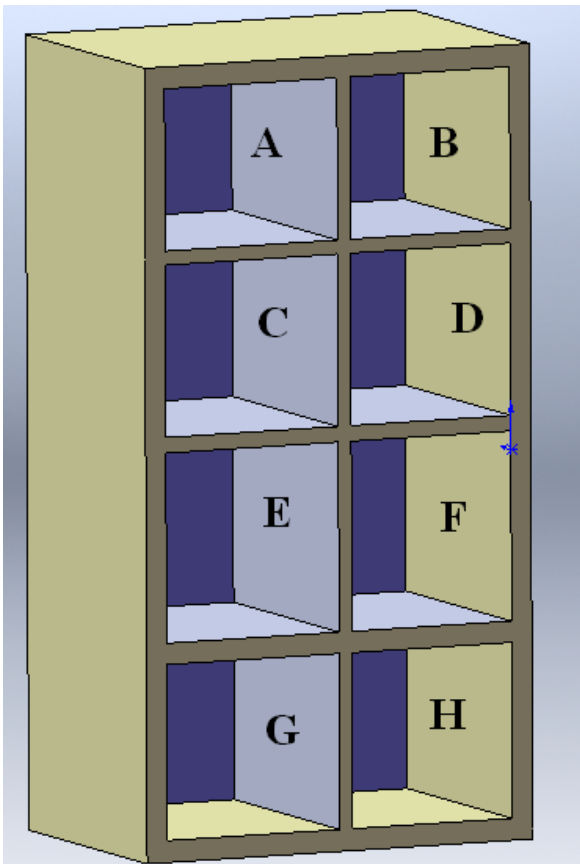
Reading a Variable to control a servo motor:

Figure 12 Variables as storage spaces

Logicator uses variables like a letter square in an office (Fig. 12). There are 8 variables from A-H available to store information. During a program a variable can be given a value (*Express*), have this value added to (*Inc*) or taken from (*Dec*). Physically this is like adding marbles into the letter square, adding more or taking them away. When required the number of marbles can be counted and the final number used.

In the next program the variable A will control the movements of the servo. It will move the servo from 75ms- 225ms in steps much like the second hand of a clock.

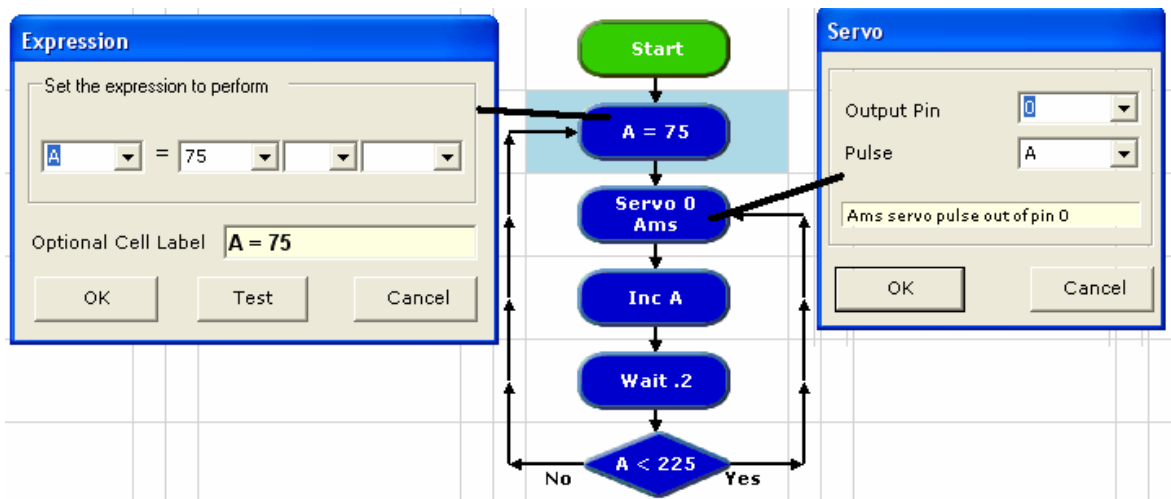


Figure 13 Servo- variable control

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.

Using Analogue Inputs to Position a Servo Motor

This section will deal with linking the reading from an analogue input and adjusting the servo output accordingly. It is important to note that the analogue inputs have a range from 0 to 255 and that the servo as stated previously has 75 to 225. This means that it is not possible to link one step in the input directly to one pulse of the servo. A practical example of this would be to get the servo to start from its home position (75ms) and move to its furthest point (225ms) as the Potentiometer (A1) moves from left to right (0 - 255). They should both reach their end points at the same time.

It is necessary to do a small mathematical calculation to allow the movement of the servo to operate smoothly with the analogue input.

- The range of the analogue input is 255 (0 – 255)
- The range of the servo is 150 (75 - 225)
- 255 steps of the input must take the servo 150 steps
- $150/255 = .588$ (.6) => close to 1 decimal point
- Every step the Pot takes must be multiplied by .6 to correspond to the servo because $255 \times .6 = 153$ (150)

Fig 14 shows the program that will achieve correct alignment between the Potentiometer and the servo.

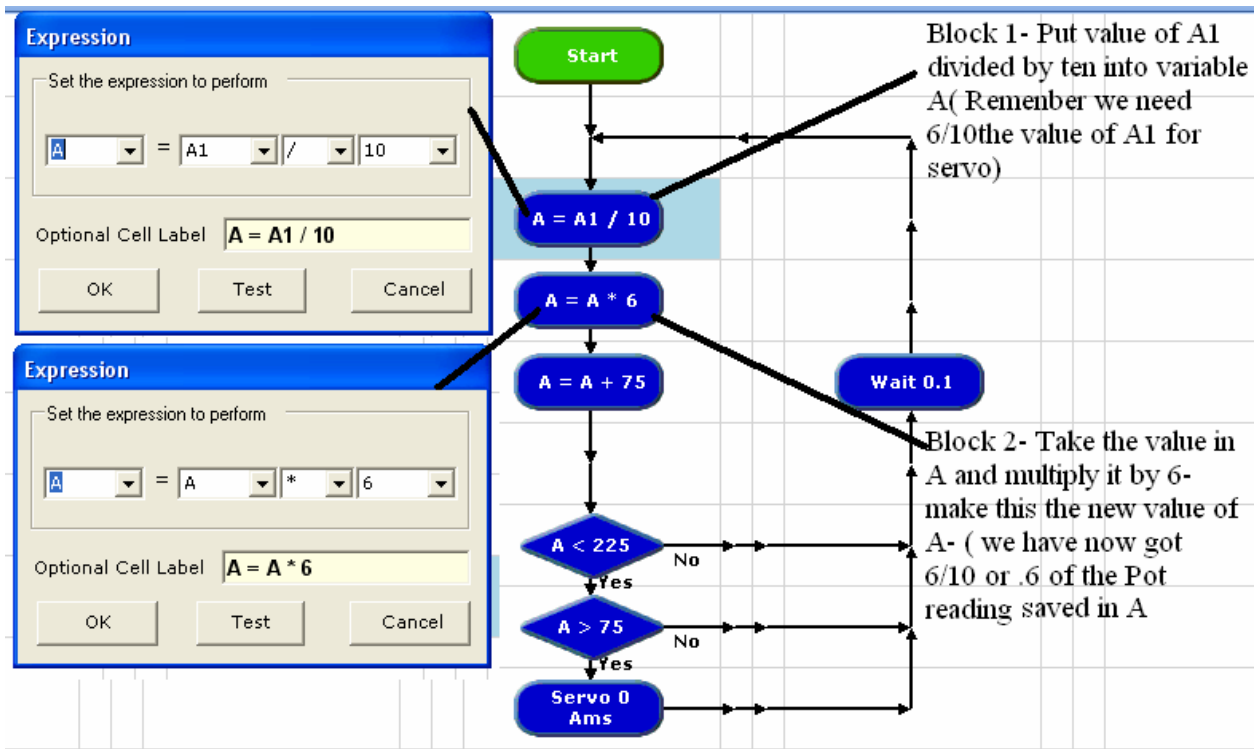


Figure 14- Potentiometer to Servo (1)

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.

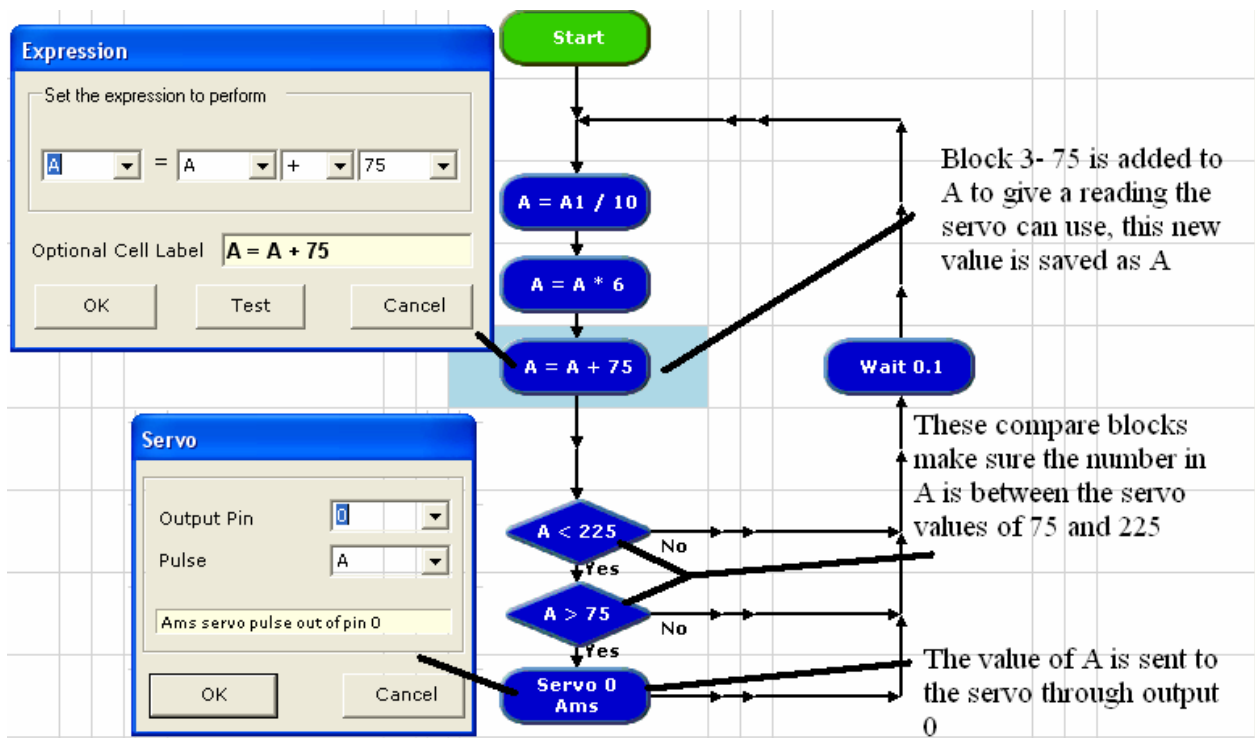


Figure 14- Potentiometer to Servo (2)

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

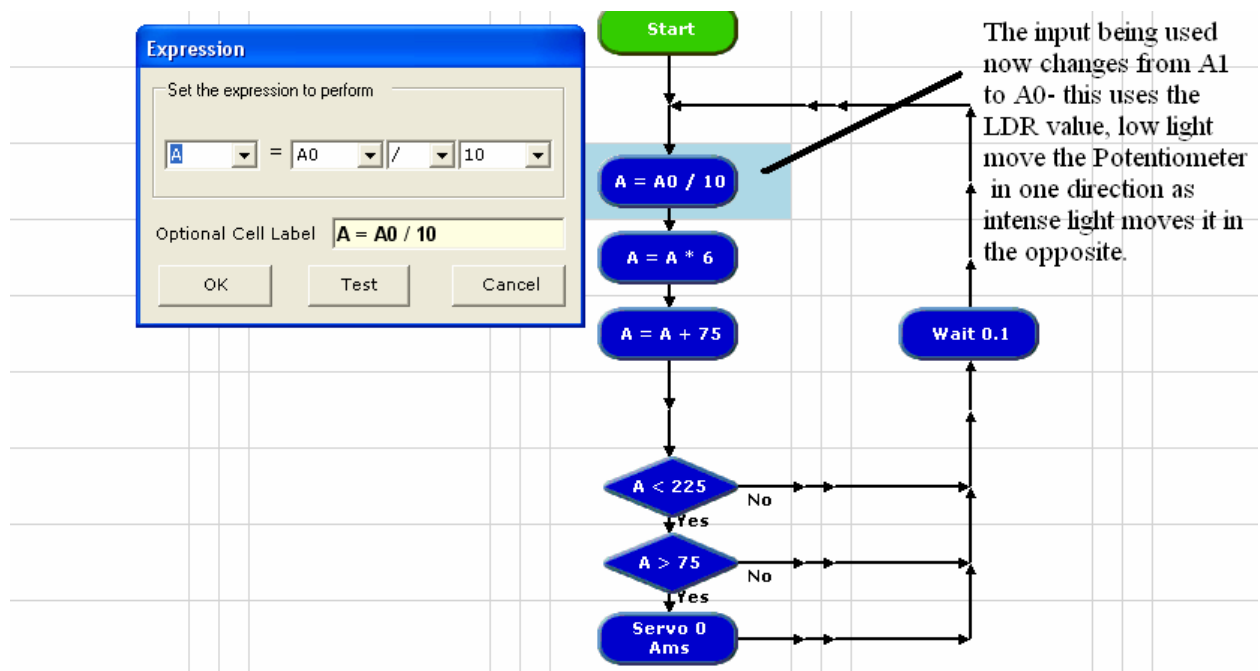


Figure 15- Light Meter

The LCD Display

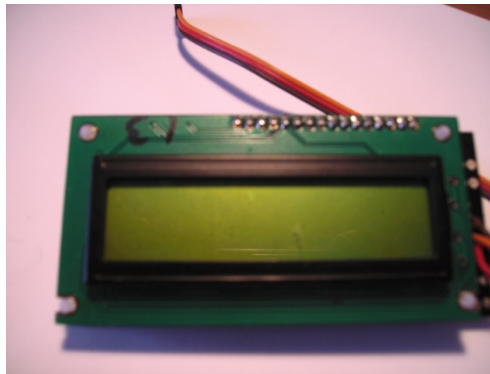


Figure 16- LCD Display

Logicator software allows for the communication with LCD displays through the use of Ascii code. This feature is both entertaining and useful. It can display messages and text and also the values of analogue inputs in real time. These displays can be bought and assembled with minimal soldering. As with the servo there are three wires connected to the display, 0v, 5v and Signal, connection to the T4 training board can be made through the servo pins and output 0.

Things to note with the LCD board

- Requires 0v and 5v at all times- Accurate 5v
- Signal must be take straight from the Pic
- Resolution button on rear of board to refine display
- Programmed using the Serout Command

Depending on the edition of Logicator the Serout command will have one or three Data lines. This does not affect the general programming of the Board, just that the edition with one data line will have more blocks

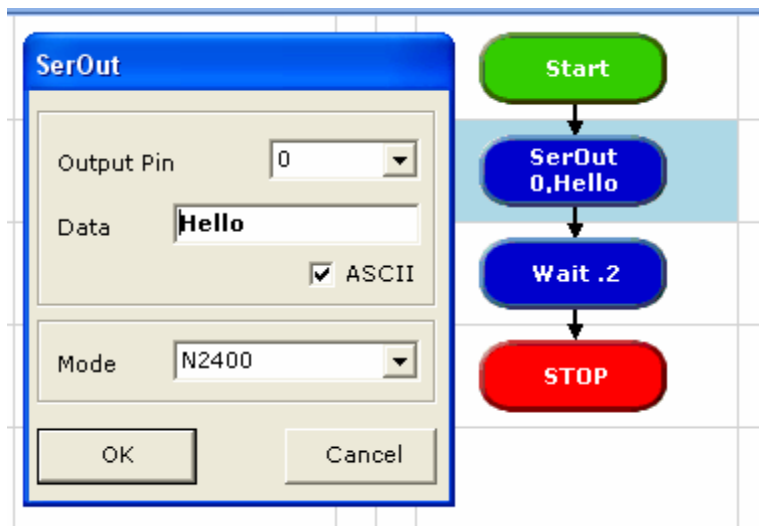


Figure 17- Basic display

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.

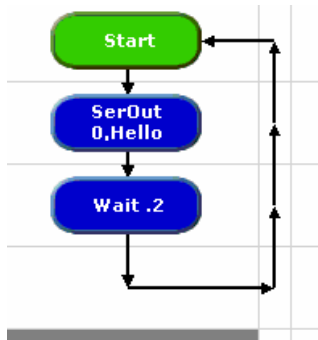


Figure 18- Constant Hello

To make full use of the display there a small number of codes that needs to be known.

- Ascii Code- (Ascii box in ser out command off)
 - Clear Display= (254,1)
 - Move to line one position 1= (254,128)
 - Move to line 1 position x ($Y=128 + x$)= (254,Y)
 - Move to line 2 position 1= (254,192)

The first command to clear screen should be used at the start of a program to make sure that any unwanted information on the display is deleted. This would stop the numerous “Hello” messages from the looped program.

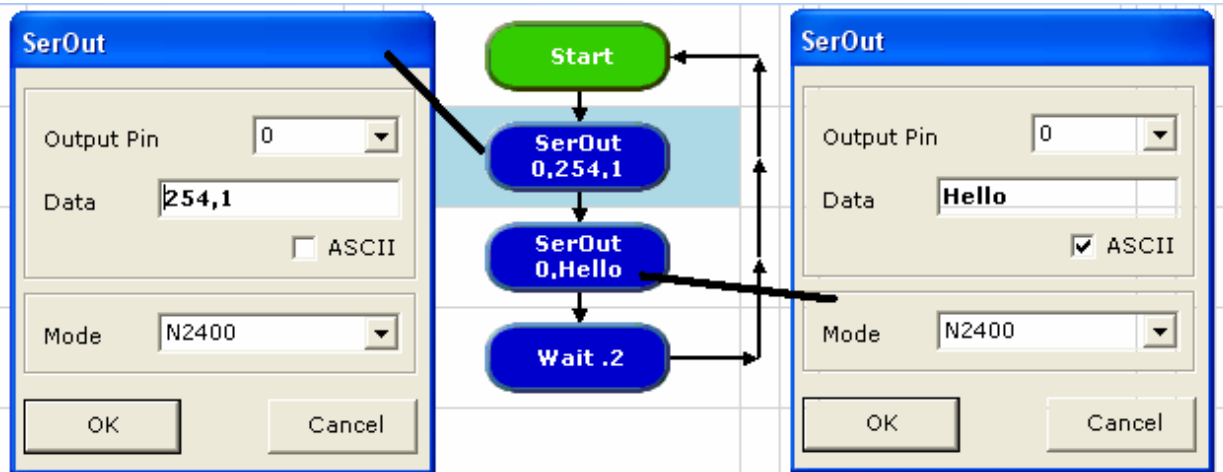


Figure 19 Using Ascii code

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.

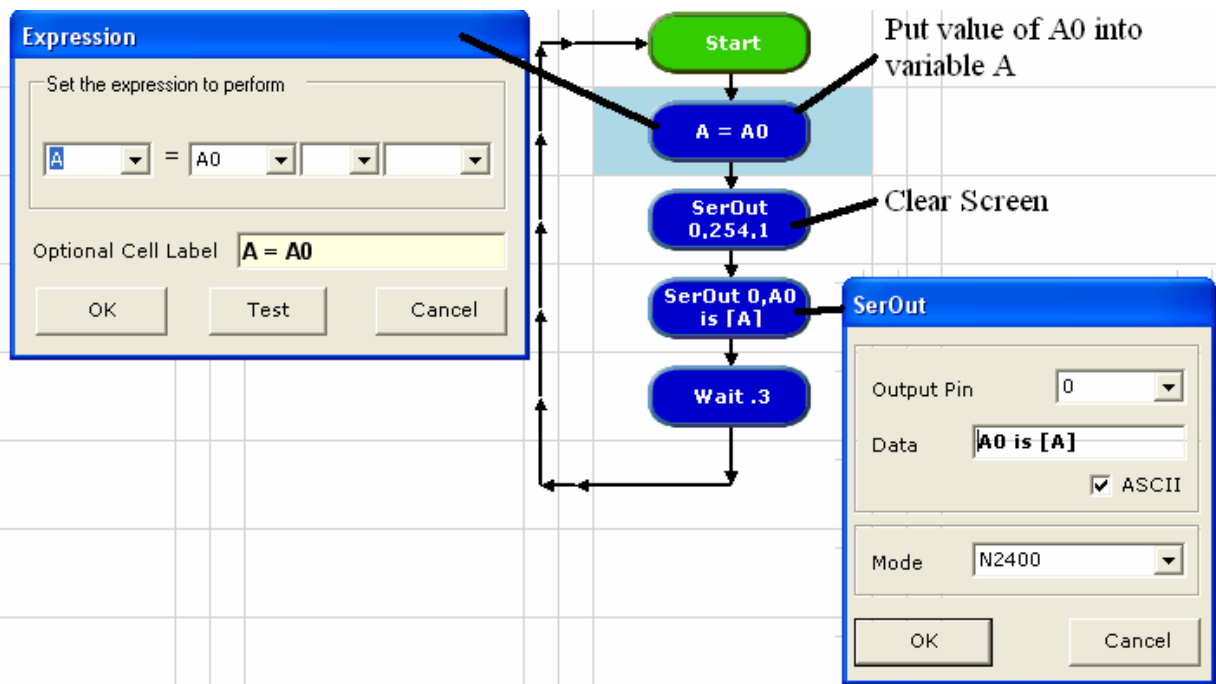


Figure 20 Displaying Input values

This program uses some aspects of previous programs where we express the value of the analogue input A0 as variable A. The two Serout blocks would be one in newer software can become one. The second Serout command contains “A0 is [A]” in the data line. This will appear on the LCD screen as A0 is (the value that is in variable A), whatever letter is place between the square brackets, the program will display the value in this variable. By placing the program on a loop it will refresh the display every .3secs, constantly updating the value of A0.

