MIAC AllCode
Getting Started Guide

MI3932 MIAC AllCode
MI5331 MIAC AllCode with Wi-Fi
MI5528 MIAC AllCode with Bluetooth

Contents

MIAC AllCode Instructional Guide

Introduction
Getting Started

MIAC AllCode Instructional Guide

Where do you start?
Congratulations. You now have a state-of-the-art MIAC AllCode industrial controller.
This instructional guide has been prepared to help you get up the learning curve as quickly as possible so you can enjoy making the MIAC do the things you want it to do.

Powering up the MIAC
To Power up the MIAC simply connect a 12V to 24V DC supply to the 2.1mm DC jack or to the 0V / V+ screw terminals. The green LED at the top left hand side of the MIAC will light when the unit is powered.

API UART
The MIAC AllCode has two serial UART peripherals which can be used to connect to the AllCode API. By default UART 1 is selected which allows access to the internal Bluetooth / WIFI module (if fitted). The API UART can be controlled using the configuration menu when powering up the MIAC. The MIAC datasheet shows the UART connections and the internal configuration jumpers to select between RS232 and RS485 communications modes.

Pairing your Bluetooth enabled device to the MIAC
The Bluetooth enabled MIAC AllCode must be paired to your Bluetooth enabled device to allow API communications to work correctly. Pages 8 through 13 show how to pair the MIAC AllCode to various different devices.

Connecting the MIAC AllCode to your WIFI network
The WIFI enabled MIAC AllCode can be connected to your WIFI network by using the configuration menu when powering up the MIAC. The Network Config setting must be set to Join Network. The Network SSID setting can then be used to scan for WIFI networks in range. The Password setting can be used to enter the password for the network.

Hosting a WIFI network from the MIAC
The WIFI enabled MIAC AllCode can be used to host a WIFI network by using the configuration menu when powering up the MIAC. The Network Config setting must be set to Host Network. The Network SSID setting can then be used to enter the WIFI network name. The Password setting can be used to enter the password for the network. The Encryption and Channel settings can be used to alter the settings of the network.

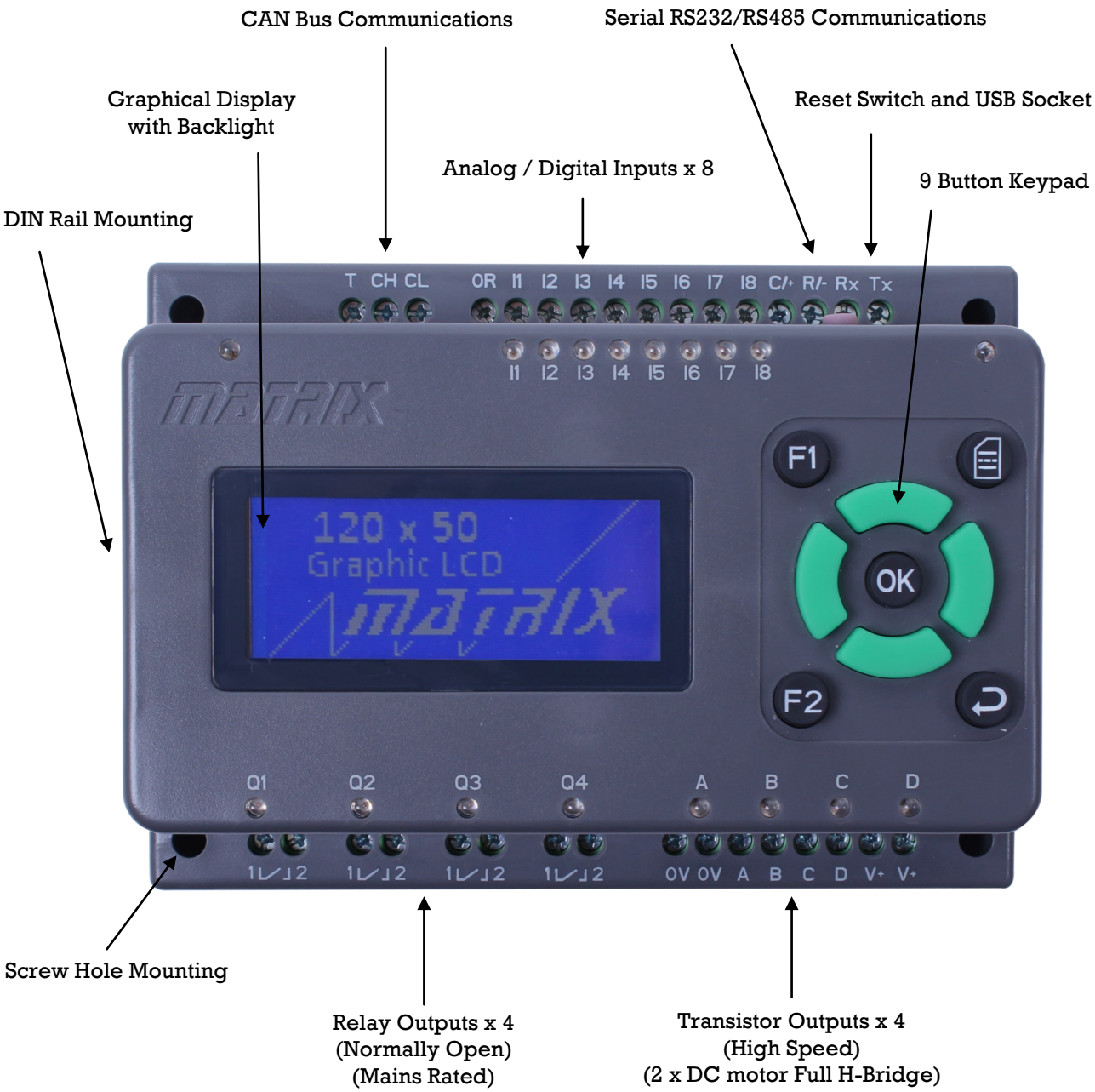
The Pre-Programmed Firmware
The original API firmware can be found on the AllCode section of the Matrix Website and can be re-programmed using the mLoader programming tool.

Take the Tour

Features of the MIAC AllCode

MIAC AllCode Instructional Guide

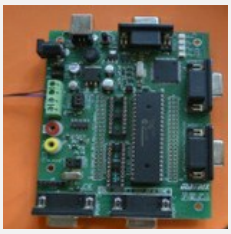
This section explores the various parts of the MIAC AllCode and explains the basic operation.



Language Neutral

Application Program Interface

MIAC AllCode Instructional Guide



Language independent, agnostic, language neutral, platform independent - what do these terms mean?

Basically it means you are not forced to use a certain programming language or platform to control the MIAC.

This is because the MIAC AllCode offers an Application Program Interface (API) that enables you to interact with the MIAC using a set of simple routines or protocols.

If you have not heard of APIs then this section will help you understand how to make use of it with the MIAC.

One way to explain this is to think about how a TV remote controller works. Although modern televisions have touch buttons or soft-touch areas along the edges of the screen, most people find it more convenient to use a remote controller to turn the TV On/Off, change a channel, adjust the volume or brightness settings, etc.

A TV remote is a very simple device consisting of a keypad and an infrared light beam. When a button is pressed its value is encoded and used to send a binary pattern, via the infrared beam, to the TV. The TV decodes the received pattern and carries out the required function.

If for some reason the TV remote failed it would be an easy task to replace it with a new one, or even purchase a universal remote (if you had a number of devices to control).

There are Apps that can be used to turn smart phones into a TV remote controller and other hardware is available that can send out TV remote codes. The only critical part is to send the correct pattern (i.e. command) when it is required.

So you could say the TV has an API that allows a remote controller (whatever form it might take) to control the intelligence or electronic control systems within the television.

The API, as used on the MIAC AllCode, offers the same platform and language independence as the TV example described above. The difference is the transmission medium for the MIAC is Bluetooth or Wifi rather than an infrared beam. This means providing you have a Bluetooth or Wifi facility on your system, you have the freedom to use your favourite platform and programming language to interact with the MIAC.

As an example, you might choose to use a Bluetooth enabled mobile phone to control the MIAC. Alternatively, a Wifi enabled PC/Mac/Raspberry Pi® or a Matrix E-block upstream board would do the same to create a simple controller.

Language Neutral

Application Program Interface

As the API reacts to a simple text-based protocol (as shown below) it means you have the freedom to use formal programming languages like C, C#, C++ or Python, or graphical or icon-based languages like Flowcode, App Inventor or LabView.

The other thing to note about the API is that some commands are bi-directional. This means that a command sent to the MIAC could result in a value being returned. A good example of this is the command to read the input terminals of the MIAC. An API command could be sent to sample the input (that effectively interrogates it) causing it to return a numerical value of the voltage present on the input terminal.

Shown below is the general format for the MIAC's API.

Text string Numeric values and/or text string

Every command starts off with a text string that identifies what the MIAC should do. This may be followed by one or more parameters. Depending on what you are trying to do with the MIAC these parameters can be numerical or textual or a mixture of both.

For example to send a value to the four relays on the MIAC you would use:

```
WriteRelays <value>
```

As the relays are grouped together and form an 4-bit row, they can be driven by sending a binary number to them. So the parameter <value> can take a value between 0 and 15.

It should be noted that the API commands for a particular language might have some subtle differences. For example, Python will use something like `"ma.WriteRelays(2)"` whereas C# would look like `"MA_DLL.MA_WriteRelays(2);"` and for App Inventor, Flowcode and LabView the appropriate icon would be selected.

Here's another example that shows how to control a transistor on the MIAC.

```
WriteTran <channel> <value>
```

The parameter labelled <channel> can take a value between 1 and 4 to define which transistor output to control. The parameter labelled <value> can take a value between 0 to 1 to define the state of the output.

If you wanted to control all transistor outputs at once you could use this command.

```
WriteTrans <value>
```

The parameters <value> can take a value from 0 to 15 to define the binary state of the transistor outputs.

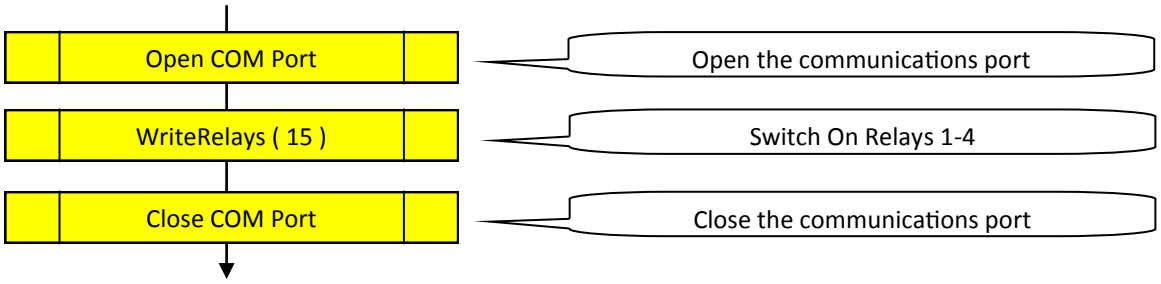
Pseudo-code

A Simple Programming Aid

MIAC AllCode Instructional Guide

Most of the examples that appear in this Instructional Guide are written with pseudo-code. This pseudo-code can be used with the many programming languages that can be used with the MIAC AllCode.

One of the reasons for this is that this Guide would be huge if all the examples were written in every language the MIAC AllCode supported, and only certain parts would be applicable to you.



API calls allow you to interact with the MIAC AllCode. The name of these calls will map across to the set of macros available in Flowcode, AppInventor, Python and other languages supported by the MIAC AllCode. A full list of available API calls is available at the end of this document.

Using pseudo-code allows you to take the first step of putting your ideas into practice in a structured way without getting tripped up by the syntax of a formal programming language. Once you have expressed your ideas, using pseudo-code, you can move on and write the actual program using your chosen programming language, which should hopefully be a fairly simple coding task.

Bluetooth setup

Windows PC

MIAC AllCode Instructional Guide

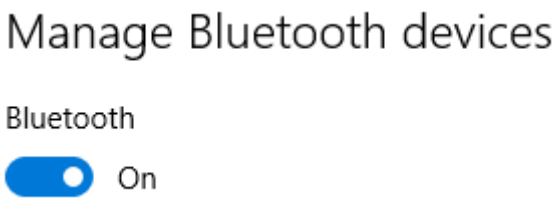
A lot of Windows devices, especially laptops and tablets, have inbuilt Bluetooth functionality. If your PC does not, you will need to use a Bluetooth 2 USB dongle.

Different versions of the Windows operating system use slightly different ways of connecting Bluetooth devices, but they all follow the same steps.

You will need to perform this process just once as Windows will remember which devices are paired.

1) Turn on Bluetooth

Often, Bluetooth is enabled by default and you can usually ignore this step. However if it is not, it can be enabled in the Windows settings and/or control panel. Very occasionally, Bluetooth needs to be switched on using a special switch or function-key. Please consult your PC or Windows help for more information.



2) Pair the MIAC

First switch on the MIAC - the Bluetooth device name, pair key and visibility can be checked and edited using the Configure menu and then selecting Bluetooth Settings when powering up the MIAC.

Again, pairing works slightly differently on the various Windows versions and so it is difficult to give specific instructions here. The Windows help and website will have guides explain how.

When pairing, you will be presented with a screen or list of available Bluetooth devices. Select the device with the name of your MIAC and click Next or Pair.

You will be asked to enter the pairing code. The MIAC AllCode uses the default code of 1234, although this can be changed to another code if you want to ensure no-one else can pair with your MIAC.

Once the code has been entered, Windows will confirm that it has paired with the MIAC.

Bluetooth setup
Windows PC

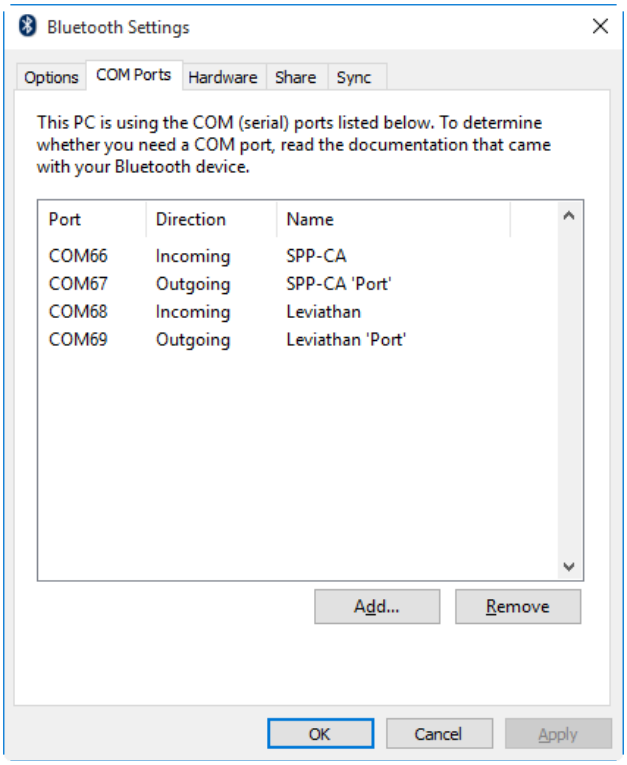
MIAC AllCode Instructional Guide

3) Determine the COM port number

You should get a popup balloon on the task bar saying device is ready to use. If you click this before it fades away then you can find out the COM port assigned to the MIAC.

You use this COM port number when communicating with the MIAC AllCode and this COM port number will stay the same as long as you do not remove or unpair the MIAC from Windows.

If you did not see the COM port when the MIAC was paired, you can find it in the Bluetooth Settings window, as shown on the right.

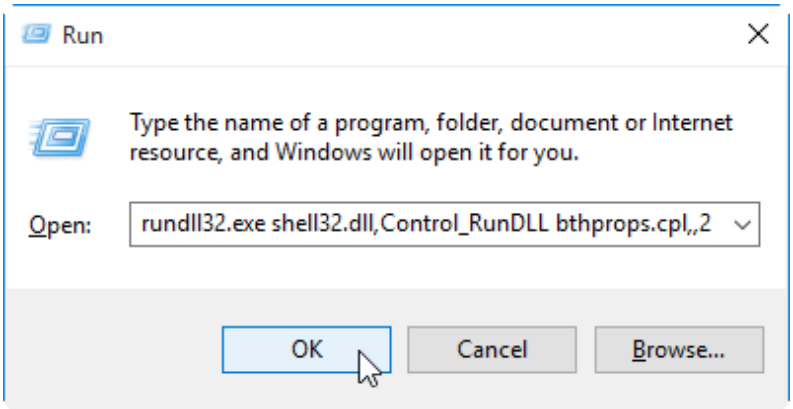


There are two COM ports listed for each MIAC. Make sure you always use the “outgoing” port number.

This window can be a bit hard to find on some versions of Windows. For example, on Windows 10 you can find this via the “More Bluetooth options” link on the Bluetooth settings screen.

Luckily, there is a guaranteed way of opening this window in all versions of Windows from version 7. Open the “Run…” window by holding the Windows key and pressing R, then type (or copy and paste) the following command into the box and press “OK”:

```
rundll32.exe shell32.dll,Control_RunDLL bthprops.cpl,,2
```



Now you have paired the MIAC and determined the COM port number, you can use any of the many programming languages available on Windows to control the MIAC AllCode.

Bluetooth setup

Android Tablet / Phone

MIAC AllCode Instructional Guide

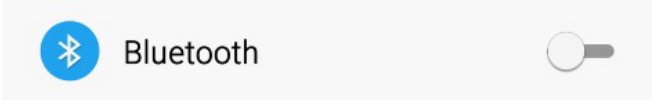
Android phones and tablets, when used with intuitive programming software such as App Inventor, provide a motivating platform for controlling the MIAC AllCode.

These devices almost always include Bluetooth and Wifi built-in.

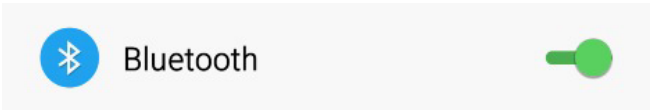
As with other devices, the MIAC AllCode must be paired with the phone or tablet before it can be used.

If your Phone or Tablet already has Bluetooth functionality built in then you may first have to enable it by clicking on Settings -> Connections.

Bluetooth Switched Off



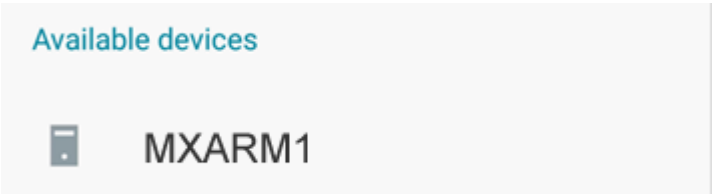
Bluetooth Switched On



Once Bluetooth is enabled you need to pair the MIAC AllCode to your phone to allow Apps to see the device.

Begin by clicking the Bluetooth option in Settings -> Connections

Next make sure your MIAC is switched on and click the Scan button on your Android device to check for new Bluetooth devices. Note you may have to scroll down to see the results from the scan.



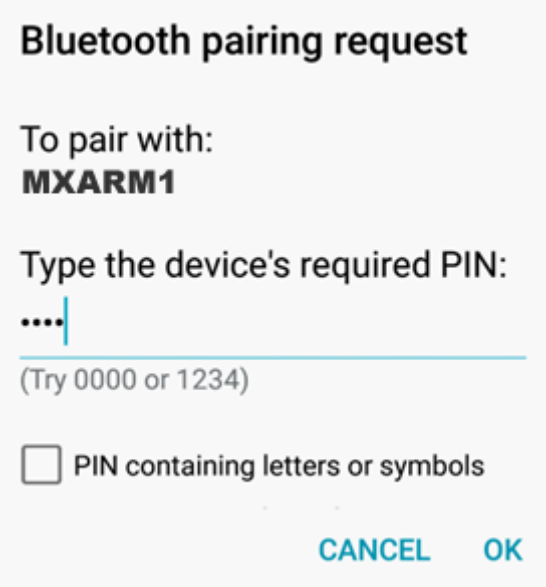
The name of the MIAC AllCode can be found by using the MIAC keypad and navigating to the Configure menu followed by the Bluetooth Settings menu when powering up the MIAC.

Bluetooth setup

Android Tablet / Phone

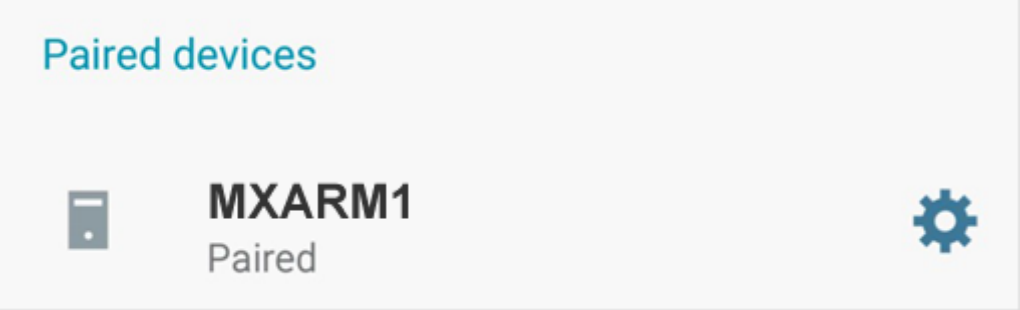
MIAC AllCode Instructional Guide

When the device name has appeared click the device name and you will be asked to enter the pair key.



The default key is 1234.

Once the device is paired it will be listed along with any other paired Bluetooth devices you might have and is ready to be used with any MIAC AllCode apps you download or create.

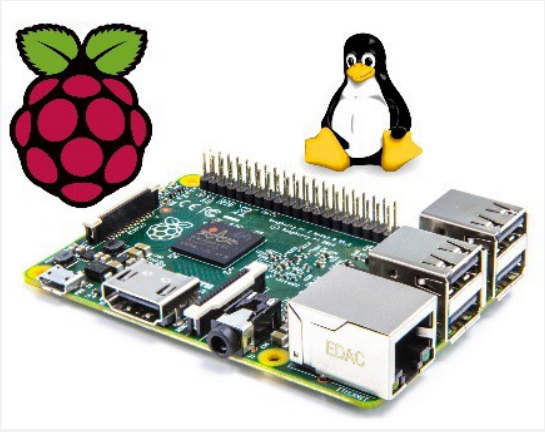


Please note: This may be subtly different on your Android device. For specifics on your Phone or Tablet please look up how to pair Bluetooth devices for your specific device.

Bluetooth setup

Raspberry Pi and Linux

MIAC AllCode Instructional Guide



The Raspberry Pi is a popular single-board computer.

The most common operating system used on the Raspberry Pi is a variety of Linux called Raspbian.

The instructions here for pairing the MIAC AllCode are not limited to a Raspberry Pi and should apply to most Linux-based computers.

Setting up Bluetooth is relatively easy on a Raspberry Pi and can be done in a number of ways. The following steps are perhaps a more complex way of setting it up, but it should work in all situations. Note the Pi needs a Bluetooth USB dongle.

Step 1 – Get your Bluetooth settings

Open a command-line terminal and type the command “hciconfig”. This will bring up a list of Bluetooth devices available on your RPi. The important thing to note is the identifier of the Bluetooth module – in my case it is “hci0”:

```
pi@raspberrypi / $ hciconfig
hci0:   Type: BR/EDR  Bus: USB
        BD Address: 00:15:83:15:A3:10  ACL MTU: 339:8  SCO MTU: 128:2
        UP RUNNING PSCAN ISCAN
        RX bytes:1420 acl:0 sco:0 events:53 errors:0
        TX bytes:452 acl:0 sco:0 commands:46 errors:0
```

Step 2 – Detect the MIAC AllCode

Switch on the MIAC and then type “hcitool scan”. When I did this, it showed two devices. Mine was the latter (“API_B”) and you will need to take note of the 6 pairs of hexadecimal numbers that are the MAC address, a unique identifier to the MIAC – in my case,

```
pi@raspberrypi / $ hcitool scan
Scanning ...
        00:BA:55:23:1C:16       FormAllCode
        00:BA:55:23:1C:20       API_B
```

“00:BA:55:23:1C:20”.

```
pi@raspberrypi / $ sudo bluez-simple-agent hci0 00:BA:55:23:1C:20
RequestPinCode (/org/bluez/2342/hci0/dev_00_BA_55_23_1C_20)
Enter PIN Code: 1234
Release
New device (/org/bluez/2342/hci0/dev_00_BA_55_23_1C_20)
```

Step 3 – Pair the MIAC with the RPi

To pair, you can use the “bluez-simple-agent” command using the “hci0” and MAC address found in the previous steps.

Bluetooth setup

Raspberry Pi and Linux

MIAC AllCode Instructional Guide

Step 4 – Making the change permanent

The final step is to make this pairing happen automatically when the RPi is next used. This can be done by editing the "/etc/bluetooth/rfcomm.conf" file (e.g. using nano) and entering the following code. Again, you will need to ensure you use the correct MAC address that was found earlier.

```
pi@raspberrypi / $ sudo nano /etc/bluetooth/rfcomm.conf
```

You will need to add a section to this rfcomm.conf file similar to the following:

```
rfcomm1 {
# Automatically bind the device at startup
bind yes;

# Bluetooth address of the device
device 00:BA:55:23:1C:20;

# RFCOMM channel for the connection
channel 1;

# Description of the connection
comment "MIAC AllCode";
}
```

The three red bits of text can be customised - you will use the MAC address found in step 2, and can use the name in the "comment" field.

If you have more than one MIAC, you can add multiple sections - just name each one "rfcomm1", "rfcomm2", etc.

Step 5 – Testing the connection

Once you are paired, you can test the connection by using the following in the command line terminal:

```
echo "WriteRelay 1 1\n" > /dev/rfcomm1
```

If all goes well, the Relay Q1 should become activated and the LED for Q1 should light up.

If this does not work and you get "permission denied" message, you may need to add yourself to the "dialout" group. To see if this is the case, use the "id" command with your username as a parameter to check which groups you belong to. If the group "dialout" is not listed, you can add yourself to the group using the following command (remember to substitute your username in place of "username"!):

```
sudo usermod -a -G dialout username
```

You will then need to logout and log back in and the "`WriteRelay`" command should now work ok.

WIFI setup

Connecting to the IP Address

MIAC AllCode Instructional Guide

WIFI Connectivity

The MIAC can be used to Host or Join a WIFI network by using the Configure and Wifi Settings menu when powering up the MIAC. Once the network settings have been specified they will be retained so you don't have to keep re-configuring the MIAC when powering up.

Network Host Mode

The host mode allows the MIAC to host a WIFI network. The various settings of the network can be specified using the Configure menu when powering up the MIAC. Once the API mode has been started the MIAC will display its local IP address you should use to connect to the MIAC.

To communicate with the MIAC you will need to connect your controller (PC/Linux/Phone/Tablet/ etc) to the WIFI network and then use the IP address provided by the MIAC.

Network Join Mode

The join mode allows the MIAC to join an existing WIFI network. The network to join and access password can be specified using the Configure menu when powering up the MIAC. Once the API mode has been started the MIAC will display its local IP address you should use to connect to the MIAC.

To communicate with the MIAC you will need to connect your controller (PC/Linux/Phone/Tablet/ etc) to the WIFI network and then use the IP address provided by the MIAC.

Controlling the MIAC
Using Flowcode

MIAC AllCode Instructional Guide

This section explains how to use Flowcode to control the MIAC AllCode.

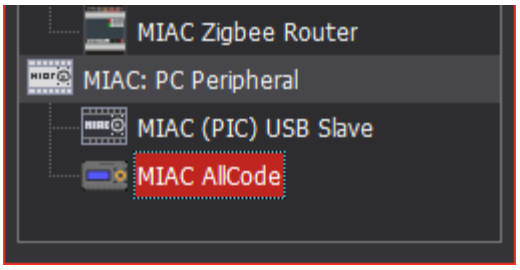
As you probably know, Flowcode provides component macros for all complex devices like CAN bus, ZigBee, and the MIAC AllCode

There are programs on the Matrix website to inspire and help you.

https://www.matrixtsl.com/allcode/resources/

This section assumes you are familiar with the basics of using Flowcode. There are two ways of using Flowcode to control the MIAC AllCode.

1) using the in-built API functionality
2) re-programming the firmware on the MIAC.

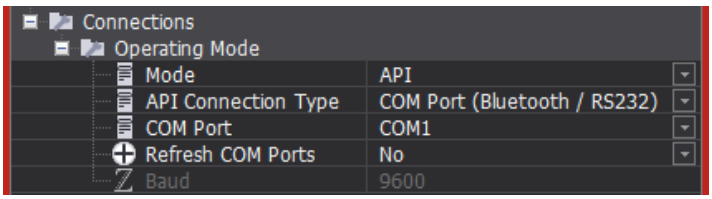


In Flowcode 7.3 and later there is a component available from the MIAC component menu to allow you to easily control the MIAC AllCode via the command API.

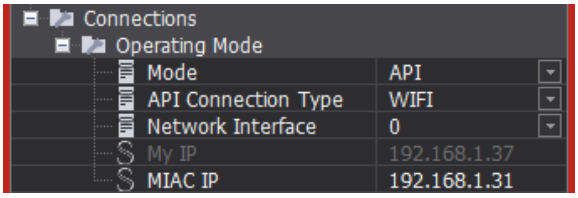
The component comes with a fully operational simulation allowing us to create and test programs before we move to the hardware.

The component's Mode property allows us to decide if we are using the component to control the simulation or the real hardware.

Bluetooth / Serial Component Properties



WIFI Component Properties



The Bluetooth and Serial mode has a COM port property that points to the MIAC AllCode. This COM port number can be found in the Bluetooth pairing section of this document.

The WIFI mode has a Network Interface option, this allows you to specify the network adapter connected to the same network as the MIAC AllCode should you have more than one Network adapter available on your PC. The MIAC IP address should be entered so that it matched the IP address shown on the MIAC display when entering API mode.

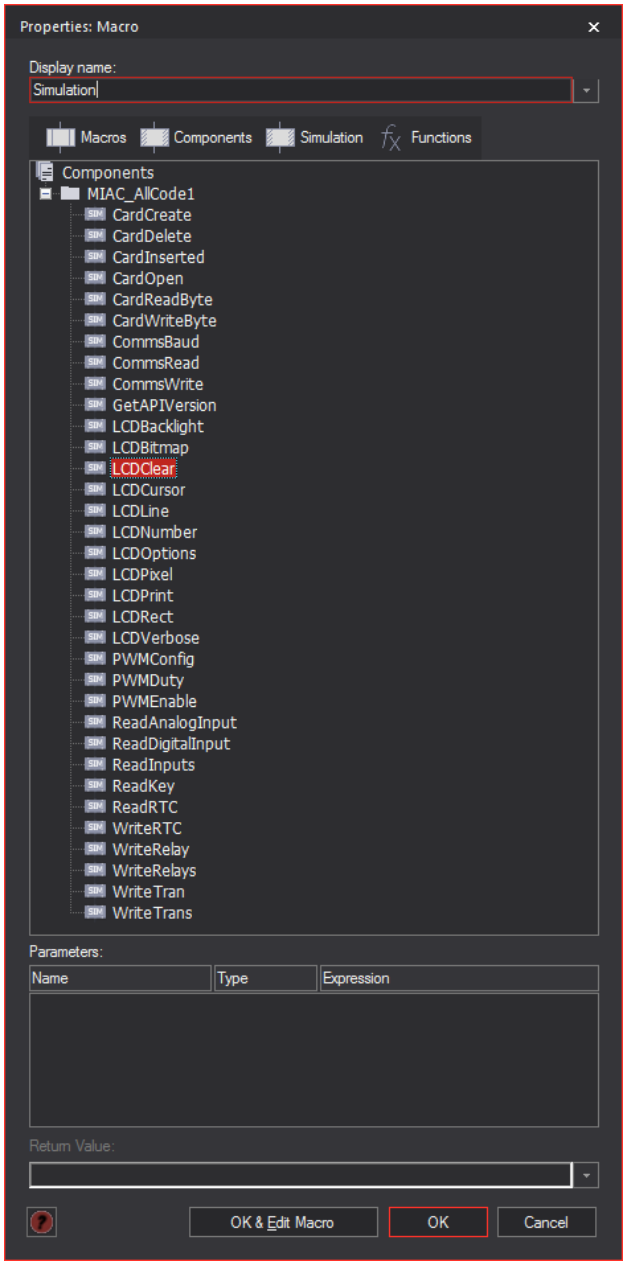
Controlling the MIAC

Using Flowcode

MIAC AllCode Instructional Guide

Direct control from Flowcode

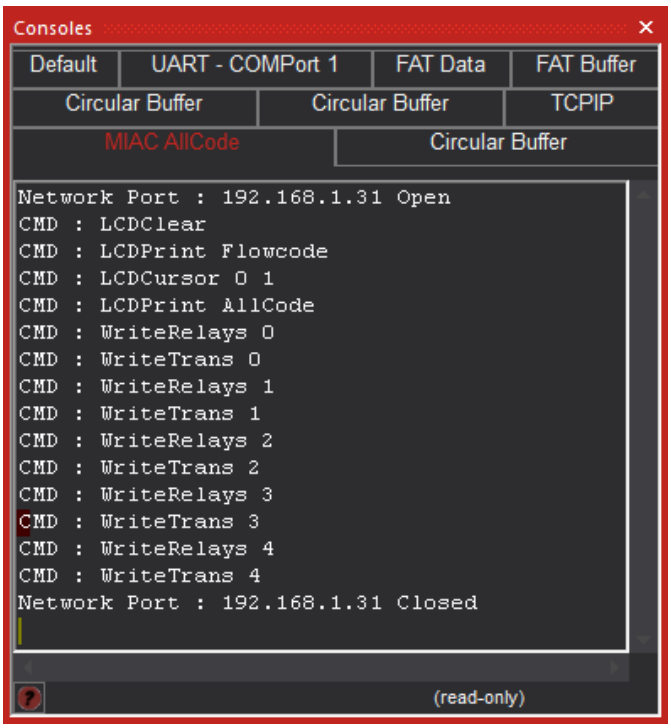
The component ("MIAC AllCode") allows us to control a MIAC AllCode from within Flowcode without compiling or downloading.



Note that the Component Macro functions appear in the Simulation tab rather than the usual components tab. This highlights the fact that the component code is not downloadable onto the MIAC.

The selected communications port is automatically opened when you start the simulation and closed when you end the simulation.

The console window shows the API commands as well as any return values allowing an easy way of seeing the communications between the simulation and the MIAC.



Downloading code to the MIAC

Flowcode also allows us to create code which will run on the microcontroller on-board the MIAC. This is not recommended unless you know what you're doing.

Note: Downloading code to the MIAC will remove the API functionality from the MIAC AllCode. Instructions on restoring the API firmware to get the MIAC back to the original factory functionality are provided on page 3.

Controlling the MIAC

Using App Inventor

MIAC AllCode Instructional Guide

App Inventor

Template

This section explains how to get started with the coding language called App Inventor that will enable you to use an Android device to control the MIAC.

These QR codes and hyperlinks will help speed-up your installation, so you can start having some fun coding.

App Inventor

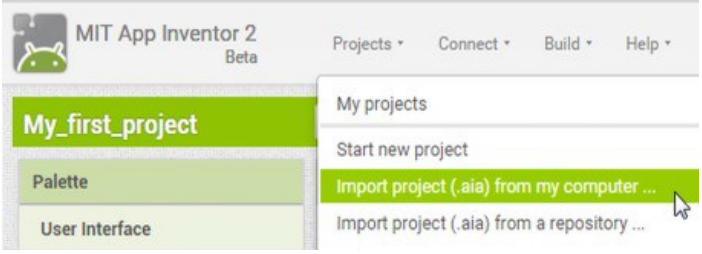
App Inventor is a freely available graphical programming language hosted on one of the cloud-computing and storage systems at Massachusetts Institute of Technology (MIT) in the United States. All you need to get started developing apps for an Android mobile phone or tablet is a web browser and a Google account. App Inventor uses colour-coded icons, shaped like jigsaw-puzzle pieces, to create an app by joining the pieces together. The system prevents you making mistakes by ensuring only certain shapes with the same colour scheme can be joined together. This technique encourages people of all ages to enjoy 'coding' and develop their confidence and ability in computer programming.

Setting up App Inventor

The key items you need are a desktop or laptop (running a modern browser like Chrome or Firefox) and a phone or tablet running the Android operating system. You will also need a QR reader so it would be a good idea at this stage to download one on to your mobile.

Just follow these simple steps to get yourself up and running really quickly.

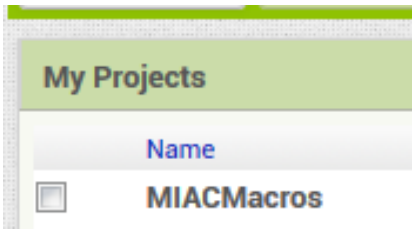
1. Set up a Google account (if you haven't already got one).
2. Go to the App Inventor website by scanning the QR code or clicking the hyperlink above and then login using your Google account.
3. Follow the online instructions, including installing the "MIT AI2 Companion App" onto your Android device.
4. You will need to link the web-based App Inventor with your phone or tablet. To do this, select 'AI Companion' from the 'Connect' menu in App Inventor.
5. Download the MIAC AllCode template onto your computer by scanning the QR code or clicking the hyperlink. Remember where you saved them on your desktop/laptop.

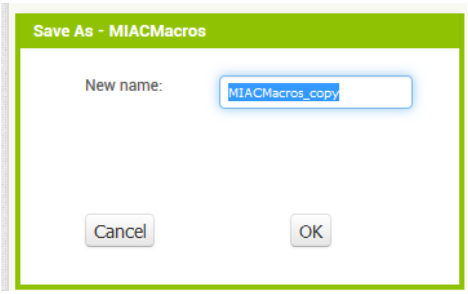


Currently AppInventor only supports the Bluetooth form of communications.

Your first program
Each time you want to start a new project, follow these steps:

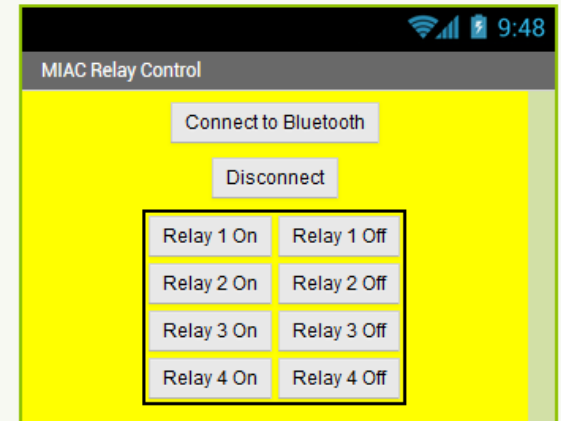
1. Load the template file by clicking 'My Projects' from the App Inventor menu and selecting the 'MIACMacros' project.





2. Save this template as a new file by selecting 'Save project as…' from the 'Projects' menu and then entering an appropriate name for your project.

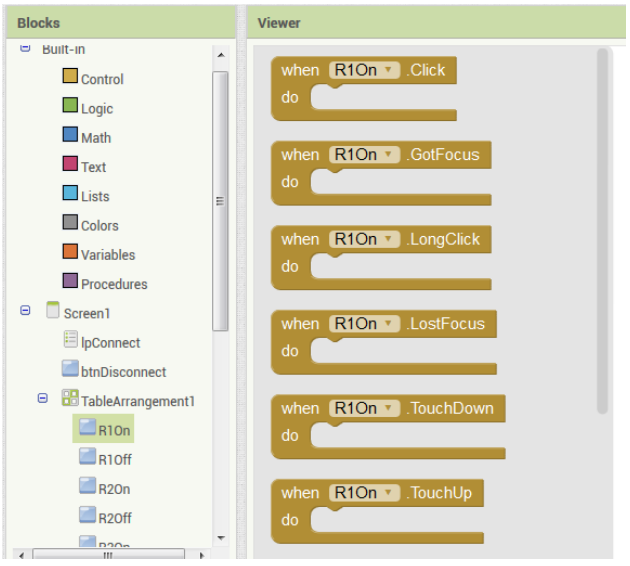
3. Click 'Screen1' from the 'Components' pane and set the 'AppName' and 'Title' in the 'Properties' pane to something suitable.



4. Drag a button from the User Interface panel onto the Viewer screen and alter its text to read "Relay 1 On". Also rename the button so it reads "R1On". Do the same again to create another button called "Relay 1 Off" with the name "R1Off". Repeat for the other relays.

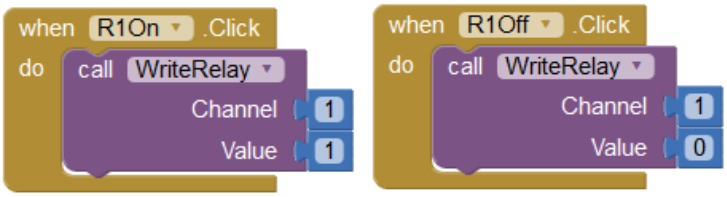
5. Switch to 'Blocks' mode and click on the 'R1On' object - a list of icons will appear.

Drag the "when R1On.Click" icon onto your program. Do the same again but this time click on the 'R1Off' object. Repeat for the other switches.



6. Click 'Procedures' from the 'Built-in' list and drag the "call WriteRelay" icon into the middle of your "when R1On.Click" and "when R1Off.Click" icons. Add literal values from 'Math' as the Channel and Value parameters to the WriteRelay code blocks.

Your two blocks should look something like this:



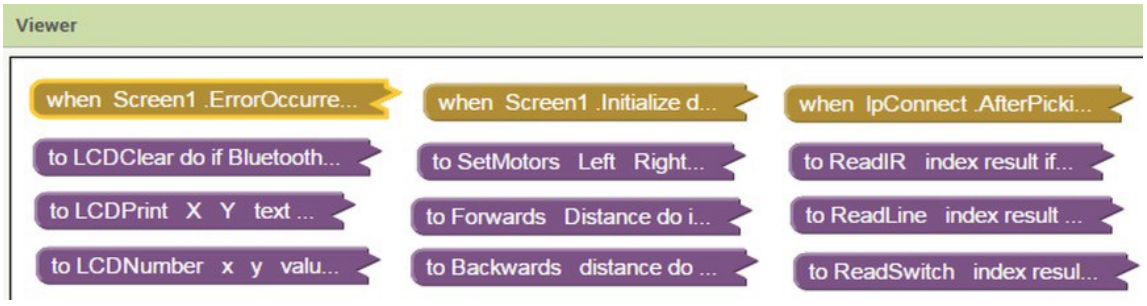
Repeat for the other switches.

7. Now you should build the project. Select "App (provide QR code for .apk)" from the "Build' menu. Once this is complete, run the "MIT AI2 Companion" app and then scan the QR code into your Android device using the "Scan QR Code" button.

Note:
If you visit the App Inventor website you will find instructions about other methods that are available for transferring your program to your Android device.

8. You can now run your program on your Android device. Click "Connect to device" and select the MIAC AllCode from the list. Clicking the "Relay 1 On" button should make Relay Q1 switch on. Clicking the "Relay 1 Off" button should make Relay Q1 switch off.

You may have noticed a number of icons at the top of the screen in App Inventor. These define the procedures for communicating with the MIAC AllCode and some standard functions to allow the Bluetooth link to be set-up.



The tan coloured icons represent **events** such as when a button is clicked, when a timer triggers or when an error occurs.

The mauve coloured icons relate to a set of **procedures** or **subroutines** that have been designed to perform certain tasks for you. You should not alter these unless you are an experienced App Inventor user.

Controlling the MIAC

Using C++ / C# / VB

MIAC AllCode Instructional Guide

A common programming tool is Visual Studio from Microsoft.

In this section we introduce various methods to communicate with the MIAC AllCode using some of the more widely known programming languages such as C++, C# and Visual Basic.

Using the MIAC AllCode with Visual Studio via the Visual C++, Visual C# or Visual Basic programming languages is fairly straightforward and consists of using a DLL library and associated files provided by MatrixTSL to communicate with the MIAC.

As will other languages, you need to use the COM port number or IP address that the MIAC is connected to. There are examples on the AllCode pages of the Matrix TSL website here: https://www.matrixtsl.com/allcode/resources/

Using C#

The program on the right shows a basic program in C#.

You should use the namespace "MIACAllCode" and place the MA_DLL library file in the same folder as your project. The DLL itself needs to be in the same folder as the EXE you create.

You will notice that the AllCode API commands are prefixed with the characters "MA_" and also need to have the COM port / IP Instance sent to them each time as the first parameter.

Remember to modify the API commands in the appendix when using them.

Also remember to close the port at the end of your program!

```
using System;
using System.Runtime.InteropServices;

namespace MIACAllCode
{
    class Program
    {
        static void Main(string[] args)
        {
            byte commsMode = 1;                                 //0 = Bluetooth or Seri
            byte PortNumber;

            if (commsMode == 0)
            {
                PortNumber = 63;                                //Assign Port Number -
                MA_DLL.MA_ComOpen(PortNumber);                  //Open Port
            }
            else
            {
                PortNumber = 0;                                 //Assign Port Number -
                MA_DLL.MA_IPOpen(PortNumber, 192, 168, 1, 31);      //Open MIAC IP
            }

            System.Threading.Thread.Sleep(1000);                //Delay to allow Comms
            MA_DLL.MA_WriteRelay(PortNumber, 1, 1);             //Relay 1 On
            System.Threading.Thread.Sleep(500);                 //Short Delay
            MA_DLL.MA_WriteRelay(PortNumber, 1, 0);             //Relay 1 Off
            System.Threading.Thread.Sleep(500);                 //Short Delay

            if (commsMode == 0)
            {
                MA_DLL.MA_ComClose(PortNumber);                 //Close Port
            }
            else
            {
                MA_DLL.MA_IPClose(PortNumber);                  //Close Network Port
            }
        }
    }
}
```

Using VB
The same program is shown on the right, this time in Visual Basic.

You will see that the program style is very similar to the C# program, with only some minor differences in syntax. The calls to the AllCode API are identical.

The MA_DLL.vb file should be added to your project.

Remember also to put the "MA.DLL" file into the same folder as your created EXE.

```
Module Module1

    Sub Main()

        'Create variable to hold the COM port number
        Dim PortNumber As Byte
        Dim CommsMode As Byte = 1  '0 = Bluetooth or Serial / 1=WIFI Network

        If (CommsMode.Equals(0)) Then
            PortNumber = 3              'Remember to change this to match your
            MA_ComOpen(PortNumber)      'Open Port
        Else
            PortNumber = 0                              'Remember to change thi
            MA_IPOpen(PortNumber, 192, 168, 1, 31)      'Open Port
        End If

        System.Threading.Thread.Sleep(1000)

        MA_WriteRelay(PortNumber, 1, 1)     'Relay 1 On
        System.Threading.Thread.Sleep(500)

        MA_WriteRelay(PortNumber, 1, 0)     'Relay 1 Off
        System.Threading.Thread.Sleep(500)

        If (CommsMode.Equals(0)) Then
            MA_ComClose(PortNumber)         'Close Port
        Else
            MA_IPClose(PortNumber)         'Close Port
        End If

    End Sub

End Module
```

Using C++
The same program is shown on the right, this time in C++.

To use the DLL with C++, you need to reference the functions by including the "MIAC_AllCode.h" header file. You also need to add the "MIAC_AllCode.lib" file to your Visual Studio project.

Also put the DLL into the same folder as the EXE you create.

As with the other languages, the calls to the AllCode API are very similar, meaning it is very easy to use the MIAC with different languages - assuming you know the basics of that language anyway!

```
#include "stdafx.h"
#include "MIAC_AllCode.h"
#include <windows.h>

int main()
{
    char commsMode = 1;                     //Communications Mode - 0 = Bluet
    char PortNumber;                        //Create variable to hold port nu

    if (commsMode == 0)
    {
        PortNumber = 63;                    //Remember to change this to matc
        MA_ComOpen(PortNumber);             //Open Port
    }
    else
    {
        PortNumber = 0;                     //Remember to change this to matc
        MA_IPOpen(PortNumber, 192, 168, 1, 31);     //Open Port - Remember to
    }

    Sleep(1000);

    MA_WriteRelay(PortNumber, 1, 1);    //Relay 1 On
    Sleep(500);

    MA_WriteRelay(PortNumber, 1, 0);    //Relay 1 Off
    Sleep(500);

    if (commsMode == 0)
    {
        MA_ComClose(PortNumber);        //Close COM Port
    }
    else
    {
        MA_IPClose(PortNumber);         //Close Network Port
    }

    return 0;
}
```

Controlling the MIAC

Using Python

MIAC AllCode Instructional Guide

Python is a widely-used computer programming language that is available on many systems. It is free, easy to learn and fun to use.

This section will show you how to set up Python for use with MIAC AllCode. It is assumed you have a basic working knowledge of Python itself. If not, there are many good resources on the internet if you wish to learn this language.

Set-up

The first thing you need is to make sure Python is installed on your computer. It is usually installed by default on a Raspberry Pi, but for Windows and other devices you will probably need to download and install it from http://www.python.org.

There are two versions of Python, 2 and 3, and either can be used to control the MIAC AllCode, but you may wish to ensure you have the latest version installed.

In addition to Python itself, you will also need to install the PySerial library. This can be found on GitHub: https://github.com/pyserial/pyserial or can be downloaded on a Linuxbased device using the following command in a terminal window:

```
sudo apt-get install python-serial
```

Now that Python and the PySerial library are installed, you should download the MIAC AllCode Python library from here: https://www.matrixtsl.com/allcode/resources/

You will find examples and other resources on this page that will help you control the MIAC in Python and many other languages.

My first Python program

```
import MA              # Import the MIAC AllCode library
ma = MA.Create()       # Create an instance of the API
ma.ComOpen(3)          # Open the COM port
ma.WriteRelay(1,1)     # Switch On Relay 1
ma.ComClose()          # Close the COM port
```

This very simple program activates relay Q1 using the API command `WriteRelay`, but there are a number of other lines of code before and after that command that may need more explanation.

The first three lines of code import the library so you can use the API commands, then an instance of the API is created and a communication channel to it is opened. The number "3" represents the COM port that was created when the MIAC was paired.

It is important to close the COM port and at the end of the program we should do that using a call to the `ComClose` API command.

Controlling multiple MIACs

By creating multiple instances of the API, we can actually control more than one at the same time. The program on the right shows how this can be done.

Just like the first program, we start by importing the MA library (note we are also importing the "time" library too). We then create 2 instances of the API and open their COM ports.

The routine for drawing the square should be self-explanatory.

Finally, both COM ports are closed.

Theoretically many MIACs can be controlled simultaneously, but unfortunately there is a practical limit due to the capability of the computer's Bluetooth device. I have found 3 or 4 is the realistic maximum.

```
# Import the libraries
import MA
import time

#Create and open 2 MIACs
ma1 = MA.Create()
ma2 = MA.Create()
ma1.ComOpen(3)
ma2.ComOpen(4)

#control the Relays
ma1.WriteRelay(1,1)
ma2.WriteRelay(1,1)

# Close the COM ports
ma1.ComClose()
ma2.ComClose()
```

Going further

We have shown only a few brief examples of how to control the MIAC AllCode using Python. If you look at the API reference at the end of this document you will find many other commands that can be used.

Controlling the MIAC

Using Labview

MIAC AllCode Instructional Guide

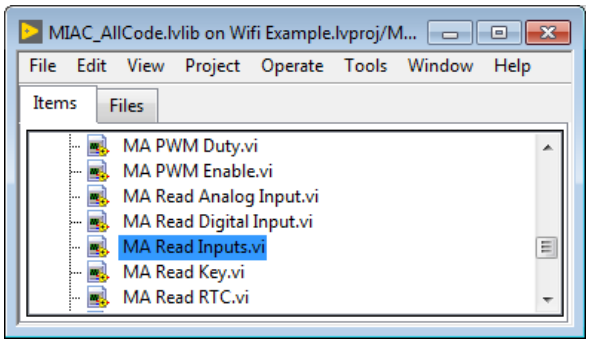
LabVIEW is a development environment for creating custom applications that interact with real-world data or signals in fields such as science and engineering.

It can also be used to control the MIAC AllCode.

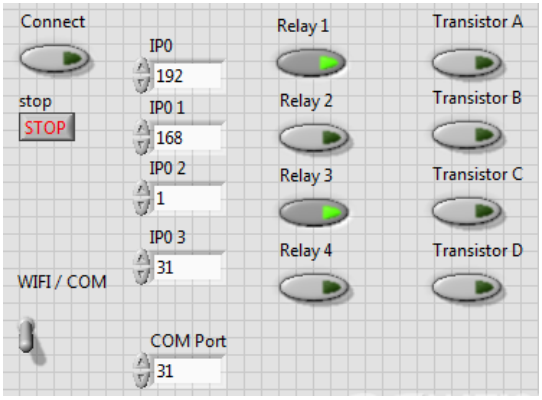
This section explains how to get started

Using the MIAC AllCode with Labview is fairly easy and consists of using a library provided by MatrixTSL. First, download the library (which consists of a DLL and a LabView library file) from the Matrix TSL website:

https://www.matrixtsl.com/allcode/resources/

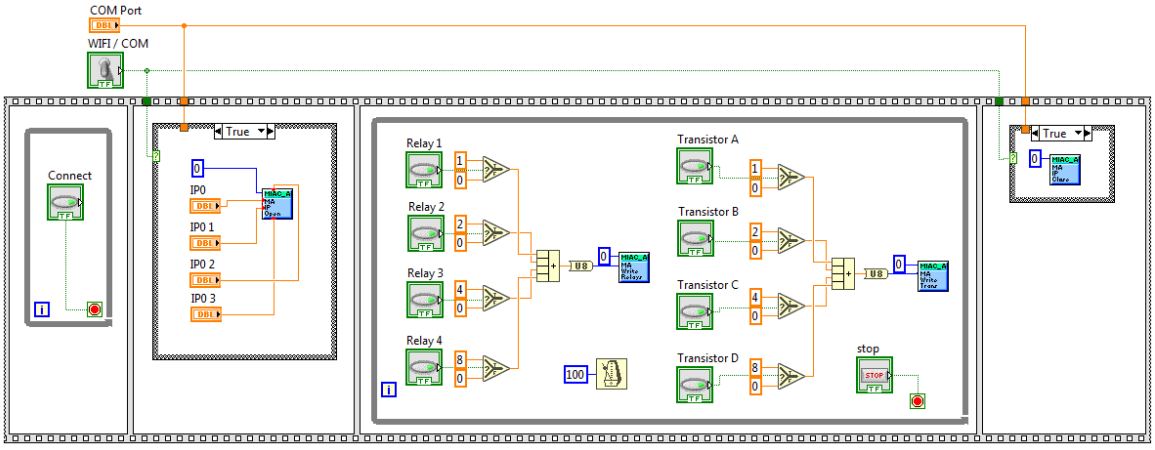


To begin, create a new blank VI and then open the "MIAC_AllCode.lvlib" file that was downloaded earlier. This contains all of the function calls to the AllCode API, as shown on the right.



A sample program is shown, using eight switches to control the state of the Relay and Transistor outputs.

The program at the bottom of the page shows a flat sequence structure that ensures the various parts of the program are called in turn. The left window executes first and wits for the connect switch to be pressed, the next window opens the communications port to the MIAC. The middle window loops until "stop" is pressed. It takes the value of the switches and sends it to the WriteRelays and WriteTrans API command every 100ms. The final window closes the communications port.



Transistor Outputs

Using PWM

MIAC AllCode Instructional Guide

There are four transistor outputs on the MIAC but there are actually six control lines from the internal microcontroller.

By default when using the transistor outputs as simple on/off controllers the extra two enables signals can be ignored.

When driving things like DC motors it can be useful to use PWM on the enable signals to set the speed of the motor.

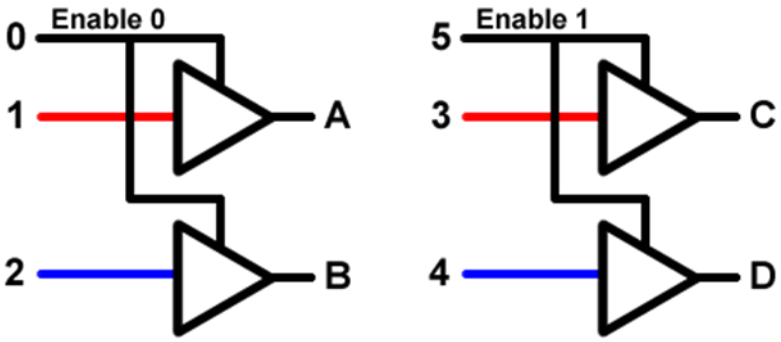
The transistor outputs can then be used to set the direction of the motor.

Two DC motors can be driven from the MIAC in full bridge mode.

For complete control all six outputs can be used with pulse width modulated (PWM) output and are numbered as follows.

- Channel 0 = Enable0 (AB)
- Channel 1 = Transistor Output A
- Channel 2 = Transistor Output B
- Channel 3 = Transistor Output C
- Channel 4 = Transistor Output D
- Channel 5 = Enable1 (CD)

Here is a block diagram of the internal workings of the transistor outputs.



The states of the control signals have the following effect on the transistor outputs.

Enable0	A	B	Output
Low	Don't Care	Don't Care	Coast
High	Low	Low	Brake
High	Low	High	Forwards
High	High	Low	Backwards
High	High	High	Brake

For simplicity the Enable0 and A/B outputs share a PWM timer, prescaler and period though the duty can vary for each output. Enable1 and C/D outputs also share their own unique PWM timer, prescaler and period.

PWM duties are 16-bit by default with the range of 0 to 65535.

The overall duty can be altered by changing the PWM period.

API Documentation

Input and Output

MIAC AllCode Instructional Guide

Connection			
Return	**Command**	**Parameter(s)**	**Description**
Status	ComOpen	Port	Open Bluetooth / Serial COM port Port = 1 to 255 Status = 0 (OK) or 255 (error)
Status	ComClose		Close Bluetooth / Serial COM port Status = 0 (OK) or 255 (error)
Status	IPOpen	IP0, IP1, IP2, IP3	Open WIFI IP Connection IP0 - IP3 = 0 to 255 Status = 0 (OK) or 255 (error)
Status	IPClose		Close WIFI IP Connection Status = 0 (OK) or 255 (error)
Version	GetAPIVersion		Returns the version number of the API Version = 1 to 65535

Transistor Outputs			
Return	**Command**	**Parameter(s)**	**Description**
	WriteTran	Channel Value	Writes a value to control a single transistor output Channel = 1 to 4 Value = 0 to 1
	WriteTrans	Value	Writes a value to control all four transistor outputs at once Value = 0 to 15
	PWMEnable	Value	Enables or Disables PWM functionality on the transistor outputs Value = 0 to 63
	PWMDuty	Channel Value	Sets a single PWM duty in terms of mark to space Channel = 0 to 5 - 0=ENAB / 1=A / 2=B / 3=C / 4=D / 5=ENCD Value = 0 to 65535
	PWMConfig	Channel Period Prescaler	Sets the PWM configuration Channel = 0 to 1 - 0=AB / 1=CD Period = 0 to 65535 Prescaler = 0 to 3 - 0=1:1 / 1=1:8 / 2=1:64 / 3=1:256

Inputs			
Return	**Command**	**Parameter(s)**	**Description**
Value	ReadDigitalInput	Channel	Reads the binary value of one of the input terminals Channel = 1 to 8 Value = 0 (Input Low) or 1 (Input High)
Value	ReadInputs		Reads the binary values of all the input terminals Value = 0 to 255
Value	ReadAnalogInput	Channel	Returns the analog value of one of the input terminals Channel = 1 to 8 Value = 0 to 1023

API Documentation
Display and Relay

MIAC AllCode Instructional Guide

Relay Outputs

Return	Command	Parameter(s)	Description
	WriteRelay	Channel Value	Writes a value to control a single relay output Channel = 1 to 4 Value = 0 to 1
	WriteRelays	Value	Writes a value to control all four relays at once Value = 0 to 15

Display

Return	Command	Parameter(s)	Description
	LCDClear		Clears all data on the LCD screen.
	LCDPrint	Text	Print text on the LCD. Text = <string>
	LCDNumber	Number	Prints a number on the LCD. Number = -32768 to 32787
	LCDCursor	X Y	Moves the LCD cursor to the selected X,Y position. X = 0 to 19 Y = 0 to 4
	LCDLine	X1 Y1 X2 Y2	Draws a line on the LCD from coordinates X1,Y1 to X2,Y2 X1 = 0 to 119 Y1 = 0 to 49 X2 = 0 to 119 Y2 = 0 to 49
	LCDRect	X1 Y1 X2 Y2	Draws a rectangle on the LCD from coordinates X1,Y1 to X2,Y2 X1 = 0 to 119 Y1 = 0 to 49 X2 = 0 to 119 Y2 = 0 to 49
	LCDPixel	X Y State	Controls a single pixel on the LCD at the specified coordinates. X = 0 to 119 Y = 0 to 49 State = 0 to 1 - 0=White / 1=Black
	LCDBitmap	X Y Filename	Reads a monochrome or 24-bit colour bitmap file from the SD card and draws on the display X = 0 to 119 Y = 0 to 49 Filename = “filename.bmp”
	LCDOptions	Foreground Background Transparent	Sets the options for drawing to the LCD Foreground = 0 to 1 Background = 0 to 1 Transparent = 0 to 1
	LCDBacklight	Brightness	Controls the brightness of the LCD backlight Brightness = 0 to 100 - 0=Off / 100=Brightest
	LCDVerbose	Value	Allows the LCD to be used to display the API calls as they are received Value = 0 to 1 - 0=Verbose Off / 1=Verbose On

API Documentation
Peripheral

MIAC AllCode Instructional Guide

Keypad

Return	Command	Parameter(s)	Description
KeyPress	ReadKey		Reads the last recorded keypress from the keypad buffer KeyPress = 0 to 255 - 0=F2 / 1=Left / 2=F1 / 3=Down / 4=OK / 5=Up / 6=Return / 7=Right / 8=Menu / 255=No Press

SD Card

Return	Command	Parameter(s)	Description
Status	CardInserted		Checks to see if an SD card is inserted Status = 0 to 1 - 0=No Card / 1=Card Present
Status	CardOpen	Filename	Attempts to open a file from the SD card Filename = "filename.txt" Status = 0 to 255 - 0=OK / 239=FileNotFound
Value	CardReadByte		Reads a byte from the open file Value = 0 to 255
Status	CardWriteByte	Value	Appends a byte to the end of the open file Value = 0 to 255 Status = 0 to 255 - 0=OK / >0=Error
Status	CardCreate	Filename	Attempts to create a file from the SD card Filename = "filename.txt" Status = 0 to 255 - 0=OK / 1=FileExists
Status	CardDelete	Filename	Attempts to delete a file from the SD card Filename = "filename.txt" Status = 0 to 255 - 0=OK / 239=FileNotFound

Real Time Clock

Return	Command	Parameter(s)	Description
Value	ReadRTC	Unit	Reads a value from the real time clock Unit = 0 to 5 - 0=Sec / 1=Min / 2=Hour / 3=Day / 4=Month / 5=Year Value = 0 to 99
	WriteRTC	Unit Value	Writes a value to the real time clock Unit = 0 to 5 - 0=Sec / 1=Min / 2=Hour / 3=Day / 4=Month / 5=Year Value = 0 to 99

Serial Communications

Return	Command	Parameter(s)	Description
Value	CommsRead	Timeout	Reads a byte from the alt communications bus Timeout = 0 to 255 Value = 0 to 255
	CommsWrite	Value	Writes a value to the alt communications bus Value = 0 to 255
	CommsBaud	Baud	Sets the speed of the alt communications bus Baud = 0 to 7 - 0=1200 / 1=2400 / 2=4800 / 3=9600 4=19200 / 5=38400 / 6=57600 / 7=115200

Matrix Technology Solutions Ltd.
33 Gibbet Street
Halifax
HX1 5BA

t: +44（0）1422 252380
e: sales@matrixtsl.com

www.matrixtsl.com

CP8656