

CAE專家教室 SAM2002系列-01
SAMD21 ARM Cortex-M0+
Microcontroller & MCC Harmony
SAM2002 v1.1



A Leading Provider of Smart, Connected and Secure Embedded Control Solutions



SMART | CONNECTED | SECURE

CAE Taiwan Team
Microchip Technology Inc.

October 30, 2024

SAM2002-01 Index

- [Software Request](#)
- [SAMD21 EVB APP045 EVB Introduction](#)
- [MPLAB® MCC Harmony Key Concept](#)
- [Harmony System Service introduction](#)
 - [TIME System Service](#)
 -  [A New MCC Harmony Project](#)
 -  [Lab1 – Delay based LED toggle](#)
 -  [Lab2 – Callback based LED toggle](#)
 - [CONSOLE System Service](#)
 -  [Lab3 –UART Console](#)
 - [DEBUG System Service](#)
 -  [Lab4 –Debug Message Control](#)
 - [COMMAND Processor System Service](#)
 -  [Lab5 –Custom Command Console](#)

SAM2002-01 Index

- USB Device CDC Library

-  Lab6 – USB Device CDC with CONSOLE System Service

- FILE System Service

-  Lab7 –SD Card with File System

-  Lab8 –FILE System Auto Mount Feature

-  Lab9 –USB Host Mass Storage with File System

-  Lab10 –NVMCTRL Internal Flash with File System

Soft Request In this Course

(Lasted Update : 2024/10/31)

- **MPLAB® X IDE v6.20** <https://www.microchip.com/mplab/mplab-x-ide>
- **MPLAB® XC32 v4.45** <https://www.microchip.com/mplab/compilers>
- **MPLAB® Code Configurator v5.5.1** (Install in MPLAB X IDE Plug-in)
- **SAMD21 DFP v3.6.144** (Device Family Pack download from X IDE \Tools\Packs)
- **CMSIS v6.1.0** (Common Microcontroller Software Interface Standard)
- **MCC Core v5.7.1** (MCC Content Manager)
- **MPLAB® MCC Harmony Framework** (From X IDE MCC Harmony Content Manager)
 - **Required Content** (Must install)
 - Csp 3.19.6
 - harmony-services 1.5.0
 - CMSIS_5 5.9.0
 - quick_docs 1.7.0
 - **Optional Content** (Manual select in Content Manager)

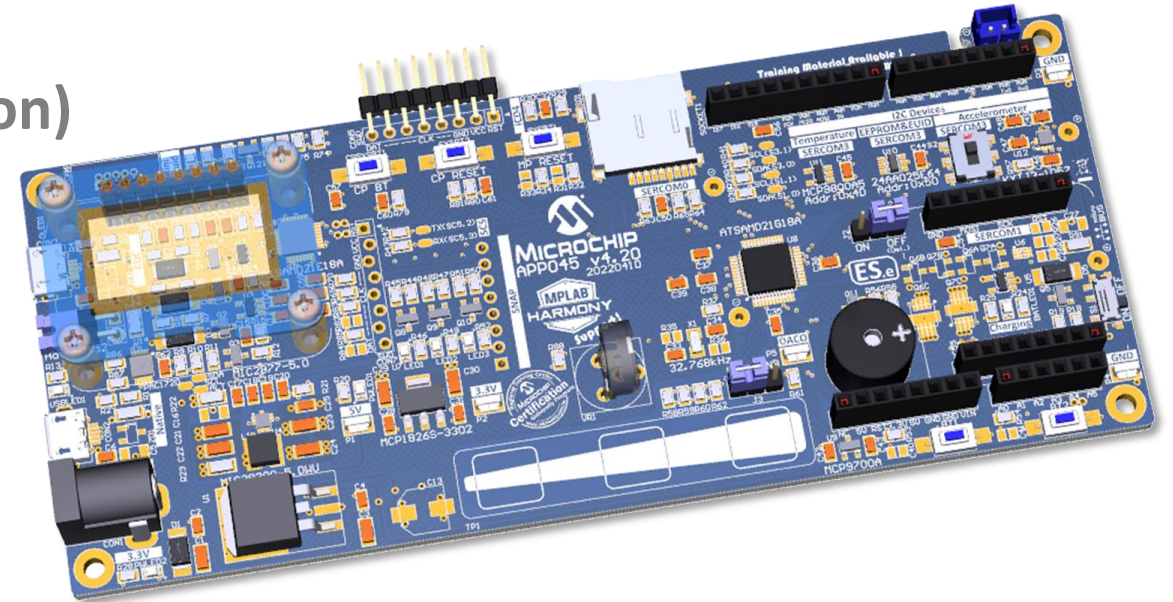
• wireless_wifi v3.11.1	• filex v6.4.0_rel
• net v3.12.3	• littlefs v2.9.3
• usb v3.14.0	• zlib v1.3.1
• dev_packs v3.18.1	• Wolfssl v5.7.0
• wolfMQTT v1.11.1	• CMSIS-FreeRTOS v10.5.1
• crypto v3.8.2	• bsp v3.20.1
• core v3.13.5	

SAMD21 EVB

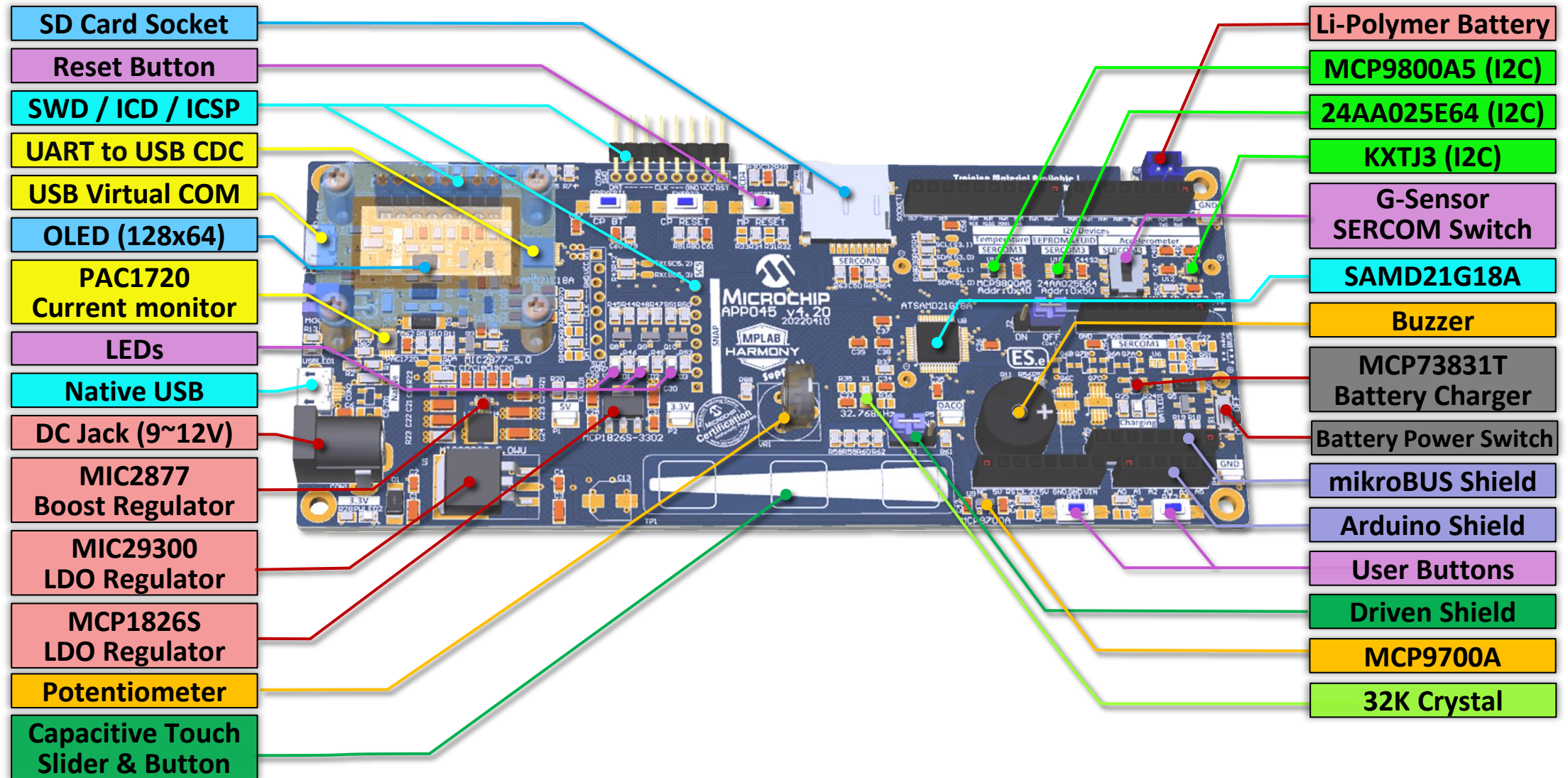
APP045 v4.2 ES.e Introduction

APP045 v4.2 ES.e Features

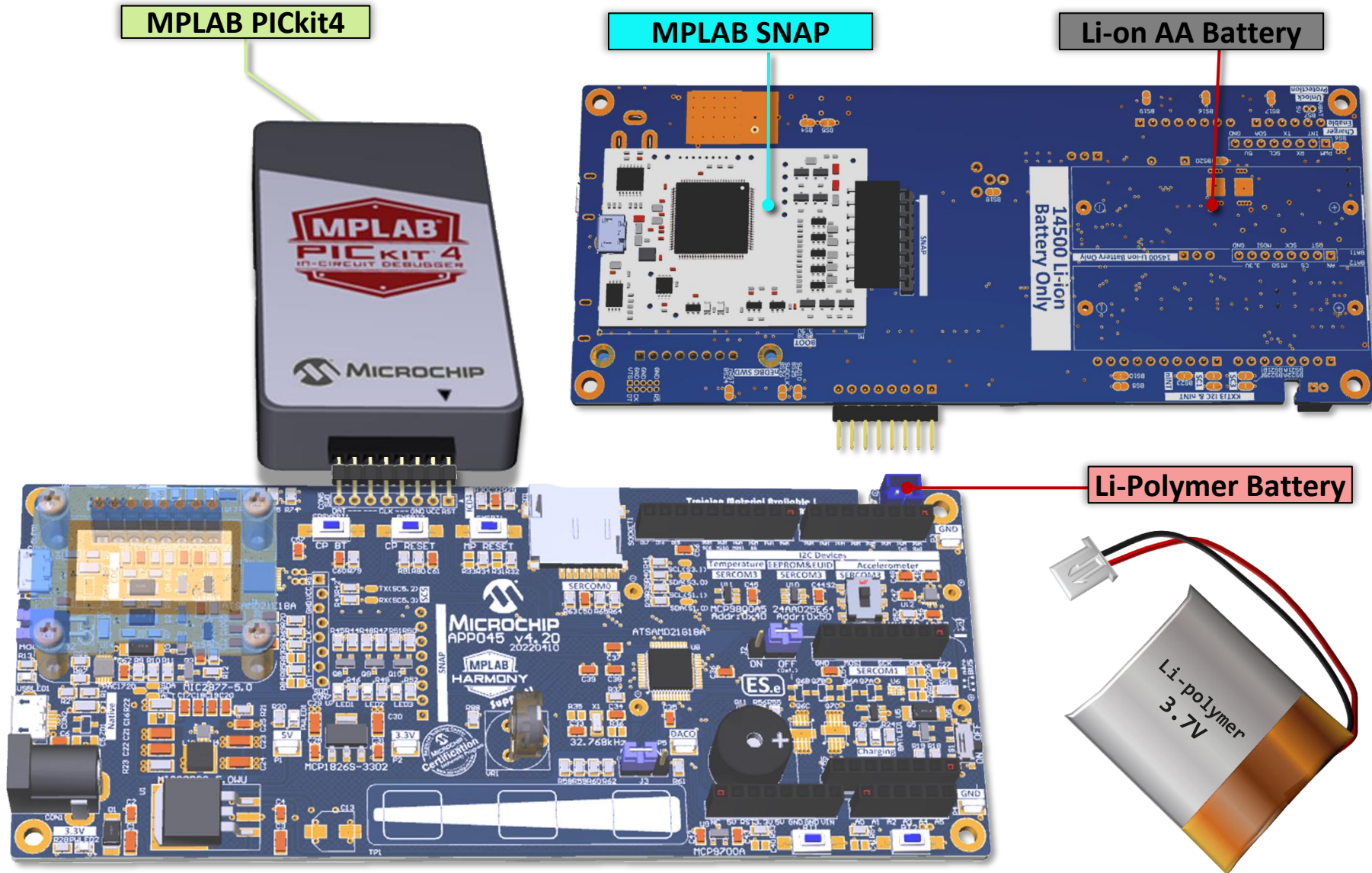
- On board **SAMD21G18A** 32Bit ARM Cortex M0+ Microcontroller
- Support plugged **MPLAB SNAP** external programmer/debugger
- Onboard **USB CDC Virtual COM Port** and **Power Current Monitor**
- Multiple Power Supply Source
- Li-on Battery, boost/charger supported (Option)
- 2 x Buttons, 3 x LEDs
- 1 x I2C Temperature Sensor
- 1 x I2C EEPROM with Unique ID
- **1 x I2C Accelerometer**
- 1 x MicroSD Socket (SPI interface)
- 1 x SPI OLED (128 x 64, SSD1306 Compliant)
- 1 x Potentiometer, Analog Temperature Sensor and **Buzzer**
- **1 x Capacitive QTouch® Slider & Buttons with Driven Shield**
- **1 x mikroBUS™ Click Board Socket**
- 1 x Arduino® Shield Socket (Arduino M0 Pro board Compliant)



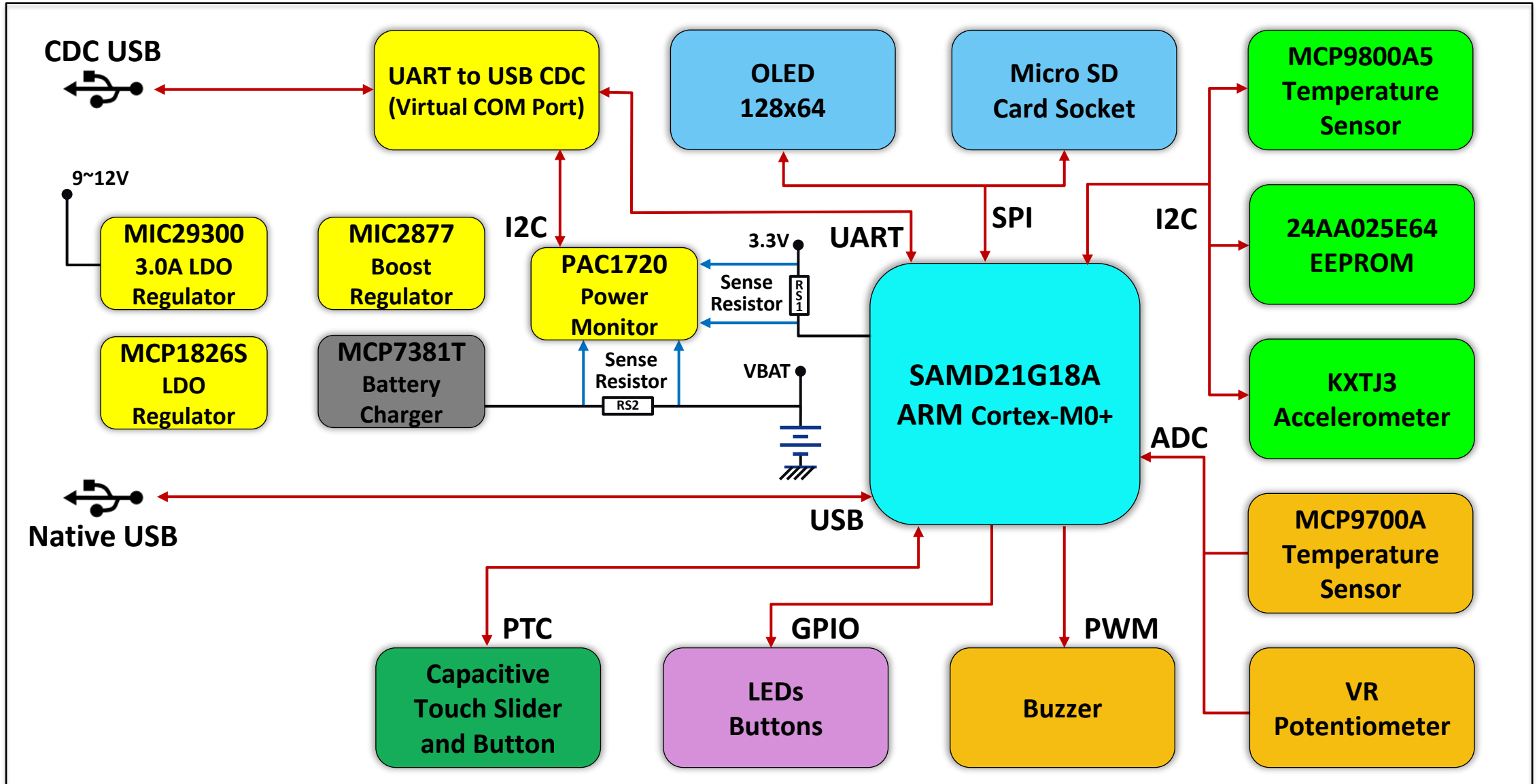
APP045 v4.2 ES.e Overview



APP045 v4.2 ES.e Programmer and Debugger



APP045 v4.2 ES.e Block Diagram

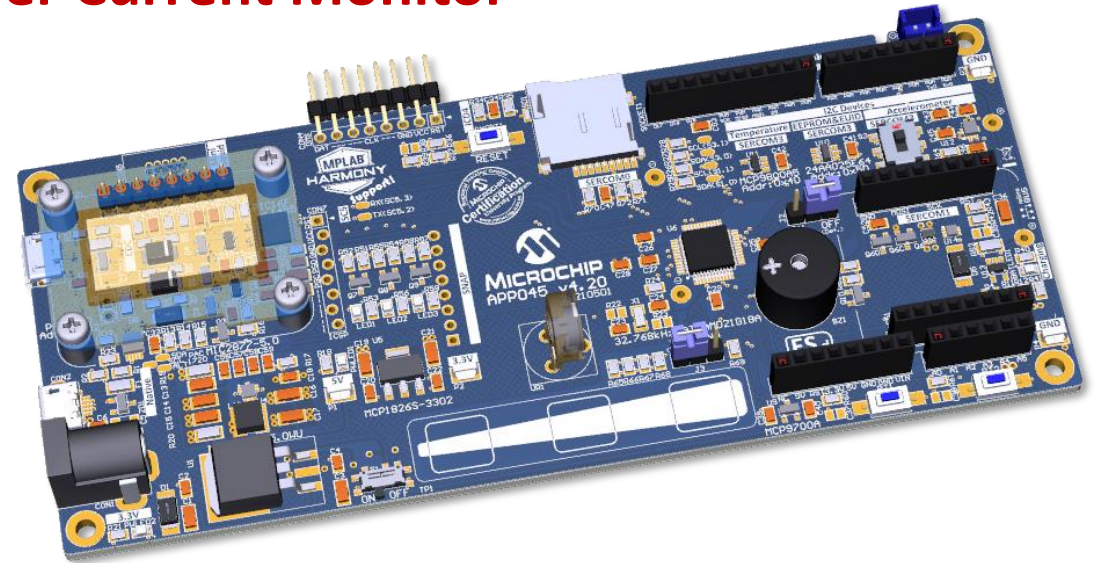


SAMD21 EVB

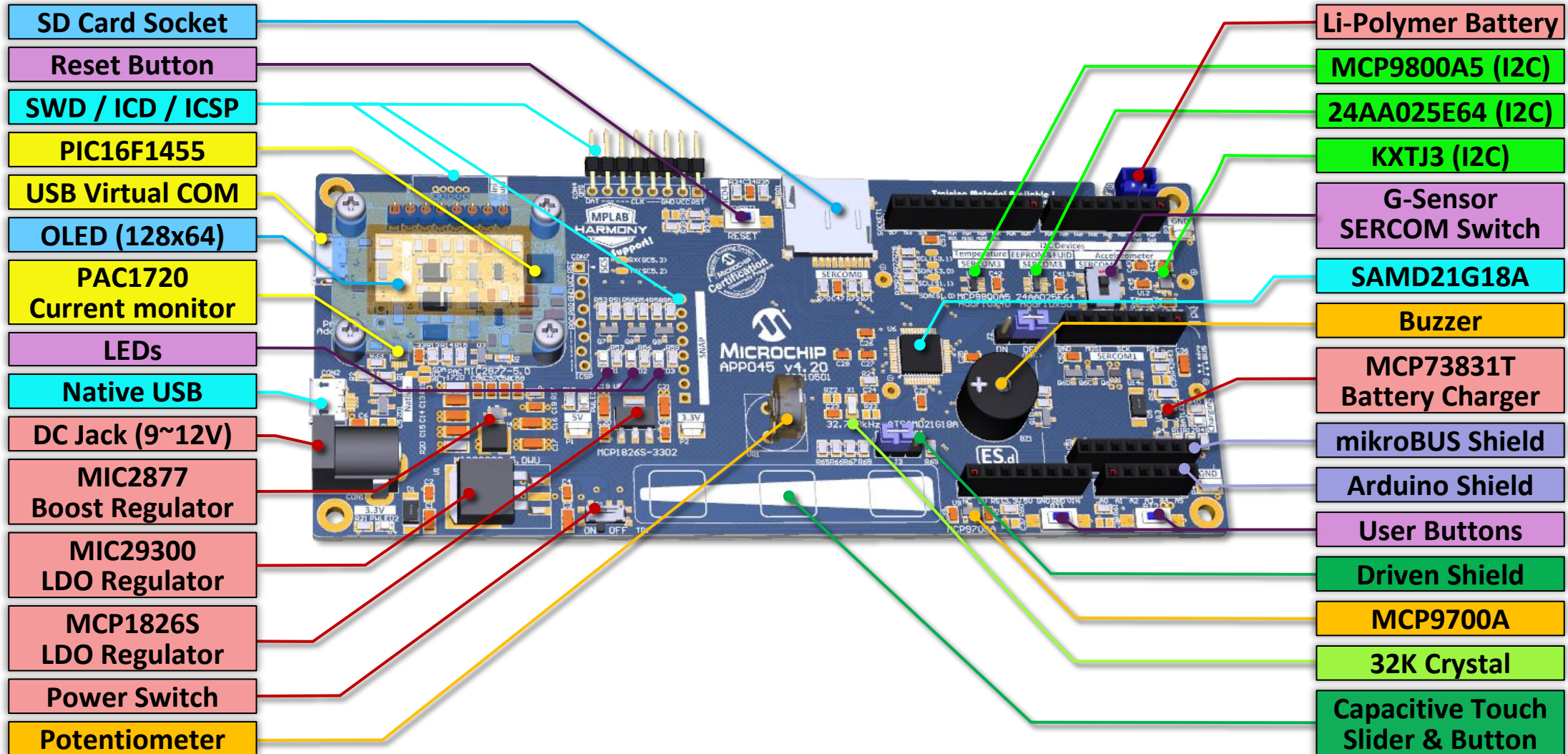
APP045 v4.2 ES.d Introduction

APP045 v4.2 ES.d Features

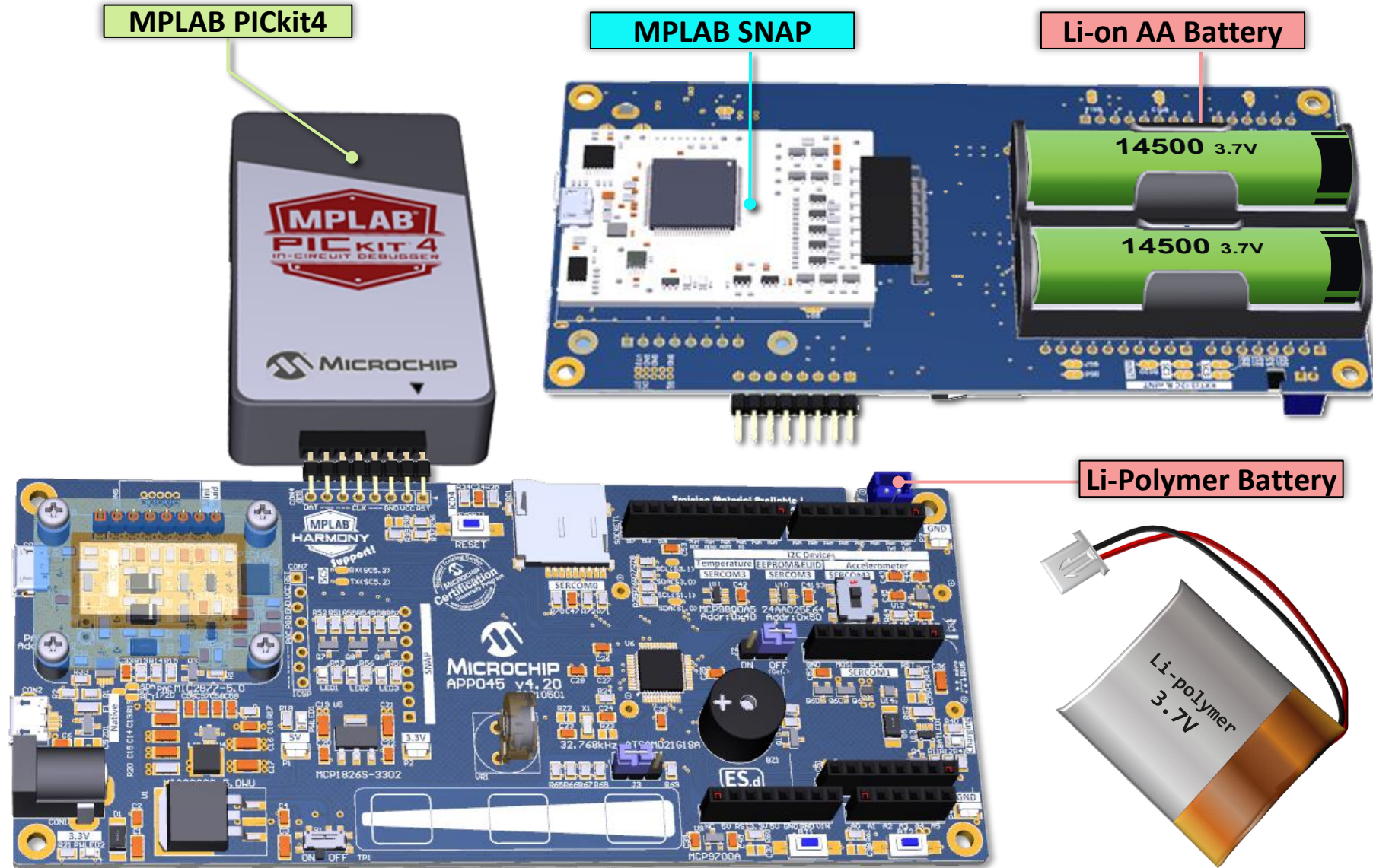
- On board **SAMD21G18A** 32Bit ARM Cortex M0+ Microcontroller
- Support plugged **MPLAB SNAP** external programmer/debugger
- Onboard **USB CDC Virtual COM Port** and **Power Current Monitor**
- Multiple Power Supply Source
- **Support Li-on Battery, boost and charger**
- 2 x Buttons, 3 x LEDs
- 1 x I2C Temperature Sensor
- 1 x I2C EEPROM with Unique ID
- **1 x I2C Accelerometer**
- 1 x MicroSD Socket (SPI interface)
- 1 x SPI OLED (128 x 64, SSD1306 Compliant)
- 1 x Potentiometer, Analog Temperature Sensor and **Buzzer**
- **1 x Capacitive QTouch® Slider & Buttons with Driven Shield**
- **1 x mikroBUS™ Click Board Socket**
- 1 x Arduino® Shield Socket (Arduino M0 Pro board Compliant)



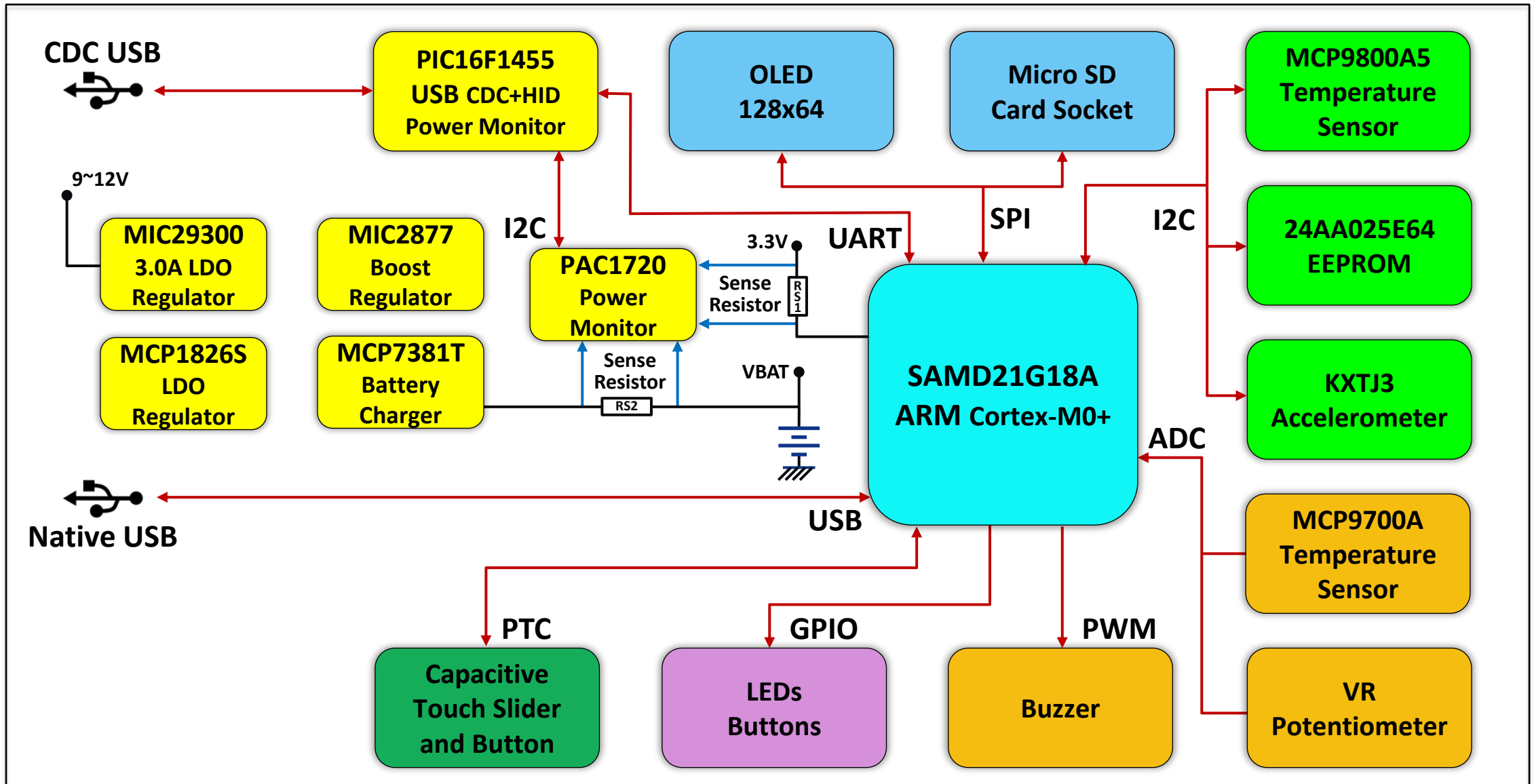
APP045 v4.2 ES.d Overview



APP045 v4.2 ES.d Programmer and Debugger



APP045 v4.2 ES.d Block Diagram

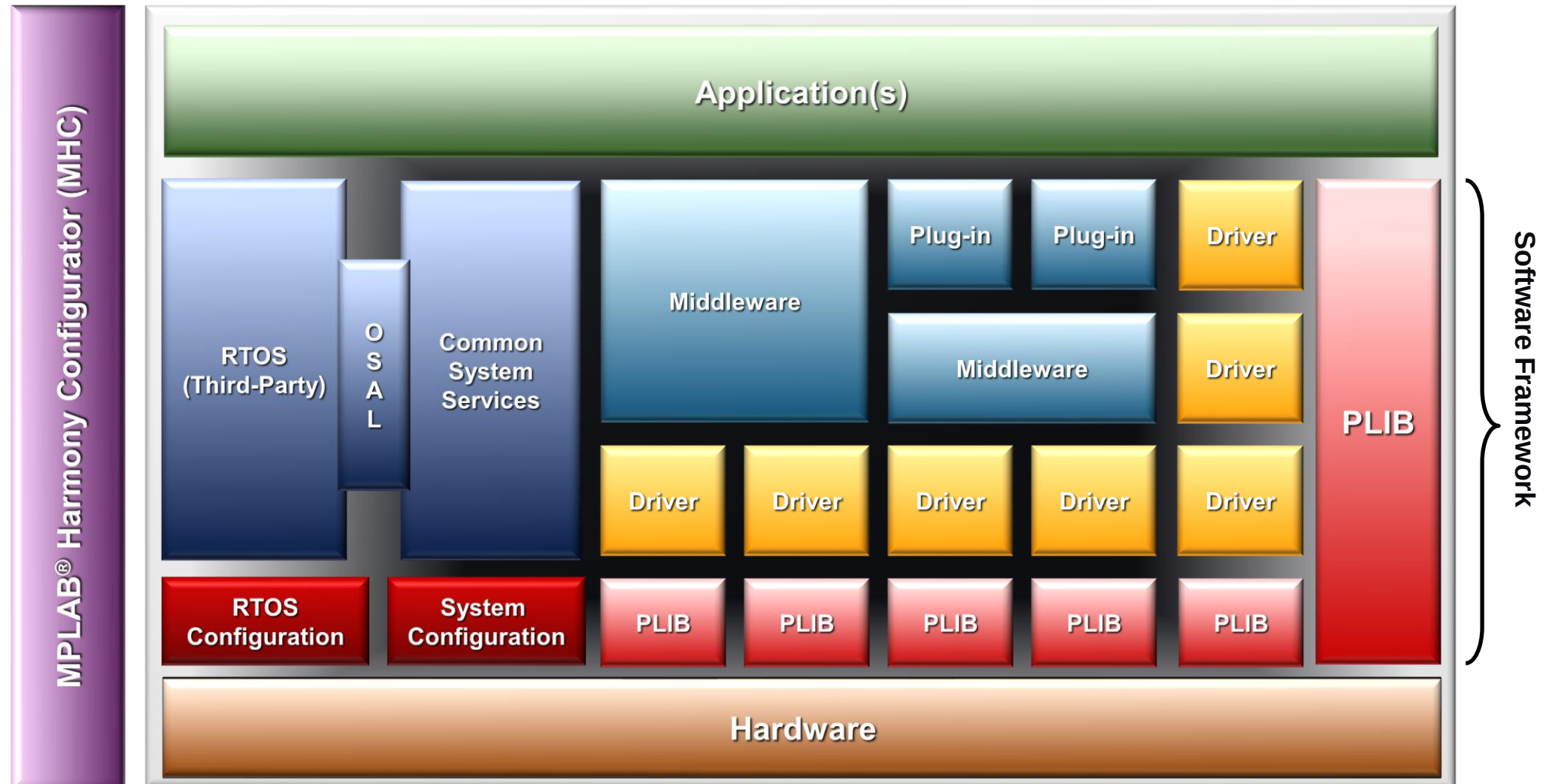


MPLAB® Harmony Key Concept

MASTERS 19 DEV8 referenced

MPLAB® MCC Harmony Key Concept

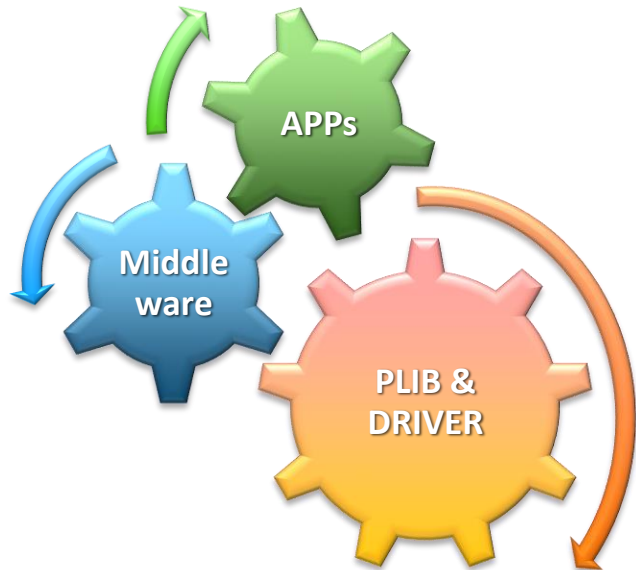
Architecture



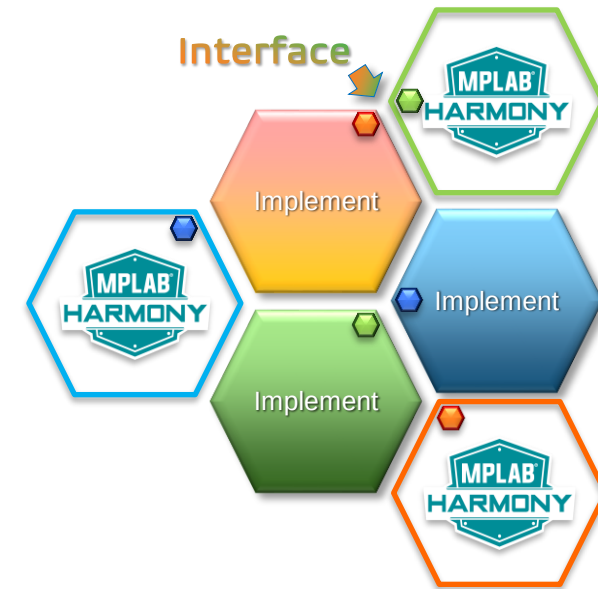
MPLAB® MCC Harmony Key Concept

Modularity and Compatibility

- Implements a single functionality
- A module has an interface and an implementation
- Hides implementation from the user
- Modules interact through interface
- No access to internal data



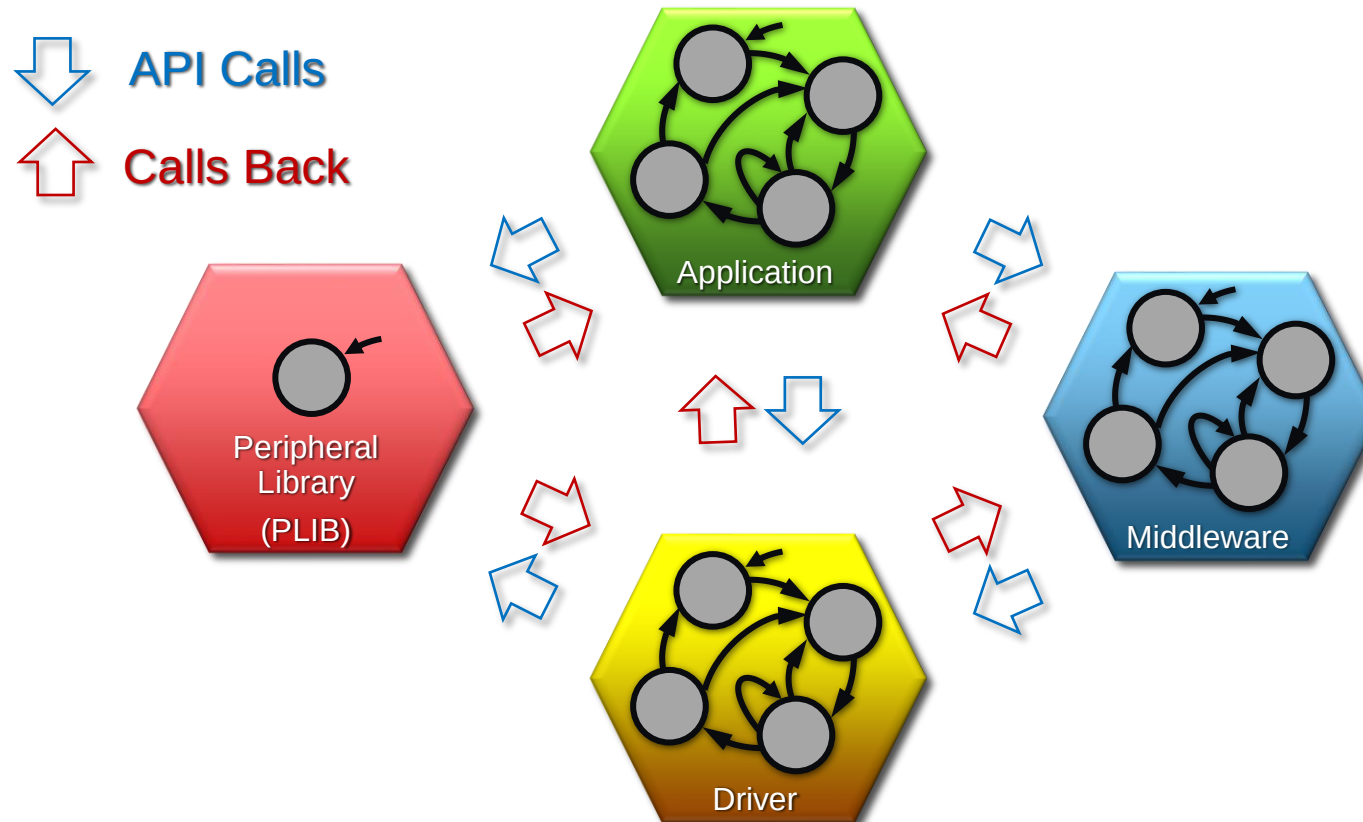
- Modules are compatible
- Adding new modules does not break existing functionality
- State machine ready for RTOS



MPLAB[®] MCC Harmony Key Concept

Compatible State Machines

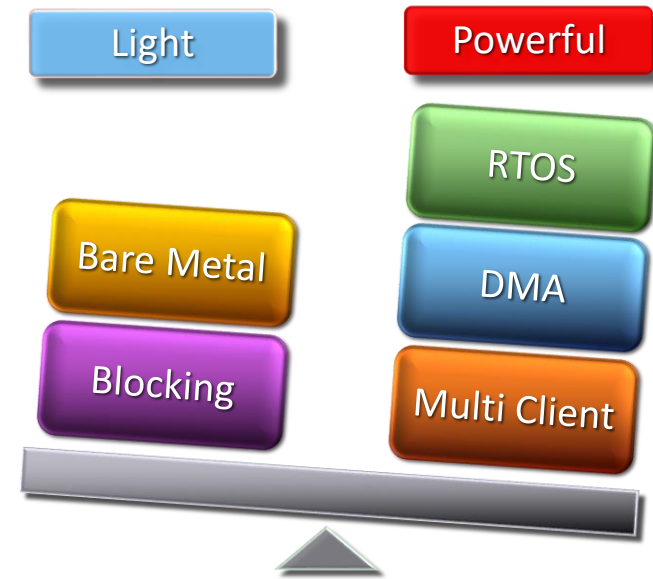
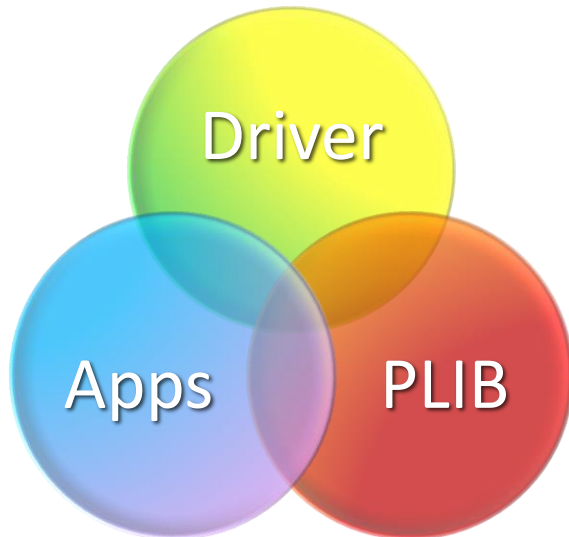
- Modules interact with each other through Interface Functions.
- Effectively Independent.
- No assumptions of sequence or timing between modules.



MPLAB® MCC Harmony Key Concept

Configurability and Flexibility

- Single-client or multi-client
- Bare-metal or RTOS
- Blocking or Non-blocking
- DMA or Non-DMA
- Other library specific options



- Use only PLIBs or combination of drivers and PLIBs
- Porting the project from one folder/machine to another
- Modular download

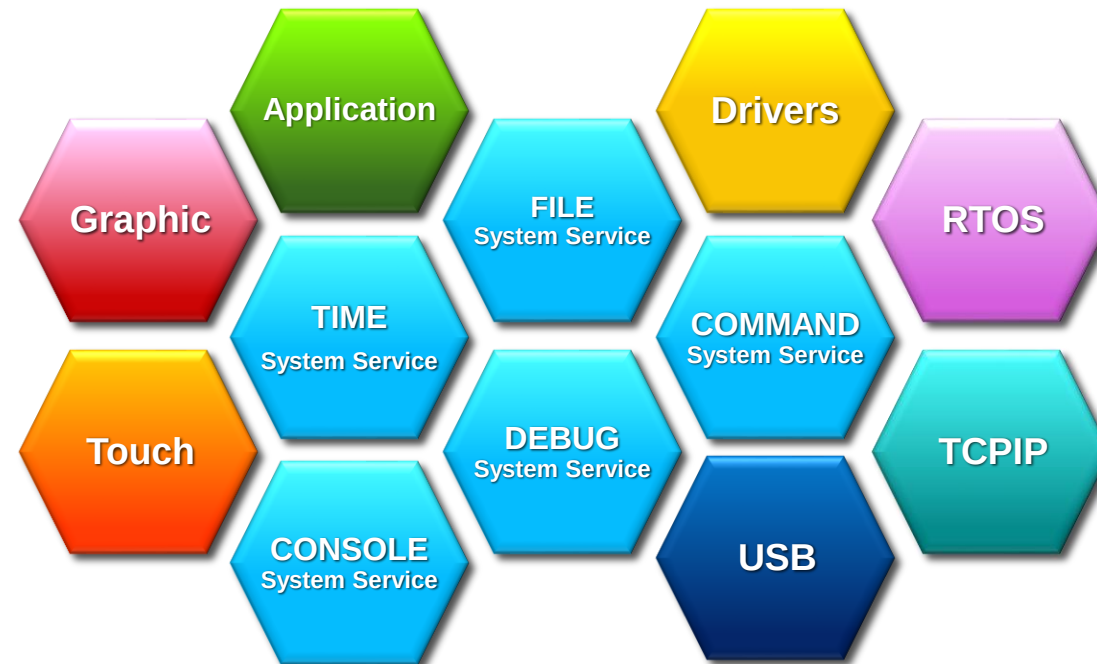
Harmony System Service Introduction

Manage the Common and Shared Resources

Harmony System Service Introduction

Manage the Common and Shared Resources

- Provides common functionality required by different drivers or library modules.
- Manages shared resources.
- Eliminates potential conflicts.
- Keeps the requests separate.



Harmony System Service Introduction

- **TIME System Service**
- **CONSOLE System Service**
- **DEBUG System Service**
- **COMMAND processor system service**
- **FILE SYSTEM service**

TIME System Service

Harmony System Service

TIME System Service

- **Introduction**

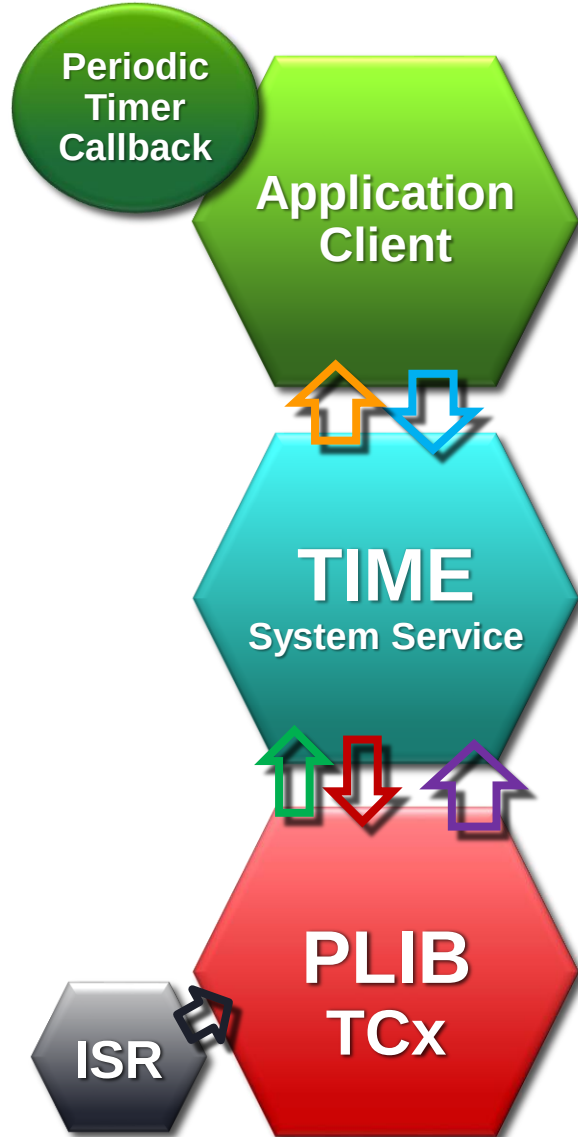
The Time System Service Library can provide periodic or one-shot alarm and delays to the user.

It uses a hardware timer peripheral library for providing its timing services.

- **Key Features:**

- Periodic notifications
- One-shot/single notification
- Delays
- Support for both tick-based and tick-less modes
 - Tick-less mode:
Instead of getting interrupted from the hardware timer periodically, the Time System Service manages the hardware timer so as to generate an interrupt only when required and at appropriate times.
 - Tick-based mode:
A periodic interrupt is generated by the hardware timer. The tick-rate is configurable by the user and defines the resolution for all the timers.

TIME System Service



Tick-less and Tick-based

- Request for periodic or non-periodic timeouts
- Set next compare match value (tick-less)
- Timer Compare Interrupt (tick-less)
- Timer Period Interrupt (tick-based)
- Application Callback (from interrupt context)

TIME System Service

- Harmony Configuration

The image displays the Harmony IDE interface for configuring the TIME System Service. It includes two side-by-side 'Configuration Options' windows and a central project diagram.

Configuration Options (Left):

- TIME
 - Use SysTick? ☐
 - PLIB Used TC3
 - Number of Clients 5
 - Operating Mode TICK BASED
 - Tick Rate (ms) 1
 - Use Floating Point Calculations? ☐
 - Achievable Tick Rate Resolution (ms): 1.0ms

Configuration Options (Right):

- TIME
 - Use SysTick? ☐
 - PLIB Used TC3
 - Number of Clients 5
 - Operating Mode TICKLESS

Project Diagram:

- TC3 Peripheral Library (TMR) is connected to the TIME System Service (SYS_TIME).
- The TIME System Service is labeled as a **Core Service**.
- A context menu is open for the TIME System Service, showing:
 - sys_time (SYS_TIME) Generic Capability
 - Current Consumers
 - Available Consumers

Available Consumers List:

- AmazonHWInterface (AmazonHWInterface)
- SD Card (SPI) (drv_sdspi)
- External Controller (gfx_driver_external)
- ILI9488 (gfx_driver_ili9488)
- SSD1309 (gfx_driver_ssd1309)
- SSD1963 (gfx_driver_ssd1963)
- LE External Controller (le_gfx_driver_external)
- LE ILI9488 (le_gfx_driver_ili9488)
- LE SSD1963 (le_gfx_driver_ssd1963)
- wolfCrypt Library (lib_wolfcrypt)
- TCPIP CORE (tcpipStack)
- Host Layer (usb_host)

TIME System Service

API functions of the Time System Service library.

- **System Functions**

SYS_TIME_Initialize	Initializes the System Time module.
SYS_TIME_Status	Returns System Time status.
SYS_TIME_Deinitialize	Deinitializes the specific module instance of the SYS TIMER module

- **Timer Functions**

SYS_TIME_TimerCreate	Creates and initializes a new 32-bit software timer instance.
SYS_TIME_TimerDestroy	Destroys/deallocates a software timer instance.
SYS_TIME_TimerStart	Starts a software timer running.
SYS_TIME_TimerStop	Stops a running software timer.
SYS_TIME_TimerCounterGet	Gets the elapsed counter value of a software timer.
SYS_TIME_TimerPeriodHasExpired	Reports whether or not the current period of a software timer has expired.
SYS_TIME_TimerReload	Reloads (or reinitializes) the software timer instance.

- **Callback and Delay Functions**

SYS_TIME_CallbackRegisterMS	Registers a function with the time system service to be called back when the requested number of milliseconds has expired (either once or repeatedly).
SYS_TIME_CallbackRegisterUS	Registers a function with the time system service to be called back when the requested number of microseconds have expired (either once or repeatedly).
SYS_TIME_DelayMS	This function is used to generate a delay of a given number of milliseconds.
SYS_TIME_DelayUS	This function is used to generate a delay of a given number of microseconds.
SYS_TIME_DelaysComplete	Determines if the given delay timer has completed.

TIME System Service

API functions of the Time System Service library. (cont.)

- Counter and Conversion Functions

<code>SYS_TIME_Counter64Get</code>	Get the common 64-bit system counter value.
<code>SYS_TIME_CounterGet</code>	Get the common 32-bit system counter value.
<code>SYS_TIME_CounterSet</code>	Sets the common 32-bit system counter value.
<code>SYS_TIME_CountToMS</code>	Converts a counter value to time interval in milliseconds.
<code>SYS_TIME_CountToUS</code>	Converts a counter value to time interval in microseconds.
<code>SYS_TIME_MSToCount</code>	Convert the given time interval in milliseconds to an equivalent counter value.
<code>SYS_TIME_USToCount</code>	Convert the given time interval in microseconds to an equivalent counter value.
<code>SYS_TIME_FrequencyGet</code>	Gets the frequency at which the hardware timer counts.

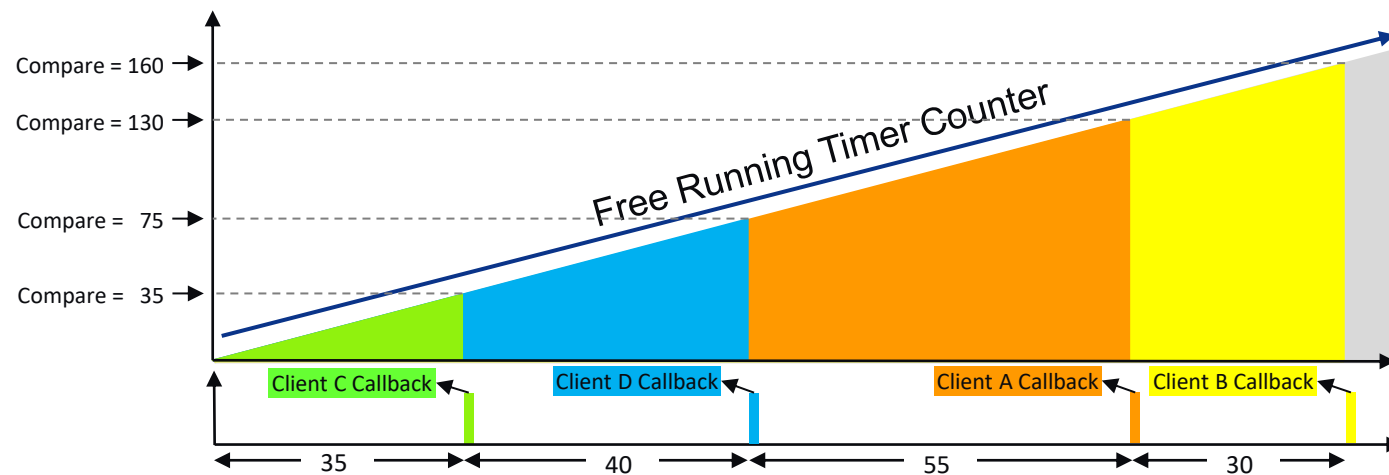
- Data Types and Constants

<code>SYS_TIME_INIT</code>	Defines the data required to initialize the TIME System Service
<code>SYS_TIME_CALLBACK</code>	Pointer to a time system service callback function.
<code>SYS_TIME_CALLBACK_TYPE</code>	Identifies the type of callback requested (single or periodic).
<code>SYS_TIME_RESULT</code>	Result of a time service client interface operation.
<code>SYS_TIME_HANDLE_INVALID</code>	Invalid handle value to a software timer instance.
<code>SYS_TIME_HANDLE</code>	Handle to a software timer instance.

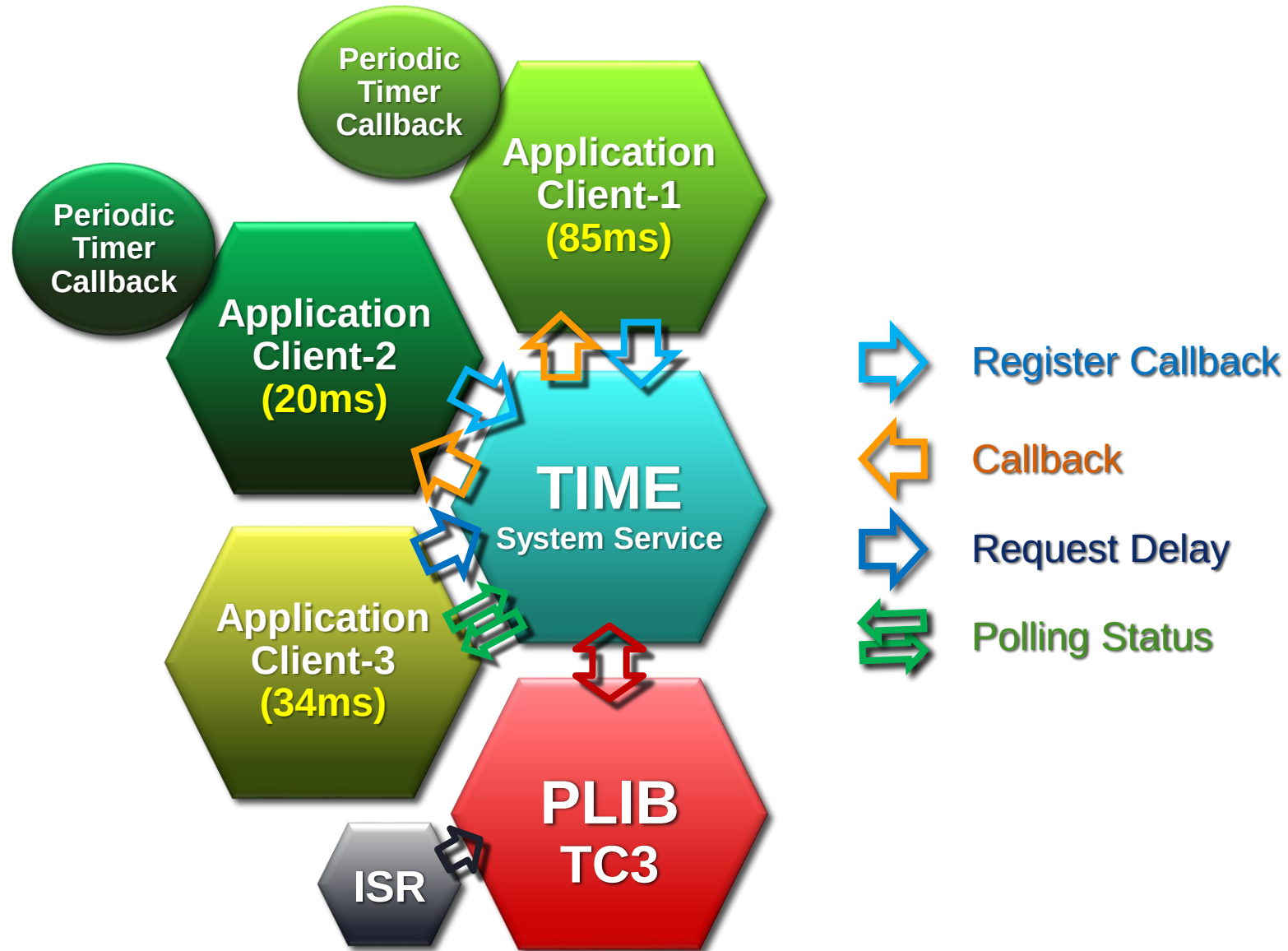
TIME System Service

- No periodic timer tick, reduces overhead of periodically servicing interrupt.
- Provide time resolution in terms of hardware time clock.
- CPU can stay in low power and wakeup only when its time to act.

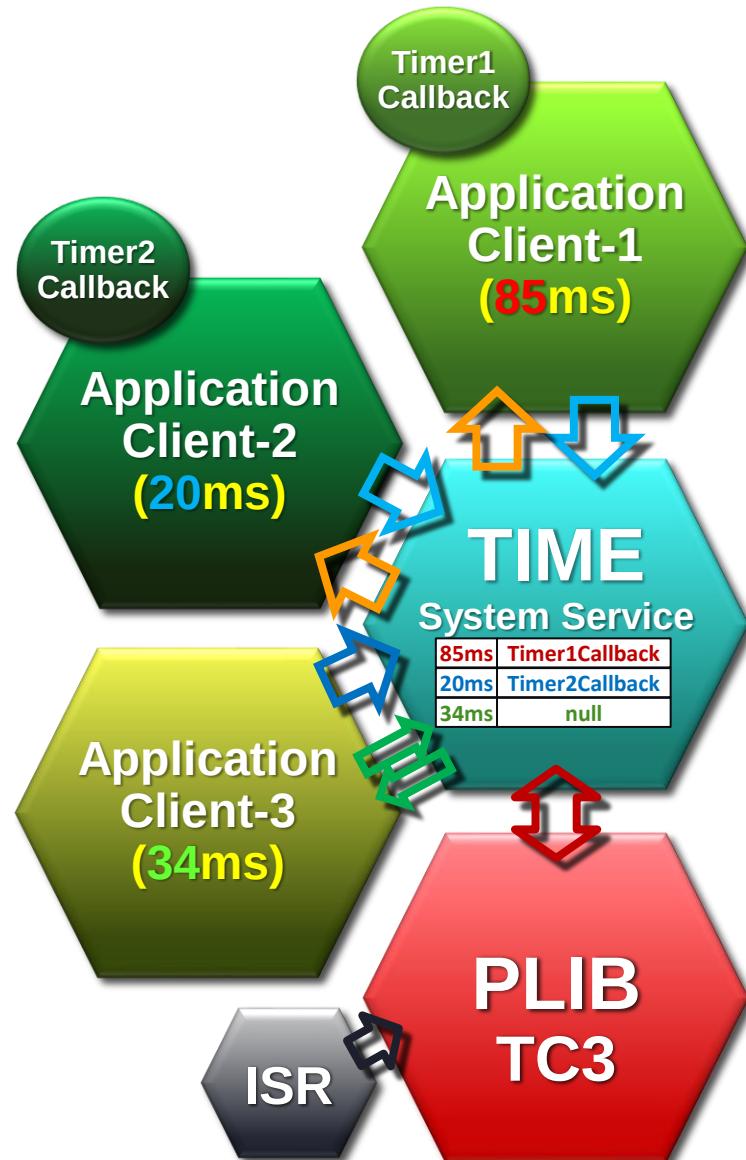
Client Request	Timer Linked List					Timer Compare
Client A : 130ms	130					Compare = 130
Client B : 160ms	130	30				Compare = 130, 160
Client C : 35ms	35	95	30			Compare = 35, 130, 160
Client D : 75ms	35	40	55	30		Compare = 35, 75, 130, 160



TIME System Service



TIME System Service



```
void Timer1Callback(...)
{
    Timer1Expired = true;
}

void Timer2Callback(...)
{
    Timer2Expired = true;
}

void App1_Task()
{
    SYS_TIME_CallbackRegisterMS(Timer1Callback, 0, 85,
                                SYS_TIME_PERIODIC);
}

void App2_Task()
{
    SYS_TIME_CallbackRegisterMS(Timer2Callback, 0, 20,
                                SYS_TIME_PERIODIC);
}

void App3_Task()
{
    SYS_TIME_HANDLE tmrHandle;

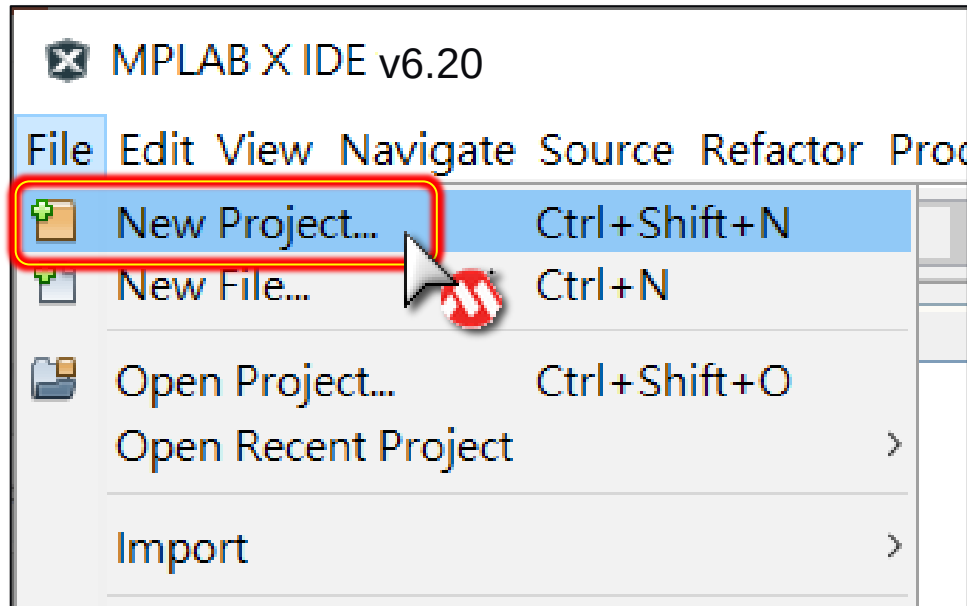
    SYS_TIME_DelayMS(34, &tmrHandle);

    while(1)
    {
        if( SYS_TIME_DelayIsComplete(tmrHandle) == true )
        {
            state = APP_DELAY_COMPLETE;
        }
    }
}
```

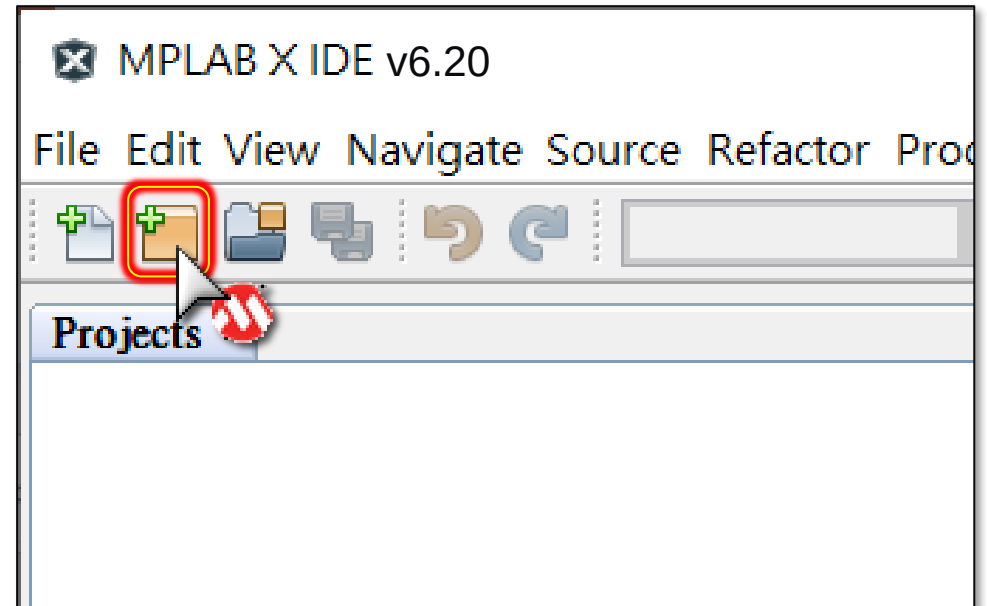
A New MCC Harmony Project

A New MCC Harmony Project

- Create a new 32 bits MCC Harmony Project



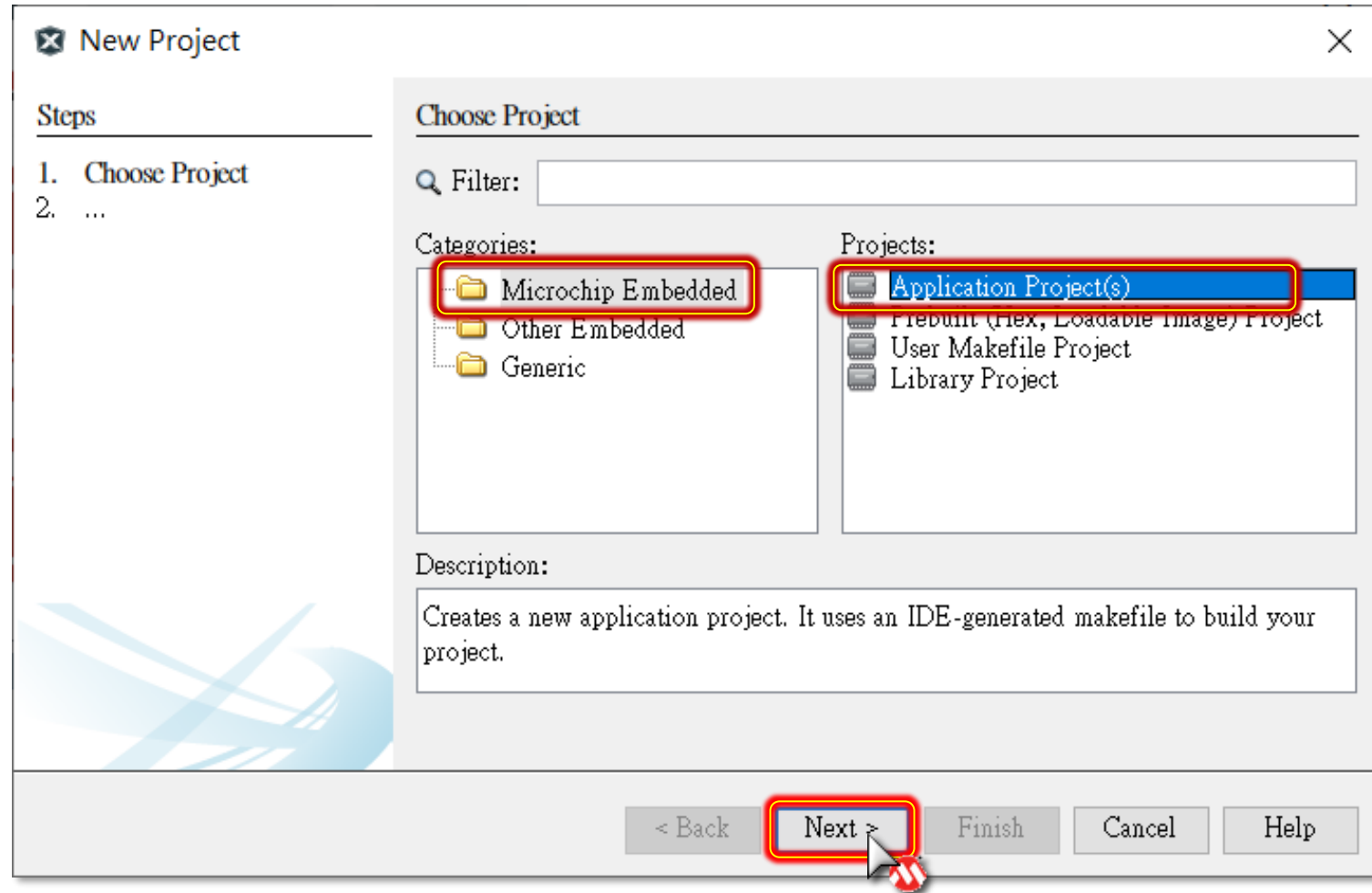
OR



Not hint for this step anymore after Lab1

A New MCC Harmony Project

- Categories : Microchip Embedded
- Projects : Application Project(s)



A New MCC Harmony Project

- Family : 32-bit MCUs (PIC32C/SAM)
- Device : **ATSAMD21G18A** Please do not miss any character of your part number
- Tool : **Snap-SN:BURxxxxxxxxxx**

New Project

Steps

1. Choose Project
2. **Select Device**
3. Select Header
4. Select Plugin Board
5. Select Compiler
6. Select Project Name and Folder
7. (Optional) Add Project

MPLAB X IDE

Select Device

Family: 32-bit MCUs (PIC32C/SAM) Optional

Device: **ATSAMD21G18A** Type in correct part number

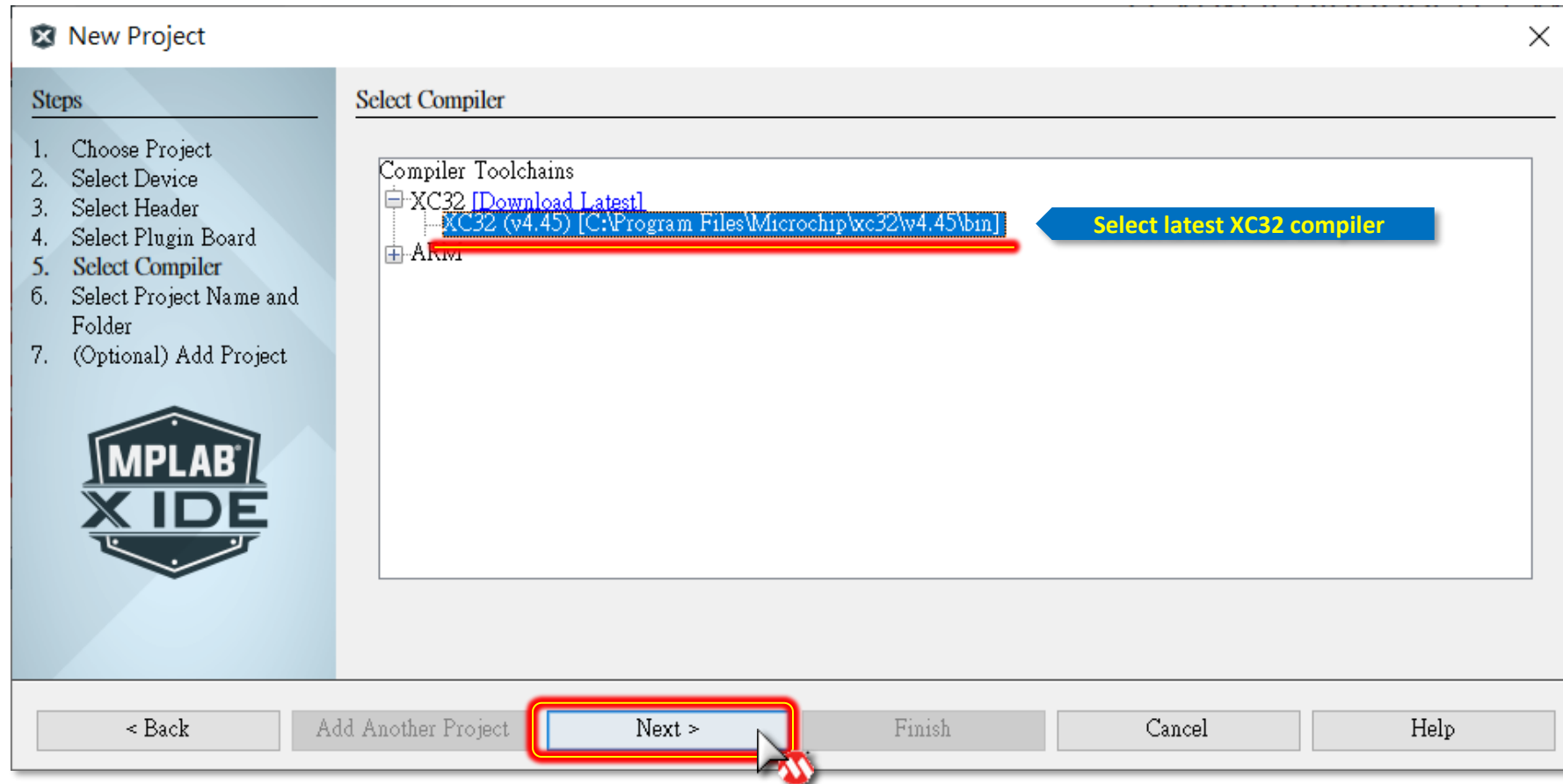
Tool: **Snap-SN:BUR183076917** Select Debugger if connected ☐ Show All

< Back Add Another Project **Next >** Finish Cancel Help

A New MCC Harmony Project

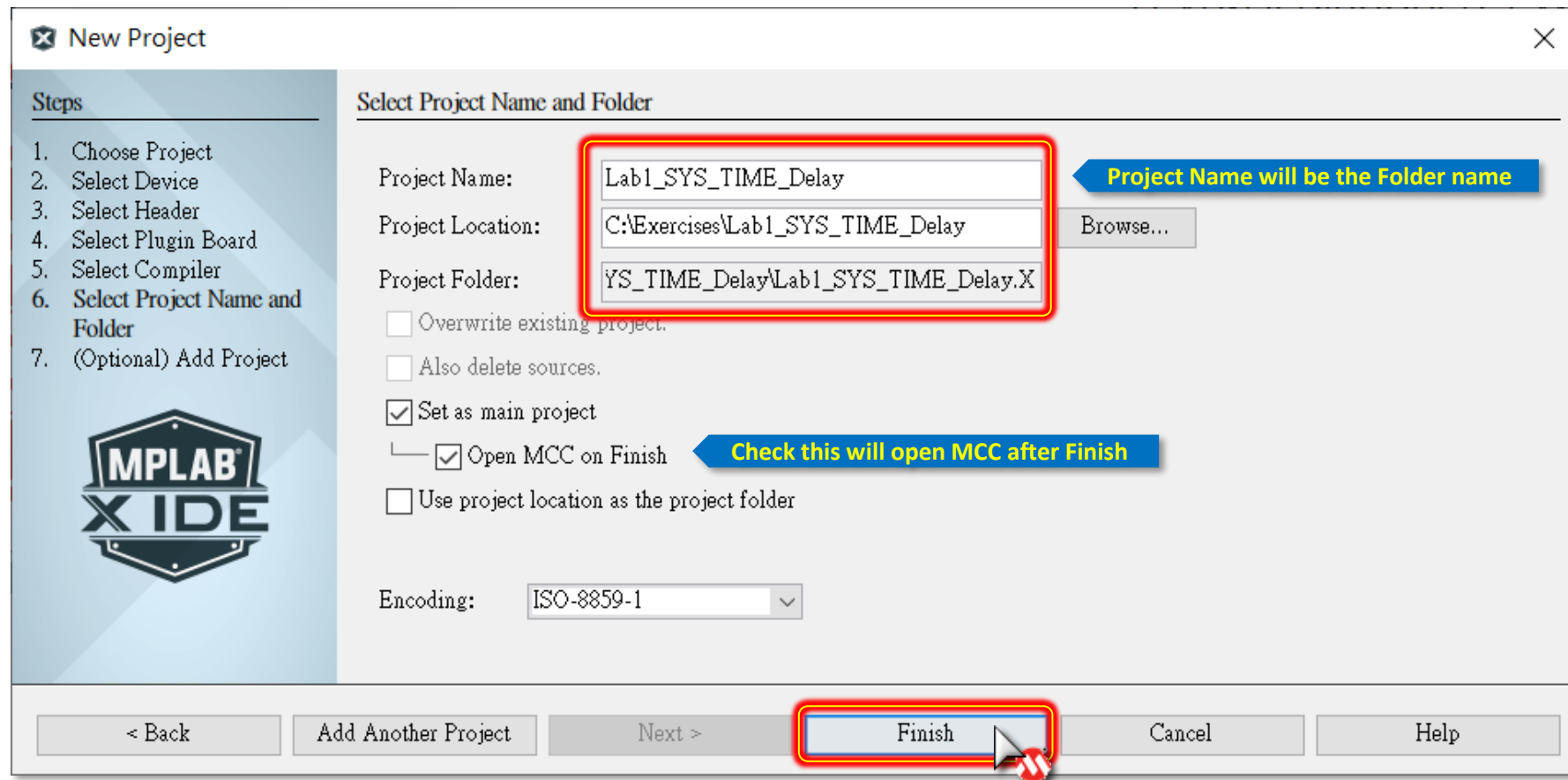
Select latest XC32 Compiler.

- **XC32**
 - XC32 (v4.45) [C:\Program File\Microchip\xc32\..)



A New MCC Harmony Project

- Project Name : Lab1_SYS_TIME_Delay
- Project Location : C:\Exercises\Lab1_SYS_TIME_Delay
- Project Folder : C:\Exercises\Lab1_SYS_TIME_Delay\ Lab1_SYS_TIME_Delay.X



New Project

Steps

1. Choose Project
2. Select Device
3. Select Header
4. Select Plugin Board
5. Select Compiler
6. Select Project Name and Folder
7. (Optional) Add Project

Select Project Name and Folder

Project Name: Lab1_SYS_TIME_Delay **Project Name will be the Folder name**

Project Location: C:\Exercises\Lab1_SYS_TIME_Delay Browse...

Project Folder: YS_TIME_Delay\Lab1_SYS_TIME_Delay.X

☐ Overwrite existing project.

☐ Also delete sources.

☒ Set as main project

☒ Open MCC on Finish **Check this will open MCC after Finish**

☐ Use project location as the project folder

Encoding: ISO-8859-1

< Back Add Another Project Next > **Finish** Cancel Help

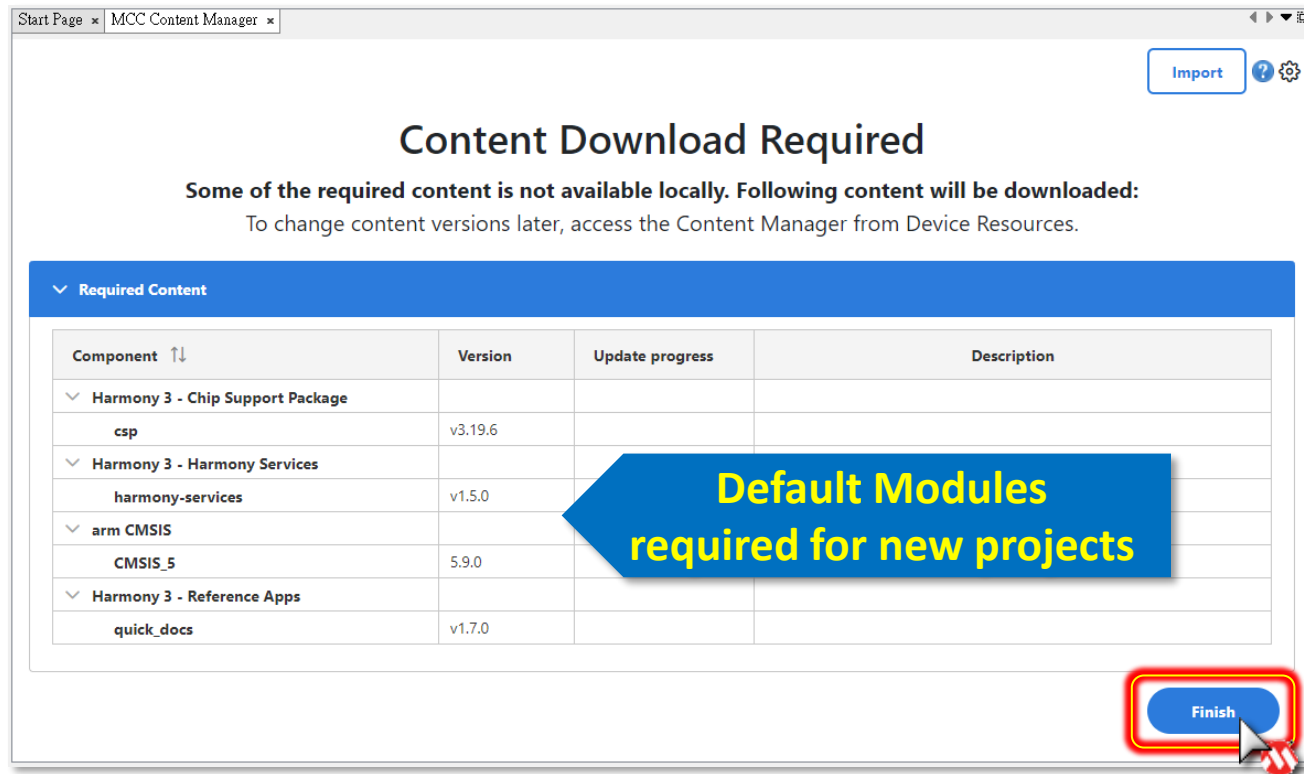
New Download of MCC Harmony Framework

MPLAB Code Configurator(MCC) Plugin v5.4.x above

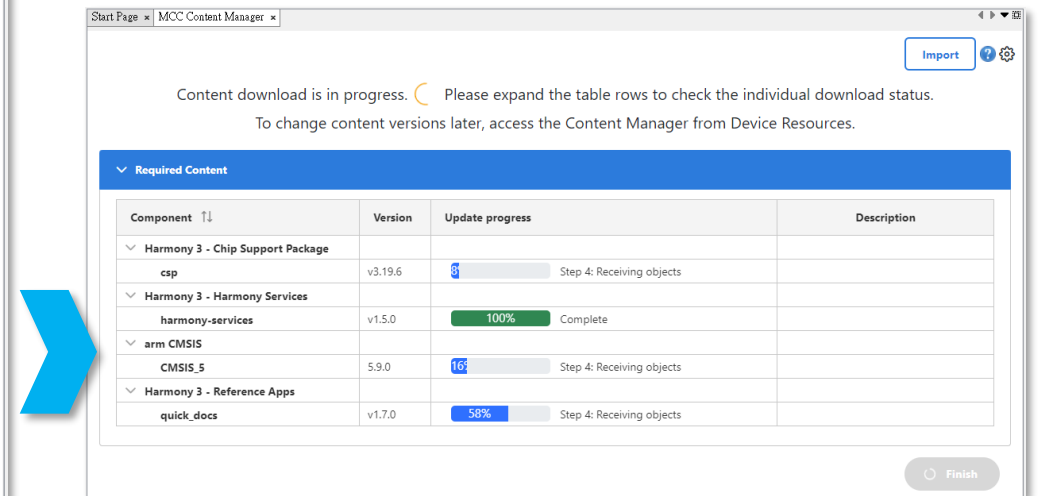
A New MCC Harmony Project

New Download of MCC Harmony Framework

- Launching the **Finish** in MCC Content Manager to download **Required Content**.
 - Default required Modules : **csp**, **harmony-services**, **CMSIS_5** and **quick_docs**.



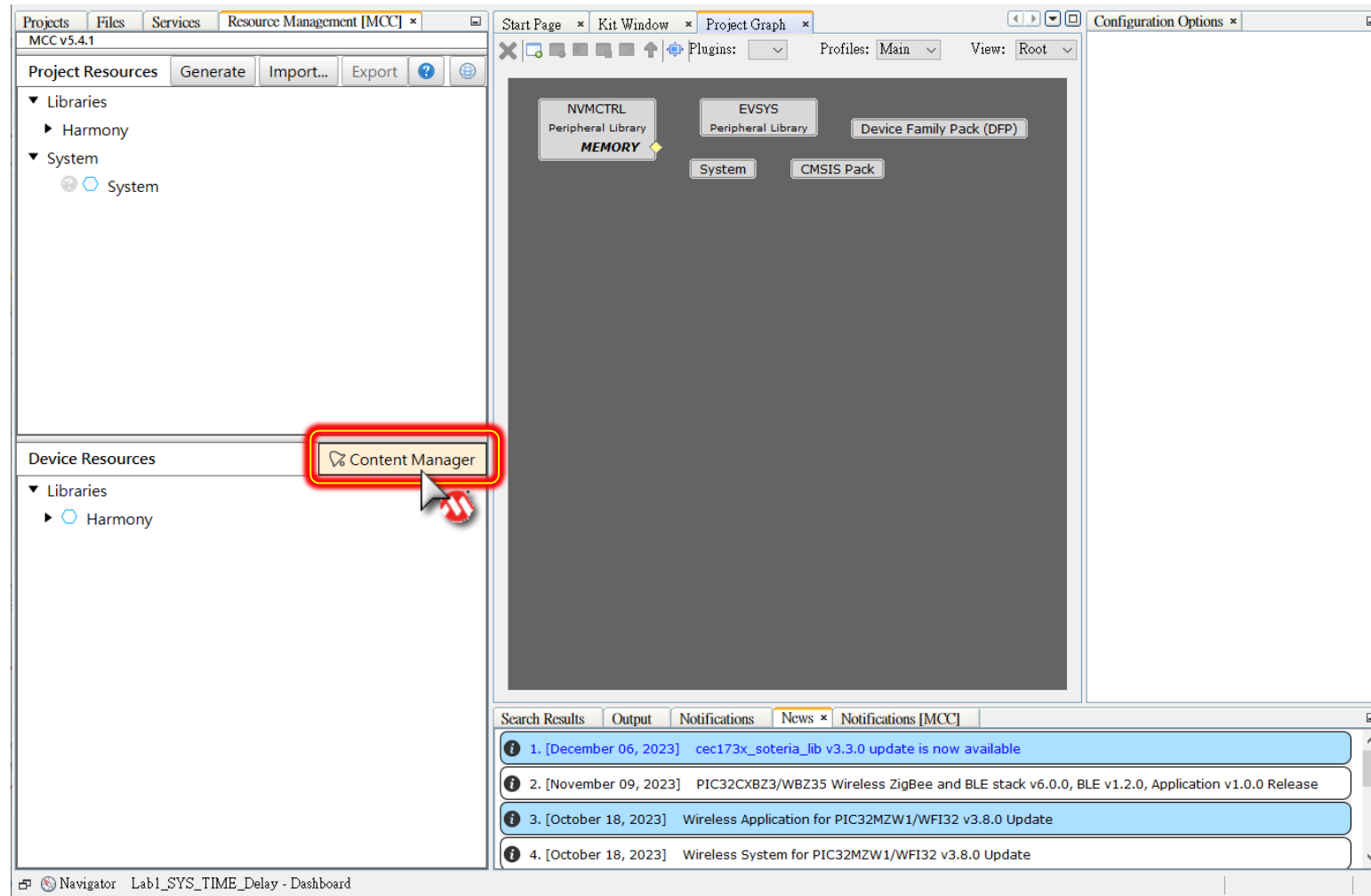
It will take about 10 mins to download



A New MCC Harmony Project

New Download of MCC Harmony Framework

- Launching the **Content Manager** in “Device Resources” window of MCC Harmony.



A New MCC Harmony Project

New Download of MCC Harmony Framework

- Input “**wireless_wifi**” in search window.
- Under “**Harmony 3 – Wireless solution**” > “**wireless_wifi**” select to “**v3.11.1**”.

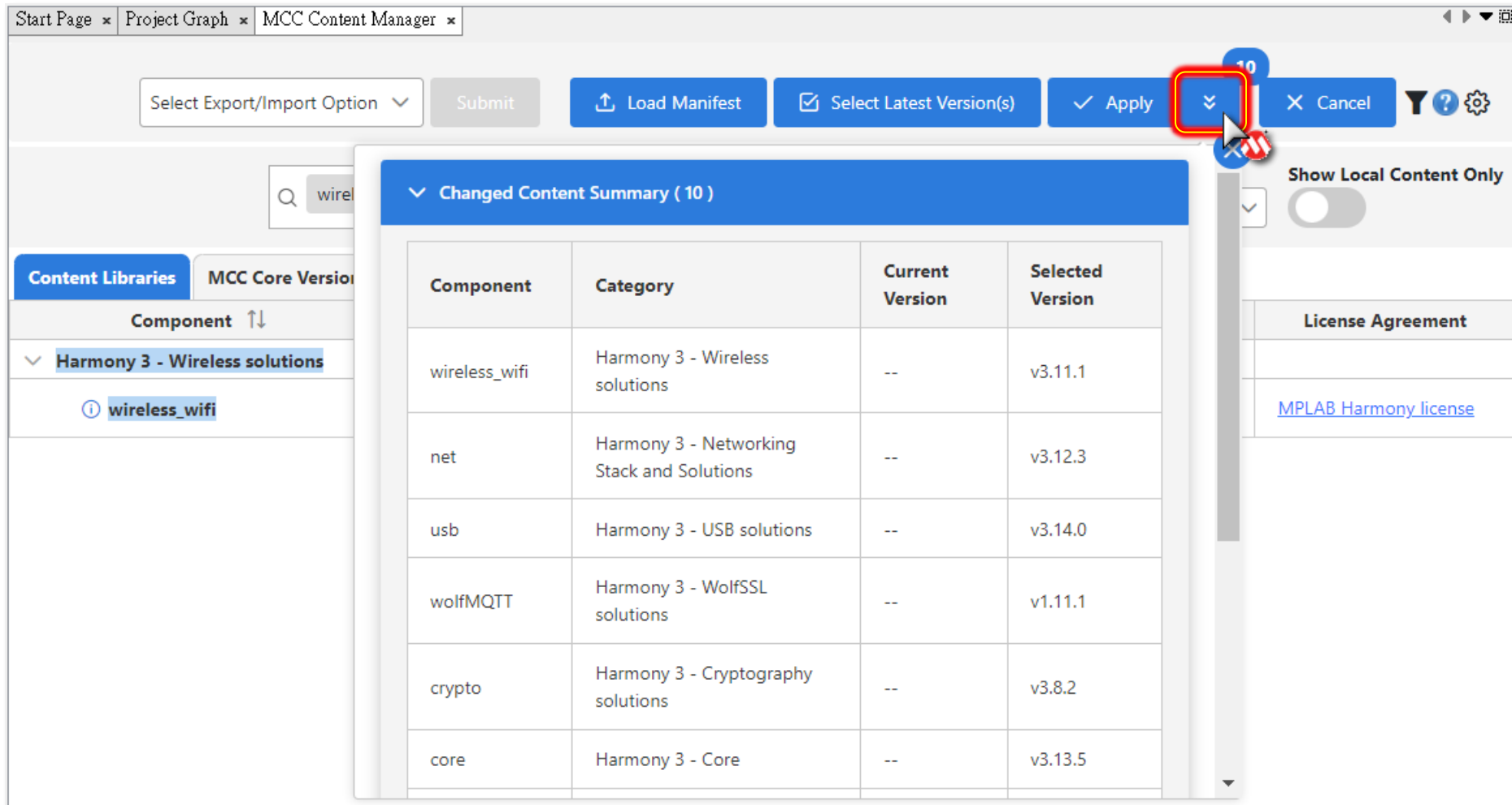
The screenshot shows the MCC Content Manager interface. At the top, there are tabs for 'Start Page', 'Project Graph', and 'MCC Content Manager'. Below the tabs, there is a search bar with the text 'wireless_wifi' entered. To the right of the search bar, there are fields for 'Device' (ATSAMD21G18A) and 'Content Type' (HARMONY). Below these fields, there is a table with columns: Component, Version, Status, Update progress, Description, and License Agreement. The table has a row for 'Harmony 3 - Wireless solutions' which is expanded to show a sub-row for 'wireless_wifi'. In this sub-row, the 'Version' column shows 'v3.11.1' with a dropdown arrow, indicating it is the selected version. The 'License Agreement' column for this row points to 'MPLAB Harmony license'.

Component	Version	Status	Update progress	Description	License Agreement
Harmony 3 - Wireless solutions					
wireless_wifi	v3.11.1				MPLAB Harmony license

A New MCC Harmony Project

New Download of MCC Harmony Framework

- In Content Manager, it will automatically select necessary dependencies modules.



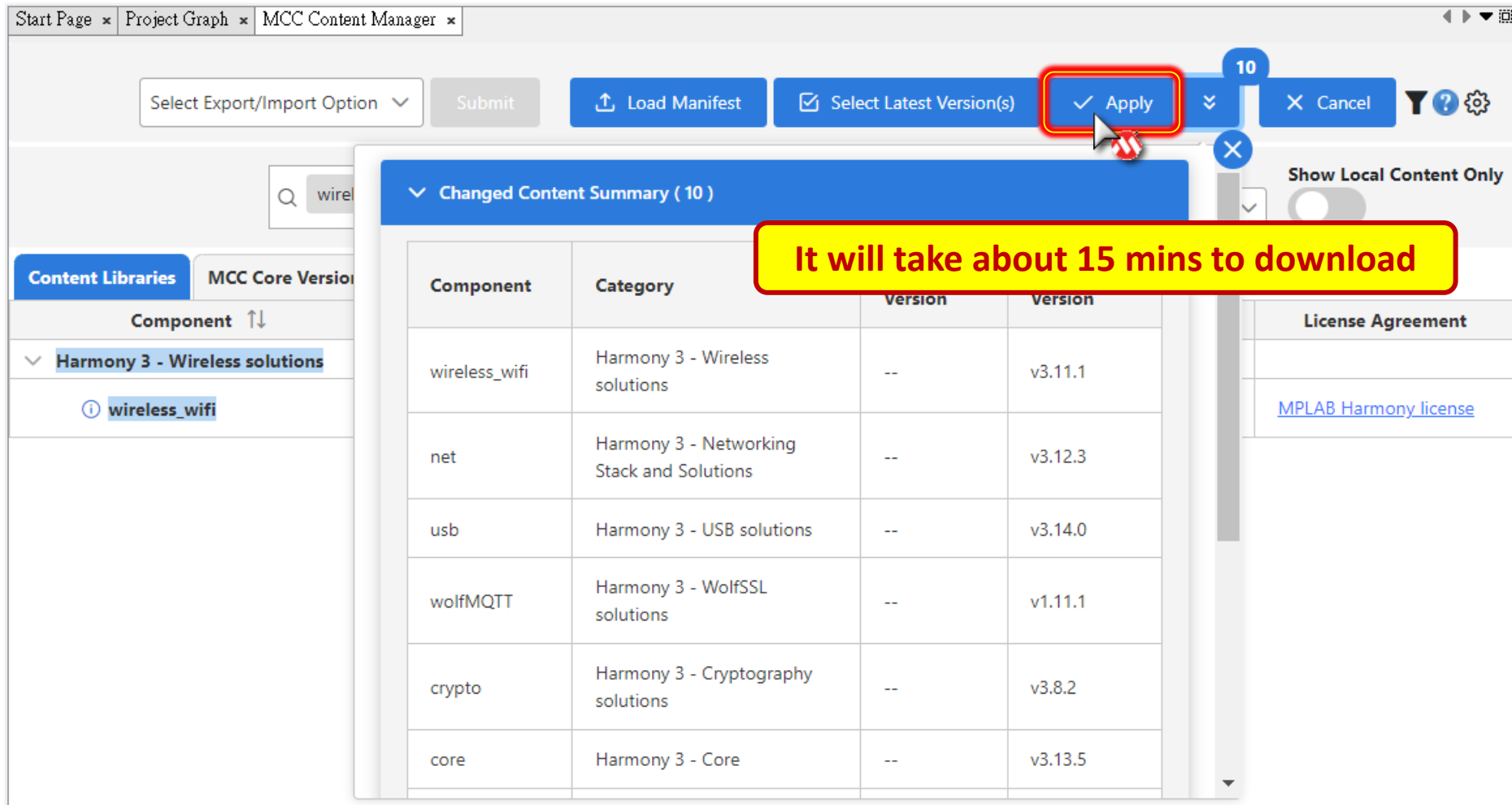
The screenshot shows the MCC Content Manager interface with a 'Changed Content Summary' dialog box open. The dialog box displays a table of components and their versions. A red box highlights the 'Apply' button in the top right corner of the dialog.

Component	Category	Current Version	Selected Version
wireless_wifi	Harmony 3 - Wireless solutions	--	v3.11.1
net	Harmony 3 - Networking Stack and Solutions	--	v3.12.3
usb	Harmony 3 - USB solutions	--	v3.14.0
wolfMQTT	Harmony 3 - WolfSSL solutions	--	v1.11.1
crypto	Harmony 3 - Cryptography solutions	--	v3.8.2
core	Harmony 3 - Core	--	v3.13.5

A New MCC Harmony Project

New Download of MCC Harmony Framework

- Launching the  in MCC Content Manager to download **Optional Content**.



The screenshot shows the MCC Content Manager interface. The 'Apply' button is highlighted with a red box. A yellow callout box with red text states: "It will take about 15 mins to download". The interface displays a table of components to be downloaded.

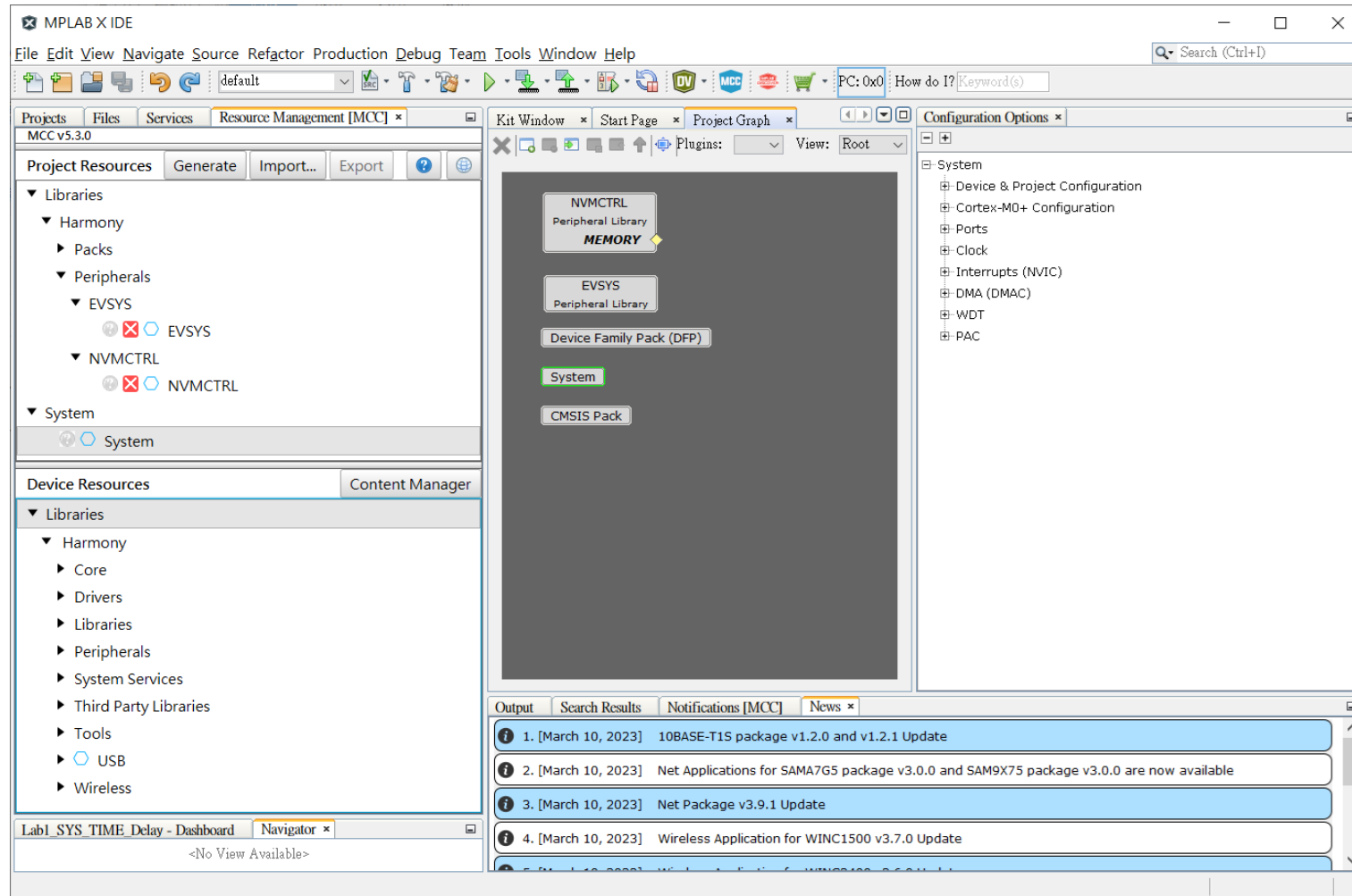
Component	Category	Version	Version
wireless_wifi	Harmony 3 - Wireless solutions	--	v3.11.1
net	Harmony 3 - Networking Stack and Solutions	--	v3.12.3
usb	Harmony 3 - USB solutions	--	v3.14.0
wolfMQTT	Harmony 3 - WolfSSL solutions	--	v1.11.1
crypto	Harmony 3 - Cryptography solutions	--	v3.8.2
core	Harmony 3 - Core	--	v3.13.5

License Agreement: [MPLAB Harmony license](#)

A New MCC Harmony Project

New Download of MCC Harmony Framework

- MCC Harmony Configurator Interface.



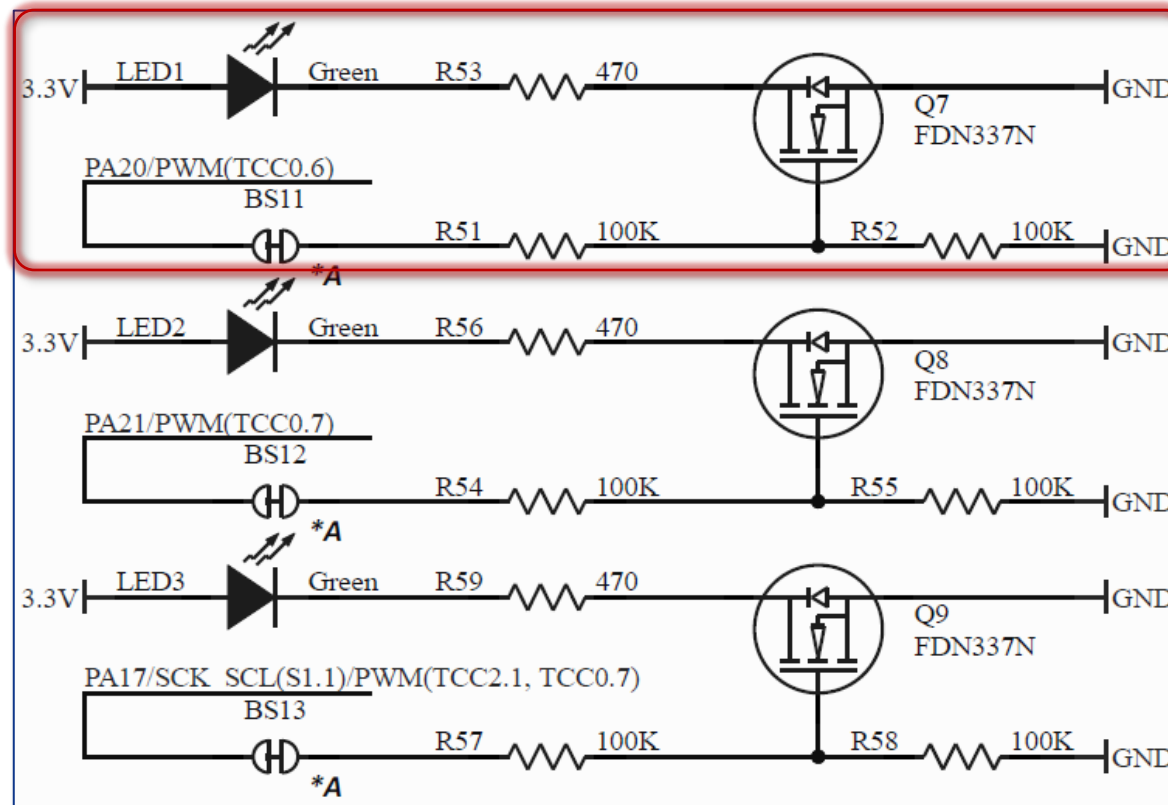
Lab 1

Delay based LED toggle

TIME System Service

⚙️ Lab1 Delay based LED toggle

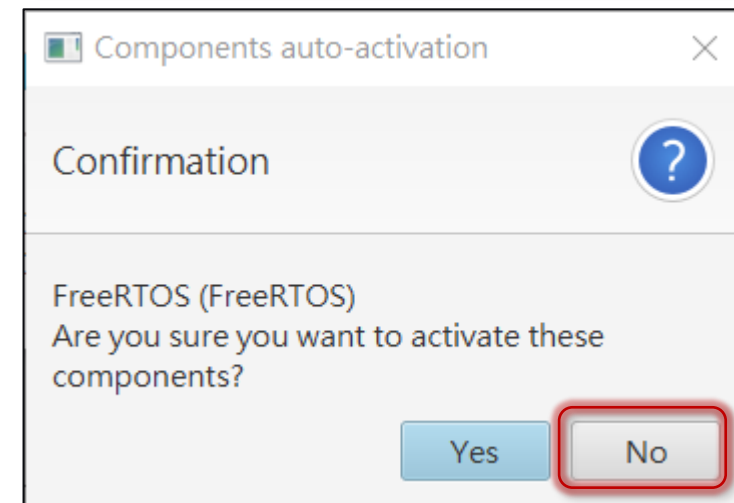
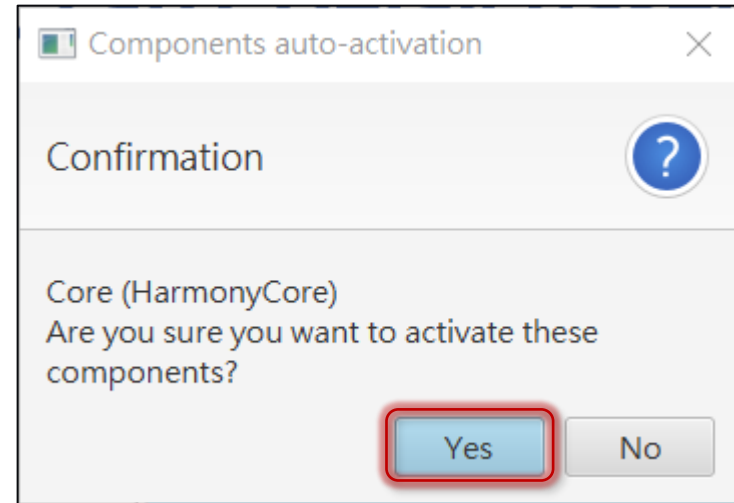
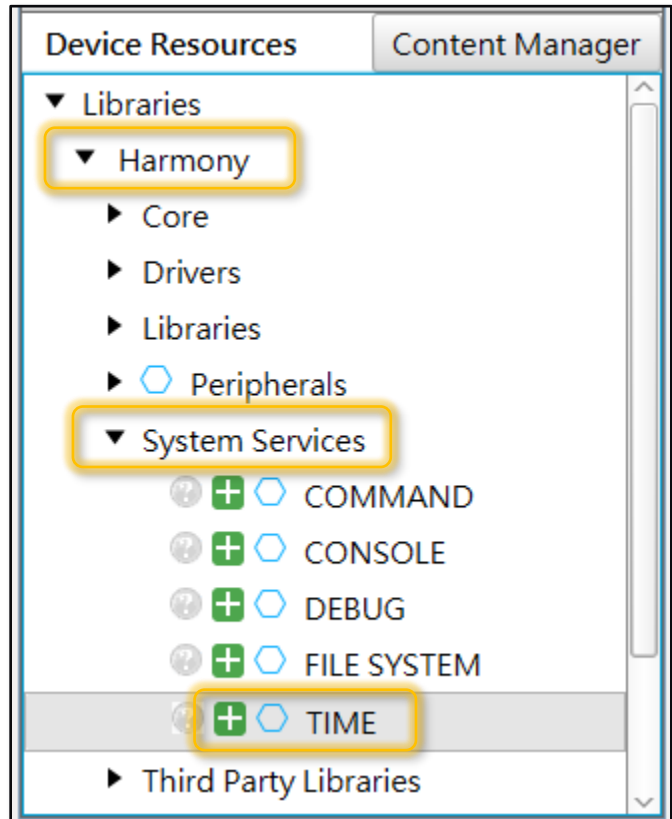
- Try add TIME System Service to create a **delay** for toggle **LED1** in **one second** period.
- The **LED1** is **high active** on pin **PA20**.
- Select **PLIB TC3** as timer source of your **TIME System Service**.



⚙️ Lab1 Delay based LED toggle

- MCC Harmony Configuration (Add TIME System Service)

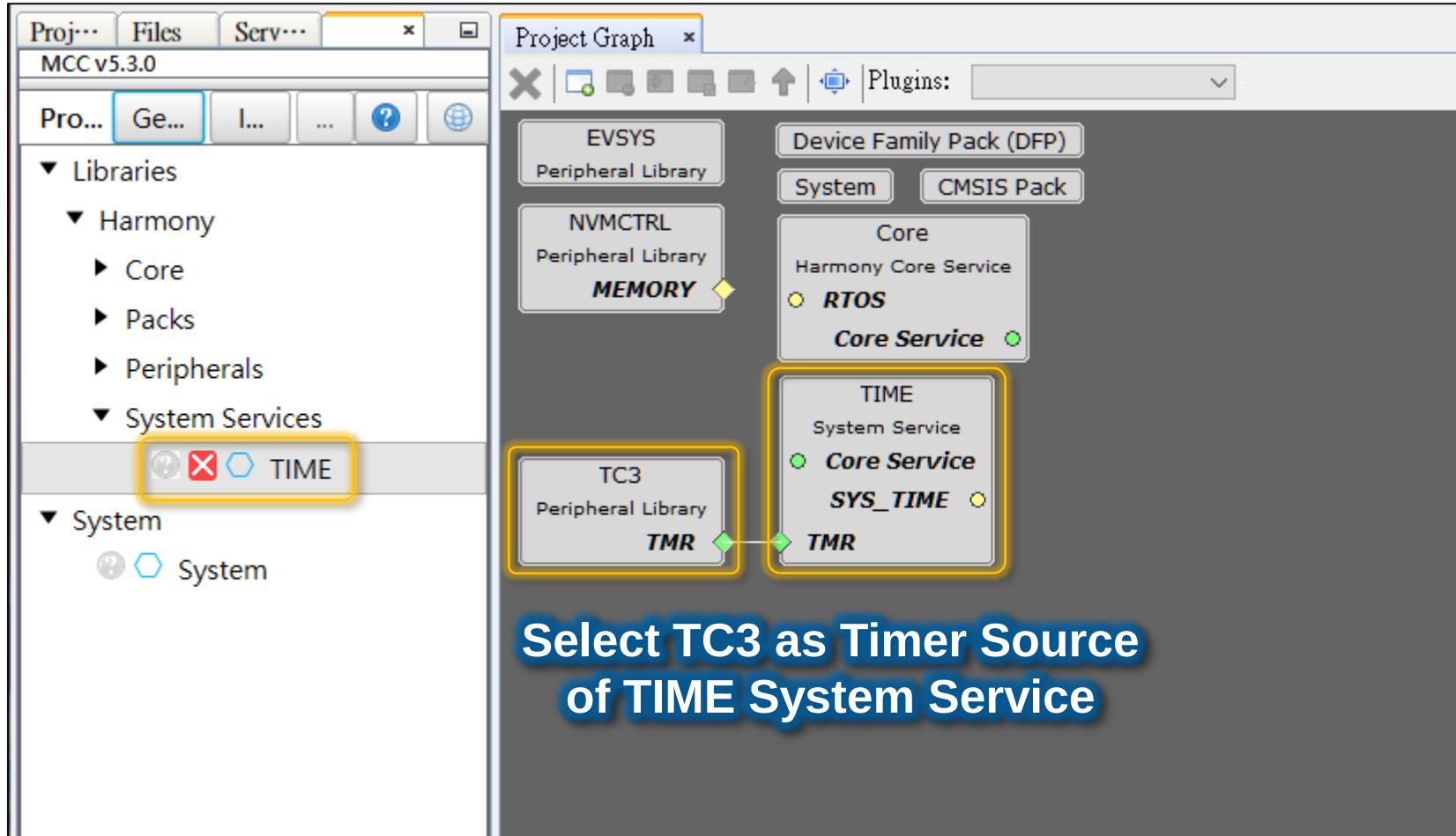
**Not hint for this step
anymore after Lab1**



Select to "No"

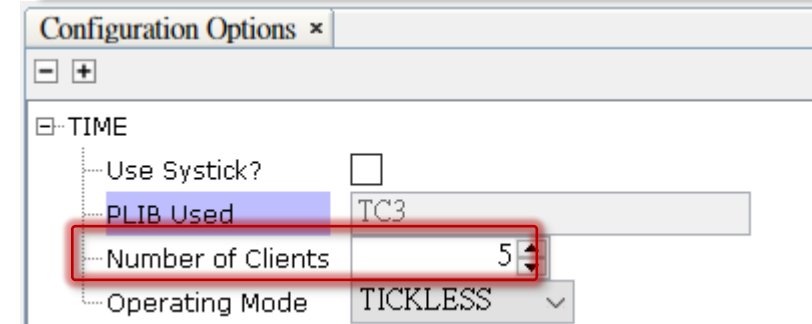
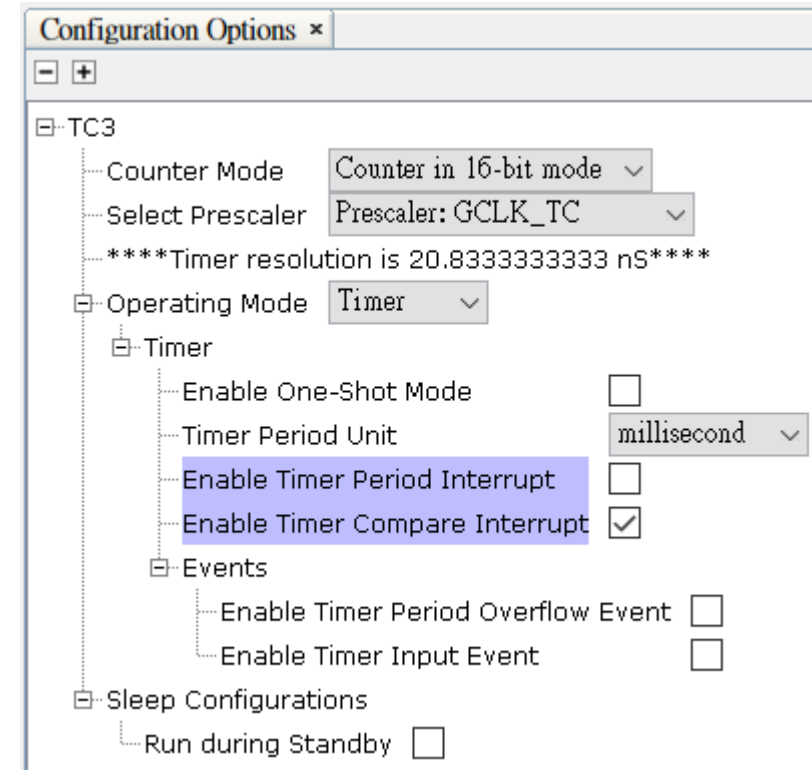
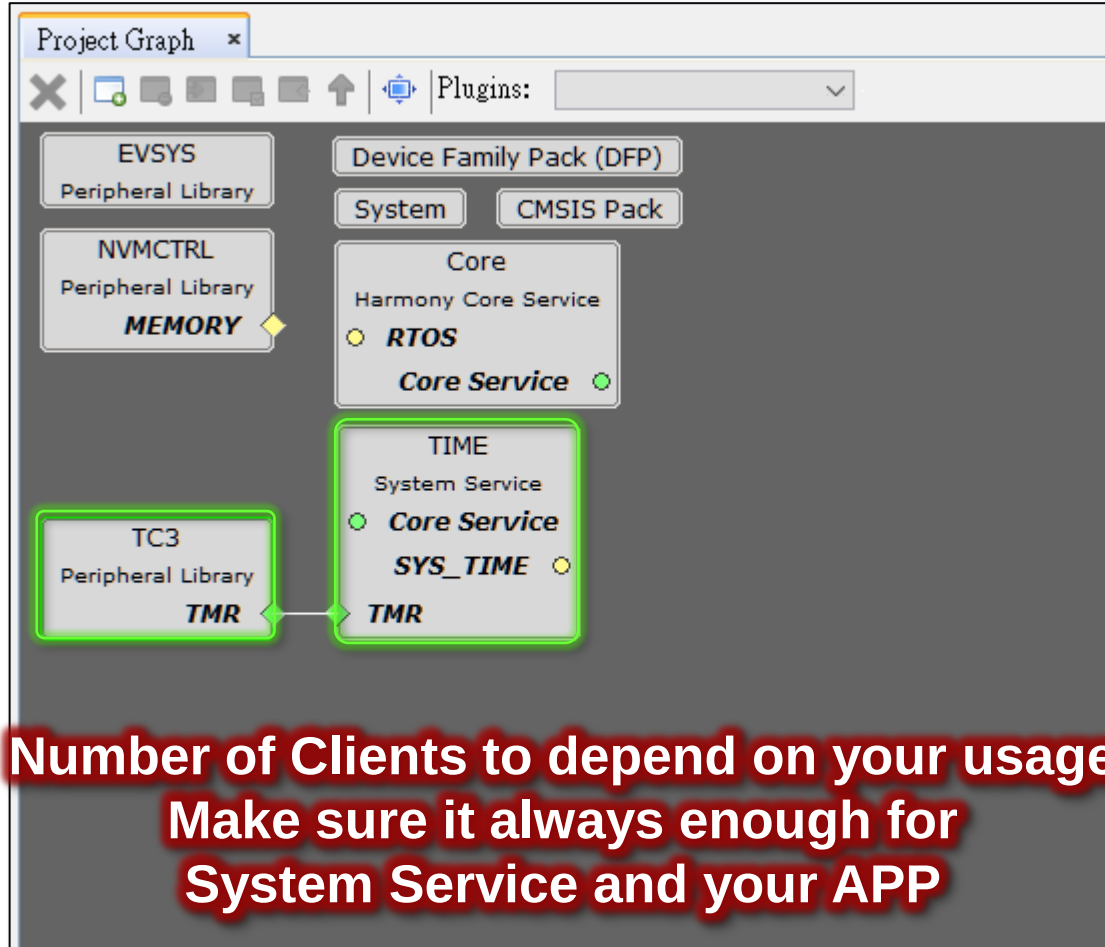
⚙️ Lab1 Delay based LED toggle

- Harmony Configuration (Add TIME System Service)



✖ Lab1 Delay based LED toggle

- Harmony Configuration



⚙️ Lab1 Delay based LED toggle

- Harmony Configuration (Setup GPIO)

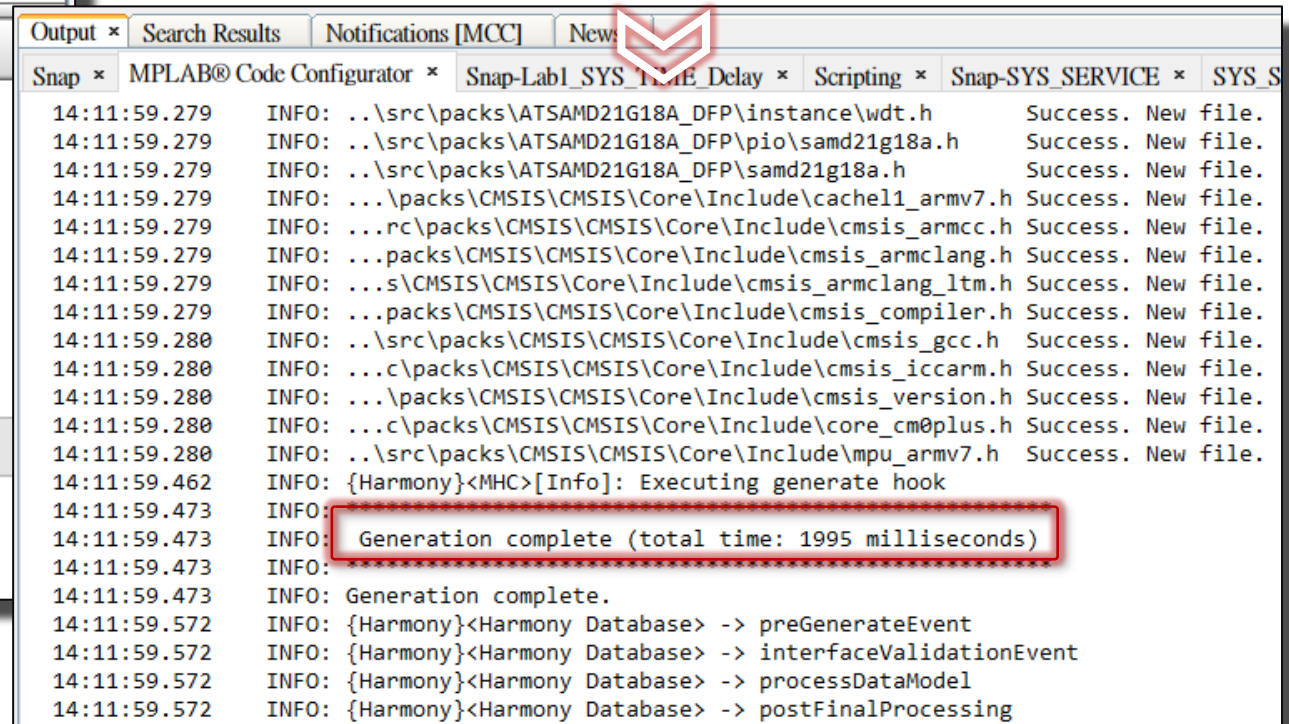
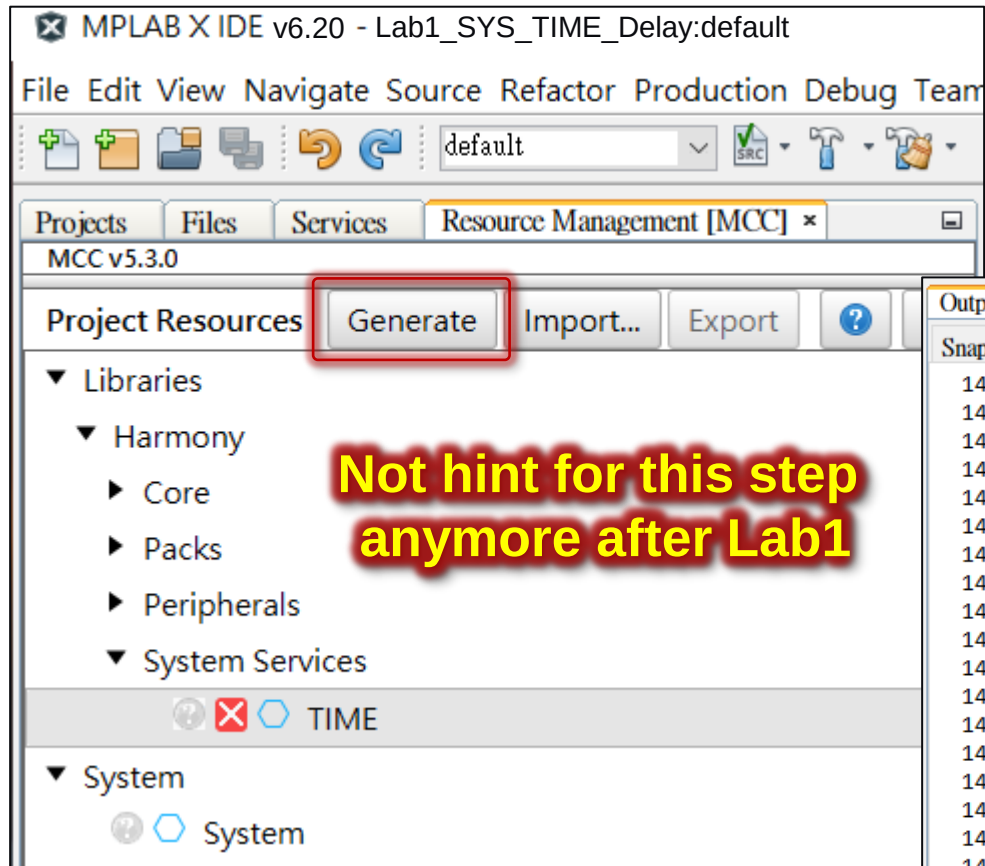
The screenshot shows the Harmony IDE configuration interface. The top part displays the Project Graph with various components like EVSYS, NVMCTRL, TC3, and TIME. The bottom part shows the Pin Settings table, which is currently set to 'Ports' order and 'Easy View' mode. The table lists pins 28, 29, and 30. Pin 29 (PA20) is highlighted with a red box and configured as a GPIO output with a custom name 'LED1' and 'Low' latch.

**Configure PA20 as GPIO output
Custom Name to LED1,
Latch is Low.**

Pin Number	Pin ID	Custom Name	Function	Mode	Direction	Latch	Pull Up	Pull Down	Drive Strength
28	PA19		Available	Digital	High Impedance	Low	☐	☐	NORMAL
29	PA20	LED1	GPIO	Digital	Out	Low	☐	☐	NORMAL
30	PA21		Available	Digital	High Impedance	Low	☐	☐	NORMAL

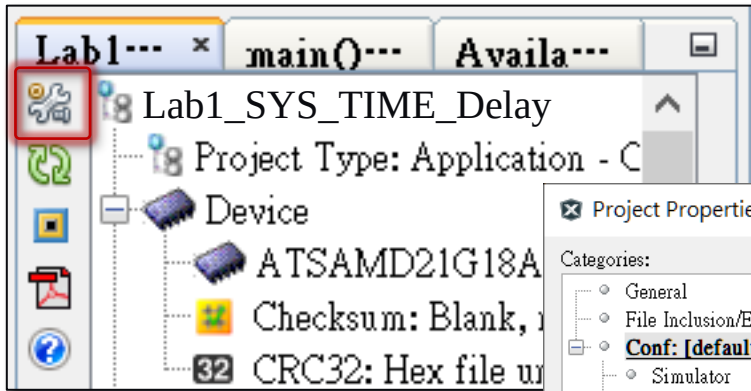
⚙️ Lab1 Delay based LED toggle

- Harmony Configuration (Generate Code)



⚙️ Lab1 Delay based LED toggle

- Configure your new project



**Double confirm the
Device : **ATSAMD21G18A****

Select correct

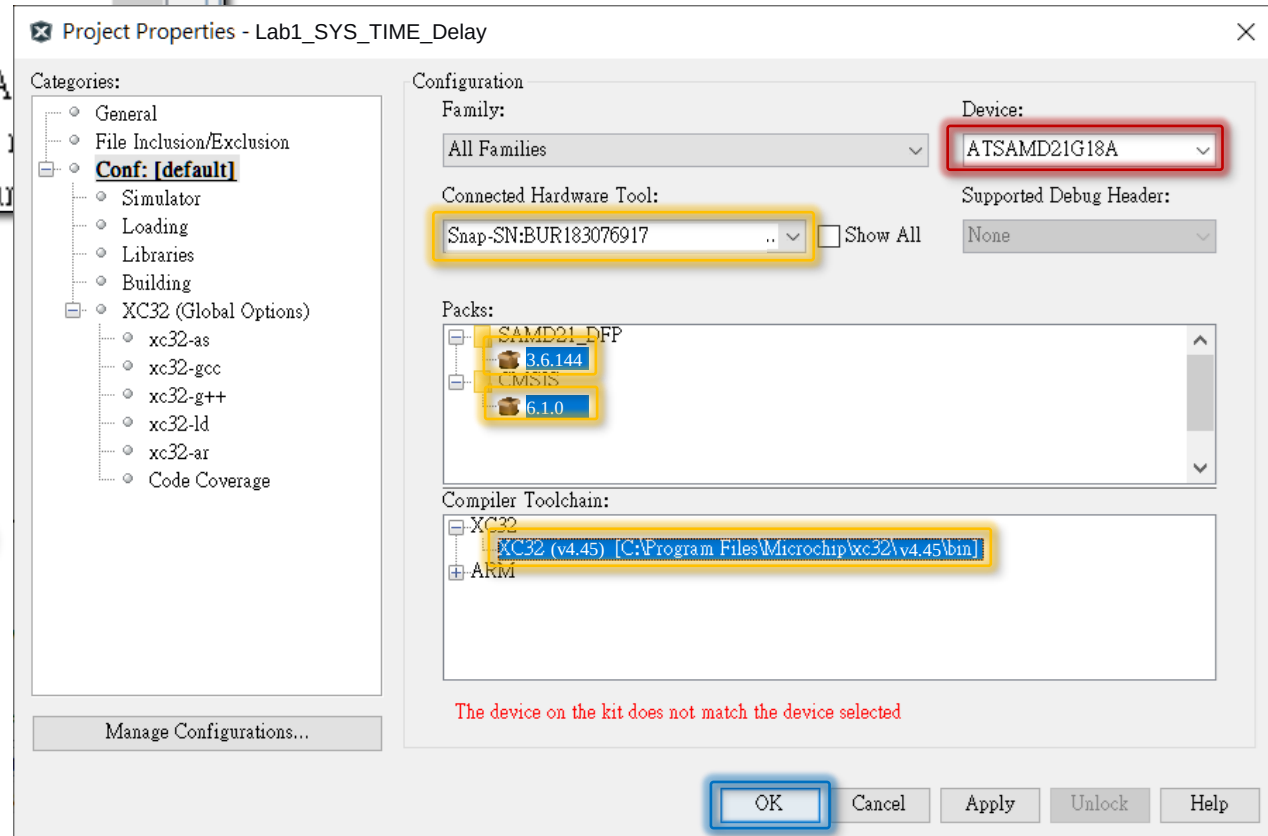
Debugger

DFP 3.6.144

CMSIS 6.1.0

XC32 (v4.45)

**Not hint for this step
anymore after Lab1**



⚙️ Lab1 Delay based LED toggle

- Modify your main.c as below

```
#include <stddef.h>           // Defines NULL
#include <stdbool.h>          // Defines true
#include <stdlib.h>           // Defines EXIT_FAILURE
#include "definitions.h"      // SYS function prototypes

SYS_TIME_HANDLE tmrHandle;

int main( void )
{
    SYS_Initialize(NULL);

    SYS_TIME_DelayMS(1000, &tmrHandle);

    while( true )
    {
        if( SYS_TIME_DelayIsComplete(tmrHandle) == true )
        {
            LED1_Toggle();
            SYS_TIME_DelayMS(1000, &tmrHandle);
        }

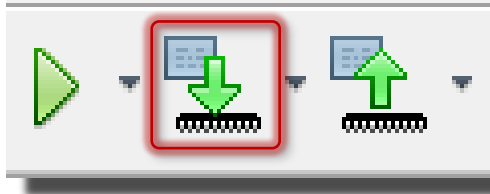
        /* Maintain state machines of all polled MPLAB Harmony modules. */
        SYS_Tasks();
    }

    /* Execution should not come here during normal operation */

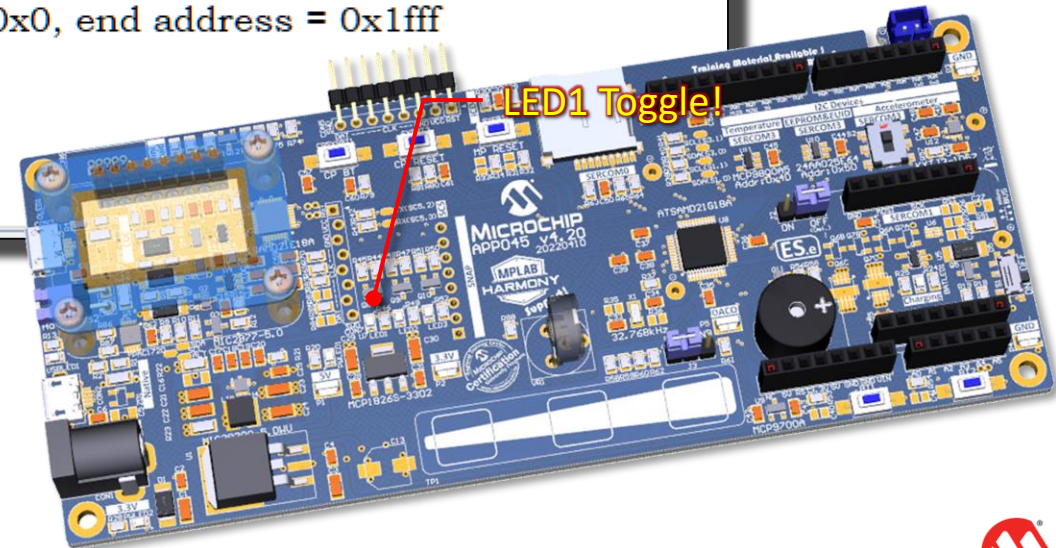
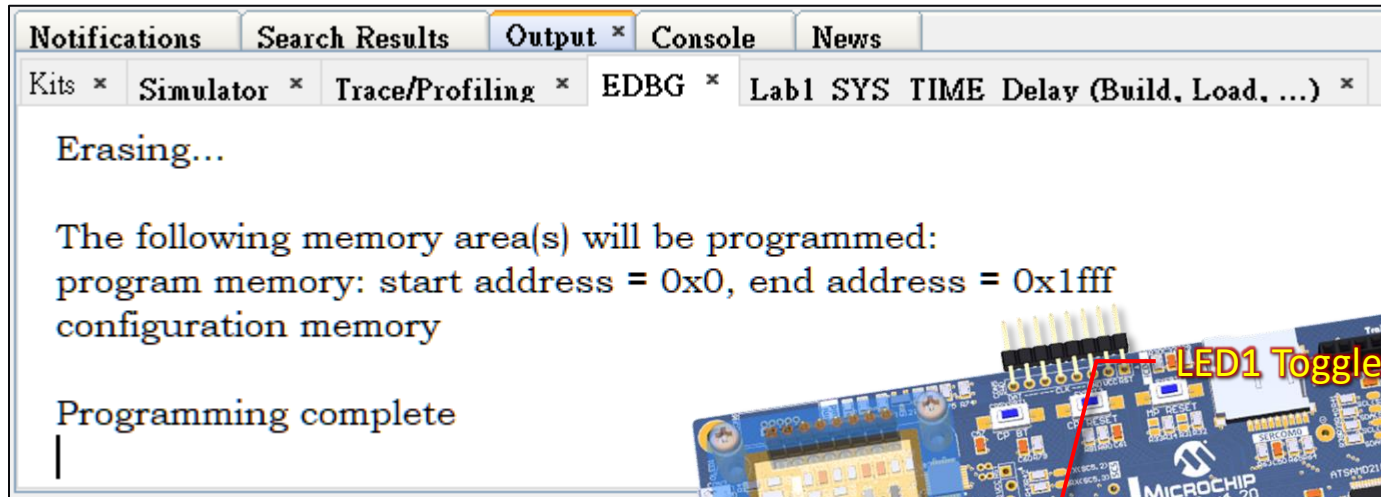
    return ( EXIT_FAILURE);
}
```

⚙️ Lab1 Delay based LED toggle

- Compiler and programming your project to EVB



Not hint for this step anymore after Lab1



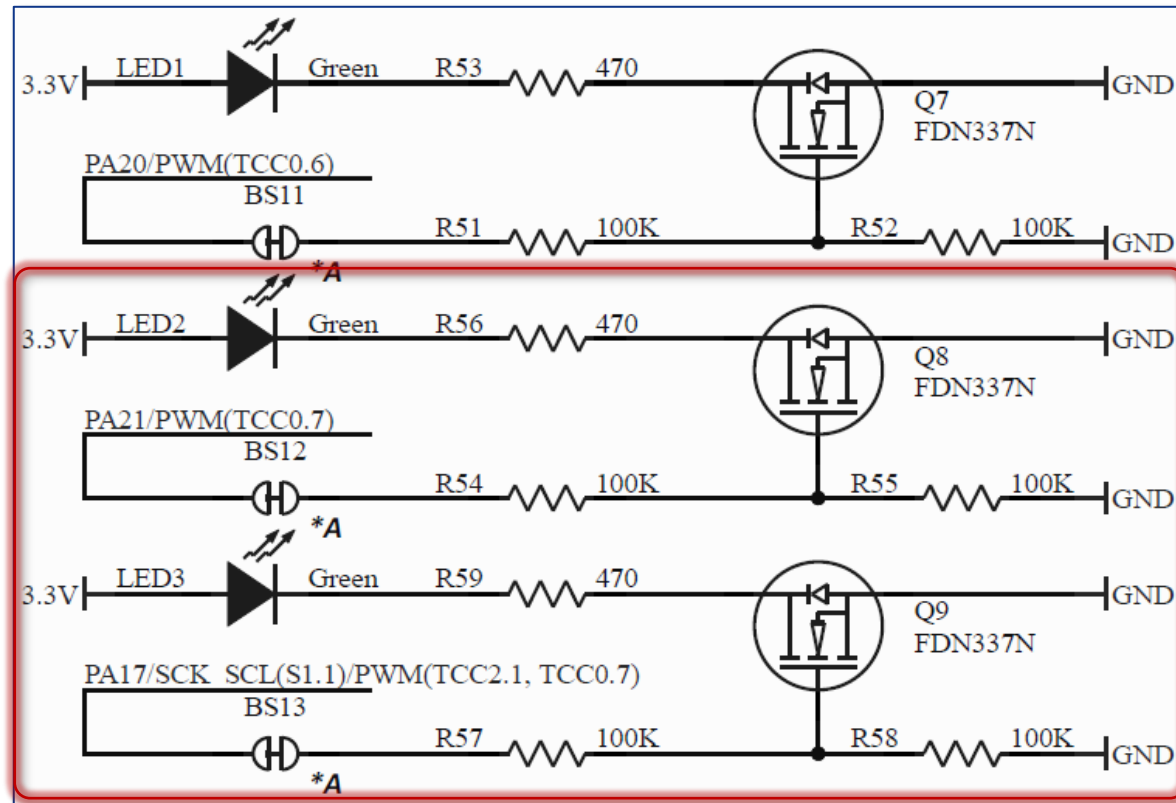
Lab 2

Callback based LED toggle

TIME System Service

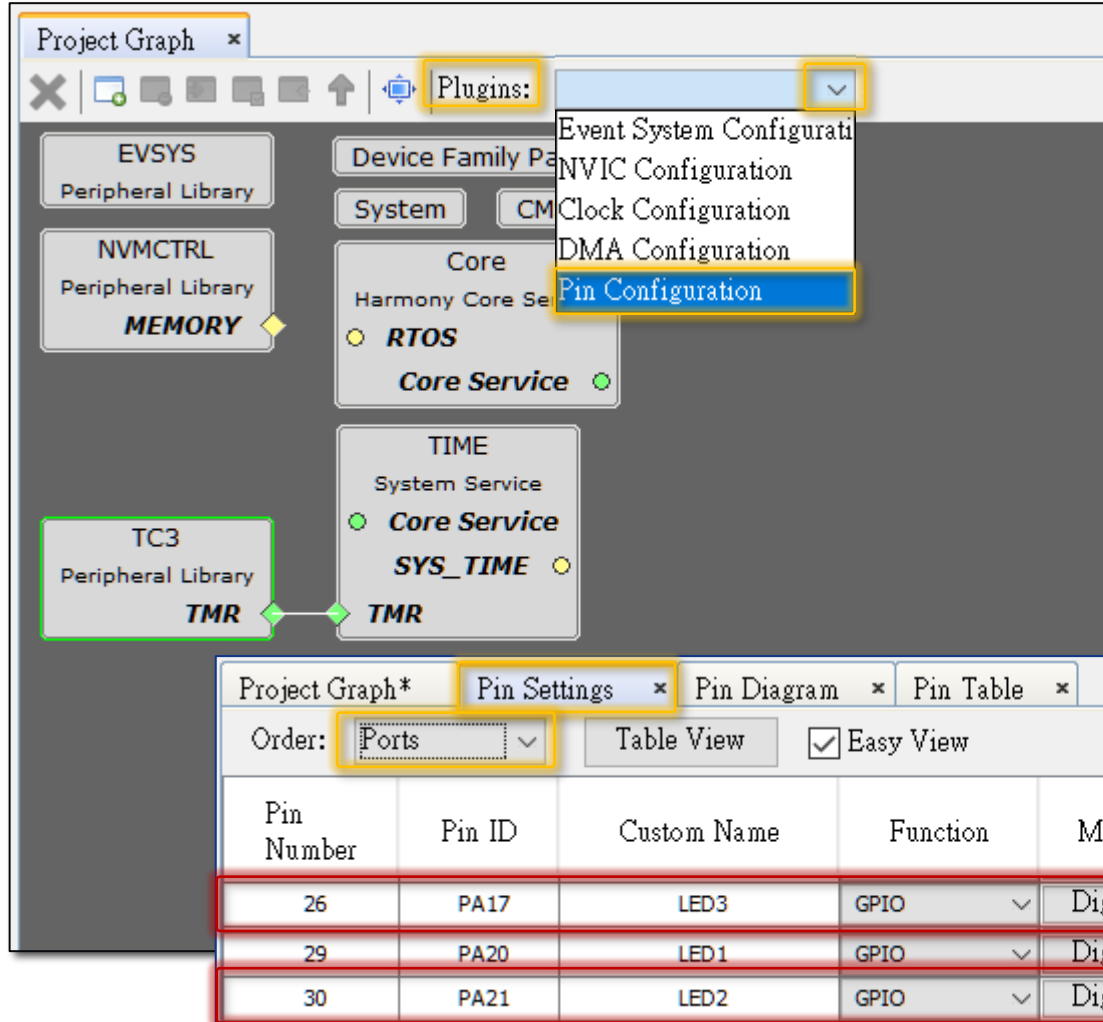
⚙️ Lab2 Callback based LED toggle

- Try use TIME System Service to create two **Periodic Callback** for toggle **LED2** in **500ms** period and toggle **LED3** in **250ms**.
- The **LED2** is **high active** on pin **PA21**.
- The **LED3** is **high active** on pin **PA17**.



✖ Lab2 Callback based LED toggle

- Harmony Configuration (Setup GPIO)



Project Graph

Plugins:

- Event System Configuration
- NVIC Configuration
- Clock Configuration
- DMA Configuration
- Pin Configuration

Device Family Pa

System

Core

Harmony Core Se

RTOS

Core Service

TIME

System Service

Core Service

SYS_TIME

TC3

Peripheral Library

TMR

Pin Settings

Order: Ports

Table View

Easy View

Pin Number	Pin ID	Custom Name	Function	Mode	Direction	Latch	Pull Up	Pull Down	Drive Strength
26	PA17	LED3	GPIO	Digi...	Out	Low	☐	☐	NORMAL
29	PA20	LED1	GPIO	Digi...	Out	Low	☐	☐	NORMAL
30	PA21	LED2	GPIO	Digi...	Out	Low	☐	☐	NORMAL

Configure
PA21 as GPIO output
PA17 as GPIO output
PA21 Custom Name to LED2
PA17 Custom Name to LED3
PA21 Latch is Low
PA17 Latch is Low

✂ Lab2 Callback based LED toggle

- Modify your main.c as below

```
#include <stddef.h>           // Defines NULL
#include <stdbool.h>          // Defines true
#include <stdlib.h>           // Defines EXIT_FAILURE
#include "definitions.h"      // SYS function prototypes

SYS_TIME_HANDLE tmrHandle;

void LED2Callback(uintptr_t context)
{
    LED2_Toggle();
}

void LED3Callback(uintptr_t context)
{
    LED3_Toggle();
}

int main( void )
{
    SYS_Initialize(NULL);

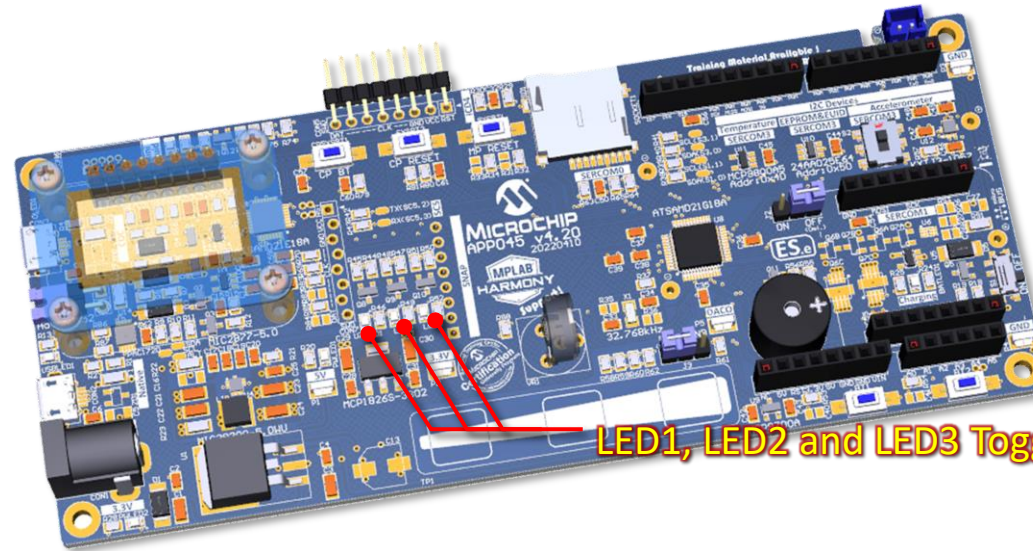
    SYS_TIME_DelayMS(1000, &tmrHandle);

    SYS_TIME_CallbackRegisterMS(LED2Callback, 0, 500, SYS_TIME_PERIODIC);
    SYS_TIME_CallbackRegisterMS(LED3Callback, 0, 250, SYS_TIME_PERIODIC);

    while( true )
    {
        if( SYS_TIME_DelayIsComplete(tmrHandle) == true )
        {
            LED1_Toggle();
        }
        ...
    }
}
```


⚙️ Lab2 Callback based LED toggle

- Result



LED1, LED2 and LED3 Toggle in binary counting !

LED1 (1000ms)	LED2 (500ms)	LED3 (250ms)
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

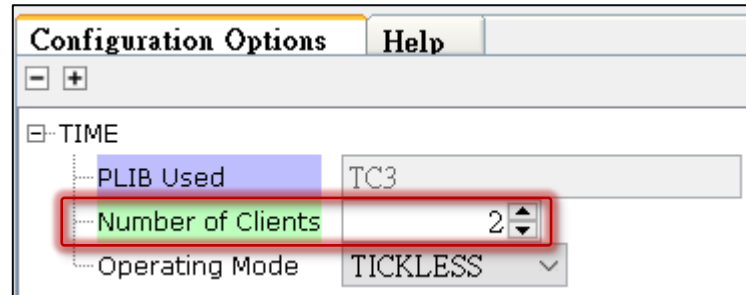
⚙️ Lab2 Callback based LED toggle

Discuss 1

- What's the effect if change Callback Type to **SYS_TIME_SINGLE** ?

```
SYS_TIME_CallbackRegisterMS(LED2Callback, 0, 500, SYS_TIME_SINGLE);  
SYS_TIME_CallbackRegisterMS(LED3Callback, 0, 250, SYS_TIME_PERIODIC);
```

- What's happen if change **Number of Client** from 5 to 2 ?

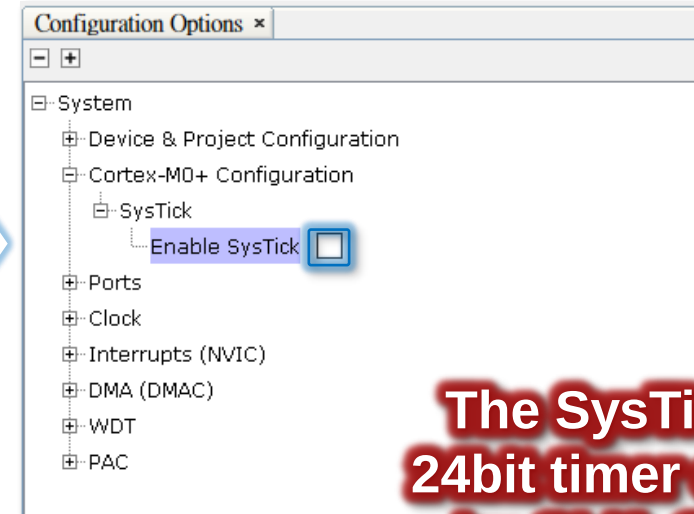
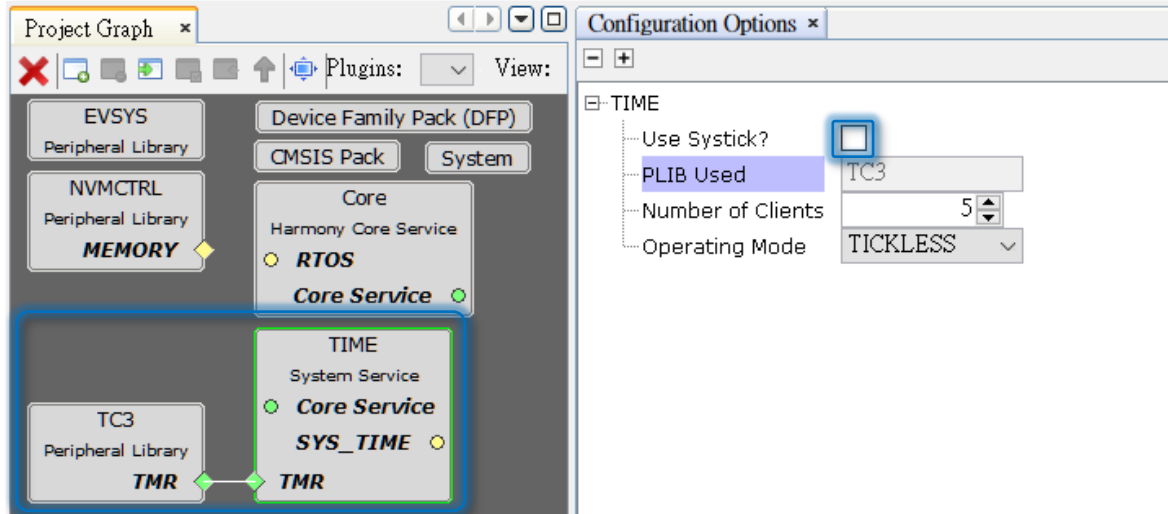


```
SYS_TIME_DelayMS(1000, &tmrHandle);  
SYS_TIME_CallbackRegisterMS(LED2Callback, 0, 500, SYS_TIME_PERIODIC);  
SYS_TIME_CallbackRegisterMS(LED3Callback, 0, 250, SYS_TIME_PERIODIC);
```

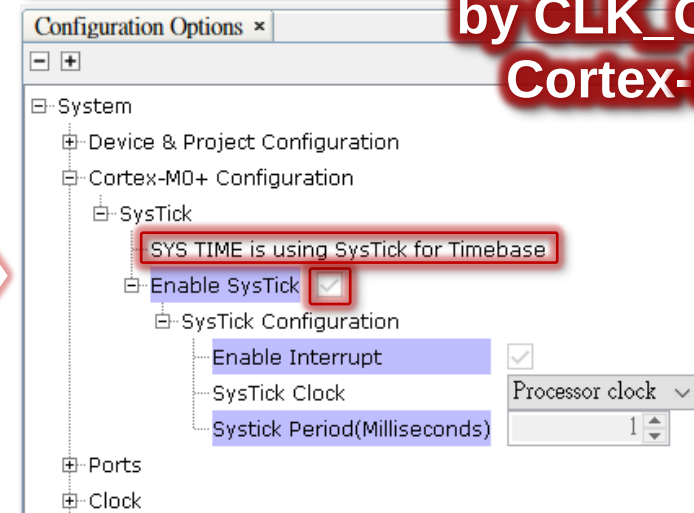
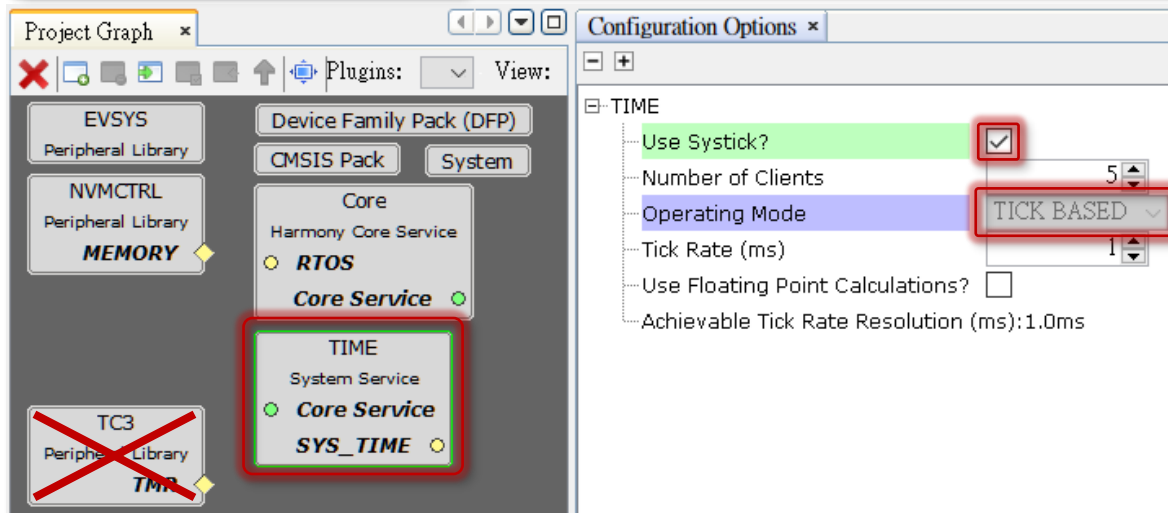
⚡ Lab2 Callback based LED toggle

Discuss 2

- How about to enable **Use Systick?** in TIME System Service ?



The SysTick is a 24bit timer clocked by CLK_CPU in Cortex-M0+



⚙️ Lab2 Callback based LED toggle

Discuss 3

- What's the **Achievable Tick Rate Resolution (ms)** in TIME System Service means ?

The screenshot displays the Microchip Harmony IDE interface. On the left, the 'Project Graph' shows the system architecture with components like EVSYS, NVMCTRL, TC3, and the TIME System Service. The TIME System Service is highlighted, showing its 'Core Service' and 'SYS_TIME' components. On the right, the 'Configuration Options' window for the TIME System Service is open. It shows the 'Operating Mode' set to 'TICK BASED' and the 'Tick Rate (ms)' set to 1.5. The 'Achievable Tick Rate Resolution (ms)' is calculated as 1.36531250194ms, which is highlighted with a red box. A second configuration window is shown below, with the 'Tick Rate (ms)' set to 1.2, and the 'Achievable Tick Rate Resolution (ms)' calculated as 1.2000000048ms, highlighted with a blue box.

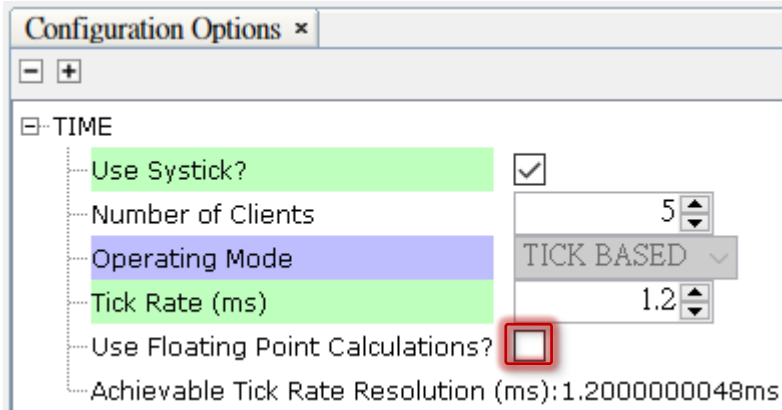
The maximum period of
16 bits TC3 @ 48MHz is
 $65535 \times (1 / 48 \text{ MHz}) =$
1.36531250194 ms
Desired Tick Rate **1.5 ms**
is not reachable

Desired Tick Rate **1.2 ms**
is $1 / 0.0012 = 833.33333... \text{ Hz}$
Truncate the decimal,
that's $1 / 833.33333 =$
1.2000000048 ms

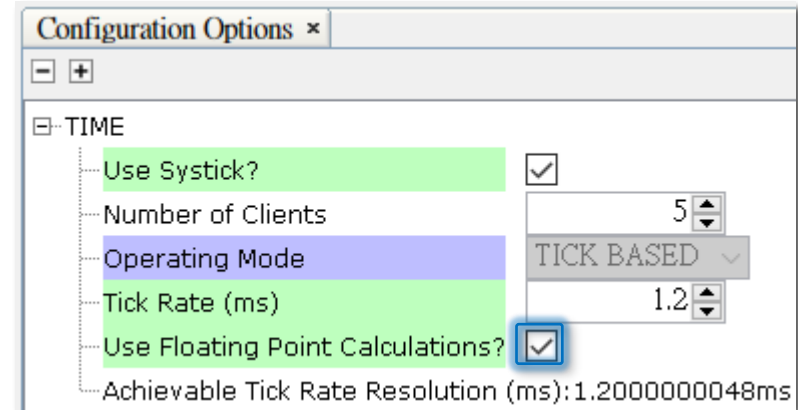
⚙️ Lab2 Callback based LED toggle

Discuss 4

- How about to enable **Use Floating Point Calculations?** in TIME System Service ?



```
#define SYS_TIME_TICK_FREQ_IN_HZ (833.33333)
uint32_t hwTimerTickFreq;
counterObj->hwTimerTickFreq = (uint32_t)SYS_TIME_TICK_FREQ_IN_HZ;
uint32_t SYS_TIME_USToCount ( uint32_t us )
{
    uint32_t count = (((uint64_t)us * gSystemCounterObj.hwTimerTickFreq) / 1000000);
    return count;
}
uint32_t SYS_TIME_MSToCount ( uint32_t ms )
{
    uint32_t count = (((uint64_t)ms * gSystemCounterObj.hwTimerTickFreq) / 1000);
    return count;
}
```



```
#define SYS_TIME_TICK_FREQ_IN_HZ (833.33333)
float hwTimerTickFreq;
counterObj->hwTimerTickFreq = SYS_TIME_TICK_FREQ_IN_HZ;
uint32_t SYS_TIME_USToCount ( uint32_t us )
{
    float count = ((us * gSystemCounterObj.hwTimerTickFreq) / 1000000);
    return (uint32_t) ((uint32_t)count + ((count - (uint32_t)count) > 0));
}
uint32_t SYS_TIME_MSToCount ( uint32_t ms )
{
    float count = ((ms * gSystemCounterObj.hwTimerTickFreq) / 1000);
    return (uint32_t) ((uint32_t)count + ((count - (uint32_t)count) > 0));
}
```

CONSOLE System Service

Harmony System Service

CONSOLE System Service

- **Introduction**

The Console System Service routes data or message traffic between a console device and a middleware layer or application. The most common use of the Console Service is to route debug or error messages to a terminal program on a host development system.

The Console System Service consists of a `core_layer` and a `console device_layer`. The `core_layer` handles module initialization and system functions. It provides a common API for use by applications and middleware libraries to send and receive data to and from one or more console devices. The Console System `core_layer` maintains the status and device descriptor information for each console instance that has been initialized.

The console device `layer` contains the functional implementation of the `core_layer` APIs. Each console device may have a different implementation, but the behavior of the API should be uniform across different devices. The console device `layer` interacts with the peripheral libraries or drivers or middle-ware to send and receive data to and from the hardware peripheral. The details of the data transaction are abstracted by the console device `layer`.

The console implementation is non-blocking in nature. For console write requests, the application data is copied into the ring buffer maintained by the console device `layer` and the call returns immediately. The return value of the API indicates the number of bytes actually copied. If the return value is less than the requested bytes, the application must retry after some time. The data transfer is controlled by a state machine implemented in the console device `layer`.

Similarly, for console read requests, the received data is copied to the application buffer from the internal receive ring buffer maintained by the console device `layer`. The return value indicates the number of bytes actually copied to the application buffer. Application must try reading again after some time, if the number of read bytes copied (indicated by the return value) is less than the bytes requested.

The size of the transmit and receive ring buffers is configurable in MHC.

CONSOLE System Service

- Harmony Configuration

The screenshot displays the Microchip Harmony IDE configuration interface for the CONSOLE System Service. The interface is divided into several panels:

- Configuration Options (Top Left):** Shows settings for the CONSOLE service:
 - Number of Instances Using UART: 2 (highlighted with a yellow circle)
 - Number of Instances Using USB: 1 (highlighted with a red circle)
 - Console Print Buffer Size (128-8192): 200
- Configuration Options (Top Right):** Shows settings for the SERCOM5 peripheral:
 - Device Used: SERCOM5
 - TX/RX Ring Buffers must be configured in the attached PLIB (highlighted with a yellow box)
- Peripheral Library (Middle Left):** Shows the USB Full Speed Driver and CDC Function Driver, both with their respective instance counts (1 and 2) highlighted with yellow circles.
- Peripheral Library (Middle Right):** Shows the SERCOM5 peripheral with its operation mode set to "USART with internal Clock" and its ring buffer sizes (TX and RX) set to 128 (highlighted with a yellow box).
- Instances (Bottom):** Shows the instances of the CONSOLE service, including Instance 0, Instance 1, and Instance 2. The instances are labeled with their respective peripheral libraries (UART, USB_DEVICE_CDC, SYS_CONSOLE) and are highlighted with yellow circles.

A red box highlights the USB Device CDC Instance, which is supported from MHC 3.5.0. A yellow box highlights the TX/RX Ring Buffers must be configured in the attached PLIB. A blue box highlights the Consumers section, which includes COMMAND (sys_command), DEBUG (sys_debug), and TCPIP CORE (tcpipStack).

USB Device CDC Instance Supported from MHC 3.5.0

CONSOLE System Service

API functions of the Console System Service library.

- **System Functions**

SYS_CONSOLE_Initialize

Initializes the console instance module and opens or initializes the specific module instance to which it is associated.

SYS_CONSOLE_Status

Returns status of the specific instance of the Console module.

SYS_CONSOLE_Tasks

Maintains the console's state machine.

- **Core Functions**

SYS_CONSOLE_DeviceGet

Returns the device type for a given console instance

SYS_CONSOLE_HandleGet

Returns a handle to the requested console instance

SYS_CONSOLE_Read

Reads data from the console device.

SYS_CONSOLE_ReadCountGet

Returns number of unread bytes available in the receive buffer.

SYS_CONSOLE_ReadFreeBufferCountGet

Returns the amount of free space in bytes available in the receive buffer.

SYS_CONSOLE_Write

Writes data to the console device.

SYS_CONSOLE_WriteCountGet

Returns the number of bytes pending for transmission in the buffer.

SYS_CONSOLE_WriteFreeBufferCountGet

Returns the amount of free space in bytes in the transmit buffer.

SYS_CONSOLE_Flush

Flushes the read and write queues for the given console instance.

SYS_CONSOLE_Message

Prints a message to the console

SYS_CONSOLE_Print

Formats and prints a message with a variable number of arguments to the console

CONSOLE System Service

API functions of the Console System Service library. (cont.)

- Data Types and Constants

STDERR_FILENO

This is macro STDERR_FILENO.

STDIN_FILENO

These are in unistd.h

STDOUT_FILENO

This is macro STDOUT_FILENO.

SYS_CONSOLE_DEVICE

Lists the available console devices.

SYS_CONSOLE_HANDLE

Handle to an instance of the console system service.

SYS_CONSOLE_MESSAGE

Print message on the default console instance

SYS_CONSOLE_PRINT

Print formatted message on the default console instance

SYS_CONSOLE_DEFAULT_INSTANCE

This is macro SYS_CONSOLE_DEFAULT_INSTANCE.

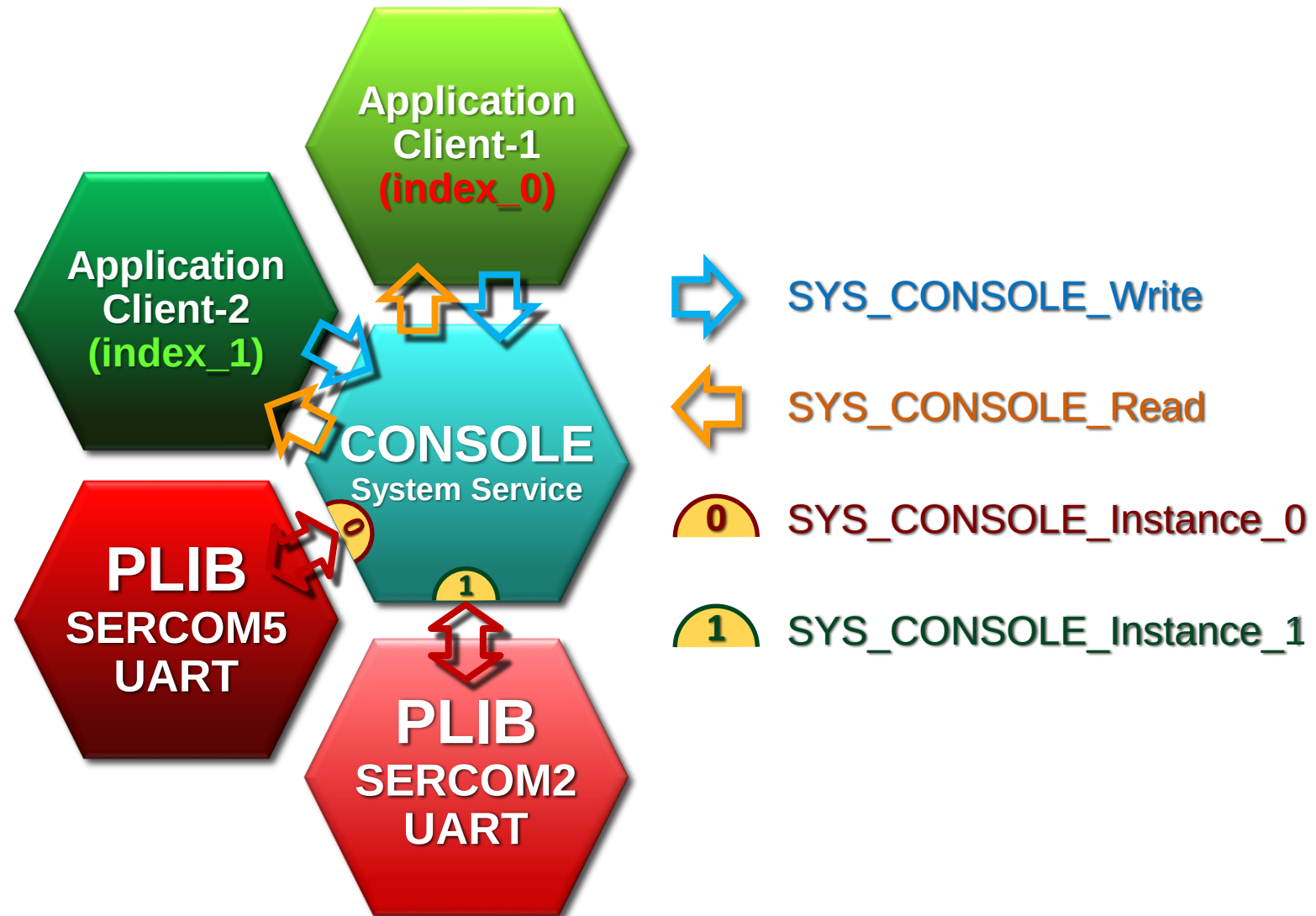
SYS_CONSOLE_HANDLE_INVALID

Invalid console handle

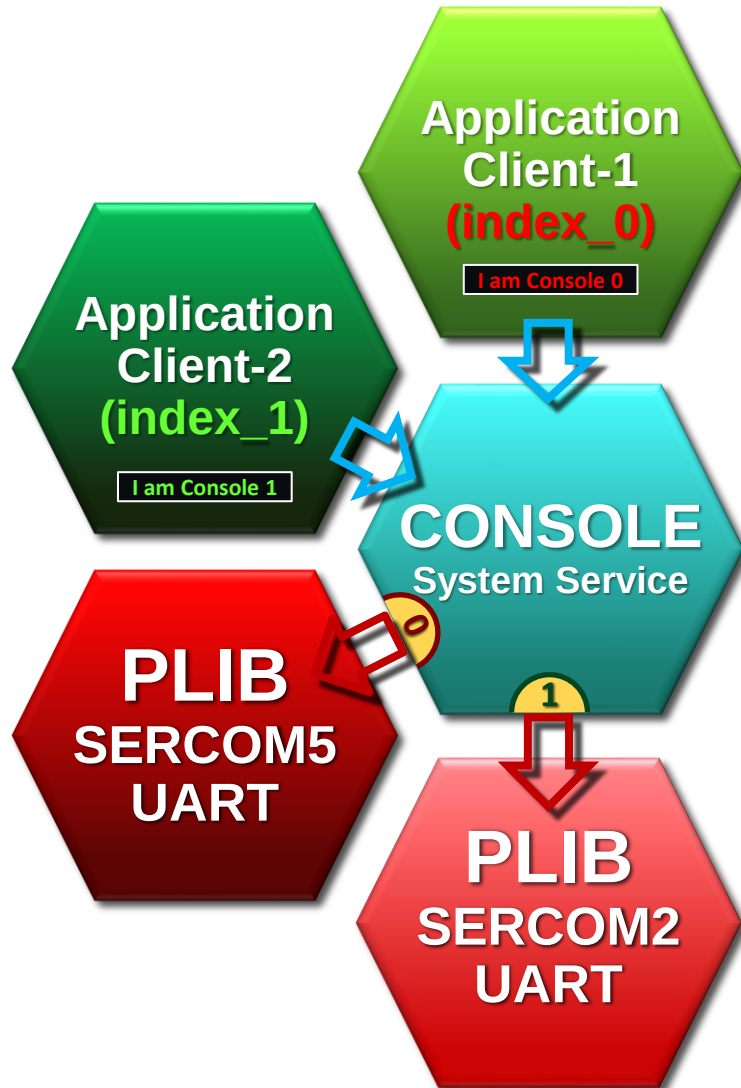
SYS_CONSOLE_STATUS

This is type SYS_CONSOLE_STATUS.

CONSOLE System Service



CONSOLE System Service



```
void App1_Task()
{
    SYS_CONSOLE_HANDLE consoleHandle0;
    consoleHandle0 = SYS_CONSOLE_HandleGet(
        SYS_CONSOLE_INDEX_0 );
    SYS_CONSOLE_Write( consoleHandle0,
        "I am Console 0\r\n", 16 );
}

void App2_Task()
{
    SYS_CONSOLE_HANDLE consoleHandle1;
    consoleHandle1 = SYS_CONSOLE_HandleGet(
        SYS_CONSOLE_INDEX_1 );
    SYS_CONSOLE_Write( consoleHandle1,
        "I am Console 1\r\n", 16 );
}
```

VT COM3 - Tera Term VT
File Edit Setup Control Win
I am Console 0

VT COM23 - Tera Te...
File Edit Setup Control Win
I am Console 1

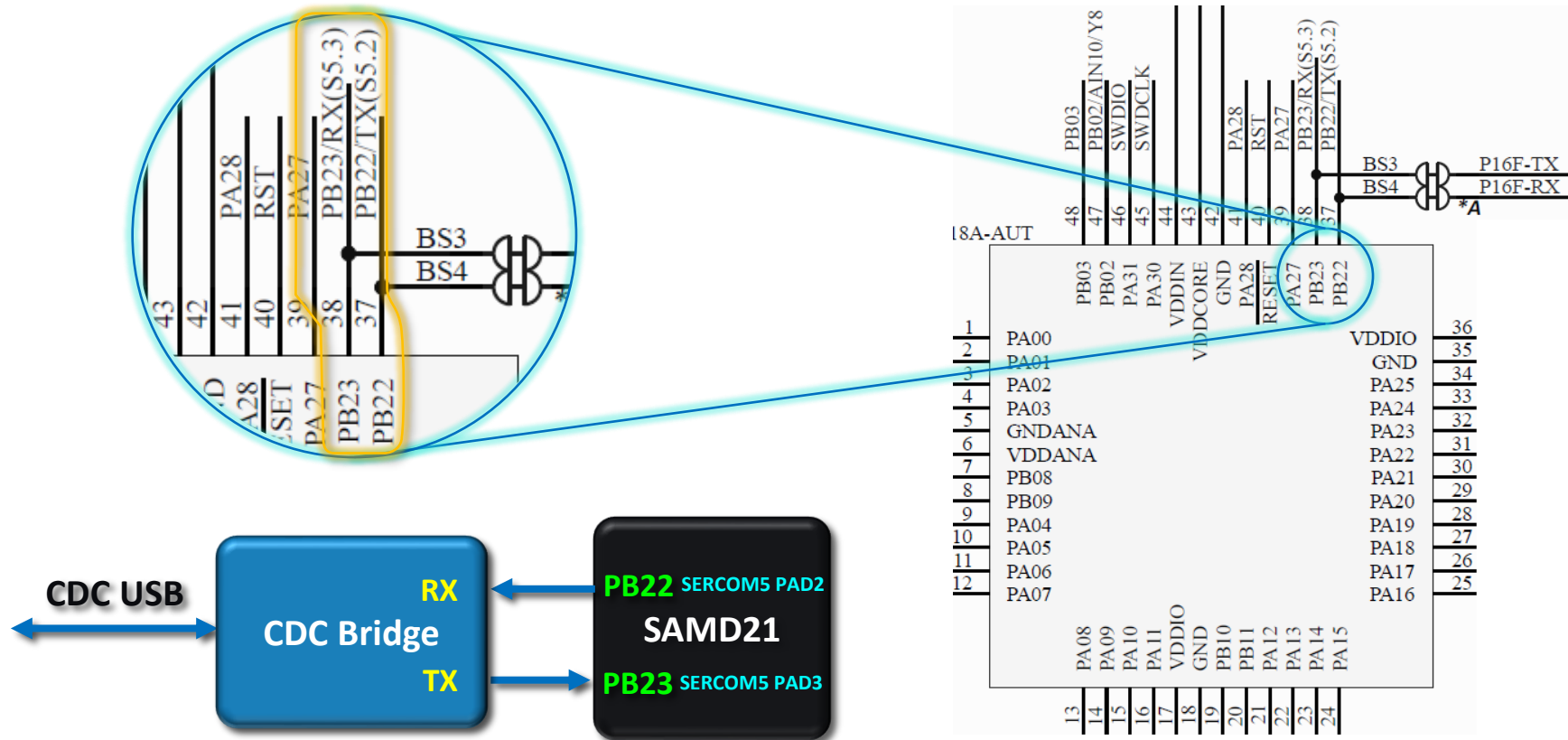
Lab 3

UART Console

CONSOLE System Service

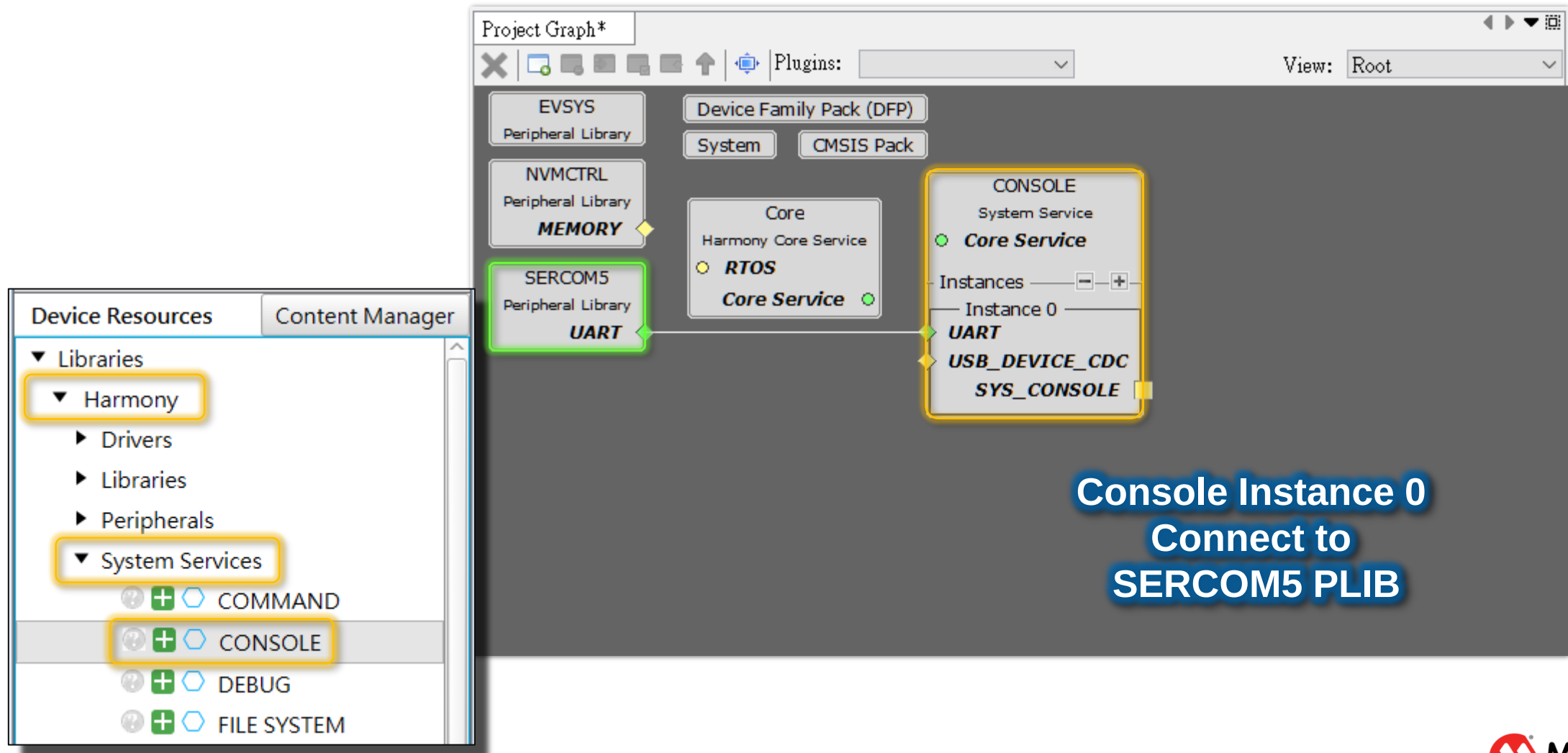
🔧 Lab3 UART Console

- Try add **CONSOLE System Service** to create a **UART console**.
- Select **SERCOM5 PLIB** as **Instance 0** of UART Console.
- Output **“Hello World!”** to UART console.
- Receive console input **X** and print it out as **“Receive : X”**



⚙️ Lab3 UART Console

- Harmony Configuration (Add CONSOLE System Service)



⚙️ Lab3 UART Console

- Harmony Configuration (CONSOLE and PLIB SERCOM5 config)

The screenshot displays the Microchip Harmony IDE configuration interface. The main Project Graph shows the system architecture with components like EVSYS, NVMCTRL, SERCOM5, and the Core Service. The SERCOM5 component is highlighted with a green box, and its UART instance is also highlighted. The Configuration Options for the CONSOLE instance are shown on the right, with the Console Print Buffer Size set to 200. The Configuration Options for the SERCOM5 instance are shown on the left, with the TX and RX Ring Buffer Sizes set to 128. A text overlay in the bottom right corner states: "Print buffer size should depend to proper size of SERCOM Ring Buffer size".

Project Graph*

View: Root

Configuration Options *

CONSOLE

System Service

Core Service

Instances

Instance 0

UART

USB_DEVICE_CDC

SYS_CONSOLE

Configuration Options *

SERCOM5

Select SERCOM Operation Mode: USART with internal Clock

Operating Mode: Ring buffer mode

Configure Ring Buffer Size-

TX Ring Buffer Size: 128

RX Ring Buffer Size: 128

Receive Enable: ☒

Transmit Enable: ☒

Frame Format: USART frame

Baud Rate in Hz: 115,200

Parity Mode: No Parity

Character Size: 8 Bits

Stop Bit Mode: One Stop Bit

Start-of-Frame Detection Enable: ☐

Receive Pinout: SERCOM PAD[3] is used for data reception

Transmit Pinout: PAD[2] = TxD; PAD[3] = XCK

Enable Run in Standby: ☐

Configuration Options *

CONSOLE

Number of Instances Using UART: 1

Number of Instances Using USB: 0

Console Print Buffer Size (128-8192): 200

Configuration Options *

CONSOLE

Device Used: SERCOM5

TX/RX Ring Buffers must be configured in the attached PLIB

Print buffer size should depend to proper size of SERCOM Ring Buffer size

Receive Pinout : SERCOM PAD[3]
Transmit Pinout: SERCOM PAD[2]

✖ Lab3 UART Console

- Harmony Configuration (Setup PLIB SERCOM5 pads)

Project Graph* Pin Diagram * Pin Table * Pin Settings *									
Order:	Ports	Table View	☒ Easy View						
Pin Number	Pin ID	Custom Name	Function	Mode	Direction	Latch	Pull Up	Pull Down	Drive Strength
37	PB22		SERCOM5_PAD2	Digital	High Impedance	n/a	☐	☐	NORMAL
38	PB23		SERCOM5_PAD3	Digital	High Impedance	n/a	☐	☐	NORMAL

Configure PB22 as SERCOM5_PAD2
Configure PB23 as SERCOM5_PAD3

Project Graph* Pin Diagram * Pin Table * Pin Settings *

Package: TQFP48

		2	23	24	25	26	27	28	29	30	31	32	33	34	39	41	45	46	47	48	7	8	19	20	37	38
		A																		B						
Module	Function	3	14	15	16	17	18	19	20	21	22	23	24	25	27	28	30	31	2	3	8	9	10	11	22	23
SERCOM5	SERCOM5_PAD0																									
	SERCOM5_PAD1																									
	SERCOM5_PAD2																									
	SERCOM5_PAD3																									

⚙️ Lab3 UART Console

- Modify your main.c as below

```
#include "definitions.h"                // SYS function prototypes
#include <stdio.h>
#include <string.h>

SYS_TIME_HANDLE tmrHandle;
SYS_CONSOLE_HANDLE consoleHandlePLIB;
char UartWriteBuf[30];
char UartReadBuf[10];

void LED2Callback(uintptr_t context) {...}
void LED3Callback(uintptr_t context) {...}

int main( void )
{
    SYS_Initialize(NULL);

    consoleHandlePLIB = SYS_CONSOLE_HandleGet( SYS_CONSOLE_INDEX_0 );
    sprintf(UartWriteBuf, "Hello world !(PLIB)\r\n");
    SYS_CONSOLE_Write(consoleHandlePLIB, UartWriteBuf, strlen(UartWriteBuf));

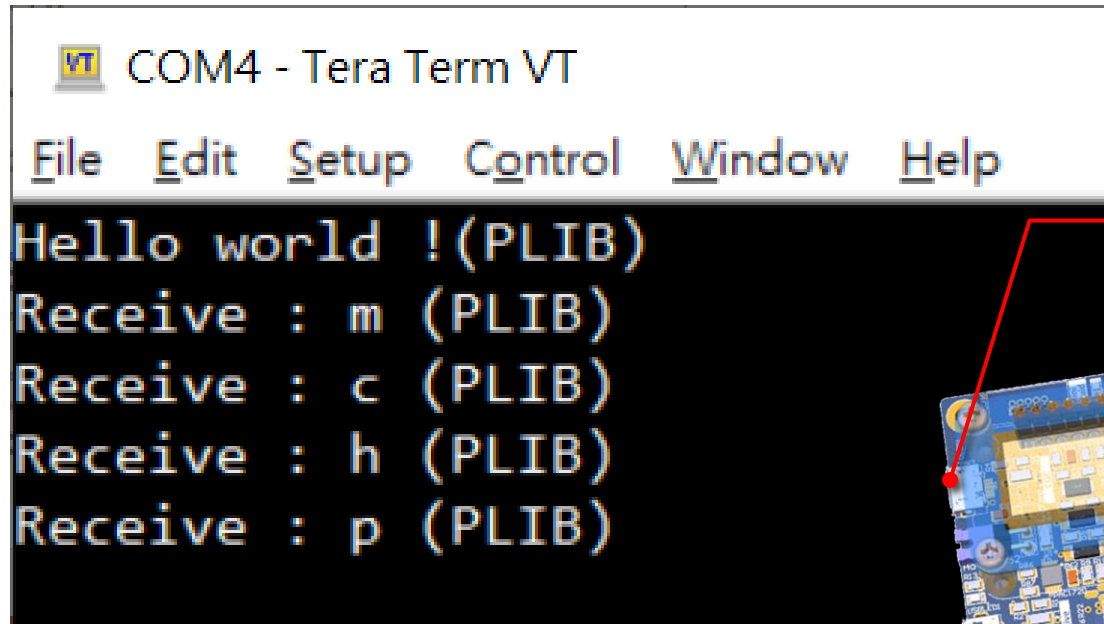
    SYS_TIME_DelayMS(1000, &tmrHandle);
    while ( true )
    {
        if( SYS_TIME_DelayIsComplete(tmrHandle) == true ) {...}

        if( SYS_CONSOLE_Read(consoleHandlePLIB, UartReadBuf, 1) > 0 )
        {
            sprintf(UartWriteBuf, "Receive : %s (PLIB)\r\n", UartReadBuf);
            SYS_CONSOLE_Write(consoleHandlePLIB, UartWriteBuf, strlen(UartWriteBuf));
        }

        SYS_Tasks ( );
    }
}
```

⚙️ Lab3 UART Console

- Compiler and programming your project to EVB.
- Open COM Port terminal console 115200-n-8-1.
- Check the UART output in console.
- Type any character in console and check the response.

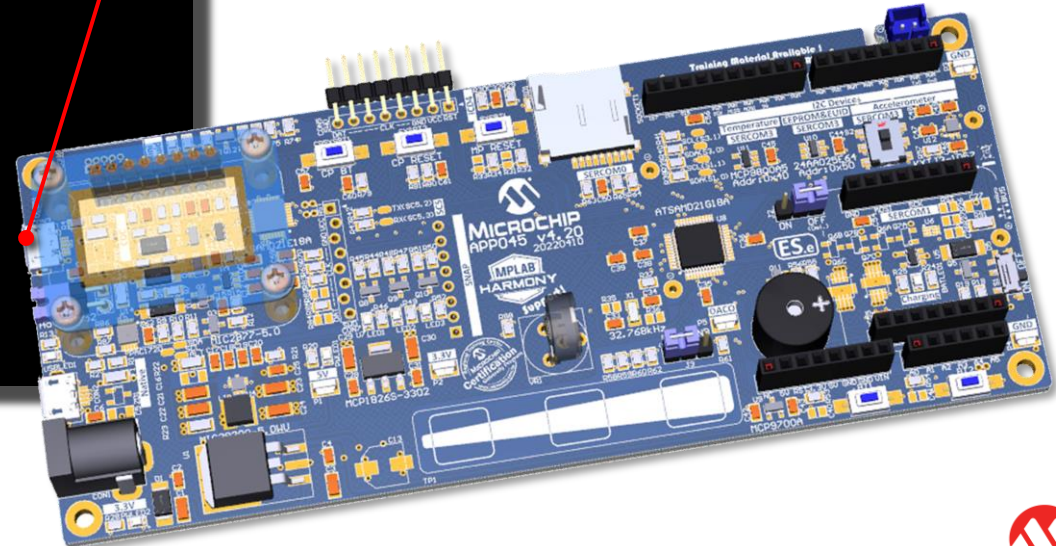


VT COM4 - Tera Term VT

File Edit Setup Control Window Help

Hello world !(PLIB)
Receive : m (PLIB)
Receive : c (PLIB)
Receive : h (PLIB)
Receive : p (PLIB)

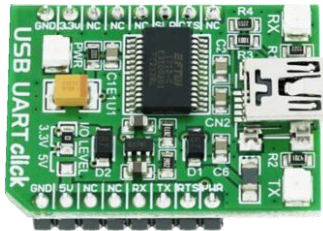
USB CDC for UART console



✖ Lab3 UART Console

Discuss 1

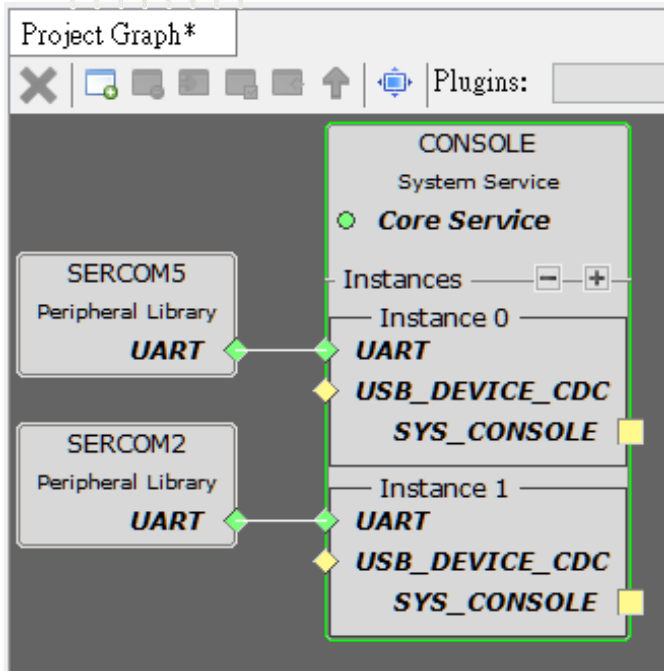
- Try to implement multi-Instance consoles of **CONSOLE** system service.
 - You can plug a USB UART click EVB for second instance of CONSOLE.



mikroBUS
USB UART click

Project Graph* Pin Settings x Pin Diagram x Pin Table x			
Order:	Ports	Table View	☒ Easy View
Pin Number	Pin ID	Custom Name	Function
15	PA10	SERCOM2_PAD2	SERCOM2_PAD2
16	PA11	SERCOM2_PAD3	SERCOM2_PAD3

MikroBUS			
1	AN	PWM	16
2	RST	INT	15
3	CS	RX	14 PA11/RX
4	SCK	TX	13 PA10/TX
5	MISO	SCL	12
6	MOSI	SDA	11
7	3.3V	5V	10
8	GND	GND	9 GND



SERCOM2

Select SERCOM Operation Mode

Enable Interrupts ?

Receive Enable

Transmit Enable

Enable Ring Buffer ?

Enable Run in Standby

Receive Pinout

Transmit Pinout

☒

☒

☒

☒

☐

SERCOM PAD[3] is used for data reception

PAD[2] = TxD; PAD[3] = XCK

COM4 - Tera Term VT

File Edit Setup Control

Hello world !(PLIB0)

Receive : m (PLIB1)

Receive : c (PLIB1)

Receive : h (PLIB1)

Receive : p (PLIB1)

COM5 - Tera Term VT

File Edit Setup Control

Hello world !(PLIB1)

Receive : 1 (PLIB0)

Receive : 2 (PLIB0)

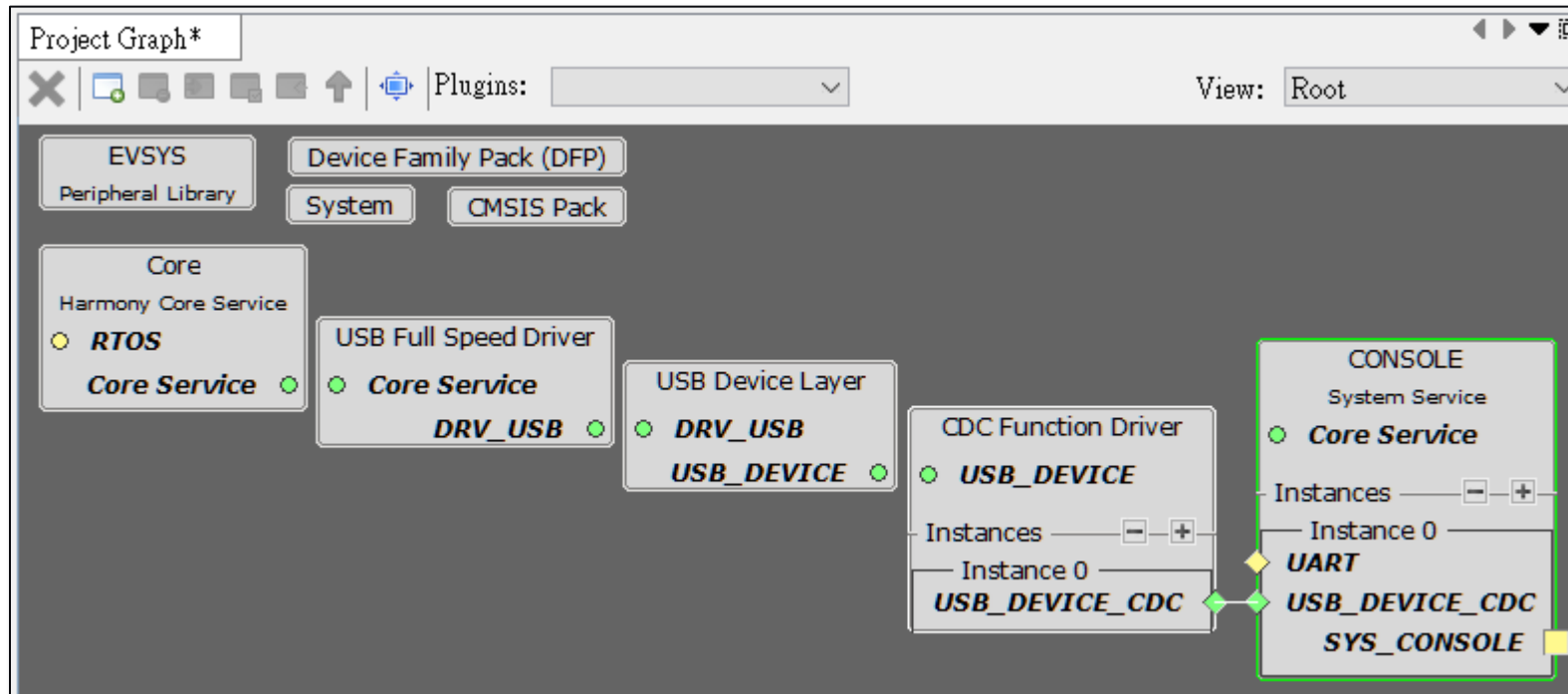
Receive : 3 (PLIB0)

Receive : 4 (PLIB0)

⚙️ Lab3 UART Console

Discuss 2

- The **CONSOLE System Service** could support the **USB Device CDC Library** to develop another USB COM Port through the **native USB interface** of SAMD21, you could practice the Lab6 firstly after you finish Lab3.
 - To go to [Lab6 – USB Device CDC with CONSOLE System Service](#)



DEBUG System Service

Harmony System Service

DEBUG System Service

- **Introduction**

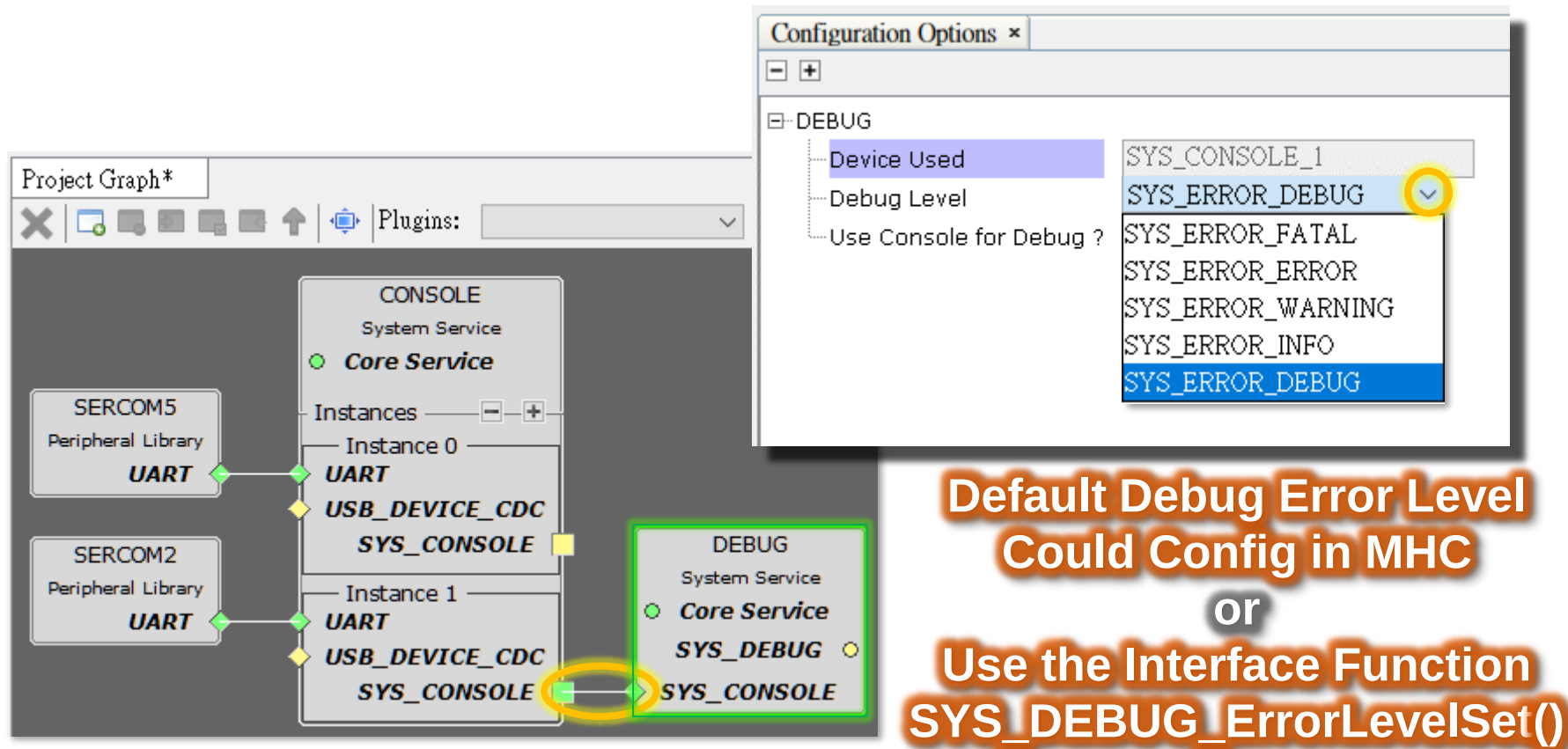
The Debug System Service provides a convenient mechanism for the application developer to send formatted or unformatted messages to a system console.

The Debug System Service provides a set of functions that allow the developer to output debug and/or error messages based on a global error level.

The messages are sent to the System Console Service, where they are routed to a console device.

The Debug System Service maintains a global error level, which may be set during initialization and changed dynamically at run-time. Both formatted and unformatted messages can be output or not, depending on the current global error level.

DEBUG System Service



**DEBUG System Service
must depend to
CONSOLE system Service**

DEBUG System Service

API functions of the Debug System Service library.

- System Functions

`SYS_DEBUG_Initialize`

Initializes the global error level and specific module instance.

`SYS_DEBUG_Status`

Returns status of the specific instance of the debug service module.

`SYS_DEBUG_BreakPoint`

Inserts a software breakpoint instruction when building in Debug mode.

- Changing System Error Level Functions

`SYS_DEBUG_ErrorLevelGet`

Returns the global system Error reporting level.

`SYS_DEBUG_ErrorLevelSet`

Sets the global system error reporting level.

- Data Types and Constants

`SYS_ERROR_LEVEL`

System error message priority levels.

`SYS_DEBUG_INDEX_x`

This is macro `SYS_DEBUG_INDEX_x`.

`SYS_DEBUG_MESSAGE`

Macro of `_SYS_DEBUG_MESSAGE`.

`SYS_DEBUG_PRINT`

Macro of `_SYS_DEBUG_PRINT`.

`_SYS_DEBUG_MESSAGE`

Prints a debug message if the specified level is $\leq$ global system error level.

`_SYS_DEBUG_PRINT`

Format prints a debug message if the specified level is $\leq$ global system error level.

- Other Functions

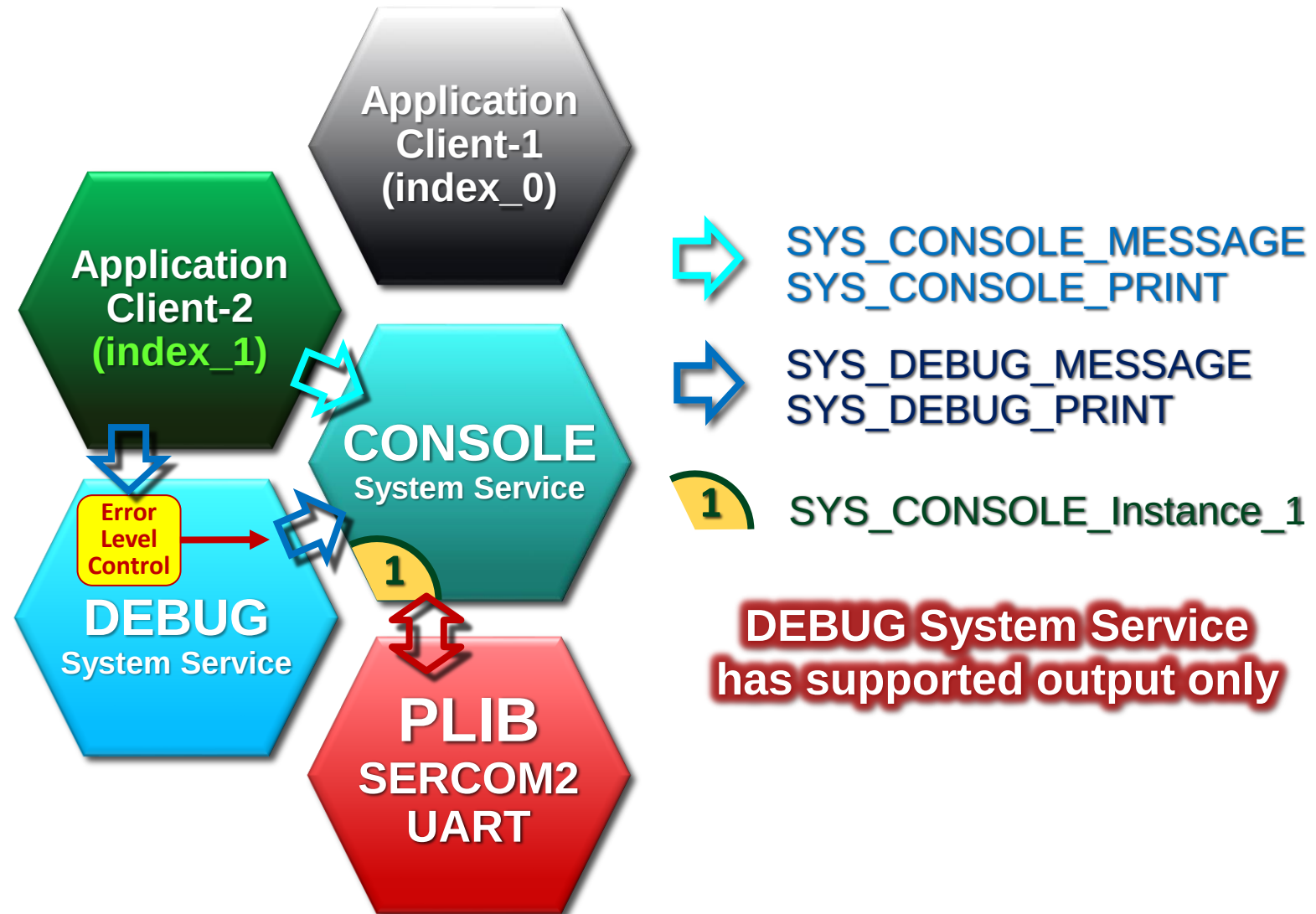
`SYS_DEBUG_Redirect`

Allows re-direction of debug system service to another console instance

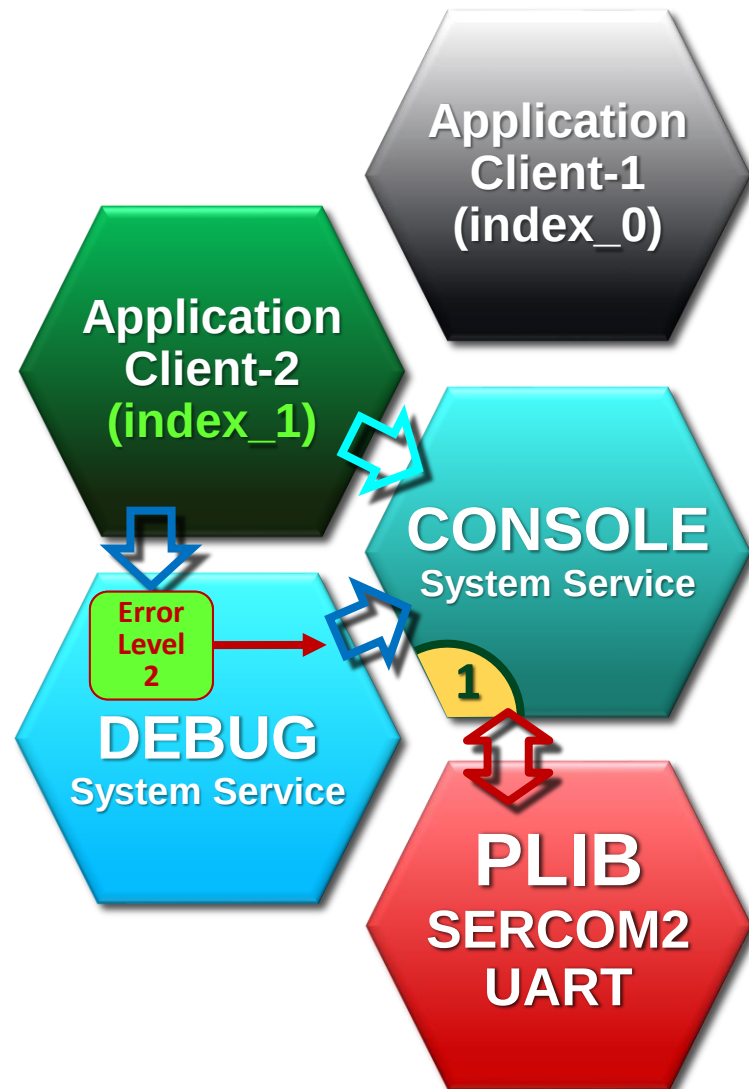
`SYS_DEBUG_ConsoleInstanceGet`

This is function `SYS_DEBUG_ConsoleInstanceGet`.

DEBUG System Service



DEBUG System Service



```
void App2_Task()...
{
    int EL = SYS_DEBUG_ErrorLevelGet();

    SYS_CONSOLE_MESSAGE("Message to Console 1\r\n");
    SYS_CONSOLE_PRINT("Current Error Level: %d\r\n", EL );
    SYS_DEBUG_MESSAGE(SYS_ERROR_DEBUG, "dErrlv 4\r\n");
    SYS_DEBUG_PRINT(SYS_ERROR_WARNING, "pErrlv %d\r\n", EL);
    SYS_CONSOLE_MESSAGE("Set Error Level to WARNING\r\n");
    SYS_DEBUG_ErrorLevelSet(SYS_ERROR_WARNING);

    SYS_DEBUG_MESSAGE(SYS_ERROR_DEBUG, "dErrlv 4\r\n");
    EL = SYS_DEBUG_ErrorLevelGet();
    SYS_DEBUG_PRINT(SYS_ERROR_WARNING, "pErrlv %d\r\n", EL);
}
```

DEBUG

Device Used	SYS_CONSOLE_1
Debug Level	SYS_ERROR_DEBUG
Debug Print Buffer Size (128-8192)	200
Use Console for Debug ?	☒

COM24 - Tera Term VT

File Edit Setup Control Window

Message to Console 1
Current Error Level: 4
dErrlv 4
pErrlv 4
Set Error Level to WARNING
pErrlv 2

Name	Lv
FATAL	0
ERROR	1
WARNING	2
INFO	3
DEBUG	4

Lab 4

Debug Message Control

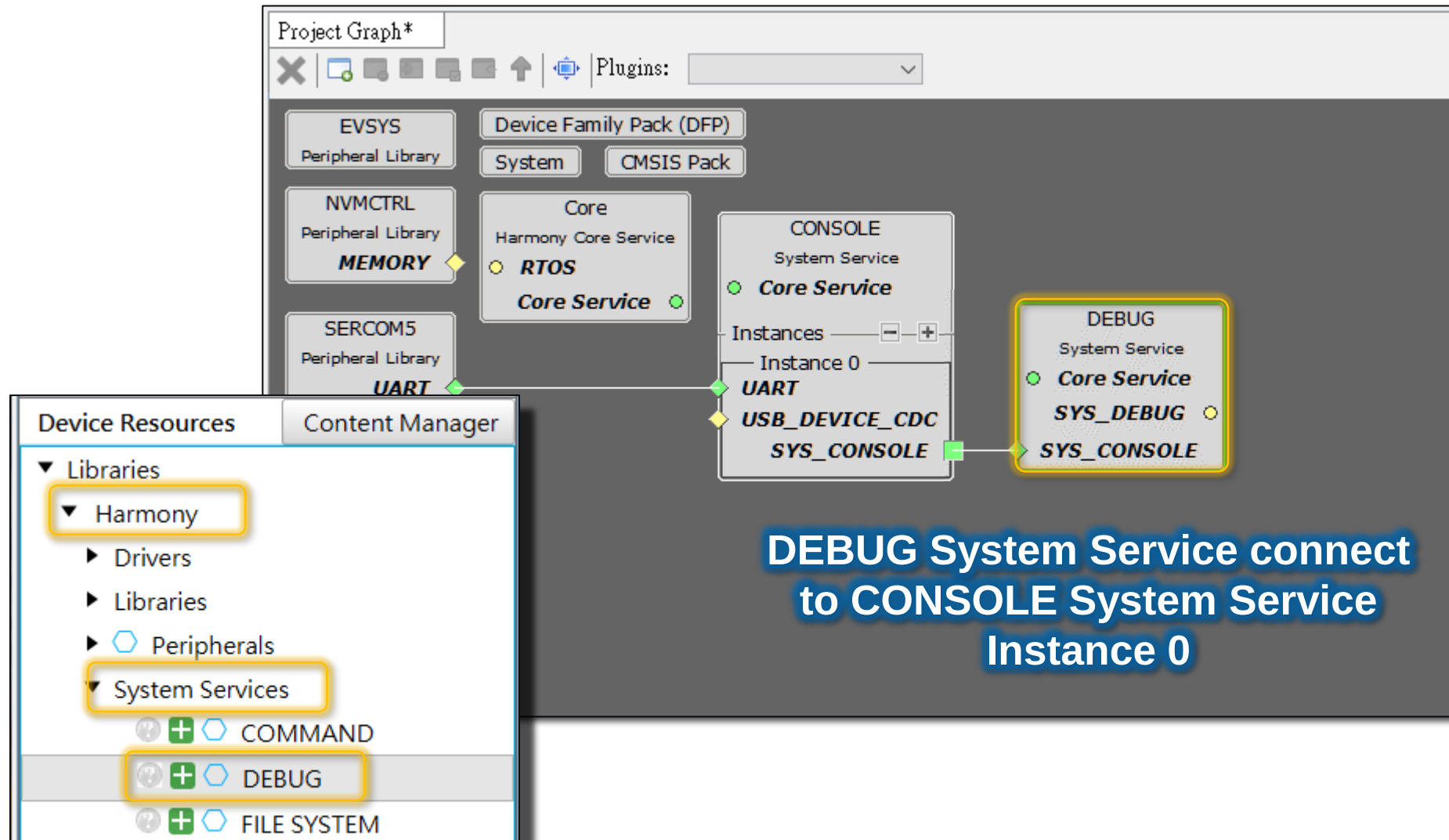
DEBUG System Service

⚙️ Lab4 Debug Message Control

- Try add **DEBUG System Service** and output message through UART to use previous added **CONSOLE System Service**.
- Use **CONSOLE System Service** Macro to output “**Hello world !**” and “**Receive : X**” message.
- Input number ‘0’, ‘1’, ‘2’, ‘3’, ‘4’ in UART console and use **DEBUG System Service** to show error-level controlled message “**Change Error Level to #**” (the # is input number) and then change the correspond **Error Level** could control five different Error Level debug message output as below.
 - “**Error Level 0 - Fatal**”
 - “**Error Level 1 - Error**”
 - “**Error Level 2 - Warning**”
 - “**Error Level 3 - Info**”
 - “**Error Level 4 - Debug**”

⚙️ Lab4 Debug Message Control

- Harmony Configuration (Add DEBUG System Service)



⚙️ Lab4 Debug Message Control

- Harmony Configuration (DEBUG System Service config)

The screenshot displays the MHC Project Graph and the Configuration Options dialog for the DEBUG system service.

Project Graph:

- Peripheral Library:** EVSYS, NVMCTRL (MEMORY), SERCOM5 (UART).
- Device Family Pack (DFP):** System, CMSIS Pack.
- Core:** Harmony Core Service, RTOS, Core Service.
- CONSOLE System Service:** Core Service, Instance 0 (UART, USB_DEVICE_CDC, SYS_CONSOLE).
- DEBUG System Service:** Core Service, SYS_DEBUG, SYS_CONSOLE (highlighted with a green box).

Configuration Options:

- DEBUG:**
 - Device Used: SYS_CONSOLE_0
 - Debug Level: SYS_ERROR_DEBUG (selected in the dropdown)
 - Use Console for Debug?: ☒

Annotations:

- Use Console for Debug is natural** (blue text with a shadow)
- Default Error Level could Config in MHC** (orange text with a shadow)

⚙️ Lab4 Debug Message Control

- Modify your main.c as below

Comment out

```
int main( void )
{
    ...
    // consoleHandlePLIB = SYS_CONSOLE_HandleGet( SYS_CONSOLE_INDEX_0 );
    // sprintf(UartWriteBuf, "Hello world !(PLIB)\r\n");
    // SYS_CONSOLE_Write(consoleHandlePLIB, UartWriteBuf, strlen(UartWriteBuf));
    SYS_CONSOLE_MESSAGE("Hello world !(PLIB)\r\n");
    ...
    while ( true )
    {
        ...
        if( SYS_CONSOLE_Read(consoleHandlePLIB, UartReadBuf, 1) > 0 )
        {
            // sprintf(UartWriteBuf, "Receive : %s (PLIB)\r\n", UartReadBuf);
            // SYS_CONSOLE_Write(consoleHandlePLIB, UartWriteBuf, strlen(UartWriteBuf));
            SYS_CONSOLE_PRINT("\r\nReceive : %s (PLIB)\r\n", UartReadBuf);

            if(UartReadBuf[0]>='0' && UartReadBuf[0]<='4')
            {
                int ErrorLevel = UartReadBuf[0] - '0';

                SYS_DEBUG_ErrorLevelSet(ErrorLevel);
                SYS_DEBUG_PRINT(SYS_ERROR_FATAL, "Change Error Level to %d\r\n",
                                SYS_DEBUG_ErrorLevelGet());
            }
            SYS_DEBUG_MESSAGE(SYS_ERROR_FATAL, "Error Level 0 - Fatal\r\n");
            SYS_DEBUG_MESSAGE(SYS_ERROR_ERROR, "Error Level 1 - Error\r\n");
            SYS_DEBUG_MESSAGE(SYS_ERROR_WARNING, "Error Level 2 - Warning\r\n");
            SYS_DEBUG_MESSAGE(SYS_ERROR_INFO, "Error Level 3 - Info\r\n");
            SYS_DEBUG_MESSAGE(SYS_ERROR_DEBUG, "Error Level 4 - Debug\r\n");
        }
    }
}
```

Comment out

✖ Lab4 Debug Message Control

- Compiler and programming your project to EVB.
- Type '0', '1', '2', '3', '4', 'a', 'b' in console to check the difference.

```
COM4 - Tera Ter...  
File Edit Setup Control Window Help  
Hello world !(PLIB)  
Receive : 0 (PLIB)  
Change Error Level to 0  
Error Level 0 - Fatal  
Receive : 1 (PLIB)  
Change Error Level to 1  
Error Level 0 - Fatal  
Error Level 1 - Error  
Receive : 2 (PLIB)  
Change Error Level to 2  
Error Level 0 - Fatal  
Error Level 1 - Error  
Error Level 2 - Warning
```

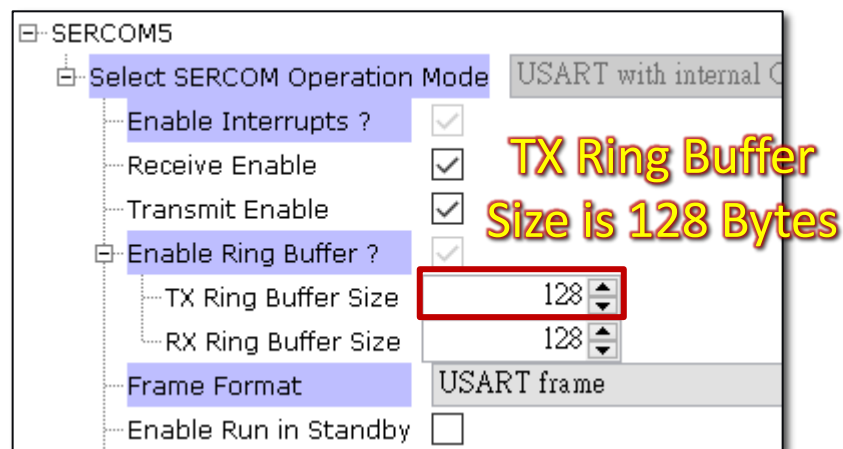
```
COM4 - Tera Ter...  
File Edit Setup Control Window Help  
Receive : 3 (PLIB)  
Change Error Level to 3  
Error Level 0 - Fatal  
Error Level 1 - Error  
Error Level 2 - Warning  
Error Level 3 -   
Receive : 4 (PLIB)  
Change Error Level to 4  
Error Level 0 - Fatal  
Error Level 1 - Error  
Error Level 2 - Warning  
Error Level 3 - 
```

```
COM4 - Tera Ter...  
File Edit Setup Control Window Help  
Receive : a (PLIB)  
Error Level 0 - Fatal  
Error Level 1 - Error  
Error Level 2 - Warning  
Error Level 3 - Info  
Error Level 4 -   
Receive : b (PLIB)  
Error Level 0 - Fatal  
Error Level 1 - Error  
Error Level 2 - Warning  
Error Level 3 - Info  
Error Level 4 - 
```

✖ Lab4 Debug Message Control

- Why DEBUG print cannot finish output ?

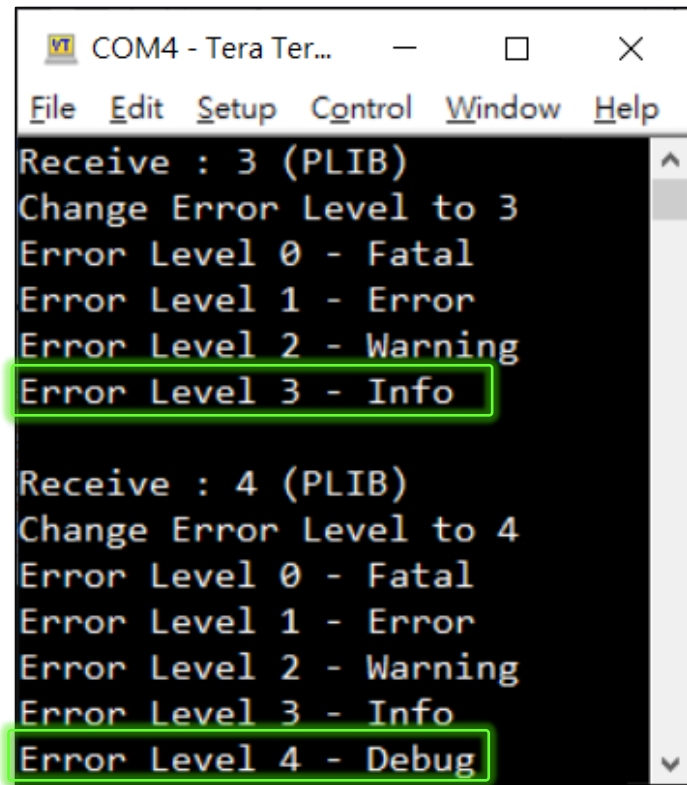
```
COM4 - Tera Ter...  
File Edit Setup Control Window Help  
Receive : 3 (PLIB)  
Change Error Level to 3  
Error Level 0 - Fatal  
Error Level 1 - Error  
Error Level 2 - Warning  
Error Level 3  
Receive : 4 (PLIB)
```



	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	0123456789ABCDEF
00	52	65	63	65	69	76	65	20	3A	20	33	20	28	50	4C	49	Receive : 3 (PLI
10	42	29	0D	0A	43	68	61	6E	67	65	20	45	72	72	6F	72	B)..Change Error
20	20	4C	65	76	65	6C	20	74	6F	20	33	0D	0A	45	72	72	Level to 3..Err
30	6F	72	20	4C	65	76	65	6C	20	30	20	2D	20	46	61	74	or Level 0 - Fat
40	61	6C	0D	0A	45	72	72	6F	72	20	4C	65	76	65	6C	20	al..Error Level
50	31	20	2D	20	45	72	72	6F	72	0D	0A	45	72	72	6F	72	1 - Error..Error
60	20	4C	65	76	65	6C	20	32	20	2D	20	57	61	72	6E	69	Level 2 - Warni
70	6E	67	0D	0A	45	72	72	6F	72	20	4C	65	76	65	6C	20	ng..Error Level
80	33																

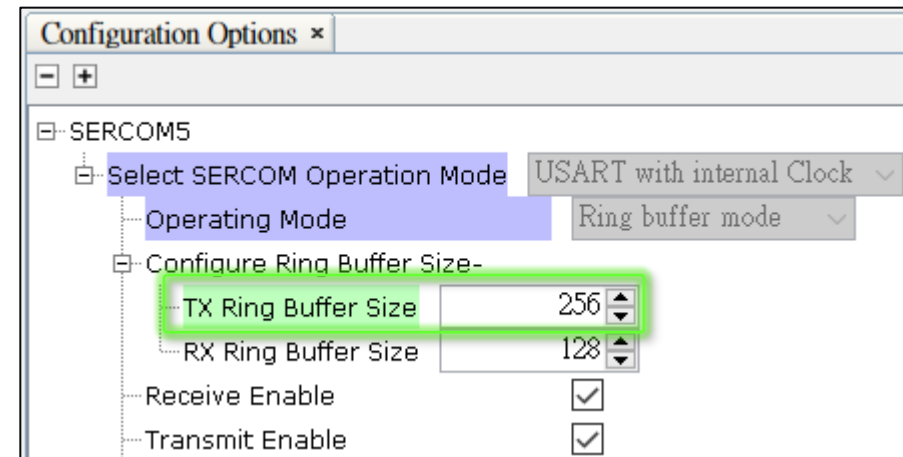
✖ Lab4 Debug Message Control

- Please increase **TX Ring Buffer Size** to **256** or add a wait write finish condition after mass message to output.



```
COM4 - Tera Ter...
File Edit Setup Control Window Help
Receive : 3 (PLIB)
Change Error Level to 3
Error Level 0 - Fatal
Error Level 1 - Error
Error Level 2 - Warning
Error Level 3 - Info
Receive : 4 (PLIB)
Change Error Level to 4
Error Level 0 - Fatal
Error Level 1 - Error
Error Level 2 - Warning
Error Level 3 - Info
Error Level 4 - Debug
```

```
...
SYS_DEBUG_MESSAGE(SYS_ERROR_FATAL, "Error Level 0 - Fatal\r\n");
SYS_DEBUG_MESSAGE(SYS_ERROR_ERROR, "Error Level 1 - Error\r\n");
SYS_DEBUG_MESSAGE(SYS_ERROR_WARNING, "Error Level 2 - Warning\r\n");
SYS_DEBUG_MESSAGE(SYS_ERROR_INFO, "Error Level 3 - Info\r\n");
SYS_DEBUG_MESSAGE(SYS_ERROR_DEBUG, "Error Level 4 - Debug\r\n");
while( SYS_CONSOLE_WriteCountGet(SYS_CONSOLE_DEFAULT_INSTANCE) );
```



⚙️ Lab4 Debug Message Control

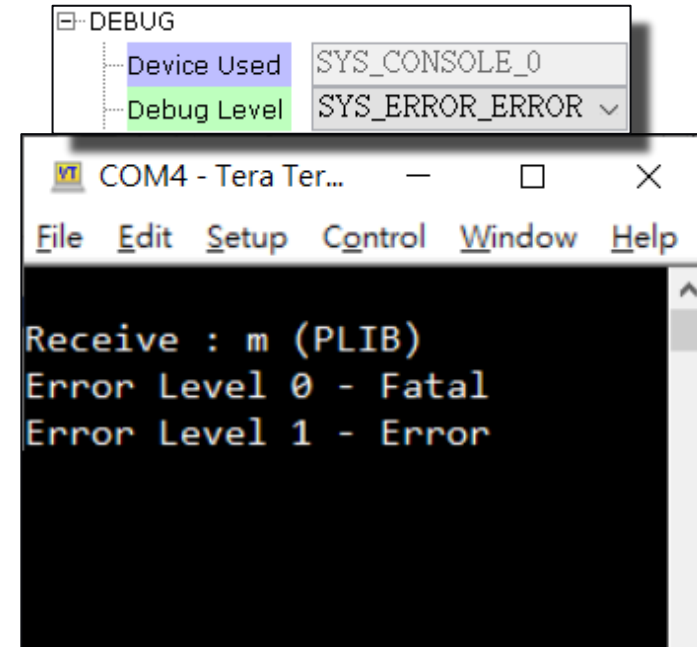
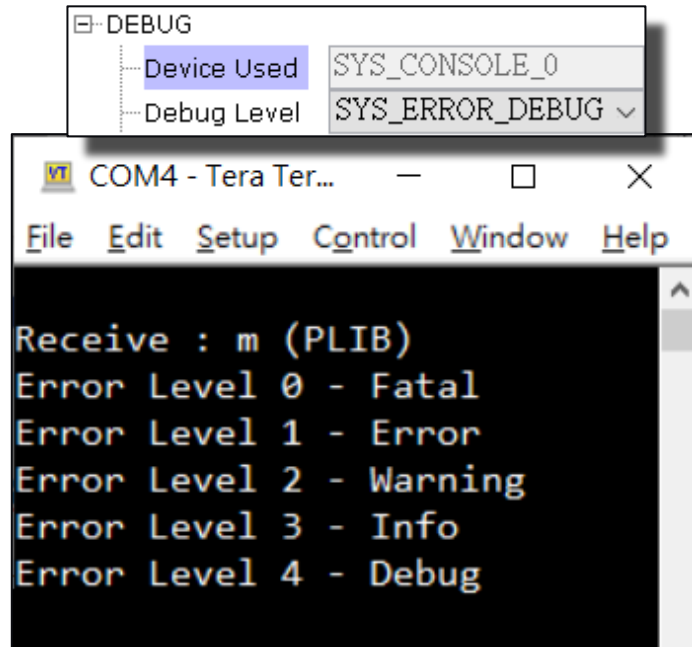
Discuss 1

- What's the effect if insert a `SYS_DEBUG_BreakPoint()` while **debugging** ?

```
if(UartReadBuf[0]>='0' && UartReadBuf[0]<='4')
{
    int ErrorLevel = UartReadBuf[0]-'0';
    SYS_DEBUG_BreakPoint();
    SYS_DEBUG_ErrorLevelSet(ErrorLevel);
}
```

```
56 if(UartReadBuf[0]>='0' && Ua
57 {
58     int ErrorLevel = UartRea
60     SYS_DEBUG_BreakPoint();
    SYS_DEBUG_ErrorLevelSet(
```

- What's the effect if change **Debug Level** configure in MHC ?



COMMAND Processor System Service

Harmony System Service

COMMAND Processor System Service

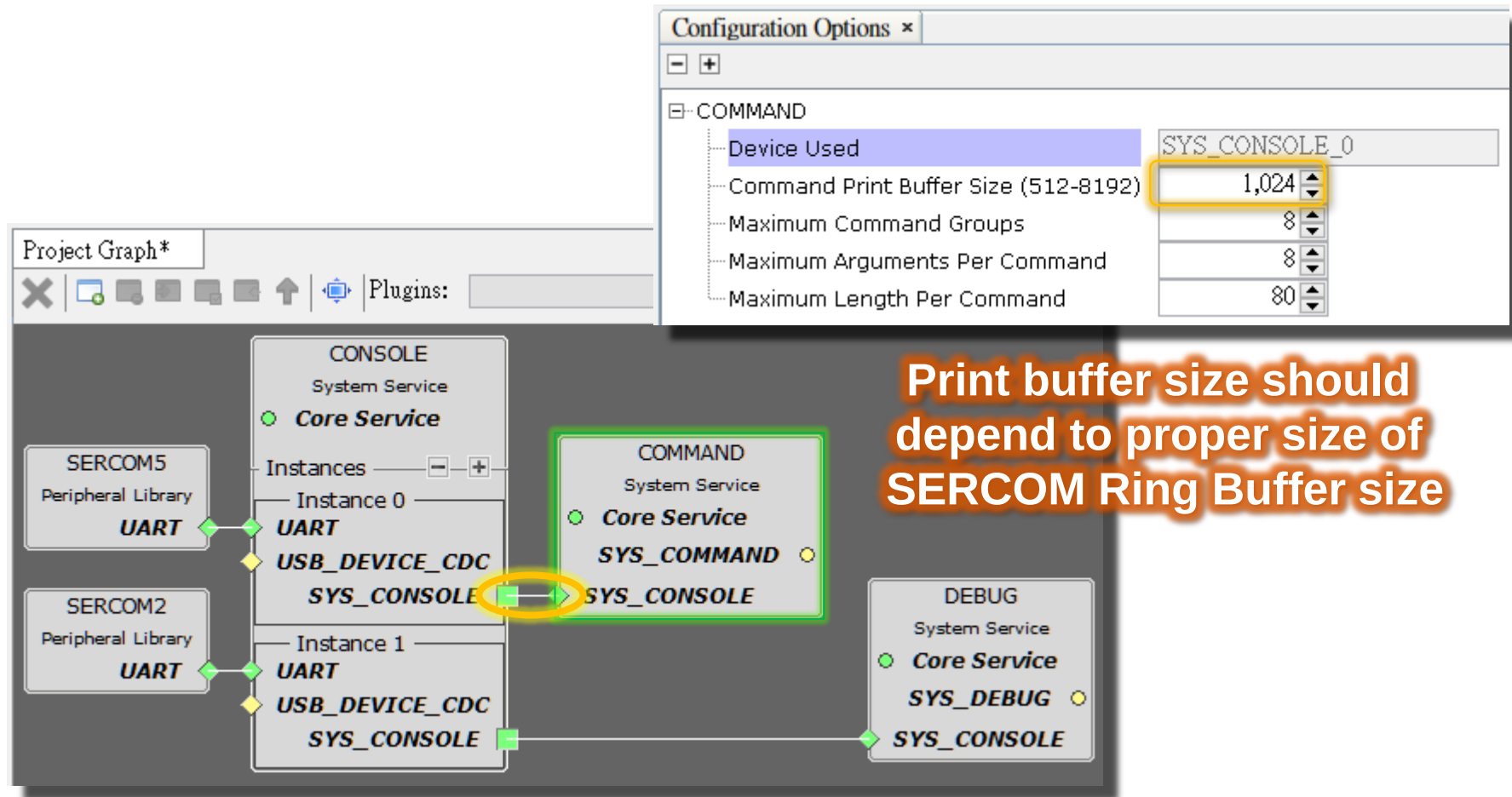
- Introduction

This library provides an abstraction of the Command Processor System Service Library with a convenient C language interface. It provides the framework for a command console that can support commands from multiple client sources.

The Command Processor System Service provides the developer with simple APIs to implement a command console.

The console may support commands from one or more client software modules.

COMMAND Processor System Service



Print buffer size should depend to proper size of SERCOM Ring Buffer size

COMMAND System Service must depend to CONSOLE system Service

COMMAND Processor System Service

API functions of the Command Processor System Service library.

- System Interaction Functions

SYS_CMD_ADDGRP

Allows clients to add command process to the Command Processor System Service.

SYS_CMD_DELETE

Removes CMDIO parameters from the Command Processor System Service console.

SYS_CMD_MESSAGE

Outputs a message to the Command Processor System Service console.

SYS_CMD_PRINT

Outputs a printout to the Command Processor System Service console.

SYS_CMD_READY_TO_READ

Allows upper layer application to confirm that the command module is ready to accept command input

SYS_CMD_READY_TO_WRITE

Allows upper layer application to confirm that the command module is ready to write output to the Console System Service.

SYS_CMD_Tasks

Maintains the Command Processor System Service's internal state machine.

SYS_CMD_Initialize

Initializes data for the instance of the Command Processor module.

- Command I/O Functions

SYS_CMDIO_GET_HANDLE

Gets the CMDIO Handle base via index.

- Data Types and Constants

COMMAND_HISTORY_DEPTH

Maximum Depth of Command History. (Up to 4 defined in sys_command.h)

MAX_CMD_ARGS

Maximum Number of Argument definitions. (MHC configurable, default is 8 and defined in sys_command.h)

MAX_CMD_GROUP

Maximum Number of Command Group definitions. (MHC configurable, default is 8 and defined in sys_command.h)

SYS_CMD_MAX_LENGTH

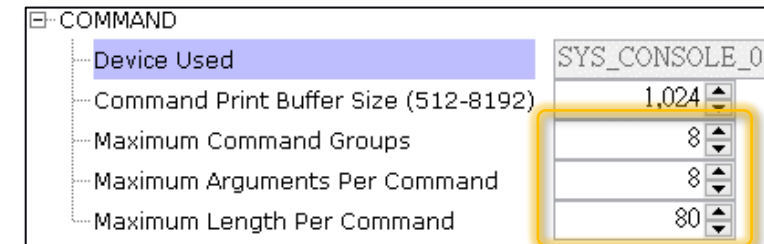
Command Buffer Maximum Length definition. (MHC configurable, Up to 80 bytes defined in sys_command.h)

SYS_CMD_READ_BUFFER_SIZE

Read Buffer Size definition. (Up to 128 bytes defined in sys_command.h)

SYS_CMD_DEVICE_NODE

Defines the data structure to store each command instance.



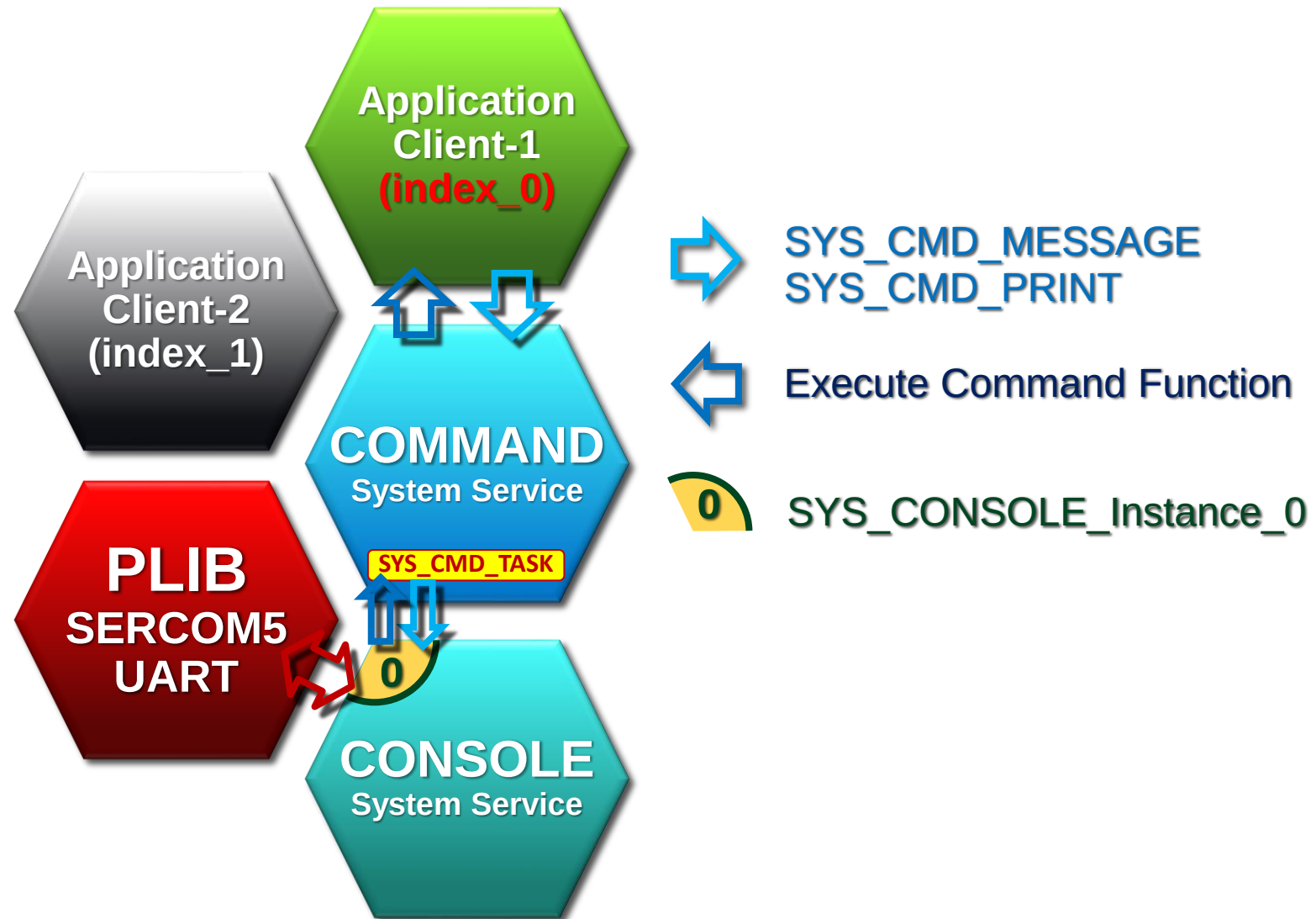
COMMAND Processor System Service

API functions of the Command Processor System Service library. (cont.)

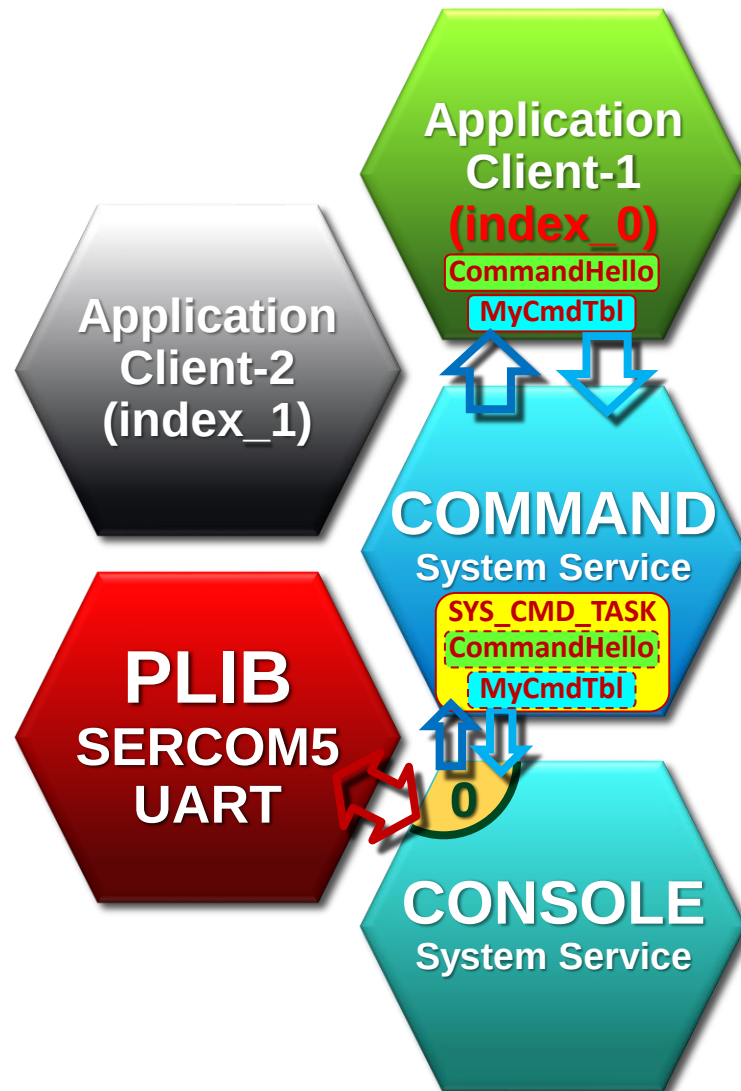
- Data Types and Constants (cont.)

<code>SYS_CMD_DEVICE_NODE</code>	Defines the data structure to store each command instance.
<code>SYS_CMD_API</code>	Identifies the Command API structure.
<code>SYS_CMD_DATA_RDY_FNC</code>	Identifies a data available function API.
<code>SYS_CMD_DESCRIPTOR</code>	A simple command descriptor
<code>SYS_CMD_FNC</code>	Identifies the command process function API.
<code>SYS_CMD_GETC_FNC</code>	Identifies a get single character function API.
<code>SYS_CMD_HANDLE</code>	Identifies a particular Command I/O instance.
<code>SYS_CMD_MSG_FNC</code>	Identifies a message function API.
<code>SYS_CMD_PRINT_FNC</code>	Identifies a print function API.
<code>SYS_CMD_PUTC_FNC</code>	Identifies a single character print function API.
<code>SYS_CMD_READC_FNC</code>	Identifies a read single character function API.
<code>SYS_CMD_CallbackFunction</code>	Command Callback Function Handle.
<code>SYS_CMD_CONSOLE_IO_PARAM</code>	Defines whether the command module is single character or full command read
<code>SYS_CMD_EVENT</code>	Identifies the Command Event Types
<code>SYS_CMD_INIT</code>	Identifies the system command initialize structure.
<code>SYS_CMD_DEVICE_MAX_INSTANCES</code>	This is macro <code>SYS_CMD_DEVICE_MAX_INSTANCES</code> .
<code>SYS_CMD_MESSAGE</code>	This is macro <code>SYS_CMD_MESSAGE</code> .
<code>SYS_CMD_PRINT</code>	This is macro <code>SYS_CMD_PRINT</code> .
<code>SYS_CMD_BUFFER_DMA_READY</code>	Define this for PIC32MX / PIC32MZ
<code>SYS_CMDIO_ADD</code>	Adds CMDIO parameters to the Command Processor System Service console.

COMMAND Processor System Service



COMMAND Processor System Service



```
void CommandHello(SYS_CMD_DEVICE_NODE* pCmdIO,
                  int argc, char** argv)
{
    SYS_CMD_MESSAGE("\r\n- Hello !");
    SYS_CMD_PRINT("\r\n-parameter count : %d\r\n", argc);
}

static const SYS_CMD_DESCRIPTOR MyCmdTbl[] =
{
    { "hello", CommandHello, ": Hello!" },
};

void App1_Task()
{
    SYS_CMD_ADDGRP(MyCmdTbl, 1, "MYCMD", ": My Commands");
}
```

VT COM24 - Tera Term VT

```
>hello
- Hello !
- parameter count : 1
```

```
>help
----- Supported command groups -----
*** MYCMD: My Commands ***
```

```
>help MYCMD
*** hello: Hello! ***
>
```

Lab 5

Custom Command Console

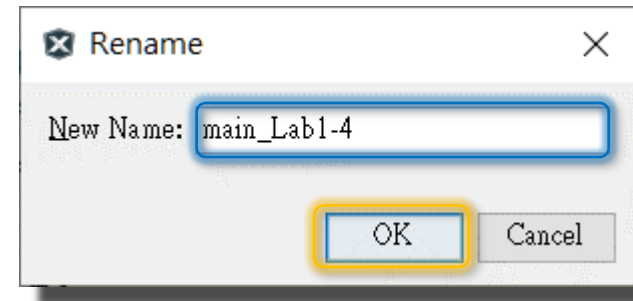
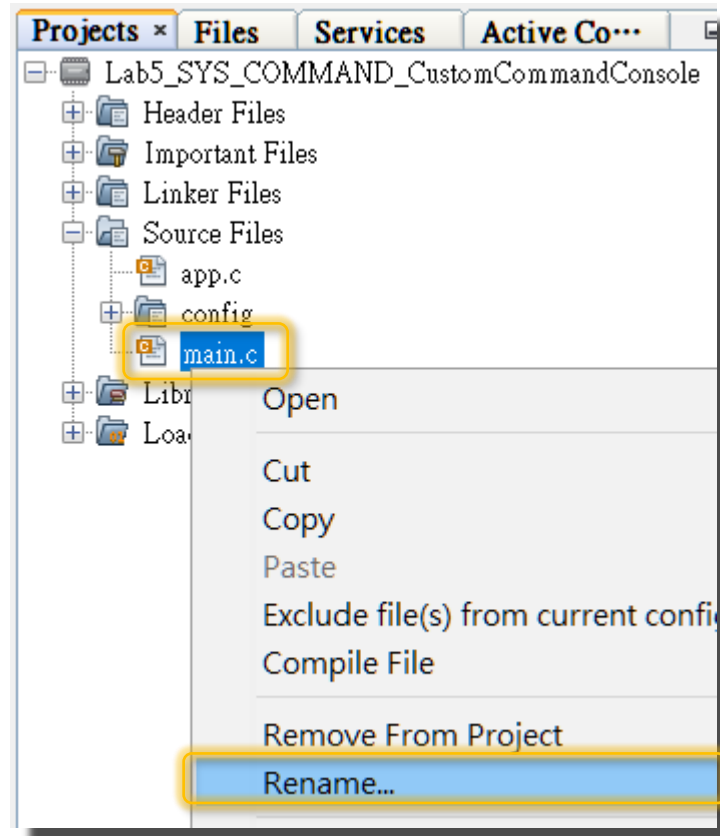
COMMAND Processor System Service

Lab5 Custom Command Console

- Try add **COMMAND processor system service** and communicate through UART to use previous added **CONSOLE System Service**.
- In this Lab, you might create a new **main.c** to continue.
- Build your custom command to have below activity :
 - hello : Print "Hello world !" on console.
 - led # on : Light LED #.
 - led # off : Dark LED #.
 - errlv # : Change Ddebug Error Level to # and output message "Change Error Level to #" then control five different Error Level debug message output as below.
 - "Error Level 0 - Fatal"
 - "Error Level 1 - Error"
 - "Error Level 2 - Warning"
 - "Error Level 3 - Info"
 - "Error Level 4 - Debug"

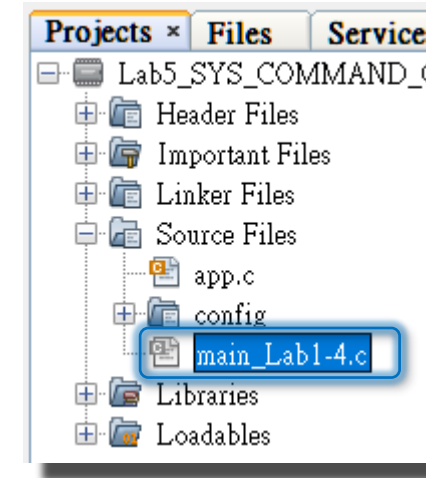
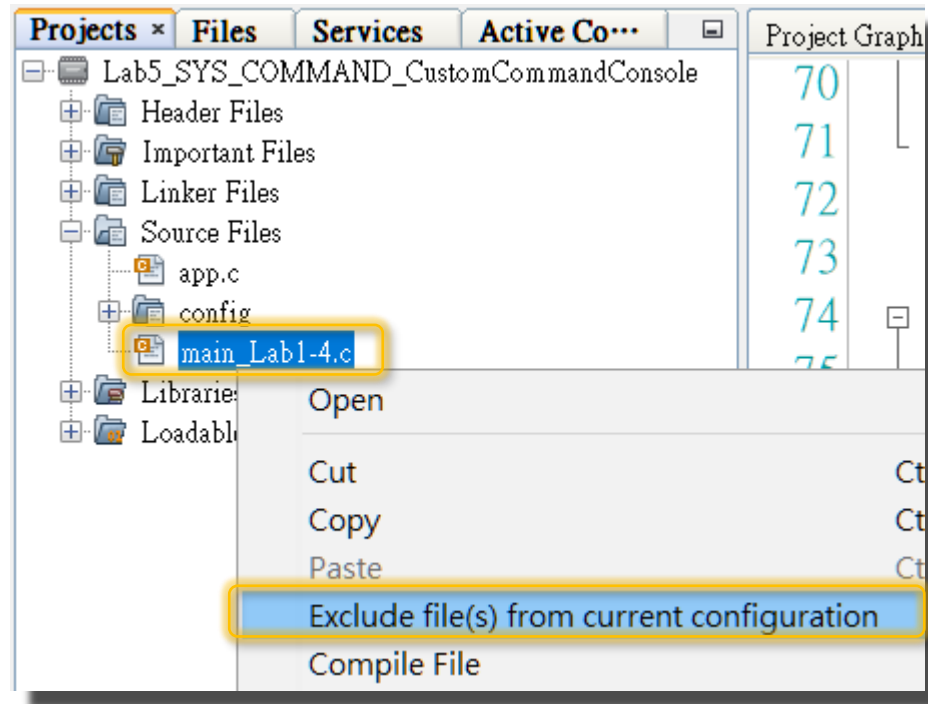
⚙️ Lab5 Custom Command Console

- Right click on main.c and rename it to **main_Lab1-4**



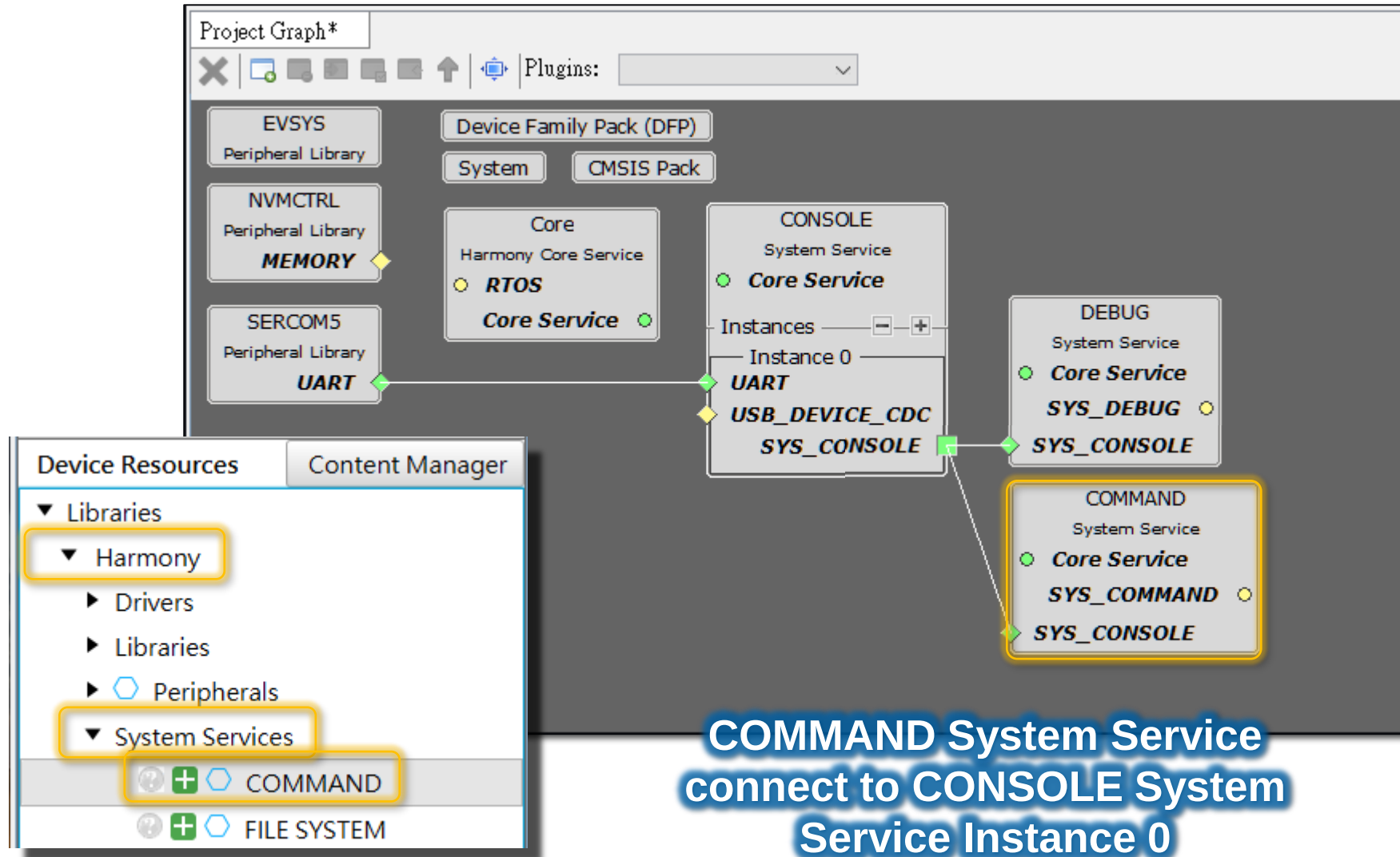
✖ Lab5 Custom Command Console

- Right click on `main_Lab1-4.c` and exclude it from current project
- The **new main.c** will **auto generate** while next [Generate] Code to be executed.



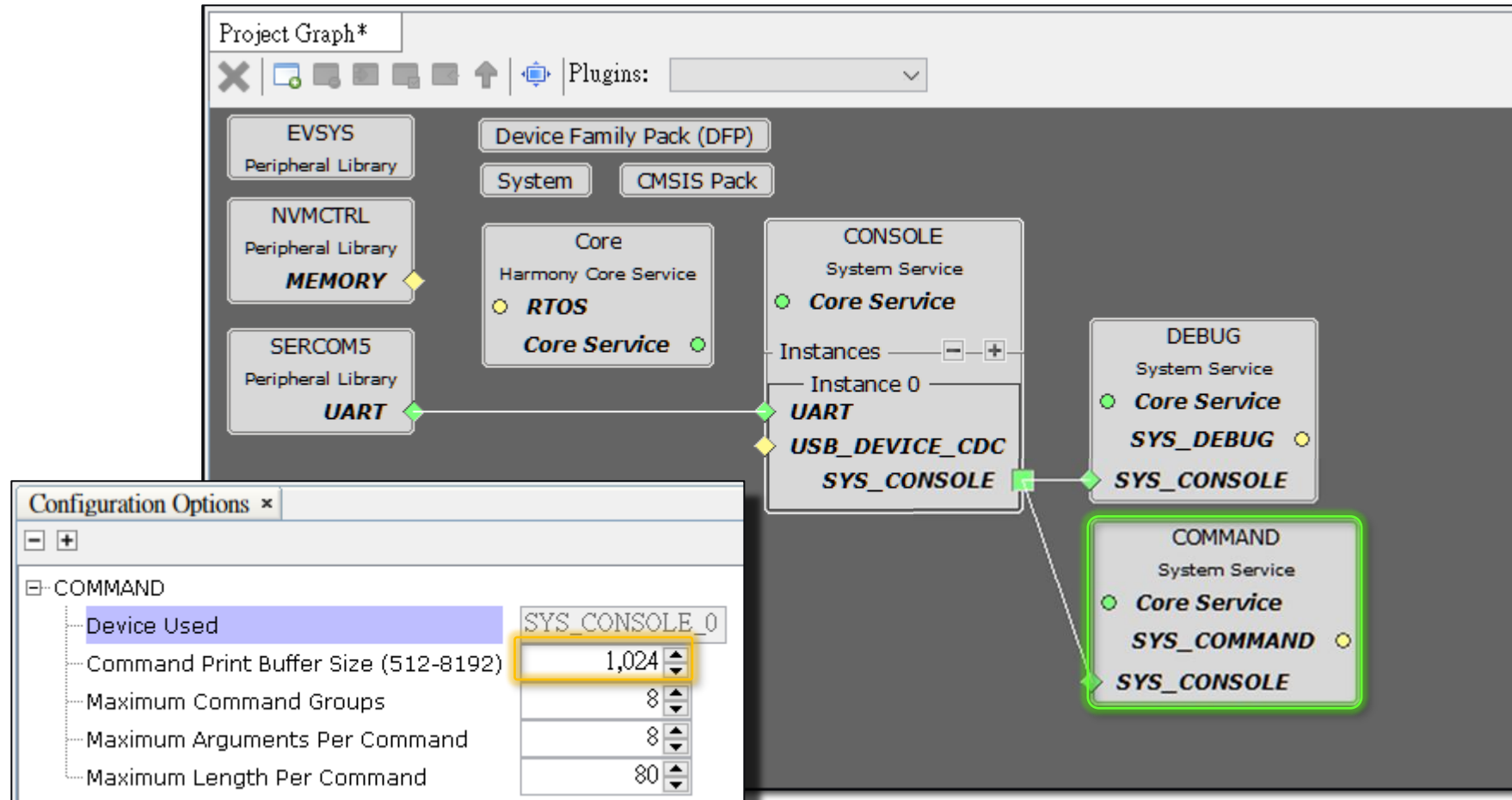
⚙️ Lab5 Custom Command Console

- Harmony Configuration (Add COMMAND System Service)



⚙️ Lab5 Custom Command Console

- Harmony Configuration (COMMAND System Service config)



Print buffer size could Config in MHC, but it should increase proper size of SERCOM Ring Buffer simultaneously

Lab5 Custom Command Console

- Modify your **new** main.c as below

```
#include <stddef.h>
#include <stdbool.h>
#include <stdlib.h>
#include "definitions.h"
#include <stdio.h>
#include <string.h>

void CommandHello(SYS_CMD_DEVICE_NODE* pCmdIO, int argc, char** argv)
{
    SYS_CMD_MESSAGE("Hello world !\r\n");
}

static const SYS_CMD_DESCRIPTOR MyCmdTbl[]=
{
    { "hello", CommandHello, ": Say Hello" },
};

int main( void )
{
    SYS_Initialize(NULL);

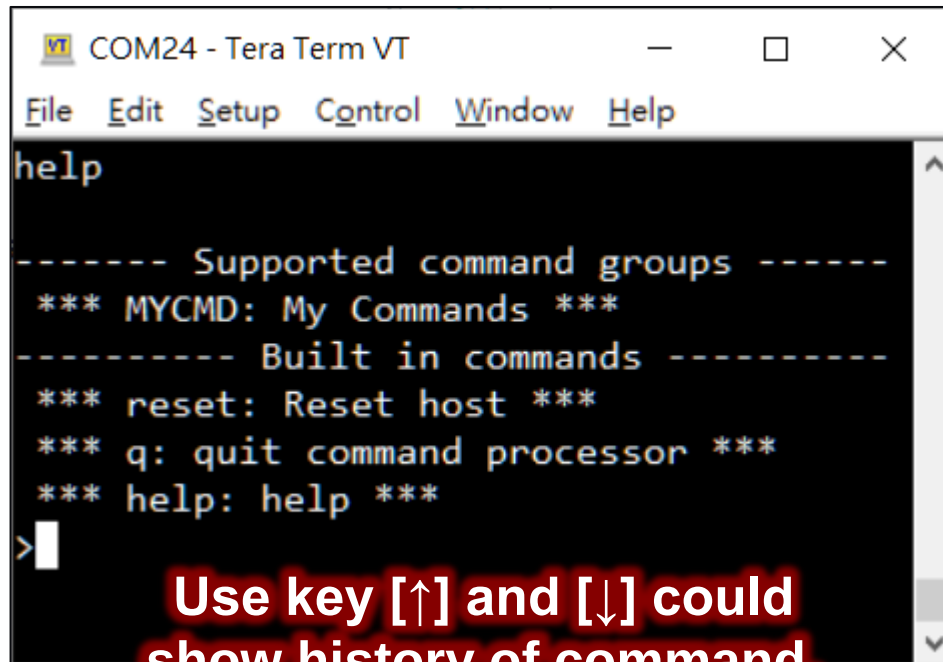
    SYS_CMD_ADDGRP(MyCmdTbl, 1, "MYCMD", ": My Commands");

    while( true )
    {
        SYS_Tasks();
    }

    return ( EXIT_FAILURE);
}
```

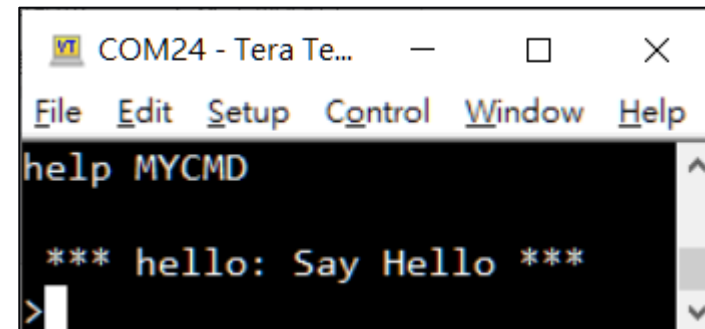
⚙️ Lab5 Custom Command Console

- Compiler and programming your project to EVB.
- Type 'help' + [Enter] will show system command help.
- Type 'help MYCMD' + [Enter] will show custom MYCMD help.
- Type 'hello' + [Enter] will execute hello command function.

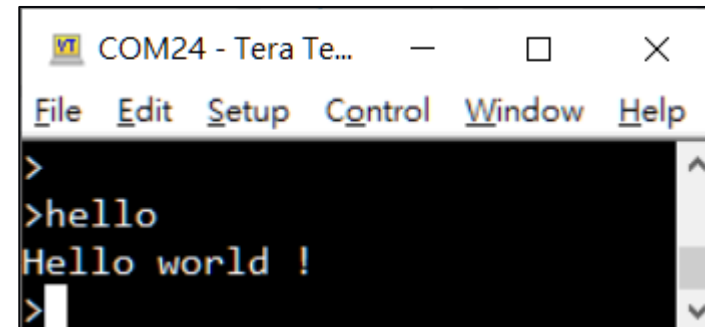


```
COM24 - Tera Term VT
File Edit Setup Control Window Help
help
----- Supported command groups -----
*** MYCMD: My Commands ***
----- Built in commands -----
*** reset: Reset host ***
*** q: quit command processor ***
*** help: help ***
>
```

Use key [↑] and [↓] could show history of command.



```
COM24 - Tera Te...
File Edit Setup Control Window Help
help MYCMD
*** hello: Say Hello ***
>
```



```
COM24 - Tera Te...
File Edit Setup Control Window Help
>
>hello
Hello world !
>
```

⚙️ Lab5 Custom Command Console

- Modify your main.c as below

```
void CommandHello(SYS_CMD_DEVICE_NODE* pCmdIO, int argc, char** argv) { ... }

void CommandLED(SYS_CMD_DEVICE_NODE* pCmdIO, int argc, char** argv)
{
    if(argc == 3)
    {
        int Led, OnOff = 0;

        if(!strcmp( argv[2], "on" ))
            OnOff = 1;

        Led = atoi(argv[1]);
        switch(Led)
        {
            case 1: (OnOff ? LED1_Set() : LED1_Clear()); break;
            case 2: (OnOff ? LED2_Set() : LED2_Clear()); break;
            case 3: (OnOff ? LED3_Set() : LED3_Clear()); break;
        }
    }
}

static const SYS_CMD_DESCRIPTOR MyCmdTbl[]=
{
    { "hello", CommandHello, ": Say Hello" },
    { "led",    CommandLED,    ": led # on[off]" },
};

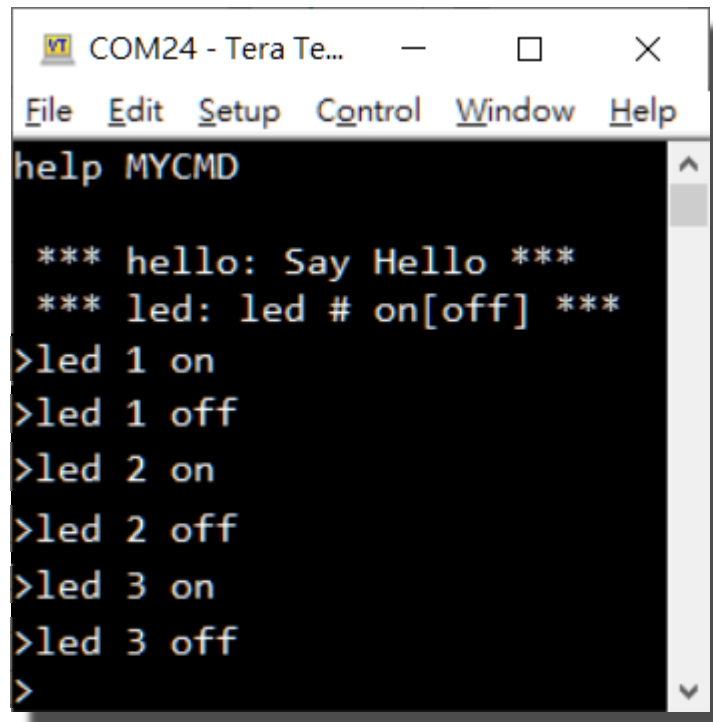
int main( void )
{
    SYS_Initialize(NULL);

    SYS_CMD_ADDGRP(MyCmdTbl, 2, "MYCMD", ": My Commands");
```

Modify this line

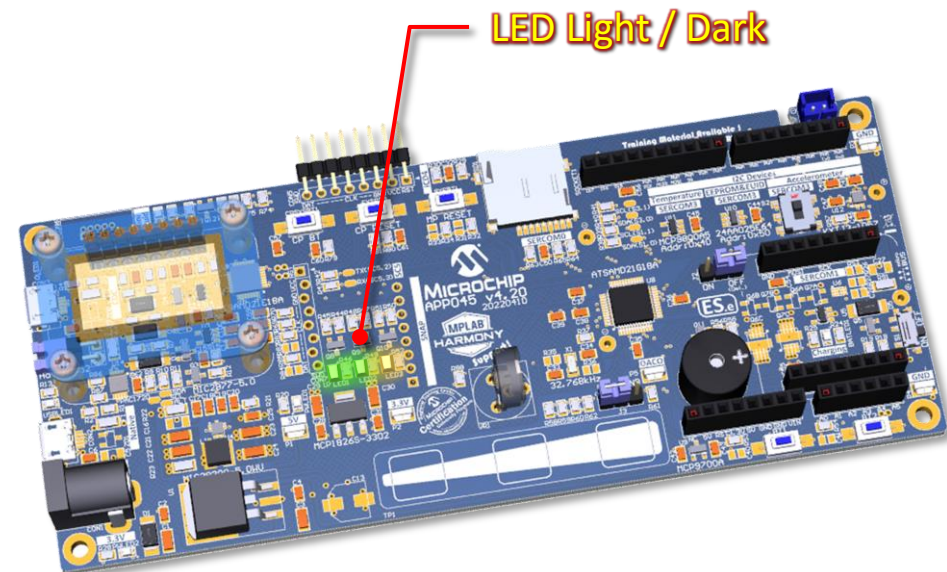
⚙️ Lab5 Custom Command Console

- Compiler and programming your project to EVB.
- Type 'help MYCMD' + [Enter] will show custom MYCMD help.
- Type 'led 1 on' + [Enter] to light **LED1**, 'led 1 off' to dark **LED1**.
- Same control command to **LED2** and **LED3**.



```
COM24 - Tera Te...  
File Edit Setup Control Window Help  
help MYCMD  
  
*** hello: Say Hello ***  
*** led: led # on[off] ***  
>led 1 on  
>led 1 off  
>led 2 on  
>led 2 off  
>led 3 on  
>led 3 off  
>
```

Use command to control LED.



⚙️ Lab5 Custom Command Console

- Modify your main.c as below

```
void CommandHello(SYS_CMD_DEVICE_NODE* pCmdIO, int argc, char** argv) { ... }
void CommandLED(SYS_CMD_DEVICE_NODE* pCmdIO, int argc, char** argv) { ... }
void CommandErrLv(SYS_CMD_DEVICE_NODE* pCmdIO, int argc, char** argv)
{
    if(argc == 2)
    {
        int ErrorLevel = atoi(argv[1]);
        if(ErrorLevel >= 0 && ErrorLevel <= 4 )
        {
            SYS_DEBUG_ErrorLevelSet(ErrorLevel);
            SYS_CMD_PRINT("Change Error Level to %d\r\n", SYS_DEBUG_ErrorLevelGet());
        }
    }
    SYS_DEBUG_MESSAGE(SYS_ERROR_FATAL, "Error Level 0 - Fatal\r\n");
    SYS_DEBUG_MESSAGE(SYS_ERROR_ERROR, "Error Level 1 - Error\r\n");
    SYS_DEBUG_MESSAGE(SYS_ERROR_WARNING, "Error Level 2 - Warning\r\n");
    SYS_DEBUG_MESSAGE(SYS_ERROR_INFO, "Error Level 3 - Info\r\n");
    SYS_DEBUG_MESSAGE(SYS_ERROR_DEBUG, "Error Level 4 - Debug\r\n");
}

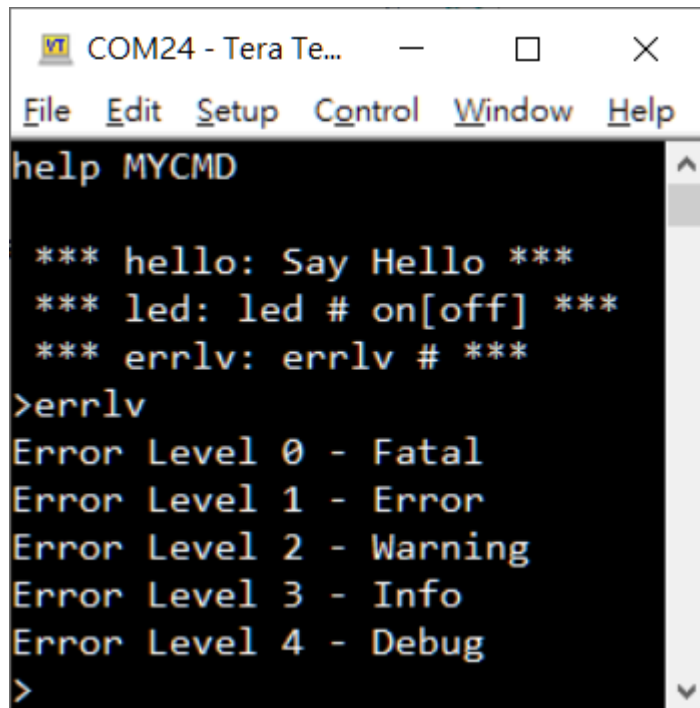
static const SYS_CMD_DESCRIPTOR MyCmdTbl[] =
{
    { "hello", CommandHello, ": Say Hello" },
    { "led", CommandLED, ": led # on[off]" },
    { "errlv", CommandErrLv, ": errlv #" },
};

int main( void )
{
    SYS_Initialize(NULL);
    SYS_CMD_ADDGRP(MyCmdTbl, 3, "MYCMD", ": My Commands");
```

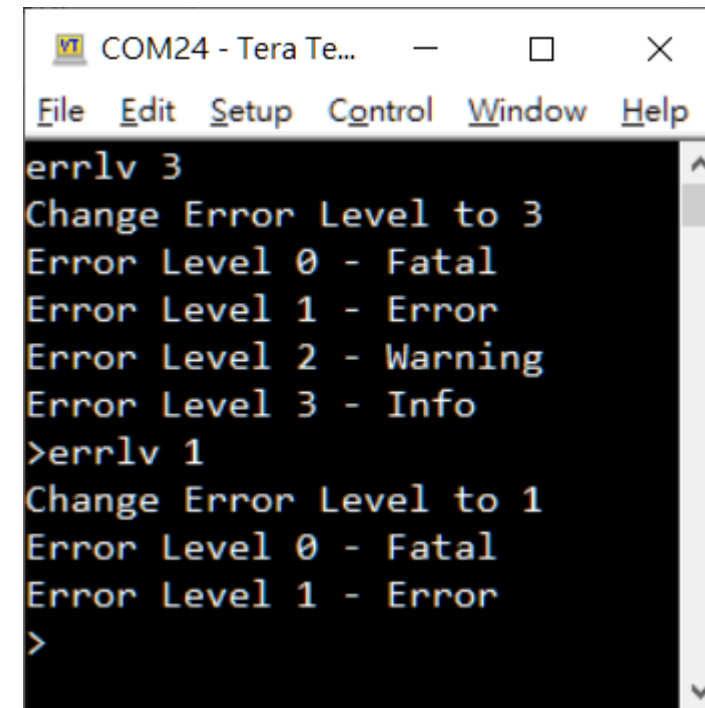
Modify this line

⚙️ Lab5 Custom Command Console

- Compiler and programming your project to EVB.
- Type 'help MYCMD' + [Enter] will show custom MYCMD help.
- Type 'errlv' + [Enter] to show five error level debug messages.
- Type 'errlv 3' + [Enter] change **error level to 3** and show only **error level <=3** debug messages, 'errlv 1' change **error level to 1**.



```
COM24 - Tera Te...  
File Edit Setup Control Window Help  
help MYCMD  
  
*** hello: Say Hello ***  
*** led: led # on[off] ***  
*** errlv: errlv # ***  
>errlv  
Error Level 0 - Fatal  
Error Level 1 - Error  
Error Level 2 - Warning  
Error Level 3 - Info  
Error Level 4 - Debug  
>
```

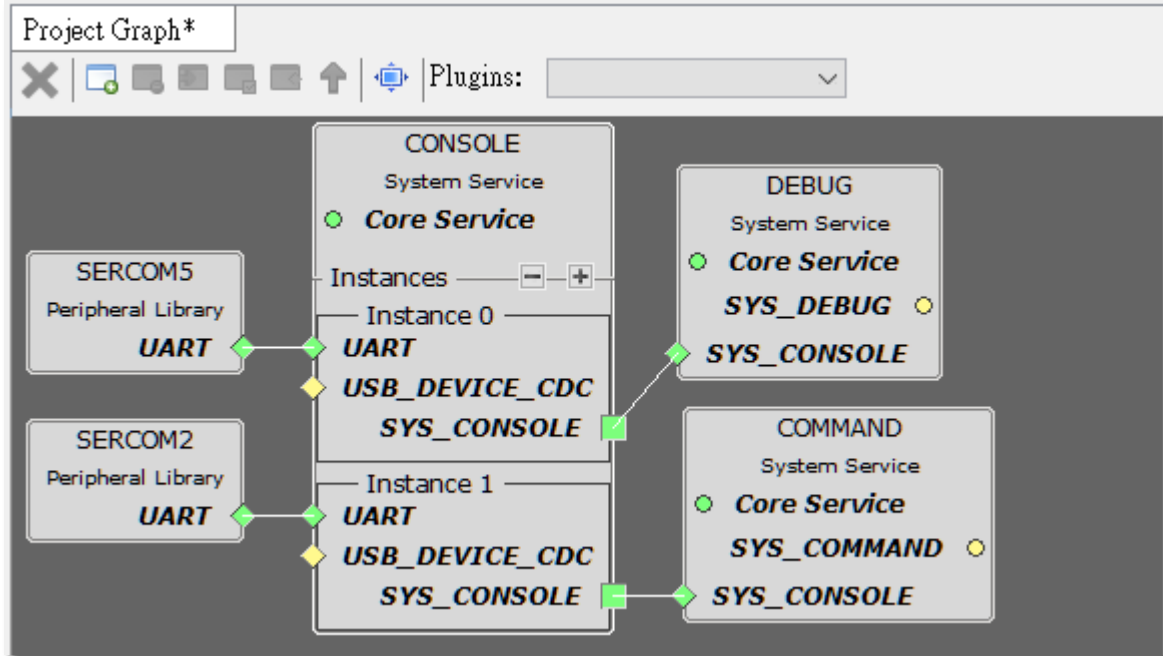


```
COM24 - Tera Te...  
File Edit Setup Control Window Help  
errlv 3  
Change Error Level to 3  
Error Level 0 - Fatal  
Error Level 1 - Error  
Error Level 2 - Warning  
Error Level 3 - Info  
>errlv 1  
Change Error Level to 1  
Error Level 0 - Fatal  
Error Level 1 - Error  
>
```

⚙️ Lab5 Custom Command Console

Discuss 1

- Try implement two Instance of CONSOLE System Service to separate DEBUG and COMMAND console.



```
COM24 - Tera Te...  
File Edit Setup Control Window Help  
DEBUG console here!  
Error Level 0 - Fatal  
Error Level 1 - Error  
Error Level 2 - Warning  
Error Level 3 - Info
```

```
COM30 - Tera Te...  
File Edit Setup Control Window Help  
COMMAND console here!  
  
>led 1 on  
>errlv 3  
Change Error Level to 3  
>
```


USB Device CDC Library

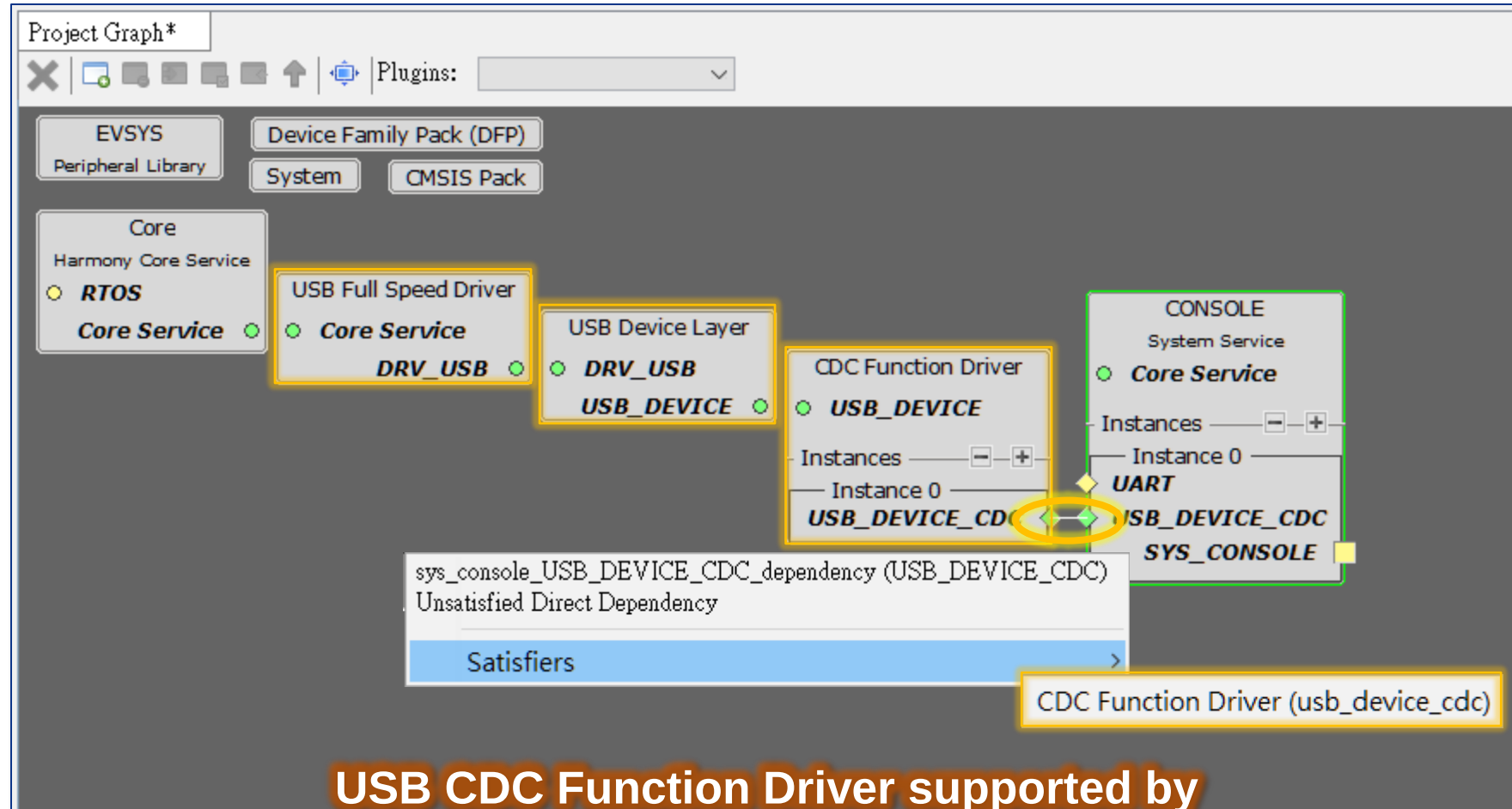
Harmony Device Library

USB Device CDC Library

- **Introduction**

The MPLAB Harmony USB Communications Device Class (CDC) Device Library (also referred to as the CDC function driver or library) provides functions and methods that allow application designers to implement a USB Device CDC. The current version of the library supports the Abstract Control Model (ACM) of the CDC specification revision 1.2 and specifically implements a subset of the AT250 command set. This library must be used in conjunction with the MPLAB Harmony USB Device Layer.

USB Device CDC Library



USB CDC Function Driver supported by CONSOLE System Service

USB Device CDC Library

API functions of the USB Device CDC library.

Functions

USB_DEVICE_CDC_EventHandlerSet

USB_DEVICE_CDC_Read

USB_DEVICE_CDC_Write

USB_DEVICE_CDC_SerialStateNotificationSend

This function registers an event handler for the specified CDC driver instance.

This function requests a data read from the USB Device CDC Function Driver Layer.

This function requests a data write to the USB Device CDC Function Driver Layer.

This function schedules a request to send serial state notification to the host.

Data Types and Constants

USB_DEVICE_CDC_EVENT_DATA_READ_COMPLETE

USB_DEVICE_CDC_EVENT_DATA_SERIAL_STATE_NOTIFICATION_COMPLETE USB Device CDC Function Driver Read and Write Complete Event Data.

USB_DEVICE_CDC_EVENT_DATA_WRITE_COMPLETE

USB Device CDC Function Driver Read and Write Complete Event Data.

USB_DEVICE_CDC_INIT

USB Device CDC Function Driver Initialization Data Structure

USB_DEVICE_CDC_EVENT_DATA_SEND_BREAK

USB Device CDC Function Driver Send Break Event Data

USB_DEVICE_CDC_EVENT

USB Device CDC Function Driver Events

USB_DEVICE_CDC_EVENT_HANDLER

USB Device CDC Event Handler Function Pointer Type.

USB_DEVICE_CDC_EVENT_RESPONSE

USB Device CDC Function Driver Event Callback Response Type

USB_DEVICE_CDC_INDEX

USB Device CDC Function Driver Index

USB_DEVICE_CDC_RESULT

USB Device CDC Function Driver USB Device CDC Result enumeration.

USB_DEVICE_CDC_TRANSFER_FLAGS

USB Device CDC Function Driver Transfer Flags

USB_DEVICE_CDC_TRANSFER_HANDLE

USB Device CDC Function Driver Transfer Handle Definition.

USB_DEVICE_CDC_EVENT_RESPONSE_NONE

USB Device CDC Function Driver Event Handler Response Type None.

USB_DEVICE_CDC_TRANSFER_HANDLE_INVALID

USB Device CDC Function Driver Invalid Transfer Handle Definition.

USB_DEVICE_CDC_FUNCTION_DRIVER

USB Device CDC Function Driver Function pointer

USB Device CDC Library

API functions of the USB Device CDC library. (cont.)

Data Types and Constants

USB_CDC_LINESTATE_CARRIER	Bit code information in line state
USB_CDC_LINESTATE_DTR	Bit code information in line state
USB_CDC_DESCRIPTOR_TYPE	Identifies the descriptor types in the CDC.
USB_CDC_INF_PROTOCOL	Identifies the protocol codes.
USB_CDC_INTERFACE_TYPE	Identifies the CDC interface type.
USB_CDC_NOTIFICATION	Identifies the notification codes available for CDC.
USB_CDC_REQUEST	Identifies the CDC specific request codes.
USB_CDC_FUNCTIONAL_DESCRIPTOR	Identifies the CDC function header type.
USB_CDC_SUBCLASS	Identifies the subclass codes for communication interface.
USB_CDC_UNION_FUNCTIONAL_DESCRIPTOR_SUBORDINATE	This is type USB_CDC_UNION_FUNCTIONAL_DESCRIPTOR_SUBORDINATE.
CS_INTERFACE	This is a macro.
USB_CDC_ACM_SUPPORT_BREAK	This is a macro.
USB_CDC_ACM_SUPPORT_COMM_FEATURE	This is a macro.
USB_CDC_ACM_SUPPORT_LINE_CODING_LINE_STATE_AND_NOTIFICATION	This is a macro.
USB_CDC_ACM_SUPPORT_NETWORK_NOTIFICATION	This is a macro.
USB_CDC_ACM_SUPPORT_NONE	Identifies the CDC ACM sub-class capabilities.
USB_CDC_CLASS_CODE	Identifies the CDC Interface Class, Subclass and protocol constants.
USB_CDC_COMMUNICATIONS_INTERFACE_CLASS_CODE	This is a macro.
USB_CDC_DATA_INTERFACE_CLASS_CODE	This is a macro.
USB_CDC_DATA_INTERFACE_PROTOCOL	This is a macro.
USB_CDC_DATA_INTERFACE_SUBCLASS_CODE	This is a macro.

USB Device CDC Library

API functions of the USB Device CDC library. (cont.)

Data Types and Constants

USB_CDC_REQUEST_CLASS_SPECIFIC

USB_CDC_SUBCLASS_CODE

USB_DEVICE_CDC_INDEX_0

USB_DEVICE_CDC_INDEX_1

USB_DEVICE_CDC_INDEX_2

USB_DEVICE_CDC_INDEX_3

USB_DEVICE_CDC_INDEX_4

USB_DEVICE_CDC_INDEX_5

USB_DEVICE_CDC_INDEX_6

USB_DEVICE_CDC_INDEX_7

CDC specific request

This is a macro.

Use this to specify CDC Function Driver Instance 0

Use this to specify CDC Function Driver Instance 1

Use this to specify CDC Function Driver Instance 2

Use this to specify CDC Function Driver Instance 3

Use this to specify CDC Function Driver Instance 4

Use this to specify CDC Function Driver Instance 5

Use this to specify CDC Function Driver Instance 6

Use this to specify CDC Function Driver Instance 7

Lab 6

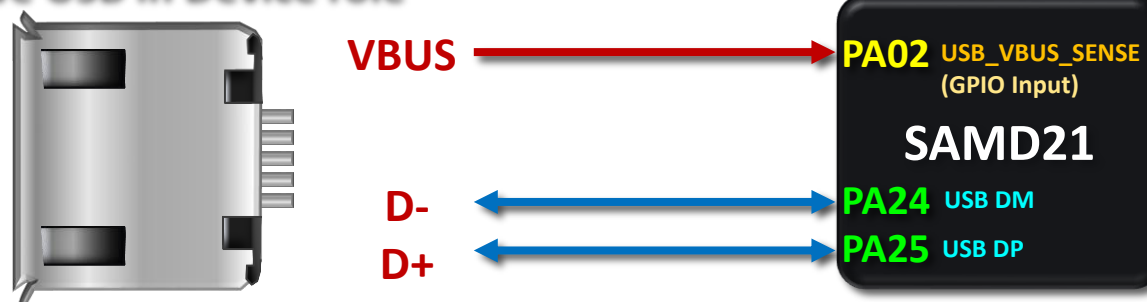
USB Device CDC with CONSOLE System Service

USB Device Library

✖ Lab6 USB Device CDC with CONSOLE System Service

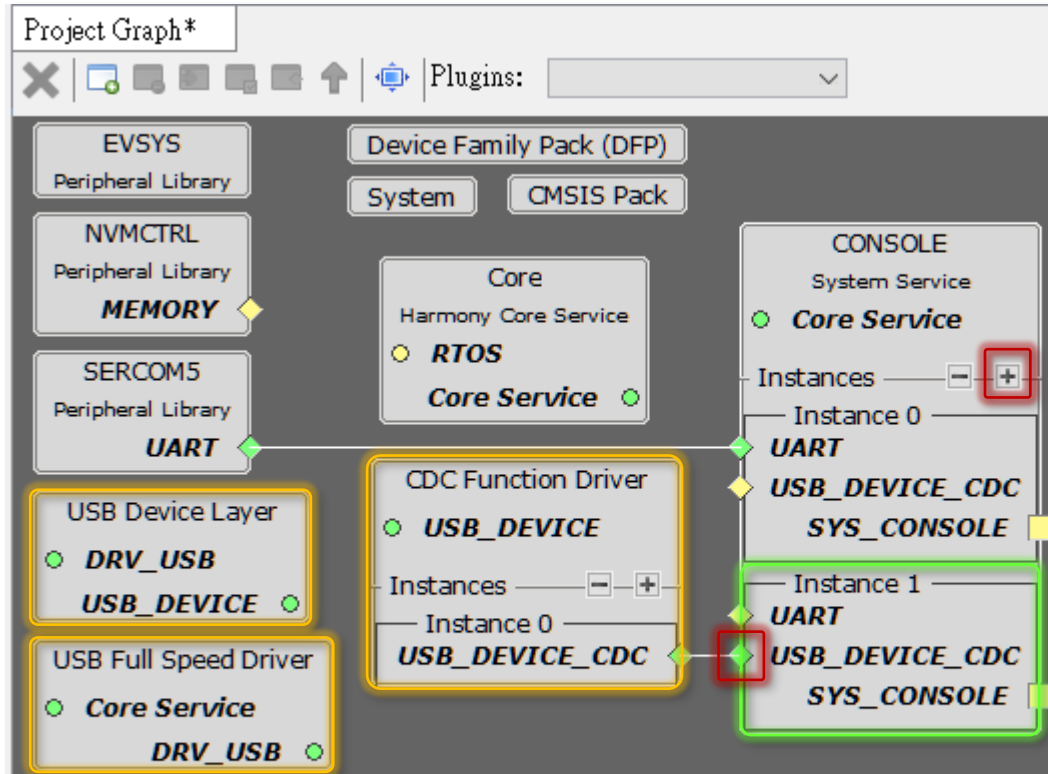
- This Lab related to **Lab3** of **CONSOLE System Service**, please start from the solution project **Lab3_SYS_CONSOLE_UartConsole.zip**
- Add another **Instance 1** of **CONSOLE System Service**.
- Connect **USB Device CDC** to **Instance 1** of **CONSOLE System Service**, the **Device Layer** and **USB Full Speed Driver** will notice to add together.
- Config **PA24/PA25** as **USB DM/DP** pins.
- Config **PA02** as **GPIO input** and custom name to **USB_VBUS_SENSE**.
- Output “**Hello World! (PLIB)**” and “**Hello World! (USB)**” to two consoles.
- Receive console input **X** and print “**Receive : X**” to opposite console.
- “**Receive : X (PLIB)**” (Receive in **PLIB** console and output to **USB** console)
- “**Receive : X (USB)**” (Receive in **USB** console and output to **PLIB** console)

Native USB in Device role



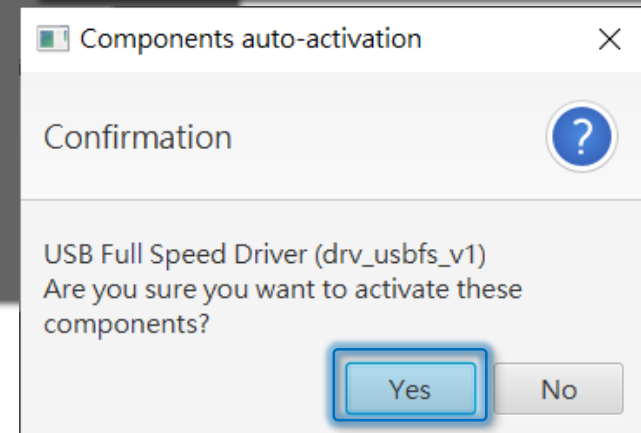
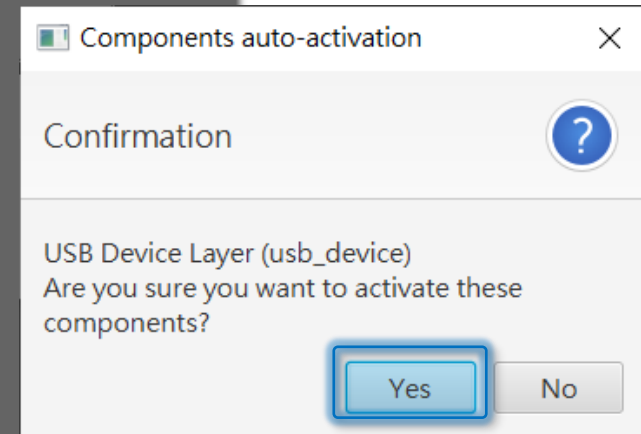
✖ Lab6 USB Device CDC with CONSOLE System Service

- Add USB CDC Function Driver and USB Device modules.



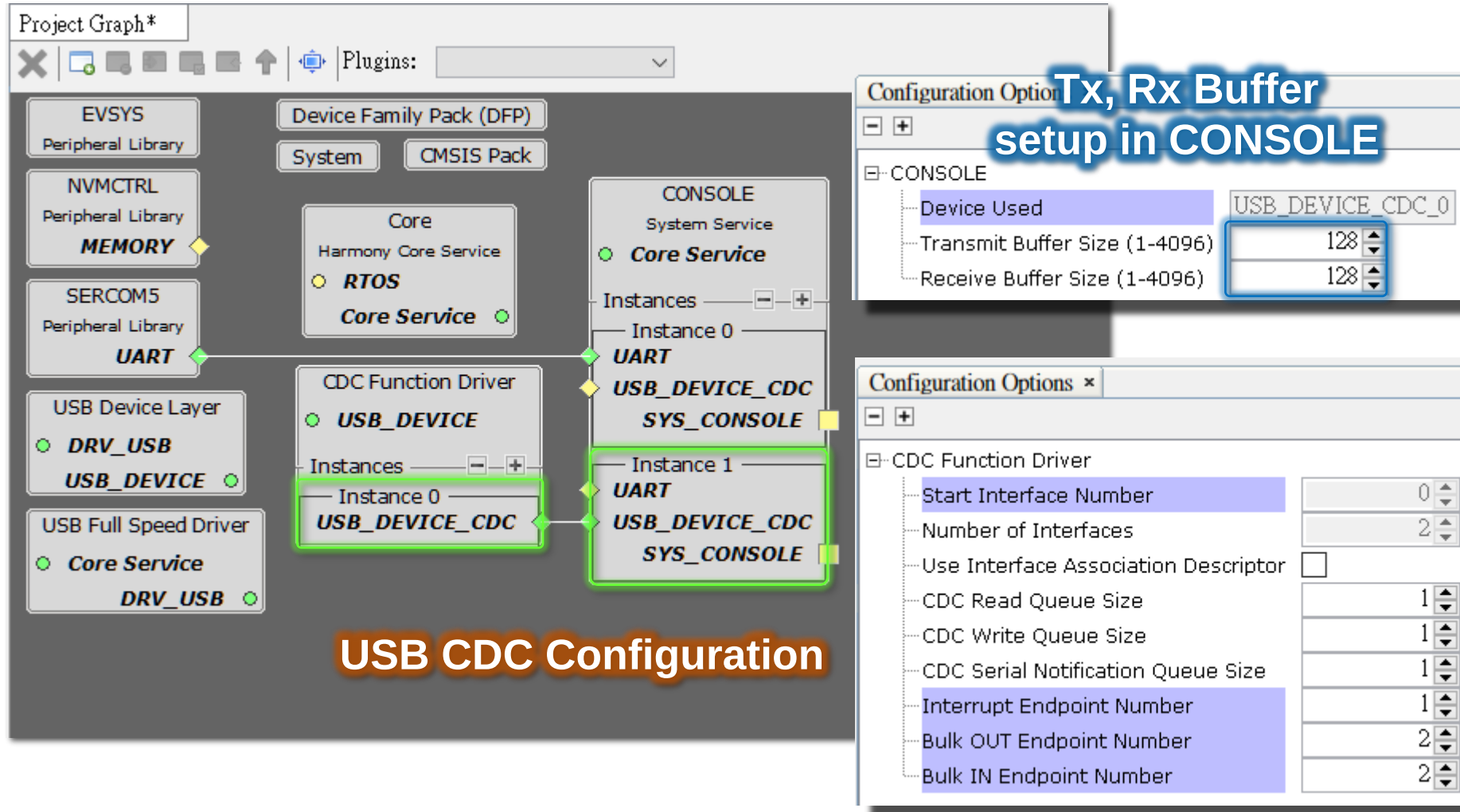
**Must Activation
USB Device Layer
USB Full Speed Driver**

**Connect CDC Function Driver to
Instance1 of CONSOLE System Service**



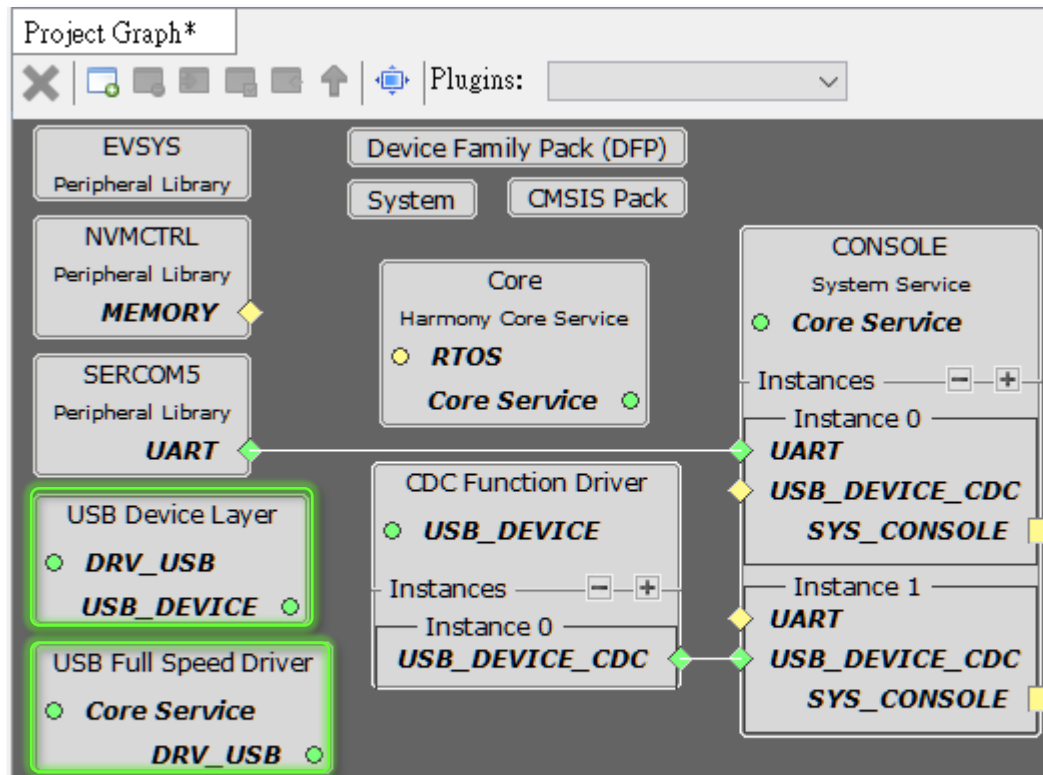
⚙️ Lab6 USB Device CDC with CONSOLE System Service

- Harmony Configuration of CONSOLE and CDC Function Driver.



⚙️ Lab6 USB Device CDC with CONSOLE System Service

- Harmony Configuration of USB Device Layer and Full Speed Driver.



Enable the VBUS Sense feature
Could auto emulate CDC Function
while USB cable to plugged-in

Setup your USB VID / PID
Or use Microchip Demo PID

USB Device Layer

- Number of Functions: 1
- Number of Endpoints: 2
- Special Events: **cdc_com_port_single_demo**
- Special Features: **cdc_com_port_single_demo**
- Endpoint 0 Buffer Size: 64
- Vendor ID: 0x04D8
- Product ID Selection: **cdc_com_port_single_demo**
- Product ID: 0x000A
- Manufacturer String: Microchip Technology Inc.
- Product String Selection: Simple CDC Device Demo
- Add Device Serial Number: ☐

VID:0x04D8 PID:0x000A
Simple CDC Device Demo

USB_VBUS_SENSE GPIO pin
Is necessary for VBUS Sense.

USB Full Speed Driver

- USB Mode Selection: Device
- Enable VBUS Sense: ☒
- VBUS SENSE Pin Name: **USB_VBUS_SENSE**

✖ Lab6 USB Device CDC with CONSOLE System Service

- Setup USB pads D-/D+ to PA24/PA25 and **VBUS Sense GPIO** PA02 to **USB_VBUS_SENSE**

Pin Settings ×									
Order:	Ports ▾	Table View	☒ Easy View						
Pin Number	Pin ID	Custom Name	Function	Mode	Direction	Latch	Pull Up	Pull Down	Drive Strength
3	PA02	USB_VBUS_SENSE	GPIO ▾	Digital	In ▾	Low	☐	☐	NORMAL ▾
33	PA24	USB_DM	USB_DM ▾	Digital	High Impedance ▾	n/a	☐	☐	NORMAL ▾
34	PA25	USB_DP	USB_DP ▾	Digital	High Impedance ▾	n/a	☐	☐	NORMAL ▾

PA02 as GPIO input and name as USB_VBUS_SENSE

Configure PA24 as USB_DM

Configure PA25 as USB_DP

Pin Table ×																											
Package: TQFP48 ▾		3	4	9	10	11	12	13	14	15	16	21	22	23	24	25	26	27	28	29	30	31	32	33	34		
		A																									
Module	Function	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25		
GPIO	GPIO																										
USB	USB_DM																										
	USB_DP																										

⚙️ Lab6 USB Device CDC with CONSOLE System Service

- Modify your main.c as below

```
SYS_TIME_HANDLE tmrHandle;
SYS_TIME_HANDLE tmrHandle_USB_hello;
SYS_CONSOLE_HANDLE consoleHandlePLIB;
SYS_CONSOLE_HANDLE consoleHandleUSB;
char UartWriteBuf[30];
char UartReadBuf[10];
...
int main( void )
{
    ...
    SYS_TIME_DelayMS(1000, &tmrHandle);
    SYS_TIME_DelayMS(1000, &tmrHandle_USB_hello);
    ...
    consoleHandlePLIB = SYS_CONSOLE_HandleGet( SYS_CONSOLE_INDEX_0 );
    sprintf(UartWriteBuf, "Hello world !(PLIB)\r\n");
    SYS_CONSOLE_Write(consoleHandlePLIB, UartWriteBuf, strlen(UartWriteBuf));

    consoleHandleUSB = SYS_CONSOLE_HandleGet( SYS_CONSOLE_INDEX_1 );

    while ( true )
    {
        if( SYS_TIME_DelayIsComplete(tmrHandle) == true ) {...}

        if( SYS_TIME_DelayIsComplete(tmrHandle_USB_hello) == true )
        {
            sprintf(UartWriteBuf, "Hello world !(USB)\r\n");
            SYS_CONSOLE_Write(consoleHandleUSB, UartWriteBuf, strlen(UartWriteBuf));
        }

        if( SYS_CONSOLE_Read(consoleHandlePLIB, UartReadBuf, 1) > 0 ) {...}
        ...
    }
}
```

⚙️ Lab6 USB Device CDC with CONSOLE System Service

- Modify your main.c as below

```
int main( void )
{
    ...
    consoleHandlePLIB = SYS_CONSOLE_HandleGet( SYS_CONSOLE_INDEX_0 );
    sprintf(UartWriteBuf, "Hello world !(PLIB)\r\n");
    SYS_CONSOLE_Write(consoleHandlePLIB, UartWriteBuf, strlen(UartWriteBuf));

    consoleHandleUSB = SYS_CONSOLE_HandleGet( SYS_CONSOLE_INDEX_1 );

    while ( true )
    {
        ...
        if( SYS_TIME_DelayIsComplete(tmrHandle_USB_hello) == true )
        {
            sprintf(UartWriteBuf, "Hello world !(USB)\r\n");
            SYS_CONSOLE_Write(consoleHandlePLIB, UartWriteBuf, strlen(UartWriteBuf));
        }

        if( SYS_CONSOLE_Read(consoleHandlePLIB, UartReadBuf, 1) > 0 )
        {
            sprintf(UartWriteBuf, "Receive : %s (PLIB)\r\n", UartReadBuf);
            SYS_CONSOLE_Write(consoleHandleUSB, UartWriteBuf, strlen(UartWriteBuf));
        }

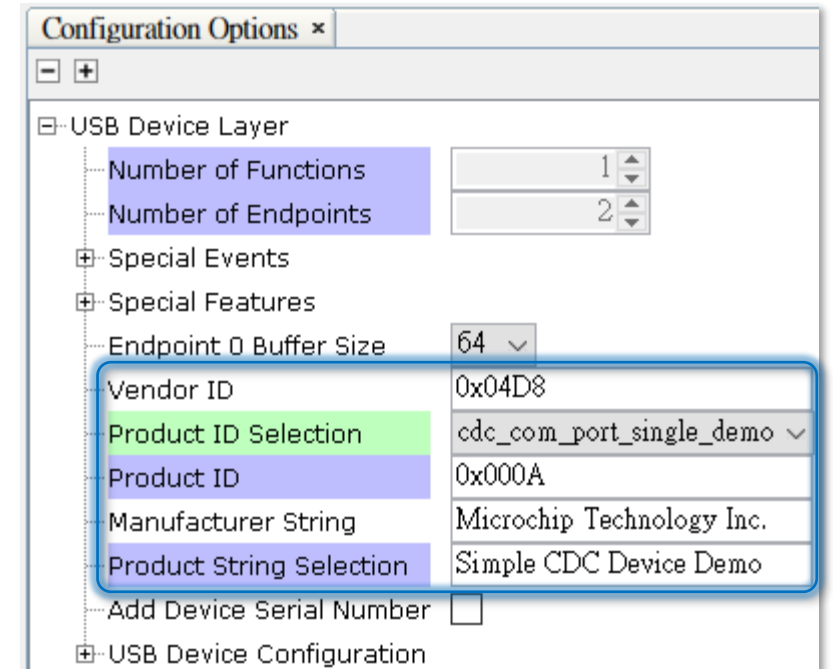
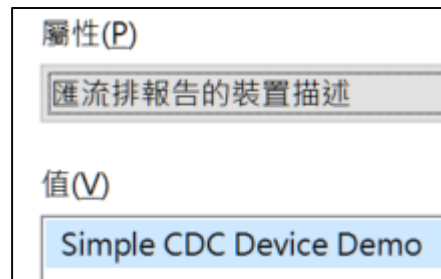
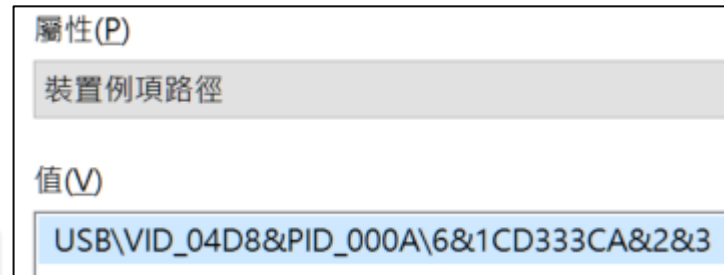
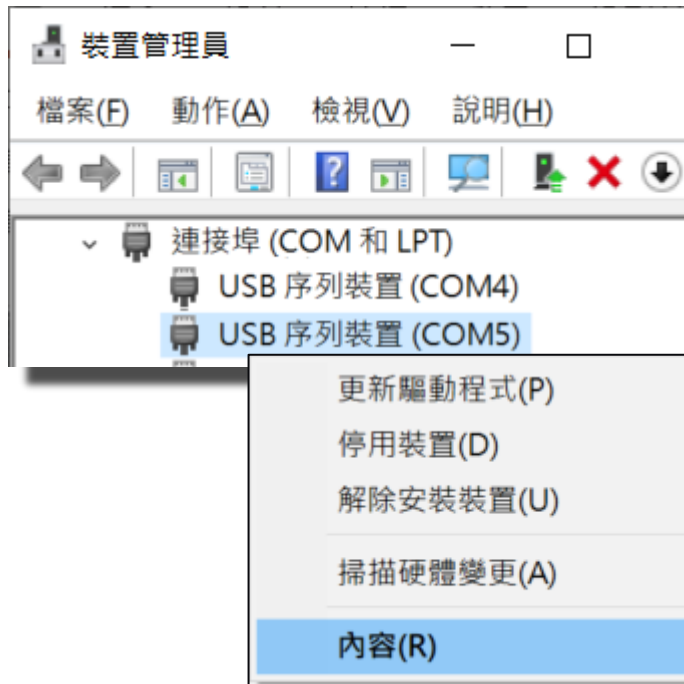
        if( SYS_CONSOLE_Read(consoleHandleUSB, UartReadBuf, 1) > 0 )
        {
            sprintf(UartWriteBuf, "Receive : %s (USB)\r\n", UartReadBuf);
            SYS_CONSOLE_Write(consoleHandlePLIB, UartWriteBuf, strlen(UartWriteBuf));
        }

        SYS_Tasks ( );
    }
}
```

Modify this line ➡

✖ Lab6 USB Device CDC with CONSOLE System Service

- Compiler and programming your project to EVB.
- Connect both two USB to your PC.
- The Device manager will detect a new USB CDC device.
- You can check the new COM Port properties.



⚙️ **裝置已就緒**
'USB 序列裝置 (COM5)' 已設定完成並隨時可用。

⚙️ **正在設定裝置**
我們正在設定 'Simple CDC Device Demo'。

🔧 Lab6 USB Device CDC with CONSOLE System Service

- If you are using Windows 7 or previous version, please install USB CDC driver from the example APP in Harmony Framework path.

<Harmony Framework Path>\usb_apps_device\apps\cdc_com_port_single\inf\

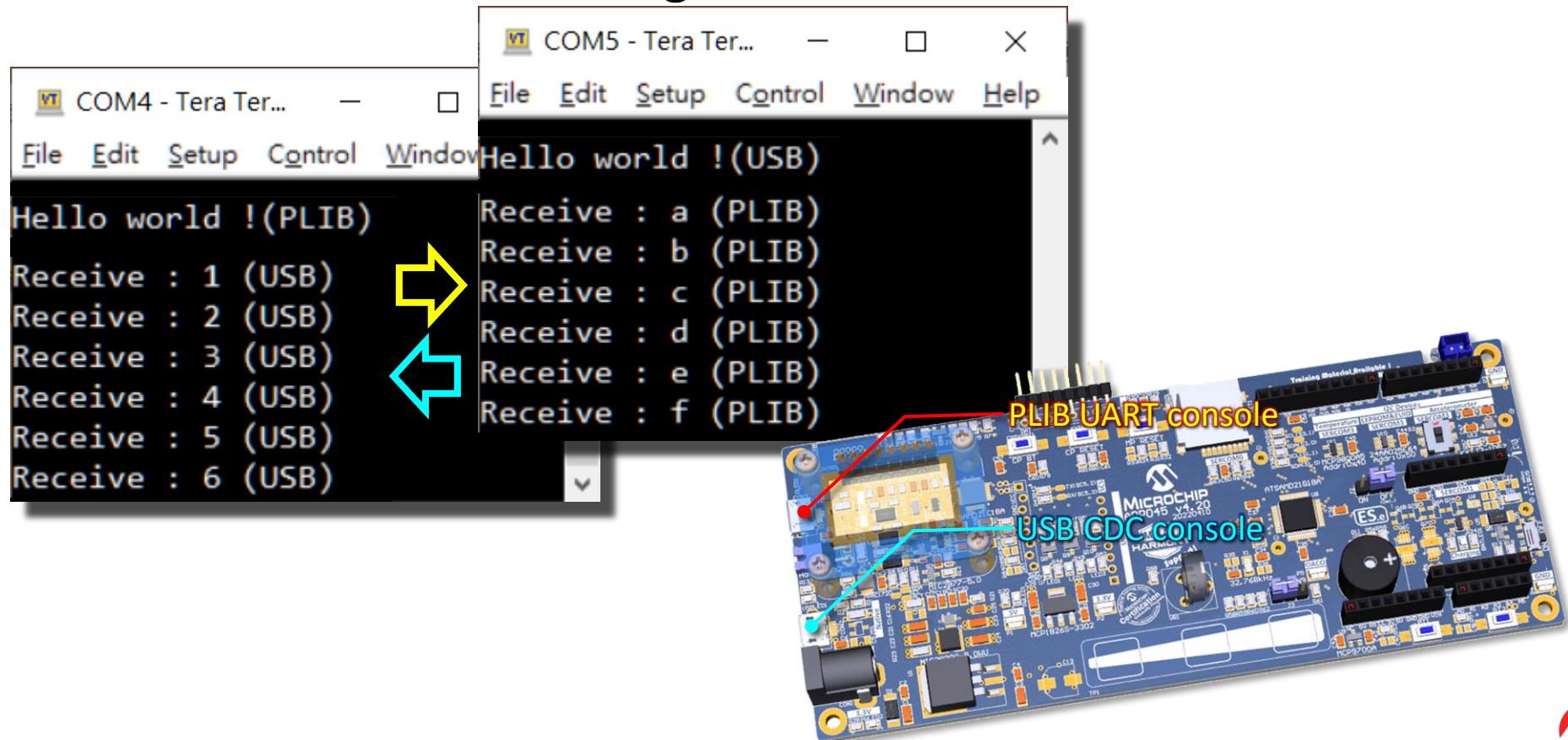
- Or Download the INF file directly from GitHub.

https://github.com/Microchip-MPLAB-Harmony/usb_apps_device/tree/master/apps/cdc_com_port_single/inf



⚙️ Lab6 USB Device CDC with CONSOLE System Service

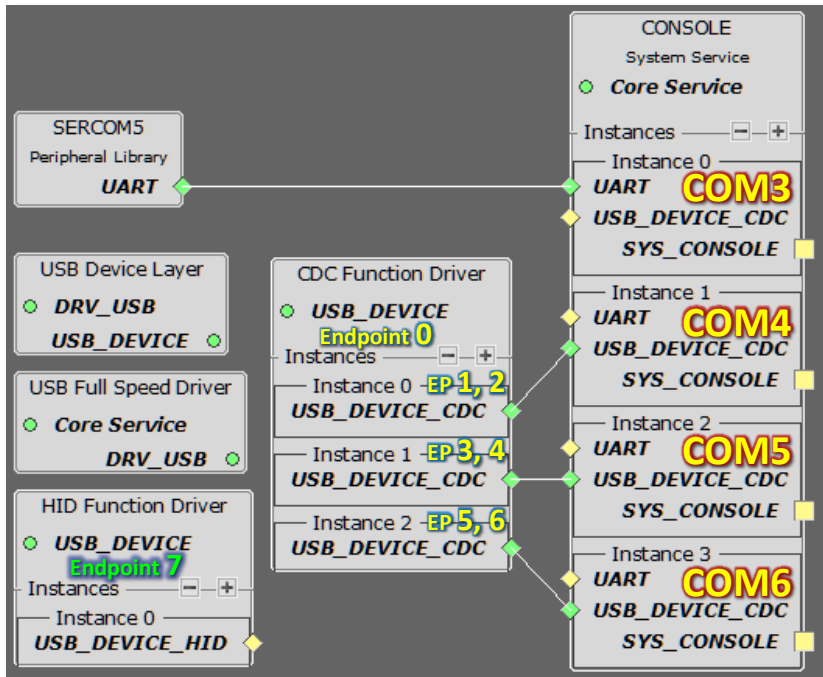
- Open two COM Port terminal console 115200-n-8-1.
- Press [Reset] button on EVB and check the 'Hello world!(xxx)' to know which one is from PLIB or from USB.
- Type-in in one console will echo message to another one.



✂ Lab6 USB Device CDC with CONSOLE System Service

Discuss 1

- How many USB device CDC instance could support by SAMD21 ?
 - USB CDC Device is a collection of the following Interfaces and Endpoints
 - Communication Interface (Device Management) on [Endpoint 0](#).
 - Communication Interface (Notification) on an Interrupt(IN) [Endpoint](#).
 - Data Interface on a Bulk(IN/OUT) [Endpoint](#).
- SAMD21 USB device mode can support up to 8 endpoints (IN/OUT), the first CDC instance will consume three endpoints, and subsequent instances will consume two.



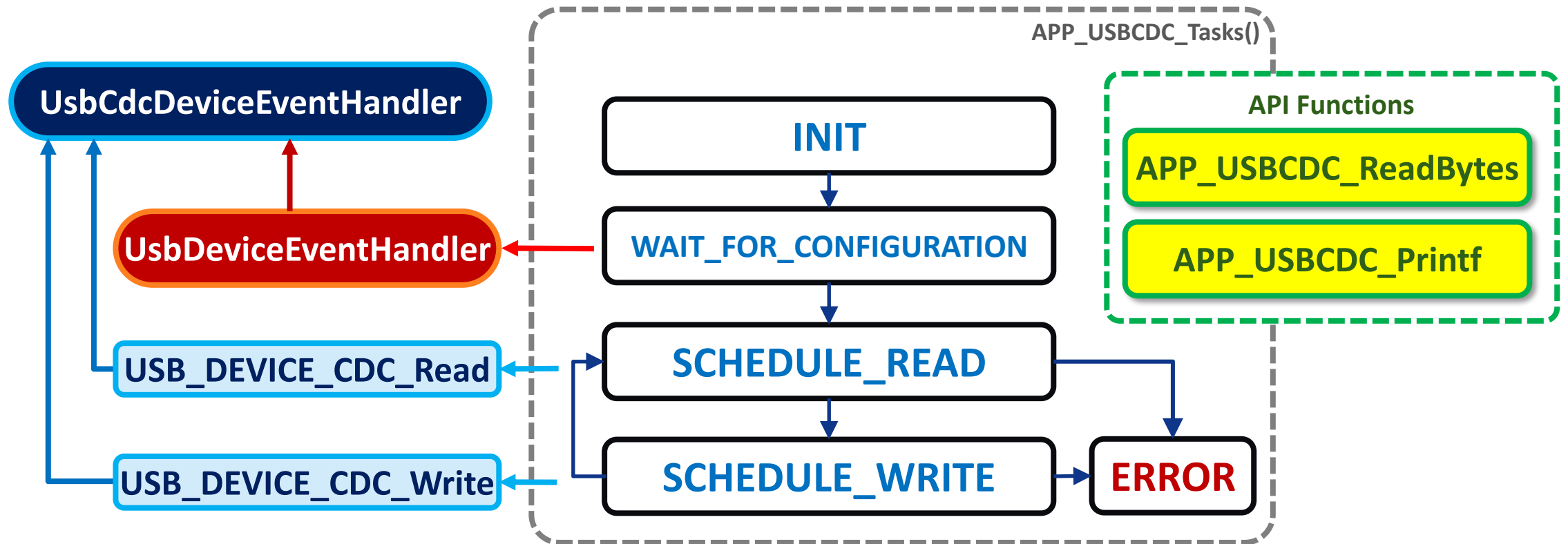
The screenshot shows the 'USB Device Layer' configuration window. The 'Number of Functions' is set to 1, and the 'Number of Endpoints' is set to 2. Under 'Special Features', 'Enable advanced String Descriptor Table' and 'Enable BOS Descriptor Support' are both unchecked. The 'Endpoint 0 Buffer Size' is set to 64. The 'Vendor ID' is 0x04D8. The 'Product ID Selection' dropdown is open, showing 'Enter Product ID' as the selected option. The 'Product ID' is 0x0000. The 'Manufacturer String' is 'Microchip Technology Inc.'. The 'Product String Selection' is 'Enter Product string here'. The 'Add Device Serial Number' checkbox is unchecked. The 'USB Device Configuration' section is expanded.

- ▼ 連接埠 (COM 和 LPT)
 - USB 序列裝置 (COM4) **PLIB SERCOM5**
 - USB 序列裝置 (COM5) **CDC Instance 0**
 - USB 序列裝置 (COM6) **CDC Instance 1**
 - USB 序列裝置 (COM7) **CDC Instance 2**
- ▼ 滑鼠及其他指標裝置
 - HID-compliant mouse **HID Instance 0**

⚙️ Lab6 USB Device CDC with CONSOLE System Service

Discuss 2

- If it possible to implement a USB device CDC console without CONSOLE System service?
 - Yes, but you will need to implement the USB Device and USB Device CDC Event Handler by yourself through the API functions of USB Device Layer and USB Device CDC library.



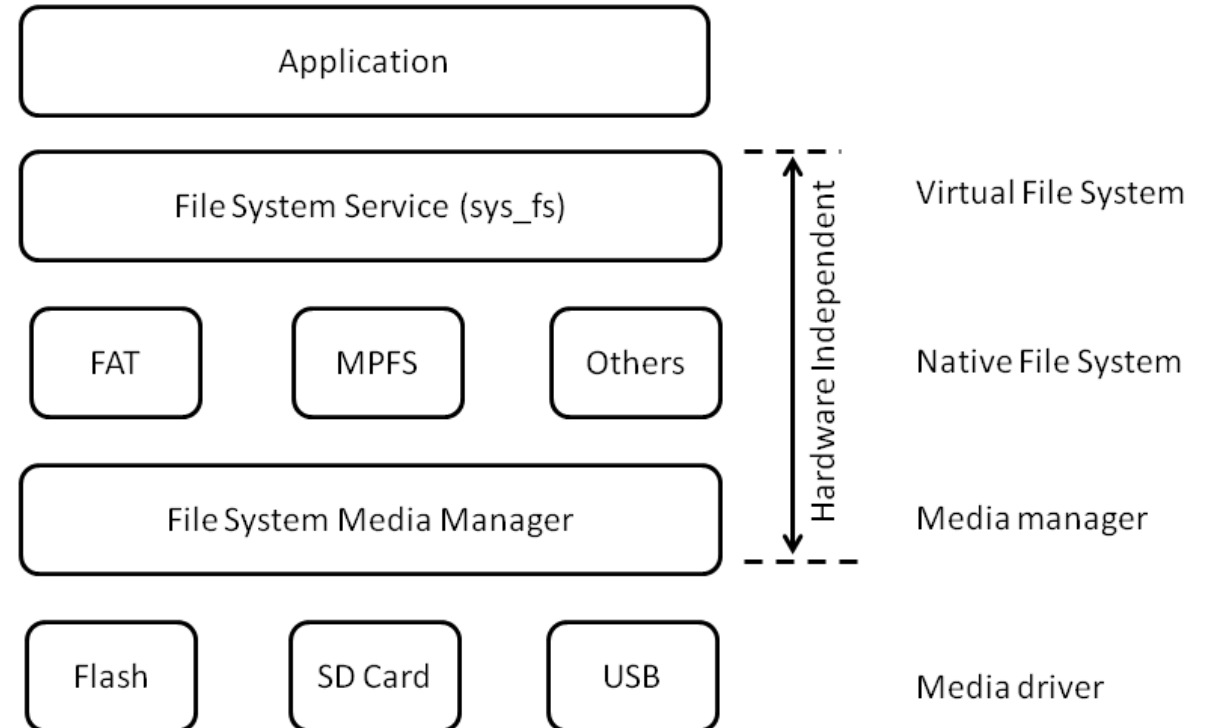
FILE System Service

Harmony System Service

FILE System Service

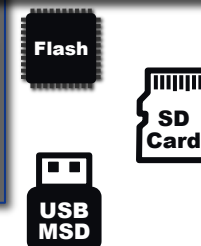
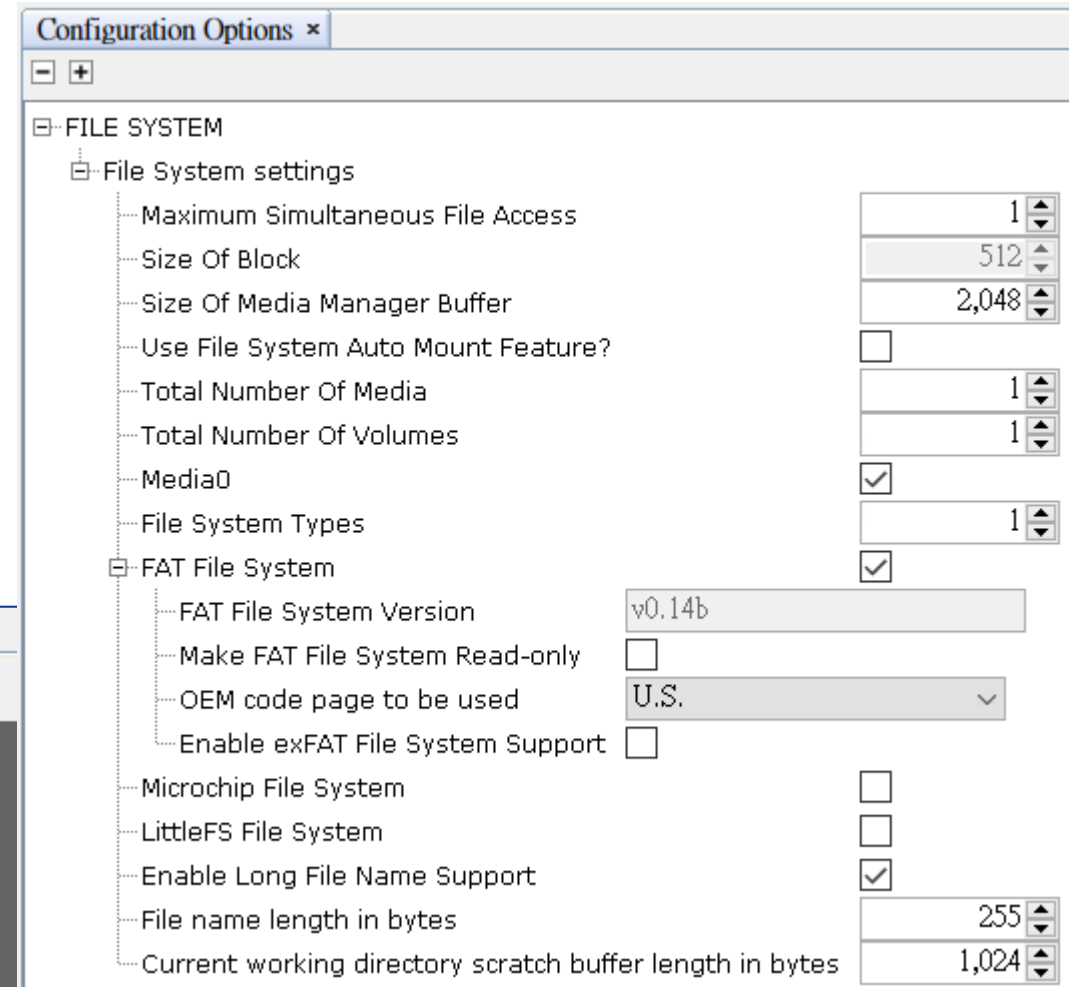
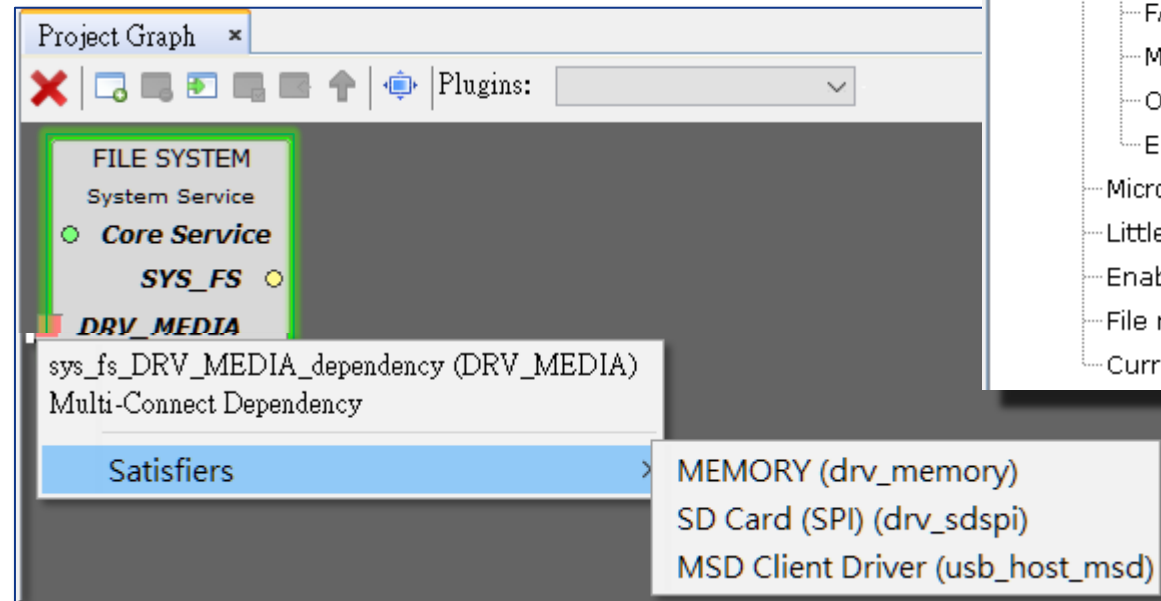
- Introduction

The File System Service provides an application programming interface (API) through which a utility or user program requests services of a file system. The File System service is really a framework designed to support multiple native file system and multiple media in the same application. Examples of native file systems are FAT12, FAT16, FAT32, and the Microchip File System (MPFS) among others. Each of these native file systems have a common set of APIs that can be used to access the files of that particular native file system. The File System Service abstracts the native file system calls and provides a common interface to the user/application_layer.

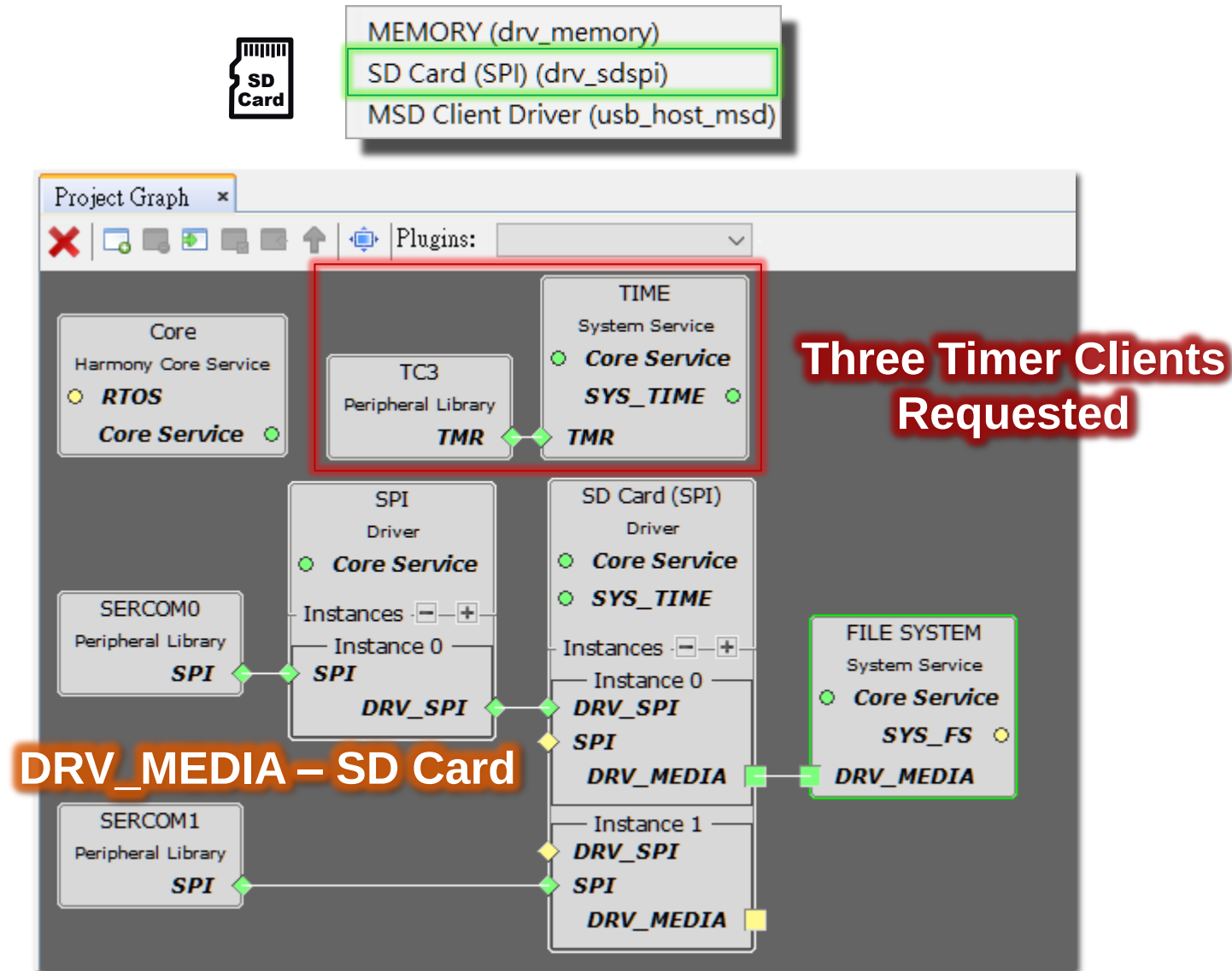


FILE System Service (DRV_MEDIA)

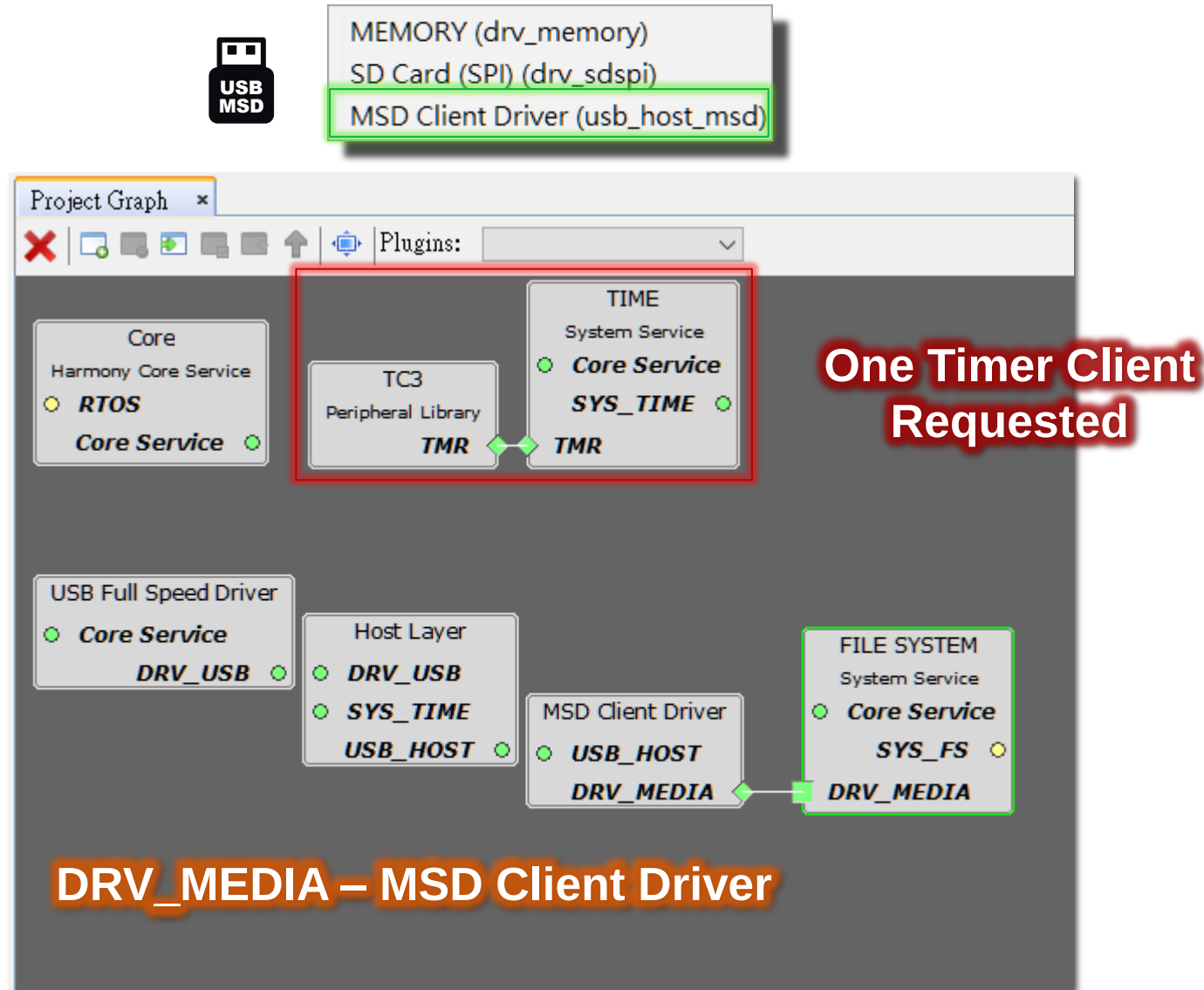
**FILE SYSTEM service
could support couple type
of Media Content**



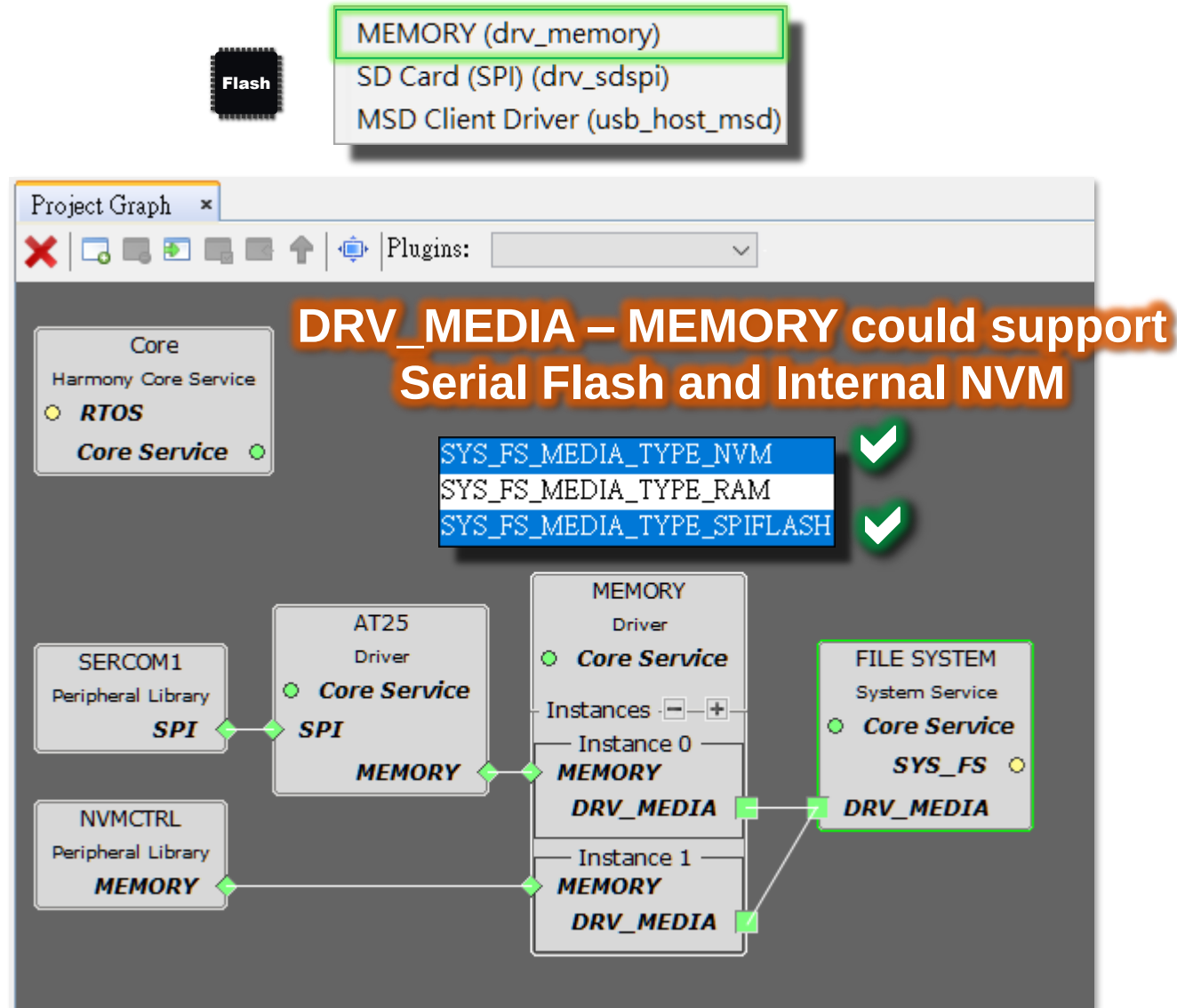
FILE System Service (SD Card)



FILE System Service (MSD Client Driver)



FILE System Service (MEMORY Client Driver)



FILE System Service

API functions of the File System Service library.

- File and Directory Operation Functions

SYS_FS_FileOpen	Opens a file.
SYS_FS_FileClose	Closes a file.
SYS_FS_FileCharacterPut	Writes a character to a file.
SYS_FS_FileEOF	Checks for end of file.
SYS_FS_FileError	Returns the file specific error.
SYS_FS_FileNameGet	Reads the file name.
SYS_FS_FilePrintf	Writes a formatted string into a file.
SYS_FS_FileRead	Read data from the file.
SYS_FS_FileSeek	Moves the file pointer by the requested offset.
SYS_FS_FileSize	Returns the size of the file in bytes.
SYS_FS_FileStat	Gets file status.
SYS_FS_FileStringGet	Reads a string from the file into a buffer.
SYS_FS_FileStringPut	Writes a string to a file.
SYS_FS_FileSync	Flushes the cached information when writing to a file.
SYS_FS_FileTell	Obtains the file pointer position.
SYS_FS_FileTestError	Checks for errors in the file.
SYS_FS_FileTruncate	Truncates a file
SYS_FS_FileWrite	Writes data to the file.
SYS_FS_DirectoryMake	Makes a directory.
SYS_FS_DirOpen	Open a directory
SYS_FS_DirClose	Closes an opened directory.

FILE System Service

API functions of the File System Service library. (cont.)

- File and Directory Operation Functions (cont.)

<code>SYS_FS_DirRead</code>	Reads the files and directories of the specified directory.
<code>SYS_FS_DirRewind</code>	Rewinds to the beginning of the directory.
<code>SYS_FS_DirSearch</code>	Searches for a file or directory.
<code>SYS_FS_DirectoryChange</code>	Changes to the directory specified.
<code>SYS_FS_CurrentWorkingDirectoryGet</code>	Gets the current working directory
<code>SYS_FS_FileDirectoryModeSet</code>	Sets the mode for the file or directory.
<code>SYS_FS_FileDirectoryRemove</code>	Removes a file or directory.
<code>SYS_FS_FileDirectoryRenameMove</code>	Renames or moves a file or directory.
<code>SYS_FS_FileDirectoryTimeSet</code>	Sets or changes the time for a file or directory.
<code>SYS_FS_Error</code>	Returns the last error.

- General Operation Functions

<code>SYS_FS_Initialize</code>	Initializes the file system abstraction layer (sys_fs layer).
<code>SYS_FS_Tasks</code>	Maintains the File System tasks and functionalities.
<code>SYS_FS_Mount</code>	Mounts the file system.
<code>SYS_FS_Unmount</code>	Unmounts the file system.
<code>SYS_FS_CurrentDriveGet</code>	Gets the current drive
<code>SYS_FS_CurrentDriveSet</code>	Sets the drive.
<code>SYS_FS_DriveLabelGet</code>	Gets the drive label.
<code>SYS_FS_DriveLabelSet</code>	Sets the drive label
<code>SYS_FS_DriveFormat</code>	Formats a drive.

FILE System Service

API functions of the File System Service library. (cont.)

- General Operation Functions (cont.)

<code>SYS_FS_DrivePartition</code>	Partitions a physical drive (media).
<code>SYS_FS_DriveSectorGet</code>	Obtains total number of sectors and number of free sectors for the specified drive.
<code>SYS_FS_EventHandlerSet</code>	Allows a client to identify an event handling function for the file system to call back when mount/unmount operation has completed. Functions active when <code>SYS_FS_AUTOMOUNT_ENABLE</code> is true.

- Media Manager Functions

<code>SYS_FS_MEDIA_MANAGER_Tasks</code>	Media manager task function.
<code>SYS_FS_MEDIA_MANAGER_TransferTask</code>	Media manager transfer task function.
<code>SYS_FS_MEDIA_MANAGER_Read</code>	Gets data from a specific media address.
<code>SYS_FS_MEDIA_MANAGER_SectorRead</code>	Reads a specified media sector.
<code>SYS_FS_MEDIA_MANAGER_SectorWrite</code>	Writes a sector to the specified media.
<code>SYS_FS_MEDIA_MANAGER_Register</code>	Function to register media drivers with the media manager.
<code>SYS_FS_MEDIA_MANAGER_RegisterTransferHandler</code>	Register the event handler for data transfer events.
<code>SYS_FS_MEDIA_MANAGER_DeRegister</code>	Function called by a media to deregister itself.
<code>SYS_FS_MEDIA_MANAGER_AddressGet</code>	Gets the starting media address based on a disk number.
<code>SYS_FS_MEDIA_MANAGER_MediaStatusGet</code>	Gets the media status.
<code>SYS_FS_MEDIA_MANAGER_VolumePropertyGet</code>	Gets the volume property.
<code>SYS_FS_MEDIA_MANAGER_CommandStatusGet</code>	Gets the command status.
<code>SYS_FS_MEDIA_MANAGER_GetMediaGeometry</code>	Gets the media geometry information.
<code>SYS_FS_MEDIA_MANAGER_EventHandlerSet</code>	Register the event handler for Mount/Un-Mount events.
<code>_SYS_FS_MountVolume</code>	Mount the volume when AUTOMOUNT feature is enabled.

FILE System Service

API functions of the File System Service library. (cont.)

- File System Data Types and Constants

SYS_FS_ERROR	Lists the various error cases.
SYS_FS_FILE_SEEK_CONTROL	Lists the various modes of file seek.
SYS_FS_FSTAT	File System status
SYS_FS_FUNCTIONS	SYS FS Function signature structure for native file systems.
SYS_FS_REGISTRATION_TABLE	The sys_fs layer has to be initialized by passing this structure with suitably initialized members.
SYS_FS_RESULT	Lists the various results of a file operation.
SYS_FS_FILE_OPEN_ATTRIBUTES	Lists the various attributes (modes) in which a file can be opened.
FAT_FS_MAX_LFN	Maximum length of the Long File Name.
FAT_FS_MAX_SS	Lists the definitions for FAT file system sector size.
FAT_FS_USE_LFN	Lists the definitions for FAT file system LFN selection.
SYS_FS_FILE_SYSTEM_TYPE	Enumerated data type identifying native file systems supported.
SYS_FS_HANDLE	This type defines the file handle.
SYS_FS_HANDLE_INVALID	Invalid file handle
SYS_FS_FILE_DIR_ATTR	Enumerated data type identifying the various attributes for file/directory.
SYS_FS_TIME	The structure to specify the time for a file or directory.
SYS_FS_FORMAT	Specifies the partitioning rule.
SYS_FS_EVENT	Identifies the possible file system events.
SYS_FS_EVENT_HANDLER	Pointer to the File system Handler function.
SYS_FS_MEDIA_BLOCK_COMMAND_HANDLE_INVALID	Defines the invalid media block command handle.
SYS_FS_MEDIA_HANDLE_INVALID	Defines the invalid media handle.

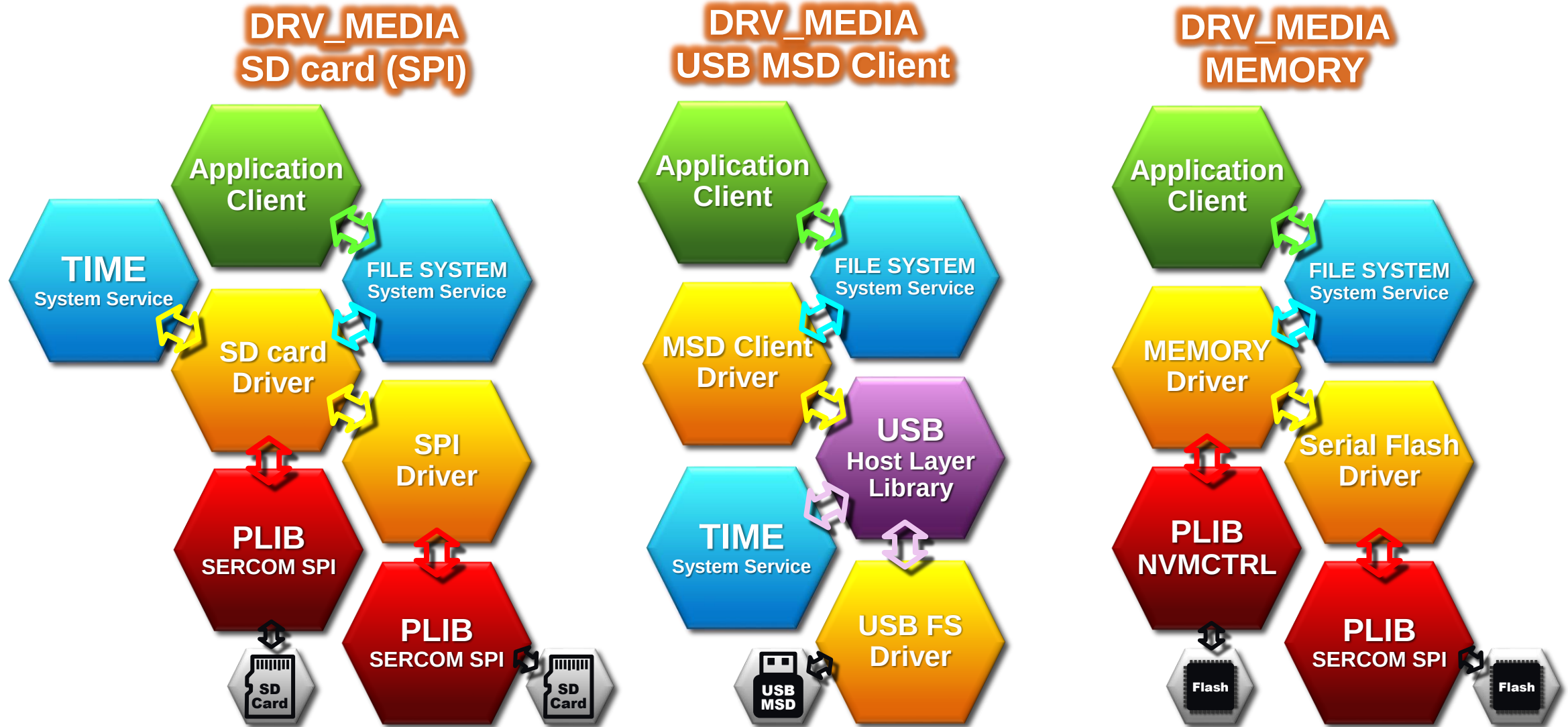
FILE System Service

API functions of the File System Service library. (cont.)

- File System Data Types and Constants (cont.)

<code>_SYS_FS_VOLUME_PROPERTY</code>	Structure to obtain the property of volume
<code>SYS_FS_MEDIA_BLOCK_EVENT</code>	Identifies the possible events that can result from a request.
<code>SYS_FS_MEDIA_COMMAND_STATUS</code>	The enumeration for status of buffer
<code>SYS_FS_MEDIA_FUNCTIONS</code>	Structure of function pointers for media driver
<code>SYS_FS_MEDIA_MOUNT_DATA</code>	Structure to obtain the device and mount name of media
<code>SYS_FS_MEDIA_PROPERTY</code>	Contains information of property of a media.
<code>SYS_FS_MEDIA_STATE</code>	The enumeration for state of media.
<code>SYS_FS_MEDIA_STATUS</code>	The state of media.
<code>SYS_FS_MEDIA_TYPE</code>	The enumeration for type of media.
<code>SYS_FS_MEDIA_BLOCK_COMMAND_HANDLE</code>	Handle identifying block commands of the media.
<code>SYS_FS_MEDIA_EVENT_HANDLER</code>	Pointer to the Media Event Handler function.
<code>SYS_FS_MEDIA_GEOMETRY</code>	Contains all the geometrical information of a media device.
<code>SYS_FS_MEDIA_HANDLE</code>	Handle identifying the media registered with the media manager.
<code>SYS_FS_MEDIA_REGION_GEOMETRY</code>	Contains information of a sys media region.
<code>SYS_FS_VOLUME_PROPERTY</code>	Structure to obtain the property of volume

FILE System Service



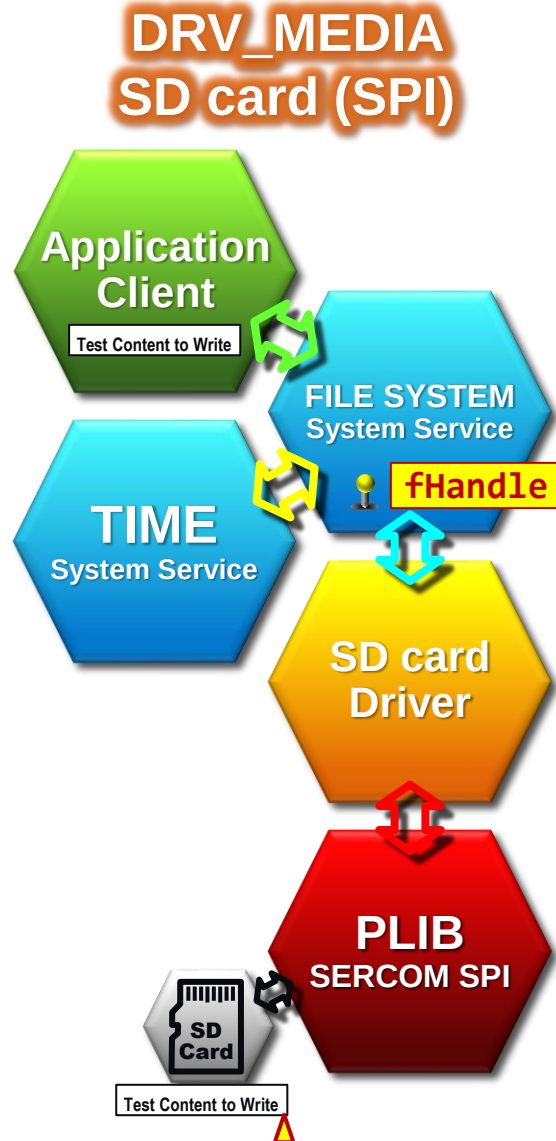
FILE System Service

- The standard names for volumes (devName) used in the MPLAB Harmony file system is as follows:

NVM Internal Flash :	nvm	[media number]	[volume number]
SD card :	mmcblk	[media number]	[volume number]
USB Mass Storage :	sd	[media number]	[volume number]
External SPI Flash :	mtd	[media number]	[volume number]
Internal SRAM :	ram	[media number]	[volume number]

- [media number] is **a, b, c...** depends on the number of connected media.
- [volume number] is **1, 2, 3...** depends on the number partitions of media.
- If a SD card with four partitions is attached to the system, the devNames will be **mmcblk**a**1, mmcblk**a**2, mmcblk**a**3, mmcblk**a**4**
- If a NVM media with one partition attached, the devName will be **nvma**1****.
- If 2nd SD card with one partition attached, the devName will be **mmcblk**b**1**.
- Finally, there will be six devNames which are available for the application to be mounted and used for the file system. (**mmcblk**a**1, mmcblk**a**2, mmcblk**a**3, mmcblk**a**4, nvma**1**, mmcblk**b**1**)
- The media devName must be preceded by the string **"/dev/"**, **the devName should use Harmony naming rule.**
/dev/mmcblka**1**
- The drive mount name must be preceded by the string **"/mnt/"**, **the mount name is customizable.**
/mnt/myDrive1

FILE System Service (SD Card)



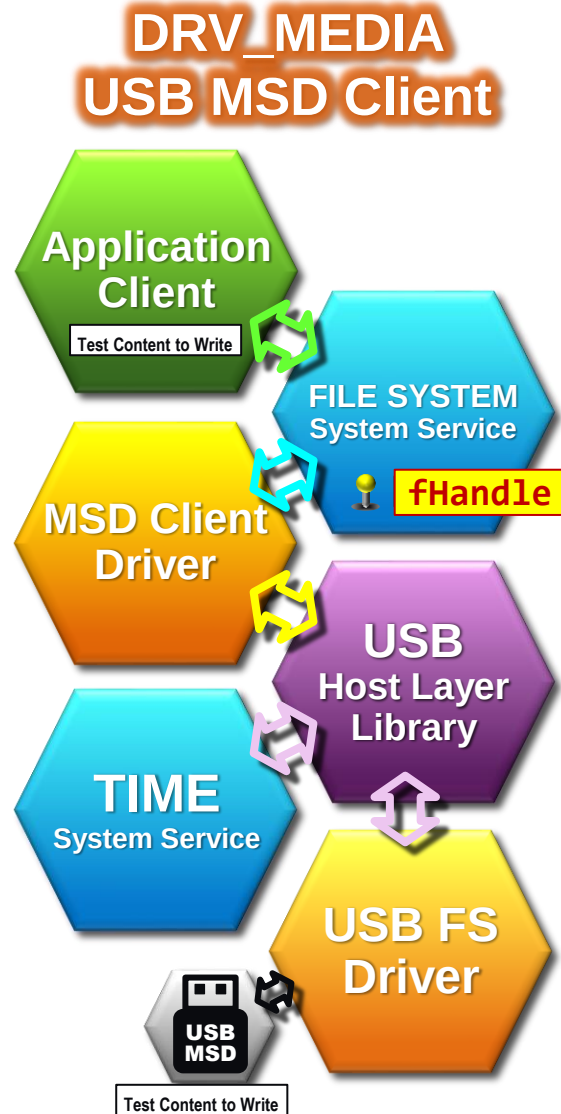
```
#define SDCARD_DEV_NAME    "/dev/mmcblk1"
#define SDCARD_MOUNT_NAME "/mnt/myDrive1"
#define FILE_CONTENT      "Test Content to Write"
SYS_FS_HANDLE fHandle = SYS_FS_HANDLE_INVALID;
char readBuffer[30];

void CommandSDCard(SYS_CMD_DEVICE_NODE* pCmdIO, int argc, char** argv)
{
    SYS_FS_Mount(SDCARD_DEV_NAME, SDCARD_MOUNT_NAME, FAT, 0, NULL);
    SYS_FS_CurrentDriveSet(SDCARD_MOUNT_NAME);
    fHandle = SYS_FS_FileOpen(argv[1], SYS_FS_FILE_OPEN_WRITE_PLUS);
    SYS_CMD_PRINT("- Create File : %s\r\n", argv[1] );
    SYS_FS_FileWrite(fHandle, FILE_CONTENT, sizeof(FILE_CONTENT));
    SYS_CMD_PRINT("- Write [%s] to File\r\n", FILE_CONTENT );
    SYS_FS_FileSeek(fHandle, 0, SYS_FS_SEEK_SET);
    SYS_FS_FileRead(fHandle, (void *)readBuffer, 30);
    SYS_CMD_PRINT("- Read content : [%s]\r\n", readBuffer );
    SYS_FS_FileClose(fHandle);
}

static const SYS_CMD_DESCRIPTOR MyCmdTbl[] =
{
    { "sdcard", CommandSDCard, ": SD Card Test" },
};
...
```

```
VI COM4 - Tera Term VT
>sdcard test.txt
- Create File : test.txt
- Write [Test Content to Write] to File
- Read content : [Test Content to Write]
>
```

FILE System Service (USB Mass Storage)



```
#define USBMSD_DEV_NAME    "/dev/sda1"
#define USBMSD_MOUNT_NAME "/mnt/myDrive2"
#define FILE_CONTENT      "Test Content to Write"
SYS_FS_HANDLE fHandle = SYS_FS_HANDLE_INVALID;
char readBuffer[30];

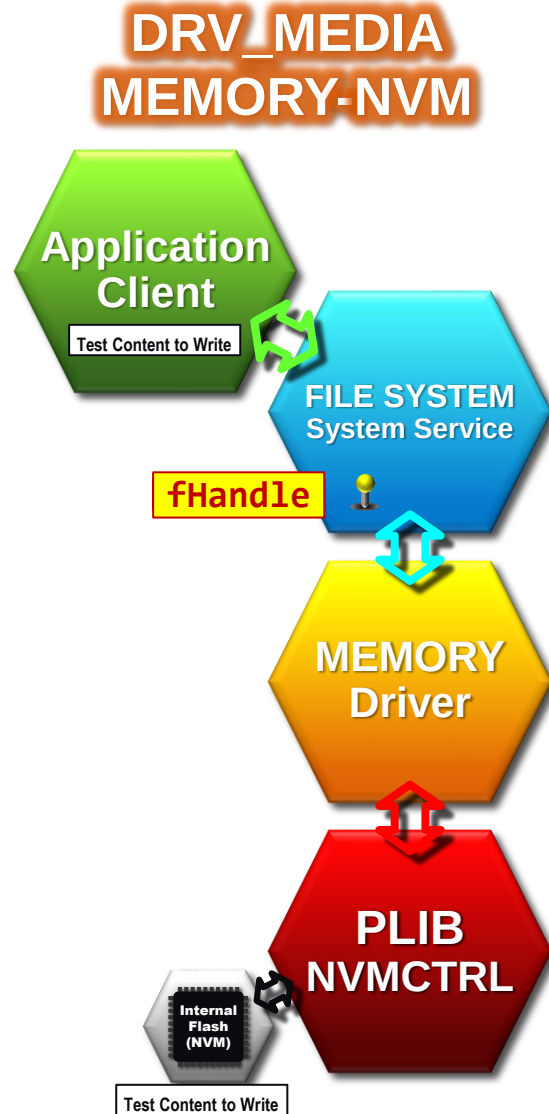
void CommandUsbMSD(SYS_CMD_DEVICE_NODE* pCmdIO, int argc, char** argv)
{
    USB_HOST_BusEnable(0);

    SYS_FS_Mount(USBMSD_DEV_NAME, USBMSD_MOUNT_NAME, FAT, 0, NULL);
    SYS_FS_CurrentDriveSet(USBMSD_MOUNT_NAME);

    fHandle = SYS_FS_FileOpen(argv[1], SYS_FS_FILE_OPEN_WRITE_PLUS);
    SYS_CMD_PRINT("- Create File : %s\r\n", argv[1] );
    SYS_FS_FileWrite(fHandle, FILE_CONTENT, sizeof(FILE_CONTENT));
    SYS_CMD_PRINT("- Write [%s] to File\r\n", FILE_CONTENT );
    SYS_FS_FileSeek(fHandle, 0, SYS_FS_SEEK_SET);
    SYS_FS_FileRead(fHandle, (void *)readBuffer, 30);
    SYS_CMD_PRINT("- Read content : [%s]\r\n", readBuffer );
    SYS_FS_FileClose(fHandle);
}

static const SYS_CMD_DESCRIPTOR MyCmdTbl[] =
{
    { "usbmsd", CommandUsbMSD, ": USB Mass Storage Test" },
};
...
```

FILE System Service (MEMORY-NVM)



```
#define MEMNVM_DEV_NAME    "/dev/nvma1"
#define MEMNVM_MOUNT_NAME "/mnt/myDrive3"
#define FILE_CONTENT      "Test Content to Write"
SYS_FS_HANDLE fHandle = SYS_FS_HANDLE_INVALID;
char readBuffer[30];

void CommandMemNVM(SYS_CMD_DEVICE_NODE* pCmdIO, int argc, char** argv)
{
    SYS_FS_FORMAT_PARAM opt = {SYS_FS_FORMAT_FAT, 0, 1, 0, 1 };
    uint8_t CACHE_ALIGN work[512];

    SYS_FS_Mount(MEMNVM_DEV_NAME, MEMNVM_MOUNT_NAME, FAT, 0, NULL);
    SYS_FS_CurrentDriveSet(MEMNVM_MOUNT_NAME);

    SYS_FS_DriveFormat(MEMNVM_MOUNT_NAME, &opt, (void *)work, 512);

    fHandle = SYS_FS_FileOpen(argv[1], SYS_FS_FILE_OPEN_WRITE_PLUS);
    SYS_CMD_PRINT("- Create File : %s\r\n", argv[1] );
    SYS_FS_FileWrite(fHandle, FILE_CONTENT, sizeof(FILE_CONTENT));
    SYS_CMD_PRINT("- Write [%s] to File\r\n", FILE_CONTENT );
    SYS_FS_FileSeek(fHandle, 0, SYS_FS_SEEK_SET);
    SYS_FS_FileRead(fHandle, (void *)readBuffer, 30);
    SYS_CMD_PRINT("- Read content : [%s]\r\n", readBuffer );
    SYS_FS_FileClose(fHandle);
}

static const SYS_CMD_DESCRIPTOR MyCmdTbl[]=
{
    { "memnvm", CommandMemNVM, ": Internal Flash (NVM) Test" },
};
```

Lab 7

SD Card with File System

FILE System Service

🔧 Lab7 SD Card with File System

- Continue this Lab from Lab5.
- Try add **FILE System Service** and select **SPI SD Card** as media.
- The **SERCOM0-SPI** as the hardware interface of SD card socket.
- Build custom command to access SD card as below :

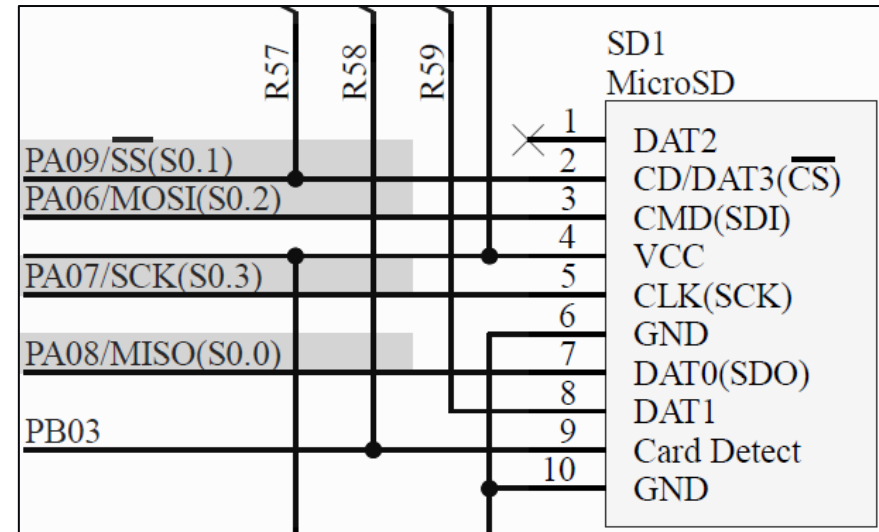
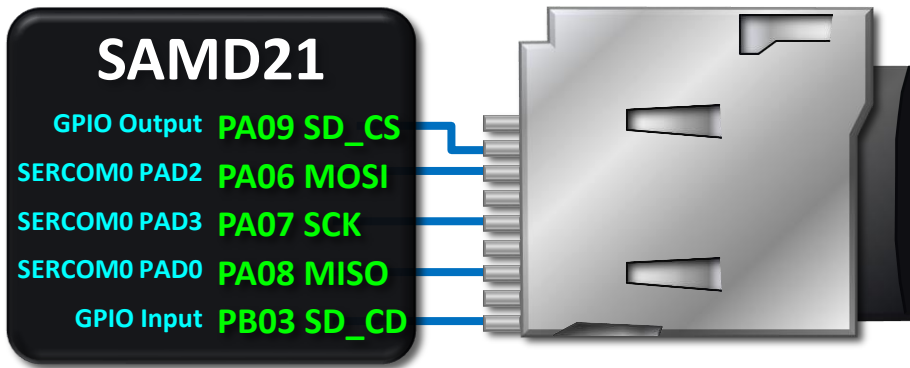
mount : Mount the SD card.

open \$: Open file \$, if not exist then create a new file.

write \$: Write string \$ to opened file.

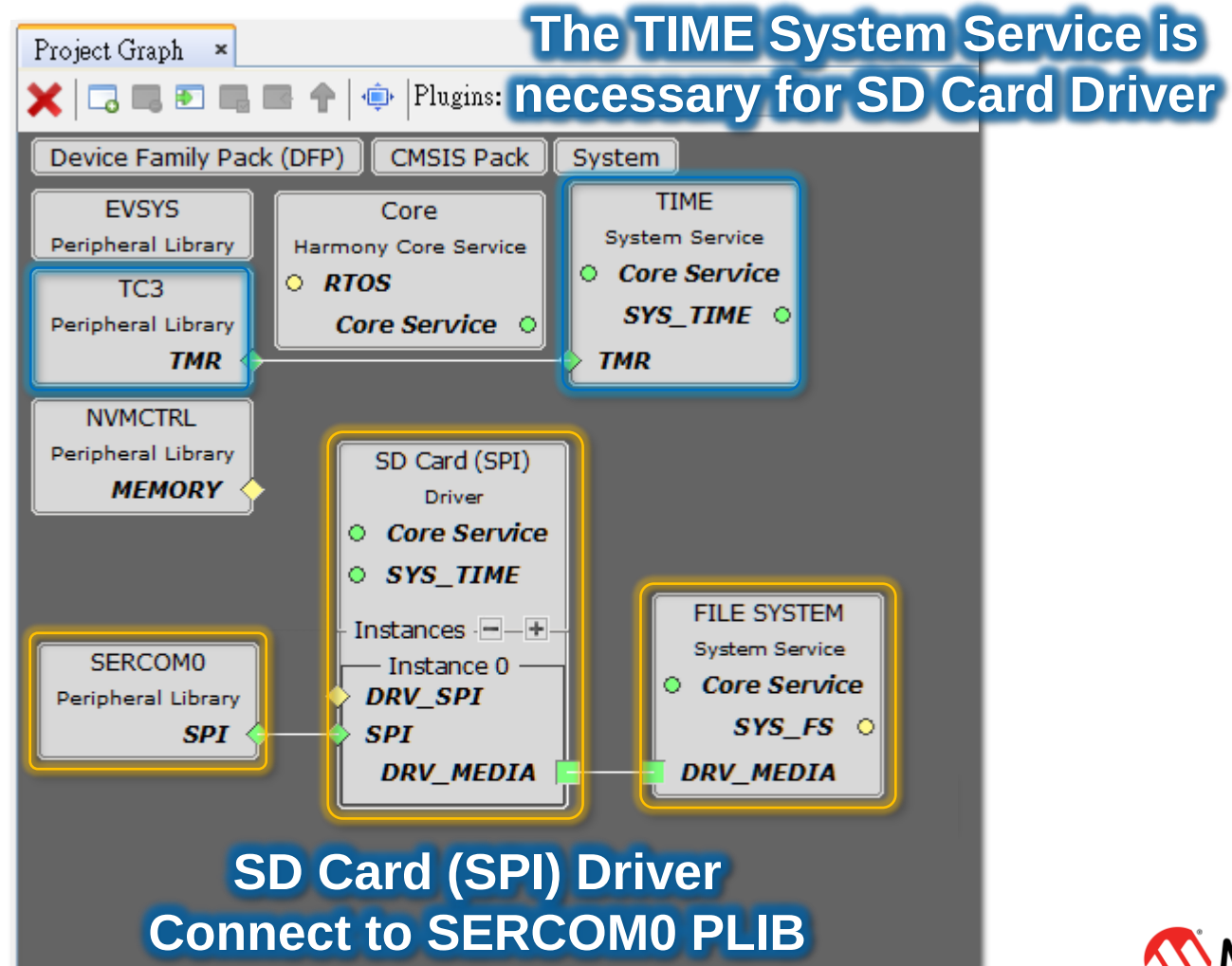
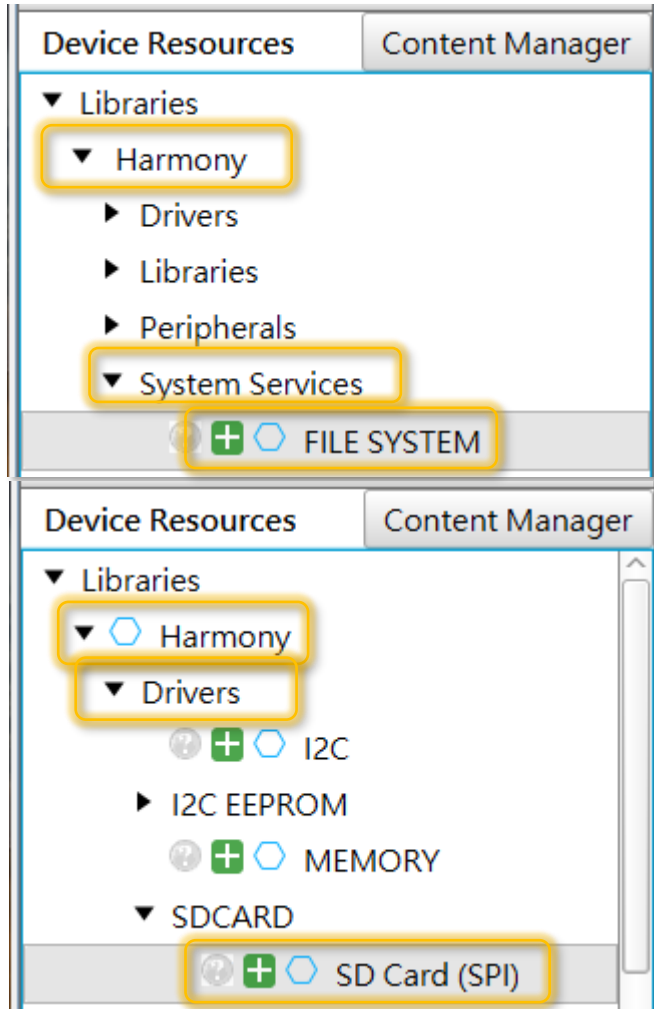
read # : Seek to start and Read # character from file.

close : Close opened file.



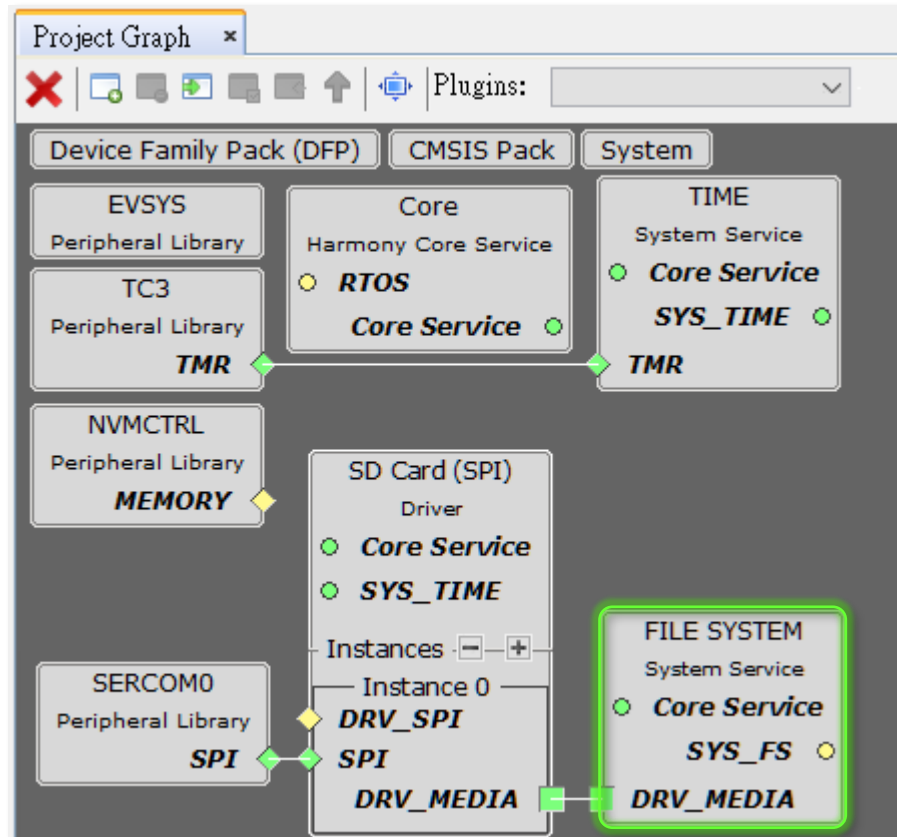
⚙️ Lab7 SD Card with File System

- Harmony Configuration (Add **FILE System Service** and **SD Card**)
- Select PLIB **SERCOM0** for SD Card (SPI) Driver.



✖ Lab7 SD Card with File System

- Harmony Configuration (FILE System Service configuration)



The Configuration Options dialog for the FILE SYSTEM service shows the following settings:

- FILE SYSTEM
 - File System settings
 - Maximum Simultaneous File Access: 1
 - Size Of Block: 512
 - Size Of Media Manager Buffer: 2,048
 - Use File System Auto Mount Feature?: ☐
 - Total Number Of Media: 1
 - Total Number Of Volumes: 1
 - Media0: ☒
 - File System Types: 1
 - FAT File System
 - FAT File System Version: v0.14b
 - Make FAT File System Read-only: ☐
 - OEM code page to be used: U.S.
 - Enable exFAT File System Support: ☐
 - Microchip File System: ☐
 - LittleFS File System: ☐
 - Enable Long File Name Support: ☒
 - File name length in bytes: 255
 - Custom File Name Length: 1,024

Keep the default settings
The Auto Mount Feature is disabled

✖ Lab7 SD Card with File System

- Harmony Configuration (PLIB SERCOM0 config)

Configuration Options

SERCOM0

- Select SERCOM Operation Mode: SPI Master
- Enable Interrupts?: ☒
- Enable operation in Standby mode: ☐
- SPI Data Out Pad: DO on PAD[2], SCK on PAD[3] and SS on PAD[1]
- SPI Data In Pad Selection: SERCOM PAD[0]
- SPI Data Order: MSB is transferred first
- SPI Speed in Hz: 1,000,000
- SPI Data Character Size: 8 bits
- SPI Clock Phase: The data is sampled on a leading SCK edge and changed on a trailing SCK edge
- SPI Clock Polarity: SCK is low when idle
- Enable SPI Master Hardware Slave Select: ☐
- SPI Receiver Enable: ☒

SPI Transfer Mode 0 is Selected

SAMD21

- GPIO Output PA09 SD_CS
- SERCOM0 PAD2 PA06 MOSI
- SERCOM0 PAD3 PA07 SCK
- SERCOM0 PAD0 PA08 MISO
- GPIO Input PB03 SD_CD

Project Structure:

- SERCOM0 Peripheral Library
- DRV_SPI
- DRV_MEDIA
- FILE SYSTEM System Service
- Core Service
- SYS_TIME
- SYS_FS

⚙️ Lab7 SD Card with File System

- Harmony Configuration (SD Card configuration)

SAMD21

GPIO Output **PA09 SD_CS**

SERCOM0 PAD2 **PA06 MOSI**

SERCOM0 PAD3 **PA07 SCK**

SERCOM0 PAD0 **PA08 MISO**

GPIO Input **PB03 SD_CD**

Configuration Options

SD Card (SPI)

PLIB Used: SERCOM0

Number of Clients: 1

SD Card Speed (Hz): 5,000,000

Transfer Queue Size: 4

Card Detection Method: Use Polling

Polling Interval (ms): 1,000

File system for SDSPI Driver Enabled: ☒

Chip Select Pin: **PA09**

!!! Configure the Chip Select pin as GPIO output under Pin Settings.!!!

Use Write Protect Pin (Active High)? ☐

Use DMA for Transmit and Receive ? ☒

DMA Channel For Transmit: 0

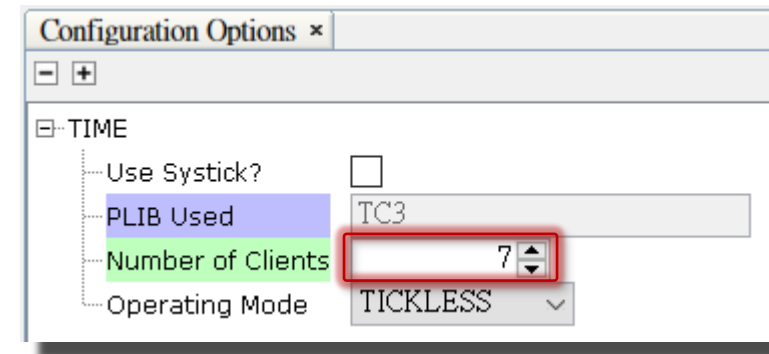
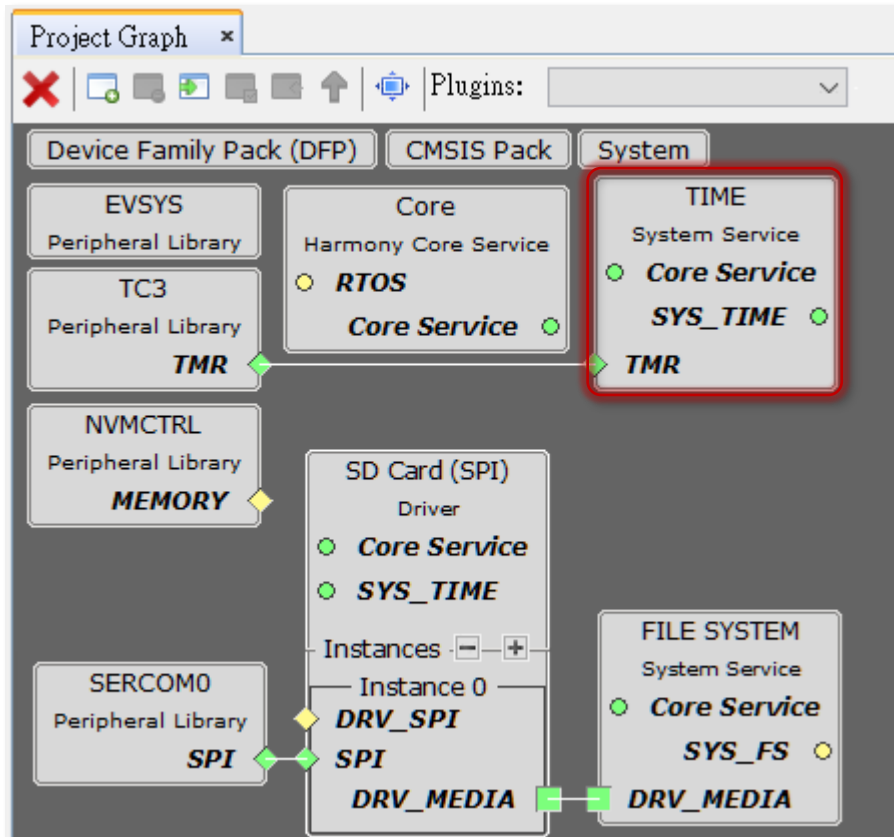
DMA Channel For Receive: 1

!!! Note: For each instance of SDSPI, connect either SPI PLIB or SPI Driver !!!

Set Chip Select pin to PA09 and Enable DMA

⚙️ Lab7 SD Card with File System

- Harmony Configuration (**TIME** System Service configuration)



**SD Card Driver requires at least
Three Timer Clients
Please make sure it's enough
for Driver and your other Applications.**

✂ Lab7 SD Card with File System

- Harmony Configuration (Setup SERCOM0 pads and GPIO pins)
- SERCOM0 on **PA06, PA07, PA08** ; GPIO on **PA09(Out), PB03(In)**.

Pin Settings * Pin Table * Pin Diagram * Project Graph*

Order: Ports Table View ☒ Easy View

Pin Number	Pin ID	Custom Name	Function	Mode
11	PA06		SERCOM0_PAD2	Digital
12	PA07		SERCOM0_PAD3	Digital
13	PA08		SERCOM0_PAD0	Digital
14	PA09	GPIO_PA09	GPIO	Digital
48	PB03	GPIO_PB03	GPIO	Digital

SD Card (SPI)

- Number of Clients: 1
- SD Card Speed (Hz): 5,000,000
- Transfer Queue Size: 4
- Card Detection Method: Use Polling
- Polling Interval (ms): 1,000
- File system for SDSPI Driver Enabled: ☒
- Chip Select Pin: PA09

Pin Table * Pin Settings * Pin Diagram * Project Graph*

Package: TQFP48

Module	Function	5	6	7	8	9	10	11	12	13	14	15	16	17	18
GPIO	GPIO														
SERCOM0	SERCOM0_PAD0														
	SERCOM0_PAD1														
	SERCOM0_PAD2														
	SERCOM0_PAD3														

SAMD21

- GPIO Output PA09 SD_CS
- SERCOM0 PAD2 PA06 MOSI
- SERCOM0 PAD3 PA07 SCK
- SERCOM0 PAD0 PA08 MISO
- GPIO Input PB03 SD_CD

✖ Lab7 SD Card with File System

- Modify your main.c as below

```
void CommandErrLv(SYS_CMD_DEVICE_NODE* pCmdIO, int argc, char** argv) { ... }

SYS_FS_HANDLE fHandle = SYS_FS_HANDLE_INVALID;
SYS_FS_RESULT fResult;
char fMedia[20] = "/dev/mmcblk1";
char fDrive[20] = "/mnt/myDrive1";
void CommandMount(SYS_CMD_DEVICE_NODE* pCmdIO, int argc, char** argv)
{
    fResult = SYS_FS_Mount(fMedia, fDrive, FAT, 0, NULL);
    if(fResult == SYS_FS_RES_SUCCESS)
    {
        fResult = SYS_FS_CurrentDriveSet(fDrive);
        if(fResult == SYS_FS_RES_SUCCESS)
        {
            SYS_CMD_PRINT("Mount %s success!\r\n", fMedia);
            SYS_CMD_PRINT("Current Drive is %s\r\n", fDrive);
            return;
        }
    }
    SYS_CMD_PRINT("Mount %s failure!\r\n", fMedia);
}

static const SYS_CMD_DESCRIPTOR MyCmdTbl[]=
{
    { "errlv", CommandErrLv, ": errlv #" },
    { "mount", CommandMount, ": Mount Media" },
};

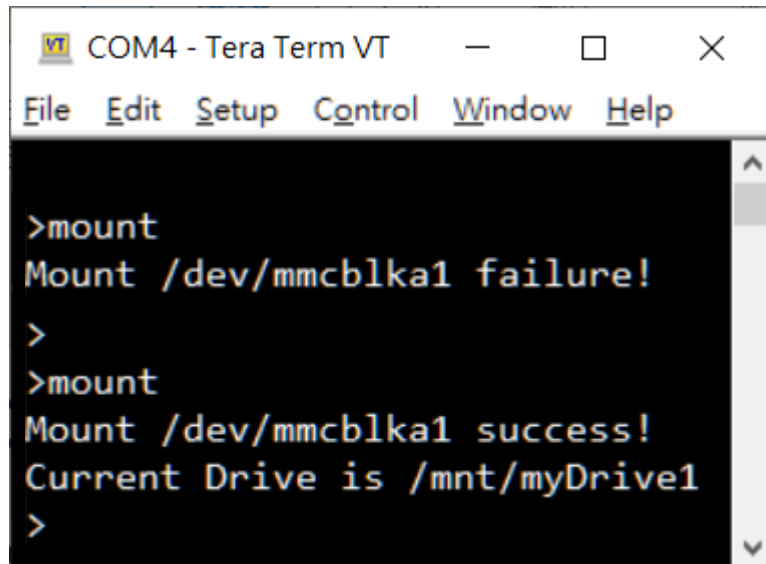
int main( void )
{
    SYS_CMD_ADDGRP(MyCmdTbl, sizeof(MyCmdTbl)/sizeof(SYS_CMD_DESCRIPTOR),
    "MYCMD", ": My Commands");
```

**Auto calculate
Command Table counts.**

Modify this line

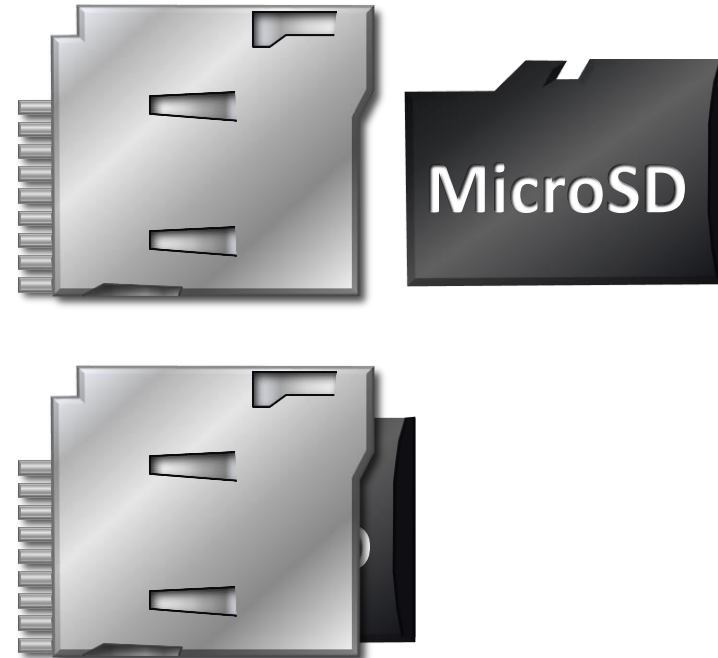
⚙️ Lab7 SD Card with File System

- Compiler and programming your project to EVB.
- **Remove SD card firstly.**
- Type-in '**mount**' to execute SD card mount operation.
- **Put in SD card.**
- Type-in '**mount**' again to see the result.



```
COM4 - Tera Term VT
File Edit Setup Control Window Help

>mount
Mount /dev/mmcblk1 failure!
>
>mount
Mount /dev/mmcblk1 success!
Current Drive is /mnt/myDrive1
>
```



⚙️ Lab7 SD Card with File System

- Modify your main.c as below

```
... void CommandMount(SYS_CMD_DEVICE_NODE* pCmdIO, int argc, char** argv) { ... }

void CommandOpen(SYS_CMD_DEVICE_NODE* pCmdIO, int argc, char** argv)
{
    fHandle = SYS_FS_FileOpen(argv[1], SYS_FS_FILE_OPEN_APPEND_PLUS);
    if(fHandle != SYS_FS_HANDLE_INVALID)
        SYS_CMD_PRINT("File [%s] opened!\r\n", argv[1] );
    else
        SYS_CMD_MESSAGE("Open failure!\r\n");
}

void CommandWrite(SYS_CMD_DEVICE_NODE* pCmdIO, int argc, char** argv)
{
    int fWriteSize = 0;

    fWriteSize = SYS_FS_FileWrite(fHandle, argv[1], strlen(argv[1]));
    if(fWriteSize != -1)
        SYS_CMD_PRINT("Write [%s](%d) to File!\r\n", argv[1], fWriteSize );
    else
        SYS_CMD_MESSAGE("Write failure!\r\n");
}

static const SYS_CMD_DESCRIPTOR MyCmdTbl[]=
{
    { "mount", CommandMount, ": Mount Media" },
    { "open",  CommandOpen,  ": open $" },
    { "write", CommandWrite, ": write $" },
};

int main( void ) { ... }
```

✖ Lab7 SD Card with File System

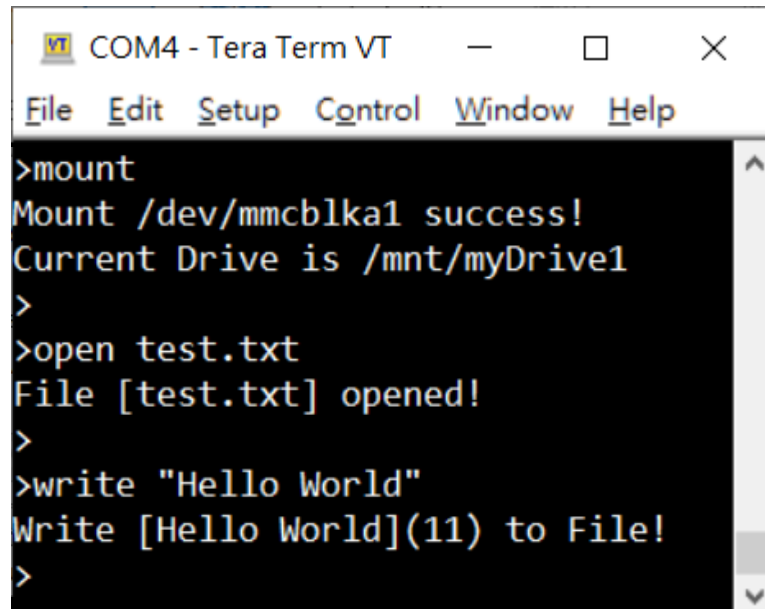
- The attribute **SYS_FS_FILE_OPEN_APPEND_PLUS** of file open operation could open target file or to create a new file if target file not exist and allow append bytes from the end of file.

```
void CommandOpen(SYS_CMD_DEVICE_NODE* pCmdIO, int argc, char** argv)
{
    fHandle = SYS_FS_FileOpen(argv[1], SYS_FS_FILE_OPEN_APPEND_PLUS);
    ...
}
```

SYS_FS_FILE_OPEN_	File exist (Open) or Not exist (Create)	Read from beginning	Read from the end	Write from Beginning	Write from the end
READ	Open / -----	✓			
WRITE	Open / Create			✓	
APPEND	Open / Create				✓
READ_PLUS	Open / -----	✓		✓	
WRITE_PLUS	Open / Create	✓		✓	
APPEND_PLUS	Open / Create		✓		✓

🔧 Lab7 SD Card with File System

- Compiler and programming your project to EVB.
- Put in SD Card and type-in 'mount' to mount the SD card.
- Type-in 'open test.txt' to execute file open operation,
if 'test.txt' not exist then create a new file for write and read.
if 'test.txt' already exist then to open it for write and read.
- Type-in 'write "Hello World"' to execute write operation.
- Remove SD Card and check 'test.txt' in the SD card on your PC.



```
COM4 - Tera Term VT
File Edit Setup Control Window Help
>mount
Mount /dev/mmcblk1 success!
Current Drive is /mnt/myDrive1
>
>open test.txt
File [test.txt] opened!
>
>write "Hello World"
Write [Hello World](11) to File!
>
```

Why not thing in test.txt ?



✂ Lab7 SD Card with File System

- Modify your main.c as below

```
void CommandMount(SYS_CMD_DEVICE_NODE* pCmdIO, int argc, char** argv) { ... }
void CommandOpen(SYS_CMD_DEVICE_NODE* pCmdIO, int argc, char** argv) { ... }
void CommandWrite(SYS_CMD_DEVICE_NODE* pCmdIO, int argc, char** argv) { ... }
void CommandRead(SYS_CMD_DEVICE_NODE* pCmdIO, int argc, char** argv)
{
    int fReadSize = 0;
    char fReadBuf[50] = "";

    SYS_FS_FileSeek(fHandle, 0, SYS_FS_SEEK_SET);
    fReadSize = SYS_FS_FileRead(fHandle, (void *)fReadBuf, atoi(argv[1]));
    if(fReadSize != -1)
        SYS_CMD_PRINT("Read [%s](%d) from File!\r\n", fReadBuf, fReadSize );
    else
        SYS_CMD_MESSAGE("Read failure!\r\n");
}

void CommandClose(SYS_CMD_DEVICE_NODE* pCmdIO, int argc, char** argv)
{
    SYS_FS_FileClose(fHandle);
    SYS_CMD_MESSAGE("File Closed!\r\n");
}

static const SYS_CMD_DESCRIPTOR MyCmdTbl[]=
{
    { "mount", CommandMount, ": Mount SD Card" },
    { "open", CommandOpen, ": open $" },
    { "write", CommandWrite, ": write $" },
    { "read", CommandRead, ": read #" },
    { "close", CommandClose, ": Close file" },
};

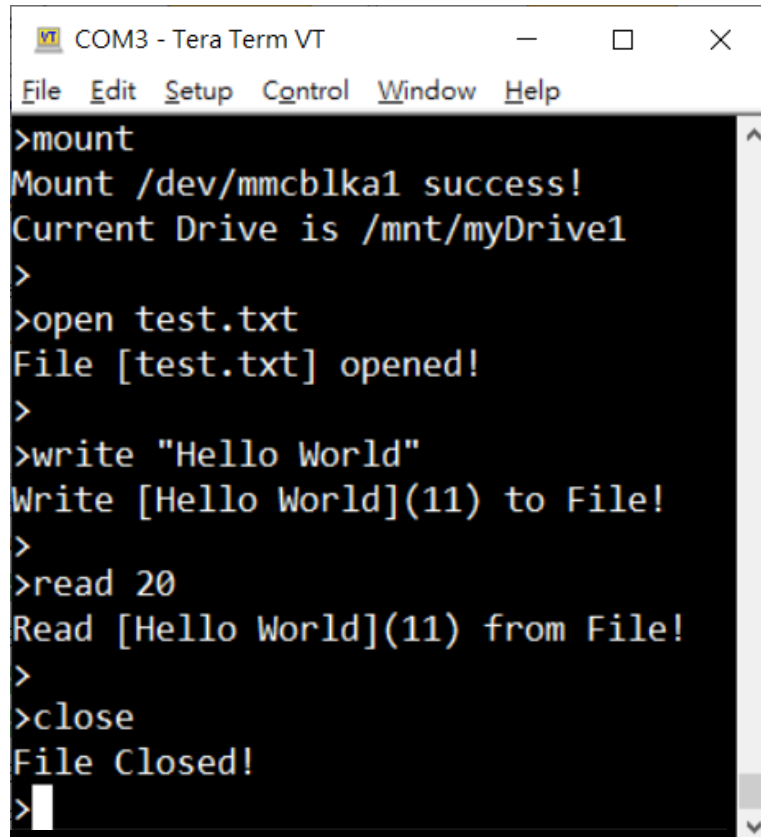
int main( void ) { ... }
```

Hello World



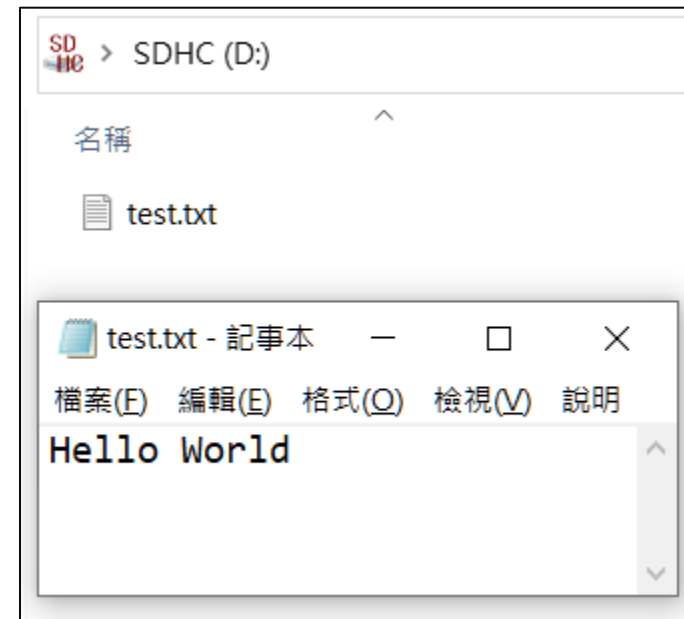
🔧 Lab7 SD Card with File System

- Compiler and programming your project to EVB.
- Run 'mount', 'open test.txt' and 'write "Hello World"' as previous.
- Type-in 'read 20' command to read 20 bytes from file.
- Type-in 'close' command to close file then check SD Card again.



```
COM3 - Tera Term VT
File Edit Setup Control Window Help
>mount
Mount /dev/mmcblk1 success!
Current Drive is /mnt/myDrive1
>
>open test.txt
File [test.txt] opened!
>
>write "Hello World"
Write [Hello World](11) to File!
>
>read 20
Read [Hello World](11) from File!
>
>close
File Closed!
>
```

"Hello World" in test.txt



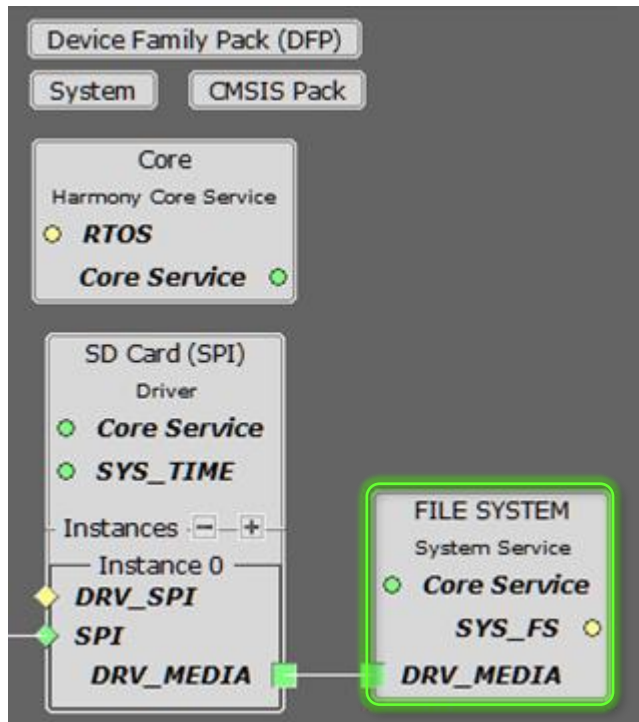
Lab 8

FILE System Auto Mount Feature

FILE System Service

⚙️ Lab8 FILE System Auto Mount Feature

- Harmony FILE system provide the **Auto Mount Feature**.
It's helpful to attach or detach media automatically.



The screenshot shows the 'Configuration Options' dialog for the 'FILE SYSTEM' component. The 'File System settings' section includes the following options:

- Maximum Simultaneous File Access: 1
- Size Of Block: 512
- Size Of Media Manager Buffer: 2,048
- Use File System Auto Mount Feature? ☒
- Total Number Of Media: 1
- Total Number File System Clients: 1

The 'Media0' section is expanded, showing the following configuration:

- Media configuration0
 - Media Type: SYS_FS_MEDIA_TYPE_SD_CARD
 - File System Type: FAT
 - Number Of Volumes: 1
- Volume0 ☒
 - Volume configuration0
 - Device Name: /dev/mmcblk1
 - Media Mount Name: /mnt/myDrive1

The 'File System Types' section at the bottom is set to 1.

⚙️ Lab8 FILE System Auto Mount Feature

- Code difference after enable the **Auto Mount Feature**.

Use File System Auto Mount Feature? ☐ **configuration.h**

```
#define SYS_FS_AUTOMOUNT_ENABLE false
```

Use File System Auto Mount Feature? ☒

```
#define SYS_FS_AUTOMOUNT_ENABLE true
#define SYS_FS_CLIENT_NUMBER 1U
...
#define SYS_FS_MEDIA_TYPE_IDX0 SYS_FS_MEDIA_TYPE_SD_CARD
#define SYS_FS_TYPE_IDX0 FAT
#define SYS_FS_MEDIA_IDX0_MOUNT_NAME_VOLUME_IDX0 "/mnt/myDrive1"
#define SYS_FS_MEDIA_IDX0_DEVICE_NAME_VOLUME_IDX0 "/dev/mmcblk1a1"
```

Use File System Auto Mount Feature? ☐ **initialization.c**

```
const SYS_FS_MEDIA_MOUNT_DATA sysfsMountTable[SYS_FS_VOLUME_NUMBER] =
{
    {NULL}
};
```

Use File System Auto Mount Feature? ☒

```
const SYS_FS_MEDIA_MOUNT_DATA sysfsMountTable[SYS_FS_VOLUME_NUMBER] =
{
    {
        .mountName = SYS_FS_MEDIA_IDX0_MOUNT_NAME_VOLUME_IDX0,
        .devName   = SYS_FS_MEDIA_IDX0_DEVICE_NAME_VOLUME_IDX0,
        .mediaType = SYS_FS_MEDIA_TYPE_IDX0,
        .fsType    = SYS_FS_TYPE_IDX0
    },
};
```

Lab8 FILE System Auto Mount Feature

- API is available when **SYS_FS_AUTOMOUNT_ENABLE** has set to **true**.

Use File System Auto Mount Feature? ☒

sys_fs.c

```
void SYS_FS_EventHandlerSet ( const void * eventHandler, const uintptr_t context )
{
    SYS_FS_MEDIA_MANAGER_EventHandlerSet( eventHandler, context );
}
```

- **Summary:**

Allows a client to identify an event handling function for the file system to call back when mount/unmount operation has completed.

- **Description:**

This function allows a client to identify an event handling function for the File System to call back when mount/unmount operation has completed.

The file system will pass mount name back to the client by calling "eventHandler" when AutoMount feature is enabled for File system.

- **Parameters:**

- | | |
|---------------------|---|
| eventHandler | - Pointer to the event handler function implemented by the user. |
| context | - The value of parameter will be passed back to the client unchanged, when the eventHandler function is called. It can be used to identify any client specific data object that identifies the instance of the client module (for example, it may be a pointer to the client module's state structure). |

✖ Lab8 FILE System Auto Mount Feature

- Modify your main.c as below

configuration.h

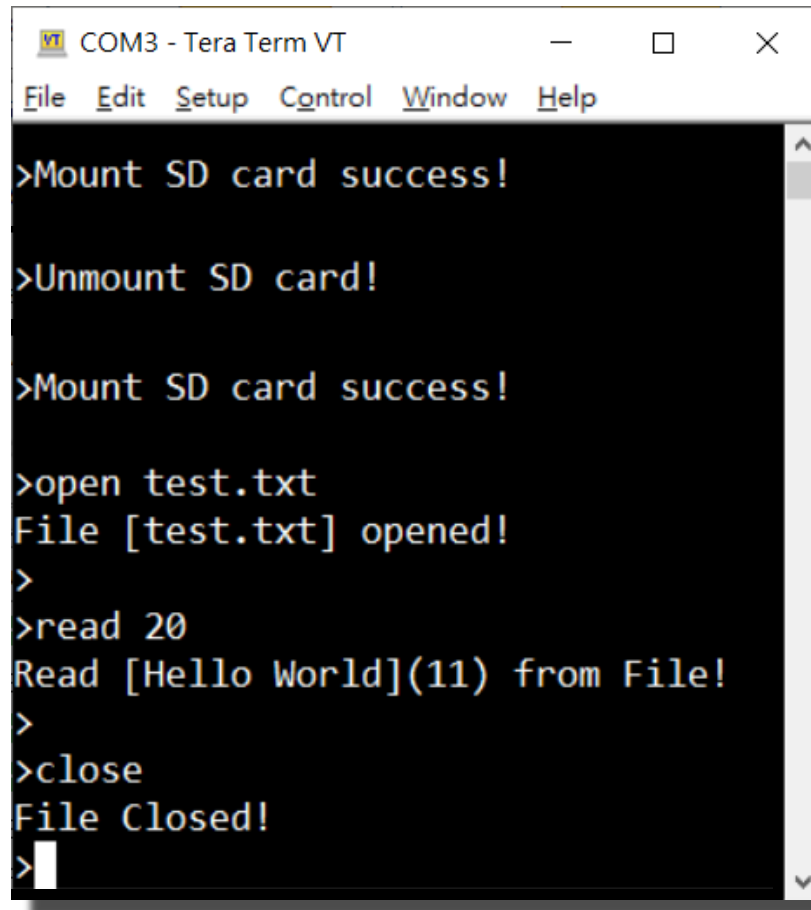
```
#define SYS_FS_MEDIA_IDX0_MOUNT_NAME_VOLUME_IDX0 "/mnt/myDrive1"  
#define SYS_FS_MEDIA_IDX0_DEVICE_NAME_VOLUME_IDX0 "/dev/mmcblk1a1"
```

main.c

```
void AutoMountEventHandler( SYS_FS_EVENT event, void* eventData, uintptr_t context )  
{  
    switch(event)  
    {  
        case SYS_FS_EVENT_MOUNT:  
        case SYS_FS_EVENT_MOUNT_WITH_NO_FILESYSTEM:  
            if(strcmp((const char *)eventData, SYS_FS_MEDIA_IDX0_MOUNT_NAME_VOLUME_IDX0) == 0)  
            {  
                SYS_CMD_MESSAGE("Mount SD card success!\r\n");  
            }  
            break;  
  
        case SYS_FS_EVENT_UNMOUNT:  
            if(strcmp((const char *)eventData, SYS_FS_MEDIA_IDX0_MOUNT_NAME_VOLUME_IDX0) == 0)  
            {  
                SYS_CMD_MESSAGE("Unmount SD card!\r\n");  
            }  
            break;  
  
        case SYS_FS_EVENT_ERROR:  
            break;  
    }  
}  
  
int main( void )  
{  
    SYS_Initialize(NULL);  
  
    SYS_FS_EventHandlerSet(AutoMountEventHandler, (uintptr_t)NULL);  
}
```

⚙️ Lab8 FILE System Auto Mount Feature

- Compiler and programming your project to EVB.
- **Remove SD card firstly.**
- **Put in** SD card and **Remove** it to check the output message.



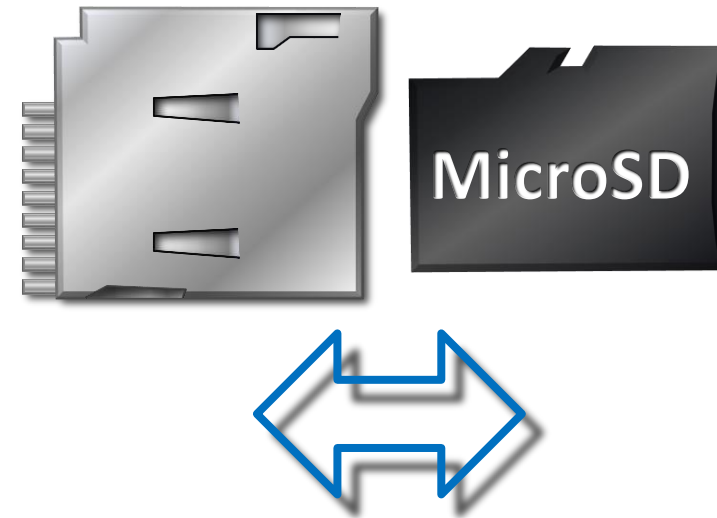
```
COM3 - Tera Term VT
File Edit Setup Control Window Help

>Mount SD card success!

>Unmount SD card!

>Mount SD card success!

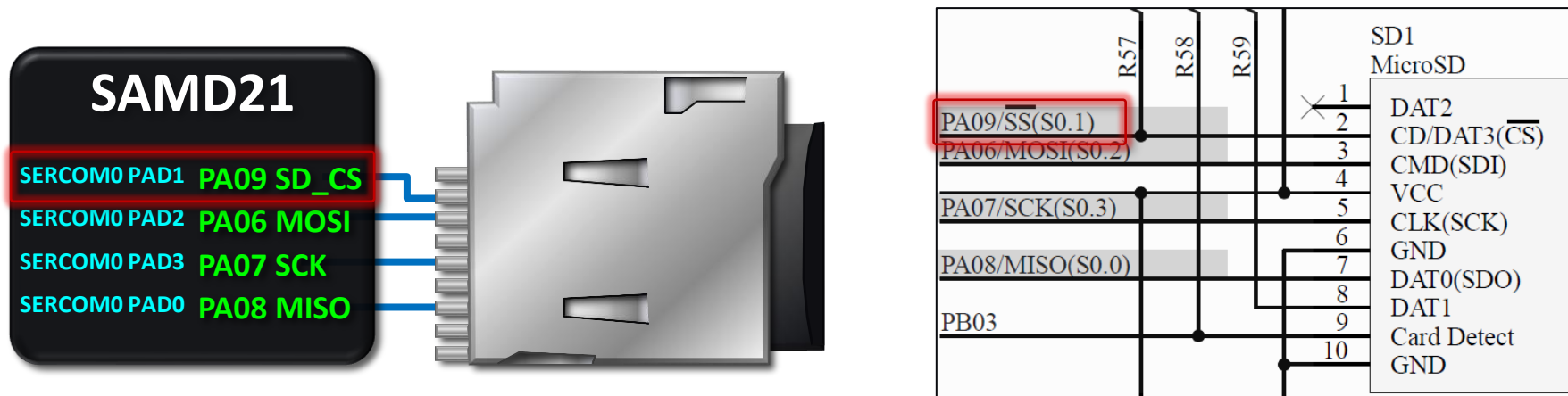
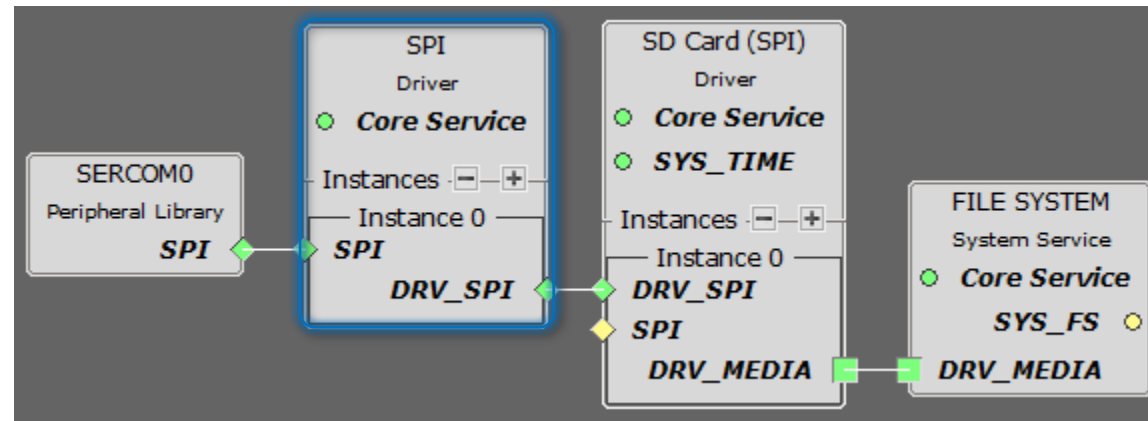
>open test.txt
File [test.txt] opened!
>
>read 20
Read [Hello World](11) from File!
>
>close
File Closed!
>
```



🔧 Lab8 FILE System Auto Mount Feature

Discuss 1

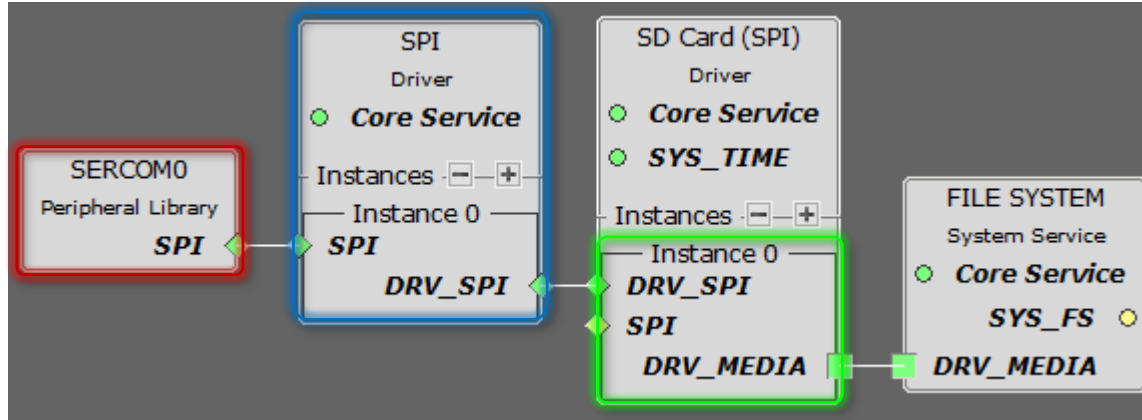
- How to use the SPI Master Hardware Slave Select (CS) but not use a GPIO to control ?
 - The SPI Master Hardware Slave Select need **SPI Driver** to support, and SPI SD Card must connect the **Chip Select** pin to a correct **SERCOM** pad.



⚙️ Lab8 FILE System Auto Mount Feature

Discuss 1

- The changes in modules if to use the SPI Master Hardware Slave Select (CS).



Enable SPI Master Hardware Slave Select in SERCOM

Keep default setting of Chip Select Pin In SD Card Driver

SERCOM0

Select SERCOM Operation Mode: **SPI Master**

Enable Interrupts ? ☒

Enable operation in Standby mode ☐

SPI Data Out Pad **DO**

SPI Data In Pad Selection **SEP**

SPI Data Order **MS**

SPI Speed in Hz

SPI Data Character Size **8 bit**

SPI Clock Phase **The**

SPI Clock Polarity **SCF**

Enable SPI Master Hardware Slave Select ☒

SPI Receiver Enable ☒

SPI Transfer Mode 0 is Selected

Enable DMA in SPI Driver

SPI

PLIB Used: **SERCOM0**

Number of Clients: **1**

Transfer Queue Size: **4**

Use DMA for Transmit and Receive? ☒

DMA Channel For Transmit: **0**

DMA Channel For Receive: **1**

Config PA09 to SERCOM0_PAD1

11	PA06		SERCOM0_PAD2	Digital
12	PA07		SERCOM0_PAD3	Digital
13	PA08		SERCOM0_PAD0	Digital
14	PA09		SERCOM0_PAD1	Digital

SD Card (SPI)

SPI Driver Instance Used: **0**

Number of Clients: **1**

SD Card Speed (Hz): **5,000,000**

Transfer Queue Size: **4**

Card Detection Method: **Use Polling**

Polling Interval (ms): **1,000**

File system for SDSPI Driver Enabled ☒

Chip Select Pin: **PA00**

!!! Configure the Chip Select pin as GPIO output under

Use Write Protect Pin (Active High)? ☐

!!! Note: For each instance of SDSPI, connect either SP

Lab 9

USB Host Mass Storage with File System

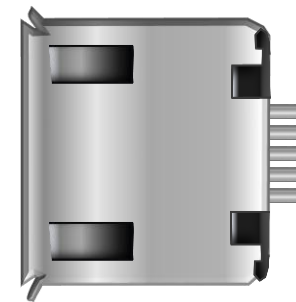
FILE System Service

✖ Lab9 USB Host Mass Storage with File System

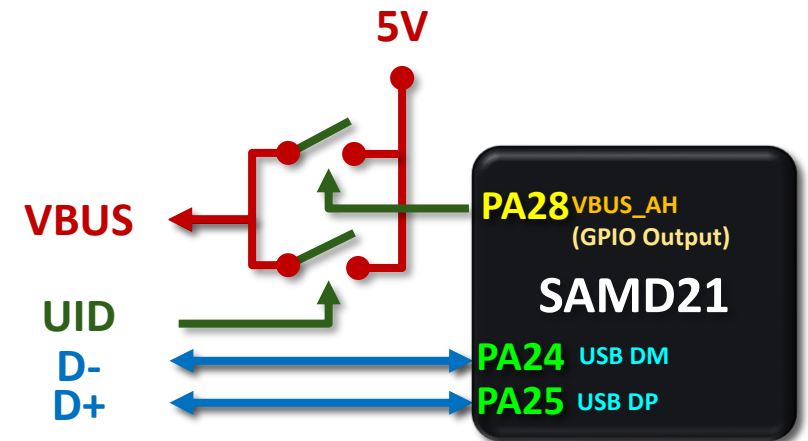
- This Lab to introduce how to use Harmony FILE system Auto Mount Feature to attach a USB pen drive in few steps.
- Try add the **USB MSD** as second media of FILE System Service .
- Configure VBUS voltage output control :
 - If use **USB OTG dongle**, the **UID pin** will tie to GND and VBUS will output.
 - If control by Driver, set **PA28** as GPIO output and named to **VBUS_AH**.
- Add USB MSD mount status in **Auto Mount Event Handler**.
- Build a command could change active media as below :
 - **media %** : Set active media to % for operation.
 - % = **SD** for SD card.
 - % = **USB** for USB pen drive.



USB OTG Dongle

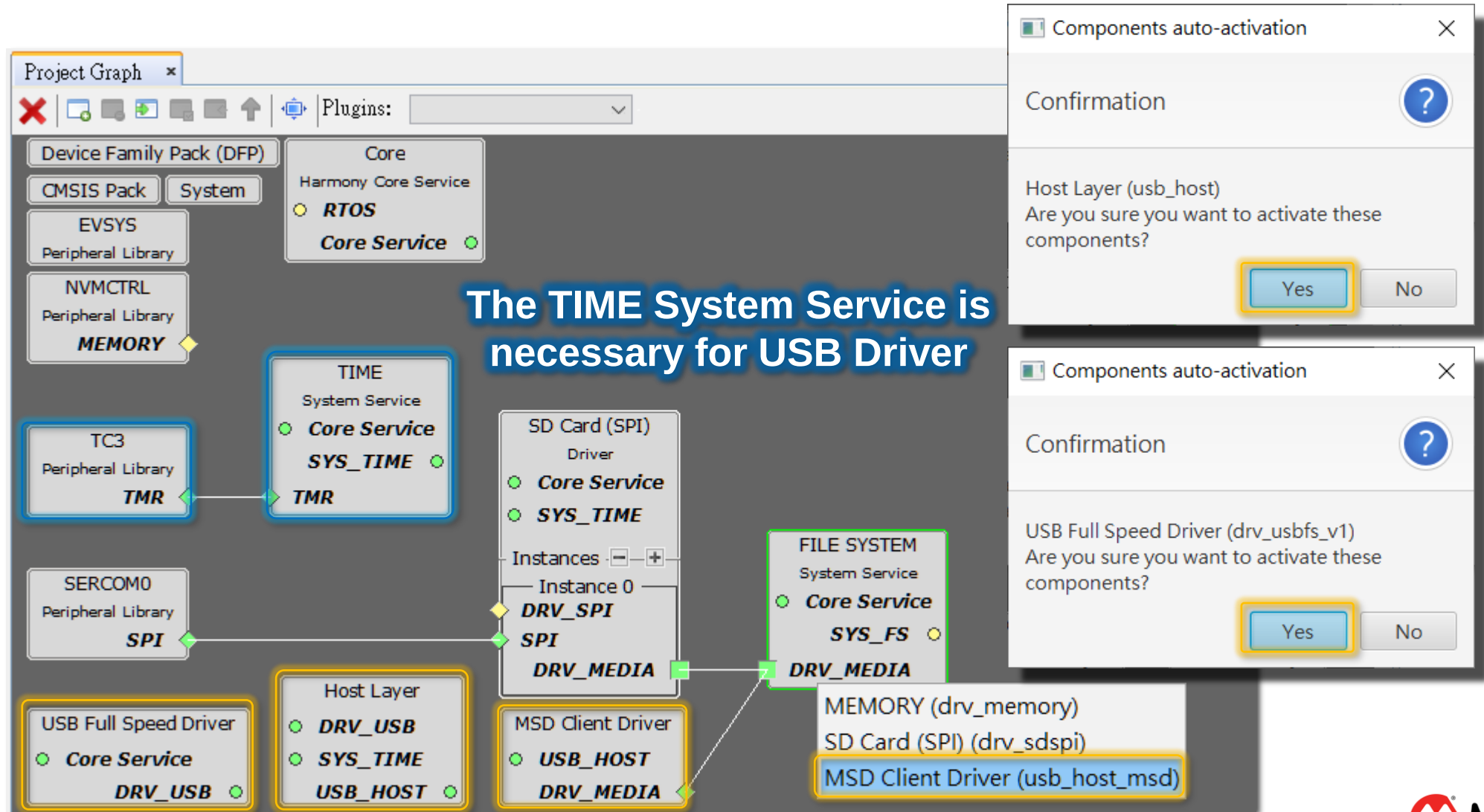


Native USB in Host role



✖ Lab9 USB Host Mass Storage with File System

- Attach media **USB MSD Client Driver** to **FILE System Service** .



✖ Lab9 USB Host Mass Storage with File System

- Configure **FILE SYSTEM** service for 2nd Media

The screenshot shows the 'Configuration Options' dialog for the 'FILE SYSTEM' service. The 'File System settings' section includes: 'Maximum Simultaneous File Access' (1), 'Size Of Block' (512), 'Size Of Media Manager Buffer' (2,048), 'Use File System Auto Mount Feature?' (checked), 'Total Number Of Media' (2), and 'Total Number File System Clients' (1). The 'Media0' section is expanded, showing 'Media1' with a checkmark. Under 'Media1', 'Media configuration1' is expanded, showing 'Media Type' set to 'SYS_FS_MEDIA_TYPE_MSD', 'File System Type' set to 'FAT', and 'Number Of Volumes' set to 1. Under 'Volume0', 'Volume configuration0' is expanded, showing 'Device Name' set to '/dev/sda1' and 'Media Mount Name' set to '/mnt/myDrive2'. The 'File System Types' section at the bottom shows 'FAT File System' checked.

Use Auto Mount Feature

Increase Number of Media

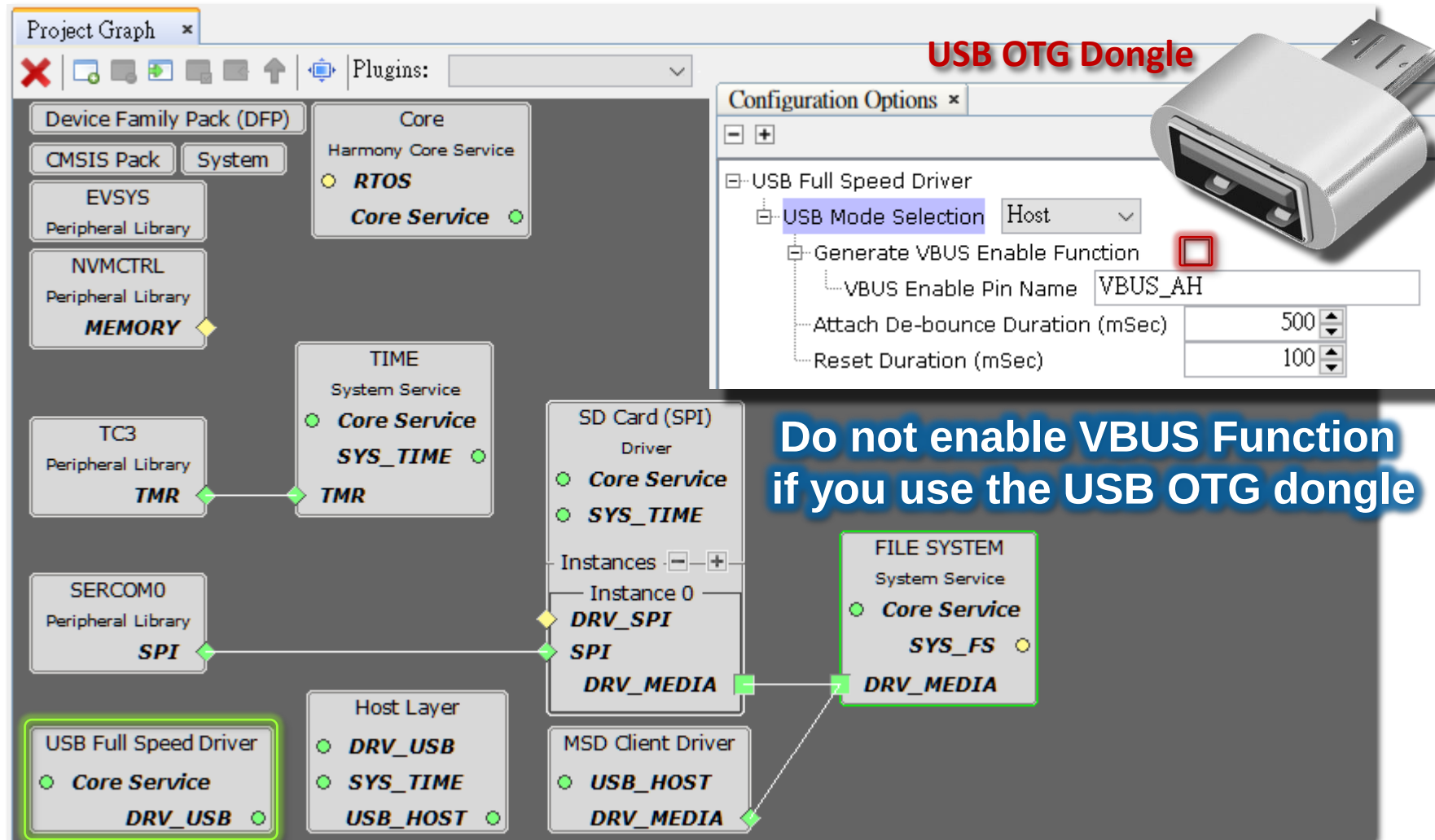
Choose 2nd Media Type to MSD
Device Name will be /dev/sda1

Modify Mount Name to /mnt/myDrive2

FILE SYSTEM
System Service
● Core Service
SYS_FS
DRV_MEDIA

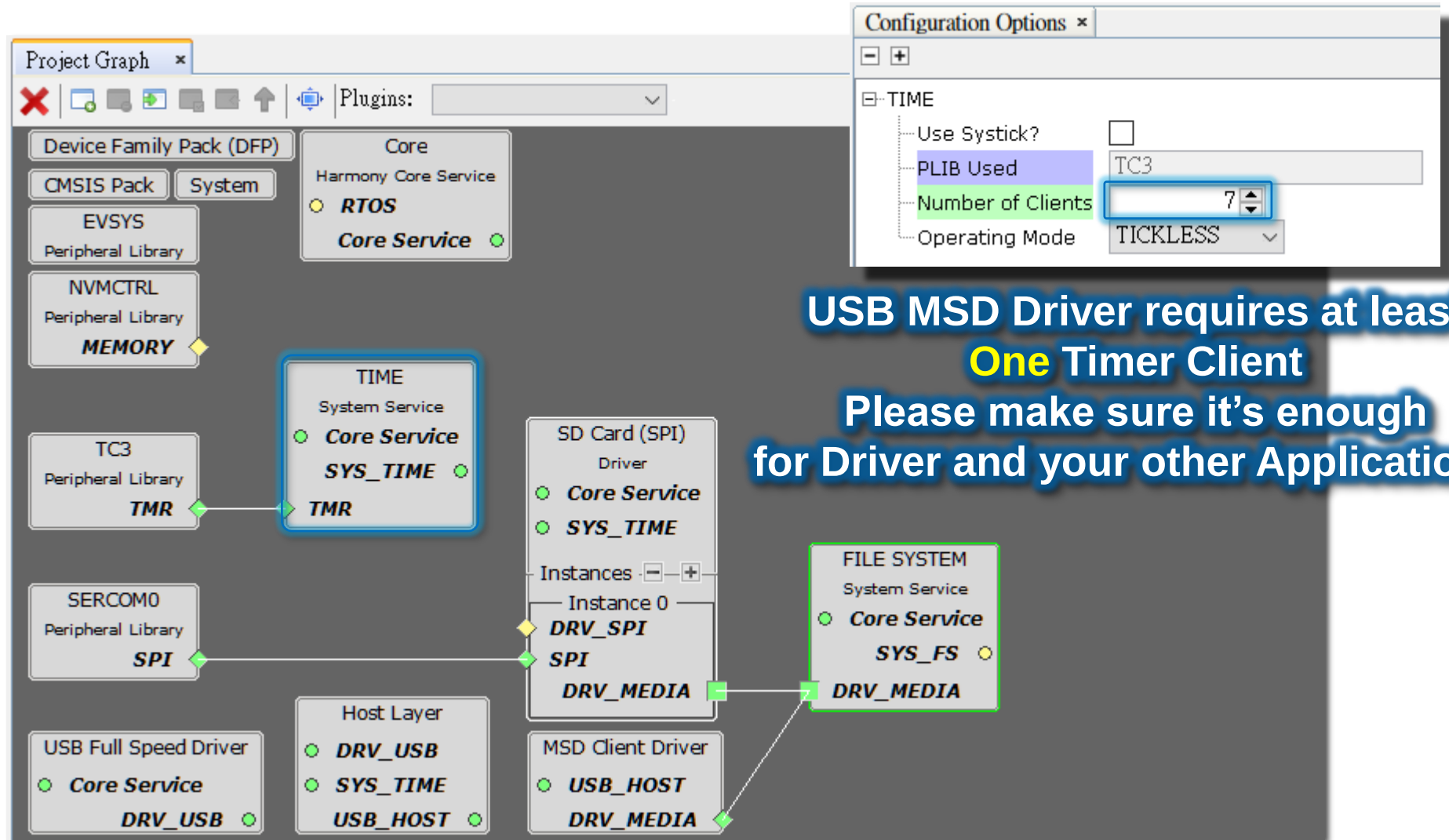
✖ Lab9 USB Host Mass Storage with File System

- Configure **USB Full Speed Driver** **without** VBUS output control.



⚙️ Lab9 USB Host Mass Storage with File System

- Configure **TIME** System Service.



**USB MSD Driver requires at least
One Timer Client
Please make sure it's enough
for Driver and your other Applications.**

✖ Lab9 USB Host Mass Storage with File System

- Configure USB D-/D+ on PA24/PA25
- Do not configure the VBUS_AH pin if no VBUS Enable Function.

Project Graph* Pin Diagram x Pin Table x **Pin Settings x**

Order: Ports v Table View ☒ Easy View

Pin Number	Pin ID	Custom Name	Function	Mode	Direction	Latch	Full Op	Full	Drive
33	PA24		USB_DM v	Digital	High Impedance v	n/a	☐	☐	NORMAL v
34	PA25		USB_DP v	Digital	High Impedance v	n/a	☐	☐	NORMAL v

USB Full Speed Driver

USB Mode Selection Host v

Generate VBUS Enable Function ☐

VBUS Enable Pin Name VBUS_AH

Attach De-bounce Duration (mSec) 500

Reset Duration (mSec) 100

Project Graph* Pin Diagram x **Pin Table x** Pin Settings x

Package: TQFP48 v

Module	Function	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	27	28	30	31	2	3
USB	USB_DM																							
	USB_DP																							
	USB_SOF_1KHZ																							

✖ Lab9 USB Host Mass Storage with File System

- Add USB MSD mount status in Auto Mount Event Handler.

```
static const SYS_CMD_DESCRIPTOR MyCmdTbl[] = { ... };

void AutoMountEventHandler( SYS_FS_EVENT event, void* eventData, uintptr_t context )
{
    switch(event)
    {
        case SYS_FS_EVENT_MOUNT:
            if(strcmp((const char *)eventData, SYS_FS_MEDIA_IDX0_MOUNT_NAME_VOLUME_IDX0) == 0)
            {
                SYS_CMD_MESSAGE("Mount SD card success!\r\n");
            }
            if(strcmp((const char *)eventData, SYS_FS_MEDIA_IDX1_MOUNT_NAME_VOLUME_IDX0) == 0)
            {
                SYS_CMD_MESSAGE("Mount USB Mass Storage success!\r\n");
            }
            break;

        case SYS_FS_EVENT_UNMOUNT:
            if(strcmp((const char *)eventData, SYS_FS_MEDIA_IDX0_MOUNT_NAME_VOLUME_IDX0) == 0)
            {
                SYS_CMD_MESSAGE("Unmount SD card!\r\n");
            }
            if(strcmp((const char *)eventData, SYS_FS_MEDIA_IDX1_MOUNT_NAME_VOLUME_IDX0) == 0)
            {
                SYS_CMD_MESSAGE("Unmount USB Mass Storage!\r\n");
            }
            break;

        case SYS_FS_EVENT_ERROR:
            break;
    }
}
```

✖ Lab9 USB Host Mass Storage with File System

- Add a new command “**media**” to select active Drive of Media.

```
void CommandClose(SYS_CMD_DEVICE_NODE* pCmdIO, int argc, char** argv) { ... }

void CommandMedia(SYS_CMD_DEVICE_NODE* pCmdIO, int argc, char** argv)
{
    if(!strcmp( argv[1], "SD"))
    {
        strcpy(fMedia, SYS_FS_MEDIA_IDX0_DEVICE_NAME_VOLUME_IDX0);
        strcpy(fDrive, SYS_FS_MEDIA_IDX0_MOUNT_NAME_VOLUME_IDX0);
    }
    if(!strcmp( argv[1], "USB"))
    {
        strcpy(fMedia, SYS_FS_MEDIA_IDX1_DEVICE_NAME_VOLUME_IDX0);
        strcpy(fDrive, SYS_FS_MEDIA_IDX1_MOUNT_NAME_VOLUME_IDX0);
    }

    SYS_CMD_PRINT("Set media to %s\r\n", fMedia);

    if(SYS_FS_CurrentDriveSet(fDrive) != SYS_FS_RES_SUCCESS)
    {
        SYS_CMD_PRINT("Drive %s not mounted!\r\n", fDrive);
        return;
    }

    SYS_CMD_PRINT("Set drive to %s\r\n", fDrive);
}

static const SYS_CMD_DESCRIPTOR MyCmdTbl[]=
{
    { "close", CommandClose, ": Close file" },
    { "media", CommandMedia, ": media %" },
};

void AutoMountEventHandler( SYS_FS_EVENT event,void* eventData, uintptr_t context ) { ... }
```

⚙️ Lab9 USB Host Mass Storage with File System

- Register the Host Event Handler and **enable USB Host Bus**.

```
void AutoMountEventHandler( SYS_FS_EVENT event,void* eventData, uintptr_t context ) { ... }

USB_HOST_EVENT_RESPONSE USBHostEventHandler(USB_HOST_EVENT event,
                                             void * eventData, uintptr_t context)
{
    switch (event)
    {
        case USB_HOST_EVENT_DEVICE_REJECTED_INSUFFICIENT_POWER: break;
        case USB_HOST_EVENT_DEVICE_UNSUPPORTED: break;
        case USB_HOST_EVENT_HUB_TIER_LEVEL_EXCEEDED: break;
        case USB_HOST_EVENT_PORT_OVERCURRENT_DETECTED: break;
    }

    return(USB_HOST_EVENT_RESPONSE_NONE);
}

int main( void )
{
    SYS_Initialize(NULL);

    SYS_FS_EventHandlerSet(AutoMountEventHandler, (uintptr_t)NULL);

    SYS_CMD_ADDGRP(MyCmdTbl, sizeof(MyCmdTbl)/sizeof(SYS_CMD_DESCRIPTOR),
                  "MYCMD", ": My Commands");

    USB_HOST_EventHandlerSet(USBHostEventHandler, (uintptr_t)NULL);

    USB_HOST_BusEnable(0);

    while( true ) { ... }

    ...
}
```

⚙️ Lab9 USB Host Mass Storage with File System

- Remove SD card and USB pen drive firstly.
- Compiler and programming your project to EVB.
- The **USB Host Layer** debug message will output after reset, that is because the **SYS_DEBUG** been used in USB stack and the error level is **SYS_ERROR_INFO (3)**.
- We could execute the command **"errlv 2"** to stop USB stack debug message.

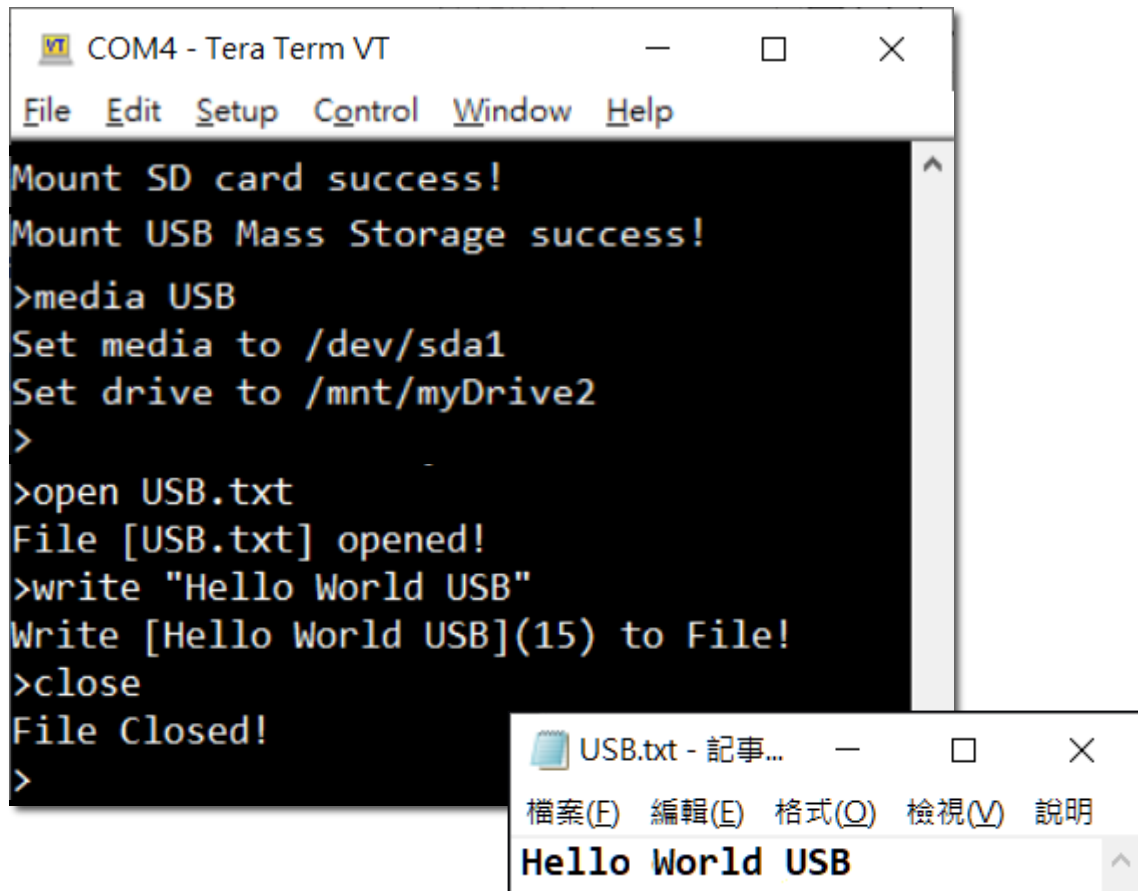
```
SYS_MODULE_OBJ USB_HOST_Initialize(const SYS_MODULE_INIT * initData )  
{  
    ...  
    result =SYS_MODULE_OBJ_INVALID;  
    SYS_DEBUG_MESSAGE(SYS_ERROR_INFO, "\r\nUSB Host Layer: Entering USB_HOST_Initialize().");  
}
```

usb_host.c

```
COM4 - Tera Term VT  
File Edit Setup Control Window Help  
USB Host Layer: Entering USB_HOST_Initialize().  
USB Host Layer: Exiting USB_HOST_Initialize() successfully.  
USB Host Layer: Bus 0 About to Open Root Hub Driver.  
USB Host Layer: Bus 0 Root Hub Driver Opened.  
USB Host Layer: Bus 0 Enabling Root Hub operation  
>  
>errlv 2  
Change Error Level to 2  
Error Level 0 - Fatal  
Error Level 1 - Error  
Error Level 2 - Warning  
>
```

✖ Lab9 USB Host Mass Storage with File System

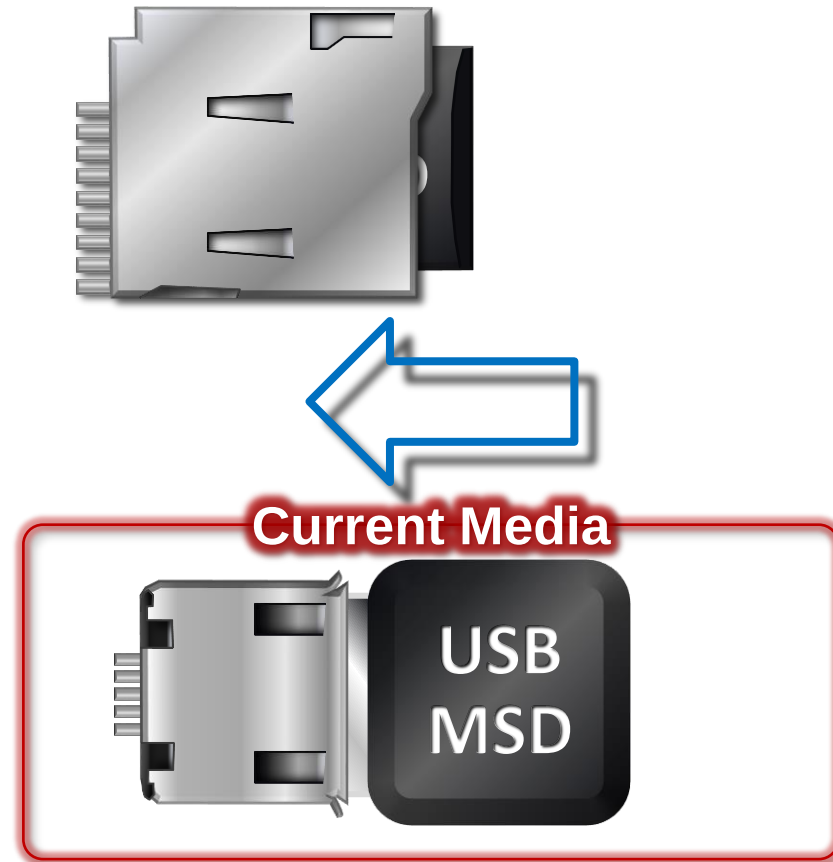
- Put in SD card and USB pen drive individually.
- Command “**media USB**” change active media to USB pen drive.
- Try file operation as below and check content in USB pen drive.



The image shows a Tera Term VT terminal window titled 'COM4 - Tera Term VT' with a menu bar (File, Edit, Setup, Control, Window, Help). The terminal output is as follows:

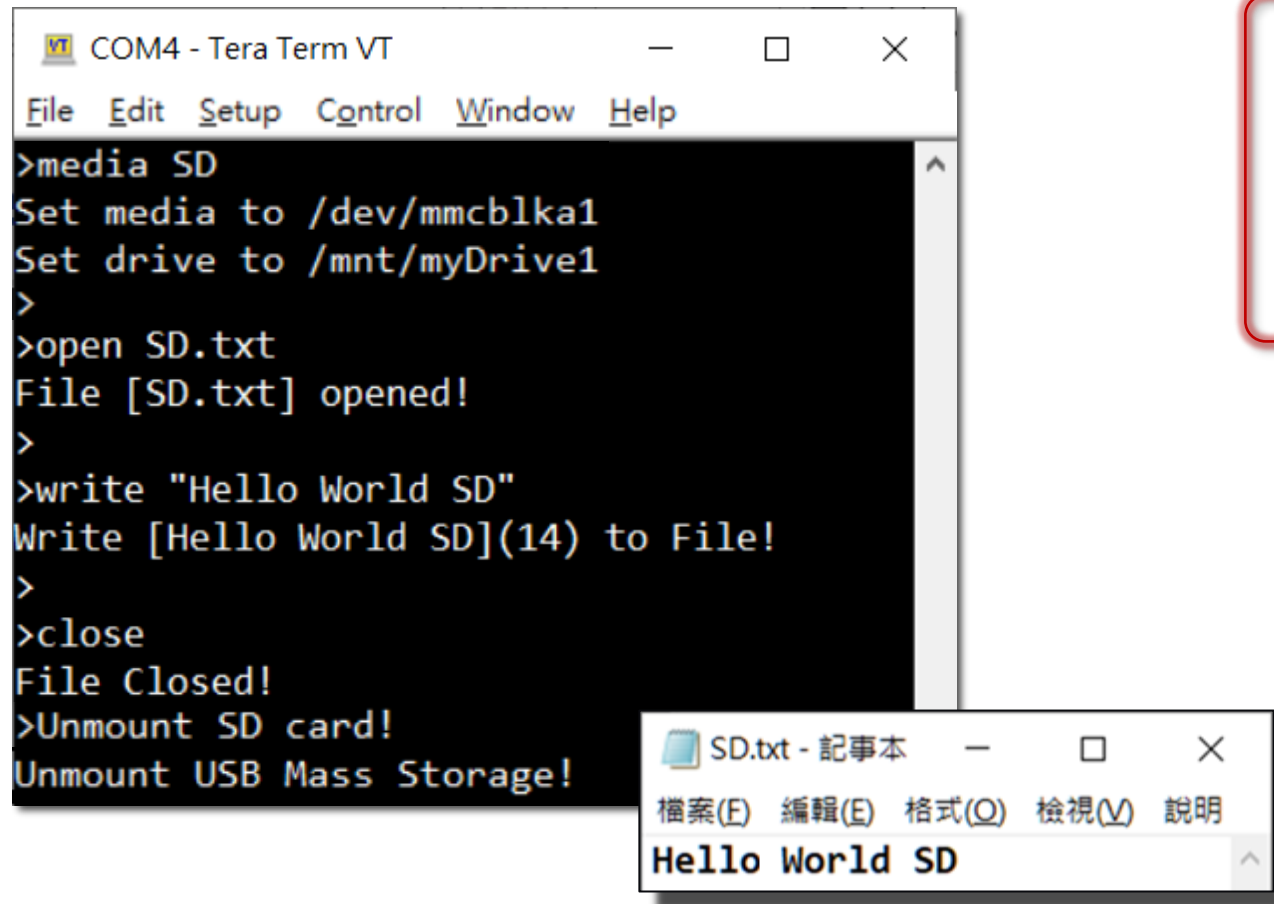
```
Mount SD card success!
Mount USB Mass Storage success!
>media USB
Set media to /dev/sda1
Set drive to /mnt/myDrive2
>
>open USB.txt
File [USB.txt] opened!
>write "Hello World USB"
Write [Hello World USB](15) to File!
>close
File Closed!
>
```

Below the terminal window is a text editor window titled 'USB.txt - 記事...' with a menu bar (檔案(E), 編輯(E), 格式(O), 檢視(V), 說明). The text 'Hello World USB' is visible in the editor.



✖ Lab9 USB Host Mass Storage with File System

- Command “**media SD**” to change active media to SD Card.
- Try file operation as below and check the content in SD card.
- **Remove SD card and USB pen drive individually.**

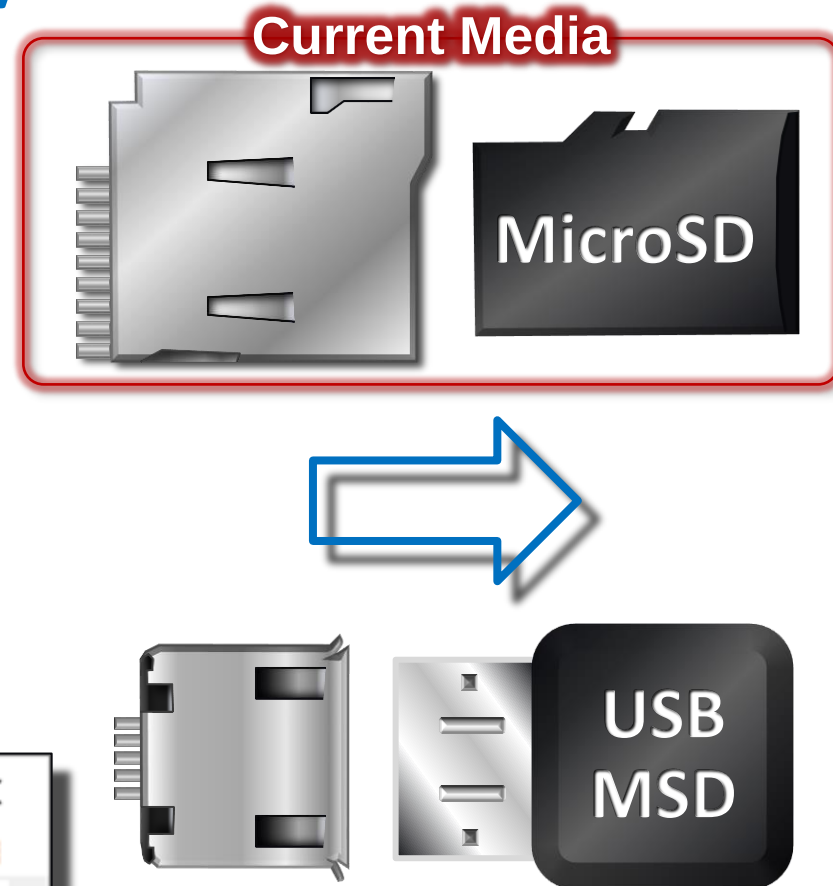


The image shows a Tera Term VT terminal window titled 'COM4 - Tera Term VT' with a menu bar (File, Edit, Setup, Control, Window, Help). The terminal output is as follows:

```
>media SD
Set media to /dev/mmcblk1
Set drive to /mnt/myDrive1
>
>open SD.txt
File [SD.txt] opened!
>
>write "Hello World SD"
Write [Hello World SD](14) to File!
>
>close
File Closed!
>Unmount SD card!
Unmount USB Mass Storage!
```

Below the terminal window is a Notepad window titled 'SD.txt - 記事本' with a menu bar (檔案(E), 編輯(E), 格式(O), 檢視(V), 說明). The text in the Notepad window is:

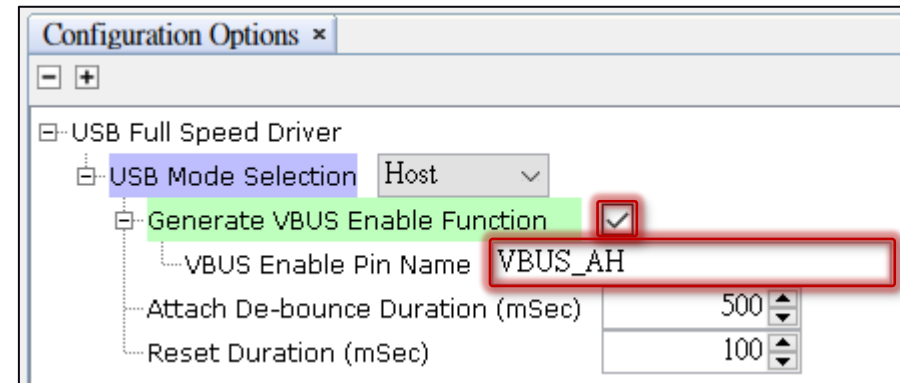
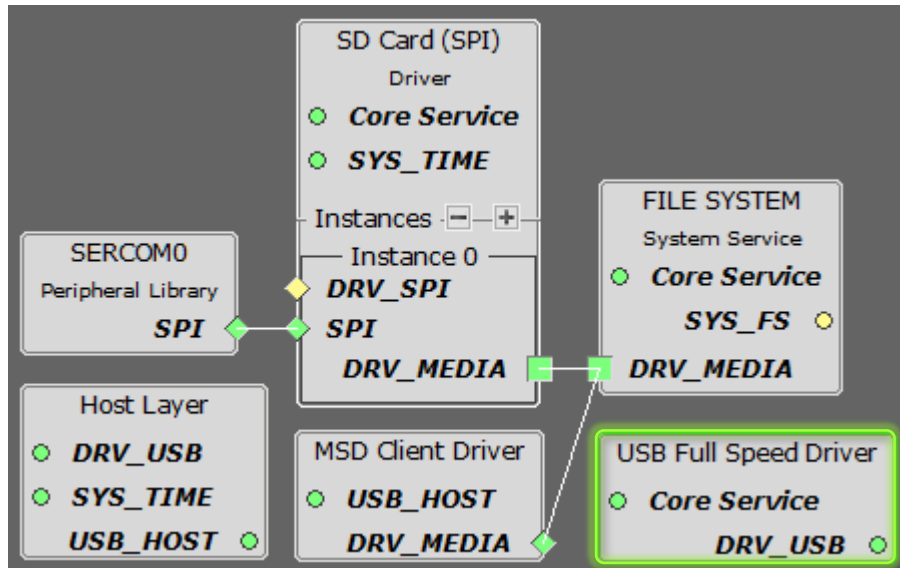
```
Hello World SD
```



✖ Lab9 USB Host Mass Storage with File System

Discuss 1

- How about if generate VBUS Enable Function in **USB Full Speed Driver** for VBUS output control by USB driver ?



✖ Lab9 USB Host Mass Storage with File System

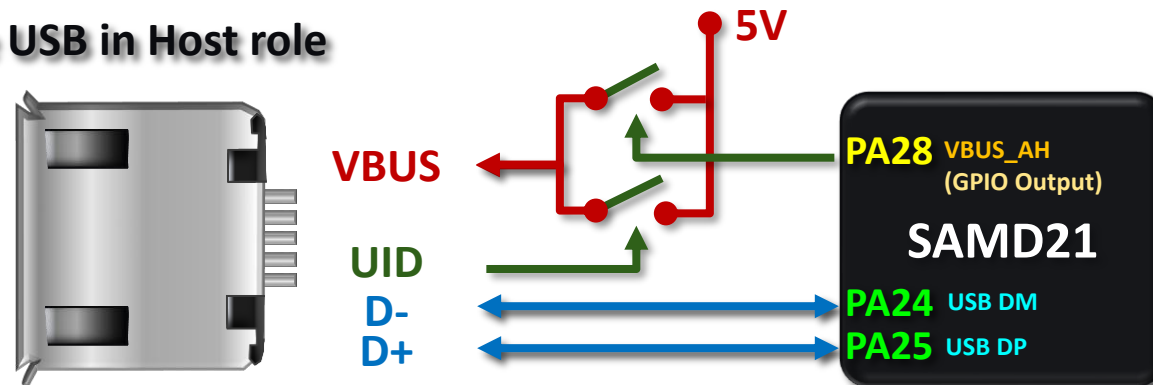
Discuss 1

- The VBUS control pin on GPIO **PA28**, Custom Name to **VBUS_AH** as the name you set in previous page, direction is **Output** and Latch to **High**. (VBUS output is **Low** active)
- Configure this only when you to generate VBUS Enable Function.**

The screenshot shows the 'Pin Settings' table and the 'USB Full Speed Driver' configuration window. The table has columns: Pin Number, Pin ID, Custom Name, Function, and Mode. The row for Pin 41 (PA28) is highlighted with a red box, showing Custom Name 'VBUS_AH', Function 'GPIO', and Mode 'Out' with 'High' selected. The configuration window on the right shows 'USB Mode Selection' set to 'Host', 'Generate VBUS Enable Function' checked, 'VBUS Enable Pin Name' set to 'VBUS_AH', 'Attach De-bounce Duration (mSec)' set to 500, and 'Reset Duration (mSec)' set to 100. Red arrows indicate the connection between the pin settings and the configuration window.

Pin Number	Pin ID	Custom Name	Function	Mode
33	PA24		USB_DM	Digital
34	PA25		USB_DP	Digital
41	PA28	VBUS_AH	GPIO	Out

Native USB in Host role



✂ Lab9 USB Host Mass Storage with File System

Discuss 1

- Add two VBUS power control functions in \Source Files\config\default\initialization.c , That are **VBUS_AH_PowerEnable()** and **VBUS_AH_PowerDisable()**.

```
/* *****  
 * USB Driver Initialization  
 * ***** */  
void VBUS_AH_PowerEnable( void )  
{  
    VBUS_AH_Clear();  
}  
  
void VBUS_AH_PowerDisable( void )  
{  
    VBUS_AH_Set();  
}  
  
void _DRV_USB_VBUSPowerEnable(uint8_t port, bool enable)  
{  
    /* Note: When operating in Host mode, the application can specify a Root  
    ...  
    used for enabling VBUS */  
    if (enable == true)  
    {  
        /* Enable the VBUS */  
        VBUS_AH_PowerEnable();  
    }  
    else  
    {  
        /* Disable the VBUS */  
        VBUS_AH_PowerDisable();  
    }  
}
```

initialization.c

USB Full Speed Driver

USB Mode Selection: Host

Generate VBUS Enable Function: ☒

VBUS Enable Pin Name: VBUS_AH

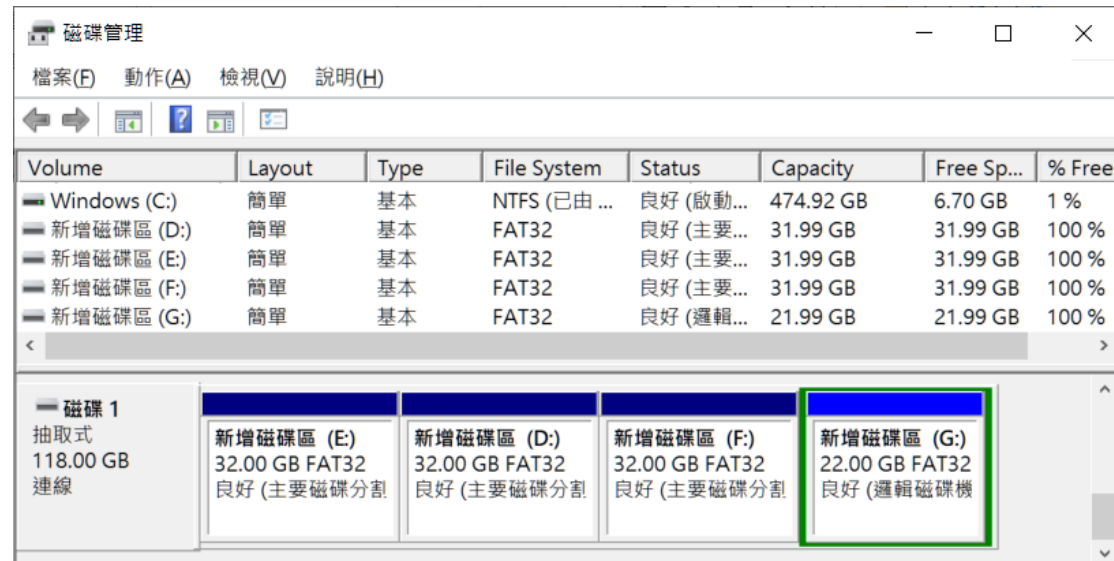
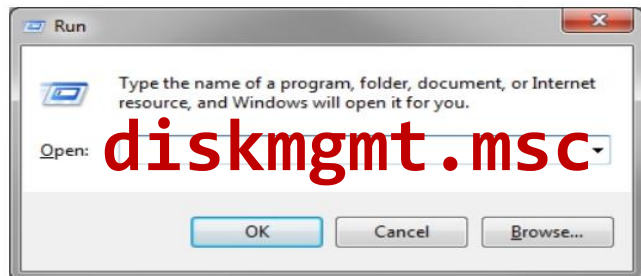
Attach De-bounce Duration (mSec): 500

Reset Duration (mSec): 100

✖ Lab9 USB Host Mass Storage with File System

Discuss 2

- How to use SD Card / USB pen drive larger than 32GB under Windows ?
 - Windows operating systems create new **FAT32** volumes only **up to 32 GB** in size. You could use Disk Manager to partition the storage to several 32GB drives.
 - microchipdeveloper.com/faq:999
Currently, the **MPLAB® Harmony** implementation of the FAT File System has been **tested up to 32 GB** SD cards. The limit for the file system is **2 TB at 512 byte/sector**. At this time, the upper size limit has not been tested.



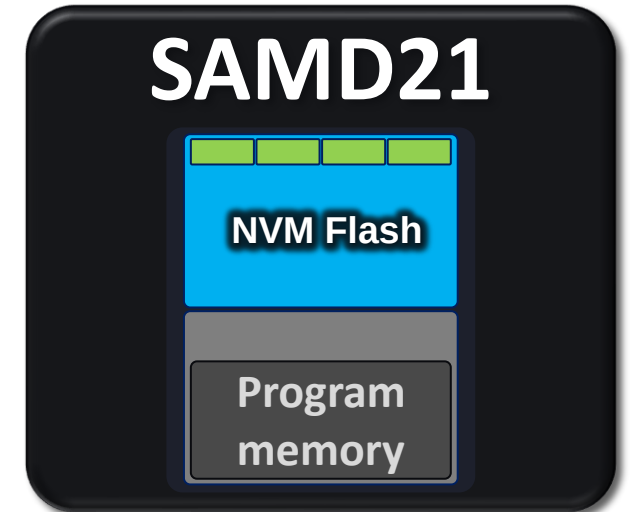
Lab 10

NVMCTRL Internal Flash with File System

FILE System Service

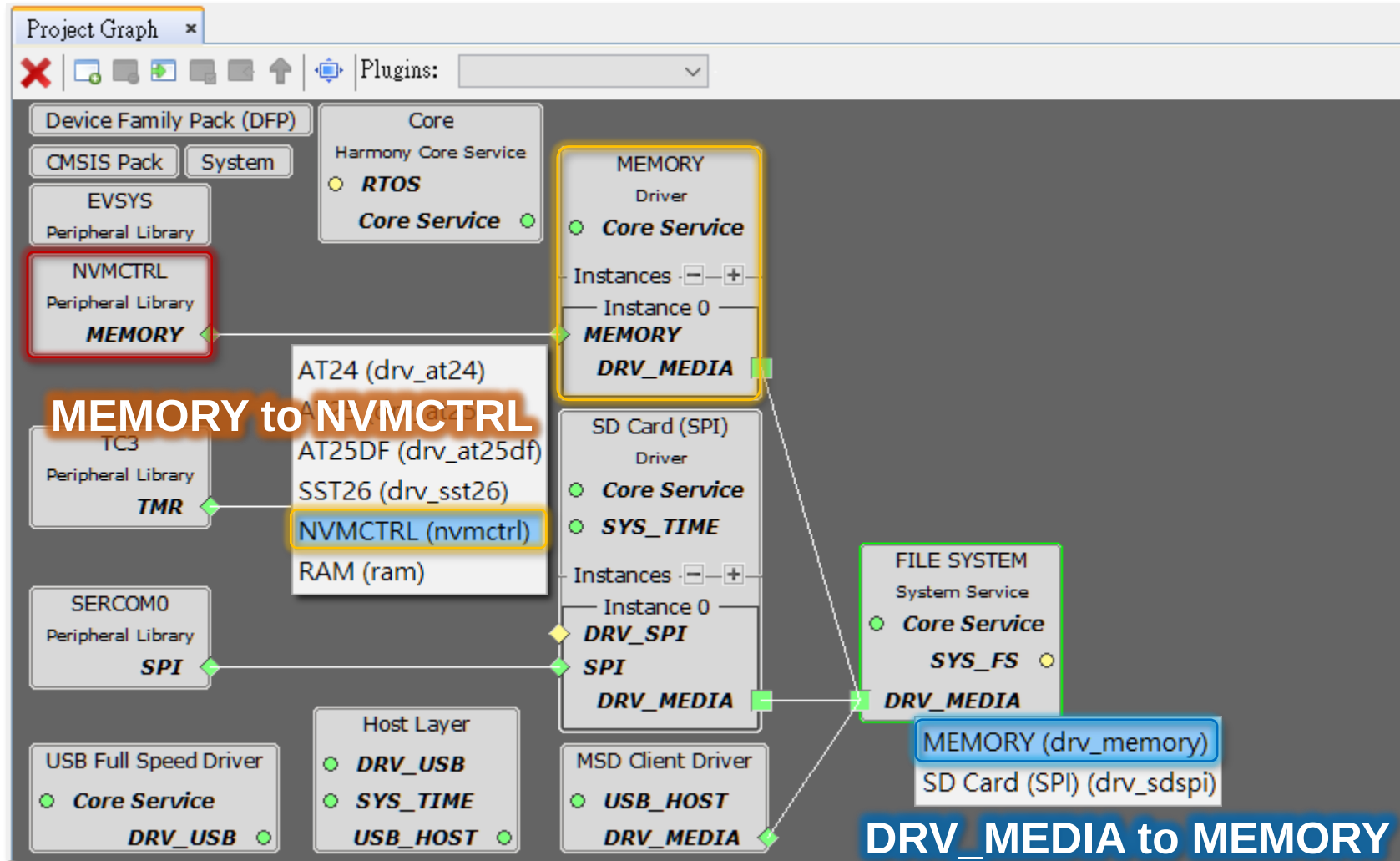
⚙️ Lab10 NVMCTRL Internal Flash with File System

- This Lab to introduce how to use Harmony FILE system to attach the internal NVM Flash as a FILE system drive.
- Try add the **MEMORY** driver as 3rd media of **FILE System Service** .
- Choose **NVMCTRL** as satisfiers of **MEMORY** driver.
- Add NVM flash mount status in Auto Mount Event Handler.
- Modify command “**media**” to support media NVM Flash :
media % : Set active media to % for operation.
 - % = SD for SD card.
 - % = USB for USB pen drive.
 - % = **NVM** for NVM flash.
- Build a new command could format active media as below :
format : **Format current media in default FAT configuration.**



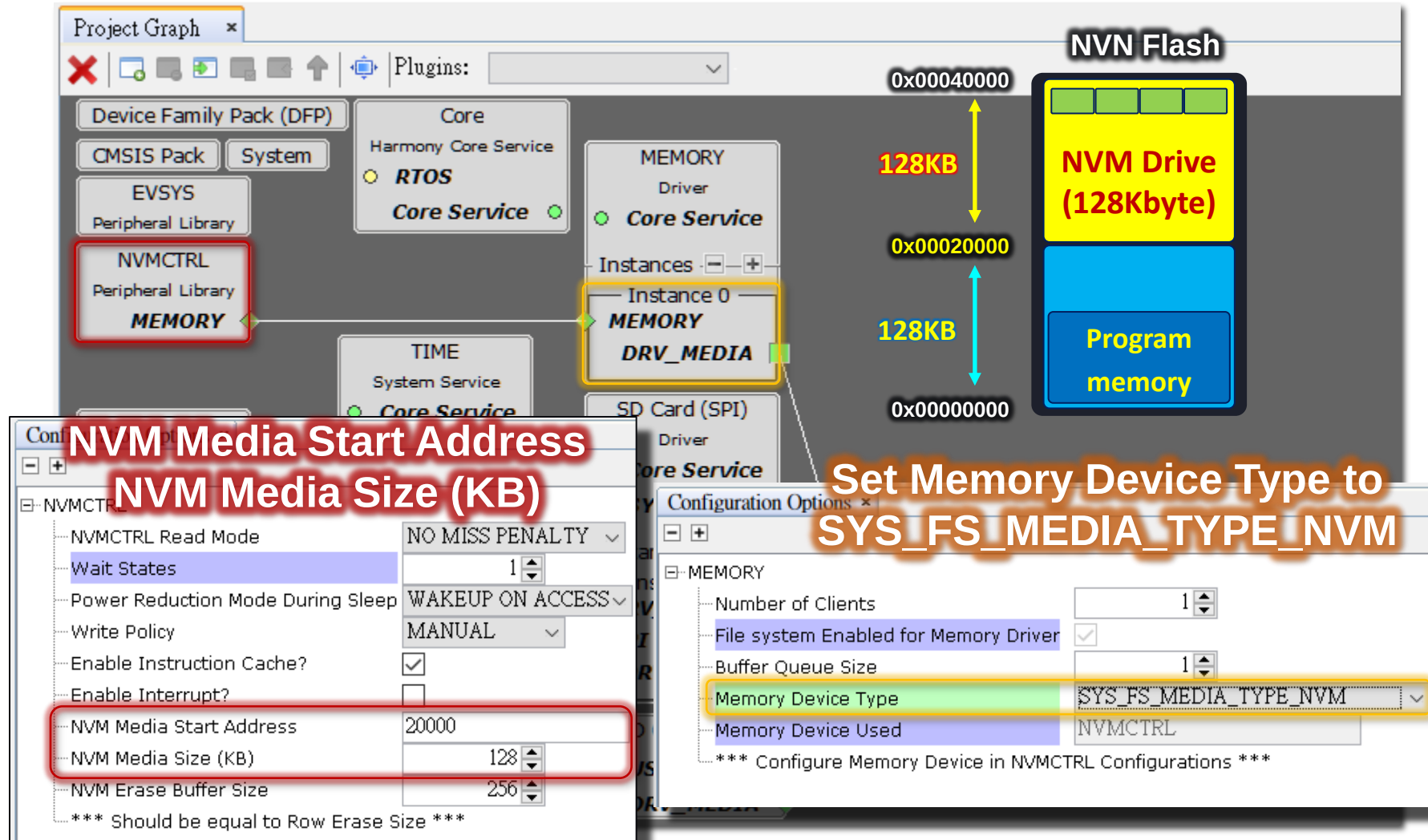
⚙️ Lab10 NVMCTRL Internal Flash with File System

- Attach **MEMORY Driver** + **PLIB NVMCTRL** to **FILE System Service** .



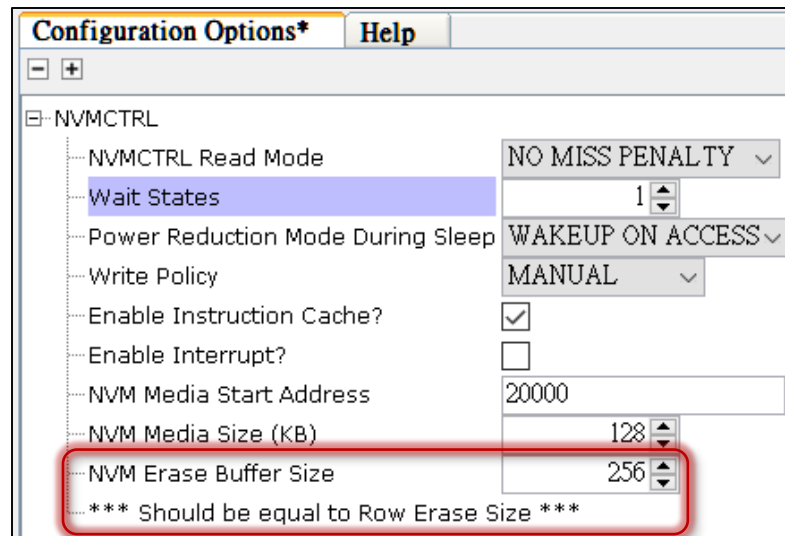
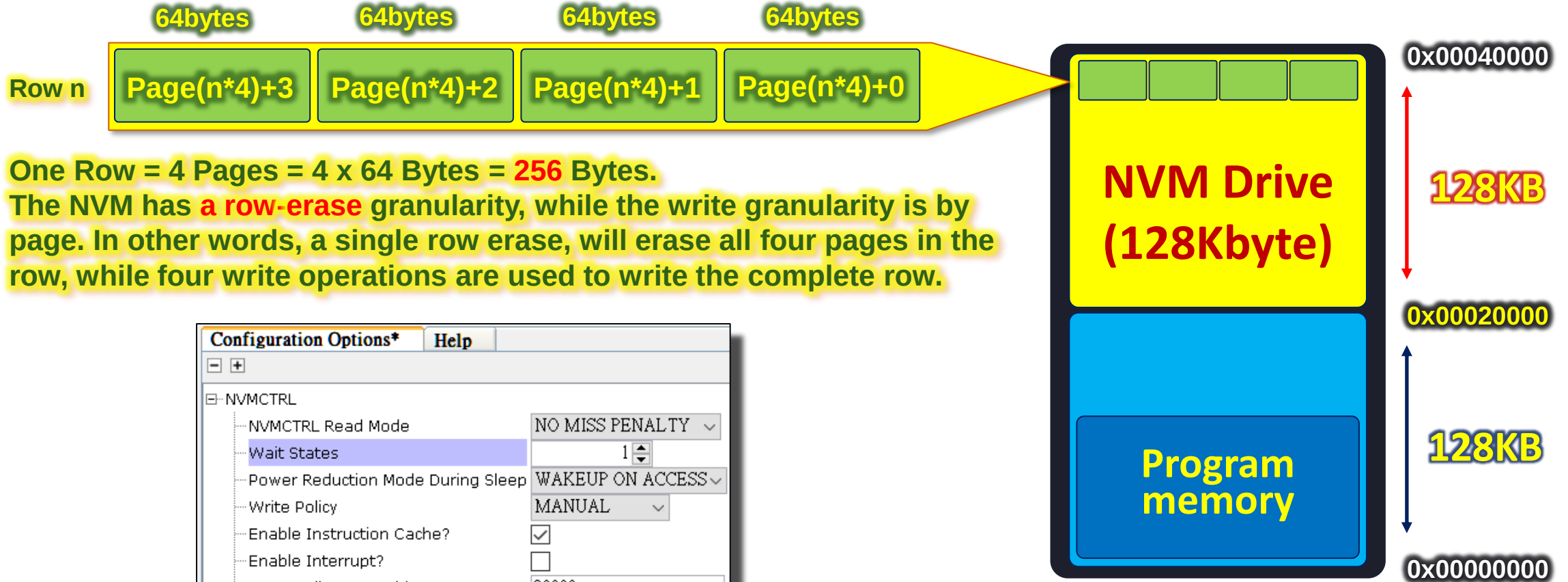
⚙️ Lab10 NVMMCTRL Internal Flash with File System

- Configuration of **PLIB NVMMCTRL** and **Instance of MEMORY Driver**



⚙️ Lab10 NVMCTRL Internal Flash with File System

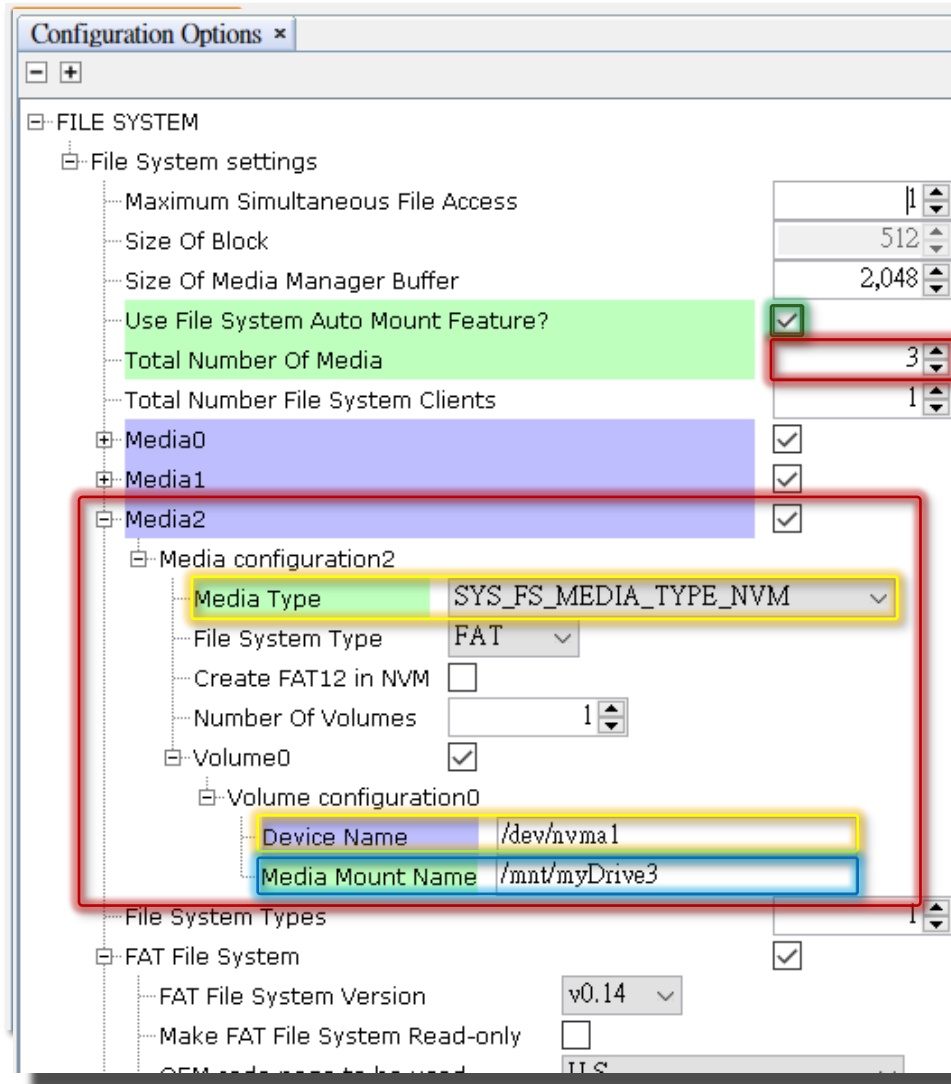
- SAMD21G18A has 256KB flash, the page size is 64 Bytes then total have 4096 pages.
- The NVM Erase Buffer Size should be equal to Row Erase Size of memory organization.



Erase buffer Size should be equal to 256 bytes

⚙️ Lab10 NVMMCTRL Internal Flash with File System

- Configure **FILE SYSTEM** service for 3rd Media

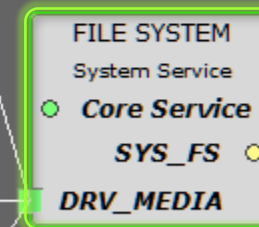


Use Auto Mount Feature

Increase Number of Media

Choose 3rd Media Type to NVM
Device Name will be
/dev/nvma1

Modify Mount Name to
/mnt/myDrive3



✖ Lab10 NVMCTRL Internal Flash with File System

- Let's check current Volume configuration of all Media.

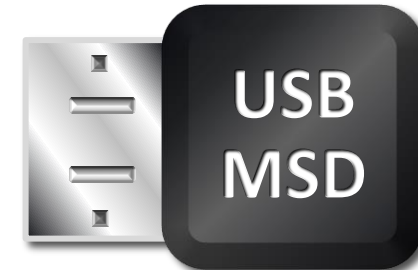
The screenshot shows the NVMCTRL configuration tool with three media entries:

- Media0** (checked):
 - Media configuration0
 - Media Type: SYS_FS_MEDIA_TYPE_SD_CARD
 - File System Type: FAT
 - Number Of Volumes: 1
 - Volume0 (checked)
 - Volume configuration0
 - Device Name: /dev/mmcblk1
 - Media Mount Name: /mnt/myDrive1
- Media1** (checked):
 - Media configuration1
 - Media Type: SYS_FS_MEDIA_TYPE_MSD
 - File System Type: FAT
 - Number Of Volumes: 1
 - Volume0 (checked)
 - Volume configuration0
 - Device Name: /dev/sda1
 - Media Mount Name: /mnt/myDrive2
- Media2** (checked):
 - Media configuration2
 - Media Type: SYS_FS_MEDIA_TYPE_NVM
 - File System Type: FAT
 - Create FAT12 in NVM: ☐
 - Number Of Volumes: 1
 - Volume0 (checked)
 - Volume configuration0
 - Device Name: /dev/nvma1
 - Media Mount Name: /mnt/myDrive3

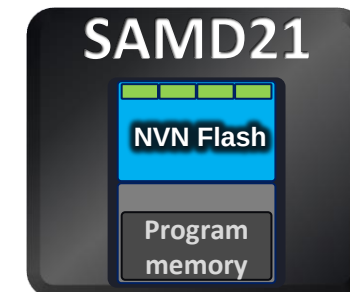
Media0
/dev/mmcblk1
/mnt/myDrive1



Media2
/dev/sda1
/mnt/myDrive2

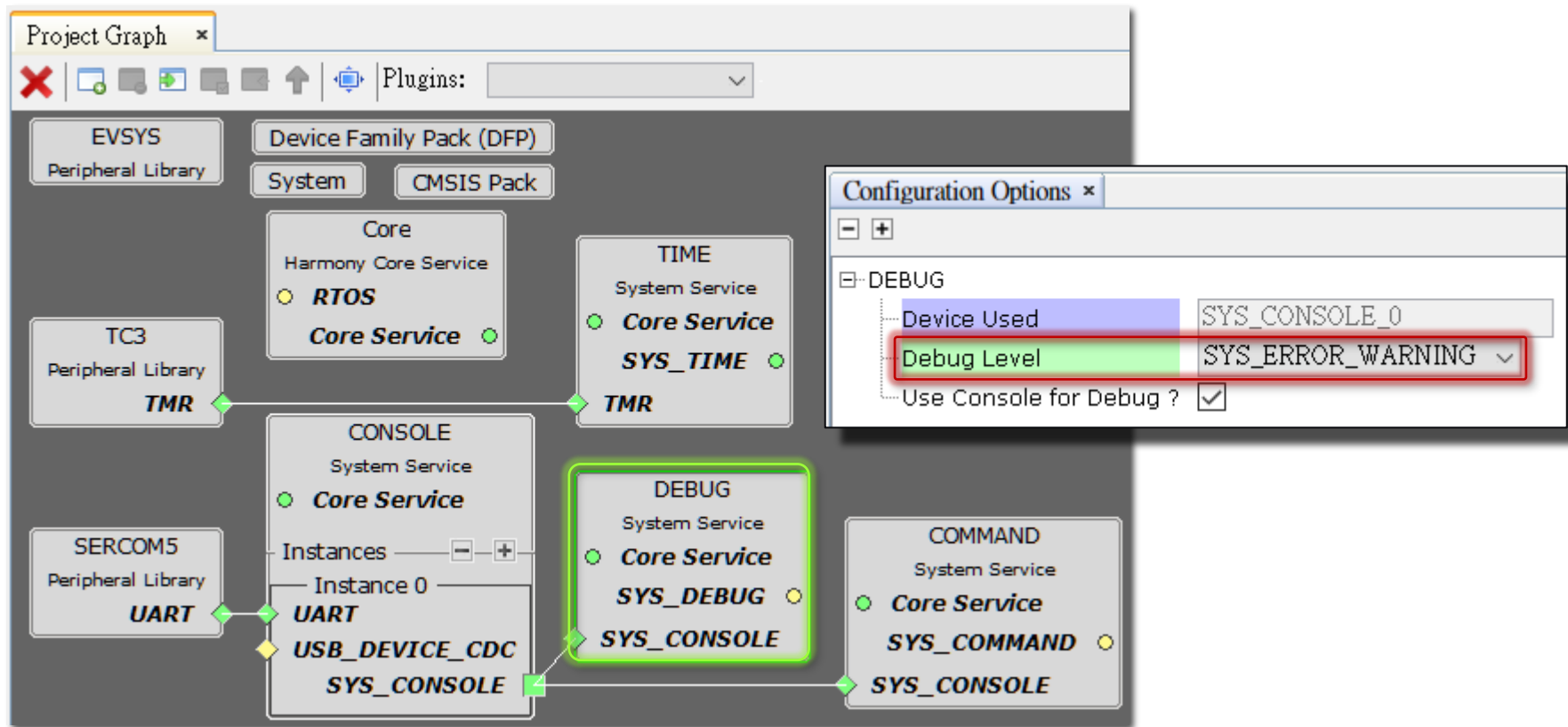


Media2
/dev/nvma1
/mnt/myDrive3



⚙️ Lab10 NVMCTRL Internal Flash with File System

- Let's change the default Debug Error Level to “**SYS_ERROR_WARNING**” in **DEBUG system Service** for disabled information message from USB Host Layer.



⚙️ Lab10 NVMCTRL Internal Flash with File System

- Add NVM Flash mount status in Auto Mount Event Handler.

```
static const SYS_CMD_DESCRIPTOR MyCmdTbl[] = { ... };

void AutoMountEventHandler( SYS_FS_EVENT event, void* eventData, uintptr_t context )
{
    switch(event)
    {
        case SYS_FS_EVENT_MOUNT:
            if(strcmp((const char *)eventData, SYS_FS_MEDIA_IDX0_MOUNT_NAME_VOLUME_IDX0) == 0)
            {
                SYS_CMD_MESSAGE("Mount SD card success!\r\n");
            }
            if(strcmp((const char *)eventData, SYS_FS_MEDIA_IDX1_MOUNT_NAME_VOLUME_IDX0) == 0)
            {
                SYS_CMD_MESSAGE("Mount USB Mass Storage success!\r\n");
            }
            if(strcmp((const char *)eventData, SYS_FS_MEDIA_IDX2_MOUNT_NAME_VOLUME_IDX0) == 0)
            {
                SYS_CMD_MESSAGE("Mount NVM Flash success!\r\n");
            }
            break;

        case SYS_FS_EVENT_UNMOUNT:
            if(strcmp((const char *)eventData, SYS_FS_MEDIA_IDX0_MOUNT_NAME_VOLUME_IDX0) == 0)
            {
                SYS_CMD_MESSAGE("Unmount SD card!\r\n");
            }
            if(strcmp((const char *)eventData, SYS_FS_MEDIA_IDX1_MOUNT_NAME_VOLUME_IDX0) == 0)
            {
                SYS_CMD_MESSAGE("Unmount USB Mass Storage!\r\n");
            }
            break;
        ...
    }
}
```

NVM Flash doesn't need the unmount message

⚙️ Lab10 NVMCTRL Internal Flash with File System

- Modify command “media” to add NVM Flash support.

```
void CommandClose(SYS_CMD_DEVICE_NODE* pCmdIO, int argc, char** argv) { ... }

void CommandMedia(SYS_CMD_DEVICE_NODE* pCmdIO, int argc, char** argv)
{
    if(!strcmp( argv[1], "SD"))
    {
        strcpy(fMedia, SYS_FS_MEDIA_IDX0_DEVICE_NAME_VOLUME_IDX0);
        strcpy(fDrive, SYS_FS_MEDIA_IDX0_MOUNT_NAME_VOLUME_IDX0);
    }
    if(!strcmp( argv[1], "USB"))
    {
        strcpy(fMedia, SYS_FS_MEDIA_IDX1_DEVICE_NAME_VOLUME_IDX0);
        strcpy(fDrive, SYS_FS_MEDIA_IDX1_MOUNT_NAME_VOLUME_IDX0);
    }
    if(!strcmp( argv[1], "NVM"))
    {
        strcpy(fMedia, SYS_FS_MEDIA_IDX2_DEVICE_NAME_VOLUME_IDX0);
        strcpy(fDrive, SYS_FS_MEDIA_IDX2_MOUNT_NAME_VOLUME_IDX0);
    }

    SYS_CMD_PRINT("Set media to %s\r\n", fMedia);

    if(SYS_FS_CurrentDriveSet(fDrive) != SYS_FS_RES_SUCCESS)
    {
        SYS_CMD_PRINT("Drive %s not mounted!\r\n", fDrive);
        return;
    }

    SYS_CMD_PRINT("Set drive to %s\r\n", fDrive);
}

static const SYS_CMD_DESCRIPTOR MyCmdTbl[] = { ... };
```

⚙️ Lab10 NVMCTRL Internal Flash with File System

- Add a new command “**format**” to format active Drive as FAT16.

```
void CommandMedia(SYS_CMD_DEVICE_NODE* pCmdIO, int argc, char** argv) { ... }

void CommandFormat(SYS_CMD_DEVICE_NODE* pCmdIO, int argc, char** argv)
{
    SYS_FS_FORMAT_PARAM opt = { 0 };
    uint8_t CACHE_ALIGN work[512];

    SYS_FS_CurrentDriveGet( fDrive );
    opt.fmt = SYS_FS_FORMAT_FAT;
    opt.au_size = 0;
    opt.n_fat = 1;
    opt.align = 0;
    opt.n_root = 1;

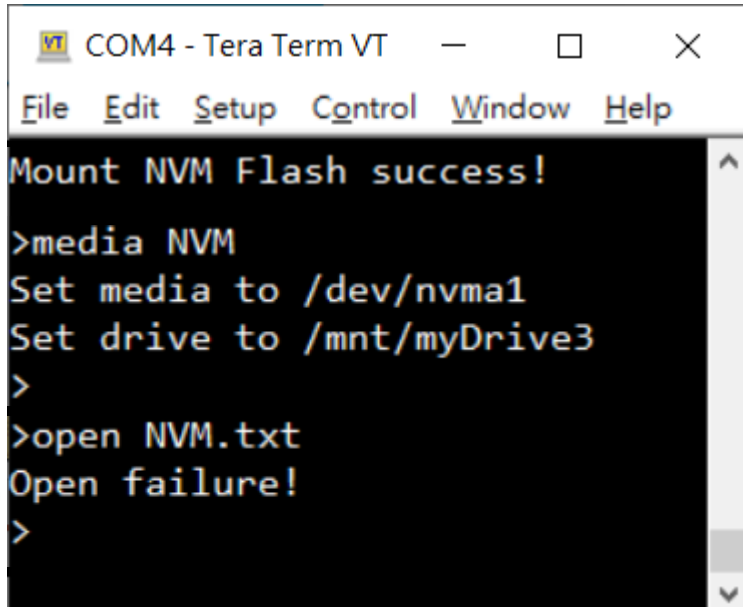
    fResult = SYS_FS_DriveFormat(fDrive, &opt, (void *)work, 512);
    if(fResult == SYS_FS_RES_SUCCESS)
    {
        SYS_CMD_PRINT("Format %s success!\r\n", fDrive);
    }
    else
    {
        SYS_CMD_PRINT("Format %s failure!\r\n", fDrive);
    }
}

static const SYS_CMD_DESCRIPTOR MyCmdTbl[] =
{
    { "close", CommandClose, ": Close file" },
    { "media", CommandMedia, ": media %" },
    { "format", CommandFormat, ": Format Media" },
};

void AutoMountEventHandler( SYS_FS_EVENT event, void* eventData, uintptr_t context ) { ... }
```

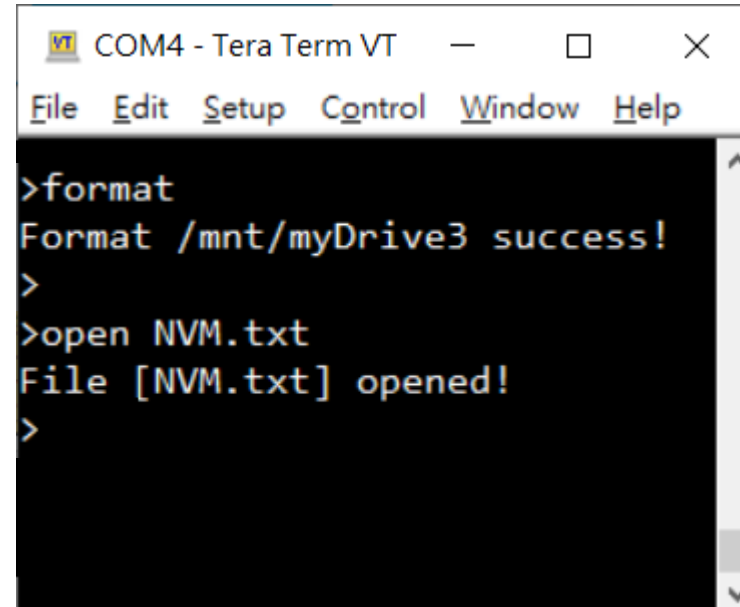
⚙️ Lab10 NVMCTRL Internal Flash with File System

- Remove SD card and USB pen drive firstly.
- Compiler and programming your project to EVB.
- The NVM should **mount automatically** while reboot.
- Command “**media NVM**” to change active media to NVM Flash.
- Command “**open NVM.txt**” try to create a new file in NVM Flash.
- Command “**format**” to format NVM Flash drive to default FAT.
- Try “**open NVM.txt**” again to check the different.



```
COM4 - Tera Term VT
File Edit Setup Control Window Help
Mount NVM Flash success!
>media NVM
Set media to /dev/nvma1
Set drive to /mnt/myDrive3
>
>open NVM.txt
Open failure!
>
```

Why? ➔



```
COM4 - Tera Term VT
File Edit Setup Control Window Help
>format
Format /mnt/myDrive3 success!
>
>open NVM.txt
File [NVM.txt] opened!
>
```

⚙️ Lab10 NVMCTRL Internal Flash with File System

- Press the **Reset** button on EVB.
- Try open “**NVM.txt**” and write “**Hello World NVM**” to file in NVM Flash.
- Press the **Reset** button on EVB.
- Try open “**NVM.txt**” and read the content we wrote from file in NVM Flash.

```
COM4 - Tera Term VT
File Edit Setup Control Window Help
Mount NVM Flash success!
>open NVM.txt
File [NVM.txt] opened!
>
>write "Hello World NVM"
Write [Hello World NVM](15) to File!
>
>close
File Closed!
>
```

```
COM4 - Tera Term VT
File Edit Setup Control Window Help
Mount NVM Flash success!
>open NVM.txt
File [NVM.txt] opened!
>
>read 30
Read [Hello World NVM](15) from File!
>
>close
File Closed!
>
```


⚙️ Lab10 NVMMCTRL Internal Flash with File System

- Put in **SD Card** and **USB pen drive** to EVB, the last drive that Auto Mount success to mount is current active media.
- Execute '**media SD**', '**open SD.txt**', '**read 20**', '**close**'
- Execute '**media NVM**', '**open NVM.txt**', '**read 20**', '**close**'
- Execute '**media USB**', '**open USB.txt**', '**read 20**', '**close**'

```
COM3 - Tera Term VT
File Edit Setup Control Window Help
Mount NVM Flash success!
Mount SD card success!
Mount USB Mass Storage success!

>media SD
Set media to /dev/mmcblk1
Set drive to /mnt/myDrive1
>open SD.txt
File [SD.txt] opened!
>read 20
Read [Hello World SD](14) from File!
>close
File Closed!
>
```

```
>media NVM
Set media to /dev/nvma1
Set drive to /mnt/myDrive3
>open NVM.txt
File [NVM.txt] opened!
>read 20
Read [Hello World NVM](15) from File!
>close
File Closed!
>
```

```
>media USB
Set media to /dev/sda1
Set drive to /mnt/myDrive2
>open USB.txt
File [USB.txt] opened!
>read 20
Read [Hello World USB](15) from File!
>close
File Closed!
>
```

⚙️ Lab10 NVMCTRL Internal Flash with File System

Discuss 1

- **How to Format SD Card / USB pen drive ?**

- The FAT(FAT12) support up to 32MBytes, it's suitable for the NVM Flash drive.
- The FAT32 support up to 2TB drive(4GB per file), will be proper for a SD Card or USB pen drive.

NVM Flash

SD/ USB drive

```
void CommandFormat(SYS_CMD_DEVICE_NODE* pCmdIO, int argc, char** argv)
{
    SYS_FS_FORMAT_PARAM opt = { 0 };
    uint8_t CACHE_ALIGN work[512];

    SYS_FS_CurrentDriveGet( fDrive );
    if(!strcmp( fDrive, "/mnt/myDrive3"))
    {
        opt.fmt = SYS_FS_FORMAT_FAT;
        SYS_CMD_PRINT("Format as FAT12!\r\n");
    }
    else
    {
        opt.fmt = SYS_FS_FORMAT_FAT32;
        SYS_CMD_PRINT("Format as FAT32!\r\n");
    }

    opt.au_size = 0;
    opt.n_fat = 1;
    opt.align = 0;
    opt.n_root = 1;

    fResult = SYS_FS_DriveFormat(fDrive, &opt, (void *)work, 512);
    if(fResult == SYS_FS_RES_SUCCESS)
    {
        SYS_CMD_PRINT("Format %s success!\r\n", fDrive);
    }
}
```

```
COM4 - Tera Term VT
File Edit Setup Control Window Help
Mount NVM Flash success!
>format
Format as FAT12!
Format /mnt/myDrive3 success!
>
>Mount SD card success!
>format
Format as FAT32!
Format /mnt/myDrive1 success!
```

Microchip Trademarks

<https://www.microchip.com/en-us/about/legal-information/microchip-trademarks>

Trademarks	Description/Appropriate Nouns	Notes
Adaptec "a" logo		Registered in other countries (not US)
Adaptec®	products, solutions, cables, software, drivers, adapters, controllers, series, family	
Adjacent Key Suppression™	technology	
AgileSwitch®	digital programmable gate driver, gate driver, drivers, family, device	US Only
AKS™	technology	
Analog-for-the-Digital Age™	AIPD family tagline	
Any Capacitor™	stable	
AnyIn™	feature, structure, capability	
AnyOut™	feature, capability	
Augmented Switching™	technology	
AVR®	devices, microcontrollers, MCUs, CPU, architecture, family	
AVR Logo		
AVR Freaks®	forum	
BesTime®	technology	
BitCloud®	SDK	
BlueSky™	firewall, subscription service, technology	
BodyCom™	technology, system, development kit	
ClockStudio™	application, software, program, utility	
ClockWorks®	Configurator, products, series, family, devices	US Only
CodeGuard™	security	
CryptoAuthentication™	development library, library, devices, ICs, family	
CryptoAutomotive™	technology, security ICs, development kits	
CryptoCompanion™	chip	
CryptoController™	solution, Trusted Platform Module (TPM)	
CryptoMemory®	cryptographic security ICs, ICs	

NOTE: *Always check with the Legal Department for the most current trademarks.

Trademarks	Description/Appropriate Nouns	Notes
PIC ³² Logo		
PICDEM™	demonstration board	
PICDEM.net™	Internet/Ethernet demonstration board	
PICKit™	starter kit, programmer, debugger, in-circuit debugger, development tool, tool, on-board	
picoPower®	technology	
PICSTART®	development programmer	
PICtail™	daughter board, board	
PolarFire®	FPGA, family, kits, devices, SoC, System-on-Chip	
Power MOS IV™		
Power MOS 7™		
Powermite 3®	package	US Only
PowerSmart™	digital control library designer, development suite	
Precision Edge®	family, product family	US Only
ProASIC®	FPGA, architecture, board, kit, family	US Only
ProASIC Plus®	FPGA, devices, family, architecture, switch	US Only
ProASIC Plus Logo		
Prochip Designer®	software suite	
PureSilicon™	oscillator, technology	
QMatrix™	sensor, devices	
QTouch®	sensor, devices, library, tools, development platform, project builder, Composer, Project Builder, board, solution, technology, Configurator	
Quiet-Wire®	enhanced EMI technology, family, technology	
Ripple Blocker™	technology, attenuators, product line, family of linear regulators	
REAL ICE™	In-circuit emulator	
RTAX™	FPGAs	
RTG4™	FPGAs	
SAM-BA®	software, application, monitor	

NOTE: *Always check with the Legal Department for the most current trademarks.

Trademarks	Description/Appropriate Nouns	Notes
CryptoRF®	transponder, devices	
dsPIC®	Digital Signal Controllers (DSCs)	
dsPIC Logo		
dsPICDEM™	demonstration board, development board	
dsPICDEM.net™	board	
Dynamic Averaged Matching™ (DAM™)	architecture	
ECAN™	technology, solution	
Espresso T1S™	device	
EtherGREEN™	platform, technology, family	
EtherSynch™	platform, technology, family	US Only
EyeOpen™	cable compensation	
Flashtec®	controller, products, family	US Only
flexPWR®	technology	
Frequency on Demand®		Registered in other countries (not US)
GestIC®	technology, sensing, controllers	
GridTime™	device, time server, GNSS time server	
HELDO®	family, regulators	
Hyper Speed Control®	family, architecture	US Only
HyperLight Load®	mode	US Only
ICSP™ or In-Circuit Serial Programming™	programming capability	
IdealBridge™	rectifier, dual MOSFET-bridge rectifier	
IGAT™	board, demonstration board	
IGLOO®	FPGAs, family, devices, series	
INICNet™	technology, sniffer	
Intelligent Paralleling™	technology	
IntelliMOS™	family, power stage family	US Only
Inter-Chip Connectivity™	technology	
JitterBlocker™	family, attenuators	
JukeBox®	technology, platform	

NOTE: *Always check with the Legal Department for the most current trademarks.

Trademarks	Description/Appropriate Nouns	Notes
SAM-ICE™	emulator	
SerGenuity®	sensors, products	
Serial Quad I/O™ or SQI™	family, flash device, product	
Silicon Storage Technology®		Japan only
simpleMAP™	protocol	
SimpliPHY™	family, products, devices	
SmartBuffer™	devices, ICs	
SmartFusion®	FPGA, evaluation kit, SoC, System-on-Chip,	US Only
SmartHLS™	IDE, compiler software, compiler, software	
SMART-I.S.™	technology	
SpyNIC®	device	
SST®		
SST Logo		
storClad™	encryption technology, technology	
SuperFlash®	technology, kit, macro, device, memory	
SuperSwitcher™	family, regulator, buck regulator	
SuperSwitcher II™	family	
Switchtec™	technology, family, switch(es), product line	
Symcom®		China only
SymmetriCom®		
SynchroPHY™	family, products, devices	
SyncServer®	server, instrument, clock, oscillator, device	
SyncWorld®	timing, Ecosystem Program	US only
Tachyon®	products, family, controllers, platform, technology	
The Embedded Control Solutions Company®	MarCom Use Only	
TimeCesium®	products, primary frequency reference	US only
TimeHub®	platform, system	US only
TimePicta®	software suite, platform, Synchronization Management System	US only

NOTE: *Always check with the Legal Department for the most current trademarks.

Trademarks	Description/Appropriate Nouns	Notes
JukeBox with design	Registered word mark JukeBox with a design	Image is not registered; this is a graphic treatment with the word mark
		
KEELO®	security ICs, protocol, technology	
Kleer®	technology	
Kleer with design	Registered word mark Kleer with a design	Image is not registered; this is a graphic treatment with the word mark
		
Knob-on-Display™/KoD™	technology, solution, HMI, user interface, touch controller, family, devices, Widget Configurator, Knob Designer, Position Wizard	
LANCheck®	online design review, online review, review	
Libero®	SoC Design Suite, tools, software, design files, IDE, designer	US registration
LinkMD®	cable diagnostics, technology, family, diagnostics engine	
MarginLink™	signal integrity testing	
maxCrypto™	controller-based encryption, encryption	
maxStylus®	solution	
maXTouch®	touchscreen controllers, technology, studio, family	
maxView™	storage manager, tools, drivers	
MediaLB®	bus, controller, device	
megaAVR®	devices, MCUs	
memBrain™	neuromorphic memory product, product, solution, technology	
Microchip Logo		
Microchip Name and Logo		
Microsemi Logo		
Microsemi®		

NOTE: *Always check with the Legal Department for the most current trademarks.

Trademarks	Description/Appropriate Nouns	Notes
TimeProvider®	series, server, clock, products, grandmaster, ePRTC, enhanced PRTC	US only
TimeSource®	Enhanced PRTC	
tinyAVR®	microcontroller, MCU	
Total Endurance™	software model	
Trusted Time™	technology	
TSHARC™	touch screen controller, controller	
Turing™	hardware security module, programmer	
UNIQ®	memory chips, bus, hardware port, communication protocol, firmware	
USBCheck™	design review	
VarSense™	technology	
VectorBox™	accelerator, software development kit, SDK, intellectual property, IP, solution, platform, tools, tool chain, tool flow	
Vectron®		
VeriPHY™	cable diagnostics algorithm, cable diagnostics software suite	
ViewSpan™	technology	
WiperLock™	technology	
XMGA®	microcontroller, MCU, devices, Custom Logic (XCL), family, series	
XpressConnect™	retimer	
ZENA™	software, analyzer or wireless adapter	
ZL®		

NOTE: *Always check with the Legal Department for the most current trademarks.

Microchip Technology Inc. Servicemarks®

Trademarks	Description/Appropriate Nouns	Notes
SQTP™	Serial Quick Turn Programming capability	

NOTE: *Always check with the Legal Department for the most current servicemarks.

MICROCHIP RESERVES THE RIGHT TO CHANGE THESE GUIDELINES SOLELY AT ITS DISCRETION.

Trademarks	Description/Appropriate Nouns	Notes
MindI™	analog simulator, analog simulator tool	
MiWi™	wireless networking protocol, protocol, network, stack, software, applications, mesh, coordinator	
MOST®	technology, NetServices, evaluation kit, board, system, device, product, network, specification, platform, cooperation, design	
MOST Logo		
motorBench®	Development Suite	US Only
MPASM™	assembler	
MPF™	products, devices	
MPLAB®	Integrated Development Environment, In-circuit Debugger, In-Circuit Emulator, IDE, ICD, REAL ICE™ in-circuit emulator, starter kit, compiler, XC compiler, C compiler, Harmony, Integrated Programming Environment, IPE, Xpress, Code Configurator, development ecosystem, PICKit™, server, library, Snap, Connect Configurator, Network Creator, Analog Designer, Discover, MindI™, Analysis Tool Suite, Discover, Universal Device Programmer, Machine Learning Development Suite, PTG, Programmer-to-Go, Programmer-to-Go Mobil App, SIC Power Simulator	
MPLAB Certified Logo		
MPLIB™	object librarian or librarian	
MPLINK™	object linker or linker	
mSIC™	MOSFETs, diodes, products, gate drivers, modules, bare die, technology, solutions	
mTouch®	solution, sensing solution, evaluation kit, technology, development kit, library	US Only
MultiTRAK™	technology	
NetDetach™	technology	
Omniscient Code Generation™ (OCG)	compilation technology	
OptoLyzer®	tool, network analysis tool, Studio	
PIC®	microcontroller, MCU, device(s), assembler	

NOTE: *Always check with the Legal Department for the most current trademarks.

