



Please note that GitHub no longer supports Internet Explorer.

We recommend upgrading to the latest [Microsoft Edge](#), [Google Chrome](#), or [Firefox](#).

Ignore

Learn more

Join GitHub today

Dismiss

GitHub is home to over 31 million developers working together to host and review code, manage projects, and build software together.

Sign up

Microchip MPLAB Harmony Configurator v3 User Guide

[Jump to bottom](#)

Bud Caldwell edited this page Jan 17, 2019 · 16 revisions

Introduction

MPLAB® Harmony 3 is a vital part of the [MPLAB development tools](#) ecosystem for working with **Microchip® 32-bit SAM®** and **PIC®** micro controllers. It consists of a Graphical User Interface (GUI) called the MPLAB Harmony Configurator (MHC) and an extensive set of interoperable firmware libraries that accelerate the process of developing embedded applications. The fundamental MHC GUI tools are the downloader and configurator tools. The downloader tool simplifies the process of downloading the firmware libraries, demonstration applications, and extensions to the MHC GUI. The configurator tool provides a convenient GUI for selecting libraries, initialization parameters, and optional features. It also makes it easy to connect everything together and generate C language code in a working configuration for your project.

This document describes how to install the MHC and how to get started using MPLAB Harmony to develop embedded applications designed with Microchip® 32-bit micro controllers.

Installation

The MHC is available as a plugin extension to the [MPLAB® X IDE](#) and as a standalone Java application for use with other tool suites. Regardless which form you choose, there are some prerequisites that you must have installed first. Additionally, you will need to have a supported [32-bit MCU board](#) on which to program, run, and debug your application.

Prerequisites for MHC use as an MPLAB X IDE Plugin

- Install the MPLAB X IDE, available from www.microchip.com/mplab.
- Install the XC32 C/C++ compiler for support of all Microchip 32-bit MCUs, available from www.microchip.com/xc32.

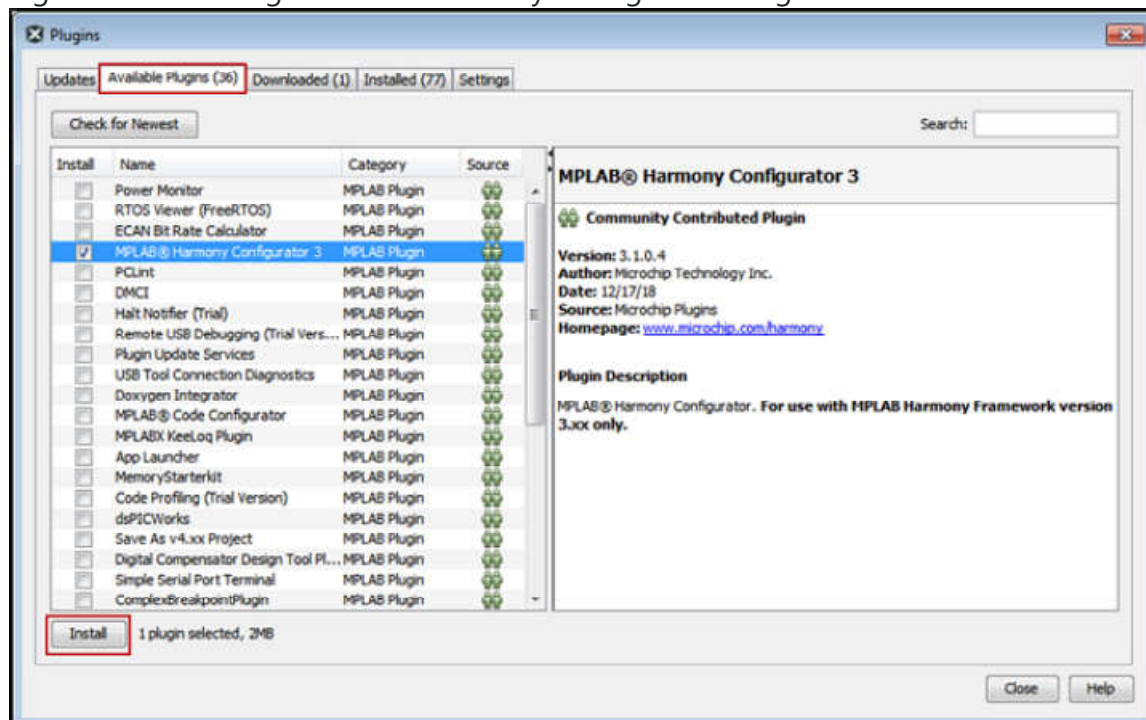
Installing MPLAB® Harmony Configurator from the Microchip Plugins Update Center

The MHC is not automatically installed with MPLAB® X IDE. You will need to take additional steps to download and install it through Microchip Plugins Update Center:

Steps

1. Open the MPLAB X IDE.
2. In the top-level menu, select to *Tools > Plugins*.
3. In the Plugins window, select the *Available Plugins* tab.
4. Select MPLAB® Harmony Configurator 3 from the list of available plugins and click the *Install* button. The Plugin Installer opens.
5. Click *Next* and review the License Agreement.
6. Click *Install* when you are ready for the Plugin Installer to begin downloading the MHC plugin. When the MHC plugin download is complete, MPLAB® X IDE will ask to be restarted.
7. Select *Restart Now* and click *Finish*. Upon restart, the plugin is installed. You can now open MHC on a new or already existing MPLAB® X IDE project.

Figure 2-1. Installing MPLAB® Harmony Configurator Plugin



Updating the MPLAB® Harmony Configurator


Whenever a new version of MHC plugin is available, the MPLAB X IDE will display a notification on the IDE window. Clicking on it will launch the plugin update wizard. In the wizard, click on the *Install* button to download and install the latest MHC plugin version.

Using the MHC with the MPLAB® X IDE

To generate code using the MHC in MPLAB® X IDE, you must first choose to create a new project or use an existing one. Then, you must follow these steps.

To Create a New Project

To create MPLAB® X IDE project, follow these steps:

1. Select *File > New Project* or click  to create a new project. The New Project wizard will open.
2. In the Choose Project pane, select the *Microchip Embedded* category.
3. In the Projects pane, select *32-bit MPLAB Harmony 3 Project*, then click *Next*.
4. Continue by following the Selecting MPLAB Harmony 3 Packages steps, below.

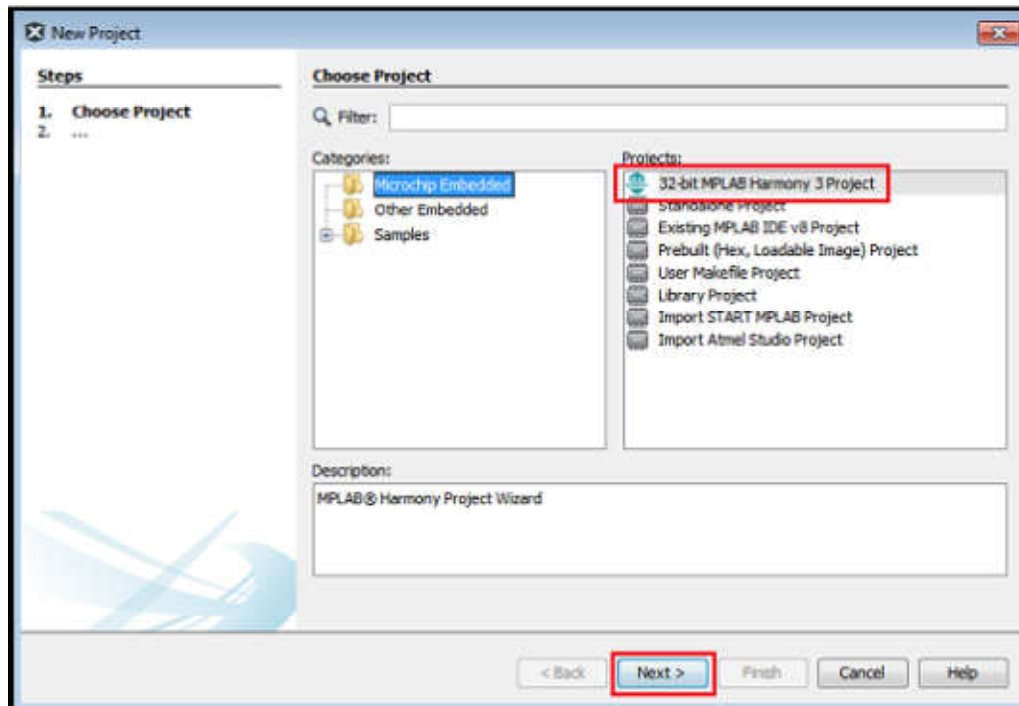



Figure 3-1. Create a new project

Note: If the 32-Bit MPLAB® Harmony 3 Project type is not listed under the Microchip Embedded category, please download and install MPLAB® Harmony 3 Configurator before continuing with these steps.

To Choose an Existing Project

To choose an existing project, follow these steps:

1. Click . The Open Project window will be displayed.
2. Navigate to the desired project's .X folder and click *Open Project*. The selected project will open in the IDE.

Note: If multiple projects are open in MPLAB® X IDE, set one as the main (active) project by Right-clicking on it and selecting “Set as Main Project”.

Selecting MPLAB Harmony 3 Packages

The Framework Downloader tool simplifies downloading of MPLAB Harmony 3 packages. Packages contain source code, templates, documentation, MHC extensions, and other collateral for MPLAB Harmony modules. Each package is maintained in a GIT repository that can be downloaded (or cloned) to your development system in a framework folder of your choice.

The Downloader tool can be launched using one of the following methods:

- **Method 1:** While creating a new project from the New Project window, click the *Launch Framework Downloader* button.

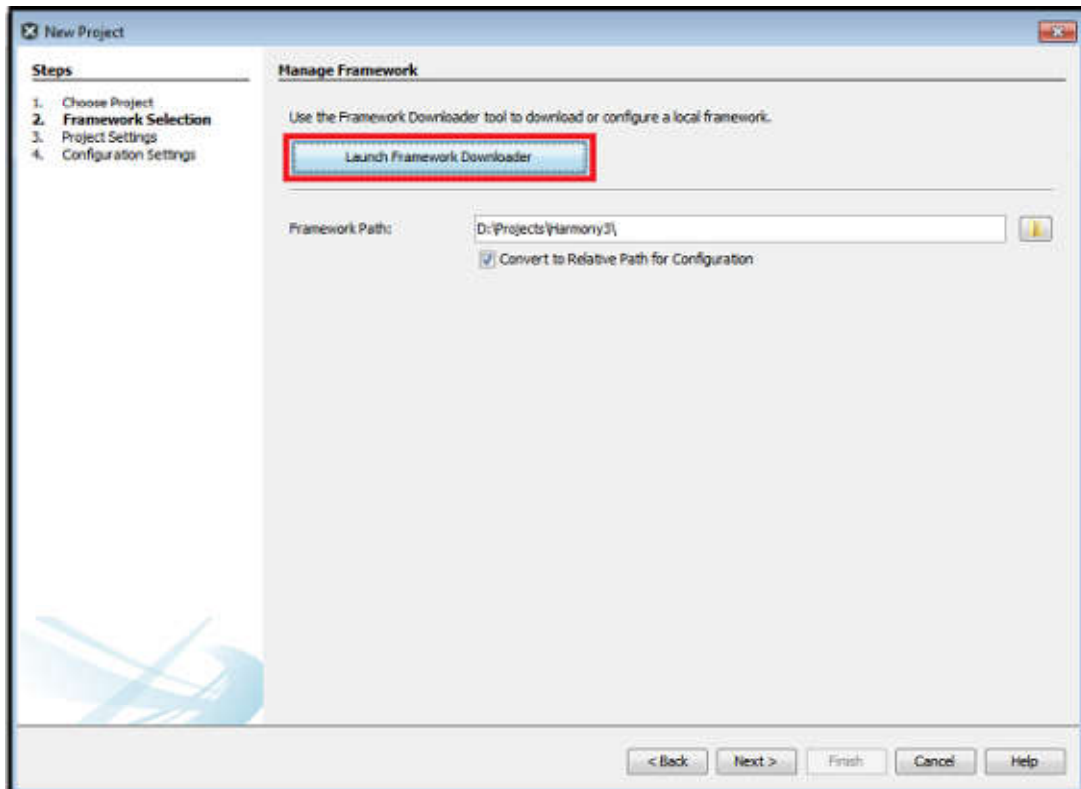


Figure 3-2. Launch Framework Downloader Button

- **Method 2:** From the *Tools > Embedded > MPLAB Harmony 3 Framework Downloader* menu option.

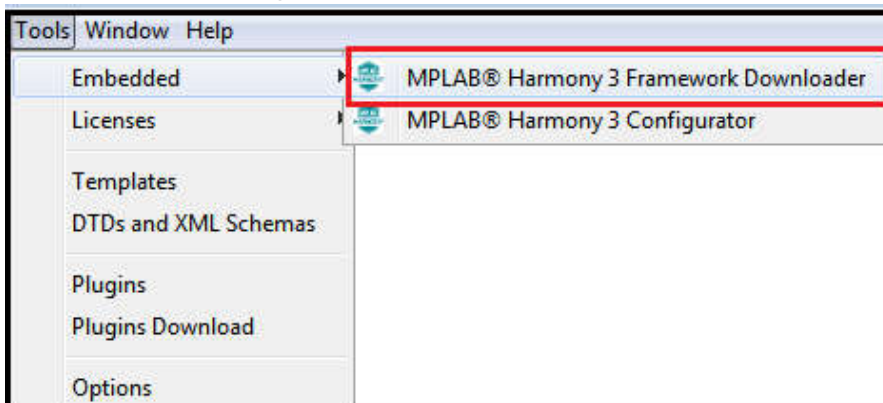


Figure 3-3. Launch Framework Downloader Menu

Downloading Harmony Packages:

After launching the MHC Downloader, you will need identify the local folder to which you wish to download the MPLAB Harmony 3 framework packages and you will need to direct the downloader to the repository server.

Server: <https://github.com/Microchip-MPLAB-Harmony/>

1. In the Framework Install Path dialog, select the desired framework installation

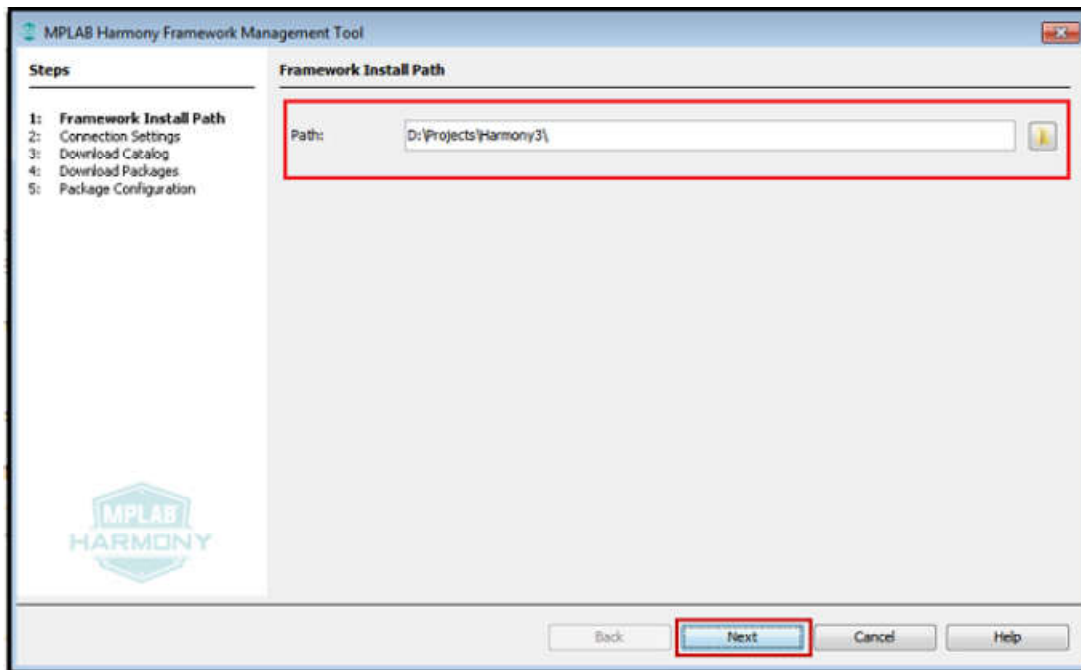


Figure 3-4. Framework Installation Path

2. Provide the repository URL and test the connection.
 - i. In the right-pane, enter information for Repository URL (previously given)
 - ii. Test connection before proceeding by clicking the *Test Connection* button. Ensure the connection is okay before proceeding.
 - iii. Click *Next*. to continue. The downloader will query the catalog (repo) repository.

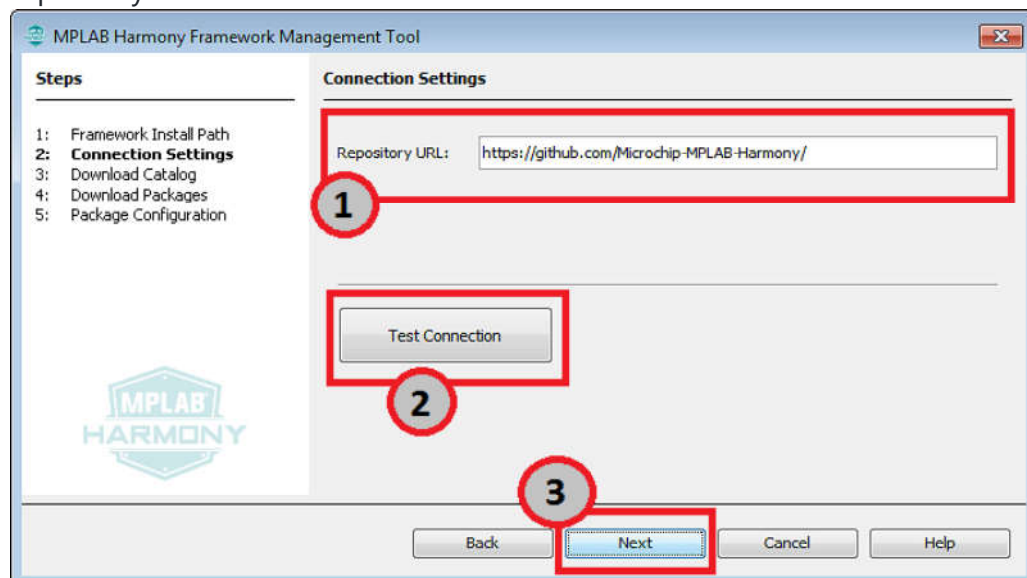


Figure 3-5. Connection Settings

3. Click *Next* when the query completes. The downloader will list all available package repositories.

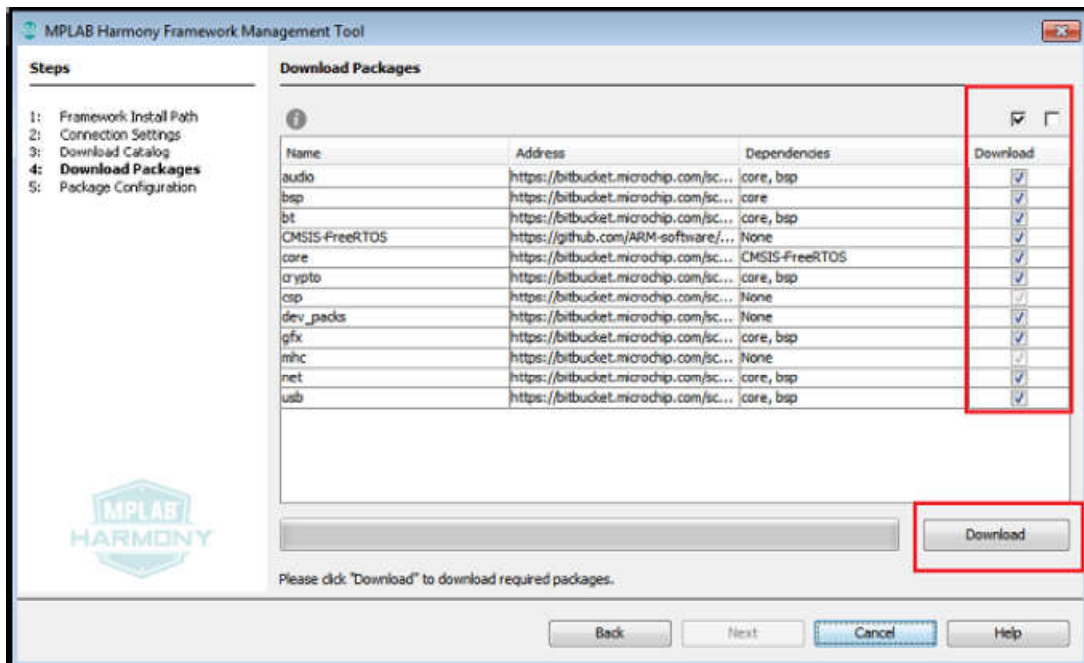


Figure 3-7. Download Packages

4. Choose only the packages in which you are interested by clicking the check-box next to its name. Required packages will be selected automatically.
5. Click "Download" to begin downloading selected packages. Progress will be shown as each package download is completed.
6. Click Finish to proceed to review the Package Configuration after all downloads have completed.

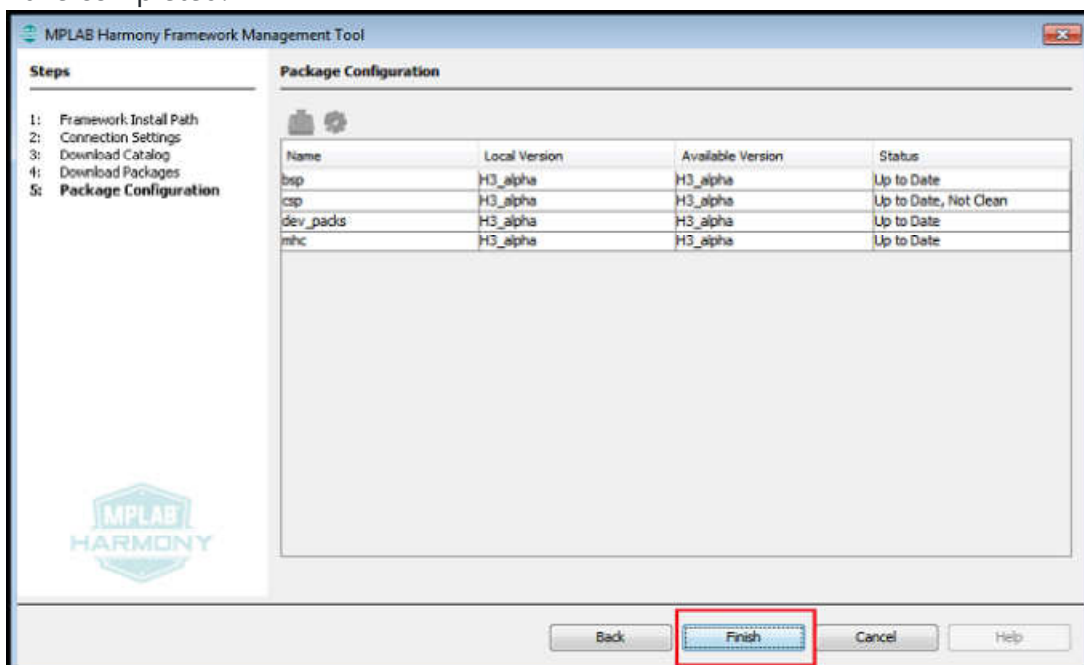


Figure 3-8. Download Catalog

7. Click *Finish* to continue to the project settings wizard.

Project Settings

In the New Project window, perform these settings:

1. Ensure that the Framework Path is correct. If it is not, navigate to the folder to which you previously downloaded the MPLAB Harmony 3 packages.
Note: You can leave *Convert to Relative Path for Configuration* selected.
2. Click *Next* to select the project Name and Location.
3. To choose the project location, navigate to the folder in which you wish to keep your MPLAB Harmony 3 projects and create a top-level folder for this project.
Note: Be sure to create a new top-level folder for this project as it will contain both the MPLAB X IDE's project folder (the ".X" folder) and the firmware folder into which the MHC will generate the selected source code.
4. In the Folder edit box, provide the name that will be used for the MPLAB X IDE's ".X" folder.
Note: This must be a valid directory name for your operating system.
5. In the Name edit box, enter the project's "Virtual" name. This is the name that will be shown from within the MPLAB X IDE.
Note: The Path box is not editable. It will update as you make changes to the other entries.
6. Click *Next* to proceed to configuration settings.

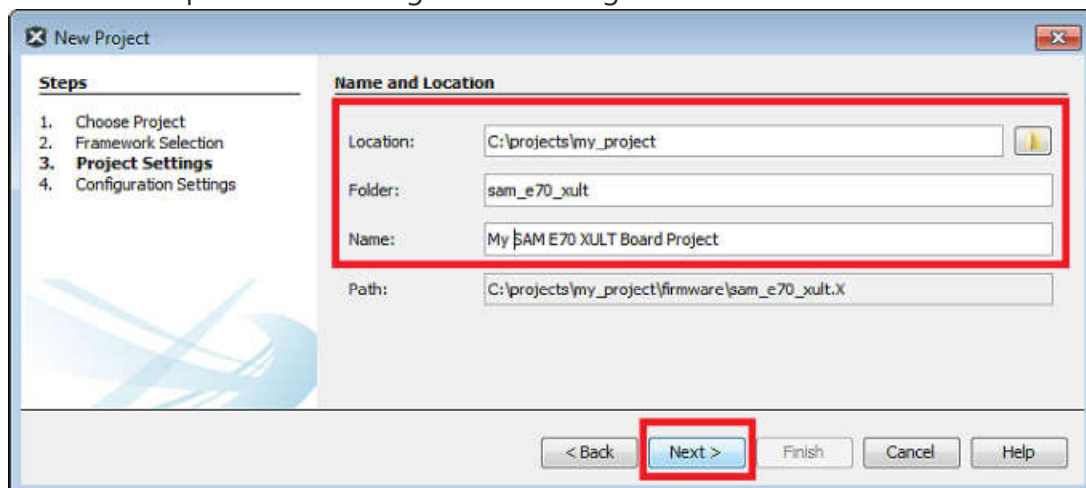


Figure 3-9. Project Name and Location

Configuration Settings

Follow these steps to select the project's configuration settings.

1. In the right pane, under Configuration Settings enter details for the configuration Name and choose the Target Device.
Note: You can select the Device Family or enter a partial device name to filter

2. After selecting the target device, Click *Finish* to launch the MHC.

Note: The New Project Wizard will first open a Configuration Database Setup dialog window to allow you to review the packages that will be used by the current project. Click *Launch* to continue to the MHC Configurator tool.

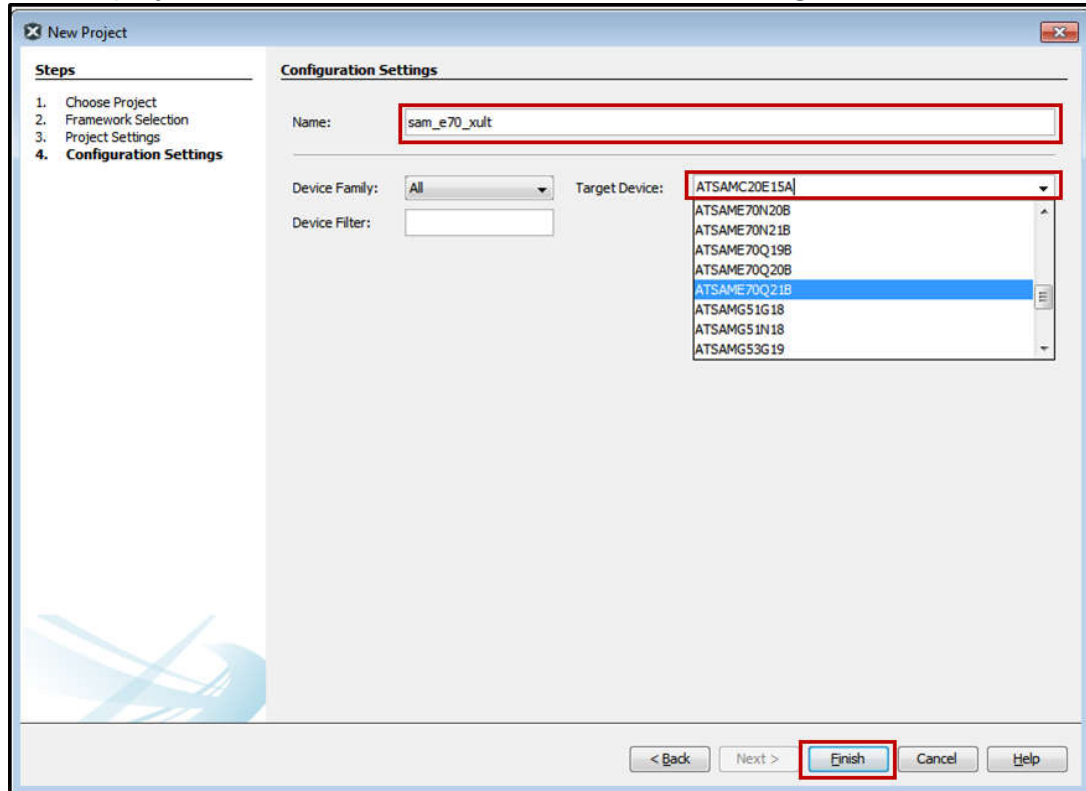


Figure 3-10. Configuration Settings

Launching MHC

Follow these steps to launch the MHC.

1. Open the MHC plugin tool. If the project has already been created, launch the MPLAB Harmony 3 Configurator by selecting *Tools* → *Embedded* → *MPLAB® Harmony 3 Configurator* from the MPLAB X IDE's menu bar.

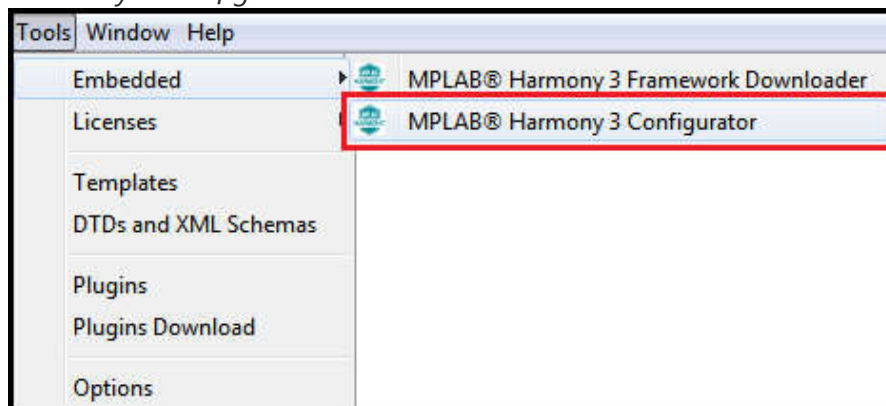


Figure 3-11. Launch MHC

The MPLAB Harmony Launcher window will be displayed.

2. If necessary, reconfigure the MPLAB X Harmony project and framework paths by clicking the *Reconfigure Paths* button. Otherwise, accept the default settings and then click *Launch*.

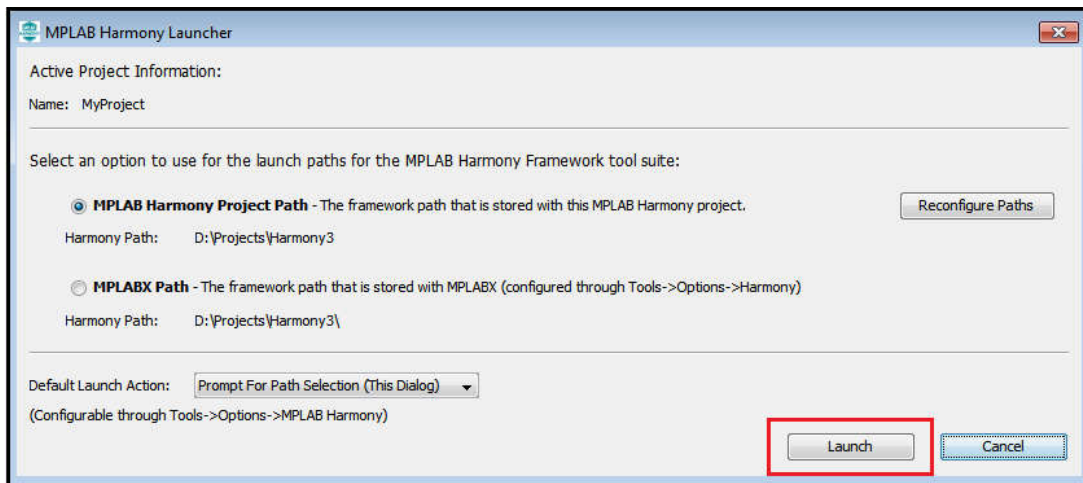


Figure 3-12. MHC Harmony Launcher

The Configuration Database Setup window will be displayed, which shows the selected and configured Harmony packages.

3. Click **Launch** to open MHC plug-in.

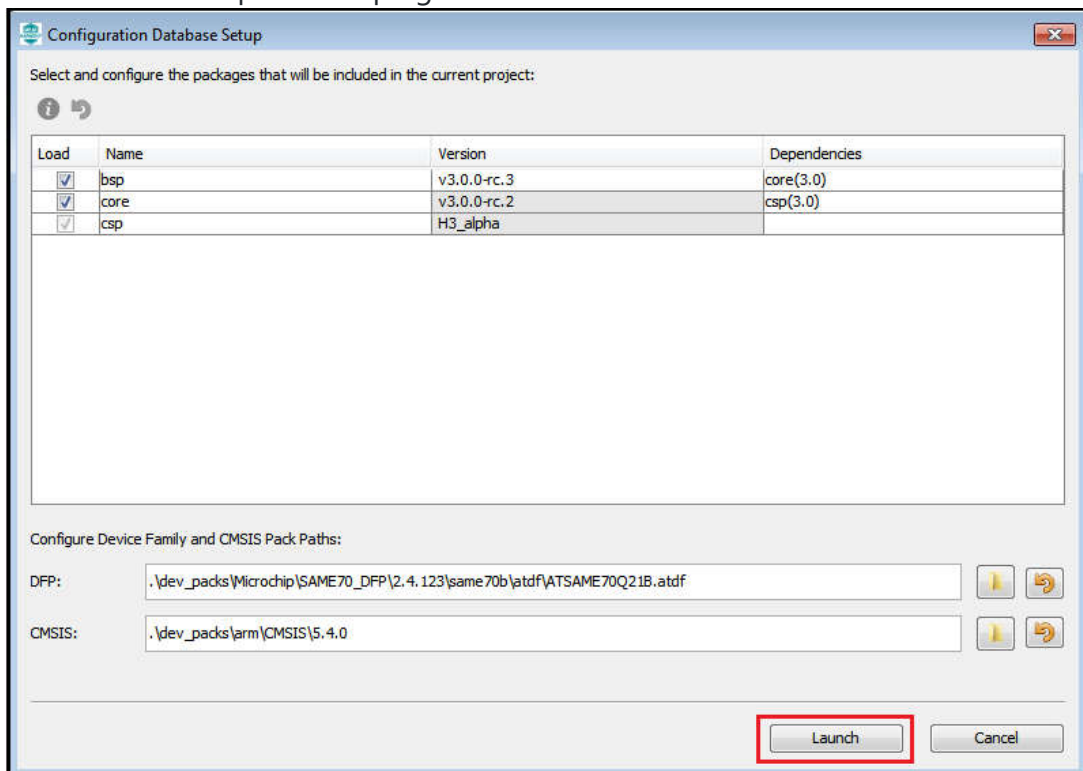


Figure 3-13. Configuration Database Setup

The MHC plugin's main window for the project will be displayed.

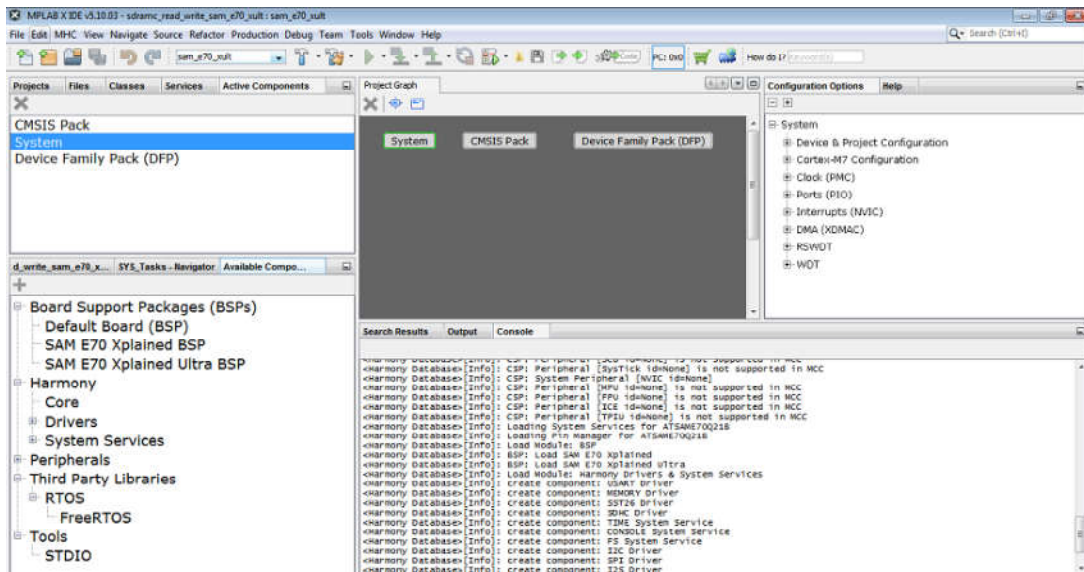


Figure 3-14. MHC Main Window (Sample Project)

Operating Areas

The MHC Graphical User Interface consists of following six major operating areas.

- **Active Components:** Displays activated/instantiated components.
- **Available Components:** Displays the available components based on the project configuration. Displays Board Support Packages (BSPs), list of available Peripheral Libraries (PLIBs), Harmony Core which consists of Drivers and System Services, Middleware Software, Third-Party Software and Tools.
- **Project Graph:** Shows the instantiated components. User can instantiate available components by double click on component. After successful component instantiation, you can see the instantiated components under Active Components panel.
- **MHC Plugins:** Consists of AFEC, DMA, MPU, NVIC, Clock and Pin configuration plugins. To open any of these plugins go to *MHC → Tools → select above mentioned available plugins*.
- **Configuration Options:** Displays the tree view of the selected component under Project Graph Area. User can do the component configuration from here.
- **Console:** Displays the MHC operation results

The following figure displays the MHC graphical user interface showing six operating areas.

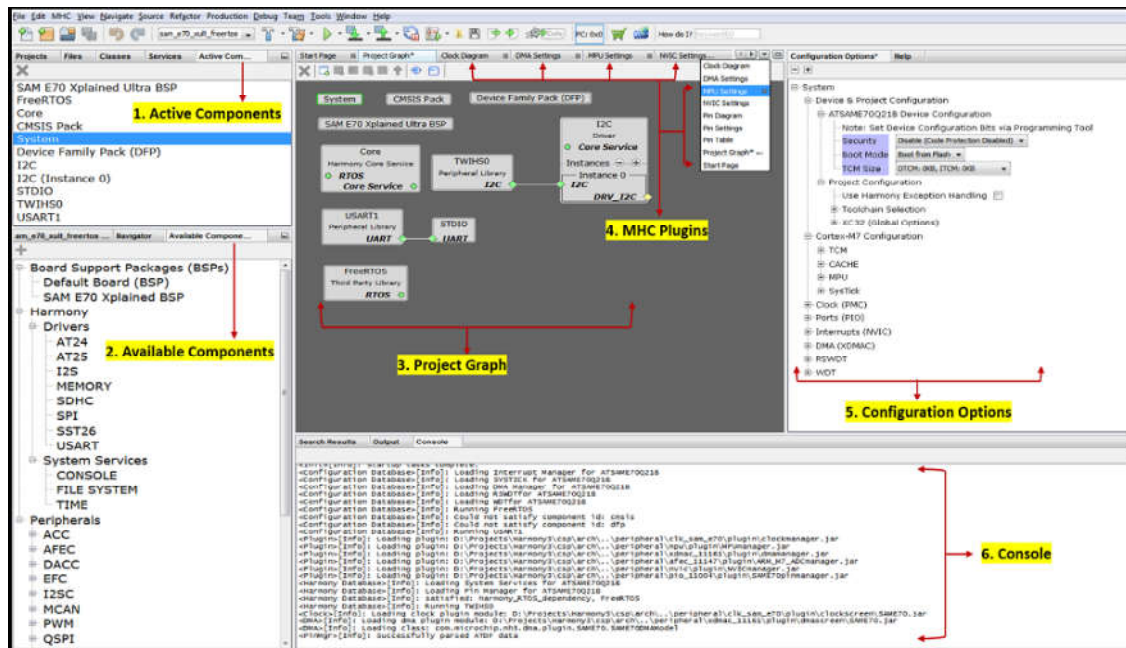


Figure 5-1. MHC Operating Areas

Note: The components shown under components are based on the Microcontroller selected while creating the project. All operating areas are dockable. Each can be dragged and dropped into another position, even out of the MPLAB® X IDE main window. However, closing the IDE however will close all MHC windows, including the ones moved outside the IDE.

Active Components, Project Graph and Configuration Options

Active Component and Project Graph displays all the instantiated components. By default, the below essential components from the Chip Specific Package (CSP) repo will get auto instantiated.

- **System:** The CSP System component is used for the device and project specific configurations.
- **CMSIS Pack:** The Cortex Microcontroller Software Interface Standard (CMSIS) Pack is a vendor-independent hardware abstraction layer for the Cortex® -M processor series and defines generic tool interfaces.
- **Device Family Pack (DFP):** This is the Microchip® Device Family Pack retrieves the device specific information and instantiates the same.

The following figure displays Active Components and Project Graph Area. Along with above mentioned default components, it consists of other active components like BSP, Harmony Core, I2C Driver, TWIHS, USART, SDTIO and FreeRTOS which are activated/instantiated by user.

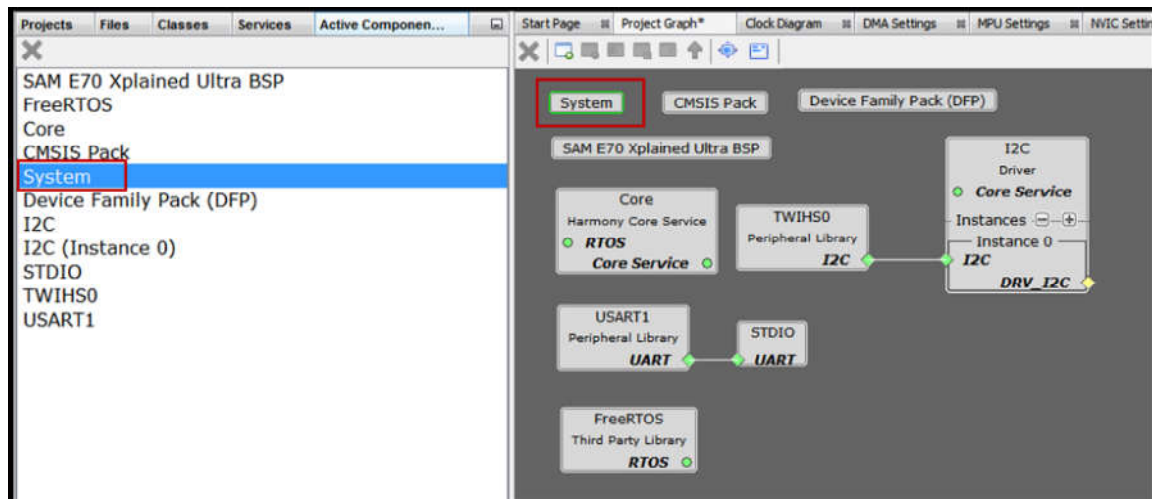


Figure 5-2. Active Components and Project Graph Areas

The following figure displays the project configuration of selected active "System" component. User can modify/configure activated component by selecting a component under Active component or under Project Graph. This way user can do the configuration of selected component.

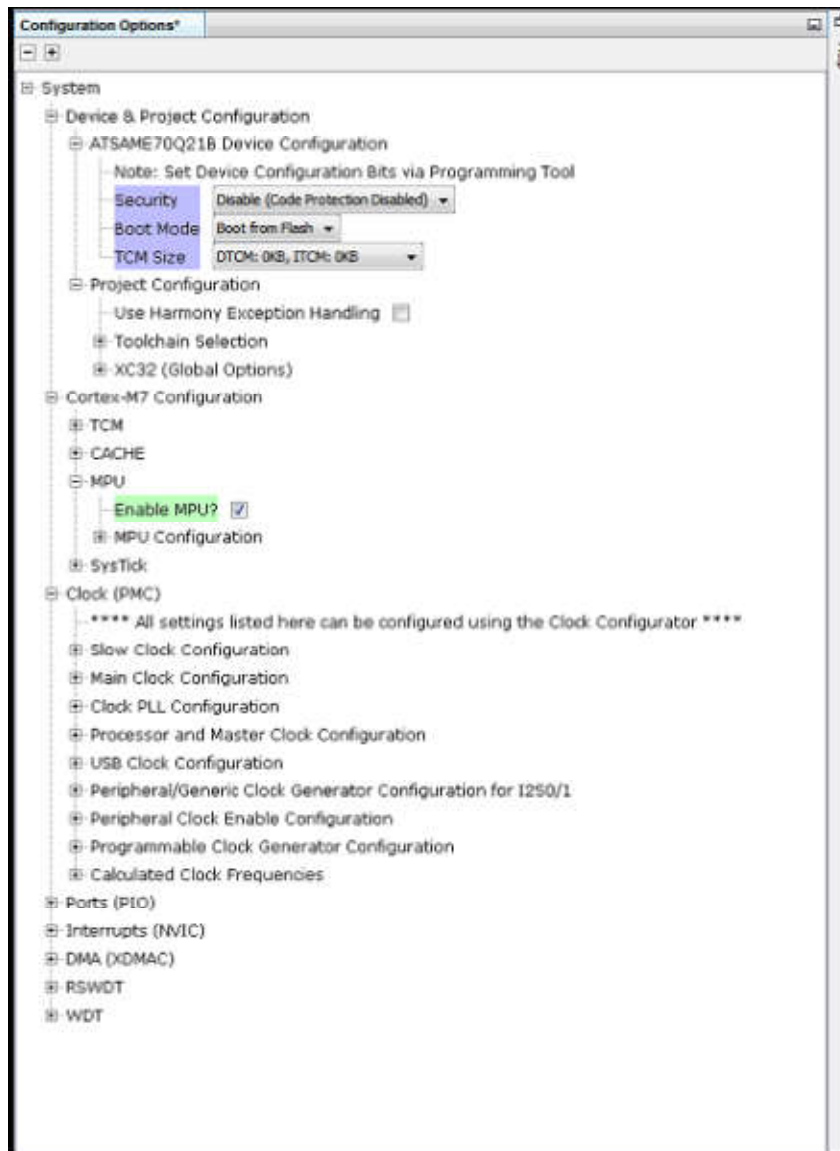



Figure 5-3. Configuration Options

Remove component

To remove a component available in Active component area, follow these steps:

1. Select it (for example, FreeRTOS) in Active Component Area or in Project Graph Area then click on button .
2. A pop-up window will display asking whether to deactivate a component. The following figure deactivating the FreeRTOS component.
3. Click Yes to deactivate the FreeRTOS component.

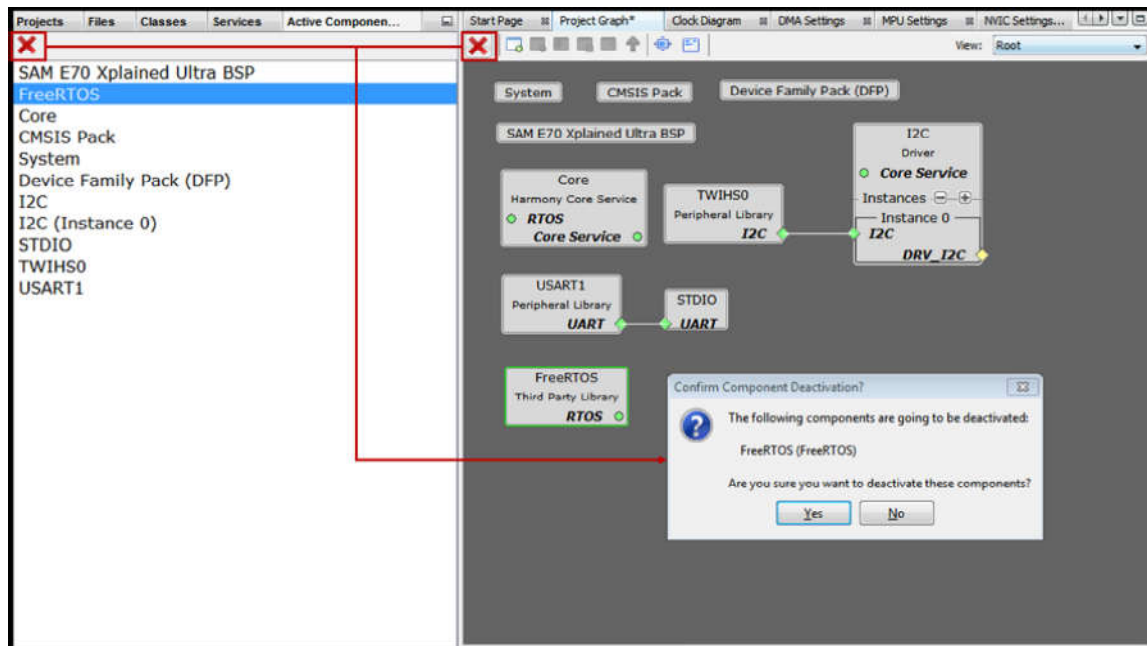


Figure 5-4. Deactivate an active System Component from Project

Note: As the "System" component is the main component in CSP, except "System" all other components can be deactivated.

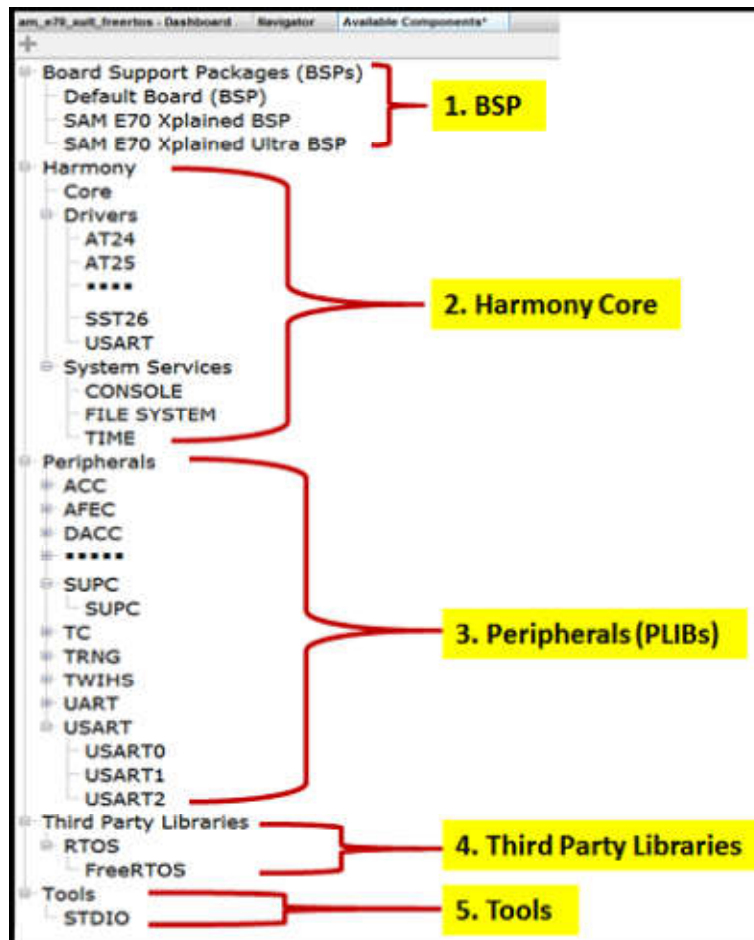
Available Components

This section displays the available components of the project. Available component list varies based on the Harmony Core repo synchronization and the Microcontroller selected while creating the project.

To instantiate a component, from the available components, double click on any available component. After the component is instantiated, the instantiated component will be displayed in the Project Graph Area. Refer "Section "Project Graph"":<https://github.com/Microchip-MPLAB-Harmony/Microchip-MPLAB-Harmony.github.io/wiki/Microchip-MPLAB-Harmony-Configurator-v3-User-Guide#project-graph> for additional information. The following sections cover available components and description, and for GUI see figure below.

1. Board Support Packages (BSPs): Includes default or board specific BSP.
2. Harmony Core: Includes Harmony Core Services i.e., Drivers and System Services.
3. Peripheral: Includes available peripherals like USART, TWIHS, TC and so on.
4. Third-Party Libraries: Lists the supported Third-Party software's. Currently supported Real Time Operating System is "FreeRTOS".
5. Tools: This lists the supported tools like Standard Input/Output (STDIO) which uses USART peripheral.

Similarly, if users sync USB or TCP/IP repositories then the respective Middleware Software section shall be displayed under the available components section.



Project Graph

The Project Graph displays all the instantiated component blocks. In Figure 4-6 BSP, USART and STDIO modules from Peripherals (CSP), I2C module from Harmony Core Driver, and FreeRTOS from Third-Party Software components are activated. MHC uses concept of Capability and Dependency for easy instantiation of modules.

Generic Capability and Dependency

A generic capability has a common name that identifies an interface that can be provided by a component but does not have an implementation of its own. One or more other components must implement the capability as shown in the following figure.

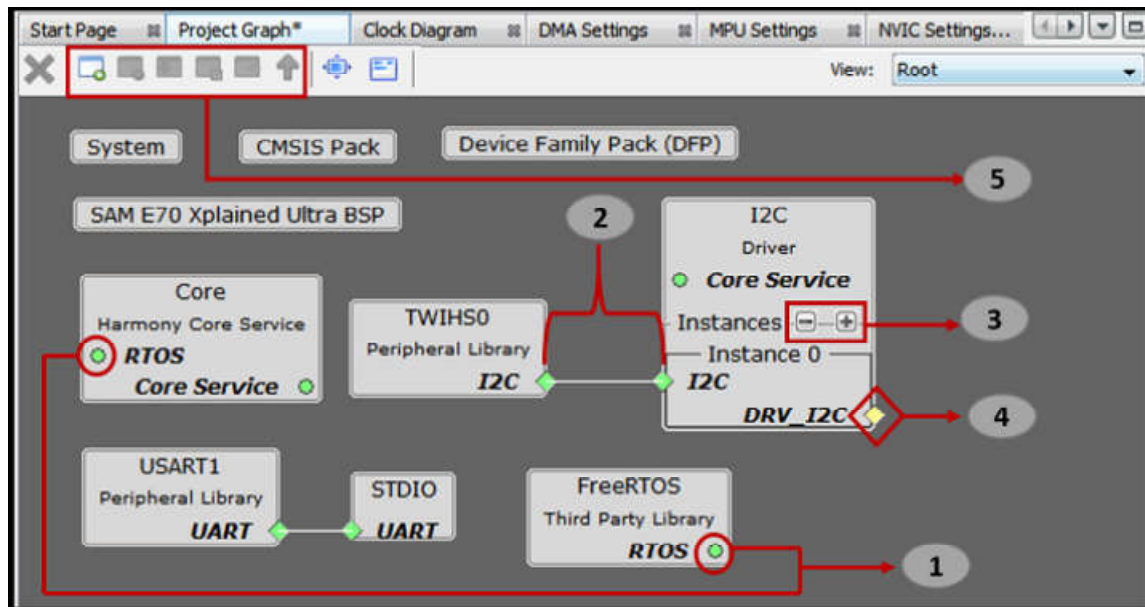



Figure 5-6. Project Graph Area

- **Point 1:** Indicates the concept of Generic capabilities and dependency, i.e. FreeRTOS has exposed a generic capability on "RTOS" on which Harmony Core Service has created a generic dependency, up on satisfying these generic dependencies the color will turn to Green. Non-required dependencies are indicated as Yellow and indicate optional functionality that will not inhibit the operation of the component if absent. Required dependencies will display as Red, indicating that an error will likely occur during generation, compilation, or at runtime if the dependency is not satisfied. Similarly, Harmony Core Service has exposed a generic capability "Core Service" on which I2C driver has created a generic dependency. Generic dependencies connect automatically to generic capabilities and no connector line is required to bind the two.
- **Point 2:** Indicates how I2C dependency is satisfied by binding with the capability TWIHS0. The left side block is a capability i.e. TWIHS0 and the right-side block is a dependency i.e. "Instance 0" I2C.
- **Point 3:** Indicates how a new component can be generated at run-time by clicking button creates a new generator database component. A generator database component can create unique instances of itself as needed. This is for things like components like bit-bang drivers that are defined in software, and thus can be added several times to a project if desired. Similarly, by clicking button deletes a component instantly.
- **Point 4:** Indicates the available capability of I2C driver. As mention above a non-required dependency is indicated as Yellow and indicates optional functionality that will not inhibit the operation of the component if absent.

- **Point 5:** Options to create, disband, add selected, view selected and configure group or container respectively. This is explained in detail in Section 5.3.2 Project Group Creation.

The following figure illustrates the unsatisfied direct dependency for I2C driver. User can see the list of satisfiers for a direct dependency by right-clicking on Red color diamond button.

Click button  to adjust the "Canvas Size" and button  for "Toggle Minimap".

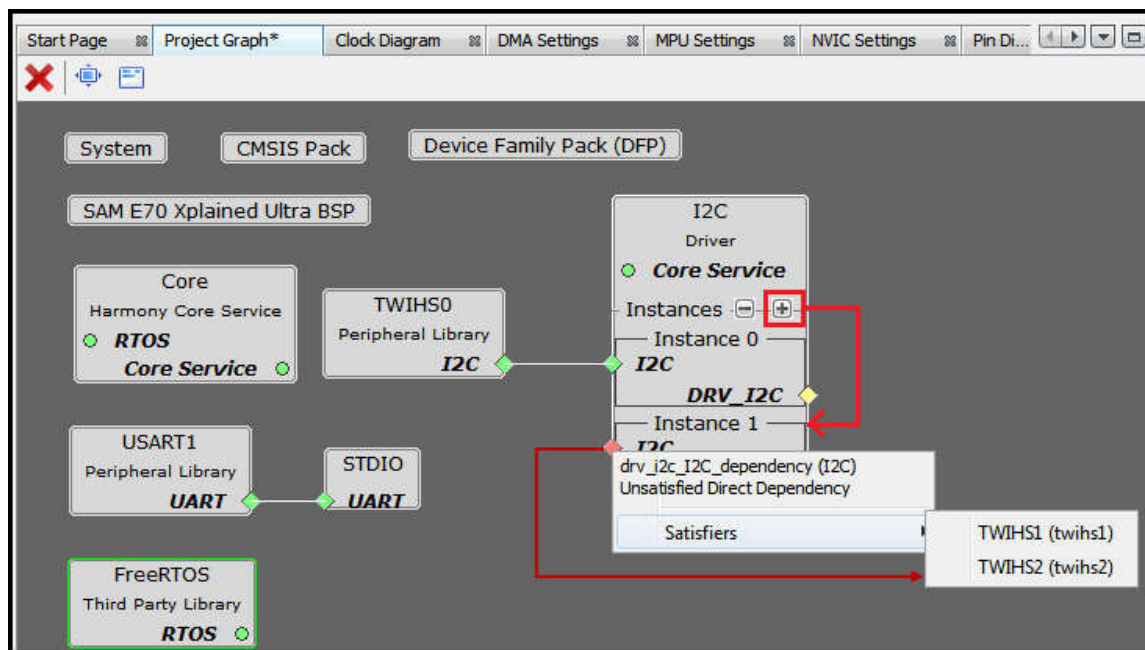


Figure 5-7. Project Graph Area: Component Generator and Satisfying a Direct Dependency

Multi-Dependency and Multi-Capability

A component can have more than one dependency and capability respectively. The following figure illustrates the multi dependency and multi-capability feature of Harmony and shown with square block unlike a diamond block in single dependency. For example, in the below figure File System component is dependent on **SD Card** (SPI), **SDHC** and **MEMORY** components respectively whereas each of these exposes a **DRV_MEDIA** capability.

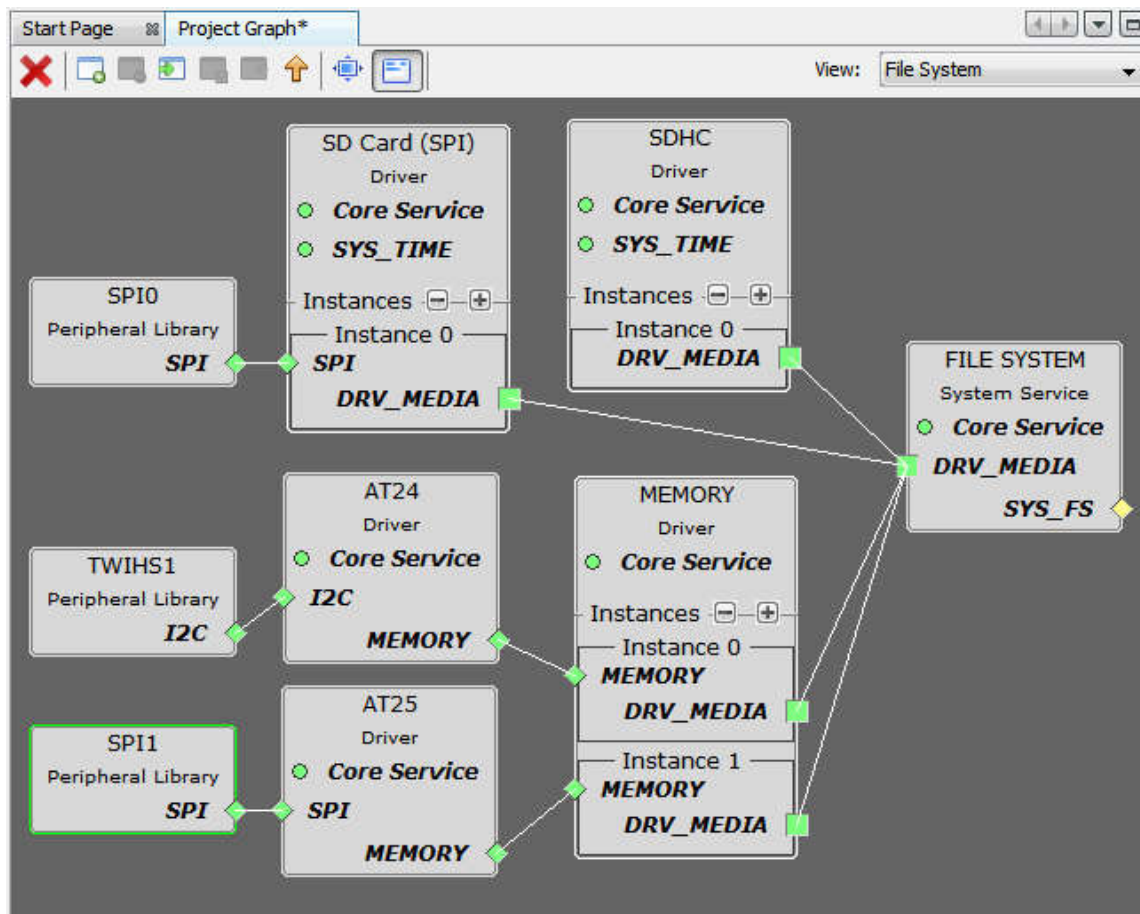
















Figure 5-8. Multi-Dependency and Multi-Capability feature

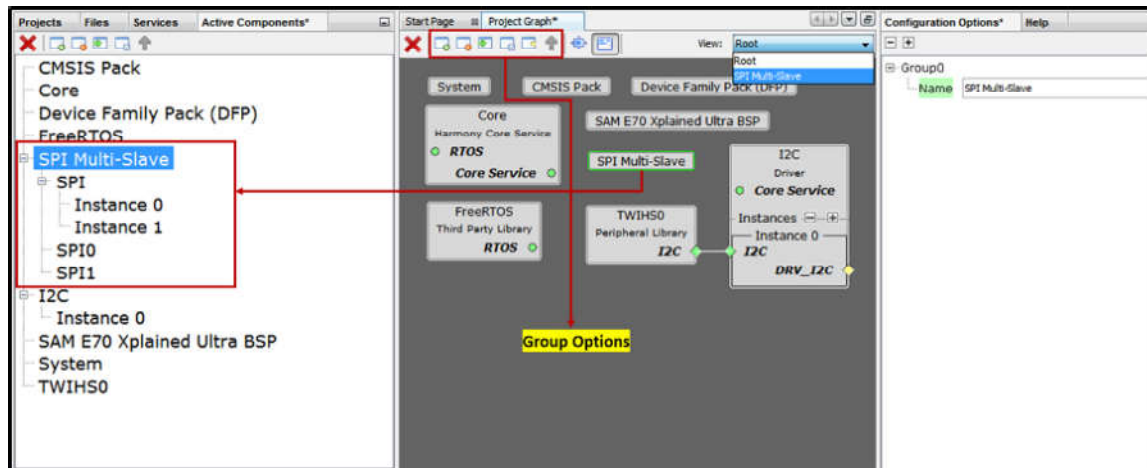
Container or Group Creation

The following are different group options. The Root is the default group. See the figure 5.9 below.

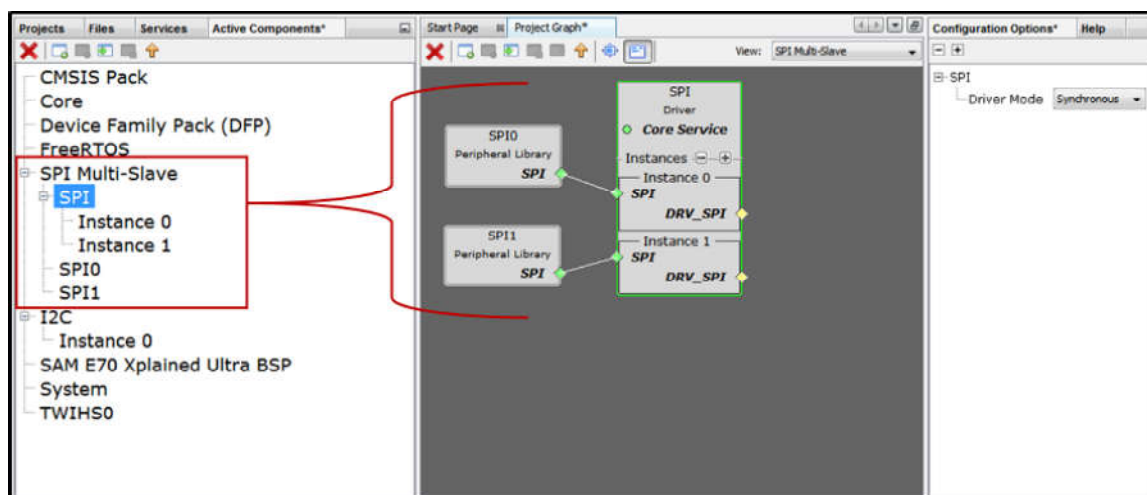
- **Create Group** or click : Creates a group. If user selects any component and clicks , then a pop-up is shown asking to automatically add selected components to the new group.
- **Disband Group** or click : Deletes a selected group. User can delete one group at a time. If user selects any component and clicks , then a pop-up is shown asking to delete selected group.
- **Add selected to Group/Container** or click : Adds selected component in to a target container. On clicking  a pop-up with drop down menu option is shown asking user to select the target container.
- **View selected Group/Container** or click : Select a group and click  or double click a group to view the components grouped together in a container.
- **Configure Group Node** or click : Select a group then click  to display the connections of the components if any. On clicking  a pop-up window is displayed showing the tabular list of available nodes, select the nodes to display

- **Select Next Highest Group** or click : Selects the next highest group available. Root is the default group. Select a group and click  or double click a group to view the components grouped together in a container then to select next highest group click  or to select a group under a drop down

View: Root



Multiple components added to group or container. This helps in simplifying the Project Graph view. The following figure illustrates the grouping of SPI Driver and SPI PLIB components. Similarly, user can create any number of group or container based on the requirement. Say for example separate container for TCP/IP and USB and so on. or based on the application need.



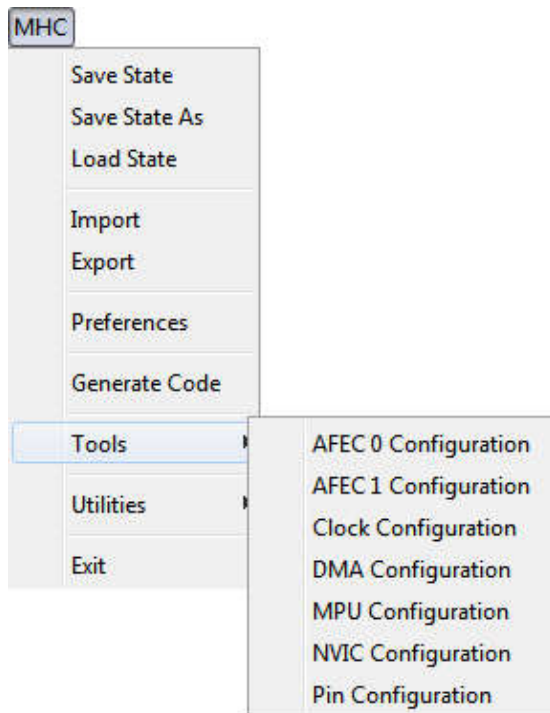
MHC Plug-ins

MHC plug-in consists of following plug-ins/managers based on the device selection while creating the project.

- **Clock Configuration:** Enables to configure Master, Generic, Peripheral and System Clocks

- Pin Configuration: Enables to configure pins in the Pin Configuration area depending upon the application requirements
- NVIC Configuration: Enables to configure enable/disable of interrupts, interrupt priority and name
- DMA Configuration: Enables to configure DMA Channels
- MPU Configuration: Enables to configure different zones of Memory Protection Unit

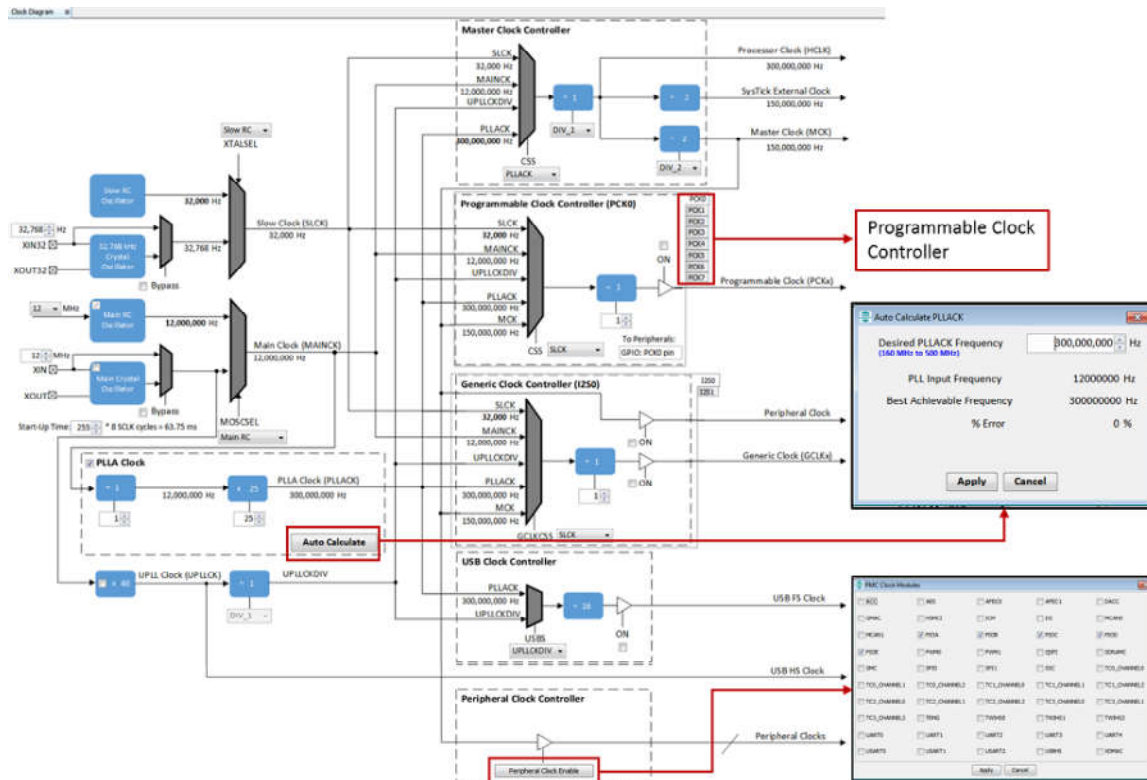
To open the MHC plug-in, perform this action *MHC* → *Tools* then click available plug-ins.



Note: MHC plugins availability will vary depending on the device selection while creating the project.

Clock Configuration

In the MHC, select the Clock Diagram Tab and verify that the clock is configured correctly for the selected target board. The Clock Diagram tab allows for easy setup of the master, system and peripheral clocks. See image below.



Pin Configuration

The Pin Manager consists of Pin Settings, the Pin Diagram, and the Pin Table tabs, which enables users to configure (assign peripheral function, set pin direction, configure pull-up or pull-down and so on) and map the I/O pins.

The following color combinations are associated with the pins in the graphical or table View:

- Gray: This pin is not usable in the selected configuration, and there is no enabled module which has any functionality on that pin. The grayed-out locks on a white background indicates the pins that are locked out by selected system functions.
- Blue: This pin is available and can be allocated to a module.
- Green (with a lock): This pin is allocated and selected for a module. The name displayed against the pin is either the name of the pin in the module's context or a custom name entered.

Pin Diagram

It is the pictorial representation of the available, assigned and not available pins of the Microcontroller. See image below.

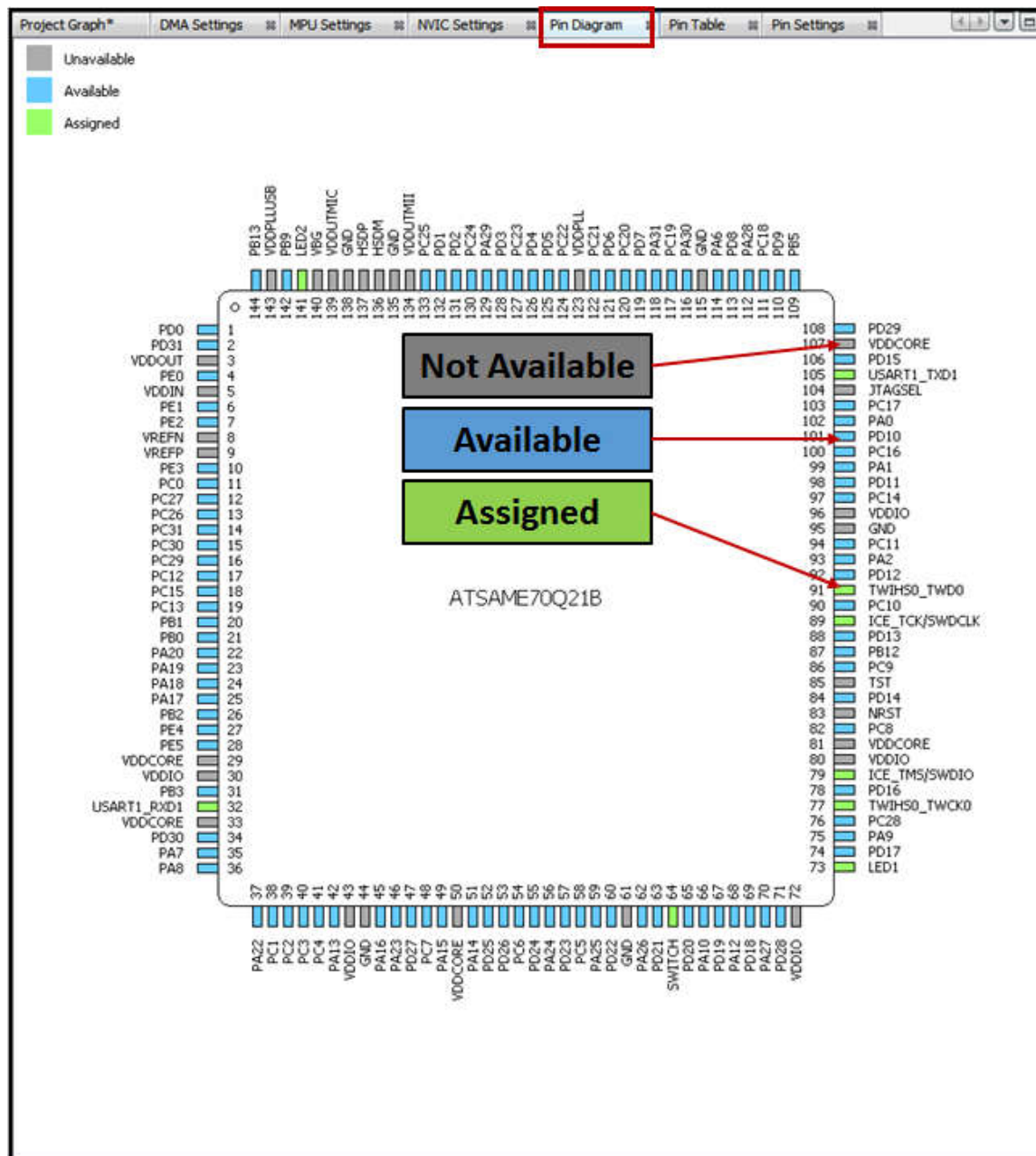


Figure 5-13. Pin Diagram

Pin Table

The pin table provides the Pin Manager Grid View. Using Pin Table users can perform these actions:

- The device package can be selected from the drop-down list.
- The package can be selected from the upper left side of the Pin Manager Table View.

The Package drop-down list shows the LQFP144 package is selected and the selected package details are displayed in the Package View. The pin numbers in the Table View provides the pin numbers for the selected package. The three leftmost columns in the Table View indicate the module's name, functionality name, and the direction.

Project Graph Pin Diagram Pin Table Pin Settings

Package: LQFP144

Module	Function	EB1_D0	VDDIN	EB1_D9	EB1_D10	VBEPN	VBEP	EB1_D11	EB1_D0	EB1_A9	EB1_A8	PC11	PC0	EB1_A1	PC12	EB1_A3	PC13	PE1	PE0	EB1_A1	PA19		
EB1	EB1_D15																						
	EB1_NANDOE																						
	EB1_NANDWE																						
	EB1_NCS0																						
	EB1_NCS1/SCKS																						
	EB1_NCS2																						
	EB1_NCS3																						
	EB1_NRD																						
	EB1_NWAIT																						
	EB1_NWR0/NWE																						
	EB1_NWR1/NB51/DQM1																						
	EB1_RAS																						
	EB1_SDA10																						
	EB1_SCK																						
EB1_SCKE																							

Figure 5-14. Pin Table

Pin Settings

Pin settings enables the user to configure the pins. Users can provide custom name for pin, change the pin function, direction, latch and other properties, see image below.

Pin Number	Pin ID	Custom Name	Function	Direction	Latch	Open Drain	PIO Interrupt	Pull Up	Pull Down	Glitch/Debounce Filter
102	PA0		Available	In	n/a		Disabled			Disabled
99	PA1		Available	In	n/a		Disabled			Disabled
93	PA2		Available	In	n/a		Disabled			Disabled
91	PA3	TWDH50_TWD0	TWDH50_TWD0	n/a	n/a		Disabled			Disabled
77	PA4	TWDH50_TWDK0	TWDH50_TWDK0	n/a	n/a		Disabled			Disabled
73	PA5	LED1	LED_AL	Out	High		Disabled			Disabled
114	PA6		Available	In	n/a		Disabled			Disabled
35	PA7		Available	In	n/a		Disabled			Disabled
36	PA8		Available	In	n/a		Disabled			Disabled
75	PA9		Available	In	n/a		Disabled			Disabled
66	PA10		Available	In	n/a		Disabled			Disabled
64	PA11	SWITCH	SWITCH_AL	In	Low		Disabled	<input checked="" type="checkbox"/>		Disabled
68	PA12		Available	In	n/a		Disabled			Disabled
42	PA13		Available	In	n/a		Disabled			Disabled
51	PA14		Available	In	n/a		Disabled			Disabled
49	PA15		Available	In	n/a		Disabled			Disabled
45	PA16		Available	In	n/a		Disabled			Disabled
25	PA17		Available	In	n/a		Disabled			Disabled
24	PA18		Available	In	n/a		Disabled			Disabled
23	PA19		Available	In	n/a		Disabled			Disabled
22	PA20		Available	In	n/a		Disabled			Disabled
32	PA21	USART1_RXD1	USART1_RXD1	n/a	n/a		Disabled			Disabled
37	PA22		Available	In	n/a		Disabled			Disabled

Figure 5-15. Pin Settings

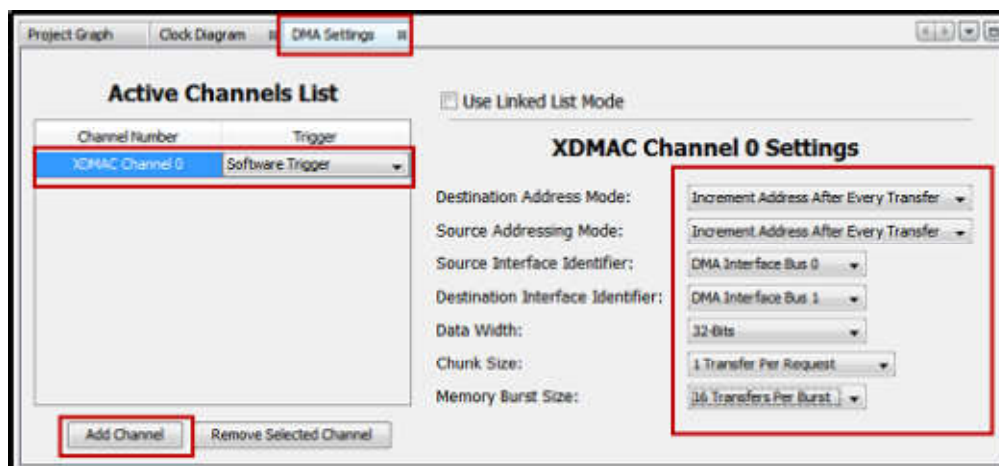
NVIC Configuration

NVIC Manager allows users to enable or disable interrupts, change priority, and change names of interrupt handlers, see image below.

Project Graph* Clock Diagram DMA Settings MPU Settings NVIC Settings				
Vector Number	Vector	Enable	Priority (0 = Highest)	Handler Name
-15	Reset (Reset Vector)	<input checked="" type="checkbox"/>	-3	Reset_Handler
-14	NonMaskableInt (Non-maskable Interrupt)	<input checked="" type="checkbox"/>	-2	NonMaskableInt_Handler
-13	HardFault (Hard Fault)	<input checked="" type="checkbox"/>	-1	HardFault_Handler
-12	MemoryManagement (Memory Management Fault)	<input type="checkbox"/>	0	MemoryManagement_Handler
-11	BusFault (Bus Fault)	<input type="checkbox"/>	0	BusFault_Handler
-10	UsageFault (Usage Fault)	<input type="checkbox"/>	0	UsageFault_Handler
-5	SVCall (SuperVisor Call)	<input checked="" type="checkbox"/>	0	SVCall_Handler
-4	DebugMonitor (Debug Monitor)	<input type="checkbox"/>	0	DebugMonitor_Handler
-2	PendSV (Pendable Service)	<input checked="" type="checkbox"/>	0	PendSV_Handler
-1	SysTick (System Tick Timer)	<input checked="" type="checkbox"/>	7	xPortSysTickHandler
0	SUPC (Supply Controller)	<input type="checkbox"/>	7	SUPC_Handler
1	RSTC (Reset Controller)	<input type="checkbox"/>	7	RSTC_Handler
2	RTC (Real Time Clock)	<input type="checkbox"/>	7	RTC_Handler
3	RTT (Real Time Timer)	<input type="checkbox"/>	7	RTT_Handler
4	WDT (Watchdog Timer)	<input type="checkbox"/>	7	WDT_Handler
5	PMC (Power Management Controller)	<input type="checkbox"/>	7	PMC_Handler
6	EFC (Enhanced Embedded Flash Controller)	<input type="checkbox"/>	7	EFC_Handler

DMA Configuration

Users can add a channel and perform the channel settings, such as setting up source and destination address mode, interface identifier, data width, chunk size, and memory burst size. See image below.



MPU Configuration

Users can configure MPU regions for memory spaces, such as ITCM, DTCM, Flash, SRAM, Peripherals, EBI, QSPI, USBHS and System. Users can also set the different attributes for these memory spaces. See image below:

Region	Enable	Memory Space	Region Name	Start Address (Hex)	Region Size	Memory Type	Data Access	Instruction Fetch	Shareable
0	<input checked="" type="checkbox"/>	ITCM	ITCM	0x0	4 MB	Normal memory, Non-cacheable	User: Read/Write, Privileged: Read/Write	<input checked="" type="checkbox"/>	<input type="checkbox"/>
1	<input checked="" type="checkbox"/>	FLASH	FLASH	0x400000	4 MB	Normal memory, Write-through cache	User: Read/Write, Privileged: Read/Write	<input checked="" type="checkbox"/>	<input type="checkbox"/>
2	<input checked="" type="checkbox"/>	DTCM	DTCM	0x20000000	4 MB	Normal memory, Non-cacheable	User: Read/Write, Privileged: Read/Write	<input checked="" type="checkbox"/>	<input type="checkbox"/>
3	<input checked="" type="checkbox"/>	SRAM	SRAM	0x20400000	8 MB	Normal memory, Write-back and write-allocate cache	User: Read/Write, Privileged: Read/Write	<input checked="" type="checkbox"/>	<input type="checkbox"/>
4	<input checked="" type="checkbox"/>	PERIPHERALS	PERIPHERALS	0x40000000	256 MB	Device Memory	User: Read/Write, Privileged: Read/Write	<input checked="" type="checkbox"/>	<input type="checkbox"/>
5	<input checked="" type="checkbox"/>	EBI_SMC	EBI_SMC	0x60000000	256 MB	Strongly-Ordered Memory	User: Read/Write, Privileged: Read/Write	<input checked="" type="checkbox"/>	<input type="checkbox"/>
6	<input checked="" type="checkbox"/>	EBI_SDRAM	EBI_SDRAM	0x70000000	256 MB	Device Memory	User: Read/Write, Privileged: Read/Write	<input checked="" type="checkbox"/>	<input type="checkbox"/>
7	<input checked="" type="checkbox"/>	QSPI	QSPI	0x80000000	256 MB	Strongly-Ordered Memory	User: Read/Write, Privileged: Read/Write	<input checked="" type="checkbox"/>	<input type="checkbox"/>
8	<input type="checkbox"/>	USBHS_RAM	USBHS_RAM	0xA0100000	1 MB	Device Memory	User: Read/Write, Privileged: Read/Write	<input type="checkbox"/>	<input type="checkbox"/>
9	<input type="checkbox"/>	SYSTEM	SYSTEM	0xE0000000	1 MB	Strongly-Ordered Memory	User: Read/Write, Privileged: Read/Write	<input type="checkbox"/>	<input type="checkbox"/>
10	<input type="checkbox"/>			0x0	32 Bytes	Strongly-Ordered Memory	User: No Access, Privileged: No Access	<input type="checkbox"/>	<input type="checkbox"/>
11	<input type="checkbox"/>			0x0	32 Bytes	Strongly-Ordered Memory	User: No Access, Privileged: No Access	<input type="checkbox"/>	<input type="checkbox"/>
12	<input type="checkbox"/>			0x0	32 Bytes	Strongly-Ordered Memory	User: No Access, Privileged: No Access	<input type="checkbox"/>	<input type="checkbox"/>
13	<input type="checkbox"/>			0x0	32 Bytes	Strongly-Ordered Memory	User: No Access, Privileged: No Access	<input type="checkbox"/>	<input type="checkbox"/>
14	<input type="checkbox"/>			0x0	32 Bytes	Strongly-Ordered Memory	User: No Access, Privileged: No Access	<input type="checkbox"/>	<input type="checkbox"/>
15	<input type="checkbox"/>			0x0	32 Bytes	Strongly-Ordered Memory	User: No Access, Privileged: No Access	<input type="checkbox"/>	<input type="checkbox"/>

Console

The Console tab displays the MHC operation results. See below image.

```

<Generate>[Info]: *** Generating Configuration ***
<Generate>[Info]:
<Generate>[Info]: Validating file list.
<Generate>[Info]: Processing file list.
<Generate>[Info]: All files successfully pre-processed.
<Generate>[Info]: Removing previously generated files and settings.
<MHC>[Info]: Removing project setting "C32:extra-include-directories" value
"./src;../src/config/sam_e70_xult;../src/packs/ATSAME70Q21B_DFP;../src/packs/CMSIS/CMSIS/Core/Include;../src/packs/CMSIS/"
<MHC>[Info]: Removing project setting "C32-LD:heap-size" value "0"
<MHC>[Info]: Removing project setting "C32-LD:no-device-startup-code" value "true"
<MHC>[Info]: Removing project setting "C32Global:mdtcm" value ""
<MHC>[Info]: Removing project setting "C32Global:mitcm" value ""
<MHC>[Info]: Removing project setting "C32Global:mstackcm" value "false"
<MHC>[Info]: Removing file: "config/sam_e70_xult/exceptions.c"
<MHC>[Info]: Removing file: "config/sam_e70_xult/initialization.c"
<MHC>[Info]: Removing file: "config/sam_e70_xult/interrupts.c"
<MHC>[Info]: Removing file: "config/sam_e70_xult/libc_syscalls.c"
<MHC>[Info]: Removing file: "/main.c"
<MHC>[Info]: Removing file: "config/sam_e70_xult/peripheral/clock/plib_clk.c"
<MHC>[Info]: Removing file: "config/sam_e70_xult/peripheral/mpu/plib_mpu.c"
<MHC>[Info]: Removing file: "config/sam_e70_xult/peripheral/nvic/plib_nvic.c"
<MHC>[Info]: Removing file: "config/sam_e70_xult/peripheral/pio/plib_pio.c"
<MHC>[Info]: Removing file: "config/sam_e70_xult/peripheral/sdramc/plib_sdramc.c"
<MHC>[Info]: Removing file: "config/sam_e70_xult/peripheral/systick/plib_systick.c"
<MHC>[Info]: Removing file: "config/sam_e70_xult/peripheral/usart/plib_usart.c"
  
```

Generating Code

The previous sections of this document cover installing MHC and operating areas. The following sections cover basic MHC operations and code generation process, which helps users to setup and configure MHC according to their needs.

Selecting and Configuring Modules

After opening the MHC, the peripheral and library modules can be added to the project by following these steps:

1. In the Available Components Area, select the peripheral or library module which needs to be configured by clicking on the module's name. See below image.
2. Configure the peripheral as required for the application in the Configuration

3. Configure the pins in the Pin Manager depending on the application requirements.

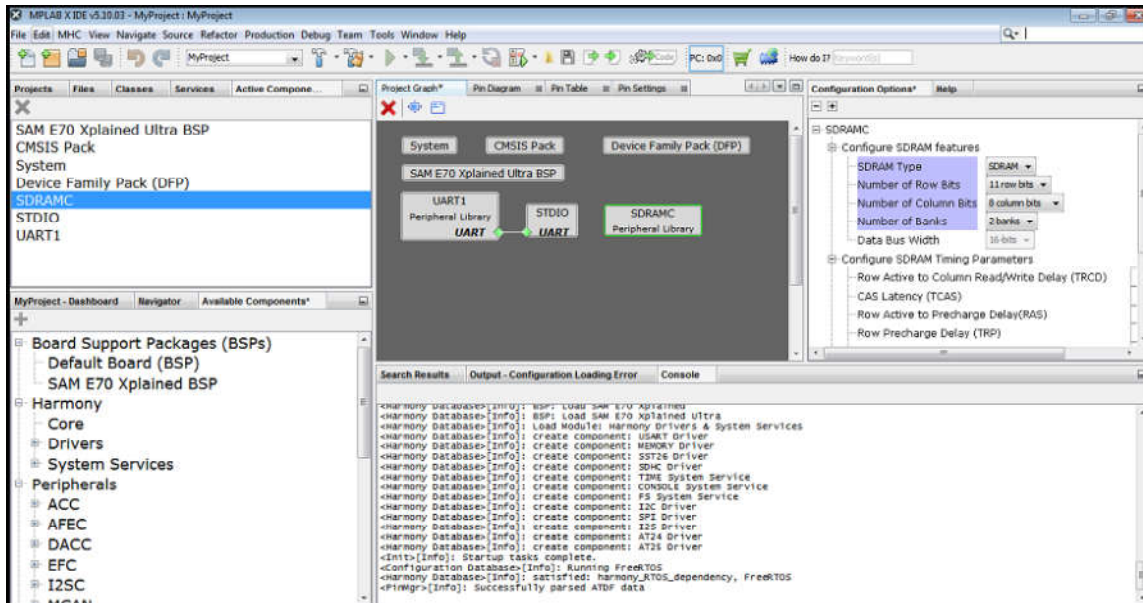



Figure 6-1. Configuring Modules

Code Generation

To perform the following actions, press  in the Project Resources window. Users can see the code generation by pressing the MHC Console tab. See [Section 5.4.6. Console](#).

1. Saving the MHC configuration file. See Figures 6-2 and 6-3.
2. Generating the project request, and changes merge strategy. See Figure 6-4.
3. Generating code for the peripheral or library module if the module's configuration has changed since the last time MHC generated code for that module.

If MHC attempts to regenerate a file on the disk that has been modified outside of MHC, the Merge [MHC] window is displayed. The Merge [MHC] window allows you to select whether to keep the modified file (default action), or to replace the modified content with the content generated by MHC.

The Merge [MHC] window is discussed in [Section 6.2.2 "The Merge \[MHC\] Window"](#).

Saving and Loading the MHC Configuration

Saving and loading the MHC configuration is integrated into the MPLAB® X IDE Save and Load functions. The MHC configuration is saved whenever the Generate button is pressed. You can also save the MHC configuration by clicking on the MPLAB® X IDE Save tool or selecting Save from the File menu.

The MHC configuration file is included in the MPLAB® X IDE project, under the Important Files folder. The configuration file uses the extension .xml. Double-clicking on the MHC configuration file will cause that MHC configuration to be loaded.

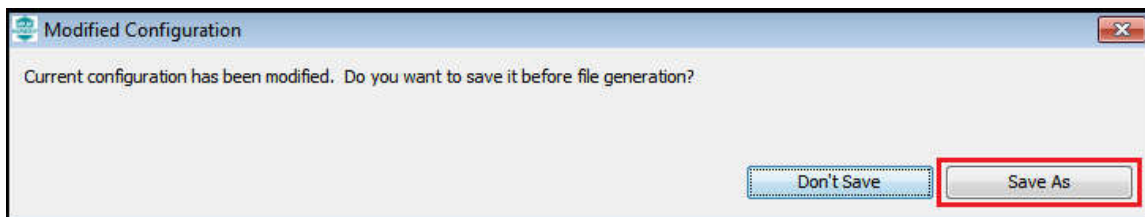


Figure 6-2. Modified Configuration

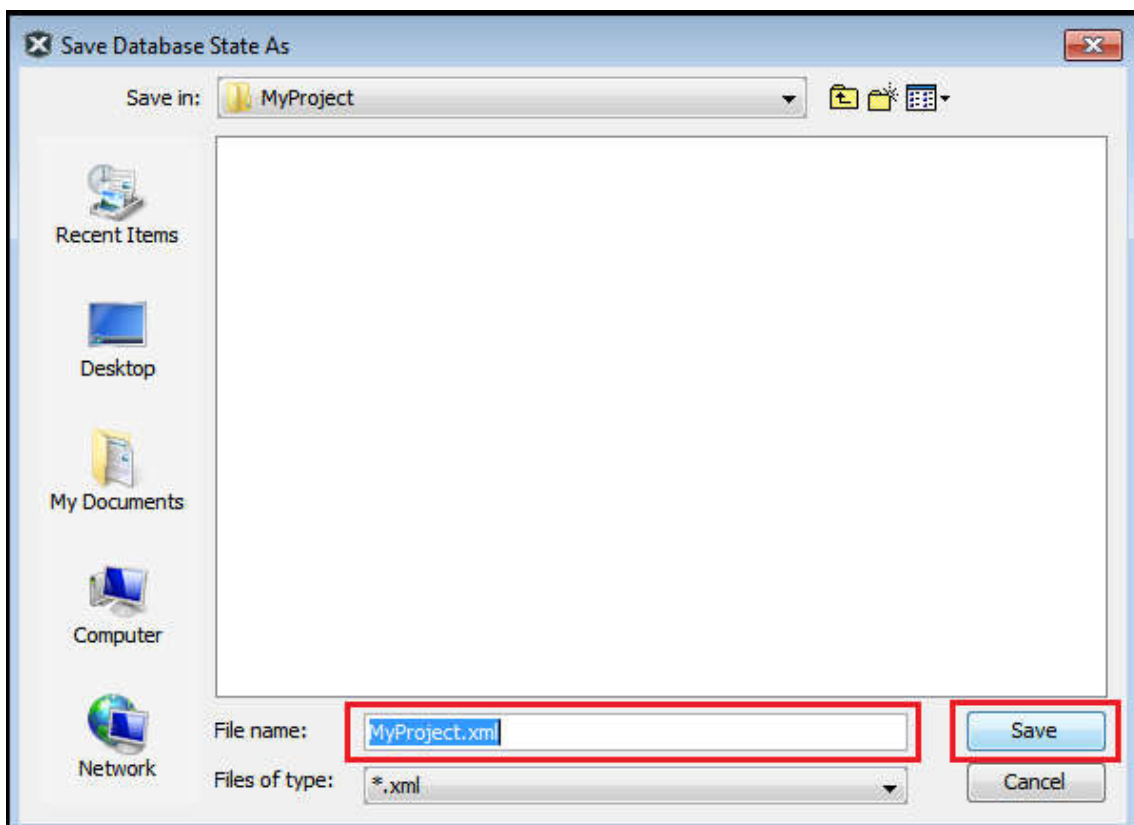


Figure 6-3. Save Database

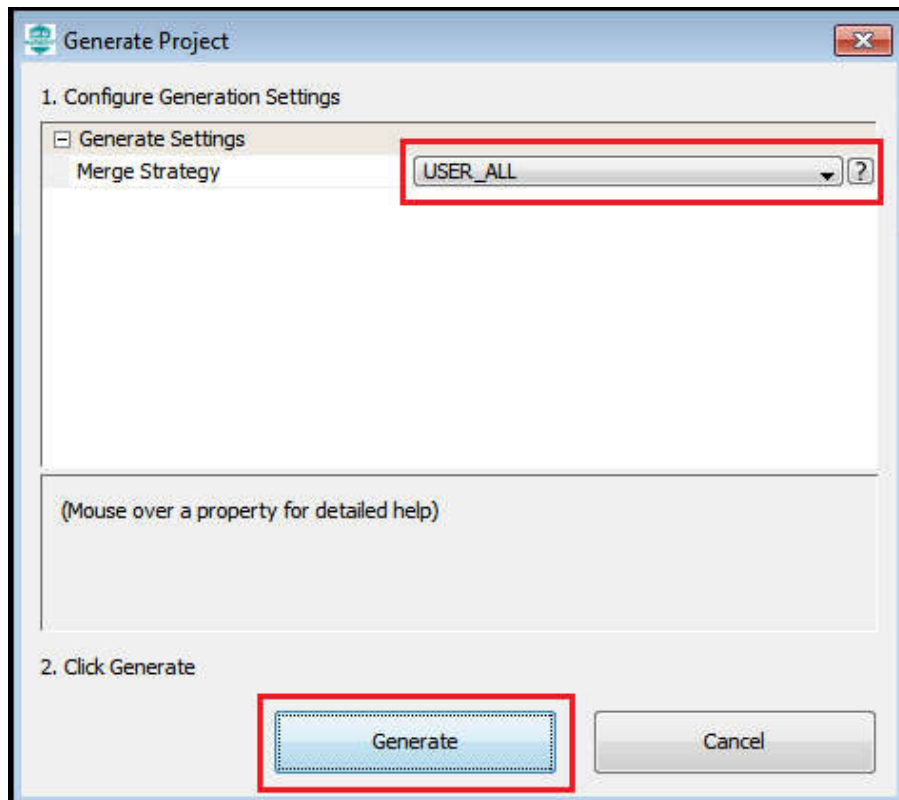


Figure 6-4. Generate Project

The Merge [MHC] Window

If any of the files generated by the MHC has been edited outside of MHC and saved to the disk, then the Merge [MHC] window will appear in the Configuration Options. The Merge [MHC] window allows you to decide whether to keep the edits, or to replace them with the MHC newly generated code.

When the newly generated MHC content has been accepted, MHC makes the changes to the file. To the MPLAB® X IDE, these changes are the same as if you typed in the new content. The normal MPLAB® X IDE edit undo can be used in the MHC Merge operation.

A list of all the files that need to be merged appear one after another after resolving the merge conflicts. You must resolve merge conflicts to ensure that all the newly generated code is incorporated into the project.

At the top of the Merge [MHC] window, in the center margin, there is an arrow, as shown in Figure 5-5. Clicking on the arrow will replace all your edits in the current file with the MHC updated code that the MHC has just generated. The numbers above the arrow indicate the current difference and the total number of differences.

The individual lines of MHC Updated Code can be selected to replace the edited code. As shown in Figure 5-5, clicking on the arrows in the right margin of the left window will copy the MHC Updated Code to the generated driver file. Once the changes are accepted, the Merge mechanism will remove the highlighted file and highlight the next file on the list. To insure all updates are completed, a warning will be generated if the Merge mechanism is closed before all the changes are accepted.

Note: Your edits will never be overwritten by the MHC generated code, except by explicit action in the Merge [MHC] window.

When changes have been made both to a generated file and in the corresponding MHC UI within the Configuration Options, the Merge [MHC] window shown in Figure 5-5 will be displayed. The Merge [MHC] window allows you to resolve the conflicts between the newly generated file and the edits you have made to the file.

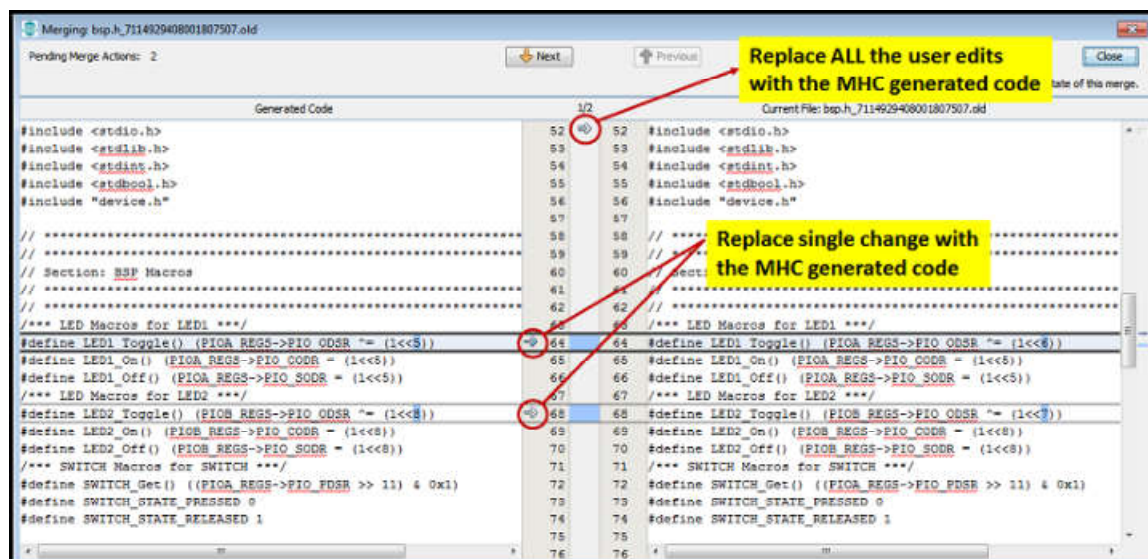


Figure 6-5. MHC Merge Tool

Generated Source and Header Files

The generated code will be included in the active MPLAB® X IDE Project as shown in the below image. The header files are shown on the top and source files are shown on the bottom.

Note: In the image below, the icons are used to organize the project virtual organization of files, not an actual one.

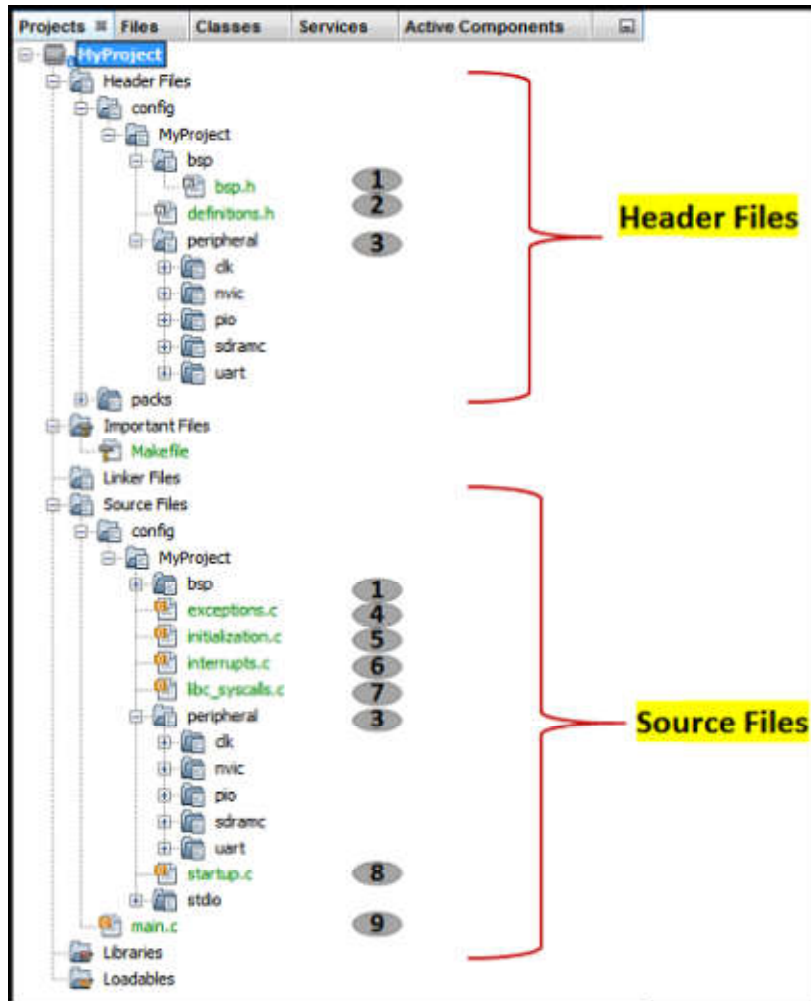


Figure 7-1. Project Area

The following table provides the Header and Source files generated from the sample project:

SI No.	Source File	Descriptions
1	bsp.h	Provides Board Support Package
2	definitions.h	Provides all library headers and definitions needed for the application
3	Peripheral (libs)	Supports peripherals used by the project
4	exceptions.c	Implements all exception handlers
5	initialization.c	Initializes all libraries and applications

6	interrupts.c	Provides the interrupt vector table
7	libc_syscalls.c	Provides the Harmony specific system calls
8	startup.c	Startup code for the application
9	main.c	Applications Main source file



► Pages 3

[Harmony Configurator v3 User Guide](#)

[Create Your First Peripheral Library Project with Harmony 3](#)

Clone this wiki locally

<https://github.com/Microchip-MPLAB-Harmony/Microchip-MPLAB-Harmony.github.i>

