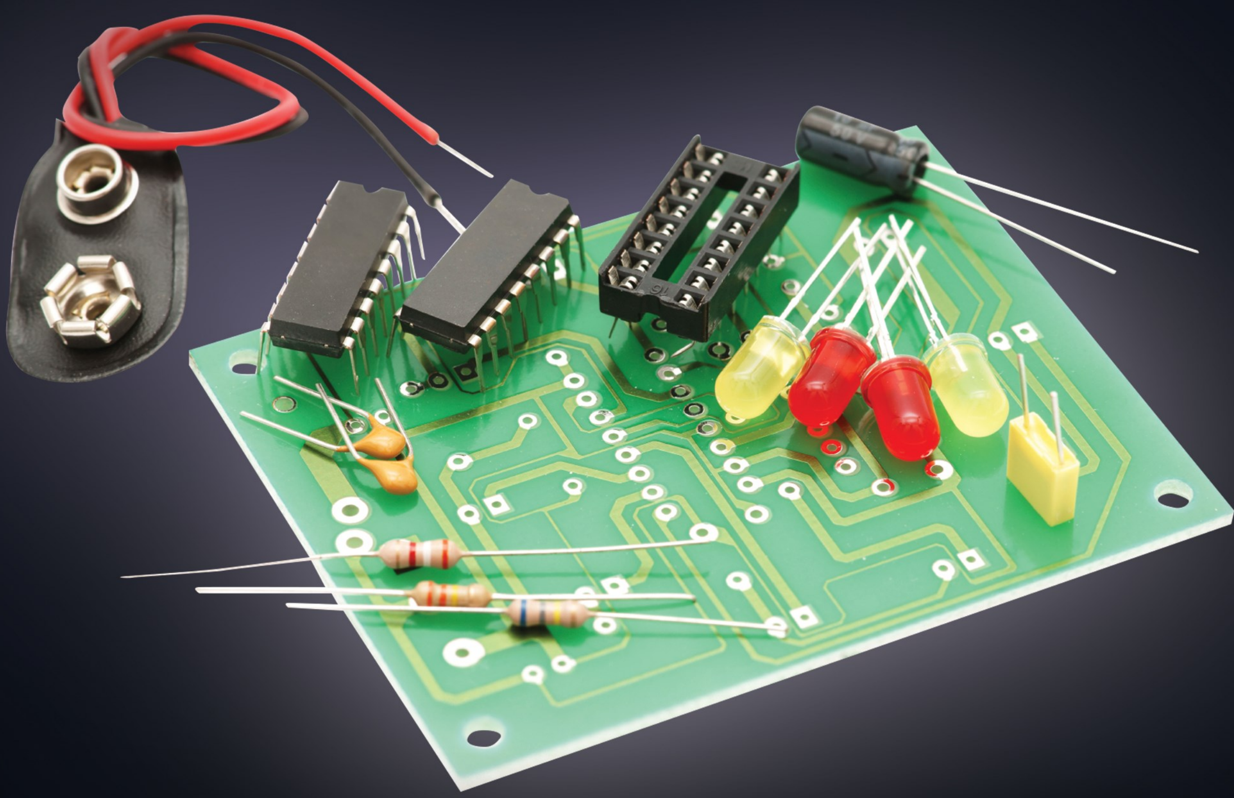
FLOWCODE6®

Simplifying Technology

Simple microcontroller circuits



SE3829

MATRIX

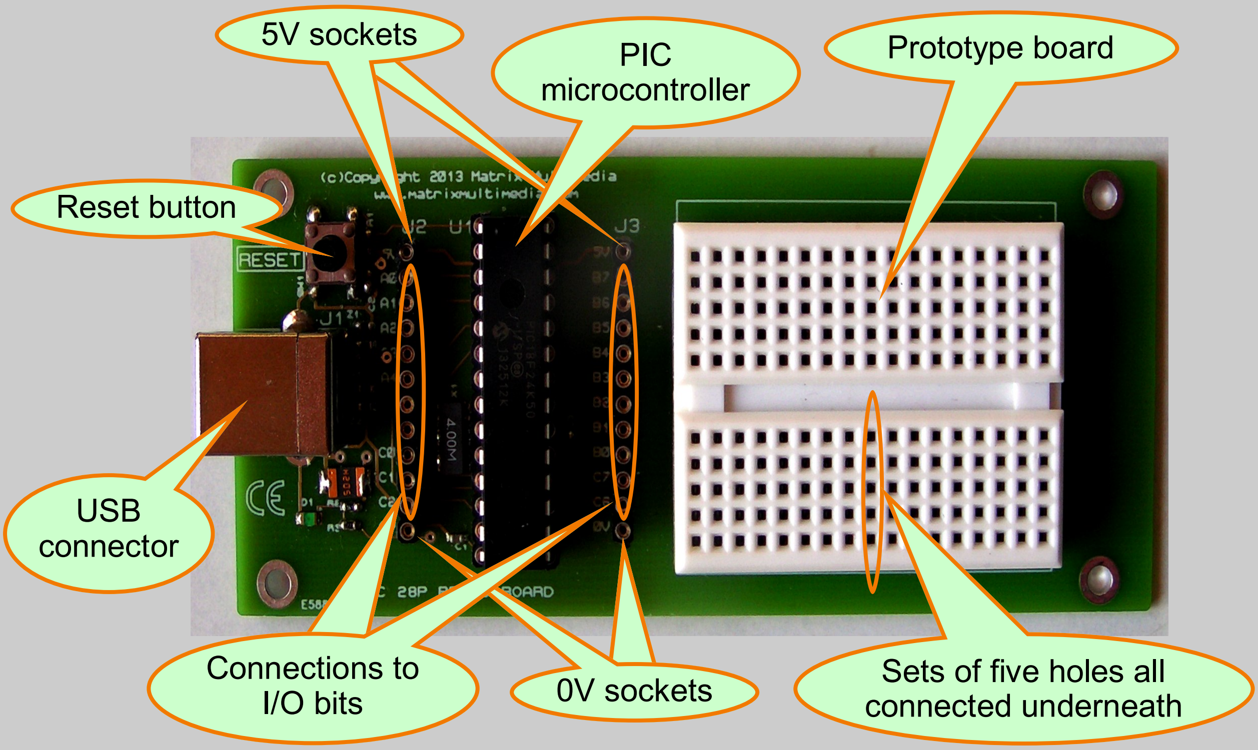
www.matrixmultimedia.com

Contents

Introduction

Simple microcontroller circuits

The hardware:



Some details:

- The USB connector is used to connect the board to a computer, so that the microcontroller can be programmed. The connection also provides electrical power for the board.
- The PIC microcontroller stays in its socket. It can be replaced, if damaged, using a chip extractor, or by gently easing up each end in turn using a flat-headed screwdriver.
- Once programming is completed, the Reset button must be pressed to start the program.
- The prototype board allows electronic components to be connected to the microcontroller, using 0.6mm wire plugged into the I/O (input / output) sockets. The holes in the prototype board are arranged in sets of five. These five are connected together by metal clips underneath the board.
- Electrical power can be provided through the USB cable, or via the 5V and 0V sockets, using 0.6mm wires.

Using the worksheets:

- The worksheets focus on car-based applications, as microcontrollers are very widely used in automotive systems. However, the examples have a much broader applicability.
- The Teacher Guide at the end of the module details all the steps needed to build the hardware, to program and to test the system. Please refer to these when necessary.

FLOWCODE 6

Worksheet 1
Switch on the LED

Simple microcontroller circuits

In today's world, we use microcontrollers more and more. These marvels of modern electronics are all-pervading, yet their use is so subtle that we often use them without realising it. Billions are sold every year!

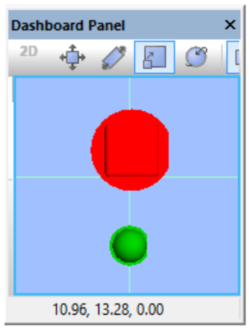
We use them in devices such as:

- the security token device used in on-line banking ;
- mobile phones, to set up and oversee our calls;
- microwave ovens to control the display and operation;
- engine management systems in our cars - increasingly we are 'driving-by-wire';
- digital cameras to control the focussing and shutter control;
- television remote controls.

This worksheet introduces a very simple application of a microcontroller - to turn on the car's courtesy light when the door switch is operated.

Over to you:

The program:

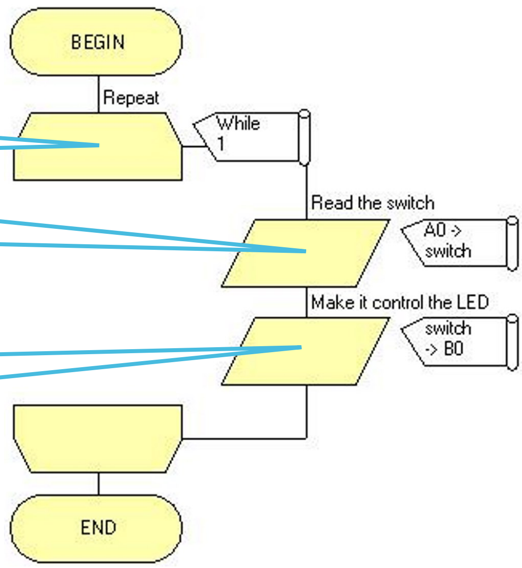


- Build the Flowcode flowchart, adding a 'Switch (Push, PCB) ' and a 'LED 5mm PCB' to the dashboard panel.
- Connect the switch to PORT A bit 0, and the LED to Port B bit 0.
- Open 'Project Explorer' using the 'View' menu.
- Click on the down arrow to the left of the 'Variables' label, and select the 'Add new' option.
- Name the new variable 'switch', and click on 'OK'.
- Configure the components as follows:

Display name	Repeat
Loop while	1
Test Loop at the	Start

Display name	Read the switch
Variable	switch
Port	PORT A
Input from	Single Bit 0

Display name	Make it control the LED
Variable	switch
Port	PORT B
Output to	Single Bit 0



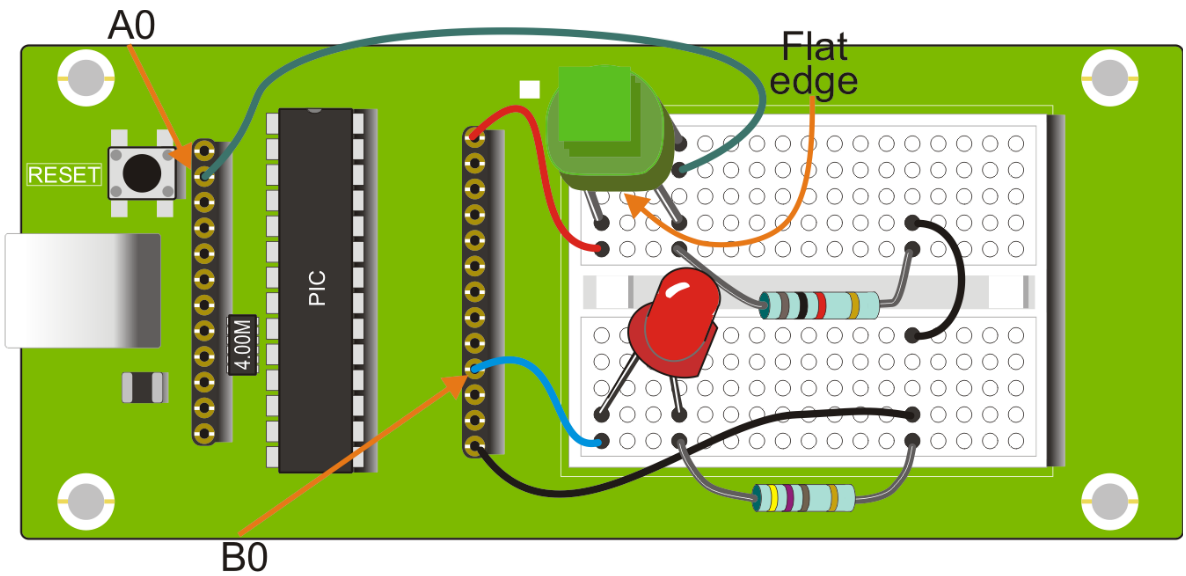
- Save the Flowcode program.
- Simulate it to check that it works.
  'Clicking' the switch should control the LED.
- Now, compile it to the chip.
- Build the circuit shown on the next page.

Worksheet 1
Switch on the LED

Over to you:

The circuit:

- Build the circuit, shown below, on the prototype board section of the hardware.
    - Notice the power supply connections to the prototype board! The red wire connects the switch to the +5V socket. The switch is then connected through the resistor to 0V - a ‘bridge’ of black wire connects the resistor to the long black wire plugged into 0V.
    - The switch forms a voltage divider with the 1kΩ resistor (brown / black / red). The flat edge of the switch is on the side indicated in the diagram. Use this to ensure that it is connected correctly.
    - The LED represents the car’s sidelight bulb. Make sure that it is plugged in the right way round! Look for the flat edge on the ‘skirt’ of the LED, or the shorter of the two legs. The short leg is connected through the resistor to 0V.
    - The LED is protected by a 470Ω resistor (yellow / purple / brown).
- Test the circuit by pressing the switch, and then releasing it. You should find that it switches the LED on and off.



What to do next:

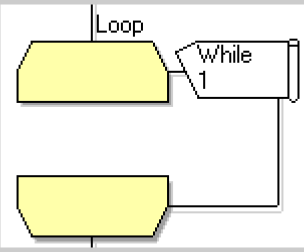
Modify the program so that the switch lights two sidelights, represented by two LEDs, one connected to Port B bit 0 and the other to Port B bit 1.

FLOWCODE 6

Worksheet 2

Make the LED flash

Simple microcontroller circuits



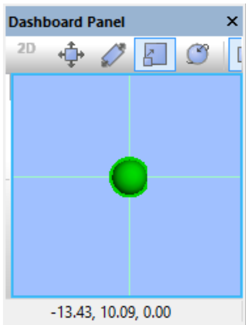
The 'loop' function appears in all the worksheets. It makes the program cycle through repeatedly.

For us, it allows us to observe what happens more easily, as each of the steps in the program lasts only microseconds.

Over to you:

The program:

- Build the Flowcode flowchart, adding a 'LED (5mm, PCB)' to the dashboard panel.
- Connect the LED to PORT B bit 0.
- Configure the components as follows:

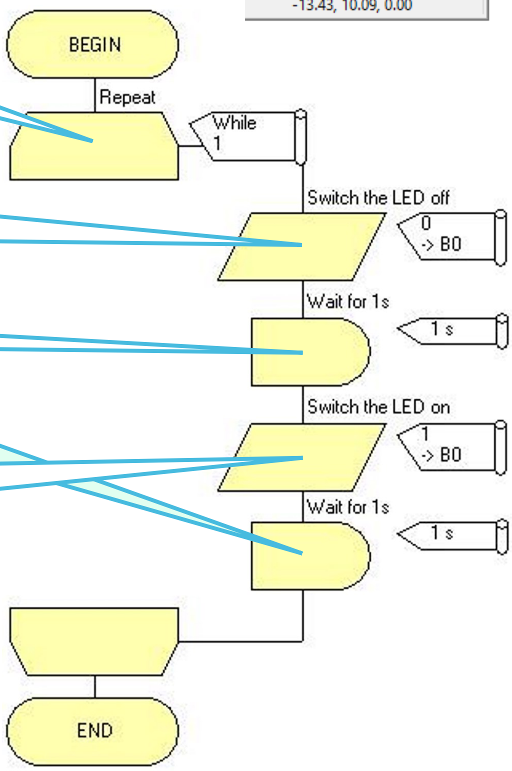


Display name Repeat
Loop while 1
Test Loop at the Start

Display name Switch the LED off
Value 0
Port PORT B
Output to Single Bit 0

Display name Wait for 1s
Delay value 1 second

Display name Switch the LED on
Value 1
Port PORT B
Output to Single Bit 0



- Save the Flowcode program.
- Simulate it to check that it works. The LED should flash on and off repeatedly.
- Now, compile it to the chip.
- Build the circuit shown on the next page.

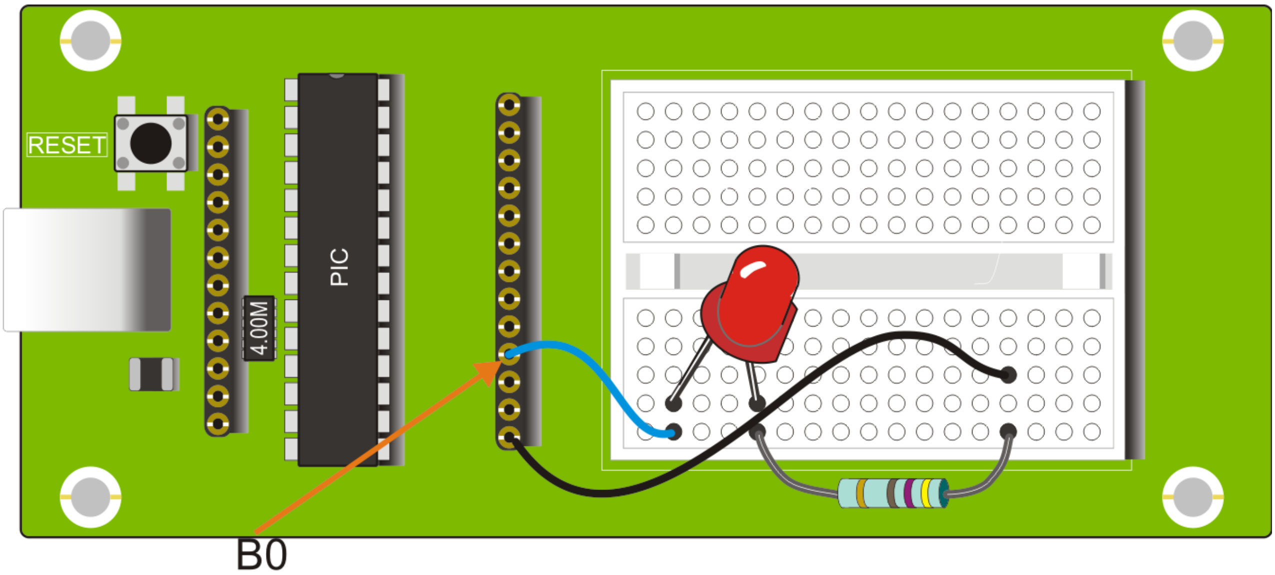
Worksheet 2

Make the LED flash

Over to you:

The circuit:

- Build the circuit, shown below, on the prototype board.
  - Make sure that the LED, representing the car indicator lamp, is plugged in the right way round! If unsure how to do this, look back at the details on the previous worksheet.
    The LED is protected by a 470Ω resistor (yellow / purple / brown).
  - Notice that there is no connection to the 5V socket this time. The LED is powered from the output of the microcontroller. The resistor is connected to 0V by the long black wire, completing the circuit.
- Test the circuit.
  The LED should flash on and off continuously - on for a second, then off for a second.



What to do next:

Modify the program so that the LED flashes twice as fast.

Modify it so that the LED flashes only ten times.

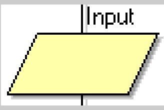
FLOWCODE 6

Worksheet 3

Keep the LED lit for a short time

Simple microcontroller circuits

The microcontroller can respond to changes in its surroundings.



To do this, it takes in information via ‘Input’ functions. In this case, it monitors the state of a switch, connected to Port A, bit 0. The information from the switch (‘on’ or ‘off’) is stored in a variable called ‘switch’.

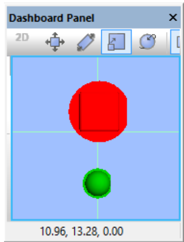
(The User Guide at the back of the module gives more information about using variables.

This worksheet shows how to set up another common application - a lamp that stays on for a short time when triggered - a car courtesy light, for example.

Over to you:

The program:

- Build the flowchart, adding a ‘Switch (Push, PCB)’ and a ‘LED (5mm, PCB)’ to the dashboard panel.
- Connect the switch to PORT A bit 0, and the LED to Port B bit 0.
- Use ‘Project Explorer to create a new byte variable called ‘switch’.
- Configure the Loop icon as before.
- Set up the other icons as follows:



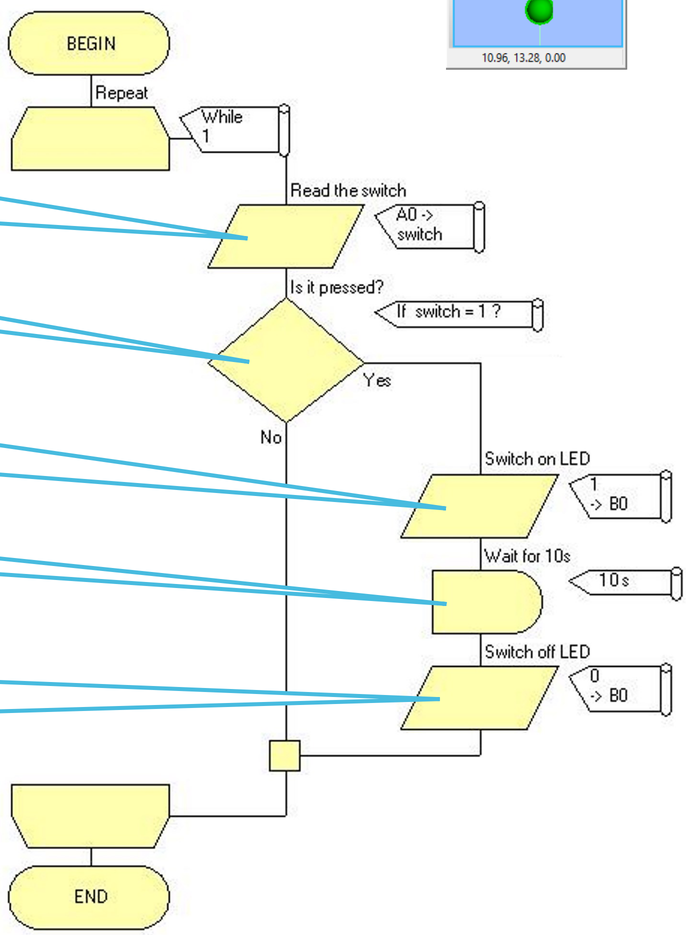
Display name	Read the switch
Variable	switch
Port	PORT A
Input from	Single Bit 0

Display name	Is it pressed?
If	switch=1

Display name	Switch on LED
Value	1
Port	PORT B
Output to	Single Bit 0

Display name	Wait for 10 s
Delay value	10 seconds

Display name	Switch off LED
Value	0
Port	PORT B
Output to	Single Bit 0



- Save the Flowcode program.
- Simulate it to check that it works. The LED switches on for ten seconds when the switch is pressed.
- Now, compile it to the chip.
- Build the circuit shown on the next page.

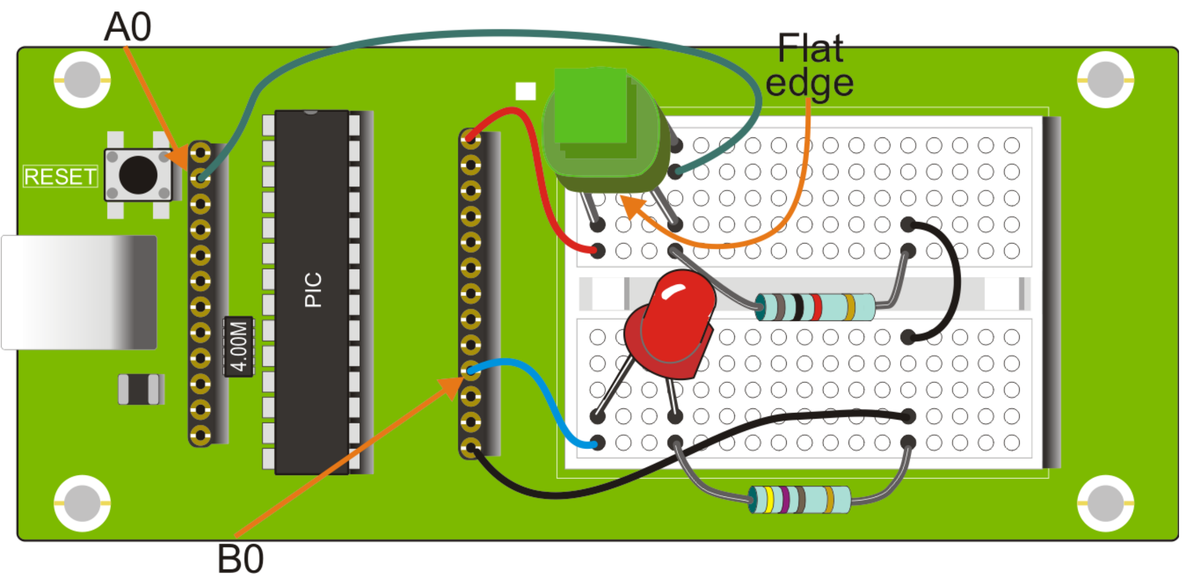
Worksheet 3

Keep the LED lit for a short time

Over to you:

The circuit:

- Build the circuit, shown below, on the prototype board.
    - It is identical to the one you built for worksheet 1. Notice the wire that takes information from the switch unit, and inputs it into Port A, bit 0 of the microcontroller!
    - The switch represents a car door switch. It forms a voltage divider with the 1kΩ resistor (brown / black / red).
    - The LED represents the car's courtesy light. Make sure that it is plugged in the right way round!
      It is protected by a 470Ω resistor, (yellow / purple / brown).
- Test the circuit by pressing and releasing the switch. The LED should light for ten seconds and then turn off.



What to do next:

Modify the program so that the LED stays on for 5 seconds when the switch is pressed.

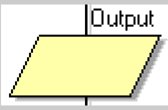
FLOWCODE 6

Worksheet 4

Set up a latch

Simple microcontroller circuits

In this application, the microcontroller turns on an alarm under the right conditions.



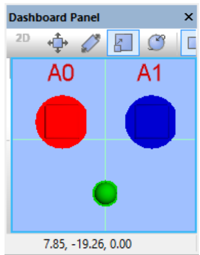
Then, the program uses an ‘Output’ function to provide power to one of its output connections.

This worksheet shows how to set up a latch.
It uses two switches but each has a specific function.
One switch turns on (sets) the alarm, an LED. in this system, and the other then turns it off (resets it.)

Over to you:

The program:

- Build the flowchart.
- Add two ‘PCB Switches’ and a ‘LED 5mm Unmounted’ to the dashboard panel.
- Connect one switch to PORT A bit 0, the second to PORT A bit 1, and the LED to Port B bit 0. Add labels to identify the switches.
- Use ‘Project Explorer to create a new byte variable called ‘switch’.
- Configure the Loop and first Input icon as before.
- Set up the other components as follows:



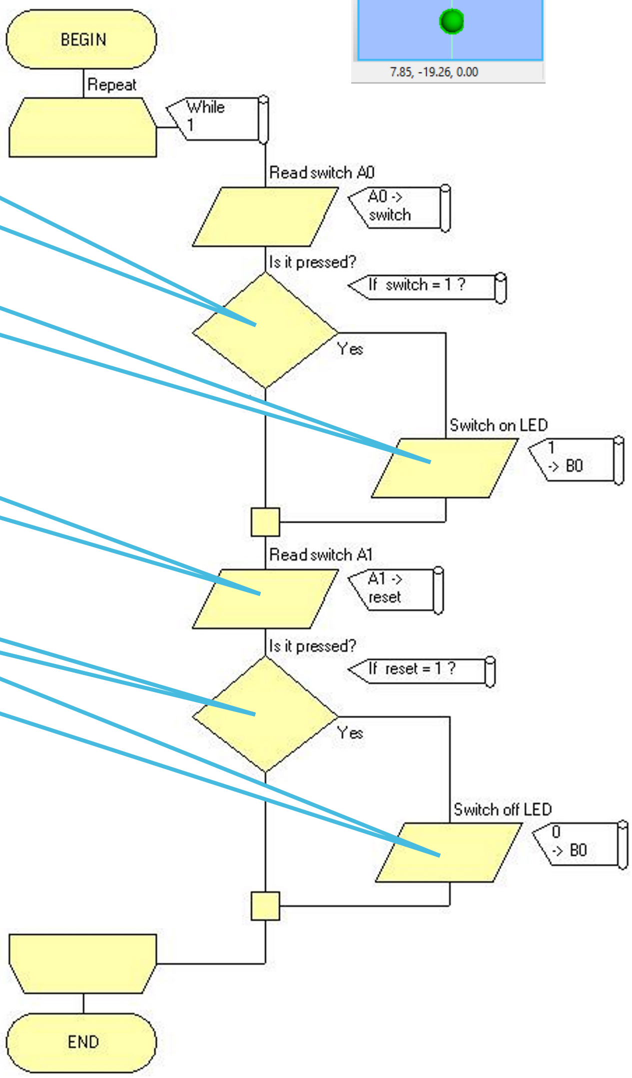
Display name	Is it pressed?
If	switch=1

Display name	Switch on LED
Value	1
Port	PORT B
Output to	Single Bit 0

Display name	Read switch A1
Variable	reset
Port	PORT A
Input from	Single Bit 1

Display name	Is it pressed?
If	reset=1

Display name	Switch off LED
Value	0
Port	PORT B
Output to	Single Bit 0



- Save the Flowcode program.
- Simulate it to check that it works. Switch A0 is known as the ‘Set’ switch, and A1 the ‘Reset’ switch.
- Now, compile it to the chip.
- Build the circuit shown on the next page.

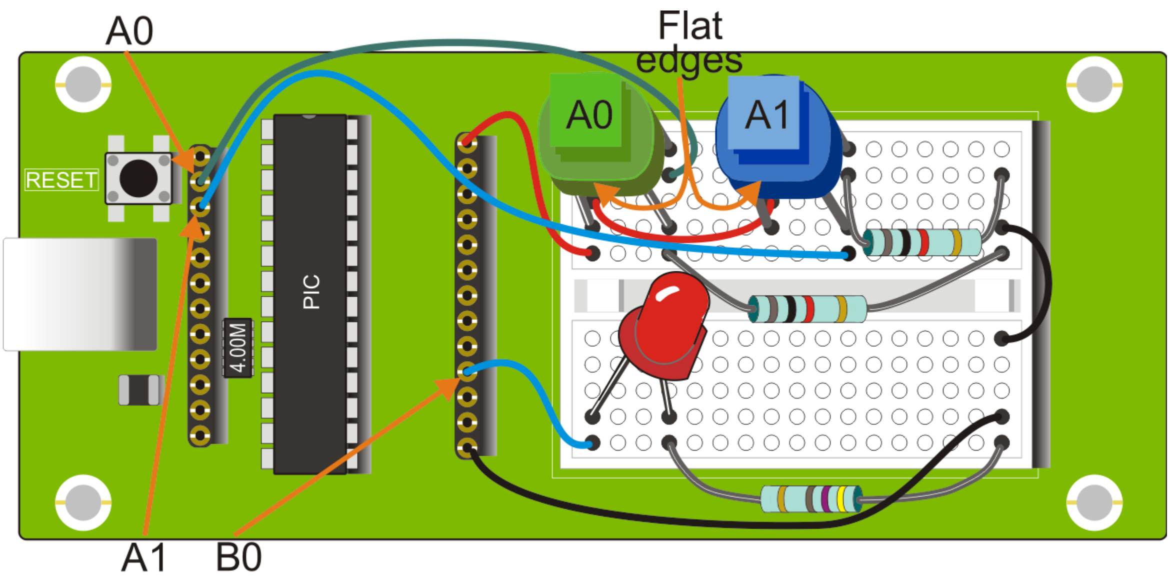
Worksheet 4

Set up a latch

Over to you:

The circuit:

- Build the circuit, shown below, on the prototype board.
    - The LED represents the 'alarm active' warning light. Make sure that it is plugged in the right way round!
      It is protected by a 470Ω resistor, (yellow / purple / brown).
    - Again, the switches form voltage dividers with the 1kΩ resistors (brown / black / red).
      The flat edges of the switches are indicated in the diagram.
      Use this information to connect the switches correctly.
- Test it by pressing switch A0 (the 'Set' switch,) and then switch A1(the 'Reset' switch).
    - Does it matter how many times you press switch A0?
    - Does it matter how many times you press switch A1?



What to do next:

Modify the program so that the LED is turned on *until* switch A0 is pressed, and then lights again when switch A1 is pressed.

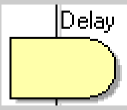
FLOWCODE 6

Worksheet 5

Set up a lighting sequence

Simple microcontroller circuits

Control systems often create sequences of operation, whether controlling motors, valves etc.

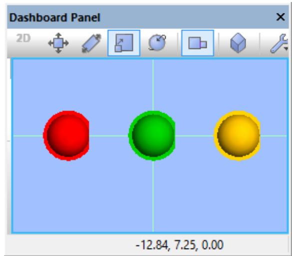


In this worksheet, the microcontroller creates a simple light sequence. The program uses delay functions to allow each stage to remain long enough to be seen.

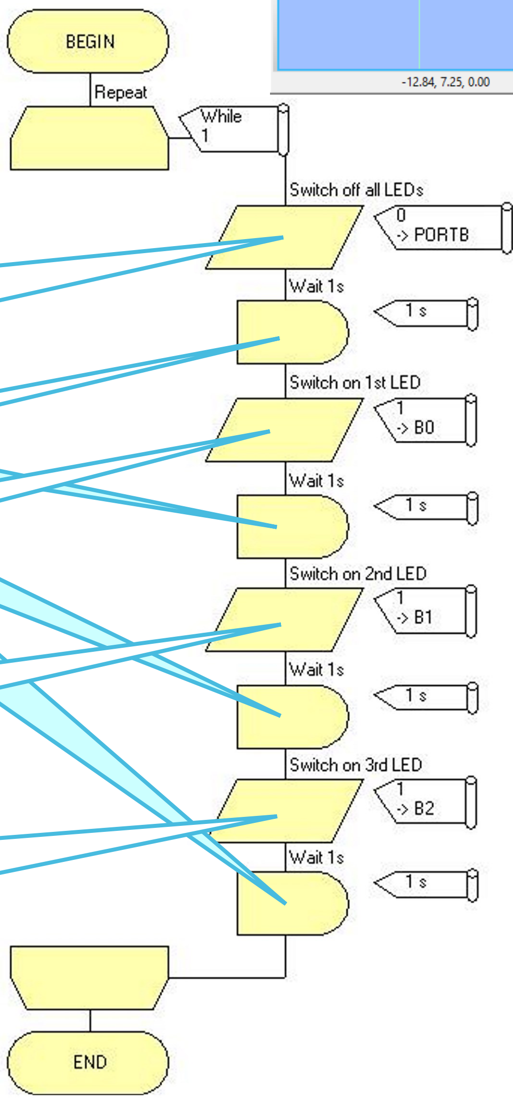
One example of a lighting sequence is that provided by traffic lights, though these are more sophisticated than the simple LEDs used here.

Over to you:

The program:



- Build the flowchart, adding three 'LED 5mm Unmounted's to the dashboard panel.
- Connect one LED to PORT B bit 0, the second to PORT B bit 1, and the third to Port B bit 2.
- Configure the Loop icon as in previous programs, and set up the other components as follows:



Display name	Switch off all LEDs
Value	0
Port	PORT B
Output to	Entire Port

Display name	Delay
Delay value	1 second

Display name	Switch on 1st LED
Value	1
Port	PORT B
Output to	Single Bit 0

Display name	Switch on 2nd LED
Value	1
Port	PORT B
Output to	Single Bit 1

Display name	Switch on 3rd LED
Value	1
Port	PORT B
Output to	Single Bit 2

- Save the Flowcode program.
- Simulate it to check the LEDs light in sequence.
- Now, compile it to the chip.
- Build the circuit shown on the next page.

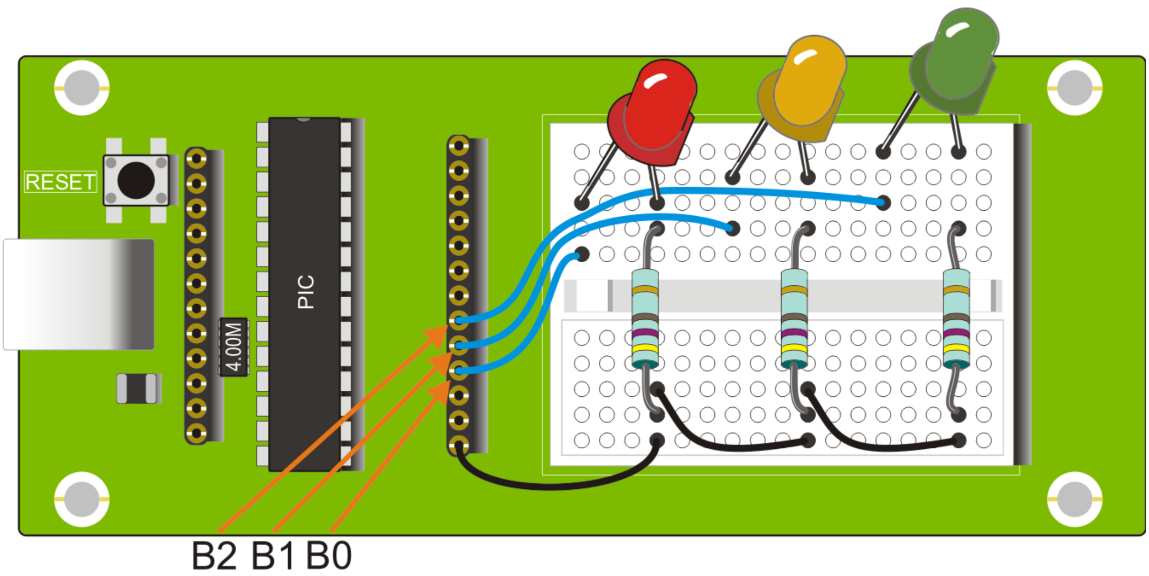
Worksheet 5
Set up a lighting sequence

Simple microcontroller circuits

Over to you:

The circuit:

- Build the circuit, shown below, on the prototype board.
    - The three LEDs are powered from separate output connections on Port B of the microcontroller , called ‘bit 0’ (B0), ‘bit 1’ (B1) and ‘bit 2’ (B2).
      Notice the black wires that make up the 0V connection to these LEDs!
    - Make sure that the LEDs are plugged in the right way round!
      Each LED is protected by a 470Ω resistor, (yellow / purple / brown).
- Test the circuit by pressing the Reset button and watching the sequence that follows.



What to do next:

Modify the program so that a switch must be pressed to make the sequence start.

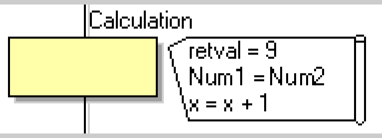
FLOWCODE 6

Worksheet 6

The AND gate

Simple microcontroller circuits

Microcontrollers can take in information from the outside world and make decisions based on it.



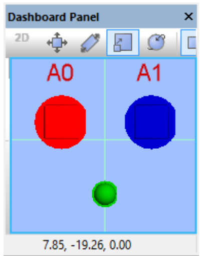
Part of this can involve using calculation functions, which perform arithmetic and, as in this case, logic operations. Several separate calculations can be included in the same icon, as the illustration shows.

In this worksheet, the car headlamps come on only if the ignition switch AND the headlamp switch are turned on.

Over to you:

The program:

- Build the flowchart, adding two switches and a LED to the dashboard panel.
- Connect one switch to PORT A bit 0, the second to PORT A bit 1, and the LED to Port B bit 0. Add labels to identify the switches.
- To make simulation easier, turn the switch action from ‘momentary’ to ‘latching’.
  - Right-click on each switch and select ‘Properties’.
  - Then click on the down arrow at the right-hand end of the ‘Operation’ row.
  - Select ‘Latching’.
- Use ‘Project Explorer to create new variables called ‘inA’, ‘inB’ and ‘result’.
- Configure the Loop icon as in previous programs, and set up the other components as follows:

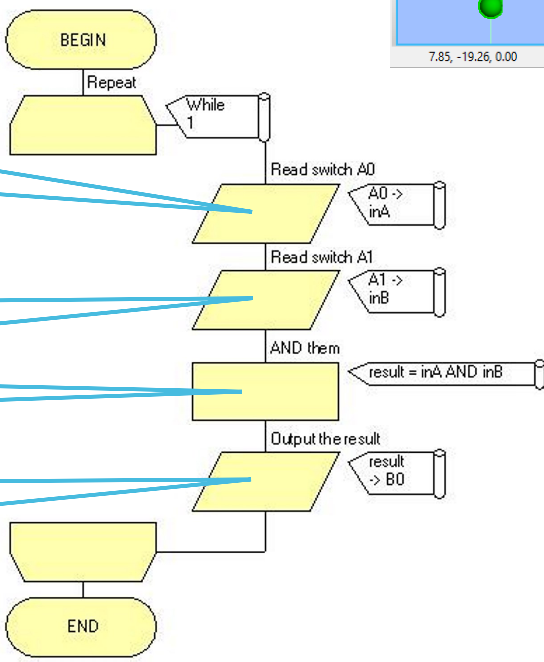


Display name	Read switch A0
Variable	inA
Port	PORT A
Input from	Single Bit 0

Display name	Read switch A1
Variable	inB
Port	PORT A
Input from	Single Bit 1

Display name	AND them
Calculation	result = inA AND inB

Display name	Output the result
Variable	result
Port	PORT B
Output to	Single Bit 0



- Save the Flowcode program.
- Simulate it to check that it works - the LED lights only when both switches are pressed.
- Now, compile it to the chip and then build the circuit shown on the next page.

Worksheet 6
The AND gate

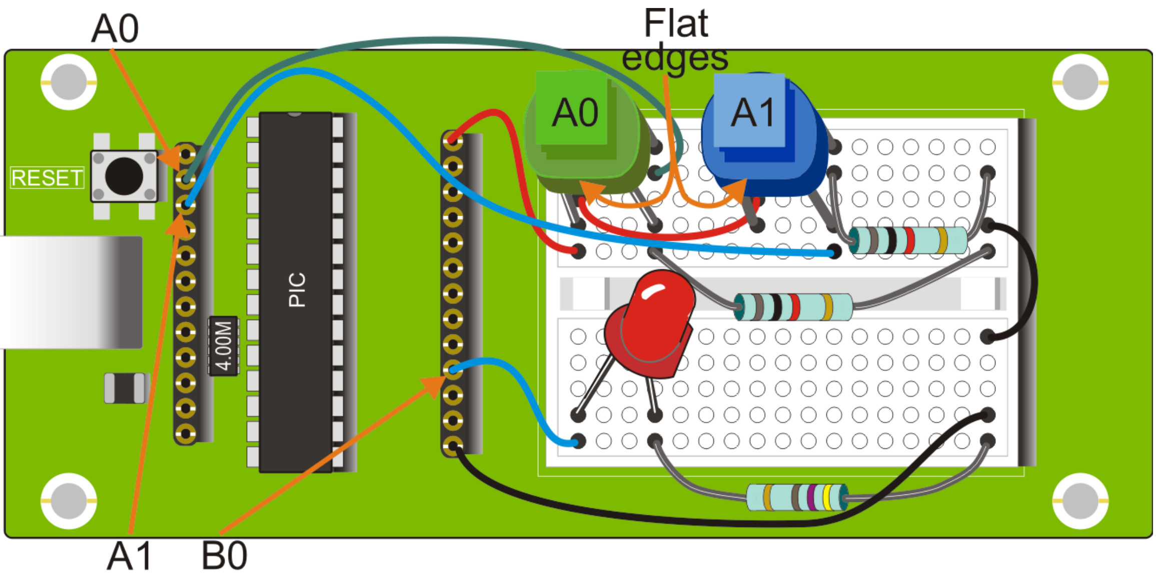
Over to you:

The circuit:

- Build the circuit, shown below, on the prototype board.
    - Switch A0 represents the ignition switch, and A1 the headlamp switch. The switches form voltage dividers with the 1kΩ resistors, (brown / black / red).

      Notice the red wires that connect the switches to the +5V supply, and the wires that take information from the switch units to the microcontroller via sockets A0 and A1.
    - The LED represents the headlamp. Make sure that it is plugged in the right way round!

      It is protected by a 470Ω resistor, (yellow / purple / brown).
- Test the circuit by pressing one or both switches in various combinations. The LED should light only when both switches are pressed.



What to do next:

Modify the program to set up an OR gate, where the LED lights if either switch A0 OR A1 OR both is/are pressed. This arrangement could be used to turn on the car's courtesy light when either front door is opened.

FLOWCODE 6

Worksheet 7
A logic system

Simple microcontroller circuits

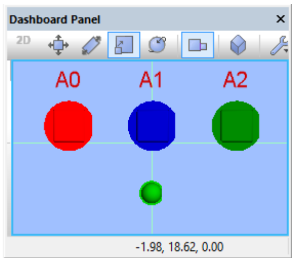
The previous worksheet showed how to set up a straightforward logic function, the AND gate. Microcontrollers can do far more powerful tasks.

This time, the car seat-belt warning sign will light when the ignition switch is turned on, someone is sitting on the passenger seat (and the pressure switch inside it,) but has **not** fastened the seat belt.

Over to you:

The program:

- Build the flowchart to light the LED only when switches A0 AND A1 are pressed AND switch A2 is NOT pressed.
- Add three ‘PCB Switches’ and a ‘LED 5mm Unmounted’ to the dashboard panel.
- Connect one switch to PORT A bit 0, the second to PORT A bit 1, the third to PORT A bit 2 and the LED to Port B bit 0.
- Add labels to identify the switches.
- Use ‘Project Explorer to create new variables called ‘inA’, ‘inB’, ’inC’ and ‘result’.
- Configure the Loop icon as in previous programs, and set up the others as follows:

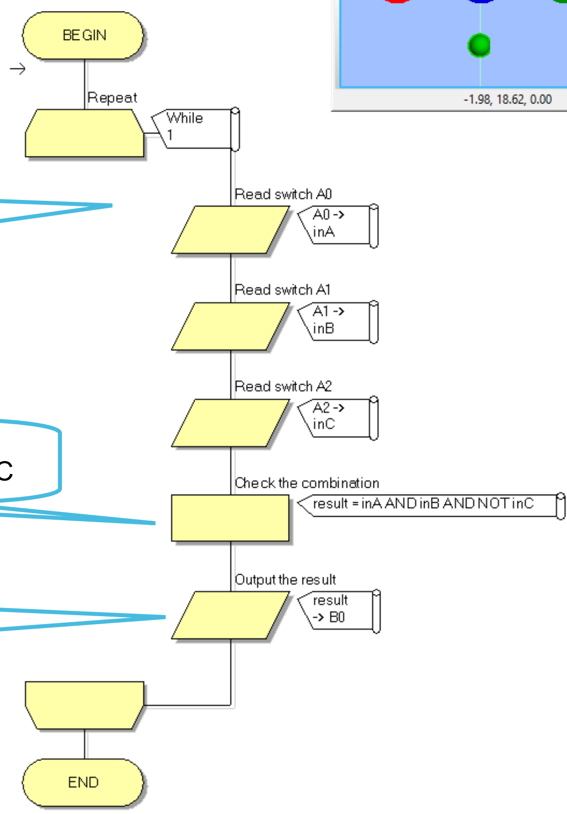


Display name	Read switch A0
Variable	inA
Port	PORT A
Input from	Single Bit 0

(Set up the other Input icons in the same way so that the variable ‘inB’ contains the state of switch A1, and ‘inC’ that of A2.)

Display name	Check the combination
Calculation	result = inA AND inB AND NOT inC

Display name	Output the result
Variable	result
Port	PORT B
Output to	Single Bit 0



- Save the Flowcode program.
- Simulate it to check that it works.
  The LED lights only when A0 AND A1 are pressed, AND A2 is NOT pressed.
- Now, compile it to the chip, and build the circuit shown on the next page.

Worksheet 7

A logic system

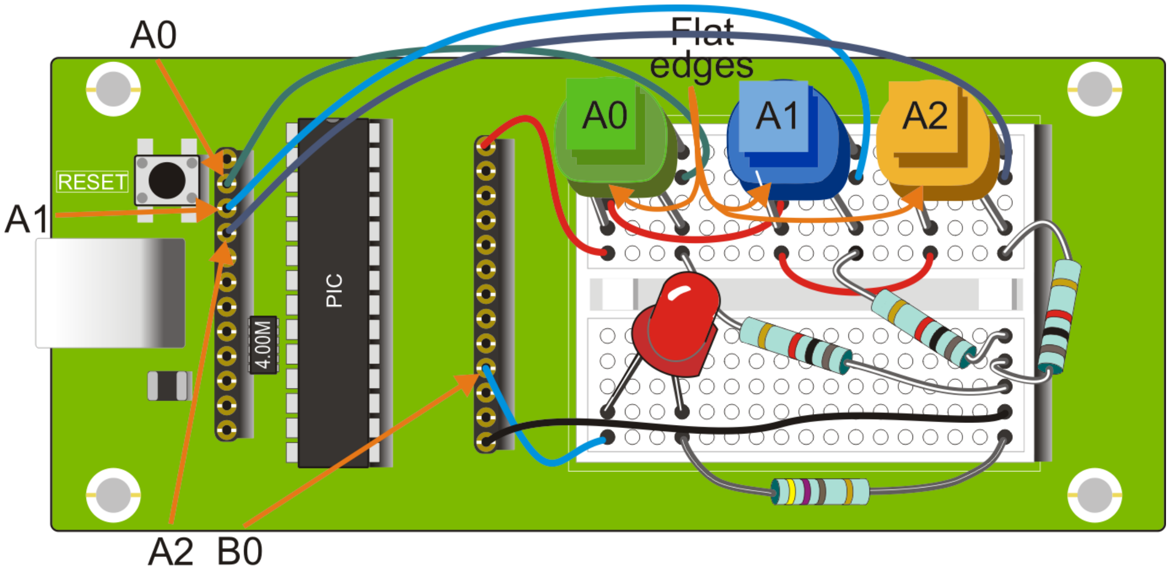
Over to you:

The circuit:

- Build the circuit, shown below, on the prototype board.
    - This time there are three switches. Switch A0 represents the ignition switch, A1 the pressure switch, inside the passenger seat, and A2 the seat-belt switch that indicates when the belt is fastened.

      They form voltage dividers with the 1kΩ resistors, (brown / black / red).
      Notice the three red wires that connect the switch units to the +5V supply, and the three wires that take information from the switch units to the microcontroller via sockets A0, A1 and A2.
    - The LED represents the seat belt warning light. Make sure that it is plugged in the right way round!

      It is protected by a 470Ω resistor, (yellow / purple / brown).
- Test the circuit by pressing the switches in various combinations. The LED should light only when switches A0 and A1 are pressed, (ignition on, and a passenger present,) and A2 is not pressed (seat belt not fastened.)



What to do next:

Modify the program so that the LED lights if switch A1 is pressed provided that switches A0 and A2 are NOT pressed.

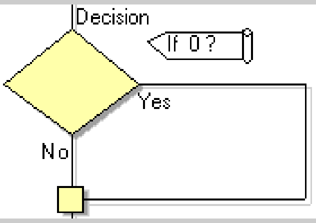
FLOWCODE 6

Worksheet 8

Is it too dark?

Simple microcontroller circuits

Many cars now have automatic headlights that switch on when the light-level outside falls too far. That is another decision that the microcontroller can make, using information from a light-sensing unit.



It involves use of a decision element in the program. In its simplest form, as used here, it makes a Yes/No decision. Is it dark, or not? It makes

Over to you:

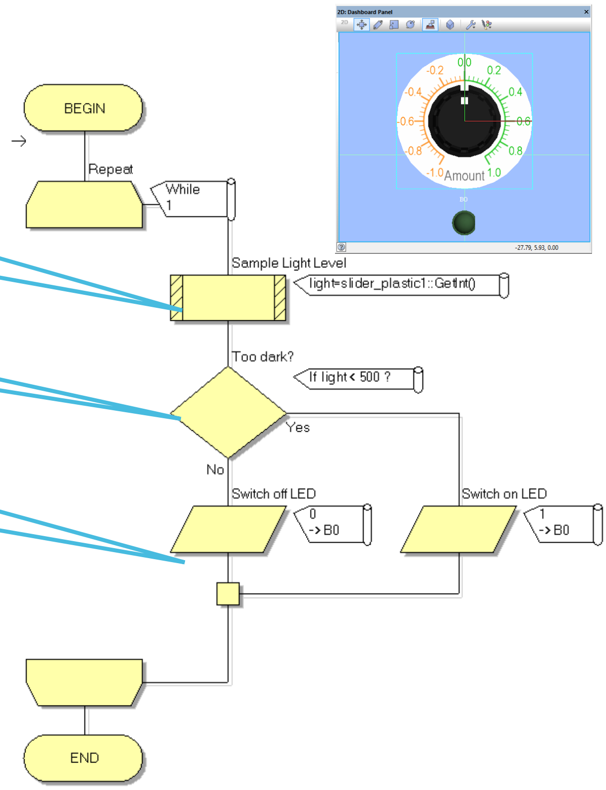
The program:

- Build the flowchart to turn on the LED only when little light shines on the LDR.
- Add a ‘Potentiometer (Slider)’ and a ‘LED (5mm PCB)’ to the dashboard panel.
- The potentiometer represents the light-sensing unit. During simulation, moving the slider represents changing light levels on the LDR. Connect it to channel An 0.
- Connect the LED to Port B bit 0.
- Use ‘Project Explorer to create a new ***integer*** variable called ‘light’.
- Configure the Loop icon as usual.
- Set up the other components as follows:

Display name	Sample light level
Component	slider_plastic1
Macro	GetInt
Return Value	light (Int variable)

Display name	Too dark?
If:	light<500

Display name	Switch off LED
Value	0
Port	PORT B
Output to	Single Bit 0



(Set up the other Output icon to output a value of 1 to Port B0, to turn on the LED.)

Save the Flowcode program.

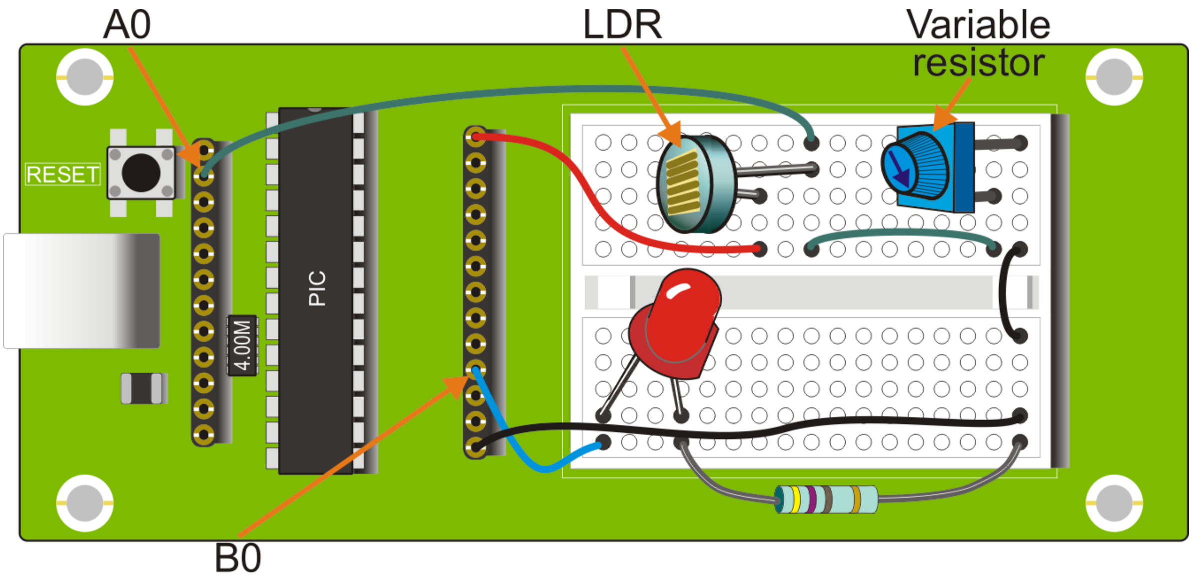
- Simulate it to check that it works. Move the slider on the potentiometer to check that it controls the state of the LED.
- Now, compile it to the chip, and build the circuit shown on the next page.

Worksheet 8
Is it too dark?

Over to you:

The circuit:

- Build the circuit, shown below, on the prototype board.
  - The LDR, (light-dependent resistor,) can be connected either way round. Be careful not to snap off the legs through excessive bending.
    It forms a voltage divider with the variable resistor, made from a 'pot' (potentiometer.) The three legs of the pot each go into a different row of the prototype board, even though only two are connected to the circuit.
  - The LED represents the headlight. Make sure that it is plugged in the right way round!
    It is protected by a 470Ω resistor, (yellow / purple / brown).
- Test the circuit by shading the LDR with your hand. When dark enough, the microcontroller will switch on the LED.
  (You may have to adjust the variable resistor to make this happen.)



What to do next:

Modify the program so that the LED operates the other way round - turns on in bright light, and off when it gets dark.

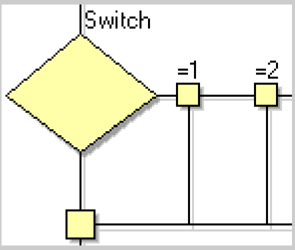
FLOWCODE 6

Worksheet 9

A tone generator

Simple microcontroller circuits

In this worksheet, a rapidly changing signal is fed to a 'sounder', a solid-state device which



vibrates when it receives such a signal. The result can be an audible sound.

The application this time is a two-tone horn, where the note produced depends on which of the

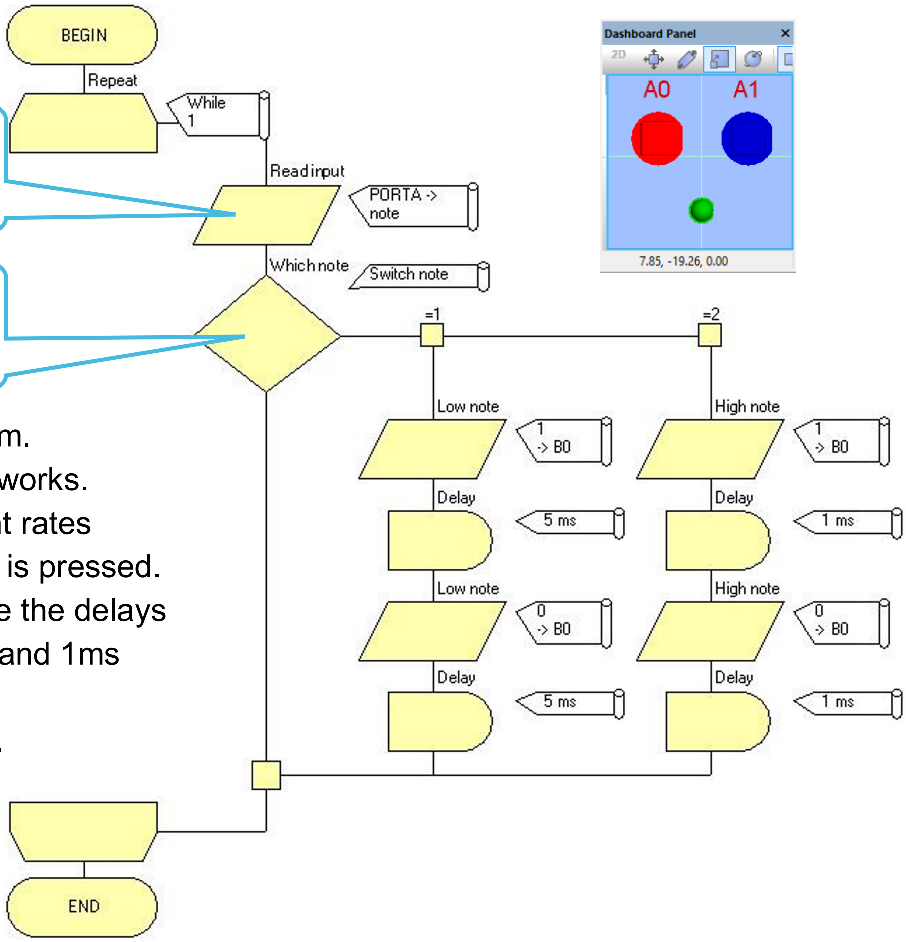
Over to you:

The program:

- Build the flowchart adding two 'PCB Switches' and a 'LED 5mm Unmounted' to the dashboard panel. Notice the use of the 'Switch' icon to allow choice of notes. (The LED represents the sounder in simulation.)
- Connect one switch to PORT A bit 0, the second to PORT A bit 1, and the LED to Port B bit 0. Add labels to identify the switches.
- Use 'Project Explorer to create a new byte variable called 'note'.
- Configure the Loop icon, Output icons and the Delay icons as before.
- Set up the other icons as follows:

Display name	Read input
Variable	note
Port	PORT A
Input from	Entire Port

Display name	Which note?
Switch	note
Cases	☑1
	☑2



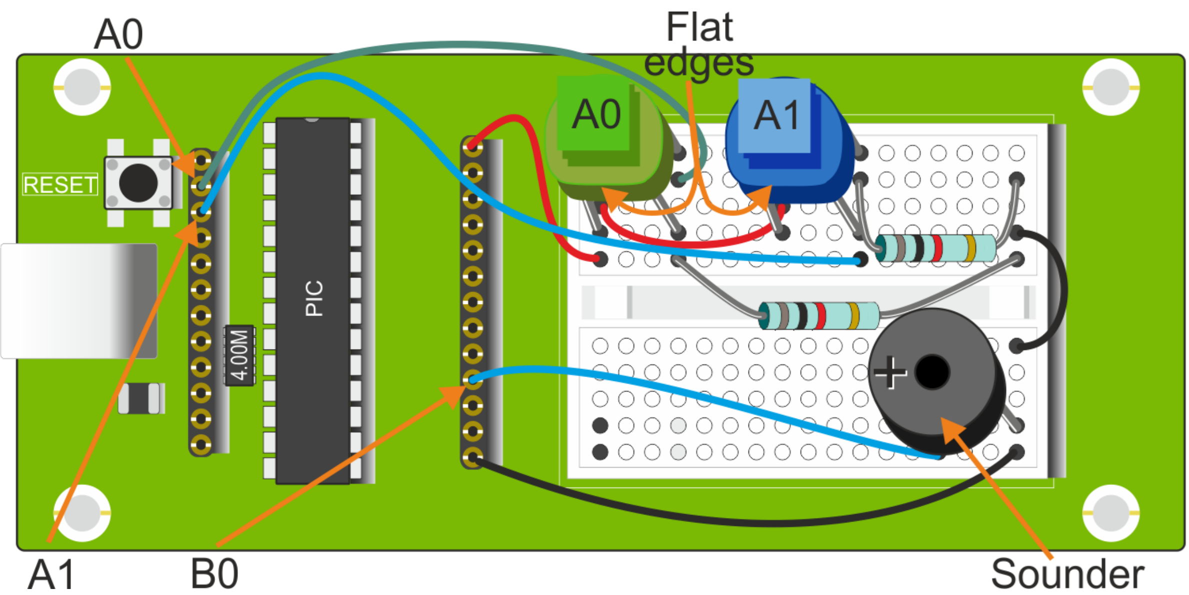
- Save the Flowcode program.
- Simulate it to check that it works. The LED flashes at different rates depending on which switch is pressed. If this is not obvious, reduce the delays - replace 5ms with 500ms, and 1ms with 100ms.
- Now, compile it to the chip.
- Build the circuit shown on the next page

Worksheet 9
A tone generator

Over to you:

The circuit:

- Build the circuit, shown below, on the prototype board.
    - The sounder represents the car horn. It must be connected the right way round, with the longer leg (+) nearer to the positive power supply (from socket B0 in this case,) and the shorter leg connected to 0V.
    - Switches A0 and A1 are used to select the two tones. They form voltage dividers with the 1kΩ resistors, (brown / black / red). Notice the red wires that connect the switch units to the +5V supply, and the wires that take information from the switch units to the microcontroller via sockets A0 and A1.
- Test the circuit by pressing each switch in turn and noticing the tones that are produced.



What to do next:

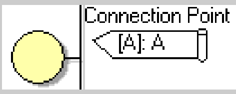
Modify the program so that it produces a third note when both switches are pressed at the same time.

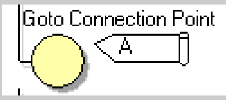
FLOWCODE 6

Worksheet 10
Counting

Simple microcontroller circuits

In modern vehicle engine management, one important factor is the engine speed. An electronic control unit counts pulses generated by a sensor attached to the rotating engine. The result is displayed on the 'rev counter, also called tachometer.



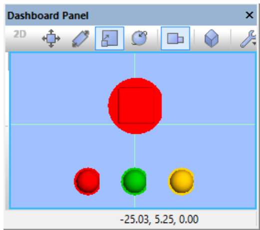


This worksheet illustrates the principle behind this. It uses 'Connection points'

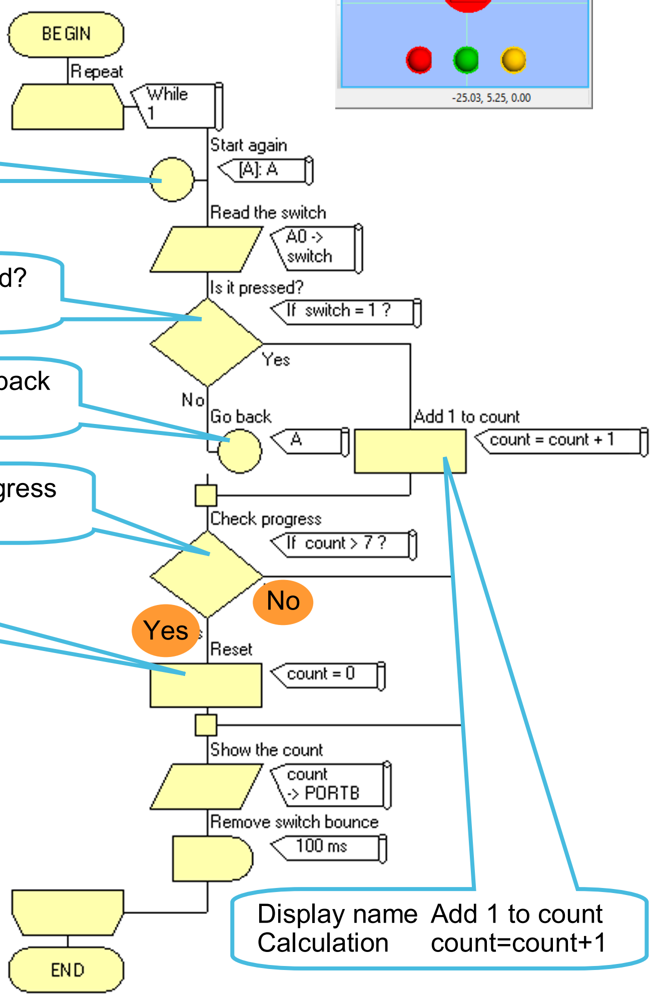
Over to you:

The program:

- Build the flowchart, adding three 'LED (5mm PCB)'s and a 'PCB Switch' to the dashboard panel. Notice the Yes / No alignment on the second Decision box!
- Connect an LED to PORT B bit 0, the second to PORT B bit 1, and the third to Port B bit 2.
- Connect the switch to Port A, bit 0.
- Use 'Project Explorer to create new byte variables called 'switch' and 'count'. Initialise 'count' variable to zero.



- Here are configuration details for some icons. Others you should be able to set up for yourself!



(The 100 ms delay means that any contact bounce within 100ms is ignored.)

- Save the Flowcode program.
- Simulate it to check that it counts.
- Compile it to the chip and then build the circuit shown on the next page.

FLOWCODE 6

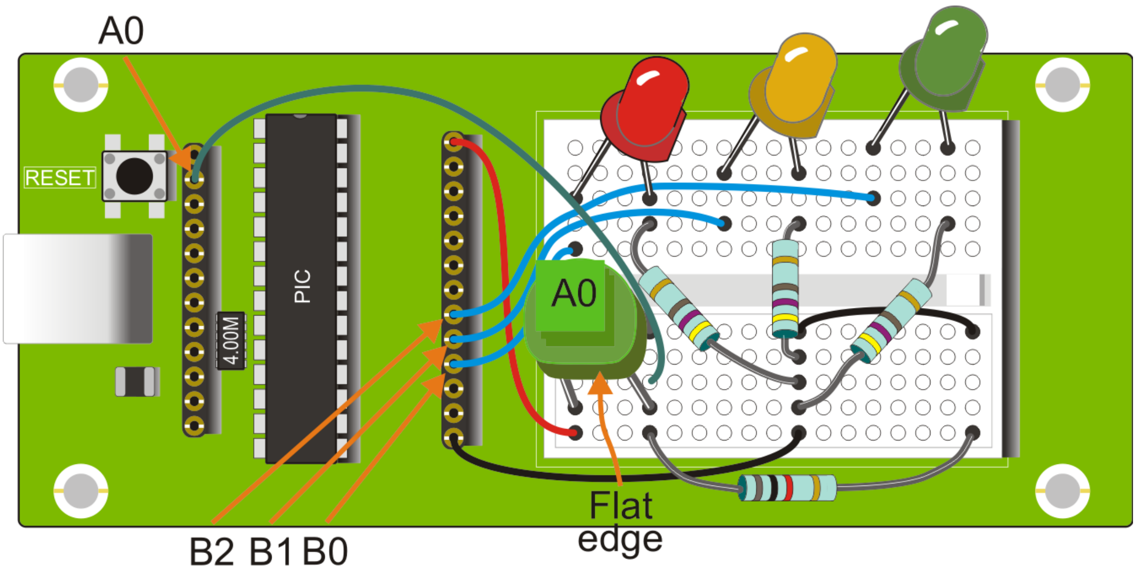
Worksheet 10
Counting

Simple microcontroller circuits

Over to you:

The circuit:

- Build the circuit, shown below, on the prototype board.
    - The switch forms a voltage divider with the 1kΩ resistor, and sends information to the microcontroller via the socket A0.
    - The number of times the switch is pressed is shown in binary on the three LEDs. This means that the LED attached to socket B0 shows 'units'; the one attached to socket B1 shows 'two's' and that attached to socket B2 shows 'four's'. If all three are lit, the count is 4 + 2 + 1 = 7.
    - Make sure that the LEDs are plugged in the right way round!
      They are protected by 470Ω resistors, as usual.
- Test the circuit by pressing the switch a number of times, and observing the resulting effect on the LEDs. Eventually, the count will return to zero (i.e. all LEDs turned off,) and then the count starts again.



What to do next:

Modify the program so that it always starts showing a count of zero.

About this course

Introduction

The course is essentially a practical one. The hardware makes it simple and quick to program and construct microcontroller circuits.

Aim

The course introduces the graphical programming language Flowcode 6, and its use in programming PIC microcontrollers.

Prior Knowledge

None needed.

Learning Objectives

On successful completion of this course, you will be able to:

- run the Flowcode 6 application;
- select a target microcontroller for it;
- create a Flowcode flowchart by adding and configuring the following icons:
    - loop;
    - input;
    - output;
    - decision;
    - delay;
    - calculation;
    - switch;
    - connection points;
    - component macros.
- create and use variables within the Flowcode program;
- open the Dashboard Panel and add and configure components on it;
- use on-screen simulation to debug the Flowcode program
- save the Flowcode flowchart;
- compile the Flowcode program into machine code;
- link the Matrix Proto Board to a computer and transfer the program;
- test a Flowcode program using hardware connected to the microcontroller.

What the student will need:

To complete the course, the student will need the following equipment:

- 1 Matrix Proto Board
- 3 1kΩ resistors
- 3 PCB switches
- 1 light-dependent resistor
- 1 10kΩ potentiometer
- 3 LEDs
- 3 470Ω resistors
- 1 sounder
- lengths of 0.6mm single strand wire.

Time:

It will take students between four to six hours to complete the ten worksheets.

Taking it further:

Once you have completed these worksheets, you will want to experiment with your own programs. The ‘Step-by-step’ guide, on the pages that follow, provides a basis for these.

Use the examples in worksheets to help when adding hardware to the prototype board . These cover a large number of hardware configurations. They show circuits with:

- a single LED;
- a switch and one LED;
- two switches and one LED;
- three switches and one LED;
- three LEDs;
- an analogue sensor (via the ‘Potentiometer (Slider)’) and one LED;
- one switch and three LEDs;
- two switches and a sounder..

Further information:

This course contains all the information you need to gain a firm foundation in the use of Flowcode 6 with the hardware provided.

Once you have mastered this introduction to Flowcode 6, you are urged to look at the extensive volume of information and examples contained in the Matrix wiki, found at

www.matrixmultimedia.com/wiki

to expand your knowledge and experience.

Step-by-step Guide to Flowcode 6 flowcharts:

1. **Open Flowcode 6:**
   When the Flowcode application starts, you are presented with four options:
   - New project;
   - Open a template;
   - Launch Flowcode Help;
   - Open an existing Flowcode project.

   Click on 'New project'.

2. **Select the target microcontroller:**
   The 'Project Options' screen opens.
   Click on the 'Misc' tab, and then on the 'Matrix Proto Board (18F24K50)' option.
   Then click on 'OK'

3. **Add the icons:**
   A new flowchart appears.
   If the 'System Panel' appears on the screen, delete it by clicking on the '**X**' in the top right-hand corner, or by opening the 'View' menu and de-selecting 'System Panel'.
   Down the left-hand edge of the workspace is the strip of possible icons. Click and drag the ones you want to make up the flowchart .

4. **Configure each icon:**
   Double-click on each icon in turn, and use information like that given in the worksheets to complete the configuration dialogue box.
   Click on 'OK' after completing each.

5. **Open the Dashboard Panel:**
   Open the 'View' menu, and click on the 'Dashboard Panel' option.

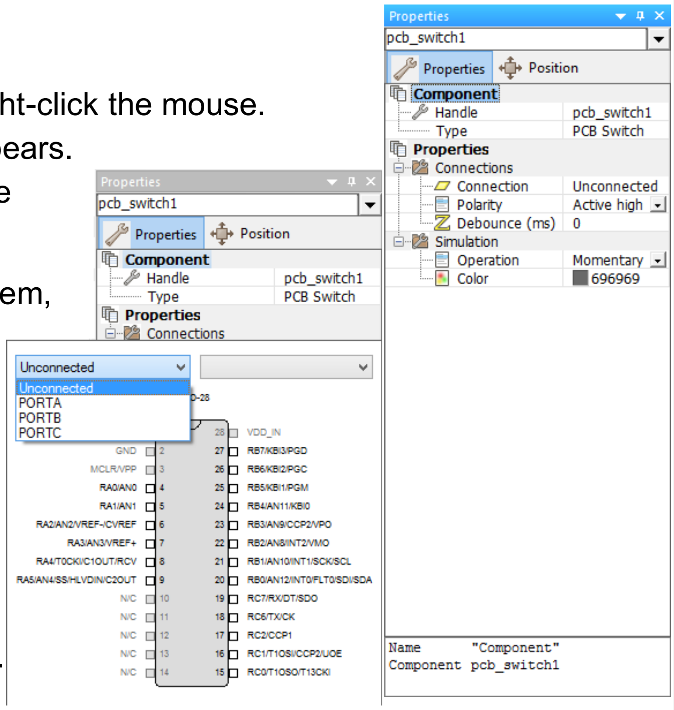
6. **Add the components:**
   Items such as switches, and the 'Potentiometer (Slider)' are found in the 'Inputs' toolbox.
   The LEDs and sounders are found in the 'Outputs' toolbox.

   Find the component you want and click on the down arrow just to the left of its name.
   Select the ''Add to dashboard panel' option.
   Some components may be hidden, and require use of the 'Search' box.

Step-by-step Guide to the Worksheets - continued...:

7. Configure properties of components:

Move the cursor over the component, and right-click the mouse.
The 'Properties' panel for the component appears.
The important one for these worksheets is the 'Connections' property, shown on the right.



Click on the text alongside the 'Connection' item, (here "Unconnected") and a diagram of the chosen microcontroller chip appears.

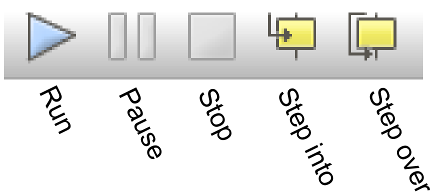
Select the pin to connect the component by:

- selecting the port and bit from the two drop-down lists;
  or
- clicking on the pin on the image of the chip.

8. Simulate the program:

You can test whether your Flowcode program works by simulating it 'on-screen'.
To simulate a flowchart:

- select the 'Run' option from the 'Debug' menu;
  or
- click the 'Run' button  on the main toolbar (or press **F5**).



Flowcode will go into simulation mode and will start to execute the program in the flowchart. A red rectangle indicates the next icon to be executed.

Simulations can be paused or stopped by selecting either the 'Pause' or 'Stop' options from the 'Debug' menu or selecting them from the simulation section of the main toolbar.

Alternatively, you can simulate the flowchart step by step, using the 'Step Into' function by pressing **F8**. You can also 'Step Over' icons by pressing **Shift+F8.**
All these options can be accessed from the 'Debug' menu or by clicking the buttons on the main toolbar.

Step-by-step Guide to the Worksheets - continued:

9. **Save the flowchart:**
   Flowcharts must be saved before they can be downloaded to the microcontroller.
   To save the flowchart:
   - select either the ‘Save’ or ‘Save As’ option from the ‘File’ menu;

   or
   - click the  button on the main toolbar.

10. **Connect the board to the PC:**
    Connect the USB socket on the Matrix Proto Board to the USB port of a computer. This allows the program to be transferred from the computer to the PIC microcontroller, and also delivers electrical power to the board.

11. **Compile the flowchart and transfer it to the chip:**
    The next step is to compile the Flowcode program, (convert it into machine code,) and then transfer it to the microcontroller .
    To do so:
    - select the 'Compile to Chip' option from the 'Build' menu;

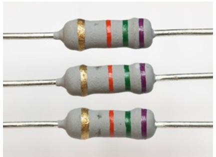
    or
    - click the ‘Compile to Chip’ icon on the toolbar.

12. **Test the program using**  **the hardware:**
    Press the ‘Reset’ switch on the Matrix Proto Board.
    Now use the mounted switches and LEDs etc. to confirm that the program does what you want it to do.

Additional information:

Resistor colour code:

It is useful to be able to identify resistors, using the resistor colour code. They often come with coloured bands across their body to show the value of the resistance.
Each colour represents a number, as shown in the table.



Black	Brown	Red	Orange	Yellow	Green	Blue	Purple	Grey	White
0	1	2	3	4	5	6	7	8	9

To read the colour code, start from the opposite end to the gold or silver band:

- Write down the number shown by the first colour band, and then the second colour band.
- Add the number of 0's shown in the next band (e.g for red, add two 0's.)
- The final band (usually gold, (5%) or silver (10%)) shows you the tolerance (how accurately made it is.)

For example, each resistor in the picture has a resistance of:

7 (purple) 5 (green) 000 (orange) = 75000Ω and a tolerance of 5%

Variables

In programming, a variable is a named location in memory, used to store information. The information may change as the program runs, but the program can always find it, by searching for the name of the variable.

Flowcode allows a number of different types of variable, resulting in efficient storage of different types of information. They include:

- Boolean variables - store just a single bit - either a single '0' or a single '1'. They take up very little room in memory, but store little information, usually the state of a digital sensor such as a switch.
- Byte variables - store a 'byte' (8 bits,) and so can store more information, numbers from 0 to 255.
- Integer variables - store integers (whole numbers.) These can include both positive and negative numbers, or allow only positive numbers, and are 16 bits long.

You can set the variable type when you create the variable. The easiest way to do this is to use the Project Explorer. You can select this from the 'View' menu on main toolbar. Choice of variable type may be decided by the component macro you use, as shown in worksheet 8.

<table>
<tr><th>Work sheet</th><th>Notes for the Instructor</th><th>Timing</th></tr>
<tr><td></td><td>Introduction:
The diagram will be particularly useful for students starting this module allowing them to identify the components on the hardware. In particular, they should note the position of the reset button and power sockets.
The Introduction gives an overview of the hardware and the way it is used.</td><td></td></tr>
<tr><td>1</td><td>The task seems trivial - a switch connected in series with the LED could do the job. However, often, such switching is carried out in software, not in hardware, the courtesy light in a car being a case in point.
‘Flowcode’ often offers several solutions to the same task. Here, the correct binary number is sent to the output port. The same could be achieved by using component macros to switch the LED on and off. The numbering of the LEDs on the robot starts with LED 0, so that LED 3 is actually the fourth one in the LED array.
Although the task is straightforward, it gives the student experience of ’drag-and-drop’ to construct the flowchart, of configuring icons, of running the simulation within ‘Flowcode’ and of transferring the program to hardware. Consequently, an inexperienced student may take longer than expected.
To create the ‘switch’ variable, open ‘Project Explorer’, click on the down arrow to the left of the ‘Variables’ label, and select the ‘Add new’ option. You then name the new variable and can give it an initial value and add a description. The reason for choosing a variable type is ultimately to reserve sufficient space in memory to accommodate the value stored in the variable. In principal, then, it is best to choose the type that most efficiently fits the data you intend to store. For the purposes of this module, unless told otherwise, it is fine to accept the default type, byte variable.
Very few problems are likely. The simulation might be made easier by turning the switch action from ‘momentary’ to ‘latching’. Do this by right-clicking on the switch and selecting ‘Properties’. Then click on the down arrow at the right-hand end of the ‘Operation’ row and selecting ‘Latching’.
If the program does not simulate correctly, then check that the student:
loaded the components to the dashboard panel successfully;
connected them to the correct pins of the microcontroller, (switch to Port A, bit 0 and LED to Port B, bit 0);
configured the icons correctly.
(If the Dashboard is not visible, use the ‘View/Dashboard Panel’ menu selection.)
If the simulation works, but the hardware doesn’t, then make sure that:
the USB lead is connected between the board and the PC;
the power supply wire links are correct;
the LED is plugged in the right way round;
it is protected by the correct value resistor;
the flat edge on the body of the switch is oriented correctly;
the switch and resistor are connected as a voltage divider;
all wire links are in the right places.
The next step is to download the program again, watching the ‘Compiler Messages’ screen. If there is still a problem, use the program provided in the ‘Solutions’ folder.
The ‘What to do next’ section invites student to modify the program to switch on a car’s two sidelights, assumed to be controlled independently. A second LED, connected to Port B, bit 1 and resistor are added. To do this, another output icon, controlling Port B, bit 1, as a single bit, is added to the program, after the existing one.</td><td>20 - 30 mins</td></tr>
</table>

Teacher Guide

Scheme of work

Simple microcontroller circuits

Work sheet	Notes for the Instructor	Timing
2	This worksheet extends the task of the first, by making the LEDs flash on and off. This time, no switch is involved, making the hardware easier to construct. Instead, the program loop makes the LED switch on and off repeatedly. Once again, the task could have been accomplished using component macros to switch the LED. If the simulation does not work, carry out the checks listed for the last worksheet. Similarly, if there are hardware problems, use the issues listed there. In the 'What to do next' section, the frequency of flashing is governed by the duration of the two delays. To double the frequency, as asked, the delays should be halved. Taking this matter still further, the first delay decides how long the LED is off (i.e. the 'space') while the second controls how long the LED is on (i.e. the 'mark'). Adjusting the relative size of these delays controls the 'mark-to-space' ratio for the signal produced. To make the LED flash only ten times requires reconfiguration of the loop, to use a loop count set to ten.	**20 - 30 mins**
3	This worksheet uses the same hardware, set up in the same way, as in worksheet 1. If there are any hardware problems, consult the list of issues given for that worksheet on the previous page. The extension mentioned in the 'What to do next' section simply requires adjustment of the duration of the delay.	
4	In electronics, a latch is a subsystem which has two stable states. It is triggered into one (the 'on' state,) by a switch called the 'set' switch. The subsystem remains 'on' until it is triggered into the other state (the 'off' state) by the 'reset' switch. As before, the state of one switch, (the 'set' switch) is stored in the variable called 'switch'. This time, a second variable ('reset',) is created to store the state of the other switch, the 'reset' switch. These can be created using the 'Project Explorer', as described on the previous page for worksheet 1. The program uses Decision icons to test whether each switch is pressed. In these, a condition tests which route the program follows - the 'Yes' (condition is true,) or 'No' (condition is false,) route. (For programming aficionados, the condition could be written simply as 'If switch' and 'If reset' meaning 'If switch / reset is *true* (i.e. =1.)) It is worthwhile stepping through the program with the students, looking at what happens when the switches are pressed or not. If the program does not simulate correctly, check the structure of the program - that the 'Yes' and 'No' loops are the right way round, (as one option in configuring the Decision icons is to reverse thm.) Then look at the advice given on the previous page, for worksheet 1. The hardware is slightly more complicated, because of the second switch. Make sure that the flat edge on the body of the switch is in the correct position, and that the signal from it is connected to port A, bit 1. Then look at the other issues listed on the previous page. The task in the 'What to do next' section should be tackled by changing the program, not the hardware!	

Work sheet	Notes for the Instructor	Timing
5	This worksheet looks at a common task for microcontrollers - controlling a number of devices, motors, heaters, solenoids, lamps etc, in sequence. This program can readily be modified to cover a wide range of similar applications. The program is linear - no branching. It progresses through a series of instructions lighting each LED in turn for a set time. It may not be obvious to students that the sequence turns on other LEDs but does not turn off those already lit until the program loops back to the beginning. They could be encouraged to experiment with the output icons, sending various decimal numbers to 'Entire Port' rather than 'Single Bit'. For example, sending the number '2' (=010_2 in binary,) will light the second LED, while turning the first and third LEDs off. Similarly, sending number '6' (=110_2,) lights the second and third LEDs, but not the first. To tackle the task given in 'What to do next', students should remind themselves of what they did in the program in worksheet 3.	**20 - 30 mins**
6	This worksheet introduces an important area in control - logic systems . In these, the microcontroller makes decisions based on the information it receives, from switches in this case. The 'AND' function requires that switch A AND switch B are pressed. The 'OR' function waits until either switch A OR switch B (or both) is/are pressed. The student first creates the variables 'InA', InB' and 'result', using 'Project Explorer' as before. The program examines each switch in turn, storing its on/off state as 1/0 in the appropriate variable. The logic operation is carried out inside the Calculation icon, generating a 1/0 value in the variable 'result', which is then displayed on Port B, bit 0. The program involves no branching and so is straightforward. If it does not simulate correctly, check the calculation inside the Calculation icon. The hardware arrangement is the same as in worksheet 4, and so the same checks apply if there are hardware problems.	**25 - 40 mins**
7	This worksheet extends the area introduced in the last one, logic systems. The last logic functions, like AND and OR. This one shows how to handle more complex logic operations. The program is again a linear one. A third switch is added to the system, and so a third variable, InC, must be created to store its state. An input icon reads the state of this third switch and stores it in InC. After that, all the hard work is carried out inside the Calculation icon, with the result processed as before. As before, if it does not simulate correctly, first check the configuration of the Calculation icon. The prototype board now looks a bit crowded, with the addition of the third switch and resistor, and it is easy to make mistakes. The usual checks apply, and should highlight any problems. The third switch provides a signal for Port A, bit 2. The modification described in the 'What to do next' section is straightforward to implement. The microcontroller is already gathering information about the states of the three switches from the variables InA, InB and InC. What it does with that is controlled by the calculation inside the Calculation icon. This modification amounts to a change in that calculation.	**25 - 40 mins**

Work sheet	Notes for the Instructor	Timing
8	Now a departure! This worksheet involves handling an analogue signal. • Digital signals have only two possible values - off and on, or equivalents. • Analogue signals, on the other hand, vary over a wide range of values. This may be a good time for the teacher to elaborate on the differences between analogue and digital quantities, and to talk about analogue-to-digital conversion. Analogue signals are created from a huge variety of devices and subsystems. To simplify the matter, Flowcode handles analogue inputs through simple devices like the 'Dashboard Knob'. Turning the knob creates an analogue signal which 'Flowcode' uses to mimic the output of the light sensor. As the light gets brighter, the output increases. The hard work is done inside the PIC chip, using a 'Component Macro', a section of program code written especially for one of the hardware components in 'Flowcode'. The 'GetInt' macro generates a numerical output representing the brightness of the light falling on the LDR. This output is stored in the integer variable 'light'. The rest of the program follows the line taken in previous exercises - using a 'Decision' icon to determine which branch the program should follow. If there is a problem with simulation, the teacher could disable the Component Macro, (by right-clicking on it and selecting the 'Disable Icon' option,) and add a Calculation icon to set the value of 'light' at 600, and later 400 to test whether the rest of the program works. If it does, then the problem lies in the configuration of the Component Macro. This could be checked against the solution provided. If there is a problem running the hardware, having checked the issues outlined earlier in this guide, check that the three legs of the variable resistor are plugged into different rows of the prototype board, that the middle leg is connected to the LDR, through the wire linkand that one end leg is connected to 0V, through another wire link. Then check that the knob is turned about half-way round to begin with. The solution to the extension problem outlined in the 'What to do next' section lies in reconfiguring the Decision box. It is possible to change the hardware arrangement, but that is electronics, not programming!	**20 - 30 mins**
9	This worksheet shows how to generate sounds, normally considered an analogue quantity. In this case, it is generated by a digital signal repeated so rapidly that the sounder vibrates in and out, producing a sound. The principle is demonstrated using LEDs, with the frequency reduced far enough to see them flashing. The program introduces a new icon, the 'Switch' icon. This is not a switch! It is related to the 'switch_case' construction in the 'C' programming language. It provides a really useful way to create menus. Here, the two switches are both connected to Port A - switch 'A0' to bit 0 and 'A1' to bit 1. When 'A0' is pressed, the 'read input' icon sends a value of 1 to the variable 'note. When 'A1 I pressed, 'note' receives value 2. (These values derive from the connections to Port A. If the switch 'A1' was connected to bit 4, it would send a value of 2^4 or 16 to 'note'.) The 'switch' icon has a branching effect. It examines the value stored in 'note', because of the way it is configured. The progress of the program depends on that value. If 'note' = 1, it goes down the path with the 5ms delays. If 'note' = 2, it goes down the path with the 1ms delays. If the value stored in 'note' does not match any of the ones specified in the configuration, then the program proceeds down the 'default' route, which, in this case, has no icons in it. Continued on next page…	**20 - 30 mins**

Work sheet	Notes for the Instructor	Timing
9	When simulating the program, the teacher may wish to change the values in the Delay icons (to 5s and 1s, for example,) to slow down the rate of flashing. If the program does not simulate, in addition to the checks mentioned on earlier pages, check the setup of the Switch icon. Make sure that the students has set up routes for ‘note’ values of 1 and 2. Make sure that the variable ‘note’ takes its value from the entire Port A, and not just a single bit. Once again, if necessary, check the program against the one provided in the Solutions folder. In troubleshooting hardware, be aware that the sounder is a polarised component, meaning that it works only when the positive leg (longer) is connected to the output of the PIC chip and the shorter leg to 0V, and not the other way round. The modification in the ‘What to do next’ section requires another branch in the ‘Switch’ icon. When both switches are pressed, the value ‘3’ is stored in ‘note’, and so the new branch is selected with that value, and contains a new value for the Delay icons.	**20 - 30 mins**
10	It is often valuable to count the number of occurrences of some event. That’s the aim in this worksheet. The event is the pressing of the switch. However, there can be an undesirable side-effect, switch bounce. Switches contain springy metal contacts. When these are thrown together, they can spring apart (several times) before finally coming to rest. This can produce a false count. To overcome this in this program, there is a delay of 100ms each time the count is updated. The idea is that the bounces should die away within this time. The delay can be increased, but that makes the program less sensitive to new events. Another new addition is the use of break points. Positioned as they are, they send the program back to recheck the switch when the first Decision icon results in the program following the ‘No’ branch. Break points should be used sparingly, as they can make it difficult to fault-find. When the switch is pressed, the value stored in ‘count’ is incremented (increased by one,) using the calculation ‘count = count + 1’. The second Decision icon then checks whether it has reached 6, as this program has set that as a maximum. When it has, the Calculation icon in the ‘Yes’ branch resets the value in ‘count’ to 0, and the whole process starts again. The value stored in ‘count’ is displayed, as a binary number, on the LEDs. The teacher should ensure that students can translate this number into decimal. Notice that the ‘Yes’ and ‘No’ branches are reversed for this Decision icon. That should be one of the checks, if there are problems with the simulation. The extent of the program can be reduced, for checking, by disabling the second Decision icon, removing the maximum value for ‘count’. If a problem arises with the hardware, run through the checks listed earlier. In particular, ensure that all three LEDs are connected the right way round, and are protected by 470 ohm resistors. Check that they are linked to the correct outputs of the PIC chip, (Port B, bits 0, 1 and 2.)	**20 - 30 mins**

Document Control

- **18 06 2014**
    - Version 1—Document creation

- **18 08 2014**
    - Version 2 - Minor image amendments