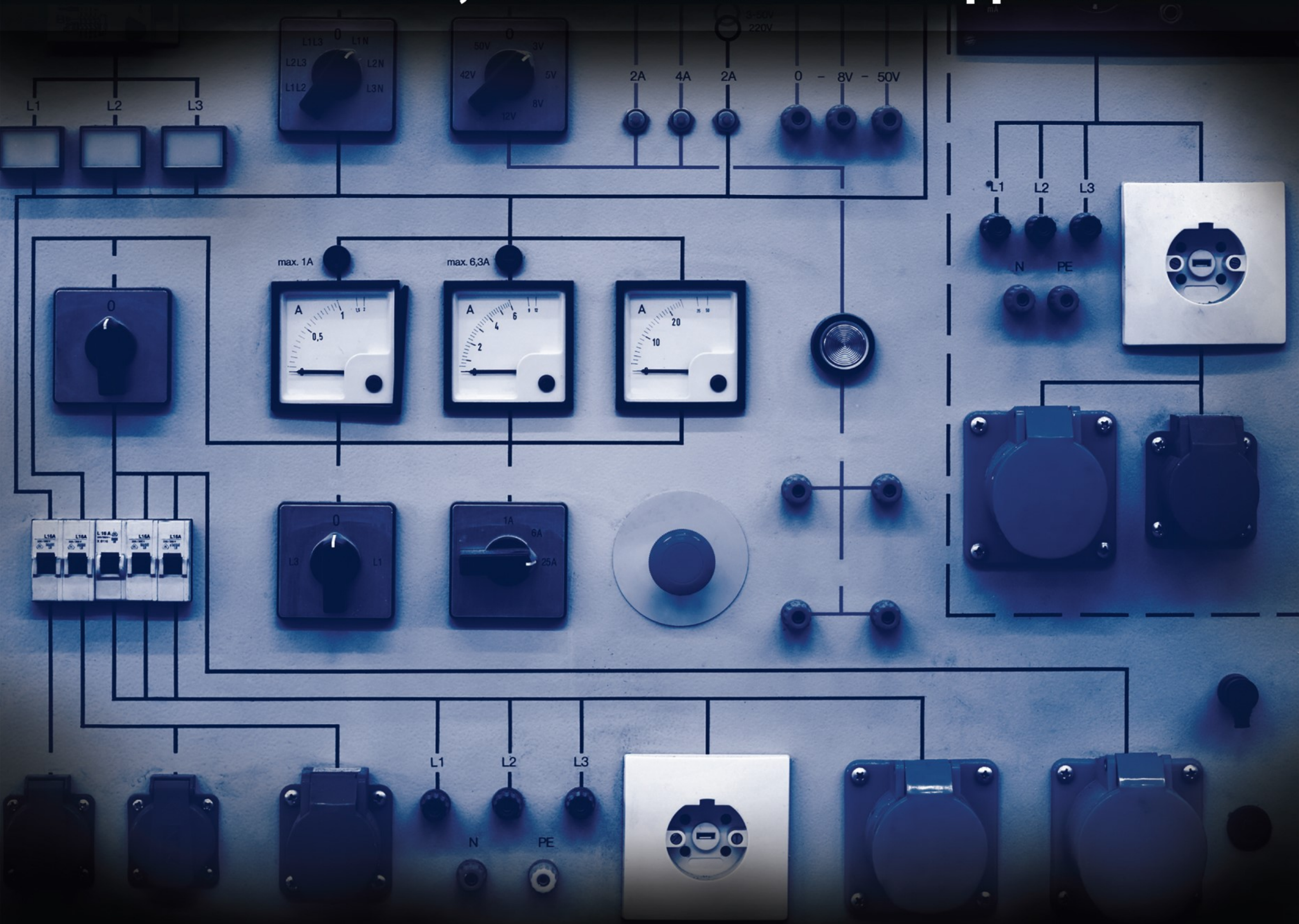
locktronics

Simplifying Electricity

Industrial sensor, actuator and control applications



LK8739

MATRIX

www.matrixmultimedia.com

Contents

Industrial sensor, actuator and control applications

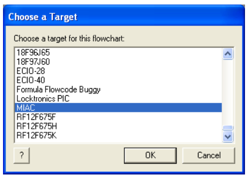
Date	Release notes	Release version
16 12 2014	Updated for RoHS compliance	LK8739-80-3
01 12 2016	Changed references from CD to website	LK8739-80-5

Course Introduction

Industrial sensor, actuator and control applications

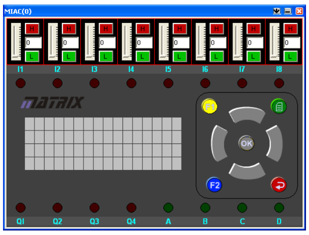
The aim of this course is to learn to program Industrial controller modules using the programming language of your choice. At the end of the document is a section which briefly details how to go about creating programs to control the MIAC using Flowcode, Labview and Visual Basic. The course includes pre compiled examples for every worksheet written in Flowcode, Labview and Visual Basic. These examples are available from the website.

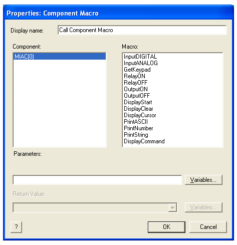
Flowcode creates a program that runs exclusively on the MIAC so for user output you will have to use the LCD display and for any user input you can use the keypad fitted to the front of the MIAC. Flowcode includes component macros to drive the different peripherals onboard the MIAC. Simply ensure that you have selected the MIAC as your target device and also ensure that the MIAC component has been added to your program.


Choose a Target

Choose a target for this flowchart:

18F96J65
18F97J60
ECIO-28
ECIO-40
Formula Flowcode Buggy
Locktronics PIC
MIAC
RF12F675F
RF12F675H
RF12F675K




Properties: Component Macro

Display name: Call Component Macro

Component: MIAC(0)

Macro:
InputDIGITAL
InputANALOG
GetKeypad
RelayON
RelayOFF
OutputON
OutputOFF
DisplayStart
DisplayClear
DisplayCursor
PrintASCII
PrintNumber
PrintString
DisplayCommand

Parameters:

Return Value:

If you are using Labview or Visual Basic then to start you will need to make sure your MIAC has been programmed with the USB Slave firmware. To do this you will need to connect the MIAC to a computer using a USB cable and then use the ‘MIACprog.exe’ tool to program the “MIAC_USB_Slave.hex” file into it. You will also need to make sure the USB driver has been installed on your PC. All of these files can be downloaded from the website. The MIAC will let you know if the USB is failing to start up correctly indicating a driver or connection problem.

Labview and Visual Basic create programs that run on a computer and control the MIAC over USB. To do this there is a list of commands that are used to control the MIAC. There is a complete listing of these functions available towards the rear of this document and there are also small sections throughout the worksheets detailing specific commands required for the task.

Example MIAC USB Commands

Command Number	**Description**	**Parameters**	**Returns**
6	Relay Control	Channel (1-4) State (0-1)	0
Command Number	**Description**	**Parameters**	**Returns**
7	Transistor Control	Channel (1-4) State (0-1)	0

Worksheet 1

Basic Outputs

Industrial sensor, actuator and control applications

Digital signals always have two unique voltages associated with them to represent the two unique digital states of 0 and 1. These voltages are commonly 0V and 5V but there are also many others. RS232 for example is a digital communications bus that uses -12V and +12V as its state representation.

Emergency Exit Light

Solenoid Valve

There are many different kinds of lamps, displays and actuators in an industrial system. Collectively these are known as 'output devices'.

Industrial programmable logic controllers (PLCs) have different types of outputs for driving devices with differing current demands.

PLC's typically feature the following output peripherals:

- Low current transistor outputs, (MIAC - labelled A to D).
- Higher current relay outputs, (MIAC - labelled Q_1 to Q_4).

Transistor outputs can vary the current delivered by rapidly pulsing the output, which means they can alter lamp brightness and motor speed.

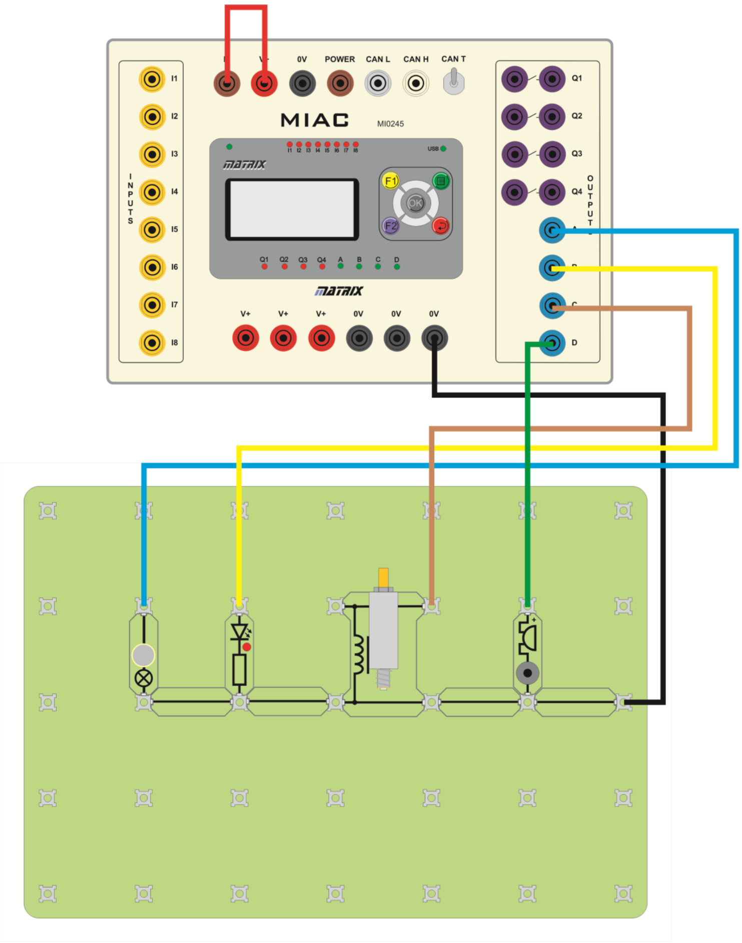
Relay outputs are relatively slow to change and exist in the on or off state. However relays can switch a great deal more current than transistors due to their mechanical nature.

Over to you:

1. Build the system shown opposite.
2. Power the MIAC using a 12V D.C. supply
3. Either a) Using Flowcode or b) using Labview or Visual Basic create a simple program to go through each of the transistor outputs switching one on at a time.
   Note: To allow the VB or Labview code to work the MIAC slave driver must be reloaded onto the MIAC.

Taking this further:

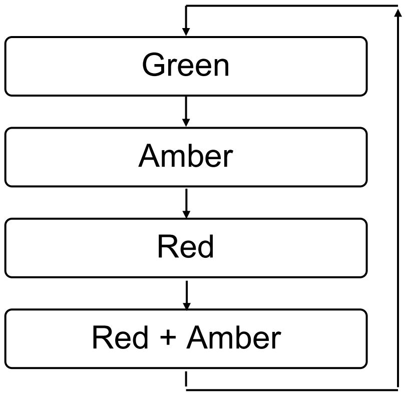
To take this further connect the output devices to the relay outputs and modify your programs to take control of the relays.



Worksheet 2

Sequenced Outputs

Industrial sensor, actuator and control applications



A useful function for a PLC is to be able to switch between a series of states. A common term for this is a finite state machine where the system can be in any one of a finite number of preset states. To the left you can see a finite state machine for a generic traffic light system.

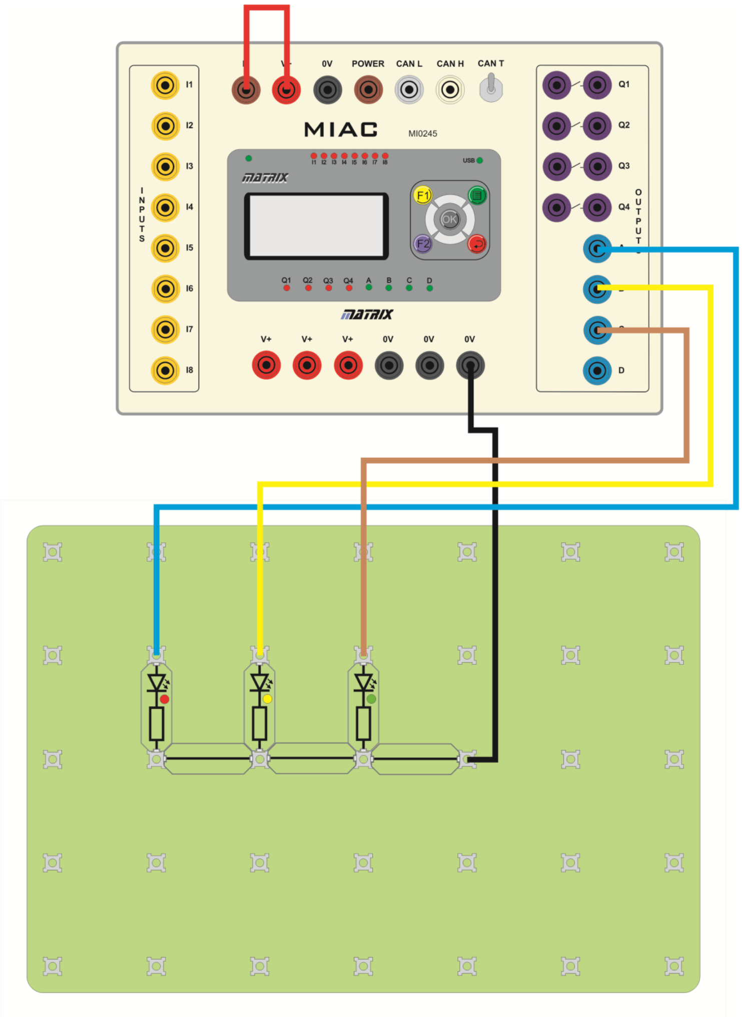
In the system shown there are four unique states that the lights can be in. Therefore the system we will design will need to be able to switch between these four states assigning different values to the outputs depending on the state they are in.

Over to you:

1. Build the system shown opposite.
2. Using Flowcode, Labview or Visual Basic create a simple program to control the LEDs to simulate a set of traffic signals.
3. Start by assigning output values for each LED for a single state.
4. Add a counter to keep track of your current state and then start adding decisions into your program based on the state.
5. Modify your output values depending on the current state.

Taking this further:

To take this further allow for the time delays between signal changes to vary to allow for a more lifelike system.



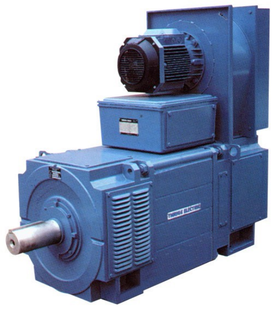
Finite state machines (FSM) are used a great deal in control systems to switch between different states or modes of operation. FSMs are so popular in industrial systems because you can use them to plan and control the operation of the task from start to finish.

Here is a basic FSM for making a cup of tea. Note the state and then the associated decision to move to the next state
Fill Kettle, Kettle Full?, Power Kettle, Water Boiled?, Pour Water, Teapot Full?, Add Teabag, Tea Brewed?.......

Worksheet 3

Pulse Width Modulation

Industrial sensor, actuator and control applications



12A D.C. Motor

So far we have considered circuits that provide only two levels of power to an output. Fully on or fully off.

However, in many situations, we want to control the speed of the motor and this requires a new technique.

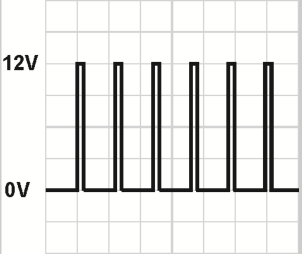
To vary the speed of a motor you need to vary the **power** supplied to it. The two easiest ways of doing this are to vary the voltage, or vary the **duty cycle** of the motor supply. Varying the duty cycle is the most common method and this is explained here.

Speed control using a varying voltage:

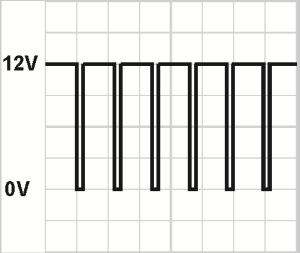
- when the MIAC output is a constant 0V then there is no power supplied to the motor;
- when the MIAC output is a constant 12V then maximum power is supplied to the motor.

Speed control using a pulsed output to the motor.

In the first oscilloscope trace, on the right, you can see that the duty cycle transistor output is seldom on (10% Mark / 90% Space). The power transferred is small, and therefore the resulting speed is low.



In the second trace, you can see that the duty cycle transistor output is mostly on (90% Mark / 10% Space). The power supplied to the motor is much higher, and therefore the resulting speed is high.



This technique of pulsing the output to vary the speed is known as Pulse Width Modulation or PWM for short.

The ratio of the time for which the pulse voltage is high to the time for which the pulse voltage is low is called the **’duty cycle’.**

Worksheet 3

Pulse Width Modulation

Most modern industrial control units will have some form of PWM output.

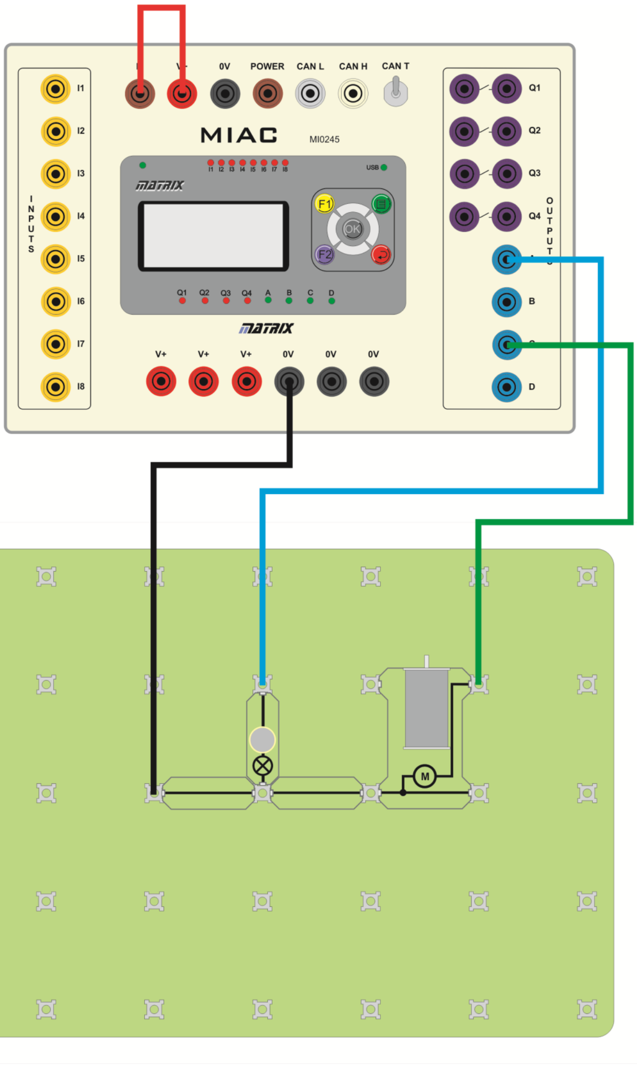
The MIAC has two hardware driven PWM channels connected to transistor outputs A & C.

Over to you:

1. Build the system shown opposite.
2. Using Flowcode, Labview or Visual Basic create a simple program to enable PWM on channel 1 (A) and then ramp up the duty from 0 to 255 and repeat.
3. Next repeat this exercise for PWM output channel 2 (C).
4. Take the black wire, disconnect it from the 0V terminal and connect it to output terminal B.
5. Switch on output B and monitor what happens to the Bulb and the motor as the PWM ramps up.

Taking this further:

By adding a switch and a potentiometer to your circuit try to create a direction and speed control for the motor.



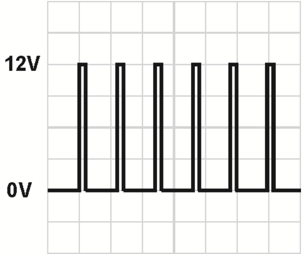
Using output B we now have control of both the speed direction of the motor.

When output B is low the current is supplied from the PWM channel and exits via output B.

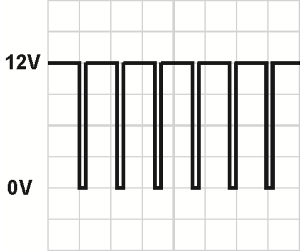
When output B is high the current is supplied from the output and exits via the PWM.

For the two examples below determine the motors direction and speed.

Output B = 1 / PWM 10% Mark & 90% Space



Output B = 0 / PWM 90% Mark & 10% Space



Worksheet 4

Basic Inputs

Industrial sensor, actuator and control applications

Indicator stalk houses several types of switch

Pressure switch

There are many different kinds of inputs and sensors in a modern industrial system. Some are controlled by an operator (like a light switch) and some by factors in an industrial process itself - like a temperature sensor.

Each sensor provides an input signal - often directly into an industrial PLC.

The sensors in any industrial system can be divided into two types: **analogue** and **digital**
The inputs on the MIAC PLC are compatible with both analogue and digital signals.

Digital sensors have a two state output, usually either 'on' or 'off'. The power supply used determines the voltages corresponding to these two states - often 10V (on) and 0V (off).

Analogue sensors have a continuous output normally ranging between the supply voltage and 0V. Analogue sensors will be covered later in the course.

Commands used in this worksheet

Command Number	**Description**	**Parameters**	**Returns**
1	Digital Input	Channel (1-8)	Input State (0-1)
Command Number	**Description**	**Parameters**	**Returns**
2	Digital Multi Input	-	Input I1 (0-1)
			Input I2 (0-1)
			Input I3 (0-1)
			Input I4 (0-1)
			Input I5 (0-1)
			Input I6 (0-1)
			Input I7 (0-1)
			Input I8 (0-1)

Single digital input channels are sampled by issuing command number 1.
The entire set of input channels can be sampled by issuing command number 2.

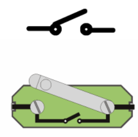
Worksheet 4

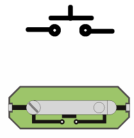
Basic Inputs

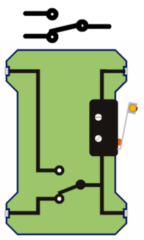
Industrial sensor, actuator and control applications

Like digital outputs, digital inputs have two distinct voltages to represent the binary values of 0 and 1. These voltages must be decided before designing your system to match the peripherals and other devices attached to the system. PLCs generally use either 12V or 24V for a logical 1 input voltage and 0V for the logical 0 input voltage.

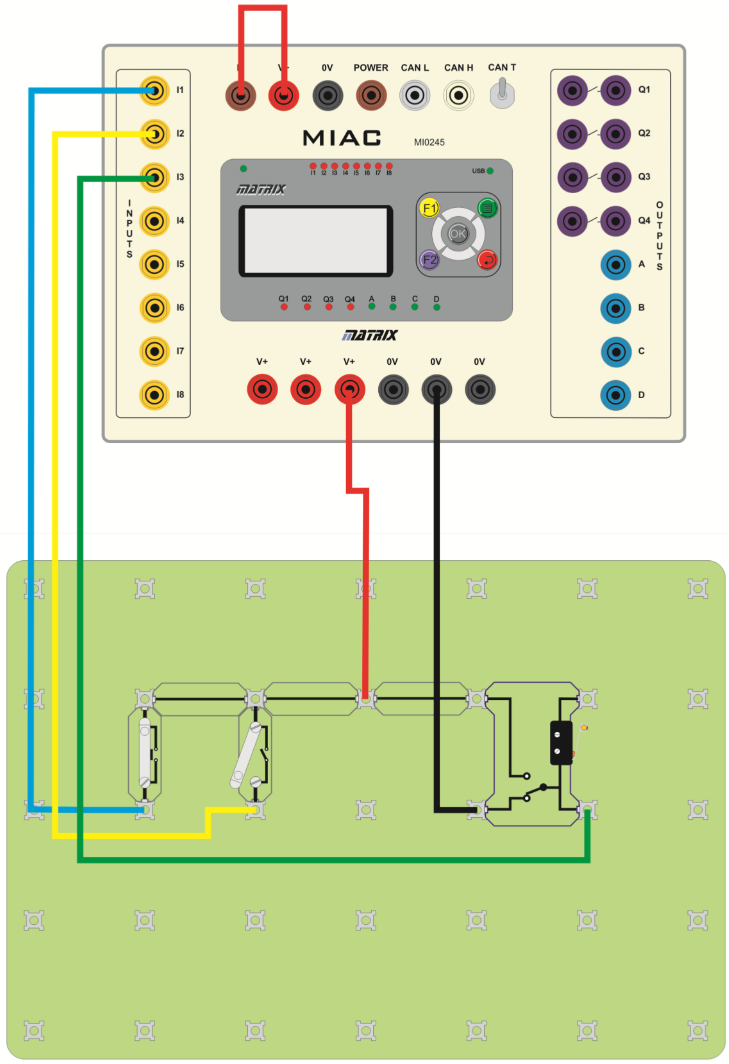
Symbols and components


Toggle switch


Momentary switch


SPDT switch

The circuit symbols for the components can be seen clearly on the carriers.



Over to you:

1. Build the system shown opposite.
2. Using Flowcode, Labview or Visual Basic create a simple program to read the digital state of the switches.
3. Display the switch values on the screen if using VB or Labview, or on the MIAC's LCD if using Flowcode.

Taking this further:

To take this further you can add more switches to your circuit and then modify your program to sample all the switches at once.

The toggle switch and momentary switch allow for a signal or voltage to be connected or disconnected.

The single pole double throw (SPDT) switch allows for a selection between two signals or voltages. Eg 0V and 12V.

Worksheet 5

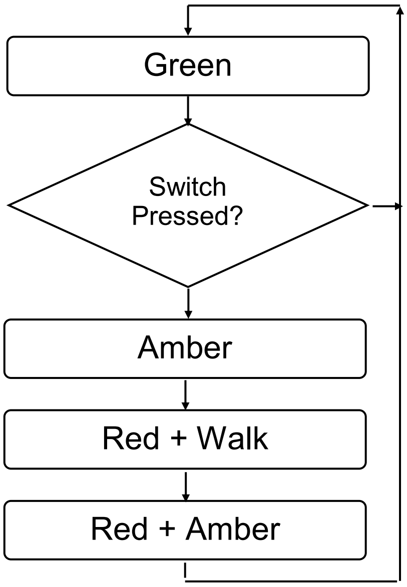
Pelican Crossing

Industrial sensor, actuator and control applications

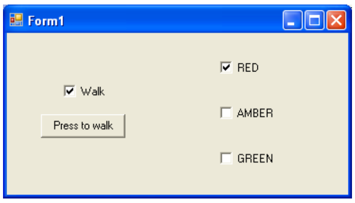
By combining the previous worksheets we can now create a fully functional pelican crossing system.

Using Flowcode to design our programs we can create a low cost embedded device to control the system.

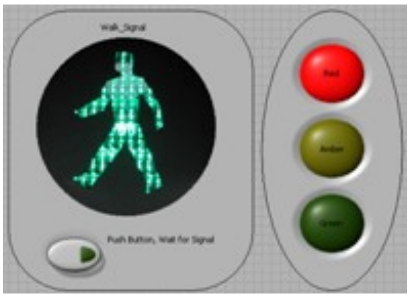
Using Labview or Visual Basic we can create a high performance front end to our embedded system which will let us monitor and control the operation of the system in real time on a PC.



Example Pelican Crossing state Machine



Visual Basic Pelican Crossing Example



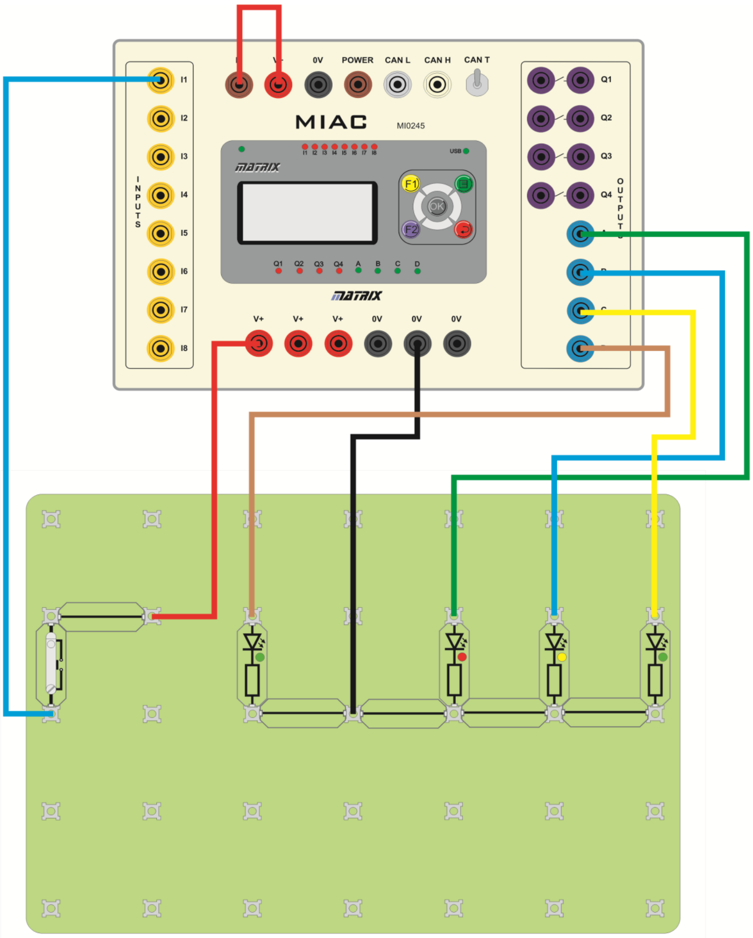
Labview Pelican Crossing Example

Over to you:

1. Build the system shown opposite.
2. Using Flowcode, Labview or Visual Basic create a simple program to control the state of all four signal lights.
3. Next poll the input with the press switch connected to stop and start the state machine

Taking this further:

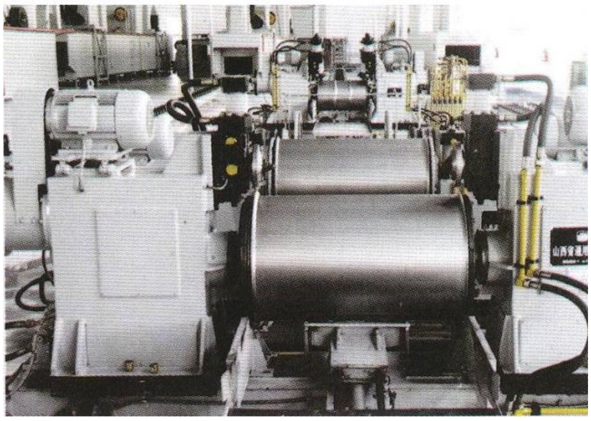
Add more states to the state machine to allow the pedestrian walk light to flash, warning the user that the lights are about to change.



Worksheet 6

Conveyer Belt

Industrial sensor, actuator and control applications



Production line using stepper motors

DC motors are cheap and reliable but are not suitable in systems which need a measured and precise amount of movement or rotation.

Car Idle valve stepper motor

Stepper motors use four internal windings. As the name suggests, these allow the rotor to be moved in small steps - forwards or backwards. Stepper motors vary in the number of steps they provide in one full rotation. The stepper motor we use takes 48 separate steps per revolution, giving it a positional accuracy of 7.5 degrees.

Stepper Motor Full Step Profile

	A	B	C	D
Step 1	0	0	0	1
Step 2	0	1	0	0
Step 3	0	0	1	0
Step 4	1	0	0	0

The four coils in a stepper motor must be energised in the correct sequence to allow the motor to rotate correctly.

Reversing the sequence allows the motor to turn in the opposite direction.

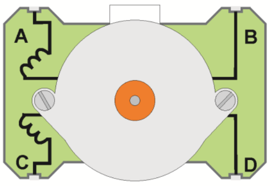
As the stepper motor uses 4 distinct steps to allow it to rotate this can again be used to create a FSM.

When the coils in a stepper motor are energised, the motor will rotate to the new position defined by the step profile. If the coils remain energised then the motor will be held in a fixed position. This is called holding torque and is useful for quickly stopping the motor or for retaining a shaft in a fixed position.

Warning: if the coils remain energised for too long then they will begin generating heat. This heat build up may decrease the life span of the motor so it is vital to switch off all outputs to the motor as soon as the motor stops moving.

New symbols and components

Stepper motor



Worksheet 6

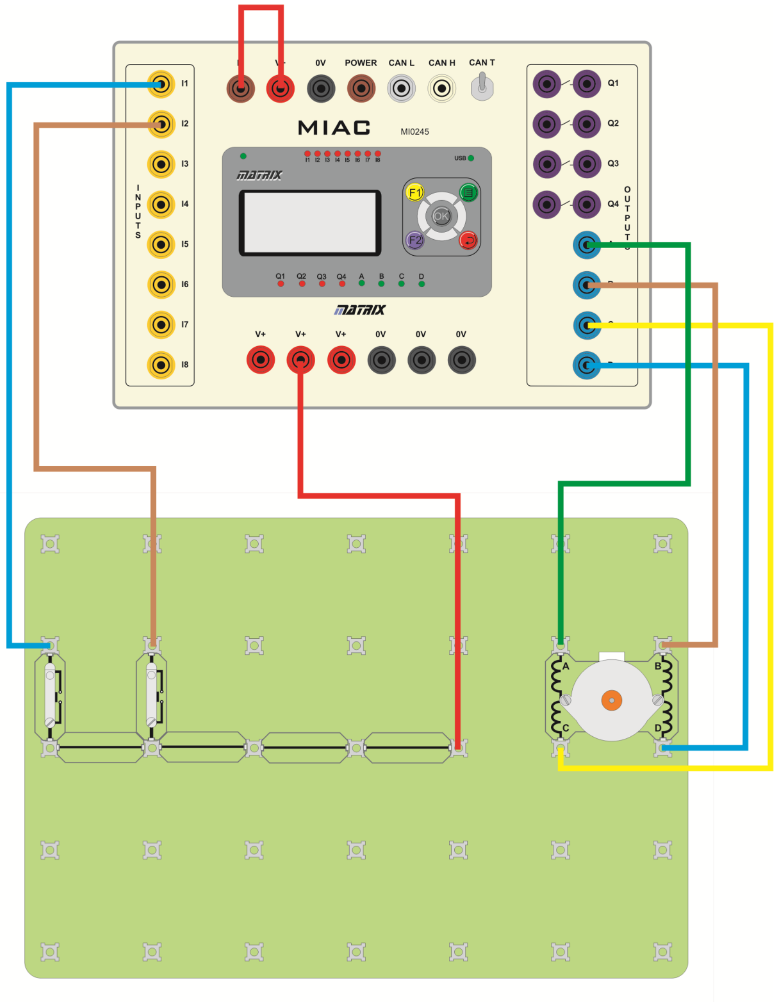
Conveyer Belt

Over to you:

1. Build the system shown opposite.
2. Using Flowcode, Labview or Visual Basic create a simple program that will rotate a stepper motor.
3. Next modify your program so that the motor only rotates when the first switch is pressed, switch off all four outputs when the switch is not pressed.
4. Next modify your program so that when the second switch is pressed the motor will rotate opposite direction.
5. Using the table below add a further 4 steps to your state machine to allow the motor to half step.

Taking this further:

To take this further add two other switches into the system and use them to speed up or slow down the motor. Alternatively you could use the switches to store and restore the current motor angle.



Half stepping effectively doubles the resolution of the stepper motor by alternating between one and two active coils.

Using half stepping the stepper motor we use takes 96 separate steps per revolution, giving it a positional accuracy of 3.75 degrees.

Stepper Motor Half Step Profile

	A	B	C	D
Step 1	1	0	0	1
Step 2	0	0	0	1
Step 3	0	1	0	1
Step 4	0	1	0	0
Step 5	0	1	1	0
Step 6	0	0	1	0
Step 7	1	0	1	0
Step 8	1	0	0	0

Worksheet 7

Analogue sensing

Industrial sensor, actuator and control applications

Fuel sensor with float

There are two types of sensor, **analogue** and **digital**.

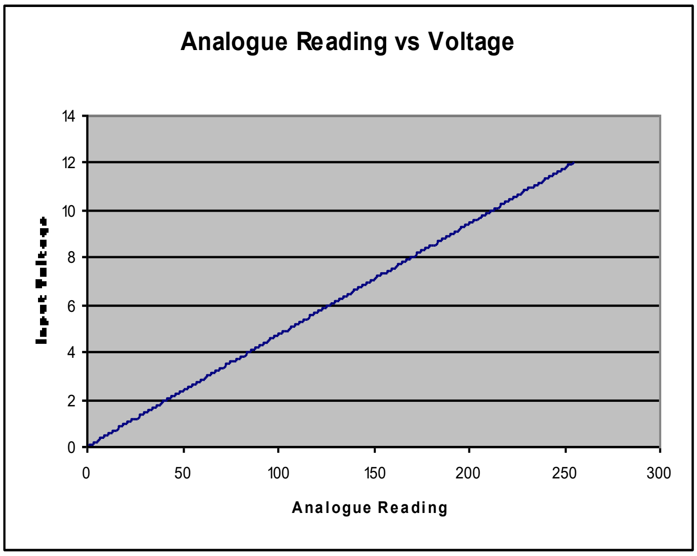
Analogue sensors provide more information than just the sensor being ’on’ or ’off’ and can be used to more accurately interpret the state of a system. They provide information about how full?, how far?, how many?, how hot? etc.

Voltage	Analogue Reading
0	0
2	42
4	85
6	128
8	171
10	213
12	255

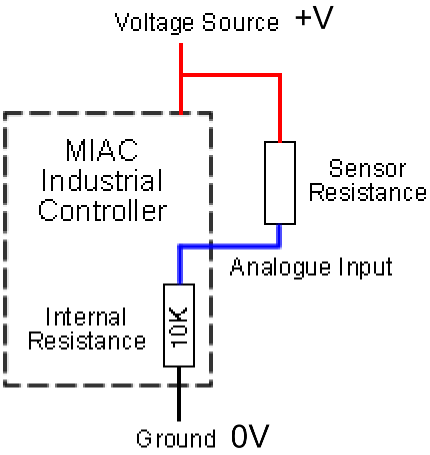
To allow a analogue signal to enter a digital based system such as a PLC or microcontroller we first have to convert the analogue continuous signal into a digital discreet signal.

The MIAC unit converts analogue signals into 8-bit digital bytes so the maximum reading can be 255 which represents an input voltage of 12V.

Analogue to digital converters (ADC) have a fixed resolution measured in bits. That is the number of binary 1s and 0s used to store the resulting signal. The MIAC is capable of generating 10-bit ADC conversions or 1024 potential signal levels. However for simplicity we have limited the MIAC to 8-bit ADC measurements or 256 potential signal levels.



ADC Conversion



Example MIAC analogue input circuit

Most analogue sensors work by varying a resistance. A potential divider circuit generates a voltage that is proportional to the resistance of the analogue sensor. To reduce the cost and number of external parts the other half of the potential divider circuitry is inside the MIAC industrial controller unit.

MIAC Slave Input Commands

Command Number	Description	Parameters	Returns
3	Analog Input	Channel(1-8)	Input(0-255)

Analogue input channels are measured (sampled) by issuing command number 3.

Worksheet 7

Analogue sensing

Industrial sensor, actuator and control applications

Converting an analogue reading to a sensor reading

An analogue input reading simply tells you the voltage present on the input pin of the device. There may be a fairly complex calculation required to convert the voltage reading to a useful sensor reading.

Here is the calculation for the Locktronics thermistor when used with the MIAC PLC device.

Resistance = 10000 x ((12 / (0.05 x ADC_Reading)) - 1)

Temperature = (1 / ((1 / 298.15) + ((1 / 3950) x LN(Resistance / 4700)))) - 273.5

Due to the fact that the calculation is fairly complex we would be better off processing the data using the PC side software rather then trying to do the calculation on the actual PLC or micro-controller. This allows our embedded device to perform much faster.

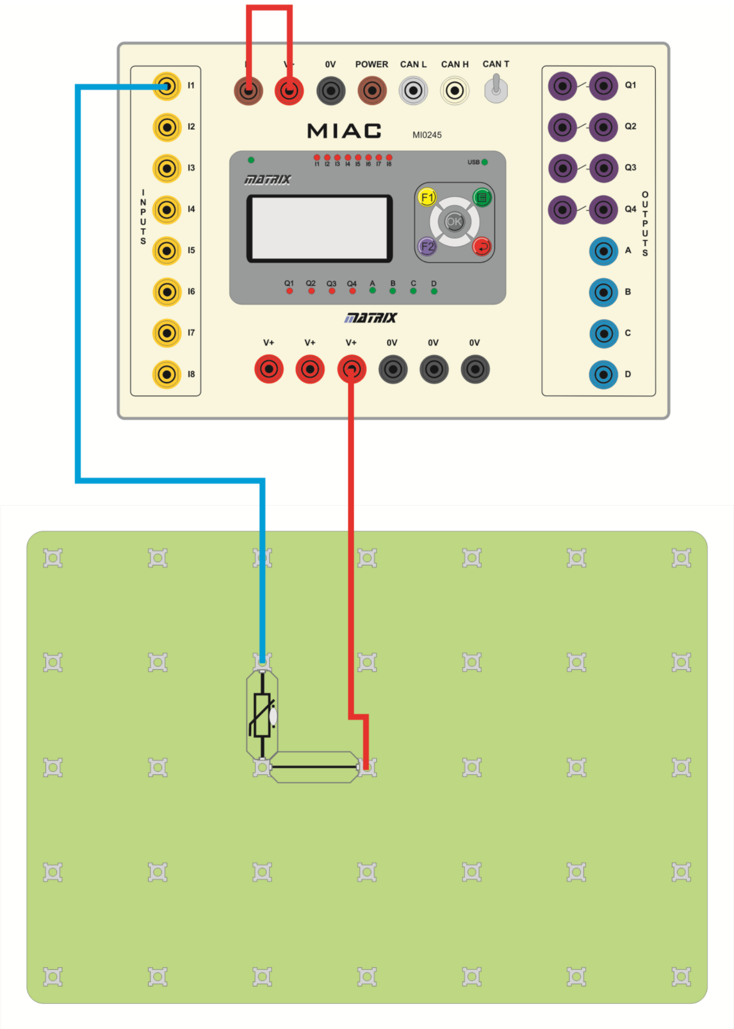
One way to perform such a calculation on an embedded device would be to use a lookup table. In a lookup table the temperature data is pre-calculated for each of the 256 possible analogue input values. The embedded device then simply has to pull a temperature value out of memory that corresponds to the current analogue reading.

Over to you:

1. Build the system shown opposite. The thermistor carrier is an analogue sensor. It is equivalent to the temperature sensor in a thermostat or freezer. Touching the temperature sensor is like increasing the heat inside the appliance.
2. Using Flowcode, Labview or Visual Basic create an application to sample the analogue sensor and display the result.
3. Use the calculation shown above to convert the analogue voltage reading into a temperature reading.

Taking this further:

To take this further you try and graph the temperature readings against time or switch LEDs on or off depending on the temperature.



Worksheet 8

Detecting Faults 1

Industrial sensor, actuator and control applications

In modern industrial systems, PLCs are used for much more than control. They can also report on the status of the system and many of its components.

In this landing gear system, the PLC can detect if the main landing gear is damaged and therefore inform the pilot and flight control so that they can both react accordingly.

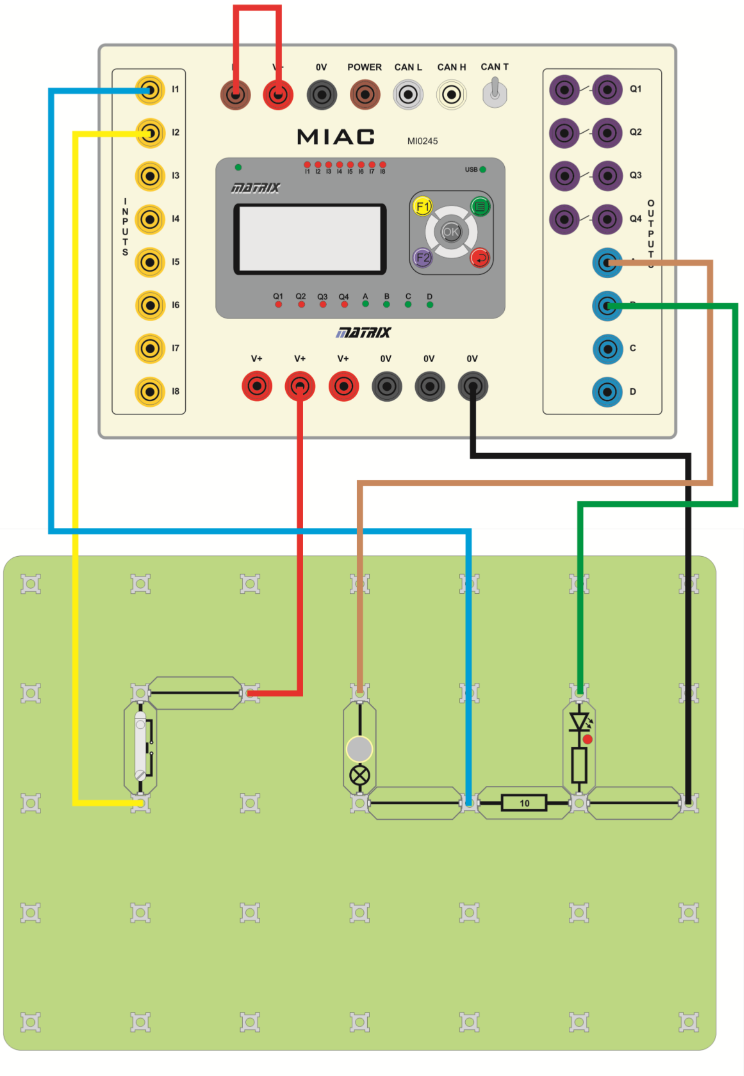
Over to you:

1. Build the system shown opposite. It contains a resistor that allows the warning bulb to be monitored.
2. Using Flowcode, Labview or Visual Basic create a program that allows you to switch the bulb on and off at will using the digital switch connected to I1.
3. Add an analogue sample function to your program for input channel I2 and watch the analogue value change as you switch the lamp on and off.
4. Add an output to your program to switch on the red LED if the analogue signal does not respond when the lamp is switched on, remove the bulb and test to ensure your broken bulb detector is working correctly.

Taking this further:

To take this further add a second bulb and resistor combination and use two analogue inputs to detect which bulb is missing or damaged.

Warning or hazard lights also need to be verified to ensure that people are aware of the warning or hazard in question. If the light fails to come on then a siren can sound indicating the light failure.



Modern industrial systems are designed with stringent fault detection in mind. Can you think of any other applications where the PLC should automatically react to faults?

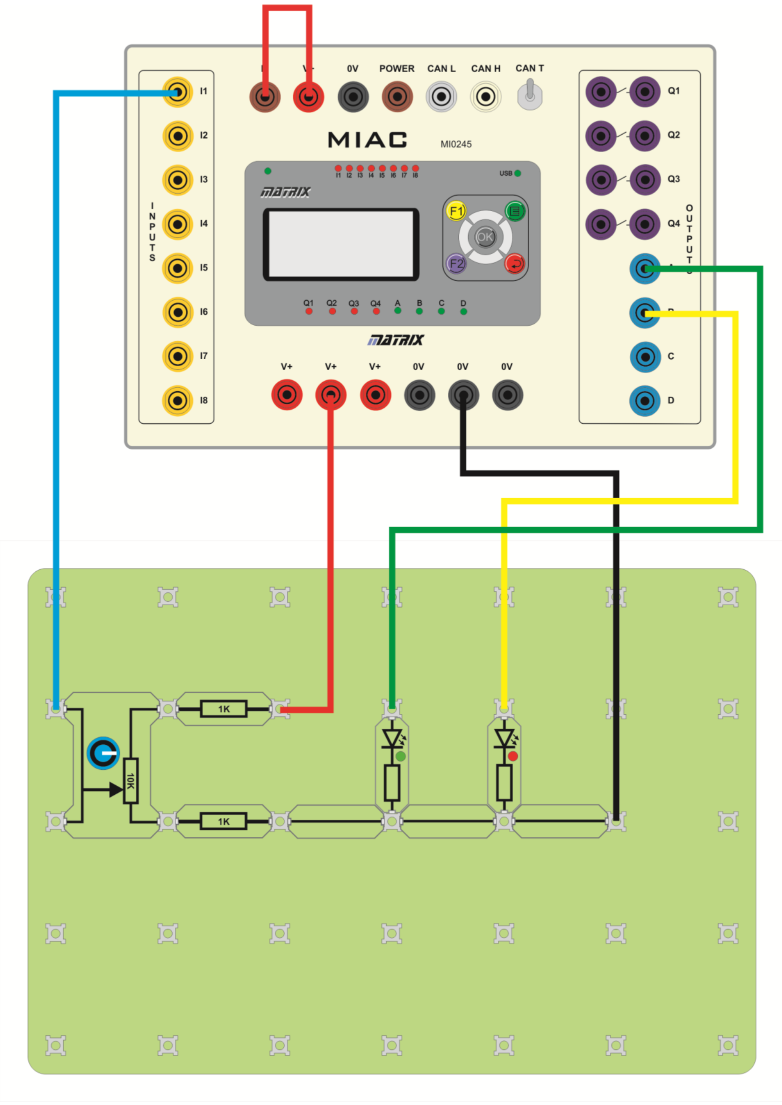
Worksheet 9

Detecting Faults 2

Industrial sensor, actuator and control applications

It is often difficult to detect a faulty reading when using analogue sensors. If a sensor is shorted to one of the power supply rails then the input into the PLC would still resemble a valid signal. We can improve this by adding a section of the analogue signal that is effectively out of bounds.

Industrial Press



Over to you:

1. Build the system shown opposite. It contains a potentiometer which we can assume is supplying our analogue sensor signal.
2. Create a Flowcode, Labview or Visual Basic program to read in the signal from the potentiometer.
3. Light up the green LED if the analogue signal is between the values 50 and 200.
4. Light up the red LED if the analogue signal is outside of the values mentioned above.
5. Short input I1 to +V and 0V and make sure that the system is detecting the fault correctly.

Taking this further:

Can you think of any other faults that could occur in a system? How could you use electronics to allow your controller to check for these faults.

In fault critical systems such as rockets and industrial processes where a fault could have catastrophic consequences it is important to track and keep on top of all problems as they develop.

Some manufacturers use multiple sensors to monitor the same events, then if the sensor reading do not tally from sensor to sensor then the PLC knows there is a problem.

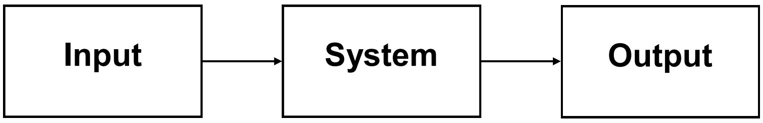
In more advanced fault detection systems there are multiple PLCs as well as multiple sensors. Each PLC then votes as to what it thinks the valid sensor reading should be, the votes are compared and the majority is accepted. Again the PLC monitoring the votes will know there is a problem if the votes do not tally correctly.

Worksheet 10

Open Loop Control

Industrial sensor, actuator and control applications

Open loop control system

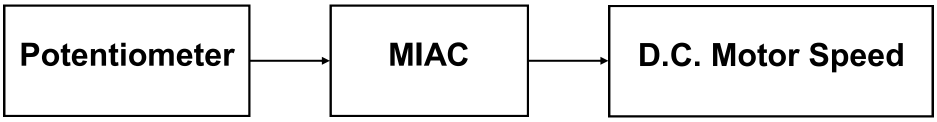


The two main types of control systems used in a PLC are **open loop** and **closed loop**.

In a open loop system, the output state of the system is being controlled but is never monitored to make sure it is correct.

In a closed loop system, the output state of the system is fed back into the input, so that the system can check when the desired outcome has been attained. More on this later.

Worksheet 10 Open loop control system



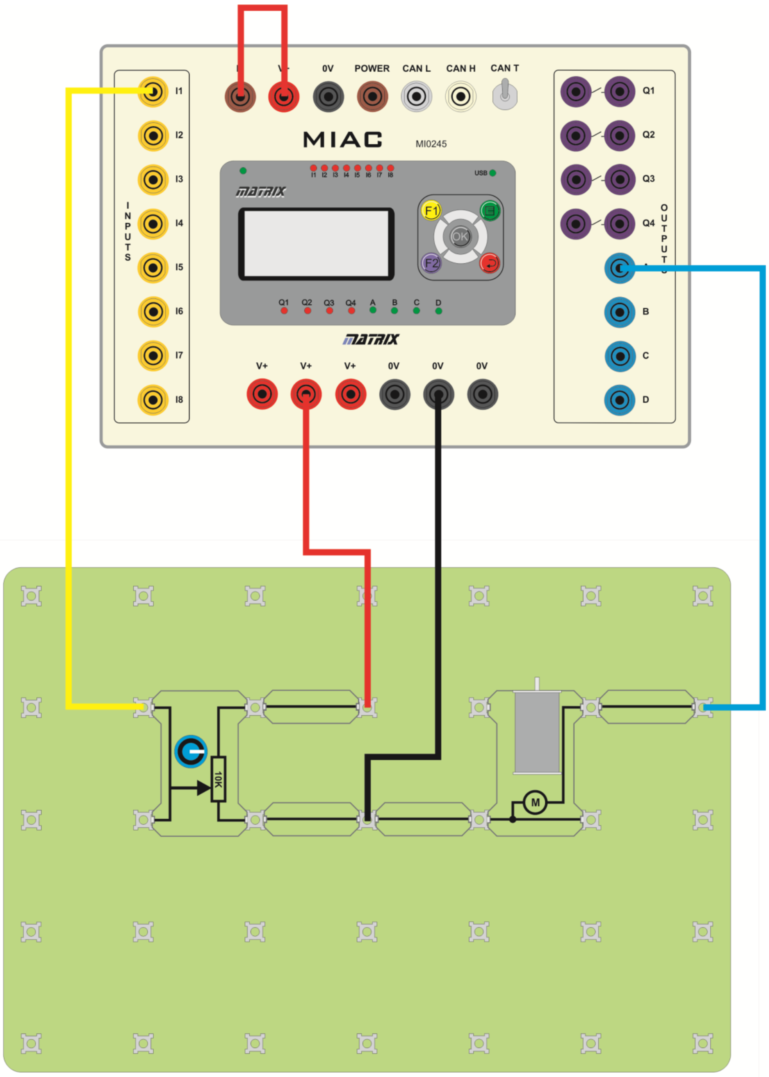
Over to you:

1. Build the system shown opposite.
2. Create a Flowcode, Labview or Visual Basic program to read in the signal from the potentiometer.
3. Forward the analogue signal from the potentiometer to the PWM driving the Motor.

Taking this further:

At the moment we have no way of knowing the motor's actual speed, its shaft position or even if the motor is connected to the system at all.

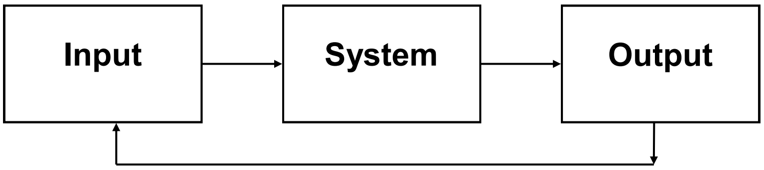
Suggest a way of improving this control system by adding an extra sensor to close the loop and allow feedback from the motor output.



Worksheet 11

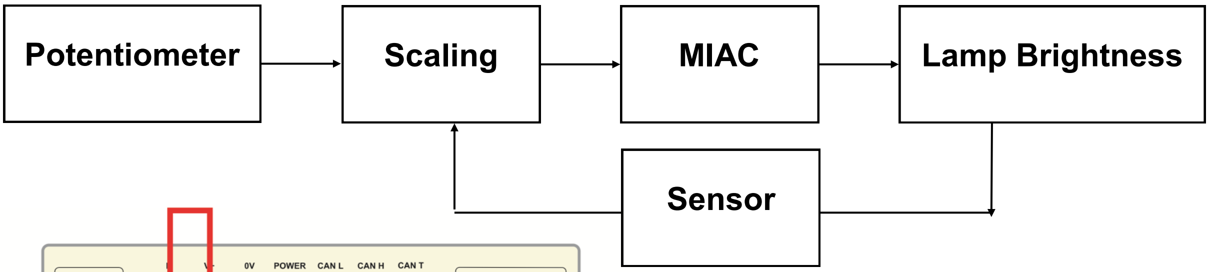
Closed Loop Control

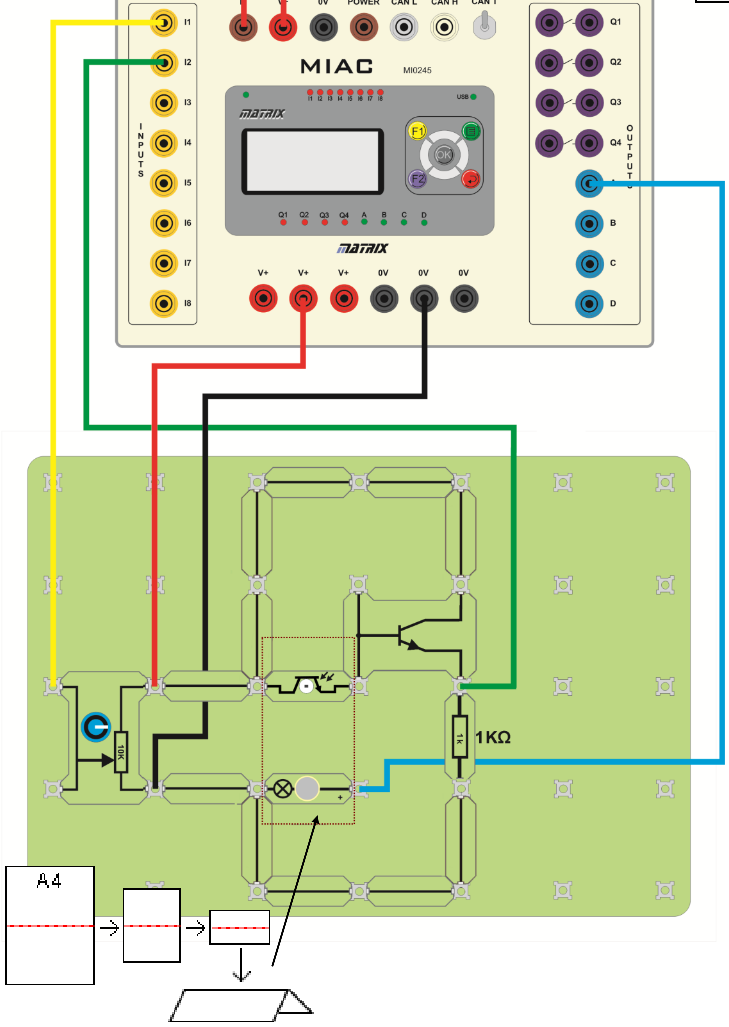
Closed loop control system



The light level sensor circuit shown below is a closed loop. The lamp lights up the area. A signal from the light sensor (phototransistor), indicating the current light level, is fed back to the system and compared with the desired light level, set by the potentiometer signal. The system can then know if and when the desired light level has been reached.

Worksheet 11 Closed loop control system





Over to you:

1. Build the system shown opposite.
2. Create a Flowcode, Labview or Visual Basic program to read in the signal from the potentiometer.
3. Forward the analogue signal from the potentiometer to the PWM driving the Lamp.
4. Modify the analogue signal to combine the signal from the LDR. The signal should be 90% potentiometer and 10% LDR error.

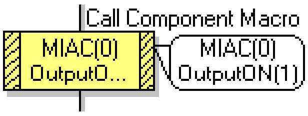
Taking this further:

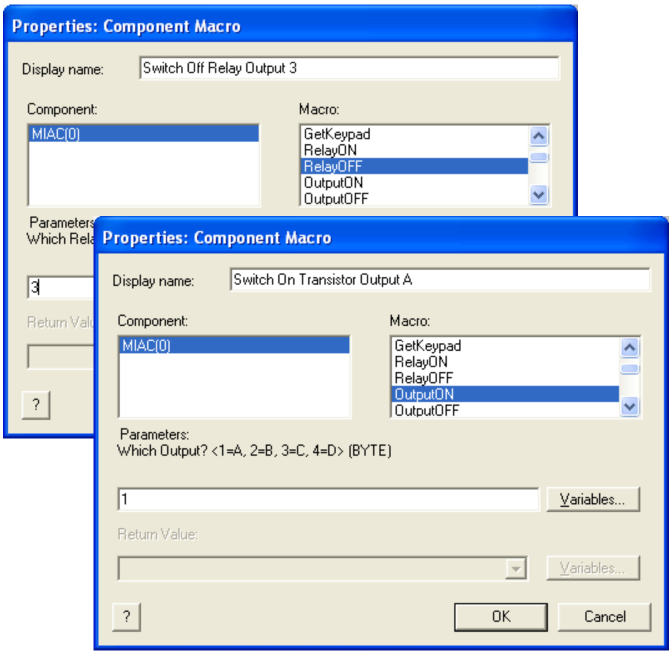
Suggest another closed loop control system and use the MIAC PLC and the Locktronics components to build and test your proposed system.

Controlling an output using Flowcode

Flowcode uses icons called component macros to call pre-made inbuilt functions on the MIAC.

Functions are available which allow both the transistor or relay outputs to be switched on or off digitally.





Properties: Component Macro

Display name: Switch Off Relay Output 3

Component: MIAC(0)

Macro: GetKeypad, RelayON, RelayOFF, OutputON, OutputOFF

Parameters:
Which Rela

3

Return Valu

Properties: Component Macro

Display name: Switch On Transistor Output A

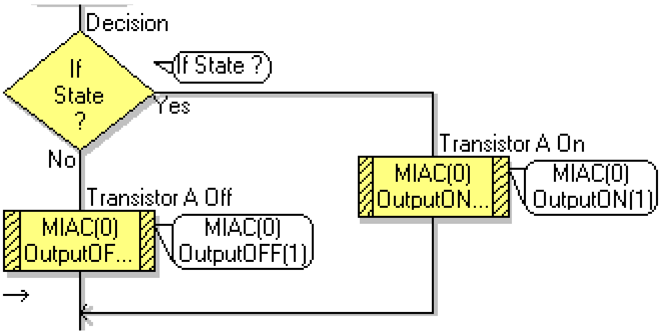
Component: MIAC(0)

Macro: GetKeypad, RelayON, RelayOFF, OutputON, OutputOFF

Parameters:
Which Output? <1=A, 2=B, 3=C, 4=D> (BYTE)

1

Return Value:

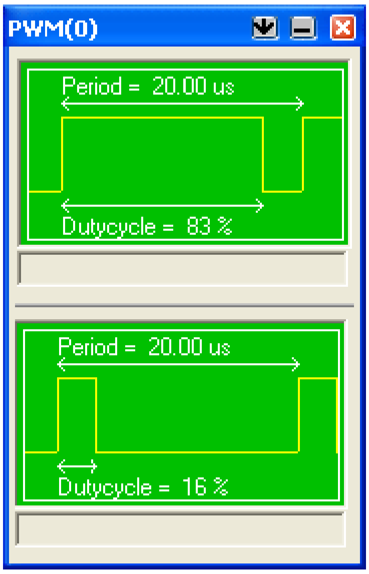


Double clicking on the component icon will cause the icon properties to open allowing the component function to be selected and parameters to be passed.

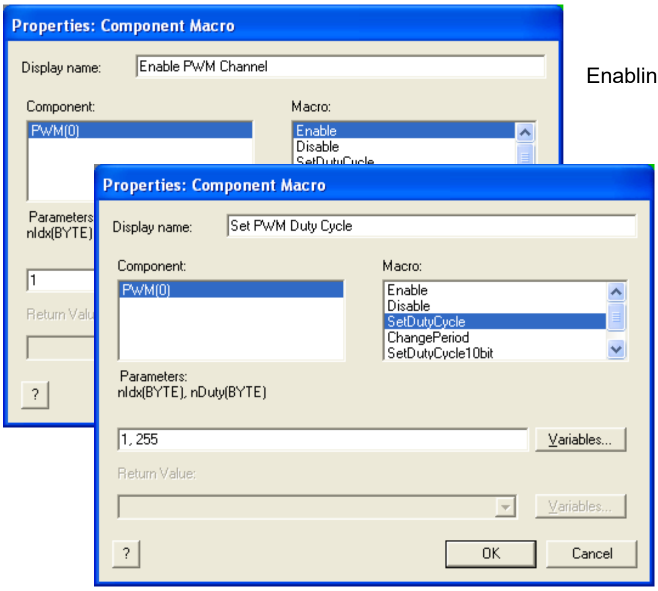
Controlling a PWM channel using Flowcode

Flowcode uses icons called component macros to call pre-made inbuilt functions on the MIAC.

Functions are available which allow two of the transistor outputs (A & C) to output pulse width modulated digital signals.



Flowcode PWM Component



Properties: Component Macro

Display name: Enable PWM Channel

Component: PWM(0)

Macro: Enable, Disable, SetDutyCycle

Parameters:
nIdx(BYTE)

1

Return Valu

Properties: Component Macro

Display name: Set PWM Duty Cycle

Component: PWM(0)

Macro: Enable, Disable, SetDutyCycle, ChangePeriod, SetDutyCycle10bit

Parameters:
nIdx(BYTE), nDuty(BYTE)

1, 255

Return Value:

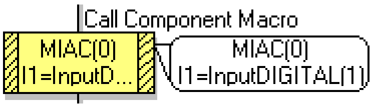
Enabling a PWM Channel

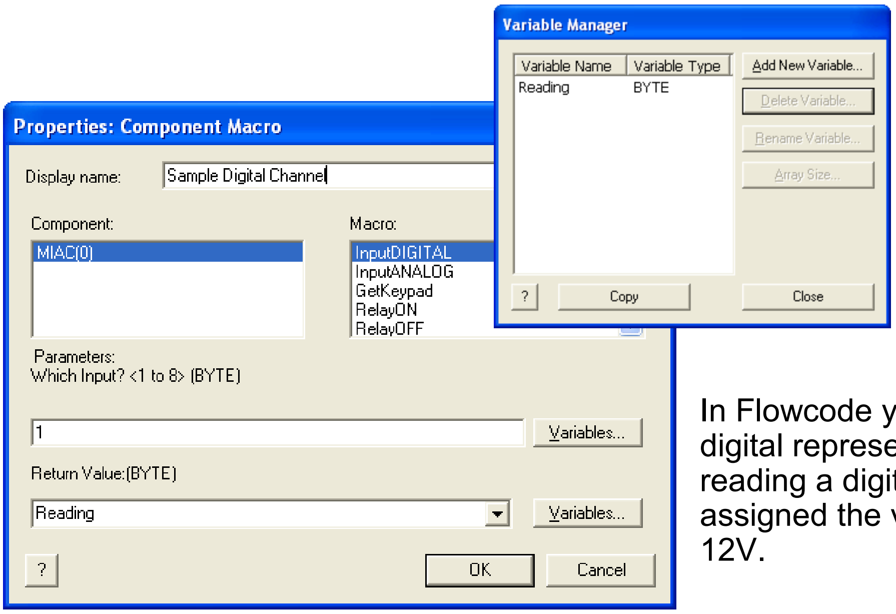
Changing the duty cycle for a PWM channel

Reading an input using Flowcode

Flowcode uses icons called component macros to call pre-made inbuilt functions on the MIAC.

Functions are available which allow inputs to be read as digital or analogue representations of the incoming signal.





Variable Name	Variable Type
Reading	BYTE

Display name: Sample Digital Channel

Component: MIAC(0)

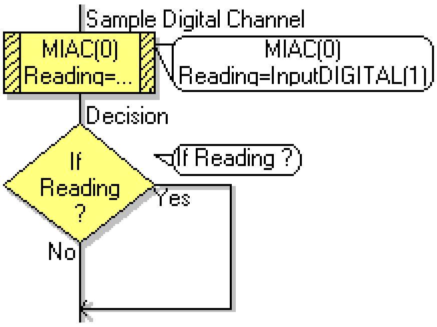
Macro: InputDIGITAL
InputANALOG
GetKeypad
RelayON
RelayOFF

Parameters:
Which Input? <1 to 8> (BYTE)

1

Return Value:(BYTE)

Reading

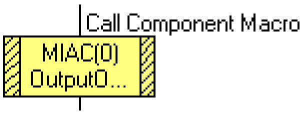


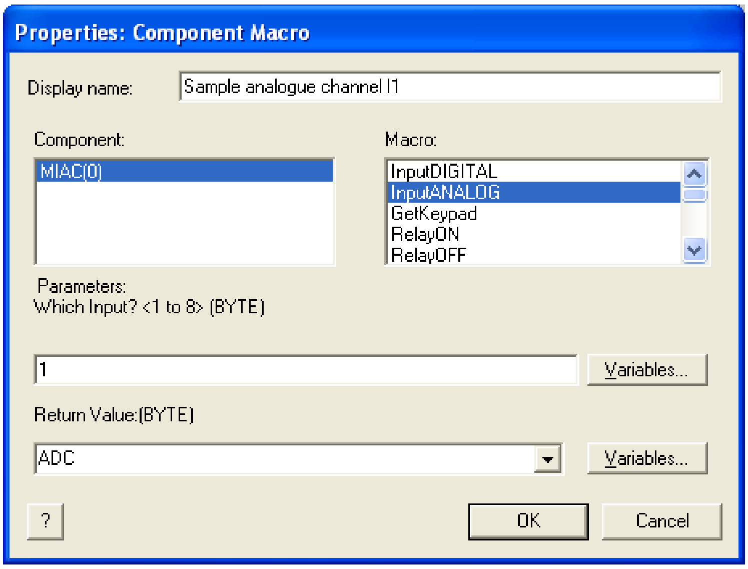
In Flowcode you require a variable to store the digital representation of the input. If you are reading a digital input then the variable will be assigned the value of 0 or 1 to represent 0V or 12V.

Reading an analogue input using Flowcode

Flowcode uses icons called component macros to call pre-made inbuilt functions on the MIAC.

Functions are available which allow the input channels to be sampled using an onboard analogue to digital converter (ADC).





Display name: Sample analogue channel I1

Component: MIAC(0)

Macro: InputDIGITAL
InputANALOG
GetKeypad
RelayON
RelayOFF

Parameters:
Which Input? <1 to 8> (BYTE)

1

Return Value:(BYTE)

ADC

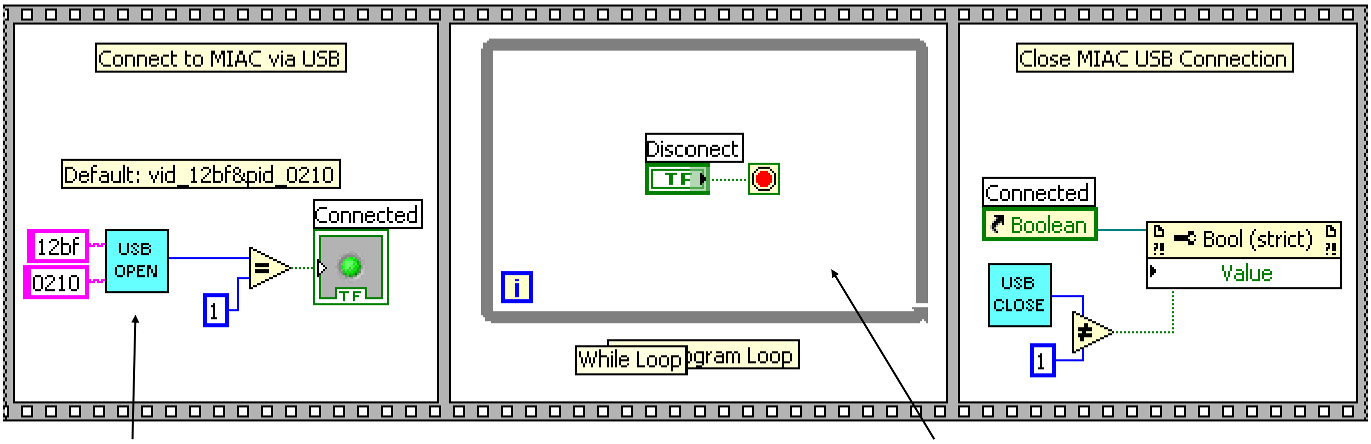
Interfacing with Labview

A program in Labview is made up of a front panel and a block diagram.

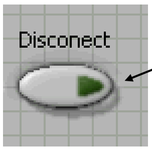
The front panel houses all of your front end graphical user interface where as the block diagram contains all the code to drive the program functionality.

The example Labview program uses a structured case statement to connect to the MIAC perform an operation and then disconnect.

Connect Process Disconnect



Several Pre-Made Labview VIs have been created to allow for easy MIAC integration. These include a connection routine, send and receive routine and a disconnection routine.



This is a loop that will run forever until you close the program or click on the disconnect button

This is the basic front panel to go with the example block diagram shown above. You notice we have a LED to signify if the MIAC is connected successfully and also a disconnect button to disconnect the MIAC once we have finished using it. This allows it to become available for any other application.

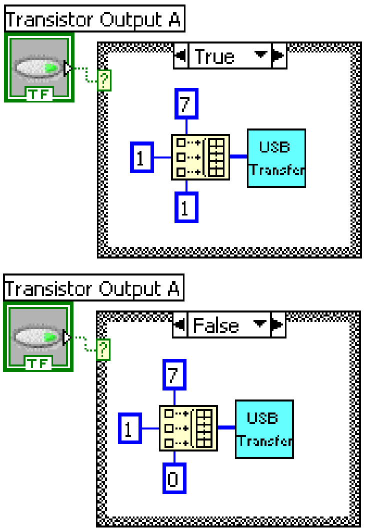
Controlling an output using Labview

Labview communicates to the MIAC device using pre-made data transfer virtual instrument named “USB_Transfer (SubVI).vi” which can be found on the website.

Transistor outputs are controlled using command number 7.
Relay outputs are controlled using command number 6.

Here is an example of how to use a Boolean switch in Labview to control a transistor output on the MIAC.

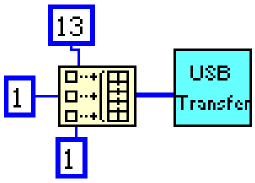
The example sends command 7 to take control of transistor output 1.



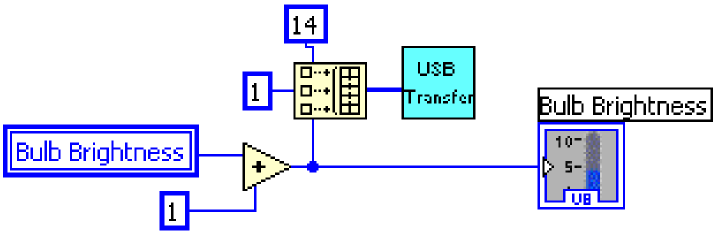
Labview output example

Controlling PWM using Labview

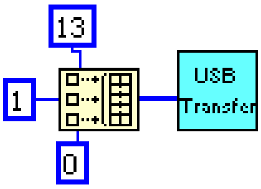
Labview communicates to the MIAC device using pre-made data transfer sub vi named “USB_Transfer (SubVI).vi”.



Enabling PWM



Modifying the PWM duty cycle

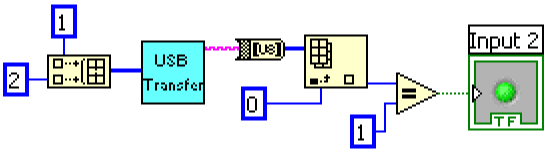


Disabling PWM

Reading an input using Labview

Here is an example of how to read an input on the MIAC and forward to a Boolean LED in Labview.

The example sends command 1 with a channel parameter of 2 to allow the digital representation of input channel I2 to be read.

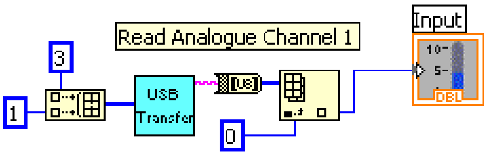


The example collects the data returned by the USB transfer VI and uses a conversion followed by an array index to retrieve the single byte return value. This is then compared with the number 1 to generate the Boolean signal required by the LED indicator.

Reading an analogue input using Labview

Here is an example of how to read an analogue input on the MIAC and forward to a numeric slider in Labview.

The example sends command 3 with a channel parameter of 1 to allow the digital representation of   input channel I1 to be read.

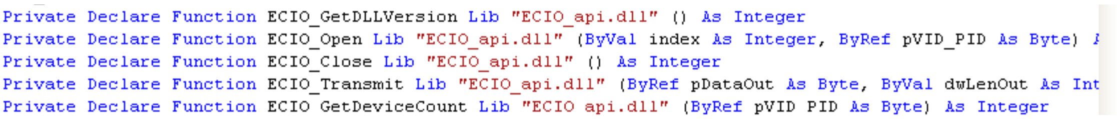


The example collects the data returned by the USB transfer VI and uses a conversion followed by an array index to retrieve the single byte return value. This is then sent directly to the input for the slider.

A program in Visual Basic is made up of a Form and a file containing basic code.

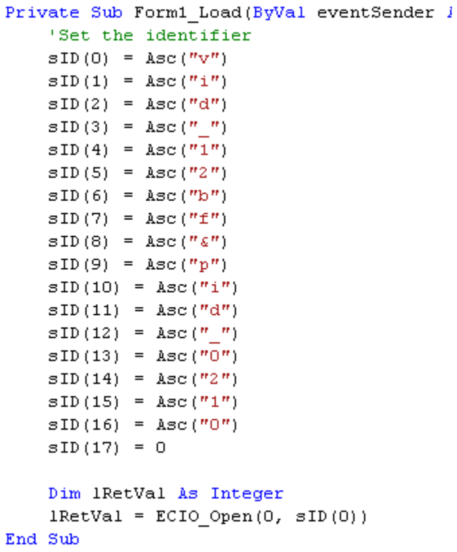
The form houses all of your front end graphical user interface where as the basic code contains all the code to drive the program functionality.

At the top of the basic code file there are a set of definitions that configure Visual Basic to talk to the MIAC PLC.

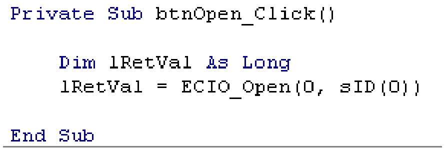


```
Private Declare Function ECIO_GetDLLVersion Lib "ECIO_api.dll" () As Integer
Private Declare Function ECIO_Open Lib "ECIO_api.dll" (ByVal index As Integer, ByRef pVID_PID As Byte) A
Private Declare Function ECIO_Close Lib "ECIO_api.dll" () As Integer
Private Declare Function ECIO_Transmit Lib "ECIO_api.dll" (ByRef pDataOut As Byte, ByVal dwLenOut As Int
Private Declare Function ECIO GetDeviceCount Lib "ECIO api.dll" (ByRef pVID PID As Byte) As Integer
```

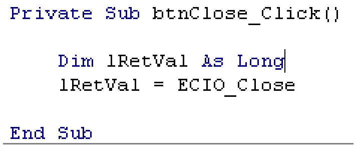
After these definitions is a form load subroutine that is run when the Visual Basic program is started. Inside this subroutine is a set of assignments that initialises the USB address for the MIAC. The form load subroutine also calls the ECIO_Open routine which automatically connects to the MIAC if it is connected to the PC.



```
Private Sub Form1_Load(ByVal eventSender A
    'Set the identifier
    sID(0) = Asc("v")
    sID(1) = Asc("i")
    sID(2) = Asc("d")
    sID(3) = Asc("_")
    sID(4) = Asc("1")
    sID(5) = Asc("2")
    sID(6) = Asc("b")
    sID(7) = Asc("f")
    sID(8) = Asc("&")
    sID(9) = Asc("p")
    sID(10) = Asc("i")
    sID(11) = Asc("d")
    sID(12) = Asc("_")
    sID(13) = Asc("0")
    sID(14) = Asc("2")
    sID(15) = Asc("1")
    sID(16) = Asc("0")
    sID(17) = 0

    Dim lRetVal As Integer
    lRetVal = ECIO_Open(0, sID(0))
End Sub
```



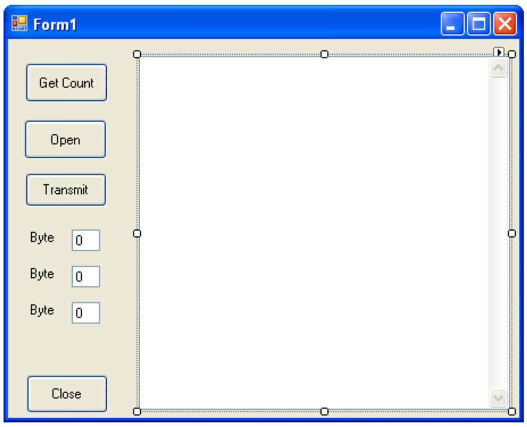
```
Private Sub btnOpen_Click()

    Dim lRetVal As Long
    lRetVal = ECIO_Open(0, sID(0))

End Sub
```



```
Private Sub btnClose_Click()

    Dim lRetVal As Long
    lRetVal = ECIO_Close

End Sub
```

The example Visual Basic application then has subroutines to demonstrate reading and writing to the device and closing the connection to the device.

Elements can be dragged into the form to create text or selection boxes and then referenced in the

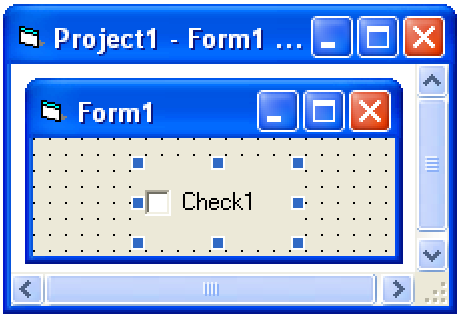


Controlling an output using Visual Basic

Visual Basic communicates to the MIAC device using pre-made data transfer routine named “ECIO_Transmit”.

Transistor outputs are controlled using command number 7.
Relay outputs are controlled using command number 6.

```
Private Sub Check1_Click()

    Dim lRetVal As Long
    Dim bDataOut(0 To 2) As Byte
    Dim bDataIn(0 To 2) As Byte
    Dim nTxDataCnt As Long
    Dim nRxDataCnt As Long

    bDataOut(0) = 7            'Digital Output Transistor Command
    bDataOut(1) = 1            'Output To channel A
    bDataOut(2) = Check1.Value 'Value of button

    lRetVal = ECIO_Transmit(bDataOut(0), 3, nTxDataCnt, bDataIn(0), 3, nRxDataCnt, 500)

End Sub
```



Visual Basic output example

Controlling PWM using Visual Basic

Visual Basic communicates to the MIAC device using pre-made data transfer routine named “ECIO_Transmit”.

Visual Basic Enable PWM Channel

```
'Enable PWM Channels
bDataOut(0) = 13           'PWM Control Command
bDataOut(1) = 1            'Output To channel A (PWM 1)
bDataOut(2) = 1            'Enable Channel
lRetVal = ECIO_Transmit(bDataOut(0), 3, nTxDataCnt, bDataIn(0), 3, nRxDataCnt, 500)
```

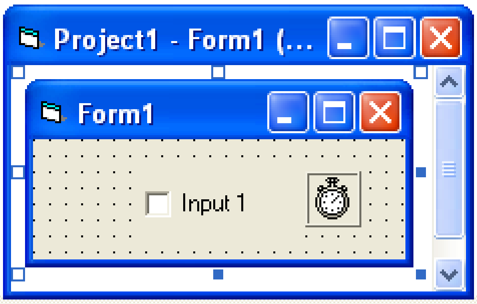
Visual Basic Specify Duty Cycle

```
bDataOut(0) = 14           'PWM Duty Control Command
bDataOut(1) = 1            'Output To channel A (PWM 1)
bDataOut(2) = PWMduty1     'Value of Duty cycle
lRetVal = ECIO_Transmit(bDataOut(0), 3, nTxDataCnt, bDataIn(0), 3, nRxDataCnt, 500)
```

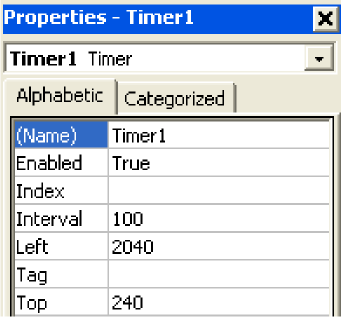
Reading an input using Visual Basic

Here is an example of how to use a Boolean check box in Visual Basic to display the digital representation of a input channel on the MIAC

The example sends command 1 to perform the digital reading.



```
Private Sub Timer1_Tick(ByVal eventSender As System.Object, ByVal eventArgs As System

    Dim lRetVal As Integer
    Dim bDataOut(2) As Byte
    Dim bDataIn(2) As Byte
    Dim nTxDataCnt As Integer
    Dim nRxDataCnt As Integer

    bDataOut(0) = 1 'Digital Input Command
    bDataOut(1) = 1 'Input from channel 1 (Press Button)
    lRetVal = ECIO_Transmit(bDataOut(0), 2, nTxDataCnt, bDataIn(0), 3, nRxDataCnt, 500)

    If (bDataIn(0) = 1) Then
        Input1.CheckState = System.Windows.Forms.CheckState.Checked
    Else
        Input1.CheckState = System.Windows.Forms.CheckState.Unchecked
    End If
```

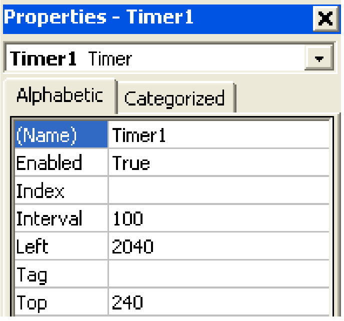


(Name)	Timer1
Enabled	True
Index	
Interval	100
Left	2040
Tag	
Top	240

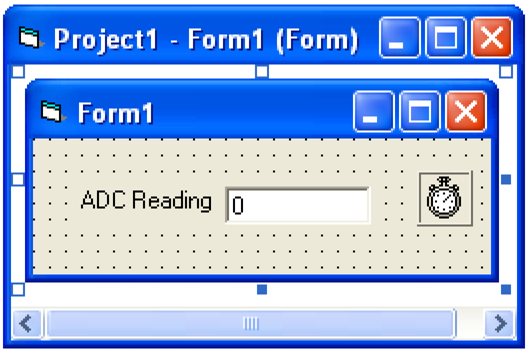
Reading an analogue input using Visual Basic

Here is an example of how to use a text field in Visual Basic to display the digital representation of an analogue input channel on the MIAC.

The example sends command 3 to perform the analogue reading.



(Name)	Timer1
Enabled	True
Index	
Interval	100
Left	2040
Tag	
Top	240



```
Private Sub Timer1_Timer()

    Dim lRetVal As Long
    Dim bDataOut(0 To 1) As Byte
    Dim bDataIn(0 To 1) As Byte
    Dim nTxDataCnt As Long
    Dim nRxDataCnt As Long

    bDataOut(0) = 3             'Perform Analogue Channel Sample
    bDataOut(1) = 1             'Channel 1
    lRetVal = ECIO_Transmit(bDataOut(0), 2, nTxDataCnt, bDataIn(0), 2, nRxDataCnt, 500)
    ADC.Text = bDataIn(0)

End Sub
```

Command List 1-2

Industrial sensor, actuator and control applications

The MIAC industrial controller has a predefined number of built in commands when using it as a slave to the PC. This allows for easier integration with PC side software such as Labview or Visual Basic. Throughout this manual the specific commands used for a particular worksheet have been highlighted. Following is a complete list of the available commands that drive the MIAC controller.

Command Number	Description	Parameters	Returns
1	Digital Input	Channel (1-8)	Input State (0-1)
2	Digital Multi Input	-	Input I1 (0-1) Input I2 (0-1) Input I3 (0-1) Input I4 (0-1) Input I5 (0-1) Input I6 (0-1) Input I7 (0-1) Input I8 (0-1)
3	Analog Input	Channel(1-8)	Input(0-255)
4	Read Keypad Buffer	-	Keypad Press (0-8 or 255)
5	Clear Keypad Buffer	-	0
6	Relay Control	Channel (1-4) State (0-1)	0
7	Transistor Control	Channel (1-4) State (0-1)	0
8	Multi Output Control	Output Value (0-255) Output Mask (0-255)	0
9	Display Clear	-	0
10	Display Cursor Position	X Coord (0-15) Y Coord (0-3)	0
11	Display Number	Number Low (0-255) Number High (0-255)	0
12	Display String	String Data	0
13	PWM Enable Control	Channel (1-2) Enabled (0-1)	0

Command List 2-2

Command Number	Description	Parameters	Returns
14	PWM Duty Control	Channel (1-2) Duty (0-255)	0
15	PWM Period Control	PWM Prescaler (0-2)	0
16	PWM Overcurrent Check	-	Overcurrent State (0-1)
17	EEPROM Write	Address (0-255) Data (0-255)	0
18	EEPROM Read	Address (0-255)	Data (0-255)
19	CAN Set Data	Number of Bytes (0-8) Data0 (0-255) Data1 (0-255) Data2 (0-255) Data3 (0-255) Data4 (0-255) Data5 (0-255) Data6 (0-255) Data7 (0-255)	0
20	CAN Send Message	CAN ID Low (0-255) CAN ID High (0-7)	0
21	CAN Check Incoming	-	CAN ID Low (0-255) * CAN ID High (0-7)
22	CAN Get Data	-	Number of Bytes (0-8) Data0 (0-255) Data1 (0-255) Data2 (0-255) Data3 (0-255) Data4 (0-255) Data5 (0-255) Data6 (0-255) Data7 (0-255)
23	Register Read	Address Low (0-255) Address High (0-255)	Register Data (0-255)
23	Register Write	Address Low (0-255) Address High (0-255) Register Data (0-255)	0

* If both the CAN ID Low and CAN ID High are 0 then no CAN message has been received.

Instructor Guide

Industrial sensor, actuator and control applications

About this course

Introduction

The course combines Locktronics equipment with a MIAC industrial controller unit to teach the basics behind industrial control. Locktronics equipment makes it both quick and simple to construct and investigate the electrical circuits used by industrial controller units. The end result can look exactly like the circuit diagram, thanks to the symbols printed on each component carrier. With the locktronics circuit assembled students are free to use their programming knowledge to create programs to drive various aspects of industrial machinery and processes. The programming languages covered by the course are Visual Basic, Labview and Flowcode but other languages could be used such as Delphi.

Aim

The course aims to introduce students to programming industrial controller units to allow them to interact with the types of sensing and control circuits used in industrial machinery.

Prior Knowledge

It is recommended that pupils have prior knowledge in the programming languages chosen to complete the course. It is also recommended that the course is completed in a single language and then maybe repeated with an alternative programming language to reinforce learning.

Learning Objectives

On successful completion of this course the pupil will have learned:

- to distinguish between analogue and digital sensors;
- that simple digital sensors have a two-state output - either open (off) or closed (on);
- that digital sensors have a high resistance when open, and a small resistance when closed;
- that simple digital sensors output a signal either at 0V or at the full power supply voltage;
- the circuit symbols for a range of switches, bulbs and sensors;
- that some components are polarised, so that they work properly only when connected the right way round;
- that a controller can be programmed to recognise a high input voltage as the switch being either ‘on’ or ‘off’;
- that output devices require a variety of current levels to make them operate;
- that relays can be used to deliver currents up to around 40A;
- that transistors are much faster then relays in switching on and off;
- how to connect a control unit to deliver current through its transistor output terminals;
- how to connect a control unit to deliver current through its relay output terminals;
- that systems typically consist of three basic elements, input, process and output subsystems;
- that analogue sensors output a continuous range of voltages;
- that a potentiometer can set a reference voltage to determine quantities such as temperature;
- that there are two commonly used types of control system, open-loop and closed-loop;
- that an analogue voltage can be produced by varying the duty cycle of a digital square wave signal;
- the advantages of a stepper motor over a simple DC motor;
- that analogue sensors may require a complex calculation to yield a useful value;
- that finite state machines are very useful in the design of a control program;
- that control units can be programmed or controlled using a variety of different languages;

What the student will need:

To complete this course the pupil will need the following equipment:

Additional documents

The instructor will need to refer to the following before starting this course:

Locktronics User Guide - explains how the Locktronics system works, and how to use the parts.

MIAC Getting Started Guide - explains how the MIAC unit works, how to download Flowcode programs or the Slave USB driver to it etc.

Power source:

The mains-powered ’plug-top’ power supply can be adjusted to output voltages of either 3V, 4.5V, 6V, 7.5V, 9V or 12V, with currents typically up to 0.5A. The voltage is changed by turning the selector dial just above the earth pin until the arrow points to the required voltage. The course uses the 12V setting exclusively. However any power supply can be used. If you prefer your students to work from exactly 12V then please use a bench top power supply.

Qty	Code	Description
1	HP2045	Shallow plastic tray
2	HP4039	Lid for plastic trays
1	HP6222	International power supply with adaptors
1	HP5540	Deep tray
2	HP7750	Locktronics daughter tray foam insert
1	HP9564	62mm daughter tray
1	HPUAB	USB AM to BM mini lead
1	LK0123	Magnet
1	LK2363	MES bulb, 12V, 0.2A
1	LK3246	Buzzer (12V)
1	LK4000	Locktronics User Guide
1	LK4025	Resistor - 10 ohm, 1W 5% (DIN)
1	LK4034	Potentiometer, 1K (DIN)
1	LK4322	Stepper Motor
1	LK5100	Locktronics current probe
2	LK5202	Resistor - 1K, 1/4W, 5% (DIN)
1	LK5203	Resistor - 10K, 1/4W, 5% (DIN)
1	LK5214	Potentiometer, 10K (DIN)
1	LK5240	Transistor - NPN, right hand feed
1	LK5243	Diode (IN4001) power 50V
14	LK5250	Connecting Link
1	LK5280	Relay 12V 10A (tranparent case)
1	LK5291	Lampholder carrier
1	LK5402	4.7K thermistor, NTC (DIN)
1	LK5404	Switch, reed
1	LK5603	Lead - red - 500mm, 4mm to 4mm stackable
1	LK5604	Lead - black - 500mm, 4mm to 4mm stackable
6	LK5607	Lead - yellow - 500mm, 4mm to 4mm stackable
6	LK5609	Lead - blue - 500mm, 4mm to 4mm stackable
1	LK5783-56	Industrial sensor, actuator and control applications inlay (DIN)
4	LK6207	Switch Press (morse key-type strip, push to make)
4	LK6209	Switch on/off (stay put, sideways swivel strip)
2	LK6430	LED, red, 12V (DIN)
1	LK6431	LED, yellow, 12V (DIN)
1	LK6432	LED, green, 12V (DIN)
1	LK6634	Microswitch
1	LK6706	Motor 3/6V D.C. 0.7A
1	LK7290	Phototransistor
1	LK6838	Solenoid
1	LK6841	MES bulb, 12V, LED, white
1	LK8275	Power supply carrier with battery symbol
1	LK8900	7 x 5 baseboard with 4mm pillars
1	MI0245	Cased MIAC with 4mm sockets

Instructor Guide

Industrial sensor, actuator and control applications

Using this course:

Our goal is to help students understand sensors and control systems in the context of industrial systems - to understand the components, the circuit diagrams, and the role of the Programmable Logic Controller (PLC). We do this by asking students, working individually or in pairs, to build a number of circuits typically found in an industrial system that uses a PLC. Students generate or select an existing PLC program that makes the circuit work in the desired way, and then take measurements, make drawings, or describe what is happening in the circuits, to reinforce learning.

Central to these activities is the MIAC. This flexible educational and industrial controller can mimic the functionality of most PLCs commonly found in industrial systems.

Programming using Flowcode:

The instructor should make sure that all students are shown how to create and send programs to the MIAC using Flowcode.

Programming using Labview or Visual Basic:

The instructor should make sure that all MIACs are pre-programmed with the appropriate MIAC hex file (“MIAC USB Slave.hex”). Details on how to load this file are found in the MIAC ‘Getting Started’ guide.

Worksheets:

It is expected that the worksheets are printed / photocopied, preferably in colour, for the students’ use. Students will need their own permanent copy, as a record of what they have learned.

Each worksheet has:

- an introduction to the topic under investigation;
- step-by-step instructions for the investigation that follows;
- a summary of the important points in a grey-filled text box

This format encourages self-study, with students working at a rate that suits their ability. It is for the instructor to monitor that students’ understanding is grasp of the ideas involved in the exercises it contains.

Time:

It will take students between six and seven hours to complete the worksheets. It is expected that a similar length of time will be needed to support the learning that takes place as a result.

Worksheet	Notes for the Teacher	Timing
1	**Basic Outputs:** The first worksheet is designed to guide students into using the MIAC PLC with Locktronics. To do this they will be asked to create a small program to switch the transistor outputs on and off one at a time. This program can be completed using Flowcode, Labview or Visual Basic. Note that if you are using Labview or Visual Basic then you will need to ensure that the MIAC PLC has been programmed with the "MIAC USB Slave" program which can be downloaded from the course website.	**40 - 60 minutes**
2	**Sequenced Outputs:** This worksheet takes the output control one step further by explaining the fundamentals of a finite state machine. This is then reinforced with the practical exercise of writing a simple traffic light controller program. The finite state machine used for this program has four unique states which represent the different combinations of lights that can be active on the signal. The movement between the preset states is also another key aspect.	**40 - 60 minutes**
3	**Pulse Width Modulation:** Worksheet three starts to look at slightly more advanced outputs by using the pulse width modulation peripheral embedded into the MIAC controller. By writing values to the PWM peripheral it is easy to show how to create analogue voltages using a purely digital based output. As a reinforcement to this you could place a capacitor between the PWM output and ground and use a voltmeter to measure the analogue output as you change the PWM mark/space parameter.	**40 - 60 minutes**
4	**Basic Inputs:** This example tackles the problem of polling digital switches. The students will learn that they will have to write a loop into their program so the switch values can be checked continuously. This example also gets the student up to speed with storing values into variables to represent the digital input voltage.	**40 - 60 minutes**

Instructor Guide

Scheme of Work

Industrial sensor, actuator and control applications

Worksheet	Notes for the Teacher	Timing
5	**Pelican Crossing:** This worksheet combines outputs, state machines and inputs to create a fully operational pelican crossing system. This reinforces the state machine idea by adding a wait for switch to the previous traffic light system.	**40 - 60 minutes**
6	**Conveyer Belt:** This worksheet introduces students to stepper motors to teach the main differences between analogue DC Motors and Digital Stepper motors. The program is created in two stages, the first being a full stepping approach and the second using half stepping techniques to effectively double the resolution of the motor.	**40 - 60 minutes**
7	**Analogue Sensing:** This worksheet introduces students to analogue sensors and also to the data manipulation often required to obtain a meaningful output. This example is also good for showing the differences between running an application on a high powered computer and running on a Microcontroller chip. Whereas the computer can process the calculation in real time we have to do a trick on the Microcontroller referred to as a look up table.	**40 - 60 minutes**
8	**Detecting Faults 1:** This worksheet introduces automatic fault sensing to the industrial controller. By detecting if an output has switched on as expected the students programs can determine if an output such as a bulb has become damaged or missing. This allows students to see for themselves how to design and create a fault tolerant system.	**40 - 60 minutes**
9	**Detecting Faults 2:** This worksheet moves on from the first fault finding worksheet to include analogue sensors and how to make these tolerant to faults such as electrical breaks or shorts. The users programs can then detect these faults and inform the user of the problem.	**40 - 60 minutes**

Instructor Guide

Scheme of Work

Industrial sensor, actuator and control applications

Worksheet	Notes for the Teacher	Timing
10	Open Loop Control: This worksheet details the basics of an open loop control system. An analogue voltage created from a potentiometer is used as the input to a DC motor speed controller. Students should note here that the motor is simply being told what to do and has no control over any aspect of the system.	**40 - 60 minutes**
11	Closed Loop Control: This worksheet details the basic of an closed loop control system. A bulb is driven directly from an input analogue voltage provided by a potentiometer. The light level from the bulb is then fed back into the system to create a error or difference between what is expected and what is sensed. This error is then used to adjust the control signal to allow the bulb to behave as expected. The changing output can be seen by removing and replacing the paper roof that sits over the bulb and sensor circuitry. When the paper is present the light from the bulb will be fairly dim as the sensor is detecting the correct amount of light. When the paper is removed the bulb will become brighter to try and raise the amount of light sensed by the sensor.	**40 - 60 minutes**