Experiment 1: Introduction to MPLAB ® X-IDE

Objectives

The aim of this tutorial is to teach the students some of the aspects of an important tool that can assist in software development. In this lab, we will use the MPLAB ® X integrated development environment (IDE) to write microprocessor C programs for the ChipKIT^{TM} Pro MX7 processor board. This described synthesis and tool analysis will be used throughout this course.

Overview

The target system for this Lab is the Digilent ChipKIT^{TM} Pro MX7 processor board. Conventional IDE systems consist of software that runs on the PC or Linux computer and special hardware that manages the microcontroller on the target system. A separate PIC microcontroller on the ChipKIT^{TM} Pro MX7 processor board provides the special hardware needed to interface with the PC that is running the MPLAB ® X IDE. The IDE assists the program developer in converting one or more text files written using C or assembly language instructions into a file that contains a set of binary instructions that is programmed into a microprocessor.

The MPLAB ® X IDE does this transformation in four steps. First, the editor is where the programmer writes his or her C language programs. A project consists of one or more text files that the compiler converts into one or more object files. The object files are combined using a linker that generates a single file that the loader then programs into the microprocessor.

Source files with the ".c" extension contain C statements that define the functionality of the file such as the interface to an LCD or the implementation of a software time delay. Files with the ".h" extension are called header files that contain C instructions and directives that allow the source code to be reused in different application. Generally each ".c" file has a companion ".h" file that contains function prototypes and definitions.

The MPLAB ® X editor has many features to help you to write C code that the compiler can convert into an object file. The compiler checks each source file for proper C syntax and that all references are resolved. Compiler errors are identified with the file name, the line number in the file where the error was detected, and a description of the error. The link and load processes will be completed only if the compiler completes its task without finding any errors. There may be times that the compiler generates warnings. In such cases, the link and load operations will still be completed. Compiler warnings are generated if the source code attempts to give the compiler an instruction that is ambiguous or confusing but the syntax is correct. In these cases, the compiler makes a reasonable but not necessarily correct interpretation of your instruction. Clean code is code that compiles with no errors or warnings when using the setting for the highest level of error and warning generation.

However, the fundamental skill to learn in this tutorial is how to use the MPLAB IDE to develop software that runs on the PIC32 processor. The steps in this exercise are designed to show you how to create a new stand-alone project and how to use the software instrumentation to verify that the program is executing as designed.

MPLAB ® X IDE Tutorial

Connecting the ChipKIT™ Pro MX7 Processor Board

Figure 4 is a picture of the chipKITTM Pro MX7 processor board. Before connecting the debugging USB cable to the PC, verify that the Board Power select jumpers shown at the upper left are positioned correctly. As shown, the processor board is powered using the debugging PC USB connection. The board power switch is located in the lower left corner. Once the board power is turned on, the RED power indication LED should be on. It is helpful to connect the processor board with the debugging USB cable and power it on before launching a new MPLAB ® X project.

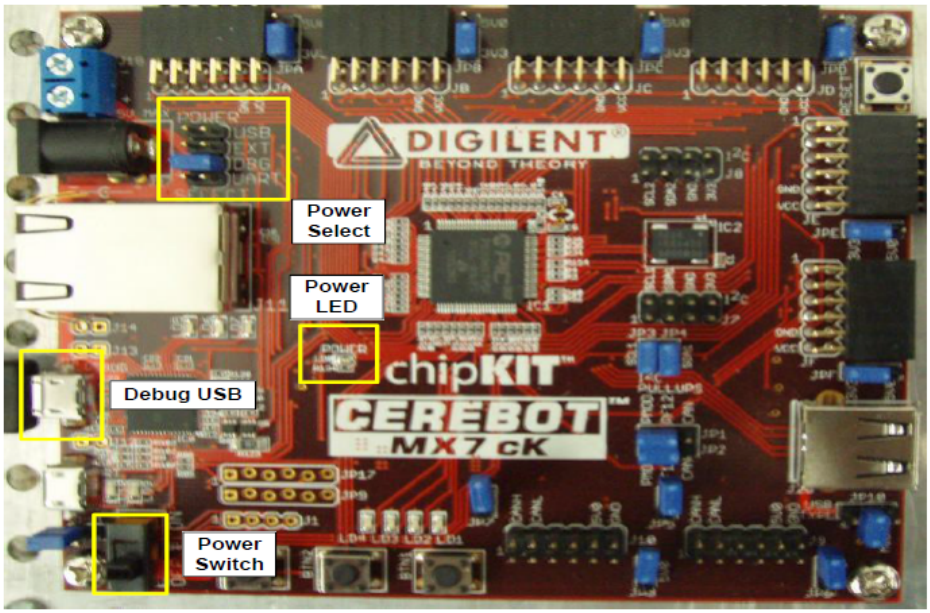

Figure 4: ChipKIT™ Pro MX7 processor board

Experiment

Part 1: Launching a New MPLAB Harmony Project

In this lab, we will be using the MPLAB Harmony Framework in order to program applications for PIC32. MPLAB Harmony is a flexible, abstracted, fully integrated firmware development platform for PIC32 microcontrollers. It takes key elements of modular and object oriented design, adds in the flexibility to use a Real-Time Operating System (RTOS) or work without one, and provides a framework of software modules that are easy to use, configurable for your specific needs, and that work together in complete harmony. [3]

MPLAB Harmony includes a set of peripheral libraries, drivers and system services that are readily accessible for application development. The code development format allows for maximum re-use and reduces time to market.

The MPLAB Harmony framework should be installed successfully on your machine before performing the following steps. It is not included by default in the MPLAB tool. It is available on the microchip website (`http://www.microchip.com/mplab/mplab-harmony`).

When installing the MPLAB Harmony framework on Windows, make sure not to change the default installation path which is usually (`C:\microchip\harmony\v1_08_01`). This path will be used later in the integration with MPLAB and creation of new harmony projects.

1. Launch the MPLAB ® X IDE by selecting the MPLAB ® X icon on the PC desktop. Initially, a window for a blank project page as shown in Figure 5 will appear.

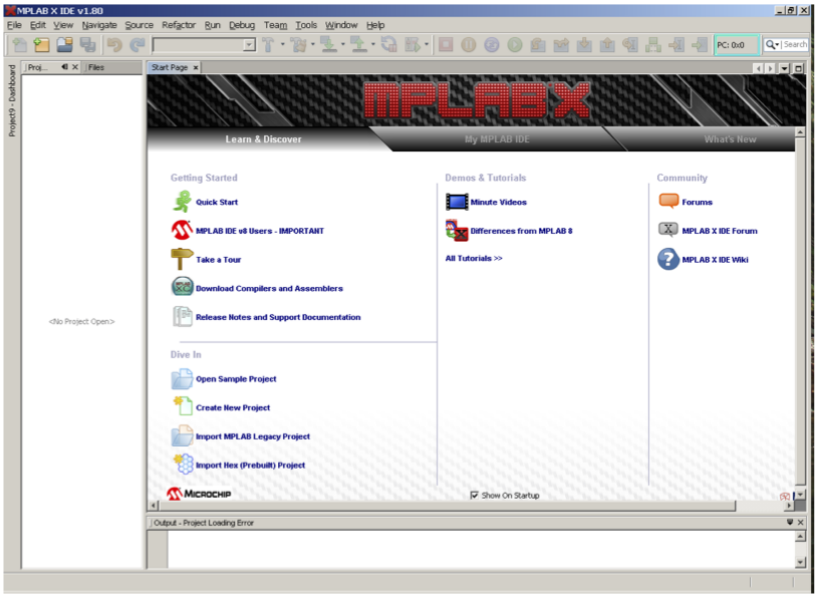

Figure 5: MPLAB ® X IDE window

2. Select **New Project** from the MPLABX IDE File menu, which opens the new Project Dialog and launches the New Project Wizard.

3. As shown in Figure 6, the first step is **Choose Project**. Select **Microchip Embedded** then the **32-bit MPLAB Harmony Project**. Then, click **Next** to open the Name and Location pane.

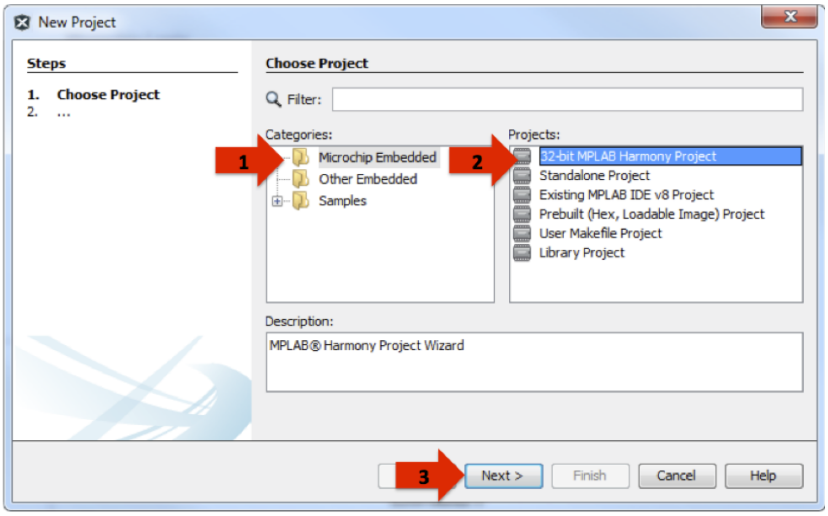

Figure 6: Creating a new MPLAB Harmony project

4. The second step is to choose the Name and Location of the project. First, you have to specify the **Harmony Path** which is where the Harmony framework is installed (`C:\microchip\harmony\v1_`

08_01). Then, choose your project location, name and initial configuration as shown in Figure 7. In this lab, we are using Chipkit mx7ck which has PIC32MX795F512L processor, so in the **Device Family** you have to choose **PIC32MX** and the **Target Device** is PIC32MX795F512L. When you click **Finish**, the New Project wizard will create several new folders on the disk within the project folder and an empty MPLAB Harmony Project.

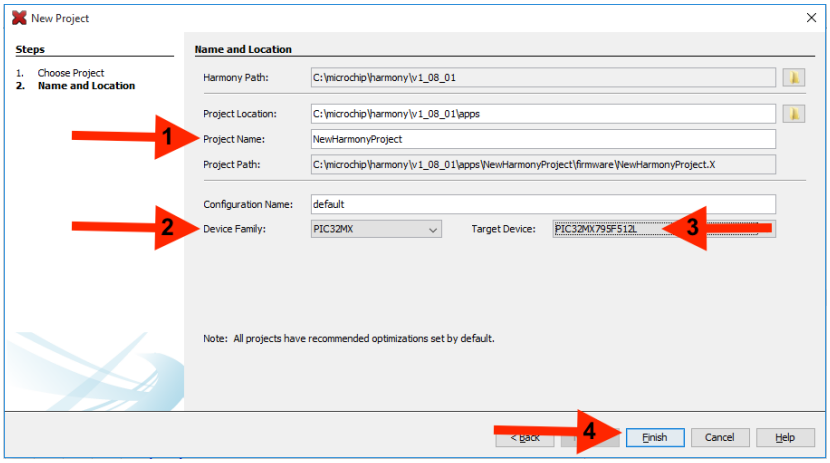

Figure 7: Selection of Name and Location of Harmony project.

MPLAB Harmony Configurator (MHC)

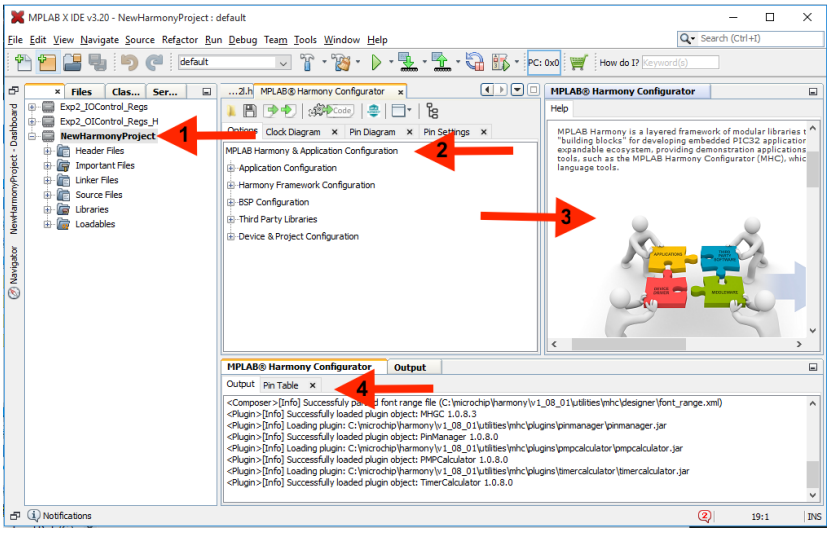

Figure 8: Different panes of MPLAB Harmony project.

After creating the empty project, the New Project wizard will launch the **MPLAB Harmony Configurator (MHC)** plugin so as to configure the project[1]. From the MPLAB X IDE (depending on the current layout of the MPLAB panes), you should see something similar to Figure 8. It displays the following items:

[1]The MPLAB Harmony new project wizard will automatically set the new project as the main project in MPLAB X IDE. This is required because the MHC will not launch if there is no currently selected main project open within the MPLAB X IDE.

1. The empty project (with no files).
2. The MPLAB Harmony Configuration tree.
3. The MPLAB Harmony Help system.
4. The MPLAB Harmony Configurator (MHC) output window.

The project you have just created (see arrow 1, in Figure 8) is empty, meaning it does not yet contain any source files. In later steps, you will use the MHC to generate an initial set of source files for the initial configuration of your project. You will use the MPLAB Harmony configuration tree (see arrow 2) to make the selections for your initial configuration. As you select items in the configuration tree, you will see help for that item appear in the help window (see arrow 3). As you use the MHC, it will display activity, warning, and possibly error messages in the output window (see arrow 4), depending on the level of output for which it is set. By default, it only displays key activity and error messages.

MHC: Configure the Processor Clock

As mentioned earlier in the introduction, the System Clock (SYSCLK) provides the time base for the CPU, peripheral clock, DMA, interrupts and flash. SYSCLK is determined from many input clocks (System PLL output (POSC or FRC with PLL), POSC, FRC, FRCDIV16, FRCDIV, LPRC, or SOSC). The default configuration for SYSCLK is programmable and can also be changed at run-time. The Primary Oscillator (POSC) can be configured for an external clock input (EC mode), or an external crystal or resonator (XT or HS modes). Standard production boards will have an 8Mhz Discera silicon resonator loaded and the EC oscillator option should be used.

System PLL Input Divider

The input divider must be chosen such that the resulting frequency applied to the PLL multiplier is between 4 and 5 MHz. The input divider options are as follows: $\div 1, \div 2, \div 3, \div 4, \div 5, \div 6, \div 10, \div 12$.

System PLL Multiplier

The multiplier options are as follows: $\times 15, \times 16, \times 17, \times 18, \times 19, \times 20, \times 21, \times 24$.

System PLL Output Divider

The System PLL output clock divider options are as follows: $\div 1, \div 2, \div 4, \div 8, \div 16, \div 32, \div 64, \div 256$.

After creating a new blank MPLAB Harmony project, the next step is to use the interactive clock diagram in the MHC to configure the processor clock. The MHC Clock Configurator is a component of the MPLAB Harmony Configurator (MHC) MPLAB X IDE plug-in. Its function is to provide a graphical user interface to configure the Oscillator module. To do this, use the following process. The numbered red arrows in the following clock diagram in Figure 9 show the location of each step in the process.

1. Select the **clock diagram** in the MPLAB Harmony Configurator pane.
2. Select the **External Clock (EC)** mode for the primary oscillator.
3. Enter the primary oscillator input clock frequency (8 MHz for the Chipkit Mx7ck board).
4. Select the primary oscillator (POSC) as the system PLL input clock.
5. Select the PRIPLL from the oscillator multiplexer (FNOSC).
6. Click **Auto-Calculate** to determine the divider and trim values in the each of the four reference clocks based on a user requested clock output frequency. .
7. Enter the desired system frequency (80 MHz for our kit).

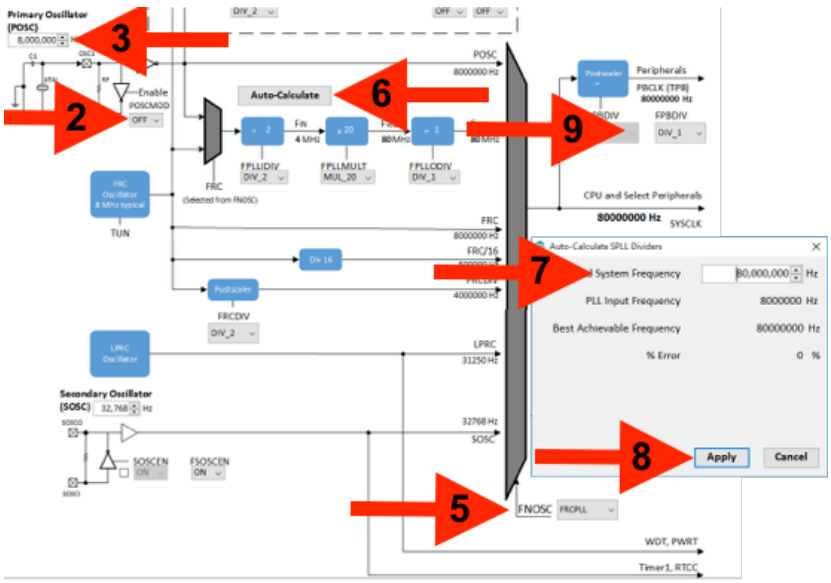

Figure 9: Configuration of processor clock (Numbers refer to text description).

8. Click **Apply**.

It is also possible to set the Peripheral Bus Clock (PBCLK) by setting the FPBDIV value. Choose DIV_2 to generate 40MHz PBCLK. For further details on this topic, please refer to the online microchip manual at `http://ww1.microchip.com/downloads/en/DeviceDoc/MPLAB%20Harmony%20Configurator%20(MHC)%20and%20MPLAB%20Harmony%20Graphics%20Composer.pdf`.

MHC: Configure Key Device Settings

There are a few key device configuration settings for most processors that must be correct to effectively run and debug the firmware. Exactly which settings are important depends on which processor is in use and what you are currently doing.

To get started, it is usually necessary to disable the Watchdog timer to avoid having the timer expire and reset the processor before the firmware has been implemented to support that functionality. To do so for PIC32MX795F512L, perform the following steps:

1. Select the Options pane in the MPLAB Harmony Configurator Window.
2. Expand the Device Configuration 1 (DEVCFG1) register settings tree.
3. Change the Watchdog Timer Enable (FWDTEN) flag to OFF.

MHC: Configure the I/O Pins

After configuring the processor clock and key settings, configure any general purpose I/O pins required by your project using the interactive Pin Diagram and Pin Settings panes provided by the MHC.

For example, the Chipkit mx7ck Kit has LED1 connected to port pin RG12. Therefore, that pin will need to be set as an output by performing the following actions:

1. Select the Pin Settings tab in the MHC pane.
2. Scroll to pin 96, RG12 and select it as a digital output pin by clicking the Out Direction (TRIS) button. (You can accept the default Low initial output level.)

3. It has a Digital mode.

Do not worry about configuring all of your I/O pins when you first create a new project. You can return to the Pin Settings and Pin Diagram tabs in the MHC at any time to update your I/O port settings. Also, predefined Board Support Packages (BSPs) will automatically configure the I/O pins for LEDs and switches for supported Microchip development boards. (Refer to Board Support Packages Help for more information on BSPs.).

MHC: Generate the Project's Starter Files

Once the initial configuration options have been selected in the MHC, click the **Generate Code** icon, as shown in the Figure 10 to generate the initial set of source files for your application.

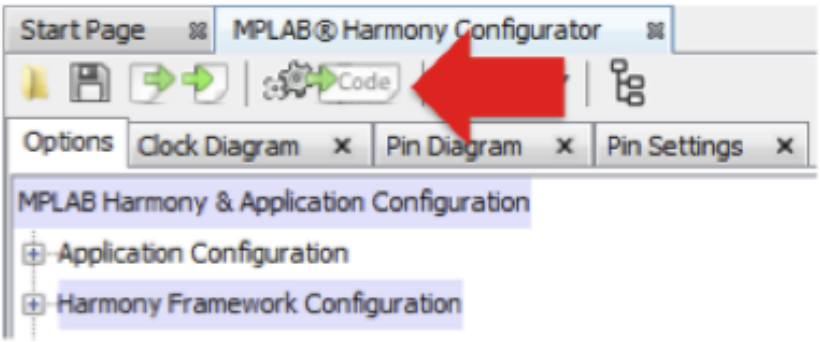

Figure 10: Generate project starter files.

Accept the default location for the configuration setting's (.mhc) file and click Save in the Modified Configuration dialog, unless you have previously saved the current configuration settings. This file stores the selections made in the MHC window. Accept the default settings and click Generate in the File Generation dialog.

This will cause the MHC to generate an initial set of project source files, based on your current configuration selections. This set of files will include the following application files, as shown in the MPLAB X IDE Project window.

Generated Header Files

The **app.h** header file contains definitions and prototypes required by the application or by other modules that use the application. The **system_config.h** header file defines build options and is included by all other files that require any configuration options. The **system_definitions.h** header file provides definitions required by the system configuration files. These files implement a configuration of an MPLAB Harmony firmware system.

Generated Source Files

The **main.c** source file contains a standard MPLAB Harmony C-language main function. The **app.c** source file contains the application logic state machine. The system files (**system_*.c**) implement the configuration of the system. The **system_init.c** file contains the standard MPLAB Harmony SYS_Initialize function (and supporting code) that is called by main to initialize all modules in the system. The **system_tasks.c** file implements the standard MPLAB Harmony SYS_Tasks function that is called in a loop by main to keep the state machines of all non-interrupt-driven modules running in the system. The **system_interrupt.c** file implements the Interrupt Service Routine (ISR) functions for any interrupt-driven modules in the system. Finally, the **system_exceptions.c** file implements any exception-handling functions required by the system.

Once the initial set of application and configuration files have been generated, you're ready to begin developing your application logic. To do this, you will modify some of the generated files, adding your own

custom code. Some of these files, such as app.h and app.c, are only generated once, when the project is initially created. These are starter files where you are intended to do most of your development. Other files (like the system configuration files) are under control of the MHC. You can, and will occasionally need to modify these files; however, you need not worry about losing your changes if you regenerate the project files. The MHC will open a diff tool when it detects that you have modified a file and allow you to choose which changes you want to keep and which ones you want to update.