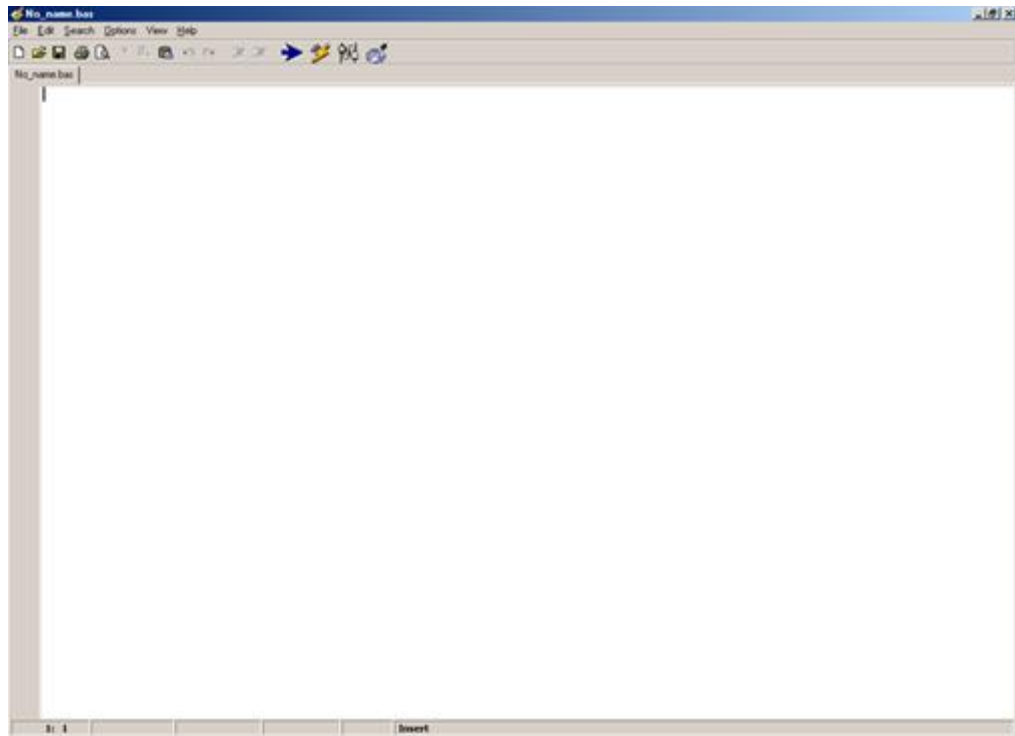


Writing your first program

Locate the program **PBP_EDITOR.EXE**, either from the **START** menu, the desktop, or directly from the folder that the compiler was installed into. You will be greeted with a blank editor. See below.



Type in the following program, or copy and paste into the editor.

' Flash all the LEDs sequentially connected to PORTD

DEVICE = 16F877

' We'll use a PIC16F877 PICmicro

XTAL = 4

' With a 4MHz crystal

DIM SCAN AS BYTE

' Declare SCAN variable

SYMBOL LED = PORTD

' Alias PORTD to LEDS

LOW LED

' Make PORTD output LOW

LOOP:

LED = 1

' First LED on

DELAYMS 300

' Delay for .3 seconds

FOR SCAN = 1 TO 7

' Go through For..Next loop 7 times

LED = LED << 1

' Shift on LED, one to left

DELAYMS 300

' Delay for .3 seconds

NEXT

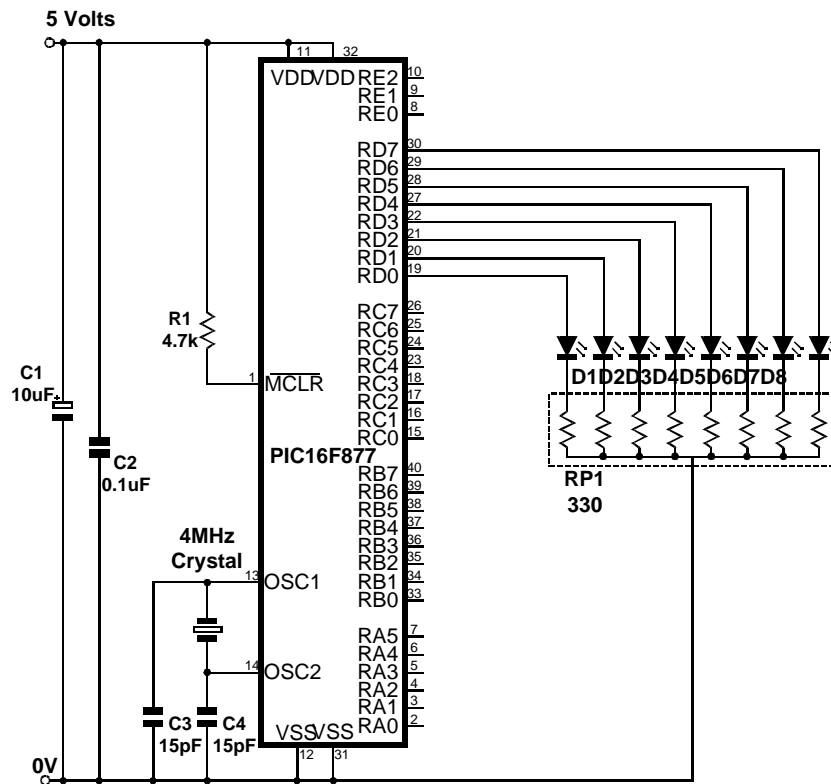
' Close the SCAN loop


GOTO LOOP

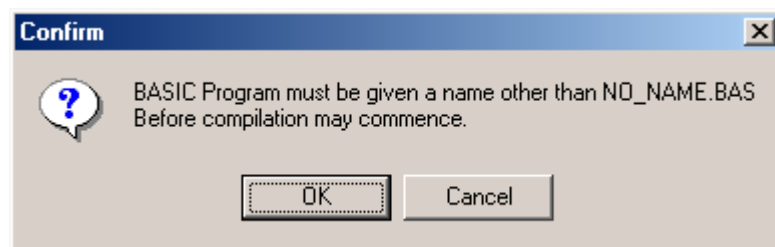
' Go back to loop and blink LED forever

If you're using a **PROTON** development board, then no extra circuitry will be required, however if not, then you will need to build a compatible circuit with 8

LEDs connected to **PORTD** of the 16F877 PICmicro. A suitable circuit is shown below. (Note that this is also a standard template for most 16F877 circuits).

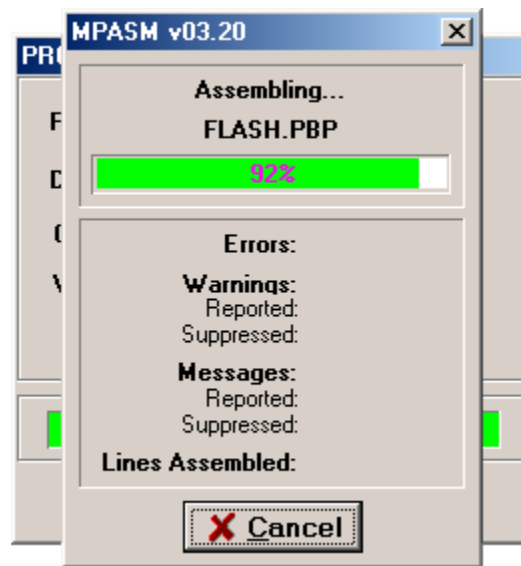


Now click on the **COMPILE** button located on the top toolbar . Because you have not given a name for the program, you will now be prompted for one.



Choose **OK**, and give the program the name **FLASH.BAS**. Not forgetting to locate the BASIC program in a suitable place on the hard drive.


One this is done, the program will compile then assemble automatically, as long as no errors where included in the typing.




You now need to place the HEX file created actually into the PICmicro. A PICmicro will sit there doing nothing at all until a program is entered into it. This is the job of a PICmicro programmer.

The PROTON+ editor gives the ability to use virtually any programmer that can accept a command line. Such as the popular EPIC programmer. See [Choosing an external Programmer](#), or [Locating the EPIC programmer on the hard drive](#) for more details concerning programmers.

If you're using the [PROTON](#) development board for design work, then there is an extremely easy method of placing the HEX file into the PICmicro, using the serial bootloader method. See [Using the serial bootloader](#) for more details.

If you've chosen a standard PICmicro programmer, then click on the **PROGRAM** icon  (located on the top toolbar). You should be presented with the programmers software. If not, then check the command line entered.

If you're using the serial bootloader option, then click the **DOWNLOAD** icon  (also located on the top toolbar).

Once the HEX file has been placed into the PICmicro, you should see each LED connected to [PORTD](#) flash in sequence. That's your first program written and operational. Welcome to the fascinating world of PICmicro microcontrollers!

It didn't work !

DON'T PANIC. There is a logical sequence of checks to make when a simple project such as flashing the LEDs hasn't worked.

First. Check your circuit for any obvious mistakes. Such as: -

Wrong component values.

- 1). Power supply correct, and correct polarity.
- 2). Is the PICmicro becoming hot to the touch. If so then check the polarity and voltage of the power supply. You may have killed the PICmicro!. Remember PICmicros do not like voltages over 6 volts. Unless they are the special voltage types.
- 3). Is the Crystal frequency correct, and crystal itself working (check either by replacement, or use an oscilloscope on CLK OUT pin).
- 4). Check that the capacitors used for the crystal are the correct values, and are not leaking.
- 5). If using a resonator with or without internal capacitors checking is important, as these devices are notorious for breaking down.
- 6). Ensure that the MCLR pin of the PICmicro is pulled up to 5 Volts with a suitable value resistor, or is connected directly to 5 Volts.
- 7). If using a breadboard for development, then ensure that it's integrity has not degraded. Breadboards become worn and loose over time, causing bad connections, or even shorts.

If you're sure that the circuit is correct, then it must be down to a software error. Just because a program compiles without any errors does not mean that it will perform the required task. There is a rather down to earth saying in the software industry "GARBAGE IN GARBAGE OUT", which means that a program written incorrectly will operate incorrectly.

Sometimes there may be clues as to what's gone wrong in the program, such as erratic behaviour. But sometimes no such clues are forthcoming, and the circuit sits there doing nothing (well, nothing that can be seen anyway!).

Check your software for any obvious errors. Such as: -

- 1). Port not set to output, or input.
- 2). **PORTA**, or **PORTE** not set to digital mode using the **ALL_DIGITAL** directive.
- 3). Wrong port used.
- 4). Wrong type of PICmicro chosen.
- 5). Wrong Crystal frequency chosen.

There are many more anomalies that could be added to the list above (countless thousands depending on the program). See [Device Specific issues](#).

When programming the device, ensure that the FUSES are set correctly. If using a 4MHz crystal, then ensure the XTAL fuse is set for XT, or HS. If a crystal frequency of more than 4MHz is used, then ensure that the fuse is set for HS. The compiler automatically assigns the appropriate fuse setting, but not all programmers do as their told! Make sure that the WATCHDOG timer is set according to the program. The compiler defaults to the watchdog timer OFF, so ensure the programmer is disabling the watchdog timer. See [CONFIG](#).

If all else fails, then you may need a helping start with a development board such as the [PROTON](#). This eases the burden of designing a circuit, and lets you get on with software development.