




Getting Started with PSF



MICROCHIP
Microchip Technology, Inc.

Microchip Technology, Incorporated
2355 W. Chandler Boulevard
Chandler, Arizona 85224
480/792-7200

REV	DATE	DESCRIPTION OF CHANGE
0.92	02-Dec-19	Initial revision
0.95	07-Jan-20	Updated the path of User guide and PSF Stack. Updated section 6 Renamed Appendix 8.1 to Non-Professional XC32 Compiler Edition Changed the order of sections - Software License, Terms and Abbreviations and Introduction Added section 8.2 Harmony Framework usage
1.00	26-Feb-20	Updated document version to align with v1.00 release
1.01	16-Mar-20	Updated Section 6 and document version to align with v1.01 release
1.02	8-Apr-20	Updated Section 6 and document version to align with v1.02 release
1.04	26-May-20	Updated document version to align with v1.04 release
1.05	24-Jul-20	Updated document version to align with v1.05 release
1.06	12-Aug-20	Updated Section 6 and document version to align with v1.06 release
1.07	08-Sep-20	Updated document version to align with v1.07 release

Table of Contents

1	Software License Agreement	4
2	Terms and Abbreviations	4
3	Introduction	4
4	PSF Overview.....	5
5	Prerequisites for PSF	5
6	Building the Project.....	6
7	Programming the PSF-EVB	8
8	Appendix	9
8.1	Non-Professional XC32 Compiler Edition.....	9
8.2	Harmony Framework Usage.....	10

1 Software License Agreement

Copyright ©[2019-2020] Microchip Technology Inc. and its subsidiaries.

Subject to your compliance with these terms, you may use Microchip software and any derivatives exclusively with Microchip products. It is your responsibility to comply with third party license terms applicable to your use of third party software (including open source software) that may accompany Microchip software.

THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

2 Terms and Abbreviations

Term	Definition
USB	Universal Serial Bus
PSF	USB Power Delivery Software Framework
EVB	Evaluation Board
PD	Power Delivery
MCU	Micro Controller Unit
MHC	MPLAB Harmony Configurator
IDE	Integrated Development Environment
IPE	Integrated Programming Environment
PDO	Power Data Object

3 Introduction

USB Power Delivery Software Framework (PSF) is an open source Power Delivery stack designed to be integrated into any suitably powerful MCU and is used to control one or more UPD350 PD controllers within a USB Type-C Power Delivery System.

This document gives an overview of the software and hardware tools required for the PSF Firmware to work properly with PSF-EVB. It also provides information on how to build the Firmware and program the hex file in the PSF-EVB.

4 PSF Overview

- The USB Power Delivery Software Framework (PSF) with USB-PD Port Controller UPD350 is an effective USB-PD solution which offers the user the ability to customize functionality to meet individual application needs.
- The PSF stack is highly portable and designed to run on different MCU Hardware platforms. Versatility towards different HW platforms is achieved through the flexibility and configurability of the PSF stack.
- The PSF stack is compliant to USB Power Delivery 3.0 & Type-C specification v1.3
- PSF supports single and multi-port solutions.

PSF User Guide

- The ‘PSF User Guide’ gives a detailed overview of the PSF stack architecture, directory structure, Supported and Not Supported messages and the various configuration options that are available in PSF.
- The User Guide also provides information about the requirements and steps for integration of PSF into a new platform.
- PSF User Guide is available at <https://github.com/MicrochipTech/usb-pd-software-framework/tree/master/Docs>

5 Prerequisites for PSF

The following are the prerequisites for PSF to work with the PSF-EVB or a Microchip MCU. To port PSF to any non-Microchip MCU the appropriate IDE, compiler and other software tools shall be used. Regardless of MCU chosen one UPD350 PD Controller is required per port.

- [MPLAB X IDE v5.30](#) or later
- [MPLAB XC32 compiler](#)
- [Atmel ICE Debugger](#)

Follow these steps for setting up the build environment needed for PSF Firmware.

1. MPLAB X IDE can be downloaded and installed from *Downloads* tab located at the bottom of <https://www.microchip.com/mplab/mplab-x-ide>
For any IDE specific information, refer the MPLAB X IDE Release Notes and User Guide of the installed version listed in the same page.
2. MPLAB XC32/32++ Compiler can be downloaded and installed from *Downloads* section located at the bottom of <https://www.microchip.com/mplab/compilers>

For any compiler specific information, refer the *Compiler User's Guide for PIC32C/SAM MCUs* listed in the *XC32 Documents* tab of the same page.

6 Building the Project

Follow these steps for building the PSF Project and generating the hex file. Building the project is demonstrated here by having "PSF_EVB_Source_Lite" project as an example. The similar approach can be followed for Source Pro, Sink and DRP projects.

1. Download PSF Firmware from <https://github.com/MicrochipTech/usb-pd-software-framework> using *Clone or Download* option. All the sub folders must be retrieved for the project to build successfully.
2. Extract the downloaded zip file usb-pd-software-framework-master.zip.
3. The entire local folder path containing the PSF source code shall have no space in the folder names.
For ex: Instead of *C:\Users\\Desktop\PSF FW\PSF*, use *C:\Users\Desktop\PSF_FW\PSF*.
4. Remove Read-Only option for the folder by right click -> Properties -> Uncheck Read Only -> Apply -> OK
5. Open MPLAB X IDE and click File -> Open Project.
Enter the path {Local_Folder}\PSF\Demo\PSF_EVB_Source_Lite\firmware which contains PSF_EVB_Source_Lite.X

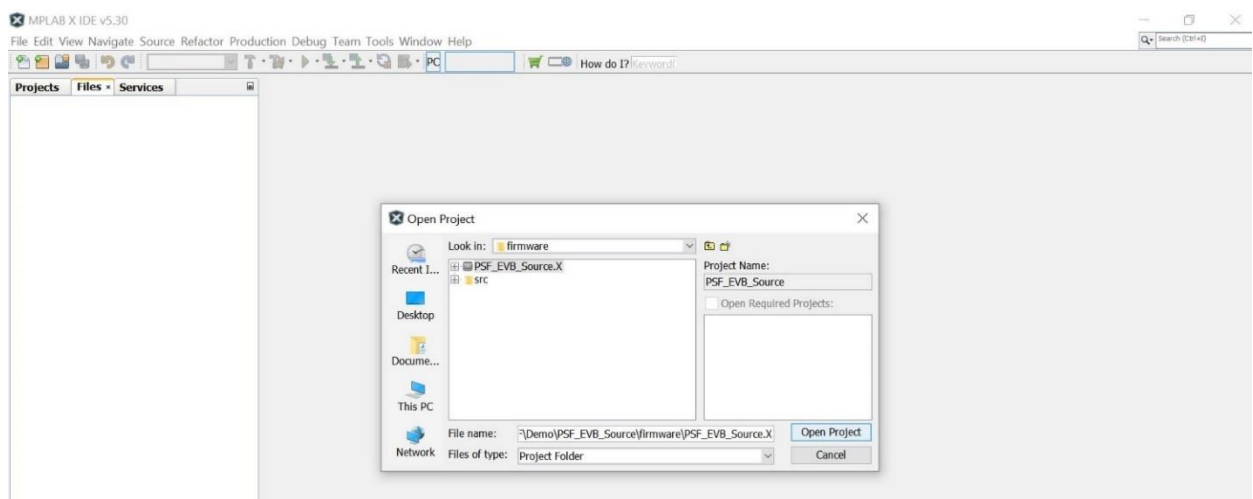


Figure 6.1 Opening project in MPLAB X IDE

6. The IDE may throw a configuration load error if the compiler toolchain is not properly linked to the project. With this error, building of the project cannot proceed.

warning: Configuration "default" builds with "XC32", but indicates no toolchain directory.

error: Configuration "default" builds with "XC32", but no toolchains of that type are installed.

Errors have occurred while loading one or more configurations.

If a specific error is not shown above, this may happen when you import a project from another computer.

- + You can add language tools in Tools->Options embedded tab.
- + You can change which language tool to use in the project properties dialog.

To resolve this, click Tools -> Options -> Embedded -> Build Tools -> Scan for Build Tools
Once done, all the compiler toolchains installed in your PC should be listed under Toolchain.

7. Choose the XC32 compiler installed by you and click Apply -> OK Once done, Compiler Toolchain will be properly listed in the project Dashboard as shown in the figure.

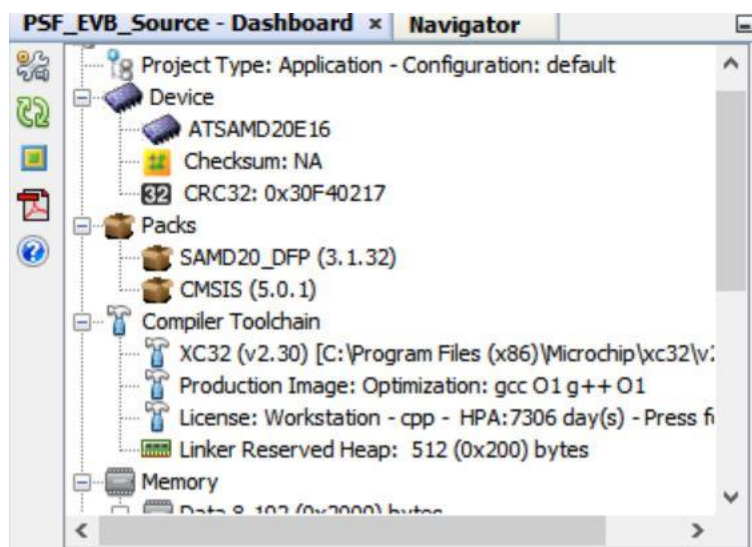



Figure 6.2 Project Dashboard in MPLAB X IDE

8. Make the project as main project by right clicking on the project name (PSF_EVB_Source_Lite) and selecting *Set as Main Project*.
9. Follow the steps mentioned in [Appendix 8.1](#) in case you are using a non-professional XC32 compiler version.
10. Right click on the project and select *Clean and Build* or use  icon at the top of IDE to build the firmware.
11. Once the build completes, a *Build Successful* message will be shown in the output window of the IDE.
12. Generated Hex file can be found under the path

{Local_Folder}\PSF\Demo\PSF_EVB_Source_Lite\firmware\PSF_EVB_Source_Lite.X\dist\default\production

7 Programming the PSF-EVB

Follow these steps to program the generated hex file in the PSF-EVB

1. Open MPLAB IPE which should have been downloaded and installed during the MPLAB X IDE installation.

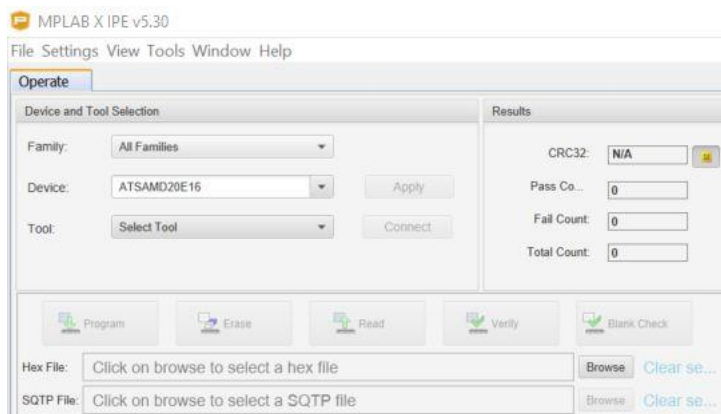


Figure 7.1 MPLAB IPE settings

2. Select *Family* as 'All Families' and *Device* as 'ATSAMD20E16' and click *Apply*
3. Connect Atmel ICE Debugger to the PC. Connect the other end of debugger to the ICE I/F(J19) of PSF-EVB. A dot will be present in Atmel ICE which gives an indication that this pin should be connected to 3v3 of J19.
4. Once debugger is connected, it will be listed under the *Tool* tab of the IPE and a yellow dot will be shown at the right side of the device.

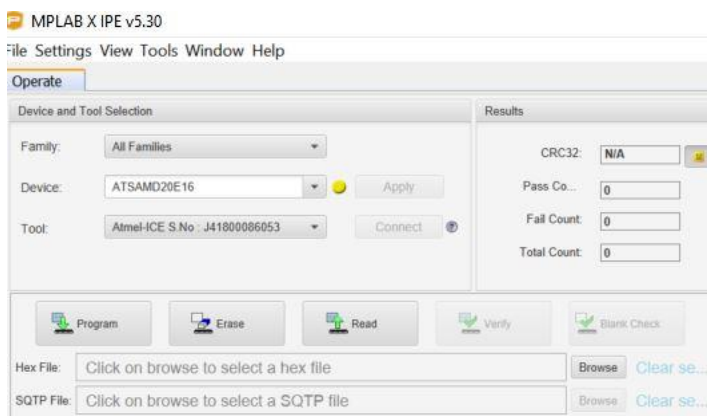
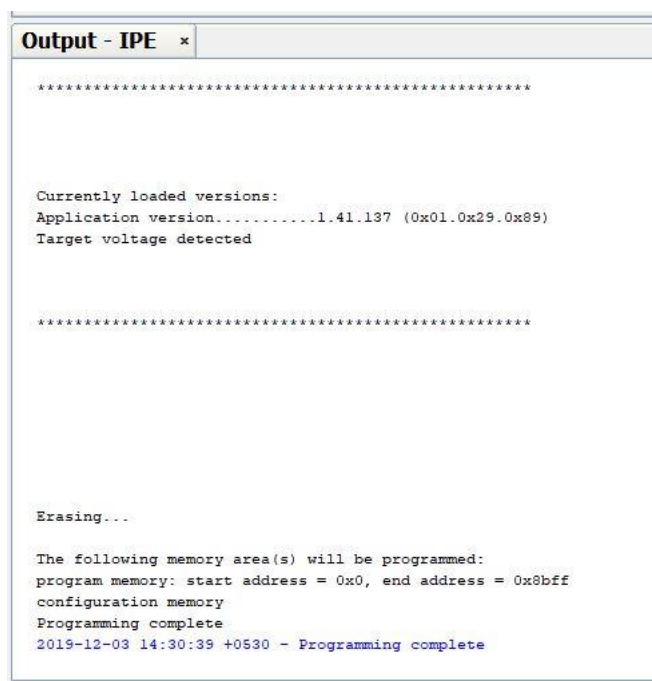


Figure 7.2 MPLAB IPE with Atmel-ICE detected

5. Browse through the path of hex file.
6. Once hex file path is given, output window of IPE will display a message like this.

*Loading code from C:\Users\Downloads\usb-pd-software-framework-public-master@6c2c24923d9\PSF\Release\v0.91\PSF_Hades_Source_V0.91.hex...
2019-12-03 14:14:08 +0530 - Hex file loaded successfully.*

7. Power on the PSF-EVB and click *Program* in the IPE.
8. Once the hex file is programmed successfully, the output window of the IPE will show the following message.



```

*****

Currently loaded versions:
Application version.....1.41.137 (0x01.0x29.0x89)
Target voltage detected

*****

Erasing...

The following memory area(s) will be programmed:
program memory: start address = 0x0, end address = 0x8bff
configuration memory
Programming complete
2019-12-03 14:30:39 +0530 - Programming complete
  
```

Figure 7.3 IPE Output Window

8 Appendix

8.1 Non-Professional XC32 Compiler Edition

If the PSF firmware is built with a non-professional XC32 compiler, build errors may occur because of compiler optimization issues. In case you are using a non-professional XC32 compiler and want the build process to be error free and successful, please follow these steps. Refer the image shown below for easier navigation.

1. Set the project as the main project by right clicking on the project and selecting *Set as Main Project*.
2. Click File -> Project Properties
3. Choose xc32-gcc under XC32(Global Options) and select *Optimization* under Option Categories.
4. Select optimization-level as 0.

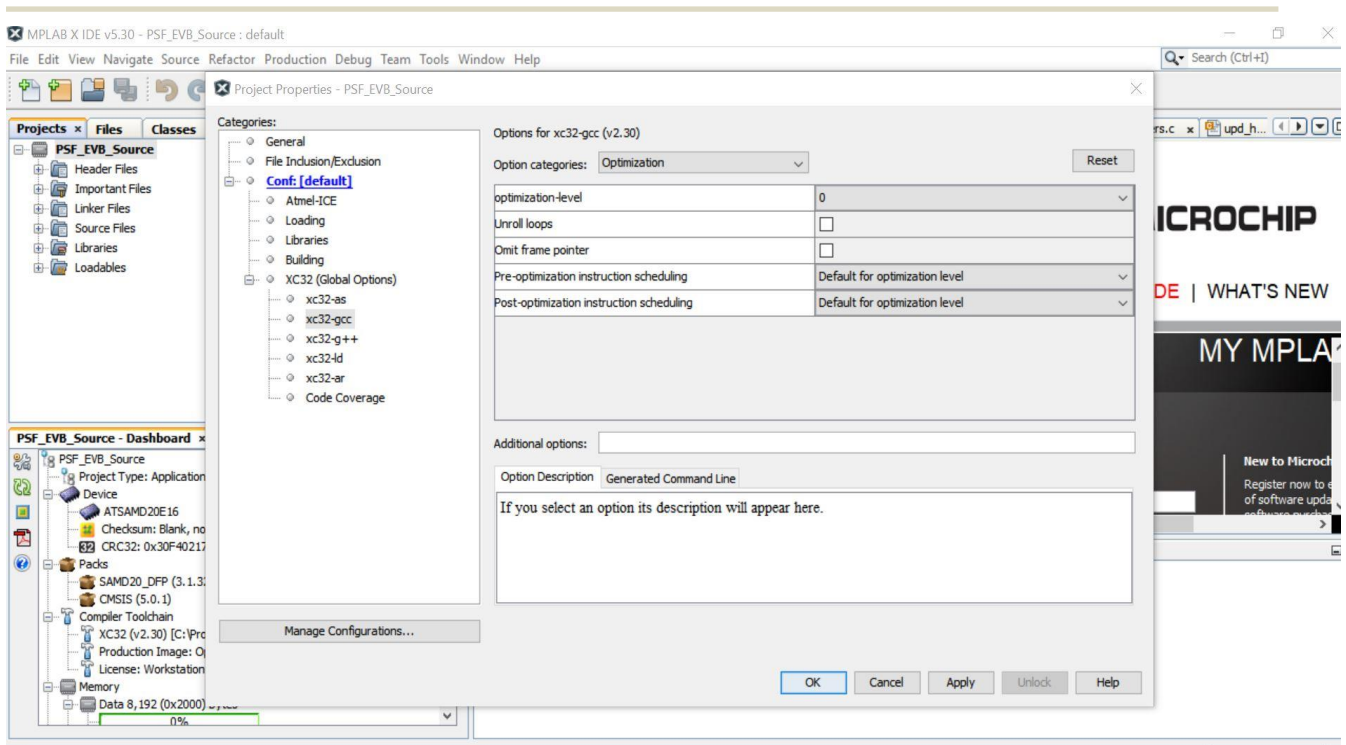


Figure 8.1 Optimization settings in MPLAB X IDE

8.2 Harmony Framework Usage

1. Download MPLAB Harmony Configurator 3 by choosing Tools -> Plugins -> Available Plugins -> MPLAB Harmony 3 Configurator -> Install in the MPLAB X IDE.

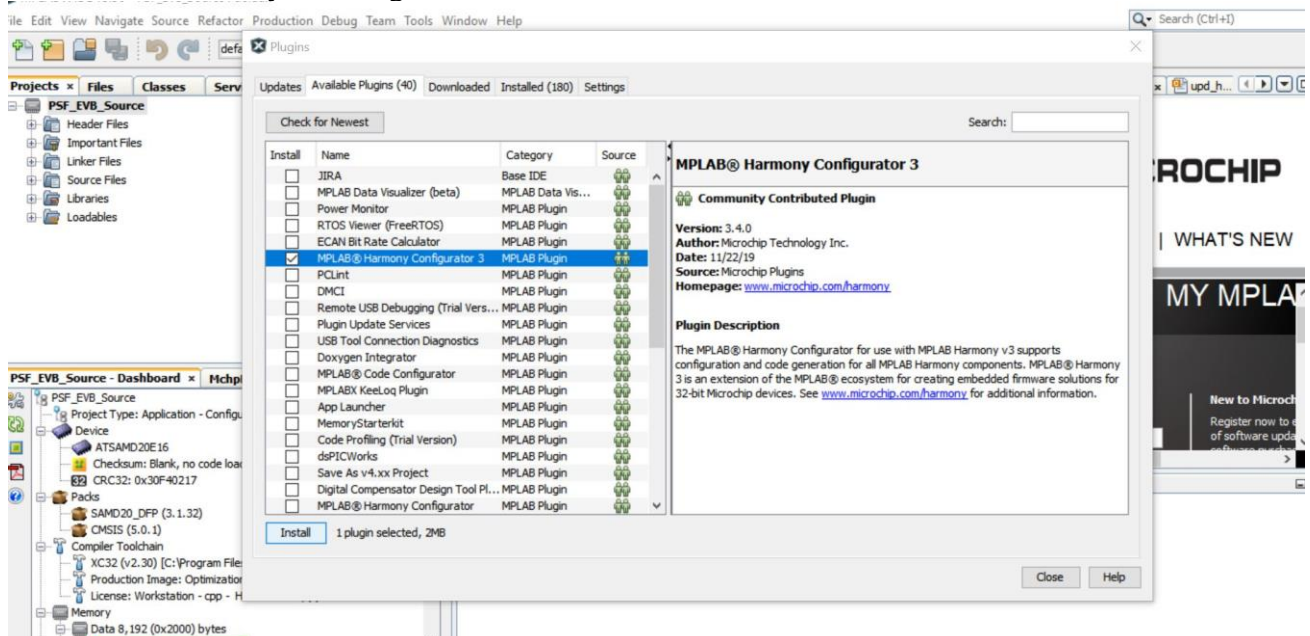


Figure 8.2 Harmony Framework installation

2. To download basic Harmony Configuration, Go to Tools -> Embedded -> MPLAB Harmony 3 Content Manager

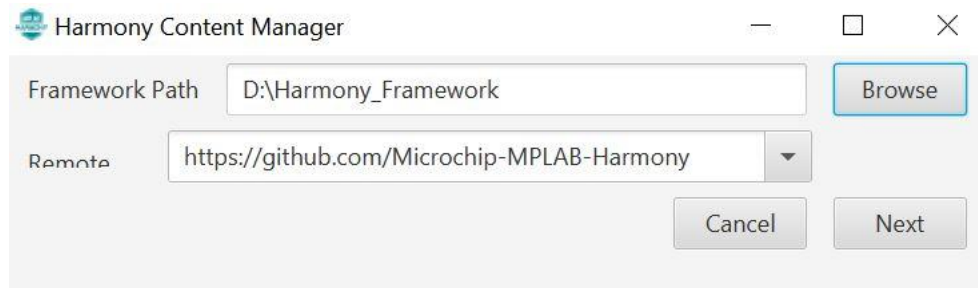


Figure 8.3 Harmony Content Manager

3. Choose a local *Framework Path* with a new folder created explicitly for downloading harmony packages, say *Harmony_Framework*. If the folder has any other local files, user might face difficulties in downloading the packages.
4. Choose GitHub Harmony path for *Remote* and click *Next*. The local Framework Path shall not have any spaces in the folder name.
The following screen will appear indicating that the Connection was successful.

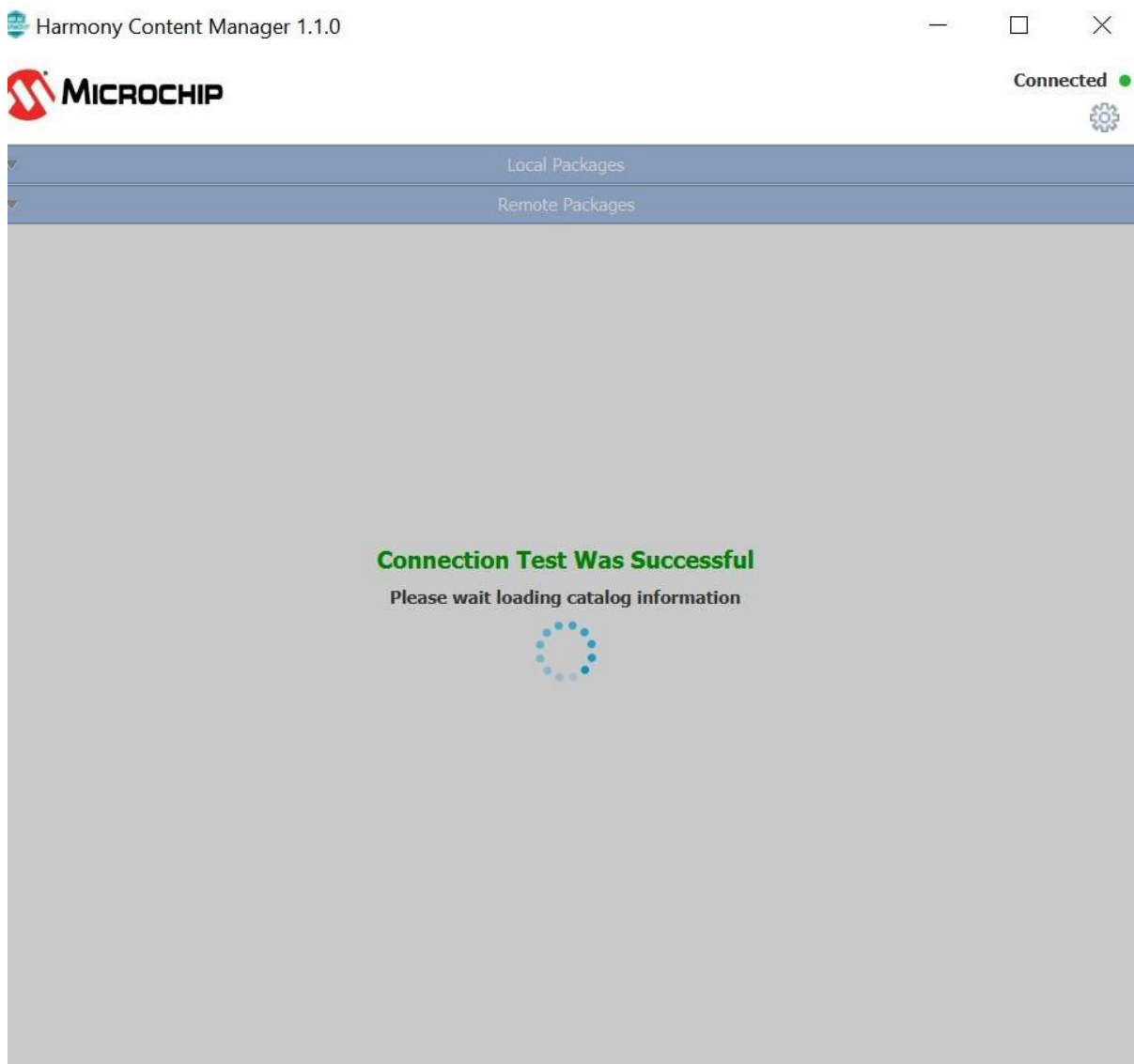


Figure 8.4 Harmony Content Manager with Connection Successful message

5. Select the Remote Packages – ‘mhc, dev_packs, bsp and csp’ and click *Download selected*.

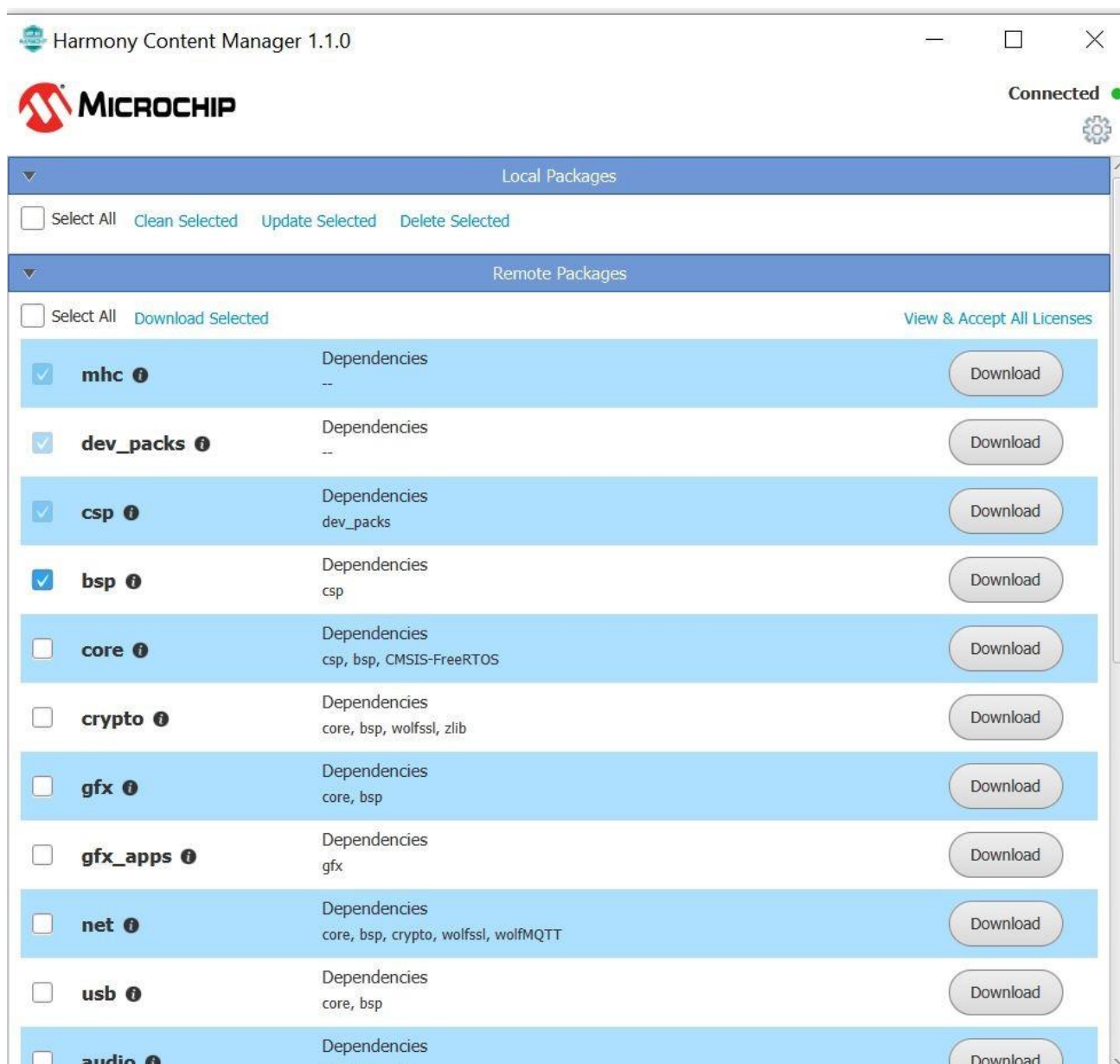


Figure 8.5 Remote Packages in Harmony Content Manager

6. A License Window will pop up. Read the license terms and accept before proceeding to the next step. Close the window.

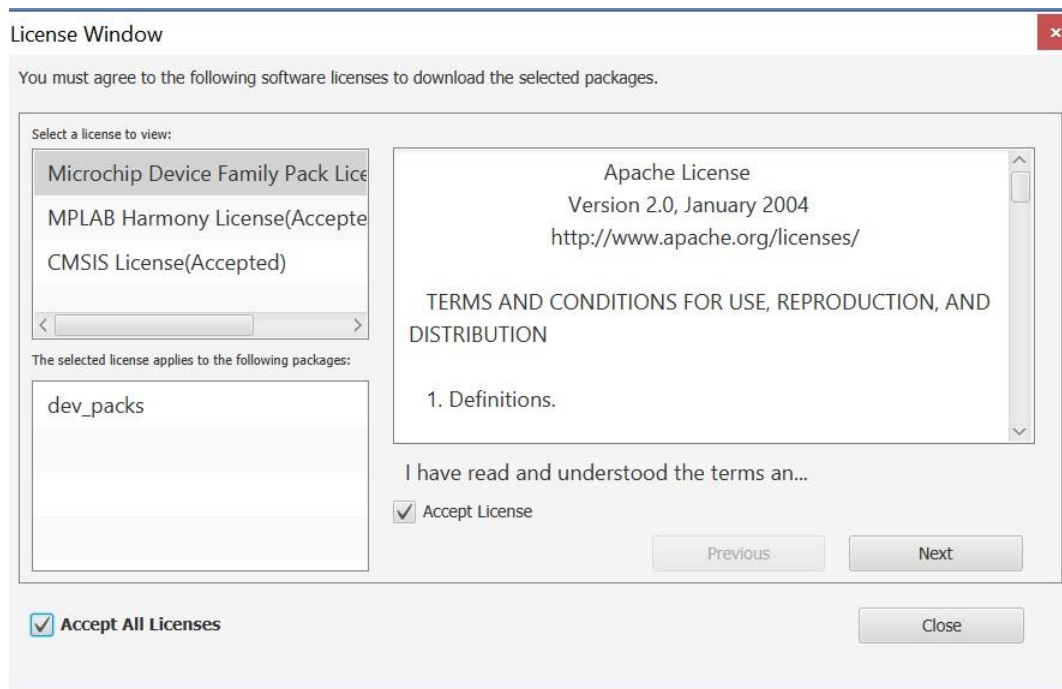


Figure 8.6 License Agreement

7. Current Download Status will be shown until remote packages are downloaded to local.

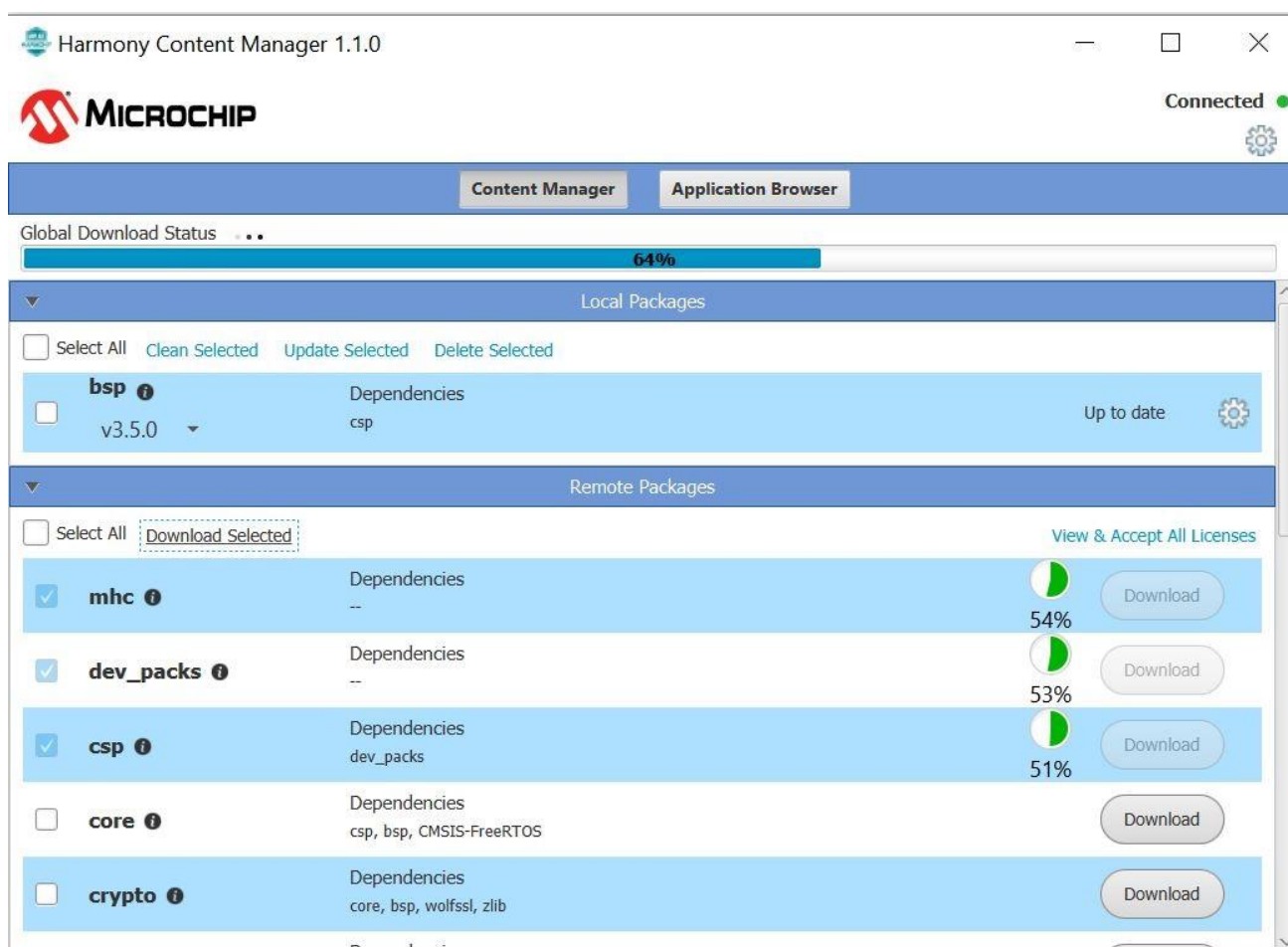


Figure 8.7 Download status of Packages

8. Once all the remote packages are downloaded, they will be listed under *Local Packages* section.

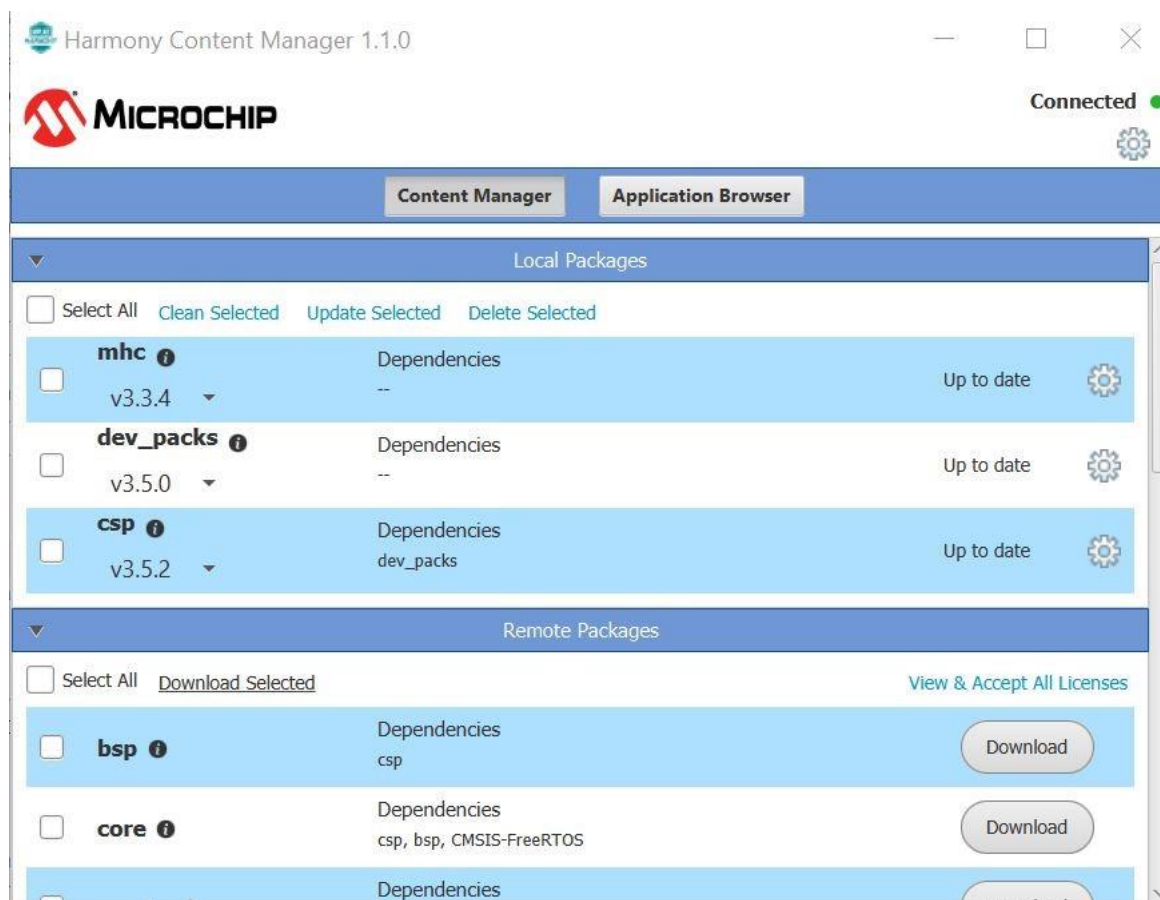


Figure 8.8 Harmony Content Manager with packages downloaded

9. To launch Harmony 3 Configurator, choose Tools -> Embedded -> MPLAB Harmony 3 Configurator

10. Select *MPLAB Harmony Project Path* and choose *Reconfigure Paths*

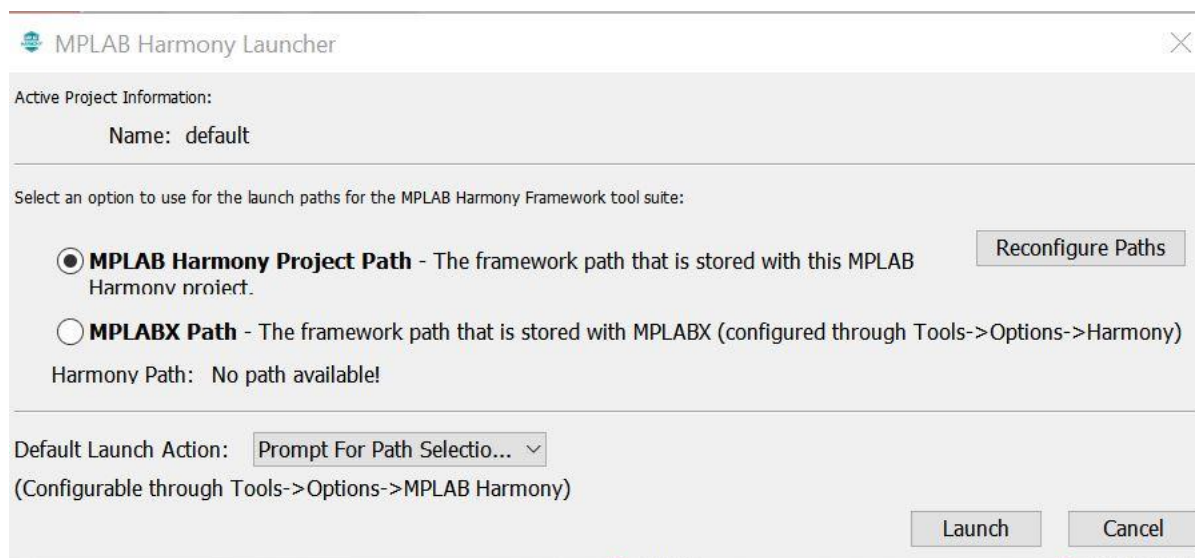


Figure 8.9 MPLAB Harmony Launcher

11. Choose the Harmony Framework path by clicking the folder icon(Green box) and click Ok

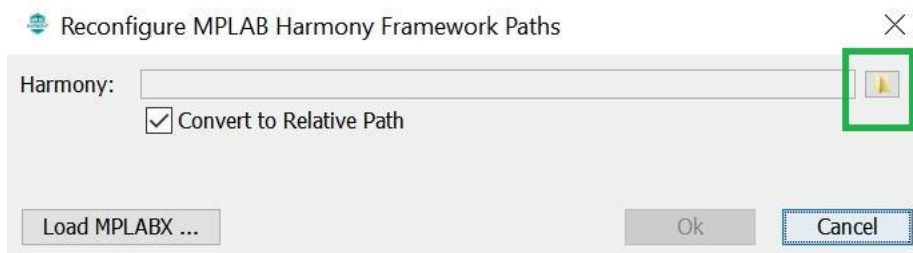


Figure 8.10 MPLAB Harmony Framework Paths

12. Click *Launch*.

13. *Window Manager Selection Dialog* window will open. Select Native window management mode if you want the Harmony tabs to be embedded into MPLABX IDE else choose *Standalone* mode for separate MHC window. This will be prompted during first time only.

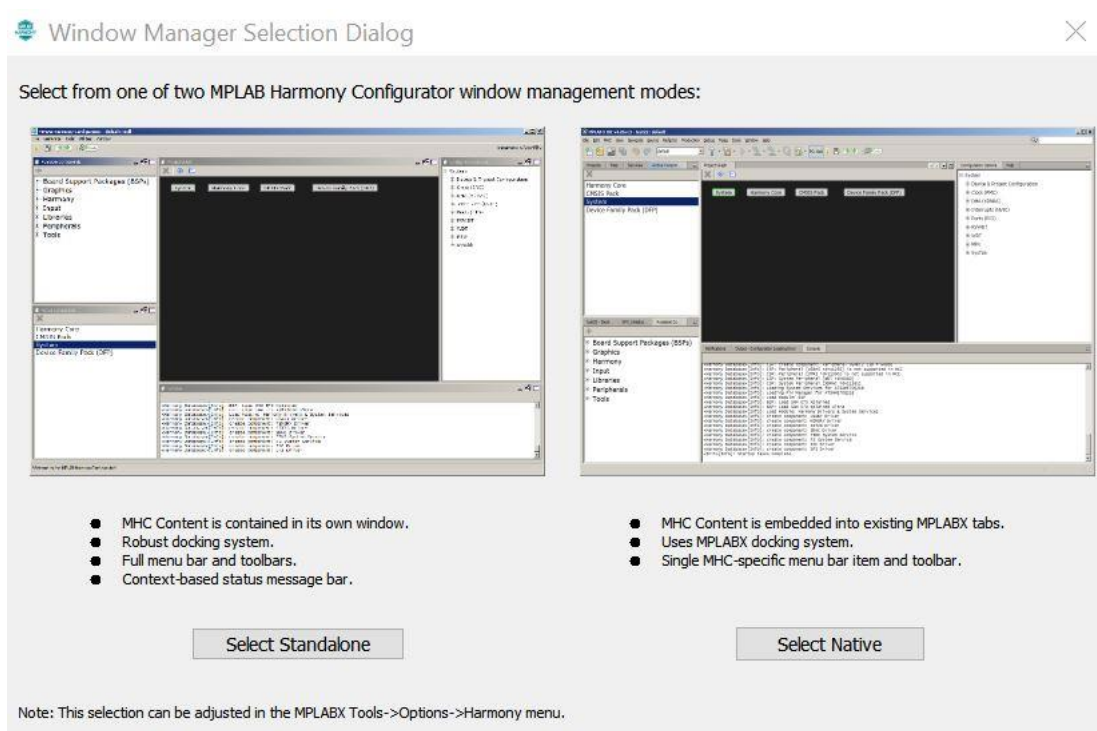


Figure 8.11 Window Manager Selection Dialog

14. After having an overview of Native Window Manager Quick Tour, click *I Understand*. This will also appear for the first time only.

15. *Configuration Database Setup* page appears with DFP and CMSIS path shown by default. Click *Launch*.

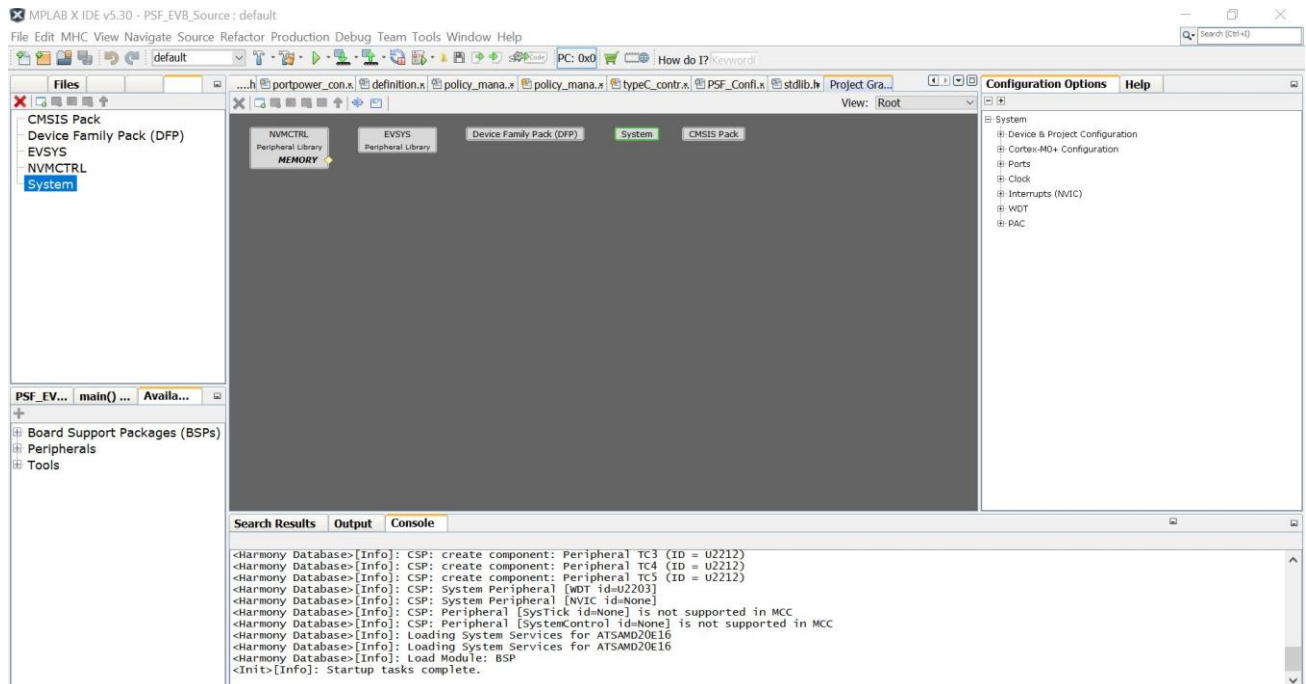


Figure 8.12 Configuration Database Setup

16. Load the Harmony Configuration File by choosing MHC -> Load State. Give the file path as {Local_Path}\PSF\PSF\Demo\PSF_EVB_Source\firmware\src\config\default\SAMD20_PSFHarmonyConfiguration.xml

17. Project Graph will be shown with the peripherals and any changes can be done through *Configuration Options* page

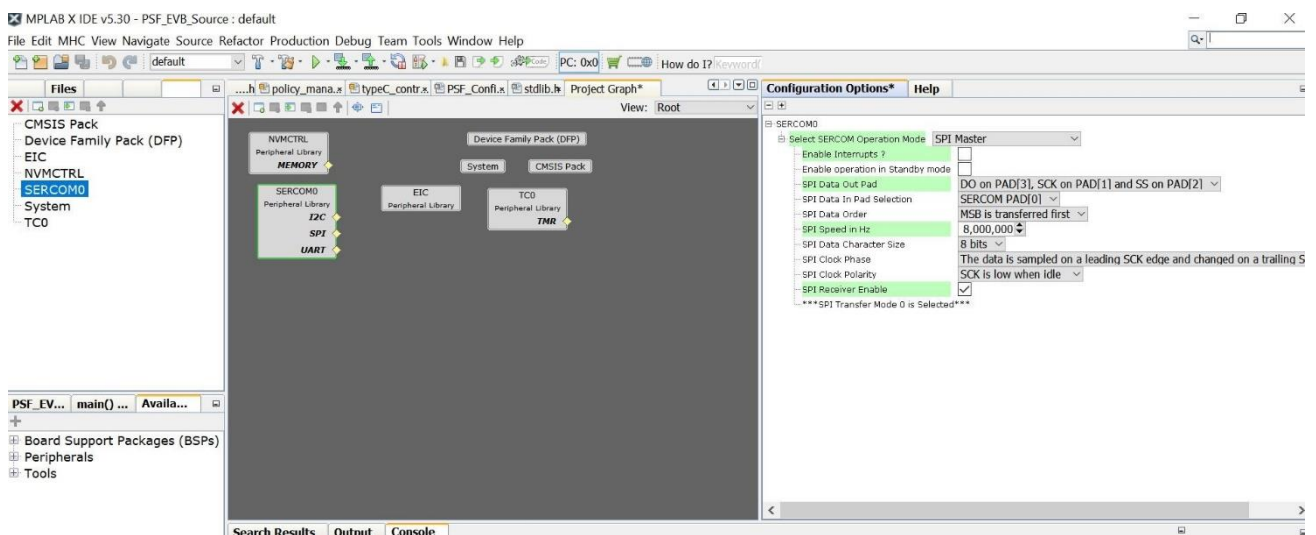


Figure 8.13 Configuration Options Page

Refer MPLAB Harmony Release Notes for any clarifications/settings specific to MHC from <https://www.microchip.com/mplab/mplab-harmony>

For more information on how to use MHC, refer <https://github.com/Microchip-MPLAB-Harmony/mhc/wiki>