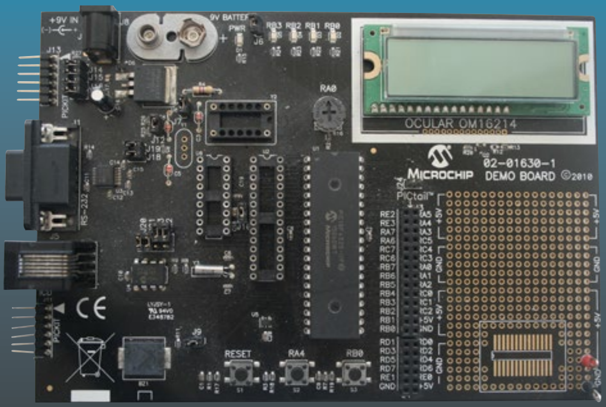
TECHNOLOGY IN ACTION™



Intermediate C Programming for the PIC Microcontroller

Simplifying Embedded Programming

—

Hubert Henry Ward

Apress®

Intermediate C Programming for the PIC Microcontroller

Simplifying Embedded Programming

Hubert Henry Ward

Apress®

Intermediate C Programming for the PIC Microcontroller: Simplifying Embedded Programming

Hubert Henry Ward
Leigh, UK

ISBN-13 (pbk): 978-1-4842-6067-8 ISBN-13 (electronic): 978-1-4842-6068-5
https://doi.org/10.1007/978-1-4842-6068-5

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Natalie Pao
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 e-mail 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 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/978-1-4842-6067-8. For more detailed information, please visit 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 and Electronic Engineering. Hubert has a 2.1 Honours Bachelor's Degree in Electrical and 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 the Skills Olympics in Seoul 2001, resulting in one of the best outcomes to date for the UK in Mechatronics.

About the Technical Reviewer

Sai Yamanoor is an embedded systems engineer working for an industrial gases company in Buffalo, NY. His interests, deeply rooted in DIY and open source hardware, include developing gadgets that aid behavior modification. He has published two books with his brother, and in his spare time, he likes to build things that improve quality of life. You can find his project portfolio at `http://saiyamanoor.com`.

Introduction

This book looks at 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 and DC motors. After reading this book, you will be well on your way to becoming an embedded programmer using the C programming language.

The Aims and Objectives of This Book

The main aim of this book is to introduce you to some useful applications of programming PIC micros such as

- Creating header files
- Controlling seven-segment displays
- Using an LCD display with two lines of 16 characters
- Pulse width modulation
- Using driver ICs such as the ULN2004A
- Controlling DC motors, including stepper motors and servo motors
- Using every aspect of the Capture, Compare and PWM, CCP module in the PIC
- Using interrupts
- Writing to the EEPROM

The Objectives of This Book

After reading through this book, you should be able to program the PIC to use all of the above. You should have a good understanding of some of the advance programming techniques for PIC micros. 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.

The Prerequisites

There are none really, but understanding the C programming language will be useful. However, I will explain how each program works as we go through them.

Also, if you understand the binary and hexadecimal number systems, it will be an advantage but there is a section in the Appendix that will help you with that.

However, to get the full use out of this book, you will need to install 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 is OK.
- A C compiler for the 8-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 are missing some useful libraries. This is why I sometimes use version 1.35.

All of these programs are freely available from the Microchip web site.

Another useful piece of software is a suitable ECAD (electronic computer-aided design) software program that supports 8-bit micros. The ECAD software I use is PROTEUS. However, it is not free, so as well as showing you how to simulate the programs in PROTEUS, 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 a prototype board.

The tools I use are either the ICD3 can (Microchip has now moved onto the ICD4 can) or the PICkit3 programmer to download the programs from MPLABX to the PIC.

The prototype board I use is the picdem2 plus DEMO BOARD and a prototype board from Matrix Multimedia (although Matrix no longer produces the more versatile board that I use).

This book was 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 details, but they shouldn't cause too many problems.

The PIC that this book is based around is the PIC18F4525. This is a very versatile 8-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 because some of the SFRs (special function registers) may differ. For example, the PIC18F4525 uses the ADCON0, ADCON1, and ADCON2 SRFs to control the ADC module but the 16F88 uses the ANSEL, ADCON0, and ADCON1 registers.

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

Before we move into the book for real, I think it will be useful to you if I explained a bit about what MPLABX is. It is an industrial IDE created by Microchip. The term IDE stands for integrated development environment. It is actually a lot of programs collected together to create a programming environment:

- There is an editor, which is slightly more than a simple text editor. However, in my early days, I used to write my programs in Notepad.
- There is also a compiler program that converts your program instructions from C to the machine code that all microprocessor-based systems use. In the very early days of programming, the programmers used to write in this machine code. This was a bit before my time, although in my early days, I wrote all my programs in assembler. Assembler is the closet language to the actual machine code that all micros use.
- There is also a linker program that will bring together any include files that we wish to use in our projects.
- As well as these programs, there are a range of programs that we can use to help debug our programs or simply simulate them.

So this IDE is a very large collection of programs that make our job of writing code much more efficient. Yet it's free; well, I use the free version, which is not as efficient as the paid version but it is more than good enough for us.

I therefore hope that you not only learn how to program the PIC micro but you also enjoy going through my book and that you produce some useful projects along the way.

CHAPTER 1

Creating a Header File

In an effort to reduce the amount of text in the program listings and reduce the amount whereby I simply repeat myself, let's create and use a series of header files. Header files are used when your programs use the same series of instructions in exactly the same way in all your projects and programs.

In this book, you will create three header files. The first will be concerned with the configuration words you write for your projects. The configuration words are used to configure how the PIC applies the essential parameters of the PIC. They have to be written for every project and program you create. Therefore, if you are going to write the same configuration words for all of your projects, you should use a header file.

The second header file will be associated with setting up the PIC to use the ports, the oscillator, the timers, etc. You will set them in exactly the same way in all of your projects, so it's useful to create a header file for this. However, in some projects you may need to modify some of the settings, so be careful when using this header file.

The third header file you will create will be used if your programs use the LCD (liquid crystal display) in exactly the same way such that

- The LCD is always connected to PORTB.
- The LCD uses just four data lines instead of eight to save I/O.
- The RS pin is always on Bit4 of PORTB and the E pin on Bit5 of PORTB.

H. H. Ward, *Intermediate C Programming for the PIC Microcontroller*,
https://doi.org/10.1007/978-1-4842-6068-5_1

- The LCD always increments the cursor position after each character has been displayed.
- The LCD always uses 2 lines of 16 characters.
- The actual characters are always on a 5 by 8 grid.

If this is all true, you should create a header file for the LCD.

These are the three header files you will create in this book. There are many more examples of when you should create a header file. The process of creating and using header files makes your program writing more efficient.

Header files can be made available for all of your projects, like global header files as opposed to local header files. Local header files are available only to the project they were created in.

Also, you can split projects up so that different programmers can write different sections of the programs and save them as header files to be used in all projects by all of the company's programmers.

Creating a Header File

Now that I have explained what header files are and why you would use them, let's create one. The first header file you will create will be for the configuration words that you will use for most of the projects in this book. It will also give me the chance to go through creating a project in MPLABX for those readers who have never used MPLABX before. The version I am using is MPLABX V5.25. It is one of the latest versions of the IDE from Microchip. Microchip is always updating the software, but the main concepts of creating a project and writing programs do not change. You will be able to follow the process even if you have an earlier version of MPLABX or a later version.

Creating a Project in MPLABX

Assuming 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 look like Figure 1-1.

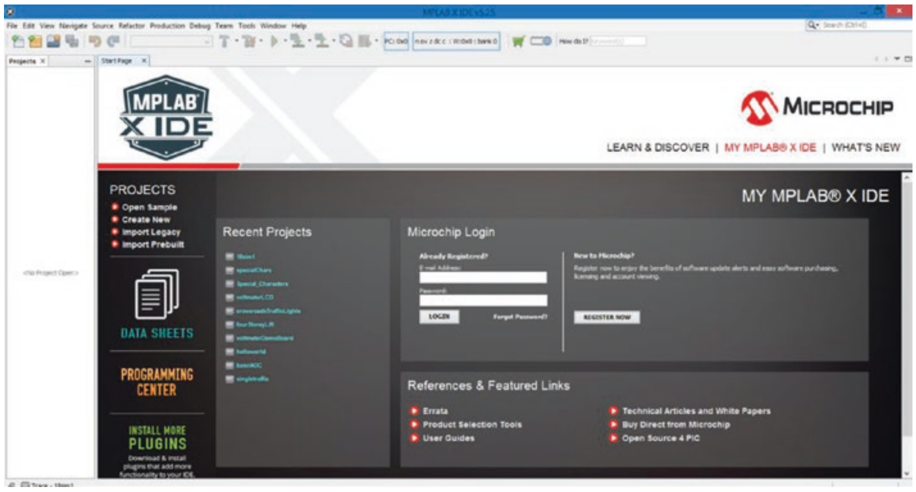
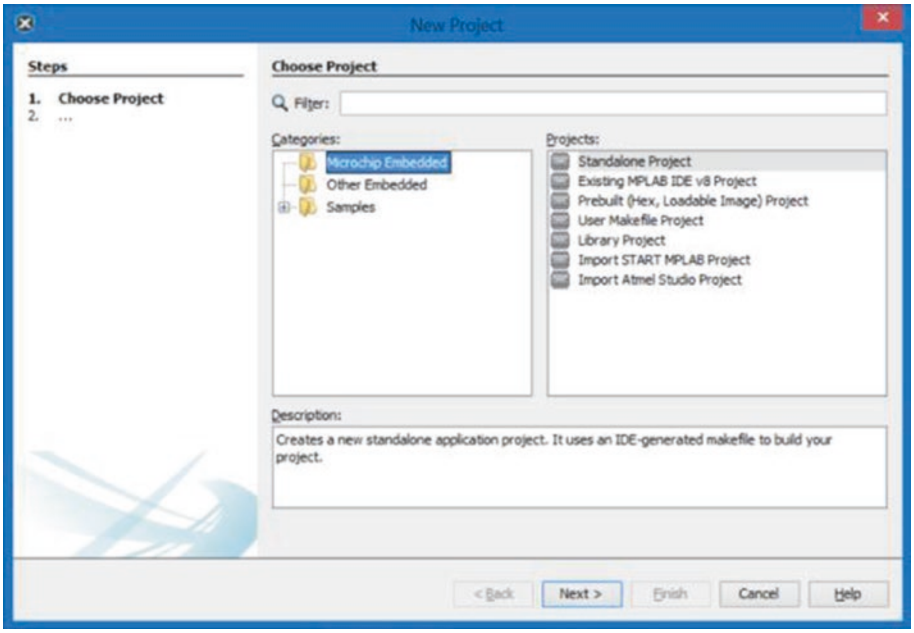


Figure 1-1. *The opening screen in MPLABX*

The project window on the left-hand side may not be shown. If you want it shown, you should select the word Window from the top menu bar. Click 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 word File, in the main menu bar, and select New project, or click 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.

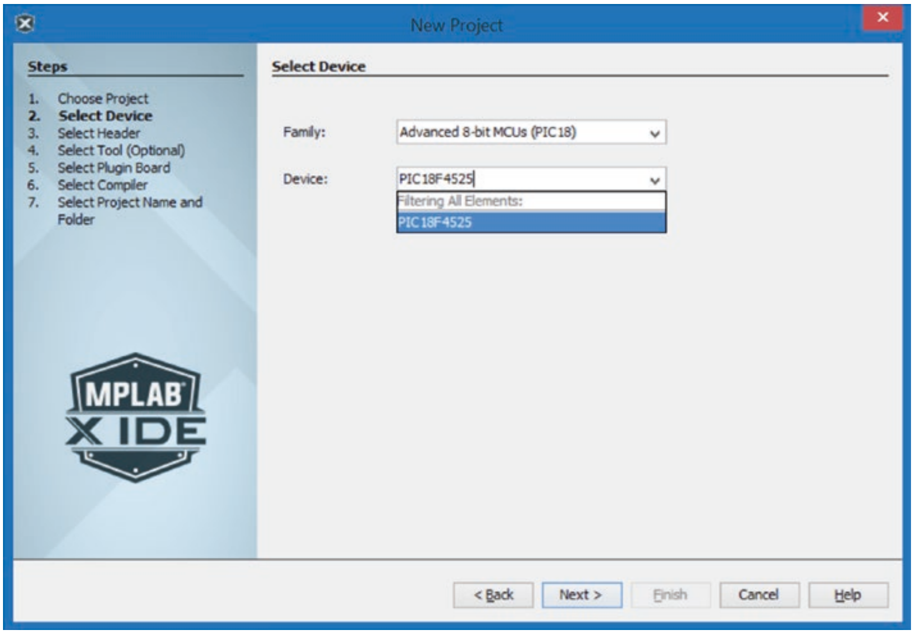
When you have selected the Create project option, you should see the window shown in Figure 1-2.



Creates a new standalone application project. It uses an IDE-generated makefile to build your project.

Figure 1-2. *The New Project window*

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



Family: Advanced 8-bit MCUs (PIC18)

Device: PIC18F4525

Filtering All Elements:

PIC18F4525

Figure 1-3. *The Select Device window*

In this window, you can choose which PIC you want to use. Select the Advanced 8-bit MCUs (PIC18) in the small box alongside Family, as shown in Figure 1-3. Then, in the Device window, select the PIC18F4525. The result is shown in Figure 1-3. To make these options visible, you need to click the small downward pointing arrow in the respective box. 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, such as PIC18F4525. Your selected device should appear in the window below.

If you are using a different PIC, select it here.

Once you are happy with your selection, click the Next button.

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.

However, if I am only simulating the program, I use the simulator option. Note that the MPLABX IDE comes with its own simulations for the PICs you may use. It also has a wide range of tools that allow us to simulate and test programs within MPLABX all without having a real PIC. You will use the simulator in this project, so select the simulator option shown in Figure 1-4.

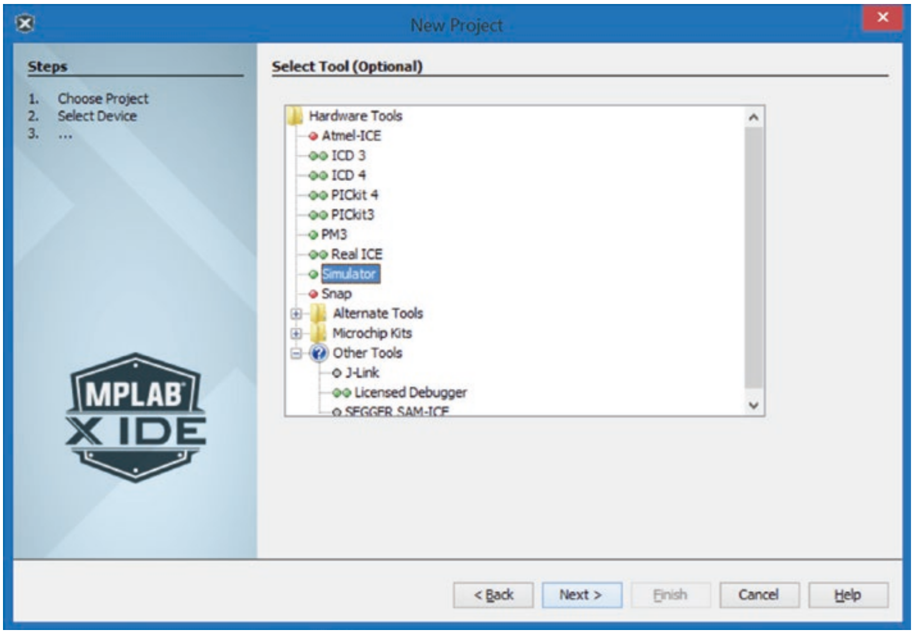
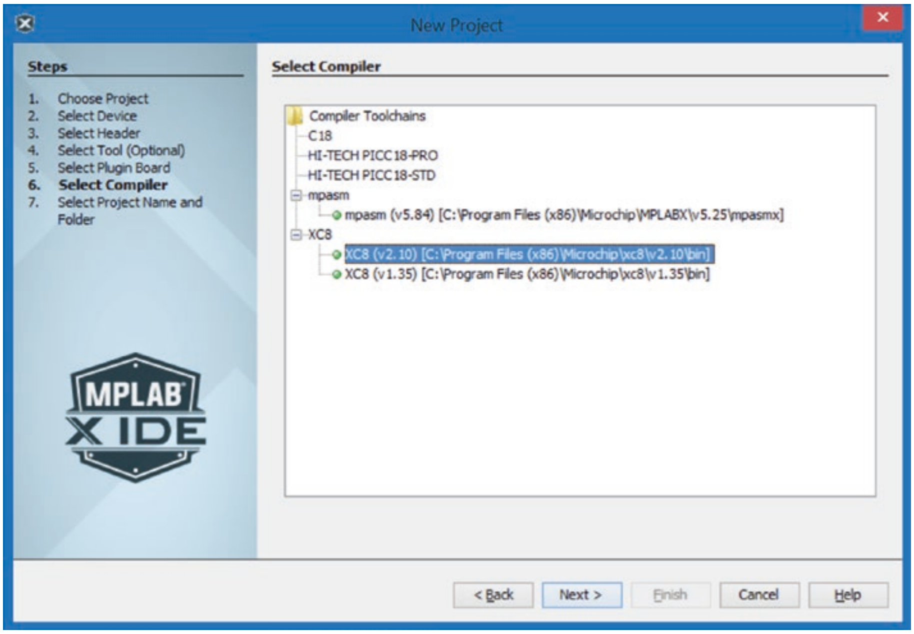


Figure 1-4. *The Select Tool window*

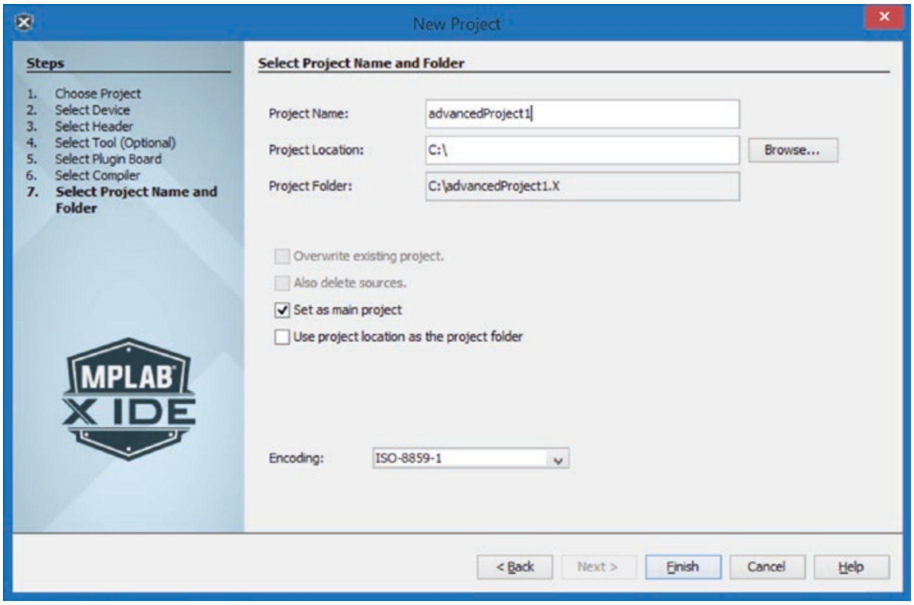
Having selected the tool you want, click Next to move on to the next window where you can select the compiler software you want to use, assuming you have downloaded the appropriate compiler software (see Figure 1-5).



Compiler Toolchains
C18
HI-TECH PICC18-PRO
HI-TECH PICC18-STD
mpasm
mpasm (v5.84) [C:\Program Files (x86)\Microchip\MPLABX\v5.25\mpasmx]
XC8
XC8 (v2.10) [C:\Program Files (x86)\Microchip\xc8\v2.10\bin]
XC8 (v1.35) [C:\Program Files (x86)\Microchip\xc8\v1.35\bin]

Figure 1-5. *The Select Compiler window*

You should select the XC8(V2.10) compiler software, although with some later projects you will use V1.35, as shown in Figure 1-5. Then click Next to move to the Select Project Name and Folder window shown in Figure 1-6.



Select Project Name and Folder

Project Name: advancedProject1

Project Location: C:\

Project Folder: C:\advancedProject1.X

Overwrite existing project.

Also delete sources.

[✓] Set as main project

Use project location as the project folder

Encoding: ISO-8859-1

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 recommended that you don't use long-winded, complicated path names for 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 advanceProject1. Note that I am using camelcase, where 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 the Finish button and the project will be created. The window will now go back to the main window, as shown in Figure 1-7.

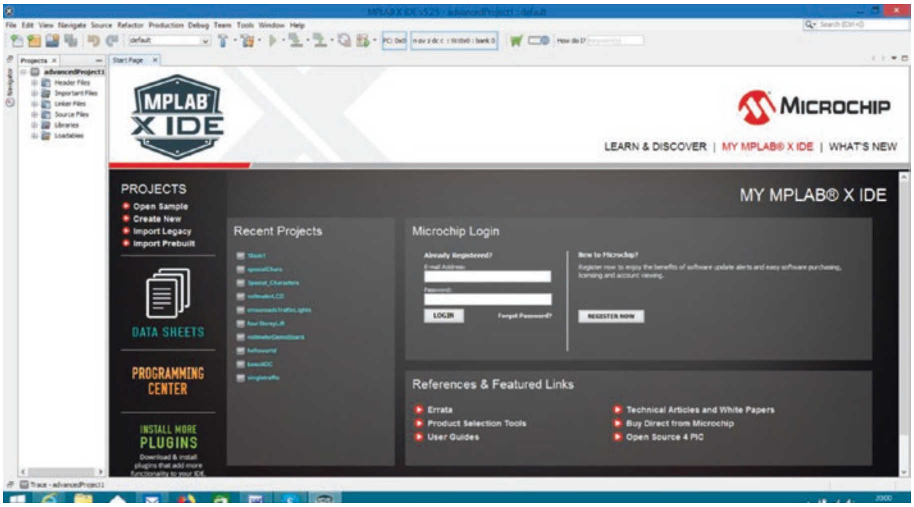


Figure 1-7. *The main window with the project created*

You should 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 about to get it the same as that shown in Figure 1-7.

Now that you have the new project created, you need to create a header file that you will use in all of your projects in this book.

To create the header file, right-click the subdirectory in the project tree named Header Files. When you do this, the flyout menu will appear, as shown in Figure 1-8.

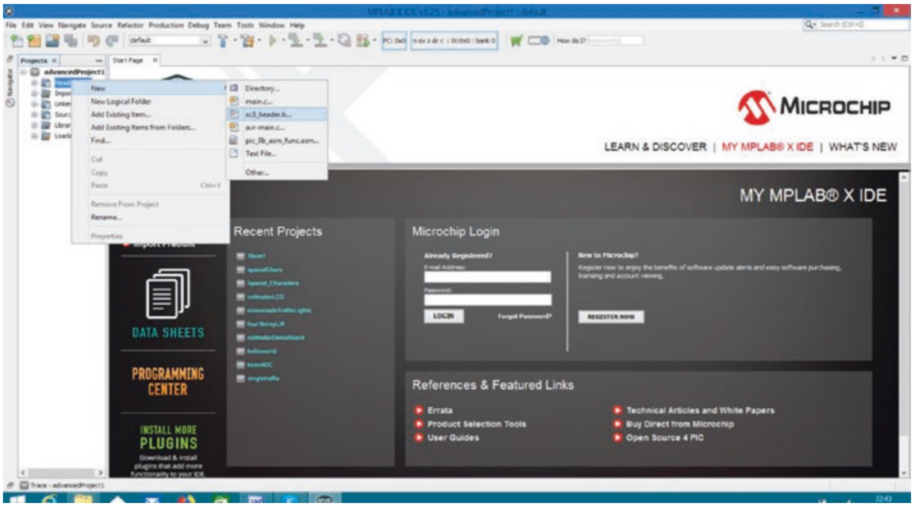
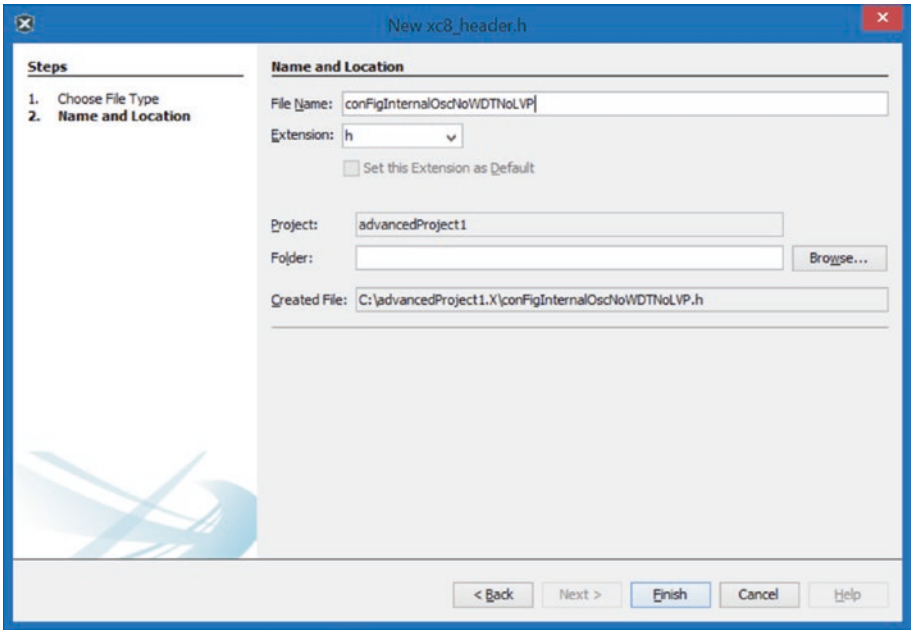


Figure 1-8. *The flyout menu for the new header file*

From that flyout menu, select New. From the second flyout menu, select `xc8_header.h`, as shown in Figure 1-8.

The window shown in Figure 1-9 will appear.



File Name: conFigInternalOscNoWDTNoLVP
Extension: h
Set this Extension as Default
Project: advancedProject1
Folder:
Created File: C:\advancedProject1.X\conFigInternalOscNoWDTNoLVP.h

Figure 1-9. *The name and location for the new header file*

All you need to do here is give the file a name. I have chosen the name conFigInternalOscNoWDTNoLVP as it gives a good description of what I want to do in this header file, which is set these three main parameters of the configuration words. Note the configuration words specify how you want to configure and so use the PIC.

The main concern is that PICs have a wide variety of primary oscillator sources and you need to tell the PIC which one you will be using. The oscillator is the device or circuit that provides a signal from which the clock signal, the signal that synchronizes the operations of the PIC, is derived. I prefer to use the internal oscillator block as the primary oscillator source. This saves buying an oscillator crystal. It also saves two inputs that would be used if I used an external oscillator. This is because I would connect the external oscillator to the PIC via those two input pins, normally RA6 and RA7.

The second major item I change is to turn off the WDT, which is the watch dog timer. This is a timer that will stop the micro if nothing has happened for a set period of time. This is a facility that you don't want in these programs, so you must turn it off. Note that the WDT is mainly used in continuous production lines. In that situation, the fact that nothing has happened for a set time usually means something has gone wrong so it's best to turn everything off.

The third item to turn off is the low voltage programming (LVP) function. The low voltage programming affects some of the bits on PORTB. Therefore, to keep the bits on PORTB available for general I/O, I normally turn off the LVP.

So this explains the header file's cryptic name. You should always give your header files a name that relates to how you want to use the file.

Once you have named the header file, click Finish and the newly created header file will be inserted into the main editing window in the software. However, Microchip automatically inserts an awful lot of comments and instructions that, at your level of programming, you don't really need. Therefore, simply select all that stuff and delete it so that you have an empty file ready for you to insert the code that you really want.

Now that you have a clean file, you can control what goes into it. The first thing you should do is put some comments in along the following lines:

- You should tell everyone that you wrote this code.
- You should say what PIC you wrote it for and when you wrote it.
- You should explain what you are trying to do with it.

There are two types of comments in C programs, which are

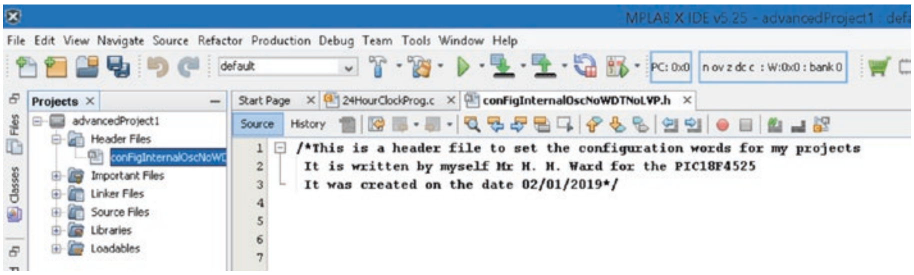
- **Single-line comments**: They start with two forward slashes (//). Anything on the same line after the two forward slashes is ignored by the compiler as they are simply comments. For example,

  ```
  //these words are just comments
  ```

- **Multiple lines of comments or a paragraph of comments**: This is text inserted between the following symbols: /* */. For example,

  ```
  /* Your comments are written in here */
  ```

So insert a paragraph of comments as shown in Figure 1-10.


MPLAB X IDE v5.25 - advancedProject1 - def
File Edit View Navigate Source Refactor Production Debug Team Tools Window Help
default
PC: 0x0
n ov z dc c : W:0x0 : bank 0
Projects
advancedProject1
Header Files
conFigInternalOscNoWD
Important Files
Linker Files
Source Files
Libraries
Loadables
Start Page
24HourClockProg.c
conFigInternalOscNoWDTNoLVP.h
Source
History

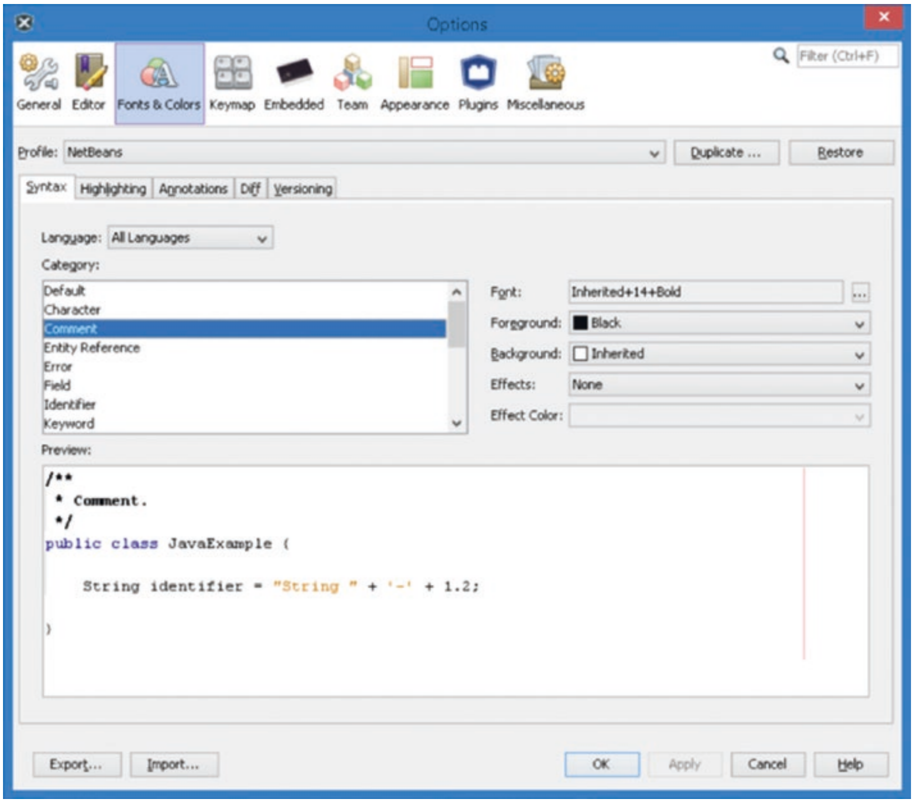
```
/*This is a header file to set the configuration words for my projects
 It is written by myself Mr H. H. Ward for the PIC18F4525
 It was created on the date 02/01/2019*/
```

Figure 1-10. *The comments for the header file*

You should insert your own comments into the editor similar to those shown in Figure 1-10.

You will notice that I changed the colour of my comments to black and bold size 14. This is to try and make them more visible than the default grey.

If you want to change the colour, you can do so by selecting the word Options from the drop-down menu that appears when you select the Tools choice on the main menu bar. You will get the window shown in Figure 1-11.



Profile: NetBeans

Language: All Languages

Category:

- Default
- Character
- Comment
- Entity Reference
- Error
- Field
- Identifier
- Keyword

Font: Inherited+14+Bold

Foreground: ■ Black

Background: □ Inherited

Effects: None

Effect Color:

Preview:

```
/**
 * Comment.
 */
public class JavaExample {

    String identifier = "String " + '-' + 1.2;

}
```

Figure 1-11. *Changing the font and colours*

Click the tag for Fonts and Colours and then select what you want to change. Once you are happy with your choice, click OK. I changed the colour of the comments to black, as shown in Figure 1-11.

Now you need to create the configuration words for your header file. As this is something you must do for all your projects, Microchip has developed a simple process for writing to the configuration words. This can be achieved using 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 1-12.

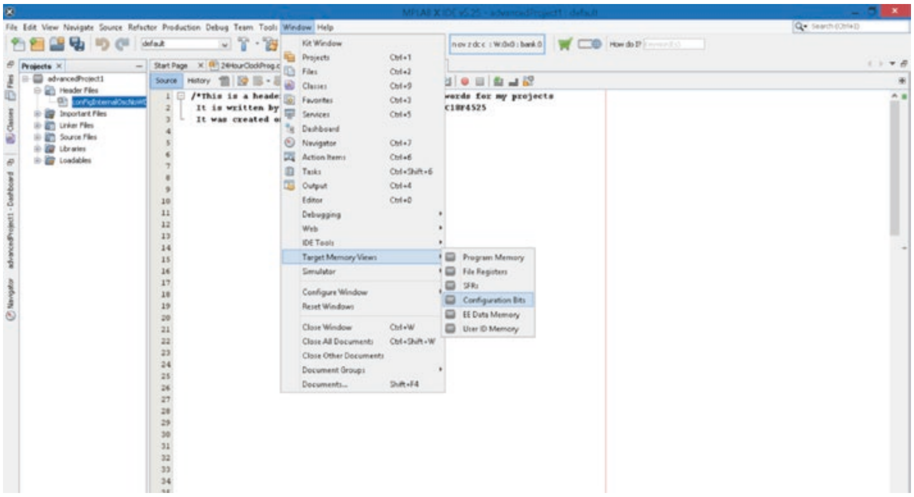
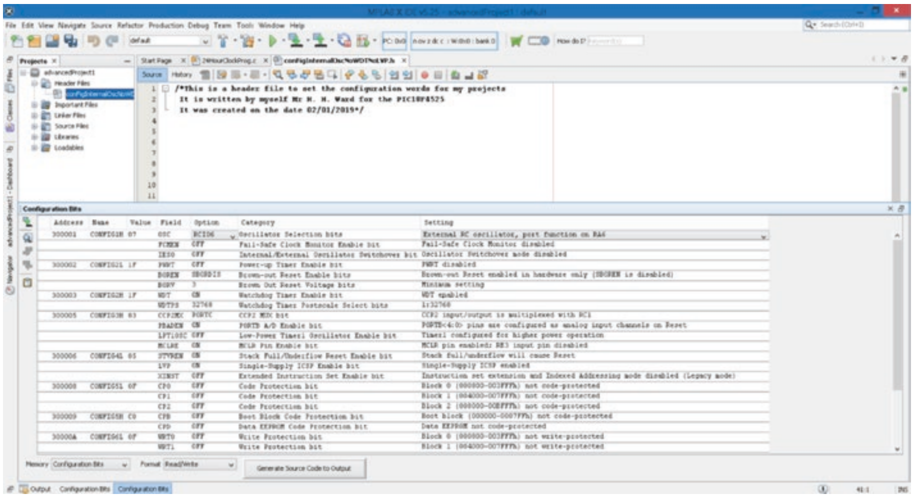


Figure 1-12. *Selecting the configuration bits*

Once you have selected the configuration bits, your main window will change to that shown in Figure 1-13.



```
/*This is a header file to set the configuration words for my projects
 It is written by myself Mr H. H. Ward for the PIC18F4525
 It was created on the date 02/01/2019*/
```

Configuration Bits

Address	Name	Value	Field	Option	Category	Setting
300001	CONFIG1H	07	OSC	RCIO6	Oscillator Selection bits	External RC oscillator, port function on RA6
			FCMEN	OFF	Fail-Safe Clock Monitor Enable bit	Fail-Safe Clock Monitor disabled
			IESO	OFF	Internal/External Oscillator Switchover bit	Oscillator Switchover mode disabled
300002	CONFIG2L	1F	PWRT	OFF	Power-up Timer Enable bit	PWRT disabled
			BOREN	SBORDIS	Brown-out Reset Enable bits	Brown-out Reset enabled in hardware only (SBOREN is disabled)
			BORV	3	Brown Out Reset Voltage bits	Minimum setting
300003	CONFIG2H	1F	WDT	ON	Watchdog Timer Enable bit	WDT enabled
			WDTPS	32768	Watchdog Timer Postscale Select bits	1:32768
300005	CONFIG3H	83	CCP2MX	PORTC	CCP2 MUX bit	CCP2 input/output is multiplexed with RC1
			PBADEN	ON	PORTB A/D Enable bit	PORTB<4:0> pins are configured as analog input channels on Reset
			LPT1OSC	OFF	Low-Power Timer1 Oscillator Enable bit	Timer1 configured for higher power operation
			MCLRE	ON	MCLR Pin Enable bit	MCLR pin enabled; RE3 input pin disabled
300006	CONFIG4L	85	STVREN	ON	Stack Full/Underflow Reset Enable bit	Stack full/underflow will cause Reset
			LVP	ON	Single-Supply ICSP Enable bit	Single-Supply ICSP enabled
			XINST	OFF	Extended Instruction Set Enable bit	Instruction set extension and Indexed Addressing mode disabled (Legacy mode)
300008	CONFIG5L	0F	CP0	OFF	Code Protection bit	Block 0 (000800-003FFFh) not code-protected
			CP1	OFF	Code Protection bit	Block 1 (004000-007FFFh) not code-protected
			CP2	OFF	Code Protection bit	Block 2 (008000-00BFFFh) not code-protected
300009	CONFIG5H	C0	CPB	OFF	Boot Block Code Protection bit	Boot block (000000-0007FFh) not code-protected
			CPD	OFF	Data EEPROM Code Protection bit	Data EEPROM not code-protected
30000A	CONFIG6L	0F	WRT0	OFF	Write Protection bit	Block 0 (000800-003FFFh) not write-protected
			WRT1	OFF	Write Protection bit	Block 1 (004000-007FFFh) not write-protected

Memory: Configuration Bits — Format: Read/Write

Figure 1-13. *The configuration bits*

You may have to drag the window up to make it as larger as shown in Figure 1-13.

This configuration window allows you, as the programmer, to select some very important options for the PIC, the most important being the primary oscillator type and source used and if you want the watch dog timer or not.

There are three main options you need to change at this point. You should change

- The OSC to INTIO67. This is done by selecting the small arrow alongside the box next to the OSC option. The default setting is usually RCI06, the resistor capacitor oscillator with bit6 on PORTA left as a normal I/O bit. You need to change this. When you click the small arrow next to the OSC option RCIO6, a small window will open. If you move the selection up to the next one, it will be the one you want, INTIO67, which means you will use the internal oscillator block as the primary source and leave Bits 6 and 7 on PORTA as normal I/O bits. Note that when you select this, a description of the change will appear in the description window alongside this tag and it will have a blue colour to the text.
- The next change is simpler. Set the WDT to OFF. It important to turn the WDT off because if nothing happens for a predefined period of time in a program, the WDT will stop the program. You don't want this to happen so you must turn the WDT off.
- The third change is to turn the LVP off.

Once you have changed these settings, you can generate the source code and then paste this code into your program. To do so, click the