

Introduction

PIC32CM devices are a family of 32-bit microcontrollers (MCUs) from Microchip Technology, based on the Arm® Cortex®-M0+ and M23 cores. These devices offer higher performance, more memory and advanced peripherals compared to traditional 8-bit microcontrollers such as the AVR® and PIC16/PIC18 families. For users migrating to a PIC32CM MCU, this document helps ease the migration process by comparing the different tools used in developing AVR, PIC16/PIC18, and PIC32CM applications, with a focus on Microsoft® Visual Studio Code (VS Code®) based tools.

Table of Contents

Introduction.....	1
1. Development Tools.....	3
1.1. Microchip Development Tools.....	3
1.2. Partner Tools.....	3
2. MPLAB Development Ecosystem.....	4
2.1. Setting up the Tools.....	4
2.2. Navigating MPLAB X IDE and VS Code.....	5
2.3. Configuring Compilers.....	6
2.4. Connecting Evaluation Kits.....	7
2.5. Finding Examples in MPLAB Discover.....	8
2.6. Adding Content Through MCC.....	8
2.7. Debug and Program.....	28
3. Development Guides and References.....	29
4. Conclusion.....	30
Microchip Information.....	31
Trademarks.....	31
Legal Notice.....	31
Microchip Devices Code Protection Feature.....	31

1. Development Tools

1.1. Microchip Development Tools

The MPLAB® Development Ecosystem is the primary development environment for all Microchip products, including PIC® and AVR MCUs. This ecosystem can be accessed using the [MPLAB® Tools](#) for VS Code as the main development tool, or through the [MPLAB® X IDE](#). The [MPLAB® XC compilers](#) are the primary compilers used for code development. Hardware tools such as the MPLAB® PICKit™ 5, MPLAB SNAP, and MPLAB ICD 5 provide programming and debugging capabilities. Microchip also offers evaluation kits and development boards, including the Curiosity Nano and Curiosity Pro, to facilitate prototyping and evaluation. Example projects, documentation and tutorials are available in [MPLAB® Discover](#), an online catalog designed to support development.

Legacy tools are still available for AVR (and SAM) microcontrollers, including [Microchip Studio](#) (formerly Atmel Studio), Atmel START, the AVR-GCC toolchain, and hardware programmers/debuggers such as the AVR Dragon, JTAGICE mkII, STK500, and Atmel-ICE. Although Device Family Packs (DFPs) are still supported in Microchip Studio, new features introduced in recent product families may not function properly. Therefore, this platform is no longer recommended for new designs.

Note: For users who continue to develop AVR projects in Microchip Studio, the MPLAB Tools for VS Code provide a more natural transition than migrating to the MPLAB X IDE.

The table below provides a summary of the development tools described above for AVR, PIC16/PIC18F, and PIC32CM microcontrollers.

Table 1-1. Summary of Tools for AVR, PIC16, PIC18, and PIC32CM Development

Tool Type	AVR	PIC16/PIC18	PIC32CM
IDE	MPLAB® Tools for VS Code, MPLAB® X IDE, Microchip Studio	MPLAB® Tools for VS Code, MPLAB® X IDE	MPLAB® Tools for VS Code, MPLAB® X IDE
Compiler/Toolchain	XC8, AVR-GCC	XC8	XC32, ARM-GCC
Programmer/Debugger	MPLAB® PICKit™, SNAP, Atmel-ICE, AVR Dragon	MPLAB® PICKit™, MPLAB SNAP, MPLAB ICD	MPLAB® PICKit™, MPLAB SNAP, MPLAB ICD
Evaluation Boards	Xplained Mini/Pro, Curiosity Nano, Arduino boards	Curiosity Nano	Curiosity Nano/Pro

1.2. Partner Tools

Third-party development tools are also available for AVR and PIC MCUs, offering support for a wide range of applications. Below is an overview of the available third-party development tools:

- [Development Tool Partners for AVR and PIC](#)

Below is a comprehensive list of third-party partners for SAM and PIC32 MCUs:

- [Third-Party Partners for SAM and PIC32 MCUs](#)

2. MPLAB Development Ecosystem

The MPLAB Development Ecosystem is part of Microchip's larger [MCU Development Tool Ecosystem](#), which supports customers through all stages of embedded design, from discovery to production. It consists of an IDE, supported compilers, code configurators, programmers/debuggers, and other integrated tools. This ecosystem is also supported in VS Code through the MPLAB Tools. This section provides a comparison of the project development experience for AVR, PIC16/PIC18F, and PIC32CM devices using the MPLAB tool ecosystem.

2.1. Setting up the Tools

This section explains how to set up the MPLAB tools. For new projects, it is recommended to use the MPLAB Tools for VS Code.

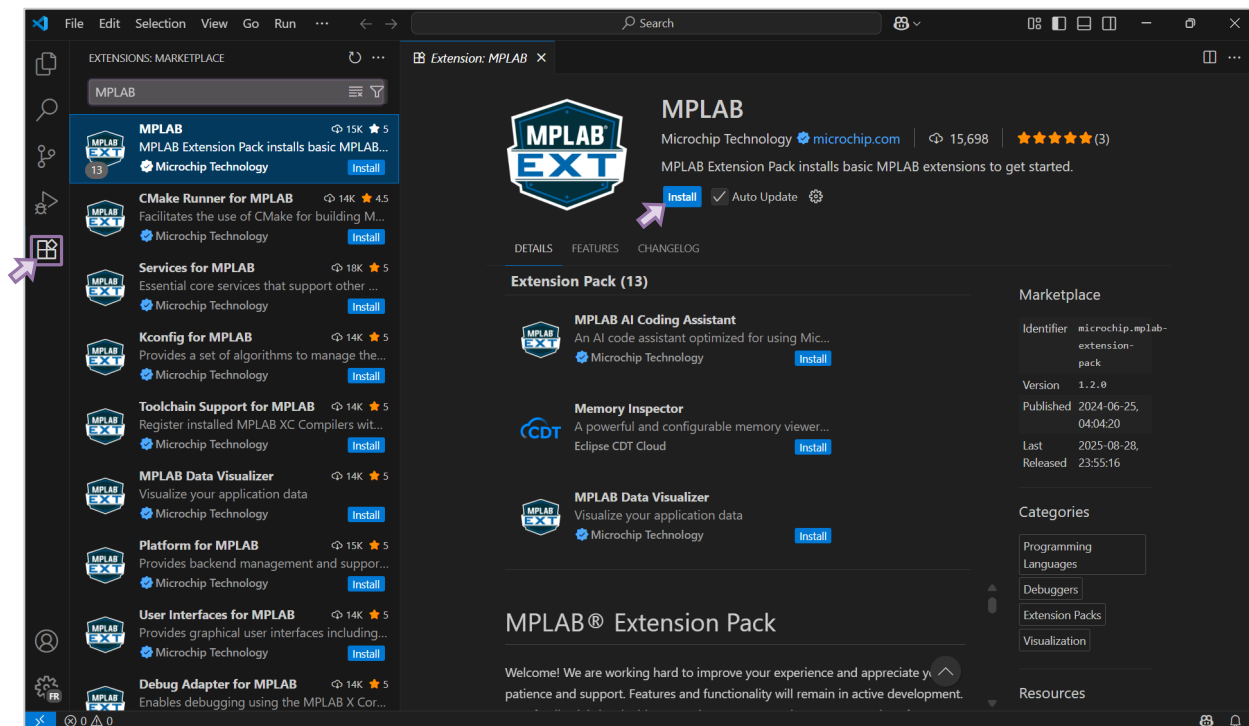
2.1.1. MPLAB® X IDE

The MPLAB X IDE installer and user guides can be downloaded from the official [MPLAB X IDE](#) website, which also provides links to other MPLAB development tools. The installation process is straightforward. The [MPLAB® X IDE user guide](#) includes installation requirements and tutorials on how to use the IDE and its plug-in tools.

2.1.2. MPLAB Tools for VS Code

The MPLAB Tools for VS Code can be downloaded from the official [MPLAB Tools](#) website or directly from the [VS Code Marketplace](#). Both resources provide a complete list of MPLAB extensions, which can be installed individually or together as part of the MPLAB Extensions Pack. Installation is performed through the VS Code Extensions interface. Tutorials are available on the official website, including links to the [Extensions Developer Help](#) and the [MPLAB Tools for VS Code Video playlist](#).

Figure 2-1. MPLAB Tools within the VS Code Extensions Marketplace



2.2. Navigating MPLAB X IDE and VS Code

For users new to the VS Code environment, there are some key differences in navigating the tools. MPLAB X features a dedicated toolbar (see the figure below) for commonly used functions such as setting up projects, building code, and launching other plugins. In contrast, VS Code uses a command palette (see the second figure) to execute these functions. Additionally, VS Code places common tool functions in various locations within the extension windows, which become visible when hovered over. Regarding the navigation, editors, and output windows, VS code arranges these elements in a similar orientation to MPLAB X.

Figure 2-2. MPLAB X IDE Windows

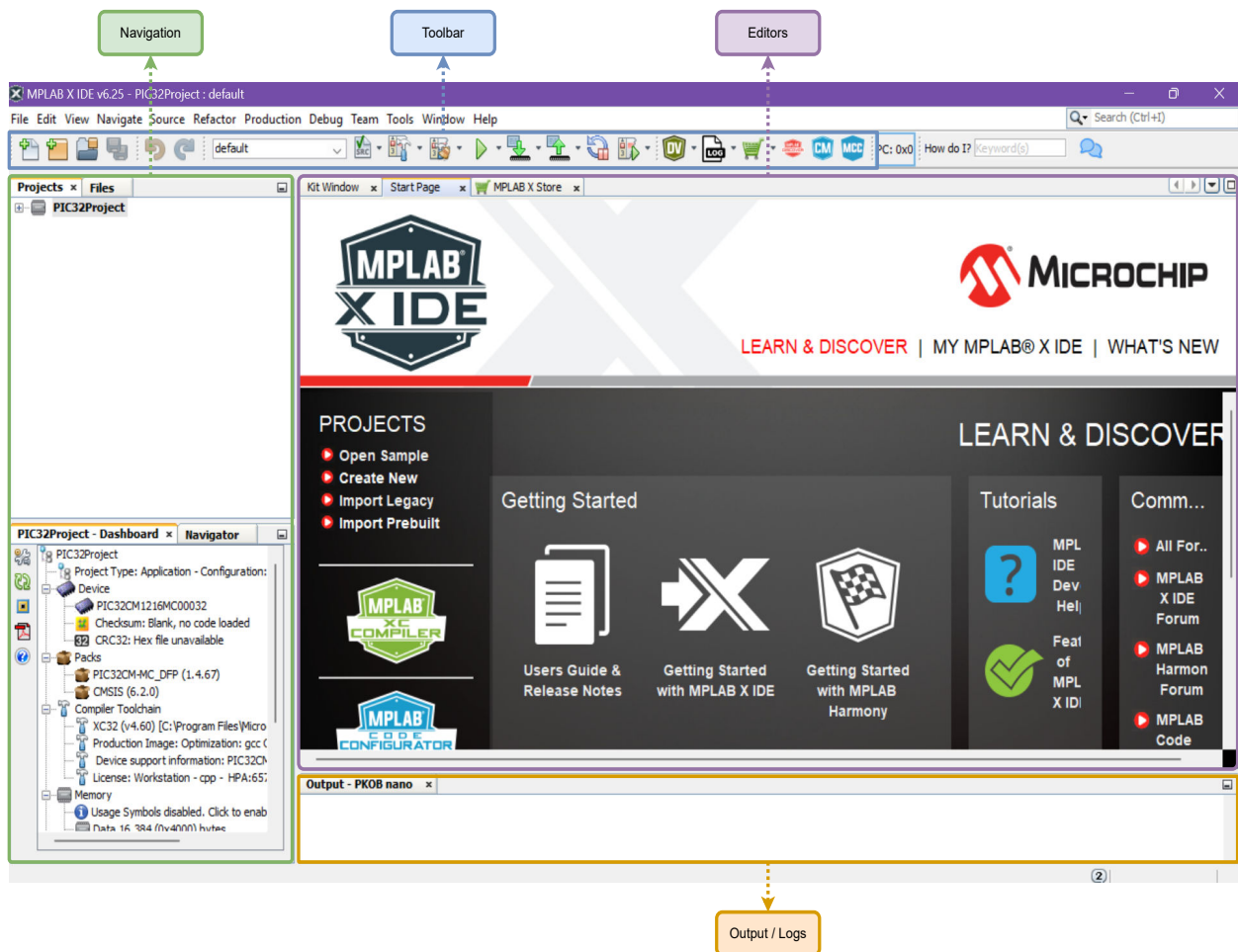
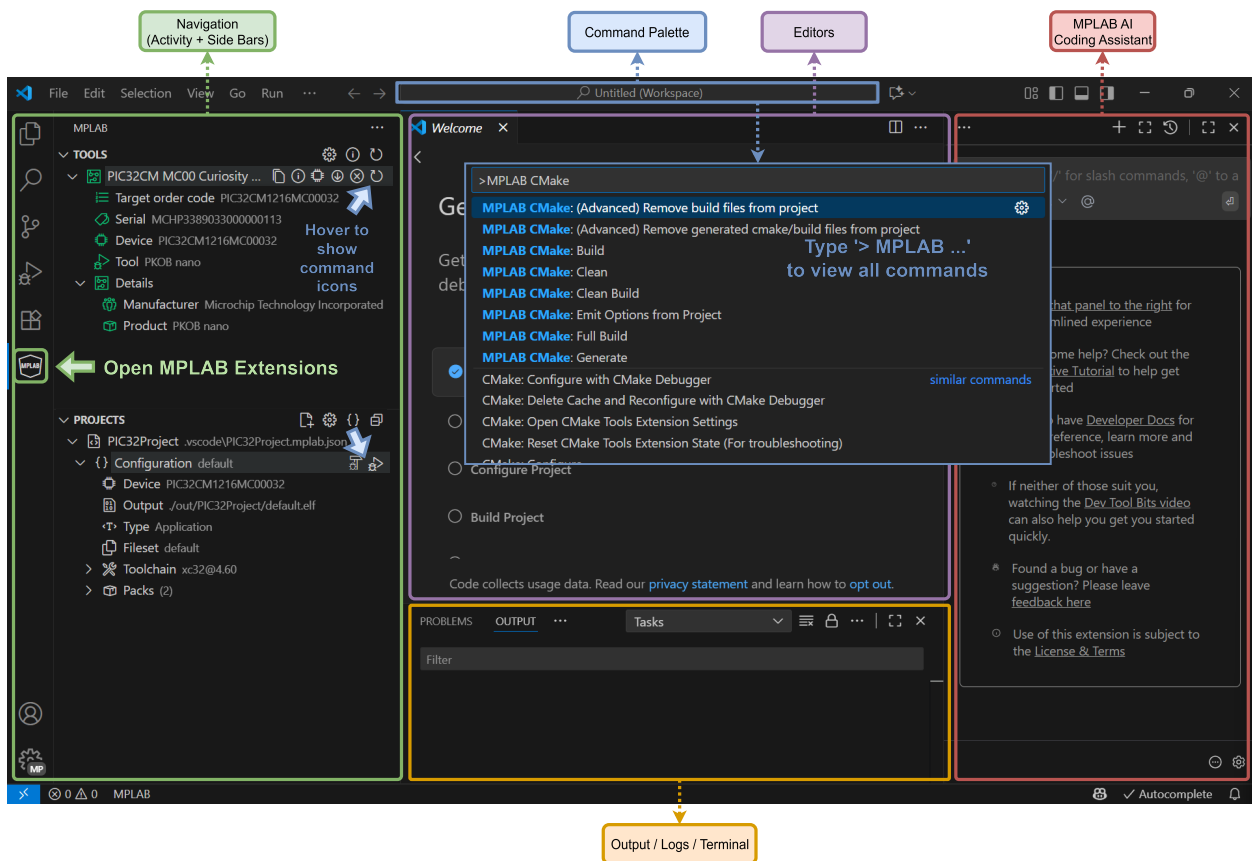


Figure 2-3. MPLAB Tools for VS Code Windows

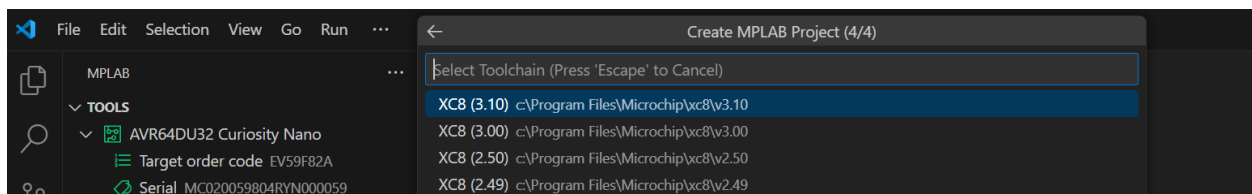


2.3. Configuring Compilers

When creating new projects or modifying existing ones, 8-bit microcontrollers such as AVR and PIC16/PIC18F use the XC8 compiler. AVR devices also support GCC compilers. PIC32CM devices, which are 32-bit microcontrollers, use the XC32 compiler and ARM-GCC. Links to the compiler pages are provided below:

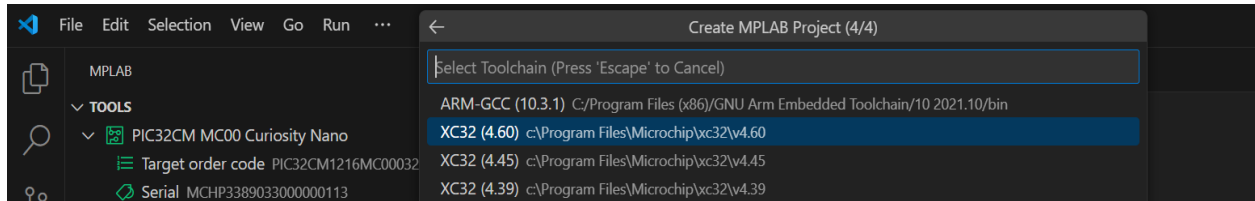
- [XC8 Compilers](#) and [GCC Compilers](#)

Figure 2-4. XC8 Compiler Selection in MPLAB Tools for VS Code



- [XC32 Compilers](#)

Figure 2-5. XC32 Compiler Selection in MPLAB Tools for VS Code



2.4. Connecting Evaluation Kits

When an evaluation kit is connected, a kit information appears, providing a description of the kit along with links to user guides, example projects from MPLAB Discover, reference designs, and schematics. In the MPLAB Tools for VS Code, the Kit Window can be accessed from the Tools information icon.

Figure 2-6. AVR DU Curiosity Nano Kit Window in MPLAB Tools for VS Code

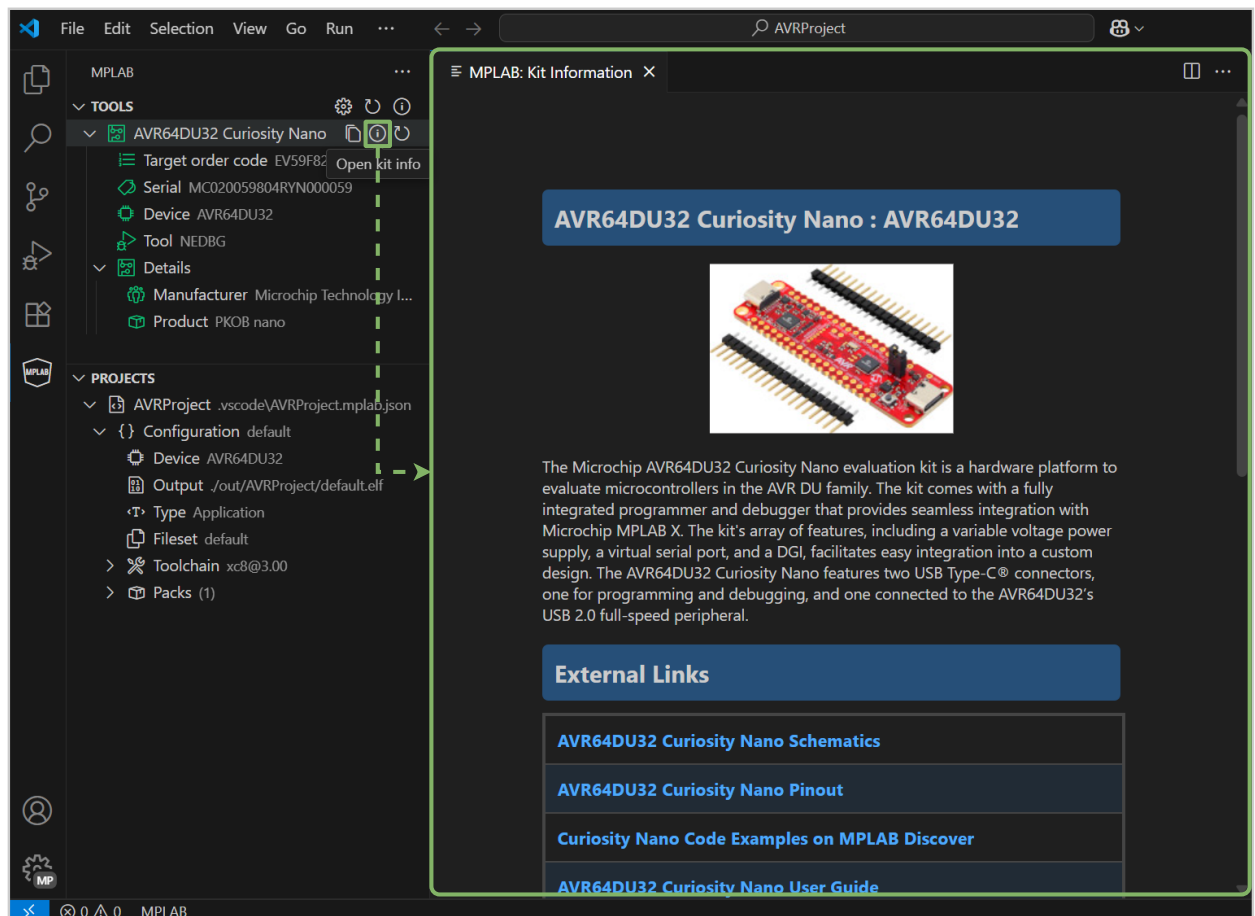
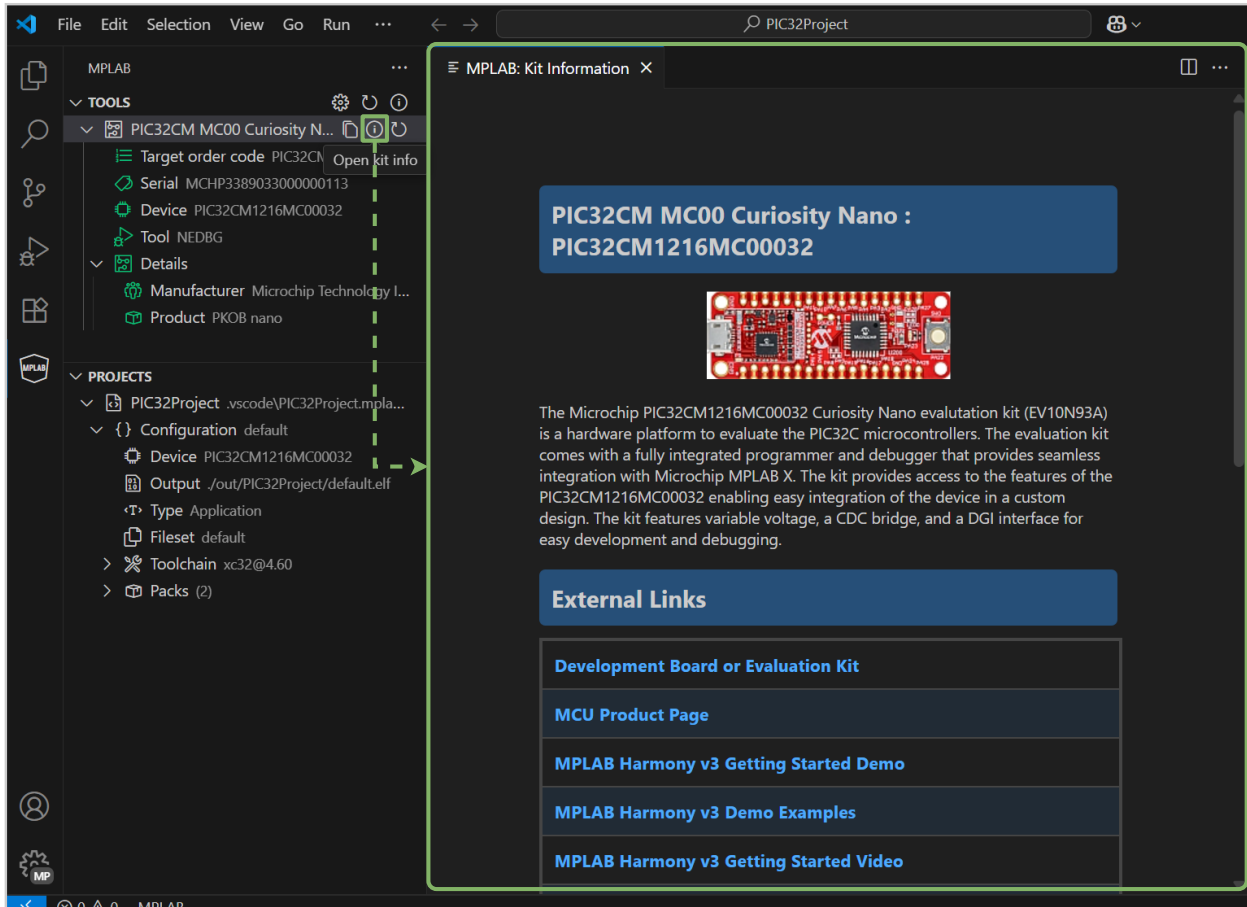


Figure 2-7. PIC32CM MC Curiosity Nano Kit Window in MPLAB Extensions for VS Code

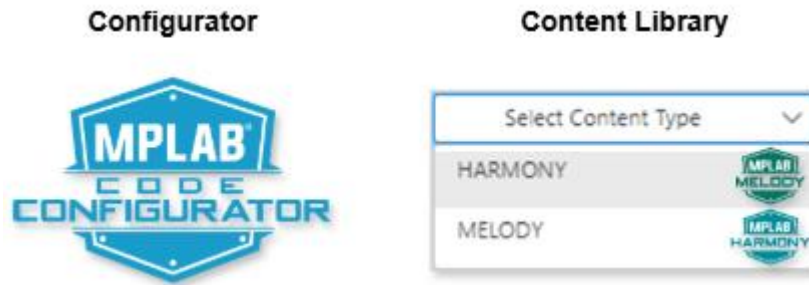
2.5. Finding Examples in MPLAB Discover

Developers can find code examples in [MPLAB Discover](#) either by clicking on the links provided in the evaluation kit's window or by manually searching the website. MPLAB Discover also offers additional resources, including documentation such as application notes, data sheets, reference designs, and information on compatible evaluation boards. Examples of MPLAB Discover content for the AVR DU and PIC32CM MC devices are provided below.

- [MPLAB Discover Examples for AVR DU Curiosity Nano](#)
- [MPLAB Discover Examples for PIC32CM MC Curiosity Nano](#)

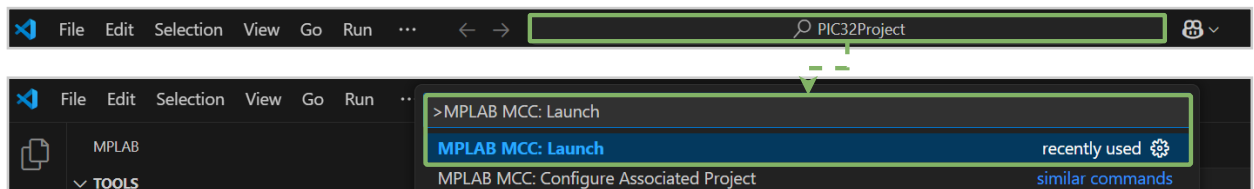
2.6. Adding Content Through MCC

MPLAB® Code Configurator (MCC) is a free graphical programming environment that generates seamless, easy-to-understand C code. MCC included content libraries that support various microcontroller families:

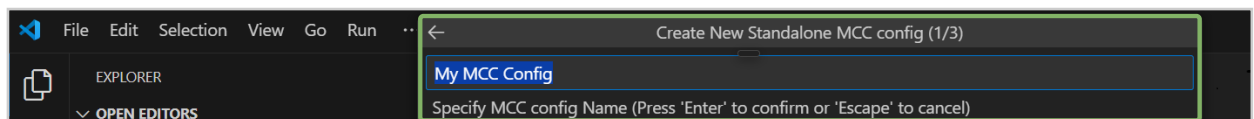
Figure 2-8. MPLAB® Code Configurator Content Libraries

- [MCC Melody](#) - Supports 8-bit PIC and AVR MCUs, and dsPIC® DSCs
 - [MCC Melody API Reference and Supported Devices](#)
- [MCC Harmony](#) - Supports all 32-bit MIPS® and Arm® Cortex®-based MCU and MPU devices
 - [MPLAB Harmony Supported Devices](#)

Launch MCC by clicking the MCC icon on the MPLAB X IDE toolbar or by typing `>MPLAB MCC : Launch` in the VS Code command palette.

Figure 2-9. Launching MCC from MPLAB® X IDE**Figure 2-10.** Launching MCC from MPLAB Tools for VS Code

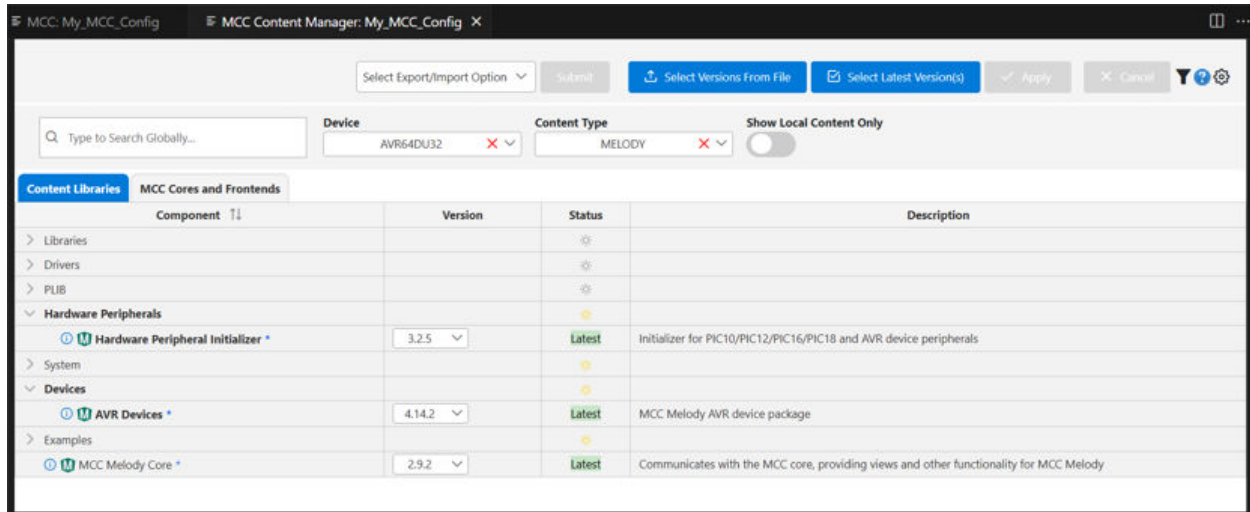
With MPLAB Tools for VS Code, users can also specify a name for the MCC Configuration. By default, MCC uses `My_MCC_Config` as the configuration name.

Figure 2-11. New MCC Configuration in MPLAB Tools for VS Code

2.6.1. MCC Content Manager

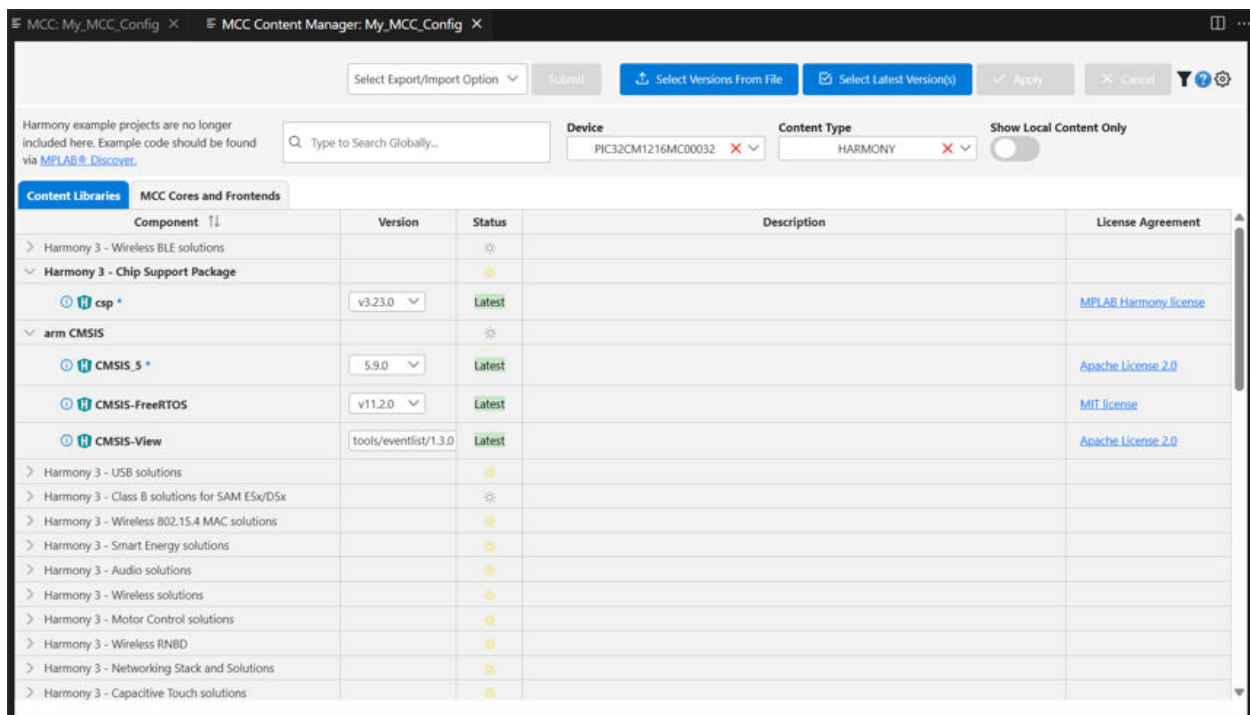
The [MCC Content Manager](#) allows users to download versions of supported drivers and libraries for a selected device. The Content Manager for MCC Melody organizes its content into categories such as Libraries, Drivers, System, PLIBs, and Examples.

Figure 2-12. Content Manager for MCC Melody



- The Content Manager for MCC Harmony displays most of its content upfront and categorizes similar items by solutions.

Figure 2-13. Content Manager for MCC Harmony



Notes:

- MPLAB® X IDE users can open the Content Manager before launching MCC by clicking the CM icon on the IDE toolbar.
- For VS Code users, the MCC Content Manager can only be accessed from within the MCC Resources Panel (refer to the next section).
- In MPLAB® X IDE, the MCC Content Manager cannot be opened while MCC is running, as it may result in unexpected or corrupted content.

2.6.2. MCC Windows

MCC Melody and Harmony offer similar views and arrangements in both MPLAB X IDE and the MPLAB Tools for VS Code. Introductions to their graphical user interfaces (GUIs) are listed below:

- [MCC Melody GUI](#)
- [MCC Harmony GUI](#)

While Melody features an *Application Builder* that provides a visual representation of added content, Harmony uses a *Project Graph* for the same purpose. Both the Application Builder and Project Graph display which driver and library modules have been added to a project and illustrate how these modules interact with each other. Additionally, both tools automatically add default system modules that are essential for every MCC project.

A key difference between Melody and Harmony is how certain modules are added. In MCC Melody, all modules open from the Project Resources into the Composer Panel. For Harmony, most modules also open from the Project Resources into the Tree View Configuration. However, some Harmony contents that uses other GUIs other than the standard tree view is added through plug-ins that open in separate windows.

- MCC Melody features an Application Builder in the middle panel, which is shared with the Pin Package view. The Pin Package and Pin Grid views automatically open at start-up. The Pin Grid view displays only the pin rows for the added modules. Pin Settings can be accessed from the Project Resource panel and show only the selected pins for the added modules.

Figure 2-14. MCC Melody Window

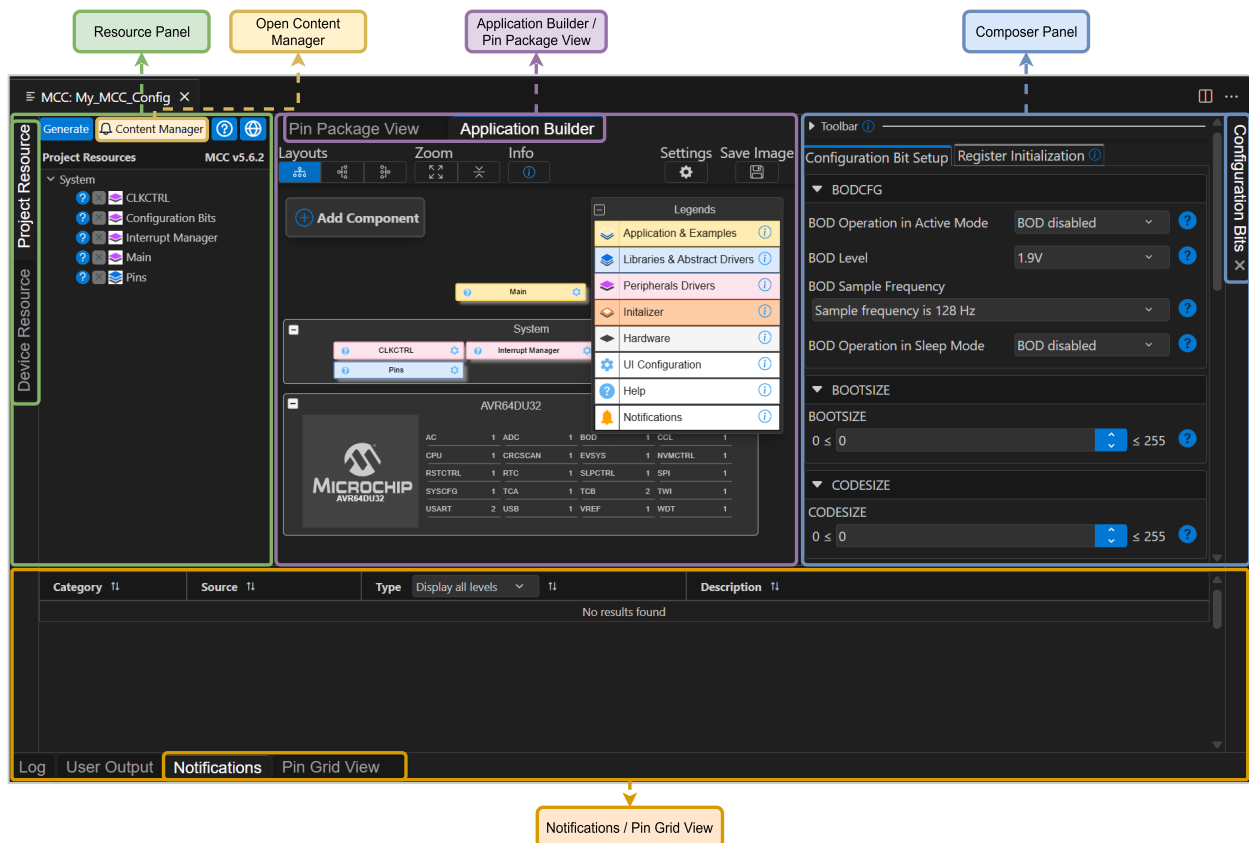


Figure 2-15. MCC Melody Pin Configurations

The screenshot displays the MCC Melody Pin Configurations interface. The top panel shows the 'Pin Package View' for the QFP32USB package, which includes a pin diagram of the AVR64DU32 microcontroller. The bottom panel shows the 'Pin Grid View' table, which lists pin configurations for various modules and functions.

Package:	QFP32U	Pin No:	30	31	32	1	2	3	4	5	9	10	11	12	13	14	15	16	17	20	21	22	23	24	25	26	27			
Module	Function	Direction																												
CLKCTRL	CLKOUT	output																												
Pins	GPIO	input																												
Pins	GPIO	output																												

- MCC Harmony features a Project Graph in the middle panel. The Pin Configurations - including the Diagram (package), Table (grid), and Settings - manually opened through plug-ins. The Pin Settings and Table views display all pin configurations for all available modules.

Figure 2-16. MCC Harmony Window

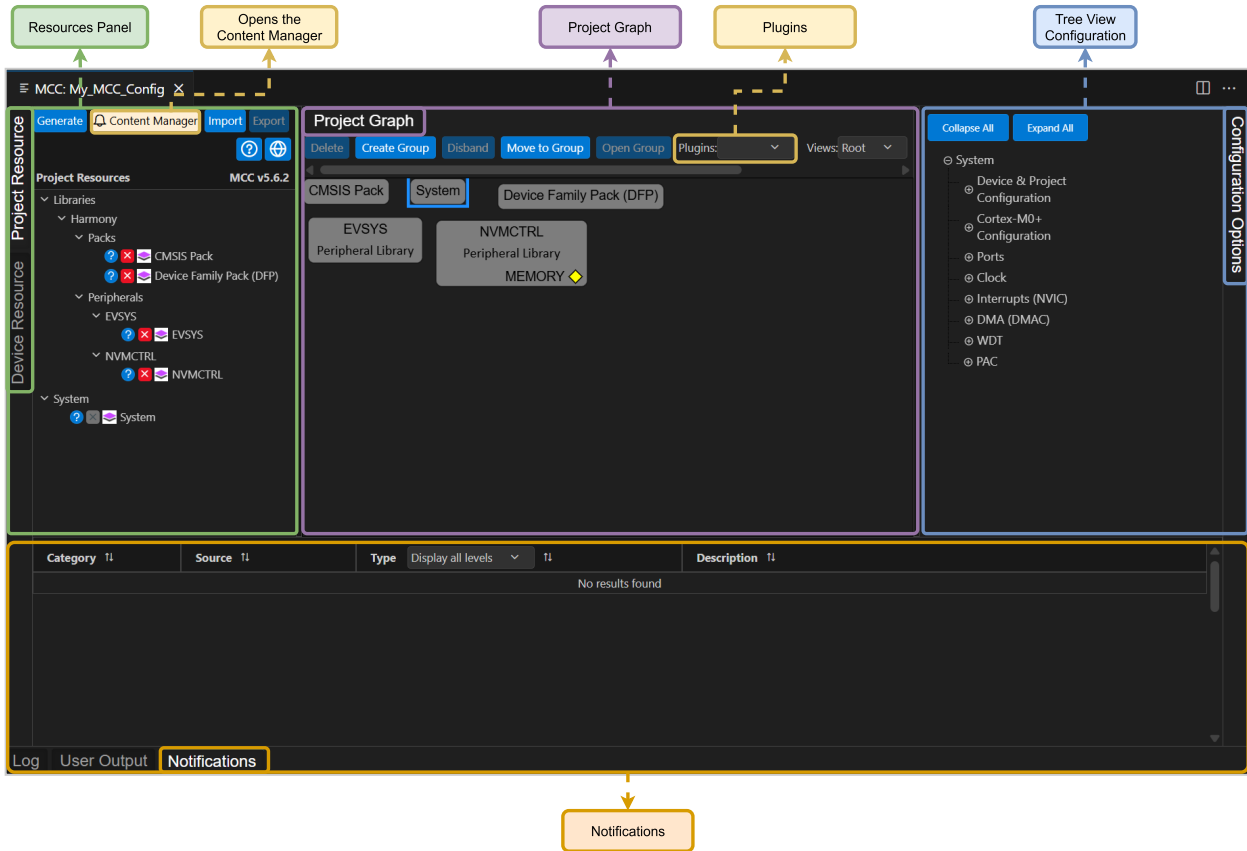
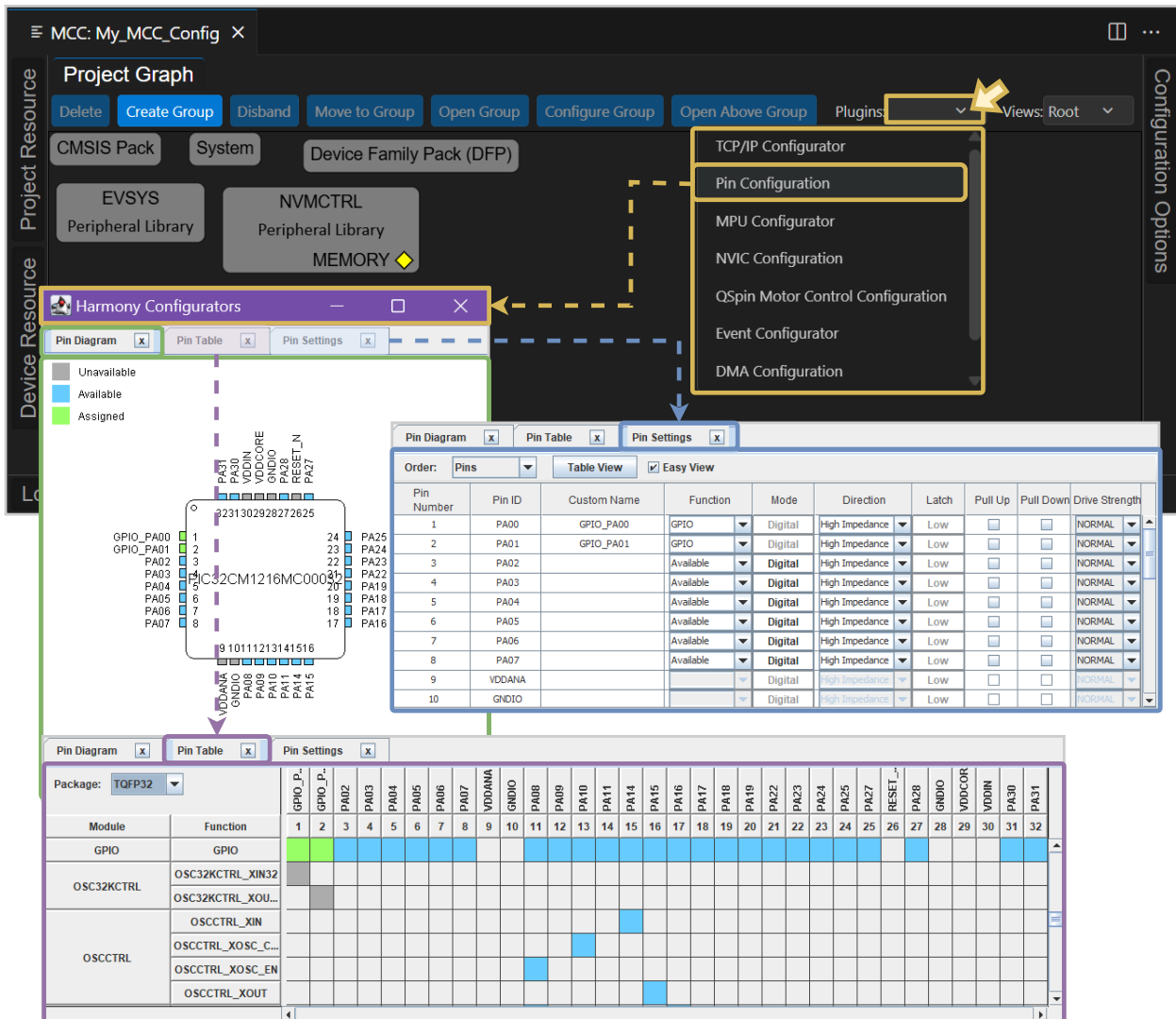


Figure 2-17. MCC Harmony Pin Configurations

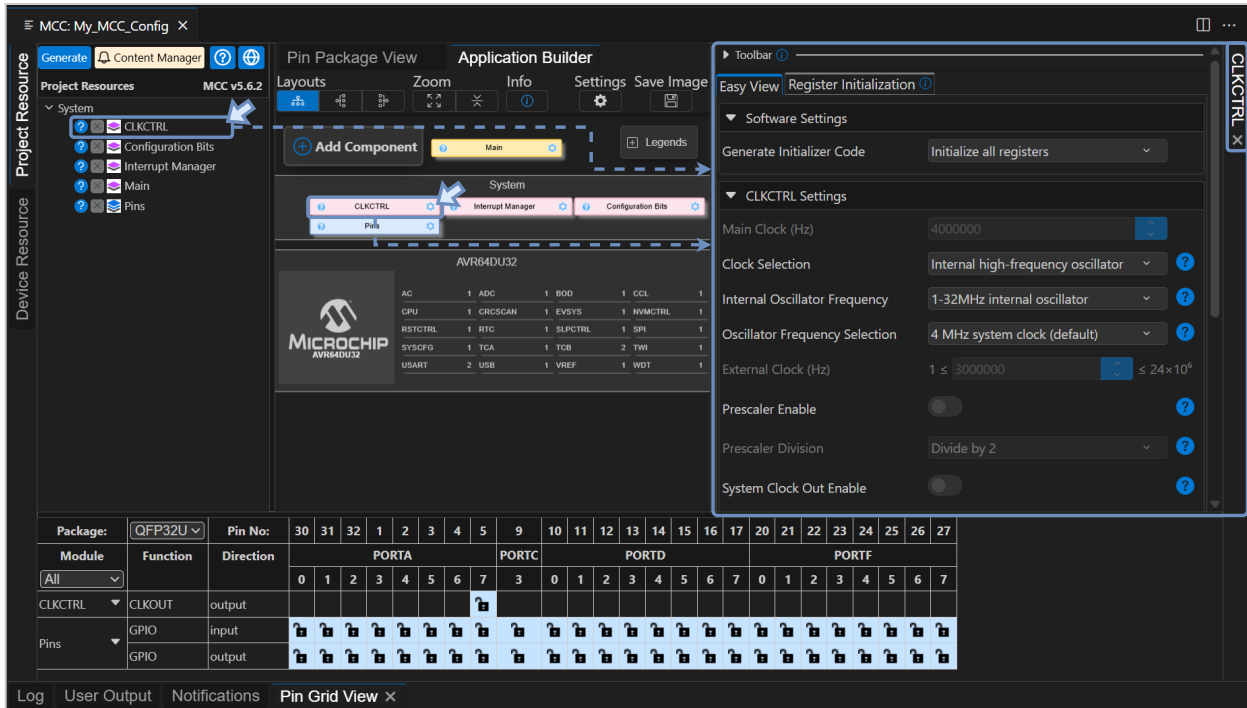


2.6.3. Clock Configurations

Configuring the MCU clock is one of the first steps in any project development. The main clock frequency determines system operation and impacts processing speed, power consumption, peripheral functionality, and overall system stability. The Clock Configuration process in MCC Melody and Harmony differs, primarily due to the more complex clock system found in PIC32 devices.

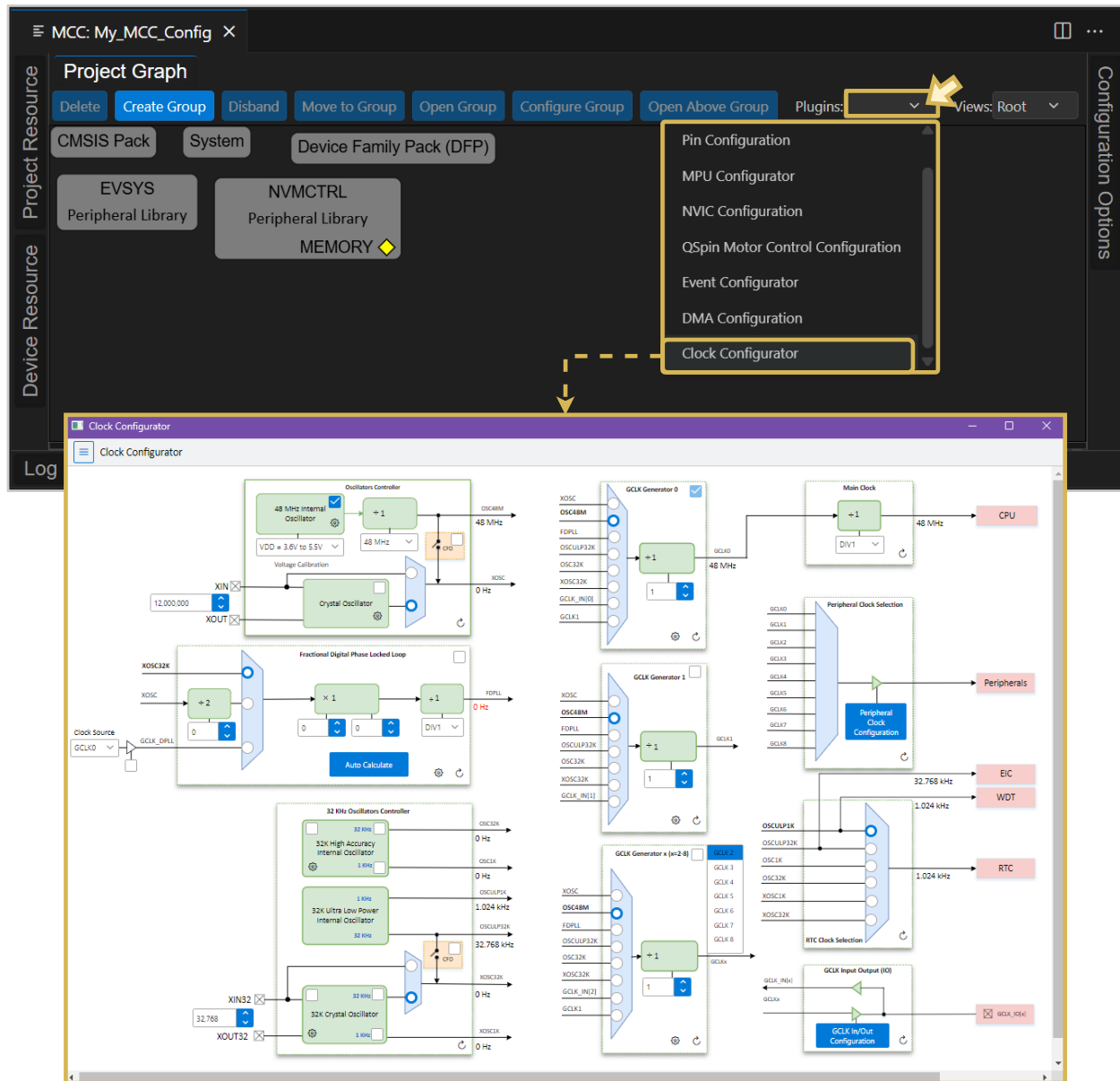
- MCC Melody Clock Control settings can be accessed from the Project Resource panel or the Application Builder and use the standard Melody user interface

Figure 2-18. MCC Melody Clock Control



- The MCC Harmony Clock Configurator uses a plug-in that graphically depicts the PIC32 clock system. It allows users to easily interconnect the various clock components and quickly configure individual blocks to achieve the desired system and peripheral clock frequencies.

Figure 2-19. MCC Harmony Clock Configurator

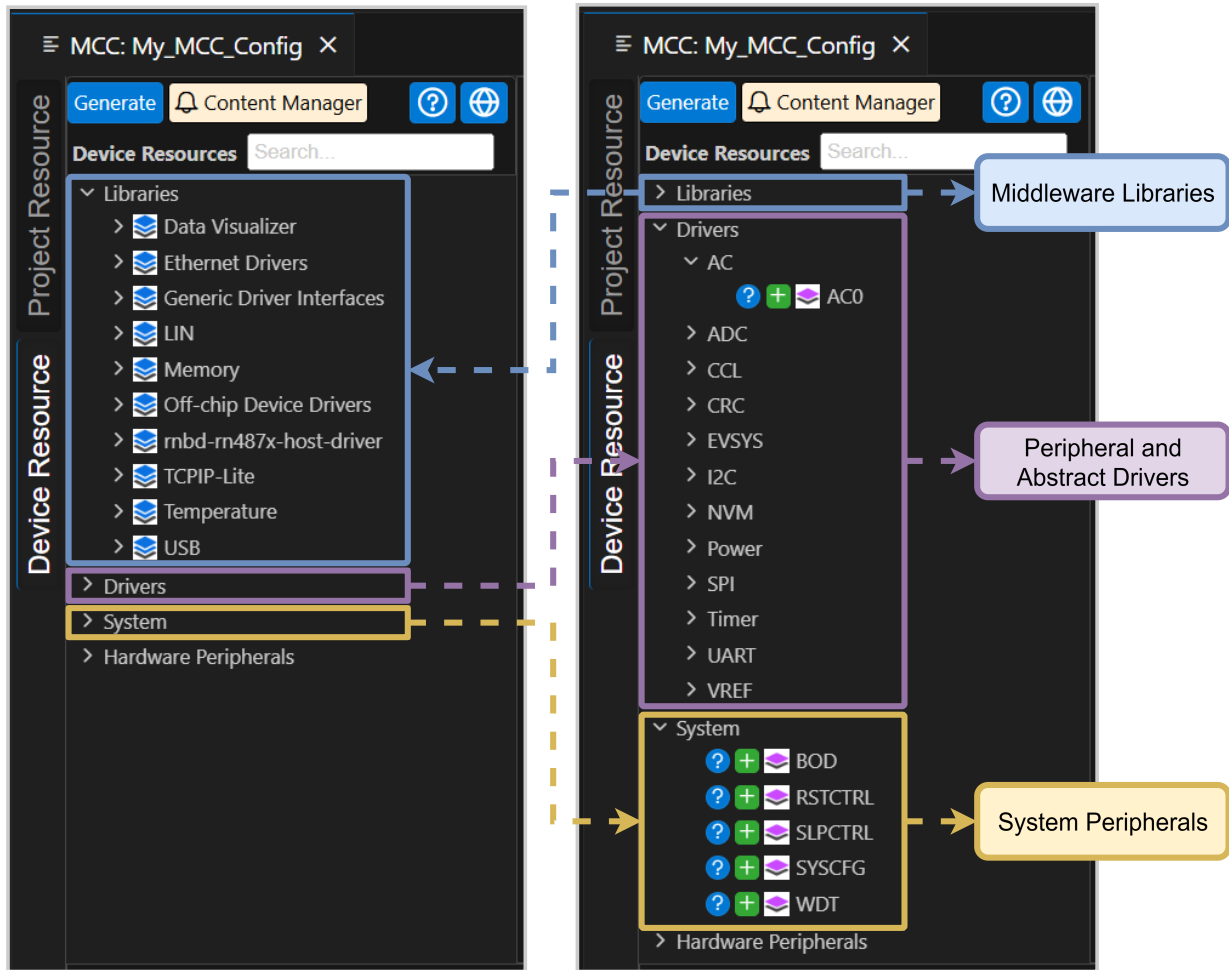


2.6.4. Drivers and Libraries

MCC displays its supported drivers and libraries in the Device Resources window.

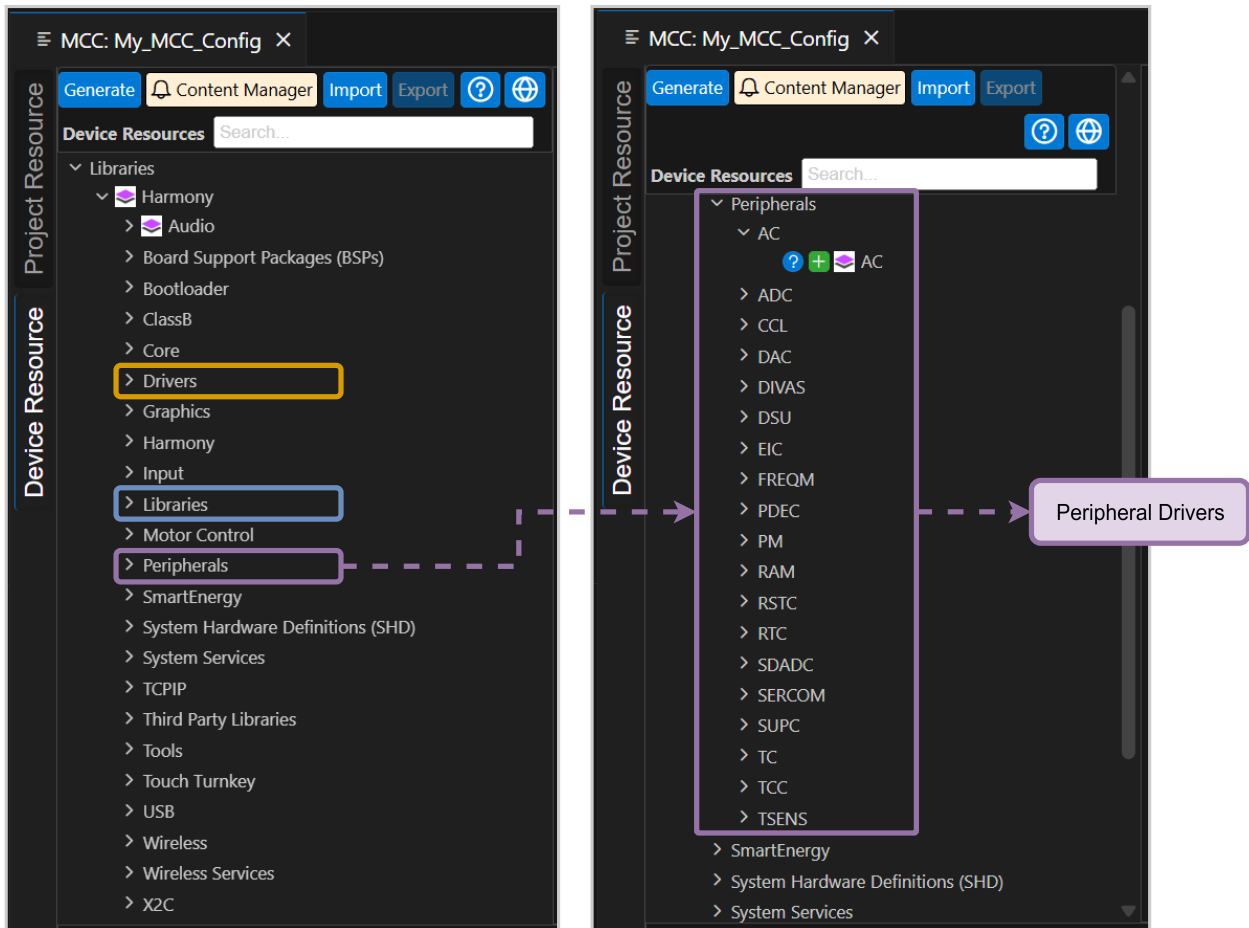
- MCC Melody uses the same categories as those found in Content Manager. Peripheral drivers are listed under *Drivers*, middleware libraries are under *Libraries*, and additional system management peripherals are under *System*. *Hardware Peripherals* contain peripheral register initializations and only generate an `Initialize()` function.

Figure 2-20. MCC Melody Device Resources



- MCC Harmony has minor differences compared to MCC Melody. In addition to *Drivers* and *Libraries*, peripheral drivers are listed under the *Peripherals* category. Solutions are organized into their own separate categories.

Figure 2-21. MCC Harmony Device Resources



2.6.5. API References

Library and driver API references can be accessed by clicking the Help icons located beside the module name in the Resources window.

Figure 2-22. MCC Melody API Reference

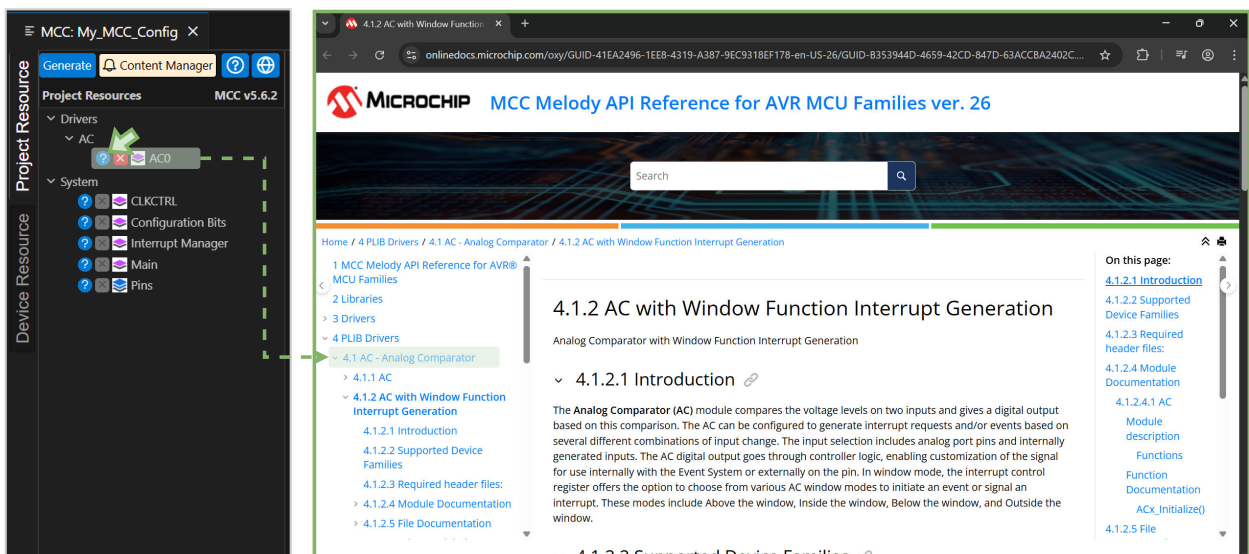
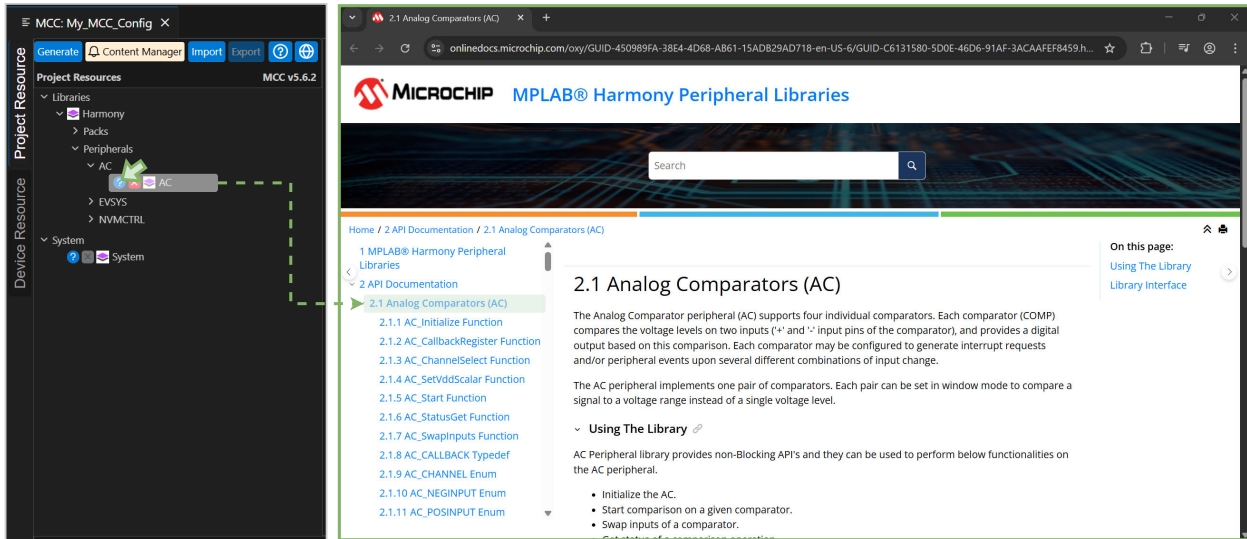


Figure 2-23. MCC Harmony API Reference

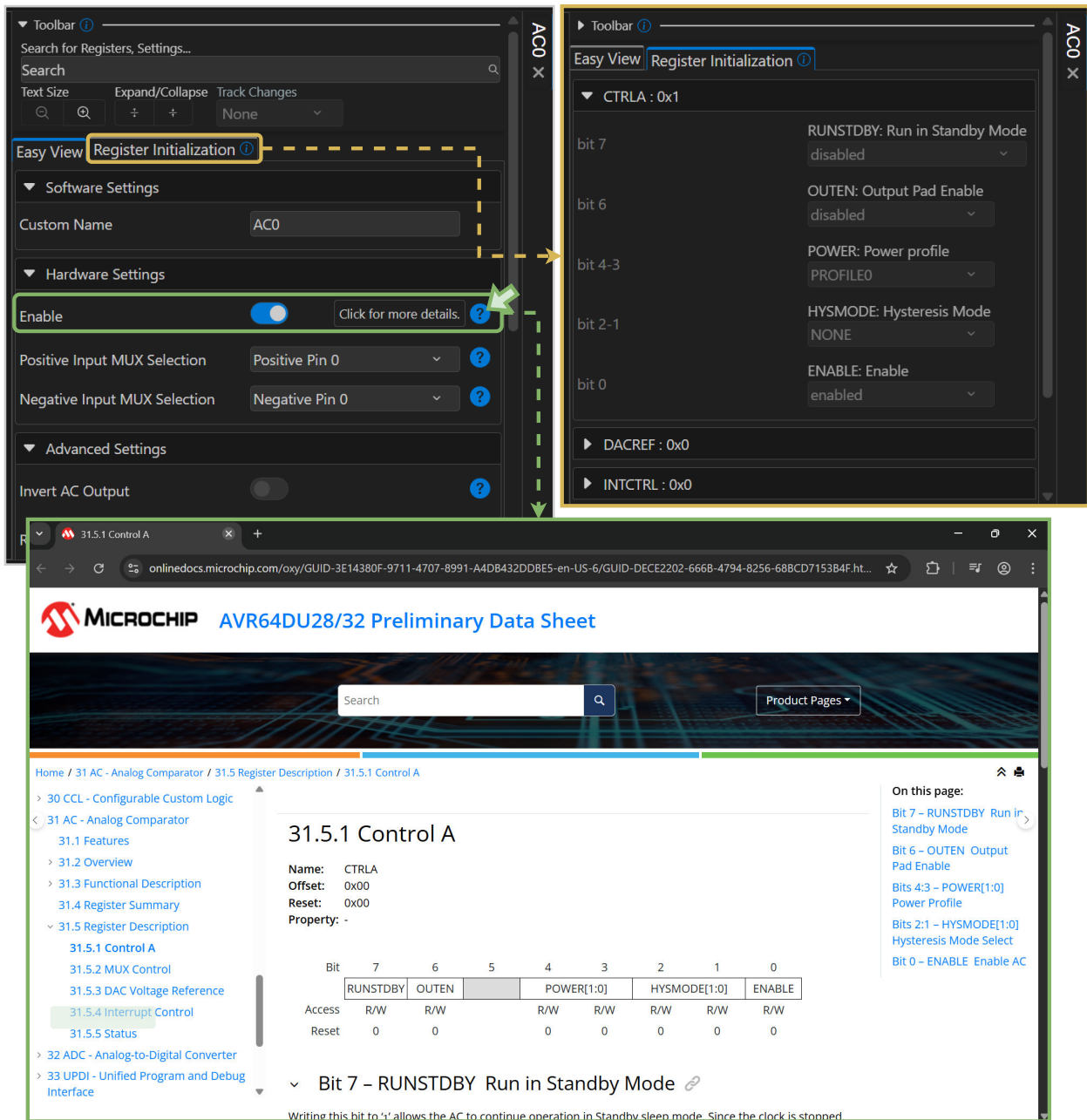


2.6.6. Module Configuration

The composer panels for MCC Melody and Harmony differ in style and structure.

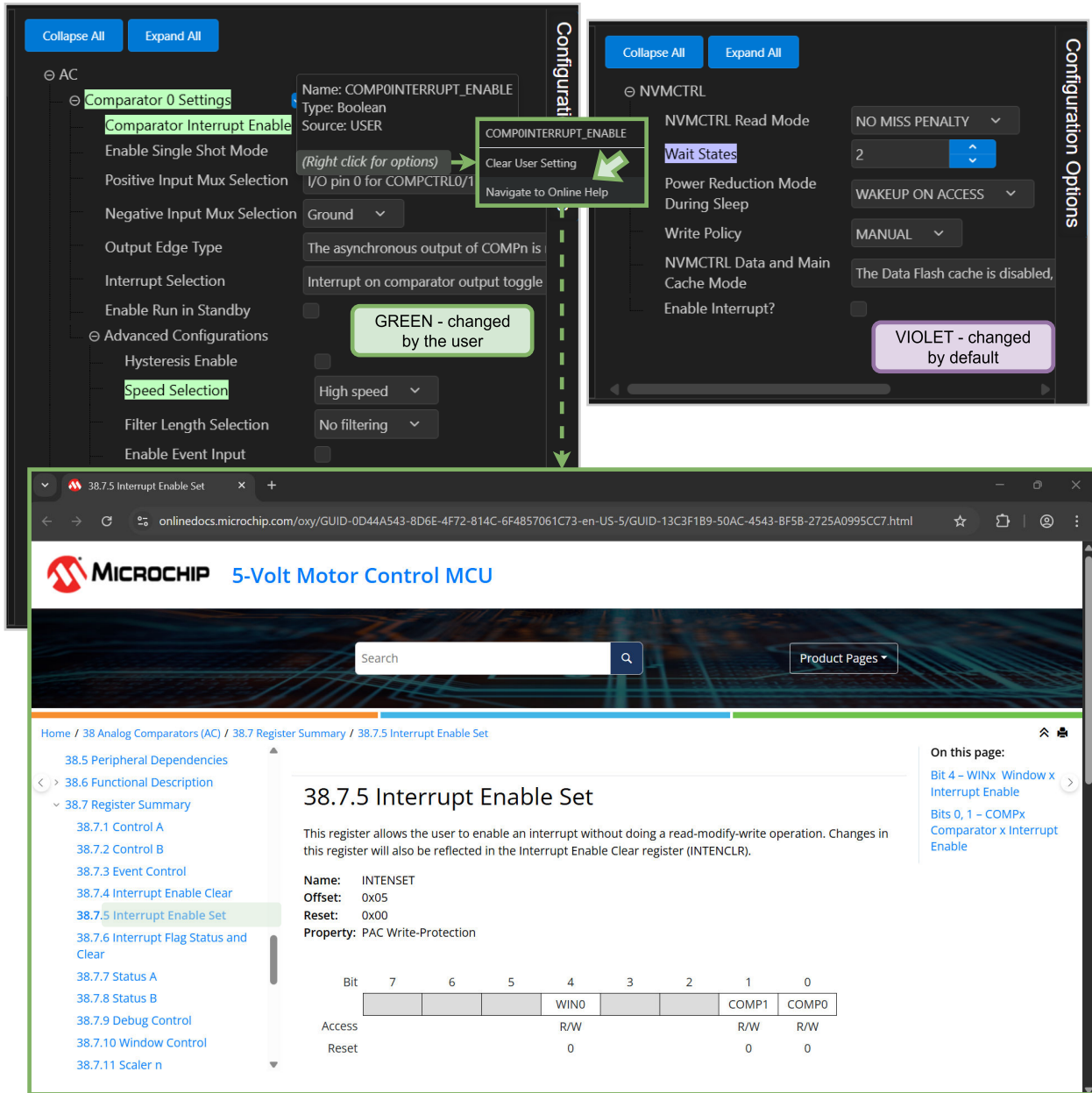
- MCC Melody uses collapsible UI groups with a toolbar that allows users to customize the view or search for keywords. UI fields include Help icons that users can click or hover over for more information. Peripheral drivers also feature Register Initialization views, providing a more accurate representation of the registers and their settings.

Figure 2-24. MCC Melody ACO Configuration



- MCC Harmony uses a collapsible list, with colors indicating which settings have been changed by default or through user configuration. Users can also hover over the UI fields to access additional information.

Figure 2-25. MCC Harmony ACO Configuration

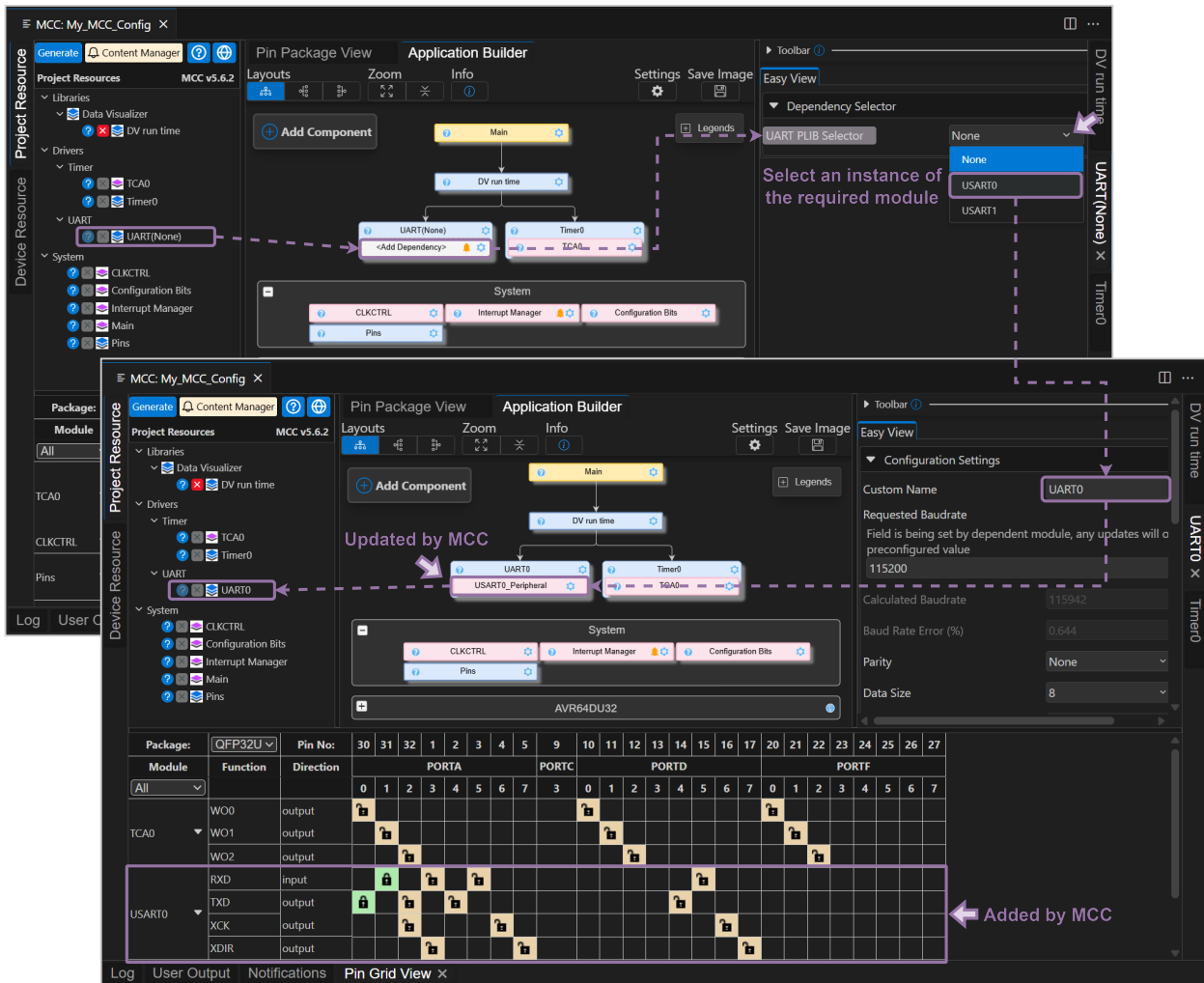


2.6.7. Module Interaction

Both MCC Melody and Harmony display how modules interact with each other.

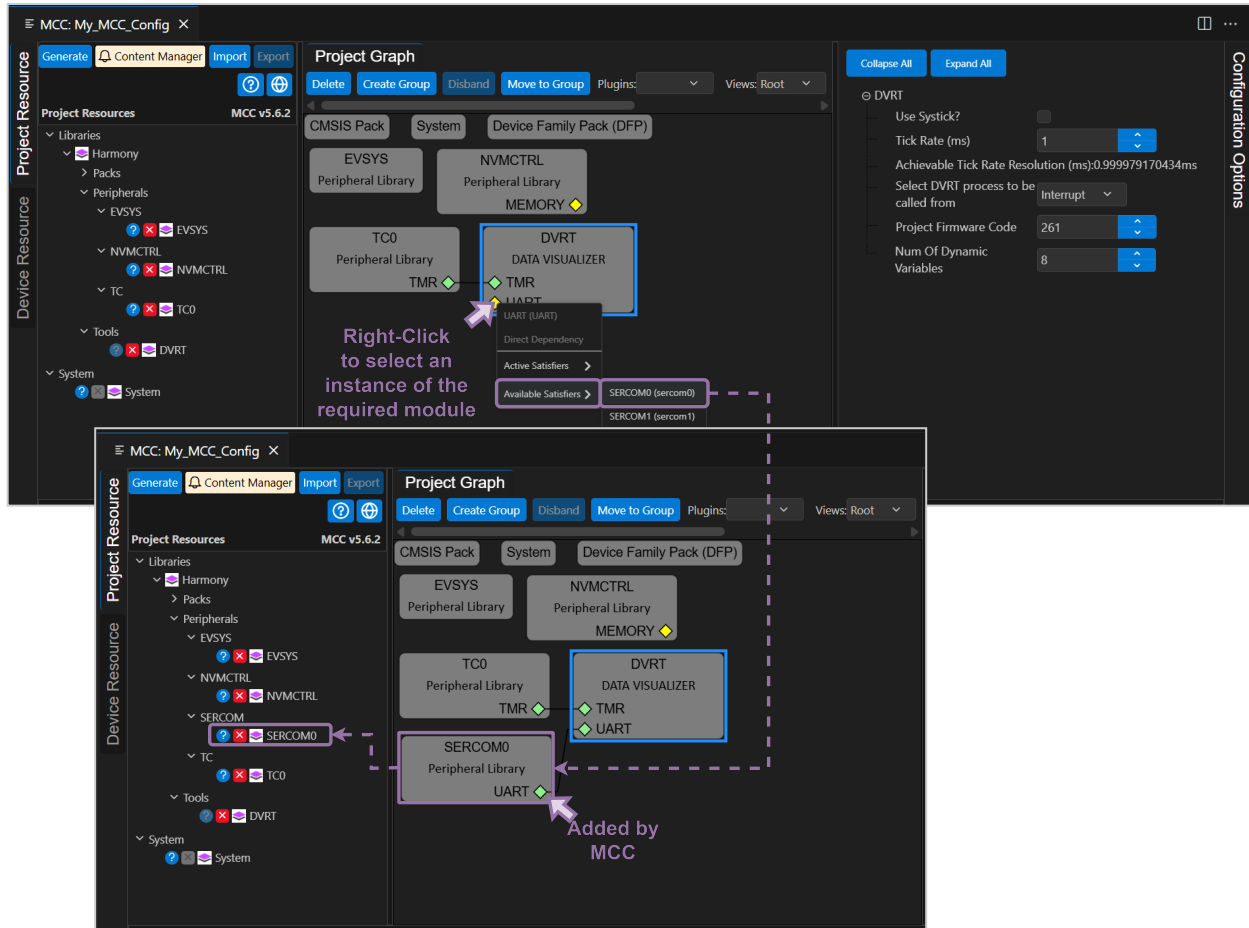
- MCC Melody module dependencies are configured in the module's Easy View. Once a module instance is selected, it is also displayed in the *Application Builder*. Dependencies with only one instance are automatically added to the project.

Figure 2-26. MCC Melody Dependency Selection



- MCC Harmony module dependencies are configured in the Project Graph by right-clicking the required module connections and selecting the appropriate instance (e.g., SERCOM0 for DVRT). The selected module instance is then automatically added to the project.

Figure 2-27. MCC Harmony Dependency Selection



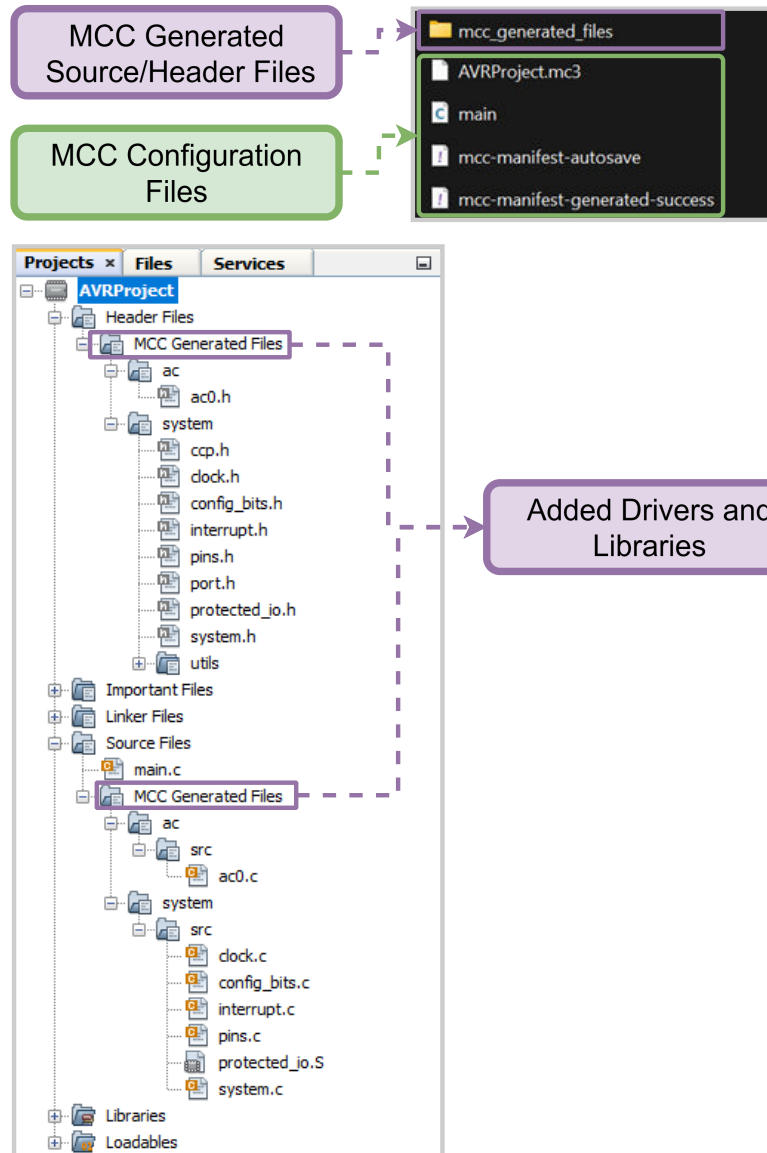
2.6.8. Code Generation

The generated file and folder structure varies between MCC Melody and Harmony, and also differs between MPLAB X IDE and MPLAB Tools for VS Code.

MPLAB X IDE

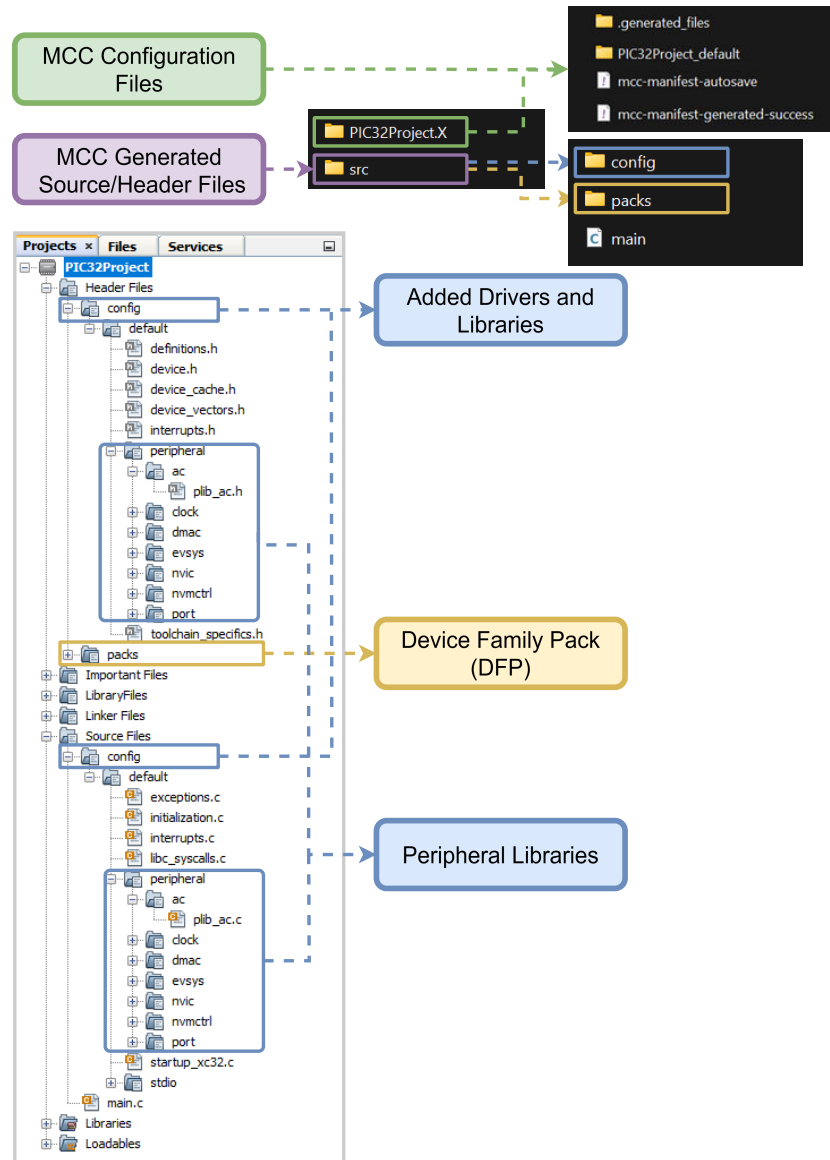
- MCC Melody generates source and header files inside a folder named `mcc_generated_files`, located within the project folder

Figure 2-28. MCC Melody Files in MPLAB X IDE

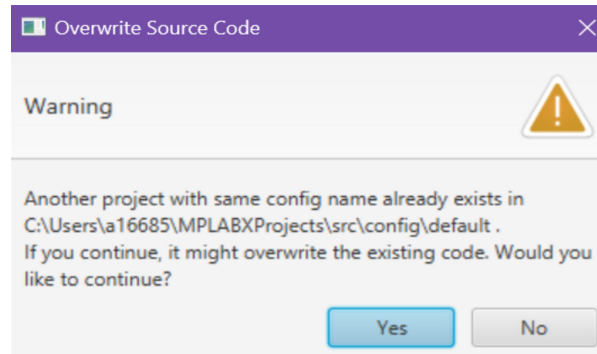


- MCC Harmony generates source and header files inside a folder named `src`, which is located outside of the main project folder

Figure 2-29. MCC Harmony Files in MPLAB X IDE



Note: MCC Harmony warns users if it detects an existing `src` folder outside the project folder. If this `src` folder belongs to a different Harmony project, move the current project folder into a different directory before generating code.

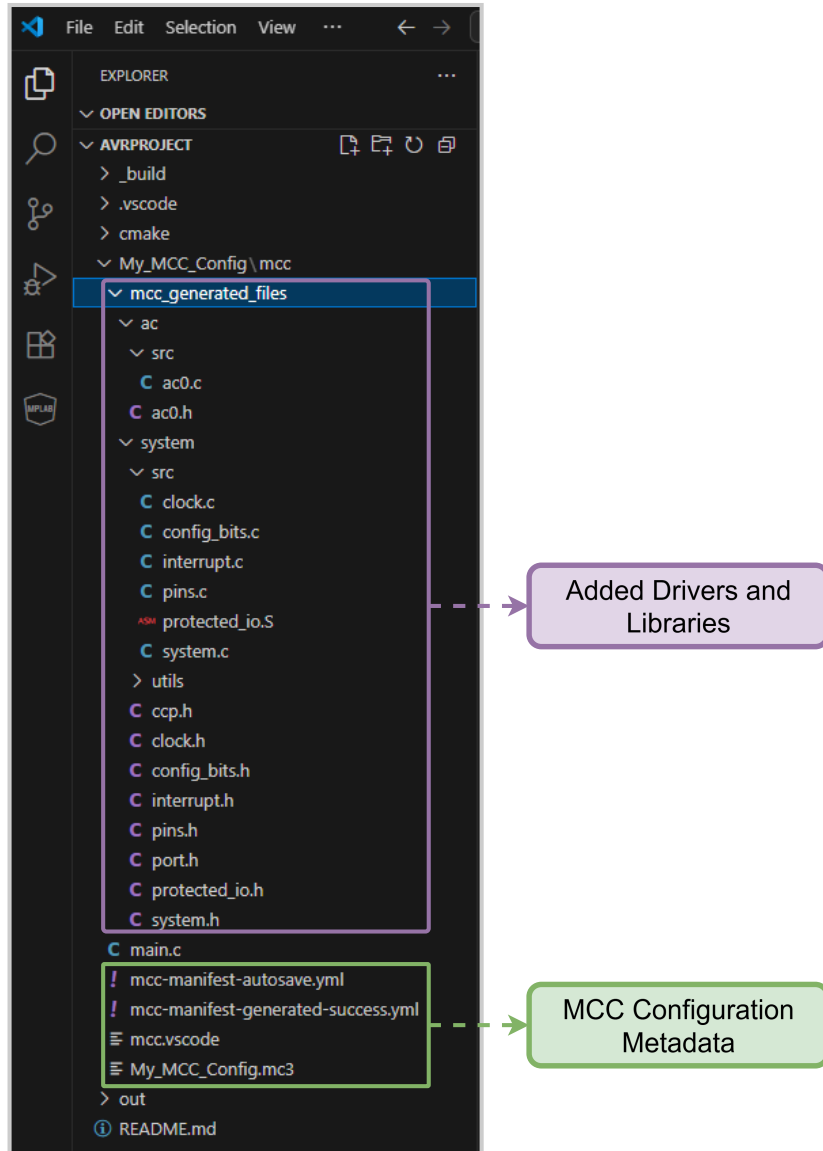
Figure 2-30. MCC Harmony Warning for an existing *src* folder

MPLAB Tools for VS Code

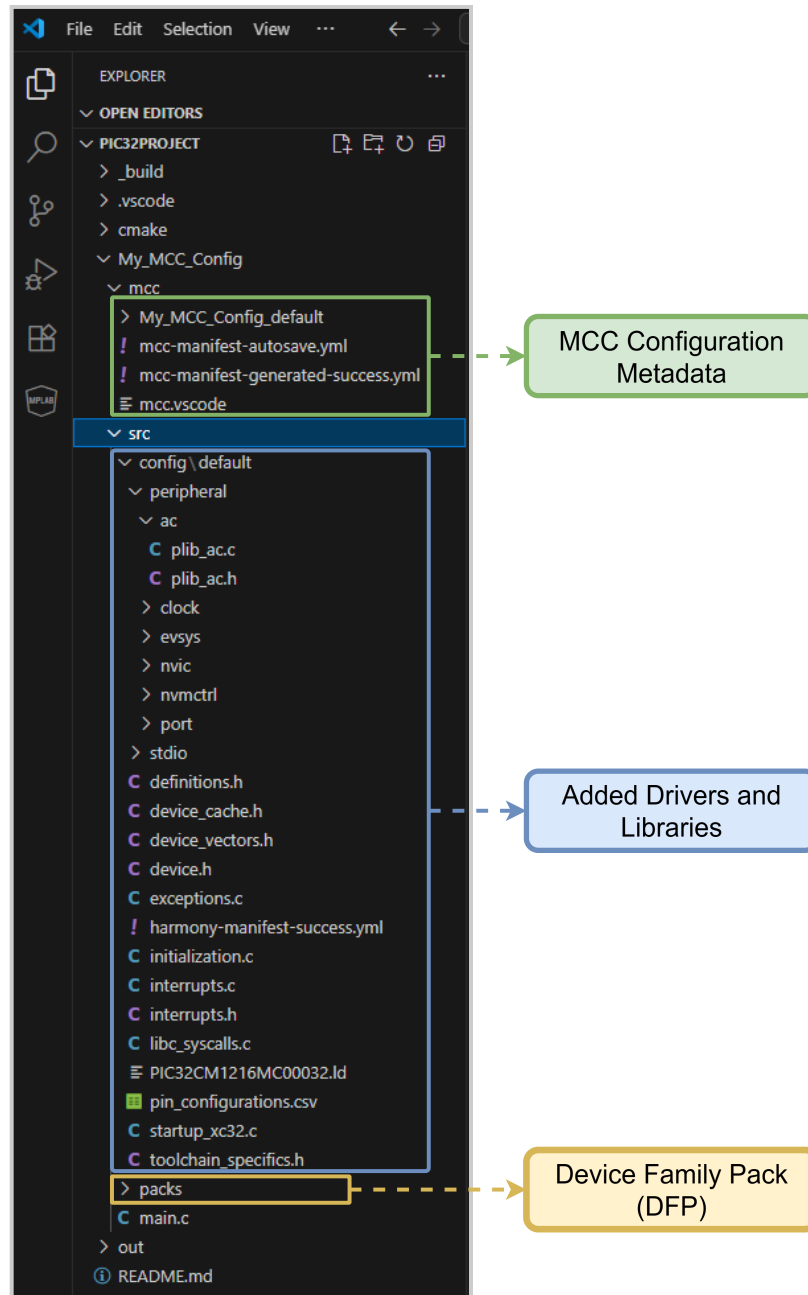
Inside the project folder, MCC generates a folder named after the MCC configuration (e.g., `My_MCC_Config`), which contains another folder named `mcc`. The structure of the contents within the MCC configuration folder differs between MCC Melody and MCC Harmony.

- MCC Melody generates both the `mcc_generated_files` folder and the MCC configuration metadata inside the `mcc` folder

Figure 2-31. MCC Melody Files in MPLAB Tools for VS Code



- MCC Harmony generates the `src` folder inside the MCC configuration folder (e.g., `My_MCC_Config`), and places the configuration metadata inside the `mcc` folder

Figure 2-32. MCC Harmony Files in MPLAB Tools for VS Code

2.7. Debug and Program

All AVR, PIC16/PIC18, and PIC32CM MCUs use the same MPLAB tools for debugging and programming, with no significant differences in tool usage. Below are references to explore the available tools:

- [Debug](#)
- [Evaluation-boards](#)
- [Qualify](#)
- [Production](#)

3. Development Guides and References

Examples, guides and tutorials are available to help get started with PIC32CM development using MPLAB Harmony:

1. [GitHub repository for pic32cm demos](#)
2. [How to Set up the Tools Required to Get Started with MPLAB® Harmony v3 and MCC](#)
3. [Videos on MPLAB Harmony v3 development](#)
4. Arm® Cortex®-Based Microcontrollers (MCUs): www.microchip.com/en-us/products/microcontrollers/32-bit-mcus/pic32-sam
5. 32-bit Microcontroller Collateral and Solutions Reference: ww1.microchip.com/downloads/aemDocuments/documents/MCU32/ProductDocuments/ReferenceManuals/32-bit-Microcontroller-Collateral-and-Solutions-Reference-Guide-DS70005534.pdf

4. Conclusion

Microchip offers a comprehensive MCU Development Tool Ecosystem through MPLAB, widely used for developing applications with AVR, PIC16/PIC18F, and PIC32CM devices. This ecosystem is also available as MPLAB Tools for VS Code, which is the recommended environment for new project development. While most tool functionalities remain consistent across device families, variations exist in the code configurators: MPLAB Melody and MPLAB Harmony.

This document provides a comparison of their interfaces and features to assist users in migrating to applications using PIC32CM MCUs. It also highlights the available user guides and references that help users get started with these development tools.

Microchip Information

Trademarks

The “Microchip” name and logo, the “M” logo, and other names, logos, and brands are registered and unregistered trademarks of Microchip Technology Incorporated or its affiliates and/or subsidiaries in the United States and/or other countries (“Microchip Trademarks”). Information regarding Microchip Trademarks can be found at <https://www.microchip.com/en-us/about/legal-information/microchip-trademarks>.

ISBN: 979-8-3371-2175-8

Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

THIS INFORMATION IS PROVIDED BY MICROCHIP “AS IS”. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

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 INFORMATION OR ITS USE, 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 THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION.

Use of Microchip devices in life support and/or safety applications is entirely at the buyer’s risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip products are strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is “unbreakable”. Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.