



# MICROCHIP 2019 MASTERS

## MPLAB Harmony 3

### Fundamentals



# Agenda

- 1. MPLAB Harmony 3 Introduction**
- 2. Overview of changes from MPLAB Harmony 2**
- 3. Introduction to the downloader and to the modular library packages**
- 4. Fundamentals of applications, projects, configuration, and peripheral libraries**
  - Lab1: Create a simple “heartbeat” LED application that flashes an LED using peripheral libraries**
- 5. Introduction to drivers, services, and middleware**
  - Lab2: Create an application that enumerates a USB CDC device and demonstrates two-way communication between the USB Device and the USB Host PC**
- 6. Review of development models**

# Agenda

- 1. MPLAB Harmony 3 Introduction**
- 2. Overview of changes from MPLAB Harmony 2**
- 3. Introduction to the downloader and to the modular library packages**
- 4. Fundamentals of applications, projects, configuration, and peripheral libraries**
  - Lab1: Create a simple “heartbeat” LED application that flashes an LED using peripheral libraries**
- 5. Introduction to drivers, services, and middleware**
  - Lab2: Create an application that enumerates a USB CDC device and demonstrates two-way communication between the USB Device and the USB Host PC**
- 6. Interoperability and RTOS support**
- 7. Review of development models**

# Agenda

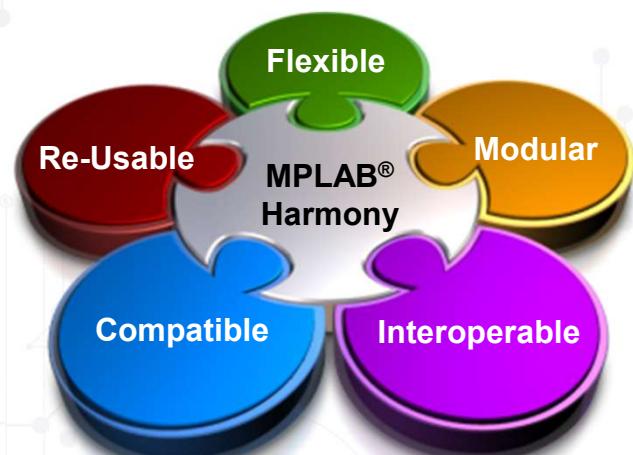
- 1. MPLAB Harmony 3 Introduction**
- 2. Overview of changes from MPLAB Harmony 2**
- 3. Introduction to the downloader and to the modular library packages**
- 4. Fundamentals of applications, projects, configuration, and peripheral libraries**
  - Lab1: Create a simple “heartbeat” LED application that flashes an LED using peripheral libraries
- 5. Introduction to drivers, services, and middleware**
  - Lab2: Create an application that enumerates a USB CDC device and demonstrates two-way communication between the USB Device and the USB Host PC
- 6. Interoperability and RTOS support**
- 7. Review of development models**

# What is MPLAB Harmony 3?

An extension of the MPLAB Ecosystem  
for creating embedded software  
solutions for 32-bit Microchip devices.

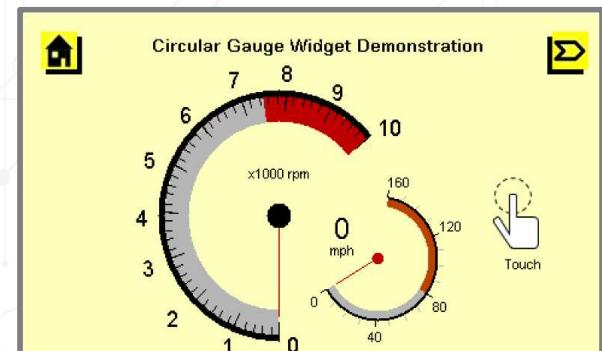
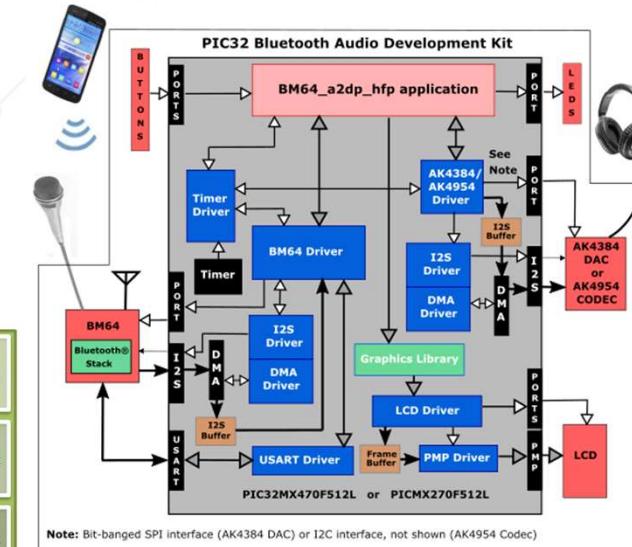
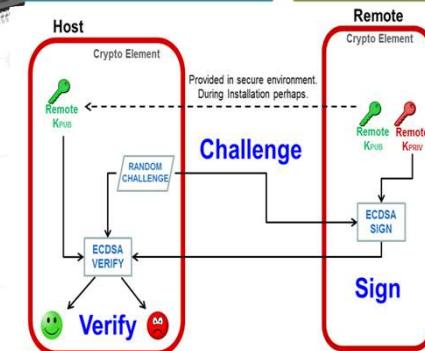
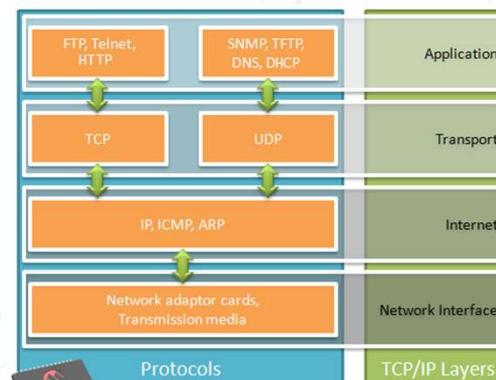
## MPLAB Harmony 3 Provides:

- **Demo/Example Applications**
- **Layered & Modular Libraries**
  - Peripheral Libraries
  - Drivers & Services
  - Middleware
- **Graphical Developer Tools**
  - Deploying, configuring, and generating the libraries.
- **An embedded software reference model.**



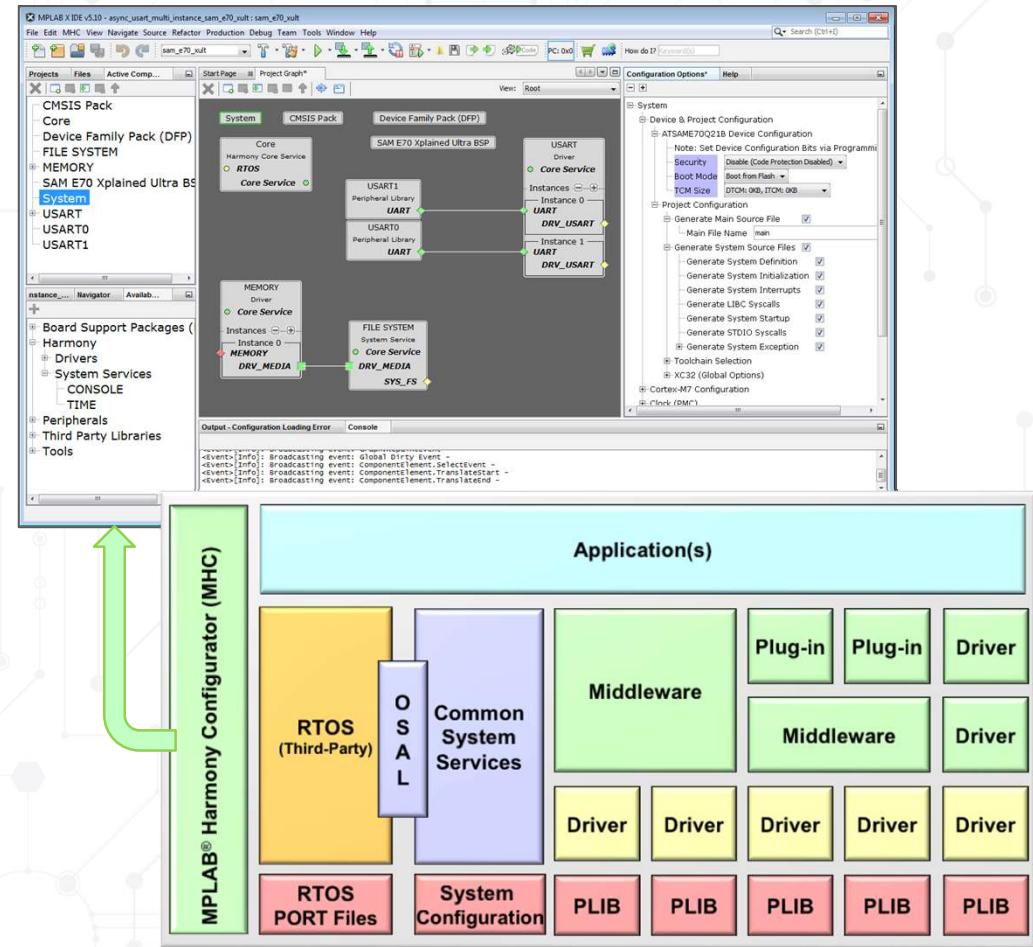
# Demo Applications

- Audio
- Bluetooth
- Bootloader
- Safety
- Crypto
- File System
- Graphics
- Motor Control
- TCP/IP
- USB



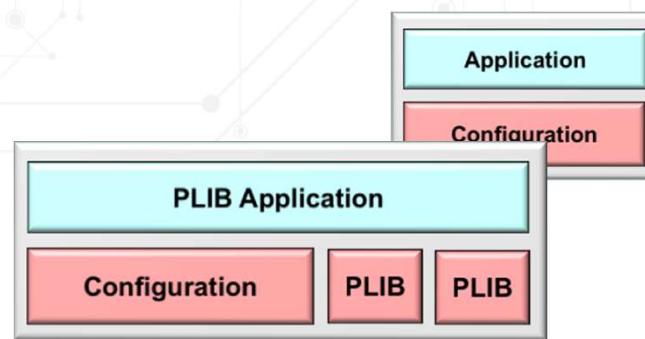
# Library Framework & GUI

- **Flexible Firmware Model**
  - Application Framework and Standalone Libraries
  - Scalable & Interoperable
- **Device Support & Peripheral Libraries (PLIB)**
  - Custom Initialization
  - Direct HW access
- **Drivers & Services**
  - Simple Interfaces
  - Modular & Interoperable
  - Abstracts HW Details
- **Middleware**
  - Value-Added SW Capabilities & Protocol Layers
  - Abstracted from Hardware
- **MPLAB Harmony Configurator (MHC)**
  - Standalone GUI & IDE Plug-in
  - Code Selection & Download
  - Device & Library Configuration
  - Project & Code Generation
- **Help Docs, Tools, & Utilities**



# Harmony CSP, Core, & Middleware

## Chip Support Package (CSP)

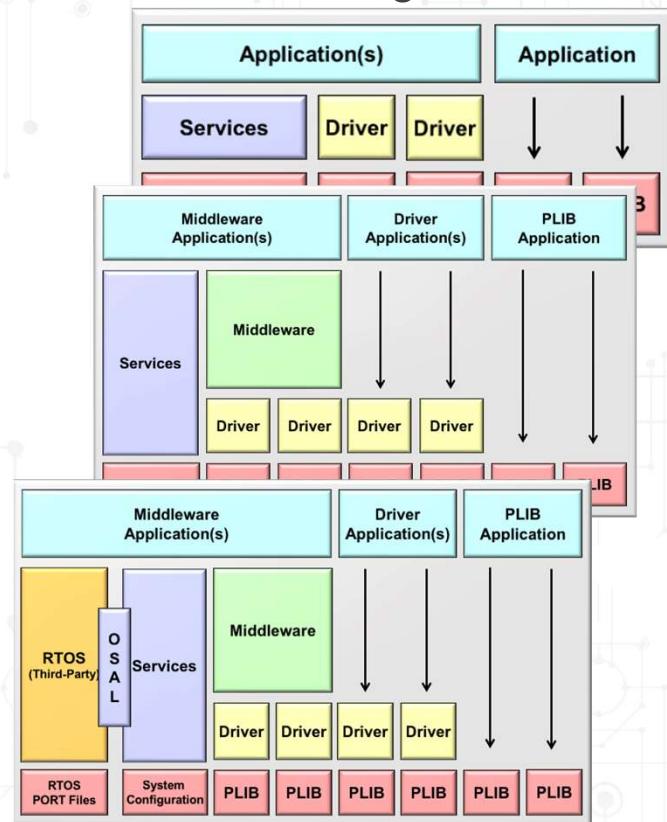


Simple Projects

PLIB-based & minimal configuration applications are supported by the CSP module (plus device pack support).

Harmony driver, service, middleware, and RTOS-based applications require Harmony core and other modules.

## Harmony Core & Middleware Packages



Advanced  
Capabilities

# Agenda

- 1. MPLAB Harmony 3 Introduction**
- 2. Overview of changes from MPLAB Harmony 2**
- 3. Introduction to the downloader and to the modular library packages**
- 4. Fundamentals of applications, projects, configuration, and peripheral libraries**
  - Lab1: Create a simple “heartbeat” LED application that flashes an LED using peripheral libraries
- 5. Introduction to drivers, services, and middleware**
  - Lab2: Create an application that enumerates a USB CDC device and demonstrates two-way communication between the USB Device and the USB Host PC
- 6. Interoperability and RTOS support**
- 7. Review of development models**

# SAM Device Support

**Harmony 3 adds support for SAM ARM Cortex and MPU devices**



Device Family	MPLAB X	bsp	dev_packs	core	csp
SAM A5D2		Yes	Yes	Yes	Yes
SAM S70/E70/V70/V71	Yes	Yes	Yes	Yes	Yes
SAM D5x/E5x	Yes	Yes	Yes	Yes	Yes
SAM D20/D21	Yes	Yes	Yes	Yes	Yes
SAM C20/C21	Yes	Yes	Yes	Yes	Yes
PIC32 MZ DA	Yes	Yes	Yes	Yes	Yes
PIC32 MK	Yes	Yes	Yes	Yes	Yes
PIC32 MZ EF	Yes	Yes	Yes	Yes	Yes

[https://github.com/Microchip-MPLAB-Harmony/Microchip-MPLAB-Harmony.github.io/wiki/device\\_support](https://github.com/Microchip-MPLAB-Harmony/Microchip-MPLAB-Harmony.github.io/wiki/device_support)

# Simplified PLIBs

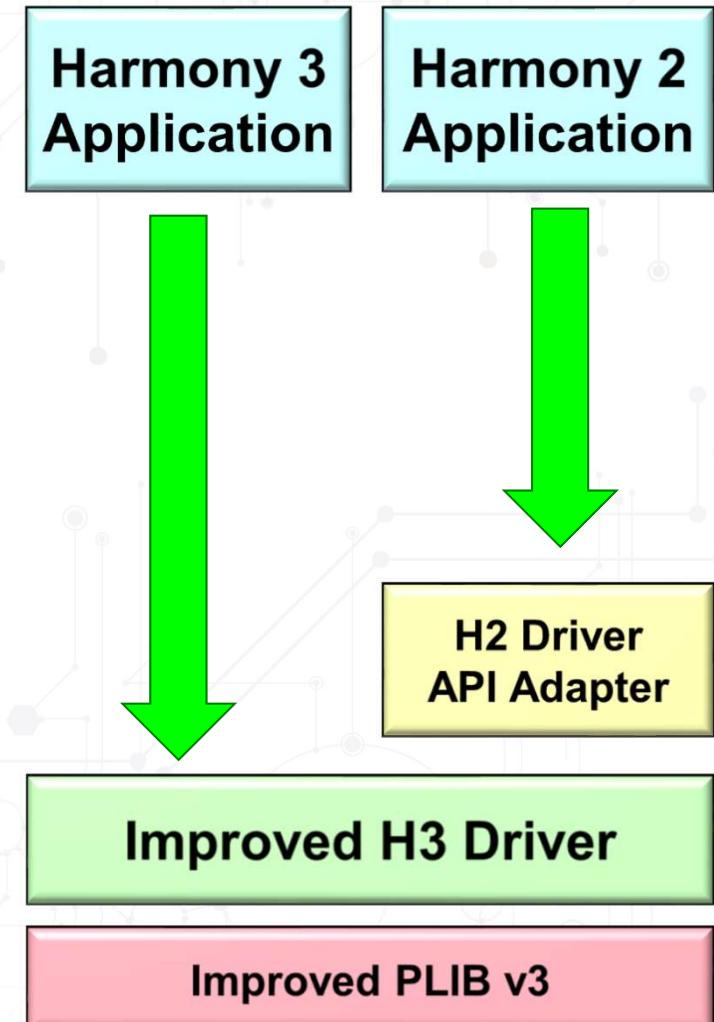
- **Easier to use APIs**
- **Similar to MCC Peripheral Libraries**
- **Single Instance**

Usage	UART	SPI	I2C/TWI
Initialization	<code>UARTx_Initialize()</code>	<code>SPIx_Initialize()</code>	<code>TWIx_Initialize()</code>
Transaction	<code>UARTx_Write()</code>	<code>SPIx_Exchange()</code>	<code>TWIx_Write()</code>
	<code>UARTx_Read()</code>		<code>TWIx_Read()</code>
			<code>TWIx_WriteRead()</code>
Status	<code>UARTx_ReadIsBusy()</code>	<code>SPIx_IsBusy()</code>	<code>TWIx_IsBusy()</code>
	<code>UARTx_ReadCountGet()</code>		
	<code>UARTx_WriteIsBusy()</code>		
	<code>UARTx_WriteCountGet()</code>		

- Will continue to be single client.
- Use actual functions instead of in-lines or macros.
- Have optional blocking calls where desirable.
- Implement ISR functionality (with call-back support).

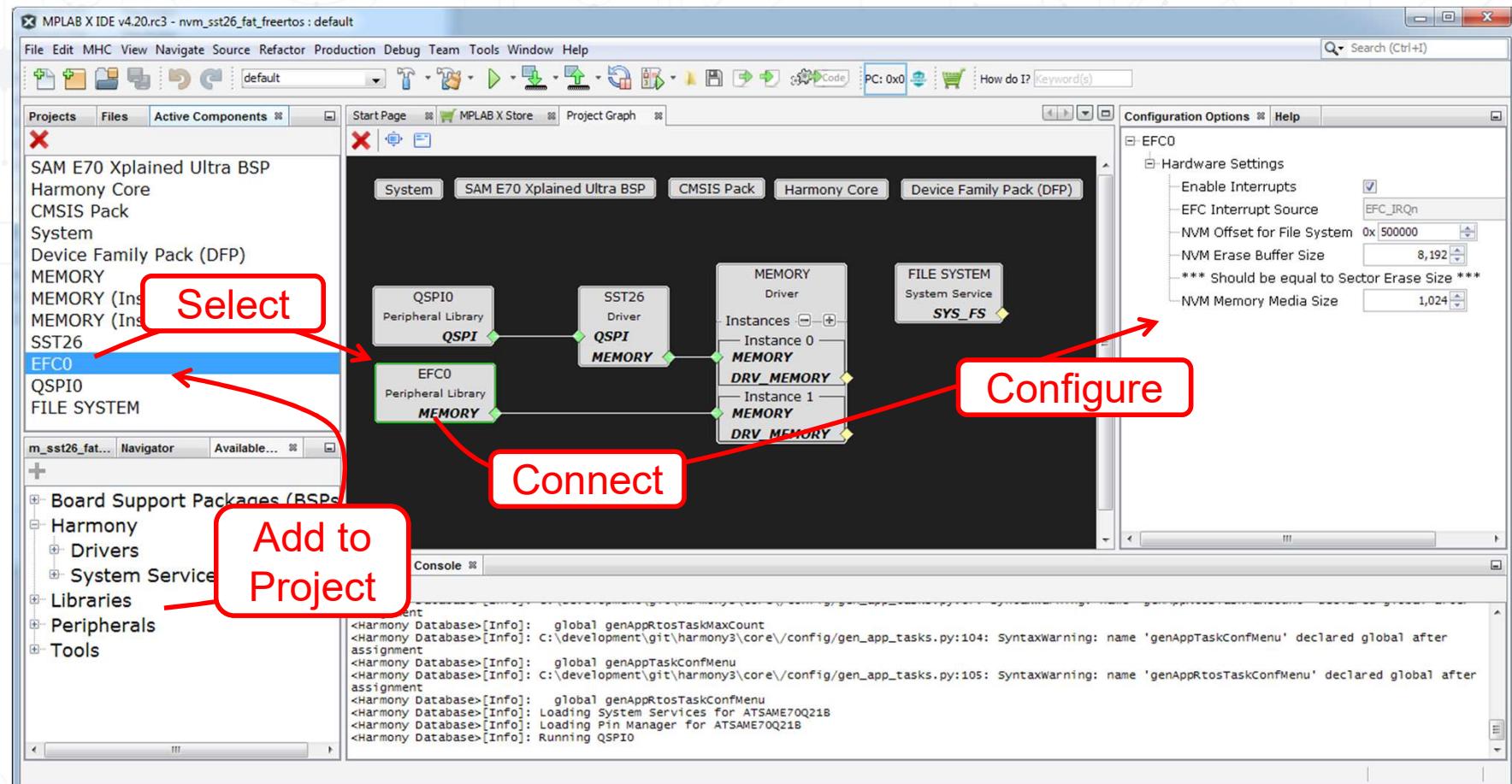
# Updated Drivers and Services

- **Same advanced Harmony functionality.**
  - Multi-client, Multi-instance,
  - Buffer queuing, RTOS, etc.
- **Updated to use new PLIBs.**
  - Will manage PLIB instances.
  - Will not use blocking APIs.
- **Some API Improvements.**
  - Usage model simplifications.
  - H2 compatibility adapters for backward compatibility\*.



\* Not available for beta releases.

# MHC Configuration Graph



- System Configuration Graph
- Component Selection & Connection
- MPLAB & Standalone
- Intuitive Resource Management

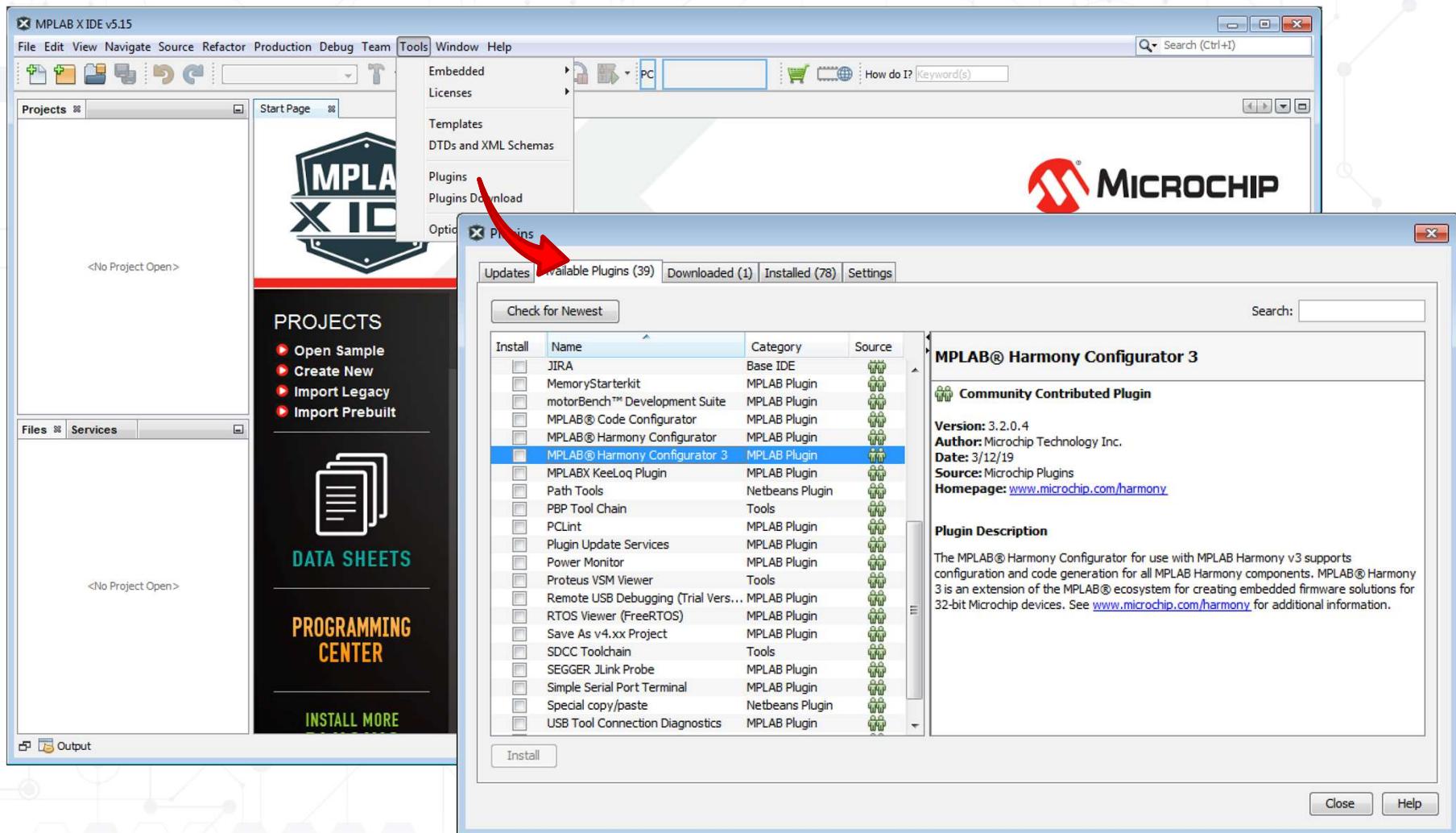
# General Notes

- **Same Great Middleware (No Changes!)**
- **RTOS Enabled by Default.**
  - Defaults to FreeRTOS for improved efficiency.
  - Still supports running cooperatively, bare-metal.
- **Modular Download Packages**
  - Separate GitHub Repo's for device, core, MW, and other packages.
  - Download/Package manager coming soon.
- **Simpler! Easier! Better!**

# Agenda

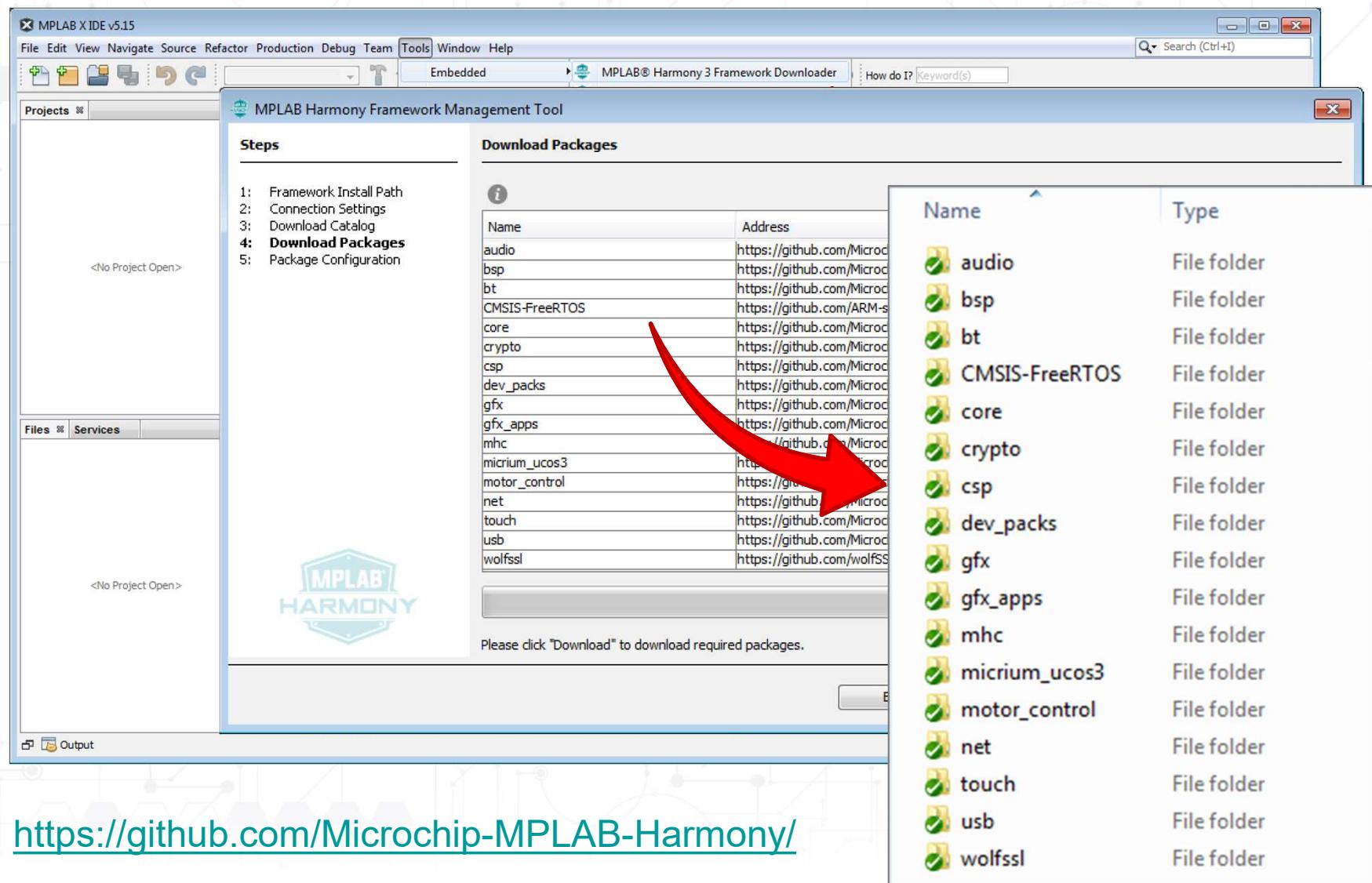
- 1. MPLAB Harmony 3 Introduction**
- 2. Overview of changes from MPLAB Harmony 2**
- 3. Introduction to the downloader and to the modular library packages**
- 4. Fundamentals of applications, projects, configuration, and peripheral libraries**
  - Lab1: Create a simple “heartbeat” LED application that flashes an LED using peripheral libraries
- 5. Introduction to drivers, services, and middleware**
  - Lab2: Create an application that enumerates a USB CDC device and demonstrates two-way communication between the USB Device and the USB Host PC
- 6. Interoperability and RTOS support**
- 7. Review of development models**

# MHC Plugin Installation



<https://www.microchip.com/mplab>

# Module Deployment



The screenshot shows the MPLAB X IDE interface with the Harmony Framework Management Tool open. The tool has two main sections: 'Steps' and 'Download Packages'.

**Steps:**

- Framework Install Path
- Connection Settings
- Download Catalog
- Download Packages**
- Package Configuration

**Download Packages:**

Name	Address
audio	<a href="https://github.com/MicrochipTechnology/audio">https://github.com/MicrochipTechnology/audio</a>
bsp	<a href="https://github.com/MicrochipTechnology/bsp">https://github.com/MicrochipTechnology/bsp</a>
bt	<a href="https://github.com/MicrochipTechnology/bt">https://github.com/MicrochipTechnology/bt</a>
CMSIS-FreeRTOS	<a href="https://github.com/ARM-software/CMSIS-FreeRTOS">https://github.com/ARM-software/CMSIS-FreeRTOS</a>
core	<a href="https://github.com/MicrochipTechnology/core">https://github.com/MicrochipTechnology/core</a>
crypto	<a href="https://github.com/MicrochipTechnology/crypto">https://github.com/MicrochipTechnology/crypto</a>
csp	<a href="https://github.com/MicrochipTechnology/csp">https://github.com/MicrochipTechnology/csp</a>
dev_packs	<a href="https://github.com/MicrochipTechnology/dev_packs">https://github.com/MicrochipTechnology/dev_packs</a>
gfx	<a href="https://github.com/MicrochipTechnology/gfx">https://github.com/MicrochipTechnology/gfx</a>
gfx_apps	<a href="https://github.com/MicrochipTechnology/gfx_apps">https://github.com/MicrochipTechnology/gfx_apps</a>
mhc	<a href="https://github.com/MicrochipTechnology/mhc">https://github.com/MicrochipTechnology/mhc</a>
micrium_uicos3	<a href="https://github.com/MicrochipTechnology/micrium_uicos3">https://github.com/MicrochipTechnology/micrium_uicos3</a>
motor_control	<a href="https://github.com/MicrochipTechnology/motor_control">https://github.com/MicrochipTechnology/motor_control</a>
net	<a href="https://github.com/MicrochipTechnology/net">https://github.com/MicrochipTechnology/net</a>
touch	<a href="https://github.com/MicrochipTechnology/touch">https://github.com/MicrochipTechnology/touch</a>
usb	<a href="https://github.com/MicrochipTechnology/usb">https://github.com/MicrochipTechnology/usb</a>
wolfssl	<a href="https://github.com/wolfSSL/wolfssl">https://github.com/wolfSSL/wolfssl</a>

Please click "Download" to download required packages.

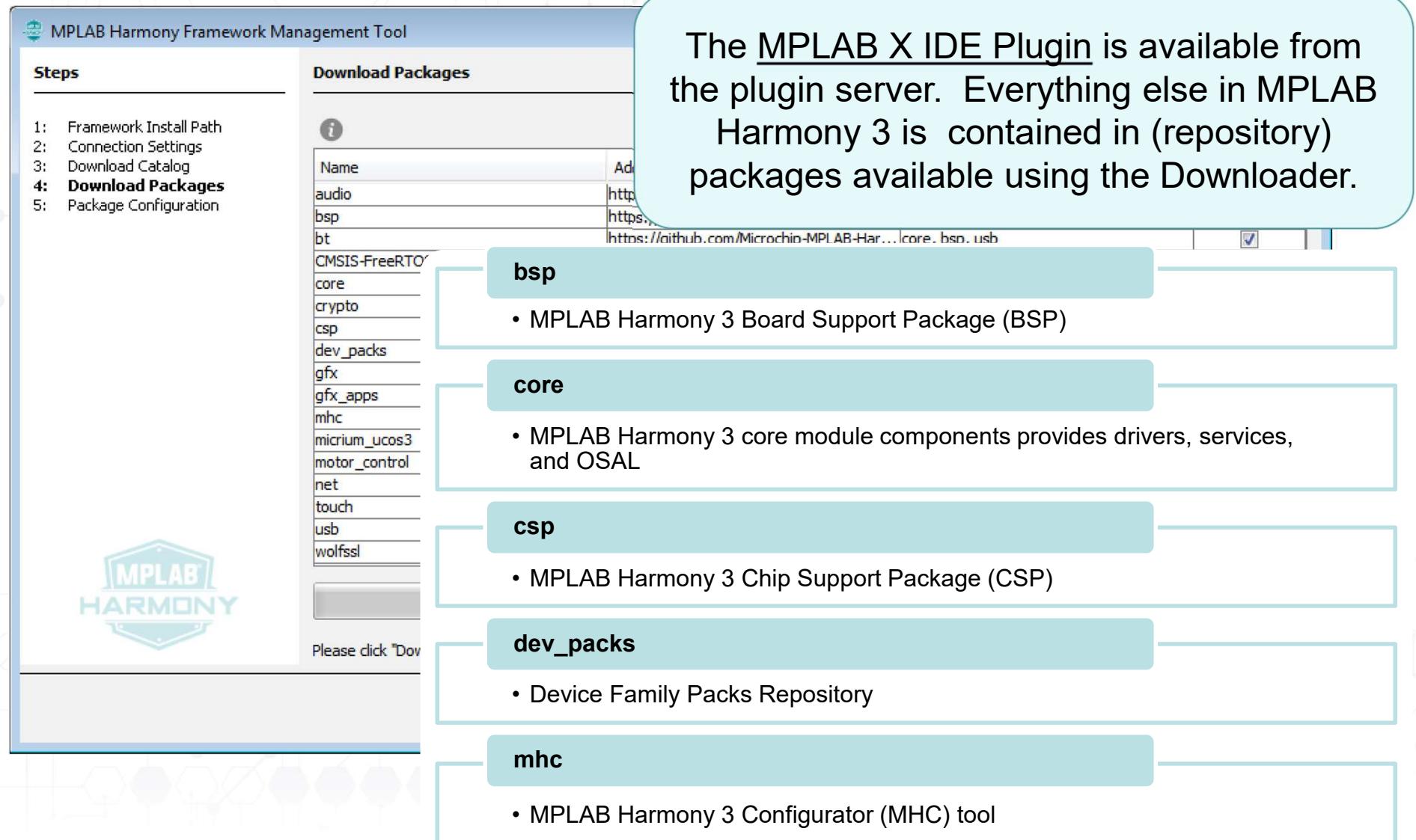
**File List:**

Name	Type
audio	File folder
bsp	File folder
bt	File folder
CMSIS-FreeRTOS	File folder
core	File folder
crypto	File folder
csp	File folder
dev_packs	File folder
gfx	File folder
gfx_apps	File folder
mhc	File folder
micrium_uicos3	File folder
motor_control	File folder
net	File folder
touch	File folder
usb	File folder
wolfssl	File folder

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

# Harmony 3 Base Packages

The MPLAB X IDE Plugin is available from the plugin server. Everything else in MPLAB Harmony 3 is contained in (repository) packages available using the Downloader.



**bsp**

- MPLAB Harmony 3 Board Support Package (BSP)

**core**

- MPLAB Harmony 3 core module components provides drivers, services, and OSAL

**csp**

- MPLAB Harmony 3 Chip Support Package (CSP)

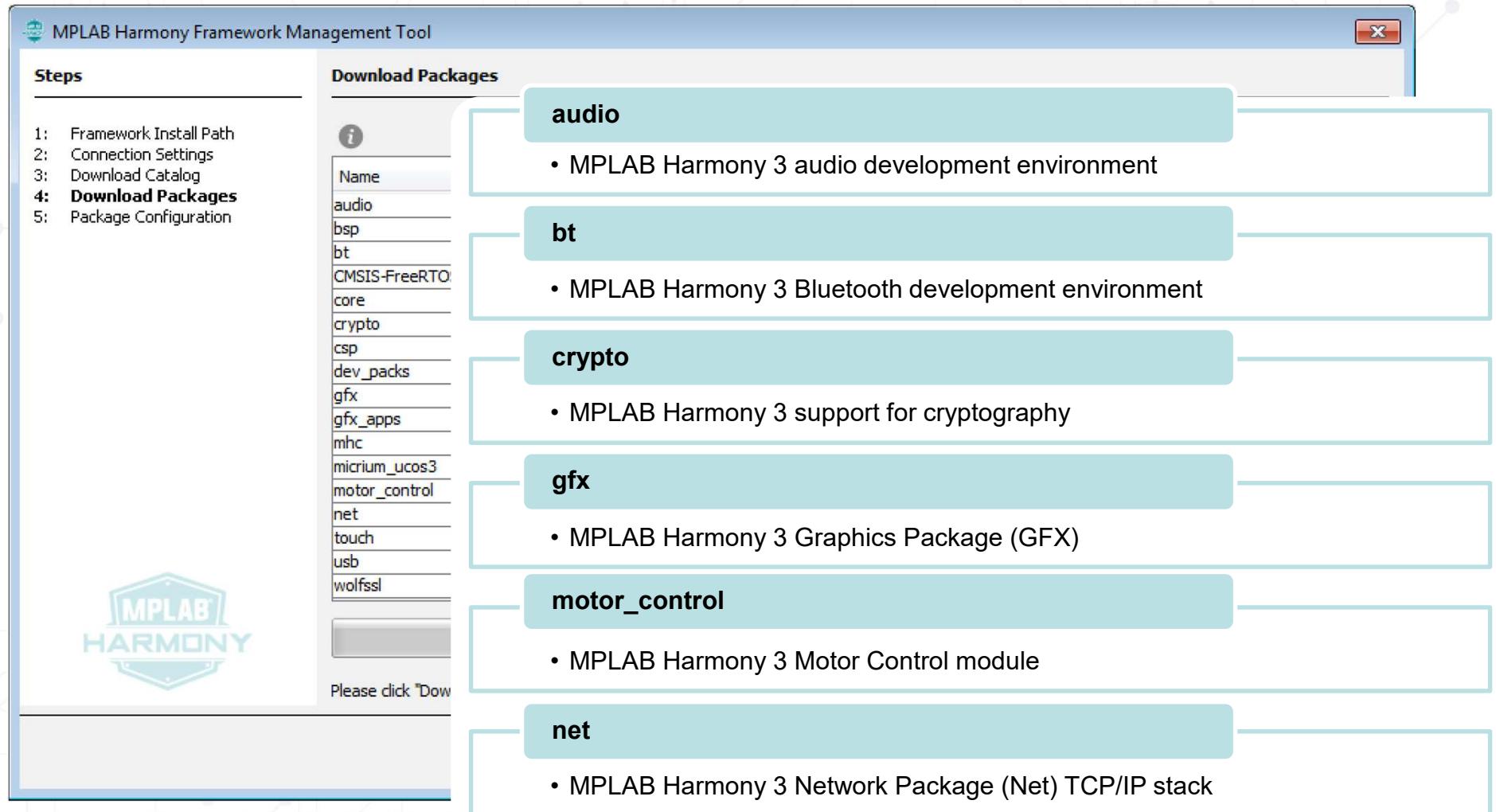
**dev\_packs**

- Device Family Packs Repository

**mhc**

- MPLAB Harmony 3 Configurator (MHC) tool

# Harmony 3 MW Packages



**MPLAB Harmony Framework Management Tool**

**Steps**

- 1: Framework Install Path
- 2: Connection Settings
- 3: Download Catalog
- 4: Download Packages**
- 5: Package Configuration

**Download Packages**

- audio**
  - MPLAB Harmony 3 audio development environment
- bt**
  - MPLAB Harmony 3 Bluetooth development environment
- crypto**
  - MPLAB Harmony 3 support for cryptography
- gfx**
  - MPLAB Harmony 3 Graphics Package (GFX)
- motor\_control**
  - MPLAB Harmony 3 Motor Control module
- net**
  - MPLAB Harmony 3 Network Package (Net) TCP/IP stack

Please click "Download" to begin the download process.

# Harmony 3 MW Packages

**MPLAB Harmony Framework Management Tool**

**Steps**

- 1: Framework Install Path
- 2: Connection Settings
- 3: Download Catalog
- 4: Download Packages**
- 5: Package Configuration

**Download Packages**

Name	Address	Dependencies	Download
audio			<input checked="" type="checkbox"/>
bsp			<input type="checkbox"/>
bt			<input type="checkbox"/>
CMSIS-FreeRTOS			<input type="checkbox"/>
core			<input type="checkbox"/>
crypto			<input type="checkbox"/>
csp			<input type="checkbox"/>
dev_packs			<input type="checkbox"/>
gfx			<input type="checkbox"/>
gfx_apps			<input type="checkbox"/>
mhc			<input type="checkbox"/>
micrium_ucos3			<input type="checkbox"/>
motor_control			<input type="checkbox"/>
net			<input type="checkbox"/>
touch			<input type="checkbox"/>
usb			<input type="checkbox"/>
wolfssl			<input type="checkbox"/>

Please click "Download" to start the download process.

**touch**

- Microchip MPLAB Harmony 3 Touch Library

**usb**

- MPLAB Harmony 3 USB module

**CMSIS-FreeRTOS**

- ARM CMSIS-RTOS adoption of FreeRTOS (third-party repo)

**micrium\_ucos3**

- MPLAB Harmony 3 applications (Micrium software not included)

**wolfssl**

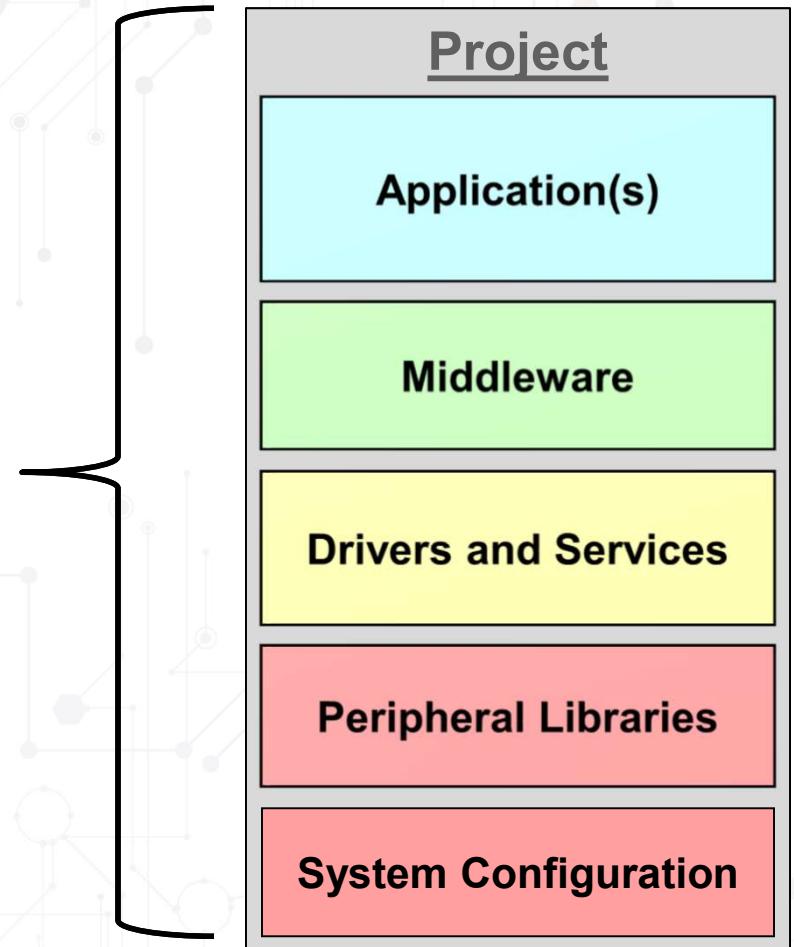
- wolfSSL embedded SSL library (third-party repo)

# Agenda

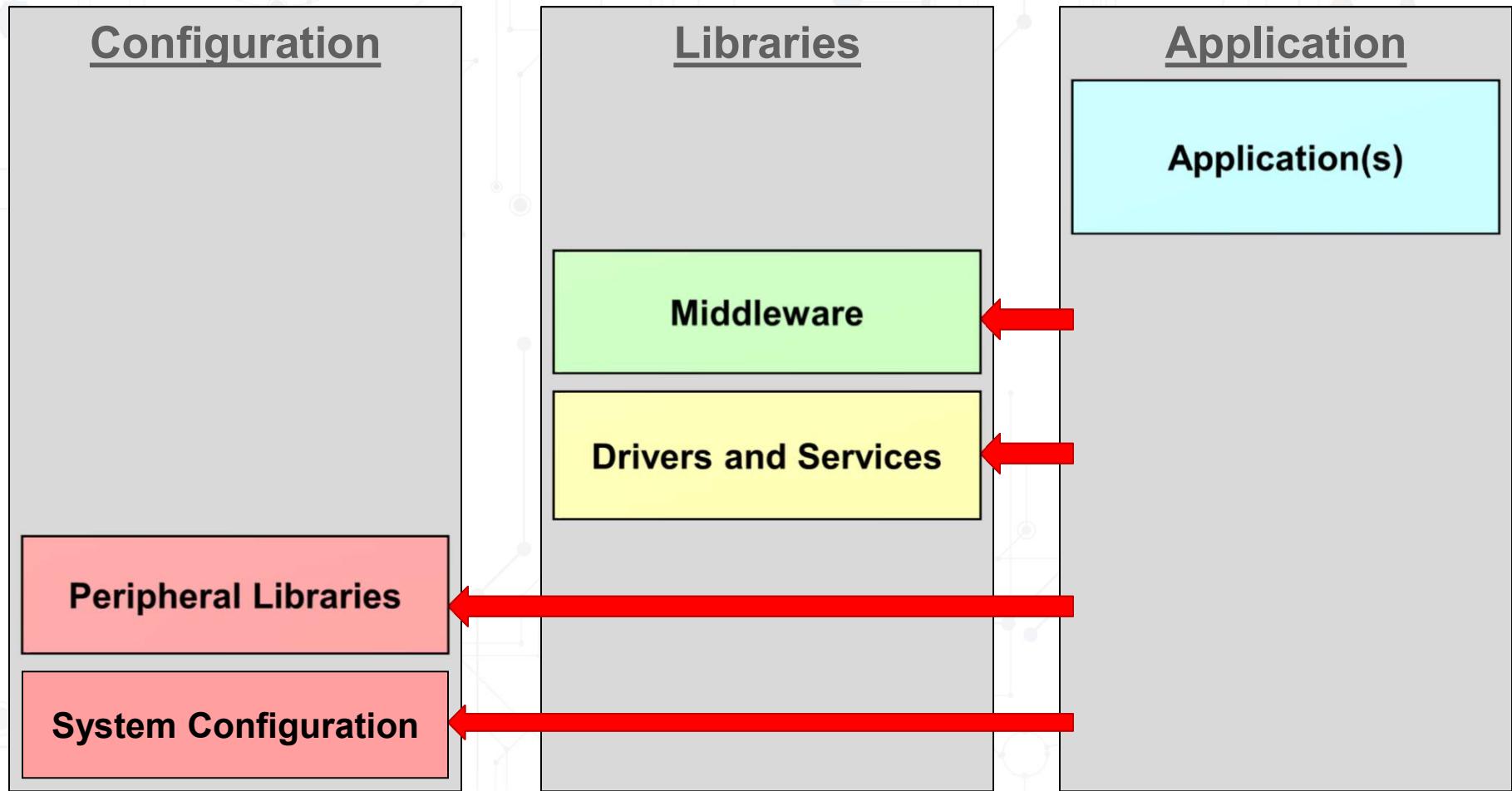
- 1. MPLAB Harmony 3 Introduction**
- 2. Overview of changes from MPLAB Harmony 2**
- 3. Introduction to the downloader and to the modular library packages**
- 4. Fundamentals of applications, projects, configuration, and peripheral libraries**
  - Lab1: Create a simple “heartbeat” LED application that flashes an LED using peripheral libraries
- 5. Introduction to drivers, services, and middleware**
  - Lab2: Create an application that enumerates a USB CDC device and demonstrates two-way communication between the USB Device and the USB Host PC
- 6. Interoperability and RTOS support**
- 7. Review of development models**

# Application Project Parts

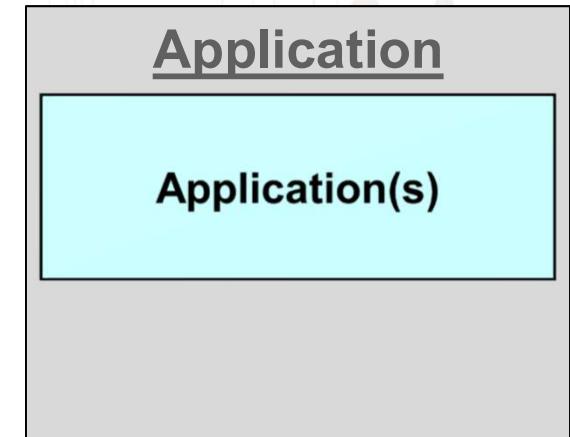
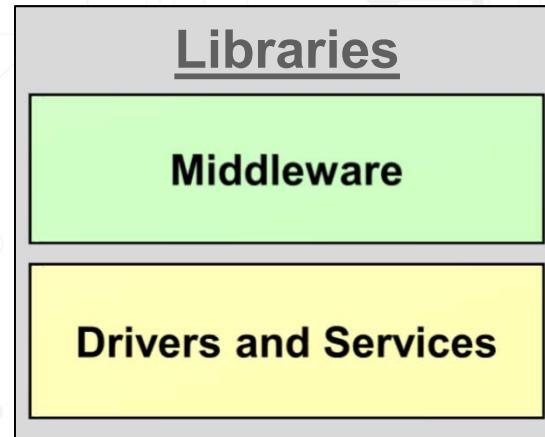
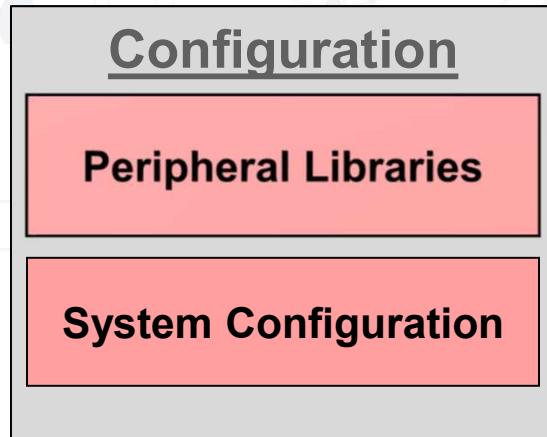
**Different Parts of an  
MPLAB Harmony 3  
application serve  
different purposes.**



# Application Factoring



# Grouping Application “Parts”



**Generated by MHC**



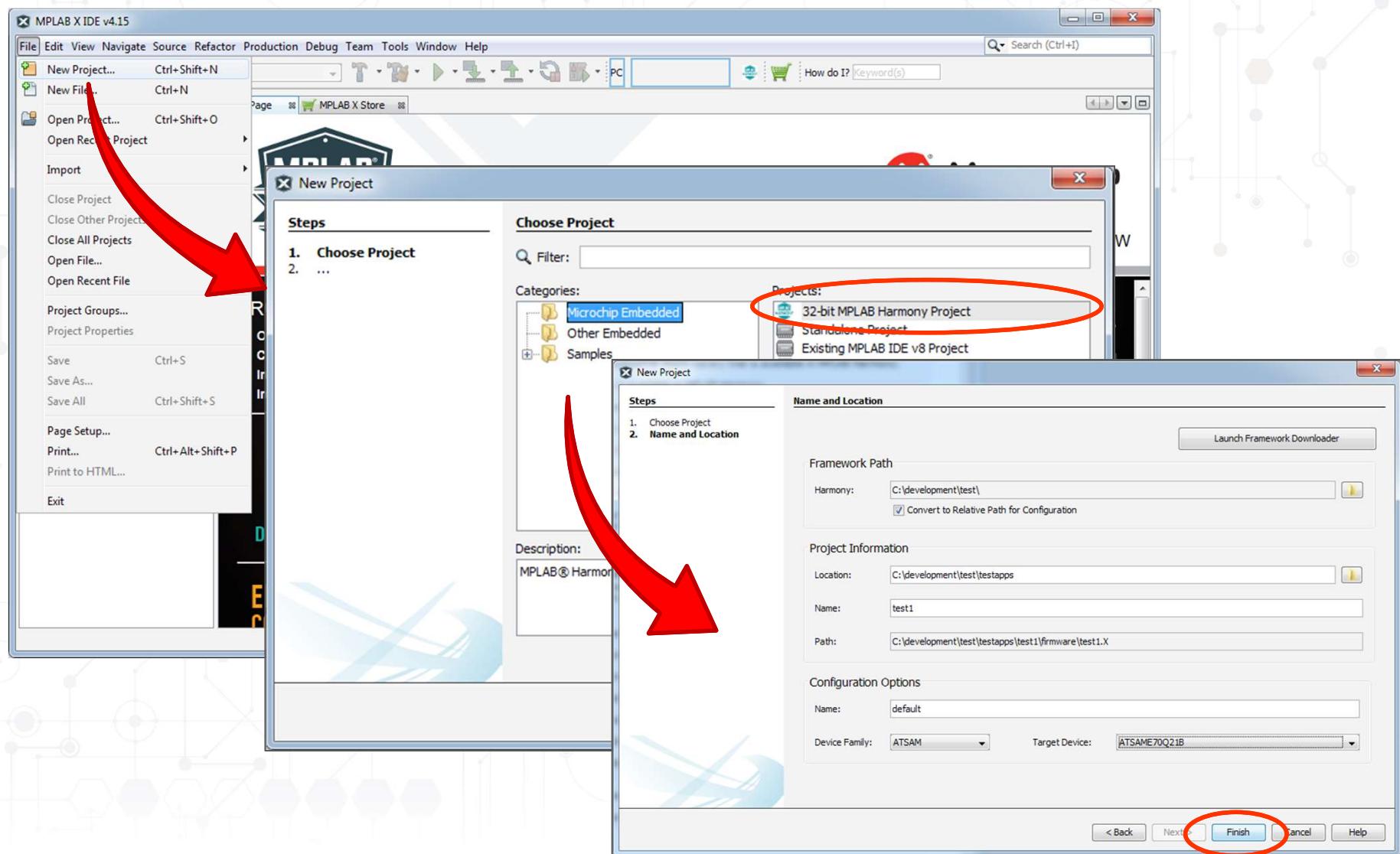
**Written by Microchip**

**&  
Third Party Partner  
Libraries**

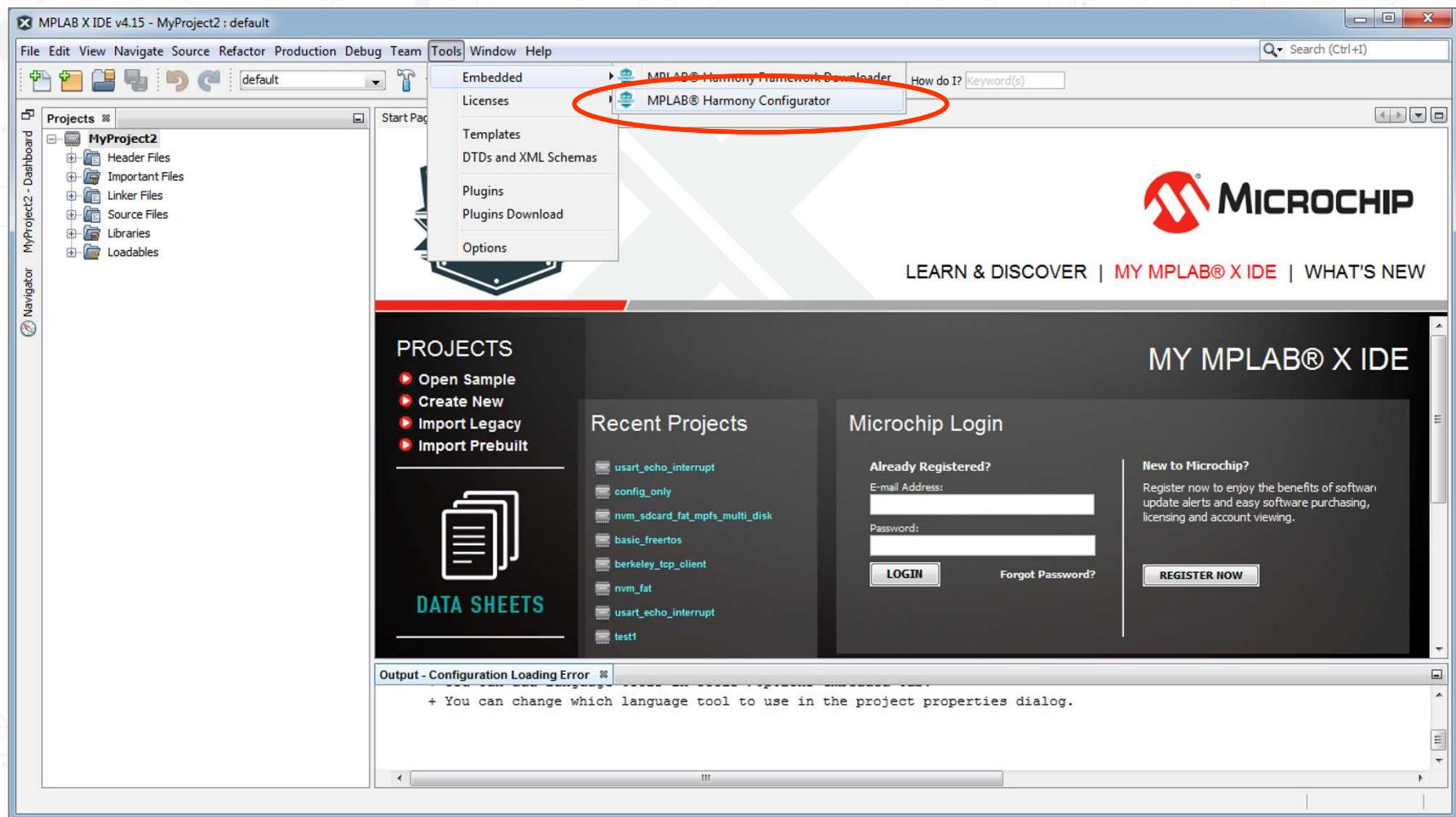


**Application  
developers focus  
on application  
development.**

# Application Project Creation

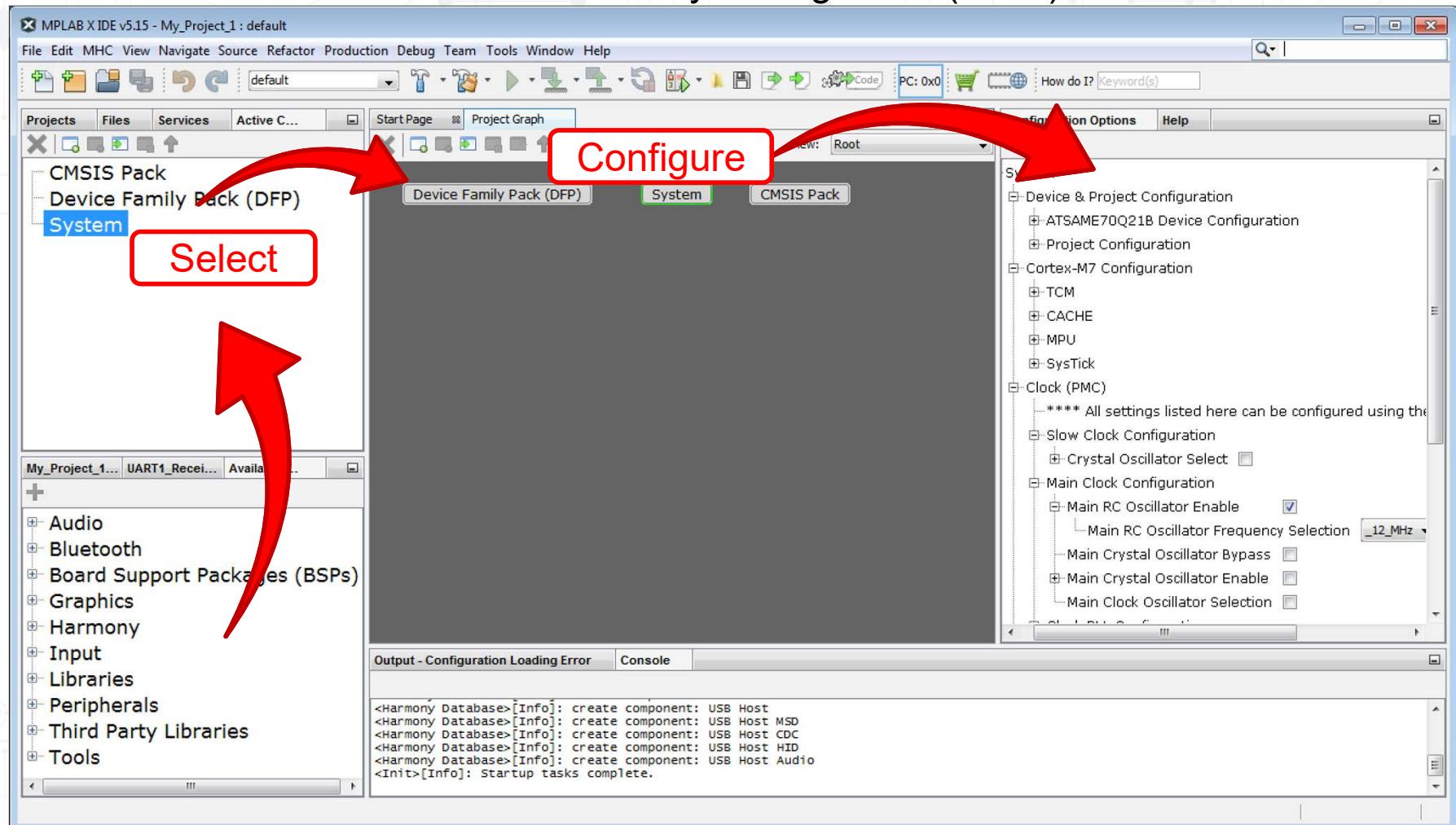


# Application Project Creation



# Configuration GUI

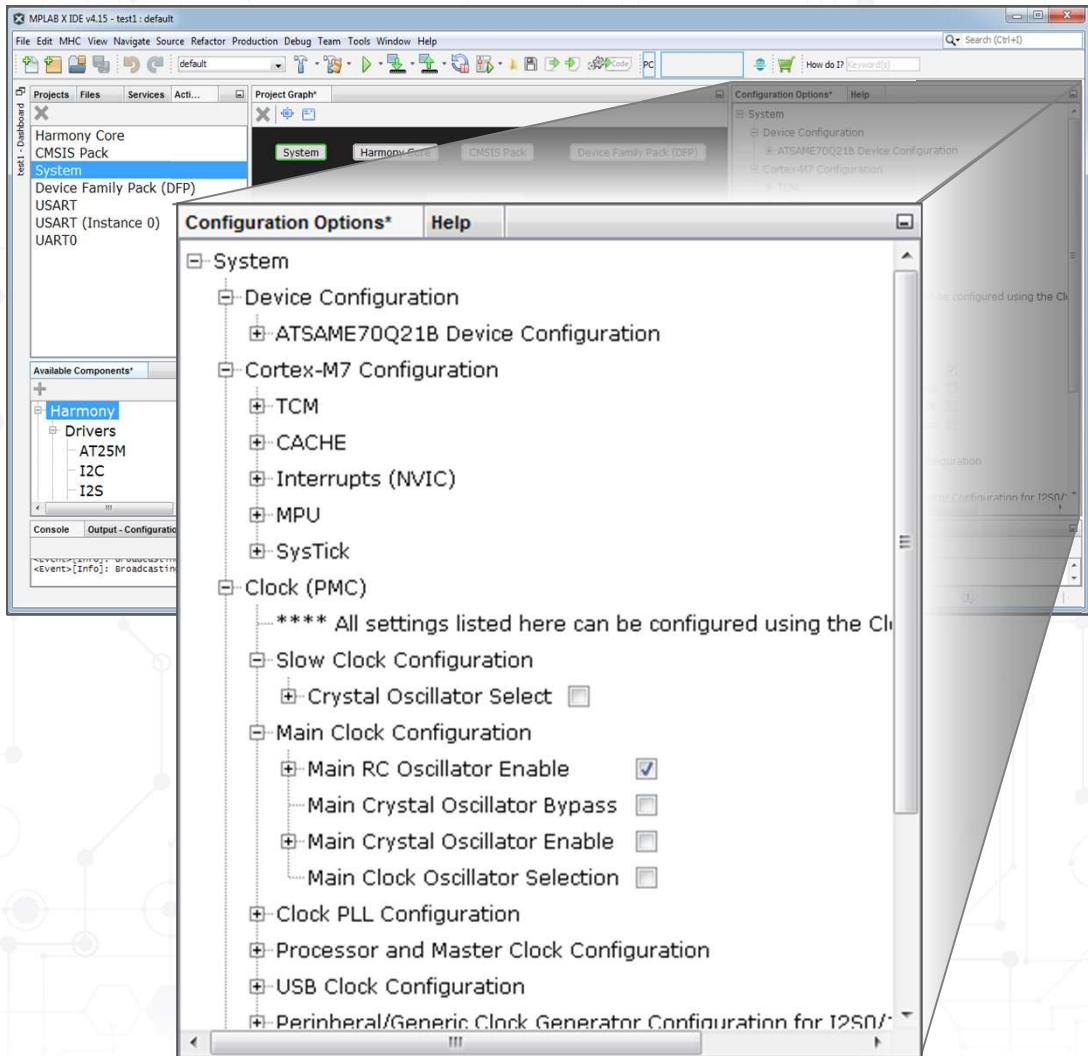
## MPLAB Harmony Configurator (MHC)



# CMSIS / DFPs

- **CMSIS Pack (Cortex MCU S/W Interface Std.)**
  - Provided by ARM
  - Core specific information/definitions
  - CMSIS always part of MPLAB X releases. ARM may update independently/asynchronously
- **Device Family Packs (DFP)**
  - Provided by Microchip for all 32-bit MCU's in Harmony 3
  - Product specific information/definitions/support
  - DFP's always part of MPLAB X releases.  
Asynchronous releases may happen in between
- **BOTH are required in MPLAB X Projects**

# Configuration Selection

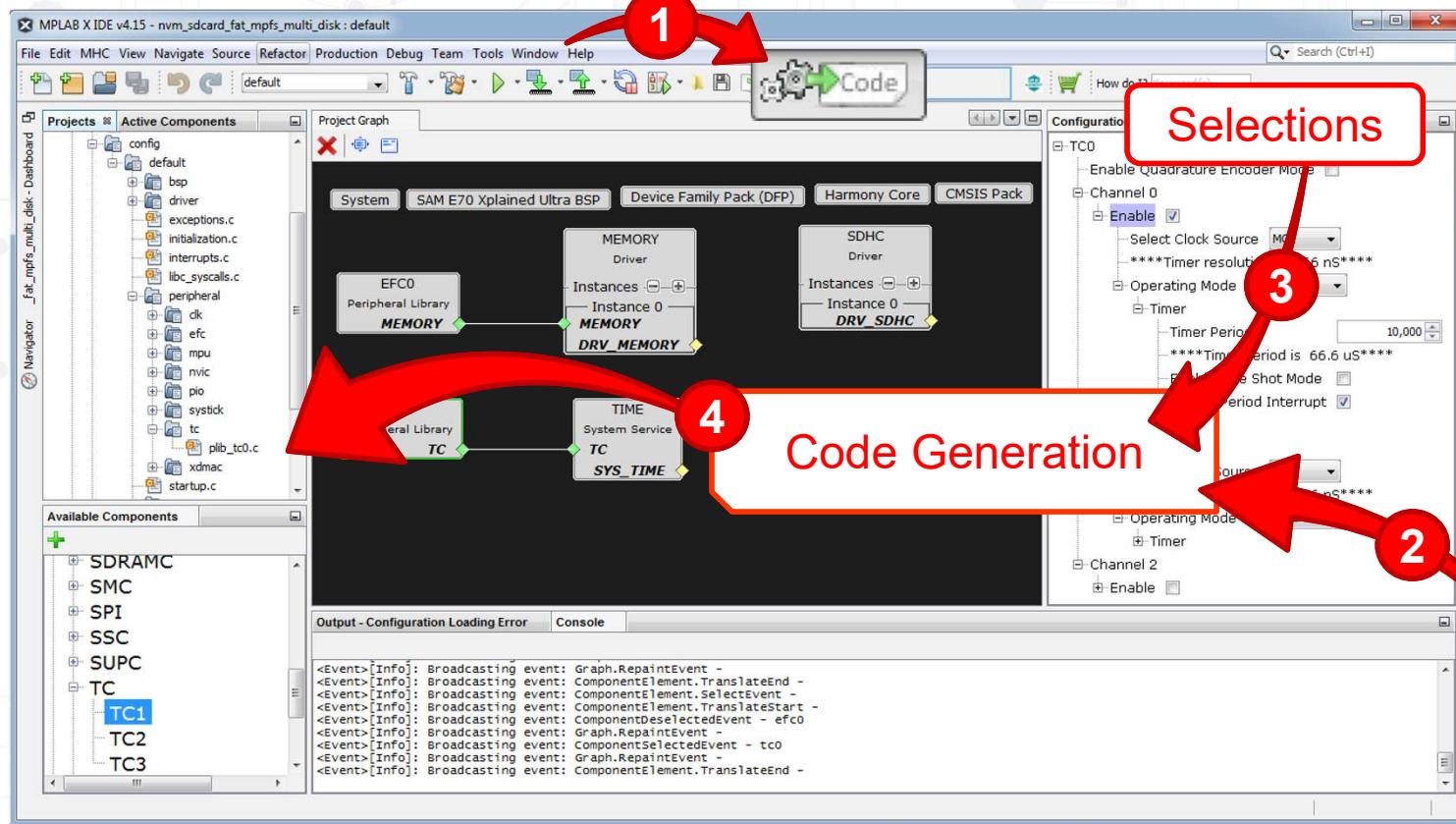


## System Settings

(Basic device settings)

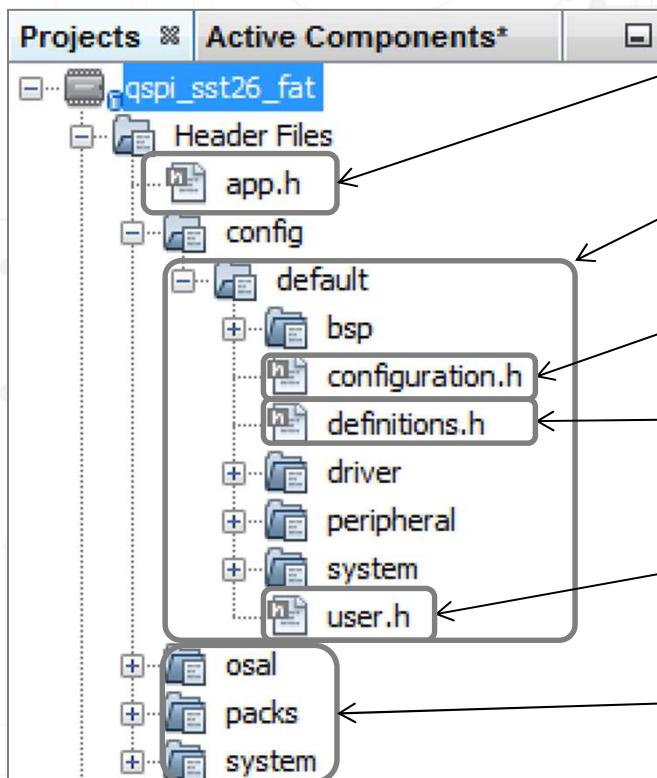
- Security, boot modes
- Cache
- Memory (TCM, MPU)
- Clocks
  - PLL/Oscillators
  - Peripheral Clocks
  - SysTick/Core Timer
- Ports/Pins Settings
- Interrupts, DMA
- Watchdog

# Code Generation



# Project Organization

## Header Files



Application header file

Configuration (name: "default") files

Build options (configuration #define) header

System definitions header (ex. sysObj)

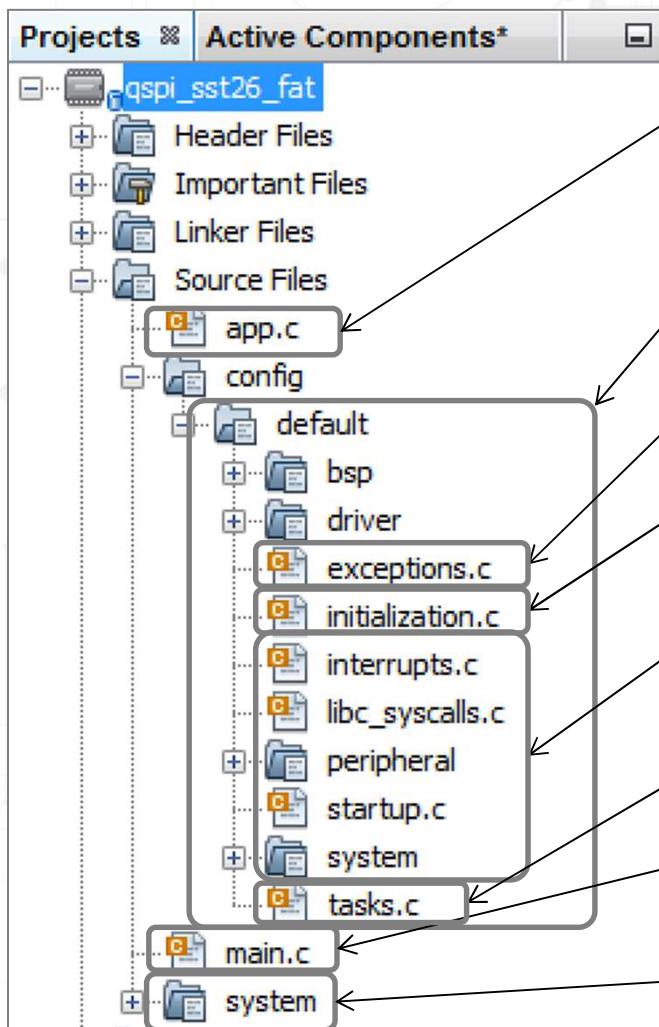
User-defined build options header

OSAL and System Service Headers

Other configuration-specific folders contain generated library headers (ex. PLIBs).

# Project Organization

## Source Files



Application source file

Configuration (name: “default”) files

Exception handlers source code

SYS\_Initialize (and init data) implementation

Interrupts, sys-calls, & startup source code

SYS\_Tasks (and thread) implementations

Main function implementation (common)

System services implementation

# Initialization Code

## Device Configuration

main.c

```
int main ( void )
{
    SYS_Initialize ( NULL );

    while ( true )
    {
        /* To Do: Develop Application */
    }

    return ( EXIT_FAILURE );
}
```



**May also include:**

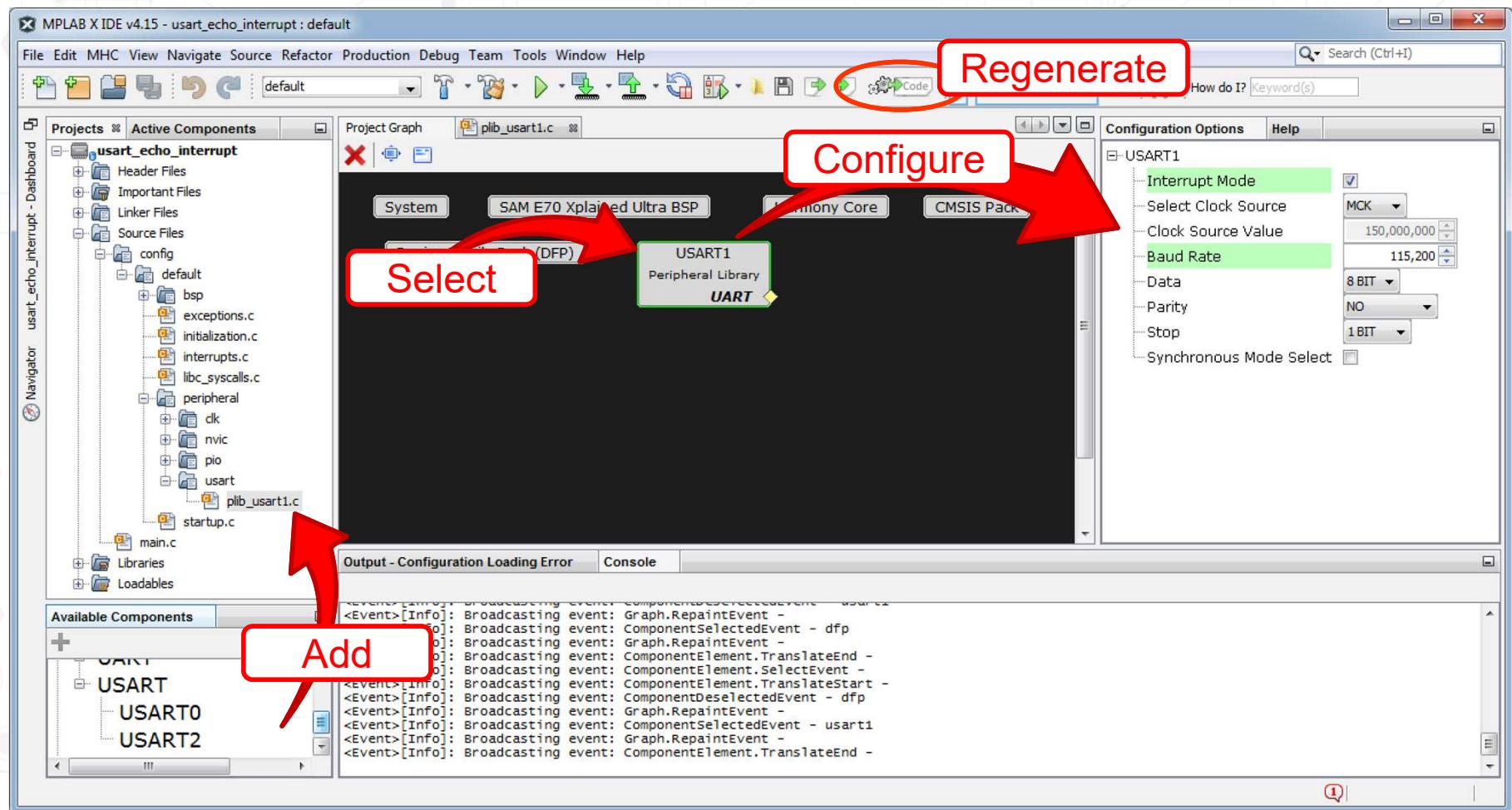
- Configuration Bits
- Startup Code
- Interrupt Vectors
- Exception/SysCall Code
- Core Peripherals

initialize.c

```
void SYS_Initialize ( void* data )
{
    CLK_Initialize();
    PIO_Initialize();
    NVIC_Initialize();

    ...
}
```

# Peripheral Libraries



# A PLIB for Every Peripheral

Microchip 32-bit Chip Support Package

Hide Locate Back Forward Home Print Options

Contents | Index | Search | Favorites |

Peripheral Library Overview > Peripheral Libraries Help > Peripheral Libraries > UART Peripheral Library Help > Library Interface

**Microchip 32-bit Chip Support Package** [Contents](#) | [Index](#) | [Home](#) [Previous](#) | [Up](#) | [Next](#) [Documentation Feedback](#) [Microchip Support](#)

**Library Interface**

**a) Initialization Function**

	Name	Description
≡	<a href="#">UARTx_Initialize</a>	Initializes given instance of the UART peripheral.

**b) Setup Functions**

	Name	Description
≡	<a href="#">UARTx_SerialSetup</a>	Sets the UART serial communication settings dynamically.
≡	<a href="#">UARTx_ReadCallbackRegister</a>	Sets the pointer to the function (and its context) to be called when the given UART's read events occur.
≡	<a href="#">UARTx_WriteCallbackRegister</a>	Sets the pointer to the function (and its context) to be called when the given UART's write events occur.

**c) Transaction Functions**

	Name	Description
≡	<a href="#">UARTx_Read</a>	Submits request to read n-Bytes of data to the given UART peripheral.
≡	<a href="#">UARTx_Write</a>	Submits a write buffer to the given UART peripheral to transfer.
≡	<a href="#">UARTx_ReadByte</a>	Submits request to read a byte of data to the given UART peripheral.
≡	<a href="#">UARTx_WriteByte</a>	Submits a byte of data to the given UART peripheral to transfer.

**d) Status Functions**

	Name	Description
≡	<a href="#">UARTx_ErrorGet</a>	Gets the error of the given UART peripheral instance.
≡	<a href="#">UARTx_ReadCountGet</a>	Returns the count of number of bytes processed for a given UART read operation.
≡	<a href="#">UARTx_WriteCountGet</a>	Returns the count of number of bytes processed for a given UART write operation.
≡	<a href="#">UARTx_ReadIsBusy</a>	Returns the read request status associated with the given UART peripheral instance.
≡	<a href="#">UARTx_WriteIsBusy</a>	Returns the write request status associated with the given UART peripheral instance.
≡	<a href="#">UARTx_ReceiverIsReady</a>	Returns the hardware status of the UART Receiver.
≡	<a href="#">UARTx_TransmitterIsReady</a>	Returns the hardware status of the UART Transmitter.

# PLIB-based Project

## Peripheral Libraries

### main.c

```
int main ( void )
{
    /* Initialize all modules */
    SYS_Initialize ( NULL );

    /* Register callback functions */
    USART1_WriteCallbackRegister( USART1 );
    USART1_ReadCallbackRegister( USART1 );
    USART1_Write( &messageStart, 1 );

    while ( true )
    {
        if(errorStatus == true)
        {
            /* Send error message */
            errorStatus = false;
            USART1_Write( &messageEnd, 1 );
        }
    }
    ...
}
```

```
void USART1_Initialize( void )
{
    /* Reset USART1 */
    USART1_REGS->US_CR = (US_CR_RSTRX_Msk | US_CR_RSTTX_Msk);

    /* Enable USART1 */
    USART1_REGS->US_CR = (US_CR_TXEN_Msk | US_CR_RXEN_Msk);

    /* Configure USART1 mode */
    USART1_REGS->US_MR = ((US_MR_USCLKS_MCK) | (0 << US_MR_
    _USCLKS_Pos));

    /* Configure USART1 Baud Rate */
    USART1_REGS->US_BRGR = US_BRGR_CD(81);

    /* Initialize instance object */
    usart1Obj.rxBuffer = NULL;
    usart1Obj.rxSize = 0;
    usart1Obj.rxProcessedSize = 0;
    ...
}
```

PLIB Application

Configuration

PLIB

PLIB

initialize.c

# Lab1:

## Create a simple “heartbeat” LED application that flashes an LED using peripheral libraries

# Lab 1: Objectives

- Be able to Install MPLAB® Harmony Configurator from the Microchip Plugins Update Center
- Be able to Select MPLAB Harmony 3 Packages
- Be able to explain Harmony project structure, configure and use Harmony Peripheral Libraries

# Lab 1: Install MHC

- **Github: <https://github.com/Microchip-MPLAB-Harmony/mhc/wiki>**
- **Follow section “Installing MPLAB® Harmony Configurator from the Microchip Plugins Update Center”**

# Lab 1: Select H3 packages

- Github: <https://github.com/Microchip-MPLAB-Harmony/mhc/wiki>
- Follow section “Selecting MPLAB Harmony 3 Packages”

# Lab 1: Create H3 project

- **Github:** <https://github.com/Microchip-MPLAB-Harmony/csp/wiki/Create-your-first-peripheral-library-project>
- **Follow section “Create your first peripheral library project”**
- **Note:**
  - The example in the above link is for SAMC21n, replace reference from “c21n” to “d21”
  - Under section “Setup MPLAB® Harmony Project Configurator to Generate Code”, point 1, Use TC3 instead of TC0. (There is no TC0 in D21 device)
  - Under section “Setup MPLAB® Harmony Project Configurator to Generate Code”, point 4, Map the LED to PB30.
  - Under section “Adding Code to main.c”, , replace reference from “TC0” to “TC3”

# Lab 1: Summary

- In this lab we have...
- Successfully created first MPLAB® Harmony project using MHC
- Learnt the structure of an MPLAB Harmony project
- Configured and used PORT Peripheral Libraries

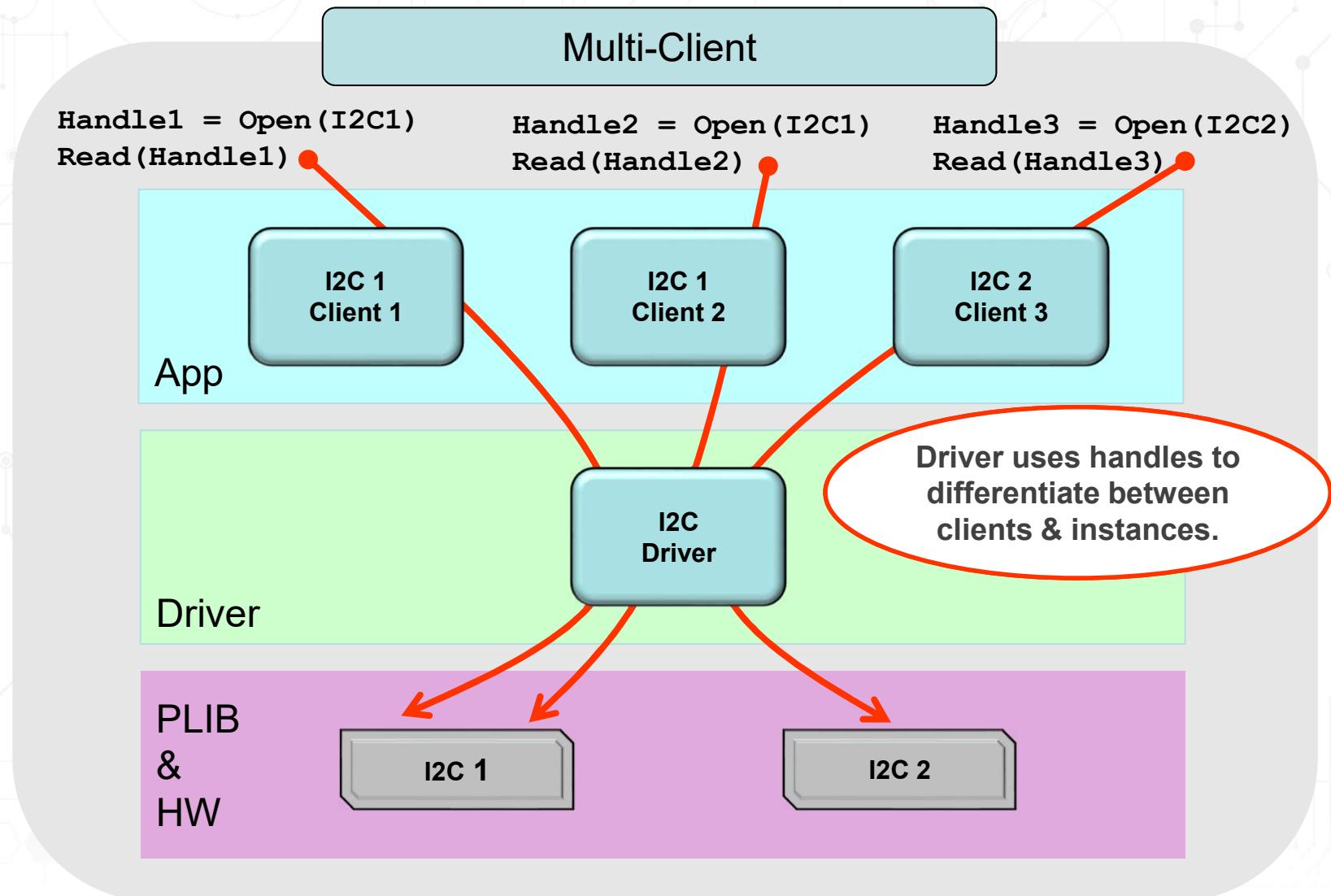
# Agenda

- 1. MPLAB Harmony 3 Introduction**
- 2. Overview of changes from MPLAB Harmony 2**
- 3. Introduction to the downloader and to the modular library packages**
- 4. Fundamentals of applications, projects, configuration, and peripheral libraries**
  - Lab1: Create a simple “heartbeat” LED application that flashes an LED using peripheral libraries
- 5. Introduction to drivers, services, and middleware**
  - Lab2: Create an application that enumerates a USB CDC device and demonstrates two-way communication between the USB Device and the USB Host PC
- 6. Interoperability and RTOS support**
- 7. Review of development models**

# Harmony Drivers

- Built on top of PLIBs
- Can be Multi-instance, Multi-client
- Simple, Abstracted Interfaces
- Portable & Backwards-compatible
- Interoperable
- Configurable Options
- Selectable Features
- Buffer Queuing & Advanced Capabilities

# Multi-Instance & Multi-Client



# Portable Interfaces

**MPLAB Harmony Help**

Framework Help > Driver Libraries Help > USART Driver Library > Library Interface

**MPLAB Harmony Help** [Contents](#) | [Index](#) | [Home](#) [Previous](#) | [Up](#) | [Next](#) [Documentation Feedback](#)  
[Microchip Support](#)

**Library Interface**

**b) Core Client Functions**

Name	Description
<a href="#">DRV_USART_Open</a>	Opens the specified USART driver instance and returns a handle to it. <b>Implementation:</b> Dynamic
<a href="#">DRV_USART_Close</a>	Closes an opened-instance of the USART driver. <b>Implementation:</b> Dynamic
<a href="#">DRV_USART_ClientStatus</a>	Gets the current client-specific status the USART driver. <b>Implementation:</b> Dynamic
<a href="#">DRV_USART_ErrorGet</a>	This function returns the error(if any) associated with the last client request. <b>Implementation:</b> Dynamic

**c) Communication Management Client Functions**

Name	Description
<a href="#">DRV_USART_BaudSet</a>	This function changes the USART module baud to the specified value. <b>Implementation:</b> Dynamic
<a href="#">DRV_USART_LineControlSet</a>	This function changes the USART module line control to the specified value. <b>Implementation:</b> Dynamic

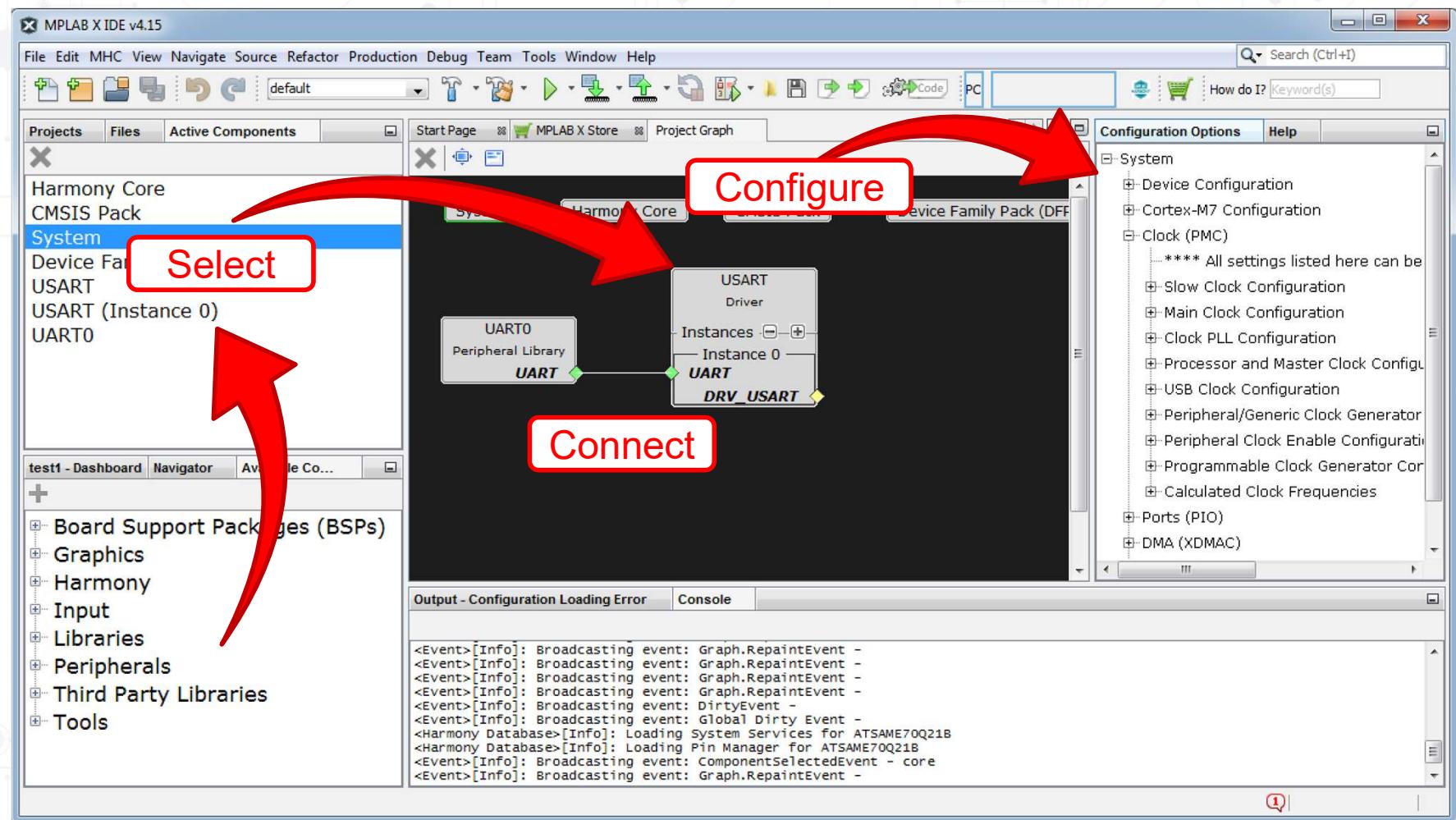
**d) Queue Read/Write Client Functions**

Name	Description
<a href="#">DRV_USART_BufferAddRead</a>	Schedule a non-blocking driver read operation. <b>Implementation:</b> Dynamic
<a href="#">DRV_USART_BufferAddWrite</a>	Schedule a non-blocking driver write operation. <b>Implementation:</b> Dynamic
<a href="#">DRV_USART_BufferEventHandlerSet</a>	Allows a client to identify a buffer event handling function for the driver to call back when queued

**USART Driver Library**

- + [Introduction](#)
- + [Using the Library](#)
- + [Configuring the Library](#)
- + [Building the Library](#)
- + [Library Interface](#)
- + [Files](#)

# Connect Driver to PLIB



# Driver-Based Project

## Drivers & Services

### main.c

```
int main ( void )
{
    SYS_Initialize(NULL);

    while(true)
    {
        SYS_Tasks();
    }

    return(EXIT_FAILURE)
}
```

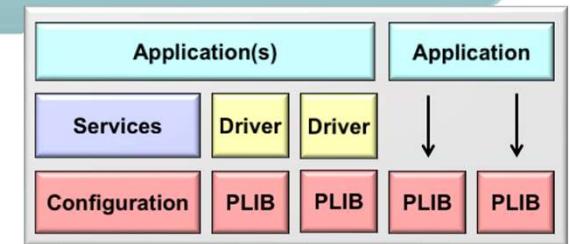
### Other Files Added

- definitions.h
- configuration.h
- tasks.c

### app.c

```
void APP_Tasks ( void )
{
    switch(app.state)
    {
        case APP_INIT:
            app.usart = DRV_USART_Open(DRV_USART_INDEX_0, 0);
            if (app.usart != DRV_HANDLE_INVALID)
            {
                app.state = APP_MESSAGE;
            }
            break;

        case APP_MESSAGE:
            if (DRV_USART_WriteBuffer(app.usart, message, sizeof(message)))
            {
                app.state = APP_RECEIVE;
            }
            break;
    }
}
```



# System Services

- **Common Functionality and Resources**
- **Avoid Duplicate Code and Conflicts**
- **Built on PLIBs, Drivers, or CPU Features**
- **Single-instance, Unified Usage Model**  
(No Open or Close Functions - Always “open”)
- **Otherwise, similar to drivers**
  - Simple, Abstracted Interfaces
  - Portable & Backwards-compatible
  - Interoperable
  - Configurable Options
  - Selectable Features
  - Advanced Capabilities

## Examples

- DMA
- Virtual File System
- Interrupts
- IO Ports
- Debug Console
- Command Console
- Time & Timers
- Random Number
- OSAL

# Middleware

- **Advanced SW Stacks & Capabilities**
- **Complex Protocol Support**
- **Built on Drivers & Services**
- **Otherwise, similar to drivers**
  - Simple, Abstracted Interfaces
  - Portable & Backwards-compatible
  - Interoperable
  - Configurable Options
  - Selectable Features
  - Advanced Capabilities

# Middleware

- HS/FS/LS
- Multi-Instance
- Device or Host
- Audio
- CDC
- HID
- MSD
- Hub

*USB*



- IPv4/IPv6
- Transport/App Layers
- Eth MAC, WiFi g
- IPv6 Gold Cert
- Dozens of Protocols: ARP, Sockets, DHCP, DNS, FTP, HTTP, ICMP, NBNS, NTP, PING, TFTP, UDS, VNC, WebSockets

*TCP/IP*



- Aria Graphics Library
- Dozens of Widgets
  - Simple, Compound
  - Graphs, Menus, Charts, Images
- Composer (GUI)
- Display Manager (GUI)
- Touch, Gestures
- Layers, Palettes, Fonts

*Graphics*



- Virtual/Multi-FS
- FAT32
- MPFS
- Multi-Device
  - Flash
  - SD Card
  - USB
  - Memory
  - Multi-Partition

*File System*



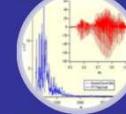
- Cryptographic Library
- Compression and Hash Libraries
  - MD5, HMAC
  - SHA1/256/384/512
- Embedded SSL Library (CyaSSL)
  - RNG, AES
  - ECC, RSA

*Embedded Security*



- 100+ functions DSP Library
- 29 functions Math Library
- Decoders: AAC, FLAC, Opus, Speex, WMA

*DSP/Math*



- Bluetooth Data Stack
- Bluetooth Audio, Voice
- BLE
- MFi
- DSP:
  - AEC
  - Filtering
  - Voice, Hi Res
  - AAC, QCPDEO

*Bluetooth/Audio*



# Middleware-Based Project

## Powerful Middleware

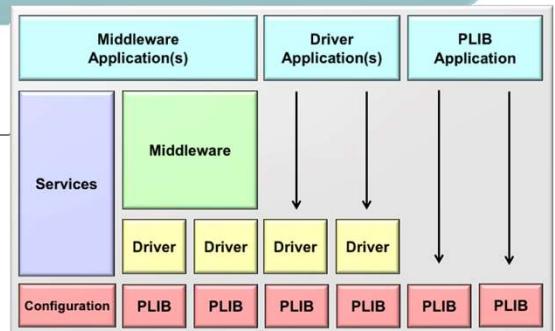
app.c

```

void APP_Tasks ( void )
{
    switch(appData.state)
    {
        case APP_STATE_INIT:
            appData.usbDevHandle = USB_DEVICE_Open(USB_DEVICE_INDEX_0, DRV_IO_INTENT_READWRITE);
            if(appData.usbDevHandle != USB_DEVICE_HANDLE_INVALID)
            {
                USB_DEVICE_EventHandlerSet(appData.usbDevHandle, APP_USBDDeviceEventHandler, 0);
                appData.state = APP_STATE_WAIT_FOR_CONFIGURATION;
            }
            break;

        case APP_STATE_WAIT_FOR_CONFIGURATION:
            if(appData.deviceConfigured == true)
            {
                appData.hidDataReceived = false;
                USB_DEVICE_HID_ReportReceive(USB_DEVICE_HID_INDEX_0, &appData.rxTransferHandle,
                                            appData.receiveDataBuffer, 64);
            }
            break;
    ...
}

```



# Lab2:

## Create an application that enumerates a USB CDC device and demonstrates two-way communication between the USB Device and the USB Host PC

# Lab 2: Objectives

- Be able to configure and use Harmony USB Device stack and CDC Function driver

# Lab 2: Create USB Device Application

- Github: <https://github.com/Microchip-MPLAB-Harmony/usb/wiki/Create-your-first-usb-device-cdc-single-application>
- Follow section “Create your first usb device cdc single application”
- Note:
  - Under section “Adding USB Device Components”, instead of component “USB High Speed Driver”, component “USB Full speed driver” would be added as SAM D21 has USB Full speed (not High Speed).

# Lab 2: Summary

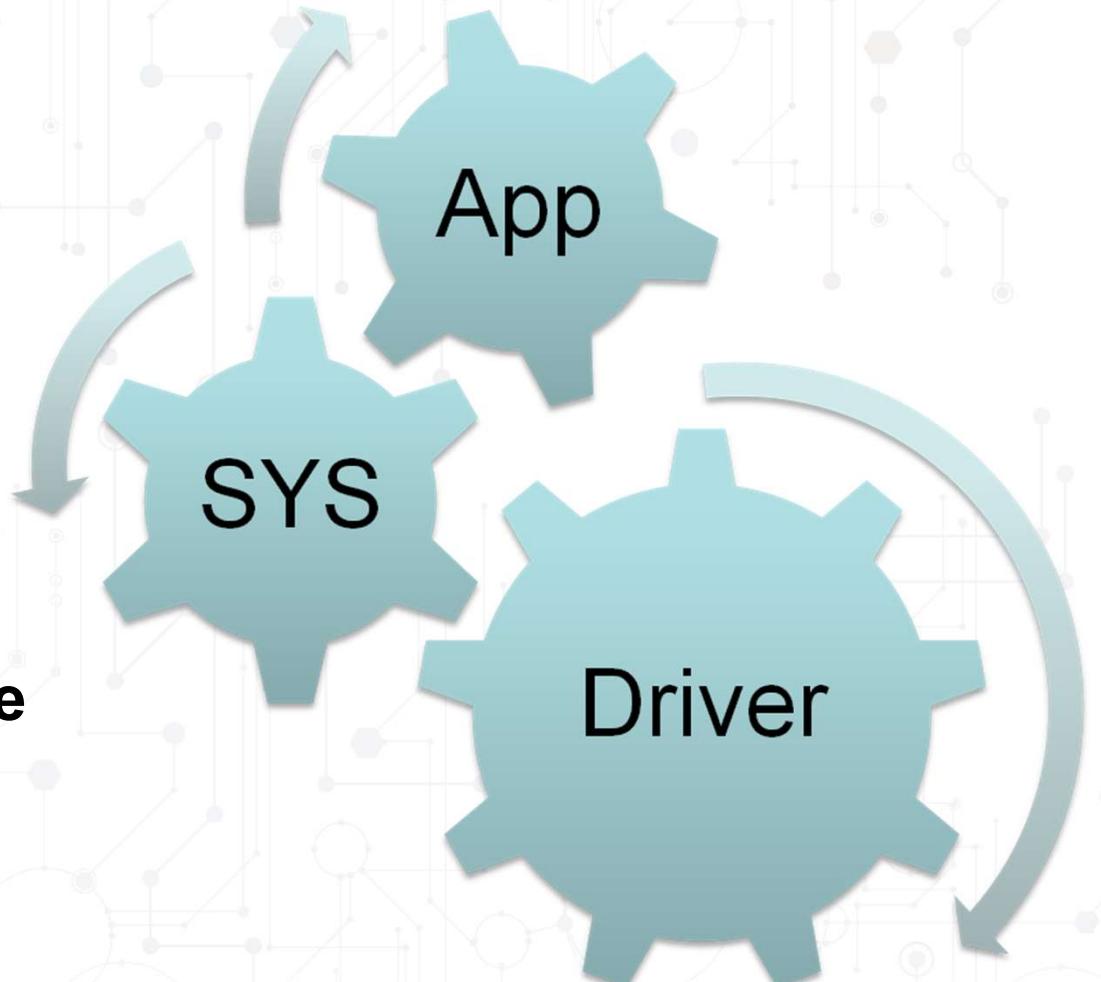
- In this lab we have...
- Successfully created USB CDC Harmony project using MHC
- Configured and used USB Device stack and CDC Function driver

# Agenda

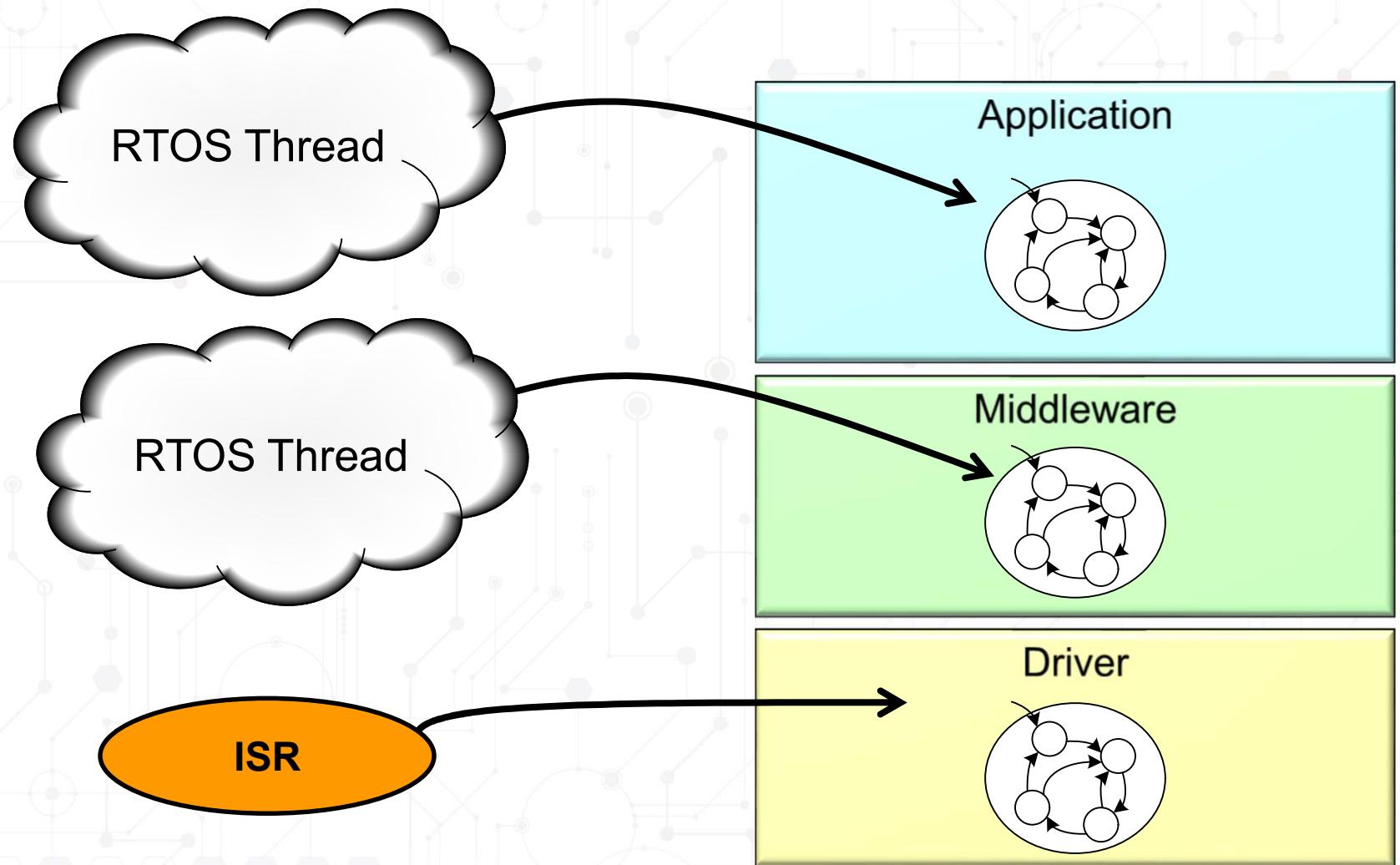
- 1. MPLAB Harmony 3 Introduction**
- 2. Overview of changes from MPLAB Harmony 2**
- 3. Introduction to the downloader and to the modular library packages**
- 4. Fundamentals of applications, projects, configuration, and peripheral libraries**
  - Lab1: Create a simple “heartbeat” LED application that flashes an LED using peripheral libraries
- 5. Introduction to drivers, services, and middleware**
  - Lab2: Create an application that enumerates a USB CDC device and demonstrates two-way communication between the USB Device and the USB Host PC
- 6. Interoperability and RTOS support**
- 7. Review of development models**

# Interoperability

- **Library modules maintain their own internal state.**
- **Execute tasks in their own threads (or interrupts).**
- **Cooperate through interface functions.**
- **No RTOS-specific code in library modules.**

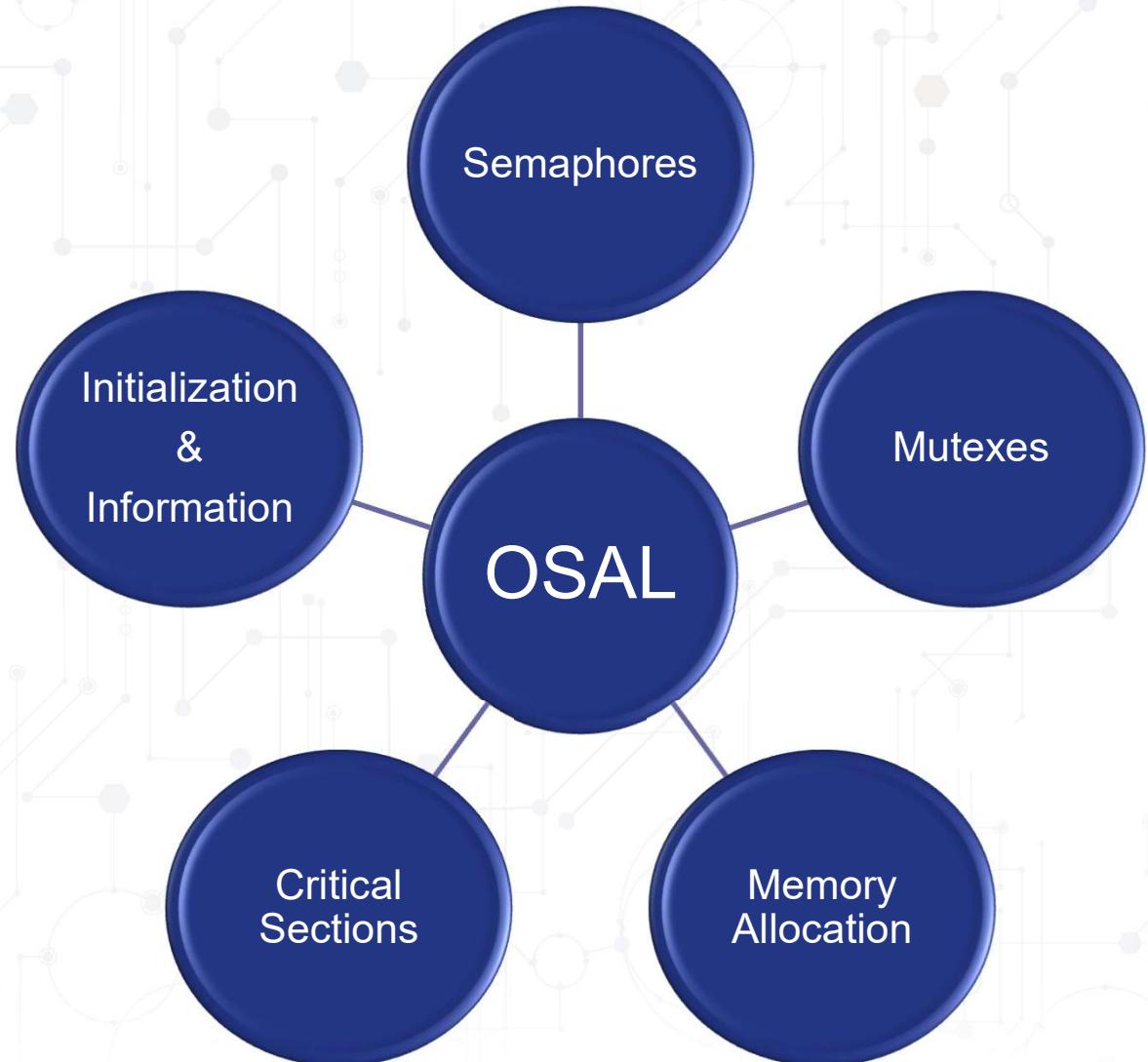


# RTOS Scheduler

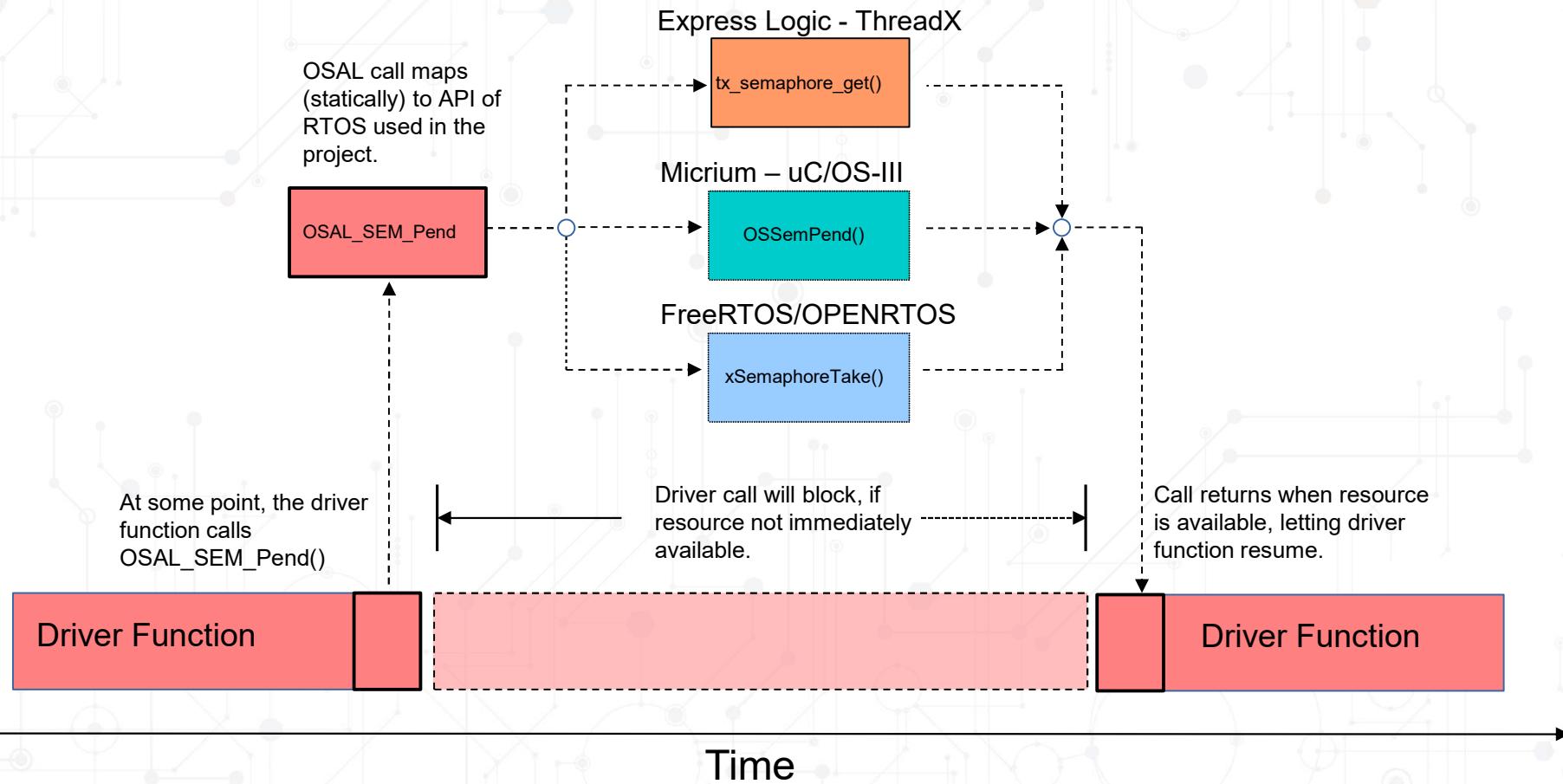


# OS Abstraction Layer

- **Multi-Threading**
  - Thread Safety
  - Synchronization
- **RTOS Compatibility**
- **OS Agnostic**  
(Or “Bare Metal”)
- **Light-weight & Minimal**



# OSAL-to-RTOS Mapping



# RTOS-Based Project

## Real-Time OS

### tasks.c

```

void _APP1_Tasks( void *pvParameters )
{
    while(true)
    {
        APP1_Tasks();
    }
}

void _APP2_Tasks( void *pvParameters )
{
    while(true)
    {
        APP2_Tasks();
    }
}

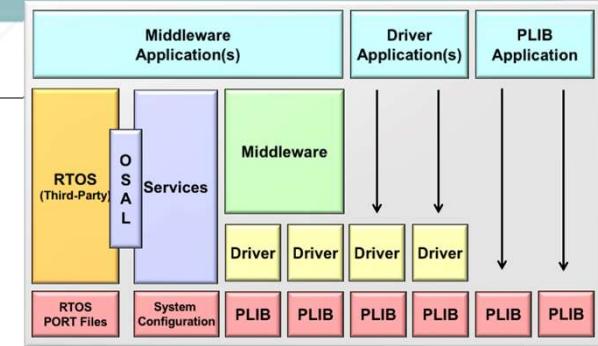
void SYS_Tasks ( void )
{
    xTaskCreate((TaskFunction_t) _APP1_Tasks, "APP1_Tasks", 1024, NULL, 2, NULL);
    xTaskCreate((TaskFunction_t) _APP2_Tasks, "APP2_Tasks", 1024, NULL, 2, NULL);

    vTaskStartScheduler(); /* This function never returns. */
}
...

```

Thread Functions

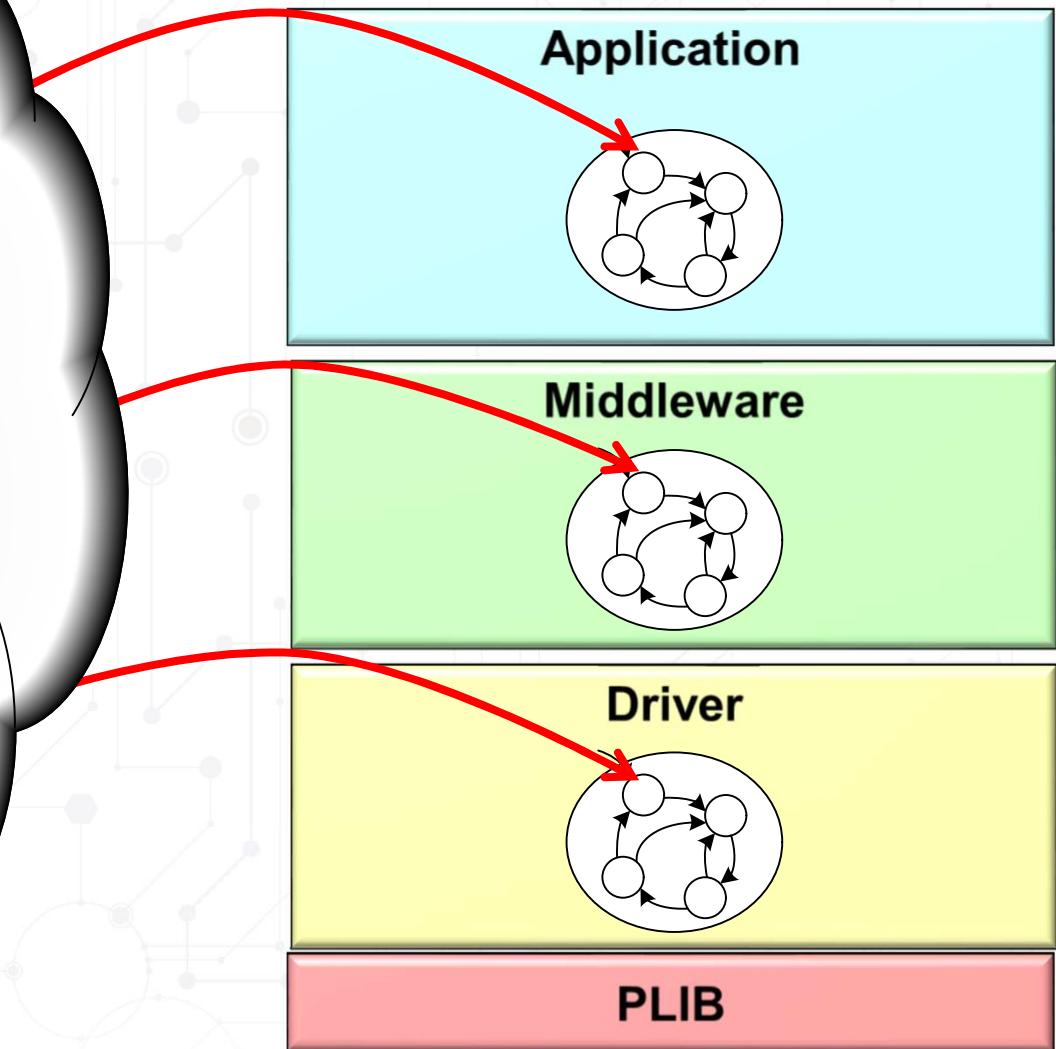
FreeRTOS Specific



# Modules Operate Independently

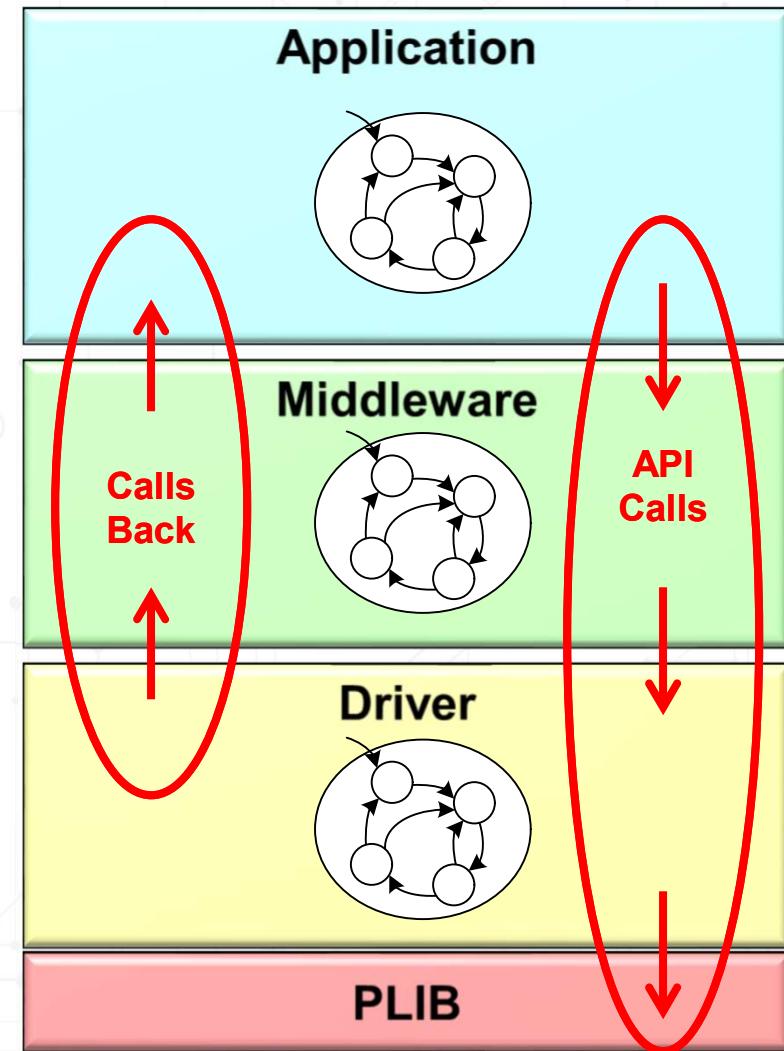
**Effectively  
Independent**

**No  
Assumptions of  
Sequence or  
Timing Between  
modules**



# Modules Cooperate Interactively

**Modules Interact  
With Each Other  
Only Through  
Interface  
Functions**



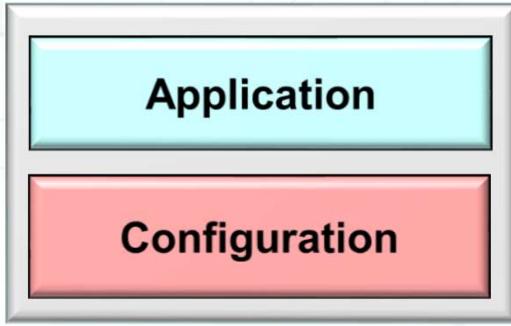
# Agenda

- 1. MPLAB Harmony 3 Introduction**
- 2. Overview of changes from MPLAB Harmony 2**
- 3. Introduction to the downloader and to the modular library packages**
- 4. Fundamentals of applications, projects, configuration, and peripheral libraries**
  - Lab1: Create a simple “heartbeat” LED application that flashes an LED using peripheral libraries
- 5. Introduction to drivers, services, and middleware**
  - Lab2: Create an application that enumerates a USB CDC device and demonstrates two-way communication between the USB Device and the USB Host PC
- 6. Interoperability and RTOS support**
- 7. Review of development models**

# Minimal Initialization

## Device Configuration

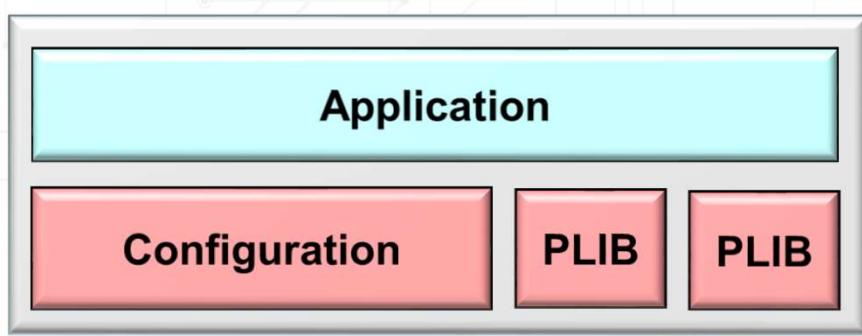
**MPLAB® Harmony Configurator (MHC)**



# PLIB-based Project

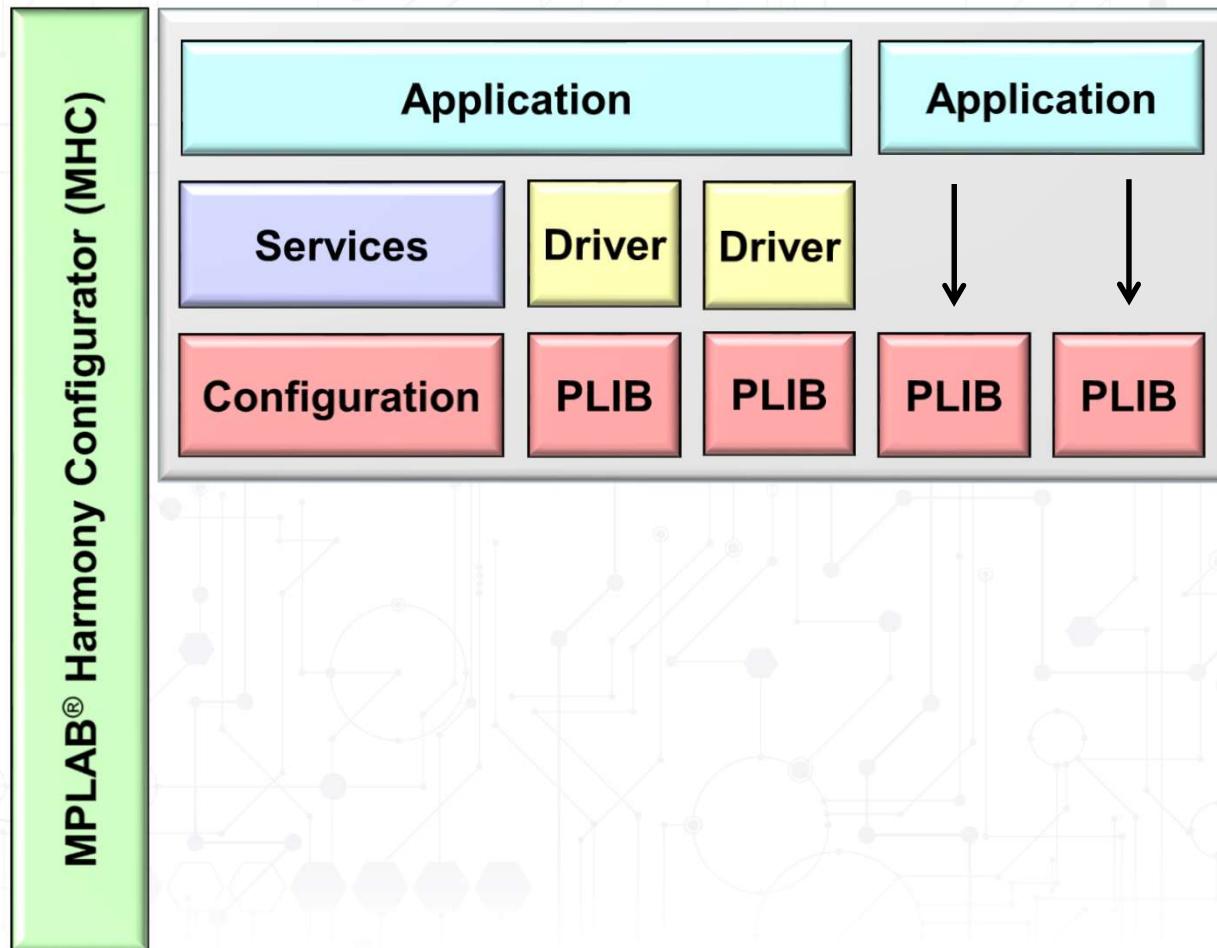
## Peripheral Libraries

**MPLAB® Harmony Configurator (MHC)**



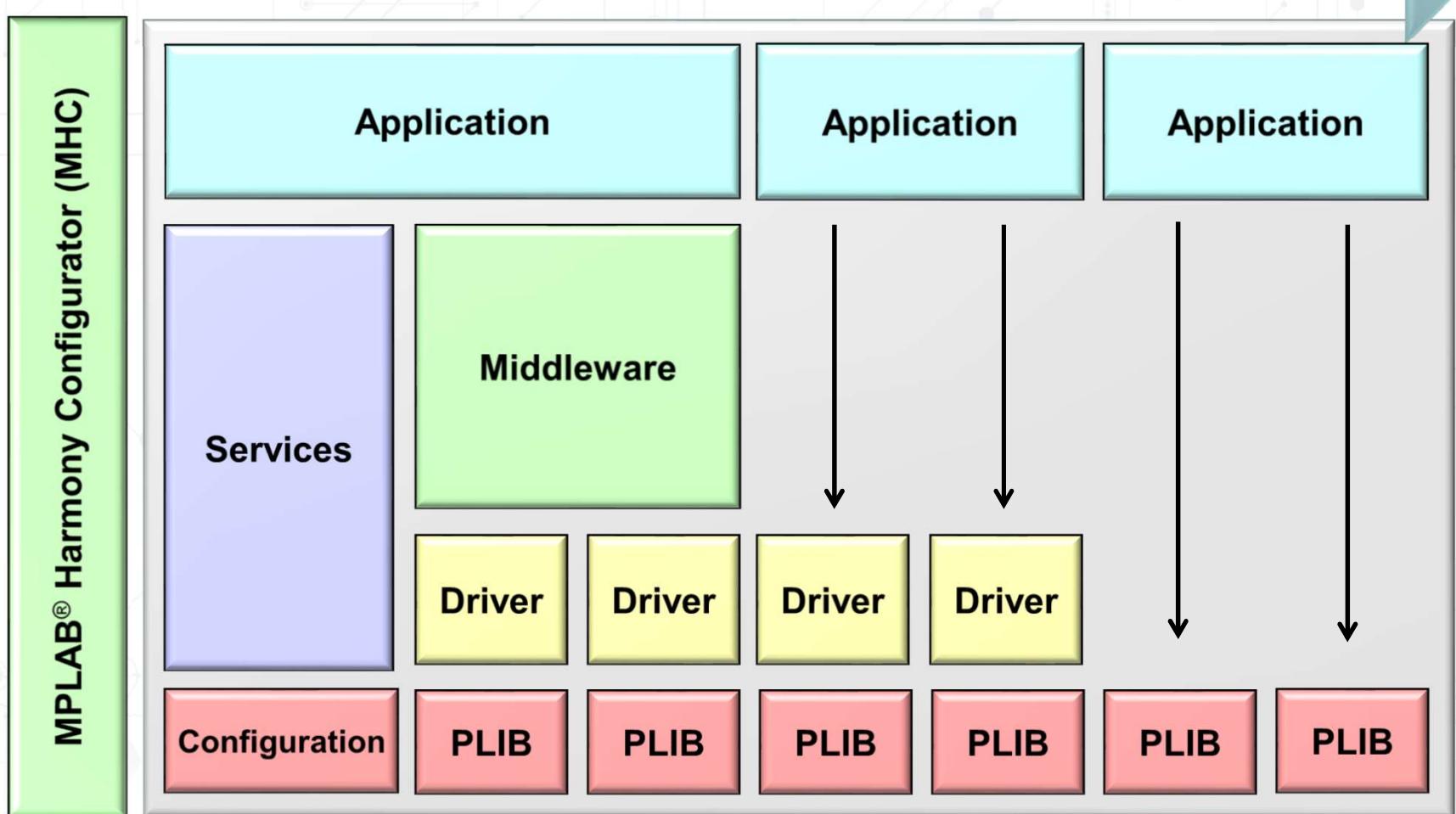
# Driver-Based Project

## Drivers & Services



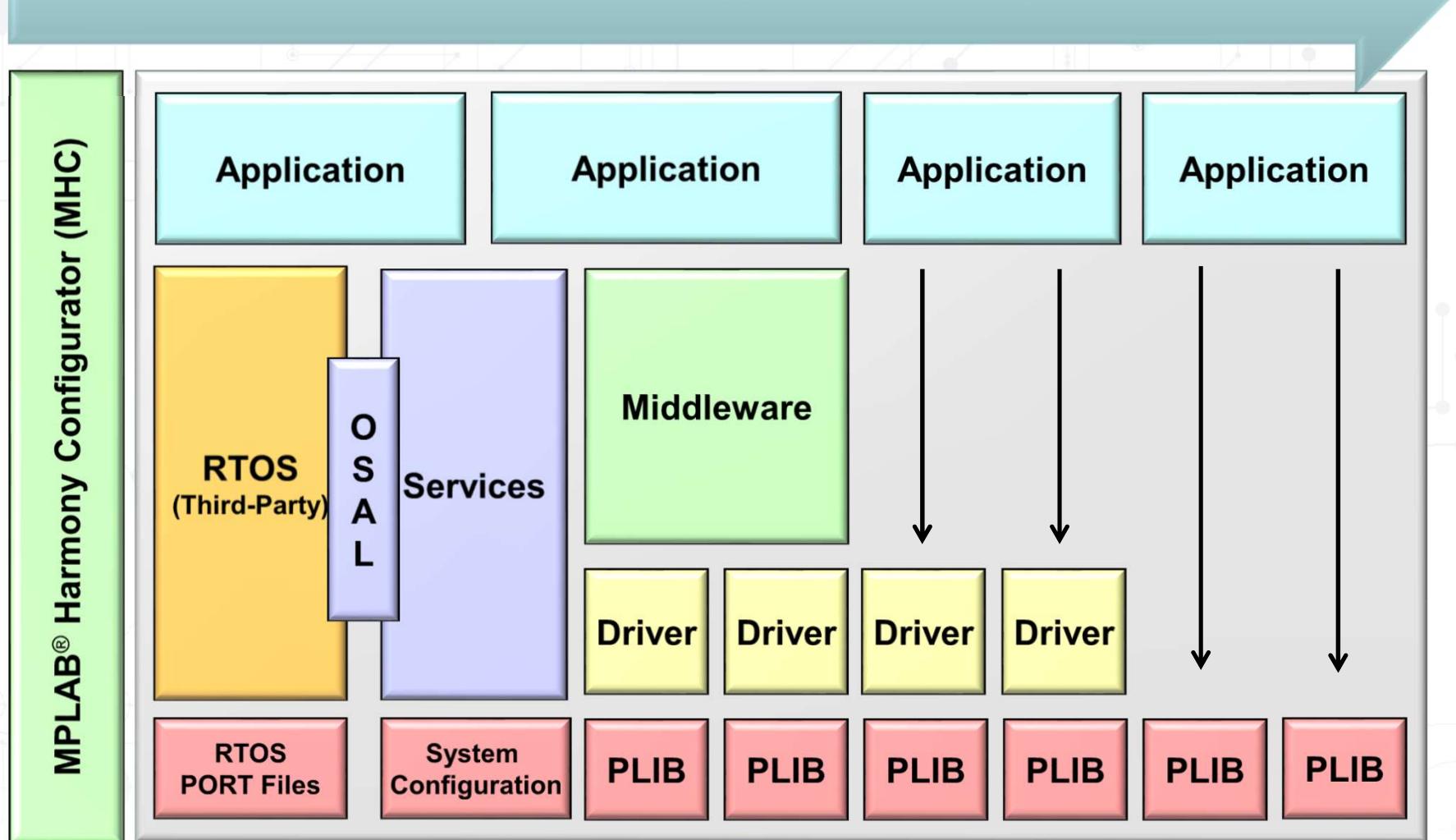
# Middleware-Based Project

## Powerful Middleware

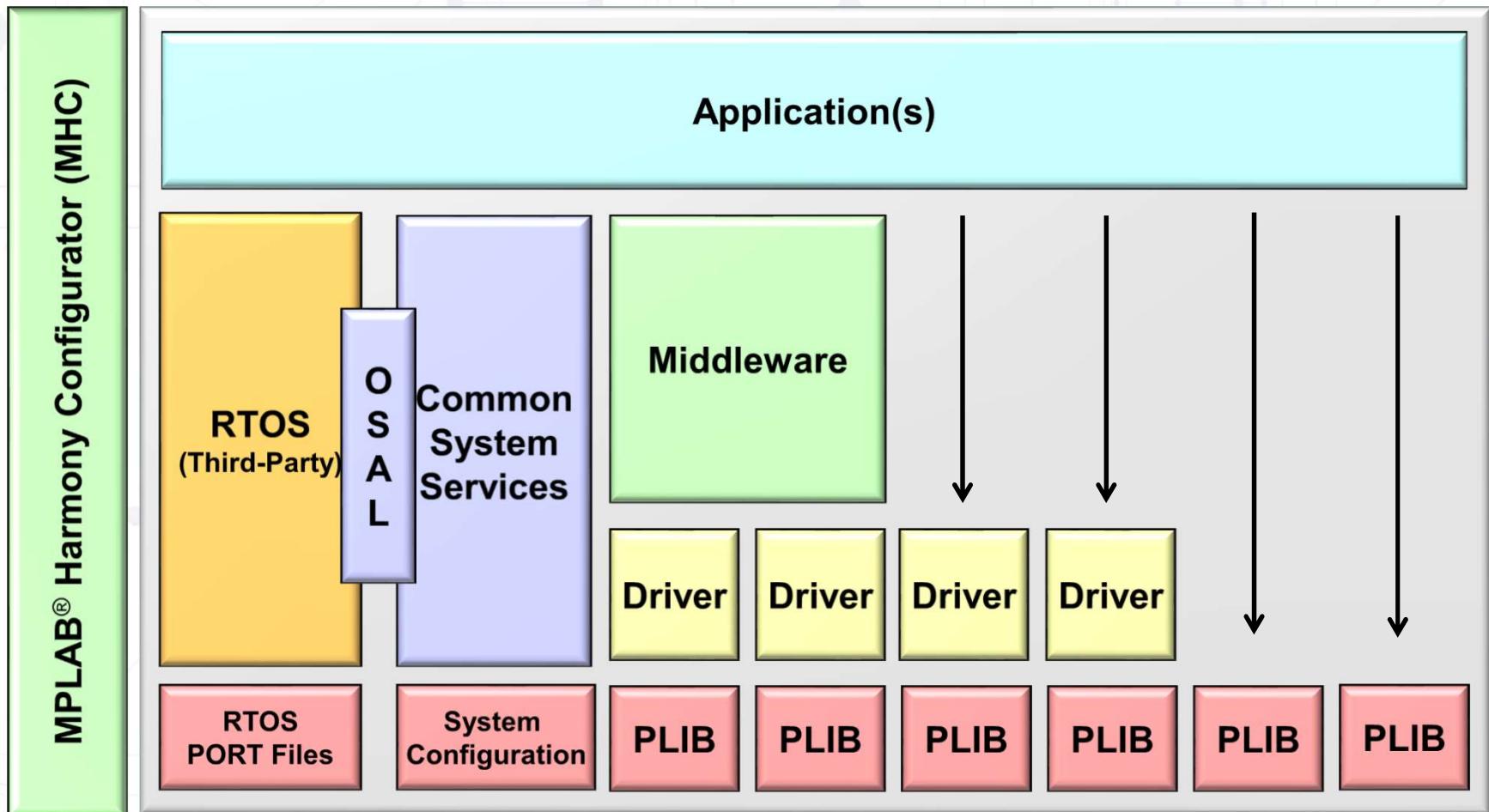


# RTOS-Based Project

## Real-Time OS



# Embedded Software Model



# Development Models

## Device Configuration

### Initialize Device

- Clocks
- Pins
- Memory
- Vectors
- Startup Code
- Essentials...

### **Peripheral Libraries**

#### Use Peripherals

- Custom Initialization
- Clean & direct
- MCC-Like
- Low overhead
- Custom generated

### **Drivers & Services**

#### Share Resources & Peripherals

- Built on PLIBs
- Interoperable
- Simple interfaces
- Advanced capabilities

### **Powerful Middleware**

#### Add MW Stacks

- Graphics
- TCP/IP Networking
- USB Host & Device
- Cryptography
- More...

### **Real-Time OS**

#### Optimize CPU Usage

- FreeRTOS Included
- OS Abstraction Layer (OSAL)
- Preemptive scheduling
- Task prioritization
- More...

**Advanced capabilities  
are built on  
fundamental support.**

**Choose the development model that best serves the applications needs!**



# Main Website

**MPLAB® Harmony - Integrated Embedded Software Development Framework**

**Beta Release of MPLAB® Harmony v3 - Integrated Embedded Software Development Framework**

**Beta Release Includes Support for SAM MCUs and Faster Development Options**

Users of MPLAB® Harmony, a flexible, abstracted, fully integrated unified embedded software development framework for 32-bit microcontrollers, are now able to use the beta version of the framework to support the first two families of Microchip's new SAM Xplained boards. This marks the first time that users can take advantage of the award-winning MPLAB development environment for the first time. The SAM E7, SAM V7X and SAM S21 microcontroller families will be the first SAM families supported, with additional SAM and PIC® families being added over the coming months.

In addition to supporting SAM microcontrollers, this beta release also provides access to highly-simplified peripheral libraries, simplified device drivers and modular software downloads which remove much of the development work from configuring your MCU. The code development format allows for maximum reuse and reduces time to market.

To provide additional flexibility to the programming experience, MPLAB Harmony can now be used as a stand-alone tool, outside of MPLAB X IDE. It is now possible to start the development of new project in your native development environment by downloading MPLAB Harmony as a stand-alone tool from GitHub's repository.

**MPLAB Harmony Features**

- \* Code Interoperability
- \* Modular architecture allows drivers and libraries
- \* Faster Time to Market
- \* Integrated single platform enables shorter development cycles
- \* Improved Compatibility
- \* Scalable across PIC® 32-bit MCUs to custom fit
- \* Quicker Support
- \* One-stop support for all customer needs including technical support, developer forums, and community
- \* Easy third-party software integration
- \* Integrates third-party solutions (RTOS, middleware, development tools, etc.)

**Features**

- Support for 32-bit SAM microcontroller devices
  - Enhanced MPLAB Harmony Configurator (MHC)
  - Board Support Package (BSP) updates for SAM Xplained boards
  - Support for SAM device startup, interrupt and exception code
  - Support for SAM device drivers
  - Support for SAM peripheral libraries
  - RTOS-based projects for optimum Central Processing Unit (CPU) utilization
- Improved user experience
  - Enhanced download manager simplifies installation
  - Application browser enables easier search of demo examples
  - RTOS enabled by default for a simpler programming model

**Get MPLAB Harmony v3 Beta**

Content is maintained in a set of GIT repositories on [github.com/Microchip-MPLAB-Harmony](https://github.com/Microchip-MPLAB-Harmony) that can be downloaded (or cloned) to your computer. Use the MPLAB Harmony v3 Downloader to obtain MPLAB Harmony v3 content.

If you already have the free, downloadable MPLAB X IDE installed, all you have to do is select the MPLAB Harmony v3 download from the extensive plugin library. If you don't already have the MPLAB X IDE on your machine, you can get it from the buttons below. After downloading MPLAB X IDE, follow the four steps below.

If you already have the free, downloadable MPLAB X IDE installed, all you have to do is select the MPLAB Harmony v3 download from the extensive plugin library. If you don't already have the MPLAB X IDE on your machine, you can get it from the buttons below. After downloading MPLAB X IDE, follow the four steps below.

[Download MPLAB X IDE for Windows OS](#)   [Download MPLAB X IDE for Linux OS](#)   [Download MPLAB X IDE for Mac OS](#)

The MPLAB Harmony embedded software development versions please read "Release Notes". Add:

Step 1: Open MPLAB X IDE.  
 Step 2: Select Tools > Plugins > Download and click Go to MPLAB X Plugin Manager.  
 Step 3: Check the install box next to MPLAB Harmony Configurator 3, click Install and follow the directions given by the Plugin Installer.  
 Step 4: From within the MPLAB X IDE, select Tools > Embedded > MPLAB Harmony v3 Framework Downloader to open the downloader plug-in.

## Harmony Landing Page(s)

**No H3 Code**  
**(H1 & H2 Archives)**

[www.microchip.com/harmony](http://www.microchip.com/harmony)

- Marketing Collateral
- Access Directions

**MICROCHIP** **MPLAB HARMONY**

**Integrated Software Framework v2.05**

**STANDARD FEATURES**

- MPLAB® Harmony is a flexible, abstracted, fully integrated framework development platform for PIC32 microcontrollers
- Broad range of Middleware Stack Libraries, including: USB, TCP/IP, WiFi™, File System, Graphics, Broadcasters, Database, Cryptography, Sensors, Cryptography, Drivers, System Services, and more.
- Over 100 Application Demonstrations with up to 600 configurations to accommodate application development
- Seamlessly integrates third-party solutions (RTOS, RTOS drivers, middleware, development tools, etc.)
- RTOS support, which includes: FreeRTOS, RTX, Express Logic ThreadX, QNX, Green Hills MicroCOS™, µC/OS-II™, Micrium (μCOS-II™)
- Middleware support, which includes: I2C/SPI, SPI, CAN, Inter-IC Technology, USART, serial, and Parallel
- Both free and enabling license terms provided

For a detailed list of features, please visit the MPLAB Harmony Web page at: [www.microchip.com/harmony](http://www.microchip.com/harmony)

From the landing page, scroll down and select the Features tab.

**COMPLIANCE**

Compliant with MISRA-C:2012 Mandatory Standards:

- MPLAB Harmony Peripheral Libraries
- TCP/IP Library

**DEVELOPMENT TOOLS**

- MPLAB X IDE v4.0 or earlier is required
- MPLAB XC32 C/C++ Compiler v1.44 (DDO 20262) with v1.44B Patch Support Patch
- MPLAB XC8 C8 Compiler
- MPLAB Harmony Configurator (MHC) v3.0.5.2

**THIRD-PARTY DEVELOPERS**

Microchip offers a range of documentation to assist you in using the design of your own software modules for MPLAB Harmony. These documents, which are provided with the installation, are also available for download from the MPLAB Harmony website (see "Download Information" for details).

- MPLAB Harmony Overview
- MPLAB Harmony Compatibility Guide
- MPLAB Harmony Creating Your First Application Tutorial
- MPLAB Harmony Application Development Guide
- MPLAB Harmony Application User's Guide
- MPLAB Harmony Configurator Developer's Guide
- MPLAB Harmony Graphics Composer User's Guide
- MPLAB Harmony Test Harness User's Guide
- MPLAB Harmony Compatibility Checklist Worksheet

## Harmony Brochure



**Other**

# Instructional

## MPLAB® Harmony 3 User's Guide

This document describes what MPLAB Harmony 3 is, explains its key architectural concepts, and provides instructions for its use.

### What is MPLAB Harmony 3?

MPLAB® Harmony 3 is an extension of the MPLAB® ecosystem for developing embedded software solutions for Microchip 32-bit devices. It is comprised of a set of tools, libraries, and example applications that extend the MPLAB® ecosystem to simplify development of embedded software for Microchip® 32-bit SAM and PIC microcontroller and microprocessor devices. MPLAB Harmony 3 provides the MPLAB® Harmony Configurator (MHC) tool, a set of modular device and middleware libraries, and numerous example applications, all of which are designed to help developers to quickly and easily develop powerful and efficient embedded software for Microchip 32-bit SAM and PIC devices.

### MPLAB Harmony 3 Provides

- Example Applications
- Modular Libraries
  - o Peripheral Libraries
  - o Drivers & Services
  - o Middleware
- Graphical Developer Tools for downloading, configuring, and generating the libraries.



The MHC is an easy to use development tool with a Graphical User Interface (GUI) that simplifies device setup, library selection and configuration, and application development. The MHC is available as a plugin that directly integrates with the MPLAB® X IDE and as a separate Java executable for standalone use with other development environments. The included MHC downloader reads an online catalog of library packages and facilitates selection and downloading of any libraries in which the developer is interested. The configurator provides convenient and powerful development tools for choosing library components from downloaded packages and configuring them for the developer's application. And, the built-in code generator produces library and application starter code (usually in source form), based on the options chosen by the developer.

The library packages provided by MPLAB Harmony 3 are distributed in separate GIT repositories containing C-language source-code (and/or templates for generating it) for components that are normally used together or that are parts of a "stack" of related library components. The Chip Support Package (CSP) contains device startup code and independent low-level Peripheral Libraries (PLIBs) that consist of simple functions to initialize and control peripherals and basic device features. The Core package provides device drivers and system service libraries that use PLIBs and that abstract hardware and Real-time Operating System (RTOS) details away from middleware and applications. Middleware libraries use drivers and system services for device



[Microchip MPLAB Harmony](#)  
[MPLAB Harmony 3 Wiki](#)  
[Create Your First Peripheral Library Project](#)  
[Get Started with Harmony 3 on the SAM3D2](#)

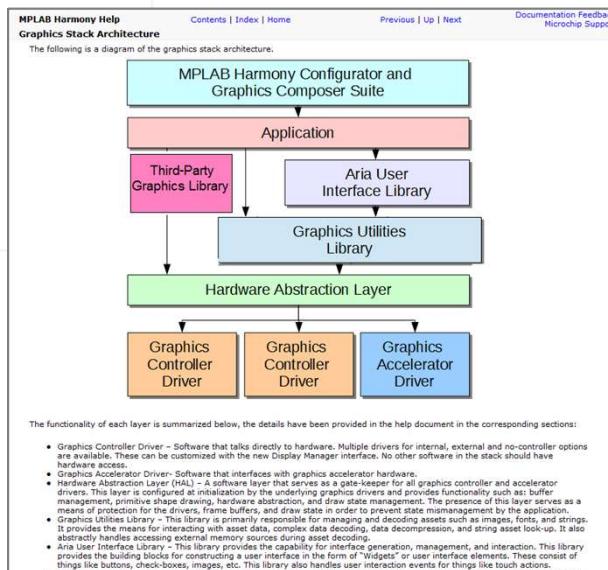
[microchip-mplab-harmony.github.io](http://microchip-mplab-harmony.github.io)

## Main Harmony 3 Introductory Material

- General development guides
- General training material,
- Links to other resources

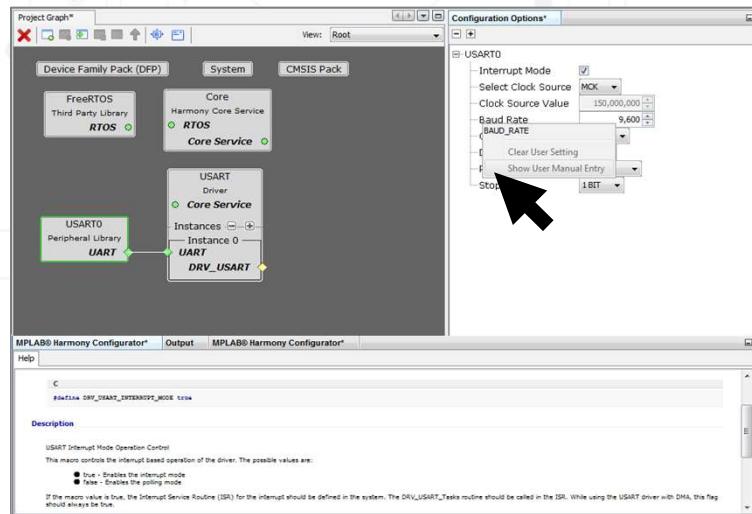
## Module-Specific WiKi Introductory Material

- Development guides, training material, & references to other resources.
- Module-specific version of main pages site.



[microchip-mplab-harmony.github.io/gfx](http://microchip-mplab-harmony.github.io/gfx)

# Reference Interactive & Online



## Module-Specific Installed & Interactive Configuration & API Help (HTML + Python)

- Generated from source code by Doc-O-Matic
- Rendered by MHC & Python
- Same HTML files as GitHub Pages.

## Module-Specific Rendered Online Automatically Configuration & API Help

- Versioned with source code.
- Rendered from <repo>/docs/html folder.
- Same HTML files as MHC interactive help.

The screenshot shows the "Driver Libraries Help" page for MPLAB Harmony 3 Core. It features a sidebar with links to various driver libraries: Driver Library Overview, AT24 Driver Library Help, AT25 Driver Library Help, I2C Driver Library Help, I2S Driver Library Help, Memory Driver Library Help, SD Card Host Controller Library, SD Card (SPI) Driver Library, SPI Driver Library Help, SST26 Driver Library Help, USART Driver Library Help, System Service Libraries Help, Command Processor System Service Library Help, Console System Service Library Help, Debug System Service Library Help, Direct Memory Access (DMA) System Service Library Help, File System Service Library Help, Interrupt System Service Library Help, Ports System Service Library Help, Time System Service Library Help, OSAL Library Help, and Applications Help. The main content area displays the "Driver Library Overview" section, which provides an overview of the driver usage and specific details for each driver library.

[microchip-mplab-harmony.github.io/core](https://microchip-mplab-harmony.github.io/core)

# Reference Downloadable

**Module specific  
Downloadable from GitHub  
Configuration & API Help  
(CHM, PDF)**

Generated by same Doc-O-Matic project as interactive  
help and GitHub Pages

Crypto Library Help      Library Interface      c) MDS Functions

**Function**  
`int CRYPT_HUFFMAN_Compress(unsigned char* out, unsigned int outSz, const unsigned char* in, unsigned int inSz, unsigned int flags)`

**CRYPT\_HUFFMAN\_DeCompress Function**  
 Decompresses a block of data.

**File**  
`crypto.h`

**C**  
`int CRYPT_HUFFMAN_DeCompress(unsigned char*, unsigned int, const unsigned char*, unsigned int);`

**Returns**  
 • negative - Error code  
 • positive - Bytes stored in out buffer

**Description**  
 This function decompresses a block of data using Huffman encoding.

**Remarks**  
 Output buffer must be large enough to hold the contents of the operation.

**Preconditions**  
 None.

**Example**  
`unsigned char cbuffer[1024];  
 unsigned char dbuffer[1024];  
 int ret;  
 ret = CRYPT_HUFFMAN_DeCompress(dbuffer, sizeof(dbuffer), cbuffer, maxlen);`

**Parameters**

Parameters	Description
out	Pointer to destination buffer
outSz	Size of destination buffer
in	Pointer to source buffer to decompress
inSz	Size of source buffer to decompress

**Function**  
`int CRYPT_HUFFMAN_DeCompress(unsigned char* out, unsigned int outSz, const unsigned char* in, unsigned int inSz)`

**c) MDS Functions**

**CRYPT\_MDS\_DataAdd Function**  
 Updates the hash with the data provided.

© 2013-2018 Microchip Technology Inc.      MPLAB Harmony v3.00      13

MPLAB Harmony Help

Contents | Index | Search | Favorites |

Volume I: Getting Started With MPLAB Harmony Libraries and Application Frameworks  
 Volume II: Supporting Hardware  
 Volume III: MPLAB Harmony Configurator (MHC)  
 Volume IV: MPLAB Harmony Development  
 Volume V: MPLAB Harmony Framework Reference

Framework Overview  
 Bluetooth Stack Library Help  
 Class B Library Help  
 Class E Library Help  
 Crypto Library Help

Introduction  
 Getting the Library  
 Configuring the Library  
 Building the Library

Library Interface

General Functions  
 a) Compression Functions  
 b) CRYPTO\_HUFFMAN\_Compress Function  
 C) MDS Functions  
 d) Random Number Generation Functions  
 e) ECC Encryption/Decryption Functions  
 f) RSA Encryption/Decryption Functions  
 g) SHA-256 Hash Functions  
 h) Triple DES (3DES) Encryption/Decryption Functions  
 i) HMAC Hash Functions  
 j) SHA-384 Hash Functions  
 k) SHA-512 Hash Functions  
 m) SHA512 Hash Functions  
 n) Data Types and Constants

Decoder Libraries Help  
 Driver Libraries Help  
 Encoder Libraries Help  
 Main Libraries Help  
 Main Library Help  
 Networking Presentation Layer Help  
 OSAL Library Help  
 Peripheral Libraries Help  
 Sample Application Service Libraries Help  
 TCP/IP Stack Libraries Help  
 Test Libraries Help  
 USB Libraries Help

Volume VI: Third-Party Products  
 Volume VII: Utilities

Volume V: MPLAB Harmony Framework Reference > Crypto Library Help > Library Interface > b) Compression Functions > CRYPT\_HUFFMAN\_DeCompress Function

**MPLAB Harmony Help**      **Contents** | **Index** | **Home**      **Previous** | **Up** | **Next**      **Documentation Feedback**  
**CRYPT\_HUFFMAN\_DeCompress Function**

**C**

```
int CRYPT_HUFFMAN_DeCompress(  

    unsigned char*,  

    unsigned int,  

    const unsigned char*,  

    unsigned int  

);
```

**Description**  
 This function decompresses a block of data using Huffman encoding.

**Preconditions**  
 None.

**Parameters**

Parameters	Description
out	Pointer to destination buffer
outSz	Size of destination buffer
in	Pointer to source buffer to decompress
inSz	Size of source buffer to decompress

**Returns**  
 • negative - Error code  
 • positive - Bytes stored in out buffer

**Remarks**  
 Output buffer must be large enough to hold the contents of the operation.

**Example**

```
unsigned char cbuffer[1024];  

    unsigned char dbuffer[1024];  

    int ret;  

    ret = CRYPT_HUFFMAN_DeCompress(dbuffer, sizeof(dbuffer), cbuffer, maxlen);
```

Volume V: MPLAB Harmony Framework Reference > Crypto Library Help > Library Interface > b) Compression Functions > CRYPT\_HUFFMAN\_DeCompress Function

MPLAB Harmony Help  
 Contents | Index | Home  
 Documentation Feedback  
 Microchip Support

Included in "docs" folder of each repository.



# Directory Online & Downloaded

The screenshot shows the GitHub repository page for Microchip-MPLAB-Harmony. At the top, there are navigation links for Code, Pull requests, Wiki, Insights, and Settings. Below the header, the repository name is displayed along with its GitHub URL. A "Home" section follows, showing a recent edit by BudCaldwell. On the left, there's a sidebar with the MPLAB Harmony logo and a "Welcome to the Microchip® MPLAB® Harmony 3 wiki!" message. This message explains that MPLAB Harmony 3 is an extension of the MPLAB ecosystem for creating embedded firmware solutions for 32-bit Microchip devices. It also provides instructions for installing the MPLAB X IDE and the MPLAB Harmony 3 Configurator (MHC) plugin.

[github.com/\[Microchip-MPLAB-Harmony\]².github.io/wiki](https://github.com/Microchip-MPLAB-Harmony/wiki)

## Main Harmony 3 WiKi

- Links to main page and other resources, support channels, how to contribute, roadmaps, and announcements.
- Basically a “readme” file for H3 overall.

## Module-Specific Readme File

- Simple format, displayed by GitHub source files.
- Provide brief module abstract.
- Provide links to instructions, tools, and documentation.
- Link to release notes (also displayed by GitHub).

The screenshot shows the GitHub repository page for aethanell/ExperimentalCore-sam. The repository has 12 stars and 8 forks. It contains 267 commits, 12 branches, 2 releases, and 3 contributors. The repository is licensed under LGPL-2.1. The README.md file describes it as an experimental new core for Arduino Due and other Atmel SAM boards. It includes links to .gitignore, .travis.yml, LICENSE, Makefile, and README.md. Below the repository details, there's a section for "Where to download" with a link to the releases page.



# Doc Summary

## Microhip.com/Harmony

[Release Notes for MPLAB® Harmony v3 – Integrated Embedded Software Development Framework](#)

[Release Notes for MPLAB® Harmony v3 – Device Family Support](#)

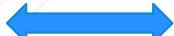
[Release Notes for MPLAB® Harmony v3 – Device Family Support](#)

[Device and Hardware Support](#)

[H3 Device Support](#)

[MPLAB® Harmony 3 Device Family Support](#)

[Welcome Page](#)



## GitHub H3 Wiki

### H3 Device Support

[MPLAB® Harmony 3 Device Family Support](#)

The following tables summarize which Microchip Harmony 3 device families have supported Microchip 32-bit device families. Refer to the read-me release notes for each individual repository for additional details.

[Device and Hardware Support](#)

[Welcome Page](#)



## GitHub H3 Pages (Instructional)

[MPLAB® Harmony 3 User's Guide](#)

This document describes what MPLAB Harmony 3 is, explains its key architectural concepts, and provides instructions for its use.

[What is MPLAB Harmony 3?](#)

MPLAB Harmony 3 is an extension of the MPLAB ecosystem for developing embedded software solutions for Microchip 32-bit devices. It is comprised of a set of tools, libraries, and frameworks designed to facilitate the development of real-time operating systems (RTOS), middleware, and various applications, all of which are designed to help developers quickly develop powerful and efficient embedded software for Microchip 32-bit SAM and PIC microcontrollers.

[MPLAB Harmony 3 Modules](#)

The HMC is an easy to use development library. A graphical user interface (GUI) lets identifies device drivers, middleware, and configuration and application development. The HMC also provides a plug-in that directly integrates with the MPLAB IDE as a separate Java application. The HMC is a graphical user interface that allows users to select a device and download read an online catalog of library packages and facilitates selection and download of any libraries for which the developer is interested. The configurator provides comprehensive support for the developer to select the required library components, download packages and configuring them for the developer's application. And, the build-in code generator automatically generates the source code for the developer's application based on the choices chosen by the developer.

The library packages provided by MPLAB Harmony, are distributed in separate ZIP repositories or single header files (and/or template files for generating C++ components) that are normally used together and are parts of a "stack" of related library components. The C/C++ libraries are the most common type of library package. These are standard C/C++ libraries that consist of simple functions to initialize and control peripherals and basic drivers. The Graphics Utilities Library is a collection of utility functions that provide the building blocks and Real Time Operating System (RTOS) details away from midware and applications. Midware libraries use drivers and system services for device

## Repo Info

### Repo GitHub Pages (API & Config Help HTML)

[Repo GitHub Pages \(API & Config Help HTML\)](#)

[Driver Library Overview](#)

[AT24 Driver Library Help](#)

[AT25 Driver Library Help](#)

[I2C Driver Library Help](#)

[Memory Driver Library Help](#)

[SD Card \(SPI\) Driver Library Help](#)

[SPI Driver Library Help](#)

[USART Driver Library Help](#)

[USBART Driver Library Help](#)

[System Service Library Help](#)

[Command Processor System Service Library Help](#)

[Central Processing Unit System Service Library Help](#)

[Direct Memory Access \(DMA\) System Service Library Help](#)

[File System System Service Library Help](#)

[Interrupt System Service Library Help](#)

[Port System Service Library Help](#)

[Timing System Service Library Help](#)

[OSAL Library Help](#)

[Applications Help](#)

### Release Notes

#### Microchip MPLAB Harmony 3 Release Notes

##### Core Release v3.1

###### NEW FEATURES

- Moved HTML doc to specific folder for better GitHub integration.
- Minor documentation.

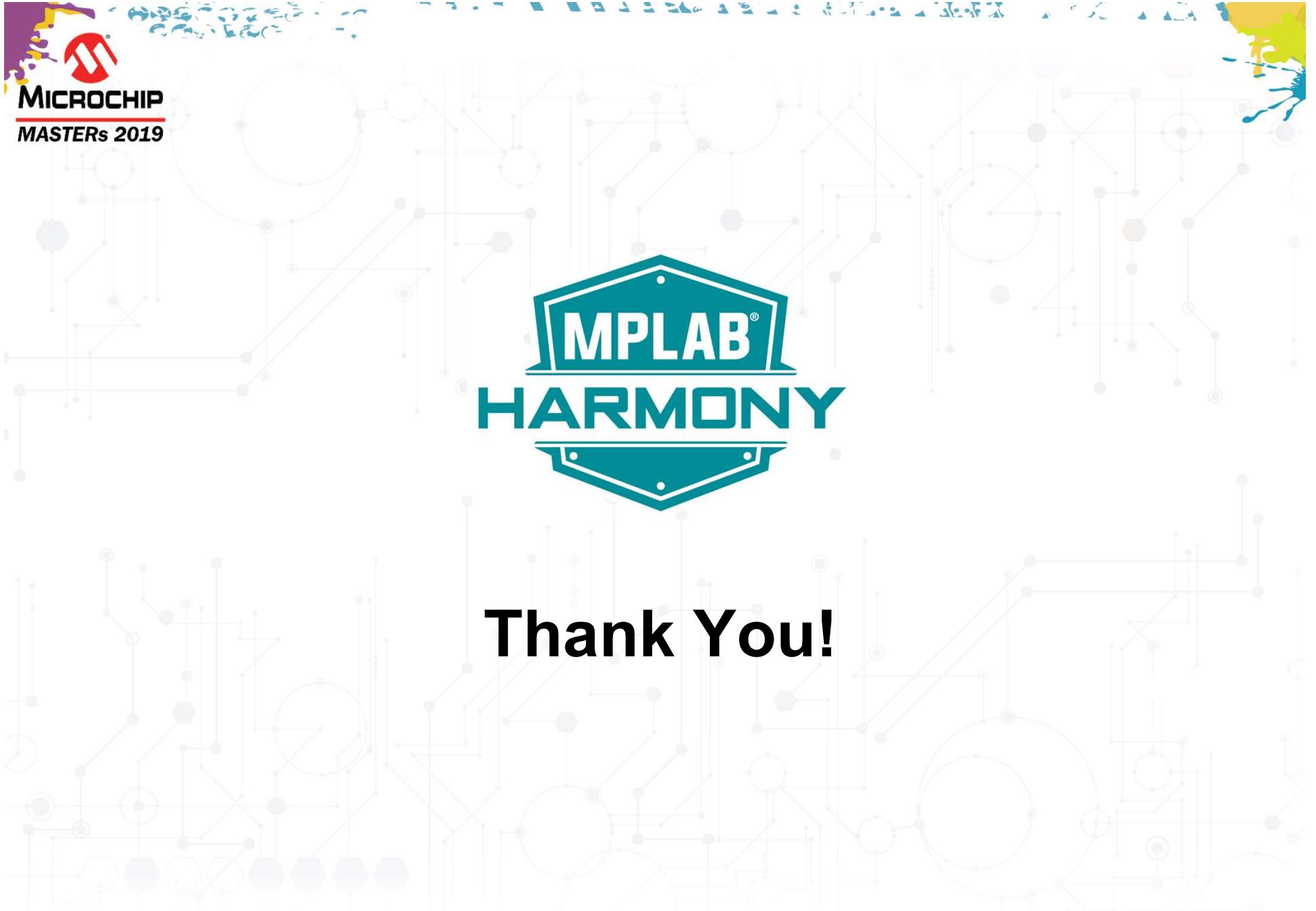
###### Core Release v3.0

###### NEW FEATURES

- New part support – VVWV families of 32-bit MCUs.
- Driver and Service System

###### Type

###### Driver



# Thank You!



# LEGAL NOTICE

**MASTERs 2019**

## **SOFTWARE:**

You may use Microchip software exclusively with Microchip products. Further, use of Microchip software is subject to the copyright notices, disclaimers, and any license terms accompanying such software, whether set forth at the install of each program or posted in a header or text file.

Notwithstanding the above, certain components of software offered by Microchip and 3<sup>rd</sup> parties may be covered by "open source" software licenses – which include licenses that require that the distributor make the software available in source code format. To the extent required by such open source software licenses, the terms of such license will govern.

## **NOTICE & DISCLAIMER:**

These materials and accompanying information (including, for example, any software, and references to 3<sup>rd</sup> party companies and 3<sup>rd</sup> party websites) are for informational purposes only and provided "AS IS." Microchip assumes no responsibility for statements made by 3<sup>rd</sup> party companies, or materials or information that such 3<sup>rd</sup> parties may provide.

MICROCHIP DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, OR STATUTORY, INCLUDING ANY IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY DIRECT OR INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND RELATED TO THESE MATERIALS OR ACCOMPANYING INFORMATION PROVIDED TO YOU BY MICROCHIP OR OTHER THIRD PARTIES, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES OR THE DAMAGES ARE FORESEEABLE. PLEASE BE AWARE THAT IMPLEMENTATION OF INTELLECTUAL PROPERTY PRESENTED HERE MAY REQUIRE A LICENSE FROM THIRD PARTIES.

## **TRADEMARKS:**

The Microchip name and logo, the Microchip logo, AnyRate, AVR, AVR logo, AVR Freaks, BeaconThings, BitCloud, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, Heldo, JukeBlox, KEELOQ, KEELOQ logo, Kleer, LANCheck, LINK MD, maXStylus, maXTouch, MediaLB, megaAVR, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, Prochip Designer, QTouch, RightTouch, SAM-BA, SpyNIC, SST, SST Logo, SuperFlash, tinyAVR, UNI/O, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

ClockWorks, The Embedded Control Solutions Company, EtherSynch, Hyper Speed Control, HyperLight Load, IntelliMOS, mTouch, Precision Edge, and Quiet-Wire are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BodyCom, chipKIT, chipKIT logo, CodeGuard, CryptoAuthentication, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet logo, Mindi, MiWi, motorBench, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PureSilicon, QMatrix, RightTouch logo, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQI, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

Silicon Storage Technology is a registered trademark of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2017, Microchip Technology Incorporated, All Rights Reserved.