Getting Started

Microchip Libraries for Applications (MLA)

Table of Contents

1 Getting Started 3

1.1 What is Microchip Libraries for Applications(MLA)? 4

1.2 Legal Information 5

1.3 Directory Organization 6

1.4 Project Layout 9

1.5 System Configurations 12

1.5.1 system_config.h 12

1.5.2 system.c 12

1.5.3 system.h 12

1.5.3.1 SYS_CLK_FrequencyInstructionGet Function 13

1.5.3.2 SYS_CLK_FrequencyPeripheralGet Function 13

1.5.3.3 SYS_CLK_FrequencySystemGet Function 14

1.6 Customer Support 15

1.7 Using a Diff Tool 16

1.7.1 Beyond Compare 16

1.7.2 MPLAB X (NetBeans) 19

Index 22

Getting Started

1 Getting Started

1.1 What is Microchip Libraries for Applications(MLA)?

Microchip Libraries for Applications (MLA) is a collection of Microchip firmware libraries and demo projects. Not all firmware libraries and demo projects from Microchip are distributed in this package; rather, this package includes a few specific libraries that tend to be used together.

By distributing libraries that are used often together, Microchip can provide example projects that integrate the use of multiple libraries.

For the specific release notes for each release, look at the release notes accompanying the release. It will also contain documentation for installation\ un-installation.

1.2 Legal Information

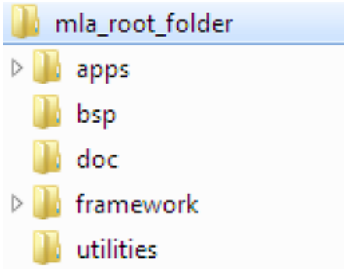
This software distribution is controlled by the Legal Information at www.microchip.com/mla_license

1.3 Directory Organization

This section summarizes the concept of the directory organization of the installation. The actual installation may differ from the screen shots in this section.

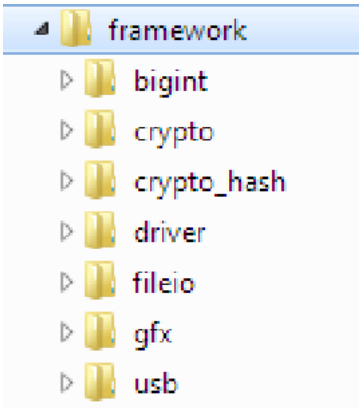
Description

The installation directory(mla_root_folder) has the following structure,



1. apps : Contains the application examples demonstrating an application which can use one or more middleware libraries or drivers.
2. bsp: Contains the board support packages for the hardware dependent code which is common and shared between various application examples.
3. doc: Contains common documentation, license files, release notes and middleware specific help files.
4. framework: Contains drivers and various middleware libraries
5. utilities: Contains common utilities that are used by multiple application examples and\or middleware libraries.

The framework folder which is the sub-folder of the installation root has structure similar to one below:

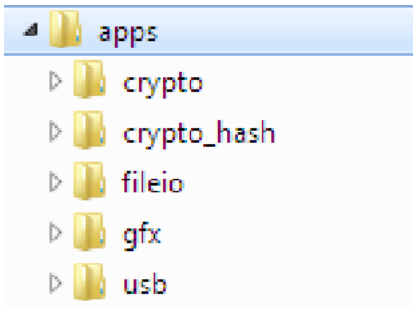


The driver folder implements common drivers. The rest of the sub-folders of the framework folders is specific to each middleware.

Each individual driver folder or each middleware folder atleast has the following set of files

- public header file(s): The public header file(s) which needs to be included in the project to allow the application to use the driver or middleware.
- src folder: The source folder contains the private header files and the source files required to implement the middleware or driver.

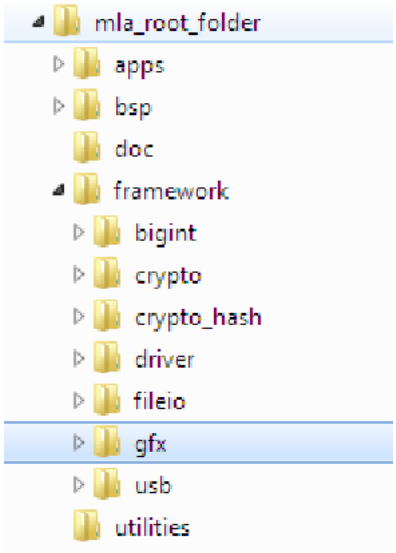
The apps folder in the installation directory has structure similar to:



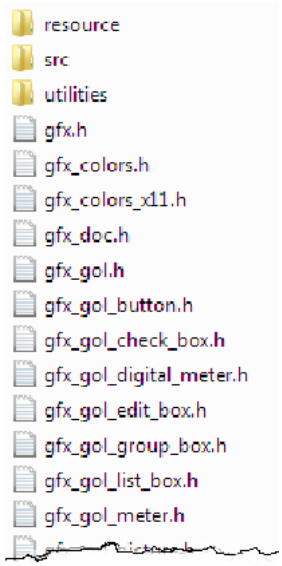
Example: Graphics Middleware and Graphics Primitive Layer Application Example

The section below demonstrates the above explained directory structure with the help of Graphics library and Primitive Layer Application Example.

The graphics library is at the location as explained by the below picture,

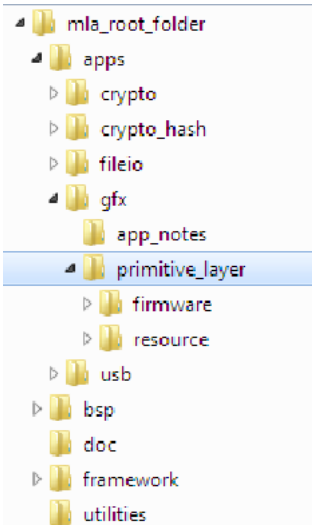


The Graphics library contains the public headers in the gfx folder. The source file and the private header files are in the src folder. The Graphics library also has resource and utilities folder as it is required for the library. The other libraries can have either, both or none of these special folders.



You need to include the public headers of the graphics library to allow you to use graphics library in the application. What to include in your project for the functioning of graphics library please refer the Graphics library help file. (The graphics library help is in mla_root_folder/doc)

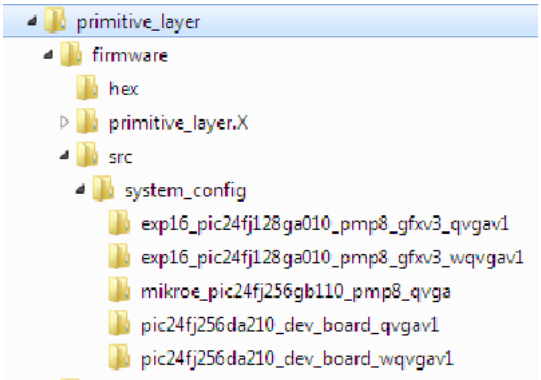
The Primitive Layer Application example for the graphics library is located in the apps folder, as demonstrated with the image below:



Each application example has

- firmware folder: Contains application example, MPLAB X project, and hardware dependent and hardware independent firmware source and header files for that application.
- resource folder: Contains resources associated with the application to work but is not firmware. This is an optional folder, will not be present if the application does not need it. For example, source images for graphics demos.
- utilities folder: Contains utilities associated with the application to work. This is also an optional folder, will not be present if the application does not need it. For example, a USB demo has a PC program to communicate with. PC Program will be here.

The firmware folder has the following structure



- hex folder: Contains the prebuilt hex files for each of the configuration for the primitive layer application example.
- src folder: Contains the hardware independent code which demonstrate the application
- system_config folder: Contains a set of folders which then contain hardware dependent code. Each individual folder in system_config folder is refered to as "hardware specific system_config folder".

For a list of abbreviations used, refer name_reference.html in mla_root_folder/doc

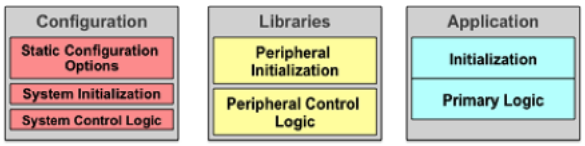
The section System Configurations, goes into details of what files are contained in each of the hardware specific system_config folder.

1.4 Project Layout

This section explains how a MLA demo projects are organized.

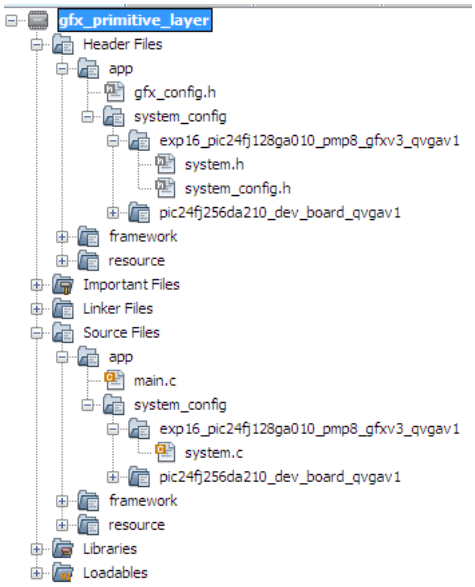
Description

To facilitate configurability, MLA projects are structured in a way that isolates the code necessary to configure a "system" from the library modules themselves and from application code. The first figure illustrates this concept, while the second figure shows how the files might appear in an MPLAB X IDE project.



The application files (main.c in the example) are separate from the configuration files in the “system_config” folder, so it is possible for a single project to have more than one configuration. (Usage of this capability can be seen in example projects included) The framework modules use the definitions provided in the selected system_config.h for its configuration.

A MLA demo Project is organized as shown below within MPLAB X.



This organization consists of a "logical" folders and source files, as described below.

The "app" folder, which contains:

- **Demo specific files**

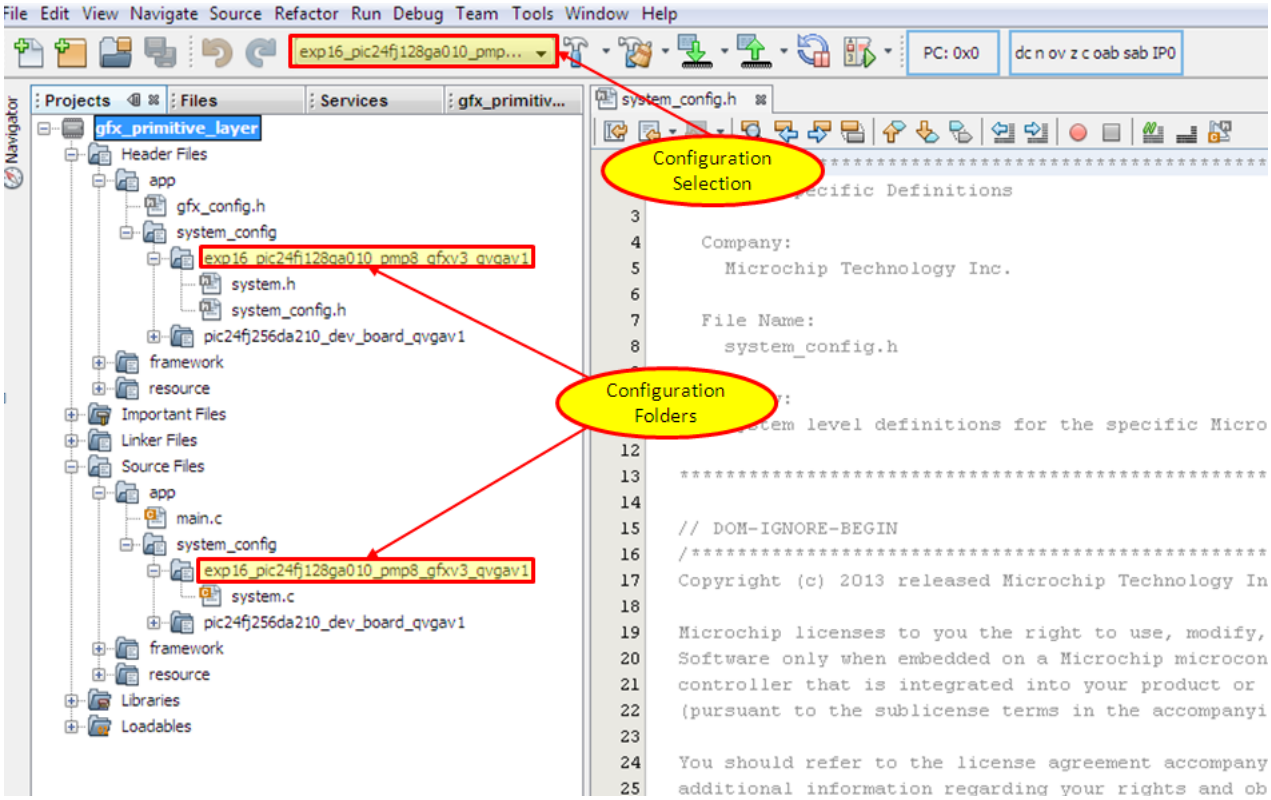
The "app" (short for "application") folder contains all of the application's firmware source files for the project. In a simple project, the "app" folder contains the demo specific file for the logic of the application itself. This is where the desired overall behavior of the application is usually implemented (although complex applications may have additional files/folders).

The app folder also consists of the system_config folder.

The "system_config" folder

The "`system_config`" folder contains one or more subdirectories, each of which corresponds to a build configuration. MLA projects can have multiple build configurations. Each build configuration consists of a specific set of properties (tools settings) in MPLAB X IDE and a set of source files that which modules are initialized and maintained in your system.

MPLAB X project layout



In most example projects distributed with MLA, the name of each MPLAB X IDE build configuration will match the name of the associated folder under the `system_config` folder in the project (the `exp16_pic24fj128ga010_pmp8_gfxv3_qvgav1` folder in the sample project). When a specific MPLAB X IDE configuration is selected, the configuration files for that configuration are included in the build and the configuration files in other configuration folders are excluded from the build.

Configuration Files:

- **`system_config.h`**
- **`system.h`**
- **`system.c`**

Above files normally make up a complete configuration of the system. The purpose of each of these files is described in more detail in the following sections. But, the basic idea is that you may want different configurations of your application for different physical hardware boards, different Microchip microcontrollers, or different feature sets, depending on your specific needs.

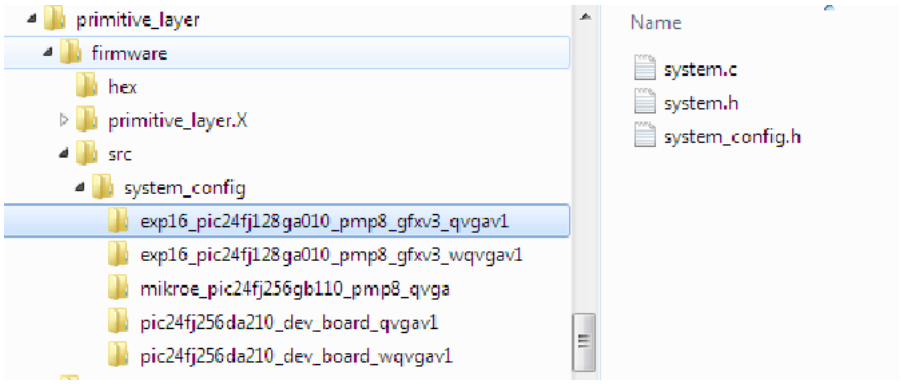
The "framework" folder

The "framework" logical folder contains all of the project relevant framework module source files. Depending on your project, there can be many files and sub folders under this folder. These files are for MLA libraries that you should not need to edit.

In most cases, the "logical folder" organization within the MPLAB X project matches exactly with the physical directory organization. This is done to keep things simple and consistent so you only need to learn a single layout. But, there are a couple of notable exceptions.

1. MPLAB has a convention of splitting out "Header Files" (".h" files) and "Source Files" (".c" files), so the virtual folder organization in the MPLAB X project separates the files and the physical directories on disk do not.
2. In an MLA example project, the "app" folder will correspond to the "src" directory on disk under the application's main folder.

The project demonstrated in the MPLAB X Project Layout above is represented as follows on the hard disk,



Project Include Path Settings

1. **Path to the framework folder:** In order for the projects to build, the include path up to the framework folder should be provided for each build configuration and must be placed in the "Includes directories" list in the compiler properties for the project. The framework module header files expect this. It is already done for the examples provided with MLA, but for the customer specific project, this needs to be done.
2. **Path to system_config.h:** For each build configuration, the path to system_config.h needs to be provided.
3. **Path to the application project src folder:** The hardware independent code resides in the src folder for the application. The path to the src folder needs to be provided for each build configuration.

For a project to be portable from one machine to another, it is recommended that the paths in the MPLAB X are relative, while adding files to MPLAB X and while specifying include paths.

1.5 System Configurations

This section describes the files that make up a system configuration.

Description

In MLA, a system configuration consists of a set of files that define the build options, how the system is initialized, and how it runs after it has been initialized. The purpose of each of these files is described in the topics in this section.

1.5.1 system_config.h

This topic describes the purpose of system configuration header file.

Description

System Configuration

In MLA, most library modules have a set of build time configuration options that define a variety of parameters (such as buffer sizes, maximum or minimum limits, and default behavior). These configuration options normally have acceptable default values. However, if you want to configure a library for your specific needs, its configuration options can be defined using C language preprocessor #define statements. The set of configuration options supported is described for each library in the "Configuring the Library" section of its help document and most libraries provide a template and example configuration header files.

To obtain its build configuration options, every library includes the same common top-level configuration file that must be called system_config.h and that must be defined as part of your system configuration.

If all the framework module configuration goes in one file, the file can be long. Hence, configuration tends to get split into sub-include files, such as gfx_config.h and usb_config.h. These can exist and can have different names. However, system_config.h must exist and the name of this file cannot be changed.

1.5.2 system.c

This section describes the system.c file.

Description

The configuration specific implementations that would exist in c source files in the configuration folder can really be organized, file wise, in any way the application developer wishes, since framework module files are not dependent on these files. In an MLA, the specific function is called from the `main` function in order to initialize all modules in the system. This function is implemented as part of a system configuration, normally in a file named `system.c`. The initialization function needs to have the same name across all the build configurations. This file may also include other necessary global system items that must be implemented in order to initialize a system such as processor configuration bits, clock settings, debug settings and system-wide global data structures.

1.5.3 system.h

This section describes the system.h file.

Functions

	Name	Description
	SYS_CLK_FrequencyInstructionGet	Gets the system clock frequency in Hz.
	SYS_CLK_FrequencyPeripheralGet	Gets the peripheral clock frequency in Hz.
	SYS_CLK_FrequencySystemGet	Gets the system clock frequency in Hz.

Description

If the library wants to access a system level prototypes, it will need to include system.h. The file system.h can include system specific implementation definitions. system.h is optional and may not exist if the libraries do not need to include these.

Clock definitions will be present in system.h to allow build configurations to define the specific clock values. The framework modules and\or application examples can use clock definitions as listed in this section.

1.5.3.1 SYS_CLK_FrequencyInstructionGet Function

Gets the system clock frequency in Hz.

File

system.h

Syntax

```
unsigned long SYS_CLK_FrequencyInstructionGet();
```

Returns

Clock frequency in Hz.

Description

This function gets the system clock frequency in Hz.

system.h can define it to a literal value #define SYS_CLK_FrequencyInstructionGet() 8000000 or system.h can define it to a function #define SYS_CLK_FrequencyInstructionGet() MyInstructionClock()

Implement the function MyInstructionClock in system.c and provide a prototype in in system.h.

Preconditions

None

Example

```
unsigned long clockOutputHz;

clockOutputHz = SYS_CLK_FrequencyInstructionGet();
```

Function

unsigned long SYS_CLK_FrequencyInstructionGet( )

1.5.3.2 SYS_CLK_FrequencyPeripheralGet Function

Gets the peripheral clock frequency in Hz.

File

system.h

Syntax

```
unsigned long SYS_CLK_FrequencyPeripheralGet();
```

Returns

Clock frequency in Hz.

Description

This function gets the peripheral clock frequency in Hz.

system.h can define it to a literal value #define SYS_CLK_FrequencyPeripheralGet() 8000000 or system.h can define it to a function #define SYS_CLK_FrequencyPeripheralGet() MyPeripheralClock()

Implement the function MyPeripheralClock in system.c and provide a prototype in in system.h.

Preconditions

None

Example

```
unsigned long perClockOutputHz;

perClockOutputHz = SYS_CLK_FrequencyPeripheralGet();
```

Function

unsigned long SYS_CLK_FrequencyPeripheralGet ( )

1.5.3.3 SYS_CLK_FrequencySystemGet Function

Gets the system clock frequency in Hz.

File

system.h

Syntax

```
unsigned long SYS_CLK_FrequencySystemGet();
```

Returns

Clock frequency in Hz.

Description

This function gets the system clock frequency in Hz.

system.h can define it to a literal value #define SYS_CLK_FrequencySystemGet() 8000000 or system.h can define it to a function #define SYS_CLK_FrequencySystemGet() MySystemClock()

Implement the function MySystemClock in system.c and provide a prototype in in system.h.

Preconditions

None

Example

```
unsigned long clockOutputHz;

clockOutputHz = SYS_CLK_FrequencySystemGet();
```

Function

unsigned long SYS_CLK_FrequencySystemGet ( )

1.6 Customer Support

The Microchip Web Site

Microchip provides online support via our web site at http://www.microchip.com. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

Product Support - Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software

General Technical Support - Frequently Asked Questions (FAQs), technical support requests (http://support.microchip.com), online discussion groups/forums (http://forum.microchip.com), Microchip consultant program member listing

Business of Microchip - Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Development Systems Customer Change Notification Service

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at www.microchip.com, click on Customer Change Notification and follow the registration instructions.

Additional Support

- Users of Microchip products can receive assistance through several channels:
- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or field application engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is available on our website.

Technical support is available through the web site at: http://support.microchip.com

Training

- Regional Training Centers: http://www.microchip.com/rtc
- MASTERs Conference: http://www.microchip.com/masters
- Webseminars: http://techtrain.microchip.com/webseminars/QuickList.aspx

1.7 Using a Diff Tool

This section will cover the basics of using a diff tool to compare two different sets of code to evaluate the differences. This section will cover a couple of different tools available and their basic functionality. This section doesn't cover all of the features available in each tool nor does this section cover all of the possible diff tools available.

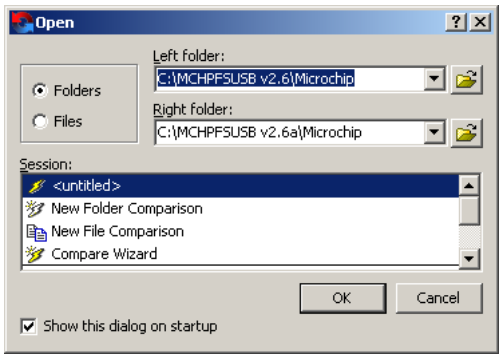
1.7.1 Beyond Compare

Beyond Compare is a commercial available differencing software from Scooter Software, Inc (http://www.scootersoftware.com/).

This demonstration is based on Beyond Compare v2.2.7. There may be interface or features changes in other versions of this software. Please refer to the software's documentation for a more detailed and updated description of the functionality.

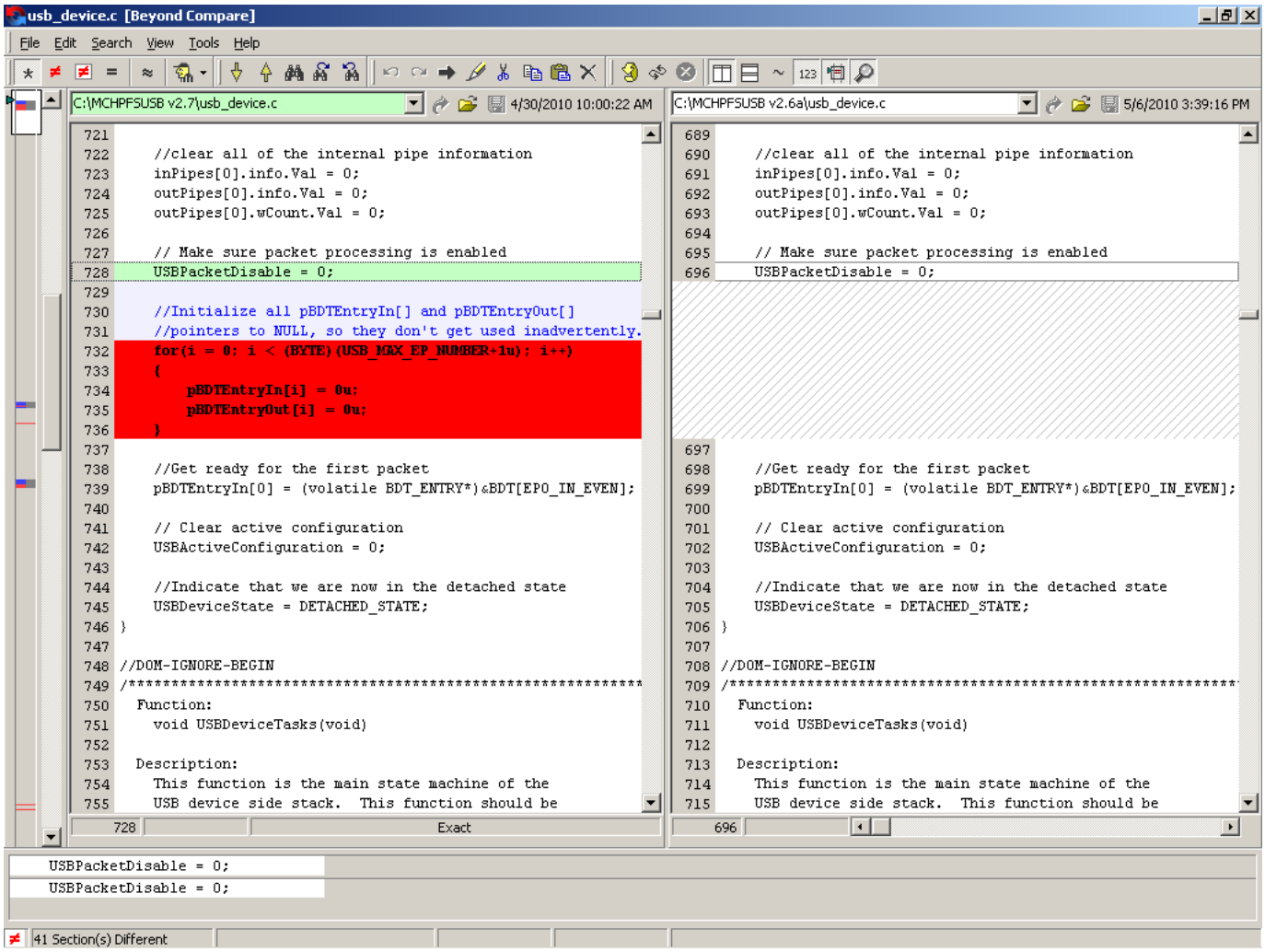
Creating a comparison

There are a couple of ways to create a comparison with Beyond Compare. The first is to open the Beyond Compare program from the Start menu. This opens a window that allows you to either compare two files or two folders.



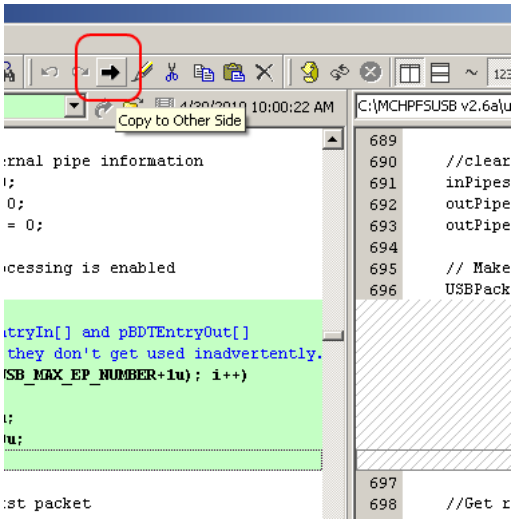
Beyond compare also has a right click menu option that allows the user to right click on two files and compare them to each other.

When two files are compared together, the differences between the two files are highlighted. On the left of the main windows there is also a difference navigation bar that shows where the differences are in the file. You can click on this bar to go to that location in the code.



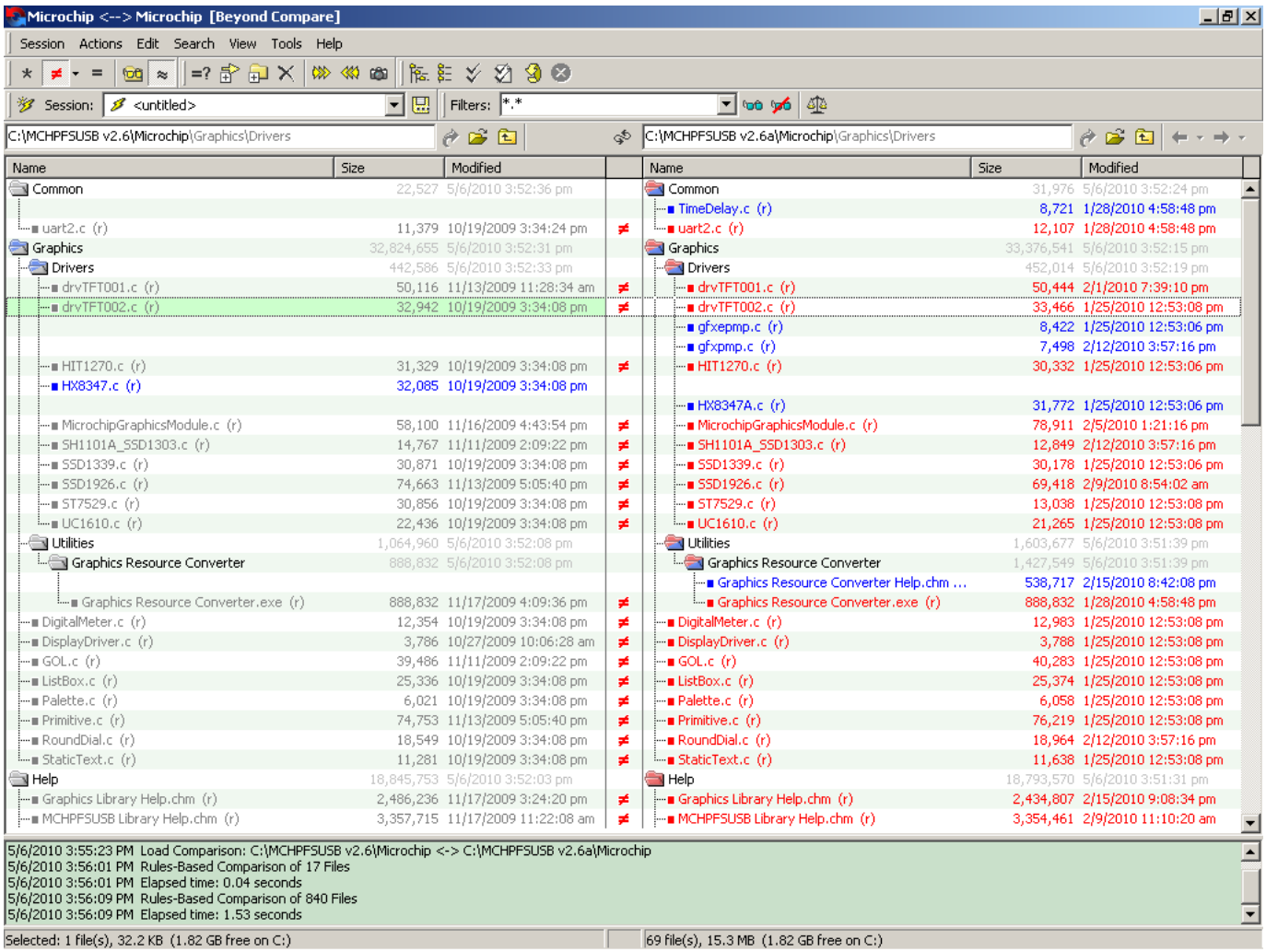
Moving changes

Once a difference between two files is detected, it is easy to move that change from one file to the other. In Beyond Compare simply highlight the lines that need to move, and click on the "Copy to other side" button that is shown below.



Comparing Folders

A feature of Beyond Compare is that it allows you to compare two folders against each other. Once two folders are selected in the program (or through the right-click menu option discussed before), the folders are compared against each other. At this point of time the contents of the folders aren't compared to see if they are different, just if the files are present or not.



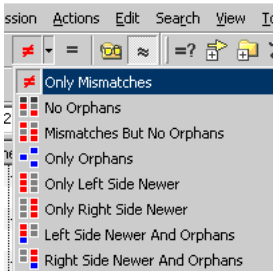
Double clicking on a file will open a file difference instance showing the difference between the two files.

To compare all of the contents against each other you can select all of the files by expanding all of the folders,

selecting all of the files,

and running the file comparison tool.

To see only files that are different, select the "Only mismatch" from the comparison tool.



1.7.2 MPLAB X (NetBeans)

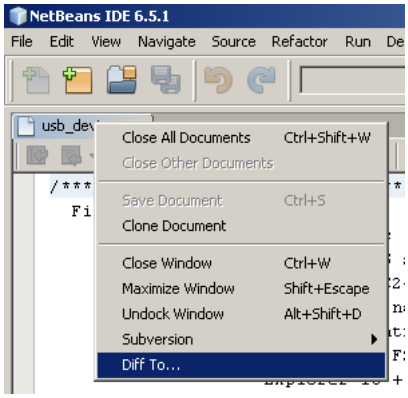
MPLAB X (NetBeans) is an open source IDE that has built in differencing functionality (www.microchip.com/mplabx and http://netbeans.org/).

This demonstration is based on NetBeans v6.5.1 but also applies to MPLAB X beta 7.02. There may be interface or features changes in other versions of this software. Please refer to the software's documentation for a more detailed and updated description of the functionality.

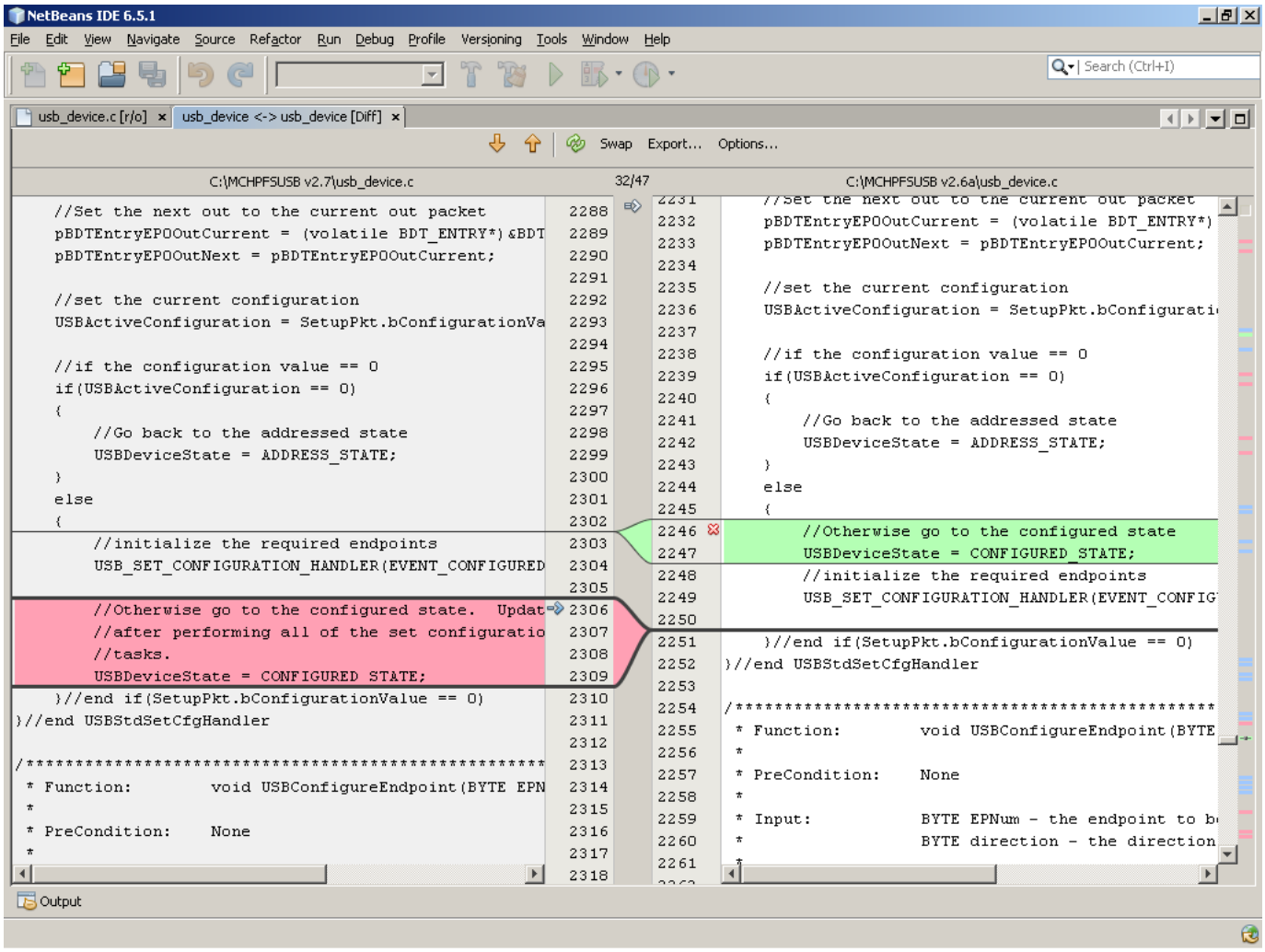
Creating a comparison

To compare two files in MPLAB X (NetBeans), one of the files must first be opened. This can be done using the "File->Open File" menu option.

Once one file is open, right click on the tab with the file name. In this menu there should be a "Diff to..." option that is available.

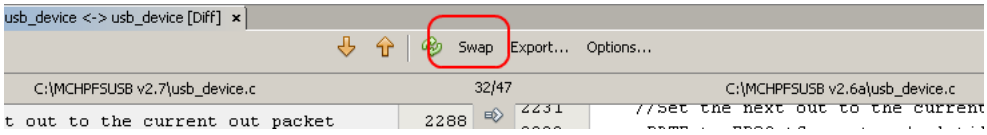


This option will open a new file window. In this file window select the file that you want to compare against. Once this file is selected the two files will sit side by side. On the right hand side of the window there is a difference navigation bar. By clicking on any of these highlighted sections, it will bring you to the difference in the two files.

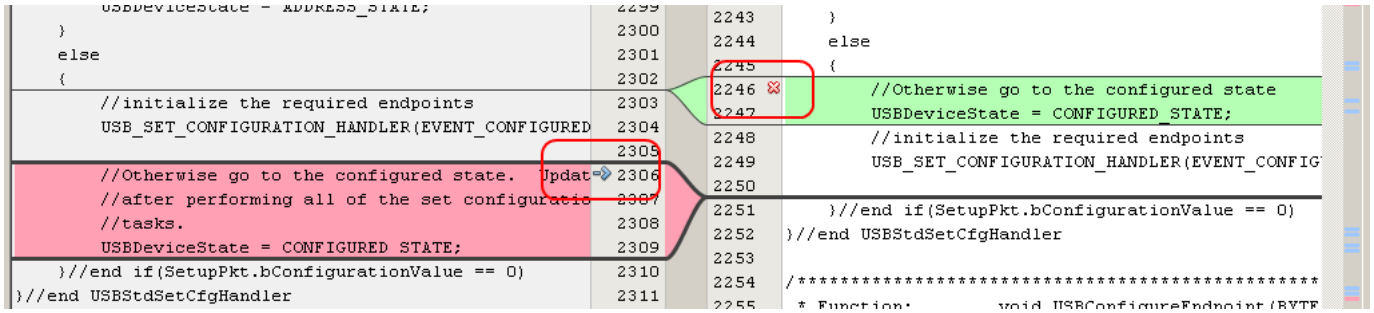


Moving changes

Changes in MPLAB X (NetBeans) are always made from the left file to the right file. If the files were opened in the opposite order as the changes that need to be made, you can click the "Swap" button that is just above both of the two files. This will exchange the position of the two files.



To move a change, click on the icon that is near the center bar between the differences. An arrow "->" indicator on the left window indicates that the changes in that section will be moved from the left file to the right file. An "X" found on the right file in a green section indicates that the source found in that section will be deleted when you click the "X".



Please note that changes made to the files via the MPLAB X (NetBeans) comparison tool may be final. You may not be able to undo these effects to please use caution when making changes.

Index

B

Beyond Compare 16

C

Customer Support 15

D

Directory Organization 6

G

Getting Started 3

L

Legal Information 5

M

MPLAB X (NetBeans) 19

P

Project Layout 9

S

SYS_CLK_FrequencyInstructionGet 13
SYS_CLK_FrequencyInstructionGet function 13
SYS_CLK_FrequencyPeripheralGet 13
SYS_CLK_FrequencyPeripheralGet function 13
SYS_CLK_FrequencySystemGet 14
SYS_CLK_FrequencySystemGet function 14
System Configurations 12
system.c 12
system.h 12
system_config.h 12

U

Using a Diff Tool 16

W

What is Microchip Libraries for Applications(MLA)? 4