# Introduction to PIC Programming

## Baseline to Enhanced Mid-Range Architecture

*by David Meiklejohn, Gooligum Electronics*

## *Lesson 0: Recommended Training Environment*

### About PICs

"PIC" refers to an extensive family of microcontrollers manufactured by Microchip Technology Inc. – see www.microchip.com.

A microcontroller is a microprocessor which has I/O circuitry and peripherals built-in, allowing it to interface more or less directly with real-world devices such as lights, switches, sensors and motors. They simplify the design of logic and control systems, allowing complex (or simple!) behaviours to be designed into a piece of electronic or electromechanical equipment. They represent an approach which draws on both electronic design and programming skills; an intersection of what was once two disciplines, and is now called "embedded design".

Modern microcontrollers make it very easy to get started. They are very forgiving and often need little external circuitry.

Among the most accessible are the PIC microcontrollers.

The range of PICs available is very broad – from tiny 6-pin 8-bit devices with just 16 bytes (!) of data memory which can perform only basic digital I/O, to 100-pin 32-bit devices with 512 kilobytes of memory and many integrated peripherals for communications, data acquisition and control.

One of the more confusing aspects of PIC programming for newcomers is that the low-end devices have entirely separate address and data buses for data and program instructions. When a PIC is described as being 8- or 16-bit, this refers to the amount of data that can processed at once: the width of the data memory (registers in Microchip terminology) and ALU (arithmetic and logic unit).

The low-end PICs, which operate on data 8-bits at a time, are divided into four architectural families:

- **Baseline (12-bit instructions)**
  These PICs are based on the original PIC architecture, going back to the 1970's and General Instrument's "Peripheral Interface Controller". They are quite limited, but, within their limitations (such as having no interrupts), they are simple to work with.

  Modern examples include the 6-pin 10F200, the 8-pin 12F509 and the 14-pin 16F506
- **Mid-Range (14-bit instructions)**
  This is an extension of the baseline architecture, adding support for interrupts, more memory and peripherals, including PWM (pulse width modulation) for motor control, support for serial, $I^2C$ and SPI interfaces and LCD (liquid crystal display) controllers.

  Modern examples include the 8-pin 12F629, the 20-pin 16F690 and the 40-pin 16F887

- **Enhanced Mid-Range (14-bit instructions)**
  As the name suggests, this is an enhancement of the mid-range architecture – quite similar, but with additional instructions, simplified memory access (optimised for C compilers), and more memory, peripherals and speed.

  Examples include the 8-pin 12F1501, the 14-pin 16F1824, and the 64-pin 16F1946.

- **High-end (16-bit instructions)**
  Otherwise known as the 18F series, this architecture overcomes some of the limitations of the mid-range devices, providing for more memory (up to 128k program memory and almost 4k data memory) and advanced peripherals, including USB, Ethernet and CAN (controller area network) connectivity.

  Examples include the 18-pin 18F1220, the 28-pin 18F2455 and the 80-pin 18F8520.

This can be a little confusing; the PIC18F series has 16-bit program instructions which operate on data eight bits at a time, and is considered to be an 8-bit chip.

## The Gooligum Baseline, Mid-Range and Enhanced Mid-Range PIC Tutorials

The Gooligum tutorials introduce the baseline, mid-range and enhanced mid-range 8-bit PIC architectures, explaining the devices' internal structure, their ports (the pins used to interface with the real world) and common peripherals such as timers and analog-to-digital converters, using assembly language and C.

The tutorials are divided into a number of series (you don't have to do them all!):

- Baseline Architecture and Assembly Language
- Programming Baseline PICs in C
- Mid-Range Architecture and Assembly Language
- Programming Mid-Range PICs in C
- Enhanced Mid-Range Architecture and Assembly Language
- Programming Enhanced Mid-Range PICs in C

Some PIC tutorials focus on a single device, usually something fairly high-end, so that, by the end of the tutorials, you will have learned that device thoroughly and be well placed to learn other similar (or simpler) PICs easily, by studying the data sheets. And some tutorials note that most professional microcontroller development these days is done using C, so you might as well start by learning C.

While those approaches are perfectly valid, the philosophy behind the Gooligum tutorials has traditionally been that it's easiest to learn by starting with the least complex PICs first, without being distracted by a massive set of options (which will mostly be ignored at first), or having to say "do this, and we'll explain why later". And, for a thorough understanding of the PIC architecture, it is best to start by learning assembly language, because it's closer to the hardware – you can see exactly what is happening.

Thus, the baseline assembly language series begins with the simplest PIC devices, allowing the most basic concepts to be explained, one at a time, with more advanced devices being introduced as necessary.

Although the baseline C series recaps those explanations, it does not go into much detail – mostly just showing how the concepts from each assembly language lesson can be implemented in C.

Similarly, the mid-range assembly language lessons start with one of the simplest mid-range PICs and begin by recapping material from the baseline tutorials, with a focus on highlighting the differences and "what's new", before moving on to topics covering facilities not available in the simpler baseline PICs.

So, although it's possible to start with the mid-range C tutorials, you will gain a better, deeper understanding by starting with baseline assembly and working your way up.

For those who want to follow that path, working your way through the baseline and then mid-range tutorials, follow-up series are available on migrating to the enhanced mid-range architecture using assembly language or C. These series are intended for those who have completed the baseline and mid-range tutorials, highlighting the differences and introducing the new features of the enhanced mid-range architecture, without recapping explanations from the earlier lessons. These migration series act as a bridge from the mid-range tutorials to the new topics covers in the later enhanced mid-range lessons.

But, that's a long road to travel. The enhanced mid-range PICs are so capable and yet inexpensive that it makes little sense (unless you're chasing every cent to cost-reduce a volume product) to use baseline or mid-range PICs in new designs. Although starting with baseline PICs is an "easy" introduction to 8-bit PICs, it means learning a number of techniques which are not needed for enhanced mid-range devices.

For that reason, the enhanced mid-range assembly language series assumes no prior knowledge and does not reference the earlier baseline or mid-range lessons. It's a fresh start, designed for new students, who want to make a start with enhanced mid-range PICs.

And because C really is a very appropriate and popular choice for microcontroller development these days, the enhanced mid-range C lessons do not assume that you have completed the assembly language lessons, repeating most of the explanations[1].

## Training / Development Environment

For PIC development, you'll need:

- A desktop or laptop PC, with a spare USB port.
  Windows 7 (32- or 64-bit is ok) will give you the greatest choice of development environments and tools.
  Linux and MacOS X are viable alternatives, with most of Microchip's development tools available for and supported on those platforms.
- Development software, including assembler and editor, preferably bundled in an integrated development environment (IDE) such as MPLAB X.
- A PIC programmer, to load your program into your PIC
- A prototyping environment, such as the Gooligum Baseline and Mid-range PIC Training and Development Board, or simply a prototyping breadboard and your own supply of components, to allow you to build the example circuits in the tutorials.

And optionally:

- A C compiler.
  The C tutorials use Microchip's XC8 compiler (running in "Free mode")[2] but you may still find the lessons helpful if you are using a different compiler.
- A hardware debugger (see below)

---

[1] of course, that does make it a little more repetitive for those who have completed the enhanced mid-range assembly language lessons – but it's not difficult to skim the duplicate material

[2] the baseline C lessons also use the free CCS compiler bundled with MPLAB 8.xx

### *PIC Programmers and Debuggers*

There are many PIC programmers available, including some that you can build yourself.

Once upon a time, PICs could only be erased by shining UV light through a window on the chip (except for parts without a window, which could only be programmed once), and programmed by placing them into a special programmer.

These days, PICs use electrically erasable flash memory. They can be programmed without having to be taken out of the prototyping (or even production) environment, through a protocol called In-Circuit Serial Programming (ICSP). But instead of worrying about designing your circuit to accommodate the ICSP protocol, it can easier (especially for small PICs, where you may not have spare pins available to dedicate to the programming function) to remove the PIC from your circuit and place it into a development board or programming adapter connected to an ICSP programmer. A programming adapter is simply a minimal circuit which allows a PIC to be programmed by an ICSP programmer.

It's a really good idea to buy an ICSP programmer; you can use it with a development board or a programming adapter, while keeping the option of later using it with your own circuitry when you're ready for that.

An excellent PIC programmer to start with is Microchip's PICkit 3, shown on the right.


Although there are cheaper equivalents available[3], the PICkit 3 is not expensive, available for around US$45. And being a Microchip product, you can be sure that it will work with Microchip's development tools, and (importantly!) PICs.

The PICkit 3 can also work as a debugger, as long as the PIC you are using supports hardware debugging (meaning that it has special debug circuitry built in). Although these tutorials don't cover hardware debugging, it is a very useful facility to have available as you develop your own projects – it allows you to see exactly what the PIC is doing by stepping through your code an instruction at a time, or stopping at a particular location (a 'breakpoint'), and being able to see the contents of the PIC's internal registers and your program's variables. A hardware debugger is great for figuring out "what on Earth is that PIC doing?" And the great thing about buying a PICkit 3 as a programmer is that you get a capable hardware debugger as well, because they do that, too.

When you become more experienced, and/or work on bigger PICs, you may want to step up to a more capable (and of course more expensive) debugger, such as Microchip's ICD 3 or REAL ICE. But to start with, a PICkit 3 is ideal.

### *Development Software*

Every PIC developer should have a copy of Microchip's MPLAB integrated development environment (IDE) – even if you primarily use a third-party *tool chain* (a set of development tools that work together).

It includes Microchip's assembler (MPASM), an editor, and a software simulator, which allows you to debug your application before committing it to the chip. Not long ago, a development environment as sophisticated as this would have cost thousands. But MPLAB is free, including support from Microchip, so there is no reason not to have it. Download it from www.microchip.com.

---

[3] Such as the PICkit 2, described in the baseline and mid-range lessons. Although a capable programmer, it does not support enhanced mid-range PICs, and therefore the PICkit 2 is no longer recommended for use with these tutorials.

MPLAB directly supports the PICkit 3 as a programmer for pretty much every modern baseline, mid-range and enhanced mid-range PIC.

Microchip provides and supports two different "MPLAB" products.

For many years, MPLAB had been only available as a Windows application, and the latest versions are numbered 8.xx – the most recent at the time of writing (October 2013) is v8.92. We can refer to this as MPLAB 8. It's very stable, easy to use, and has all the features we need.

However, Microchip has found it difficult to add additional features to MPLAB, or to meet requests to have it run on platforms other than Windows. Therefore, Microchip has developed a "next generation" replacement, called MPLAB X, which also runs on Linux and Mac OS X. MPLAB 8 is now effectively be retired, with all new development, including support for new tools, compilers and PICs, being for MPLAB X.

Therefore, although the older baseline and mid-range tutorials describe both MPLAB 8 and MPLAB X, the enhanced mid-range tutorials focus only on MPLAB X[4].

MPLAB 8 includes a free copy of CCS's PCB C compiler for Windows, which supports most baseline PICs, including those used in these tutorials. Although it's now a little dated (at the time of writing, the version bundled with MPLAB was 4.073, while the latest commercially available version was 5.013), it remains useful and is used in the baseline C tutorials.

The Microchip XC8 C compiler can be downloaded from www.microchip.com. It supports all of the baseline, mid-range and enhanced mid-range PICs, and is available for Windows, Linux and OS X. It can be used for free, when running in "Free mode", but with most optimisations disabled – meaning that it generates much larger code than the paid-for commercial versions. However, code size doesn't matter much for the small example programs in these tutorials, so the free version of XC8 is used in the baseline, mid-range and enhanced mid-range C tutorials.

### *Prototyping*

If you use an ICSP programmer, then you'll need a way to connect your PIC to it, for programming. You also need to be able to test your PIC in a real circuit, before building the final design.

One solution, satisfying both these purposes, is Microchip's "PICkit 3 Low Pin Count (LPC) Demo Board". It is available for around US$26, including a PIC16F1829 and PIC18F14K22. Or you can buy a "PICkit 3 Starter Kit" bundle, including a PICkit 3 programmer, LPC Demo Board, PIC16F1829 and PIC18F14K22, for around US$60. That's excellent value; given that the MPLAB software is free, that's everything you need to get started, including a programmer, demo board and PIC chips, for only US$60!

It includes four LEDs, a pushbutton switch and a trimpot (variable resistor). Above this, a prototyping area is provided. The IC socket supports all of the modern (flash memory based) 8-, 14- and 20-pin baseline, mid-range and enhanced mid-range PICs. Note that it **does not** support the 6-pin 10F PICs, even when they are in 8-pin DIP package.

Most of the I/O lines are brought out to the 14-pin header on the side of the board, allowing it to be connected to another circuit. For example, a prototype circuit can be constructed on a breadboard and the power and signal lines connected back to the header on the LPC Demo Board. This arrangement allows you to develop a complex circuit with no need to remove the PIC from its socket for programming; the PICkit 3 can remain plugged into the LPC Demo Board during development.

---

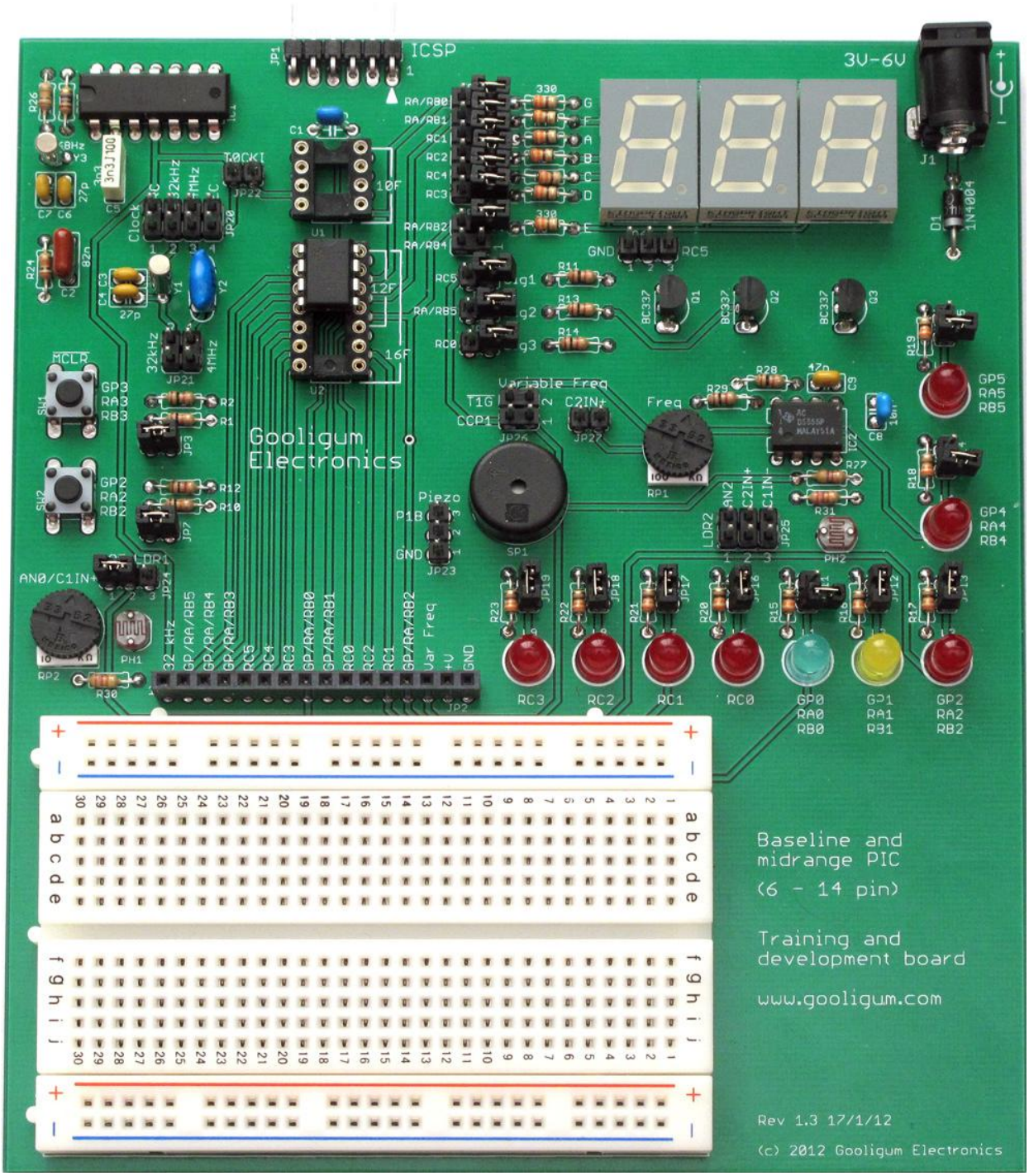
[4] MPLAB 8 supports the devices used in the enhanced mid-range PIC tutorials and can be used with those lessons if you are already comfortable with MPLAB 8. However, it is unlikely to support future enhanced mid-range devices.

Unfortunately, the LEDs on the LPC Demo Board can only be used with 14- and 20-pin PICs, not the 8-pin devices. The board also doesn't come with jumpers installed; it's a good idea to add them, so that the LEDs and trimpot can be selectively disconnected, to avoid interference with the rest of your circuit.

Many of the tutorial lessons require the use of parts not included on the LPC Demo Board, such as photocells, crystal-driven oscillator circuits, and 7-segment LED displays. Although it is possible to build all of these circuits on a breadboard, connected to the LPC Demo Board, it is a little cumbersome to do so, for some of the more complex circuits.

And of course you need to obtain all the necessary parts.

To avoid these problems, we have developed a training and development board, specifically for use with the baseline, mid-range and introductory enhanced mid-range tutorials, as shown below.



It works with a PICkit 3 programmer, and supports all 8- and 14-pin baseline, mid-range and enhanced mid-range PICs, as well as all 6-pin 10F devices (in an 8-pin DIP package). It is fully configurable using the provided jumpers, and comes with all of the hardware needed for the baseline and introductory mid-range and enhanced mid-range tutorials, including all the required PICs. Add-on parts kits, such as a motor-control kit, allow the board to be used with more topics covered in more advanced lessons.

Every PIC pin is brought out to the header at the bottom of the board, allowing the easy prototyping on the breadboard of circuits not provided by the onboard components, including circuits which require more than the 20 mA that a PICkit 3 can deliver, through the use of an external regulated DC power supply.

A number of other prototyping boards are available from various of sources, including Microchip. Some of these include more advanced peripherals, such as LCD displays, while others are intended to be an introduction to "mechatronics" (microcontroller-controlled robotics), and include motors, gears, etc. And some are intended to be general development boards, offering as much flexibility and expansion as possible. Most of these boards can of course be adapted for use with these tutorials.

However, the examples in these tutorials are intended to be used directly, without modification with the Gooligum Baseline and Mid-range PIC Training and Development Board.

### *Recommendation*

To make a start in PIC development, it's difficult to do better than the low-cost combination of:

- PICkit 3 programmer
- Gooligum Baseline and Mid-Range PIC Training and Development Board
- MPLAB X integrated development environment

These tutorials assume that you are using that recommended combination. However, most of the lesson content is of course applicable to other development environments, including the Microchip Low Pin Count Demo Board, but you may need to modify the examples to work correctly in those environments.

Other than a PC, the only other thing you need is a PIC!

If you purchase the Gooligum training board, it will come with all the necessary PICs.

Otherwise, to follow all the lessons exactly, you will need one each of:

- PIC10F200 or PIC12F508
- PIC12F509
- PIC16F506
- PIC12F629
- PIC16F684
- PIC12F1501

It is possible to adapt the lessons to other baseline and mid-range PICs by reading the data sheets, but of course it's easier to work with those listed here.

Good luck!!