MATRIX

Digital Techniques in Aviation

Contents

# Worksheet 1 - Understanding Flowcode Embedded

AEROSPACE

Code for microcontrollers can be developed in a number of ways: using a low level language like C or even assembly code, or using a high level language like Flowcode.

High level languages are compiled to low level languages and then binary which the microcontroller understands.

Flowcode allows you to develop programs using flow charts. In this worksheet you will explore how Flowcode works.

The photograph shows a low level programming language.
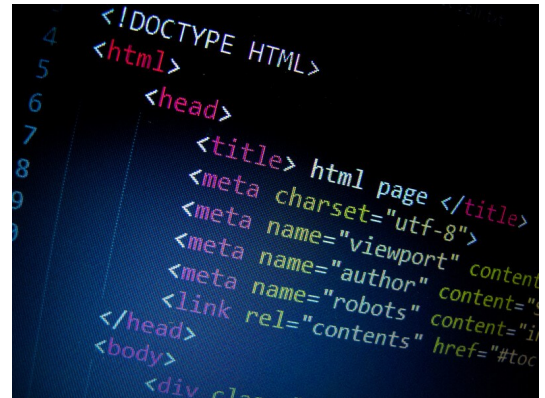
Over to you:

This section allows those who are new to Flowcode to understand how it can be used to develop programs. It allows you to create programs step-by-step.

We advise you to work through every section to familiarise you with the options and features of Flowcode and introduce you to a range of programming techniques. As you work through each part, please also refer to the Flowcode help file, the Flowcode Wiki and the Flowcode web site (at www.flowcode.co.uk).

Specifically in this section you will learn:

- how to use each Flowcode icon (except the C code icon);
- how the fundamental components in Flowcode work - the LED, LCD, ADC, switch, 7-segment display, 7-segment quad display, keypad and EEPROM components.

What is Flowcode Embedded?

Flowcode Embedded allows you to create microcontroller applications by dragging and dropping icons on to a flowchart to create programs. These can control external devices attached to the microcontroller such as LED's, LCD displays etc.

Once the flowchart has been designed, its behaviour can be simulated in Flowcode before it is compiled, assembled and transferred to a microcontroller.

The process:

- Create a new flowchart, specifying the microcontroller that you wish to target.

- Drag and drop icons from the toolbar onto the flowchart to program the application.

- Add external devices by clicking on the buttons in the Components Libraries toolbar.

- Edit their properties, including how they are connected to the microcontroller, and configure any macros they use.

- Run the simulation to check that the application behaves as expected.

- Transfer the application to the microcontroller by compiling the flowchart to C, then to assembler code and finally to object code.
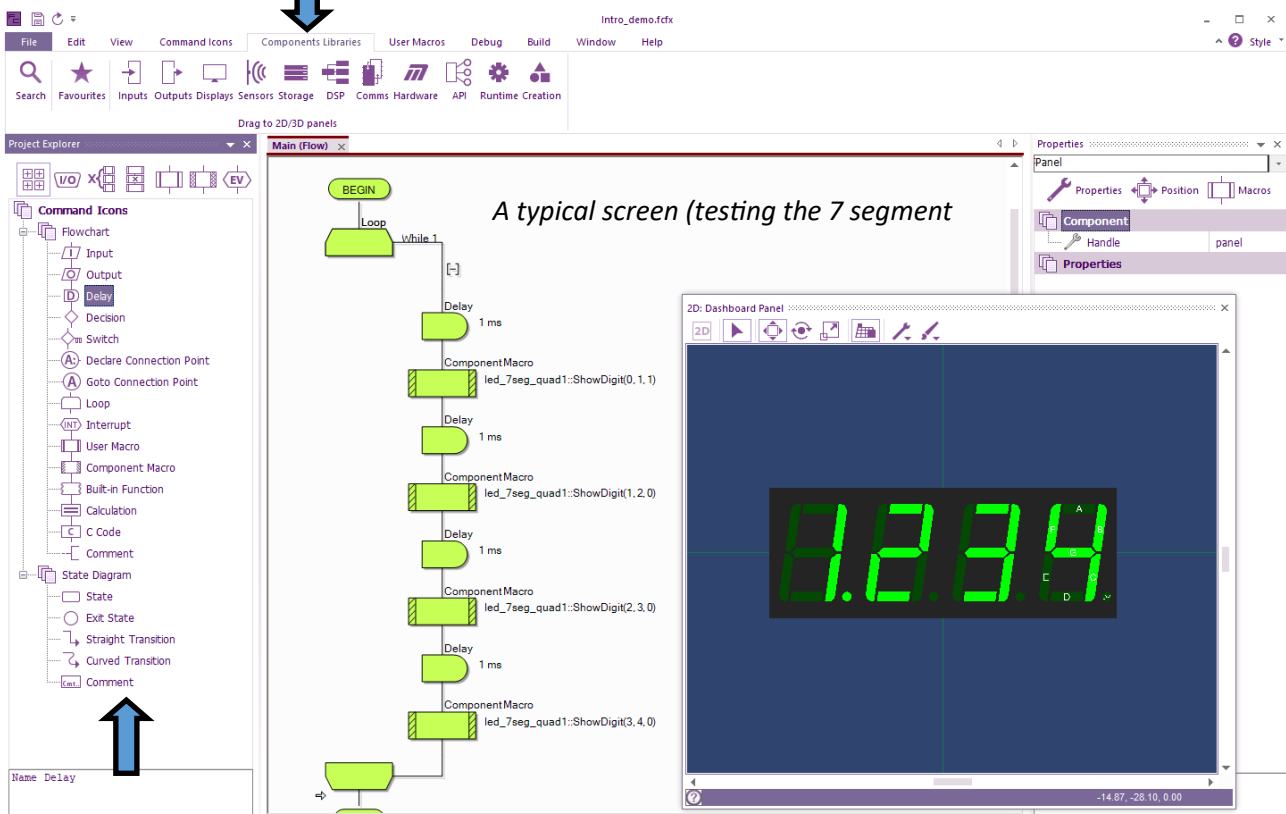
Worksheet 1 - Understanding Flowcode Embedded

The Flowcode environment consists of:

- a main work area in which the flowchart windows are displayed;

- a number of toolbars that allow icons and components to be added to the flowchart;

- the System and Dashboard panels that display the attached components and provide basic drawing capabilities;

- the Project Explorer panel that shows project variables, macros and component macros;

- the Icon List panel that shows bookmarks, breakpoints and search results;

- windows that allow the status of the microcontroller to be viewed;

- windows that display variables and macro calls when the flowchart is being simulated.

Components Libraries toolbar

Connect external components to the microcontroller or use basic panel drawing commands. Components are grouped in different categories that appear as drop down menus. Click on a component and it will be added to the microcontroller and   appear on the panel. The pin connections and properties of the component can then be

A typical screen (testing the 7 segment

edited.

Worksheet 1 - Understanding Flowcode Embedded

2D: Dashboard and 3D: System Panels

The components that you connect to the microcontroller will be displayed on a dashboard panel. Panels also provide basic drawing features like lines, shapes, and images.

The Dashboard Panel is primarily for 2D use although offers a 3D view. It is generally used as an interface where buttons and switches of interactive components are kept.

The System Panel is the main 3D panel, offering many more features and options:

- full camera control;
- editable background environments with default 'Sky Dome' and 'World Dome' views;
- the option to use an image as the background;
- 'Shadow mode' offering both 'Tabletop' and 'Object' shadow options.

More details on these panels are found in the 'Flowcode - Getting Started Guide'.
(View > 3D: System Panel) / (View > 2D: Dashboard Panel)
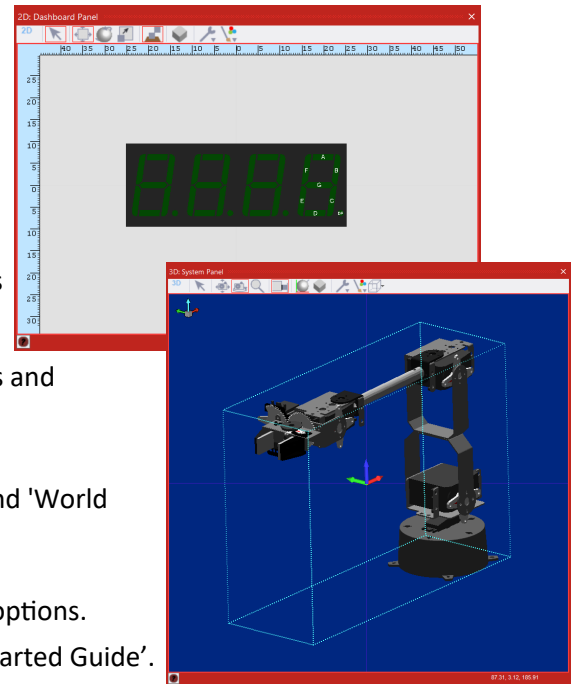
Component Properties panel

All items on the panel, including the panel itself, have associated properties that are displayed in the Properties pane when the item is selected.

Some are read-only while others can be manipulated.

Some, like size and position, change as you interact with the item.

Others allow access to more advanced features of the selected item.

The Properties pane typically docks to the right hand side of the screen but looks like this when undocked: **(View > Component Properties)**
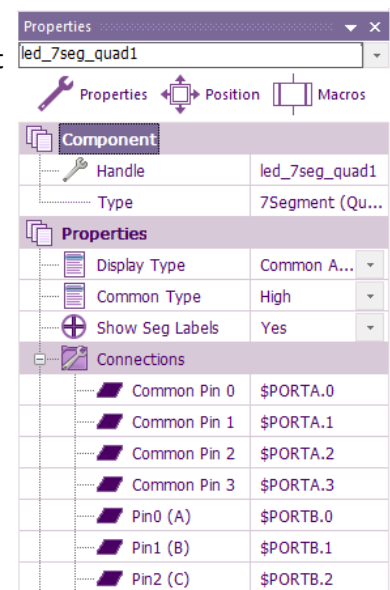
Project Explorer

The buttons along the top of this panel allow you to select 'Ports', 'Globals', 'Macros' and 'Components'.

The 'Ports' view shows variable names assigned to the microcontroller ports.

The 'Globals' view shows any constants and variables that have been defined for use in the current project.

The 'Macros' view shows user-created macros in the current program and allows the user to drag them into the current flowchart.

The 'Components' view is very similar except that it also lists components that are present in the panel. **(View > Project Explorer)**
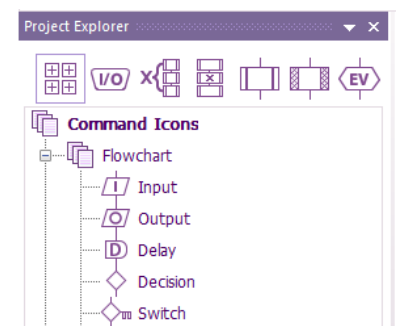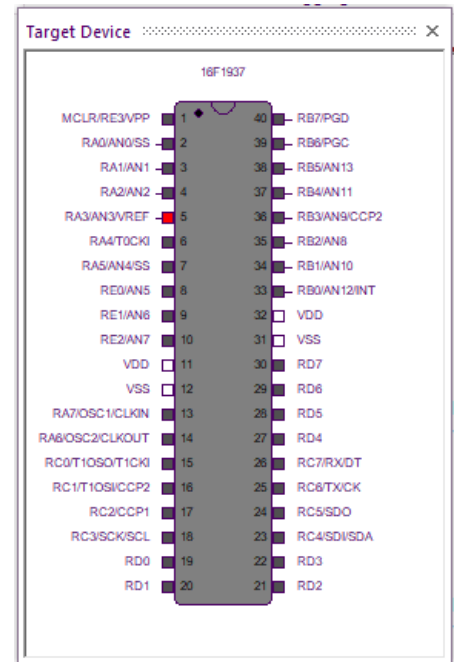
Worksheet 1 - Understanding Flowcode Embedded

Target device window

The pinout for the currently selected microcontroller chip is displayed.

When the flowchart is being simulated, the state of the microcontroller I/O ports are shown on the microcontroller as red and blue, for high and low outputs respectively.

- **(View > Target Device)**

Docking and undocking the toolbars and panes

Toolbars and panes can be undocked from their default positions and either be left free floating, or docked to the top, bottom or the sides of the Flowcode window.
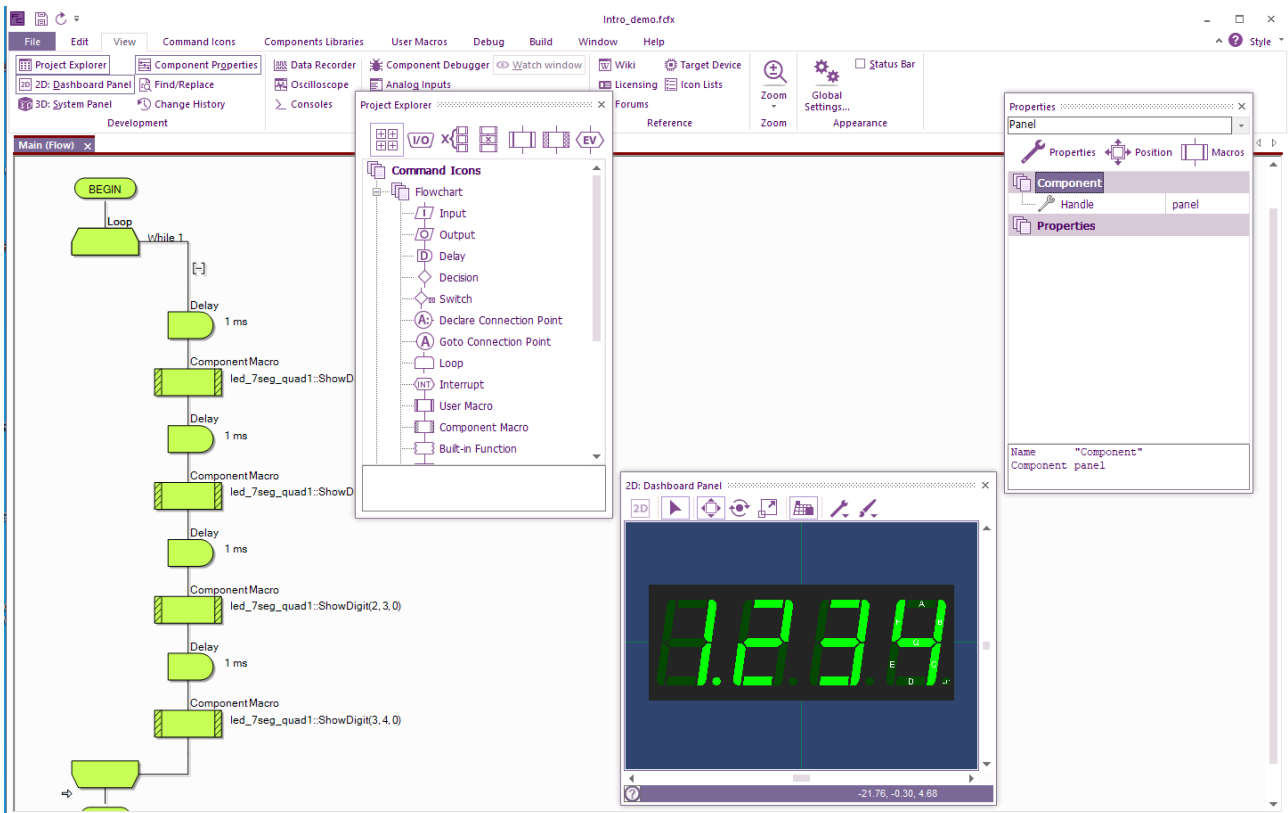
An example showing floating toolbars:

To undock a docked toolbar, simply click and hold on the toolbar 'grab bars' (the top of the toolbar). Drag the toolbar to its new position. To dock it again, double-click on the **grab bar**.

Worksheet 1 - Understanding Flowcode Embedded

Flowchart window

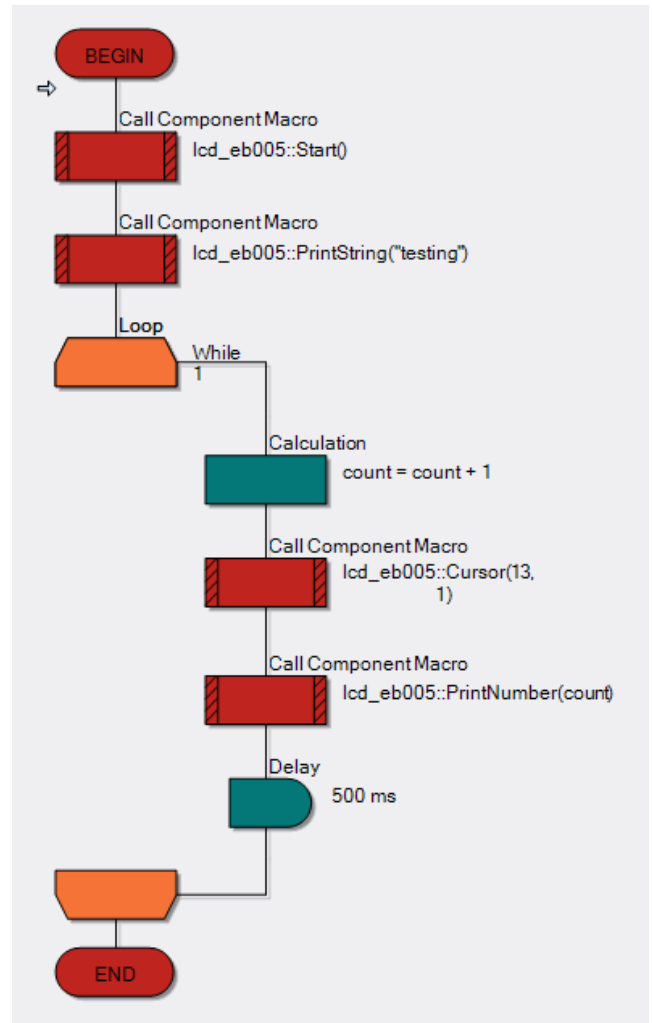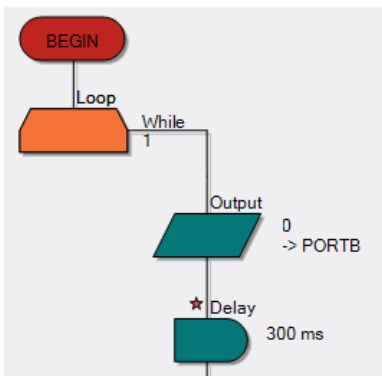The icons that make up the flowchart are displayed in this main space. The text will change depending on properties selected, component macros called etc. The display names can be changed by the user to aid project organisation.

A red star alongside an icon indicates that the flowchart has not been saved in its current form.

Simulation

When simulating a program in Flowcode the red rectangle indicates the icon to be executed next.
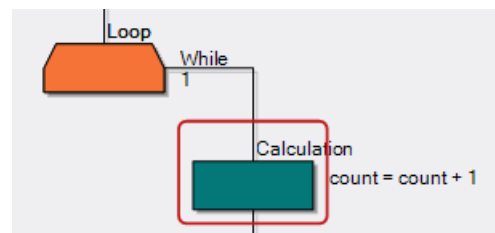
Simulation Debugger

When simulating a flowchart, the current values of any variables used in the program can be seen in this window. These are updated after a command is simulated unless the simulation is running at full speed - 'As fast as possible'.

If you simulate a flowchart and then press the pause button, you can click on variables in this window to change their value. This allows you to test your flowchart under known conditions.

The window also shows the current macro being simulated under the 'Macro Calls' section, useful when one macro calls another during the simulation process.
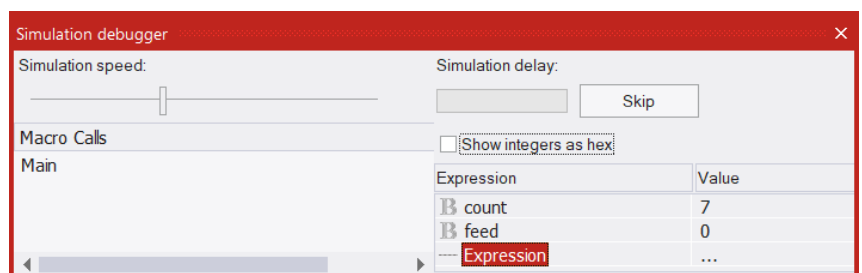
Worksheet 1 - Understanding Flowcode Embedded

Starting a new flowchart

- Create a new flowchart by clicking on the 'New Project' button or by selecting **File > New Project**.
- Select the microcontroller that you wish to target from the list presented.
- Click the 'New Embedded Project' button.

Opening an existing project

There are a number of ways of opening an existing Flowcode project:

- Select the Open option from the File menu **(File > Open Project)**.

    or

- Select the file from the list of most recently used files in the File menu.

    or

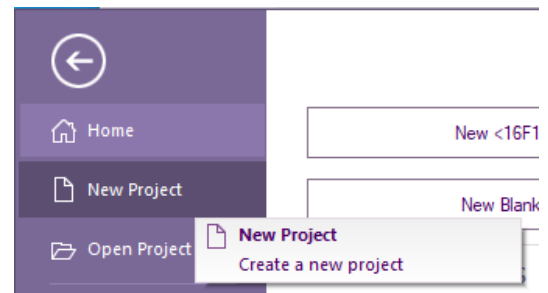- Double-click on a Flowcode (.fcfx) file in Windows Explorer to launch Flowcode and open the file.

Saving a Flowchart

To save a flowchart, select either the 'Save' or 'Save As' options from the File menu **(File > Save / Save As)**. Flowcharts must be saved before they can be compiled to C or transferred to a microcontroller.

Saving Flowchart Images

To save an image of the currently active flowchart, select 'Save current Flowchart...' from the 'Save Image' sub-menu in the 'File' menu **(File > Save Image > Save current Flowchart...)**.

This function saves an image of the program to any file in the format chosen from the list:

- Bitmap (*.bmp);
- JPEG (*.jpg;*.jpeg);
- GIF (*.gif);
- PNG (*.png).

Note that the current zoom rate is used to determine the resolution of the image saved. If you need high quality images for printing then increase the zoom setting.

From the 'Save Image' menu, you also have the option to save the current image of either the 'Dashboard Panel' or the 'System Panel' **(File > Export > Save Dashboard image... / Save System image...)**.

These images can be saved to any file format chosen from the list:

- Model (*.mesh)
- Bitmap (*.bmp)
- JPEG (*.jpg;*.jpeg)
- GIF (*.gif)
- PNG (*.png)

Worksheet 1 - Understanding Flowcode Embedded

The View menu

This dictates which panels and toolbars appear on the workspace, a useful feature when trying to simplify its appearance.

It also has a **Zoom** menu, which allows you to display more icons in the workspace window than when using the default zoom setting.

The current zoom setting is displayed on the Zoom sub menu, and on the right hand side of the status bar, at the bottom of the Flowcode window.

The size of each icon is dictated by the zoom level - for larger icons, zoom in - for smaller icons, zoom out. Use the Print Preview function to optimise the appearance of your flowchart on the paper.

The **Zoom** menu can also be accessed by right-clicking on the flowchart workspace.

Function key shortcuts:

- Increase Zoom (F3) - increases zoom size by 5%;
- Decrease zoom (F2) - decreases zoom size by 5%;
- Default zoom (F4) - set zoom to 75%;
- Zoom to fit - Zooms to fit the whole flowchart into the current window;
- Zoom to fit width - Zooms to fit the width of the flowchart into the width of the window.

Global Settings

The **View** menu also includes a **Global Settings** for configuration of application and flowchart

(View > Global Settings)   Then select the appropriate Tab.

Application Tab

This tab enable setting of general application settings, such as language, document appearance, autosave, feature, code generation options and web access.

The OpenGL graphics engine can here be set as hardware or software mode.

The **Override language** option allows the user to override the default Flowcode language settings and to display Flowcode in a specified language. To do so, select the language from those available on the drop down list and restart Flowcode. It will do so in the selected language, provided the relevant language pack has been installed.
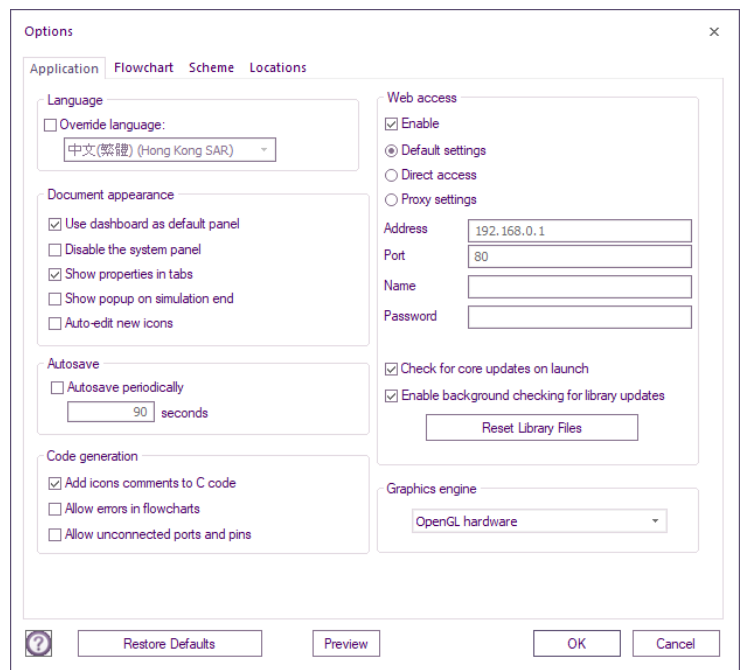
Worksheet 1 - Understanding Flowcode Embedded

Flowchart Tab

This tab enables settings for flowchart display styles, text size and font, as well as annotation and tooltip style customizations.
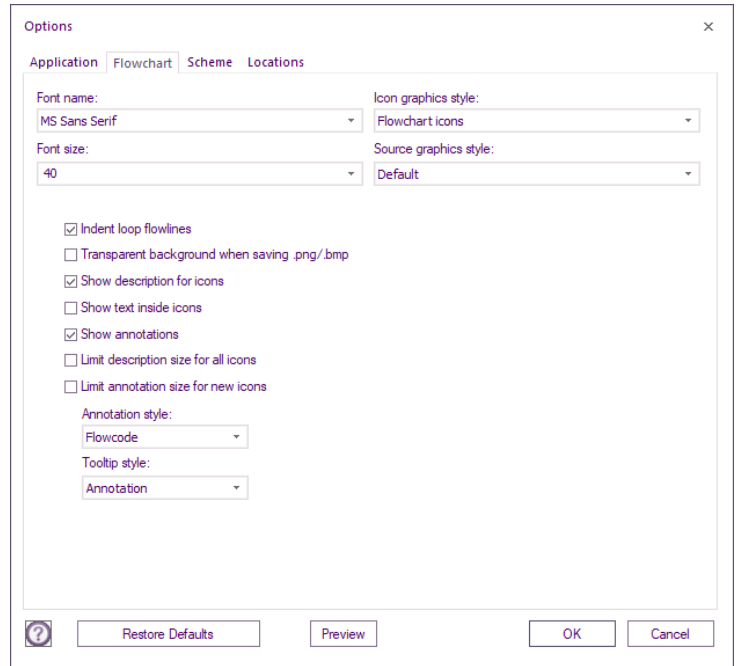
Scheme Tab

This tab contains the settings for changing the appearance of the flowchart, including icon colours and graphics, background colours and patterns etc.
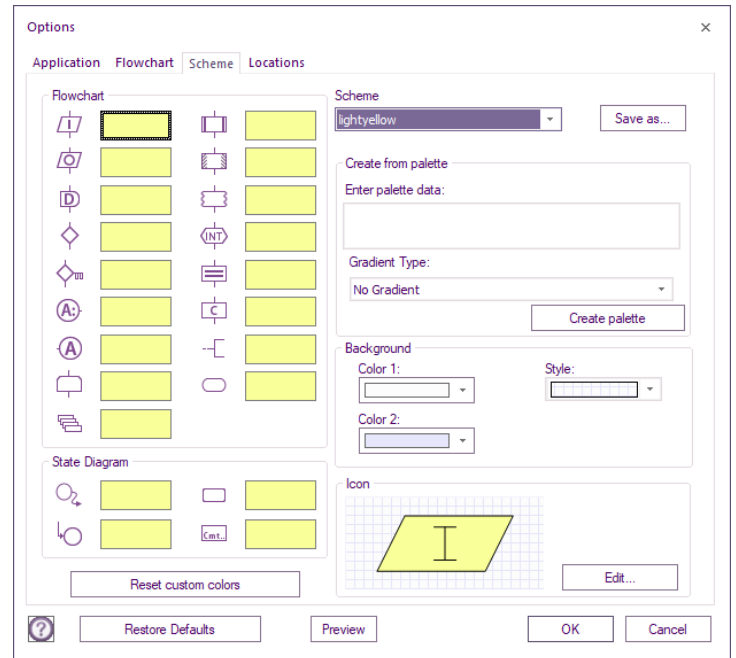
Worksheet 1 - Understanding Flowcode Embedded

Locations Tab

This tab enables the setting of backup filenames and the location of toolchain directories, listing programming tools that can be used in developing the program.

Additional directories can be added for the location of custom components.
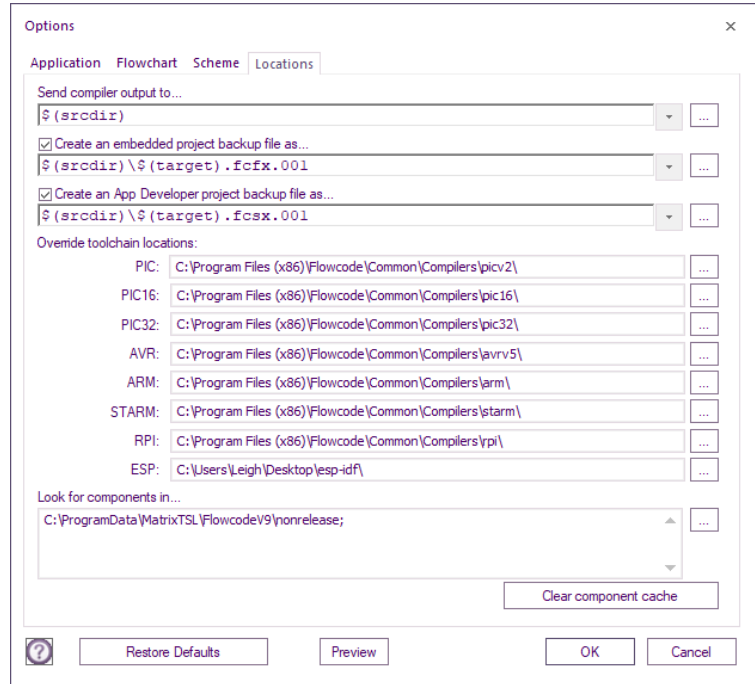
View Windows (Simulation)

Analog Inputs and Digital Pins

Analog input values can be set and digital pins monitored and set via

View > Analog Inputs

and

View > Digital Pins

Worksheet 1 - Understanding Flowcode Embedded

AEROSPACE
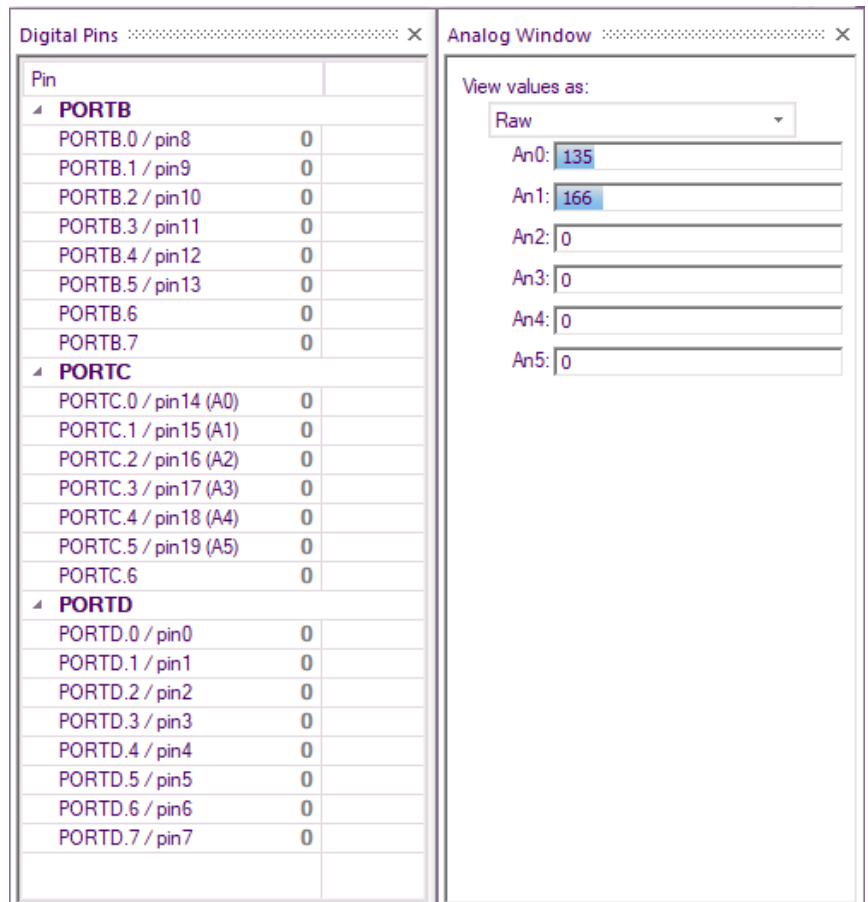
Getting Help with Flowcode

Flowcode has within it and online an extensive **wiki** which can be accessed through the Help toolbar menu or via an internet browser and visiting this page:

http://www.flowcode.co.uk/wiki/

Additionally every single component within Flowcode has a page on the **wiki** which explains all the macros within it, and usually includes some examples as well.

To access the component help simply right-click your mouse on any component in either the 2D or 3D panel and select **Help**.

From here you can see:

- Explanation of the component
- Some examples of the component in use
- Macro references explaining what each macro plus the parameters and return values.

Library Updates

Flowcode components and target device information is kept up to date via an online system accessed from the Help menu **(Help > Library Updates)**

Worksheet 2 - Flowcode first program

In modern aircraft, there are no switches, indicators or displays connected directly to an electronic system. Instead, they connect to a microcontroller system.

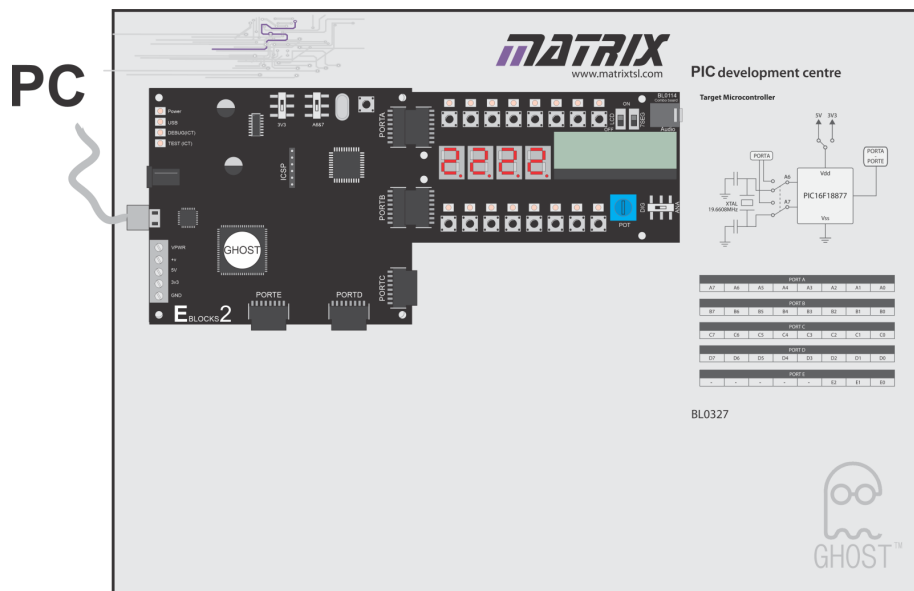In this section you will learn about the basics of microcontrollers, how they are programmed and how they connect to circuit elements around them.

The photograph shows typical cockpit with night time displays

Over to you:

- Set up the equipment as in the diagram above. Plug in the power supply. Load Flowcode Embedded.
- Read 'Introduction to microcontrollers' in the Reference section and then refer to it as necessary as you progress through the course. This will tell you about the basics of how microcontrollers work
- Read the 'Using E-blocks' document in the Reference section and then refer to it when needed as you progress.
- Familiarise yourself with the E-blocks II datasheet which includes circuit diagrams of all of the E-blocks you will use on this course. You will need this when doing the exercises.
- Work through the following pages that show you how to set up your first simple program using Flowcode and download it to the hardware. You will create a program that lights an **LED** attached to the microcontroller. This program introduces the topic of how to control a digital output.
- For the components you have used sketch the circuit diagram, to include the microcontroller chip, the LEDs and their connection to the microcontroller, the crystal, and the power connections.

Worksheet 2 - Flowcode first program

Starting a new project

Select 'New Project' at the welcome screen, or via the menu **(File > New Project)**
On the "Embedded" tab choose a target device or development board.

You will notice that the selection list includes details of the features and peripherals of each target.
This is useful when selecting a device for a particular project.

For our new project, if we are using for example the MatrixTSL E-blocks2 PIC development board,
choose the BL0011 target from the "Free targets" list.

Arduino users please:
- *select an appropriate Arduino development board;*
- *use ports C and D as appropriate. (Port C on the Arduino 'Maps' to Port A of the Combo board).*
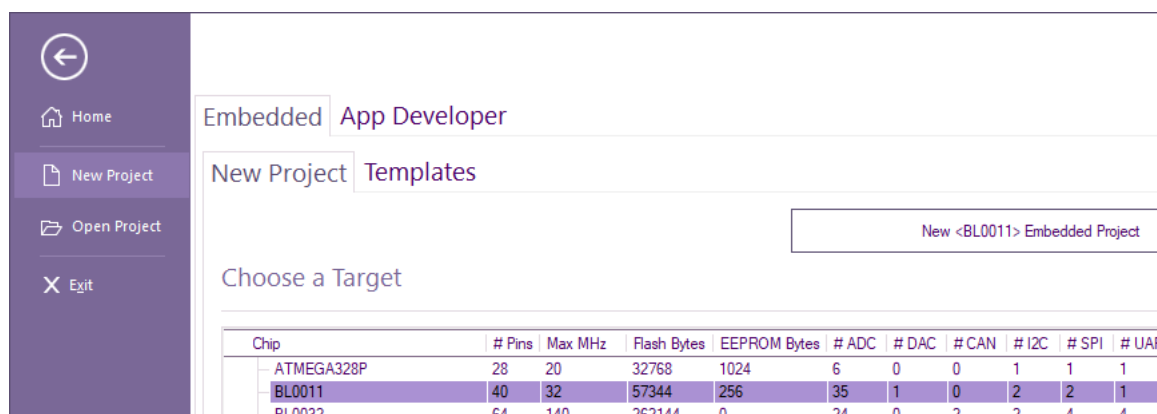
In this case the target choice also selects the correct 16F18877 device and presets the correct values for clock oscillator frequency and other settings.

Click on the "New <BL0011> Embedded Project" button to start the project.

TIP: The project target device can be changed later via the menu **(Build > Project Options)**

Worksheet 2 - Flowcode first program

Add an LED Array (PCB) to the 3D system panel.

Click on the 'Component Libraries' tab and select the LED array from the **Outputs** section .

(Component Libraries > Outputs > LED Array (PCB) > Add to 3D system panel)
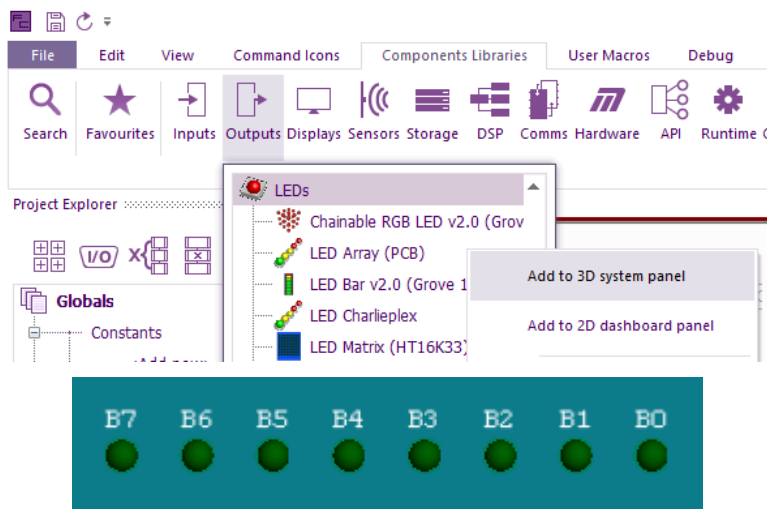
Create the Flowchart.

Move the cursor over the **Loop** icon, in 'Command Icons' . Click and drag it over to the work area.

While dragging it, the normal cursor changes into a small icon. Move it in between the 'BEGIN' and 'END' icons.

As you do so, an arrow appears showing you where the **Loop** icon will be placed. Release the mouse button to drop the icon in between the 'BEGIN' and 'END' boxes.

Add an **Output** icon within the loop on the flowchart in the same way.

TIP: The colours of the icons on your system may be different.
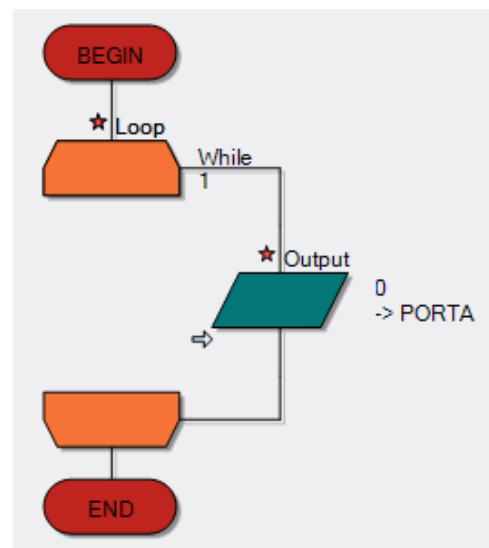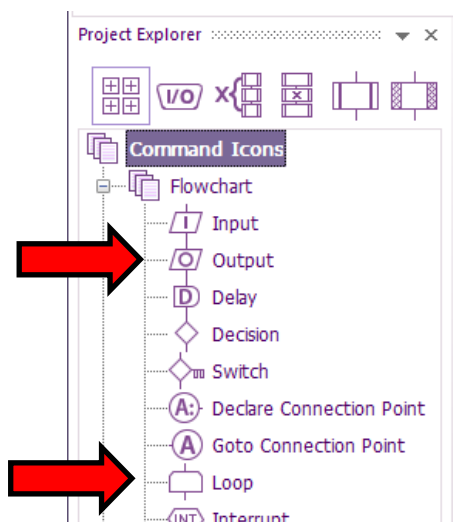
Worksheet 2 - Flowcode first program
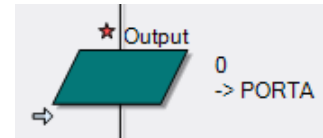
Changing port settings

- Double click on the Output icon in your flowchart and the **Properties** box will appear and show that it is currently connected to Port A.

    The LEDs in your 3D system panel are currently attached to Port B, so we need to connect the **Output** to that port.

- Select Port B. Input a value of 1.
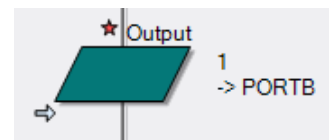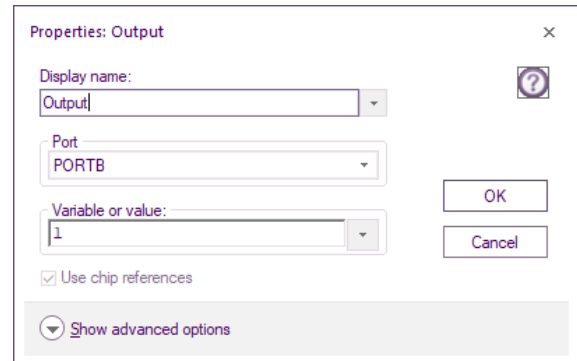
Run the simulation.

Select the **Go** icon from the **Debug** menu bar and the program simulation will light up the LED in the 3D system panel.
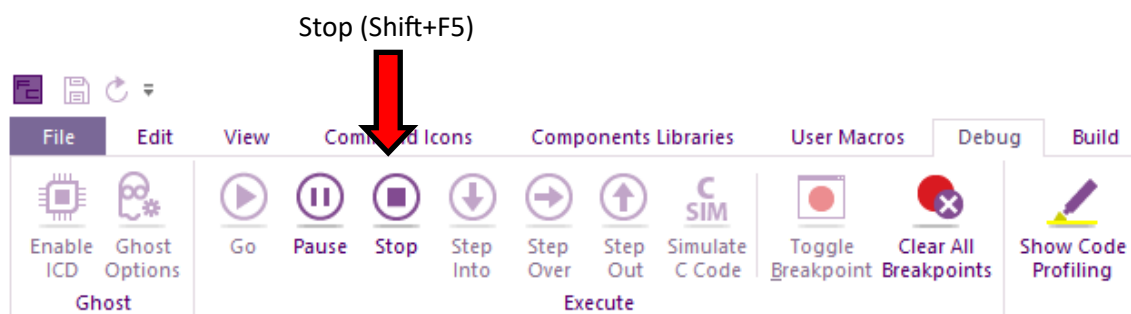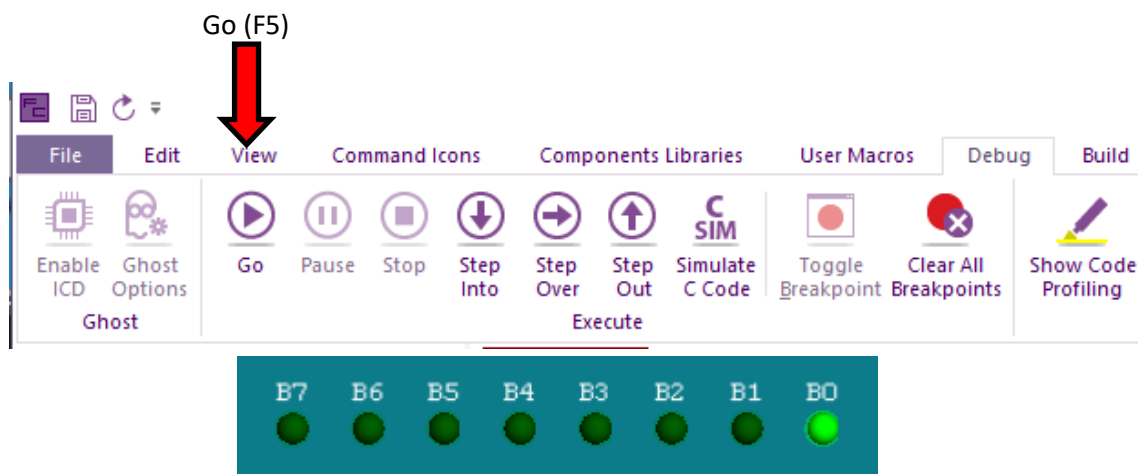
Click on the **Stop** icon and the program simulation will end.

TIP: Remember to stop your simulation before doing anything else. (If Flowcode isn't doing as you expect, check that you haven't accidentally left your simulation running).

Worksheet 2 - Flowcode first program

Saving your program:

Save your program. **(File > Save)**

Connect the target development board to a power supply.

Connect the USB programming lead to your PC.

Click the **Compile to Target** button on the **Build** menu as shown below: **(Build > Compile to target)**
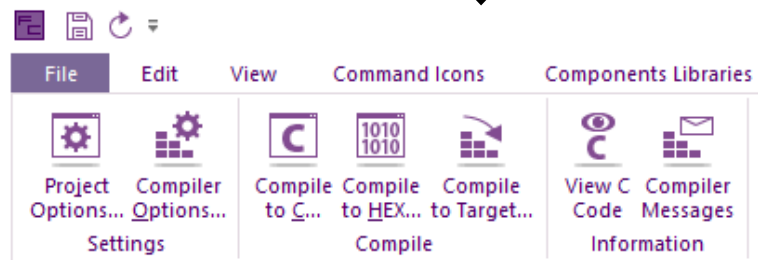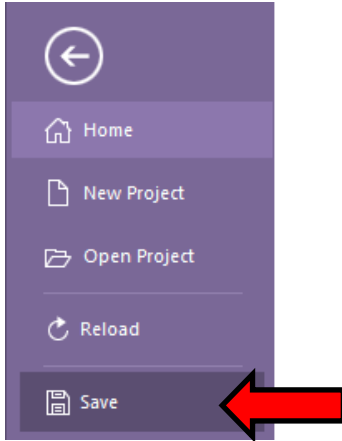
Having successfully lighting your LED, try these changes .

Highlight the image of the LED array in the 3D system panel and right click to select the **Properties**.

Change the number of LEDs in your array by changing the value under **count**.

Change the colour of the LEDs in the simulation as shown below.

(Wording layout and font size modified.)
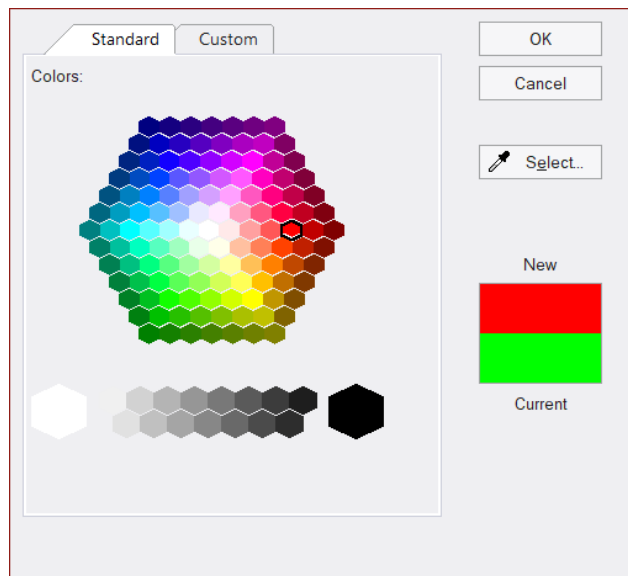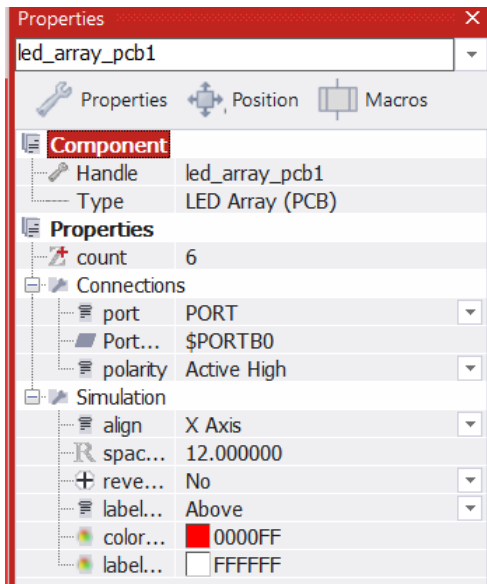
Property settings for six red LEDS.

Six red LEDS in simulation.

Worksheet 2 - Flowcode first program

Changing the port settings.

Bring up the **Output** icon properties (double click on the icon) and change the Port settings to **Port A.**
Highlight the image of the LED array in the 3D system panel and right click to select the **Properties**.
Change the Port settings to **Port A** and insert a value of 1.
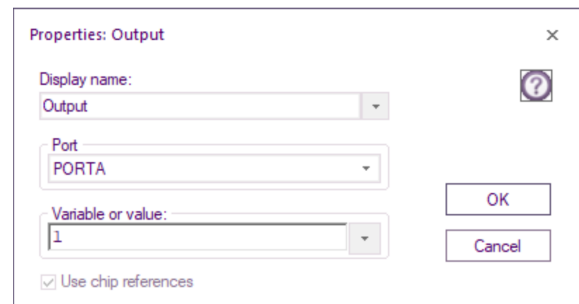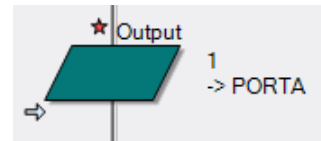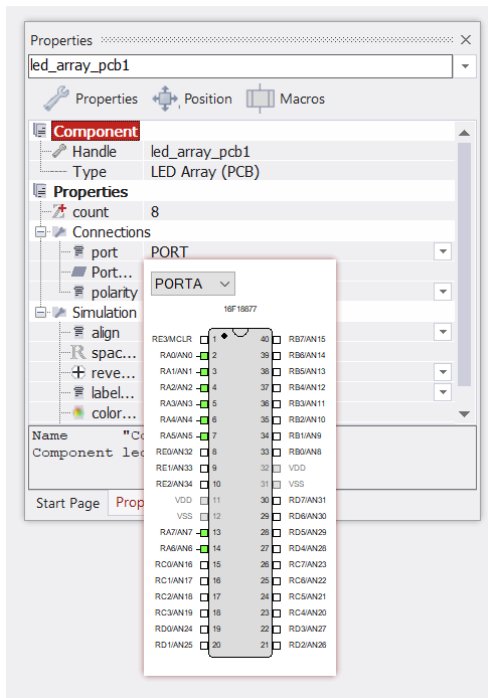
Run in simulation mode and then compile to chip.
You should see the first LED on the upper row light up.

Practise changing the ports by changing them back to **Port B**.
Change the **value** from 1 to **255**.
Test in simulation mode and then compile to chip (all eight LEDs light up).
Experiment using other values.
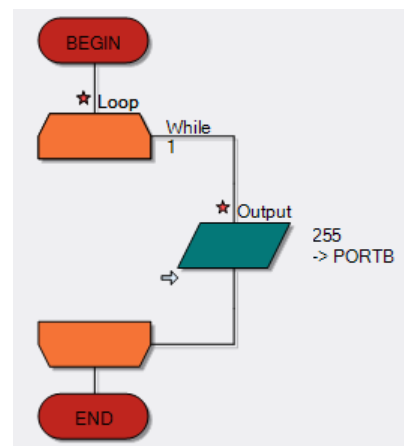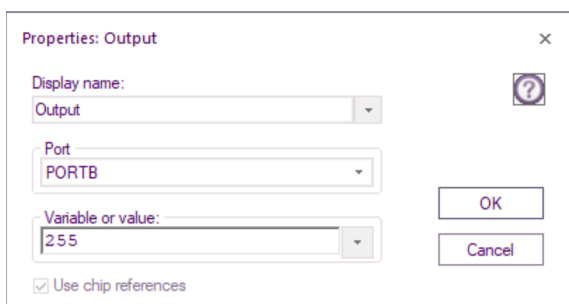
Worksheet 3 - Binary and hexadecimal

Digital electronic devices can't cope with decimal numbers (0, 1, 2, ..9 etc.).

Instead, they use the binary system, which uses only two numbers 0 and 1. The number 1 could be represented by a high voltage signal, such as 5V, while number 0 could be a low voltage, such as 0V. (Wording moved up from lower in the page.)

The photograph shows a display using binary numbers.

Number sent to output	LED array
51	B7 B6 B5 B4 B3 B2 B1 B0 ○○○○○○○○
204	B7 B6 B5 B4 B3 B2 B1 B0 ○○○○○○○○
195	B7 B6 B5 B4 B3 B2 B1 B0 ○○○○○○○○
_____	B7 B6 B5 B4 B3 B2 B1 B0 ●●●●●●●●
_____	B7 B6 B5 B4 B3 B2 B1 B0 ●●●●●●●●
_____	B7 B6 B5 B4 B3 B2 B1 B0 ●●●●●●●●

Over to you:

Complete the table above by:

- shading in the LEDs that light, in the first three rows.
- working out what number produces the LED patterns shown in the last three rows.

Use Flowcode to check your answers.

Can you light the same LED patterns using Hex?

(Enter a hex number into Flowcode by preceding it with '0x').

So what?

The table opposite shows how decimal and binary number systems compare:

The decimal system uses ten numbers, 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9.
On reaching the last of these, '9', we start again with '0', but add another number in front.

For example, after '8' and '9' comes '10', and after '18' and '19' comes '20' and so on. When we reach '99', both of these go back to '0''s but with a '1' in front, to make '100'.

Decimal	Same in binary
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010

Worksheet 3 - Binary and hexadecimal

AEROSPACE

In binary, the same thing happens, but a lot more often, because it uses only '0''s and '1''s.

Counting up starts with '0', then '1', then back to '0' with a '1' in front, making '10' (not ten - it's two!)
Next comes '11' (three) and then, start again, with two '0's but with a '1' in front, to give '100' (four) and so on.

Notice that each time the binary '1' moves one place to the left, it doubles in value of the number in decimal, as the second table shows.
We can use this idea to convert between number systems.

TIP: In any binary number, the bit at the left-hand end, the Most
Significant Bit (MSB), has the highest value.
The one at the right-hand end, the Least Significant Bit (LSB), is worth least

Decimal	Same in binary
1	1
2	10
4	100
8	1000

BINARY VALUE				
16	8	4	2	1

Hex numbers

Hexadecimal, 'hex' for short, is a more convenient form than binary (for humans) for representing numbers.

- A binary digit is either 0 or 1.
- A decimal digit varies between 0 and 10.
- A hex digit has sixteen possible states.

Clearly having sixteen states is a problem, as we have only the digits from 0 to 9. To get around this, we use the letters **A** to **F** to provide the additional six digits.

Working with eight bit binary numbers is handy as computers (and the PIC MCU) store information in groups of eight bits.

A single memory cell inside a PIC device can store a number ranging from 0000 0000 and 1111 1111.
In decimal this range is 0 to 255.
The equivalent in hex is 0 to FF.

Decimal	Binary	Hex
0	00000000	0
1	00000001	1
2	00000010	2
3	00000011	3
4	00000100	4
5	00000101	5
6	00000110	6
7	00000111	7
8	00001000	8
9	00001001	9
10	00001010	A
11	00001011	B
12	00001100	C
13	00001101	D
14	00001110	E
15	00001111	F

Binary value					Decimal value
16	8	4	2	1	
				1	1
			1	0	2
		1	0	0	4
	1	0	0	0	8
			so		
		1	1	1	7
	1	0	0	1	9
1	0	1	0	0	20
1	1	1	1	1	31

Worksheet 4 - Flowcode examples

Microcontroller circuits contain only a small number of components - the microcontroller itself plus switches, indicators, controls, displays, actuators, as needed.

The huge variety in the functionality of circuits is dictated by the software used.

In this section you will use the same basic hardware to create six very different systems using Flowcode.

The photograph shows a turboprop aeroplane cockpit.
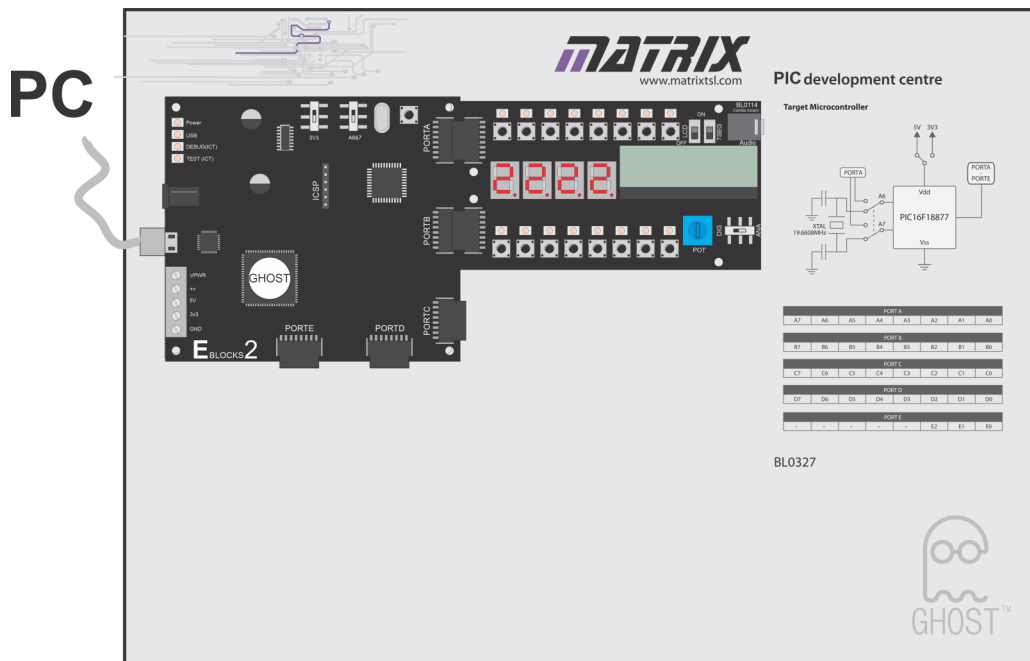
Over to you:

Set up the hardware as shown in the diagram above.

Use it to work through the examples below, which are described in more detail in the following pages:

- example 1: Adding digital inputs
- example 2: Using loops
- example 3: The LCD display
- example 4: A stopwatch
- example 5: Binary adder
- example 6: Binary logic in control

For each example:

- download the program to the hardware and check out its functionality.
- sketch the circuit diagram to show the components used, their connection to the microcontroller, the crystal, and the power supply to the microcontroller.

Worksheet 4 - Flowcode examples

 AEROSPACE

Example 1: Adding digital inputs - Where's the fire?

The scenario!

A large building has a number of heat sensors in its fire alarm system. When a fire occurs, the fire brigade needs to know where it is. In other words, they need to know which heat sensor has triggered the alarm.

The system is controlled by a PIC device. There are five heat sensors, connected as inputs to Port A. Port B is set up as the output port and connected to a set of five LEDs. If a heat sensor detects a fire, the corresponding LED lights up.

The PIC fire monitor
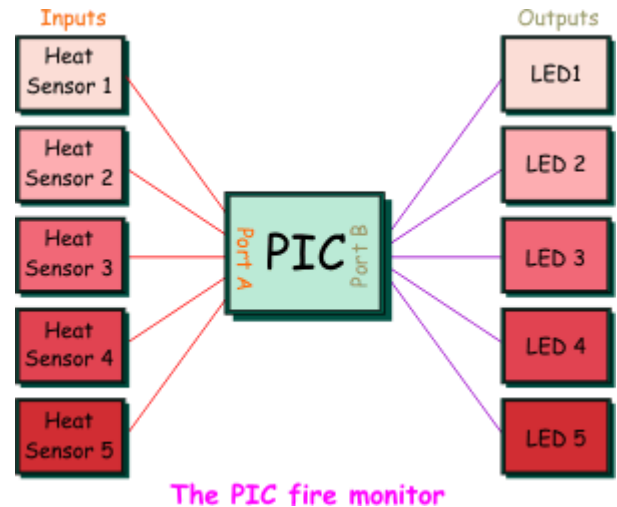
Setting up the flowchart

Open Flowcode and create a new project suitable for the board you are using.

Drag the **Loop icon**, the **Input icon** and the **Output icon** into your Flowchart from the icon toolbar to create the Flowchart shown.

Connect **Input** to Port A and **Output** to Port B.

Arduino users:
please use Ports C and D as appropriate.
(Port C on the Arduino 'maps' to Port A of the Combo board).

Worksheet 4 - Flowcode examples

AEROSPACE
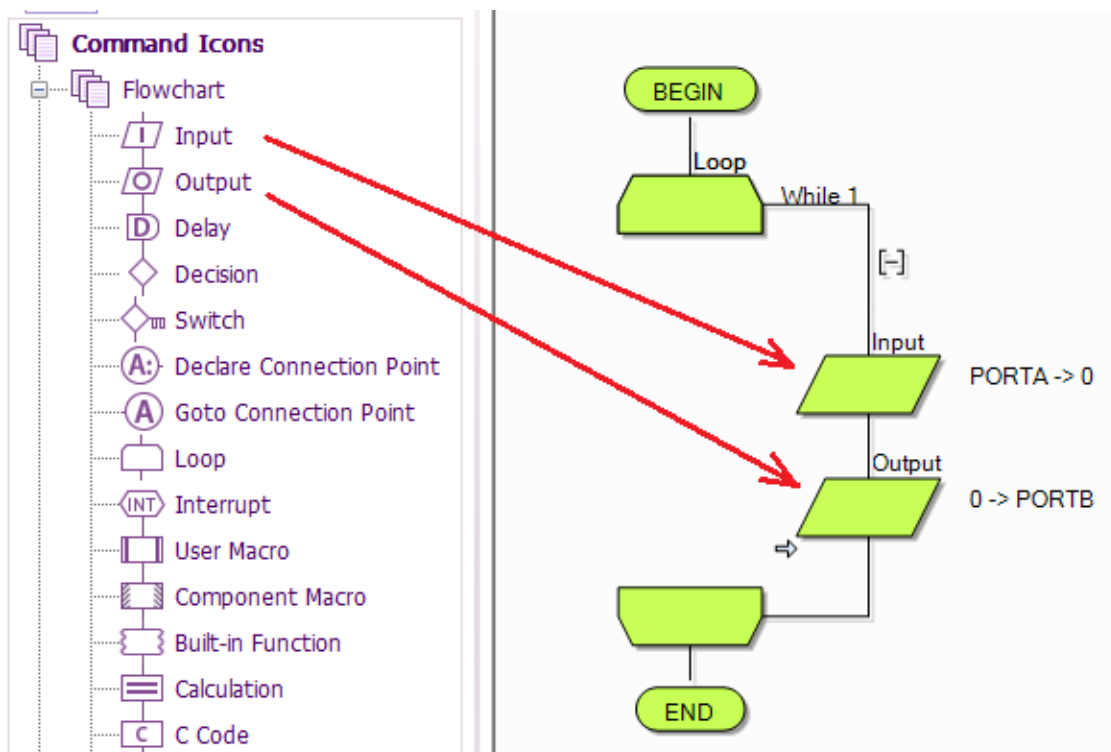
Example 1: Adding digital inputs - Where's the fire? .....

Creating the variables

Right-click on the Input icon, and select 'Properties' from the menu. The Input Properties dialogue box appears, shown opposite.

This allows us to add a 'variable'. But what is a variable?

A variable is a place where we can store information, in particular, information that changes as the program runs.

In this case, the variable is the number of the heat sensor that triggers the alarm.

It might be sensor 1 that goes off, or sensor 5…. .
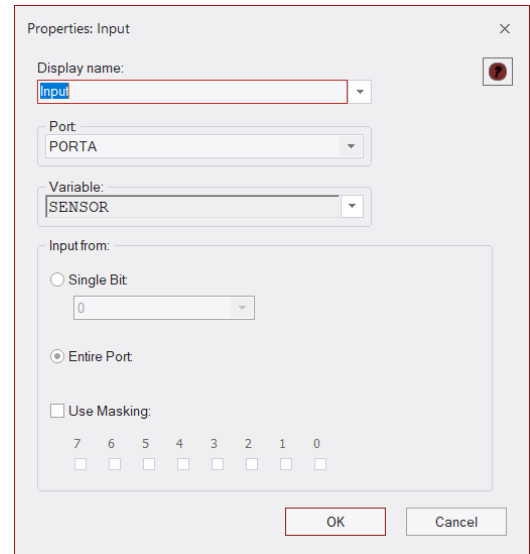
We are going to use a variable called **SENSOR** to store the information on which sensor has ▼ been triggered.

Click on the arrow next to the 'Variable:' box.

You will see the next dialogue box:

Hover over the word 'Variables' and an arrow appears.

Click on it and select 'Add new'.
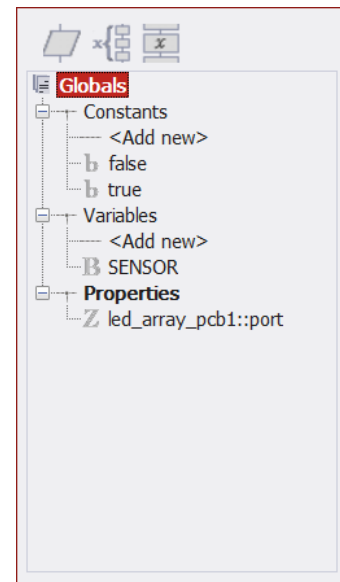
Another dialogue box, shown opposite, appears, offering a large choice of variable types. For now, accept the default type of 'Byte', a variable which can store numbers from '0' to '255'.

Type the name "SENSOR" (without quotation marks) as the name of the new variable and click on the 'OK' button. It now appears in the list of variables that the flowchart can use.

Double-click on the name of the variable to use it, or alternatively click and drag the name into the variable box.

You now see the input 'Properties' box again. Notice that you need to tell the system which port you are going to use to input the data the system needs. It is set to port A at the moment, and we are going to leave it that way.

In this case, the system needs to monitor the heat sensors and so each sensor will be connected to a different bit of port A.

Click on 'OK' to close the Input Properties box.
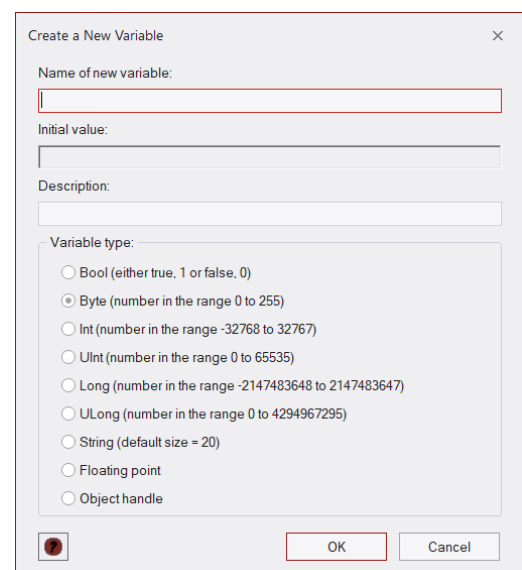
Worksheet 4 - Flowcode examples

Example 1: Adding digital inputs - Where's the fire? .....

More on variables
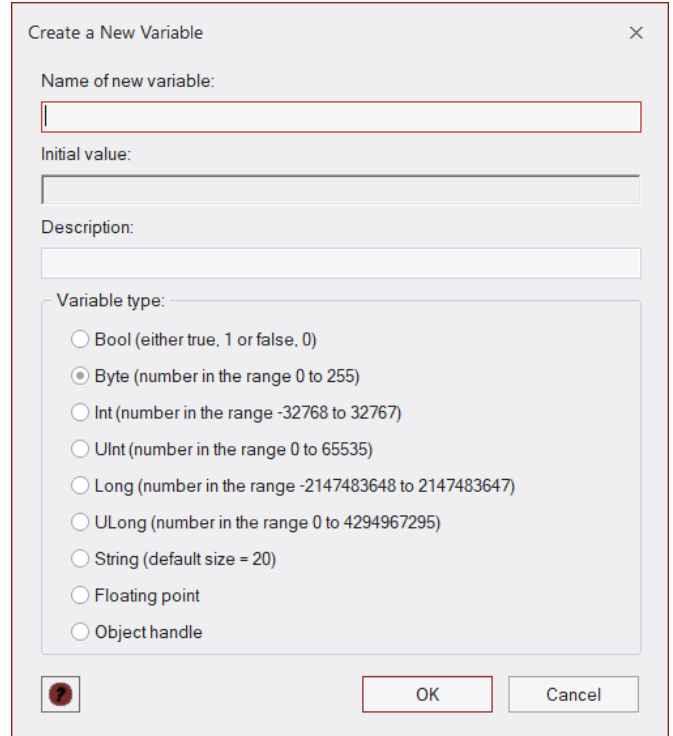
In the previous section you added a variable to the program using the variable dialogue box:

Computer signals consist of streams of binary '0's and '1's along each wire. A group of eight wires can carry eight 'bits', (binary digits,) simultaneously.

This grouping of eight bits, known as a 'byte', is used for much of the internal wiring inside microcontrollers and for the registers that hold and process data.

It is also used within memory subsystems. The contents of a memory register having eight bits can vary from '0' to '255'.

A variable inside Flowcodecan be configured to use just one memory register or more than one.

Flowcode variables:

Flowcode offers eight different types of variables:

- a 'Bool' (Boolean) variable, which can be either '1' or '0' (true or false);
- a single register, known as a 'Byte' variable, which can store numbers from '0' to '255';
- a double register, known as an 'Int' variable, which can store numbers from '-32768' to '+32767';
- the double register can also be unsigned, then known as a 'UInt' variable, which can store numbers from '0' to '65535';
- a quad register, known as a 'Long' variable, which can store numbers from '-2147483648' to '2147483647';
- the quad register can also be unsigned, then known as a 'ULong' variable, which can store numbers from '0' to '4294967295'.

TIP: Use a 'Byte' variable for simple counters and for variables that will not go above the value '255'. It is the most economical in terms of memory space and also the fastest. Mathematical processes involving two bytes (often referred to as '16 bit arithmetic') take longer to execute. A multiple register, known as a 'String' variable, can consist of a number of 'Byte' variables - the default in Flowcode is 20.

Other variable issues:

Floating point numbers, (that contain a decimal point somewhere in them,) can also be used, although they represent a much wider range of values than an integer. They suffer a loss of accuracy over large ranges.

Finally an 'object handle' is used to reference a more complicated piece of data (such as a file, component or a block of text) whose internal format is not known.

Why worry?:

The number of registers inside a microcontroller is limited, and in larger applications the number and types of variables must be managed carefully to ensure that there are enough. On downloading a program, the variables in Flowcode are implemented in the Random Access Memory (RAM) part of the PIC device. In the 16F1937 there are 512 Bytes of memory. This means you can have 512 'Byte' variables, 265 'Int' variables or 25 'Strings' each consisting of twenty 'Bytes' or characters.

Worksheet 4 - Flowcode examples

Example 1: Adding digital inputs - Where's the fire? .....

Setting up the outputs

- Next, right-click on the Output icon, and select 'Properties' (or just double-   click on it. The Output Properties box appears.

- Click on the arrow,   , next to the 'Variable:' box. You will see the 'SENSOR' variable listed.

- Double-click on the word 'SENSOR' or click and drag it to the 'Variable:' box. The Output Properties box now shows that the system is set to output whatever data is stored in the 'SENSOR' variable.

- Change the port used to Port B, (following the instructions in worksheet 2.)

- Click on 'OK' to close the Output Properties box.

- The flowchart should now look like the one opposite:

  Notice the arrows in the icon annotations. They show that information will flow from Port A into the flowchart, via 'SENSOR', (Input icon) and from the flowchart, via 'SENSOR', out to Port B (Output icon).
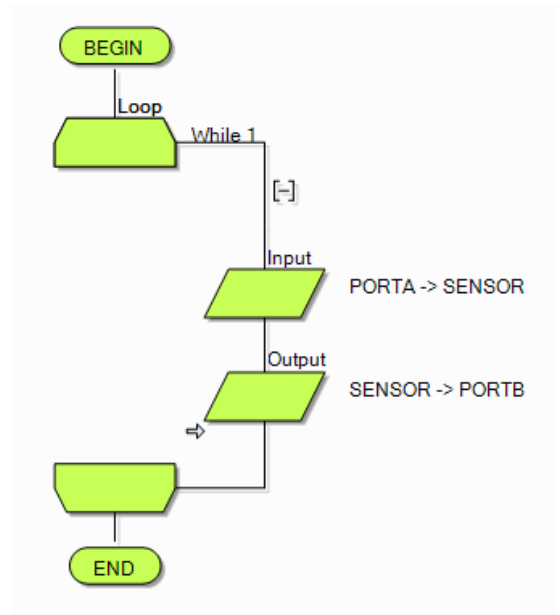
Adding the LEDs

- Now click on the 'Outputs' button and select the LED Array icon. 'Click-and-drag' it onto the System Panel.

- Click on the box next to the 'Count' property and use the keyboard to change the 'Count' property under the 'Simulation' section to value '5'.

- Click next to 'Port' under the 'Connections' section to open an interactive view of the chip, showing the compatible pins.

- Click on the drop-down menu and select the 'PORT B' option. You have now connected the LEDs to the pins on port B.
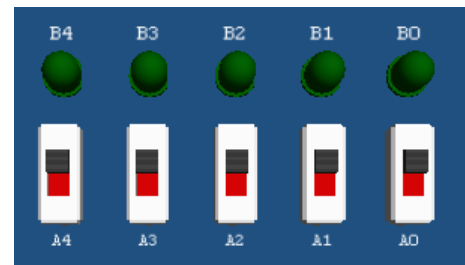
(Arduino users: please use ports C and D as appropriate).

Adding the Switches

- You are going to use switches to simulate the five heat sensors. The switch that is 'on' (closed) is the heat sensor that has triggered the fire alarm. Click on the 'Inputs' button and select the Switch Array. Drag it into a suitable spot on the System Panel.

- Click on the box next to the 'Count' property and change the value to '5'. Check that the component is connected to 'PORTA'.

Simulating the program

- Click once on the 'Step Into' button. The 'Simulation Debugger' window appears but ignore it for now.

- Move the cursor over one of the switches and click, to simulate detecting a fire. The switch graphic toggles to the closed position. Click the 'Step Into' button a few more times to simulate the complete program.

The program is finished, and working! You have just detected a fire, which turned on a heat sensor.

Worksheet 4 - Flowcode examples

AEROSPACE

Example 2 - Using loops:

Counting sheep, badly at first, but without falling asleep!

The plan is straightforward - when a sheep passes through the gate, it breaks a light beam. This sends a pulse to a counting system, which then adds one to the total stored in the system.

We display this total on the LED Array.

The plan seems straightforward - but there will be problems!

(Note that Flowcode has a 'Beam Breaker' component, based on the 'Collision Detector'. Although this would do a far better job, for now we detect the light beam interruption using more basic methods.)
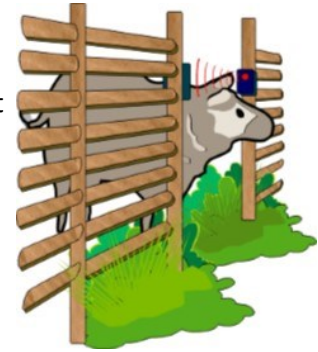
Setting up the flowchart

Launch Flowcode and start a new flowchart.

Create the flowchart shown opposite.

It contains a 'Loop' icon, an Input icon, an Output icon and a 'Calculation' icon (which you have not used before.)

(Arduino users: please use ports C and D as appropriate).
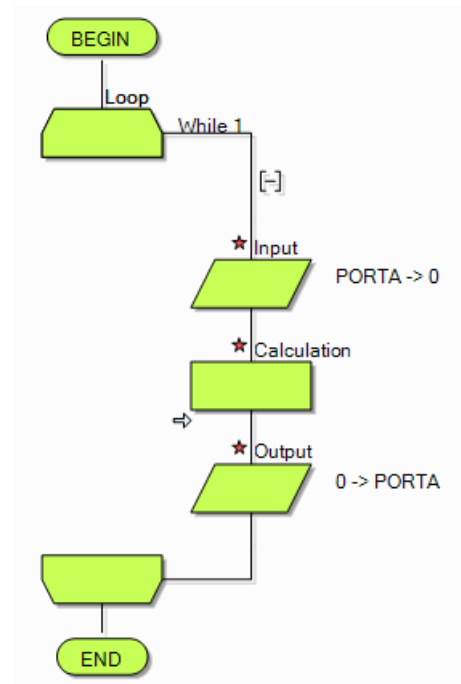
Creating the variables

We are going to create two variables, one called 'SHEEP' and the other called 'TOTAL'. The 'SHEEP' variable will show whether a sheep is present or not. The variable 'TOTAL' will store the total number of sheep recorded so far.

- Click 'View' on the menu bar, and ensure that 'Project Explorer' is checked (View > Project Explorer).
- Click on the 'Globals' button at the top of the Project Explorer panel:
- Hover over 'Variable:' in the Project Explorer panel and click 'Add new'.
- You now see the 'Create a New Variable' dialogue box. Type in the name "SHEEP" and then click on 'OK'. You can leave the variable type as 'Byte' as there will not be that many sheep!
- Create a variable named "TOTAL" in the same way.

Worksheet 4 - Flowcode examples

AEROSPACE
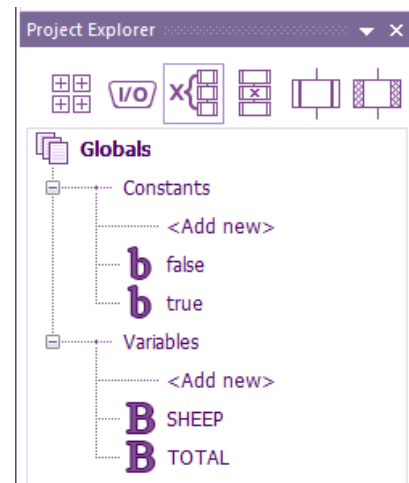
Example 2 - Using loops .....

Setting up the calculation

- Double-click on the 'Calculation' icon to open the 'Properties' dialogue box.

- Change the 'Display name' to "New total".

- Create the calculation by typing the following in the 'Calculations' window:

    TOTAL = TOTAL + SHEEP

- We will simulate breaking the light beam using a push switch marked 'SW0' on Port A bit 0.

    The 'Input' properties are set up to store whatever number appears on Port A in the variable called 'SHEEP'. Initially, that number is '0'. When the switch is pressed, the number on Port A, stored in 'SHEEP' is '1'. (With only one switch, the biggest number we can create on Port A is 1.)

- When the 'Calculation' icon is executed, the number stored in 'SHEEP' is added to the 'TOTAL' variable. Hence, when a sheep breaks the light beam, 'TOTAL' is increased by 1. With no sheep present, 'TOTAL' remains unchanged.

Click on the 'OK' button, to close the dialogue box.
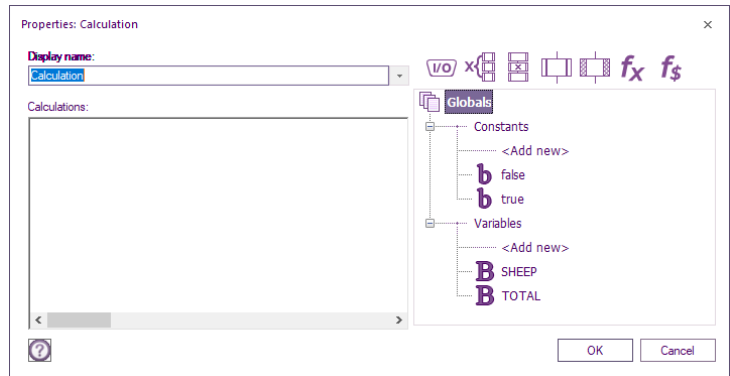
Configuring loop properties

- Double-click on the 'Loop' icon to open its 'Properties' dialogue box.

This shows the options for controlling the loop. Next to the 'Loop while:' statement is the loop control text box, where you write the loop condition - the program continues looping until this condition is met.
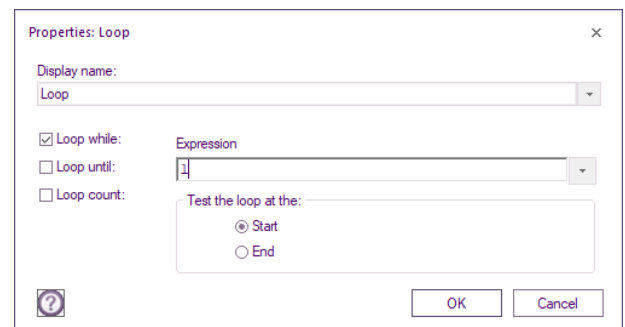
Examples of loop conditions:

- count = 10   (Loop runs as long as the variable 'count' = 10)
- count > 4   (Loop runs as long as the 'count' is greater than 4)
- count = preset (Loop runs as long as the 'count' is the same as the variable 'preset')

In all of these, looping continues as long as the condition in the 'Loop while' text box is 'true'.

In programming 'true' has a special meaning. It is assigned a numerical value of '1' so that a test can determine if something is 'true'. Similarly 'false' is assigned the numerical value '0'.

The default condition in the 'Loop while:' text box is '1' - this condition is always 'true' and so with this value, the loop will run forever. Programs usually contain a 'loop forever' structure. If they do not, the program will end suddenly and the computer will just sit there doing nothing.

When to test?

You can configure the properties to test the loop condition either at the start of the loop or at the end. Understanding this option is important. It can affect the number of times that the program will loop.

Loop for a set number of times

Sometimes, you just want to run a loop for a set number of iterations. To do this, check the 'Loop count:' box and enter the number of loops you want in the associated text box.

Worksheet 4 - Flowcode examples

AEROSPACE

Example 2 - Using loops .....

Setting up the input

- Right-click on the 'Input' icon, and select 'Properties' from the menu, to see the dialogue box shown opposite:

- Double- ▼ click on 'Input' in the 'Display name:' box and type "Check the sensor" to change the display name.

- Click on the next to the 'Variable:' box to open the 'Variable Manager'.

- Double-click on the word 'SHEEP' to insert it into the 'Variable:' box.

- By default, the input is PortA, which is the one we want. Click on 'OK' to close the dialogue box.
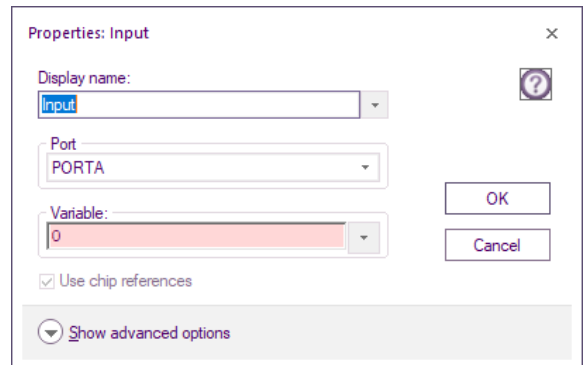
Setting up the output

- Double- ▼ click on the 'Output' icon to open the output 'Properties' dialogue box.

- Click on the next to the 'Variable:' box.

- Double-click on the word 'TOTAL' to insert it into the 'Variable:' box.

- In the output' Properties' box, change the port used to 'PORTB'.

- Click on 'OK' to close the dialogue box.

The flowchart should now look like that opposite:
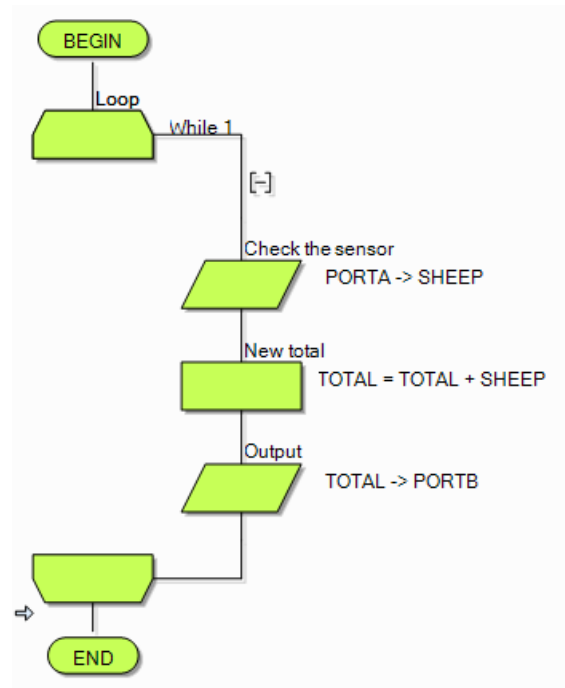
(Arduino users: please use ports C and D as appropriate).

Adding the LED Array

- Click once on the 'Outputs' box and select the 'LED Array' icon .

- Place it on the System Panel by moving the cursor over it and then 'clicking-and-dragging' it into position.

- Change the value of the 'Count' property to '8' to set the number of LEDs in the array.

- Click the 'Connections' property in the 'Properties' pane. Select 'PORTB' from the drop-down menu to connect the LEDs to the pins on Port B.

- You can change the colour of the LED Array in the 'Colors' section.

Worksheet 4 - Flowcode examples

Example 2 - Using loops .....

Adding the switch

- A single push switch will represent the light beam sensor. Select **Switch (Push,Panel)** from **Component Libraries > Inputs** and add or drag it onto the System Panel.
- On the 'Properties' pane 'Connections' section, check that the 'Connection' property for the switch is '$PORTA.0' i.e. the switch is connected to Port A bit 0.

- Select Label from **Component Libraries > Creation**
- Click on the Label property in the 'Properties' pane and replace the default text with "Light beam interruption".
- To adjust the size of the text, click on the 'Position' tab and change the values of 'Width' and 'Height' under the 'World size' section. Move the text to a suitable position next to the switch.
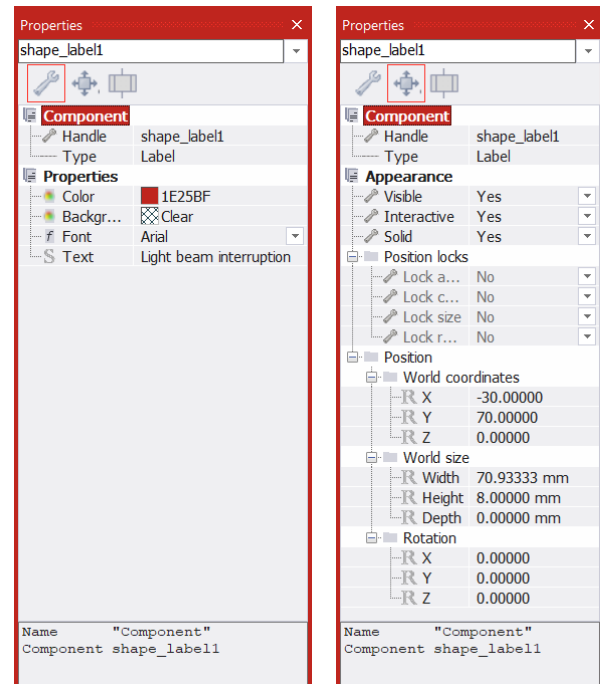
You should now have a project that looks something like this:
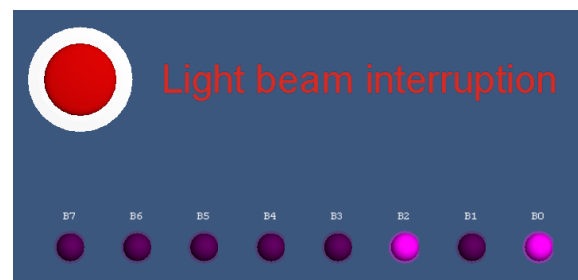
Simulating the program

- Now run the simulation by clicking on the 'Run' button   .
- The 'Simulation debugger' window appears. Close it as it is not needed.
- Move the cursor over the switch and give the briefest mouse click you can.

What happens depends on how quickly you click, and how fast the PC works!

We want only the 'B0' LED to light, to show a total of 1 sheep. The program runs at high speed, however, and so keeps cycling through the 'Input' and 'Calculation' steps. As a result, before you have time to release the push switch, the total has incremented (increased by one) several times.

This problem is explored in the next section.

Worksheet 4 - Flowcode examples

AEROSPACE

Example 2 - Using loops .....

The Solution: Adding a Delay

The problem - the program runs too fast!

Before we have time to release the switch, the program has run through several times, adding one to the total each time.

We need to slow it down by adding a delay.

- Move the cursor over the 'Delay' icon.
- Drag it onto the main work area and drop it between the Calculation and the Output icons.

The flowchart should now looks like this:
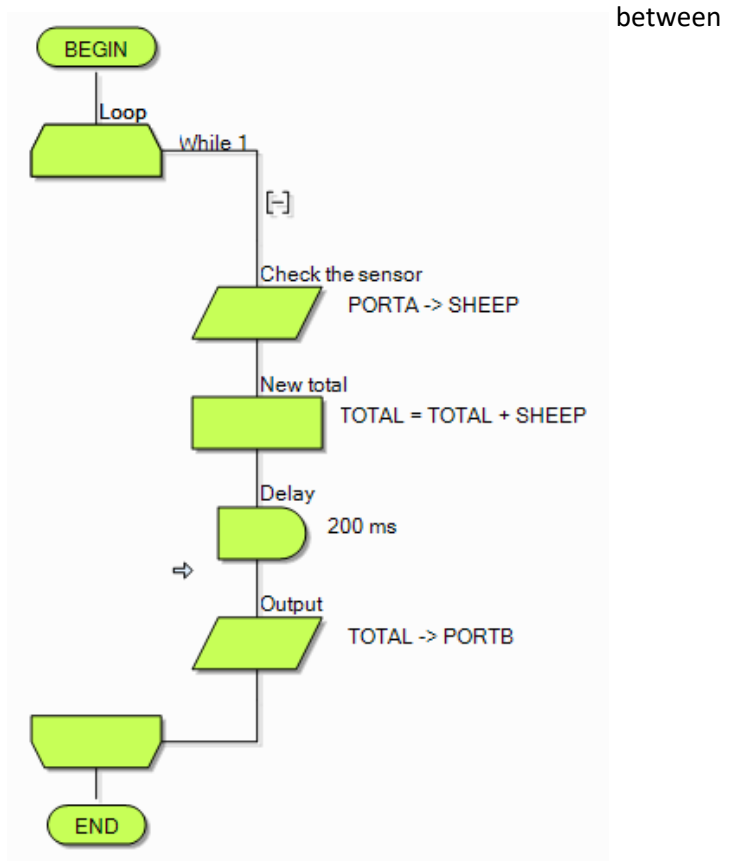
- Double-click on the 'Delay' icon to open the 'Properties' dialogue box.
- Change the value in the 'Delay value or variable:' box to '200' and then click on the 'OK' button. This causes a 200 millisecond (0.2 second) delay whenever the 'Delay' icon is activated. In other words, the system just sits there and does nothing for 0.2 seconds.

- Now run the simulation again. Providing you don't keep it pressed for too long, you should find that the LED array shows an increase of 1 each time you press the switch.
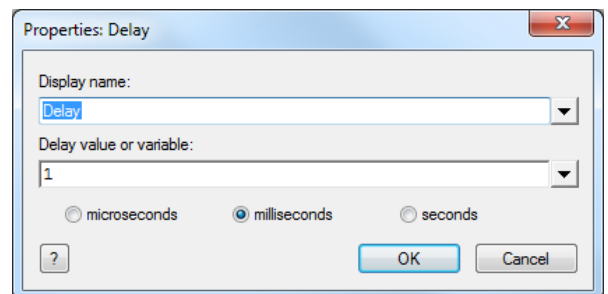
  The program now works satisfactorily, providing the sheep rush through the light beam in less than 0.2 seconds. The delay could be increased to allow for slower sheep!

  Note: This program shows the total number of sheep in binary format.

Worksheet 4 - Flowcode examples

AEROSPACE

Example 3: The LCD display

Programs using the LCD display need to use the crystal oscillator.
In Flowcode:

- select 'Build' from the main menu;
- then 'Project Options...';
- and finally the 'Configure' tab.
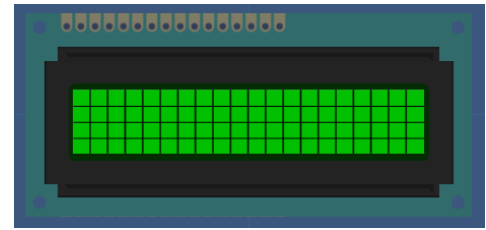
Select the crystal oscillator from the list of options **(Build > Project Options... > Configure).**

LCD displays

Flowcode comes with a number of components that add commonly used subsystems, such as the LCD display, the 7-segment display, and several analogue inputs devices.

In this worksheet, we look at the LCD display, the basic text display subsystem on a range of electronics devices, from calculators to mobile phones, which can display text or numbers on one or more rows.

In most programming languages, the LCD is one of the last things you learn, as it is quite a complicated device to program. However, Flowcode takes care of the complexities, making the LCD simple to use. The LCD display referred to here is the one used on the E-Blocks Combo board and on the LCD display - a two row, sixteen character display.

Adding the LCD component

Before you can use the LCD, you need to add a LCD component to a Flowcode panel.

- Select the **LCD (Generic, 20x4)** component from **Component Libraries > Displays** and add it to the System Panel. A LCD display mimic now appears on the panel.

- At the top of the 'Properties' pane, the 'Component' section identifies the component just selected. By default, the LCD is added to Port B. You could change this, but we will keep it on Port B.

- The LCD display displays letters and numbers conveyed as serial data on a five wire bus, and so requires five connections. The techniques involved go beyond this tutorial. Fortunately, Flowcode has some embedded routines that take care of the complexities.

- Drag a 'Component Macro' icon onto the flowchart and open up the corresponding macro dialogue box by double-clicking on it.

- Now scroll through the 'LCD' section in 'Components' and select the macro called 'Start'. This initiates the LCD, clears the display and gets it ready for action. We examine more LCD macros in the next couple of sections, but for now scroll through the available macros and take a quick look at each.

AEROSPACE
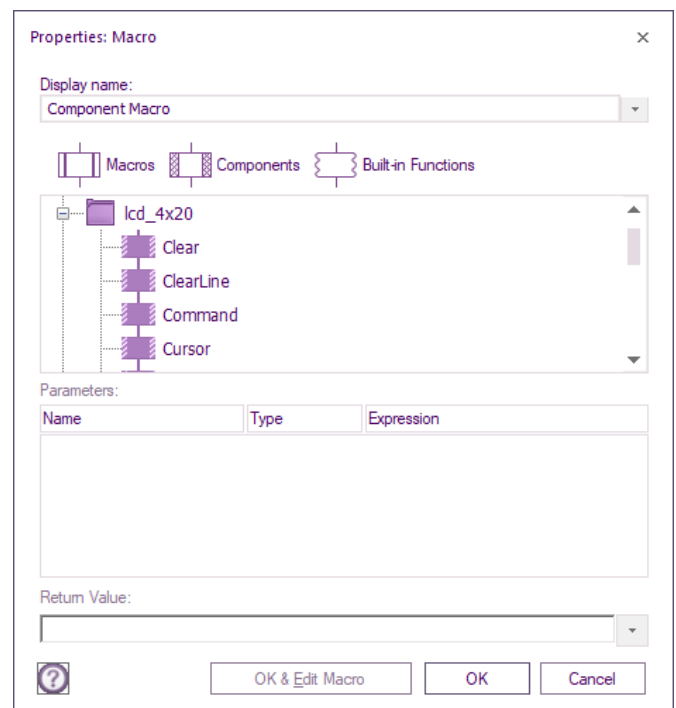
Example 3: The LCD display .....

Writing Messages

To display text on the LCD, simply type it in!

- Add another 'Component Macro' to the flowchart and open the macro dialogue box.

- Select the LCD macro called 'PrintString'. This requires a single parameter (item of data), 'Text', - the text to be printed.

- Type the text into the parameter box surrounded by quotation marks, e.g. "Hello World"
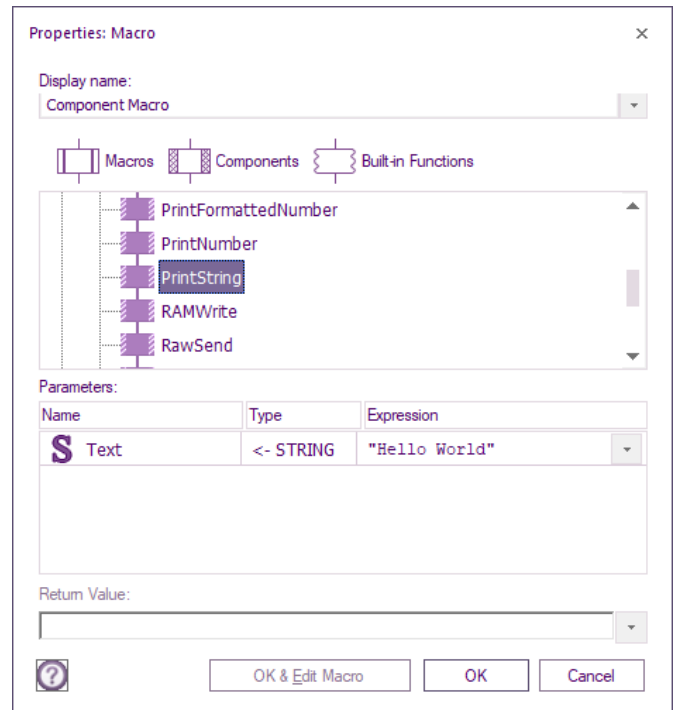
- Run the program and the text will be sent to the LCD display.
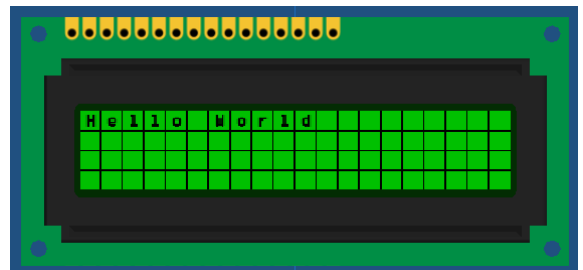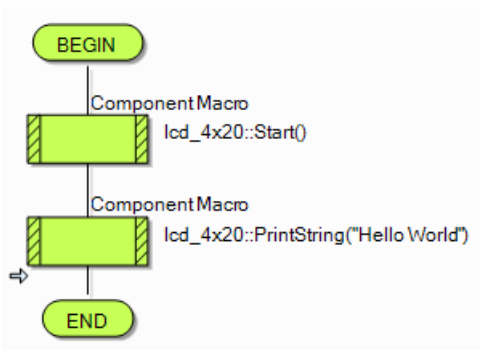
Other LCD functions

There are a number of other useful functions in the LCD macro list:

- 'Clear'   - Clears the display and resets the cursor position, (where the display prints next,) to '0,0' i.e. top left.

- 'Cursor'   - Moves the cursor to the specified location. The two parameters, 'X' and 'Y' select the horizontal and vertical positions of the cell respectively. '0,0' is the top left cell, '0,1' the first cell on the second line, '3,2' the fourth cell on the third line … .

- 'PrintNumber'   - Works like 'PrintString' but prints a number instead of a string. It can be used with variables, or with actual numbers.

Example 3: The LCD display .....

Using PrintNumber - an example:

Altogether we will add four **Component Macros** to the flowchart:

- To the first **Component Macro** add '**Start'**.
- To the second select '**PrintString'** and add **"Hello World"** (with quotation marks).
- To the third select '**Cursor'** and add **0,1** to the parameters.
- To the fourth select '**PrintNumber'** with the parameter value as **123.**
- Click 'Run' to simulate the program.
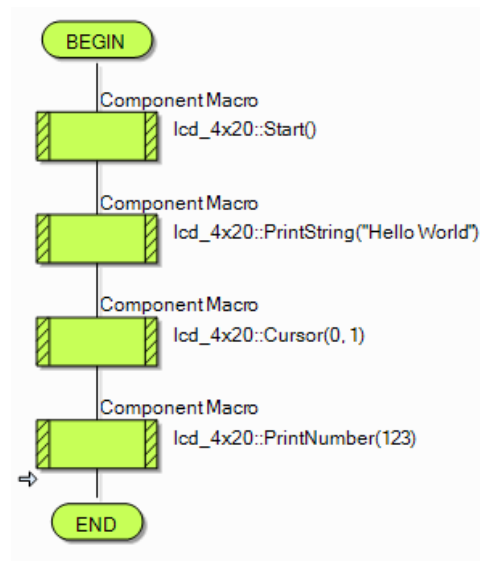
You should see results similar to those shown below:
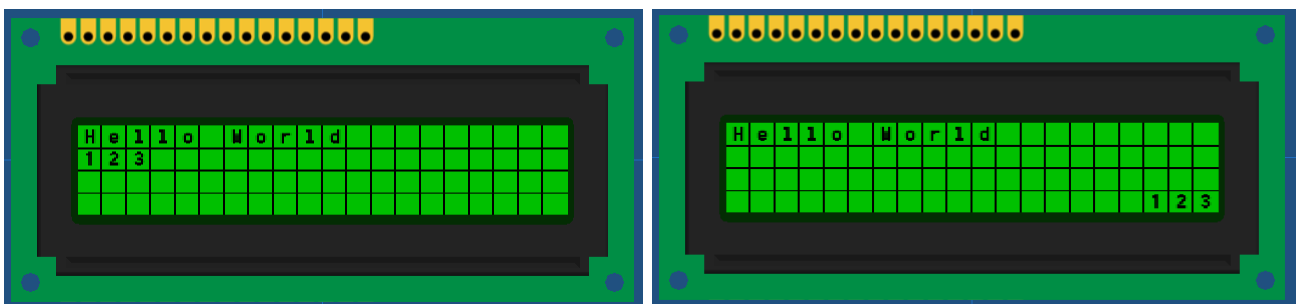
TIP: Try changing the '**Cursor'** parameters and see where the numbers print.

The 'y' value needs to be between 0 and 3.

The 'x' value needs to be between 0 and 19, (between 3 and 17 to see all three figures 1 and 2 and 3).

AEROSPACE
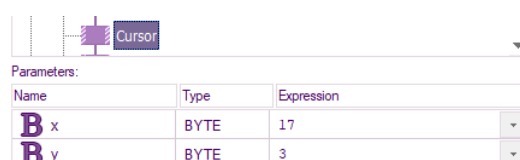
Example 4: A stopwatch

This example uses example 3 (Using PrintNumber) as a starting point.

- Expand the '**Using PrintNumber'** program by dragging a **Loop** icon below the **PrintString Component Macro**.

- Change the text in the 'PrintString' Component Macro to "Hundredths:" (with quotation marks).

- Drag a 'Calculation' icon into the loop.

- Create a variable called 'Count' as an 'Int' type with initial value 0.

- Double-click on the 'Calculation' icon and type "Count = Count + 1" in the 'Calculations:' text box. This will add 1 to the value of variable count every time the icon is executed.

- Next drag another 'Component Macro' into the Loop.

- Double-click this 'Component Macro' and find 'Cursor' under the 'LCD' macros.

- Enter '0,1' as parameters, to position the cursor on the first character of the second line.

- Next, drag a third 'Component Macro' into the Loop.

- Select 'PrintNumber' and enter 'Count' as the parameter.

- Now, add a 'Delay' icon to the flowchart and set the delay to 10ms (which equals one hundredth of a second).

- Refine the program by clicking on each icon and entering comments on what the icon does.
  This may seem to be a lot of effort, but it saves time later as your program will be easier to follow.

- Run the program. You have now made a counter that will count (approximately) the time elapsed in hundredths of seconds.
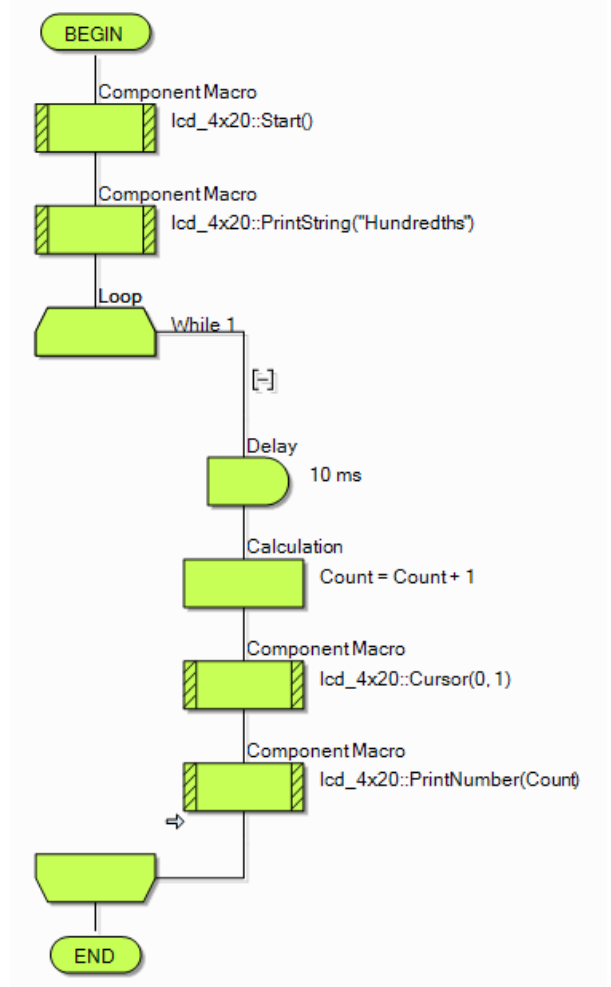
TIP: You can refine the program by clicking on each icon and entering comments to describe what the icon does.
It may seem like a lot of effort, but it can help with more complex programs.

Worksheet 4 - Flowcode examples

Example 5 - Using binary numbers - A binary adder

In this section you build a system that makes the microcontroller add together two numbers.

The simplest way to input a binary number is to use a set of switches attached to the input port.

To input two numbers, we need two sets of switches and two input ports.

To see the result of the calculation, we will use a LED Array, connected to the output port.

We need a PIC chip with three ports!
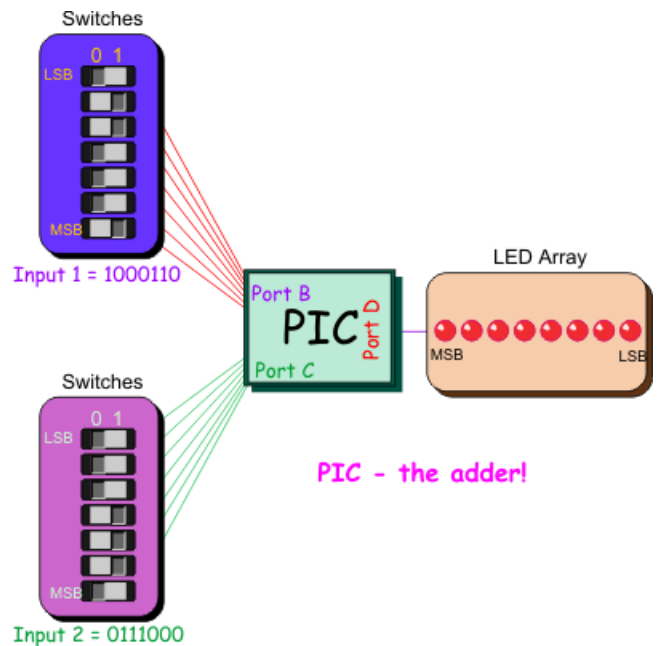
Setting up the Flowchart

- Launch 'Flowcode' and start a new flowchart.
- We need a PIC with at least three ports.
- Pull down the slide bar to find the 16F1937 PIC.
- Click on it to select it and then click on 'OK'.
- Click-and-drag a Loop' icon between the 'BEGIN' and 'END' boxes.
- Click-and-drag an 'Input' icon and drop it between the ends of the loop.
- Click and drag a second 'Input' icon and drop it in between the ends of the loop.
- Click and drag an 'Output' icon and drop it just below the 'Input' boxes.
- Click and drag a 'Calculation' icon and place it in between the second 'Input' icon and the 'Output' icon.
- Your flowchart should now look like this (though this one has had descriptions and variables added):

Worksheet 4 - Flowcode examples

AEROSPACE
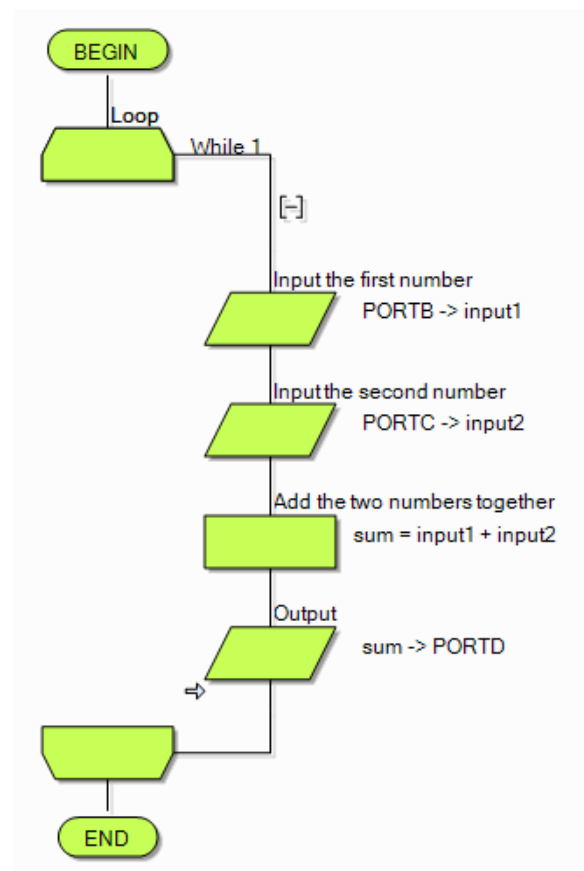
Example 5 - Using binary numbers - A binary adder .....

Creating the variables

- Click 'View' on the menu bar and ensure that 'Project Explorer' is checked (View > Project Explorer).

- Click on the 'Globals' button at the top of the Project Explorer panel.

- We are going to create three variables, called 'input1', 'input2' and 'sum'. The first two store the numbers fed in from the switches. The variable 'sum' stores the result of adding them .

- Hover over 'Variables' in the 'Project Explorer' panel then click on the that appears.

- Click 'Add new' and the 'Create a New Variable' dialogue box appears.

- Type in the name "input1", and click on the 'OK' button - leave the variable type as 'Byte'.

- Create variables, 'input2' and 'sum' in the same way.
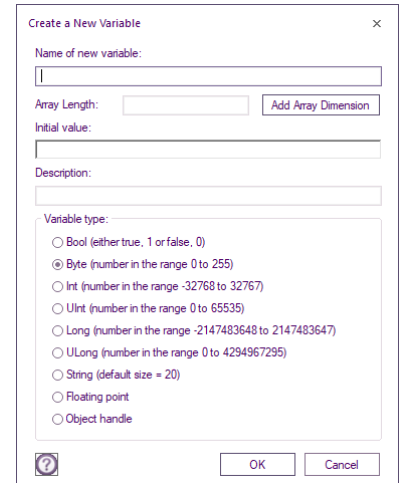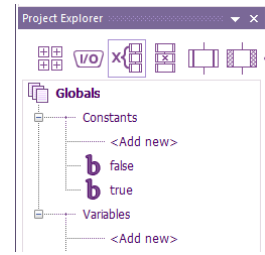
Setting up the inputs

- Right-click on the top 'Input' icon, and select 'Properties'. The 'Properties: Input' dialogue box appears.

- Double-click on the word 'Input' in the 'Display name:' box to highlight it.

- Type "Input the first number" to replace it. This will appear alongside the 'Input' icon in the flowchart.
  (Adding labels like this helps users to understand what is happening.)

- Click on the arrows next to the variable box to open the 'Variable Manager'. This lists the three variables that you just created.

- Double-click on 'input1' to use this variable in the input box.

- Back in the 'Input Properties' dialogue box, click on the down arrow at the end of the port window, and select 'PORTB' to replace 'PORTA'.

- Click on 'OK' to close the dialogue box.

- Double-click on the second 'Input' icon. (a quicker way to open the 'Properties' box.)

-
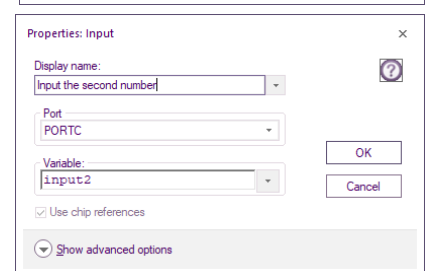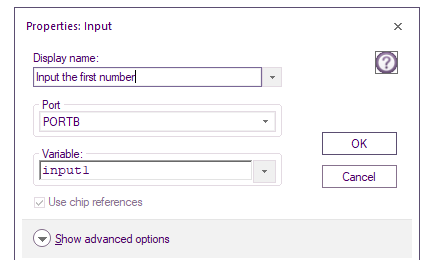
Configure this input to:

- display the label 'Input the second number';

- use the variable 'input2';

- use 'PORTC'.

- Then close the dialogue box by clicking the 'OK' button.
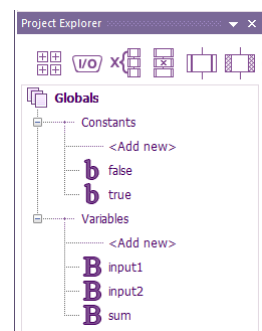
Arduino users: These two Ports will need to be set as follows:

Input 1 set to PORTC (to use Port A switches on the Combo board).

Input 2 set to PORTD (to use Port B switches on the Combo board).

Worksheet 4 - Flowcode examples

AEROSPACE

Example 5 - Using binary numbers - A binary adder .....

Set up the calculation

- Double-click on the 'Calculation' icon to open the Properties dialogue box.
- Change the 'Display name:' to 'Add the two numbers together'.
- In the 'Calculations:' box insert:

  sum = input1 + input2

  (Either type this directly, or drag in variables from the right window and insert '=' and '+' signs!)
- Then click on the 'OK' button, to close the dialogue box.
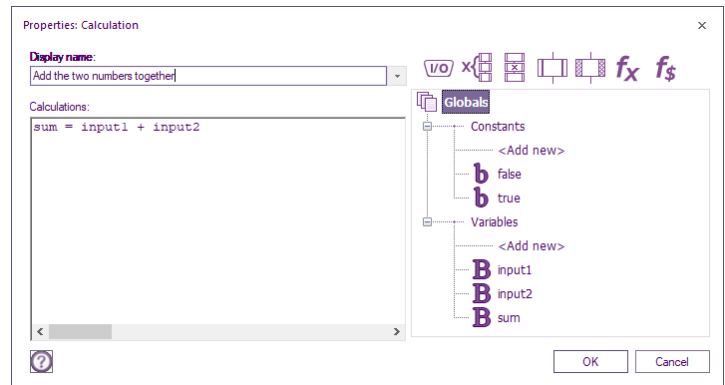
Setting up the output

- Double-click on the 'Output' icon, to open the output 'Properties' dialogue box.
- Click on the arrow next to the 'Variable:' box.
- Double-click on 'sum' to insert it in the box.
- Change the port used to 'PORTD'.   *(For Arduino PORTB)*
- Click on 'OK' to close the dialogue box.
- The flowchart should now look like the one opposite:
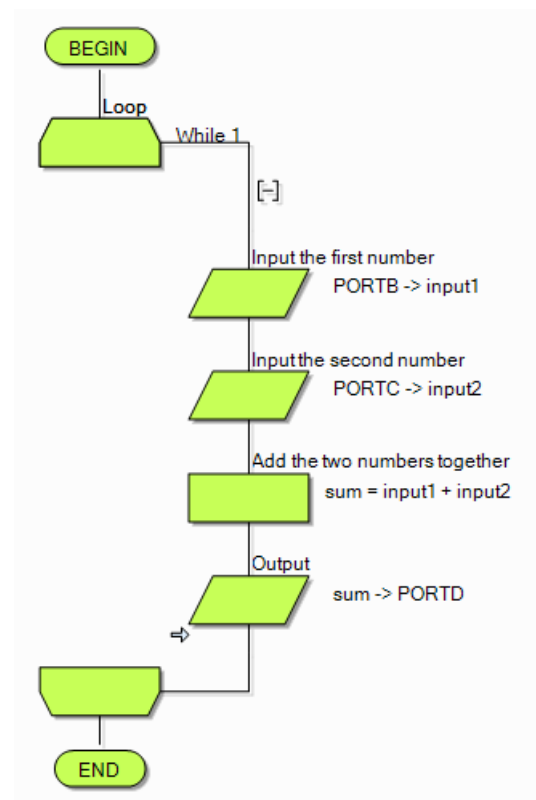
Adding a LED Array

- Click on the 'Outputs' tab and select 'LED Array' .
- Place it in the middle of the System Panel by moving the cursor over the component and then clicking-and-dragging it into position, (or right-clicking it and selecting 'Center all objects').
- Click next to the 'Count' property under the 'Simulation' section on the Properties pane and change the number of LEDs to seven.
- Click next to the 'Port' property and select 'PORTD' from the drop-down menu to connect the LEDs to the pins on port D.   *(For Arduino PORTB)*
- Change the colour of the LED Array to red (0000FF) by changing the 'LED 0' property while the 'Same Color' property is set to 'Yes'.

Worksheet 4 - Flowcode examples

AEROSPACE

Example 5 - Using binary numbers - A binary adder .....

Adding the Switches

Two sets of switches are used, one for each binary number.

The output port has only eight bits. The biggest number it can output is 1111 1111, (= 255 in decimal). We will limit ourselves to inputs no bigger than seven bits meaning that the biggest input we can have is 111 1111, (= 127 in decimal). Bigger numbers would overflow the capacity of the output.

- Click on the 'Inputs' tab, select 'Switch Array' and drag it onto the System Panel   above the LED Array.
- Open the 'Properties' pane   for the switch array and connect it to Port B, using the   next to the 'Port' property to open the drop down menu.
  (Arduino users: Use PORTC)
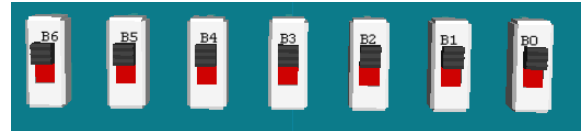
- Add a second 'Switch Array' to the System Panel in the same way. Position it under the 'LED Array' and connect it to 'PORTC'. *(Arduino users: Use PORTD)*
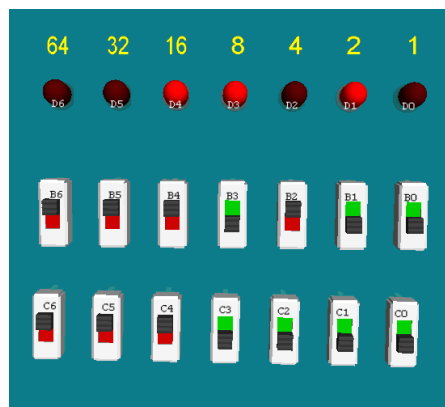
Slow Simulation

As described earlier, Flowcode allows you to progress through the flowchart one step/icon at a time, to see the effect of each on the variables and on the output.

There are three ways to simulate the program step-by-step:

- Click on **Go** on the Debug toolbar and on the **Step Into** button **(Debug > Step Into)**
- Press the F8 function key on the keyboard.
- Click on the 'Step Into' button on the main toolbar in the simulation section.

  Do one of these!

Several things happen:

- a red rectangle appears around the 'BEGIN' icon, showing that this is the current step;
- the 'Simulation debugger' window appears - containing 'Variables' and 'Macro Calls';
- the 'Variables' section lists the three variables that you defined for this program, and shows their current values - all zero at the moment.

  Ignore the 'Macro Calls' section for the moment.

Worksheet 4 - Flowcode examples

![AEROSPACE logo]

Example 5 - Using binary numbers - A binary adder .....

Testing the program

Now set up two numbers on the switch components.

- Move the cursor over the switch box connected to Port B.

- Click on switches B0, B1, and B3, to activate them.

- The switches now look like this:
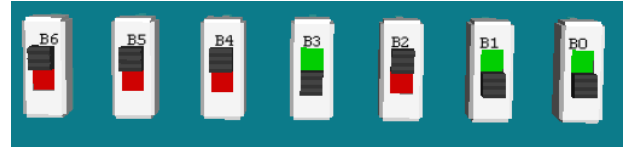
  Switch 'B6' gives the most significant bit and 'B0' the least significant bit, so you have set up the binary number 000 1011 (= eleven in decimal.)

- Set up the number 000 1111 (fifteen) on the switches connected to port C.
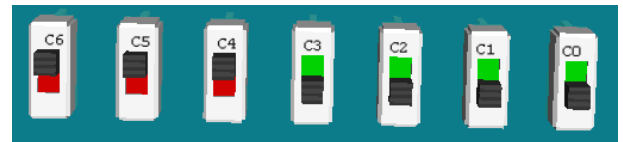
- Now 'Step Into' to the next icon in the program by, for example, pressing F8 once more.

- The red rectangle moves on to the next icon, the 'Loop' icon, but little else happens.

- Press F8 once again. The red rectangle moves on to the first 'Input' icon.

- Press F8 again and the 'Variables' box shows that the 'input1' variable now contains eleven - the result of the input instruction just carried out.

- Press F8 again and the 'Variables' section shows that 'input2' now contains fifteen.

- Press F8 again and the calculation is carried out. The 'sum' variable stores the result.

- Press F8 again. The value stored in 'sum' is transferred to the LEDarray.

The result looks like:

Reading from the most significant bit ('D6') to the least significant bit ('D0'), the LED array shows the number 0001 1010. In decimal, this is the number 26. No surprises there then!

Try different numbers!

Repeat the same procedure using different numbers and step through the program to check what the sum of the numbers is.

TIP: Explore adding graphics to your binary calculator to make it easier to read. **Component Libraries > Creation** to add digits above your LEDs.

Worssheet 4 - Flowcode examples

AEROSPACE

Example 6. Binary logic in control.

Electronic systems can make decisions. They rely on specific combinations of circumstances in order to take some particular action. Very often, these are of the form "If this AND this is true, then..." or "If this OR this is true, then...". These are examples of using binary logic.

The answer to the "If…" question is either "Yes" / "No", or "True" / "False", i.e. one of two possibilities (so a binary solution). This could be expressed as a logic 0 or a logic 1 and electronically by a high voltage or a low voltage. We can program Flowcode to make exactly the same decisions.

6A. Controlling a microwave oven

For safety reasons, a microwave oven has a door sensor to make sure that it will not operate if the door is open. In other words, the generator operates if the door is closed AND one of the heating control switches is pressed. We can build this condition into a Flowcode program.

Setting up the flowchart

- Launch Flowcode with a new flowchart.
- Create the flowchart shown on the next page, using a loop icon, two input icons, three output icons, two decision icons, two calculation icons and a delay icon.
- Create four variables:
    - 'door' (to store the state of the door switch).
    - 'control' (to store the state of the on/off control switch)
    - 'output' (to control whether the microwave switches on or not)
    - 'count' (to monitor how many times the 1s delay has occurred.
      Give it an initial value of ten, so that the microwave oven will operate for 9s).
    - Use the default configuration for the loop icon.
- Configure one input icon to store the state of the door switch (on Port A bit 0) in the variable 'door'.
- Configure the other input icon to store the state of the control switch (on Port A bit 1) in the variable 'control'.
- The upper calculation icon checks to see whether the door AND the control switch have been pressed.
- Configure it using the equation output = control & door. (The & signifies the AND operation.)
  The result of this operation (0 or 1) is stored in the variable 'output'.
- The upper decision icon checks the value stored in 'output'. (If output? is shorthand for If output=1?)
  Configure this decision icon.
- When the result of the calculation is 0 the program follows the 'No' route from the decision icon and the left-hand output icon is executed. This sends a logic 0 to the LED, ensuring that it (and the microwave generator) is switched off.
- When the result of the calculation is 1, the program follows the 'Yes' route. The 'Turn on' output icon sends a logic 1 to the LED turning it on. Configure both of these output icons.
- The lower calculation icon reduces the number stored in the variable 'count' by one.
  Configure it using the equation count = count - 1
  The initial value of 'count' is ten. Provided the number stored in 'count' has not reached zero, the program follows the 'No' route. Eventually, after looping enough times, the number stored does reduce to zero. The program then follows the 'Yes' route and executes the 'Turn off' output icon, which is configured in the same way as the other 'Turn off' icon, to switch off the microwave generator.

Worksheet 4 - Flowcode examples

AEROSPACE

Example 6. Binary logic in control .....

- Add a switch array to the System Panel and configure it to have two switches, one connected to Port A, bit 0 and the other to Port A, bit 1.
- Add an LED connected to port B, bit 0 to represent the microwave generator.
- Add labels to the System Panel to identify the components. Position them using the 'World' coordinates under the 'Position' tab of the label properties.
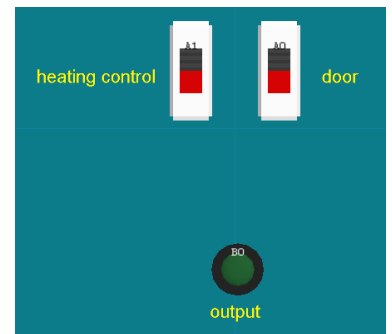
- Now simulate the program step-by-step, using the F8 function key repeatedly.
- Check what happens for different combinations of switch states and interpret this in terms of the behaviour of the microwave oven.

  What happens, for example, if the door is opened while the microwave generator is operating?
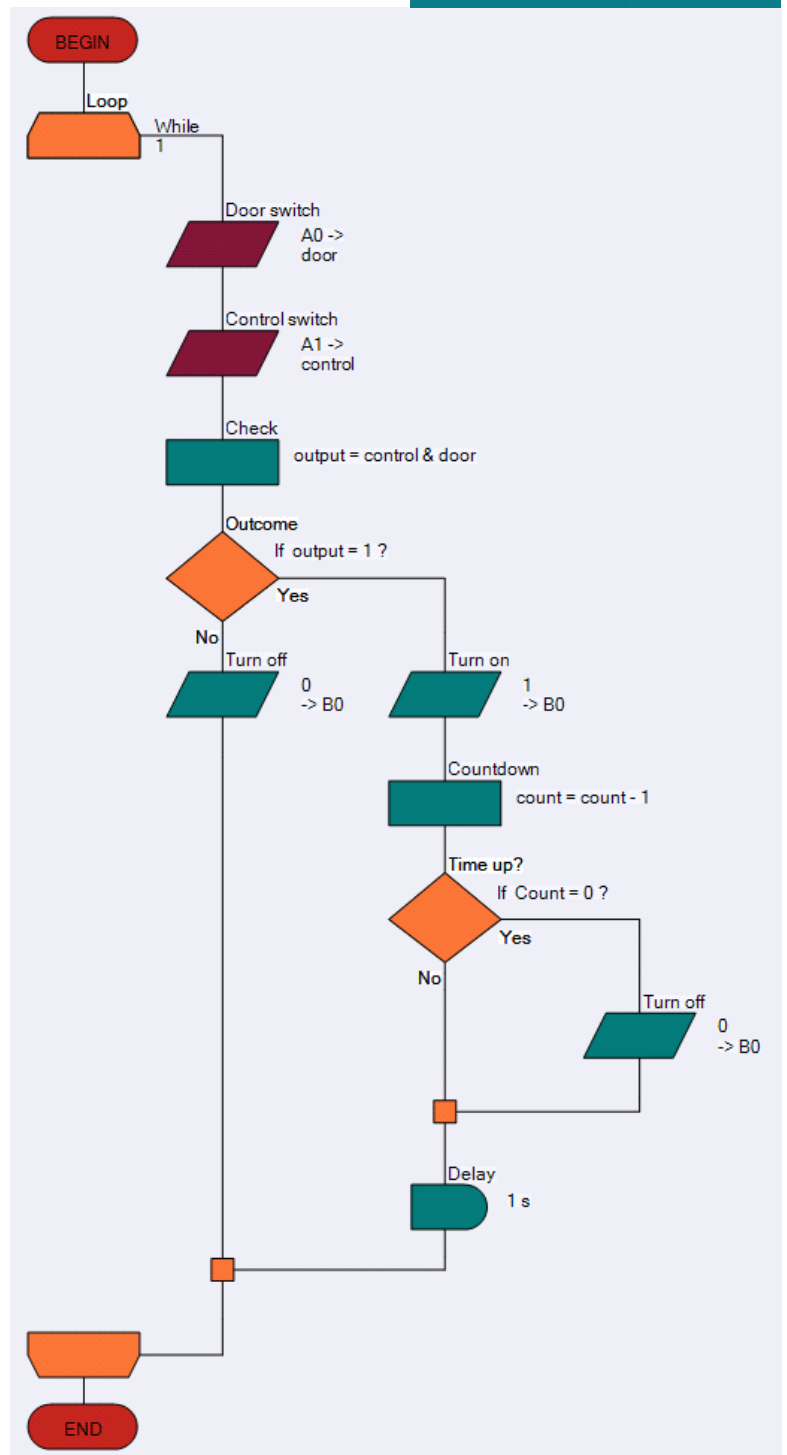
For Arduino the Ports need to be set to PORTC and PORTD (equivalent to A and B on the Combo board).

Worship 5 - Sensors and actuators

Only a small number of additional electronic components are needed to make up microcontroller circuits, such as switches, indicators, sensors, displays and actuators.

The huge variety in the functionality of these circuits is dictated by the software used with them.

In this section you will use the same basic hardware to create six very different systems using Flowcode.

The photograph shows a turboprop aeroplane cockpit.
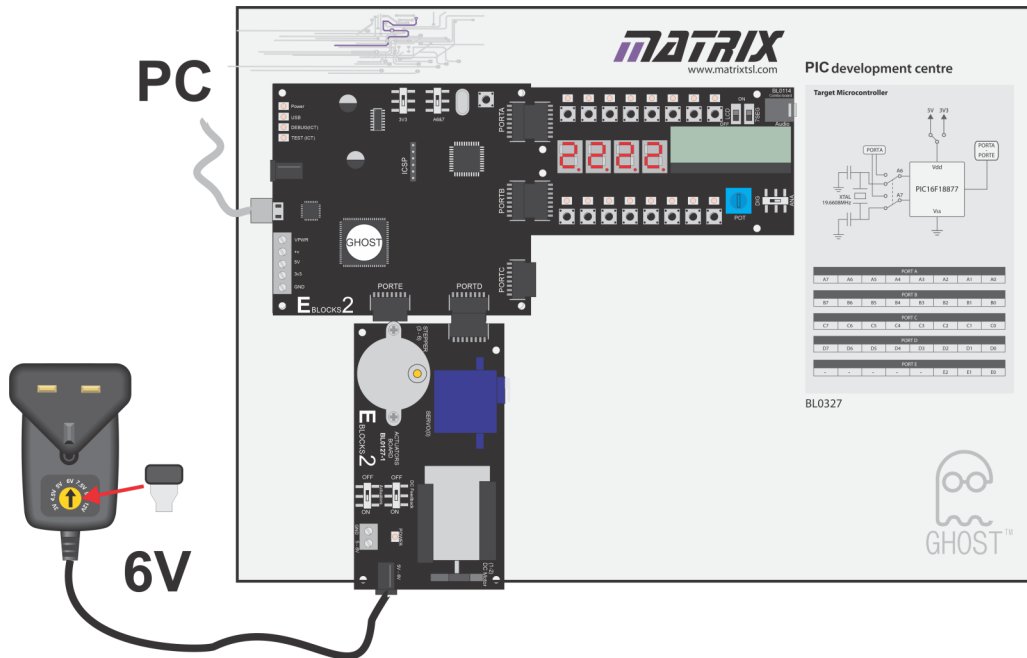
Over to you:

- Add the actuators panel to your hardware as shown in the diagram above. You will need a 6V power supply for the actuators panel.

- Familiarise yourself with the circuit diagram of the actuators panel, given in the E-blocks II datasheet. Rather than entering the programs into Flowcode as you did before, in this worksheet you just download pre-written programs into Flowcode, product code AV4234, look at how they work and take measurements with a multimeter and oscilloscope.

- First of all, download example 7 'A to D conversion and sensors'. The LCD shows the 'value' of both the light sensor and the potentiometer sensor input in the range 0 - 255.

- Use a multimeter to measure the voltage at the wiper of the potentiometer and complete the table below by adding the digital reading for each voltage.

	•0V	•1V	•2V	•3V	•4V	•5V
Digital value:						

Worapsheet 5 - Sensors and actuators

- Download Example 8 'PWM motor control'.
  In this program you can use the potentiometer on the Combo board to vary the speed of the motor.

- Use an oscilloscope to monitor the voltage at the motor terminals.

- Make sure you understand PWM and how it is used in a single output to control the speed.

- Using the E-blocks II datasheet, sketch the circuit diagram including:

  - the potentiometer,
  - the display,
  - the motor driver chip,
  - the motor,
  - the microcontroller, (including pin numbers),
  - the crystal,
  - power connections,
  - protection circuitry.

- Write a short description of how the program works - you can look at the program in Flowcode to help you.

- Download Example 9 'Stepper motor control'.
  In this program you can use the potentiometer on the Combo board to vary the speed of the motor.

- Using the E-blocks II datasheet, sketch the circuit diagram including:

  - the potentiometer,
  - the display,
  - the motor driver chip,
  - the microcontroller, (including pin numbers),
  - the crystal,
  - power connections
  - protection circuitry.

- Write a short description of how the program works - you can look at the program in Flowcode to help you.

Worksheet 6 - Multiplexed systems

Microcontrollers are not electrically or mechanically rugged. To make them so, engineers add protection circuitry and encase them in metal canisters with rugged connectors.

In aircraft and cars these are called Electronic Control Units - ECUs. These interconnect using a bus to form a distributed control system for aircraft. The two main bus types used in aircraft are ARINC and the CAN bus. Here we look at CAN.

The photograph shows an aircraft ECU.
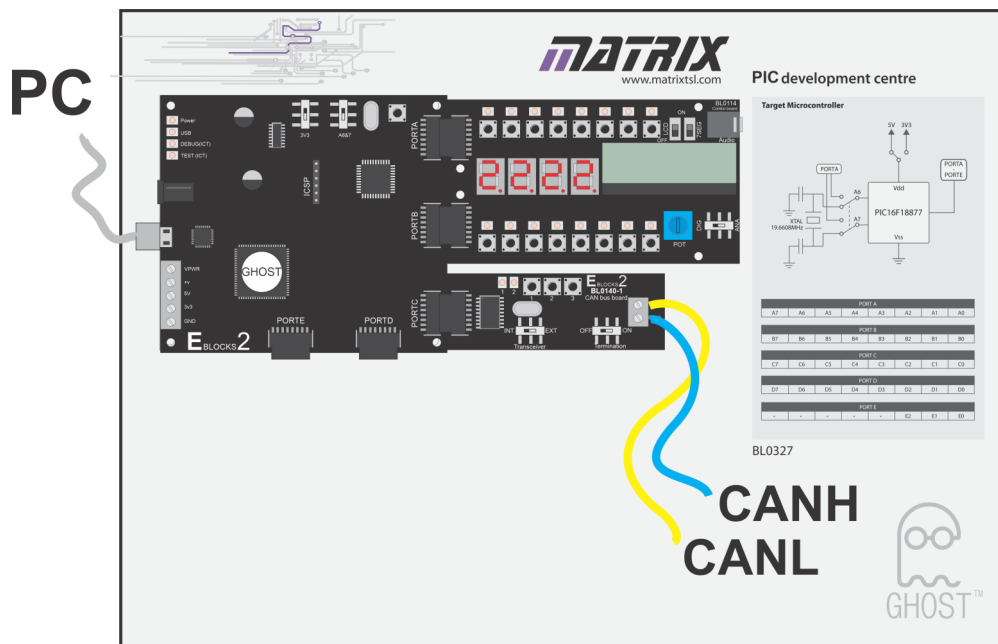
Over to you:

- Connect the E-blocks CAN bus board into the CAN bus as shown in the diagram above.
- Navigate to a screen on the EFIS module that shows the compass as a graphical object.
- Remove the compass from the sensor assembly. The EFIS will have a red cross to show that a sensor is missing. The error light on the EFIS module will come on.
- Download program example 10 to the E-blocks system. This is a simple compass simulator where the angle of the compass is dictated by the potentiometer on the Combo board.
- The EFIS red cross should go out. Clear the 'compass sensor missing' fault on the PC Avionics software.
- Using the console, identify message 0241.
  How does this differ from the message generated by the compass module?

So what?

- The E-blocks system behaves like the compass module in that it generates messages, correctly structured, sent on the CAN bus, with the right timing and something like the right content. The only difference is that rather than a compass sensor, it uses a potentiometer.
- Whether CAN signals are created using an ECU, bare-board electronics, or a real sensor, the CAN signal has just the same structure and voltages.

Avionics system

Teacher's notes

Reference - About this course

AEROSPACE

This workbook is intended to reinforce the learning that takes place in EASA module 5 - Digital Techniques and Electronics Instrument Systems.

Coverage of module 5 is split across several Matrix products as follows:

	EASA Electronic fundamentals (LK9282)	Electronic Flight information systems (AV3737)	Microcontroller systems in Aviation (BL2976)
5.1 Electronic instrument systems			
5.2 Numbering systems		✔	✔
5.3 Data conversion			✔
5.4 Data buses		✔	✔
5.5 Logic circuits	✔		
5.6 Basic computer structure			✔
5.7 Microprocessors			✔
5.8 Integrated circuits			✔
5.9 Multiplexing		✔	✔
5.10 Fibre optics	✔		
5.11 Electronics displays		✔	✔
5.12 Electrostatic sensitive devices			
5.13 Software management and control		✔	
5.14 Electromagnetic environment			
5.15 Typical electronic / digital aircraft systems		✔	

Reference - About this course

Within this package the learning outcomes are as follows:

- Decimal, binary, hexadecimal
- Microcontroller chips and types
- Microcontroller technology: CPU, ROM, RAM, ALU, inputs, outputs, clock, internal peripherals
- Microcontroller circuits and systems
- Microcontroller programming:
    - Flow chart programming techniques
    - Inputs, Outputs, Delays, IF...THEN, While, Goto points, Calculations, Decisions, Subroutines
    - Compilers, Assemblers, Linkers
    - Variables, open loop control, closed loop control
- LED indicators, 7-segment LED displays
- Switches - push to make and slide
- Serially addressed LCD displays
- Potentiometers and sensors
- PWM control of motors, stepper motors, servo motors,
- Microcontroller communications and multiplexing
- Digital to Analogue conversion and Analogue to Digital conversion (through PWM)

Prior Knowledge

Students should have completed the Electronics Fundamentals part of the course - EASA module 4.

Using this course

It is expected that these worksheets should be distributed to the students in electronic or printed format.

Many students will be happy reading a PDF from a computer screen - but the full manual can be printed for them if desired.

Worksheets usually contain:

- an introduction to the topic under investigation;
- step-by-step instructions for the practical investigation that follows;
- a section headed 'So What?' which aims both to challenge learners by questioning their understanding of a topic and also provides a useful summary of what has been learned.
  It can be used to develop ideas and as a trigger for class discussion.
- a section headed 'For Your Records' which provides important summary information that students should retain for future reference. Students can either write on the worksheet itself or make notes on a separate document.

This format encourages self-study, with students working at a rate that suits their ability. It is for the tutor to monitor that students' understanding is keeping pace with their progress through the worksheets and to provide additional work that will challenge brighter learners. One way to do this is to 'sign off' each worksheet, as a student completes it and, in the process, have a brief chat with the learner to assess their grasp of the ideas involved in the exercises that it contains.

AEROSPACE

Teaching Microprocessor systems

This course is a practical one: learning microprocessor systems purely from a book is dull. This is an engaging course that will teach elements of programming as well as how microcontroller based hardware works. Whilst not strictly in the EASA syllabus, programming is a key skill which teaches fault-finding techniques and logical thinking. All modern engineers need some programming ability.

This course teaches about microcontrollers rather than microprocessor systems. Microprocessor systems are very much an old technology and in the last 20 years microcontrollers have taken over as far as flight systems are concerned. Once a student has understood microcontroller systems, the learning curve for microprocessors will be significantly shorter.

The course uses 'Flowcode Embedded', an easy platform from which to study graphical programming software for microcontrollers.

Learning at home

Flowcode is free of charge for hobbyists - so students can download a copy and use it at home if they wish.

Arduino vs PICmicro

This course makes use of a Matrix PICmicro development panel. This is an advanced learning solution that includes instrumentation and debug facilities including In-Circuit Debug and In-Circuit Test. These features are great for debugging programs and seeing code execute in a different way.

Much of the course is also compatible with Arduino and there are some notes in the Appendix on using Arduino. Students can buy an Arduino or Arduino clone board for as little as $20 and continue their learning at home.

The learning path

For Worksheets 1 to 6, the student is guided step-by-step in constructing the flowchart programs. This helps students to understand the structure and individual commands and stimulates learning.

As the programs get more complex the benefit of this approach diminishes and we ask students to load prewritten programmes that carry out various tasks.

These are:

- Example programme 7: Potentiometer / light sensor project;
- Example programme 8: DC motor control using Pulse Width Modulation;
- Example programme 9: Stepper motor control;
- Example programme 10: CAN bus compass mimic.

Time:

The timings are approximate. It will take most students between 8 and 12 hours to complete the full set of worksheets. It is expected that a similar length of time will be needed to support the learning in a class, tutorial or in a self-study environment.

The E-blocks equipment and software provided can support up to 60 hours of learning. The Matrix product 'Introduction to Microcontroller Programming' (product code CP4375) is available free of charge from the Matrix web site for those that want to learn more.

Teacher's notes

AEROSPACE

Worksheet	Notes for the Tutor	Timing
1	In this worksheet students are simply tasked with becoming familiar with the Flowcode environment. The text shows students several areas of the package to investigate. Students should be encouraged to work through this methodically, but we are aware that many will just dive in and work their own way round the software.	30 minutes
2	In this student students get hold of the E-blocks hardware and build their first programme: lighting an LED. They learn about using flow charts for programming, about simulation, about microcontroller ports and understand how to download programs to the microcontroller.	30 minutes
3	Students extend the activities in worksheet 2 and learn about the binary and hexadecimal numbering systems.	60 minutes
4	Now that students are gaining confidence the worksheets become more task orientated. In this worksheet students construct several examples using inputs, loops, and displays to make a stopwatch and adder and a logic circuit in a microwave oven. Students should be given time to explore and make variations. Brighter students who progress faster can easily be given additional exercises.	5 hours
5	Students move on to the actuators panel. At this point students are asked to download a sequence of pre-written examples. If you have more time then you could ask students to create the full programme: the issue here is not capability but time available. Program 7 - potentiometer and light sensor Program 8 - PWM DC motor control Program 9 - Stepper motor control	2 hours
6	In worksheet 6 students connect the CAN bus board to the system and download a program that mimics the Compass senor in the EFIS system. The objective here is to help students understand that electronics systems in aviation are simply the equivalent to E-blocks systems but more compact.	60 minutes

Bill of materials

AEROSPACE

There are two parts to this course:

1. CP5715 - EFIS for Aviation

2. CP7244 - Digital techniques in Aviation

CP5715 - EFIS for Aviation requires the following parts:

1	AV2209	EFIS display panel - with GPS antenna
1	AV5322	EFIS sensor panel
1	COM00170	Pipette filler
1	COM4177	4mm diameter tubing, 300mm
1	HP2666	Power supply
2	HP4039	Lid for plastic trays
2	HP5540	Deep plastic tray
1	HP7750	Locktronics daughter tray foam insert
1	HP9564	62mm daughter tray
1	HPUSB	USB lead
1	LK5202	Resistor - 1K, 1/4W, 5% (DIN)
2	LK5206	Resistor - 120 ohm, 1W 5% (DIN)
1	LK5209	Resistor - 5K6, 1/4W, 5% (DIN)
5	LK5214	Potentiometer, 10K (DIN)
28	LK5250	Connecting Link
2	LK5603	Lead - red - 500mm, 4mm to 4mm stackable
2	LK5604	Lead - black - 500mm, 4mm to 4mm stackable
2	LK5660	Lead - black - 1000mm, 4mm to 4mm stackable
2	Unknown	Lead - red - 1000mm, 4mm to 4mm stackable
8	LK5607	Lead - yellow - 500mm, 4mm to 4mm stackable
8	LK5609	Lead - blue - 500mm, 4mm to 4mm stackable
2	LK8900	7 x 5 baseboard with 4mm pillars
1	MI0550	MIAC NXT
1	HP2666	Adjustable power supply
2	LK5620	Lead, yellow, 1000mm 4mm to 4mm stackable
2	LK5640	Lead, blue, 1000mm, 4mm to 4mm stackable

CP7244 - Digital techniques in Aviation requires the following parts:

1	BL0127	E-blocks 2 actuators board
1	BL0140	E-blocks 2 CAN bus board
1	BL0161	Patch board
1	BL0562	PIC development centre and printed panel
1	FCXXX	Flowcode for education
1	HP2666	Adjustable power supply
2	HP4039	Tray Lid
1	HP5540	Deep tray
1	HP9564	62mm daughter tray
1	HPUSB	USB lead

Reference - the full EFIS system
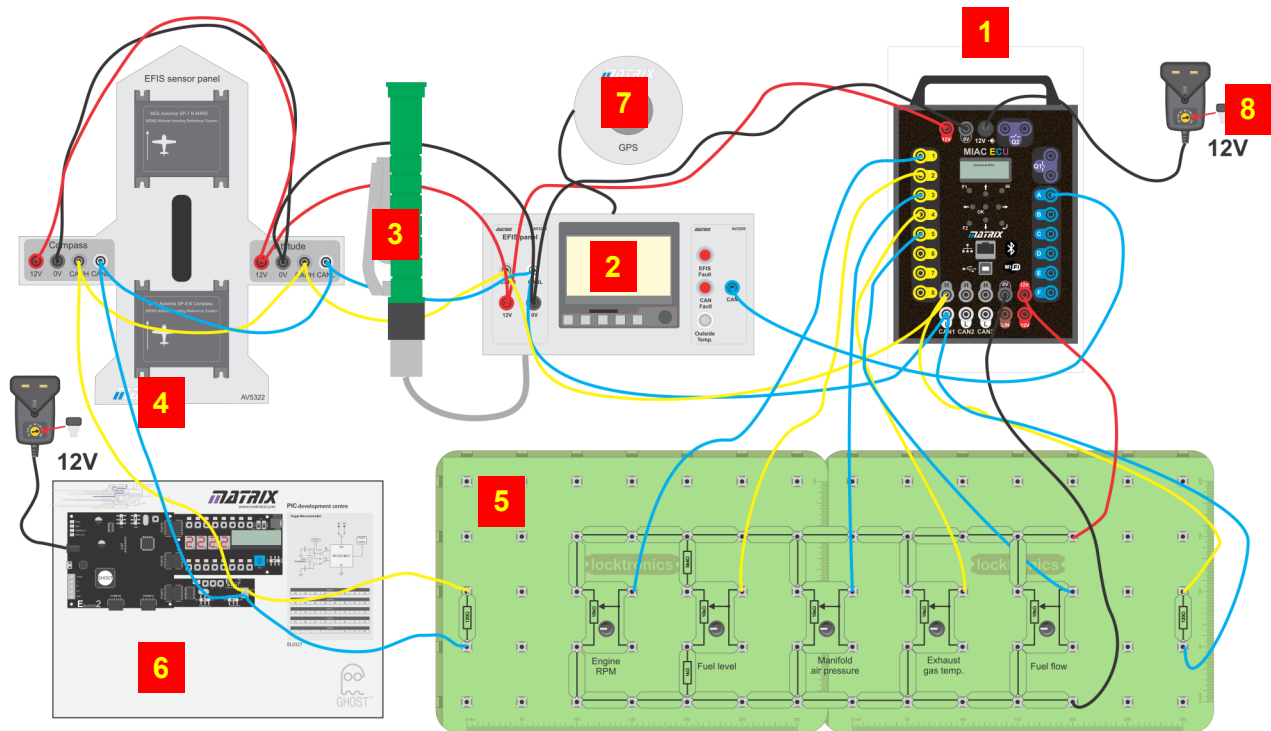
The image above shows all the parts of the EFIS system.

They are:

1. The MIAC NXT - this is a controller with 4mm sockets that has eight analogue or digital inputs, two high power relays, six high power transistor outputs, three CAN bus interfaces, one LIN bus interface, a LCD display, a keypad and integrated USB, Wifi, Ethernet and Bluetooth communications systems. This is a rugged educational product designed by Matrix TSL for teaching and learning. Further details below.

2. The EFIS system - see below for further details - which includes the EFIS module, 4mm connectors for power, CAN and a warning signal from the MIAC. Further details below.

3. A pipette and tubing that can be used to alter the pressure on the EEFIS module internal pressure sensor.

4. The EFIS sensors assembly with Attitude and Compass sensors. Each sensor has 4mm connectors for power, ground and CAN bus. Further details below.

5. The Locktronics base boards with components

6. An E-blocks II development system with microcontroller upstream board, Combo board, and CAN bus interface board. Further details below. This is not integral to the solution. This is used in another module on microcontroller programming and systems and it provides alternative compass data via CAN bus.

7. The GPS antenna which plugs directly into the back of the EFIS module using a micro BNC plug/socket.

8. A 12V power supply.

Not shown:

- The Windows PC
- The Avionics analyser software which is detailed below.
- Flowcode embedded software - detailed below.

Reference - The E-blocks system
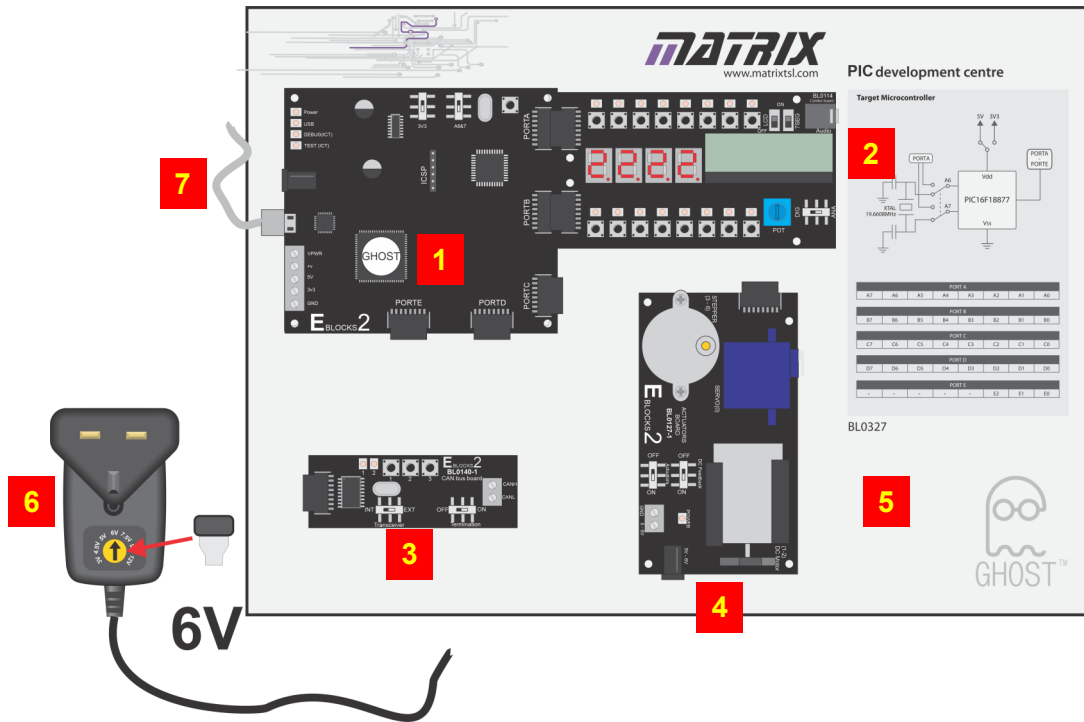
The E-blocks system allows you to create CAN bus messages at a microcontroller level.

The parts are:

1. PICmicro microcontroller upstream programming board. This allows you to reprogram the resident PIC device using Flowcode and presents all of the PIC input /output pins on neat multiway headers so that other boards can be added to the system.

   This board includes full In-Circuit debugging and instrumentation that shows what is happening on the pins on the PIC device.

2. E-blocks Combo board. This includes switches, LEDs, an LCD display, 7-segment displays and simple sensors. The Comb board allows you to create simple programs with a user interface.

3. CAN bus board: plugs into the serial port of the PICmicro device and allows you to create fully industrial compatible CAN messages that interface to the system.

4. Actuators board. This includes a DC motor with both analogue and digital feedback, a servo motor and a stepper motor plus the circuitry to drive the motors.

5. Backplane

6. Power supply - used for actuators board at 6V.

7. USB lead which provides power for the E-blocks system and communication with the PC.

The E-blocks datasheet (code BL9983) contains all technical information on the E-blocks hardware - including full circuit diagrams. This is available from the Matrix web site.

Appendix 1:

Introduction to Microcontrollers

Microcontrollers are tiny devices used to control other electronic devices. They are found in a huge range of products. In automotive systems they can be found in engines, anti-lock brakes and climate control systems. In domestic electronics they can be found in TVs, VCRs, digital cameras, mobile phones, printers, microwave ovens, dishwashers and washing machines.

A **microcontroller** is a digital integrated circuit, consisting of a central processing unit, a memory, input ports and output ports.

What is a microcontroller?

The Microcontroller
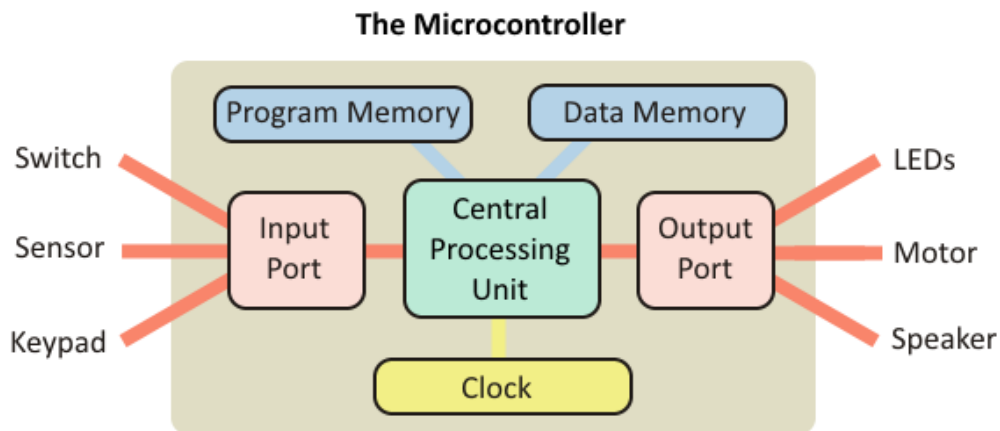
At their heart (or is it brain?) there is a Central Processing Unit (CPU). This processes the digital signals, does calculations and logic operations, creates time delays, sets up sequences of signals etc.

How does it know what to do? It is following a program of instructions, stored in part of the memory, called the 'program memory', inside the PIC.

From time to time, the CPU needs to store data, and then later retrieve it. It uses a different area of memory, called the 'data memory' to do this.

The clock synchronises the activities of the CPU. It sends a stream of voltage pulses into the CPU that controls when data is moved around the system and when the instructions in the program are carried out. The faster the clock, the quicker the microcontroller runs through the program. Typically, the clock will run at a frequency of 20MHz (twenty million voltage pulses every second.)

To talk to the outside world, the microcontroller has 'ports' that input or output data in the form of binary numbers. Each port has a number of connections - often referred to as 'bits'. An 8-bit port handles an 8-bit (or one byte) number.

Information from sensors is fed into the system through the input port(s). The microcontroller processes this data and uses it to control devices that are connected to the output port(s). The ports themselves are complex electronic circuits - not simply a bunch of terminals to hang components on.

What is a microcontroller?

Microcontrollers - PIC and AVR

The name PIC, (Peripheral Interface Controller), refers to a group of microcontrollers, produced by Arizona Microchip.

When we use a PIC microcontroller, we have to specify how we want the ports to behave. The ports are bi-directional, meaning that they can act as either input ports or output ports. When we write a program for the PIC, we start by configuring the ports, telling them whether they are to behave as input ports or output ports.

The input port can receive data (information) in one of two forms, as an analogue signal, or as a digital signal. It is important that we understand clearly the difference between these.

The Digital World

Much of our everyday information is described in numerical format.

For example:

- "It is 2 o'clock."
- "The temperature outside is 21 degrees C."
- "The car was travelling at 48 kilometres per hour."

It is easy to understand data in this form.

For example, the table shows how the speed of a car changes over a period of time.

However, you might wonder what happened at time 35 seconds.

Was the car moving faster or slower than 25 km/h at that moment?

Time in seconds	Speed in kilometres per hour
0	0
10	15
20	21
30	25
40	22
50	20
60	16

The Analogue World

Now the information is given in the form of an analogy! In other words, we use something that behaves in a similar way.

For example:

- The hour glass egg timer:

The greater the time elapsed, the deeper the sand in the bottom of the egg timer.

- The mercury-in-glass thermometer

The hotter it gets, the further the mercury moves up the tube.

- The car speedometer

The higher the speed, the further the pointer moves around the dial.

The problem with analogue data is that you have to do some work to extract it.

For the speedometer, and thermometer, you have to work out where the pointer sits on the scale. On the other hand, it is easy to judge how the temperature of a body or speed of a car is changing - watch how quickly the mercury is moving along the tube or how fast the pointer moves round the dial.

Data representation
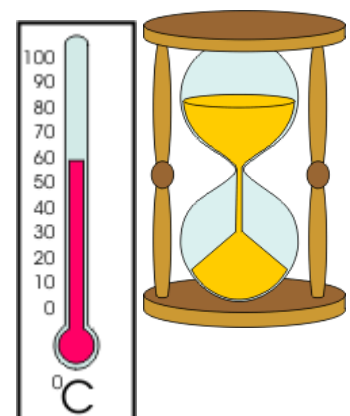
Analogue Data

Many electronic sensors provide signals in analogue form. For example, a microphone provides an electrical 'copy' of a sound wave.

Another - the temperature sensor!

Here is the circuit diagram for one type of temperature sensor.

The output voltage increases when the temperature increases.

It is an analogue signal because the voltage copies the behaviour of the temperature.

An electrical analogue signal can have any voltage value, limited only by the power supply used.

In this case, the output of the temperature sensor could, in theory, go as high as 5V, or as low as 0V.

Over a period of time, the output voltage could change as shown in the diagram. This is an analogue signal.
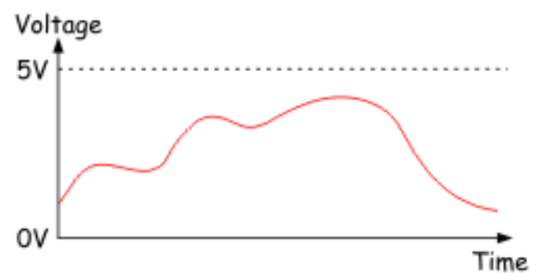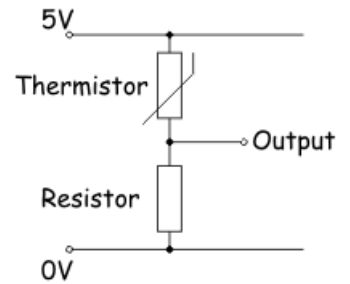
Digital Data

A digital signal carries its information in the form of a number. Electronic systems usually employ the binary number system, which uses only the numbers '0' and '1', coded as voltages.

We could decide on the following code: '0' = 0V, '1' = 5V, for example.

Digital signals, then, have only two possible voltage values, usually the power supply voltage, or as close to it   as the system can get, and 0V.

How can we enter these numbers into an electronic system?

One (very slow) way would be to use a switch (an example of a digital sensor.)

The circuit diagram shows such a digital sensor.

- When the switch is open (not pressed,) the output is 'pulled down' to 0V by the resistor. This output could represent the binary number '0'.

- With the switch closed (pressed,) the output is connected to the positive supply, 5V in this case. This could represent the binary number '1'.
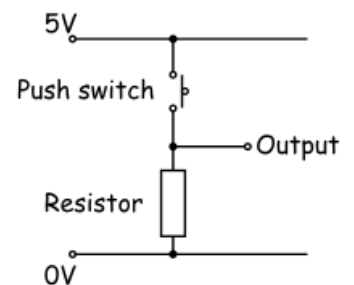
(Note - if the positions of the switch and resistor were reversed, pressing the switch would put a logic 0 signal on the pin etc.)

The following diagram shows a more complex digital signal.
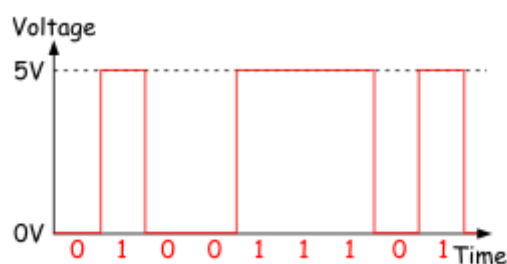
- The nine bit binary number represented by the signal is given under the waveform.

Analogue to digital conversion

- Much of our 'real world' data is analogue, but computers (including microcontrollers) can only process digital data. Fortunately microcontrollers often contain a subsystem that can convert information from analogue format to digital format. This is called an Analogue-to-Digital Converter - usually shortened to 'ADC' or 'A/D'.

- The ADC inside the a microcontroller divides the range of possible analogue voltages into equal steps. The lowest step is given the number '0', and the highest step is given the highest number that the A/D converter can handle.

- This highest number is determined by the resolution of the ADC, which, in turn, depends on number of 'bits' the internal circuitry of the ADC can handle. The resolution of PIC ADCs is 8, 10 or 12 bit.

- For example, if the biggest analogue voltage is 5V, and the PIC has an 8-bit ADC:
    - the highest 8-bit number is 1111 1111 (= 255 in decimal);
    - the first step is 0000 0000 (= 0 in decimal);
    - meaning that there are 256 voltage levels;
    - so stepping from one level to the next involves a voltage jump of 5V/256, or about 20mV.

- When this microcontroller processes an analogue signal, it first divides it by 20mV, to find out how many steps the signal includes. This gives the digital equivalent of the analogue signal.

- The next graph illustrates this process.
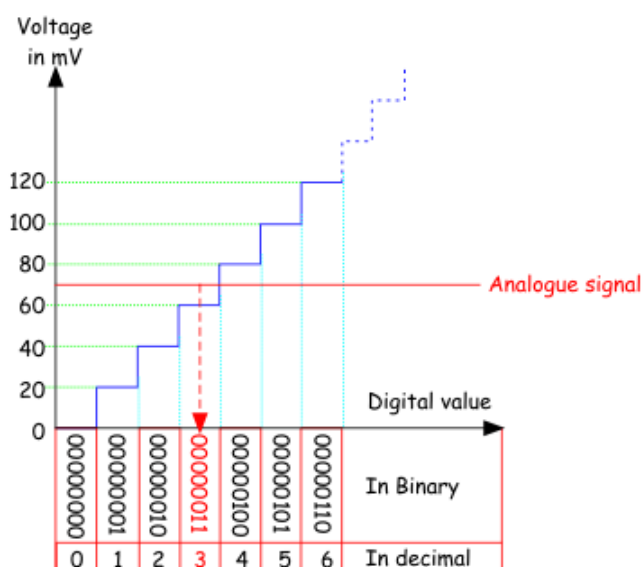
- In this example, the converter outputs '0000 0000' for any analogue signal up to 20mV, outputs '0000 0001' for analogue signals between 20 and 40mV, and so on.
  The analogue signal shown in the graph produces an output of '0000 0011'.

Inputting data into a microcontroller AEROSPACE

- The PIC microcontroller is a digital device, but data can be entered in both analogue and digital forms. Programmers choose whether pins on the PIC are used as analogue inputs, digital inputs or digital outputs. This flexibility leads to complex labelling.

- The diagram shows the pinout for a PIC 16F1937 chip. It has five ports, known as A, B, C, D and E. The pins on port A are labelled RA0 to RA7; pins on port B are labelled RB0 to RB7 etc. Ports A, B, C and D have eight pins but port E has only four.

- For example, up to eight digital sensors can be connected to port A of the 16F1937.

- Pin 2 is marked as '**RA**0/**AN**0', meaning that it can be used as bit 0 of port A, (**R**egister **A** bit 0) or as **AN**alogue input 0.

- The function of each input/output pin is determined by setting the contents of internal registers, called 'data-direction' registers inside the PIC device.

- Pins RA6 and RA7 are also labelled as 'OSC1' and 'OSC2'. They can be connected to an external oscillator circuit or be used for digital input /output.

- Analogue sensors must be attached to the pins labelled with an 'ANx' (ANalogue) label. These, found on ports A, B and E, can handle analogue signals between V_{DD} (5V) and V_{SS} (Gnd).

- Most pins have alternative functions. For example pin 25 is labelled as 'RC6/TX/CK', meaning that it can be Register C bit 6, or the transmit (TX) pin of the internal serial interface, or the **C**loc**K** pin of the internal serial interface.

- Fortunately Flowcode takes care of the internal settings that dictate pin functionality for you.
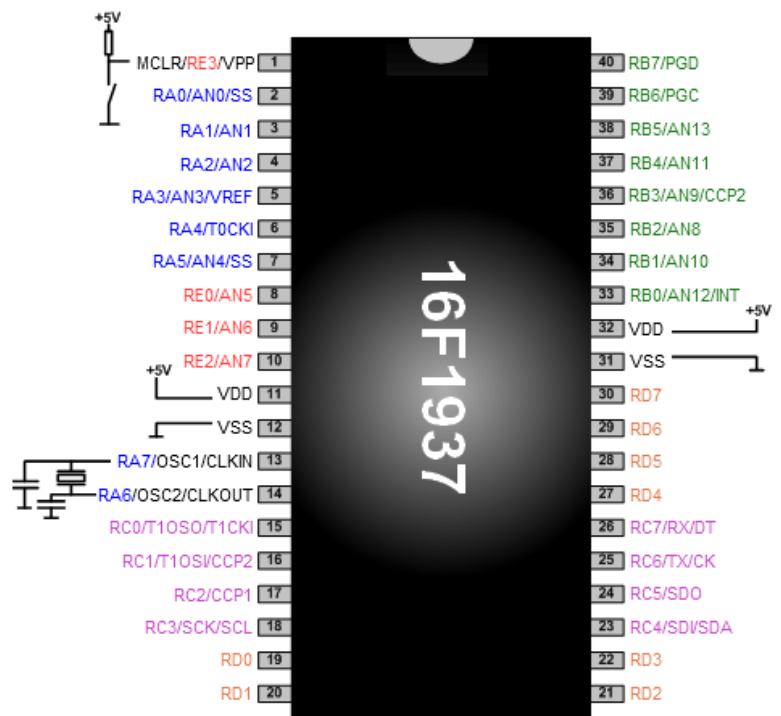
Outputting data

- The microcontroller is a digital device - as we have said several times already! It outputs a digital signal. In most cases, we use this to turn something on and off - '0' = 'off' and '1' = '0n', for example.

- Suppose that we set up port B as the output port, (or let Flowcode do it for us). There are eight pins on port B, so we can switch eight devices on and off. It is important to plan how we connect these devices, as otherwise they might work the opposite way round!

Outputting data

AEROSPACE

The diagram shows eight LEDs connected to port B of a PIC16F84 microcontroller:

The four red LEDs are connected between the positive supply rail and the port pins. For these LEDs, PIC is 'sinking' current.

The four green LEDs are connected between the pins   and the 0V rail. For these, PIC is 'sourcing' current.
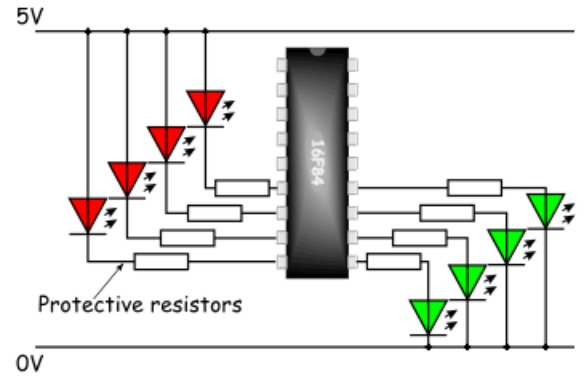
- Each red LED lights up when its pin is at a low voltage, outputting '0' in other words.
  Each green LED lights when its pin is at a high voltage, outputting a '1'.

- There are limits as to how much current the ports can control. Typically, one output pin can handle up to 25mA. This is enough to drive LEDs and buzzers directly, but higher-powered devices will need additional circuitry to interface with the PIC (dealt with later). However, the maximum current for the whole port is around 100mA, so not all pins can output 25mA at the same time.

Maximum current sunk/sourced by any I/O pin	25mA
Maximum current sunk by all ports	200mA
Maximum current sourced by all ports	140mA
Maximum current out of VSS (Gnd) pin	95mA
Maximum current into V_{DD} (5V) pin	70mA

Current Limits

- As you have seen, Flowcode has a simulation mode that allows you to attach LEDs to show the status of the pins on the microcontroller when they are used as outputs. The LED simulation function inside Flowcode assumes that current is **sourced** from the PIC device - like the green LEDs in the diagram above.

- At some stage, you will need to use the PIC pin specifications in order to use them as digital inputs, analogue inputs, or as digital outputs. In particular, there are limitations on the output capabilities of the device. Exceeding these limits, even for a short time, may cause permanent damage to the PIC.

- Fortunately the E-block boards used on this course all have current limiting resistors which protect the PIC device. When using the prototype or patch boards, however, there is no such protection and care must be taken not to damage your device.

Storing Data

- Electronic sub-systems that store data are known as 'memory'. They can store only digital data.

- One item of data is stored in one location in the memory. This data could be the correct combination to disarm a burglar alarm, or the target temperature of a car engine block.

- Each memory location has a unique address, a number used to identify the particular location. This means that we can draw up a map of the memory, showing what data is held in each location.

- The decimal version of the address is included to make the table easier to read.

Address		Data stored
In decimal	In binary	
0	000	11101001
1	001	00100101
2	010	10000101
3	011	11001101
4	100	01110100
5	101	00011011
6	110	11110011
7	111	10000101

- Electronic systems understand only binary numbers. This very small memory has eight locations.
- Notice that numbering normally starts at '0'! It needs a 3-bit binary number to create unique addresses for each location. It allows us to store items of data that are eight bits long, (one 'byte' (1B).
- Our example memory could be called a 8 x 1B memory. Memory systems used in computers are much larger. Data is often stored as 32 bit numbers, allowing the use of much larger numbers. There are many more locations, too. A typical computer memory now has millions of memory locations!

Types of Memory

- There are several types of electronic memory, each with a slightly different job to do.
- We can divide them into two main groups, ROM and RAM.

Read-Only Memory (ROM)

These devices are normally only read (i.e. the contents are accessed but not changed 'written',) during the running of a program.

- The contents are not volatile. (The data remains stored even when the power supply is switched off.)
- They are often used to store the basic programs, known as 'operating systems', needed by computers.
- The group includes:
- PROM (Programmable Read Only Memory),
- EPROM (Erasable Programmable Read Only Memory),
- EEPROM (Electrically Erasable Programmable Read Only Memory)

- A PROM is a one-shot device, which arrives blank, ready to receive data. Data can then be 'burned' into it, but only once. After that, it behaves like a ROM chip that can be read many times but not altered.
- With an EPROM, shining ultraviolet light through a window in the top of the chip erases the contents. New data can then be 'burned' into the memory. Some older PIC devices operate in this way.
- The EEPROM devices work in a similar way to an EPROM, except that the contents are erased by sending in a special sequence of electrical signals to selected pins. 'Flash' memory is a form of EEPROM, widely used as the storage medium in digital cameras, (the memory stick) and in home video games consoles.

Random Access Memory (RAM)

- RAM allows both read and write operations during the running of a program.
- The contents are volatile and disappear as soon as the power supply is removed. (The exception is NVRAM, Non-Volatile RAM, where the memory device may include a battery to retain the contents, or may include an EEPROM chip as part of the memory to store the contents during power loss.)
- They are often used for the temporary storage of data or application programs.
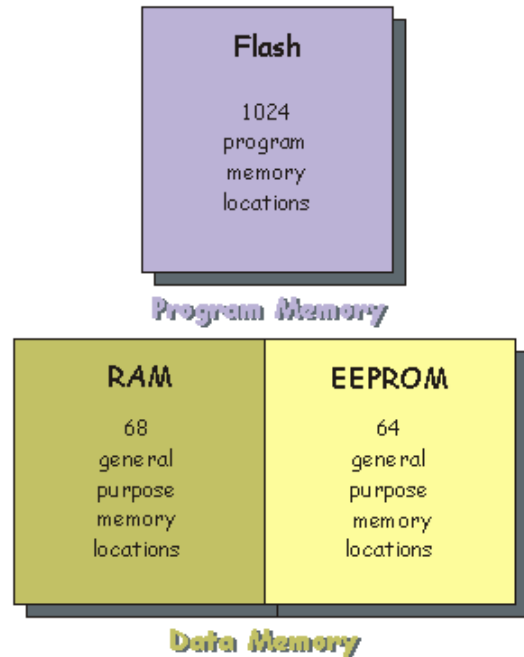
Memory types

Microcontroller memory

PIC chips have three separate areas of memory:

- program memory (Flash);
- user variable memory (RAM);
- EEPROM.

The names give strong hints as to the purpose of the areas!

For the eighteen pin PIC16F84 the graphic illustrates the organisation of the memory:

Program memory is used to store the program!

In most PICs, such as the 16F1937, this uses 'Flash' technology, meaning that it can be programmed and cleared many times. Older PIC's use PROM for the program memory so that many of these can be programmed only once.

Flash

1024 program memory locations

Program Memory

RAM	EEPROM
68 general purpose memory locations	64 general purpose memory locations

Data Memory

Data memory is used to store data!

Part of this uses RAM and part uses EEPROM.

The EEPROM allows us to preserve important data even if the power supply to the system is switched off.

For example, suppose that the PIC is part of a temperature controller that keeps an incubator at a set temperature. It might make sense to store the target temperature value in EEPROM so that we do not have to enter it into the system every time we switch the incubator on.

Programming

Microcontrollers are programmable devices. They do exactly what they are told to do by the program, and nothing else! A program is a list of instructions, along with any data needed to carry them out.

The only thing microcontrollers understand is numbers. There's a problem! We don't speak in numbers, and they don't understand English!

There are two solutions, and both need some form of translator:

- Write the program in english, or something close, and then have the result translated into numbers.
- We can think through the program design in English and then translate it ourselves into a language   that is similar to numbers, known as 'assembler'. From there, it is a swift and simple step to convert   into the numerical code that the microcontroller understands.

These two extremes are known as programming in a **high-level language** (something close to English) or in a **low-level language** (assembler).

The first is usually quicker and easier for the programmer, but takes longer to run the program, because of the need to translate it for the microcontroller.

The second is much slower for the programmer, but ends up running very quickly on the microcontroller.

If you think that this sounds very complicated, you are right. It is! Fortunately, Flowcode works using flowcharts - the easiest, and highest level, of programming and then takes care of all translation needed.
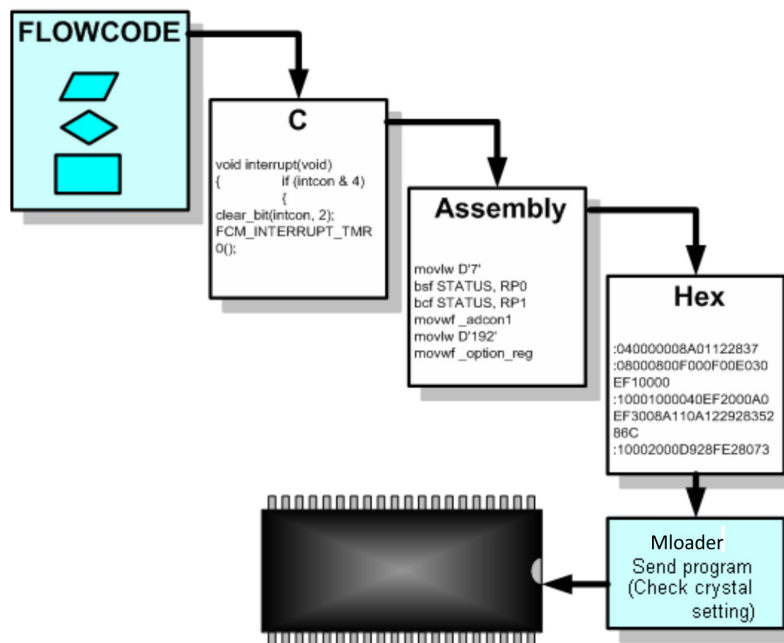
The Flowcode process

AEROSPACE

The Flowcode process

- 'Flowcode offers an easy way to program microcontroller chips, as you will see. Once the flowchart is designed on-screen, one press of a button causes the software to translate it into numerical code!

- Flowcode passes the program through a number of processes before it gets sent into the microcontroller. The flowchart is processed:

  - first into C code,

  - then into Assembler,

  - and finally into hexadecimal numbers or 'Hex', which the microcontroller 'understands'.

- The Hex code is then sent into the microcontroller, using a subsidiary program called 'Mloader'.

- When you select *Build > Project Options... Configure* from the Flowcode menu, the program 'Mloader' runs. It controls a number of options and configurations by setting the value of registers inside the device when you download a program.

- The Hex code is 'burned' into the microcontroller program memory. Since Flash memory is used to form the program memory, the program is not lost when the microcontroller is removed from the programmer. This allows you to use it in a circuit. Equally, use of Flash memory means that you can reuse the microcontroller and overwrite the program memory with a new program.

Running the Program

- As soon as the microcontroller is powered up and supplied with clock pulses, it will start to run whatever program is stored in program memory (Flash).
- When you press the reset button on the microcontroller programming board, the program restarts from the beginning.
- During programming the microcontroller stops while the program is being loaded. When that is completed, it then restarts and runs the downloaded program.

The PIC16F877

## Different types of microcontroller

There are a large number of microcontroller devices available, from the humble 16F84 to larger more complex microcontrollers, such as the 40 pin 16F1937. Different microcontrollers have different number of ports, or I/O pins, analogue inputs, larger memory, or advanced serial communications capabilities such as RS232 or SPI bus.

Deciding on which device to use for a project can be a task in itself. For this course we use a 16F1877 device, a 40 pin PIC that has many internal subsystems (like an A/D converter, and a serial port).

PIC16F1877 Architecture

As this course uses the PIC16F18877 PIC, it is important that you understand a little more about what it does and how to use it. This section details the pins that are available on the 16F1877 and the connectors they use on the programmer board. (The section on 'Using E-blocks' looks at how these connections are made).

At this point in a traditional programming course, you would be introduced in some detail to the various internal circuit blocks of the PIC device. You would need this information to write code for the PIC in C or assembly code. No need - Flowcode takes care of these details!
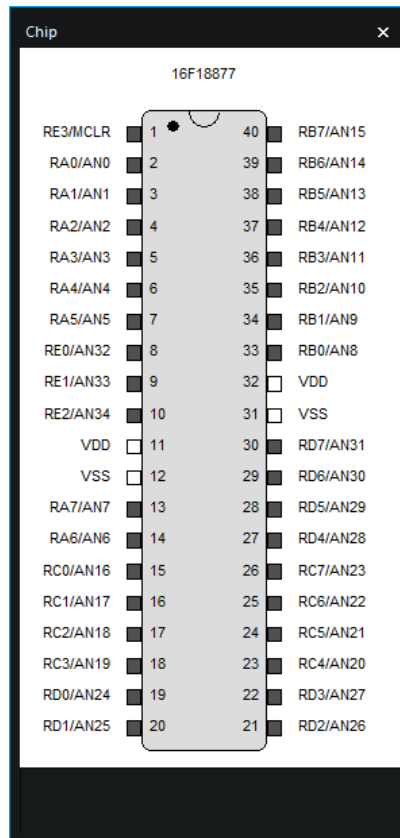
However, you do need to understand the input and output connections of the PIC, the memory available and the role of the other subsystems in the PIC.
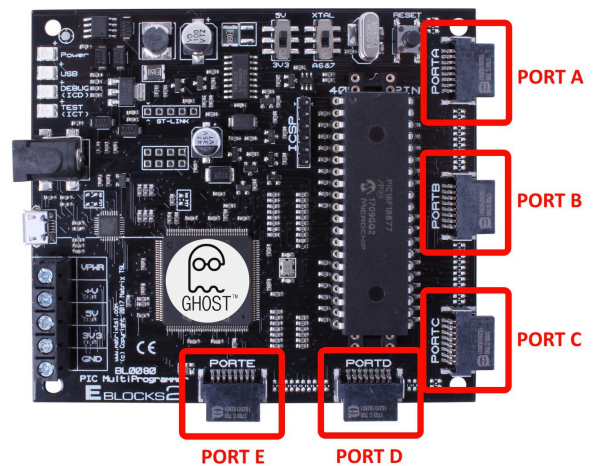
Ports - The PIC16F18877 PIC has five ports, labelled 'A' to 'E', connected to the rest of the microcontroller internals by an 8-bit bus system.
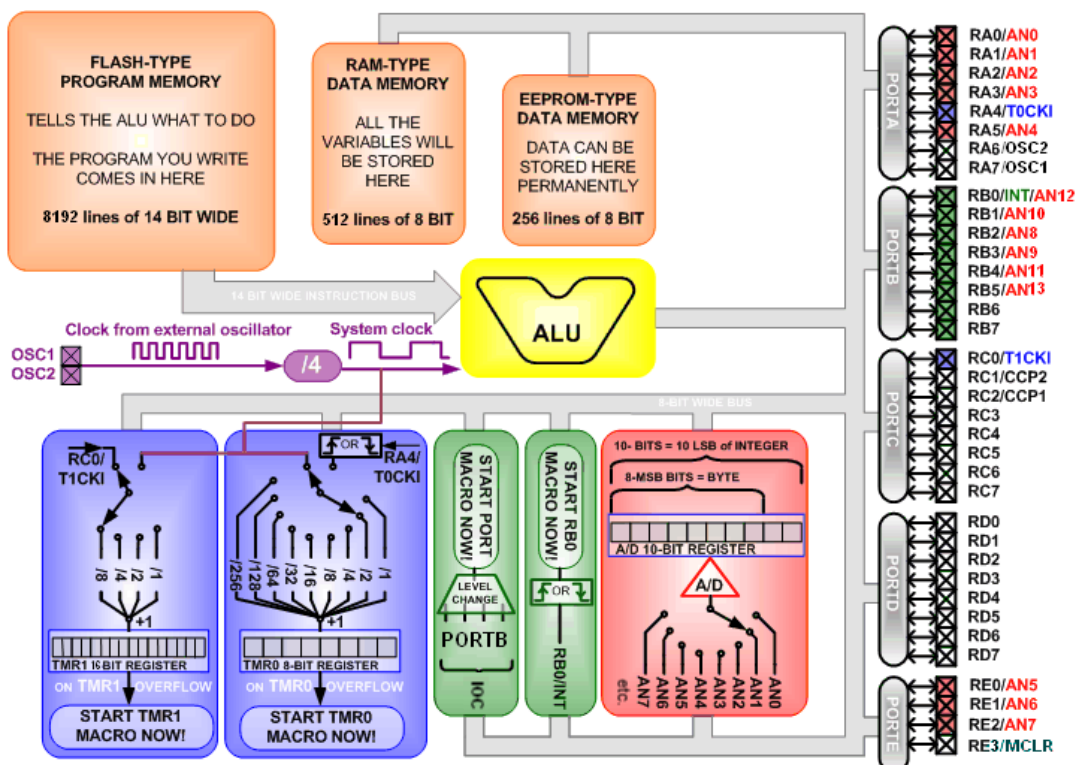
The PIC16F18877

The PIC16F877 pin out:

Other subsystems in the PIC16F18877:

Memory:

Flash

- Flash memory is used to store the program you write.
- This program is 'compiled' by the computer to binary code and then downloaded into the Flash memory .
- You can read from, and write to it and it is retained, even after a power cut.
- The Flash memory contained in the 16F18877 can store up to 32768 program commands.

RAM

- Data from inputs, outputs, analogue inputs, calculations etc. is typically stored in 'variables' (values in the program that alter as it runs). RAM is where these are stored.
- This memory is erased every time the power gets cut or a reset occurs.
- It also contains system 'registers' which control and report the status of the device.
- The RAM memory in the 16F18877 can store up to 4096 bytes of data.

EEPROM

- EEPROM is where data can be permanently stored
- This memory is of the PROM-type - preserved every time the power cuts or a reset occurs.
- The EEPROM of the 16F18877 can store up to 256 bytes of data.

ALU:

- The ALU (Arithmetic Logic Unit) is at the heart of the PIC's data processing.
- All data passes through this unit.
- The program in the Flash memory tells the ALU what to do.
- The ALU can send data to, and fetch data from all the separate blocks and ports in the PIC using the 8-bit wide data-bus.
- The ALU needs four external oscillator clock pulses to execute one whole instruction.
- The ALU works in a very complicated way. Fortunately Flowcode programmers do not need to know how it works.

Timer 1 (TMR1):

- This timer interrupt is used to provide the microcontroller with exact timing information.
- It is 'clocked' either by the system clock or by an external clock on pin RC0.
- Either clock can be divided by 1, 2, 4 or 8 by configuring the Prescaler of TMR1 in Flowcode. The resulting output triggers TMR1 and increments the TMR1 register.
- TMR1 is a 16-bit register, which 'overflows' when it reaches '65536'.
- At the instant it overflows, it generates an interrupt and the TMR1 register is reset to '0'.
- This TMR1 Interrupt stops the main program immediately and makes it jump to the TMR1 macro.
- After this finishes, the main program continues from where it left off just before the interrupt.

For example:

External clock oscillator frequency (crystal oscillator)	19 660 800 Hz
System Clock (four clock pulses per instruction)	4 915 200 Hz
Set prescaler to '8' (divides by 8)	614 400 Hz
Overflow frequency when TMR1 = '65536'	9.375 Hz

Result: TMR1 interrupts the main program and execute the TMR1 macro 9.375 times per second.

Timer 0 (TMR0):

- This timer interrupt also provides the microcontroller with exact timing information.
- It is 'clocked' either by the system clock or by an external clock on pin RA4.
- This system clock runs exactly four times slower than the external oscillator clock.
- Either clock can be divided by 1, 2, 4 or 8, 16, 32, 64, 128, or 256 by configuring the Prescaler of TMR0 in Flowcode. The result triggers TMR0 and increment the TMR0 register.
- This TMR0 register is an 8-bit register, which overflows when it reaches 256.
- At the instant it overflows, it generates an interrupt and the TMR0 register is reset to 0.
- A TMR0 Interrupt stops the main program immediately and makes it jump to the TMR0 macro.

After this finishes, the main program continues from where it left off just before the interrupt.

For example:

External clock oscillator frequency (crystal oscillator)	19 660 800 Hz
System Clock (4 clock pulses per instruction)	4 915 200 Hz
Set prescaler to 256 (divides by 256)	19 200 Hz
Overflow when TMR0 = 256	75 Hz

- **Result:** TMR0 interrupts the main program and execute the TMR0 macro 75 times per second.

The PIC16F18877

AEROSPACE

RB0 External Interrupt:

- A logic level change on pin RB0 can be configured to generate an interrupt.
- It can be configured in Flowcode to react to a rising or to a falling edge on RB0.
- If set to react to a rising edge, when one occurs:
  - it immediately stops the main program;
  - the RB0 related macro is executed;
  - then the main program continues from where it left off just before the interrupt.
- This happens every time a rising edge is detected at pin RB0.

PORT B External Interrupt:

- A logic level change on any combination of pins on port B can generate an interrupt.
- This can be configured to occur on a rising or a falling edge, or both.
- When one of these interrupts occurs:
  - it immediately stops the main program;
  - the port B related macro is executed;
  - then the main program continues from where it left off just before the interrupt.
- This happens every time a level change is detected on one of the pins selected on port B.

A/D:

- The 16F18877 has fourteen pins that have an extra A/D function but only one 10-bit A/D converter.
- This implies that these fourteen analogue inputs can't all be read at the same time.
- A built-in analogue switch, configured in Flowcode, selects which inputs are sampled.
- After the 'sample' instruction, the analogue switch points to the correct input and this is converted into a 10-bit binary value.
- In Flowcode, you can opt to use only the eight most-significant bits (MSB's) of this 10-bit value, by using the 'GetByte' instruction, or to use the full ten bits by using the 'GetInt' instruction. The ten bits will fill up the ten least-significant bits (LSB's) of the selected 16-bit integer variable.
- After this, the program can select to read another analogue input.

Busses:

- PIC and AVR (Arduino) microcontrollers use Harvard architecture.
- This means that there are separate busses for instructions and for data.
- The data bus is 8-bits wide and connects every block and port together.
- The instruction bus is 14-bits wide and transports instructions, which are 14-bits long, from the program memory to the ALU.

Introduction to 'clocks'

- Every microcontroller needs a clock signal to operate. Internally, the clock signal controls the speed of operation and synchronises the operation of the various internal hardware blocks.
- In general, microcontrollers can be 'clocked' in several ways, using:
  - an external crystal oscillator;
  - 'RC' mode, where the clock frequency depends on an external resistor and capacitor;
  - an internal oscillator.
- The 'RC' mode exists for reasons partly historical and partly economic. It was introduced as a low cost alternative to a crystal oscillator and is fine for applications that are not timing critical.

Appendix 2:

Using E-blocks

E-blocks are small circuit boards that can easily connect together to form an electronic system. There are two kinds of E-Blocks. **Upstream** boards and **Downstream** boards.

A variety of boards can be combined to create a full system with downstream boards connected to upstream boards.

E-blocks are ideal companions to Flowcode software, allowing users to test and develop their Flowcode programs. Programs can be compiled directly to the boards, providing ideal development environments.

Using E-blocks

E-blocks consist of upstream boards and downstream boards.

Upstream boards

'Upstream' is a computing term indicating a board that controls the flow of information in a system. They are usually programmed in some way.

Any device which contains 'intelligence' and can dictate the direction of flow of information on the bus can be thought of as an 'upstream' device.
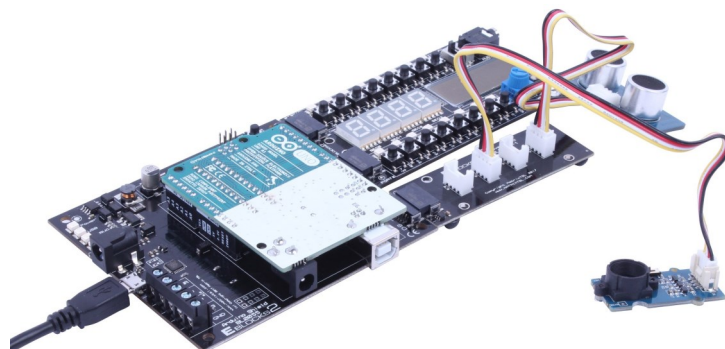
Examples include microcontroller boards, and Programmable Logic Device boards.
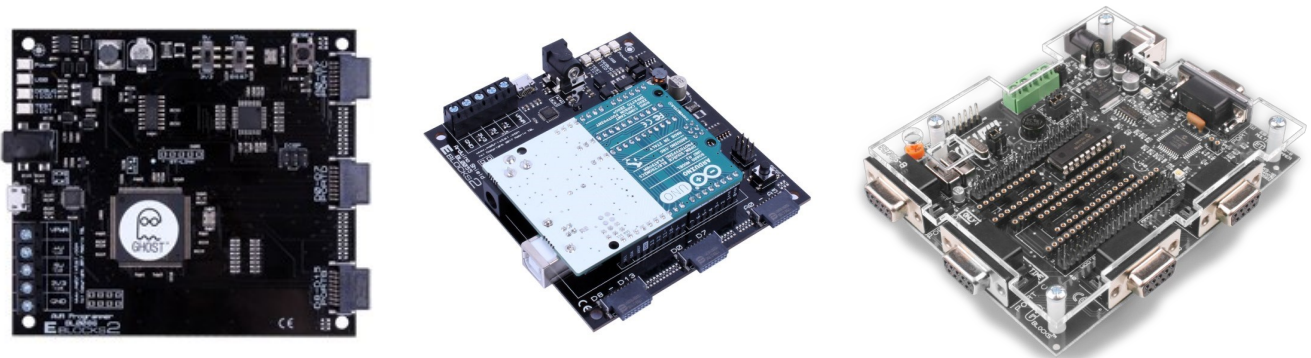
Downstream boards

'Downstream' boards are controlled by an 'upstream' board, but information can flow into or out of them. Examples include LED boards, LCD boards, RS232 boards etc.

Upstream and downstream boards combined to form a full system, with the downstream boards plugging into the upstream 'intelligent' boards.

BL0011 PIC Programmer

- The board has five ports, labelled A to E.

- Ports 'B', 'C' and 'D' offer full 8-bit functionality.

- Port 'A' has 6-bit functionality (8-bit if the internal oscillator is selected).

- Port 'E' has 3-bit functionality.

- It can be powered from an external power supply, delivering 7.5V to 9V or from a USB supply.

- If the Reset switch is pressed, the program stored in the microcontroller will restart.

- The board is USB programmable via a programming chip. This takes care of communication between Flowcode and the microcontroller.

- The microcontroller executes one instruction for every four clock pulses it receives.

- (Note - a single instruction is NOT the same as a single Flowcode symbol, which is compiled into C and then into Assembly and probably results in a number of instructions).

- This course uses an 8MHz crystal which is multiplied up to 32MHz internally.

- Switches allow the user to select a number of options, such as external or USB power supply.

- Where the microcontroller uses an internal oscillator, all eight bits of port A can be used for I/O operation

- A PICKit3 tool from Microchip can be used via the ICSP header.

- It comes with a surface mounted PIC16F18877 device.

- It provides power to the downstream E-blocks boards via the port connectors.

- It contains the Matrix Ghost chip for real time in-circuit debugging when combined with Flowcode.

Using E-blocks

BL0114 Combo Board

The board combines on one compact board the functionality found on a number of individual E-blocks boards:

- BL0167 LED board (x2)
- BL0169 LCD board
- BL0145 Switch board (x2)

For this course, the port connectors attach to female connectors on ports A and B of the upstream board.

The board provides a set of eight switches and eight LEDs for port A and the same for port B.

With the main switch in the DIG position, port A is routed to its push switches (SA0 to SA7), to LEDs (LA0 to LA7) and to the quad 7-segment display.

With the main switch in the ANA position, port A is switched to the analogue sensor section of the board, so that pin RA0 is connected to the on-board light sensor and pin RA1 is connected to the potentiometer to give a variable output voltage, (simulating the action of an analogue sensing subsystem).

Note: With the switch in the ANA position, the on-board switches and LEDs LA0 and LA1 will not operate.

Port B I/O pins are routed to its push switches (SB0 to SB7), to the LEDs (LB0 to LB7), to the quad 7-segment displays and to the LCD display.

The quad 7-segment display is turned on by switch '7SEG'. It is connected to both port A and B.

- Port B is used to control the LED segments and the decimal point).
- Port A, bits 0 to 3, select which display is activated.

The LCD is a 20 character x 4 lines module, turned on by switch 'LCD'.

Normally a complex device to program, Flowcode takes care of the complexities, unseen by the user.
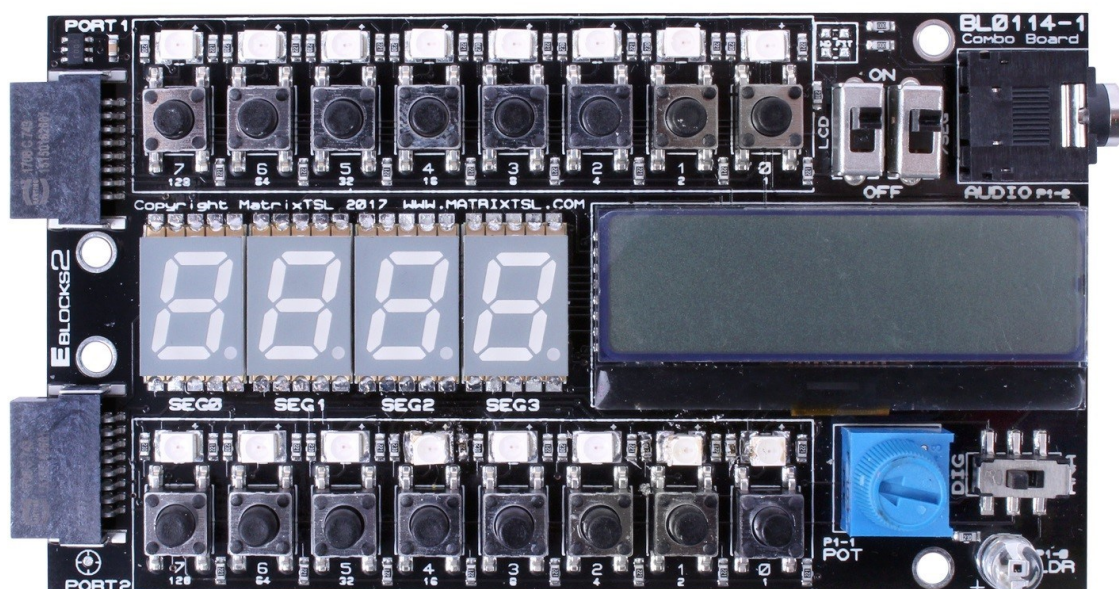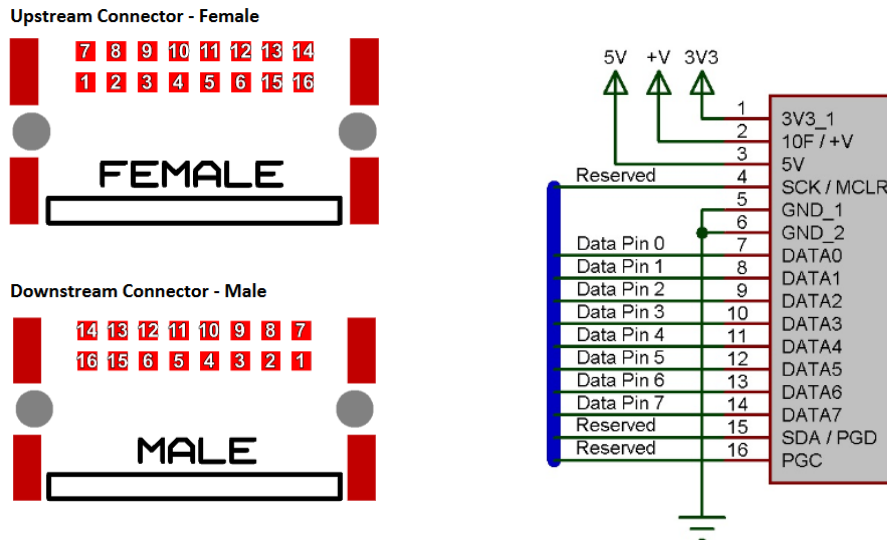
Using E-blocks

Connecting E-blocks together

E-blocks2 are built on a bus-based concept. Each E-block connects together with a 16 pin Har-flex connector, with the female ports attached to the 'intelligent' upstream boards and the male connectors attached to downstream boards.
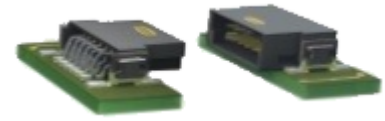
The diagram below shows that the first three pins are used to transfer the power to the downstream board, pins 4,15 and 16 are reserved. Pins 5 and 6 are connected to ground while pins 7-14 are the pins which transfer our 8 bits worth of data between the boards.

Using E-blocks on the bench

You do not need a backplane to use E-blocks - you can simply connect them together on the bench. In each E-blocks package you will find a four small rubber feet to facilitate this. These provide a degree of protection for the E-blocks boards and help prevent short-circuits from tinned copper wire and other metal objects on the bench. The disadvantage is that your E-blocks system is less portable as the connectors will be under more stress as the system is moved about.
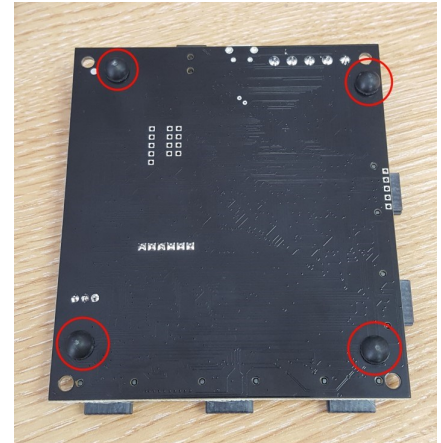
Protecting E-blocks circuitry

Where possible, leaded components have been used on E-blocks boards for devices that are susceptible to electrical damage. This makes the task of replacing them simpler should they be damaged.

To protect 'upstream' components, all 'downstream' E-blocks boards include protective resistors. Should errors occur when declaring the nature of port pins, e.g. an input declared as an output, no damage will be caused.

However there are circumstances where it is possible to cause damage:

- Care is needed when using screw terminal connectors and patch/prototype boards.
- Where possible, use protective resistors for the lines you need to connect when connecting two 'upstream' boards together with a gender changer E-block.
- Make sure you are earthed before handling E-blocks circuit boards to minimise the risk of static damage. If you have not got an antistatic wrist band, then touch a radiator or other earthed metal object.
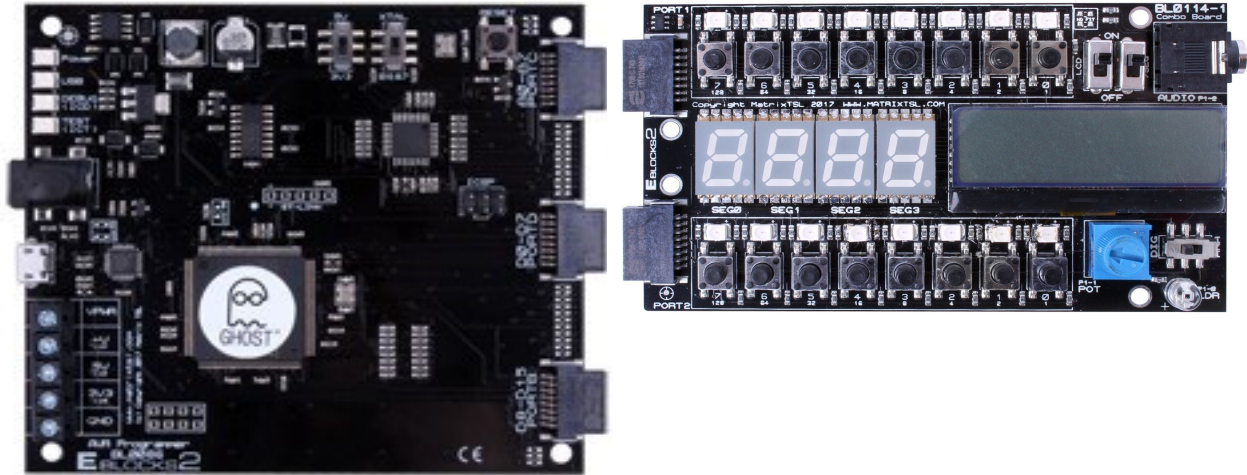
Before making any changes to the E-blocks system, turn off the power supply.

Introduction

The hardware:

Most exercises use the BL0011 / BL0080 Multiprogrammer and BL0114 Combo board.

Most can also be completed using the Arduino Uno Shield (BL0055).

However, these require different PORT settings.

Hardware and software settings used to test most programs:

BL0011 / BL0080		Jumper settings
Microchip PIC MCU	16F18877	
Target voltage	5V	J15
Voltage source	PSU	J11
Programming Source	USB	J12,13,14: USB
Oscillator Selector	OSC	J18,19
Port A E-block	E-blocks Combo board	
Port B E-block	BL0114	
Port C E-block		
Port D E-block		
Port E E-block		

Flowcode and download settings:

Menu	Name	Setting
Build > Project Options... > Choose a Target	Family - PIC	16F18877
Build > Project Options... > General Options	Clock speed (Hz)	32 000 000
	Simulation speed	Normal
Build > Project Options... > Configure	Oscillator	HS Oscillator
	Watchdog Timer Enable bit	Disabled

Appendix 3:

Arduino adjustments

# Appendix 1

ARDUINO: SECTION A

BL0055 Arduino Shield

- The board has three ports, labelled **A0-A5**, **D0-D7** and **D8-D13**.

- Port **D0-D7** offers full 8-bit functionality.

- Port **A0-A5** and **D8-D13** has 6-bit functionality.

- It can be powered from an external power supply, delivering 7.5V to 9V or from a USB supply.

- If the Reset switch is pressed, the program stored in the Arduino will restart.

- The board is USB programmable via a programming chip. This takes care of communication between **Flowcode** and the **Arduino** device.

- The Arduino executes one instruction for every clock pulse it receives.

  (Note - a single instruction is NOT the same as a single Flowcode symbol, which is compiled into C and then into Assembly and probably results in a number of instructions).

- This device uses a **16MHz crystal**.

- The board will detect whether an external power supply or USB power supply should be used.

- An AVR ISP tool from Microchip can be used via the ICSP header.

- It is usually supplied with an Arduino Uno device.

- It provides power to the downstream E-blocks boards via the port connectors.

- It contains the Matrix Ghost chip which allows for real time in-circuit debugging and pin monitoring when combined with Flowcode.

Appendix 1

AEROSPACE
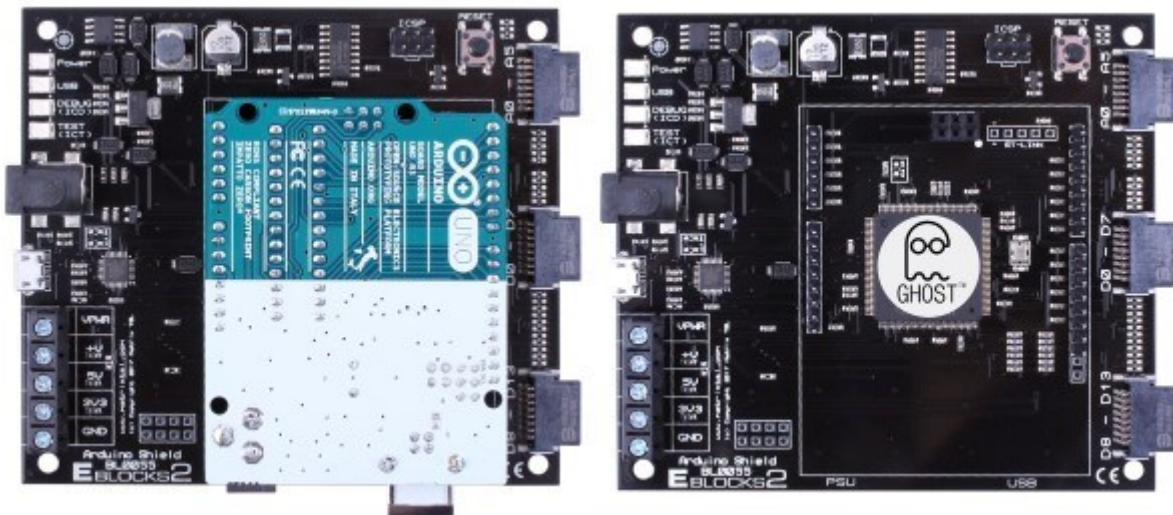
ARDUINO: SECTION B

**Selecting Arduino in Flowcode**

On opening Flowcode, you are presented with the 'Welcome' screen. Click on **New Project**.
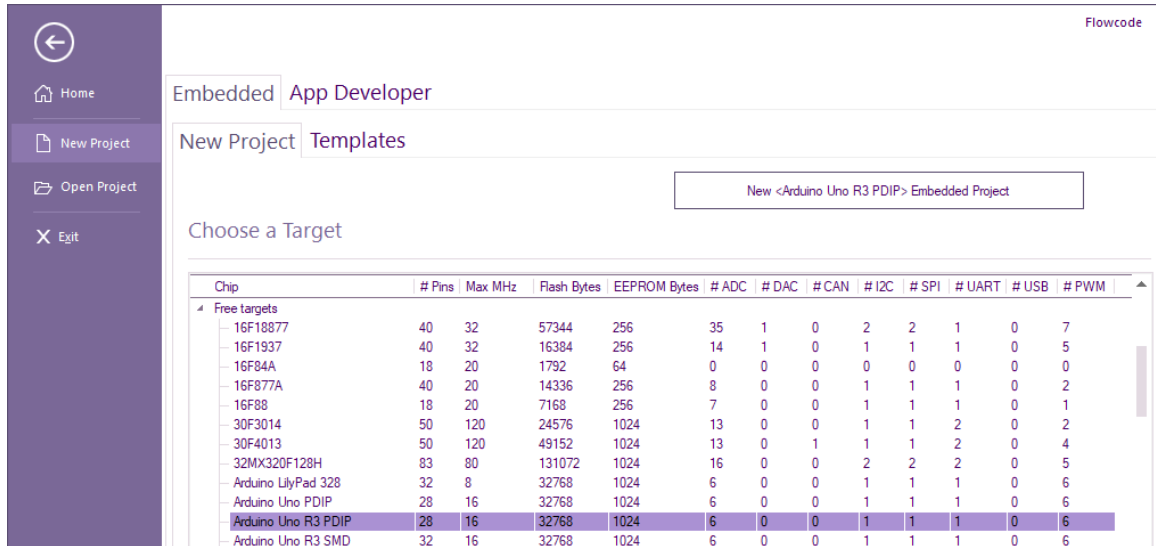
Select **Arduino Uno R3 PDIP** from the Free targets list. Click **"New <Arduino..."** button above

This brings up the standard Flowcode environment.

A flowchart can now be developed into a program that can be tested within the Flowcode simulation mode,   or be saved and compiled to the Arduino board.

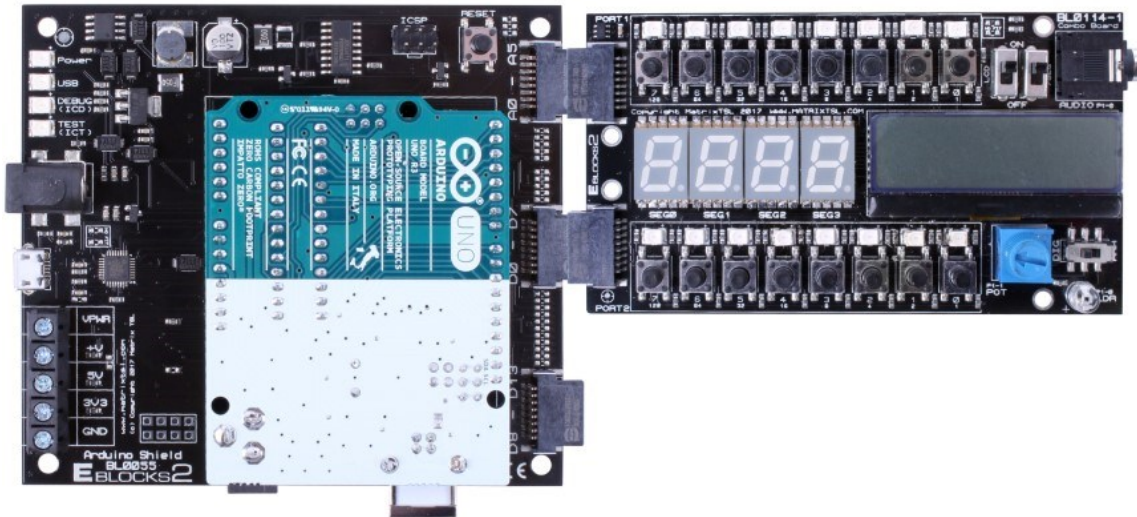Follow the Examples and Exercises, taking Port changes into consideration where required.
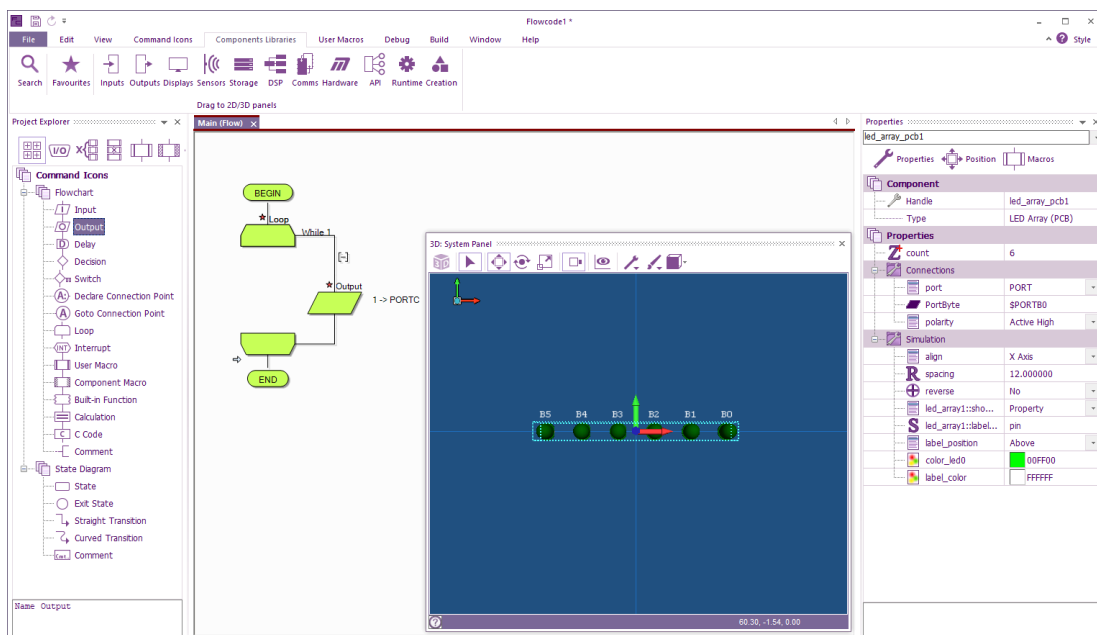
For example, the image above shows how **Flowcode First Program** (Page 42) would look to an Arduino user.

Here, Arduino users are using **PORTC** instead of **PORTA**.

(PORTC on the Arduino 'Maps' to PORTA of the Combo board)

Appendix 1

AEROSPACE

E-blocks2:

Eblocks2 uses the 'Click' boards for its SPI connections.

Using the BL0106 'Click' board E-block, you can put the board into the (D8-D13) port as shown in the picture above:

Appendix 1

ARDUINO: SECTION D

Setting up the hardware:

This diagram shows you how to set up the E-blocks hardware with Arduino.

Plug your Arduino into the L0055 board as shown, then connect the combo board into the ports labelled **(A0-A5)** and **(D0-D7)**.
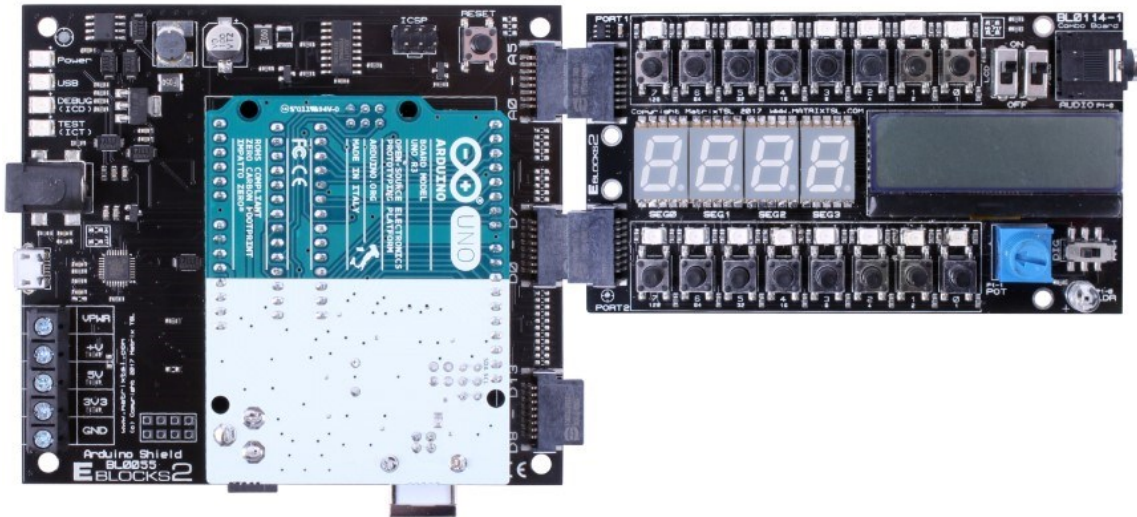
Note: Despite having two hardware port connections between the EB0114 Development board and the BL0055 Shield, the Arduino Uno can only provide 6 general purpose I/O connections on port C, (A0-A5).

Therefore, LEDs '6' and '7' and switches '6' and '7' on Port 1 of the Development board, cannot be used with the Arduino Uno.

In order to program the Arduino Uno board directly from within Flowcode, you must ensure that the appropriate drivers are installed. We recommend you visit the Arduino site and download the latest drivers from there.

Version Control

15 07 22   First beta

23 11 22   First release

28 07 23   Reformatting to new layout