MICROCHIP MASTERs 2019

MPLAB Harmony 3

Fundamentals

SMART | CONNECTED | SECURE

MICROCHIP

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

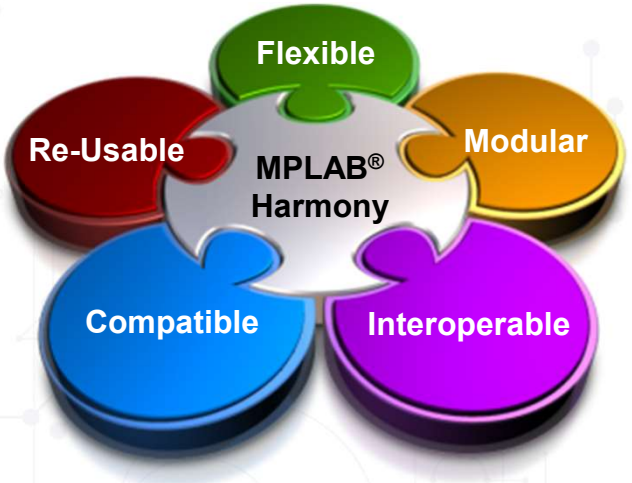
Agenda

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.

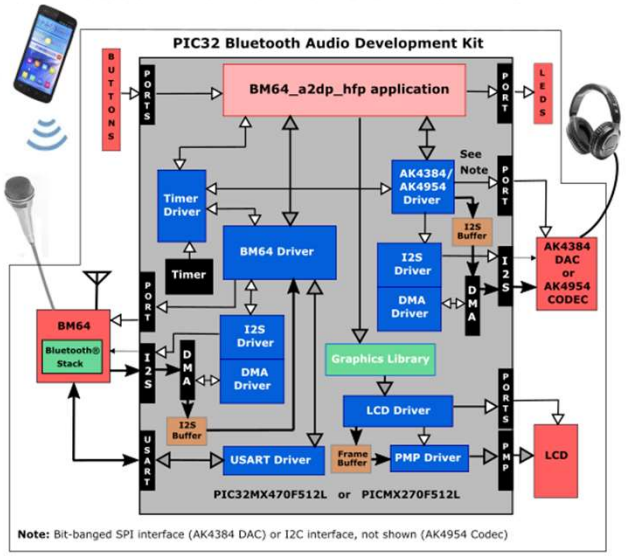
MPLAB Harmony 3 Provides:

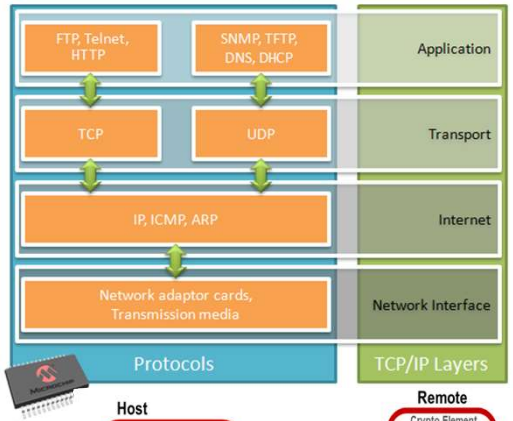


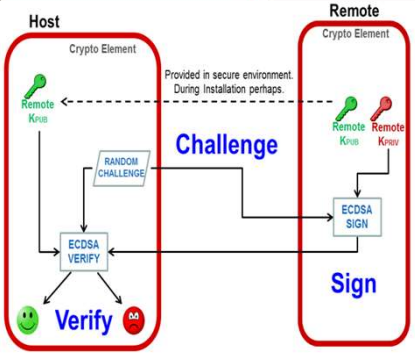
- **Demo/Example Applications**
- **Layered & Modular Libraries**
  - Peripheral Libraries
  - Drivers & Services
  - Middleware
- **Graphical Developer Tools**
  - Deploying, configuring, and generating the libraries.
- **An embedded software reference model.**

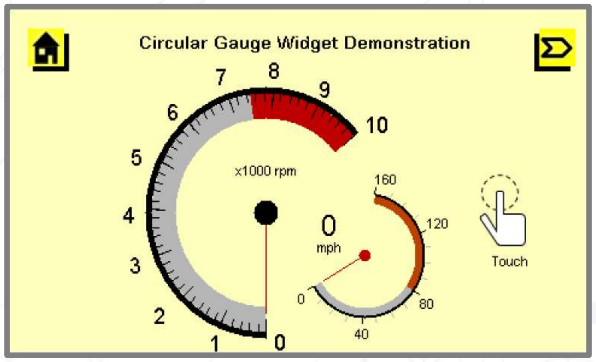
Demo Applications

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



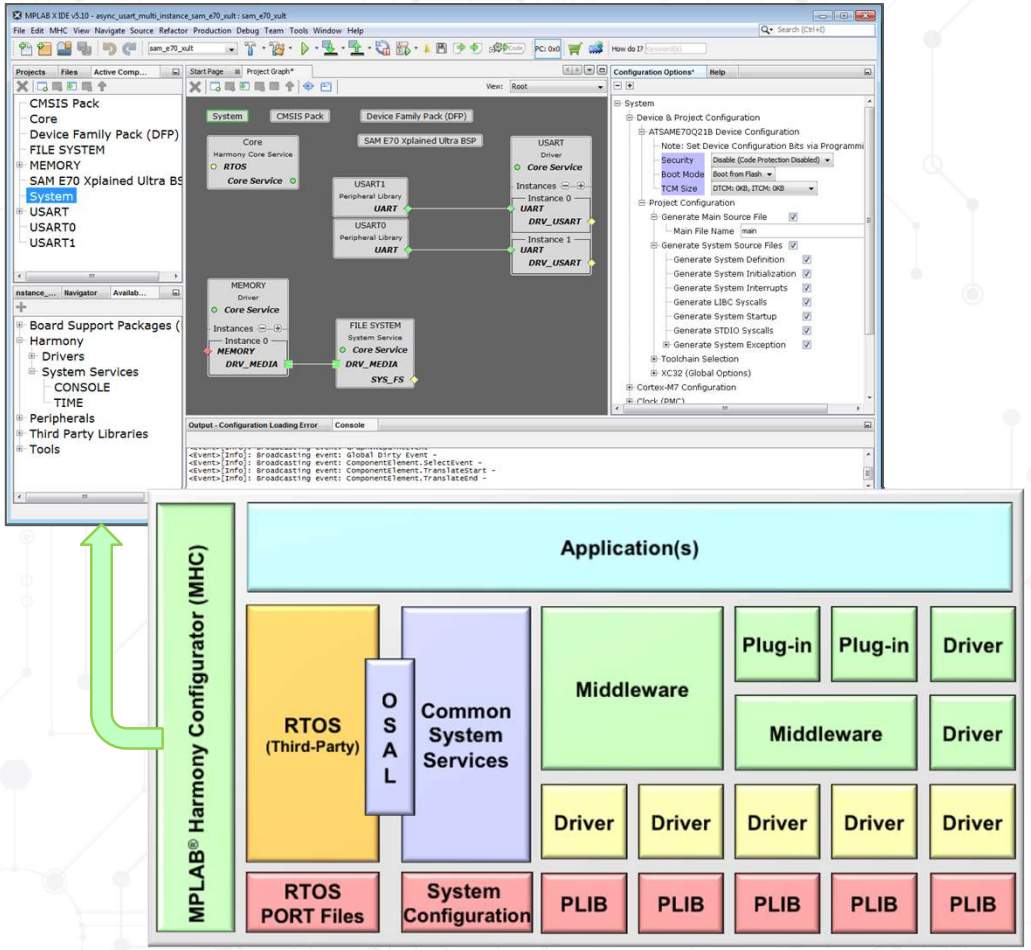






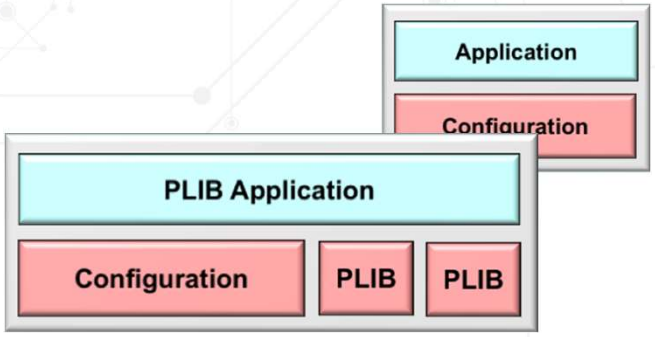
Library Framework & GUI

- **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)

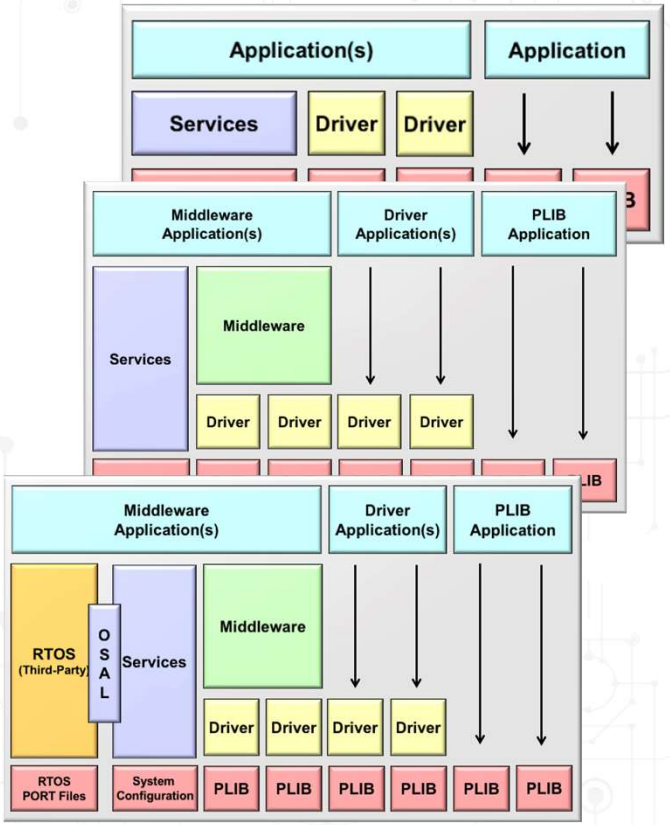


Simple Projects

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



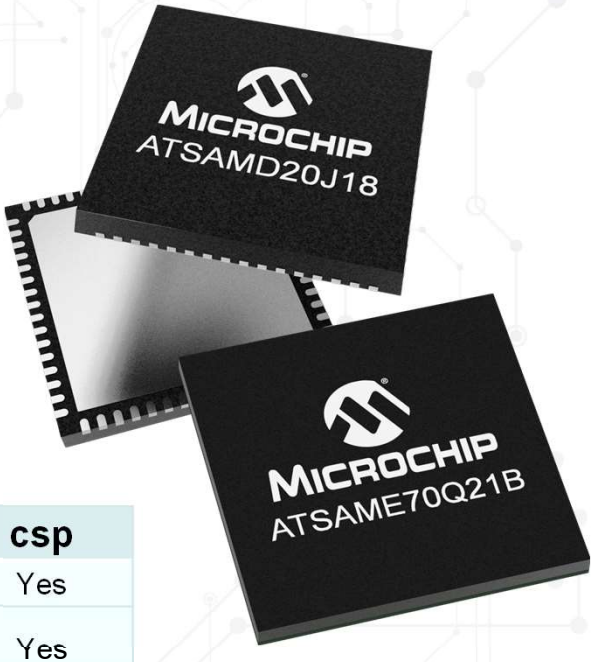
Advanced Capabilities

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

Simplified PLIBs

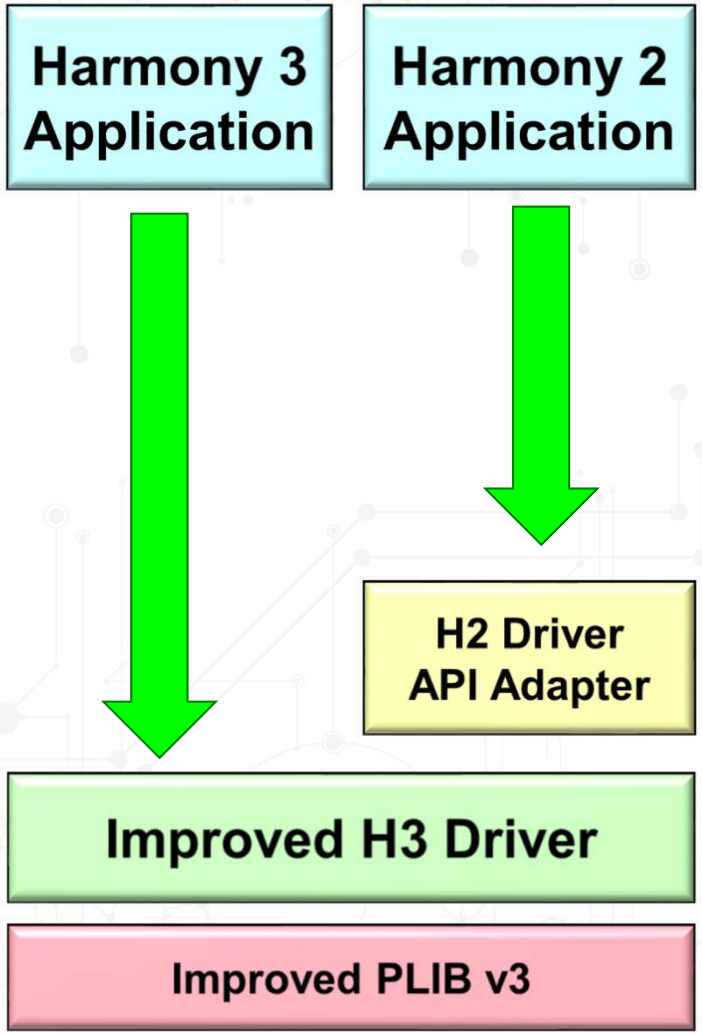
- Easier to use APIs
- Similar to MCC Peripheral Libraries
- Single Instance

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

- 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).

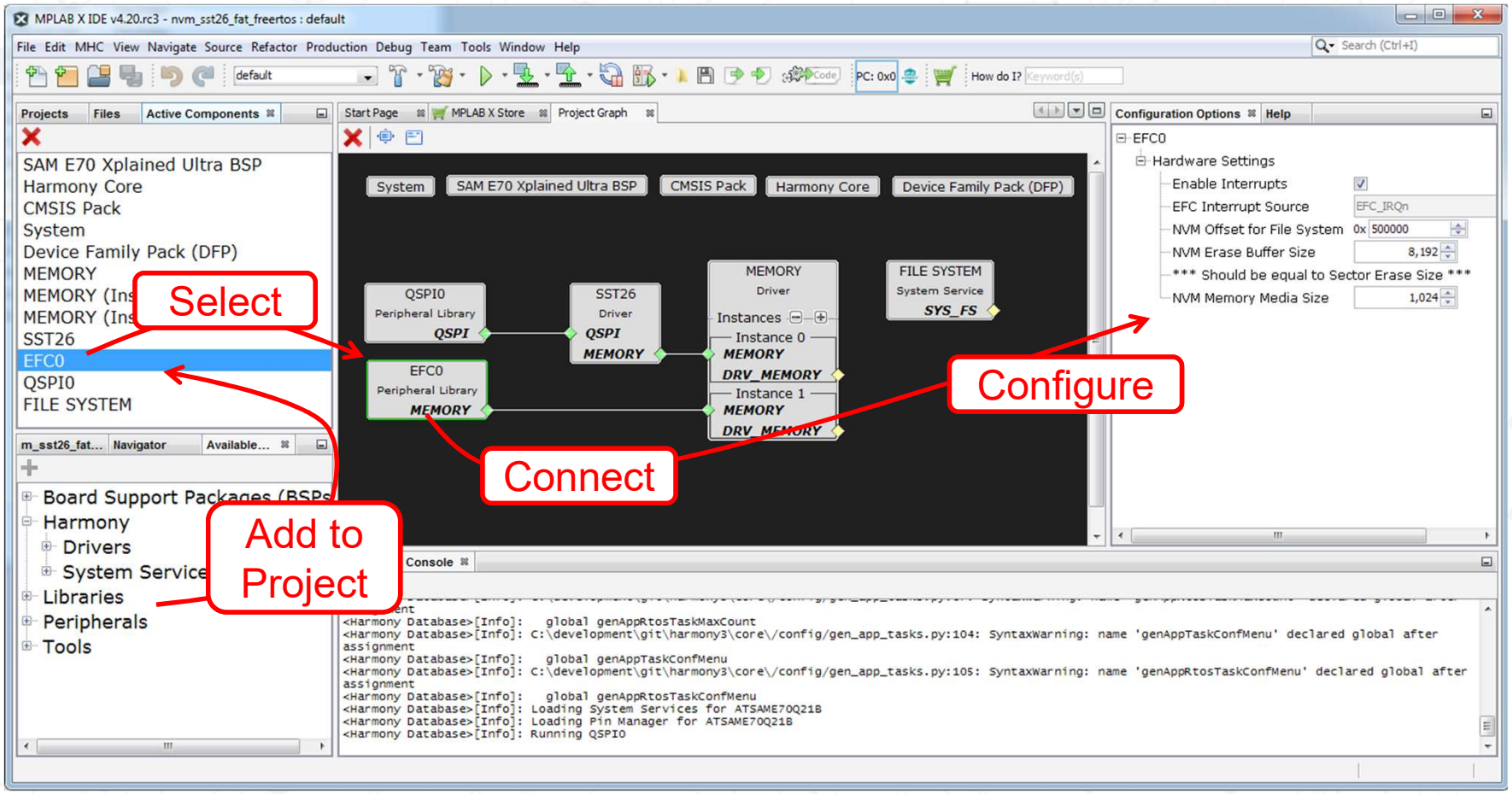
Updated Drivers and Services

- **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

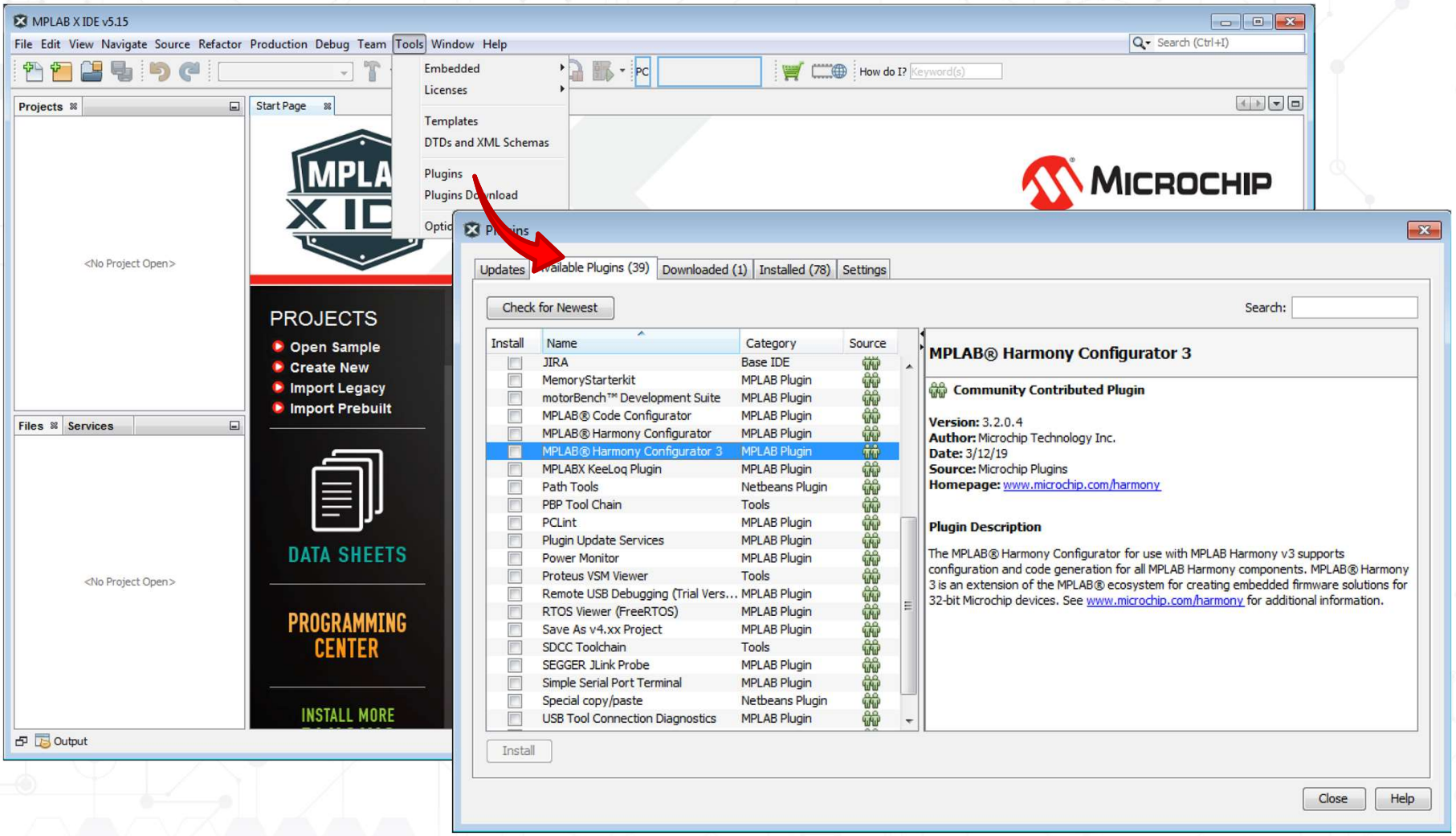
General Notes

- **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

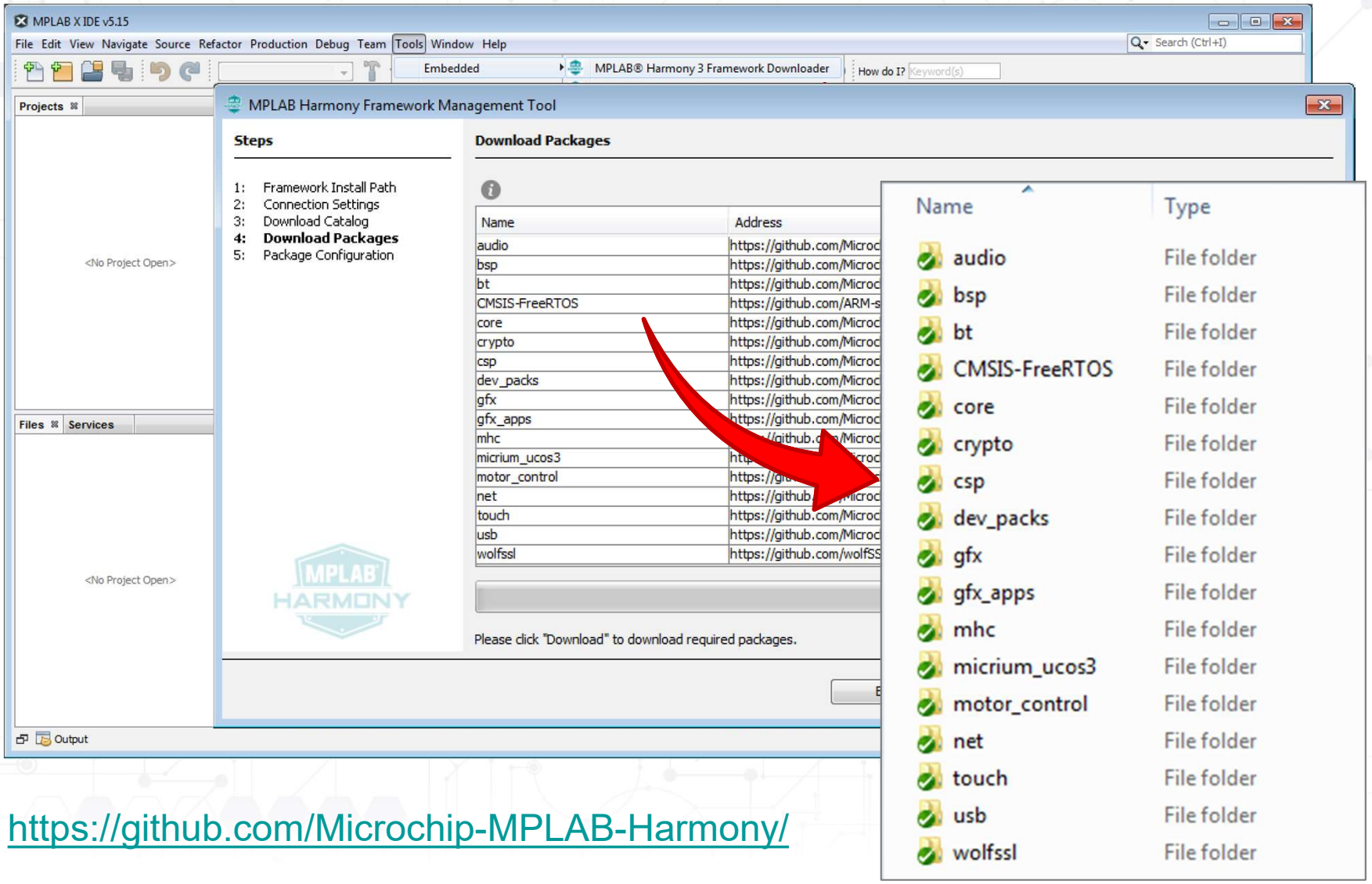
1. MPLAB Harmony 3 Introduction
2. Overview of changes from MPLAB Harmony 2
3. **Introduction to the downloader and to the modular library packages**
4. Fundamentals of applications, projects, configuration, and peripheral libraries
   - Lab1: Create a simple “heartbeat” LED application that flashes 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



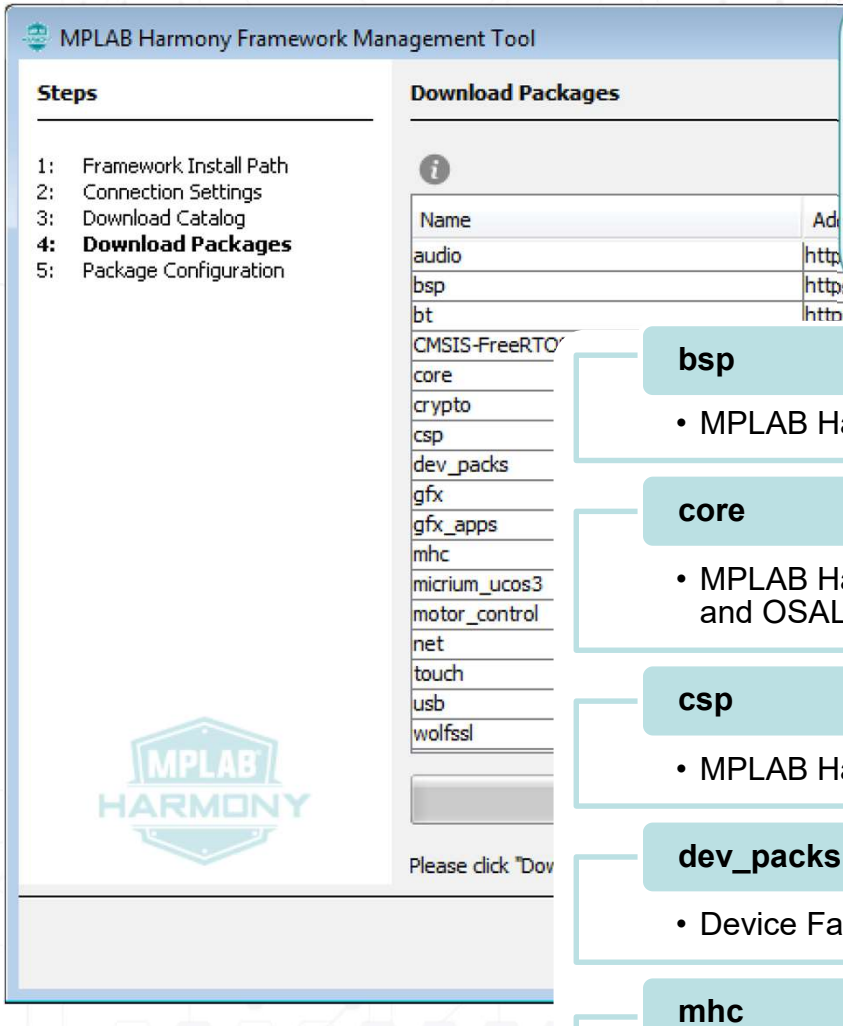
https://www.microchip.com/mplab

Module Deployment



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)

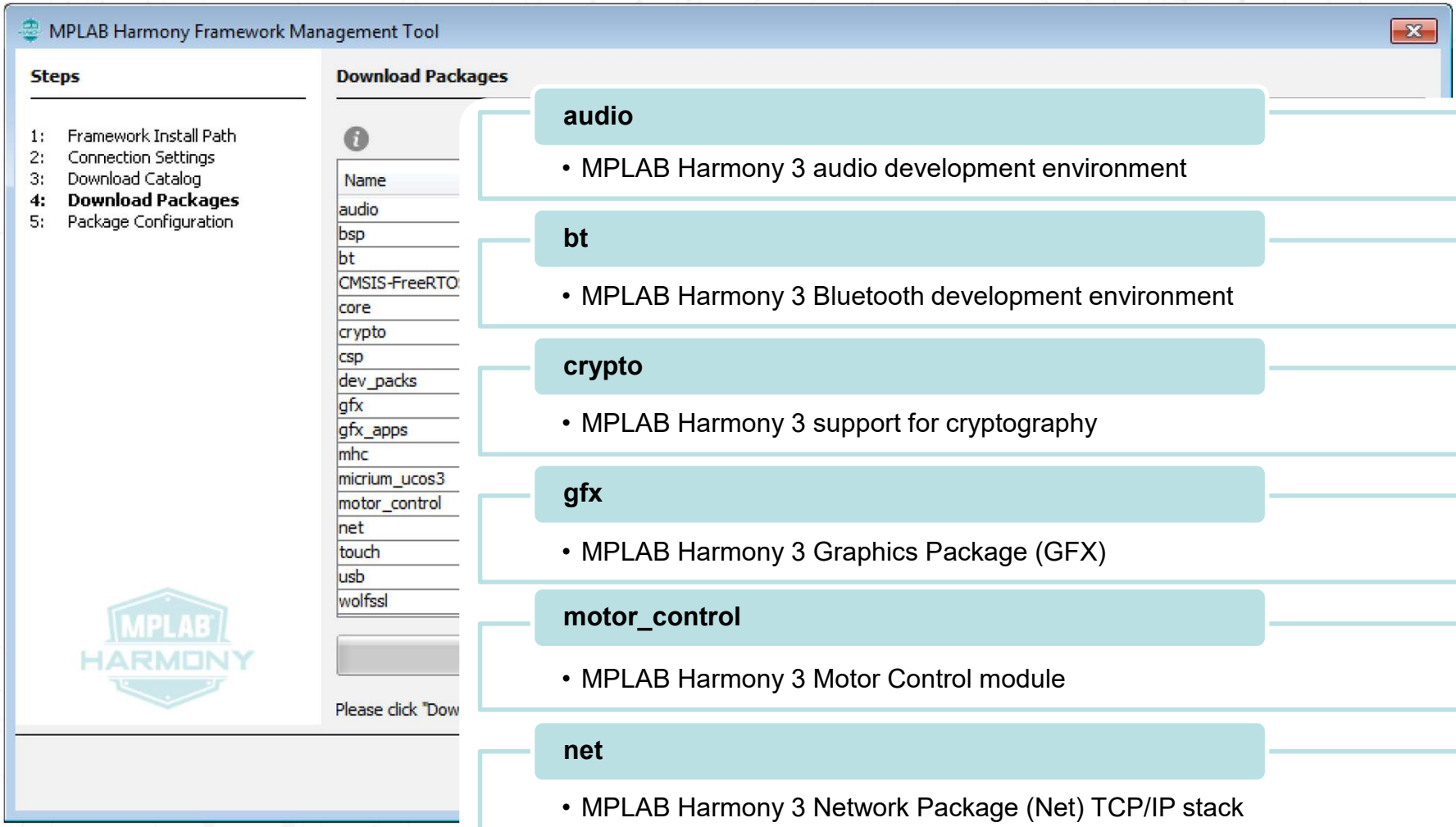
dev_packs

- Device Family Packs Repository

mhc

- MPLAB Harmony 3 Configurator (MHC) tool

Harmony 3 MW Packages



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)

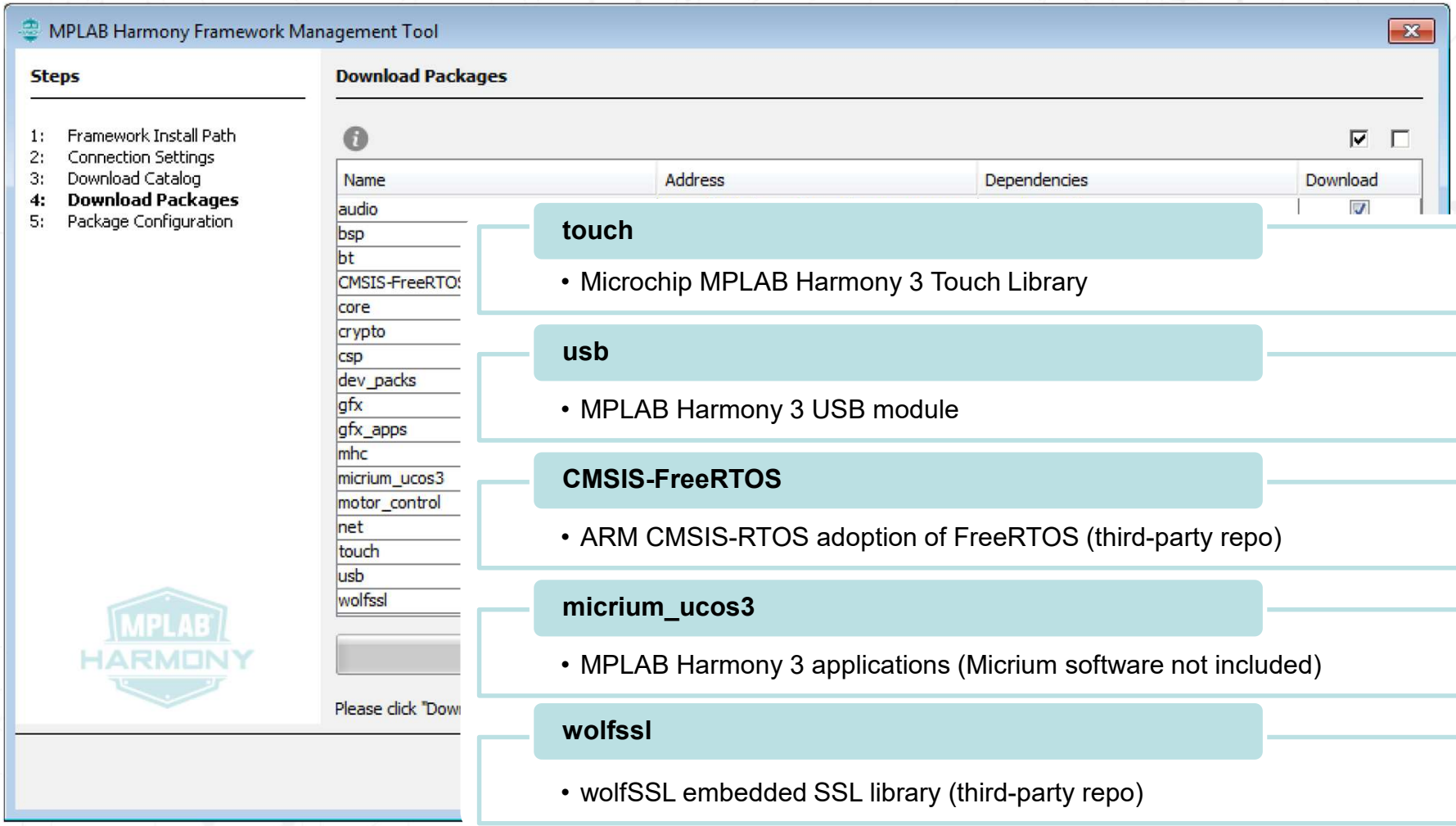
motor_control

- MPLAB Harmony 3 Motor Control module

net

- MPLAB Harmony 3 Network Package (Net) TCP/IP stack

Harmony 3 MW Packages



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

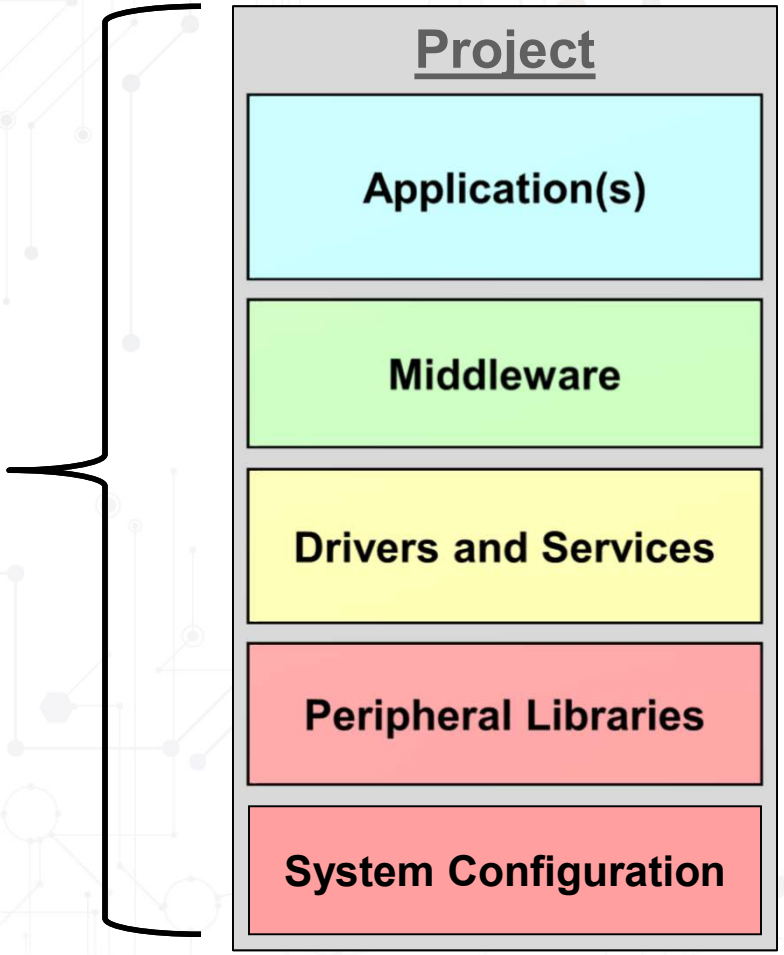
- wolfSSL embedded SSL library (third-party repo)

Agenda

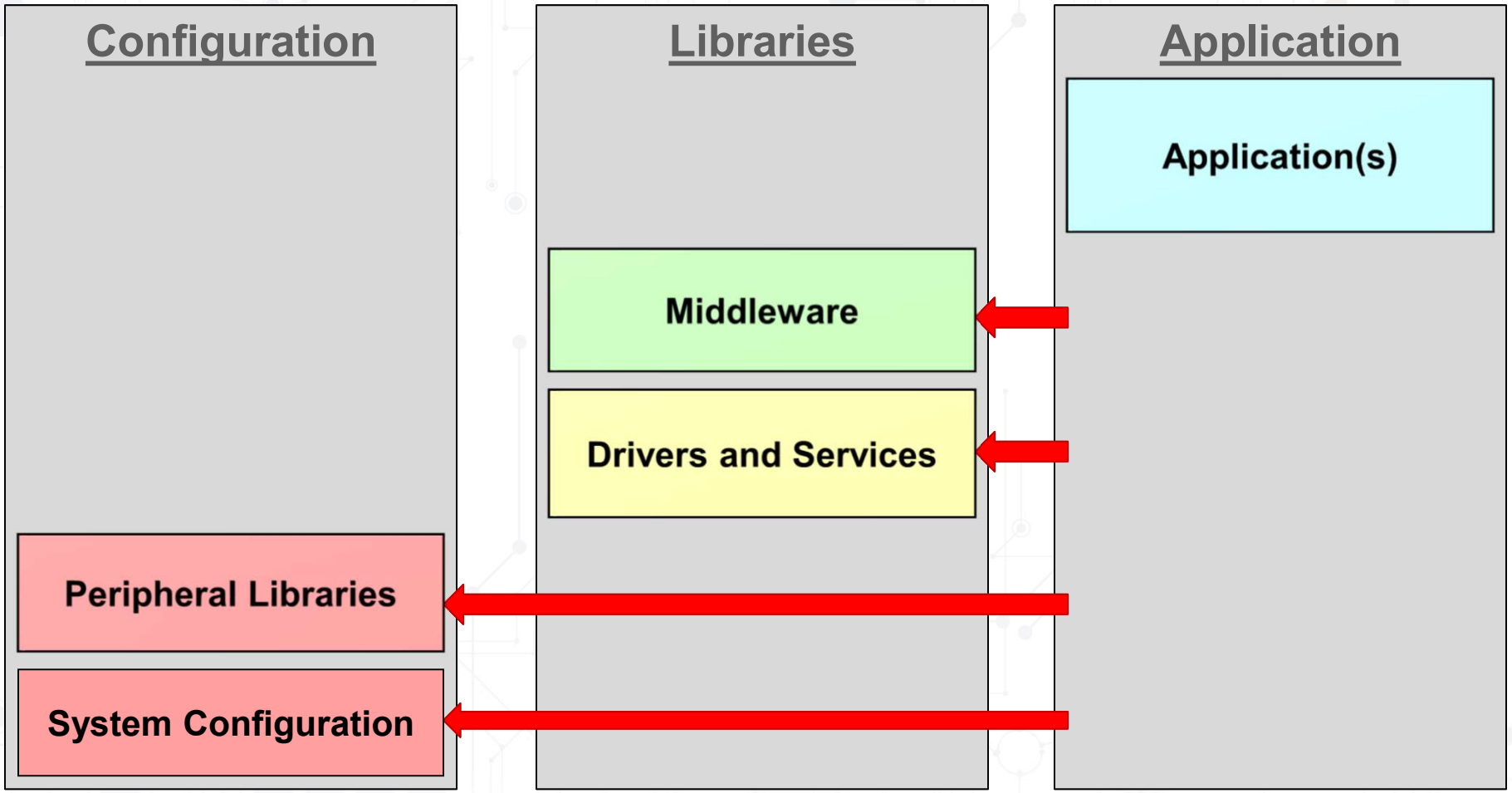
1. MPLAB Harmony 3 Introduction
2. Overview of changes from MPLAB Harmony 2
3. Introduction to the downloader and to the modular library packages
4. **Fundamentals of applications, projects, configuration, and peripheral libraries**
   - Lab1: Create a simple “heartbeat” LED application that flashes 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

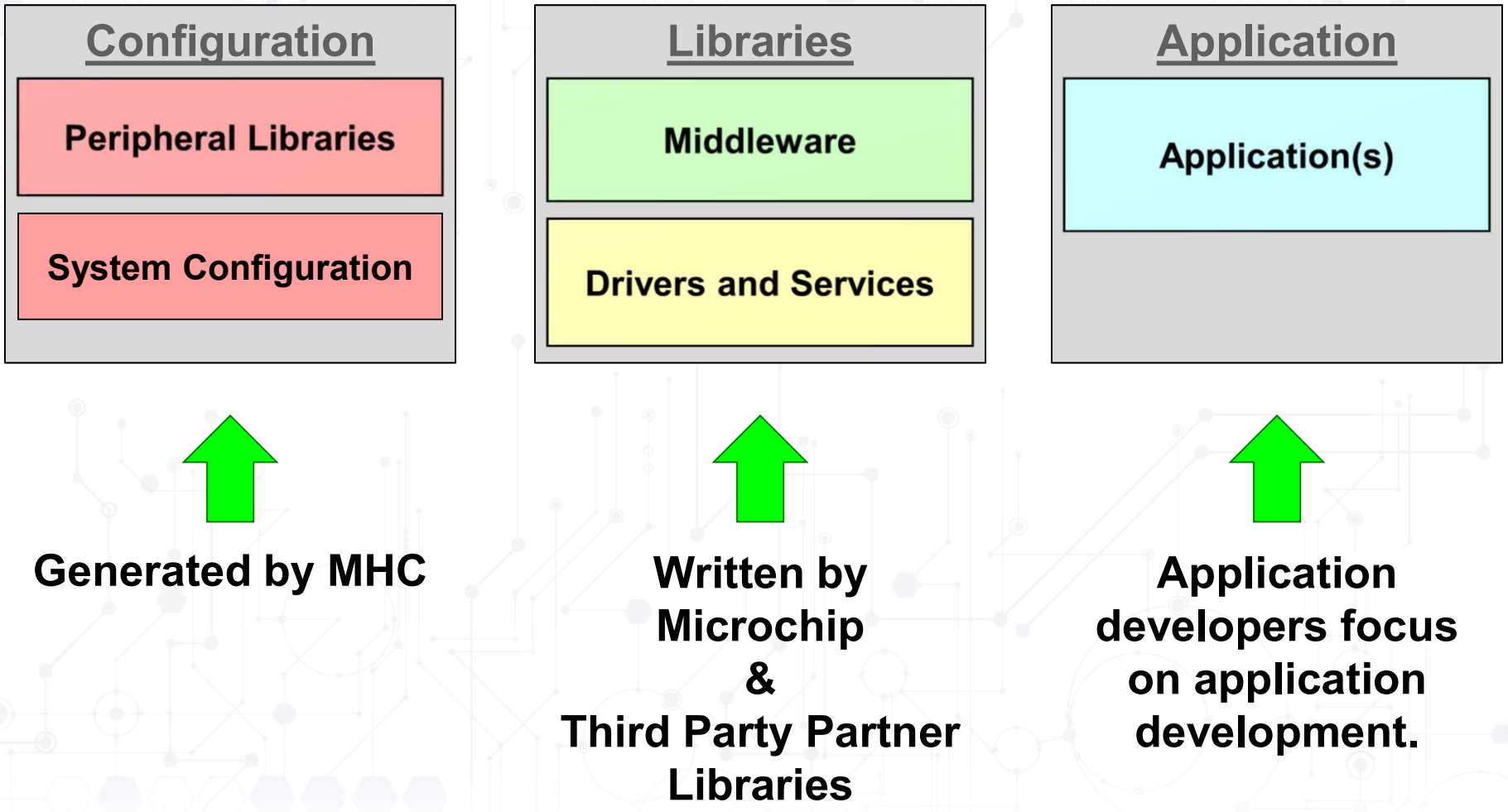
Different Parts of an MPLAB Harmony 3 application serve different purposes.



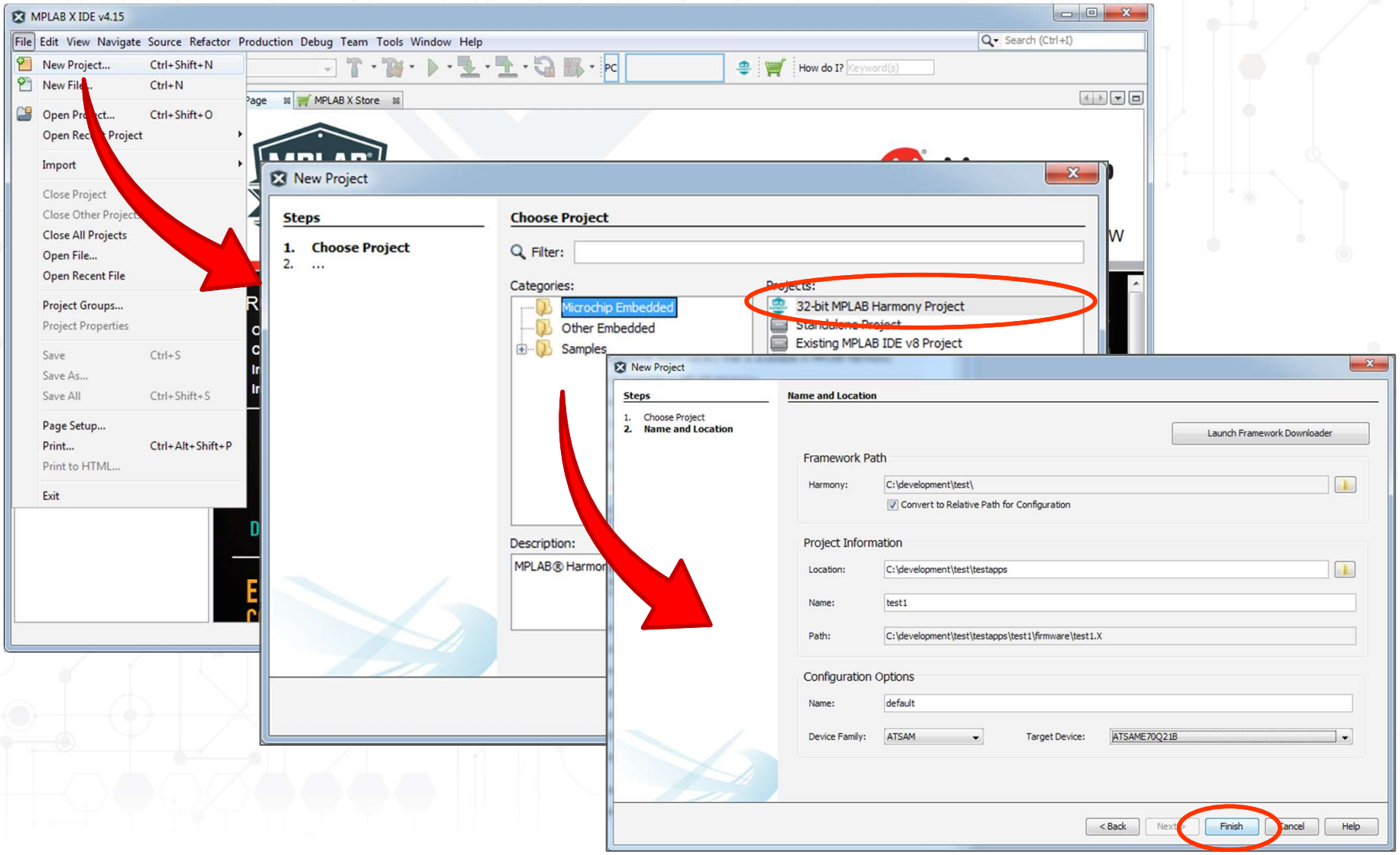
Application Factoring



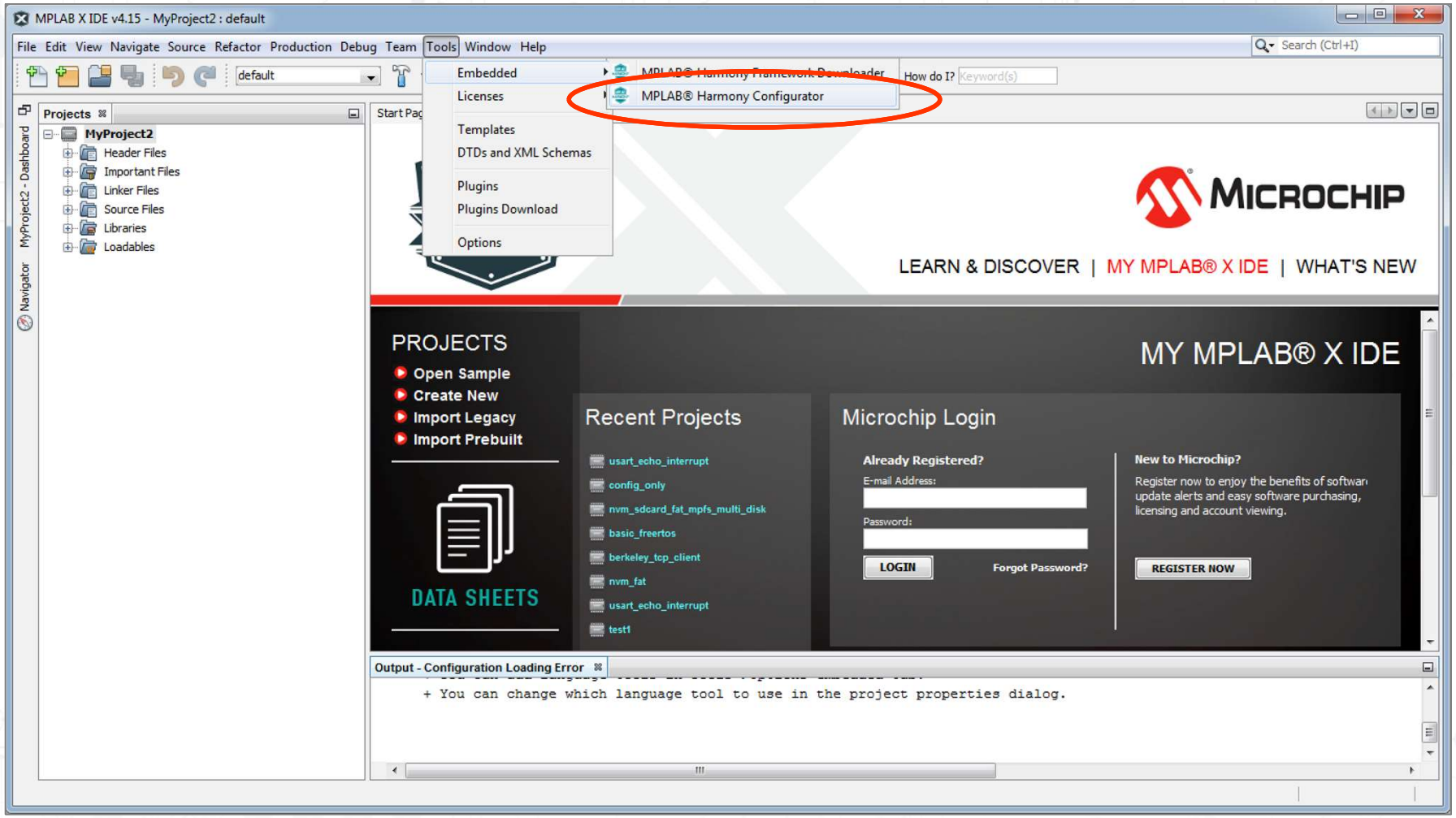
Grouping Application "Parts"



Application Project Creation

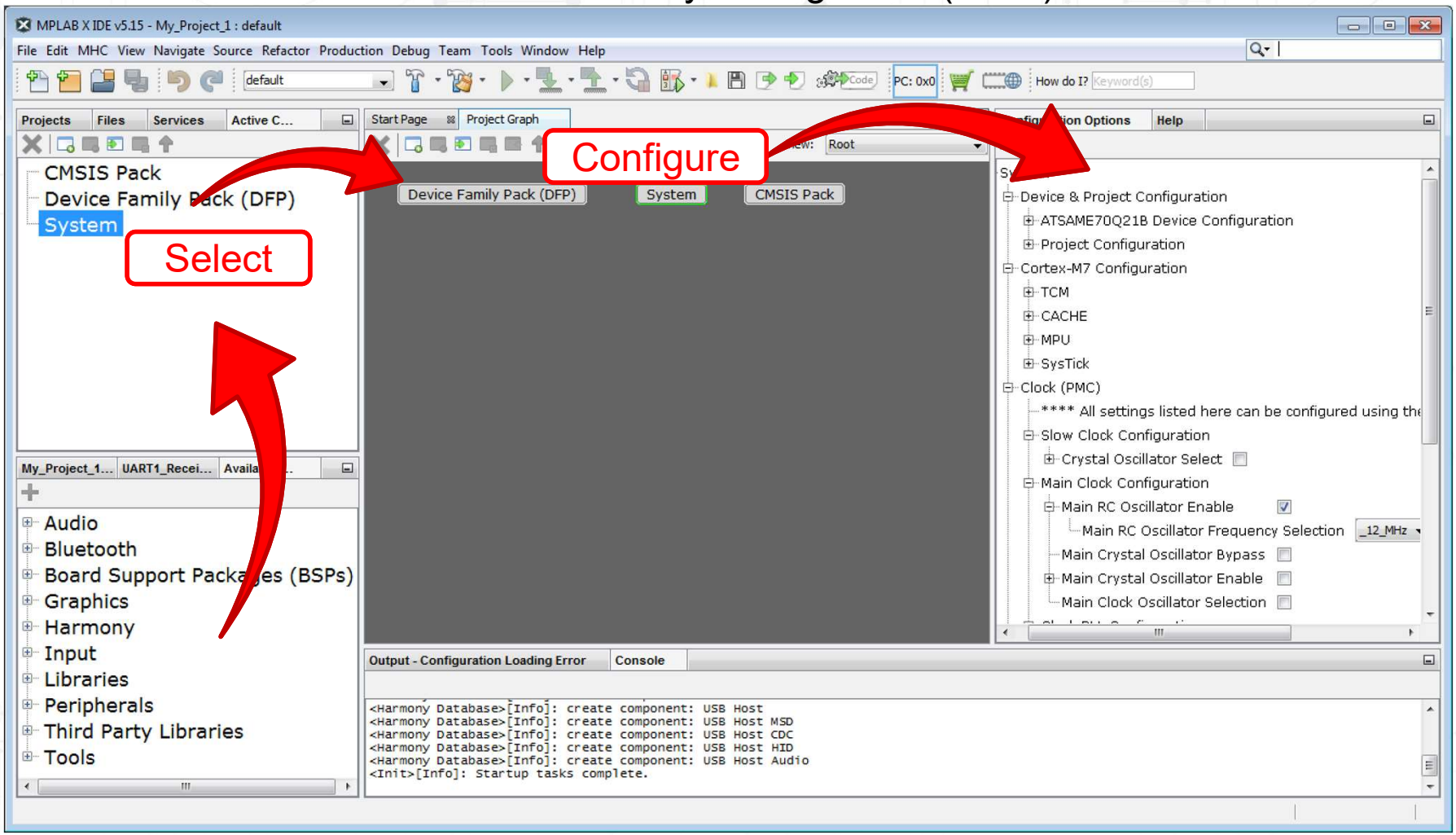


Application Project Creation



Configuration GUI

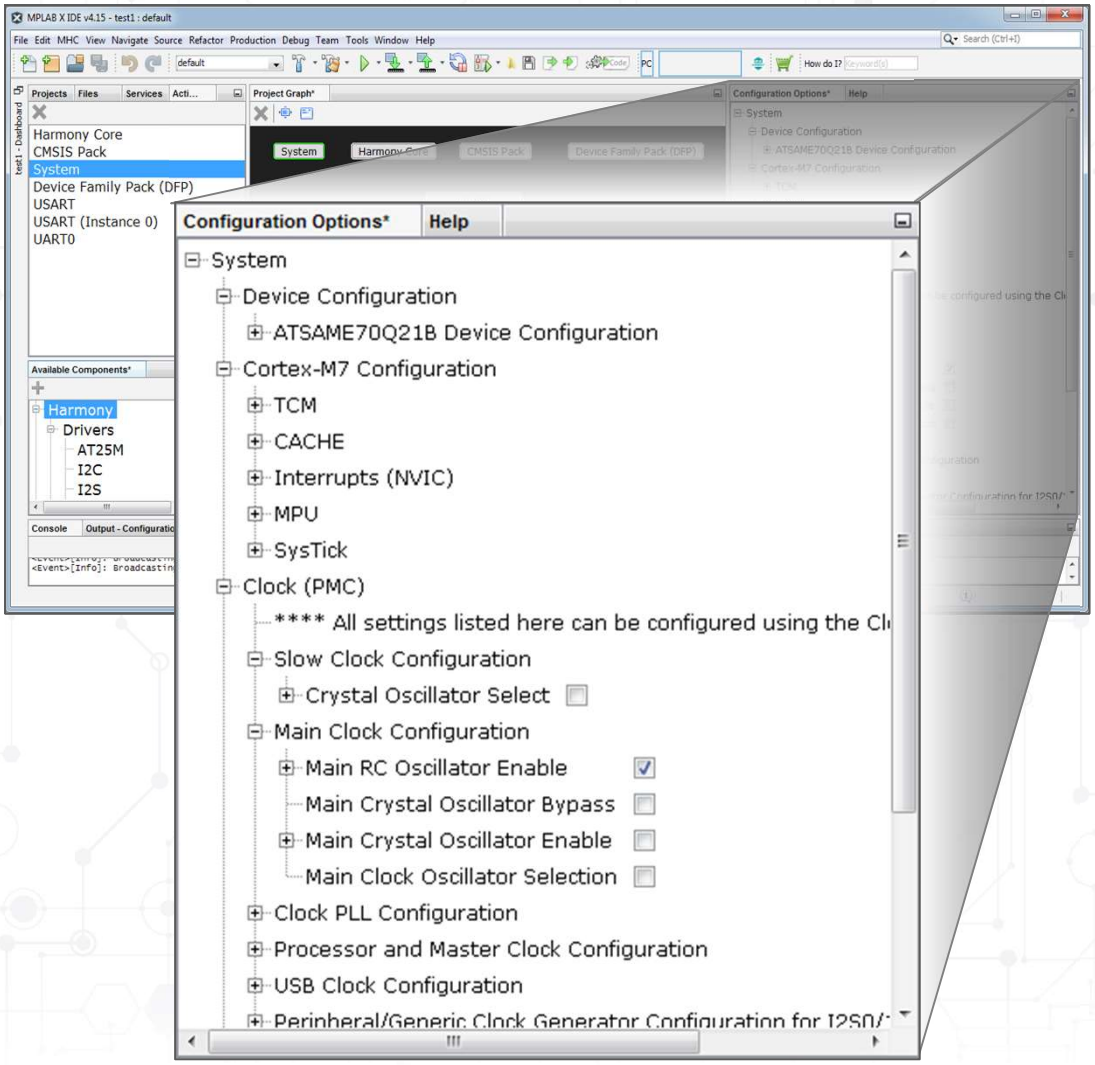
MPLAB Harmony Configurator (MHC)



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

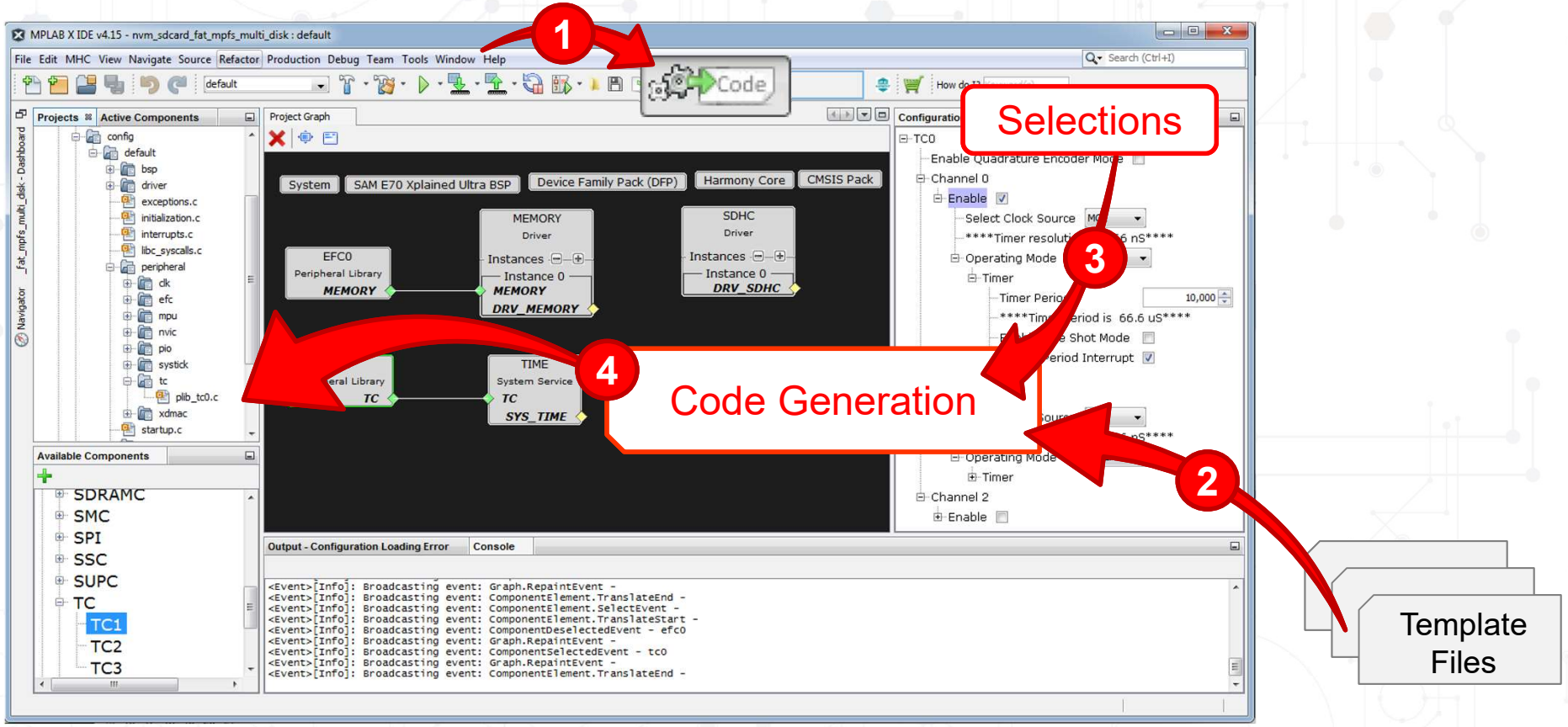


System Settings

(Basic device settings)

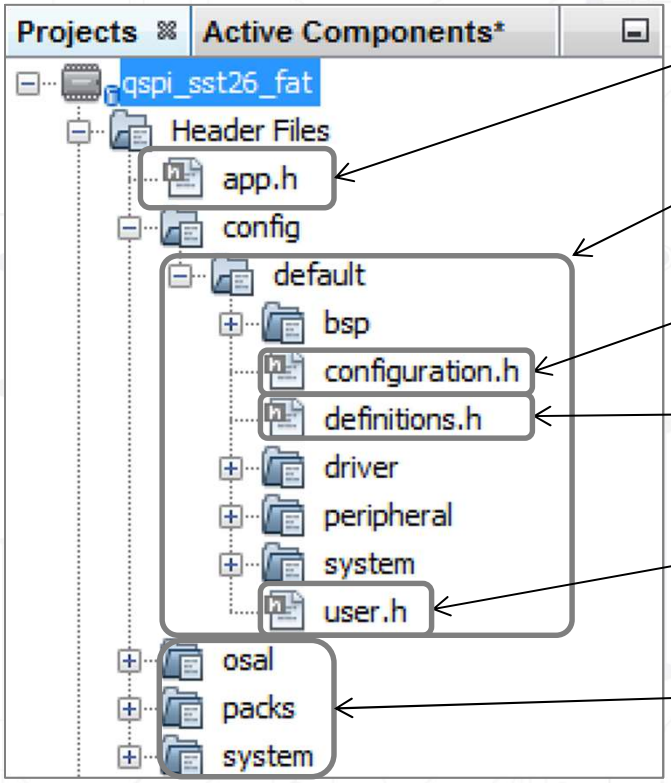
- Security, boot modes
- Cache
- Memory (TCM, MPU)
- Clocks
  - PLL/Oscillators
  - Peripheral Clocks
  - SysTick/Core Timer
- Ports/Pins Settings
- Interrupts, DMA
- Watchdog

Code Generation



Project Organization

Header Files



Application header file

Configuration (name: “default”) files

Build options (configuration #define) header

System definitions header (ex. sysObj)

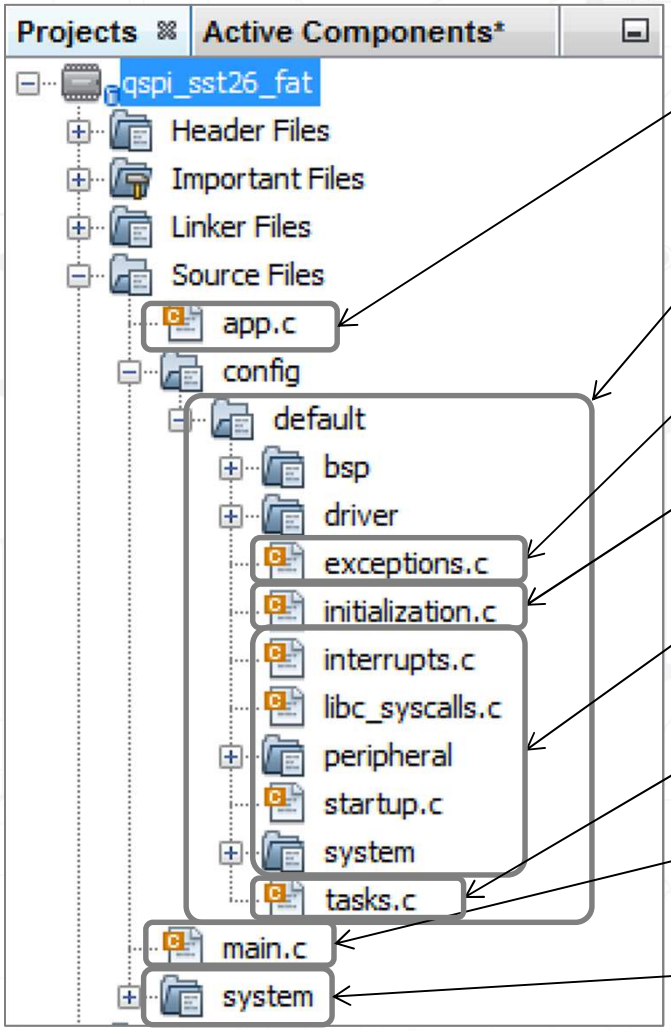
User-defined build options header

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

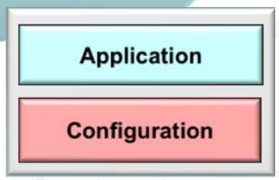
SYS_Tasks (and thread) implementations

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 );
}
```

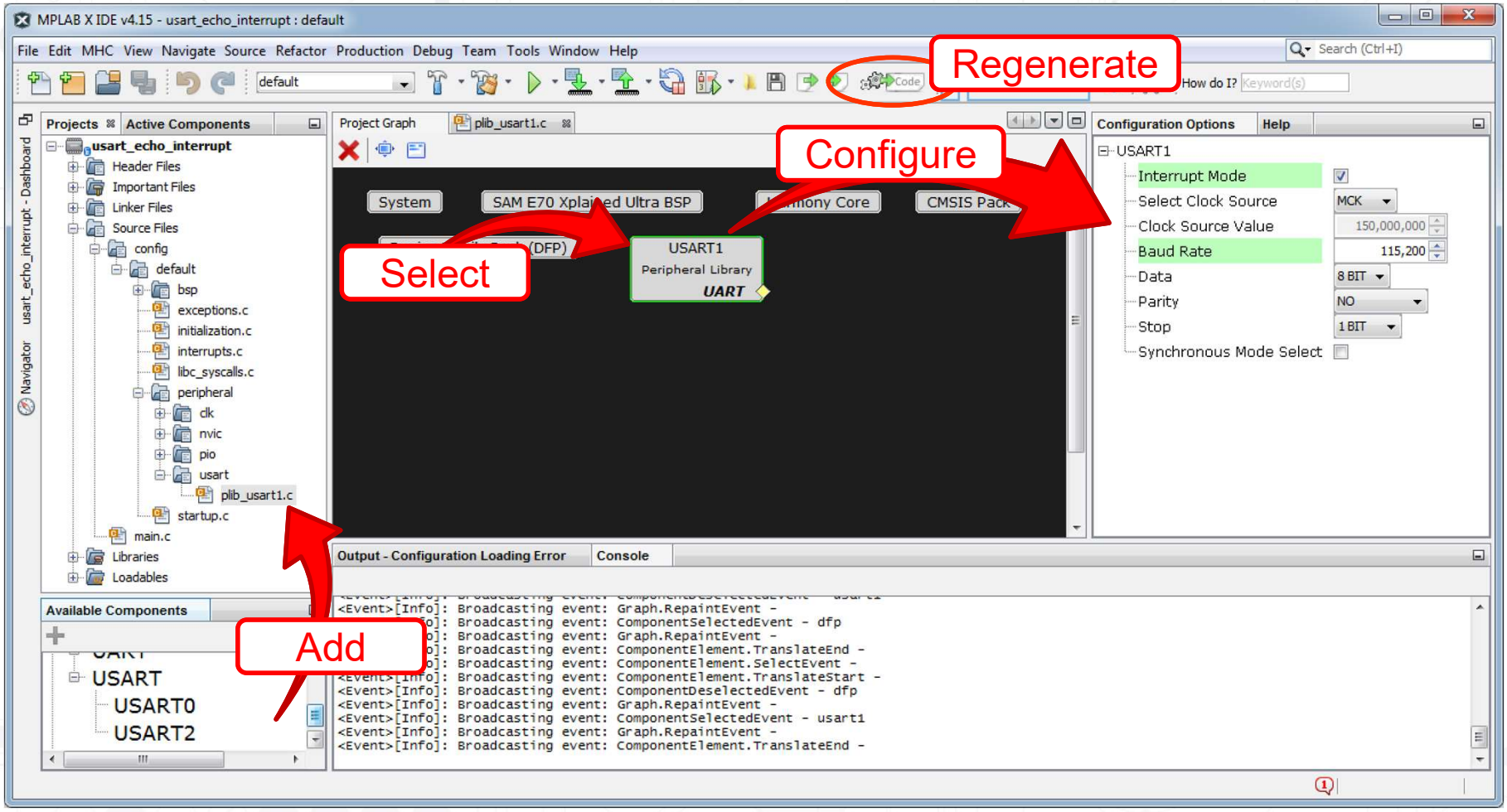
May also include:

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

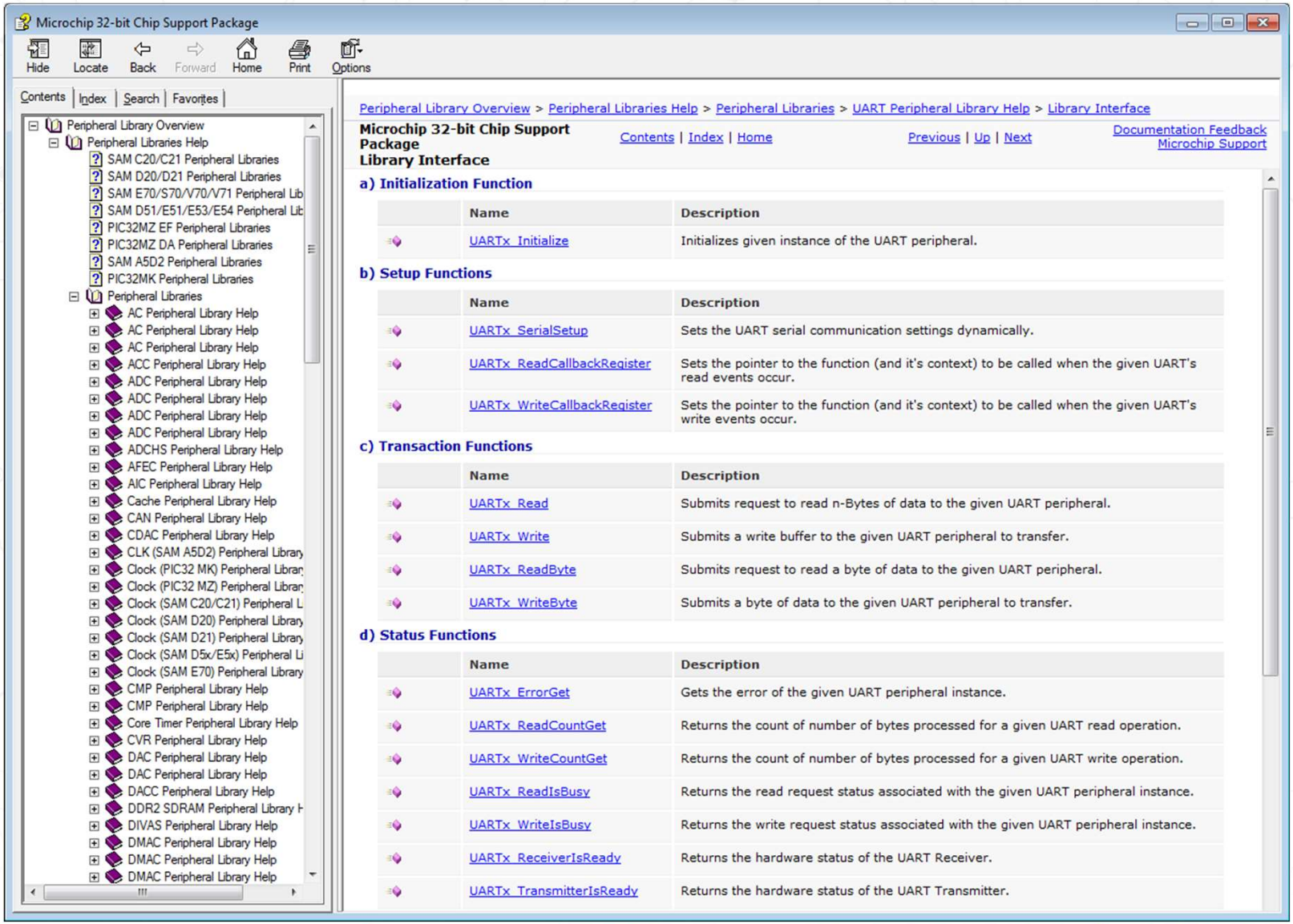
initialize.c

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

Peripheral Libraries

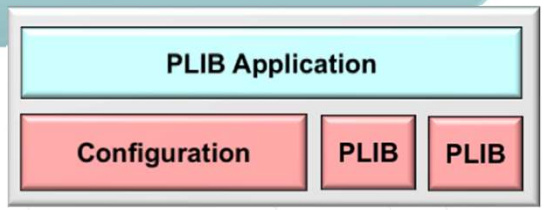


A PLIB for Every Peripheral



PLIB-based Project

Peripheral Libraries



main.c

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

    /* Register callback functi
    USART1_WriteCallbackRegiste
    USART1_ReadCallbackRegister
    USART1_Write(&messageStart,

    while ( true )
    {
        if(errorStatus == true)
        {
            /* Send error messag
            errorStatus = false;
            USART1_Write(&messag
        }
...
```

initialize.c

```
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_

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

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

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

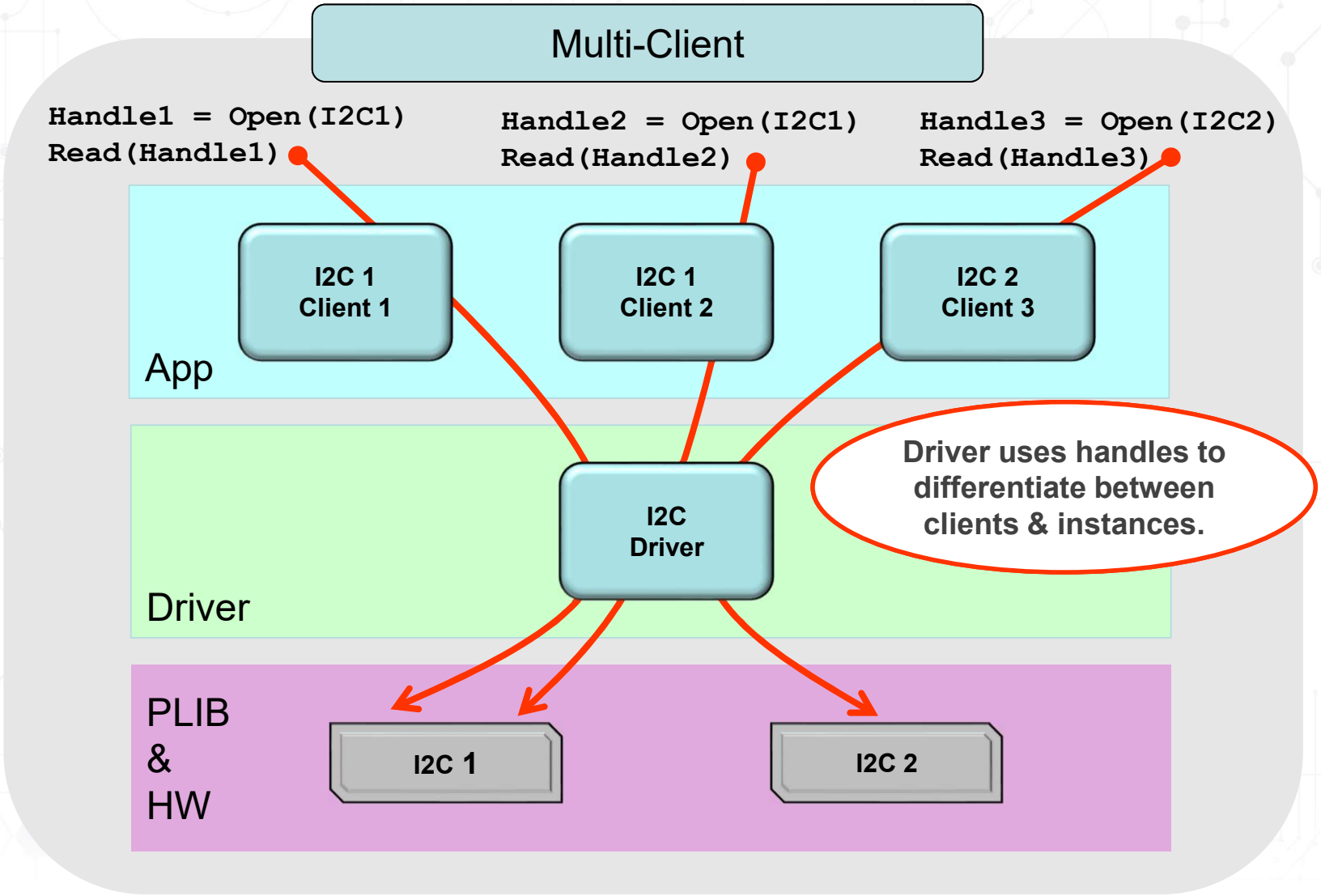
Agenda

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

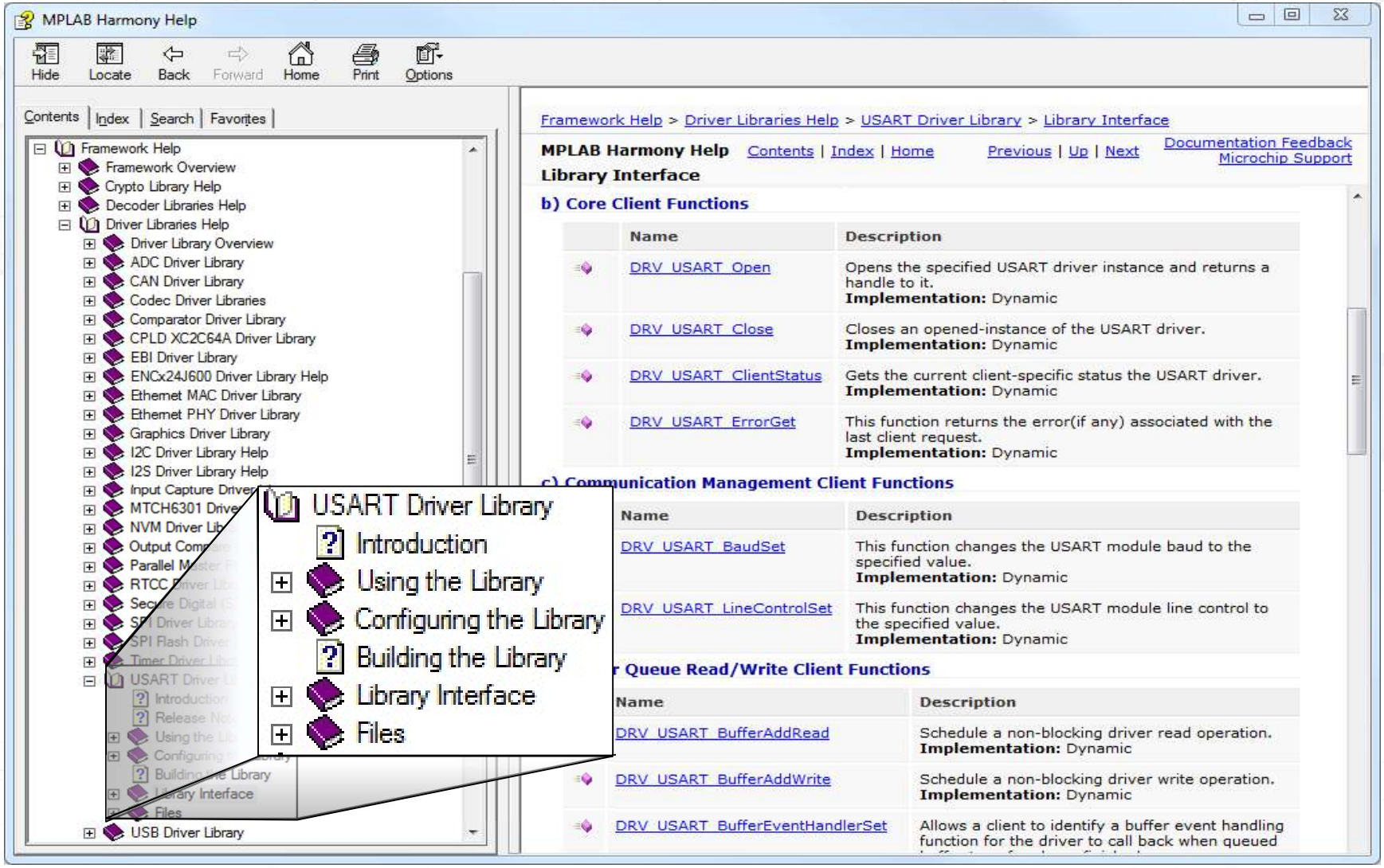
Harmony Drivers

- 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

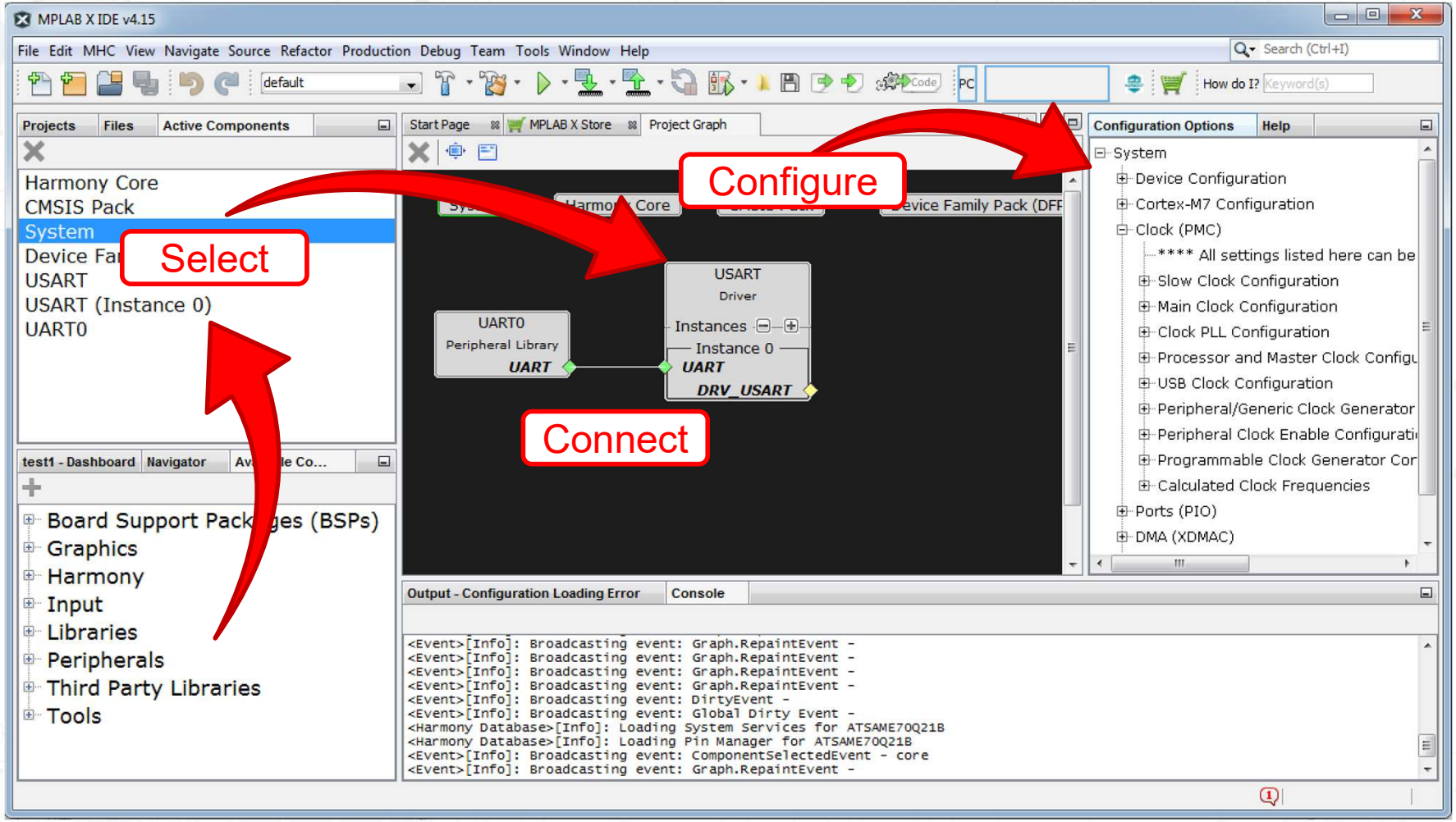
Multi-Instance & Multi-Client



Portable Interfaces

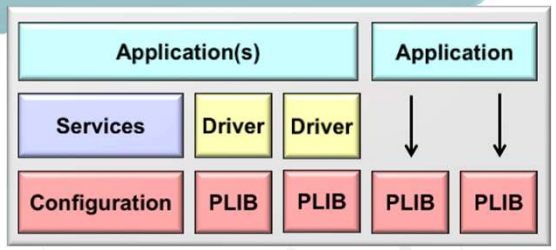


Connect Driver to PLIB



Driver-Based Project

Drivers & Services



main.c

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

    while(true)
    {
        SYS_Tasks();
    }

    return(EXIT_FAILURE
}
```

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(mess
            {
                app.state = APP_RECEIVE;
            }
            break;
...
```

Other Files Added

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

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,

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

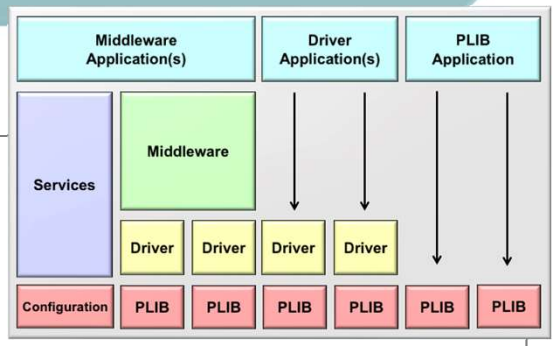
DSP/ Math

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

Bluetooth/ Audo

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_USBDeviceEventHandler, 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

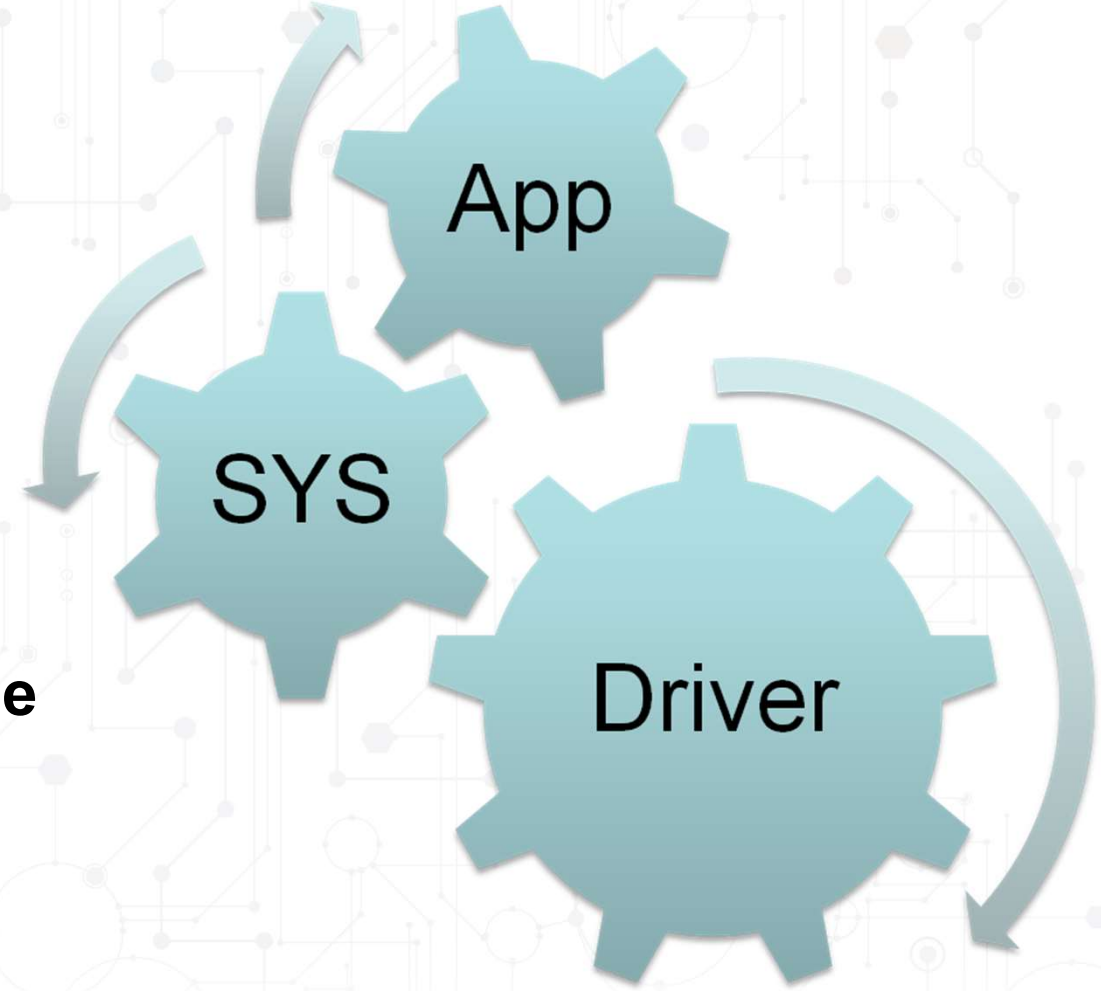
- Configured and used USB Device stack and CDC Function driver

Agenda

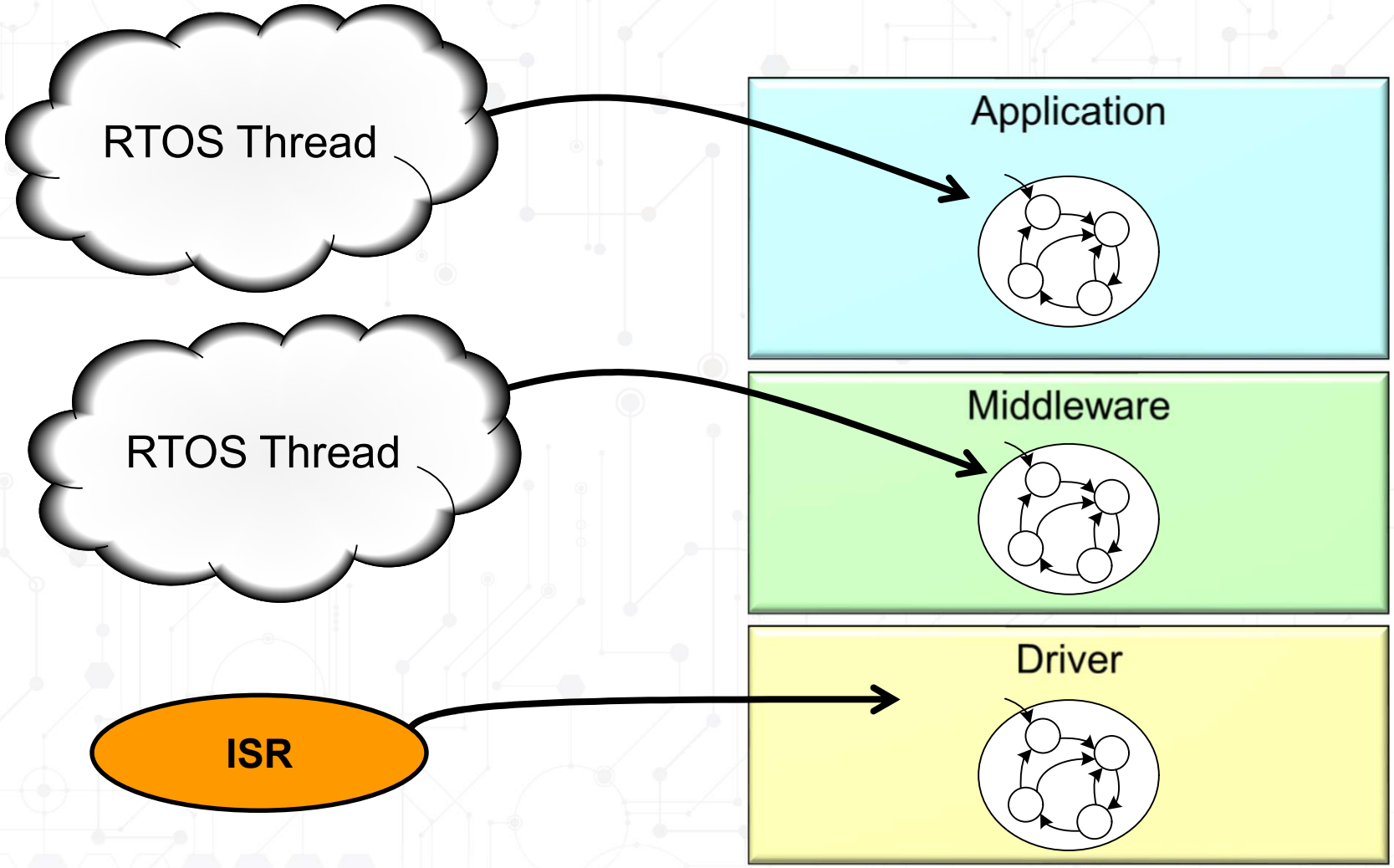
1. MPLAB Harmony 3 Introduction
2. Overview of changes from MPLAB Harmony 2
3. Introduction to the downloader and to the modular library packages
4. Fundamentals of applications, projects, configuration, and peripheral libraries
   - Lab1: Create a simple “heartbeat” LED application that flashes an LED 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.**

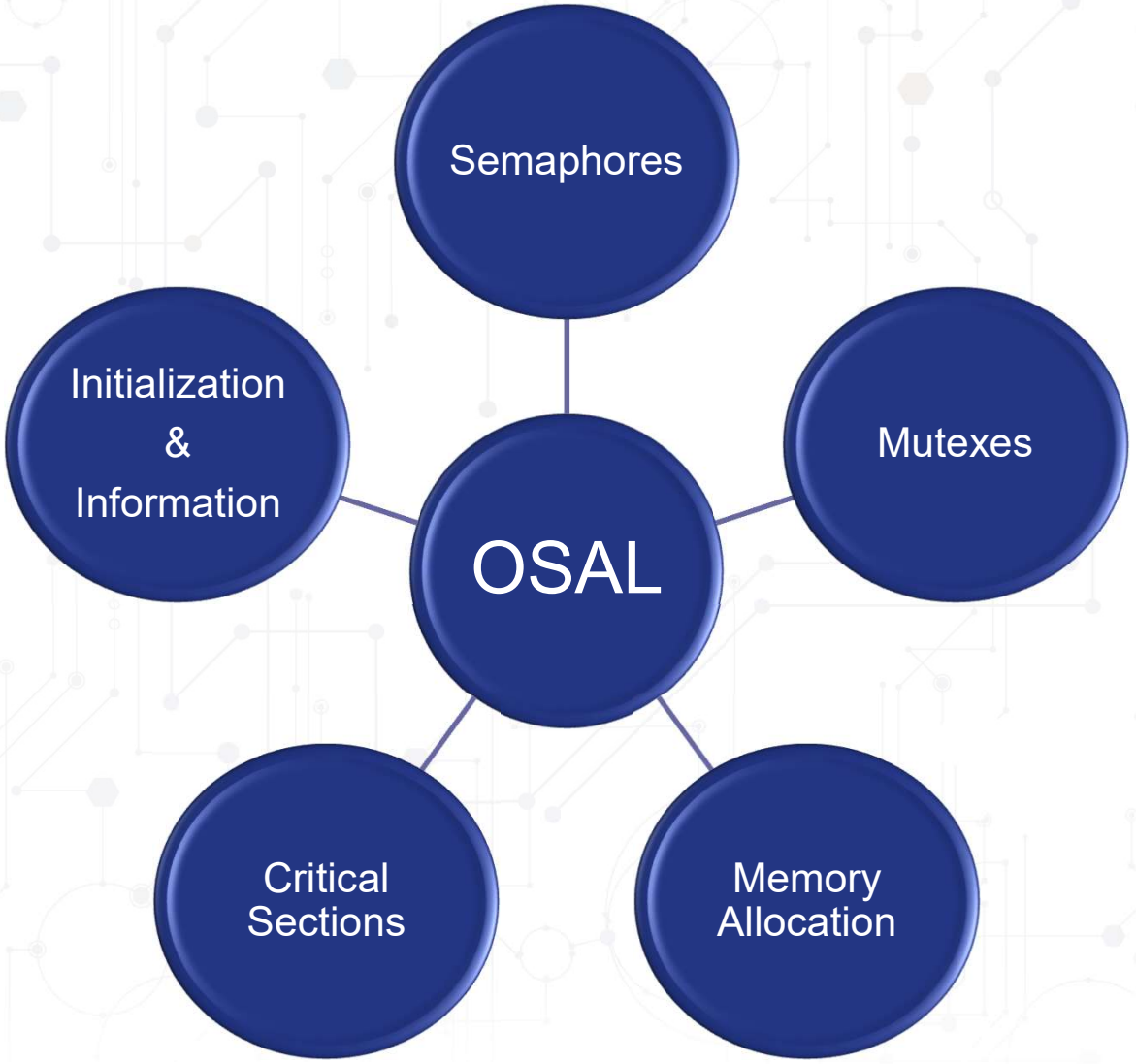


RTOS Scheduler

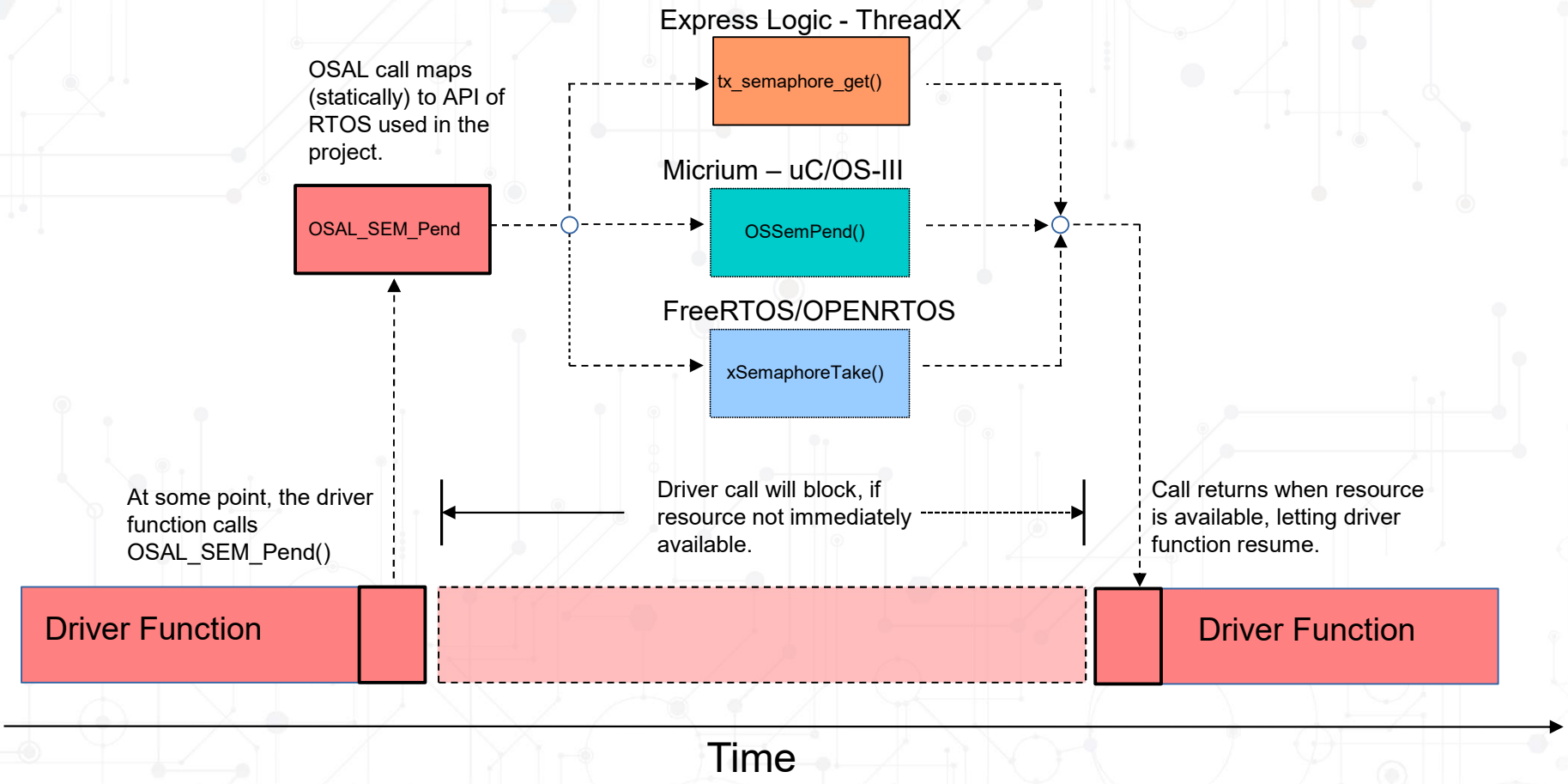


OS Abstraction Layer

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

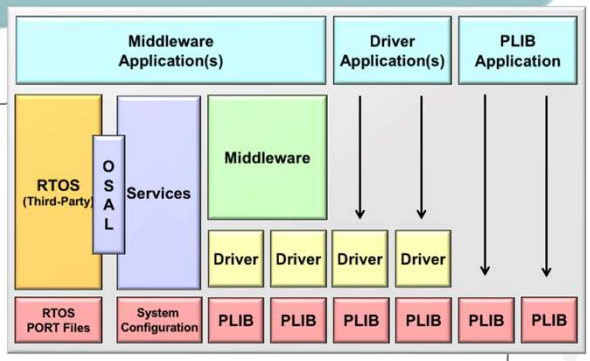


OSAL-to-RTOS Mapping



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. */
}
...
```

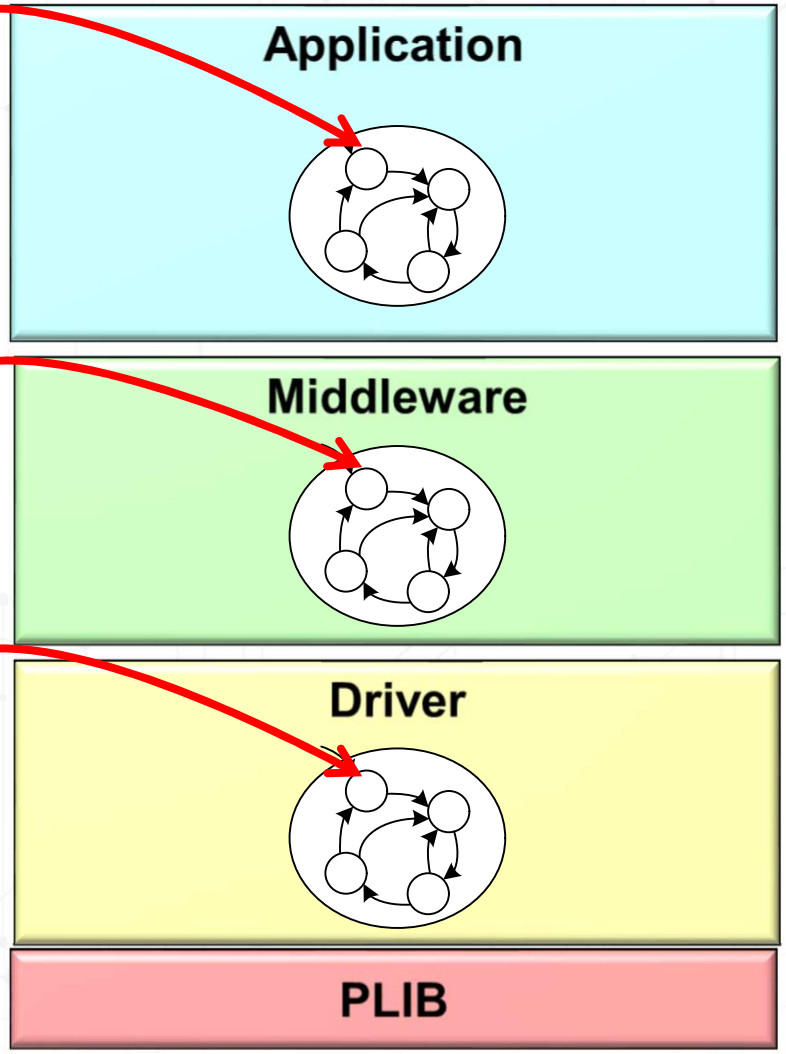
Thread Functions

FreeRTOS Specific

Modules Operate Independently

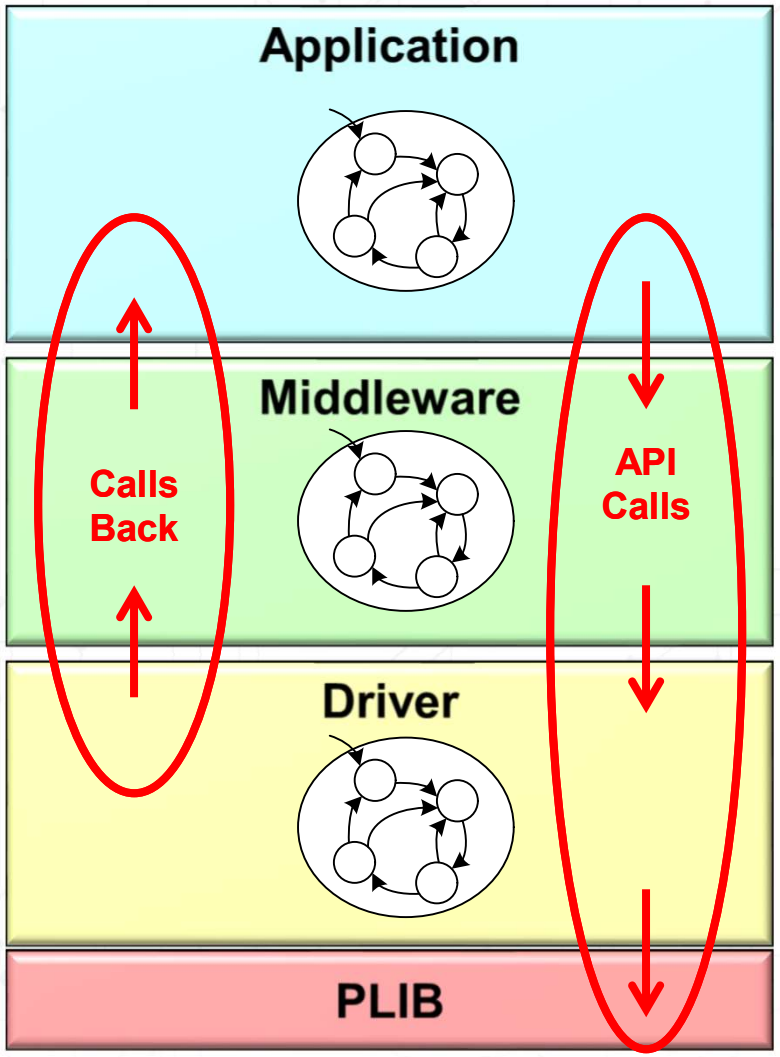
Effectively Independent

No Assumptions of Sequence or Timing Between modules



Modules Cooperate Interactively

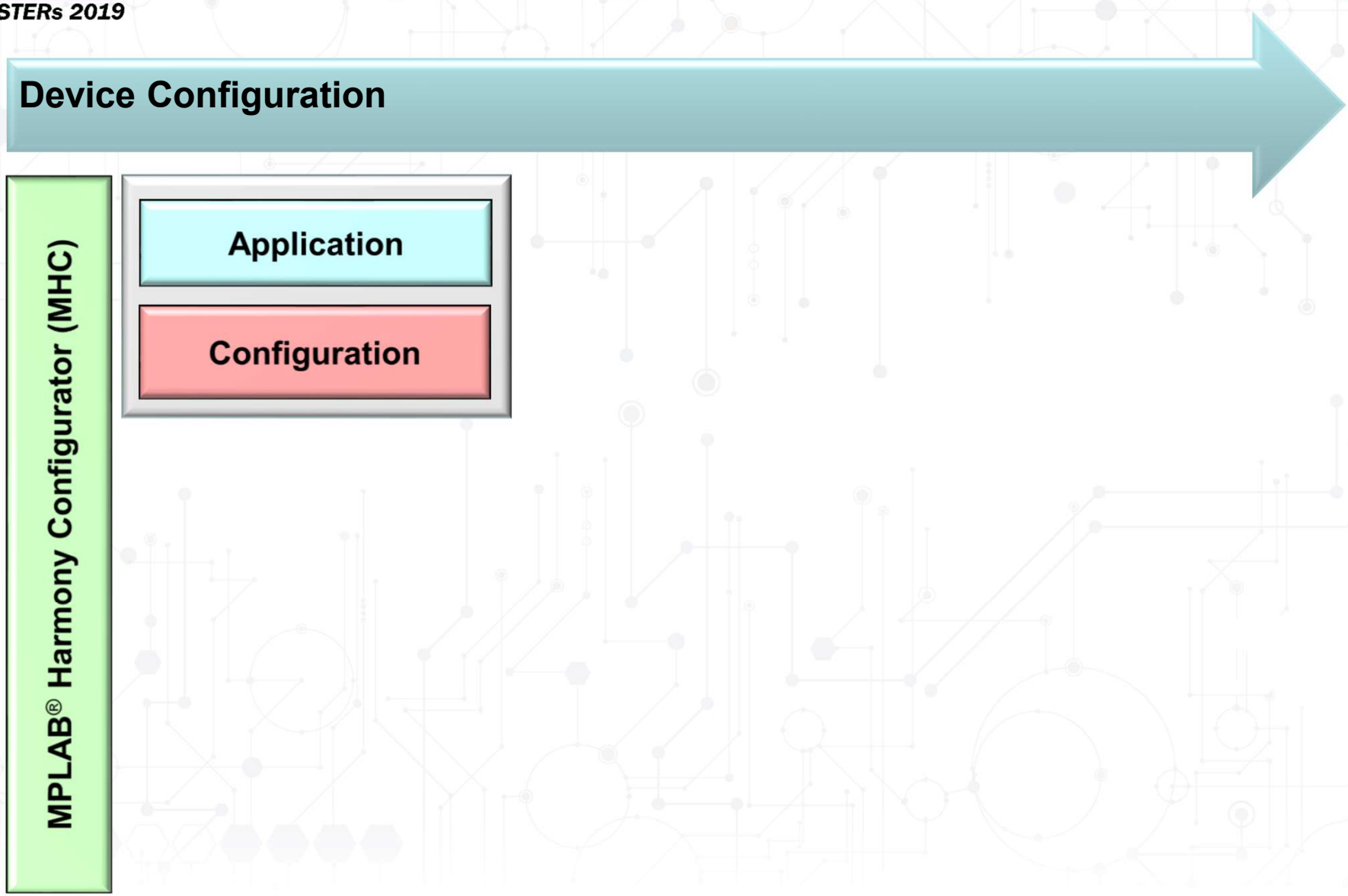
Modules Interact With Each Other Only Through Interface Functions



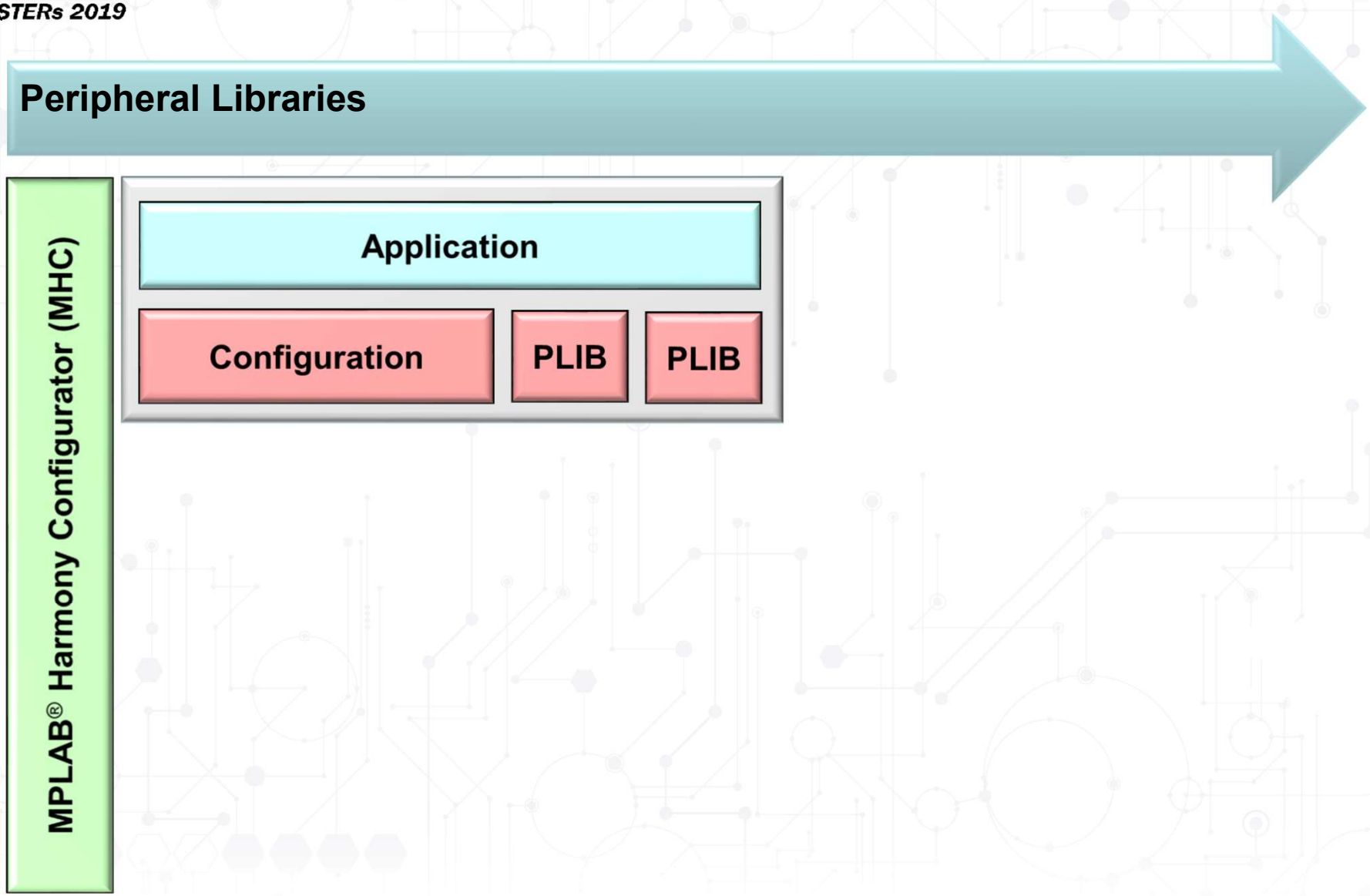
Agenda

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**

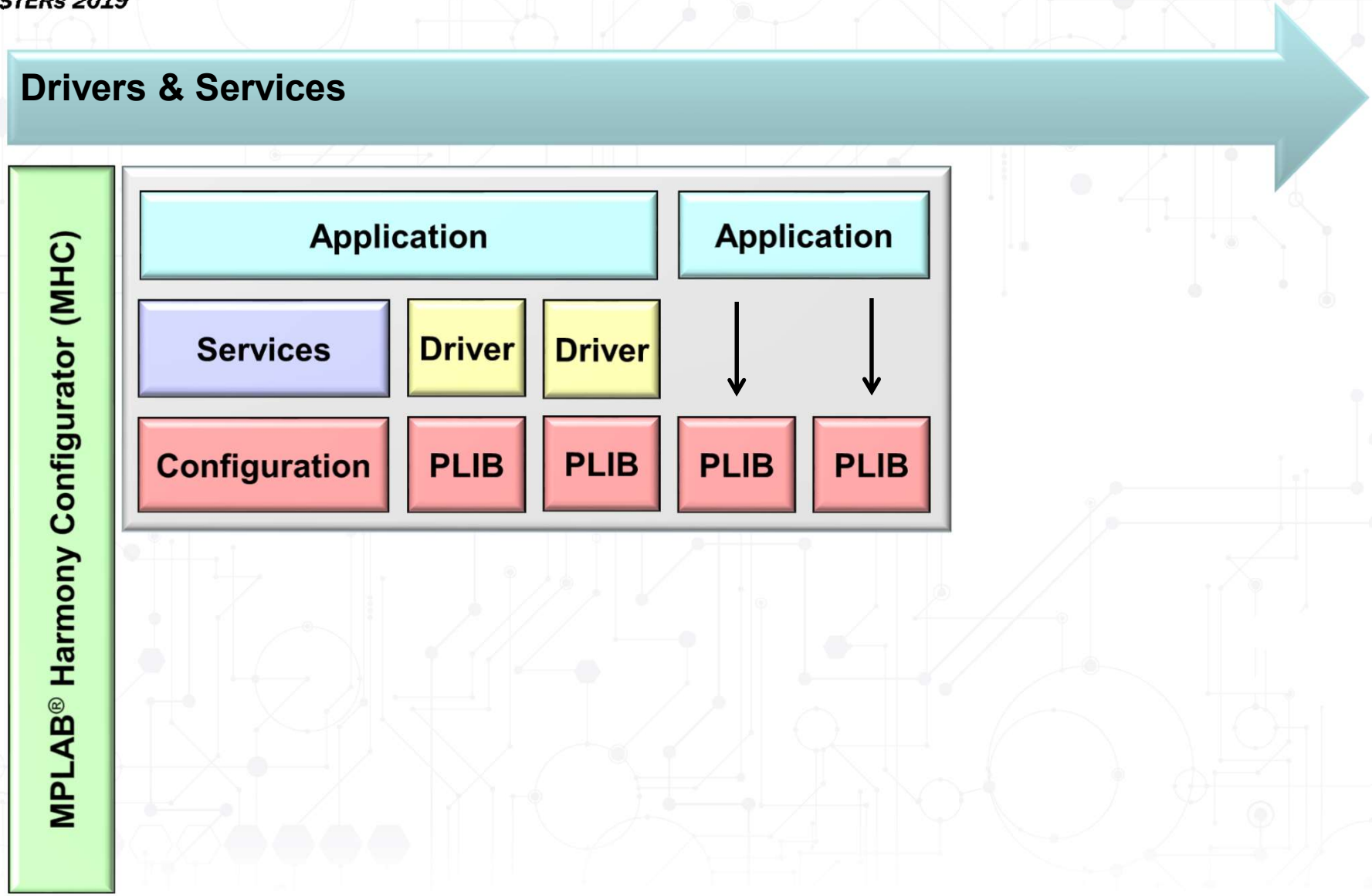
Minimal Initialization



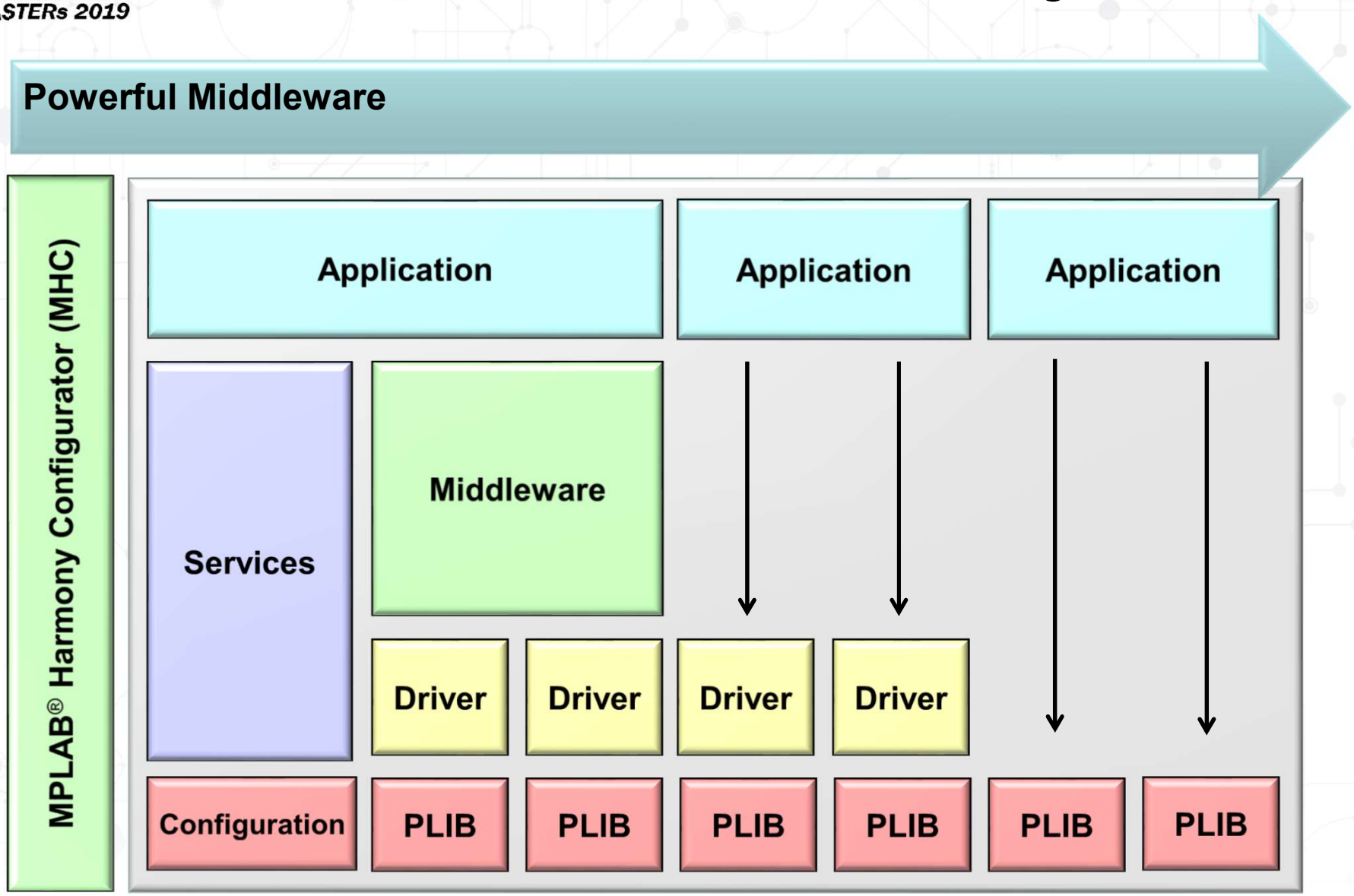
PLIB-based Project



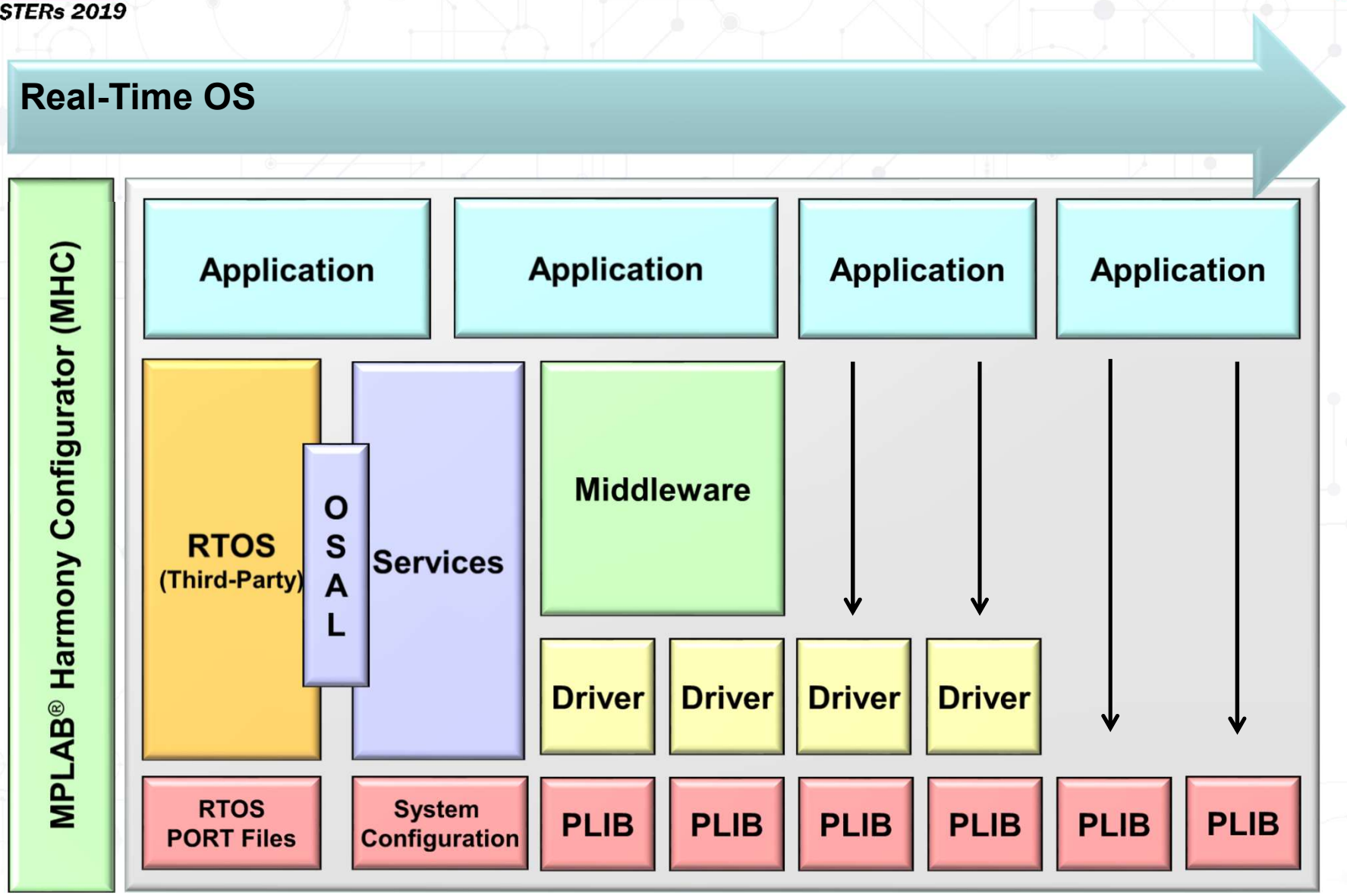
Driver-Based Project



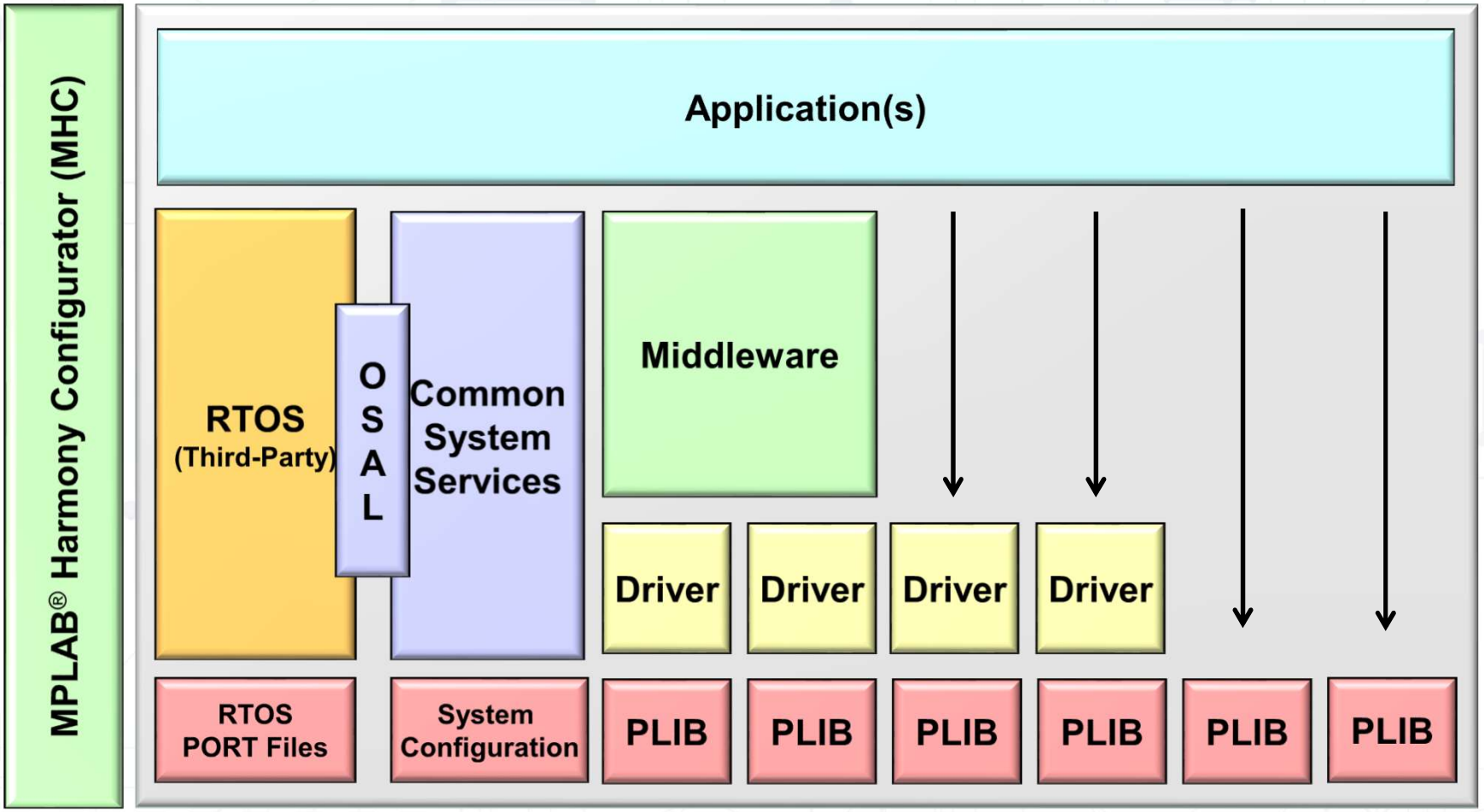
Middleware-Based Project



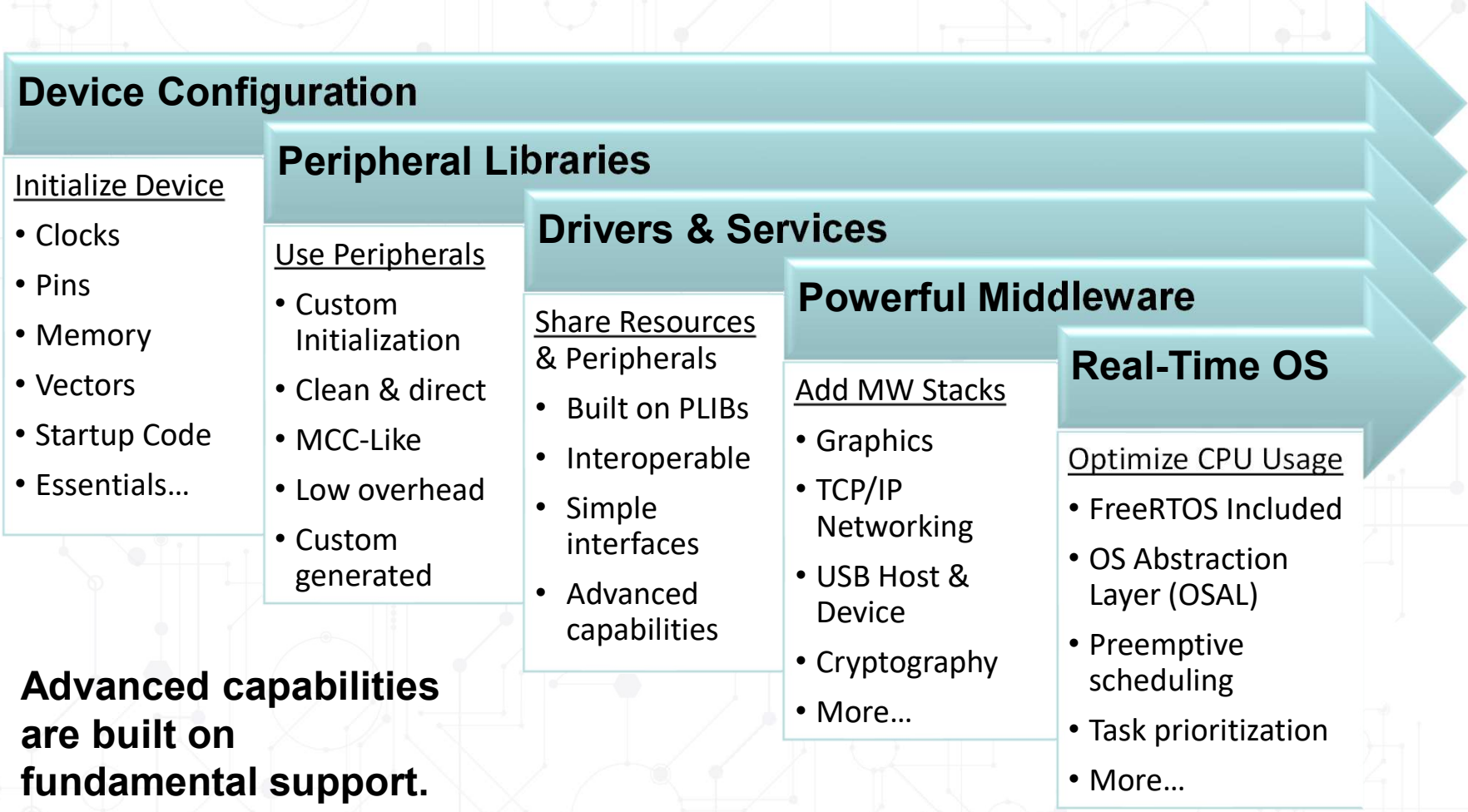
RTOS-Based Project



Embedded Software Model



Development Models



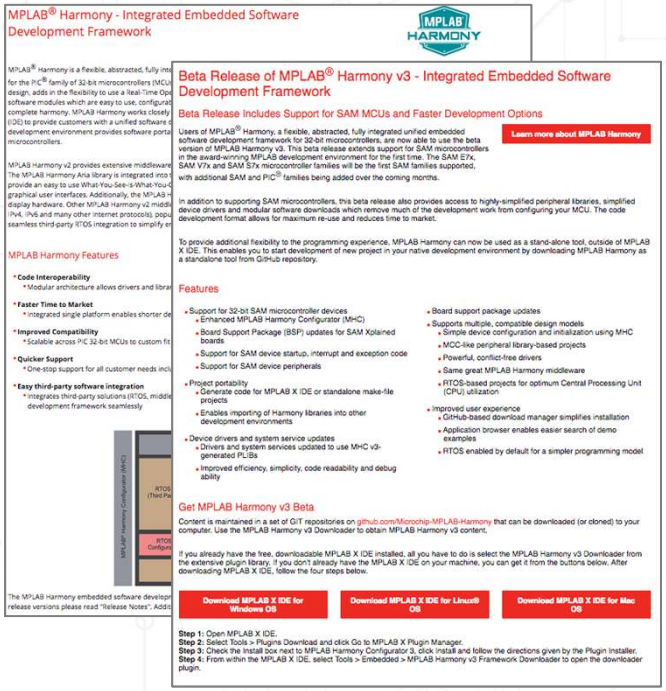
Advanced capabilities are built on fundamental support.

Choose the development model that best serves the applications needs!

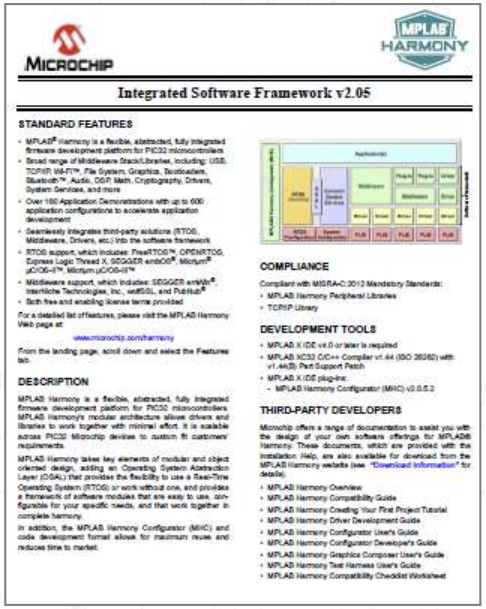
Main Website

www.microchip.com/harmony

- Marketing Collateral
- Access Directions



Harmony Landing Page(s)



Harmony Brochure

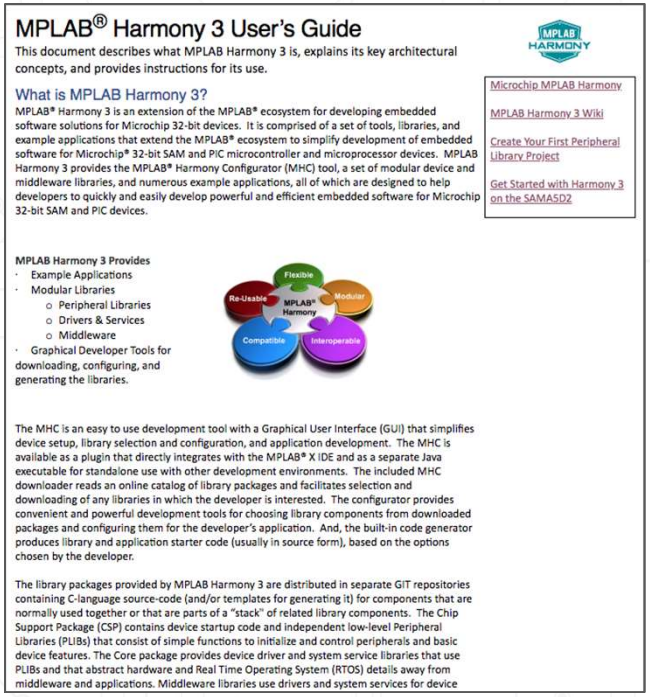
?

Other

No H3 Code

(H1 & H2 Archives)

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.

Microchip MPLAB Harmony

MPLAB Harmony 3 Wiki

Create Your First Peripheral Library Project

Get Started with Harmony 3 on the SAMA5D2

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
  - Peripheral Libraries
  - Drivers & Services
  - 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 driver 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.github.io

Main Harmony 3

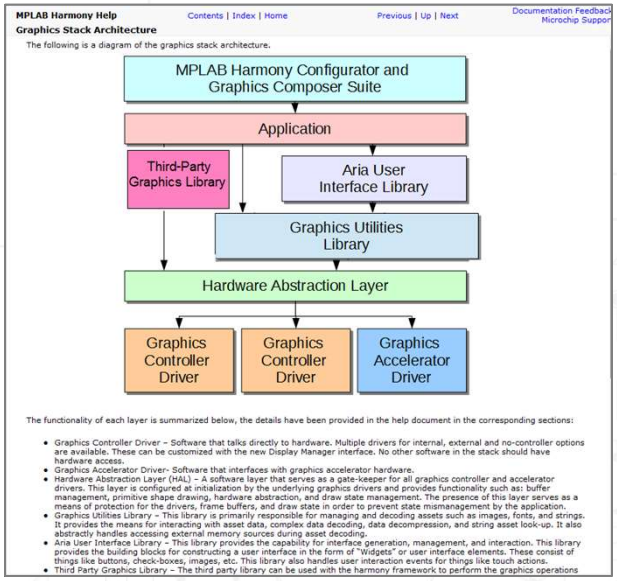
Introductory Material

- 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.



MPLAB Harmony Help — Contents | Index | Home — Previous | Up | Next — Documentation Feedback, Microchip Support

Graphics Stack Architecture

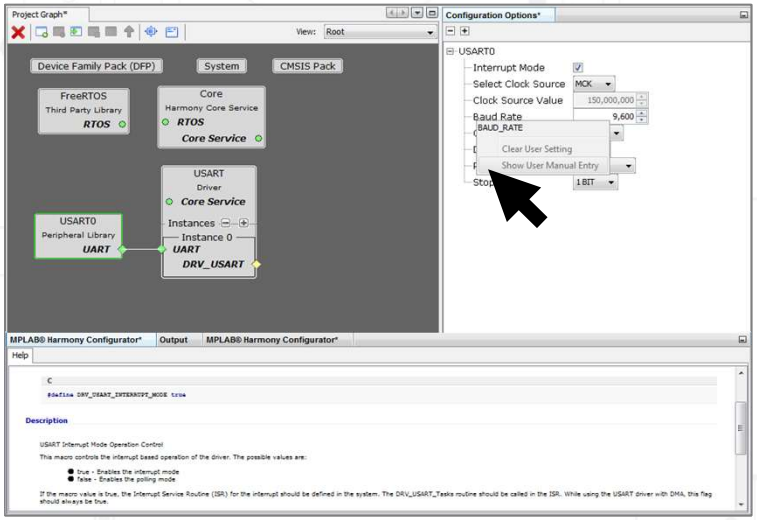
The following is a diagram of the graphics stack architecture.

The functionality of each layer is summarized below, the details have been provided in the help document in the corresponding sections:

- Graphics Controller Driver – Software that talks directly to hardware. Multiple drivers for internal, external and no-controller options are available. These can be customized with the new Display Manager interface. No other software in the stack should have hardware access.
- Graphics Accelerator Driver- Software that interfaces with graphics accelerator hardware.
- Hardware Abstraction Layer (HAL) – A software layer that serves as a gate-keeper for all graphics controller and accelerator drivers. This layer is configured at initialization by the underlying graphics drivers and provides functionality such as: buffer management, primitive shape drawing, hardware abstraction, and draw state management. The presence of this layer serves as a means of protection for the drivers, frame buffers, and draw state in order to prevent state mismanagement by the application.
- Graphics Utilities Library – This library is primarily responsible for managing and decoding assets such as images, fonts, and strings. It provides the means for interacting with asset data, complex data decoding, data decompression, and string asset look-up. It also abstractly handles accessing external memory sources during asset decoding.
- Aria User Interface Library – This library provides the capability for interface generation, management, and interaction. This library provides the building blocks for constructing a user interface in the form of "Widgets" or user interface elements. These consist of things like buttons, check-boxes, images, etc. This library also handles user interaction events for things like touch actions.
- Third Party Graphics Library – The third party library can be used with the harmony framework to perform the graphics operations

microchip-mplab-harmony.github.io/gfx

Reference
Interactive & Online

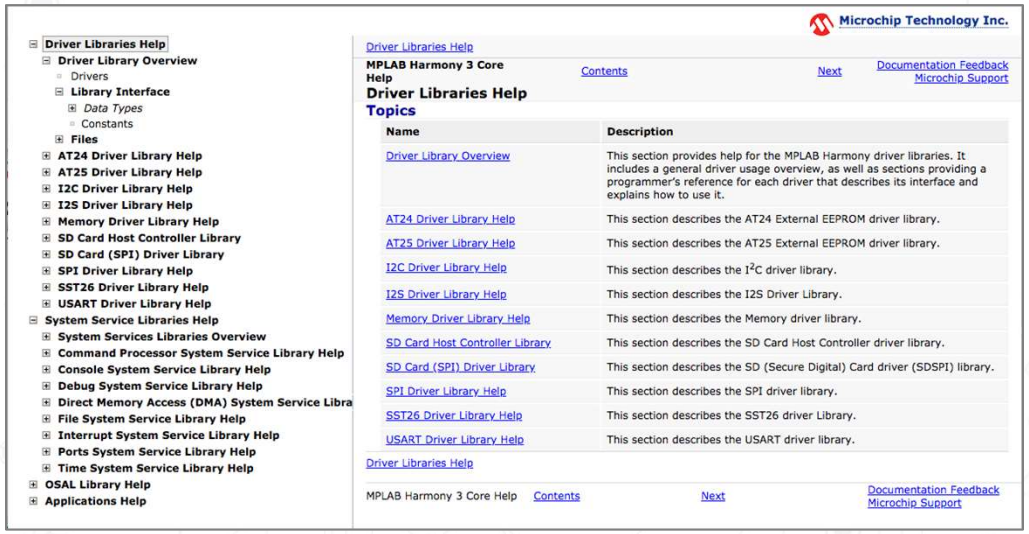


Module-Specific
Installed & Interactive
Configuration & API Help
(HTML + Python)

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

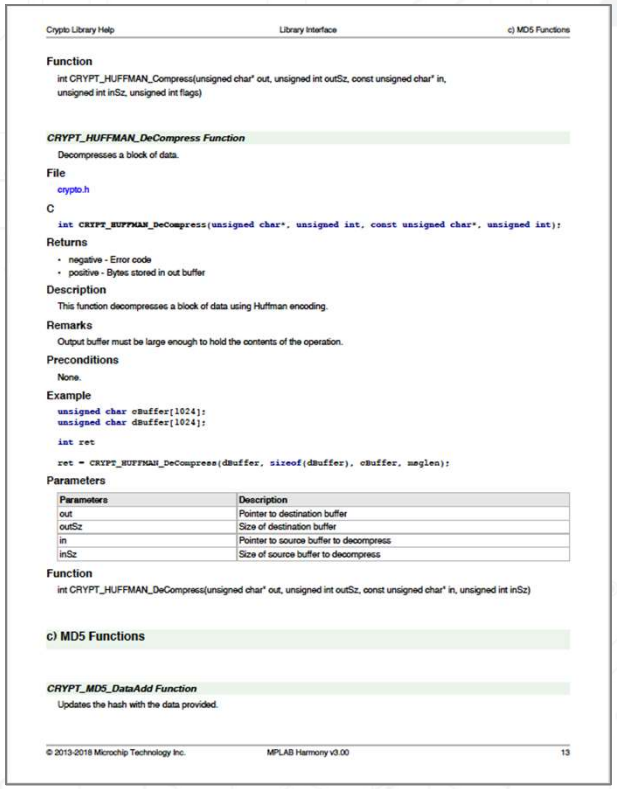
Module-Specific
Rendered Online Automatically
Configuration & API Help

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



microchip-mplab-harmony.github.io/core

Reference Downloadable


Crypto Library Help Library Interface c) MD5 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, msglen);
```

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) MD5 Functions

CRYPT_MD5_DataAdd Function

Updates the hash with the data provided.

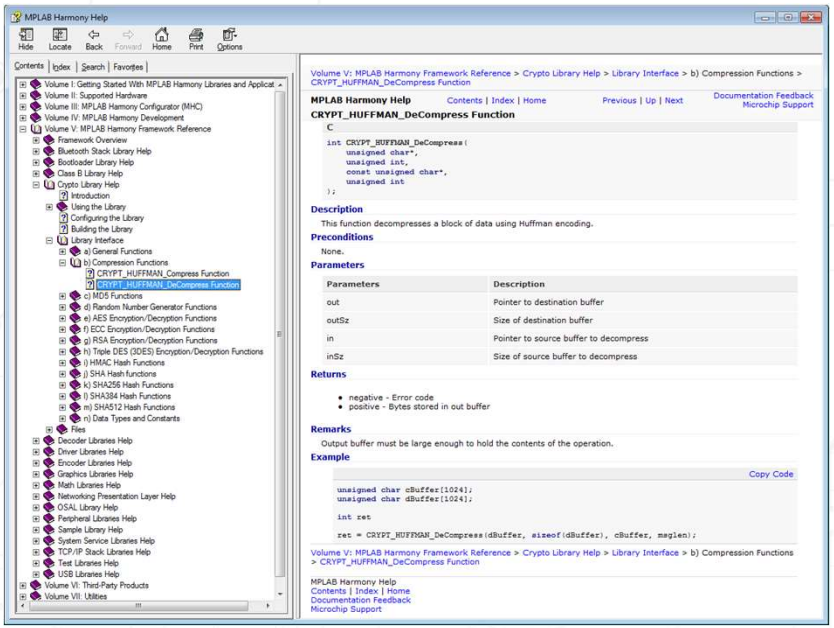
© 2013-2018 Microchip Technology Inc. MPLAB Harmony v3.00 13

Module specific Downloadable from GitHub

Configuration & API Help

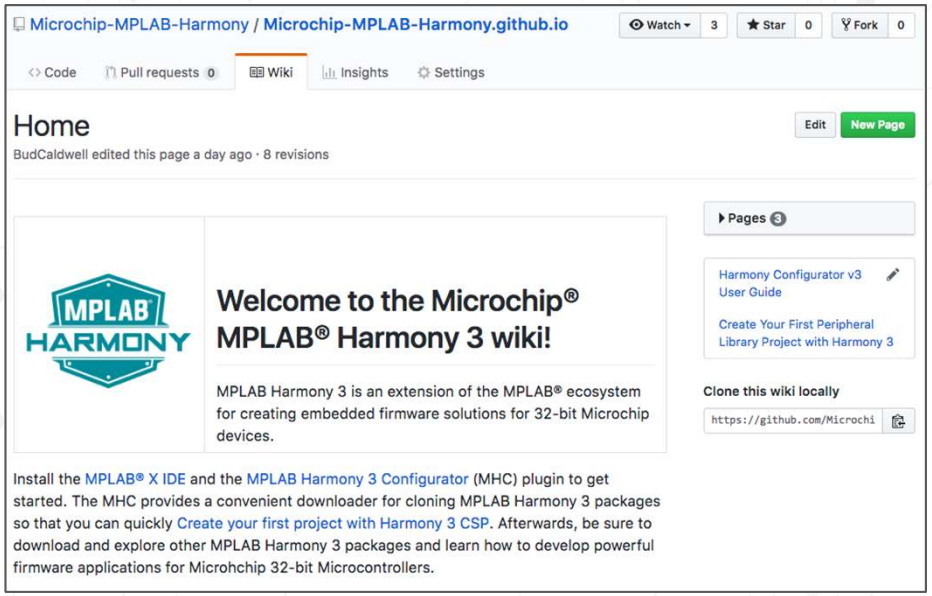
(CHM, PDF)

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



Included in ”docs” folder of each repository.

Directory
Online & Downloaded



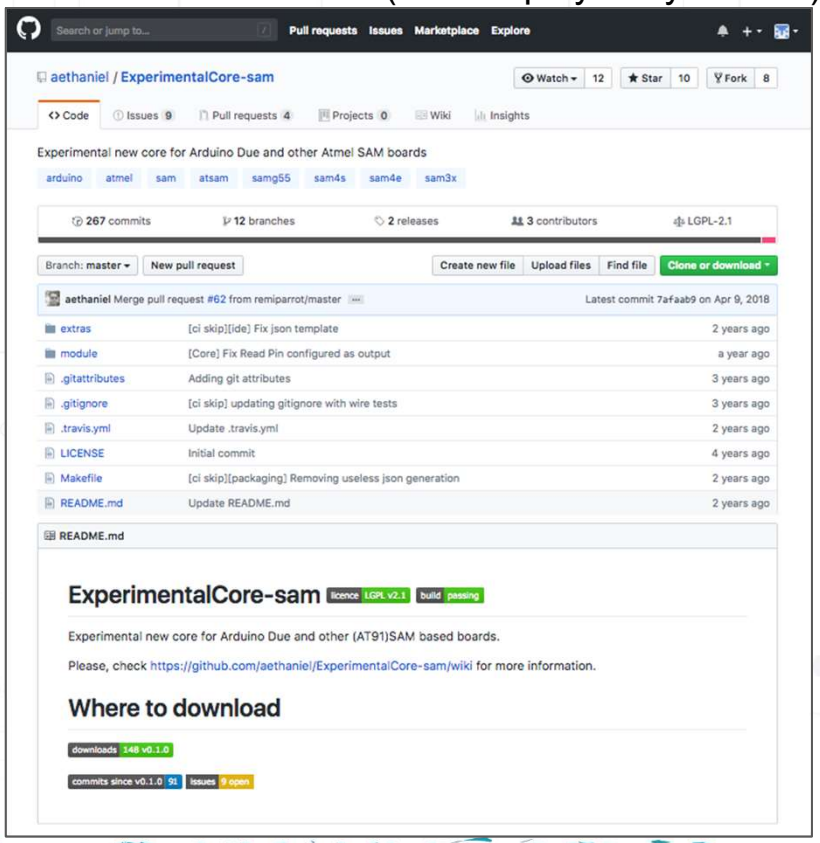
github.com/[Microchip-MPLAB-Harmony]²/wiki

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).



Doc Summary

Microhip.com/Harmony

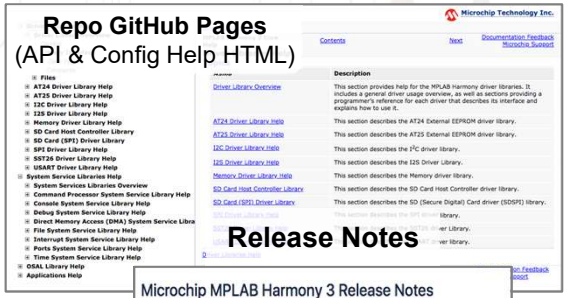
GitHub H3 Wiki

H3 Device Support

Welcome Page

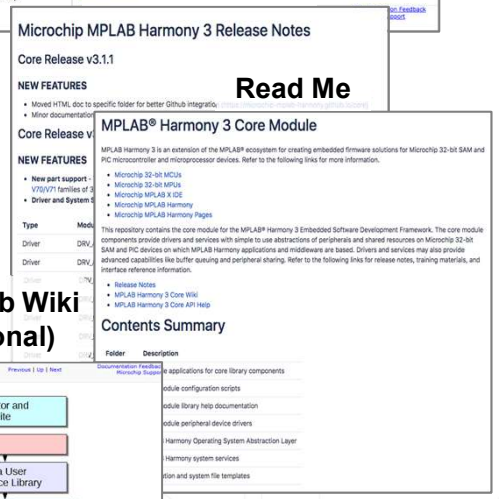
Repo Info

Repo GitHub Pages
(API & Config Help HTML)

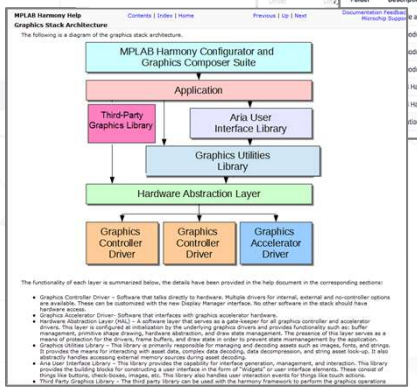


Release Notes

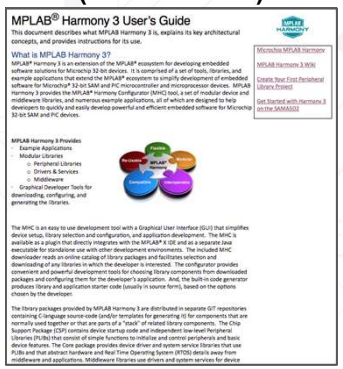
Read Me



Repo GitHub Wiki
(Instructional)



GitHub H3 Pages
(Instructional)



On GitHub Web

- HTML & Wiki
- Readme
- Release Notes

Downloadable

- CHM & PDF

Thank You!

LEGAL NOTICE

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 3rd 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 3rd party companies and 3rd party websites) are for informational purposes only and provided "AS IS." Microchip assumes no responsibility for statements made by 3rd party companies, or materials or information that such 3rd 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 POSSIBLITY 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.