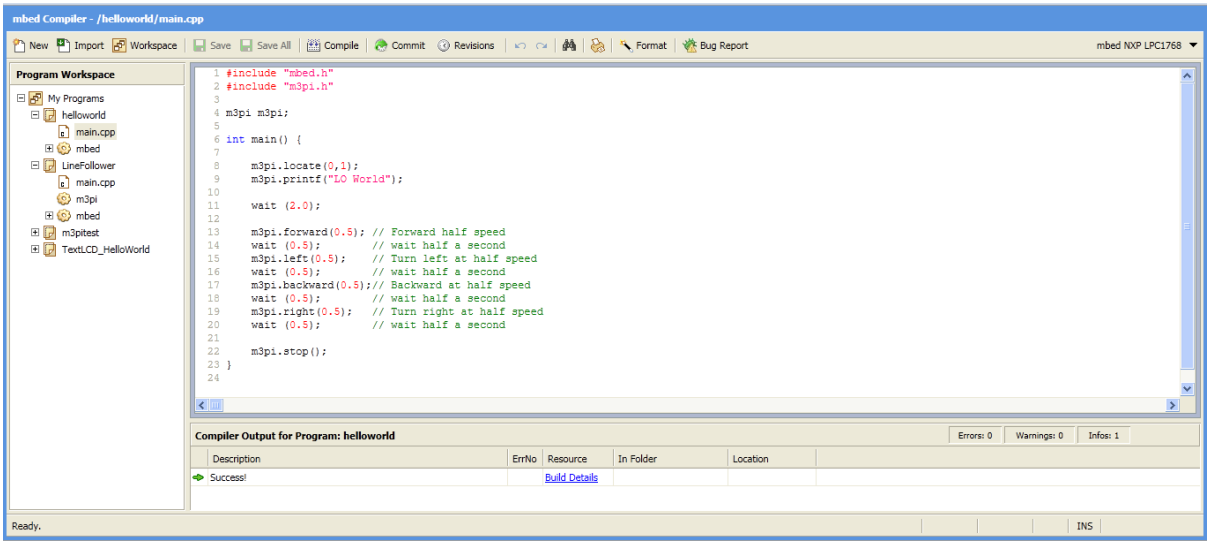
Programming Basics

Some simple syntax

This sheet will tell you the basics of programming. As an example, let's use the Hello World program, as it is nice and simple. Below is the program as you would see it in the compiler; then let's break it into pieces to see what each part does.



First let's look at how the program is structured. It is quite simple, but needs to be known if you are going to change the program or write your own.

Each line is either blank or contains a **command**; this tells the robot to do a single action.

Each line with a command also has a semicolon after it.This is to tell it that the line has finished.

The green writing after the // are **comments**.The robot does not see these, but they are very useful to the programmer in telling them what's going on. It also makes it easier for someone else to use your program, or quickly look for a certain part of the program.

Chunks of program are enclosed in curly brackets. These are important; a little like round brackets in maths. These chunks are referenced separately and may be repeated as a whole. You can also get smaller chunks within larger chunks. In this program there is only one chunk of program, after the first part. It is customary to move the part of programming inside the chunk out by a few spaces. This makes it much easier to see the chunks. You can see this has been done with the int main () {chunk}.

Remember each chunk must have a start and end, and after the end you place the closing bracket on the same vertical line as the line which started the chunk. This is all in the interests of neatness, which is very important in programming.

The program

Now let's look at the program itself, to find out what it does.

```
1 #include "mbed.h"
2 #include "m3pi.h"
3
4 m3pi m3pi;
5
```

This is the first bit of program. The first two lines have libraries, entitled mbed.h and m3pi.h. These have lists of **functions** which you can now use in the program.

The word m3pi m3pi; are used to connect the pins on the microcontroller to the robot itself (after all the microcontroller doesn't know that it's connected to a robot, it just does what you tell it to, these will ensure that the robot works properly.

```
6 int main() {
7
8     m3pi.locate(0,1);
9     m3pi.printf("LO World");
10
11    wait (2.0);
12
13    m3pi.forward(0.5); // Forward half speed
14    wait (0.5);        // wait half a second
15    m3pi.left(0.5);    // Turn left at half speed
16    wait (0.5);        // wait half a second
17    m3pi.backward(0.5);// Backward at half speed
18    wait (0.5);        // wait half a second
19    m3pi.right(0.5);   // Turn right at half speed
20    wait (0.5);        // wait half a second
21
22    m3pi.stop();
23 }
24
```

This may look complicated at first, but it is quite a simple program. Lines 8 and 9 reference the LCD display and the part in quotation marks is printed when the robot is turned on. Look out for this when you turn your robot on. See if you can change what it says (remember you can only have up to 8 characters).

The wait (2.0); command is simple. It tells the robot to wait for 2 seconds before going onto the next line. You can experiment with the length of this wait.

The next few commands are the parts that make the robot move. The comments are very useful here as they explain what each command does. You can see that the speed varies from 0 to 1, and that the speed at which it moves is contained within the curly brackets after the function.

As an example let's look at lines 13 and 14. Line 13 tells the robot to go forward at half speed. It will keep going until another function tells it not to (either by stopping or moving in a different way). The wait(0.5); tells the robot to wait (remember from above) except this time the robot is moving, so it will be going forwards for 0.5 seconds. See if you can work out the rest of the program yourself.