

What is Control It PC?

Control It PC is a new Control program from Deltronics based on Control 95. The same familiar language statements and on screen box are used, but there are several new features and improvements.

The language itself is considerably more powerful and versatile, and a lot faster.

Features previously offered as separate programs (such as Virtual Models and Junior Control) are now integrated into one package.

Procedure editing is now a lot easier, and a procedure step and trace feature allows you to follow the progress of the procedures line by line as they run.

Procedures can now write output to the screen e.g. `Write("the motor is on")`

Whether you're a new user or upgrading from Control95, Control It PC is easy to use.

Installation

System requirements

The program will run on any system capable of running Windows 95 or above, having at least 8MB RAM and 4MB free disk space.

To use an interface, one free COM port is required.

Place the Setup Disk in the a: drive and Select **Start Run...** and type **a:\setup.exe** in the box and click **OK**.

During the setup procedure you will be asked for your name, Company (or School) name, and serial number. The serial number can be found on the disk and on the envelope containing the disk.

Please read the copyright notice on the envelope.

To view the serial number after installation, select **About Control It...** from the **Help** menu.

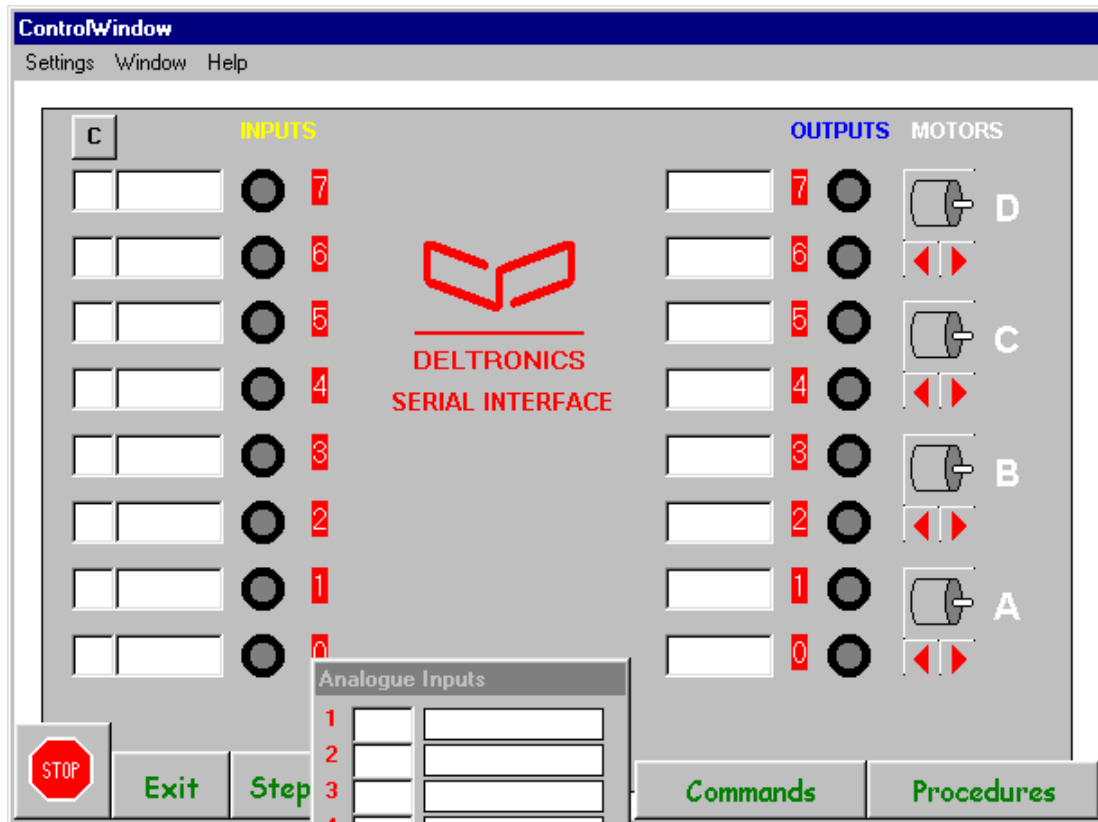
Running Control It PC

Select **Start.. Programs.. Control It.. Control It**

There are two main Windows, the Control Window and the Procedures Window.

Control It PC. main windows

The Control Window



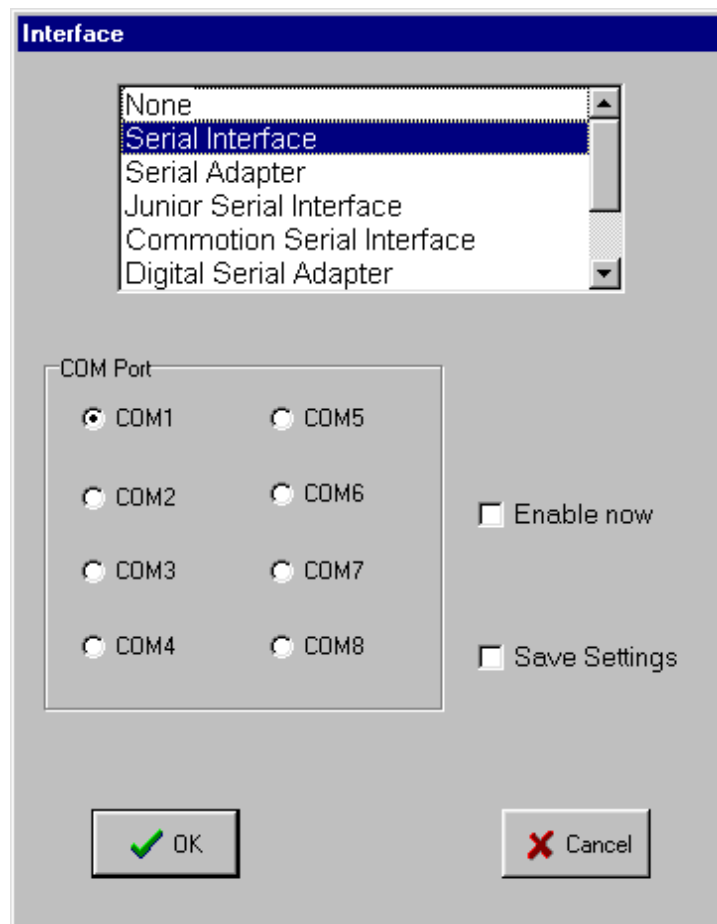
Inputs

On the left of the window is a row of 8 LEDs corresponding to the input LEDs on the box; each has a label which can be used to identify the input and a counter which can count the number of changes on the input.

If you have a box connected, the LED will indicate the state of the real inputs on the box.

NOTE : When you first run the software, the interface is set to 'none'. To configure the program for your interface, select **Interface...** from the **Settings** Menu, this brings up the Interface dialogue

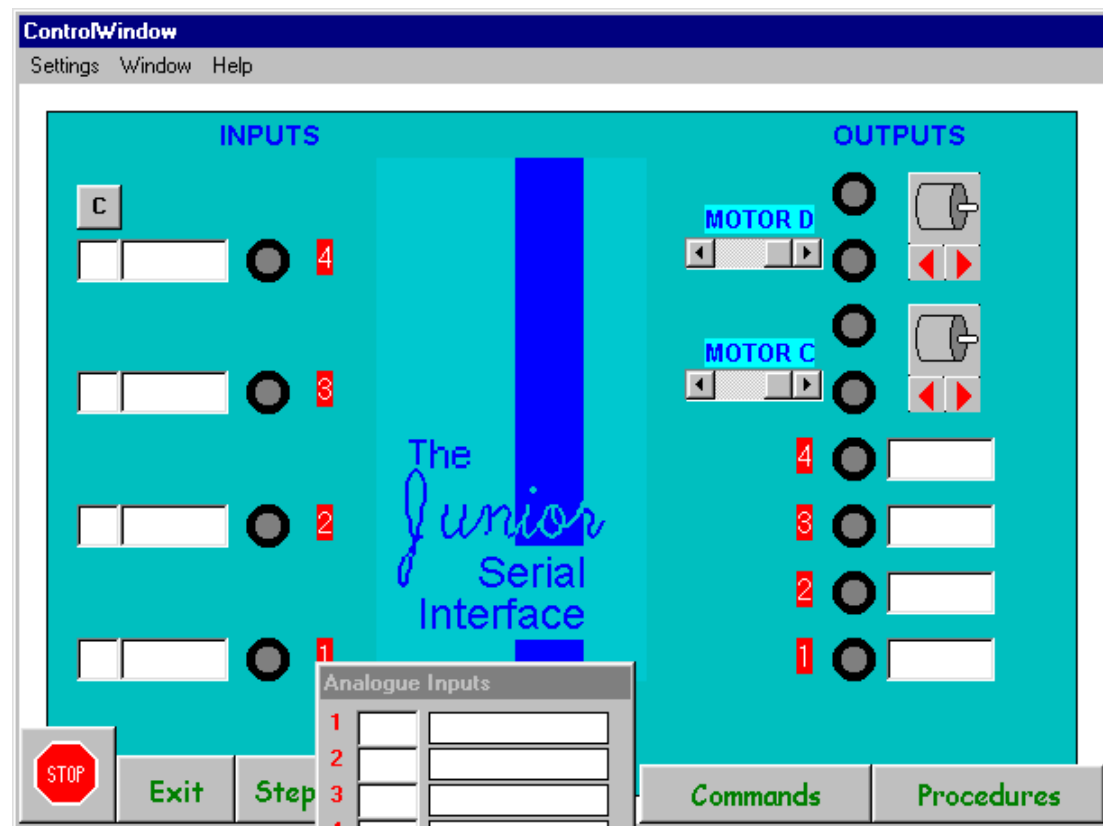
The Interface dialogue



Select the interface from the list and select the COM port to which you have connected it. If the box is connected and switched on, check the 'Enable now' check box. To save the settings for future use, check the 'Save Settings' check box.

The screen appearance will change to reflect the type of box that you selected.

If you selected **Junior Serial Interface**, the window will look like this :-



Outputs

Back to the Control Window - On the right is another set of LEDs representing the outputs. Click on the centre of the LED to turn it on or off, If you have a box connected and enabled, the output on the box will go on or off as well.

The outputs also have labels which can be used to identify them.

Motors

Next to the outputs are four Motor Icons; each icon is composed of three controls, the main motor icon turns the motor on and off and the arrow icons determine the direction. Again, if you have a box connected, and a motor connected to it, the motor will turn.

Analogue

If you selected an interface that has analogue channels, there will be a small window with numerical and bar graph displays of the four analogue channels.

Buttons

in the bottom left corner are three buttons

Stop - turns off all the outputs and stops any procedure that is running

Exit - Exits the program, if you have un-saved procedures you will be asked if you want to save them first.

Step - Used to step through a procedure, one line at a time (see Procedures section)

Speed (Power) Scroll Bars

Select **General...** from the **Settings** menu. Select Scroll Bars from the Display section and click **OK**.

The output labels will be replaced by scroll bars which can be used to set the speed of motors or the brightness of lamps connected to the outputs.

The Procedures Window

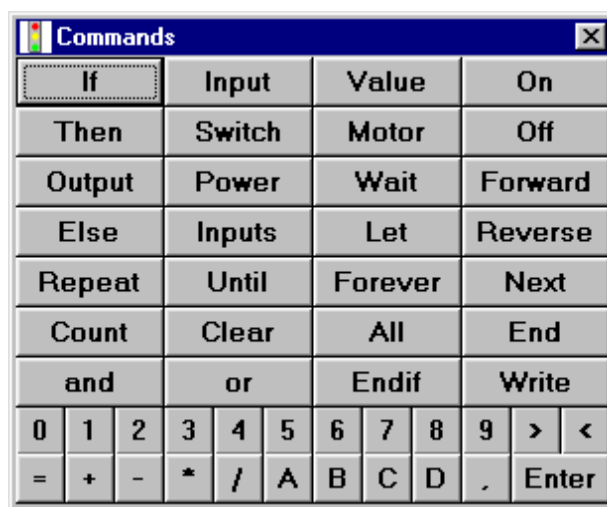
This is where the procedures are created and edited. To start writing a new procedure, Click the **Procedures** button and select **New...** from the pop-up menu or Select **New...** from the **File** menu in the Procedures window. Type the name of your new procedure and click **OK**.

A window for editing your procedure is opened.

You can have several procedures open at once. To edit a procedure, just click on the title bar of its window, or select the procedure name in the **Window** menu, or click the **Procedures** button and select **Change...** from the pop-up menu.

Click the **Commands** button to bring up the Commands Window, which contains all the key words necessary for building a procedure.

The Commands Window



Try this simple procedure :-

```

Switch On 1
Wait 1
Switch Off 1
Wait 1
Switch On 1
End

```

you can click on the buttons in the commands window, or type on the keyboard. If you are using the keyboard, remember to put spaces in-between the each word or number.

To run the procedure, click the Procedures button and select Run... from the pop-up menu, then select the name of the procedure. Alternatively select **Run *name*** from the **Run** Menu (e.g. if your procedure was called test, this menu item will be **Run test**)

Output 1 should go on and off.

Now select **Trace** from the **Run** menu, and run the procedure again; this time the lines in the procedure will be highlighted as the program encounters them. You can also select **Step** from the **Run** menu, then when you run the procedure you can step it through one line at a time by pressing the **Step** button in the Control Window.

To turn off Step or Trace, select it again in the Run menu.

Inputs - using the If statement

Type in the following procedure :-

```
Repeat Forever
  If input 1 On then Switch On 4
  If input 1 Off then Switch Off 4
next
end
```

We ask the program to check the state of an input and do something as a result, using the **If ... Then...** statement

the program only does this once, so if you want to continuously monitor an input you have to put the statement inside a loop. That is the function of the **Repeat Forever** and **Next** commands

Run the procedure. Output 4 should go on when input 1 is on and off when input 1 is off.

To stop the program, press the **STOP** button in the bottom left of the Control Window.

You can simplify this procedure by writing :-

```
Repeat Forever
  If input 1 on switch on 4
  Else switch off 4
next
end
```

this is an **If... Then... Else...** statement. The program performs the **Else** part if the **If** part is **NOT** true; in this case if input 1 is not on.

Notice that the **Then** word has been left out, that is because it is optional.

Saving procedures

To save a procedure, first, select it by clicking in its window title bar, or by selecting it in the **Window** menu. Then select **Save/Save as...** in the File menu. Select the destination folder and click OK. To save the Input and Output labels with the procedure, make sure the **Save Labels** item is ticked.

Repeat Loops

Repeat loops are very useful elements in building most procedures. With a repeat loop you can make the procedure repeat a series of commands a fixed number of times, or until a certain event happens or forever (until the STOP button is pressed)

The basic forms are :-

```
Repeat
  [Statement]
  [Statement]
  ....
  ....
Until [Condition]
```

For Example

```
Repeat
  Switch On 1
  Wait 5
  Switch Off 1
  Wait 5
Until Input 3 On
```

This flashes Output 1 on and off until input 3 is on.

```
Repeat n
  [Statement]
  [Statement]
  ....
  ....
Next
```

e.g.

```
Repeat 4
  Motor A Forward
  wait 2
  Motor A Off
  wait 2
Next
```

The commands in-between **Repeat** and **Next** are repeated four times. Notice that they have spaces in front of them to put them further to the right, this helps to make your procedure more readable by clearly identifying which parts are inside the loop and which parts are not. This is known as indenting.

The program ignores spaces, but obviously not inside key words e.g. Sw itch is not allowed.

Finally there is the Repeat Forever statement

```
Repeat Forever
  [Statement]
  [Statement]
  ....
  ....
Next
```

This repeats everything in-between Repeat and Next 'Forever' i.e. until you press the **STOP** button.

Output Labels - Giving names to the outputs

You can label an output by clicking on it and then typing the name. You can then refer to it by that name in a procedure. For example, if you label some outputs Red, Amber and Green, this procedure will implement a traffic light sequence.

```
Switch Off All
Switch On Red
Wait 5
Switch On Amber
Wait 1
Switch Off All
Switch On Green
Wait 5
Switch Off Green
Switch On Amber
Wait 1
Switch Off Amber
Switch On Red
End
```

NOTE: The wait values should be set to seconds (Select **General...** from the **Settings** menu to change the wait values to Seconds or Tenths).

The procedure should go through the sequence from red to green and back to red again. The first line makes sure that all the

outputs are off before starting.

Example - Pelican crossing

This example should help illustrate some of the basic features of the Control It language.

Specification -

We all know what a pelican crossing does, but it helps to put it down in black and white before we start trying to write any procedures.

Normally, the traffic lights are on Green, and the red man is on.

If the button is pressed, the traffic lights will go to amber, and then to red.

the red man then goes off and the green man comes on.

After a delay, the traffic lights go to flashing amber while the green man flashes.

Finally the traffic lights go to green while the green man goes off and the red man comes on.

This example is best done using sub procedures, that way we can break the task down into smaller parts which are easier to program.

These are the procedures

Pelican - The main procedure

sequence - goes through the whole sequence from red to green and back again.

Flash - flashes the amber light and green man

Bleep - sounds the bleeper while it is safe to cross.

Here's the main procedure - Pelican

```
Switch Off All
Switch On Green
Switch On RedMan
Repeat Forever
    if input 0 on then sequence
Next
```

The first line switches off all the outputs so that everything is in a known state. Then the green light and red man are switched on.

Then we simply go round in a loop testing input 0, if it is on, we go through the sequence.

The sequence procedure

```
Switch off Green
Switch on Amber
Wait 30
Switch Off Amber
Switch On Red
Switch Off RedMan
Switch On GreenMan
Bleep
Switch Off Red
Flash
Switch On RedMan
Switch On Green
end
```

First, the light goes from green to amber, then a pause of three seconds (Wait 30) this assumes that the wait value is set to tenths of a second. Select **General...** from the **Settings** menu to bring up a dialogue to set this.

Then from amber to red, and the man goes from red to green.

Then call another sub procedure, **Bleep** which operates the bleeper (the number and duration of bleeps will determine the time allowed to cross the road)

Then switch off the red light and call another sub procedure **Flash** which flashes the green man and the amber light. Finally, switch on the red man and the green light, to return to the original state.

Notice that all procedures must end in an end statement. when the procedure gets to the end statement, it returns to the procedure that called it.

Now we need to write the procedures Bleep and Flash

Bleep

```
Repeat 20
  Switch on Bleeper
  Wait 5
  Switch off Bleeper
  Wait 5
Next
End
```

Flash

```
Repeat 10
  Switch On Amber
  Switch On GreenMan
  Wait 5
  Switch Off Amber
  Switch Off GreenMan
  Wait 5
Next
End
```

Remember that the wait values must be set to tenths. Select **General...** from the **Settings** menu.

Also, remember to label all the outputs - Red, Green, Amber, RedMan, GreenMan, Bleeper.

NOTE: there should be no spaces in RedMan and GreenMan.

Examples

The following pages contain ten examples which introduce most of the features of the Control It PC language.

Any additional features can be found in the command reference and in the program help.

1. Lighthouse

This is a simple procedure for turning a light on and off.
Build the lighthouse from Lego or similar and attach a bulb to the top. Connect the bulb to one of the output sockets on the Control It or Serial Interface.

Build the following procedure (call it 'light')

```
Switch On 2
Wait 1
Switch Off 2
Wait 1
Switch On 2
Wait 1
Switch Off 2
End
```

The procedure assumes that Output 2 will be used. Make sure that the wait values are set to Seconds (select **General...** in the **Settings** Menu). Run the procedure. (Remember to select the interface from the Configure Menu) The light should flash on and off twice.

Loops - making the procedure repeat

You can make the procedure repeat any number of times by using **Repeat**.

Change the procedure to :-

```
Repeat 10
  Switch On 2
  Wait 1
  Switch Off 2
  Wait 1
Next
End
```


Lighthouse (continued)

Run the procedure, and the light will flash on and off ten times.

Using Repeat, the light flashed ten times instead of twice, but has used one less line to do it. The word Next indicates the end of the loop.

You can put any number after Repeat or you can use Repeat Forever, to make the light flash continuously. In this case the program can be stopped by using the **STOP** button.

Labels - giving names to the Outputs.

Click on the label next to Output 2 and type the word LAMP.

Now change the procedure to :-

```
Repeat 10
  Switch On Lamp
  Wait 1
  Switch Off Lamp
  Wait 10
Next
End
```

Saving the Procedure.

Select the procedure by clicking on its title bar or by selecting it from the **Window** menu Select **Save/Save As** from the **File** Menu. Select the destination folder and click OK.

2. Traffic lights

This procedure goes through a traffic light sequence.

First, label the lower three outputs, Red, Amber and Green.

Then build this procedure :-

```
Switch Off All
Switch On Red
Wait 4
Switch On Amber
Wait 1
Switch Off Red
Switch Off Amber
Switch On Green
Wait 4
Switch Off Green
Switch On Amber
Wait 1
Switch Off Amber
Switch On Red
```

Run the procedure, and it should go through the sequence from Red to Green and back to Red again. The first line makes sure that all the lights are off before starting.

To make the sequence repeat continuously, use **Repeat** as follows:-

```
Switch Off All
Repeat Forever
  Switch On Red
  .....
  .....
  .....
  Switch Off Amber
Next
End
```

3. Using Inputs - A Simple Alarm

For this procedure, you will need a buzzer and a magnetic proximity switch or a microswitch.

Attach the proximity switch to a door or window so that the switch is **On** when the door is shut and **Off** when it is open. Connect it to one of the inputs, say Input 1. Connect the buzzer to one of the outputs, say Output 4; you can label the Output, 'Buzzer'.

Build the following procedure :-

```
Repeat Forever
  If Input 1 On Then Switch Off Buzzer
  Else Switch On Buzzer
Next
End
```

Run the procedure, and the buzzer should sound when the door is opened. You can use labels on the inputs as well, therefore if you label Input 1 as 'Door' , you can say :-

```
If Door On Then Switch Off Buzzer
```

Using Sub Procedures.

Suppose you want the buzzer to go on and off instead of sounding a continuous tone.

Build this procedure (call it Buzz)

```
Repeat Forever
  Switch On Buzzer
  Wait 5
  Switch Off Buzzer
  Wait 5
Next
End
```

Alarm (continued)

Then modify the Alarm procedure to :-

```
Repeat Forever
  If Door On then Switch Off Buzzer
  Else Buzz
Next
End
```

Adding other inputs

You can add other sensors to the alarm. for example a pressure mat. Connect the pressure mat to one of the inputs, and label that input 'Mat'.

Modify the procedure to

```
Repeat Forever
  If Door On Then Switch Off Buzzer
  Else Buzz
  If Mat Off Then Switch Off buzzer
  Else Buzz
Next
End
```

Notice that the pressure mat is normally 'Off' but goes to 'On' when it is stepped on.

4. Car Park

This procedure counts the cars going into and coming out from a car park. You can use a microswitch or a light switch to detect the cars going in and out. Two bulbs are needed for the 'SPACES' and 'FULL' signs.

Build this procedure:-

```
Repeat Forever
  Let A = Count 1
  Let B = Count 2
  If (A-B) > 9 Then FullSign
  Else SpacesSign
Next
End
```

and the Sub Procedures FullSign :-

```
Switch On Full
Switch Off Spaces
End
```

and SpacesSign :-

```
Switch On Spaces
Switch Off Full
End
```

Label one of the outputs 'Spaces' and another 'Full'. The program assumes that there are ten spaces in the car park.

Car Park (continued)

This procedure uses two variables A and B. Variables are used whenever you need to store or remember a number or perform calculations - in this case A is the number of cars that have entered the car park and B is the number of cars that have left. Here, the **'If'** command is used to test the result of a comparison. i.e. "is (A-B) greater than 9 ?"

You can read more about how Control It PC performs arithmetic and comparisons in the Arithmetic section.

You can monitor the value of a variable while a procedure is running, by selecting **Variables** from the **Window** menu in the Control Window.

Note that the sub procedures FullSign and SpacesSign, only have two lines each (excluding the end statement) The Car Park program could have been written as one procedure like this :-

```
Repeat Forever
  Let A = Count 1
  Let B = Count 2
  If (A-B) > 9 Then
    Switch On Full
    Switch Off Spaces
  Else
    Switch On Spaces
    Switch Off Full
  EndIf
Next
End
```

This is the long form of the **If... Then... Else..** construction which is :-

```
If [condition] Then
    [statement]
    [statement]
    .....
    .....
Else
    [statement]
    [statement]
    .....
EndIf
```

The important rule is that the first statement in each block must be on the line after the '**If**' or the line after the '**Else**'.

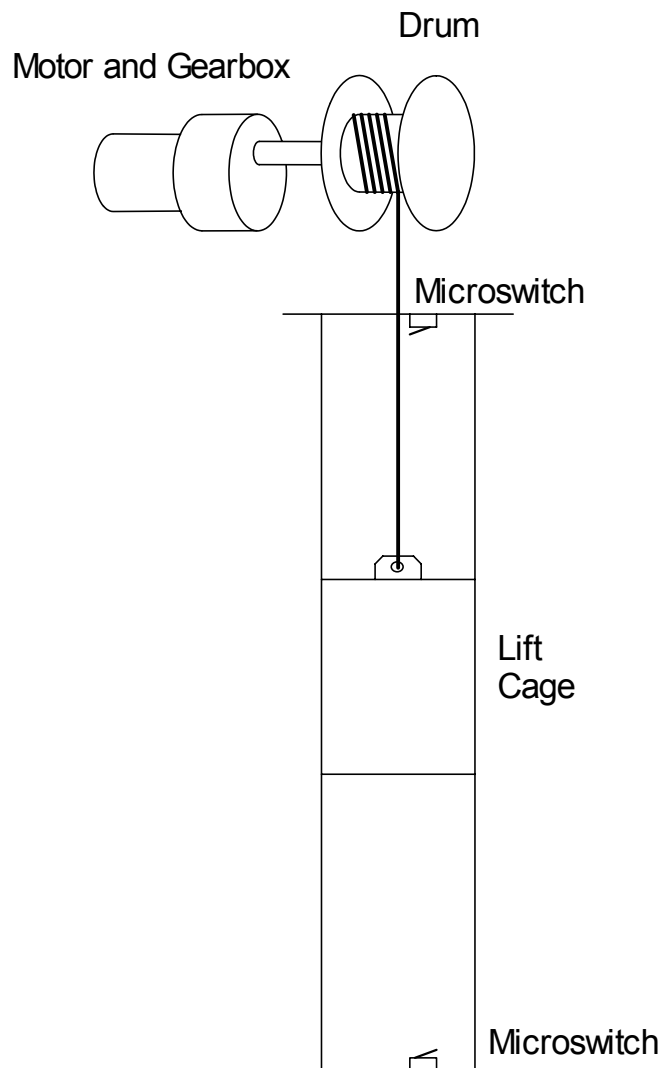
The **EndIf** statement must also be present.

The **Else** block is optional. i.e. you can have :-

```
If [condition]
    [statement]
    [statement]
    .....
EndIf
```

5. Lift

This program operates a simple two floor lift.



To build the lift you will need a motor (geared) and a pulley or drum (cotton reel or similar). The lift cage will need to run smoothly in a set of guides. You will also need two microswitches or magnetic proximity switches.

Lift (continued)

First write a simple procedure to send the lift to the top.
Connect the motor to Motor C and the top microswitch to Input 1.

Build this procedure (call it 'Up')

```
Repeat
  Motor C Forward
Until Input 1 On
Motor C Off
End
```

This procedure uses a simple example of feedback, i.e. the effect of an Output is monitored by an input in order to decide when to switch the output on or off .

Add two push buttons; one for up and one for down. Connect them to Inputs 3 and 4, then write the overall lift procedure as follows :-

```
Repeat Forever
  If Input 3 On then Up
  If Input 4 On Then Down
Next
End
```

Before you run this, you should also write the procedure 'Down' which is very similar to 'Up'

```
Repeat
  Motor C Reverse
Until Input 2 On
Motor C Off
End
```

(Input 2 should be connected to the bottom microswitch)

6. Street Lights

This procedure automatically switches on a street light when it becomes dark. It introduces the use of Analogue Inputs. You will need a light level sensor.

Connect the sensor to Analogue Channel 1, and a light bulb representing the street light to Output 1.

Build this procedure:-

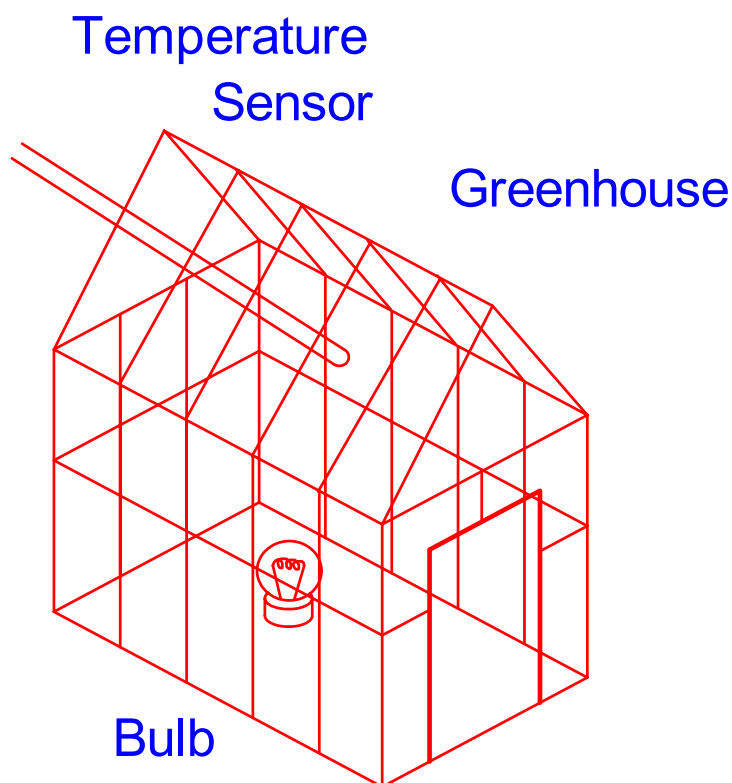
```
Repeat Forever
  If Value 1 < 50 Then Switch On 1
  Else Switch Off 1
Next
End
```

When you cover the light sensor the light should come on; uncover it and the light should go out. (You may have to experiment with the value in line 2 to get it right).

7. More analogue - Temperature Control - A Greenhouse

This procedure is a temperature control system for a greenhouse. You will need a 5W bulb as a heater (the small bulbs in the Deltronics accessory pack are not suitable for this) and a temperature sensor.

Build a model greenhouse (don't make it too large or it will take a long time to warm up) and place the bulb and temperature sensor inside it. Ideally, for demonstration purposes the temperature sensor should be located at the top, above the lamp, so that it warms up quickly. In a real greenhouse it should be located low down so that the whole greenhouse warms up before the heater switches off.



Connect the bulb to Output 1 and label it 'Heater'. Connect the temperature sensor to Analogue Channel 1. Select **General...** from the **Settings** Menu and set degrees C instead of 0 to 100% .

Greenhouse (continued)

Build this procedure:-

```
Repeat Forever
  If Value 1 > 24 then Switch Off Heater
  Else Switch On Heater
Next
End
```

It will take a few minutes to observe the system switching on and off, but eventually it should stay within a degree or two of the value that you set.

8. Lift 2 - Adding an extra floor

NOTE: This procedure is unsuitable for the Junior Serial Interface and the PIC boards because they don't have enough inputs.

There is an extra problem with having more than two floors. If you want to go to the middle floor, which way you turn the motor will depend on whether you are above that floor or below it. One way of doing this is to have a variable ('floor') which records which floor the lift is on.

You will need three inputs for pushbuttons (label them B0, B1 and B2) and three microswitches or proximity switches for the floors (label them F0, F1 and F2)

The main procedure will look something like this:

```
Ground
Repeat Forever
  If B0 On Then Ground
  If B1 On Then First
  If B2 On Then Second
Next
End
```

The procedures Ground, First and Second send the lift to the ground, first and second floors respectively.

The procedures Ground and Second are the easiest to write, and are the same as 'Up' and 'Down' in the previous lift example.

Ground

```
Motor A Reverse
Repeat
Until F0 On
Motor A Off
Let Floor = 0
End
```

Lift 2 (continued)

Second

```
Motor A Forward  
Repeat  
Until F2 On  
Motor A Off  
Let Floor = 2  
End
```

Now, the procedure 'First'. Which way we turn the motor (or if we turn it at all) depends on the variable 'Floor'

First

```
If Floor = 2 Then Motor A Reverse  
If Floor = 0 Then Motor A Forward  
If Floor = 1 Then Motor A Off  
Repeat  
Until F1 On  
Motor A Off  
Floor = 1  
End
```

Notice the line Floor = 1 , the 'Let' was left out, that is because it is optional.

Lift2 (continued)

Improvements

Add a light to each floor which comes on when the lift arrives; add a buzzer which sounds when the lift arrives; add a door to the lift.

The most difficult thing about adding a door is constructing the door; use a motor with a lot of reduction gearing, and a switch to detect when the door is open (or closed). You can either put doors on the fixed part of the lift, in which case each floor has its own door, or you can fit one door to the lift cage, but this means that the door motor and associated wiring has to be carried on the lift.

The best way to write the program is to use another two sub procedures 'Open' and 'Close'.

9. Traffic Lights 2

This program implements a road junction with two sets of lights

There are four sub procedures

Procedure R1 sends set 1 to red

Procedure G1 sends set 1 to green

Procedure R2 sends set 2 to red

Procedure G2 sends set 2 to green

Label the Outputs Red1, Amber1, Green1 and Red2, Amber2, Green2

Procedure R1

```
Switch Off Green1
Switch On Amber1
Wait 2
Switch Off Amber1
Switch On Red1
End
```

Procedure G1

```
Switch On Amber1
Wait 2
Switch Off Red1
Switch Off Amber1
Switch On Green1
End
```

Procedures R2 and G2 are the same, but use Red2, Green2 etc.

Traffic lights 2 (continued)

Main Procedure

```
Switch Off All  
Switch On Red1  
Switch On Red2  
Repeat Forever  
  G1  
  Wait 10  
  R1  
  Wait 3  
  G2  
  Wait 10  
  R2  
  Wait 3  
Next  
End
```

Each road has ten seconds of Green light, and there is an overlap when both sets are on Red for 3 seconds.

10. Level Crossing

This program operates a level crossing barrier and lights. You will need a train set (or at least a short length of track), a magnetic proximity switch or microswitch to detect the train coming past a certain point on the track, a motor and gearbox for the barrier, another switch to detect when the barrier is down, some bulbs and a buzzer.

Connect the train detecting switch to, say Input 1, the barrier motor to motor A and the 'barrier down' switch to Input 2.

Write the following procedure:-

```
Repeat
  Until Input 1 On
  Motor A Forward
  Repeat
    Flash
  Until Input 2 On
  Motor A Off
  Repeat 20
    Flash
  Next
  Motor A Reverse
  Wait 20
  Motor A Off
End
```

Sub procedure 'Flash'

```
Switch On 3
Wait 3
Switch Off 3
Wait 3
End
```

Level crossing (continued)

For these procedures, the wait values should be set to tenths of a second. (select **General...** from the **Settings** Menu)

The first two lines wait for the train to activate the switch. The next part starts winding down the barrier and flashing the lights. The barrier motor is stopped when the 'barrier down' switch is activated, but the lights continue to flash twenty more times. The motor is then reversed to wind up the barrier. (The wait value here will depend on the speed of the motor)

The number of repeats after the barrier has come down should be enough to let the train go past.

The duration of flashes should be kept short, otherwise the barrier could over-run before it is switched off.

What happens if the train slows down or stops after it has triggered the switch but before it has reached the crossing?

To overcome this problem, put another switch on the far side of the crossing and change **Repeat 20** to **Repeat Until Input n On** where n is the input to which the sensor is connected.

Improvements

Modify the procedure 'Flash' to operate two lights to flash alternately (as in a real level crossing). You could also add a buzzer here.

Command Reference

Clear

Clear, clears a specific input counter,

e.g. Clear 5

Count

Count represents a count of the number of Off to On transitions on an input since the last Clear (either by command or by the on screen button) The form is **Count *n*** , where ***n*** is the input number.

Count can be used in any arithmetic expression or logical expression

e.g. If Count 5 > 20 Then Switch Off 2

Let X = Count 3

Let Y = Count 1 + Count 2

Do

The **Do** command is the same as **Repeat n** (see **Repeat**). it is used to repeat a series of commands, a fixed number of times. The form is

```
Do n
  [statement]
  [statement]
  ....
EndDo
```

n is the number of times to repeat the statements.

Else

(see **If**)

End

Every procedure must end in an **end** command. In the main procedure, it tells the procedure to stop. In a sub procedure, it tells the procedure to return to where it was called from. If you leave out **End**, the compiler will assume it, but it is good practice to put an end statement after all your procedures.

EndDo (see **Do**)

EndIf (see **If**)

EndWhile (see **While**)

EndLoop (see **Loop**)

Forever

Forever is a keyword used in conjunction with **Repeat** (see **Repeat**)

Forward

Forward is a keyword used in conjunction with **Motor** (see **Motor**)

If

If, along with Then is used to construct conditional statements.
The general form is:-

If [Condition] then [Command]

e.g. If Input 1 On Then Switch On 7
If Value 4 < 40 Then Motor A Reverse

The 'Then' is optional, i.e.

If Input 1 On Switch On 7

is the same as: If Input 1 On Then Switch On 7

You can test more than one input at once using the 'and' and 'or' keywords e.g.

If Input 1 On and Input 2 On Switch on 6
If Input 3 On or Input 6 On Motor A forward

If you want to perform more than one action in an If command, you can use the long form of the If statement which has the form:-

```
If [condition]
    [statement]
    [statement]
    ....
EndIf
```

The statements **must** start on the line **after** the If [condition], otherwise the compiler will treat it as the short form. You **must** include the EndIf to show where the block of statements ends, and the rest of the procedure begins.

An If statement can be used on its own or with an Else statement.

Else

Else is always used in conjunction with an If statement. An If.. Then.. Else.. construction carries out one action if the 'if' condition is true or another if it is false.

e.g. If Input 1 on then Switch on 4
Else Switch off 4

The Else must be on the line which immediately follows the If.

Long form of If.. Then.. Else..

If.. Then.. Else.. also has a long form which is:-

```
If [condition]
    [statement]
    [statement]
    ....
Else
    [statement]
    [statement]
    ....
EndIf
```

The statements following else must start on the line after the else, and you must include the EndIf.

Input

Input is always used in conjunction with If, Until or While to test the state of an input bit.

e.g. If Input 2 On Then

 If Input 2 Off Then

 Until Input 4 On

You can test more than one input at once using the 'and' and 'or' keywords e.g.

 If Input 1 On and Input 2 On Switch on 6

 If Input 3 On or Input 6 On Motor A forward

Inputs

Inputs is used to check the state of all the inputs at once, and is always used in conjunction with If, Until or While

e.g. If Inputs = 96 Then Switch on 5
 Else Switch off 5

If the binary byte represented by the inputs is '96 decimal' then output 5 will be switched on.

'>' and '<' are not allowed with this command, only '='.

Let

Let is used to assign a value to a variable. Variable names can consist of numbers and letters, but must start with a letter.

The general form is :-

Let [Variable] = [expression]

Variable is any variable name, and expression can be a constant, another variable, or an arithmetic expression,

e.g. Let floor = 1
Let items = items + 1
Let B = F / 3

Let is optional, i.e.

is the same as : floor = 1
Let floor = 1

See the section on Arithmetic for more details.

Loop

Loop can be used to construct a continuous loop, and is an alternative form of a Repeat Forever loop (see Repeat)

The form is :-

```
Loop
    [statement]
    [statement]
    ....
EndLoop
```

The only way to stop a Loop is to press the **STOP** button.

Motor

Motor is used to set the state of one of the four motors to Off, Forward or Reverse,

e.g. Motor A Forward
Motor C Reverse
Motor A Off

Next Next is used in conjunction with Repeat (see Repeat)

Off

Off is used in conjunction with Input, Switch or Motor (see the relevant commands)

On

On is used in conjunction with Input or Switch (see Input and Switch)

Or

Or can be used when testing inputs e.g.

If input 1 on or Input 2 On Then Switch on 4

Switches output 4 on if either input 1 **or** input 2 is on.

Or is also used in relational expressions (see Arithmetic section)

Output

Output is used to output a byte to the outputs,

e.g. Output 48 causes the binary equivalent of 48 to appear on the outputs, overriding any previous Switch On or Switch Off commands. Constants or variables can be used,

i.e. Let A = 48
Output A

Has the same effect as Output 48

See the section on Arithmetic for more information on variables.

Power

Power is used to set the power (speed) of one of the eight outputs or one of the four motors. It can be used to control motor speed or lamp brightness.

The general form is :-

Power [n] [p]

n is the output number or motor reference and p is the power,

e.g. Power 3 20
Power B 15
Power 0 31

Power values can range from 0-31. 0 is stationary and 31 is full speed.

Note: The relevant Motor or output must be switched on before this command can take effect.

Repeat

Repeat is used along with Until or Next to construct loops.

There are three general forms:-

(a)
Repeat
 [statement]
 [statement]

Until [Condition]

This repeats all the statements until the condition becomes true.

Condition can be a reference to Inputs, Input counts or Values, or an arithmetic expression.

e.g. Until Value 2 > 28
 Until Count 7 = 100
 Until A = 7
 Until X=Y+3
 Until Input 2 On

(b) Repeat *n*
 [statement]
 [statement]

 Next

This repeats all the statements in-between Repeat and Next, *n* times. For example:-

 Repeat 4
 Switch on 1
 Wait 1
 Switch off 1
 Wait 1
 Next

Flashes output 1 on and off four times.

(c) Repeat Forever
 [statement]
 [statement]

 Next

Repeats all the statements in-between Repeat and Next, 'Forever' i.e. until the STOP button is pressed.

Reverse

Reverse is used in conjunction with Motor (see Motor)

Switch

Switch is used to switch on or off one of the outputs.
Constants or variables can be used, e.g.

Switch On 5

Switch Off 6

X=5

Switch On X

is the same as: Switch On 5

You can also switch on or off more than one output at a time e.g.

Switch On 2,4,6

Then (see If)

Until (See Repeat)

Value

Value is used to read one of the four analogue channels and is always used in conjunction with If, Until, While or Let

e.g. If Value 1 > 50 Then Switch On 3

Value returns values of 0-100 for an input voltage range of 0 to 1.8V. You can make the Value command return a value in Degrees C

Select **General...** from the **Settings** menu and check the Degrees C button in the Analogue Values box.

Analogue values can also be assigned to variables (See Arithmetic).

Wait

Wait causes the program to wait for a specified number of tenths of a second or seconds.

To set Tenths or seconds, Select **General...** from the **Settings** menu and select Tenths or Seconds in the Wait Values Box.

e.g. Wait 20

While

While is used in conjunction with **EndWhile** to construct loops. The general form is :-

```
While [condition]
    [statement]
    [statement]
    ....
EndWhile
```

Condition can be a reference to Inputs, Input counts or Values, or an arithmetic expression.

e.g. While Input 2 On
 While Count 7 < 100
 While A = 0
 While X<Y+3

The program will repeat the set of statements *while* the condition is true ie *until* the condition becomes false.

Write

Write is used to write text to the screen. The text appears in the Write Window.

The form is:-

Write "[text]"

Where [text] is the text that you want to appear in the Write Window

e.g. Write "The motor is on"

The Write window will appear automatically if it is not shown. To show the Write window, select **Write** from the **Window** menu, to hide it, select **Write** again. Every **Write** command will start a new line in the Write window.

To change the font of the text in the Write window, select **Write Font...** from the **Fonts** menu in the **Procedures** window.

Arithmetic

Control It PC can perform integer and Boolean arithmetic.

Variables

Variables are created using the Let command (see Command Reference). Any reference to a variable not created in this way will result in a 'Variable not found' error.

e.g. `Let A = B + 4`

would result in an error if B had not previously been referred to on the left hand side of a Let assignment.

The general form of an arithmetic statement is :-

`Let [variable] = [expression]`

expression can be a constant e.g. `Let A = 6`

Or another variable e.g. `Let A = B`

Or a general expression.

e.g. a simple expression has the form

`[operand] [operator] [operand]`

Each operand can be a variable or constant and operator can be one of

`+ - * /`

The operators perform addition, subtraction, multiplication and integer division respectively. (Integer division discards the remainder e.g. $9 / 4 = 2$)

The Let command can be used to assign one of the Analogue values to a variable.

e.g. Let V = Value 2

or Let V = Value 1 + 12

Variables can be used as parameters for certain commands.

e.g. Let VAR = 5
 Switch On VAR

Or

 Output VAR

Relation Operators

Relation operators can be used in If, Until or While to test for various conditions e.g. $X > 6$

The four relational operators are $>$, $<$, $=$ and $\#$.

$>$	-	"Greater Than"
$<$	-	"Less Than"
$=$	-	"Equals"
$\#$	-	"Not Equal to"

Examples -

If Count 1 > 10

Until $X=Y*(A+3)$

If Floor # 1