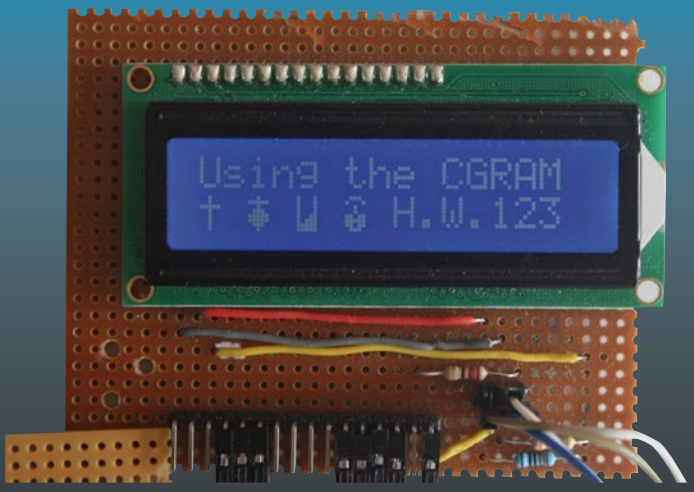
TECHNOLOGY IN ACTION™



Programming Arduino Projects with the PIC Microcontroller

A Line-by-Line Code Analysis and Complete Reference Guide for Embedded Programming in C

—

Hubert Henry Ward

Apress®

Programming Arduino Projects with the PIC Microcontroller

A Line-by-Line Code Analysis and Complete Reference Guide for Embedded Programming in C

Hubert Henry Ward

Apress®

Programming Arduino Projects with the PIC Microcontroller: A Line-by-Line Code Analysis and Complete Reference Guide for Embedded Programming in C

Hubert Henry Ward
Leigh, UK

ISBN-13 (pbk): 978-1-4842-7232-9 ISBN-13 (electronic): 978-1-4842-7230-5
https://doi.org/10.1007/978-1-4842-7230-5

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Susan McDermott
Development Editor: James Markham
Coordinating Editor: Jessica Vakili

Distributed to the book trade worldwide by Springer Science+Business Media New York, 1 NY Plaza, New York, NY 10004. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please email bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/ 978-1-4842-7232-9. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

Table of Contents

About the Author

Hubert Henry Ward has nearly 25 years of experience as a college lecturer delivering the BTEC, and now Pearson's, Higher National Certificate and Higher Diploma in Electrical & Electronic Engineering. Hubert has a 2.1 Honors Bachelor's Degree in Electrical & Electronic Engineering. Hubert has also worked as a consultant in embedded programming. His work has established his expertise in the assembler and C programming languages, within the MPLABX IDE from Microchip, as well as designing electronic circuits, and PCBs, using ECAD software. Hubert was also the UK technical expert in Mechatronics for three years, training the UK team and taking them to enter in the Skills Olympics in Seoul 2001, resulting in one of the best outcomes to date for the UK in Mechatronics.

About the Technical Reviewer

Massimo Nardone has more than 22 years of experiences in Security, Web/Mobile development, Cloud, and IT Architecture. His true IT passions are Security and Android.

He has been programming and teaching how to program with Android, Perl, PHP, Java, VB, Python, C/C++, and MySQL for more than 20 years.

He holds a Master of Science degree in Computing Science from the University of Salerno, Italy.

He has worked as a Project Manager, Software Engineer, Research Engineer, Chief Security Architect, Information Security Manager, PCI/SCADA Auditor, and Senior Lead IT Security/Cloud/SCADA Architect for many years.

Introduction

This book looks at using some useful aspects of the PIC microcontroller. It explains how to write programs in C so that you can use the PIC micro to control a variety of electronics devices and DC motors. After reading this book, you will be well on your way to becoming an embedded programmer using the C program language.

I have two main aims in writing this book. First and foremost, I have a real passion for writing programs in that I am enthralled by the way we can use simple '1's and '0's, really '5V' and '0V', to make a circuit do almost anything we want it to do. The extent of what we can get these microcontrollers to do is limited only by our imagination and experience. In my youth, we thought the flip phone used by Captain James T. Kirk in *Star Trek* was just a gimmick and we would never get there. Well, our mobile phones do a lot more than just video calls. All of what we do with our phones is done with just '1's and '0's. In writing this book, my primary goal is to hopefully inspire you guys into wanting to understand how we make the '1's and '0's work for us. That is why I will explain how all the instructions in my programs actually work and how they achieve the outcomes we want.

The second aim in writing this book is to transfer all those Arduino Sketches, which use a range of useful peripheral devices, and show you how to get them to work on a PIC microcontroller. In doing so, I hope to show you that the world of PIC controllers is not complicated and really anyone can work with them. It does not matter what experience you have with PICs, as I will assume you know nothing. This book will take the complete novice and not only allow them to run those Arduino Sketches on the PIC, but it will explain in detail every aspect of how the programs work. After reading this book, you will hopefully not only enjoy

programming PIC micros in the 'C' programming language, but should be well prepared to take on a new career as an embedded programmer.

In addition to getting to know how to program in 'C', I will show you how to build a range of circuits so that you end up using the peripherals in a practical environment. I will show you how, if you so wish, you can make your own prototype boards to program your PICs and run the programs you write. We will learn about the electronic principles of the components we use to interface the PIC to a wide range of useful peripherals.

Goals of the Book

After reading through this book, you should be able to program the PIC to use all the following peripherals:

- The seven-segment display
- The eight-by-eight matrix display
- The joystick controller
- The HC-04 ultrasonic sensor
- The stepper motor
- THE SERVO MOTOR
- The DS1307 real-time clock module
- The DHT11 humidity and temperature sensor
- The DC motor and fan
- The membrane keypad module

You should also gain a good understanding of some of the basic and advanced programming techniques for PIC micros, such as the following:

- The configuration words
- The TRIS registers and PORTS

- The timer registers
- The ADC (analog-to-digital converter) module
- The CCPM (capture, compare, and pulse width modulation) module.
- The MSSP (master synchronous serial port) module using SPI (serial peripheral interface) and I^2C (inter-integrated circuit) communication protocols.
- The EUSART (enhanced universal synchronous/asynchronous, receive and transmit) module
- Interrupts

You will be able to use an industrial IDE to write your programs in C, compile and download your programs using a variety of tools, and use the simulator in MPLABX, the IDE, to debug your programs.

You should be able to download your programs to your PIC in a practical situation where you have the ability to design and build some useful projects.

You will have learned the basic principles of a range of electronic components such as resistors, capacitors, transistors and the Darlington array, diodes, and light-emitting diodes.

The Prerequisites

There are none really, but if you understand 'C' programming it would be useful. However, I will explain how each of the programs work as we go through them.

Also, if you understand the binary and hexadecimal number systems, it would be an advantage, but there is a section in the appendix that will help you with that and I will explain the important aspects as we need them.

However, to get the full use out of this book, you will need to have installed the following software:

- MPLABX, which is the IDE from Microchip. The version in the book is MPLABX version 5.25. However, any version later than 2.20 will be OK.
- A 'C' compiler for the eight-bit micro. I use XC8 (V2.10), but with some programs I use XC8 (V1.35) compiler software. However, you should be aware that some of the later compilers have some useful libraries missing. That is why I sometimes use version 1.35.

All these programs are freely available from the Microchip website.

Another useful piece of software would be a suitable ECAD (electronic computer-aided design) software that supports eight-bit micros. The ECAD software that I have used in the past is PROTEUS. However, this is not free and so I will show you how to use a suitable prototype board to run the programs in a practical situation.

If you want to go down the practical route, you will need to purchase a programming tool and you may need to buy a prototype board. However, I do have a chapter where I show you how to make your own prototype board using vero board.

The programming tools I use are either the ICD3 can; now Microchip has moved on to the ICD4 can, or the PICkit3 programmer, to download the programs from MPLABX to the PIC.

The off-the-shelf prototype board that I may use is the PICdem2 plus demo board.

This book has been written based around using MPLABX V5.25. However, the principles of how to create projects and write programs are transferable to earlier and later versions of MPLABX. There may be some slight differences in the detail; however, these shouldn't cause you too much of a problem.

I will base the book around the following two PICs:

- The PIC16f88: This is a very versatile eight-bit micro that comes in an 18-pin or 20-pin dual inline package.
- The PIC18F4525: This is a very versatile eight-bit micro that comes in a 40-pin dual inline package.

As long as the PIC you want to use has the same firmware modules, then the programs in the book can easily be used on other PIC micros with some minor modifications. However, you should always refer to the data sheet for the particular PIC you use, as some of the SFRs (special function registers) may differ. For example, the PIC18F4525 uses the ADCON0, ADCON1, and ADCON2 special function registers to control the ADC module, but the 16F88 uses the ANSEL, ADCON0, and ADCON1 registers. I will show you how to use the datasheets, as it is important that an engineer can obtain the relevant information from datasheets.

Throughout the book, I will be including some program listings. I will then go through an analysis of any new instructions that the listings introduce. With respect to the first listing, I will assume all the instructions are new to you the reader.

I hope you will enjoy reading and learning from my book and that it will ignite that spark in you that motivates you to move into a career as an embedded programmer. Good luck and happy reading.

CHAPTER 1

Introducing MPLABX

In this chapter we are going to learn about the MPLABX, an industrial integrated development environment (IDE) from Microchip. We will learn what an IDE is and how to create a project with MPLABX. We will also learn about the configuration bits for programmable interrupt controller (PIC) micros. Finally, we will learn about header files: why we use them and how to create one in MPLABX.

MPLABX: The IDE from Microchip

An IDE is actually a collection of the different programs needed to write program instructions in our chosen language, and then convert them to the actual machine code that the micro understands and also link together any bits of program we may want to use.

The programs we need in the IDE are as follows:

- A text editor to write the instructions for the program. Note that the simple text editor Notepad could be used, but the text editor in MPLABX is by far a more advanced text editor.
- A compiler to change the instructions into a format the micro can understand.
- A linker to combine any files the programmer wants to use.

H. H. Ward, *Programming Arduino Projects with the PIC Microcontroller*,
https://doi.org/10.1007/978-1-4842-7230-5_1

- A driver that will allow the programming tool we use to load the program into the micro.
- A variety of debug tools to allow the programmer to test the program live within the micro.

All these are in, or should be in, the IDE we choose to use. Microsoft has Visual Studio, Microchip has MPLABX, and Freescale uses CodeWarrior. The Arduino has its own IDE. There is also CODEBLOCK, which is an IDE for writing generic 'C' programs that will run on your PC. As this book is based on the PIC micro, we will concentrate on MPLABX. MPLABX has an improved text editor to give the text different color codes when we save the file as an .asm or .c for c program file, such as light blue for keywords, light gray for comments, and so on.

There are some other organization programs within MPLABX, such as those that support the ability to write to the configuration registers for the PIC. There is also the ability to debug your programs within the IDE. All this makes MPLABX a useful tool for programming PICs.

There is also a program called MCC (Microchip Code Configurator). This will actually produce a lot of the code you need to use various aspects of the PIC. However, I firmly believe that you should produce the code you use yourself so that you fully understand it. I will not cover the use of MCC. Also, Microchip has not written the MCC for all its PICs, and the 18F4525 is one it has missed so far. Really, when asked who the programmer is, you should be able to say that you are and not the MCC. When you take the time to study how to write your own code, you will find it is not as hard as you first thought. Also you will get a better self-reward if you write it all yourself.

The only aspect of the programs that I let Microchip do for me is to write the code configuration bits that set up the PIC. This is only because it is so simple to do this and it covers all the #pragma statements that we used to write ourselves.

Creating a Project in MPLABX

Once you have downloaded both the MPLABX software and the XC8 (V2.10) compiler software or XC8 (V1.35), when you open the software the opening screen will be as shown in Figure 1-1.

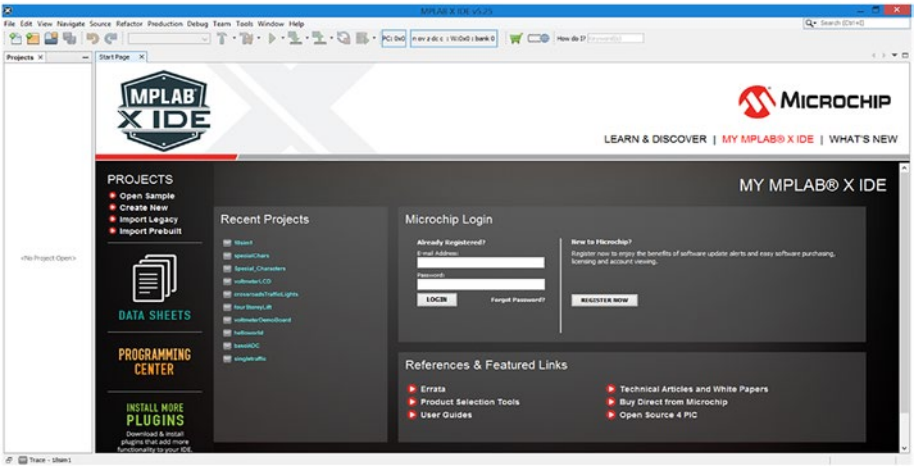


Figure 1-1. *The opening screen in MPLABX*

The project window on the left-hand side may not be shown. If you want it shown, then you should select the word "window" from the top menu bar. You should then click the mouse on the word "projects", with the orange boxes in front of it, and the window should appear. You may have to move the window about to get it in the position shown.

Now, assuming you are ready to create a project, you should either click the mouse on the word "file", in the main menu bar, and select new project, or click the mouse on the orange box, with the small green cross, on the second menu bar. This is the second symbol from the left-hand side of the second menu bar.

When you have selected the create project option, you should now see the window shown in Figure 1-2.

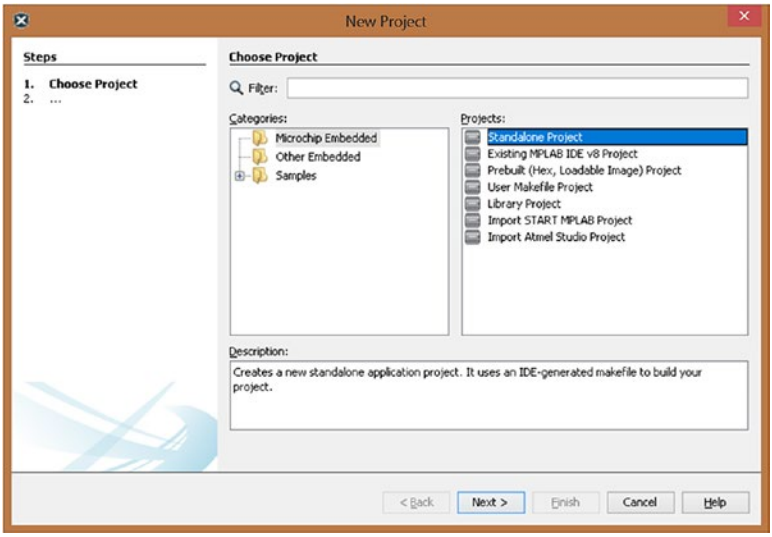


Figure 1-2. *The new project window*

Most of the projects we will create are Microchip Embedded and Standalone. Therefore, make sure these two options are highlighted and then click "Next". The select device window should now be visible as shown in Figure 1-3.

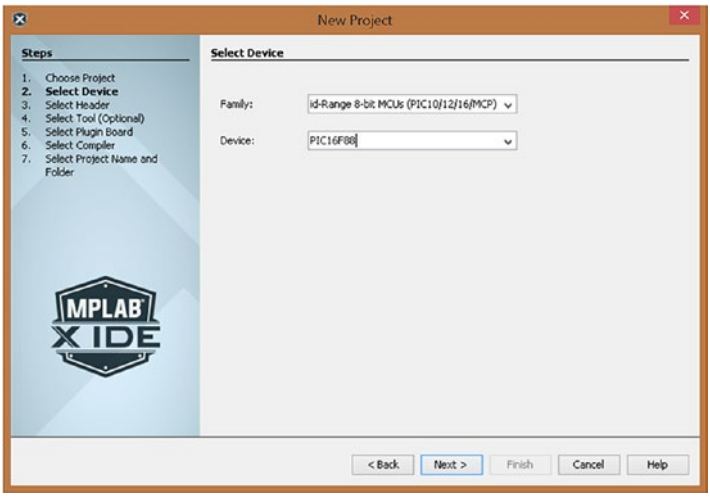


Figure 1-3. *The select device window*

In this window, we can choose which PIC we want to use. For our first project, we are going to use the PIC16F88 micro. You need to select the midrange eight-bit MCUs (PIC10, 12, 16/MCP) in the small box alongside Family, as shown in Figure 1-3. Then, in the device window, you need to select the PIC16F88. The result is shown in Figure 1-3. To make these options visible, you need to click the small downward-pointing arrows in the respective boxes. The different options should then become visible. If the device window is highlighted in blue, you could simply type in the PIC number you want (i.e., PIC16F88). Your selected device should appear in the window shown in Figure 1-3.

If you are using a different PIC then you should select it here.

Once you are happy with your selection, you need to click the next box in the window.

The next window to appear is the select tool window. This is shown in Figure 1-4. With this window, you can select the programming tool you want to use to download the program to your prototype board. There are a range of tools you can use. I mainly use the ICD3 CAN or the PICkit3 tool. We will select the PICkit3 as shown in Figure 1-4. If you are using a different tool to download your program, simply select it here. Do not worry if you don't see the serial number of your device, as shown in Figure 1-4. This would be because you have not yet connected it to your laptop.

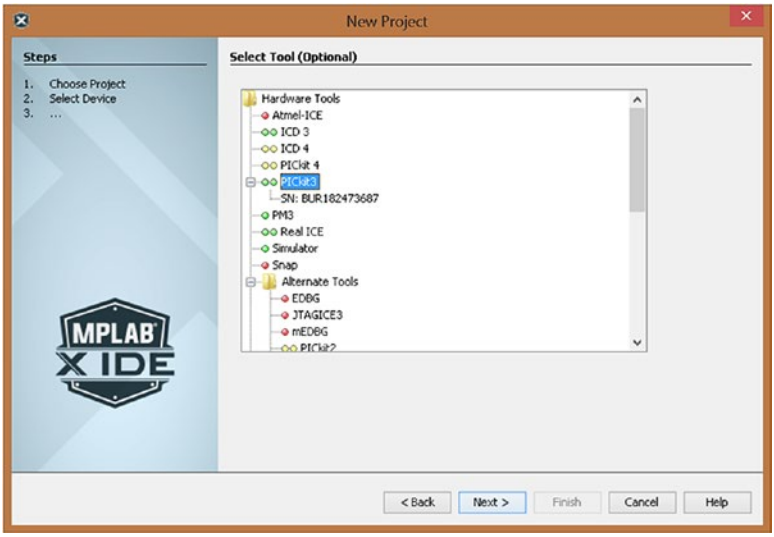


Figure 1-4. *The select tool window*

Having selected the tool you want, simply click "Next" to move on to the next window, where you can select the compiler software you want to use, assuming you have downloaded the appropriate compiler software.

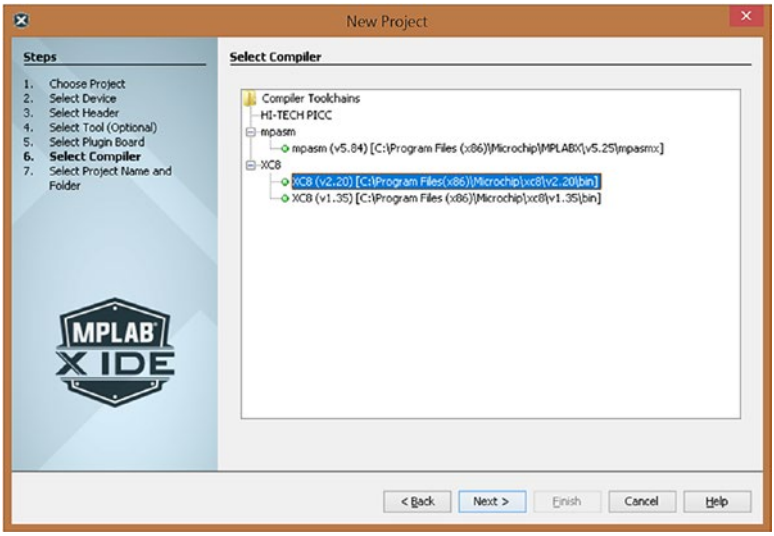


Figure 1-5. *The select compiler window*

You should select the XC8 (V2.20) compiler software, although with some later projects we will use V1.35, as shown in Figure 1-5. Then click "Next" to move to the select project name and folder window, as shown in Figure 1-6.

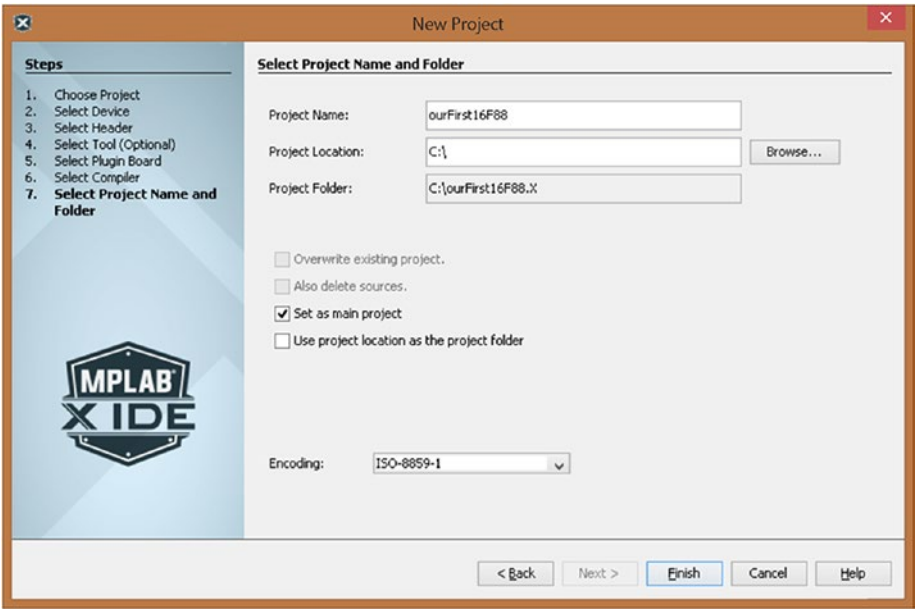


Figure 1-6. *The select project name and folder window*

In this window, you will specify the name of the project and where you want to save it. The software will create a new directory on your computer with the project name you create here. It is not recommended to use long-winded complicated path names to the new folder, so I normally save all my projects on the root directory of my laptop.

I have suggested a project name for this new project as ourFirst16F88. Note that I am using camel case, in which two words, or more, are combined together. The first letter of the first word is in lowercase and the first letters of any subsequent words are in uppercase. In this way, multiple words can be combined together to make one long word.

As you type the name for your project you should see that the folder is created on the root drive, or wherever you have specified it should be. The folder name will have a .X added to it.

It will be in this new folder that all the files associated with the project will be saved as well as some important subdirectories that are created.

Once you are happy with the naming of the project, simply click "Finish" and the project will be created. The window will now go back to the main window as shown in Figure 1-7.

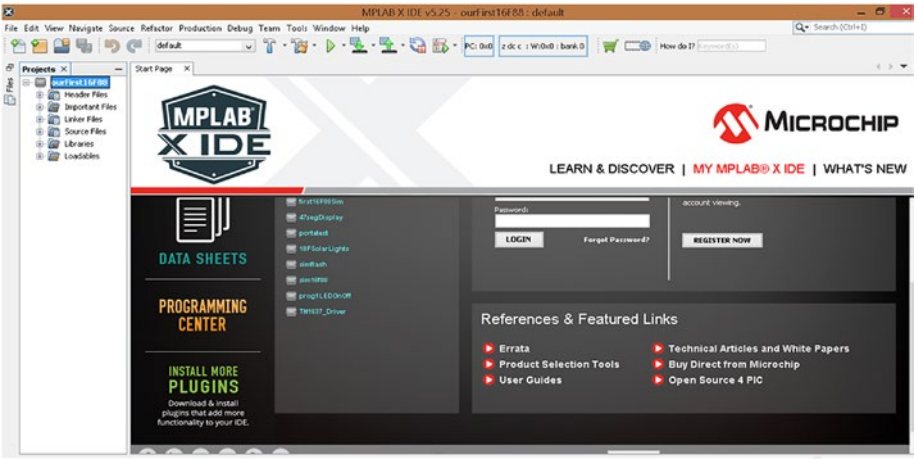


Figure 1-7. *The main window with the project created*

You should now see the project window at the left-hand side of your screen as shown in Figure 1-7. Note that you may need to move the window around to get it the same in Figure 1-7.

The Configuration Words

This section discusses one of the major differences between the PIC microcontrollers and the Arduino. It may seem rather cumbersome, but it reflects the versatility of PICs. It will also give me an opportunity to introduce you to header files and show you how we can create them.

With the Arduino, it is most likely that the configuration work has been done for you; this may make it seem that the Arduino is easier to use. However, I think it has removed the versatility of the device and restricts your understanding of the microcontroller. Certainly, MPLABX, the industrial IDE developed by Microchip, does not have any preconfigured setups for their PICs. This does give you the full range of their versatility, but it does mean you have an extra bit of studying to do before you can program the PICs. Thankfully, it is not a lot to study and Microchip has made it very easy for you to configure the PICs as you want.

The configuration words actually configure, or set up, certain aspects of the PIC and how we want to use them. The most important of these aspects is the source of the oscillator we will be using. The oscillator produces the clocking signal that synchronizes all the actions of the PIC. Microchip allows us to choose from a very wide range of sources from the simple RC (resistor capacitor) oscillator, for low frequencies that don't need to be very accurate, to the more accurate crystal oscillators with frequencies above 20Mhz. These are external devices that can be connected to the PIC. However, Microchip also gives us the choice of using oscillators that are internal (i.e., already set up inside the PIC). This can save the cost of buying an oscillator and save the use of two pins that would have to be used to connect an external oscillator to the PIC. When you use the Arduino, you don't have to worry about which oscillator source you are using because you have had that choice done for you.

There are other aspects of the PIC you need to configure, such as the WDT (watch dog timer) and the Low Voltage Programming (LVP); we will look at them now.

I have tried to explain the use of the configuration words; now let's see how we can configure them. As this is something you have to do for all your projects and for all the different PICs you might use, Microchip has tried to make it easier for you. I know when I first started writing programs for PICs it wasn't quite as easy as it is now.

To write the code for the configuration words, we use a special window in the MPLABX IDE. To open this window, click the word "window" on the main menu bar and then select "Target Memory Views" from the drop-down menu that appears. Then select "configuration bits" from the slide-out menu that appears. This process is shown in Figure 1-8.

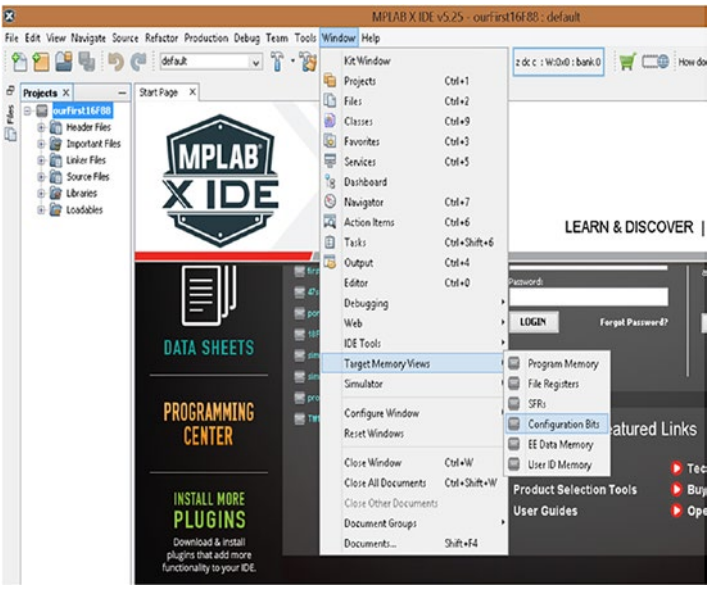


Figure 1-8. *Opening up the configuration bits*

Once you have selected the configuration bits option, your main window will change to that shown in Figure 1-9.