FORMULA FLOWCODE

An introductory course

Introduction

Formula Flowcode it intended to be used with students from 14+ as a motivating resource for learning about electronics and technology. With this in mind this guide sets out the first 5 stages of learning that can be carried out with Formula Flowcode.

In each of the stages we outline the concepts that will be introduced to students and the Flowcode icons that are to be introduced. We then develop example programs to illustrate these concepts and finally provide a summary of the concepts learned and suggestions fro further work.

The approach teachers should take here will depend on the background and capability of students. For those with little experience of electronics and technology we have provided a suite of hardware macros which enable students to drive the Formula Flowcode robot very easily. These macros are available to Flowcode users when they put the Formula Flowcode component in the Flowcode workspace.

This means rather than students having to understand how Pulse Width Modulation is used for motor speed control, students are able to simply set the speed of a motor (between 0 and 255), and also to set the motor in a forwards or reverse direction. Students can also set the robot to spin, take a light sensor reading, etc. These macros also mean that students avoid issues of how sensors are connected to the PICmicro microcontroller, and issues concerning A/D sampling etc.

Other, more experienced, students might find this approach a little superficial and will want to immediately 'get under the hood' of Flowcode by understanding the overall circuit diagram, PICmicro microcontroller connections, etc.

Teachers using Formula Flowcode may go through these exercises twice - a first time to introduce students to the concepts of control, and a second time to delve deeper into the subject.

However as most problems occur at a lower level this resource will concentrate on typical first time users to Formula Flowcode.

Note that all of these challenges are well within the capabilities of the free version of Flowcode shipped with the Formula Flowcode robot.

How to use this resource

It is possible to use this document as a stand-alone course to give to students, but this depends on the level of the students and how well they can direct their own learning experience.

More likely, this course can be used to give teachers a short introduction to the capabilities of the Formula Flowcode robot and how it can be used in an educational context.

The structure of the course can be retained and used as a framework around which the topics of control can be introduced. With each topic, students should be shown the basic concepts and then very quickly asked to develop their own programs. In practice it should be possible to teach someone of 12+ using this framework.

Resources you can use to help you

There are several resources you can use for assistance

- With Flowcode there are 28 example files that demonstrate how Flowcode works.
- With this document there are 13 sample files that you can use to demonstrate the individual concepts.
- On the Matrix Multimedia web site you will find a number of videos that show how you can get started with Flowcode.
- On the Matrix Multimedia web site there is a downloadable learning resource entitled 'An introduction to microcontroller programming' which gives a detailed introduction to PICmicro microcontrollers and Flowcode programming.
- On the Matrix Multimedia web site there is a user's forum where you can ask questions on Flowcode and/or the Formula Flowcode robot.

www.matrixmultimedia.com

FORMULA FLOWCODE

Level 1 - Nuts and bolts

Fundamentals of programming the robot

Concepts

- Creating a flowchart program
- The Formula Flowcode component
- Simulation of the program
- Running the program on the robot

- Hardware macro icon
- Delay icon
- Loop icon

Program 1.1 - Turning an LED on

This first program will simply turn an LED on for a few seconds, and then turn it off again.

1) Run Flowcode V3.

2) Select "Create a new Flowcode flowchart…" on the opening screen and click "OK".

3) Select the "Formula Flowcode Buggy" as the target device and click "OK" (see screenshot above).
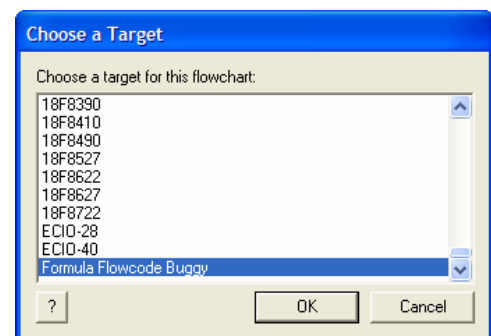
4) Click the "Formula Flowcode" component icon:

5) Drag the "Component Macro" icon onto the flowchart between the "Start" and "Stop" icons.

6) Your program will now look like this:
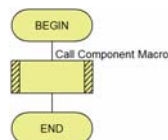
7) Double-click this new icon.

8) The window on the right will be displayed. Select the "FormulaFlowcode(0)" component, then the "LEDOn" macro. Type "3" into the "parameters" box. Also add an appropriate comment into the "Display name" box. Once you have finished, click the "OK" button.
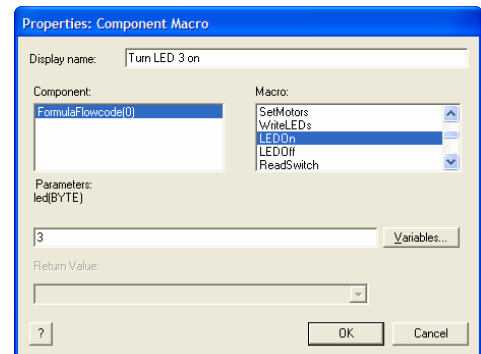
9) Next, drag a "Delay" icon onto the flowchart immediately below the previous icon. Double-click this delay icon and select 2 seconds as the delay time. Also, add an appropriate "Display name" and then click "OK".

10) Finally, add another "Component Macro" icon to the end of your flowchart. Edit its properties so that LED 3 is turned off (using the "LEDOff" macro). Your finished program should look similar to that on the right.

11) Save your program (File...Save), giving it an appropriate name.

12) Simulate the program by clicking the "Run" button. You should see LED 3 on the Formula Flowcode window light for a few seconds.
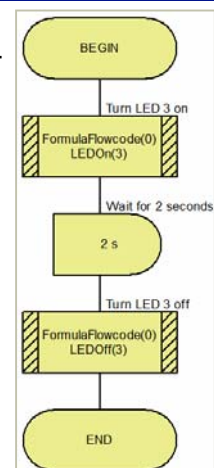
Ex 1.1

FORMULA FLOWCODE

Program 1.2 - Flashing the LEDs

The next program will use a loop to flash all of the LEDs on and off.

1) Create a new flowchart as before. Remember to select the "Formula Flowcode Buggy" as the target and click the "Formula Flowcode" component icon.

2) Very often in programming, it is essential that the microcontroller runs a program forever. For this, we will use the "Loop" icon. This icon can either loop a certain number of times, or it can loop while a certain condition is true.

3) Add the "Loop" icon to your blank flowchart and edit its properties so that it loops forever. To do this, edit the "Loop while" box so it displays "1=1" (1 is always equal to 1, so the loop condition is always true and the loop is repeated forever). Alternatively, write "1" into this box - this is also always true.

4) Within the two Loop icons, add a "Component Macro" icon. Change it so that it calls the "WriteLEDs" macro and set its parameter to "255" . This will turn all of the LEDs on.

5) Why use 255 to turn on all of the LEDs? This macro uses a binary value to represent the state of each LED - 0 for off and 1 for on. To turn on all 8 LEDs, we need to send the binary value of "11111111" to the macro (which is 255 in decimal).

6) Add additional icons to your program so that the finished program looks like the one on the right.

7) Save this program and then simulate it using the "Run" button. You can also pause the program and step through each icon one at a time during simulation.

8) The LEDs on the simulation should flash on and off. This will continue forever until you stop the simulation.

9) To send the program to the Formula Flowcode robot, connect the robot to the PC using a USB cable, turn it on, and then press the "Compile to chip" button.

10) Once the program has been downloaded, press one of the switches (RB4 or RB5) to make the program run.
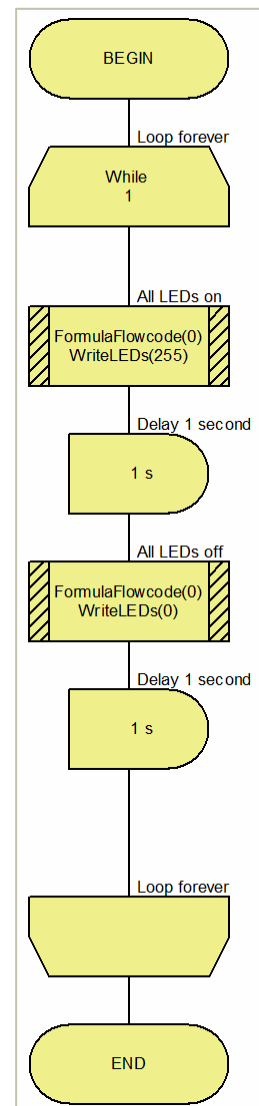
Ex 1.2

Summary

You should now know be able to:
- Create basic flowchart programs for the Formula Flowcode robot.
- Simulate these programs on screen.
- Download these programs to the robot and see them running.

Further work

- Display alternating patterns on LEDs (e.g. 10101010 to 01010101 OR 11110000 to 00001111)
- Create an "LED chaser" program (Cylon or Knight-Rider effect).
- Experiment with other hardware macros for Formula Flowcode, e.g.
    - Driving forwards for 1 second and then stopping
    - Use the "PlayNote" macro to make some sounds

FORMULA FLOWCODE

Level 2 - Shape 'n' shuffle

Taking the robot for a drive.

Concepts

- Controlling the motion of the robot
- Motor characteristics
- Open-loop control

- More about loops
- Modular programming
- Macros

Program 2.1 - Forward and back again

The program on the right shows a simple use of the inbuilt macros "Forward", "Reverse" and "Stop" to control the movement of the Formula Flowcode robot.

Create this program in Flowcode and then download it to the robot. Once it has been downloaded, you will need to press one of the switches at the front of the robot to start the program (this is an option in the "properties" window of the Formula Flowcode component).

One thing you might notice is that the robot bends slightly to the left or right. This is because the motors are not turning at exactly the same speed due to slight mechanical variations between the motors. The macro "SetMotors" allows you to specify a different value for each motor and would allow you to compensate for this issue.

A nicer program would be one that drives forwards, turns around, and then drives back. We will modify the program on the right so that it does this.
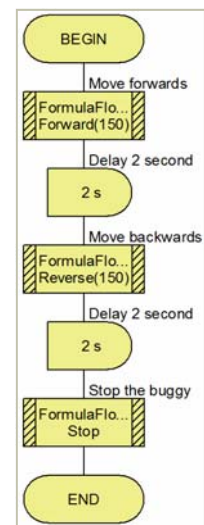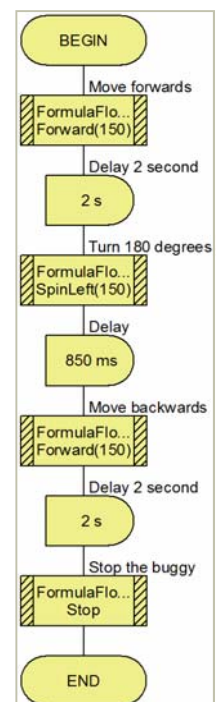
Ex 2.1

Program 2.2 - Turning around

The obvious way to make the robot turn around is to use the SpinLeft (or SpinRight) macro, but there is an immediate problem - how to we know when the robot has completed a turn of 180°? The simple answer is that we don't! We need to use trial and error to determine the required time delay.

The program on the right shows the basic principal - 2 additional icons have been added (the "SpinLeft" macro call and an extra delay), and the macro call to "Reverse" has been altered to "Forward". You will need to experiment with the values of the delay (or the speed of the "SpinLeft" macro) to find the one that works for you.

This program is an example of "open-loop control". The program will work ok (once you have found the correct delay value), but not in all circumstances. If you place the robot onto a different surface (e.g. from a table-top to a carpet), then the spin will no longer be 180°.

There are many factors that will affect the actual turning speed of the robot - the friction of the surface under the robot, the slope of the surface, the power of the batteries and variations in the motors themselves are the principal factors. But how do we compensate for these differences? We will investigate this later.

Ex 2.2

Details correct at time of going to press. Matrix Multimedia reserves the right to change specification.

HP794-60-3

PICmicro and PIC are trademarks of Microchip Technology Inc.

www.matrixmultimedia.com

Copyright © 2007
Matrix Multimedia Limited

FORMULA FLOWCODE

## Program 2.3 - The power of macros

Up to now, we have been using "component macros" to our programs. These macros perform tasks that are called again and again throughout our programs. In addition to these pre-made "component macros", we can also build and use our own macros (simply called "macros"). This is a very powerful feature of Flowcode and allows our programs to become modular.

In the previous program, we worked out a time delay that allowed the robot to turn 180°. In this program, we will use this to create a macro that makes the robot turn 90°. We will then use this macro to recreate the 180° turn of the previous program.

First, open the previous program and then select "Save as" from the "File" menu. To create a new macro, select "Macro...New…" from the menu. This will bring up a window where we can type the name of the macro, a description, and some other information. For now, simply type "Turn90" as the name and then click "OK".   We will now see a blank flowchart.

Add the "SpinLeft" icon and the delay icon to this new macro. Remember that the delay time will be approximately half that of the delay used to turn 180°. Next, go back to the "Main" flowchart (use the menu "Window...Main").

In the main flowchart, delete the old "SpinLeft" icon and delay icons. In their place, add to "Call macro" icons and in each, select the "Turn90" macro. Download this new program to the robot and confirm that is works like the previous one.
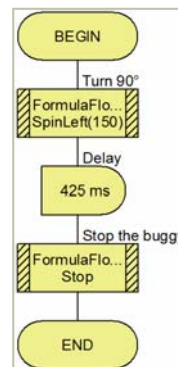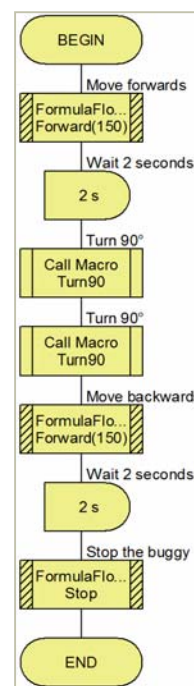
Program 2.4 - Square dance

In level 1, we introduced the loop icon and how it is used to create an endless loop. In this program, we will use the loop icon to perform some tasks a set number of times. The finished program will make the robot drive in a square.

To drive in a square, the robot needs to follow this sequence: go forward, turn 90°, go forward, turn 90°, go forward, turn 90°, go forward. Rather than writing these tasks one after another, a better way would be to repeat the "go forward" and "turn 90°" tasks 4 times.

Create a new flowchart, add a "Loop" icon and double-click it to edit it. Instead of the default "Loop while 1" condition, select "Loop count" and type 4 into the box. Add the icons to go forwards (remember the delay) and turn 90° inside the 2 loop icons. And also add the "Stop" icon at the end of the program (although this is not really necessary).

Summary

You should now know be able to:

- Use component macros to make the robot move.
- Create and use your own macros.
- Use loops to repeat a sequence of commands.

Further work

- Create programs to drive the robot in other shapes (e.g. triangle, hexagon, circle).
- Create other macros that may be useful (e.g. TurnLeft, TurnRight, Forwards10).
- Use "SetMotors" instead of "Forward" to compensate for motor differences.
- Create a basic obstacle course or maze for the robot and create a program to get the robot from one side to the other without using the sensors (i.e. pre-program the robot with the required route).

Turn90 macro

Ex 2.3

Ex 2.4

FORMULA FLOWCODE
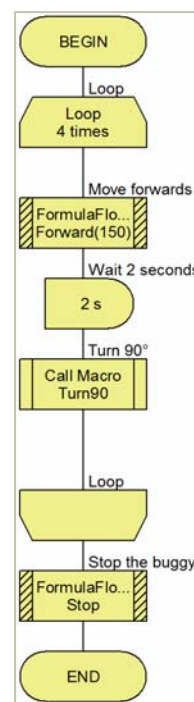
Level 3 - Robopop

Make the robot dance and sing.

Concepts

- Basic inputs
- Variables
- Calculation icon

- Decision icon
- Microphone sensor
- Buzzer output

Program 3.1 - Show me your moves

Dance is basically a series of movements often (but not always) performed to a piece of music. Creating a series of movements should be easy enough, but reacting to sound is something new and we'll cover this a bit later. So first, let's create some moves…

We don't want the robot to travel too far, so we need to keep the dance moves fairly short and to try to start and finish in the same place. To do this, use pairs of Forward/Backward and SpinLeft/SpinRight macro calls (remember the short delay after each). You could also add a few Stop calls within your dance moves. If these moves within a macro, you can combine them to create a full dance routine. An example (the Wiggle macro) is shown on the right.

Once you have a number of different moves, sequence them together by calling them from the main flowchart.
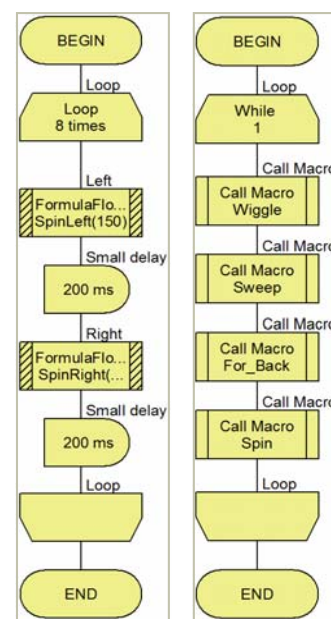
Wiggle macro   Ex 3.1

Program 3.2 - Let you robot sing

The Formula Flowcode robot has an on-board buzzer which can create basic sounds by using the PlayNote macro. This macro takes 2 parameters, "note" and "delay_ms", where "delay_ms" is the length to play the note (in milliseconds) and "note" is the pitch of the note. The table on the right shows approximate values for some musical notes and the program beyond it plays the first few notes of "twinkle, twinkle little star".

To make things easier, we could use "variables" to hold the numbers. To create the variables, select "Edit...Variables" from the menu. From the resulting window, select "Add new" and enter the name as "C" and click "OK" (leave the variable type as "byte"). Also add variables called "G" and "A", then click "Close" to continue.

Once we have added these variables, we need to give them appropriate values using a "Calculation" icon. Add this icon to the beginning of your program and double-click it to edit it. Enter "C = 64", "G = 127" and "A = 141" (without the quotes) on separate lines and then click "OK".

You can now use the variables "C", "G" and "A" in place of the numbers in your musical programs - PlayNote(G, 400). You can also define other variables and set these appropriate values.

note	value
G	0
G#	14
A	28
A#	40
B	53
C	64
C#	75
D	85
D#	94
E	103
F	112
F#	120
G	127
G#	135
A	141
A#	148
B	154
C	159
C#	165
D	170
D#	175
E	179
F	183

Ex 3.2

www.matrixmultimedia.com

FORMULA FLOWCODE
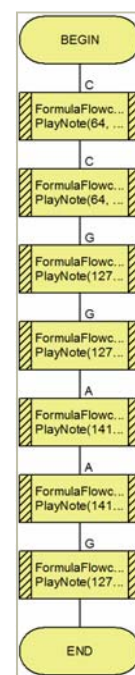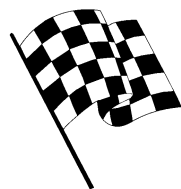
Program 3.3 - Let the music begin

Ok, the robot can now dance and it can play music. Now, let's make it dance to some real music. For this, we need to use the microphone sensor and the ReadMic macro. We will modify the earlier dance program so that the robot only dances when it hears a sound.

By default, all sensor readings are "analogue" values and are read as "bytes" (i.e. a value between 0 and 255). So first of all create a new "byte" variable called "sound". To read the microphone sensor value, use the "ReadMic" macro and set the "return value" to the variable "sound".

Now add a "Decision" icon. The robot should dance if the sound level is greater than a certain value, so in the "if" box type "sound > 50". If the sound level is greater than 50, then we want the robot to dance. If not, it should stop dancing. The ">" symbol is used to mean "*is greater than*". Similarly, "<" means "*is less than*" and "=" means "*is equal to*". In addition ">=" and "<=" mean "*is greater than or equal to*" and "*is less than or equal to*".

The completed program is shown on the right.

There are other sensors on the robot - an LDR (Light Dependant Resistor) for detecting the light level and three IR (Infrared) sensors which are used to detect how close walls and other objects are. These sensors can be used in exactly the same way as the microphone sensor.

In addition to these analogue sensors, there are a 4 digital sensors - 2 switches and 2 line-followers. These can also be used like the microphone sensor, but the values returned are either 1 or 0.
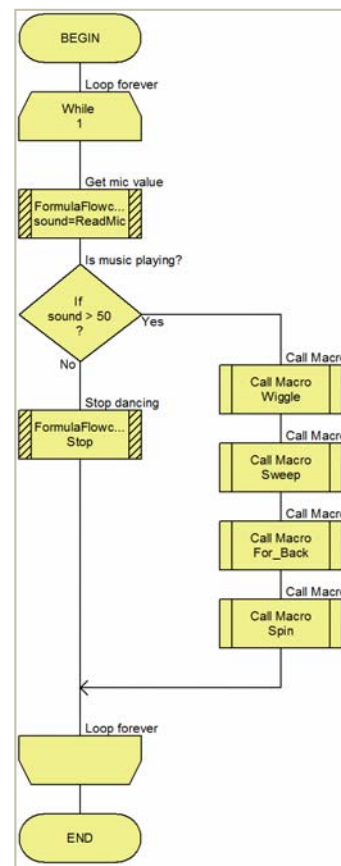
Ex 3.3

Summary

You should now know be able to:

- Create and use BYTE variables.
- Read sensor values.
- Use the decision icon with an appropriate condition to alter program flow.

Further work

- Make more dance moves.
- Play different melodies and make your own up.
- Make the robot stop dancing as soon as the music stops.
- Make the robot perform different dances depending on the loudness of the sound.
- Create a program that makes the robot move forward when you clap you hands.
- Create a "Theremin-like" instrument that plays a frequency dependant on the LDR light level.

Details correct at time of going to press. Matrix Multimedia reserves the right to change specification.

HP794-60-3

www.matrixmultimedia.com

FORMULA FLOWCODE

Level 4 - Follow my line

Program the robot to follow a line on paper.

Concepts

- Line-follower sensors
- Comments in programs
- Feedback

- Closed-loop control

Program 4.1 - What lies beneath?

The line follower sensors are digital sensors, returning 1 when the sensor is over a black surface and 0 when it is over white. The macro call that returns this sensor value is "ReadLineSensor". It takes a parameter which specifies either the left sensor (0 or 'L') or the right sensor (1 or 'R').
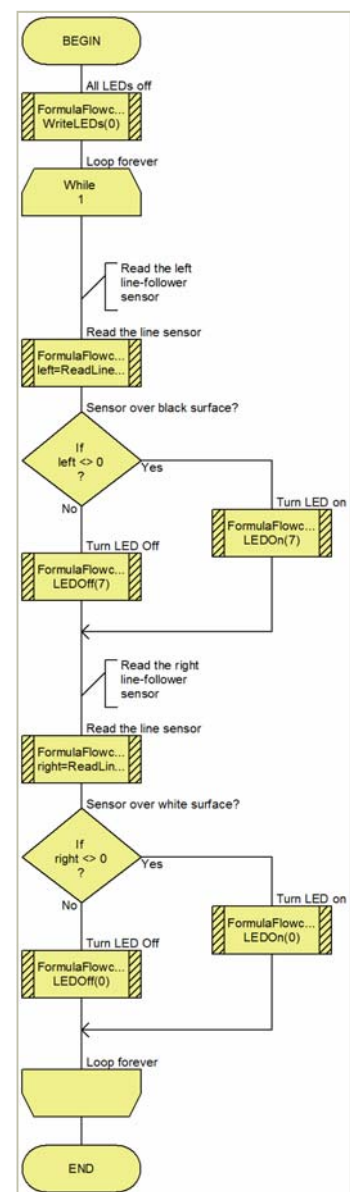
The program on the right is a simple one which displays the state of both line-follower sensors onto the LEDs. Download this program to the robot and place its line-follower sensors over various surfaces and observe the resulting output on the LEDs.

One new feature of this program is the inclusion of 2 comment icons. You can add these to the program by dragging the "Comment icon" symbol onto your flowchart. Comments like these are very important in programs and should be used where appropriate, especially to describe a group of icons that together form a distinct task. Note that icons can also have their own individual comment by changing their "Display name" property.

In the "Decision" icons, the "if left < > 0" line means "*if the variable 'left' does not equal zero*".

Now we know how to read the line follower sensors, we can put them to use. A common use of these sensors is to allow the robot to follow a line around a maze or race track. You can make your own track by sticking black tape (or drawing with a black marker pen) on a white background, or you can use the track that is on the reverse of the Formula Flowcode user guide.

There are two ways to accomplish this task. One way is to make both of the robot's line followers remain above the black line, although for this method, the line needs to be fairly thick. Another method is to make sure one sensor is above the black line and the other is off it. It does not matter which method you use, but in the next program we will use the latter (with the right sensor above the line and the left sensor below it).

Ex 4.1

Details correct at time of going to press. Matrix Multimedia reserves the right to change specification.

HP794-60-3

PICmicro and PIC are trademarks of Microchip Technology Inc.

www.matrixmultimedia.com

Copyright © 2007
Matrix Multimedia Limited

FORMULA FLOWCODE

Program 4.2 - Start your engines...

Up to now, we have been using a method of control called "open-loop control". This does not work very well in practice because there is no feedback about the position of the robot so it basically "guessed" which direction to drive in. For this program, we will use feedback to create a "closed-loop control" program.

The program, which is displayed on the right, is fairly straight-forward. There is an endless loop with two parts inside - the first part reads the values for each sensor and the second part uses these values to decide which direction to move the robot.

There are 2 variables to store the sensor values - "*L_Line*" and "*R_Line*", which are both read by the "ReadLineSensor" macro.

We have decided to use the line-following method where we retain the right sensor above the line and the left sensor off it. So the first decision is easy - if the left sensor is on the line, then spin to the left until the left sensor is not on the line.

If the left sensor is not above the line, then we check the right sensor. If it is on the line, then we are ok and should continue forwards. If not, then spin right until it is above the line.

Despite its simplicity, the program on the right works surprisingly well. The robot's movement is a sometimes a little jerky, but it should be able to complete a lap of the track within a few seconds.
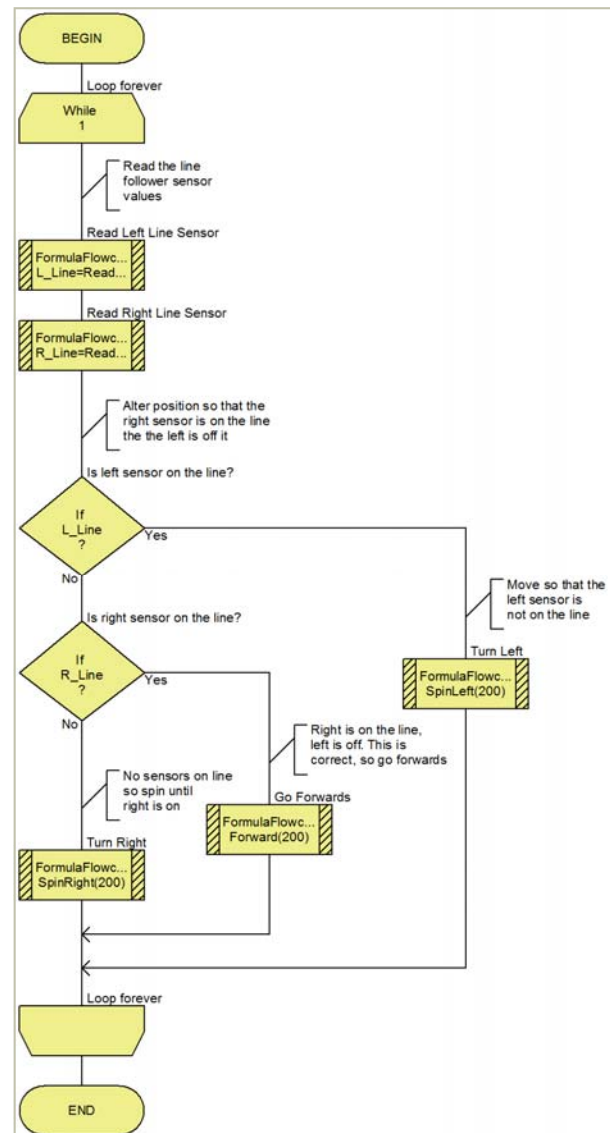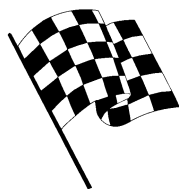
Ex 4.2

Summary

You should now know be able to:

- Use comments in your programs to aid readability.
- Explain the differences between open-loop and closed-loop control.
- Use feedback in your programs to affect program flow.

Further work

- Create your own mazes using white card and blank sticky-tape.
- Modify or rewrite the program to improve your lap time.
- Race against your friends to see who has written the most efficient program.
- Add flashing lights (using the LEDs) or a siren (using the buzzer) to your program.

Details correct at time of going to press. Matrix Multimedia reserves the right to change specification.

HP794-60-3

www.matrixmultimedia.com

PICmicro and PIC are trademarks of Microchip Technology Inc.

Copyright © 2007 Matrix Multimedia Limited

FORMULA FLOWCODE

Level 5 - I can see the light

Program the robot to follow the light source.

Concepts

- LDR sensor
- Feedback
- Closed-loop control

- Binary representation
- Hexadecimal representation
- The 8-bit microcontroller 'brain'
- Conversion between decimal, binary and hex

Program 5.1 - Displaying the light

The Formula Flowcode robot has an on-board LDR (Light Dependant Resistor) which detects the light level at the front of the robot. This sensor works in a similar way to the microphone sensor, but there is a significant difference - if the light is bright, then the returned value is low. If the robot is in the dark, then the LDR sensor will return the maximum value (255).
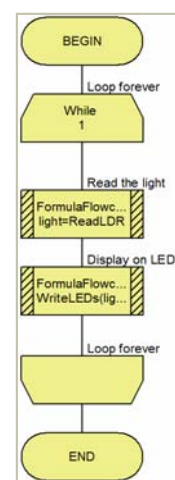
As an example of this, the program on the right will display the value of the LDR on the LEDs. As discussed previously, the sensor gives a value between 0 and 255. This program will display this value on the LEDs as a pattern representing the binary equivalent of this value (a series of 1's and 0's). This is because microcontrollers (indeed, almost all computers) store numbers and perform calculations in binary. This means that a basic understanding of binary can be very useful.

A more convenient way of representing numbers in microcontrollers is by using hexadecimal notation, i.e. "hex" or base-16. In this system, there are 16 digits used to represent numbers (0 to 9, followed by A, B, C, D, E, F), and every combination of 4 binary bits can be represented as a single hex digit, as shown in the table on the right. And 2 hex digits can represent 8 binary bits, i.e. any number between 0 (00) and 255 (FF). In Flowcode, you can write a hex number by writing "0x" before it - e.g. 0xFF. Similarly, binary numbers are written like this: 0b01010101 (= 0x55).

The heart (or is it the brain) of the robot is an 8-bit microcontroller, because numbers inside it are stored (and calculations are performed) in groups of 8 bits. This is why we have sensor values up to 255 - because 255 is 11111111 in binary (or 0xFF in hex).

Being able to convert between decimal, hex and binary numbers is useful skill when using microcontrollers. One way is to write the appropriate power of two above each binary digit (called a "bit") - just like we can write the powers of ten above columns of decimal numbers. Here are some examples:

Ex 5.1

Hex	Bin	Dec
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

$2^7=128$ $2^6=64$ $2^5=32$ $2^4=16$ $2^3=8$ $2^2=4$ $2^1=2$ $2^0=1$

= 0b00000111 = 0x07 = 4+2+1 = 7

= 0b10000010 = 0x82 = 128+2 = 130

= 0b11000011 = 0xC3 = 128+64+2+1 = 195

= 0b01110110 = 0x76 = 64+32+16+4+2 = 118

7 6 5 4 3 2 1 0

FORMULA FLOWCODE

Program 5.2 - Finding your way

In program 2.2, we created a program which turned the robot 180° using open-loop control. This did not work very well because there was no feedback about the position of the robot so it basically "guessed" when to stop turning. We will now implement the same program using feedback.

The sensors available to us are switches, line sensors, microphone, LDR and IR sensors. Ideally, we would want a compass sensor so the robot would know the direction it faced (perhaps this would make a nice add-on project). Realistically, there are only 2 choices - use the line followers with markers on the ground or use the LDR and a torch to direct the robot back to its starting point. We will use the latter method, where the robot drives towards the light.

There are a number of approaches we can take for our program. One way involves spinning around until the light level starts to decrease (and then spin back the other way a little). The program needs to spin whilst constantly comparing the LDR sensor value with its previous value. Once the light level reading has decreased, the program knows the robot has spun past the brightest point. There are some important issues to consider - the light levels in the dark part of the room will probably vary (due to reflections, etc) and the light level may decrease before the brightest reading has been read. Also, many light sources do not emit a constant brightness - in fact, they often flash too quickly for our eyes to see, but fast enough for the LDR reading to be affected.

Another way would be to spin at least 360°, reading the LDR all the time and updating a variable for the maximum brightness (i.e. the minimum value of the LDR). Once the spin has completed, spin again until the robot points to the bright spot (i.e. when the LDR is reading this minimum value). The flowchart on the right shows a macro that implements this method.

Note the final loop condition - "While LDR > (min_LDR + 5)". A 5 adds a "tolerance" to the sensor reading and has been added to make sure the brightest point is definitely found.
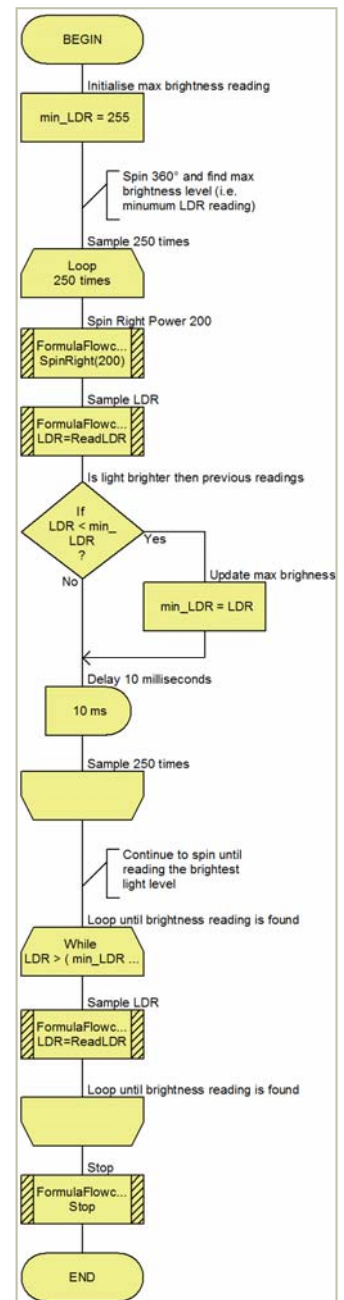
Summary

You should now know be able to:
- Create quite complex programs to solve problems.
- Convert between binary, decimal and hexadecimal numbers.
- Use the LDR sensor.
- Use feedback from analogue sensors to affect program flow.

Further work

- Experiment with binary and hexadecimal values within Flowcode programs.
- Refine program 5.2 so it works better.
- Use different light sources with the program.
- Implement other programming strategies to accomplish a 180° turn.
- Investigate closed-loop control using other sensors.
- Create a program that makes the robot follow a moving light source.

FindLight Macro

FORMULA FLOWCODE

Summary

On completion of this course, you should be able to use Flowcode to create fairly complex programs for the Formula Flowcode robot using almost all of its features.

Along the way, you should have gained a knowledge of the following concepts:

- Modular programming using 'macros',
- Variables,
- Closed-loop control,
- Feedback,
- Analogue and digital values,
- Sensors,
- Motor control,
- Binary and hexadecimal numbers.

You will have used these outputs:

- LEDs,
- Motors,
- Buzzer.

And these inputs:

- Switches,
- An LDR sensor,
- A microphone input,
- Line follower inputs,

And hopefully, you've had some fun as well!

On the right are some ideas for further work.

Further work

Drag race
Time trial race along a straight line

Remember, you'll need some form of feedback to make sure the robot stays on the line. It is probably best solved by using the line-follower sensors.

Lefty
Solve a maze by following the left hand wall

Use the IR sensors to guide the robot around a walled maze by following the left-hand side.

Daytona 5
Race 5 times around an 8x8 maze

Create a track and compete with each other to see who completes the race in the fastest time.

Pimp my ride
Create your own robot mechanics

Either create a shell for the robot, or design your own chassis. Add extra electronics via the E-Block port, I2C or UART connectors (e.g. add a display or Bluetooth module, or design your own electronics).

Full maze
Move on to full maze solving problems

Following on from "Lefty", but now with full maze-solving capabilities. Again, use the IR sensors.

Under the hood
Program the robot directly using flowcharts or C

Talk to the microcontroller directly without using the Formula Flowcode component. Read sensors using the ADC module and drive the motors using the PWM module. Either use flowcharts or C, or even assembly language!

www.matrixmultimedia.com