SAMD21 ARM Cortex-M0+ Microcontroller & MCC Harmony

SAM2001

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

SMART | CONNECTED | SECURE

CAE Taiwan
Microchip Technology Inc.

March 19, 2025

Class Objectives

- **When you finish this course, you will be able to:**
  - Be familiar with MPLAB X IDE and MCC Harmony
  - Understand how to configure and use the SAMD21 series peripherals.
- **There have two level course included.**
  - Major course : General study of basic features.
  - Appendix : Advance features and appendix.

SAMD21 ARM Cortex-M0+ Microcontroller Introduction

Major Course (1/3)

- SAMD21 ARM Cortex-M0+ Microcontroller Introduction
- MPLAB® X IDE, MCC Harmony and Development Tools Introduction
- SAMD21 EVB APP045 v4.20 ES.d Introduction
- Getting Started with First Project
  - 💻 Lab0 First Project
- MPLAB® Code Configurator Harmony Interface introduction
- PORT I/O Architecture
  - 💻 Lab1 PORT Output
  - 💻 Lab2 PORT Input and Output
- Timer/Counter, TC Architecture
  - 💻 Lab3 TC Timer Method Polling
  - 💻 Lab4 TCs Timer Method Polling

Major Course (2/3)

- NVIC Architecture
  - 💻Lab5 TC Timer Method Interrupt
  - 💻Lab6 TCs Timer Method Interrupt
  - 🎯Study Advanced Features
- Clock System Architecture
  - 💻Lab7 System Clock XOSC32K
  - 💻Lab8 System Clock DFLL48M
- Pin Multiplexer (PINMUX)
- SERCOM - UART Architecture
  - 💻Lab9 SERCOM - UART Tx Polling
  - 💻Lab10 SERCOM - UART Tx/Rx Interrupt
  - 💻Lab11 SERCOM - UART Tx/Rx Interrupt with myprintf

Major Course (3/3)

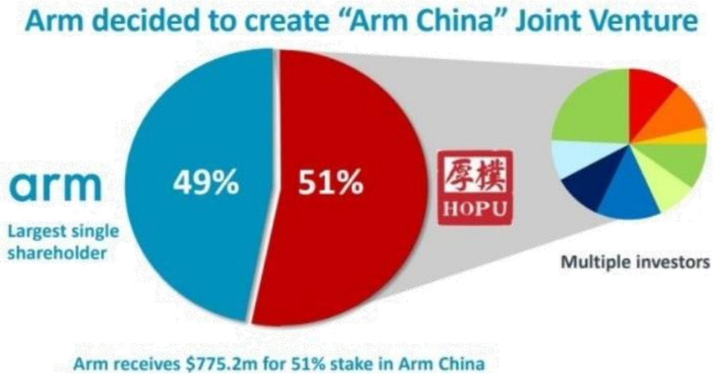
- High Speed ADC Architecture
  - 💻 Lab12 ADC Polling Software Trigger
- ADC Interrupt and Pin Scan
  - 💻 Lab13 ADC Pin Scan
- TCC – PWM Architecture
  - 💻 Lab14 TCC NPWM
  - 💻 Lab15 TCC NPWM ADC Duty Control
  - 💻 Lab16 TCC NPWM Auto Duty Control
- SERCOM-SPI (OLED SSD1306)
  - 💻 Lab17 SERCOM - SPI SSD1306 OLED

SAMD21 ARM Cortex-M0+ Microcontroller Introduction

MICROCHIP

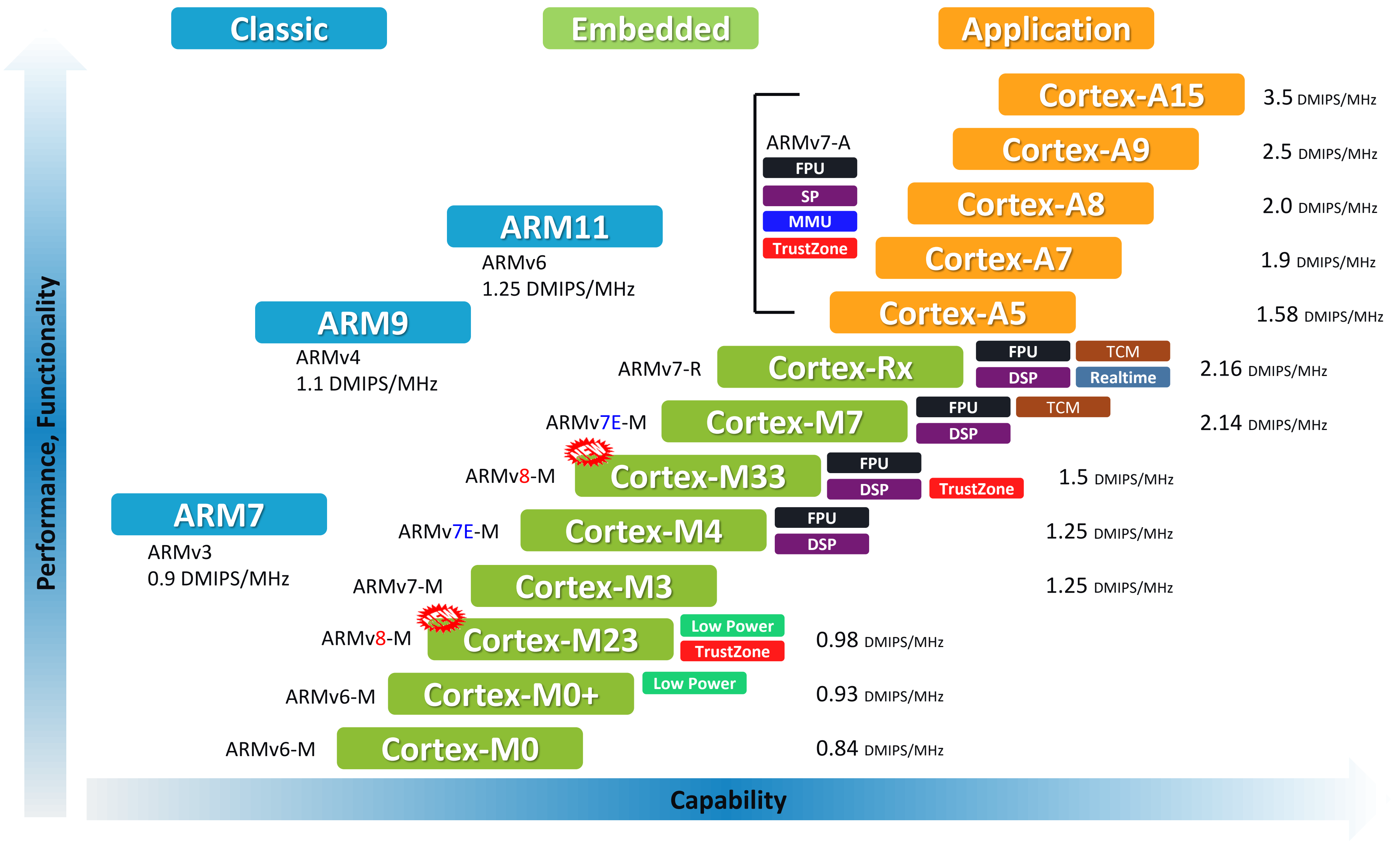
ARM's history

- **ARM**, previously **A**dvanced **R**ISC **M**achine, is a family of reduced instruction set computing (RISC) architectures for computer processors. Arm Holdings develops the architecture and licenses it to other companies, who design their own products that implement systems-on-chips (SoC) and systems-on-modules (SoM).
- SoftBank complete acquisition of ARM Holdings on September 5, 2016 (GMT) for a total acquisition price approximately USD 31.0 billion or JPY 3.3 trillion.
- SoftBank confirmed on June 4, 2018 that Arm Limited agreed sell a majority stake in its Chinese subsidiary Arm Technology (China) Co., Ltd. Under the terms of the Transaction, Arm will sell 51% of its equity interest in Arm China for USD 775.2 million to the Direct Investors.
- On 13 September 2020, it was announced that Nvidia would buy Arm from SoftBank for $40 billion, subject to regulatory approval, with the latter acquiring a 10% share in Nvidia.



ARM in Partnership

ARM core MCU/MPU brief



MICROCHIP

ARM core MCU/MPU success

MICROCHIP

Microchip ARM Core Products

Explore 32-bit MCUs

Explore 32-bit MPUs

PERFORMANCE

Related Family

PIC32CM JH
ARM® Cortex® M0+, 48 MHz 5 Volt

SAMC
ARM® Cortex® M0+, 48-64 MHz 5 Volt

PIC32CM MC
ARM® Cortex® M0+, 48 MHz 5 Volt

SAM R30
ARM® Cortex® M0+, 48 MHz

SAM D
ARM® Cortex® M0+, 48 MHz

Related Family

PIC32CM Lx
ARM® Cortex® M23, 48 MHz

SAM L21/22
ARM® Cortex® M0+, 32-48 MHz

SAM L10/11
ARM® Cortex® M23, 32 MHz

Related Family

PIC32CK
ARM® Cortex® M33, 120 MHz

PIC32CK SG
ARM® Cortex® M4F, 120 MHz

SAM E5x
ARM® Cortex® M4F, 120 MHz

SAM D5x
ARM® Cortex® M4F, 120 MHz

SAM G
ARM® Cortex® M4F, 120 MHz

SAM 4
ARM® Cortex® M4F, 120 MHz

PIC32CK BZ
ARM® Cortex® M4F, 64 MHz

SAM 3
ARM® Cortex® M3, 96 MHz

Related Family

PIC32CZ CA
ARM® Cortex® M7, 300 MHz

SAM V7x
ARM® Cortex® M7, 300 MHz

SAM E7x
ARM® Cortex® M7, 300 MHz

SAM S7x
ARM® Cortex® M7, 300 MHz

32-bit MPU Product Families

SAM9 Series MPU
Arm926EJ-S™, up to 800 MHz

SAMA5 Series MPU
ARM® Cortex® A5, up to 940 DMIPS

SAMA7 Series MPU
ARM® Cortex® A7, up to 1900 DMIPS

MICROCHIP

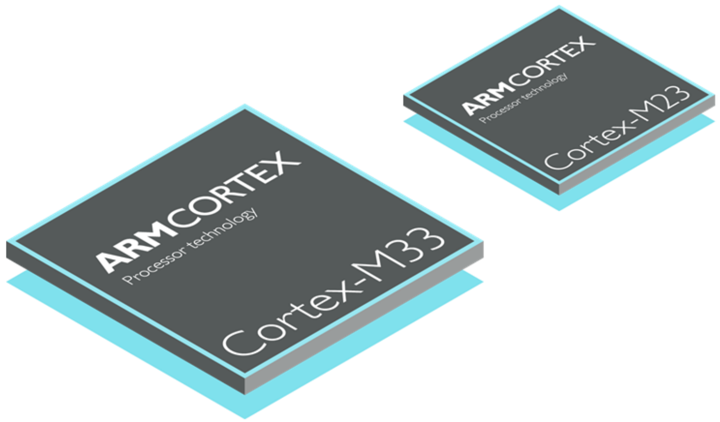
Microchip Cortex-M MCU Series

- **Microchip 32 Bits (ARM core) MCU include below series :**
- **Cortex-M0+**
  - SAMD09 / SAMD1x New
    - Low pin count, high c/p.
  - SAMD2x
    - Rich peripheral.
  - SAMC2x / PIC32CM-JM New
    - Support 5V and CAN bus.
  - SAML2x / PIC32CM L2x New
    - Lowest power consumption and peripheral.
  - SAMDAx
    - Automotive certificated.
- **Cortex-M3, Cortex-M4 (Legacy)**
  - SAM3x, SAM4x, SAMG5x.

MICROCHIP

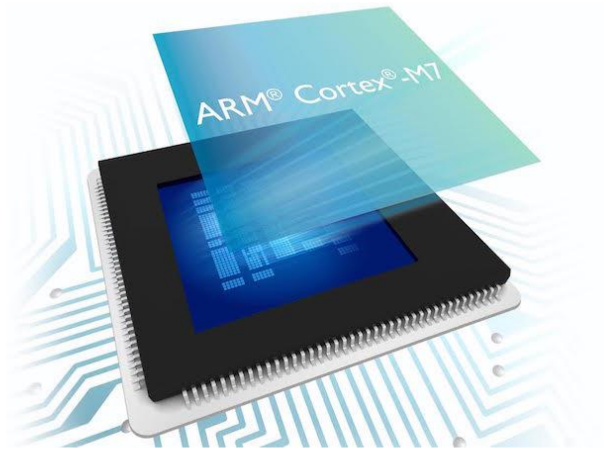
Microchip Cortex-M MCU Series (cont.)

- **Cortex-M4 (New generation)**
  - SAMD5x
    - Support float point FPU, crypto engine, USB , low power.
  - SAME5x / PIC32CK New
    - FPU, crypto engine, USB, Ethernet, CAN-FD, Low power.
- **Cortex-M23**
  - SAML1x / PIC32CM L1x New
    - Very low power, support TrustZone.
- **Cortex-M33 (Coming soon)**
  - SAMD6x
    - High performance, support TrustZone.



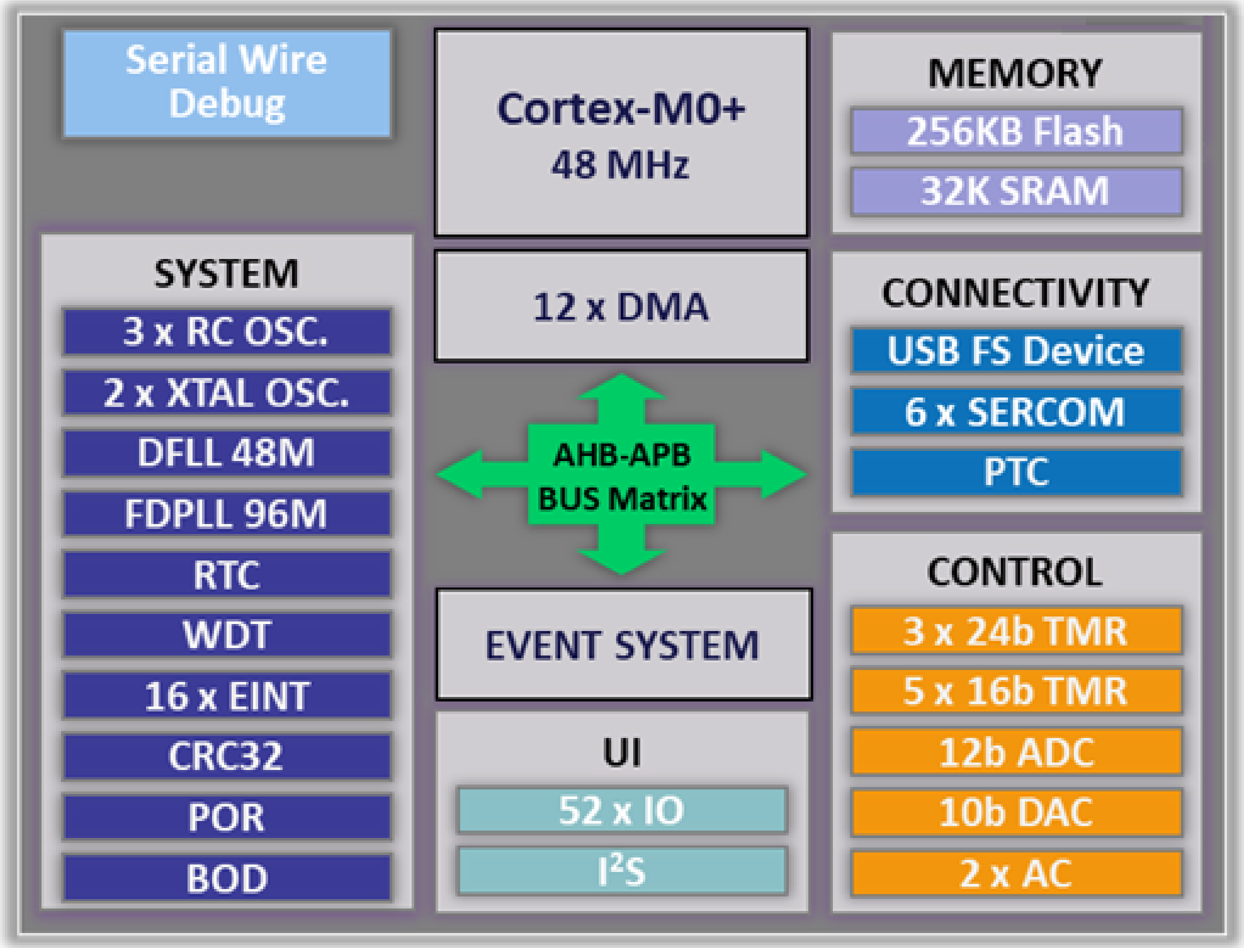
Microchip Cortex-M MCU Series (cont.)

- **Cortex-M7**
  - SAMS7x
    - High performance, rich peripheral.
  - SAME7x / PIC32CZ New
    - Ethernet support, CAN-FD support.
  - SAMV7x / PIC32CZ New
    - Ethernet support, CAN-FD support, automotive certificated.



SAMD21 Block Diagram

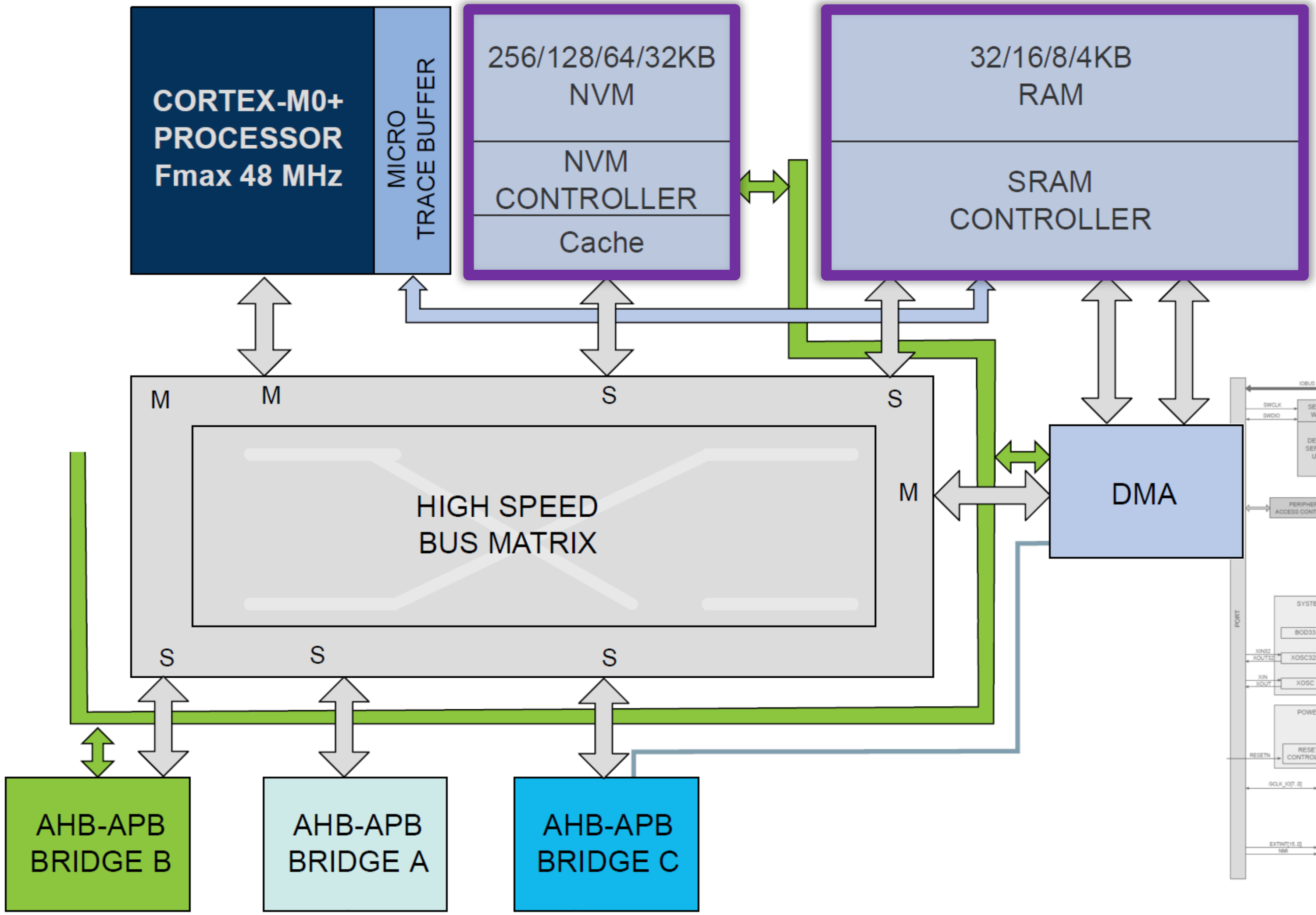
- The SAMD21 is a series of low-power microcontrollers using the ARM® Cortex® M0+ processor (Cortexv6-M)。
- Maximum frequency of 48MHz and reach 2.46 CoreMark®/ MHz。
- All devices include intelligent and flexible peripherals, Event System for inter-peripheral signaling, and support for capacitive touch button, slider and wheel user interfaces。

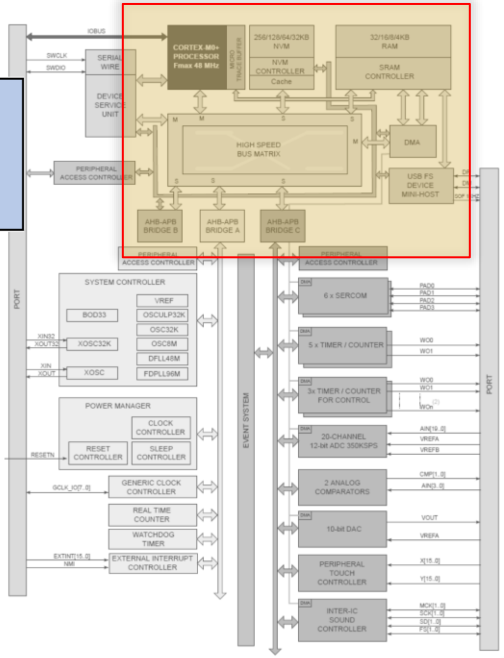


MICROCHIP

Architecture

- **Cortex M0+ is Von Neumann architecture, RISC**





MICROCHIP

Architecture

- **Cortex M0/M0+ Von Neumann**

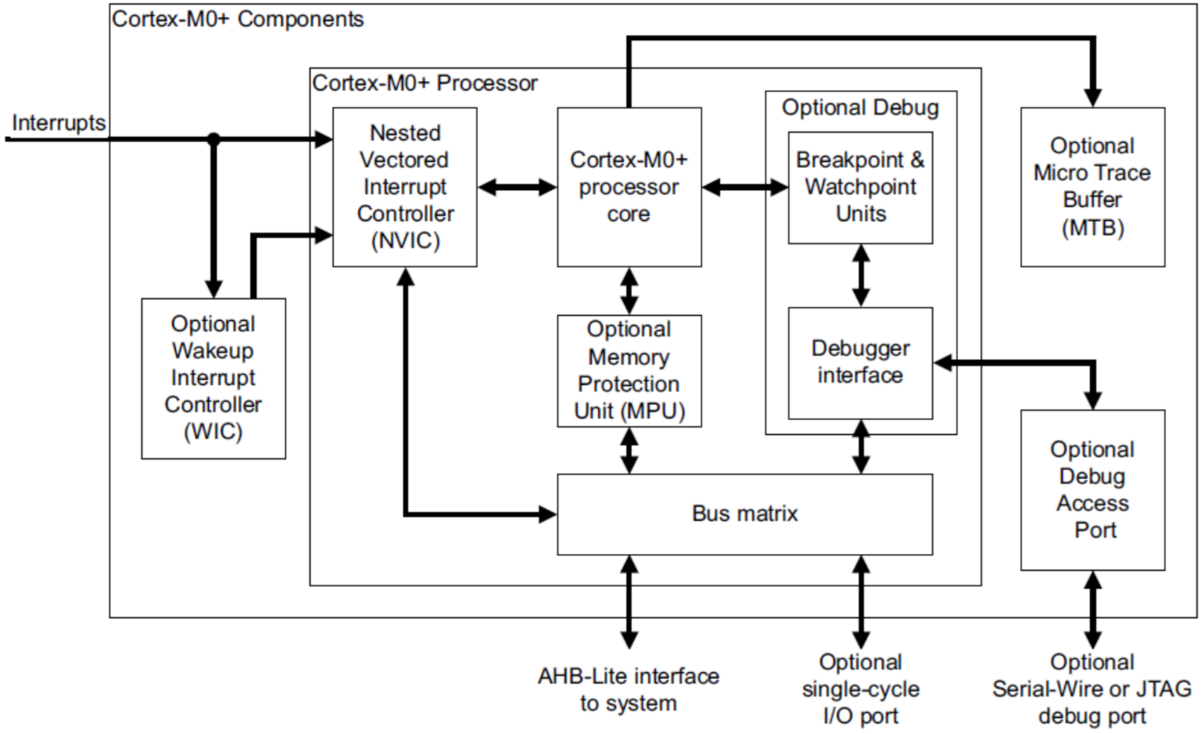


Figure 1-1 Cortex-M0+ implementation

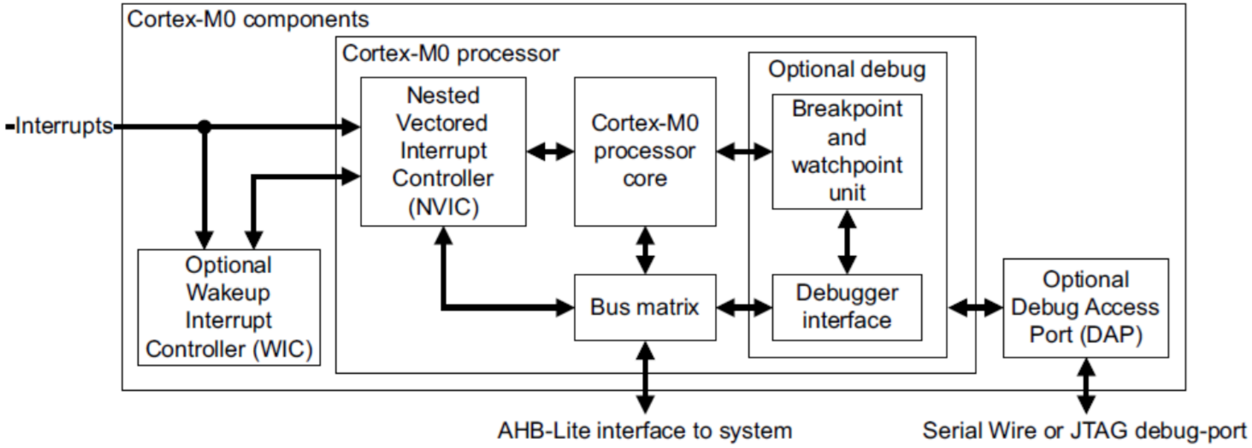


Figure 1-1 Cortex-M0 implementation

Architecture

- **Cortex M3 Harvard**

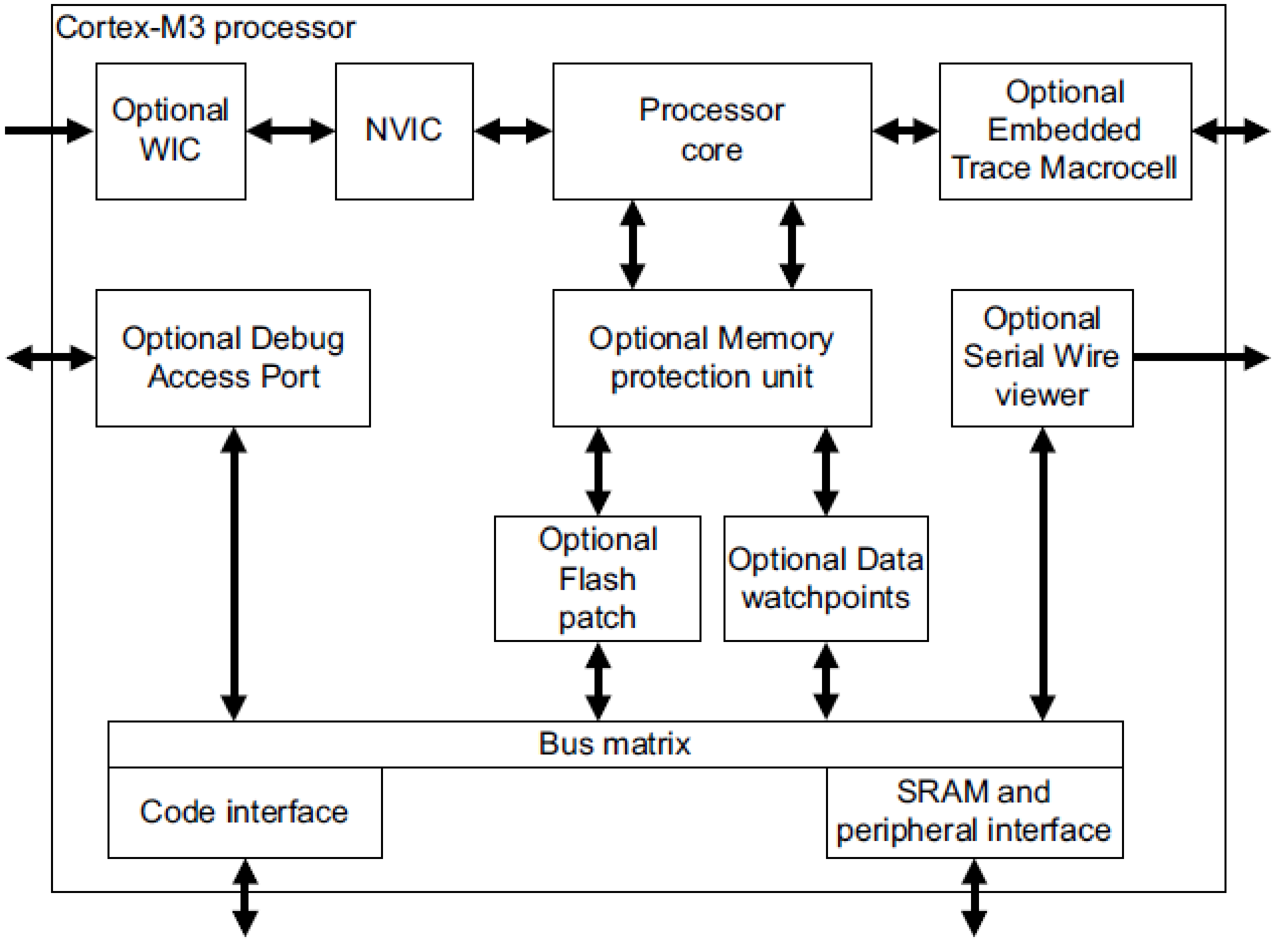


Figure 1-1 Cortex-M3 implementation

Architecture

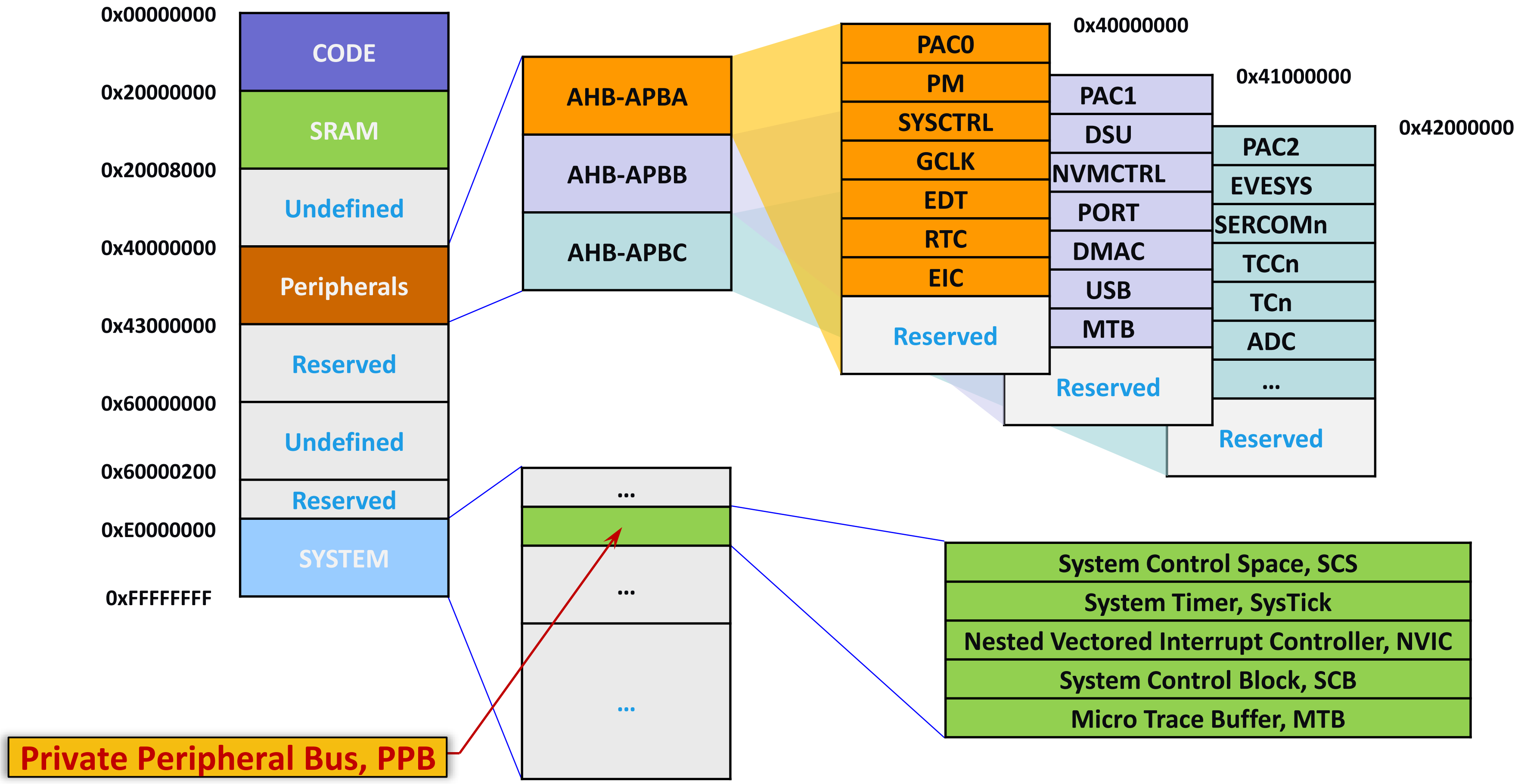
ARM Cortex-M instruction variations

Arm Core	Cortex M0[2]	Cortex M0+[3]	Cortex M1[4]	Cortex M3[5]	Cortex M4[6]	Cortex M7[7]	Cortex M23[8]	Cortex M33[12]	Cortex M35P	Cortex M55
ARM architecture	ARMv6-M[9]	ARMv6-M[9]	ARMv6-M[9]	ARMv7-M[10]	ARMv7E-M[10]	ARMv7E-M[10]	ARMv8-M Baseline[15]	ARMv8-M Mainline[15]	ARMv8-M Mainline[15]	Armv8.1-M
Computer architecture	Von Neumann	Von Neumann	Von Neumann	Harvard	Harvard	Harvard	Von Neumann	Harvard	Harvard	Harvard
Instruction pipeline	3 stages	2 stages	3 stages	3 stages	3 stages	6 stages	2 stages	3 stages	3 stages	4 to 5 stages
Thumb-1 instructions	Most	Most	Most	Entire	Entire	Entire	Most	Entire	Entire	Entire
Thumb-2 instructions	Some	Some	Some	Entire	Entire	Entire	Some	Entire	Entire	Entire
Multiply instructions 32x32 = 32-bit result	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Multiply instructions 32x32 = 64-bit result	No	No	No	Yes	Yes	Yes	No	Yes	Yes	Yes
Divide instructions 32/32 = 32-bit quotient	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Saturated instructions	No	No	No	Some	Yes	Yes	No	Yes	Yes	Yes
DSP instructions	No	No	No	No	Yes	Yes	No	Optional	Optional	Optional
Single-Precision (SP) Floating-point instructions	No	No	No	No	Optional	Optional	No	Optional	Optional	Optional
Double-Precision (DP) Floating-point instructions	No	No	No	No	No	Optional	No	No	No	Optional
Half-Precisions (HP)	No	No	No	No	No	No	No	No	No	Optional
TrustZone instructions	No	No	No	No	No	No	Optional	Optional	Optional	Optional
Co-processor instructions	No	No	No	No	No	No	No	Optional	Optional	Optional
Helium technology	No	No	No	No	No	No	No	No	No	Optional
Interrupt latency (if zero-wait state RAM)	16 cycles	15 cycles	23 for NMI 26 for IRQ	12 cycles	12 cycles	12 cycles 14 worst case	15 no security ext 27 security ext	12 no security ext ?? security ext	TBD	TBD

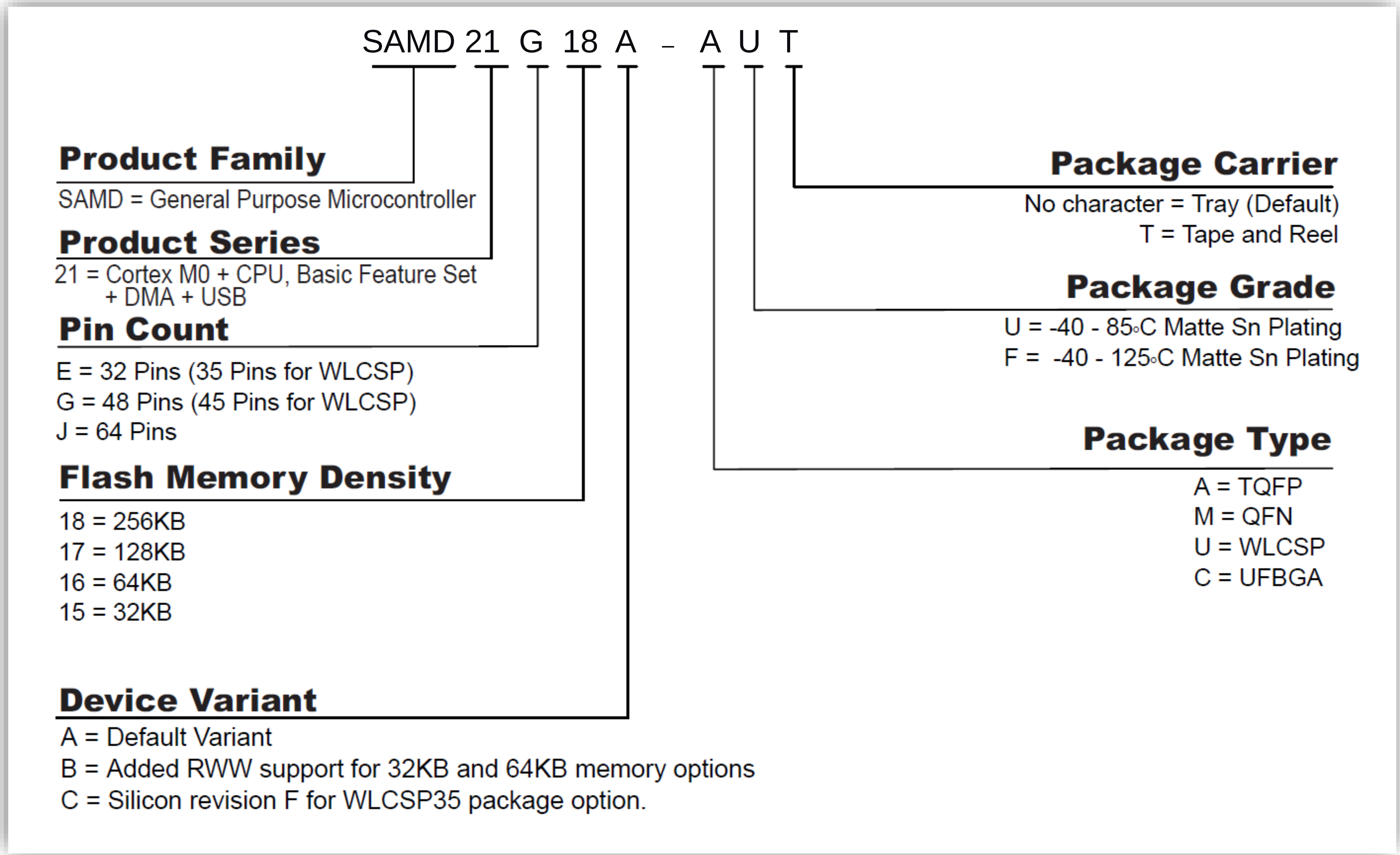
Link : https://en.wikipedia.org/wiki/ARM_Cortex-M

SAMD21 Memory

- Up to 256KB Flash and 32KB SRAM



Part Number



MICROCHIP

Soft Request In this Course

- **MPLAB® X IDE v6.25 https://www.microchip.com/mplab/mplab-x-ide**
- **MPLAB® XC32 Compiler v4.60 https://www.microchip.com/mplab/compilers**
- **MPLAB® Code Configurator (MCC) v5.5.2**
- **MCC Core v5.7.1**
- **MCC Harmony Framework (From X IDE MCC Content Manager)**
  - MCC Harmony Core v1.5.5
  - Harmony 3 - Chip Support Package
    - csp v3.22.1
  - Harmony 3 - Harmony Services v1.5.0
  - Harmony 3 - Reference Apps v1.7.0
  - arm CMSIS v5.9.0
- **MPLAB® X IDE Packs**
  - SAMD21_DFP v3.6.144 (Device Family Pack)
  - CMSIS v5.4.0 or Newest (Device Family Pack)
  - SNAP_TP v2.8.1532 or Newest (Tool Firmware Pack)

➢ **MCC Core and Framework Update from X IDE > MCC > Content Manager**

➢ **SAMD21_DFP and SNAP_TP Update from X IDE > Tools > Packs**

(Lasted Update : 2025/03/17)

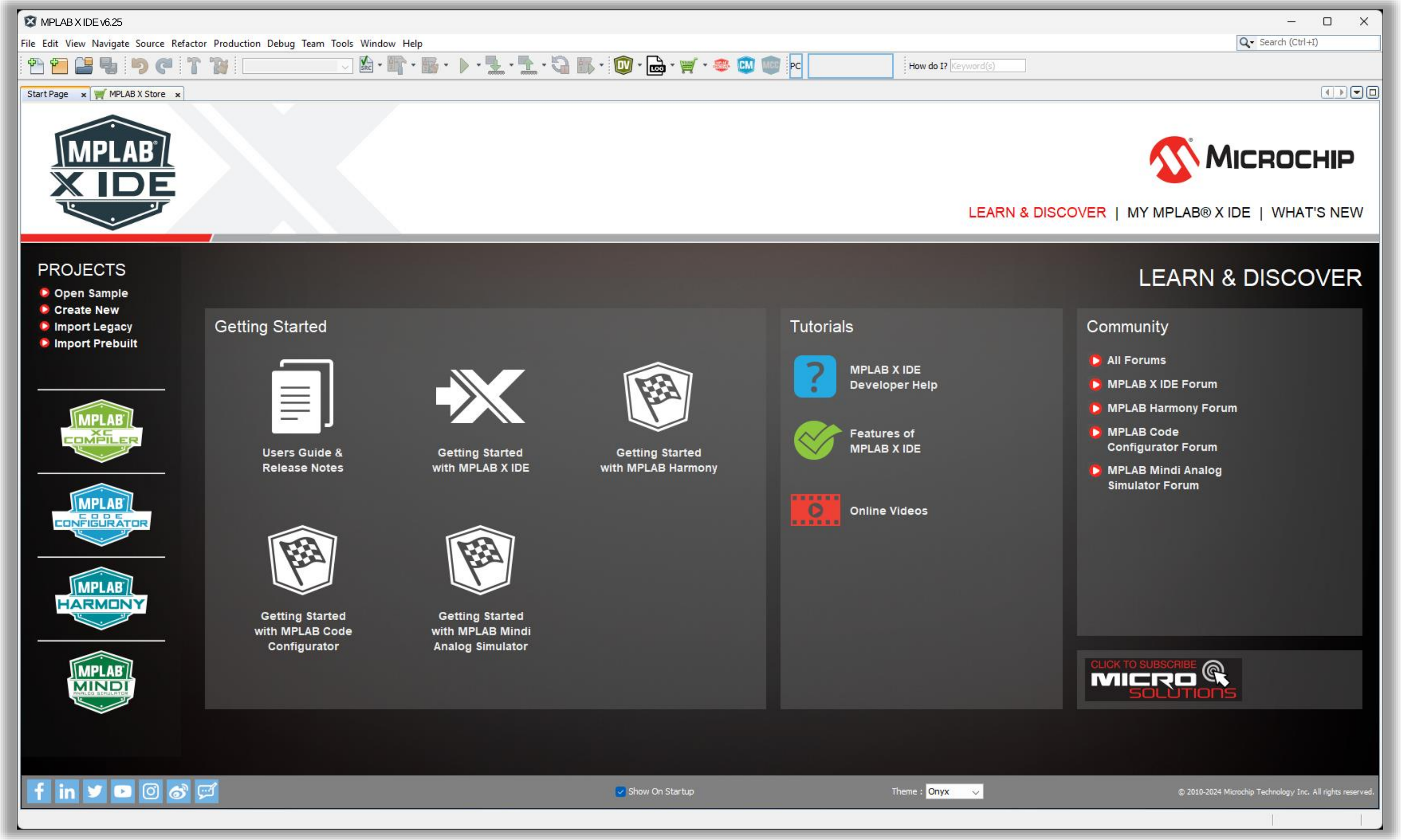
MICROCHIP

MPLAB® X IDE, MCC Harmony and Development Tools Introduction

All in One Development

MPLAB® X IDE

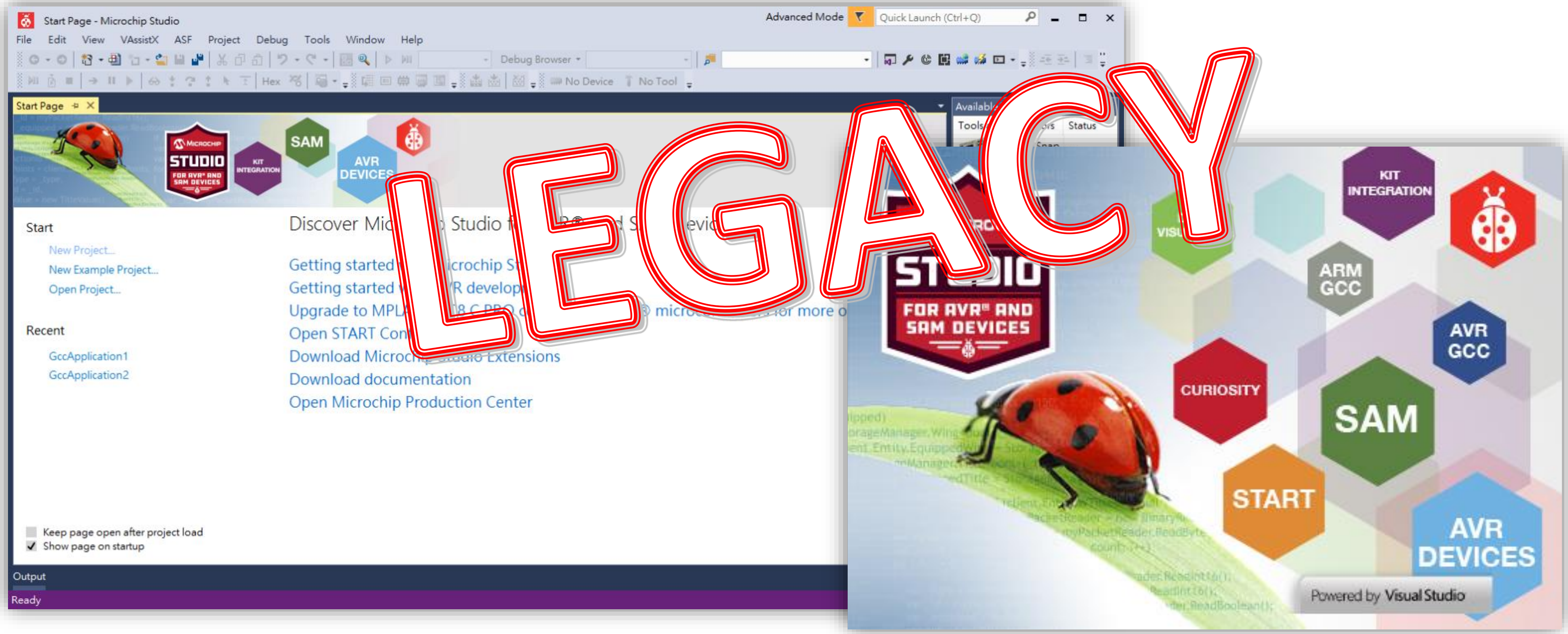
- **New generation integrated Development Tools, Support PIC, dsPIC, AVR, CEC and SAM Series MCU/MPU, Provide Plug-in function to extend more advance function. Java Based, Cross platform, Current version is v6.25.**



AVR/SAM Series Development

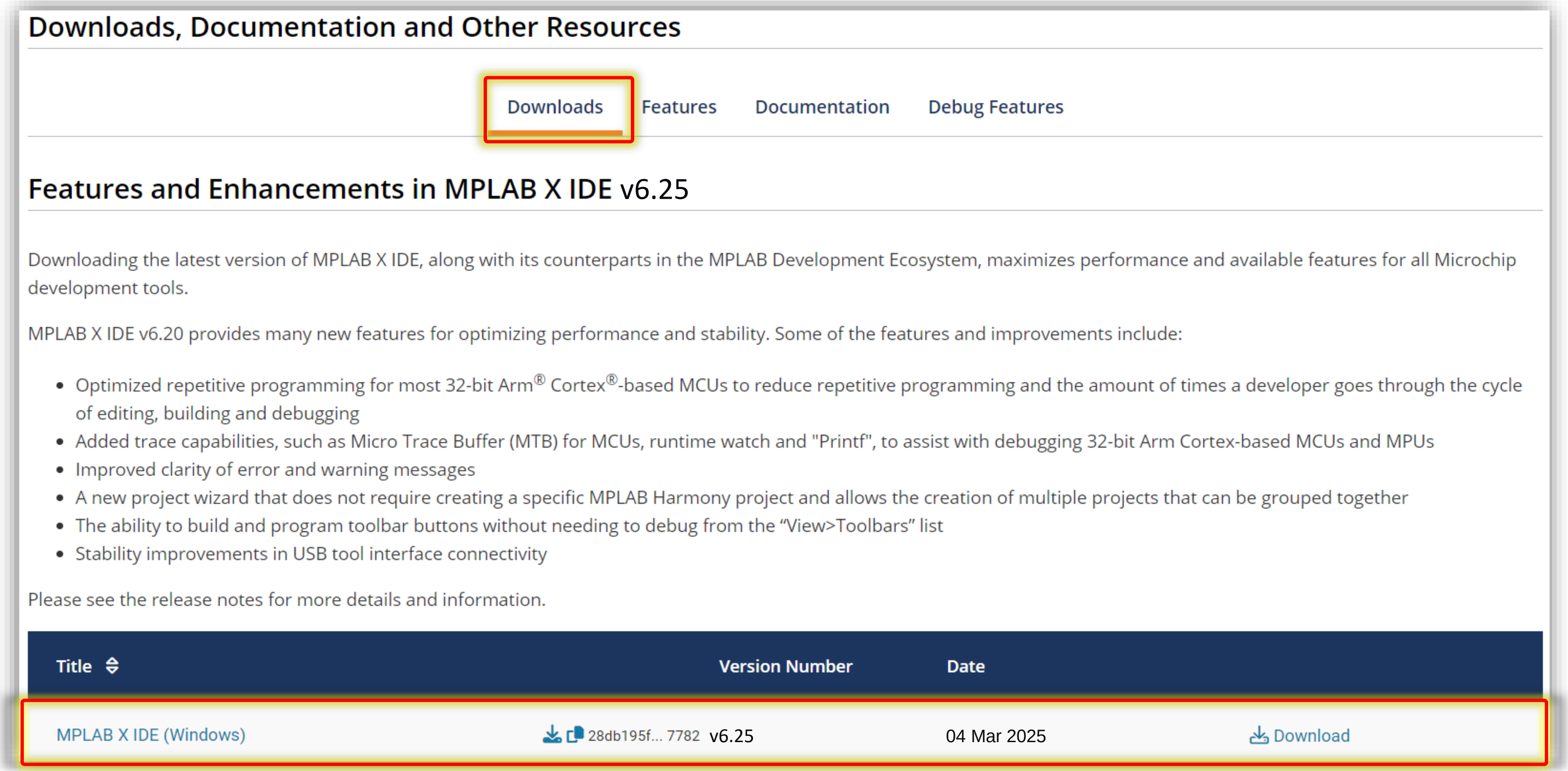
Atmel Studio 7 → Microchip Studio

- New generation integrated Development Tools, Support AVR, SAM Series MCU/MPU, Current version is v7.0.2594.
- The GCC compilers already include support AVR/SAM series.



Official Site Download Page

www.microchip.com/mplabX



Downloads, Documentation and Other Resources

Downloads | Features | Documentation | Debug Features

Features and Enhancements in MPLAB X IDE v6.25

Downloading the latest version of MPLAB X IDE, along with its counterparts in the MPLAB Development Ecosystem, maximizes performance and available features for all Microchip development tools.

MPLAB X IDE v6.20 provides many new features for optimizing performance and stability. Some of the features and improvements include:

- Optimized repetitive programming for most 32-bit Arm® Cortex®-based MCUs to reduce repetitive programming and the amount of times a developer goes through the cycle of editing, building and debugging
- Added trace capabilities, such as Micro Trace Buffer (MTB) for MCUs, runtime watch and "Printf", to assist with debugging 32-bit Arm Cortex-based MCUs and MPUs
- Improved clarity of error and warning messages
- A new project wizard that does not require creating a specific MPLAB Harmony project and allows the creation of multiple projects that can be grouped together
- The ability to build and program toolbar buttons without needing to debug from the "View>Toolbars" list
- Stability improvements in USB tool interface connectivity

Please see the release notes for more details and information.

Title		Version Number	Date	
MPLAB X IDE (Windows)	28db195f... 7782	v6.25	04 Mar 2025	Download

Programmer/Debugger

MPLAB® PICkit 5 (Part No. PG164150)

- **The MPLAB® PICkit™ 5 In-Circuit Debugger/Programmer allows fast and easy debugging and programming of all PIC®, dsPIC®, AVR, SAM and CEC flash microcontrollers and microprocessors.**
- **Features**
  - Improved Programmer-to-Go (PTG) support.
    - Connect wirelessly from your smartphone via Bluetooth®
    - Select from multiple saved program images on the SD Card
    - Start programming from the app or by pressing on the logo
  - Supply 150 mA to the target.
  - Option to be self-powered from the target (2.7V to 5V).
  - USB Type-C® connector and cable.
  - No external power needed when the device is powered.
  - Use the 8-pin single in-line header.

$94.99

Programmer/Debugger

MPLAB® PICkit™ Basic (Part No. PG164110)

- **The MPLAB® PICkit™ Basic In-Circuit Debugger/Programmer allows fast and easy debugging and programming of most PIC® and AVR® MCUs, dsPIC® DSCs and SAM Arm® Cortex®-M based MCUs and MPUs.**
- **Features**
  - USB Type-C® interface for easy PC connection
  - 8-pin single in-line header with beginner-friendly flying leads
  - Target voltage of 1.20V to 5.5V
  - Supports 4-wire JTAG, Serial Wire Debug (SWD), UPDI, PDI, SPI, debugWIRE and TPI programming
  - Backward compatible with two-wire JTAG and ICSP
  - Includes an adapter to convert from the 8-pin inline to the standard 10-pin Arm SWD connector
  - Easy firmware recovery with a push button

$29.99

Programmer/Debugger

MPLAB® SNAP (Part No. PG164100)

- **The MPLAB® SNAP In-Circuit Debugger/Programmer allows fast and easy debugging and programming of most PIC®, dsPIC®, AVR, SAM and CEC flash microcontrollers.**
- **Features**
  - Affordable performance
  - Matches silicon clocking speed
  - Target voltage of 1.20V to 5.5V
  - Portable USB-powered
  - CE and RoHS-compliant
  - 8-pin single in-line header
  - Backward compatible for demo boards, headers and target systems using 2-wire JTAG and ICSP

$14.99

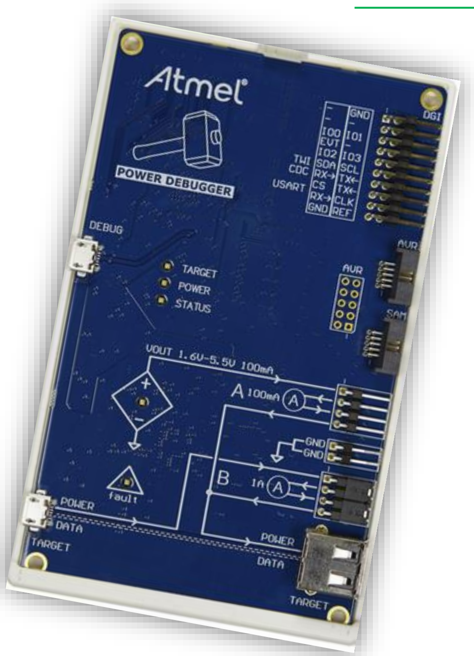
MICROCHIP

Programmer/Debugger

Power Debugger (Part No. atpowerdebugger)

- **Simultaneous debug & power measurement**
  - Debugging over CMSIS-DAP
  - Extended to support SWO trace, same as Atmel-ICE
- **Two independent channels for power measurements**
  - Channel A for high accuracy, lower currents
  - Channel B for higher currents with lower resolution
- **CDC virtual COM port**
- **DGI for streaming application & power data**
  - Open specification, docs provided
- **Provides target power, 1.6 – 5.5V,**
  up to 100mAStatus LEDs for debugging & target supply

$290.94



Programmer/Debugger

Atmel-ICE (Part No. ATATMEL-ICE)

- **Supports programming and debugging all Cortex-M based SAM, AVR and UC3 based parts**
  - Programming/ Debugging AVR/SAM MCU/CPU over JTAG, SD, PDI, SPI, TPI and aWire etc.
  - Target operating voltage range 1.62V to 5.5V
  - ITM serial trace support on Cortex-M based SAM MCUs
- **Supported in Atmel Studio, IAR, Keil and more**
  - Any IDE that supports the CMSIS-DAP protocol can support Atmel-ICE
- **3 different editions**
  - USD 59.99 - Low cost PCBA only
  - USD 79.99 - Basic edition with most common cables
  - USD 99.99 - Including all cables and adapters

$79.99

Programmer/Debugger

Debug Adapter Board (Part No. AC102015)

- **The Debug Adapter Board Gives ICD 4 / PICkit 4 cable compatibility with Atmel-ICE, Power Debugger and J-Link style connectors.**
  It supports JTAG, SWD and ICSP
  protocols in multiple connector formats.
- **Package Contents**
  - Debugger Adapter Board
  - Cable, FEMALE FLAT RIBBON IDC CONNECTOR, 15CM
  - Cable, FEMALE FLAT RIBBON IDC CONNECTOR, 12CM
  - 8 CONDUCTOR MODULAR CABLE, 6"
  - Cable, HCSD Dual Row Socket IDC Assembly 14pin (7 ea) 10"

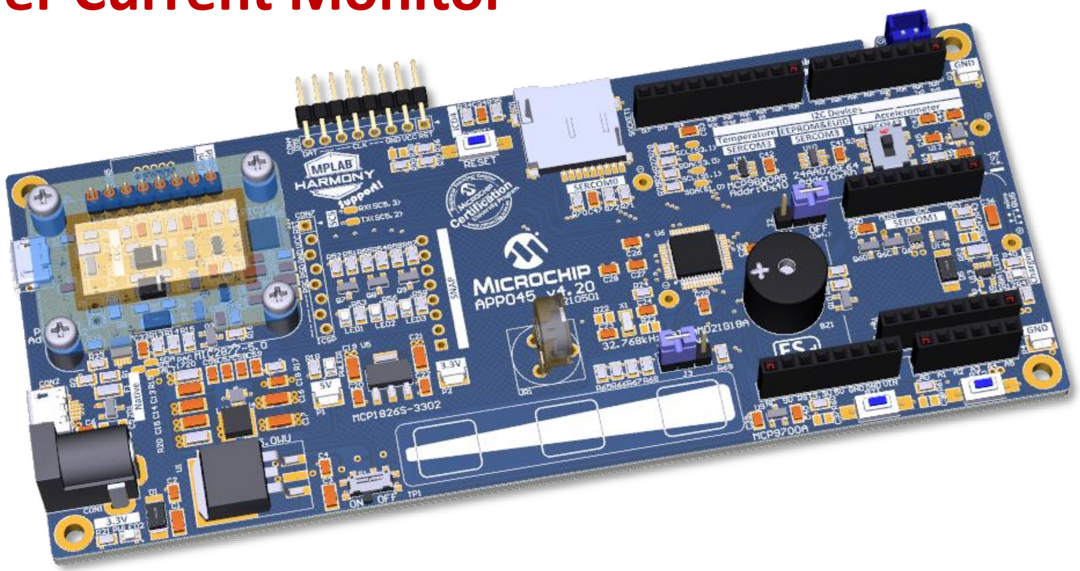
$34.99

MICROCHIP

SAMD21 EVB
APP045 v4.2 ES.d Introduction

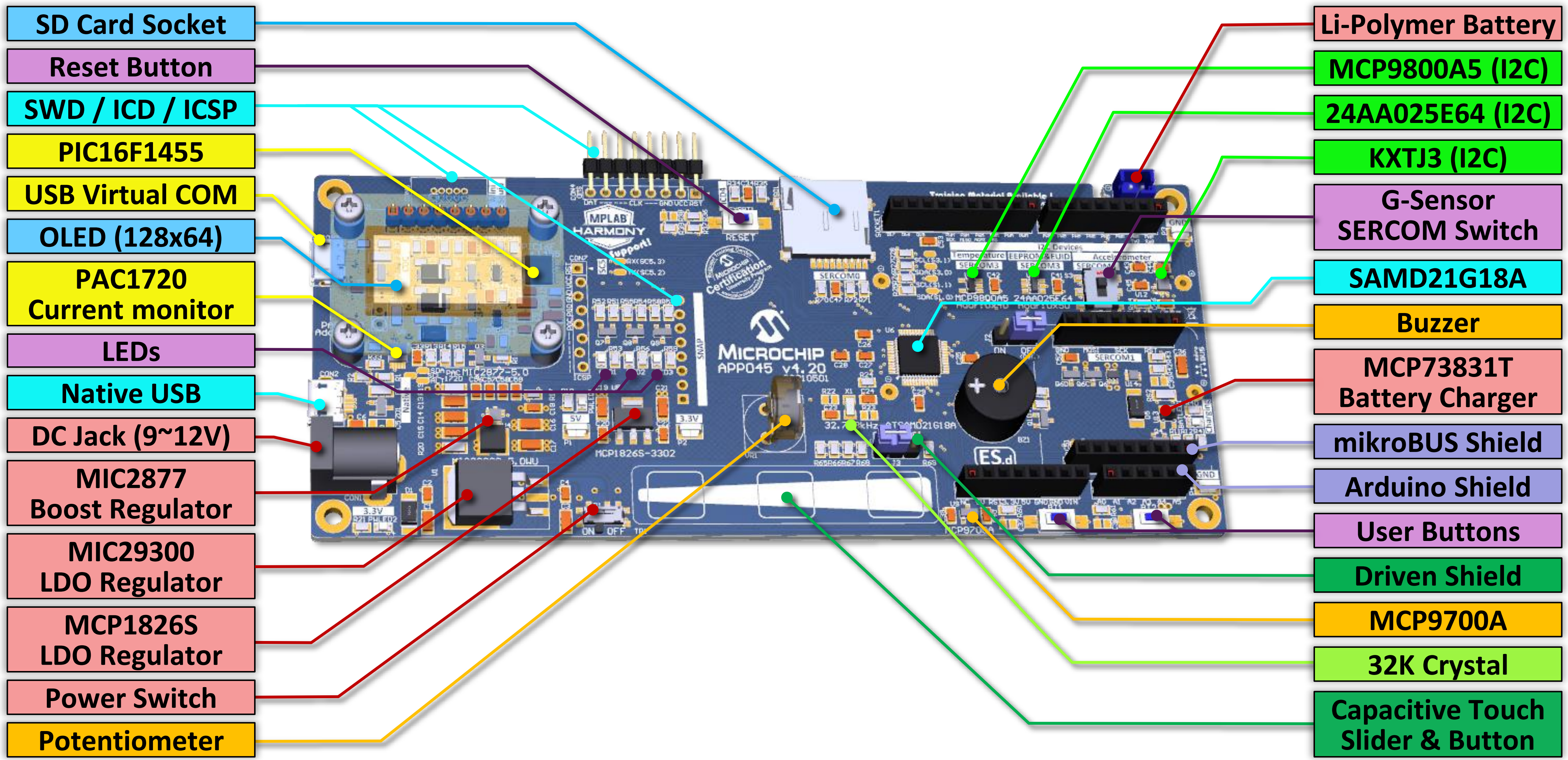
APP045 v4.2 ES.d Features

- 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)

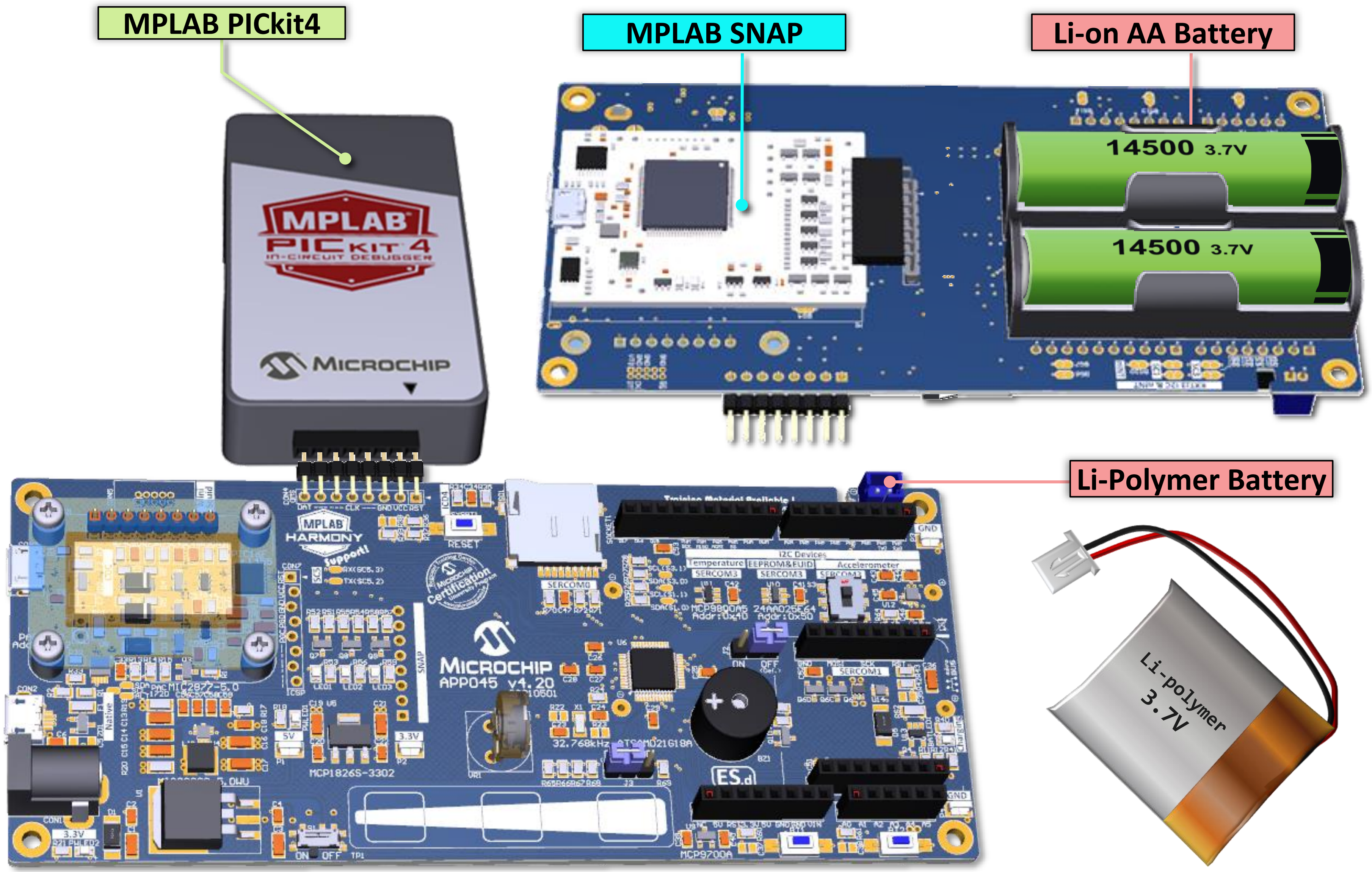


MICROCHIP

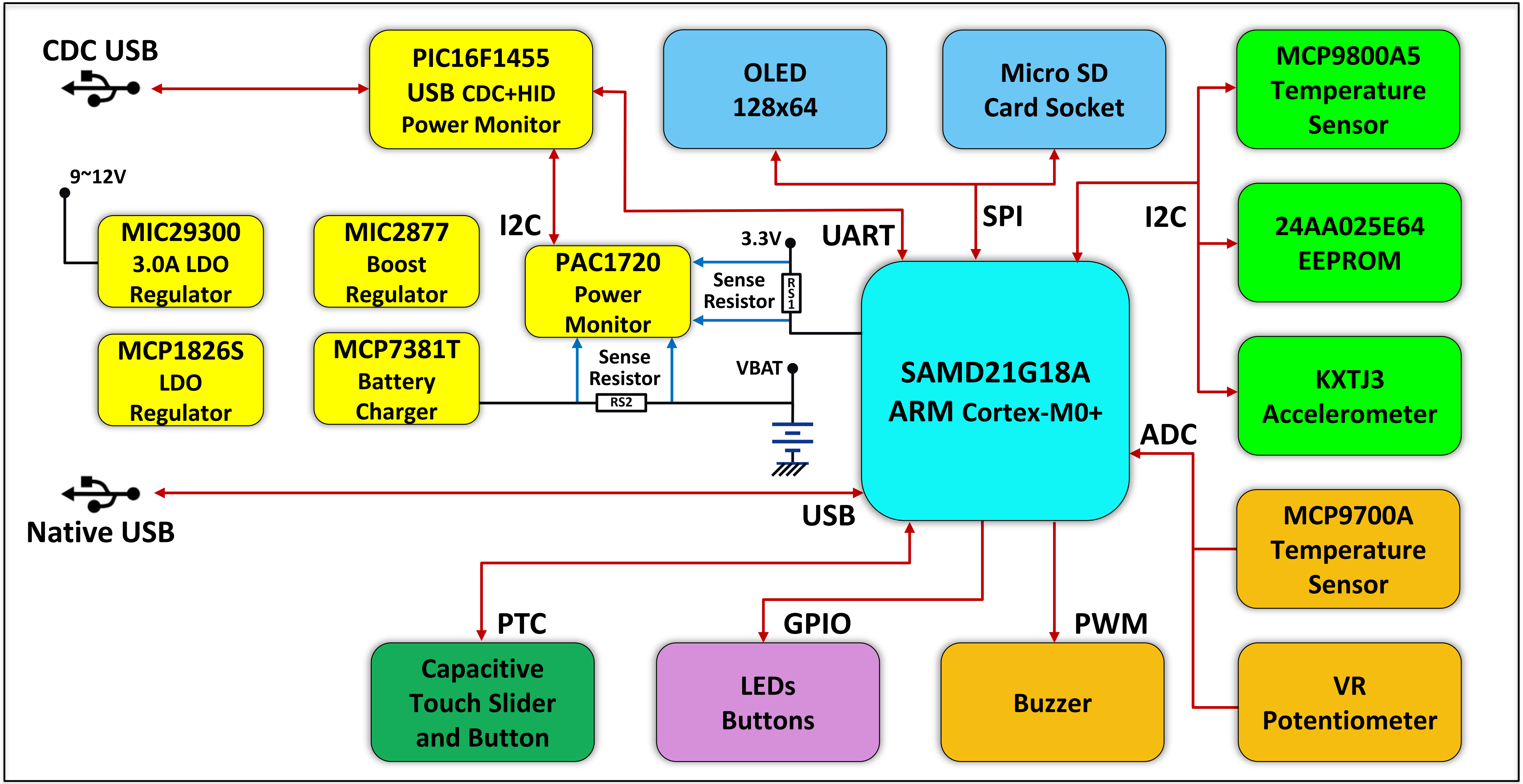
APP045 v4.2 ES.d Overview



APP045 v4.2 ES.d Programmer and Debugger



APP045 v4.2 ES.d Block Diagram



Getting Started with First Project

Lab0 First Project

- **Try to create your first project to use MPLAB X IDE.**
- **Learn how to use edit, build, project manager, debug and programming functions.**

- **How to Start？**

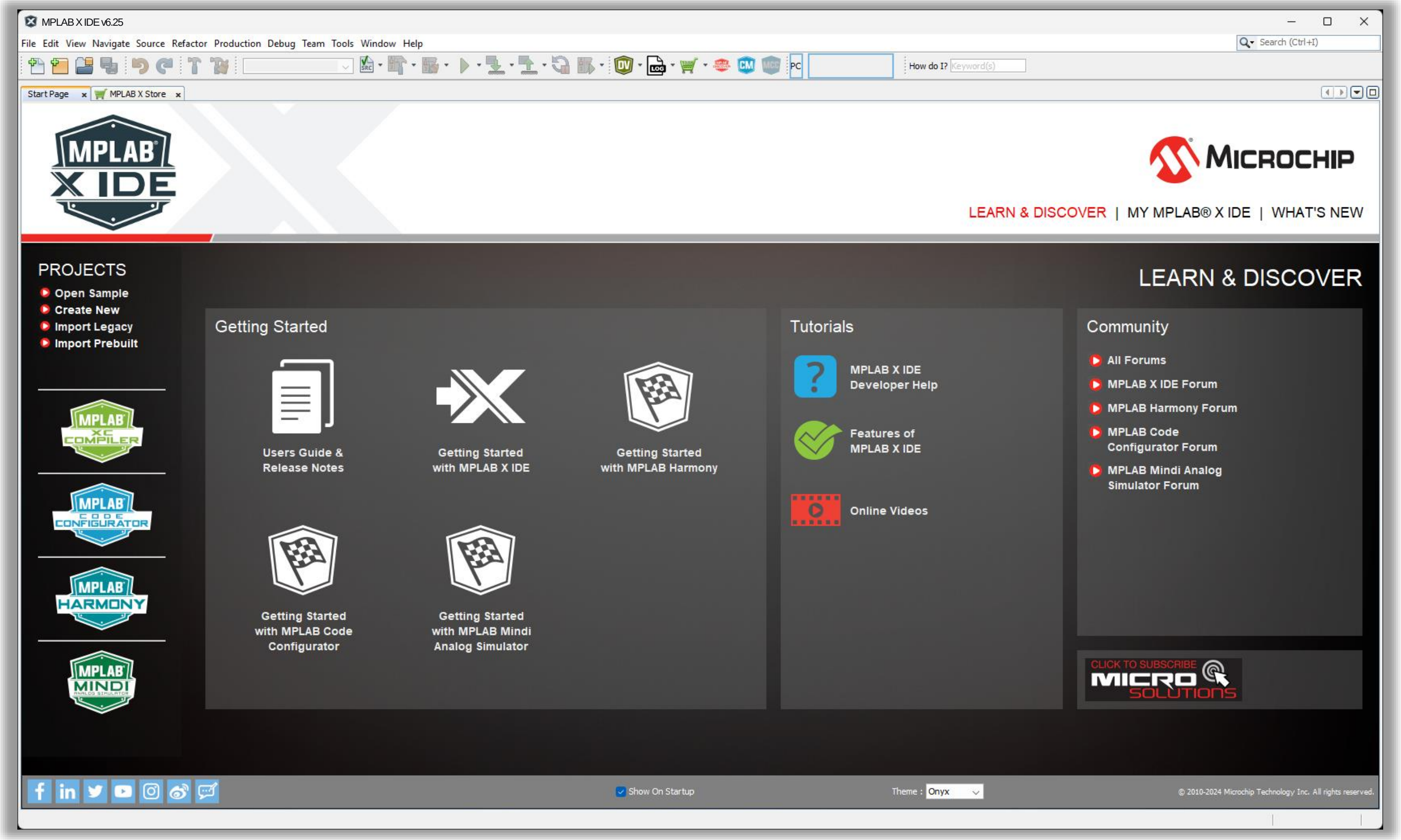
MICROCHIP

Lab0 First Project

Step 1

- Open **MPLAB X IDE v6.25** firstly.

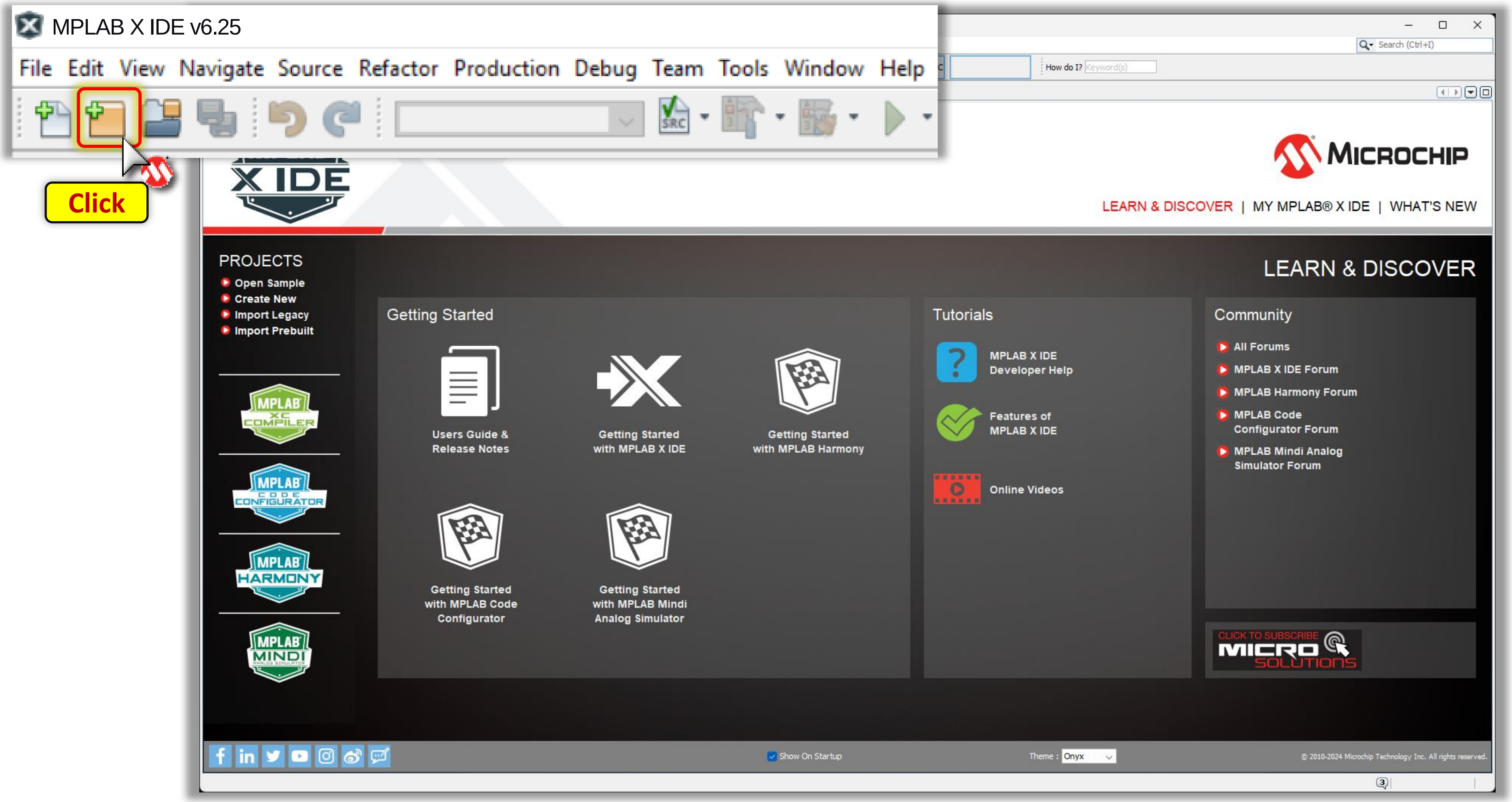
MPLAB X IDE
v6.25



Lab0 First Project

Step 2

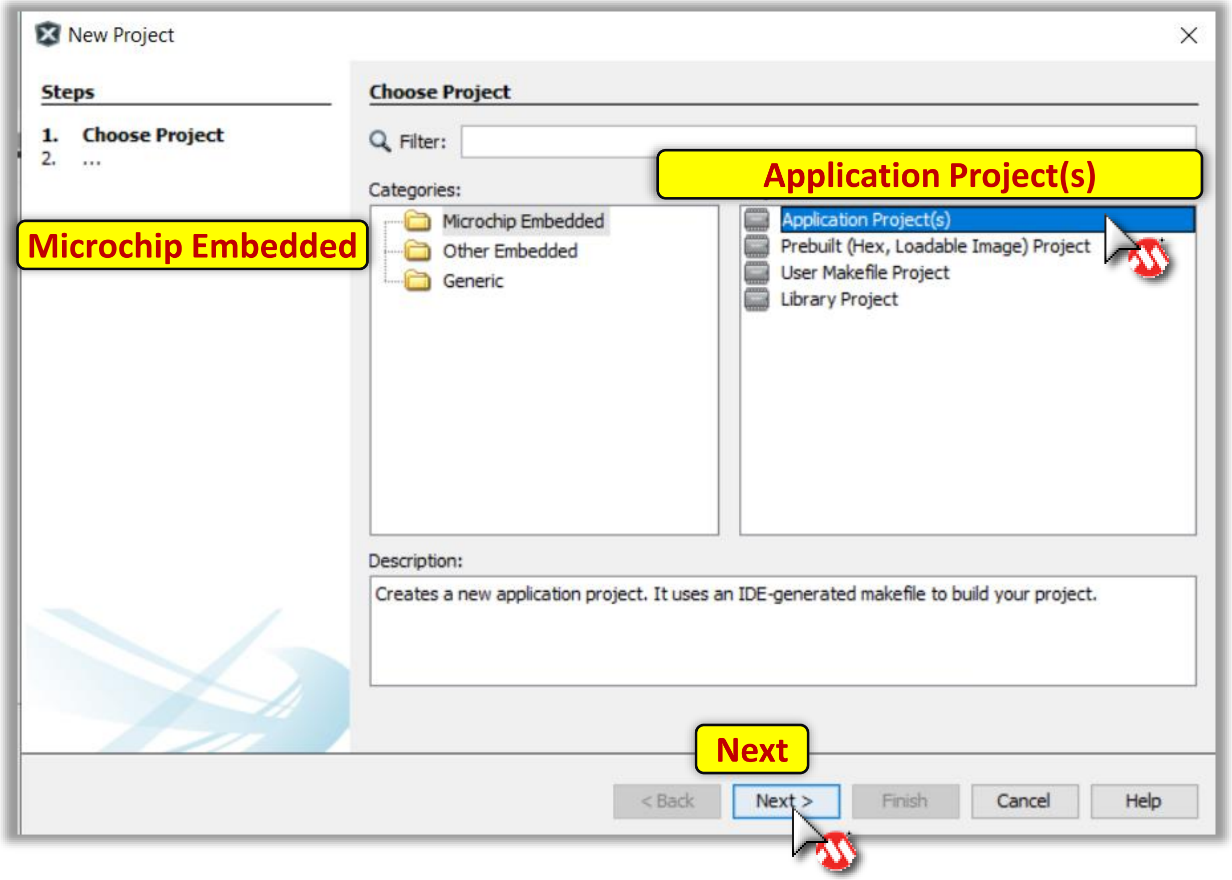
- Select to File > New > Project or Click icon



Lab0 First Project

Step 3

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



Choose

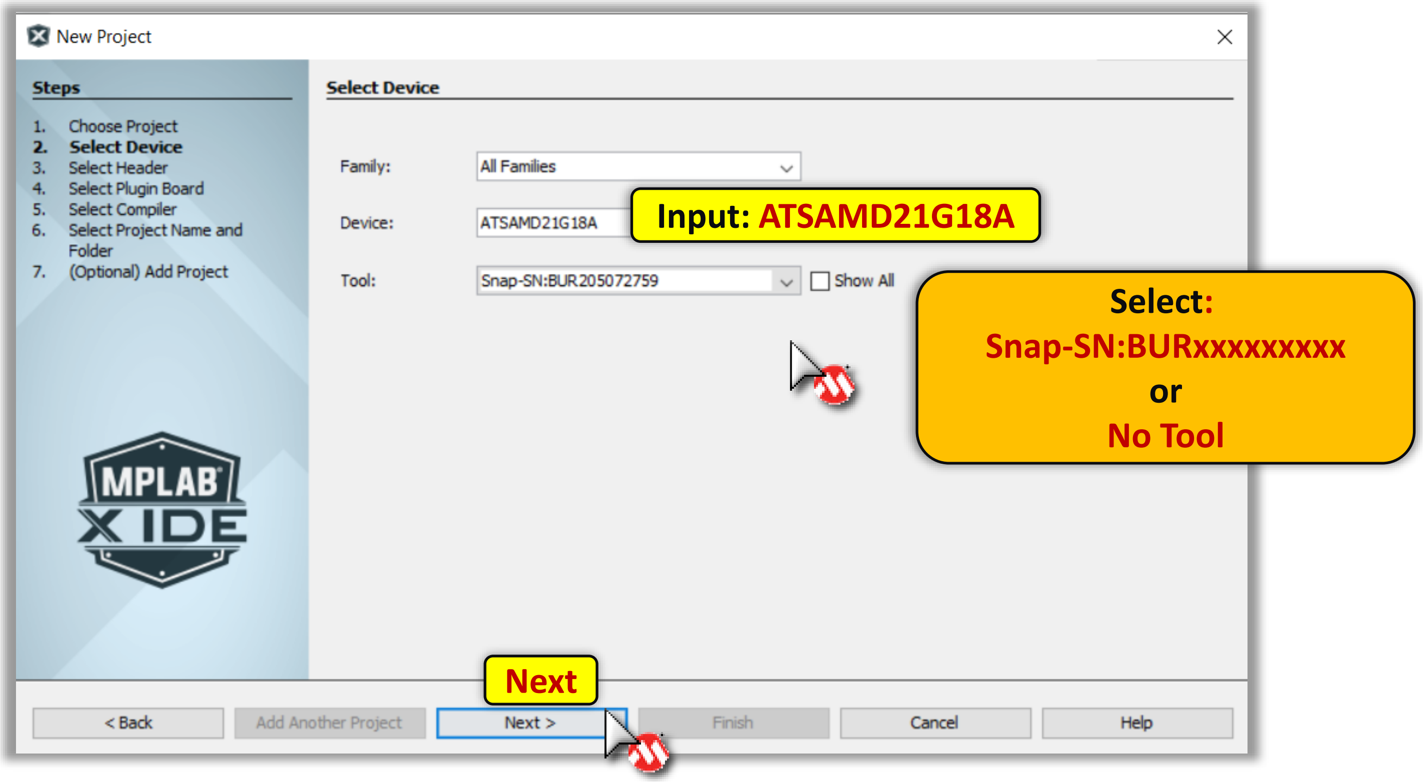
32-bit MCC Harmony Project

In MPLAB X IDE legacy version

Lab0 First Project

Step 4

- **Family** : All Familes
- **Device** : ATSAMD21G18A **Don’t miss any character**
- **Tool** : Snap-SN:BURxxxxxxxxx or No Tool

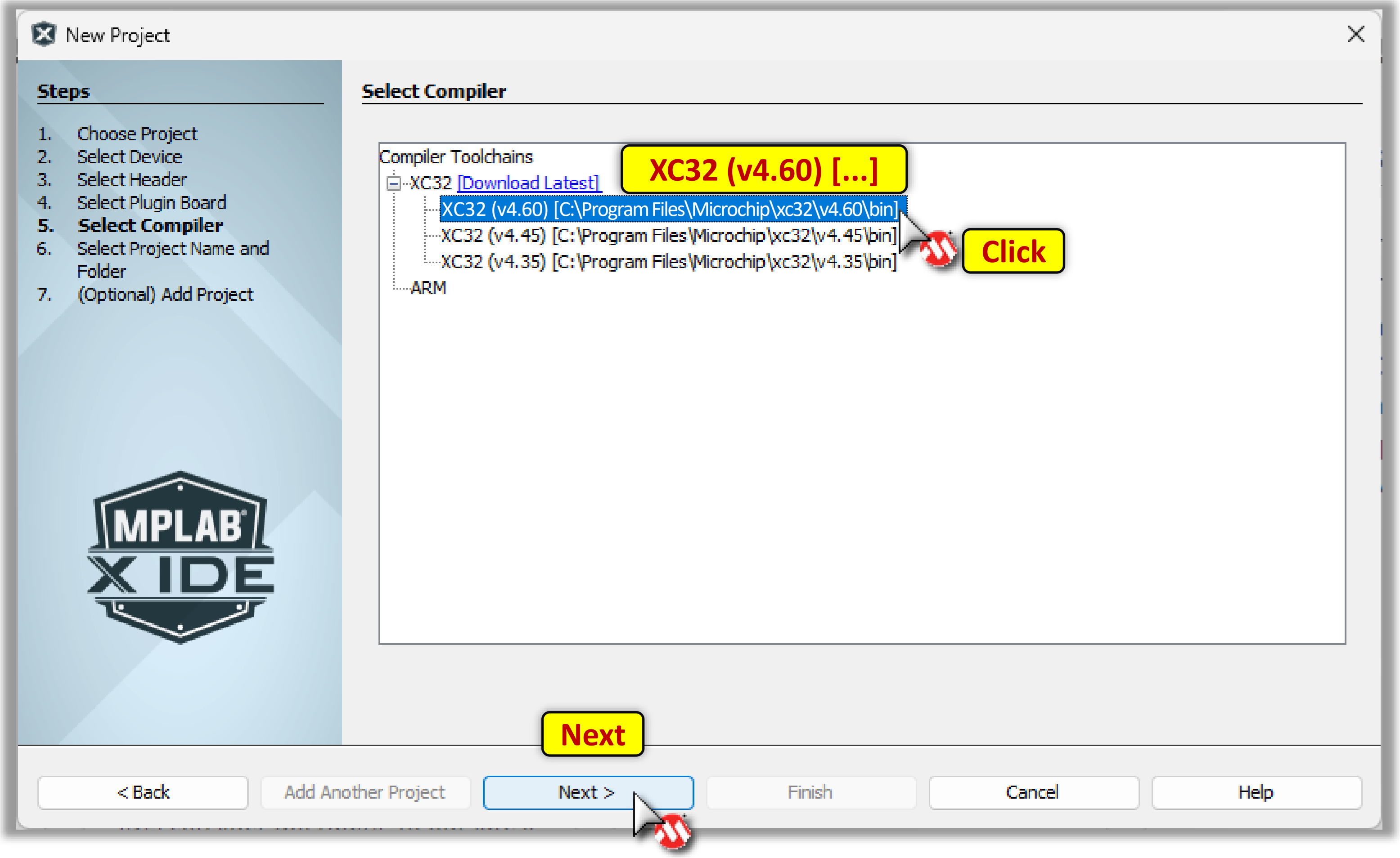


Lab0 First Project

Step 5

Assign Compiler Toolchains

- XC32: XC32 (v4.60)[…]

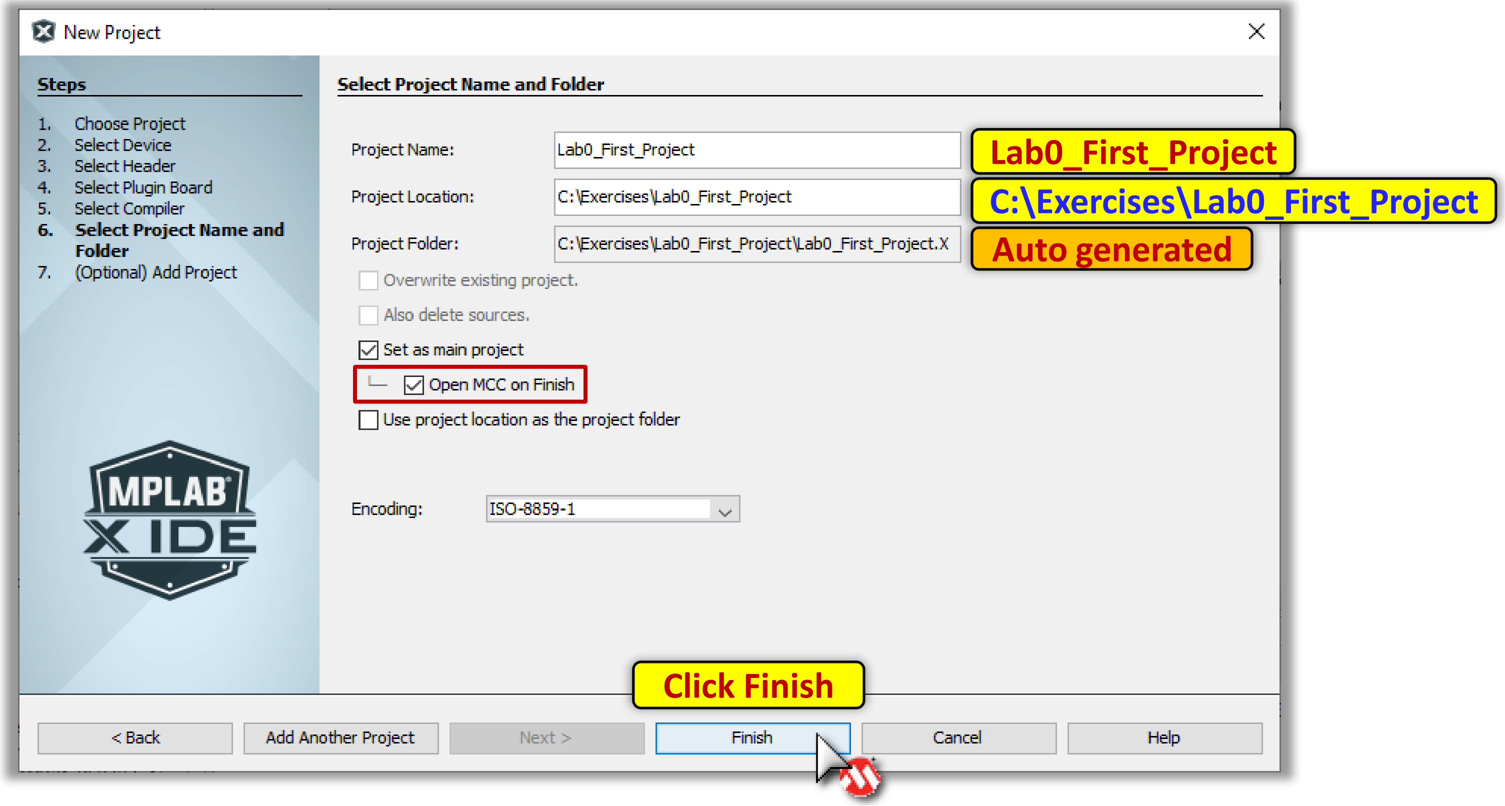


Lab0 First Project

Step 6

Configure the Project Location and Project Folder Name.

- **Project Name : Lab0_First_Project** No white space of project folder
- **Project Location : C:\Exercises\Lab0_First_Project** Don't forget Lab0_First_Project
- **Project Folder : C:\Exercises\Lab0_First_Project\Lab0_First_Project.X**

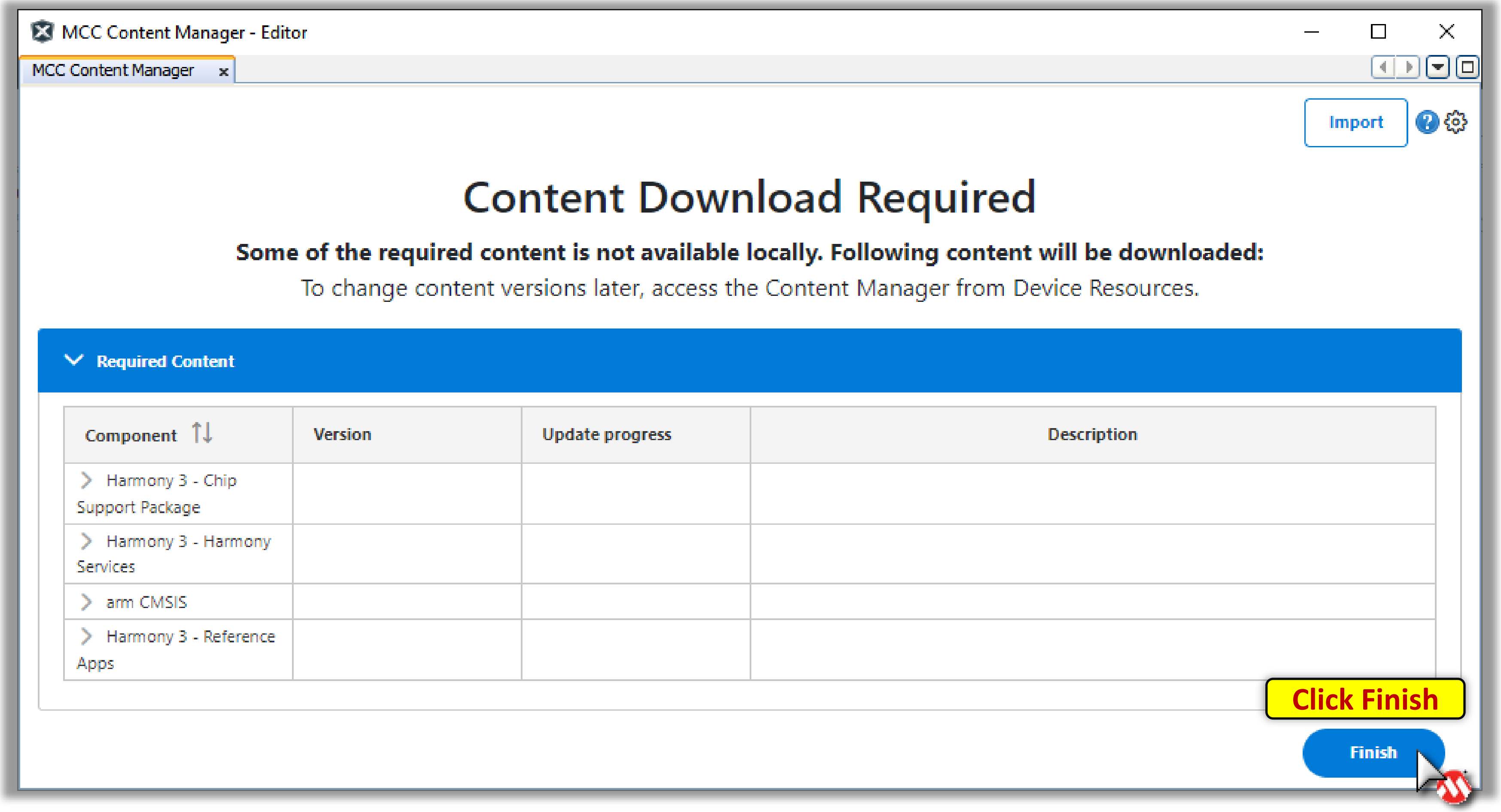


MICROCHIP

Lab0 First Project

Step 7

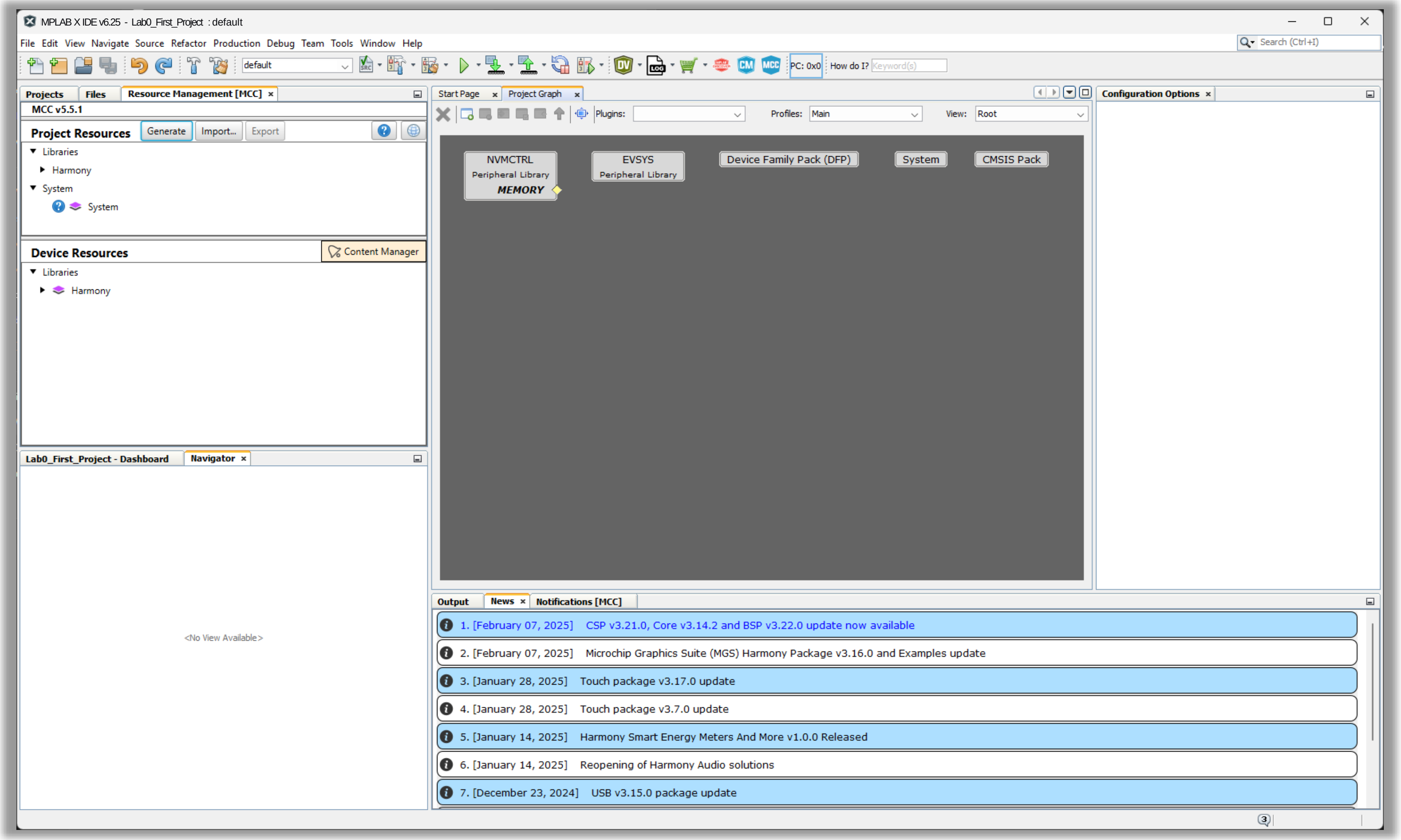
- Click Finish and Wait Download Framework. (Appears only on first execution.)



Lab0 First Project

Step 8

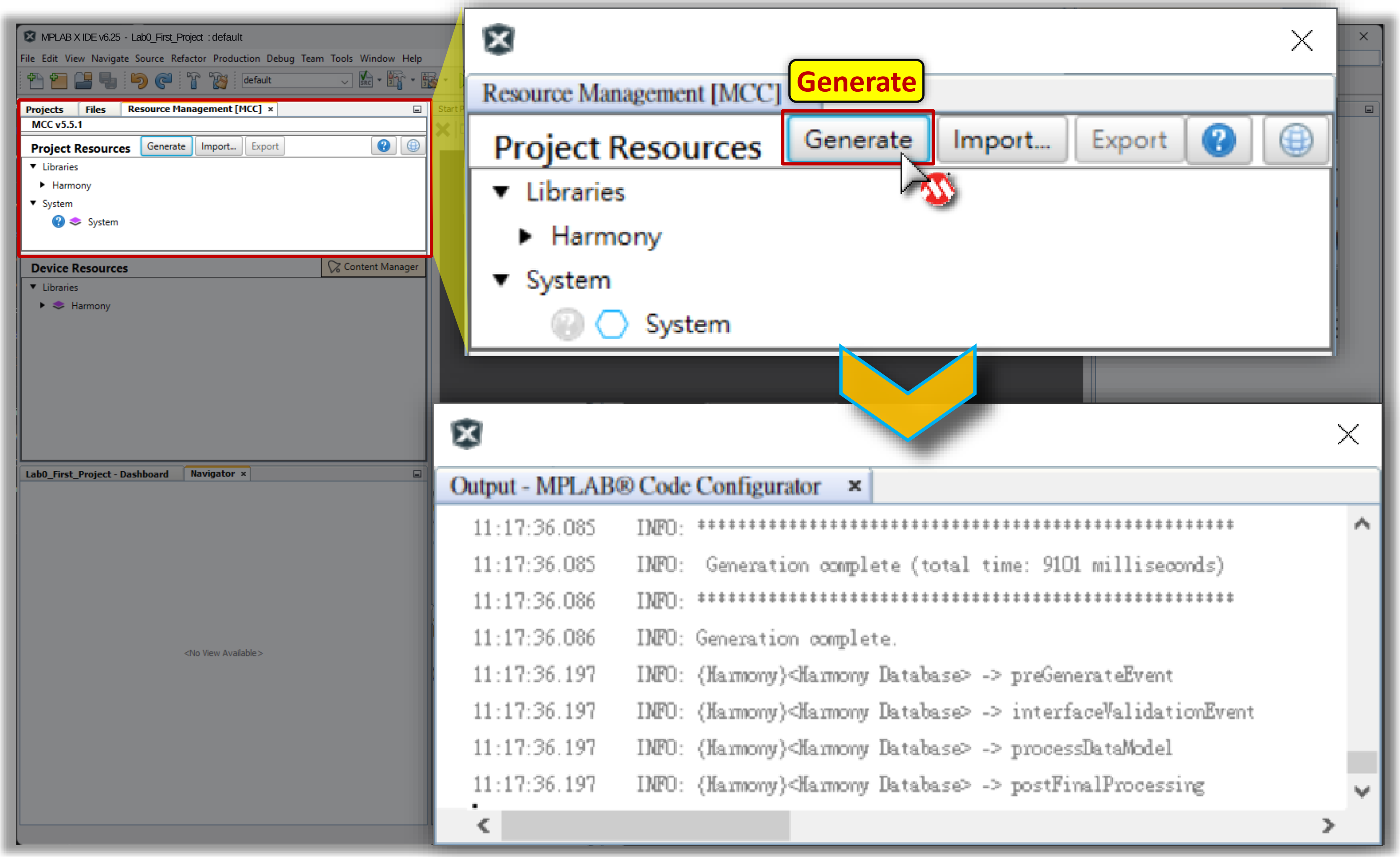
- **MCC Harmony Configurator interface show up, let's come back later.**



Lab0 First Project

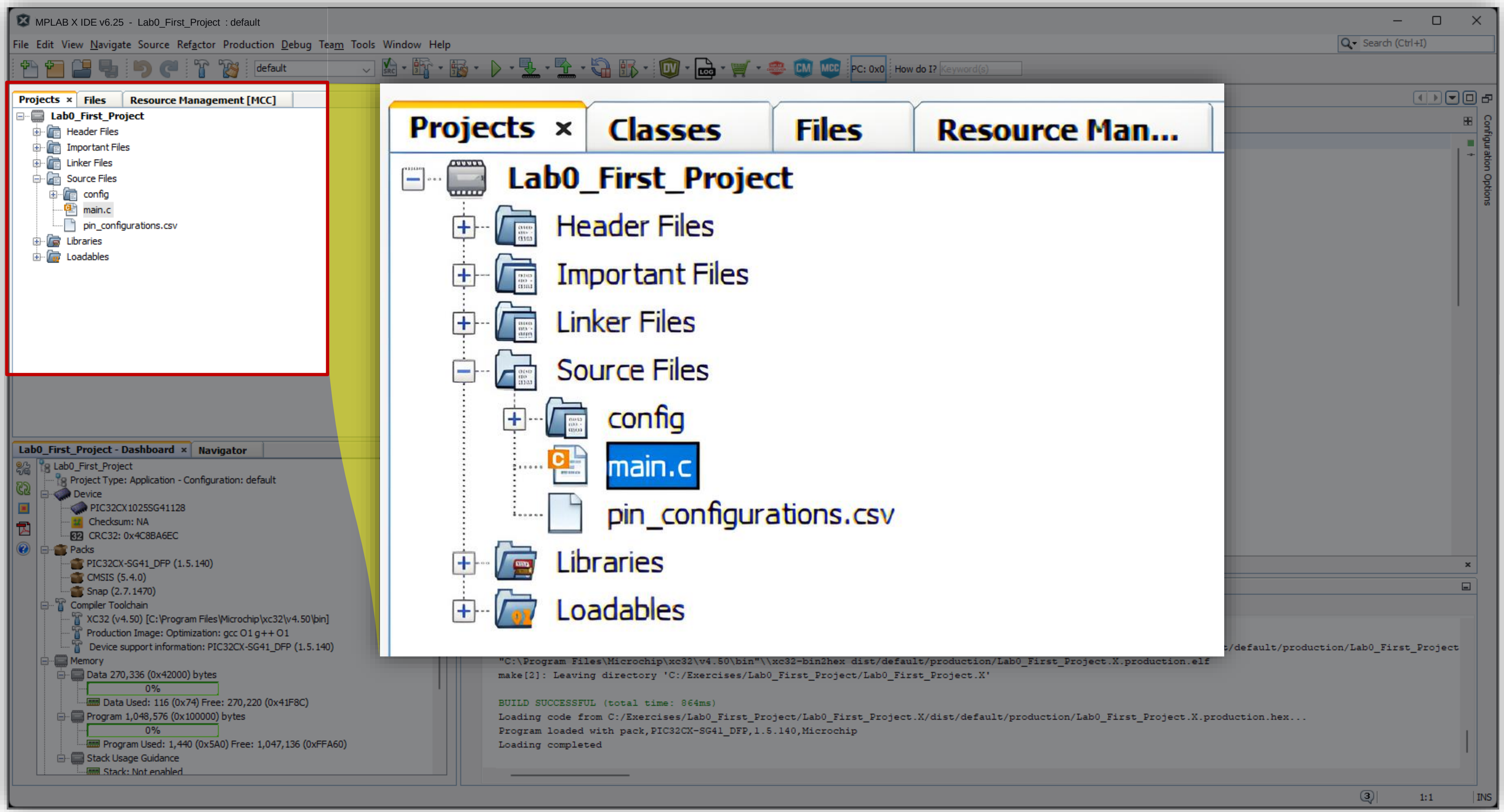
Step 9

- Click "Generate" Button to Generate your Code.



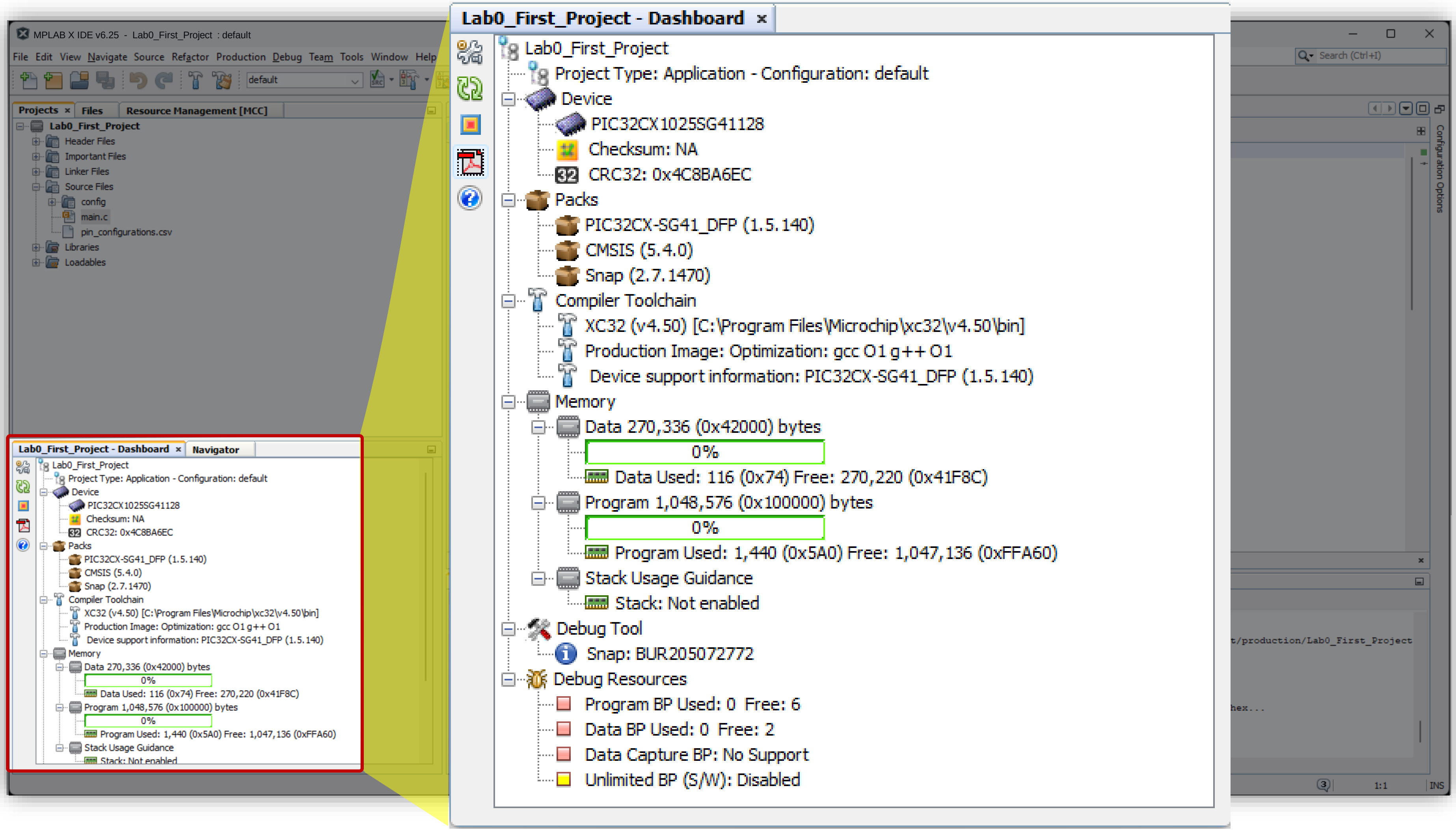
MPLAB X IDE Interface

- **Project window.**



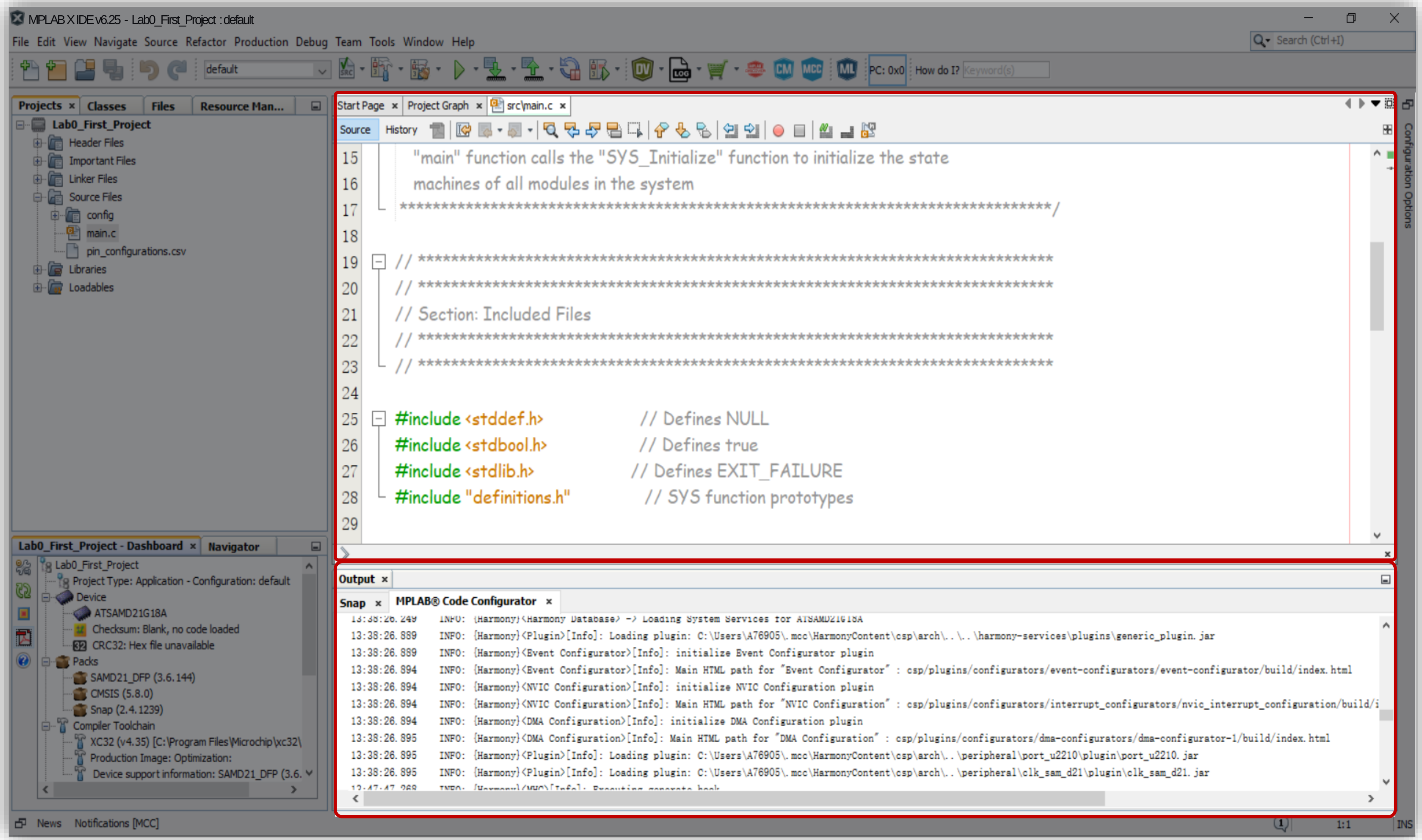
MPLAB X IDE Interface

- **Dashboard window.** (The version of modules might be different to yours.)



MPLAB X IDE Interface

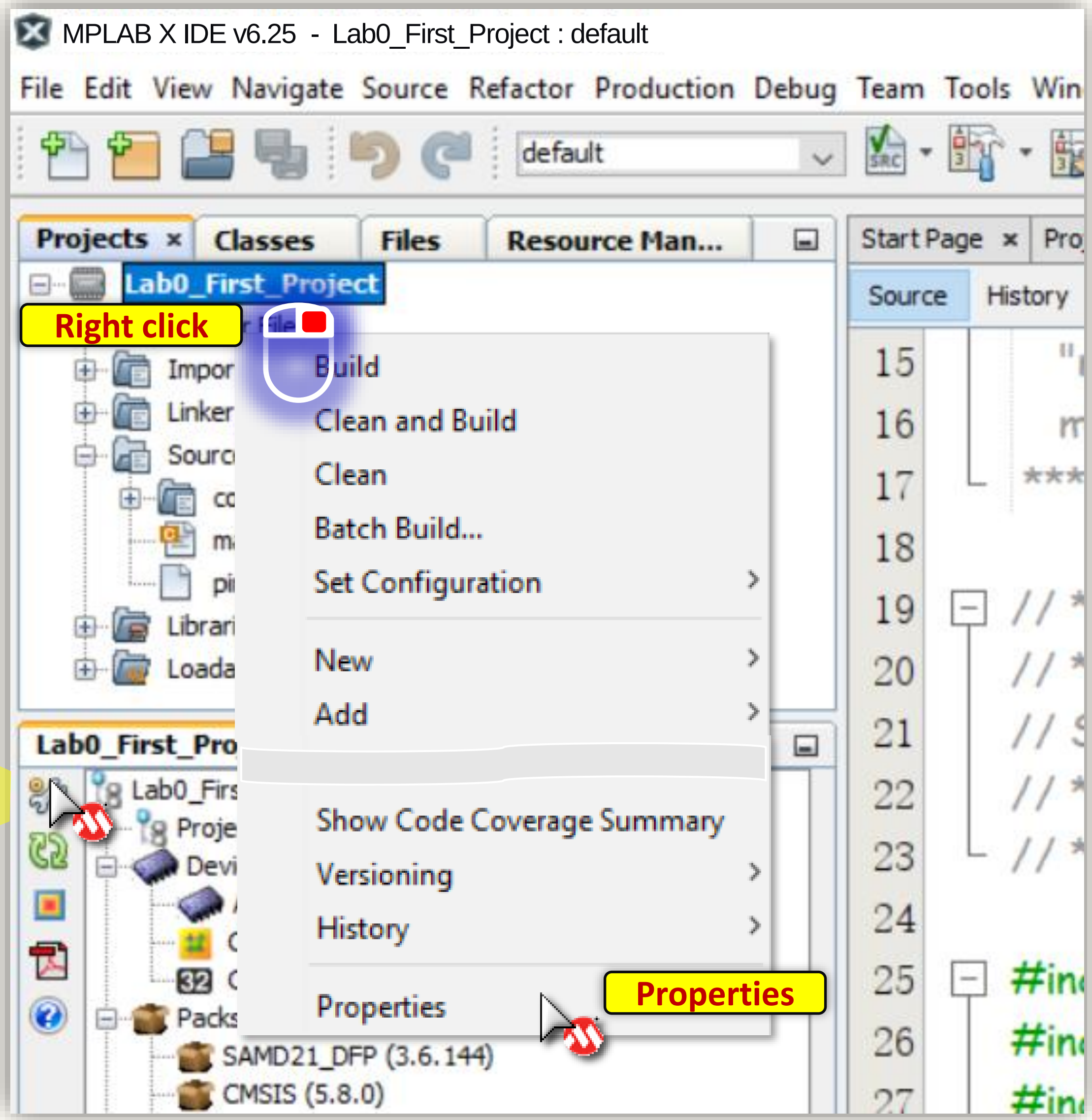
- **Source code and Output window.**



Project Properties

Hint

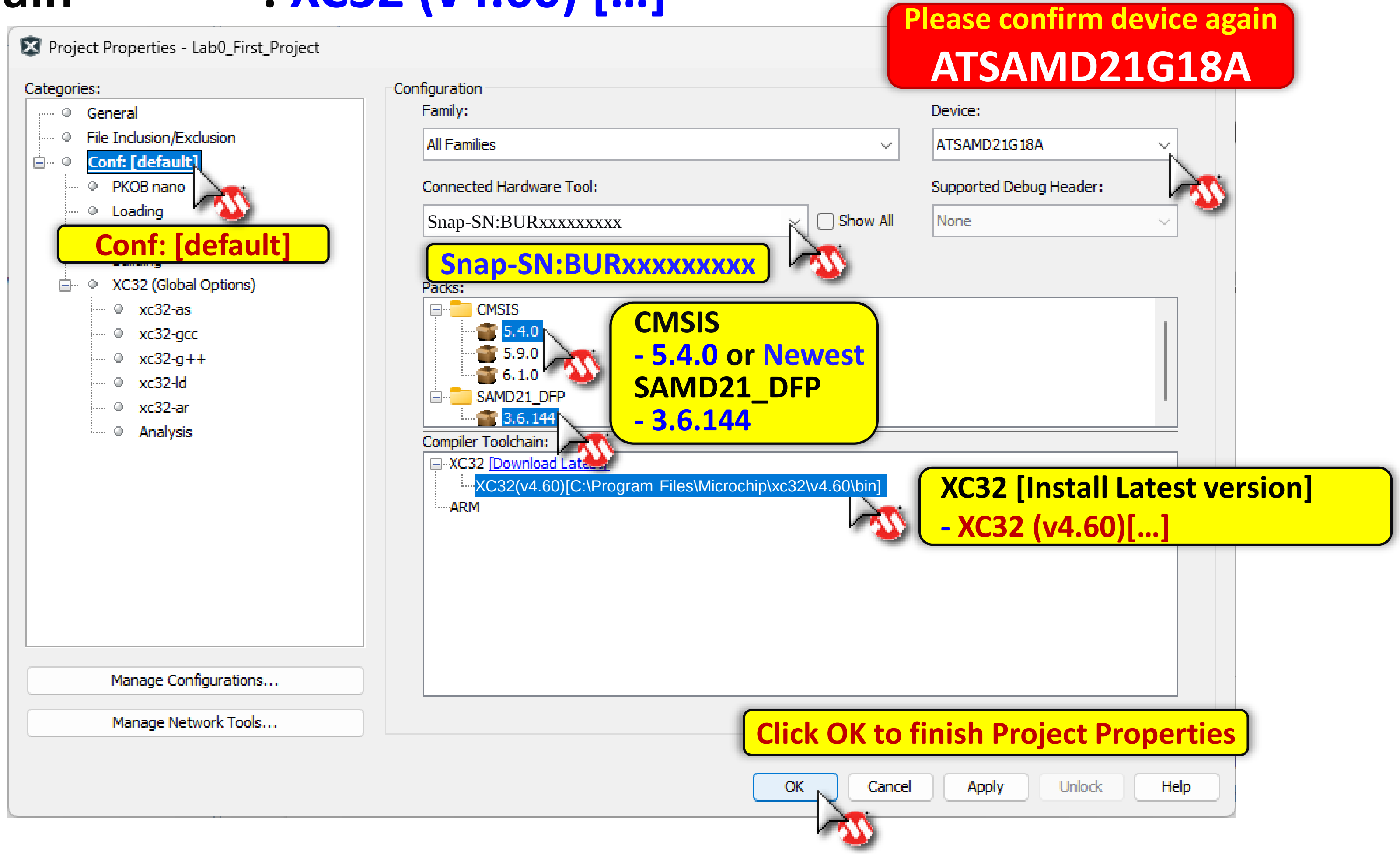
- Now we can check settings in project properties :
- Click on Dashboard or Right click on project and select to **Properties**.



MICROCHIP

Project Properties

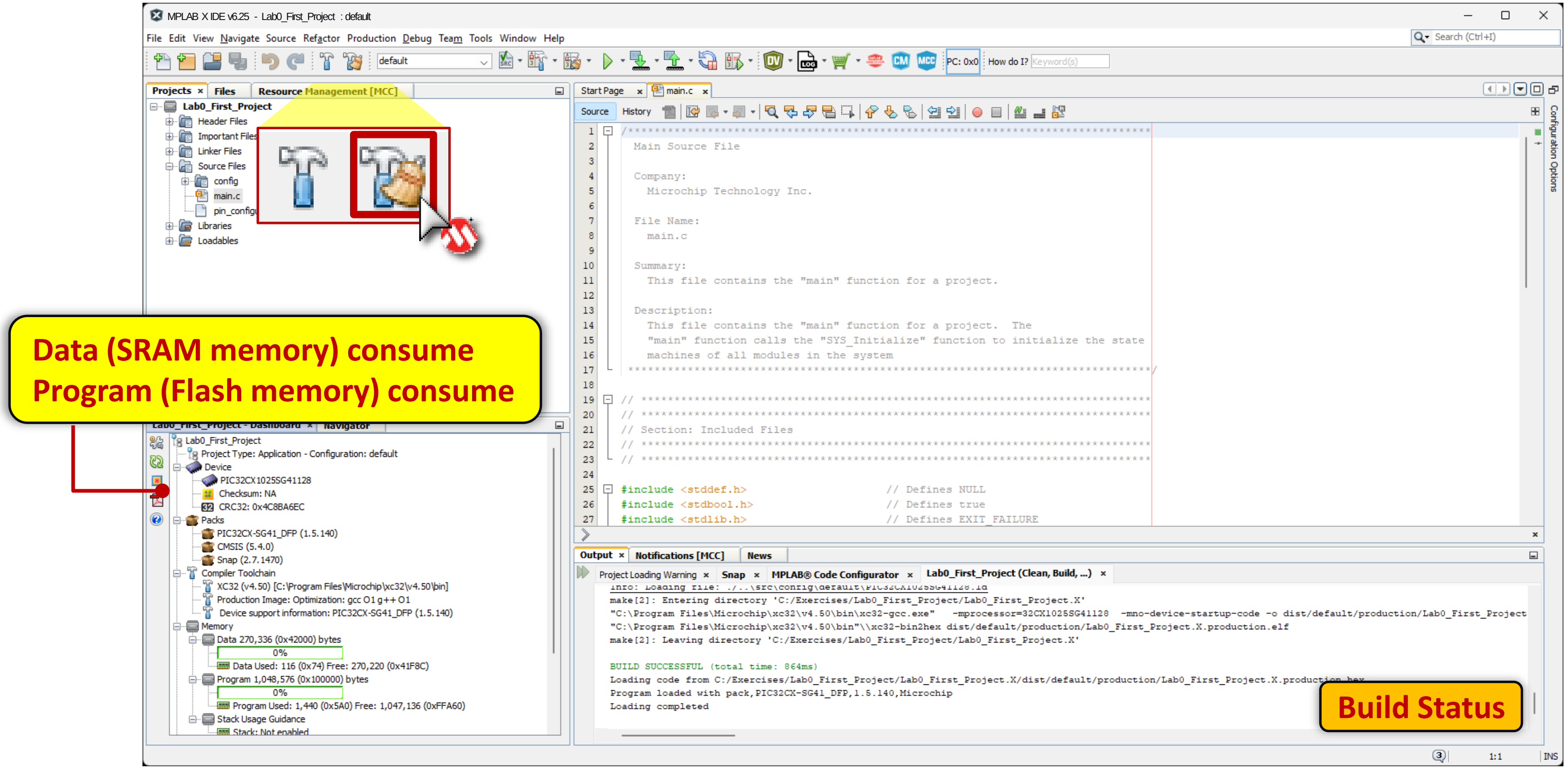
- Connected Hardware Tool : Snap-SN: BURxxxxxxxxx
- Packs : CMSIS : 5.4.0 or Newest
  SAMD21_DFP : 3.6.144
- Compiler Toolchain : XC32 (v4.60) [...]



Lab0 First Project

Step 10

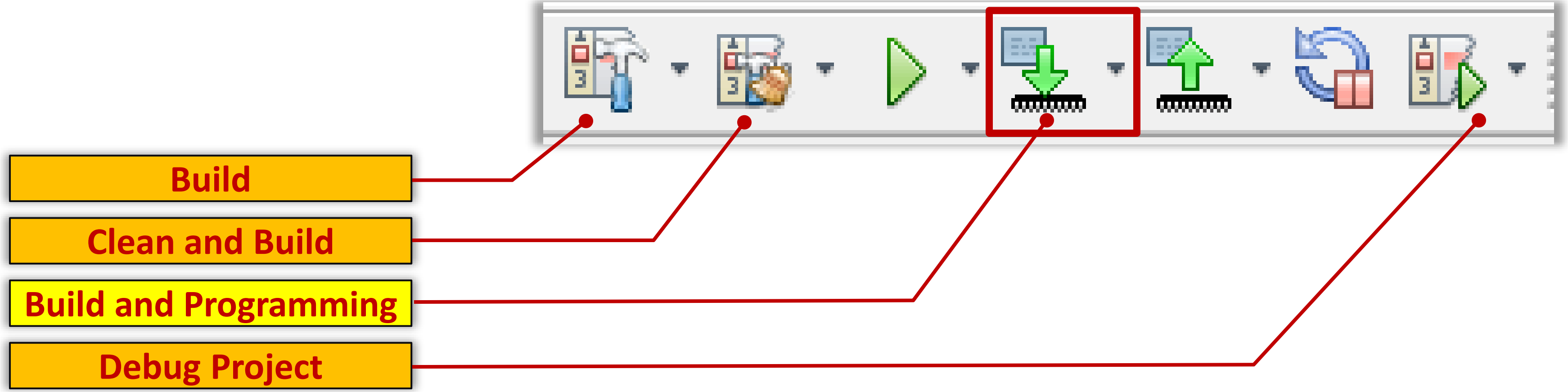
- Click the  icon to Clean and Build your project.
- Or press **Shift + F11** to Clean and Build your project.
- Compile result message will show in the output windows.



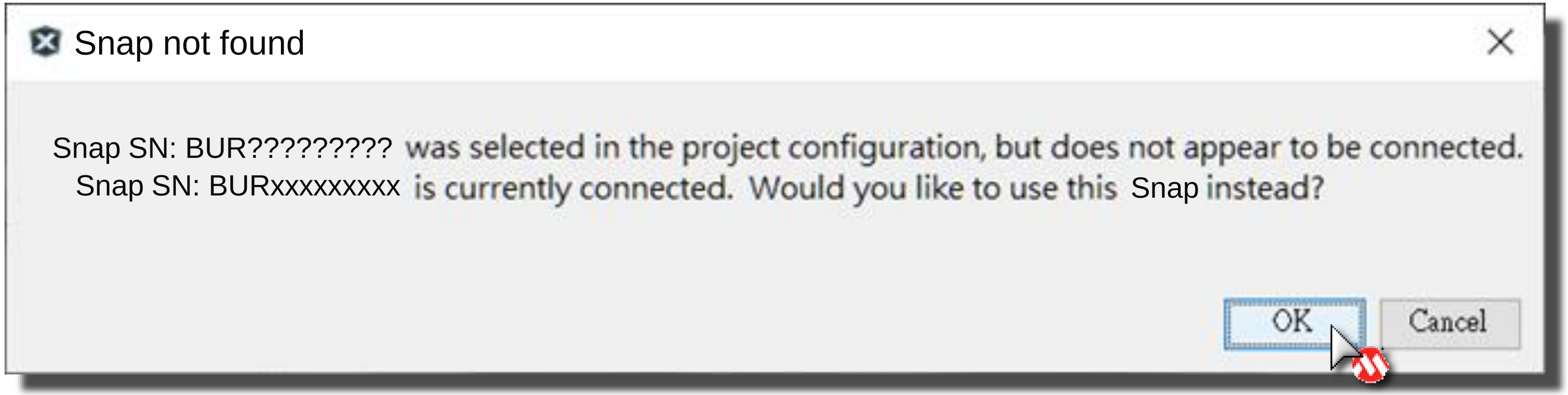
Lab0 First Project

Step 11

- Click Icon  to build and program firmware to EVB.

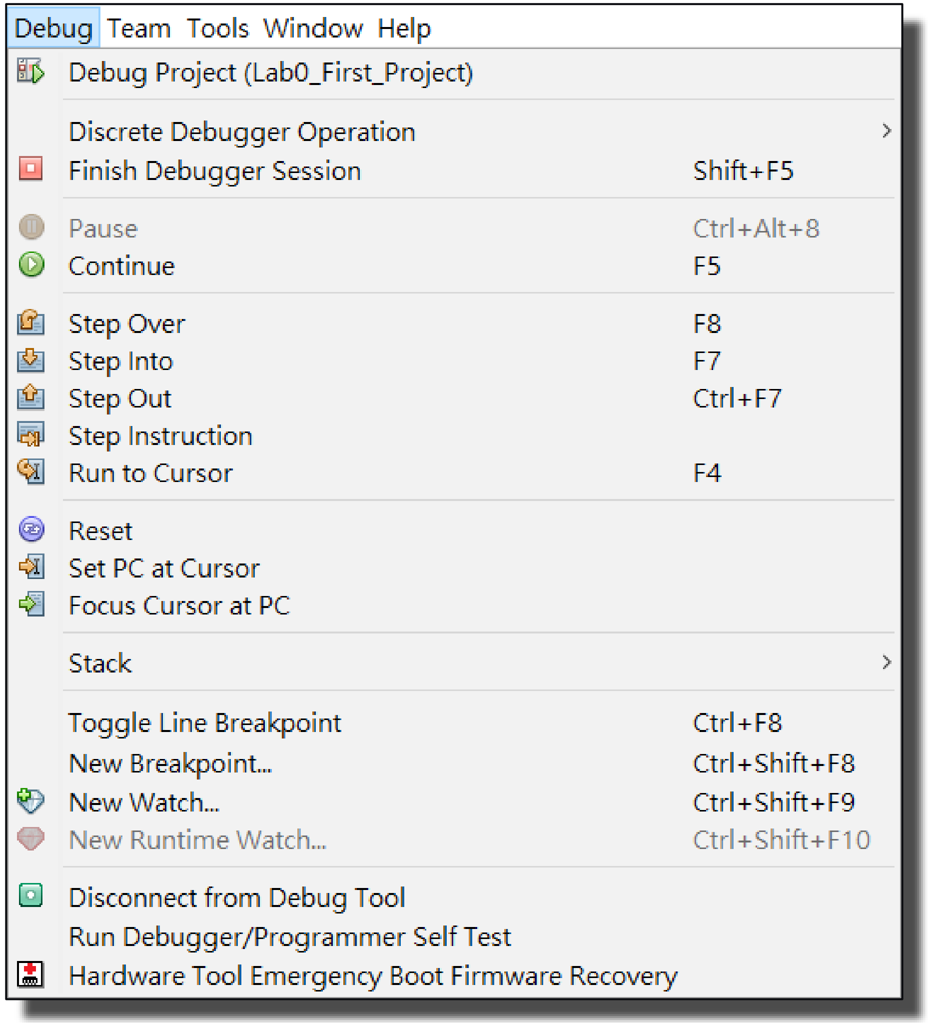


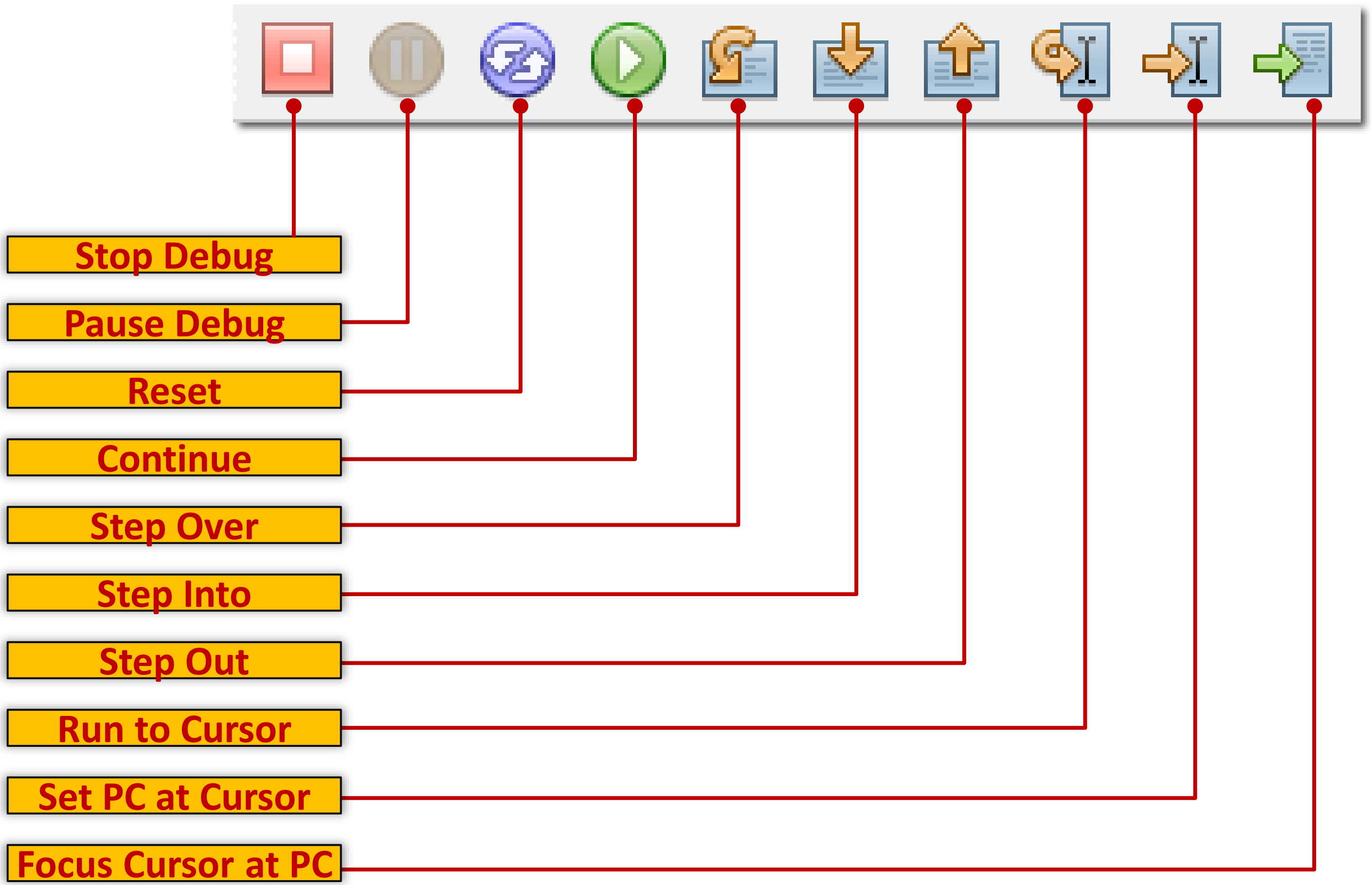
- The number of Programmer SN of your project while programming is different.



Debug Tools

- **MPLAB X IDE provide a lot of debug window, select the Debug.**



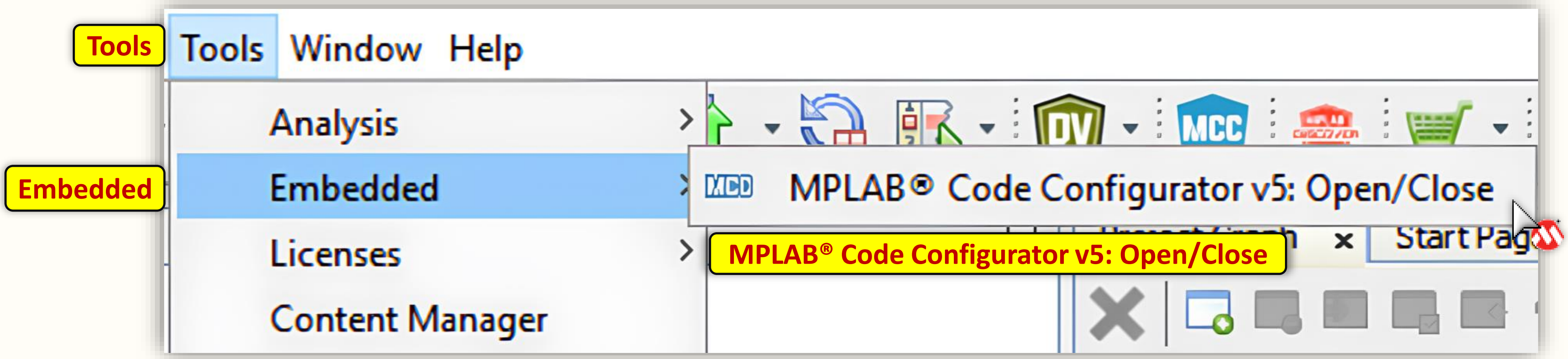


MPLAB® MCC Harmony Interface introduction

MCC Harmony Interface

Hint

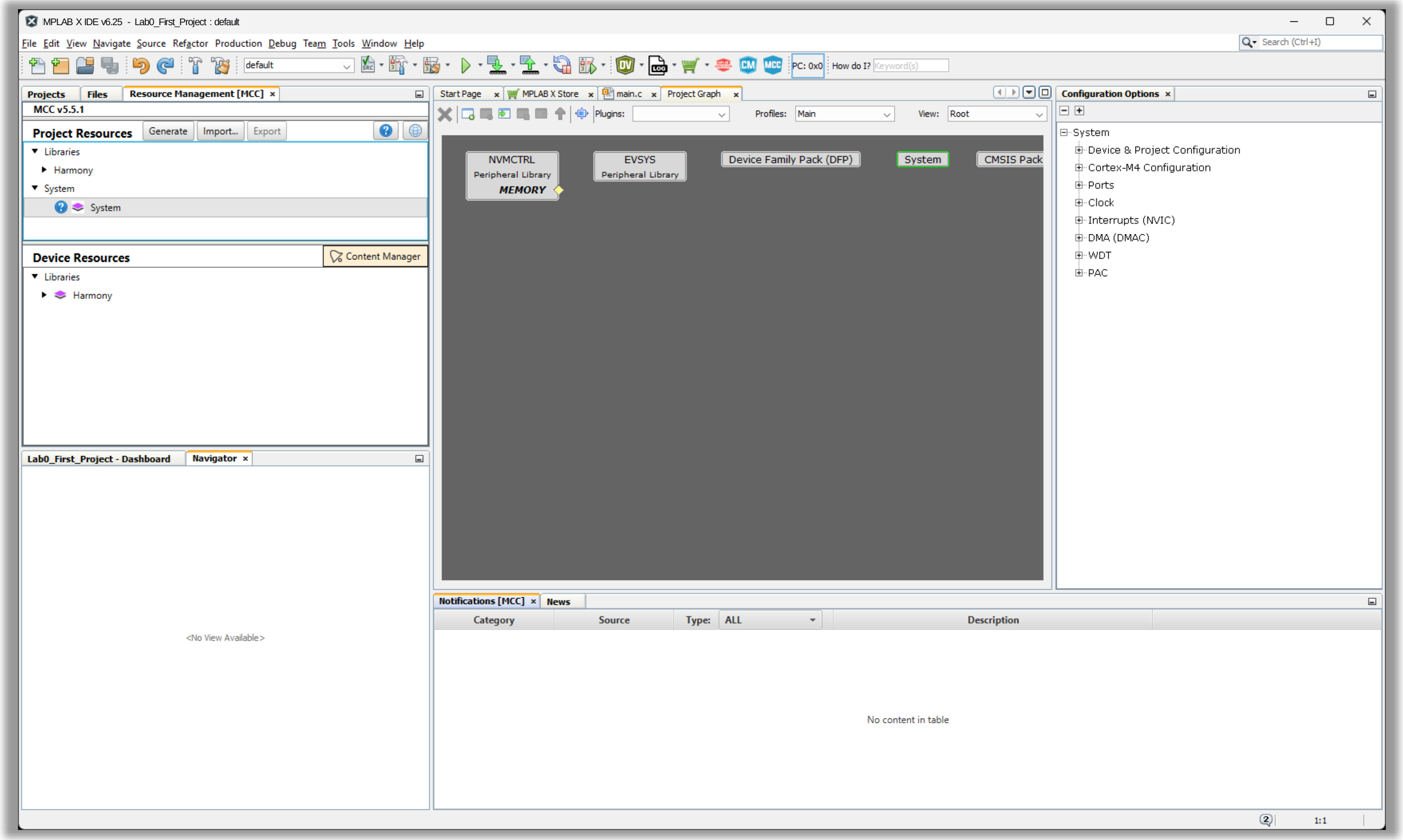
- **Notice ! If you close MCC Harmony already , please open it through below steps.**
- **Select to Tools > Embedded > MPLAB® Code Configurator v5: Open/Close**



- **Or Click  Icon.**

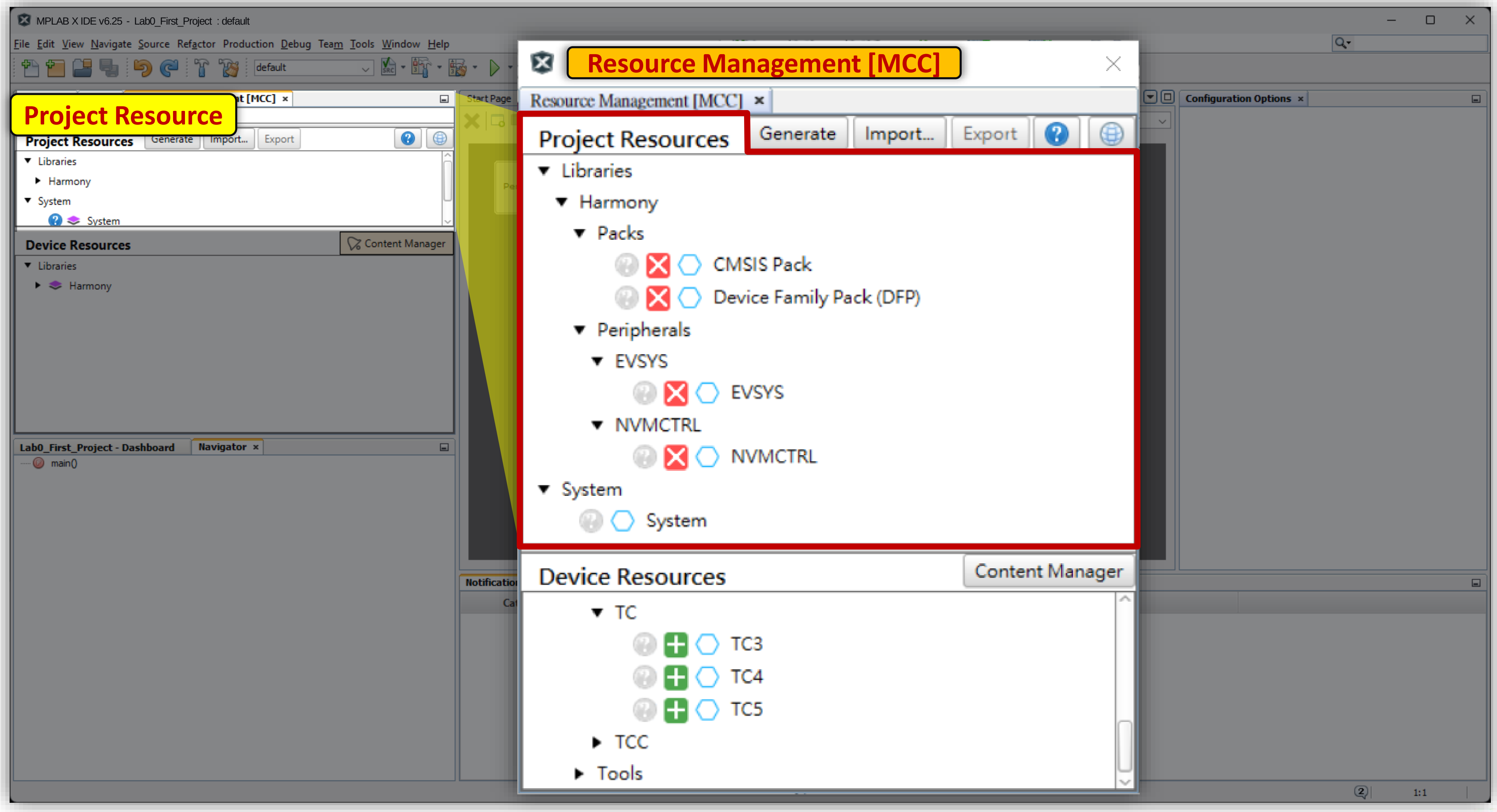
MCC Harmony Interface

- **Return to MCC Harmony, let's go through its interface.** (The version of modules might be different to yours.)



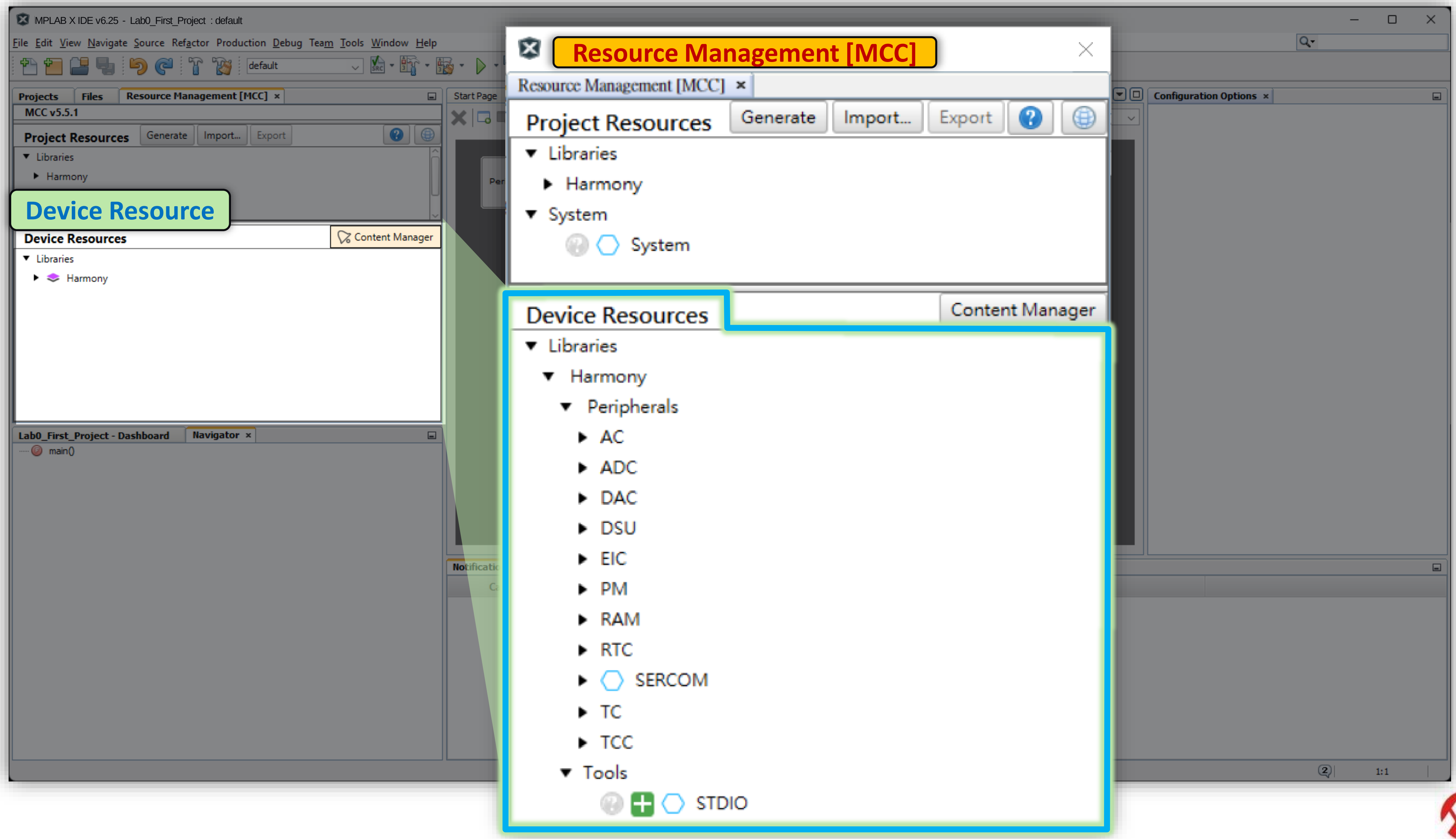
MCC Harmony Interface

- **Project Resource. (The modules already added in current project)**



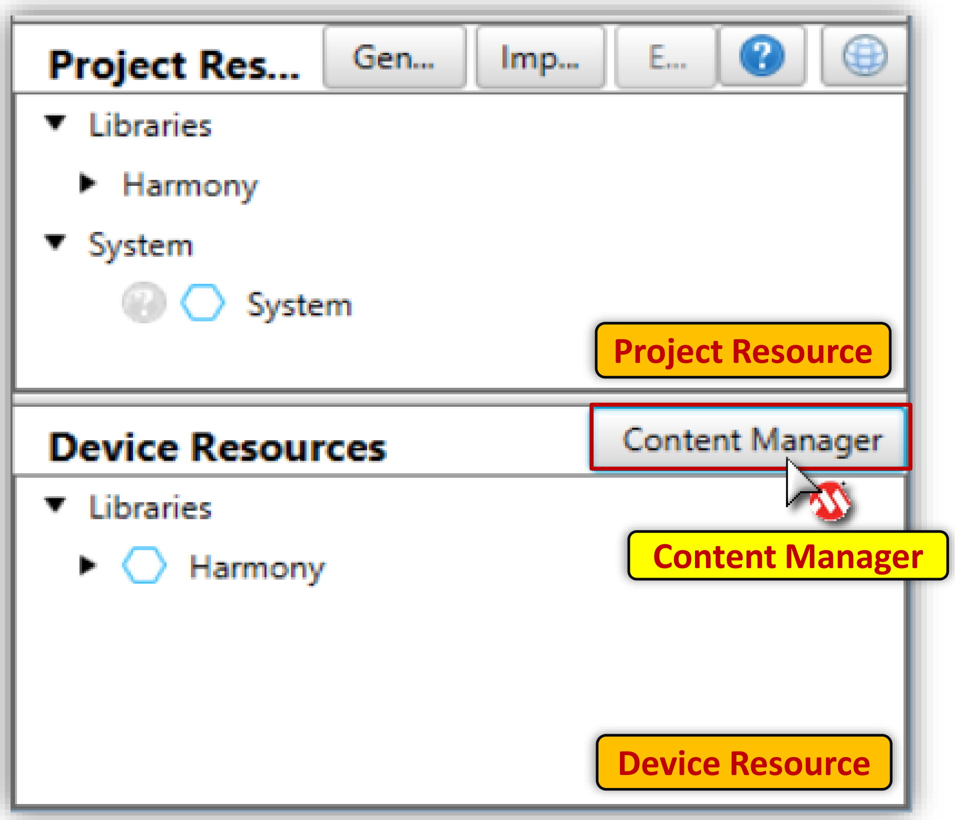
MCC Harmony Interface

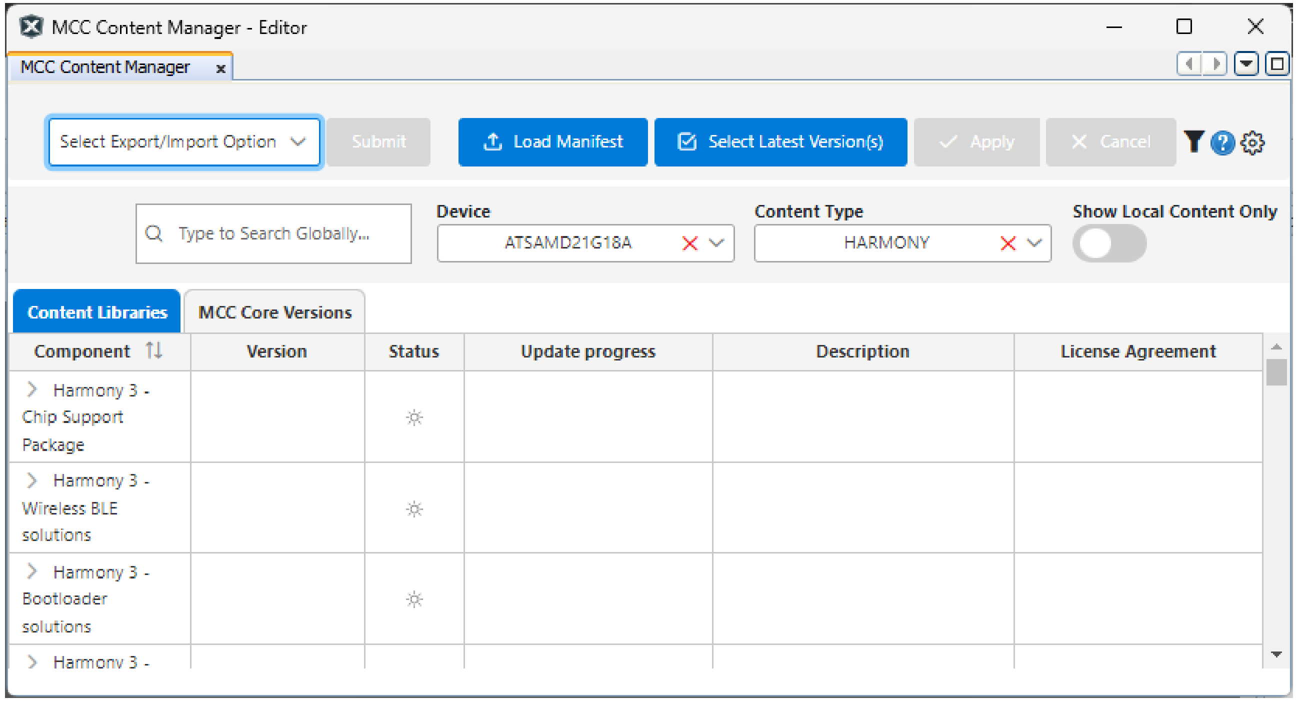
- Device Resource. (The modules not in current project but available to be added)



MCC Harmony Interface

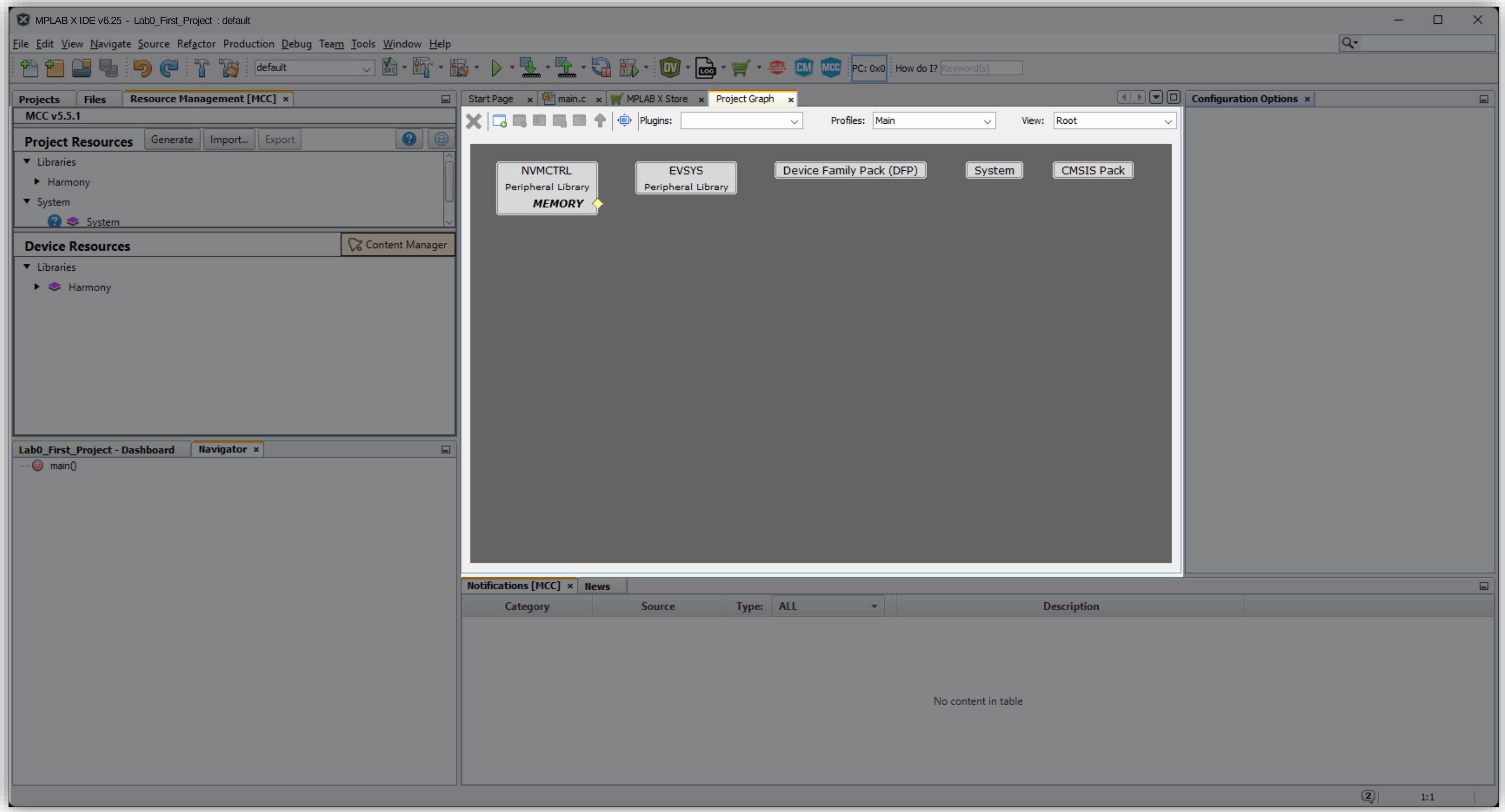
- **Content Manager. (Existing Framework)**





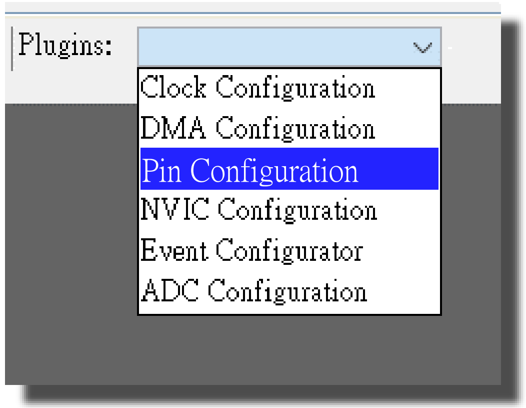
MCC Harmony Interface

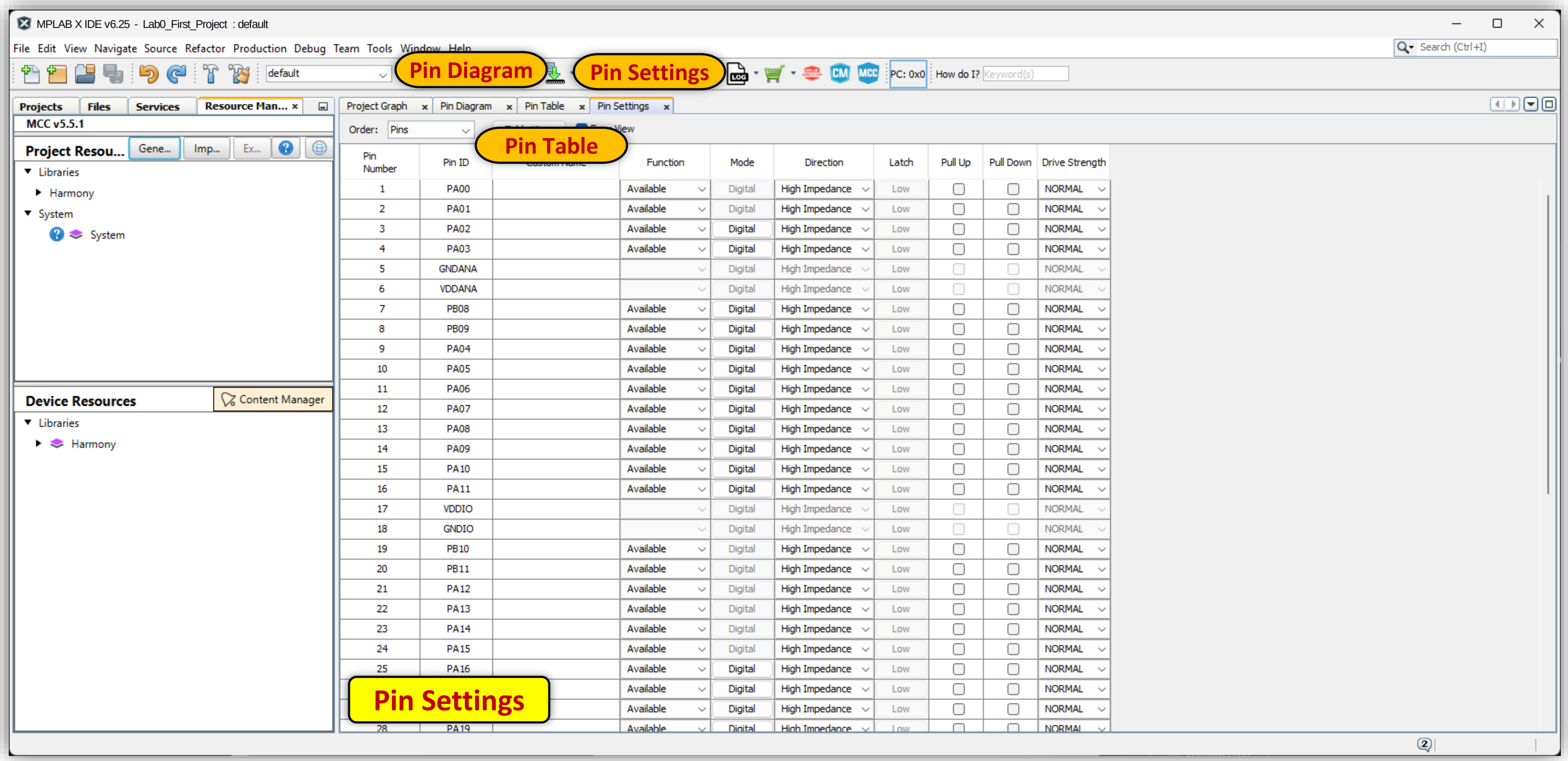
- **Project Graph. (The connection diagram between modules)**



MCC Harmony Interface

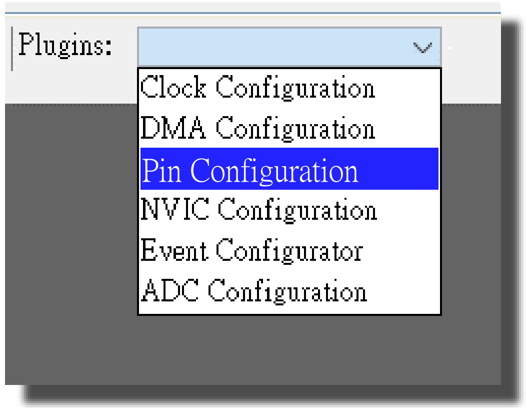
- Project Graph > Plugins : Pin Configuration

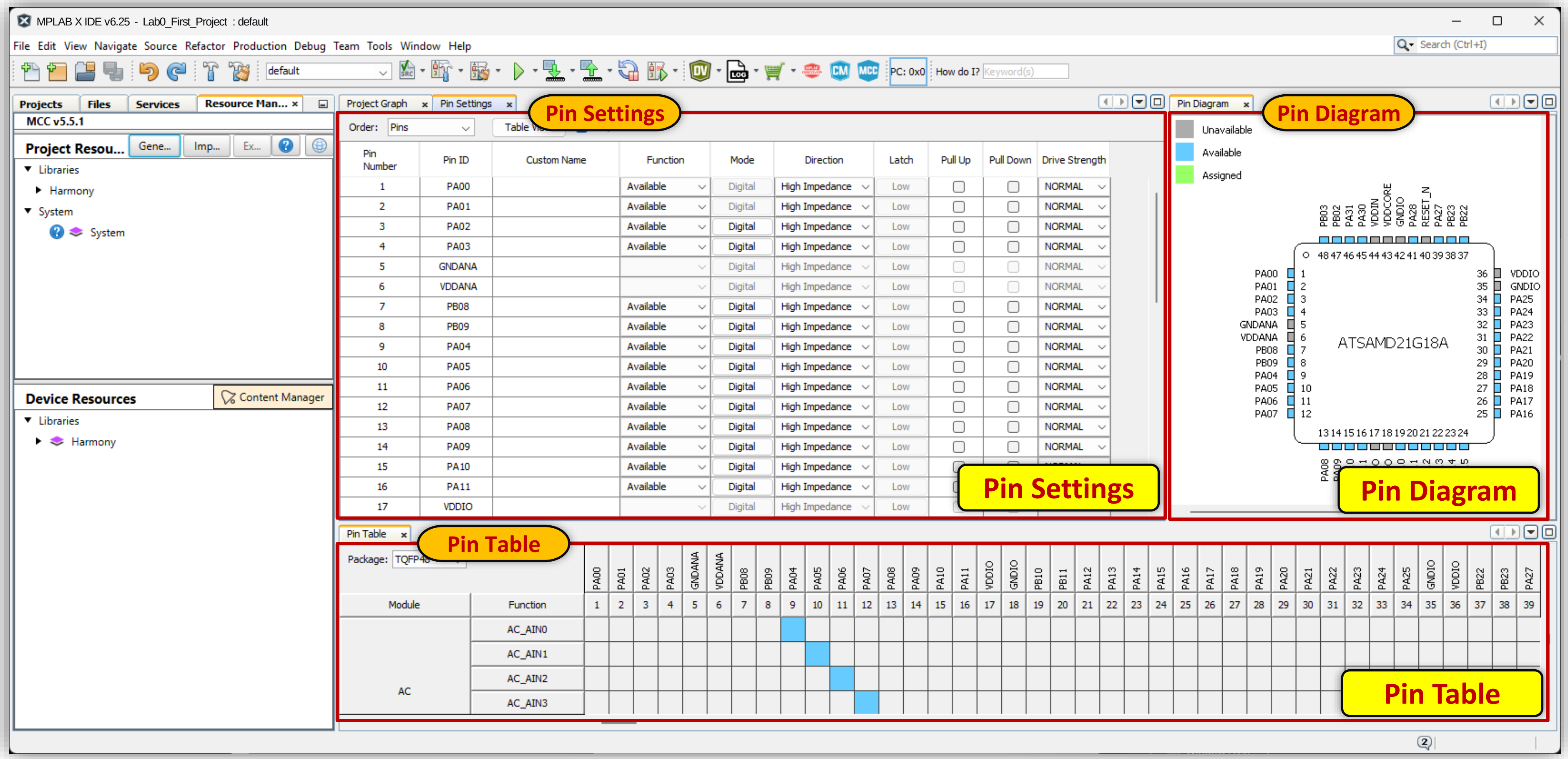




MCC Harmony Interface

- Project Graph > Plugins : Pin Configuration (Dock three windows to different locations)

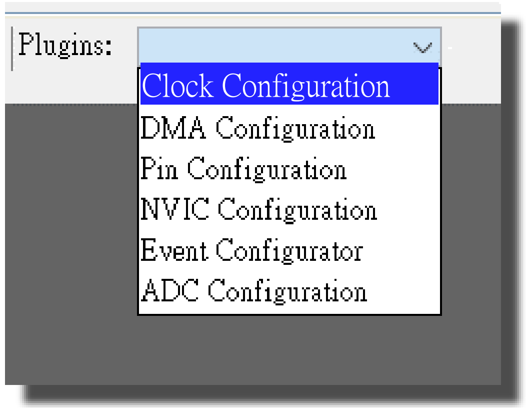


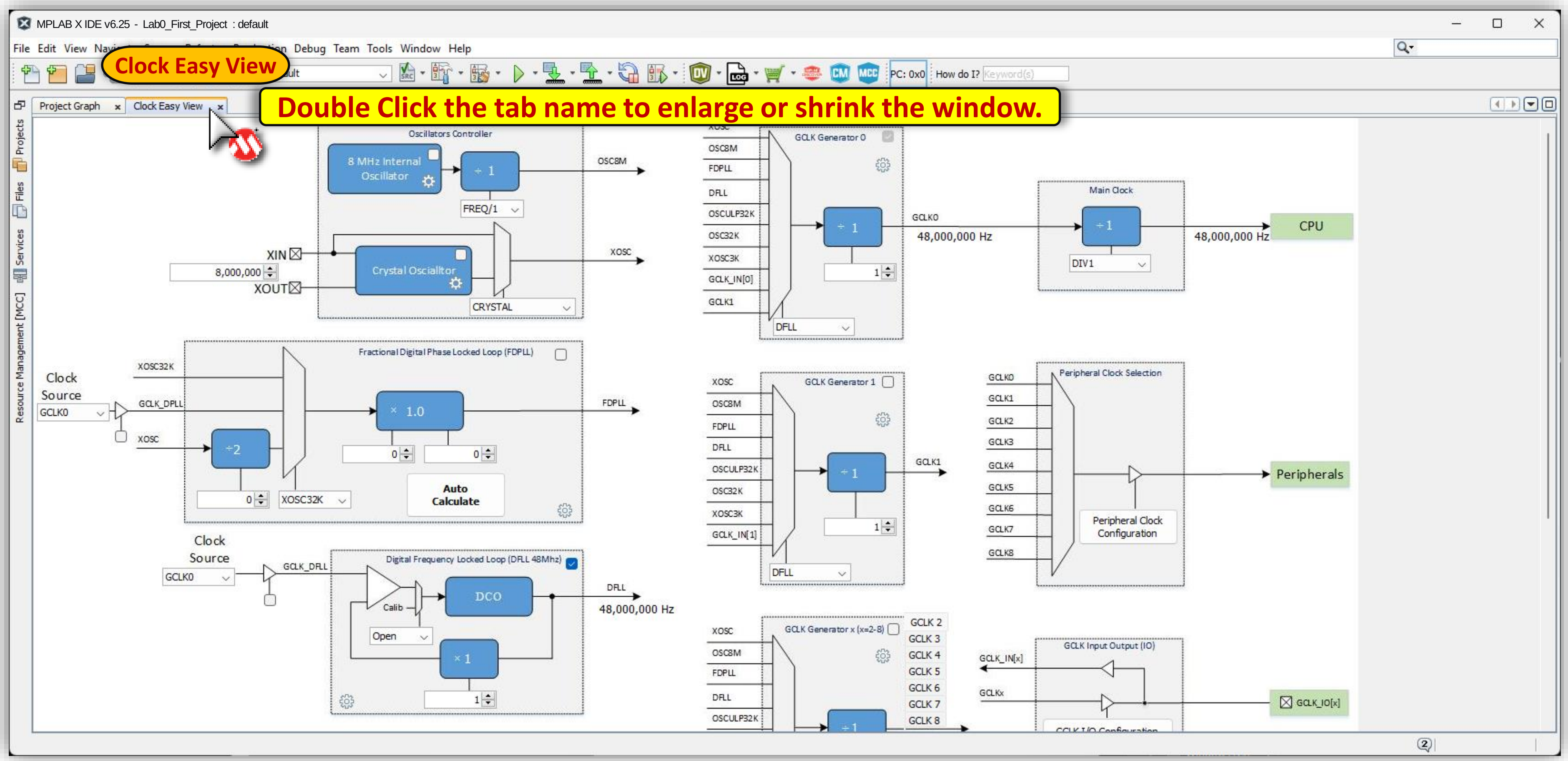


MICROCHIP

MCC Harmony Interface

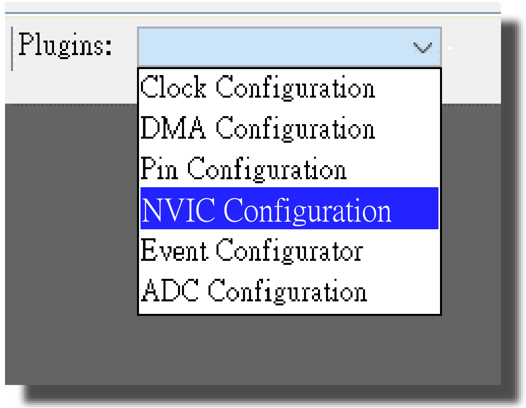
- **Project Graph > Plugins : Clock Configuration**

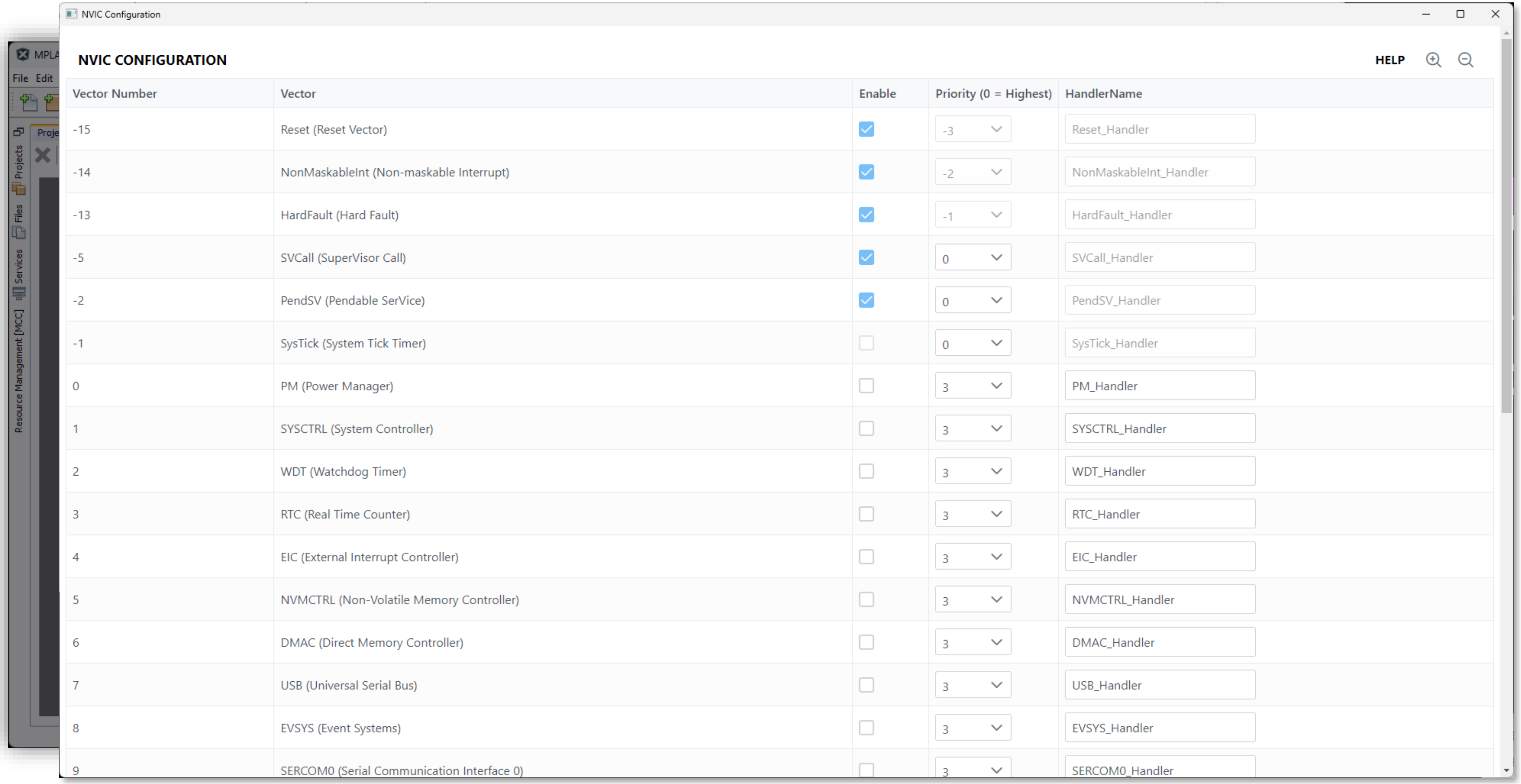




MCC Harmony Interface

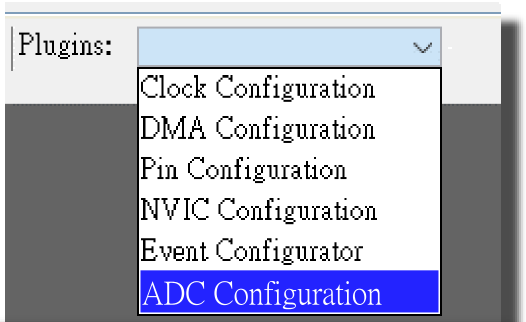
- **Project Graph > Plugins : NVIC Configuration (Open a new window after clicked)**



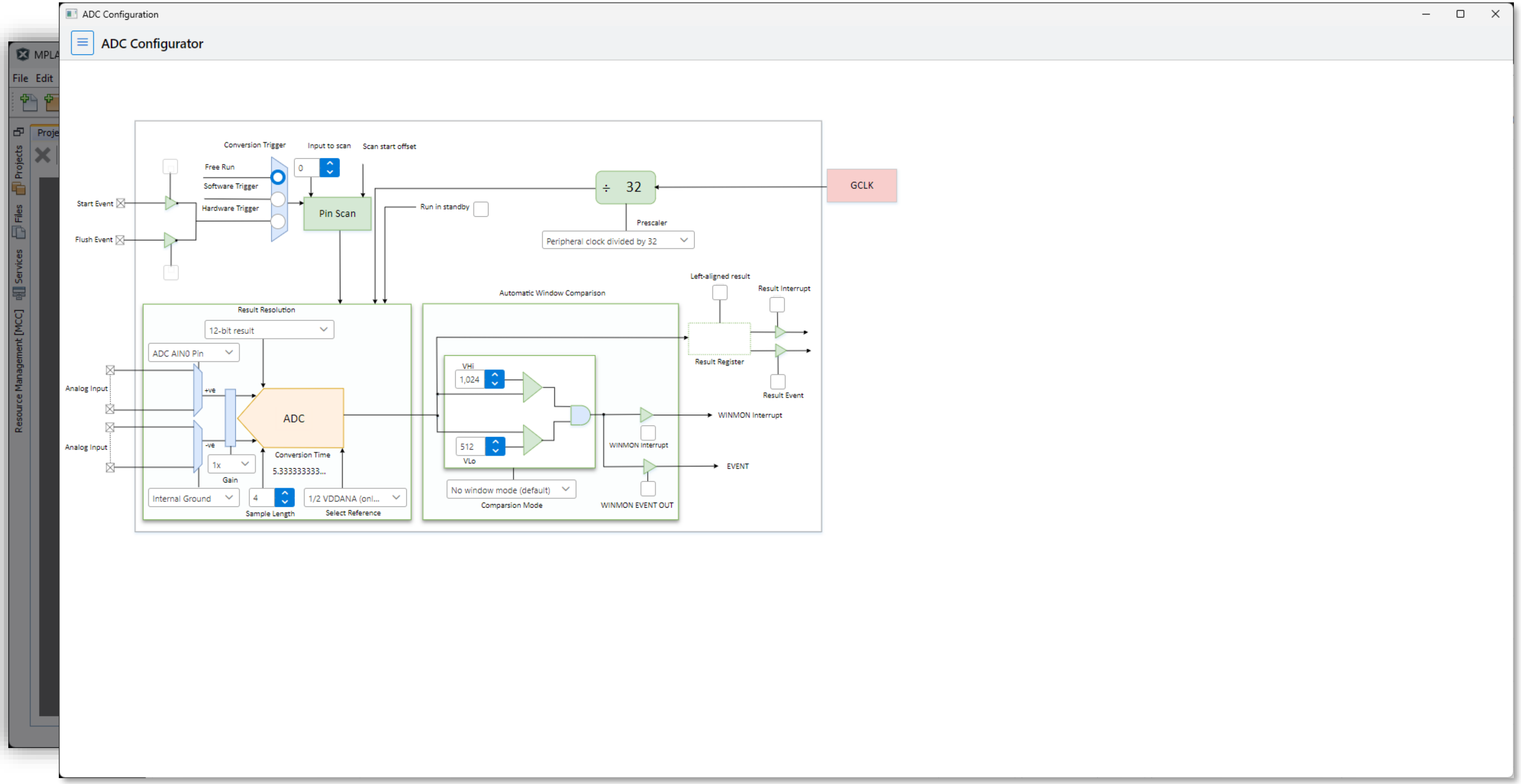


MCC Harmony Interface

- **Project Graph > Plugins : ADC Configuration**

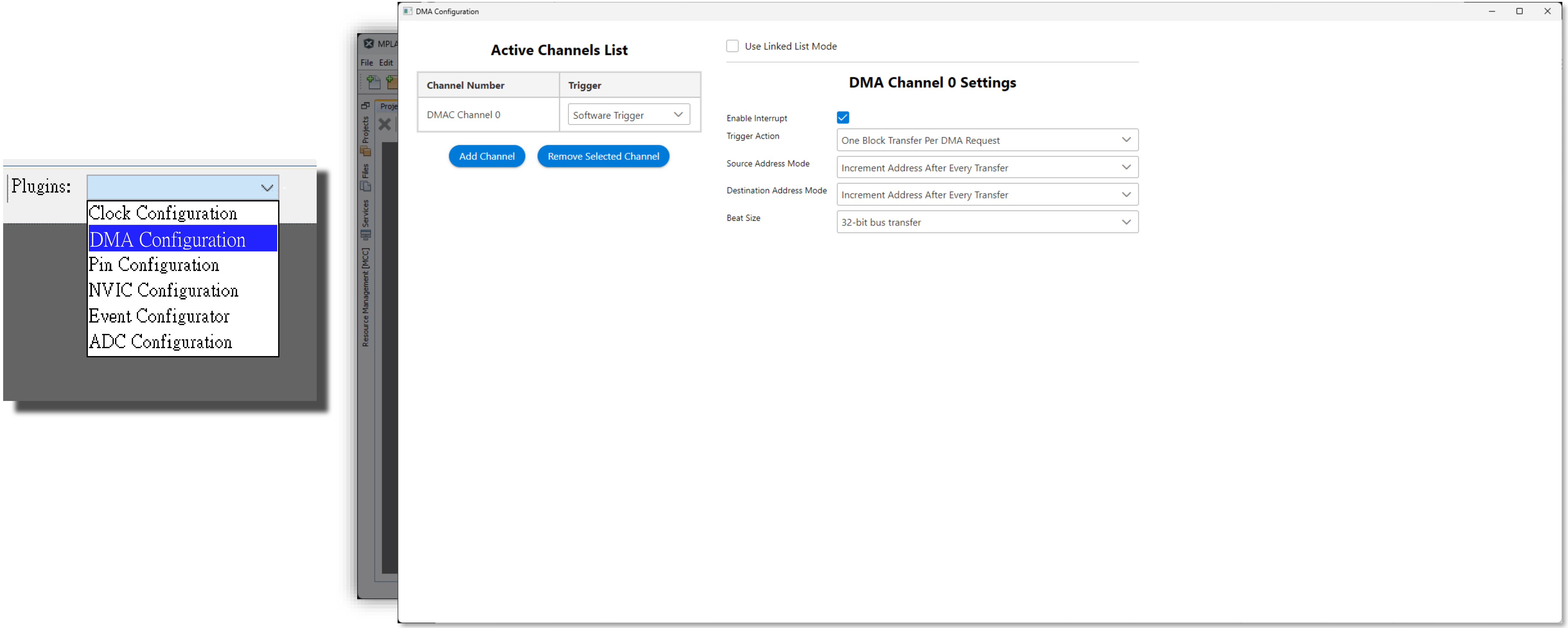


**ADC Configuration
will not show in a new project**



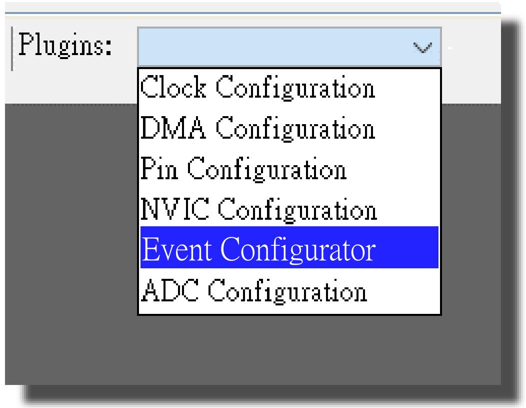
MCC Harmony Interface

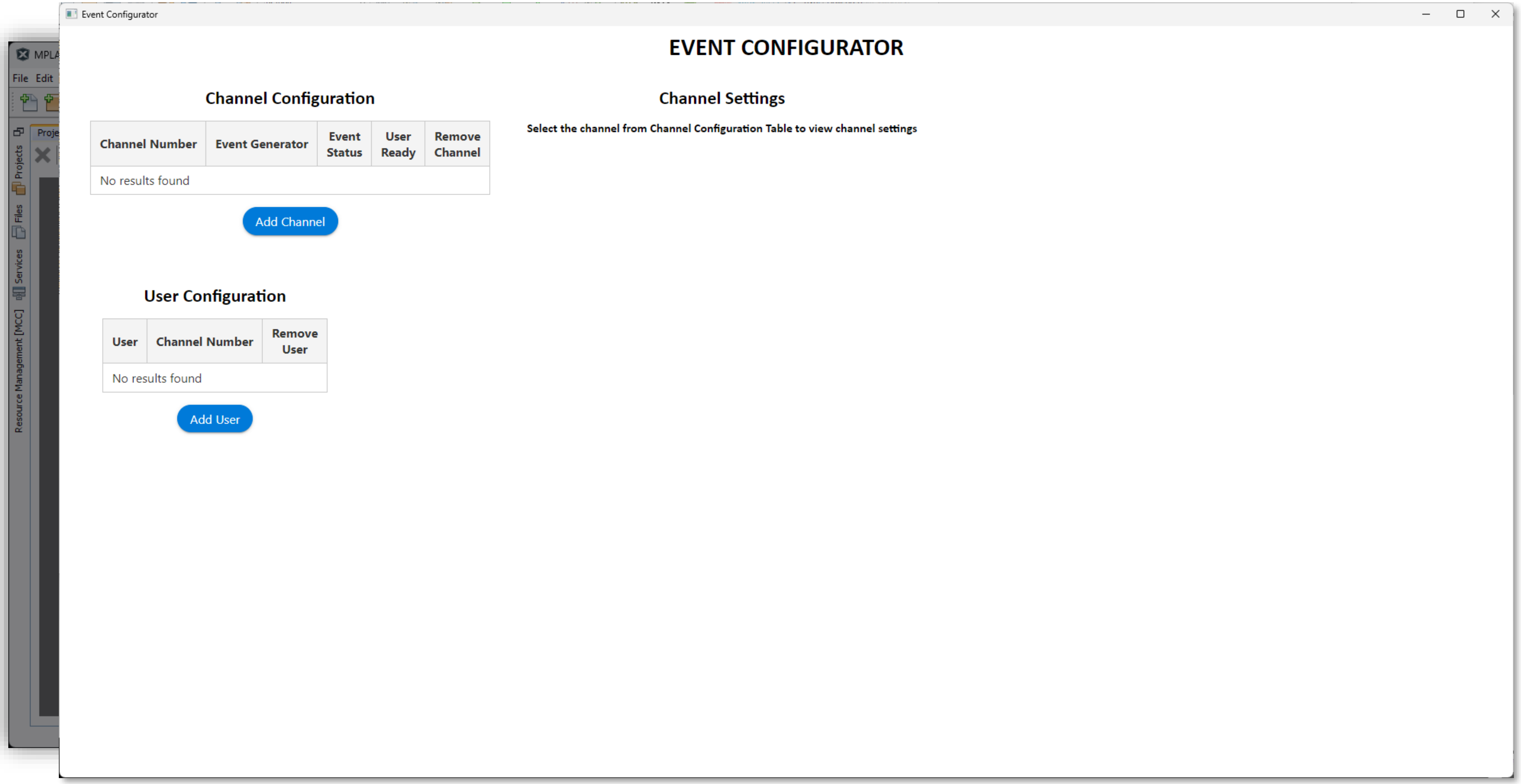
- **Project Graph > Plugins : DMA Configuration (Open a new window after clicked)**



MCC Harmony Interface

- Project Graph > Plugins : Event Configurator (Open a new window after clicked)

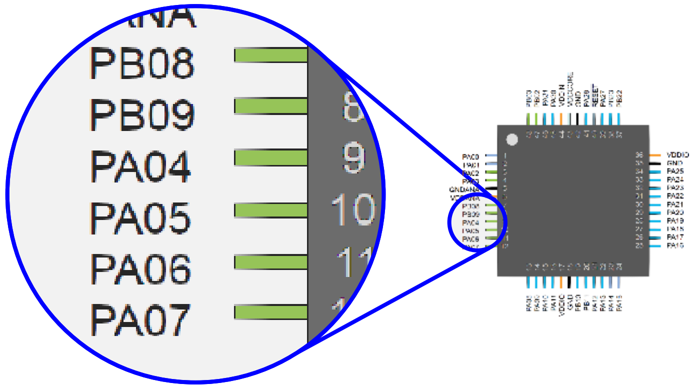


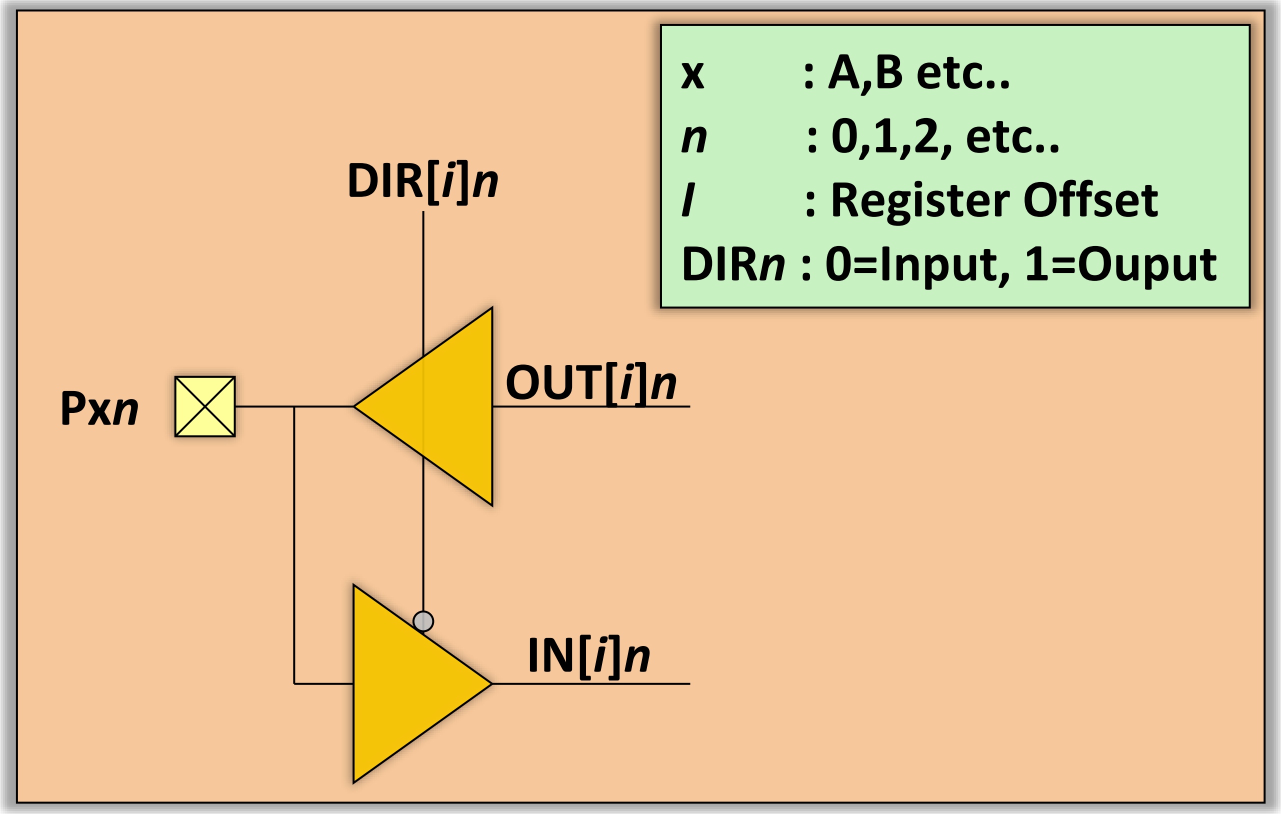


PORT I/O Architecture

PORT Block Diagram

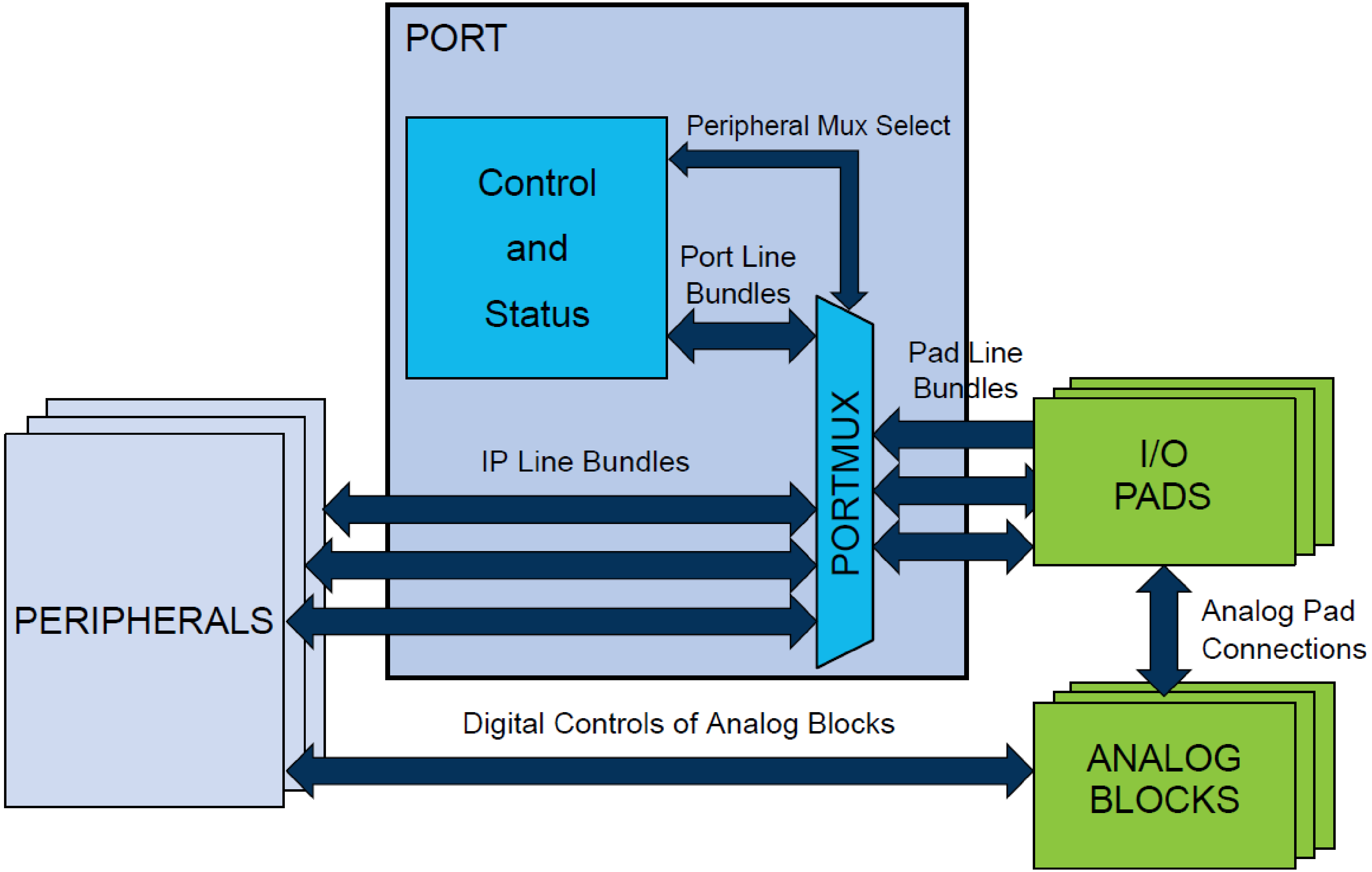
- Please refer to the block diagram, this is the concept for programming model. The real and detail block diagram please refer to the following slide.
- DIR: use to determine in or out. 0 for input, 1 for output.
- OUT: set output drive value for Output pins.
- IN: reaction logical level from Input pins.



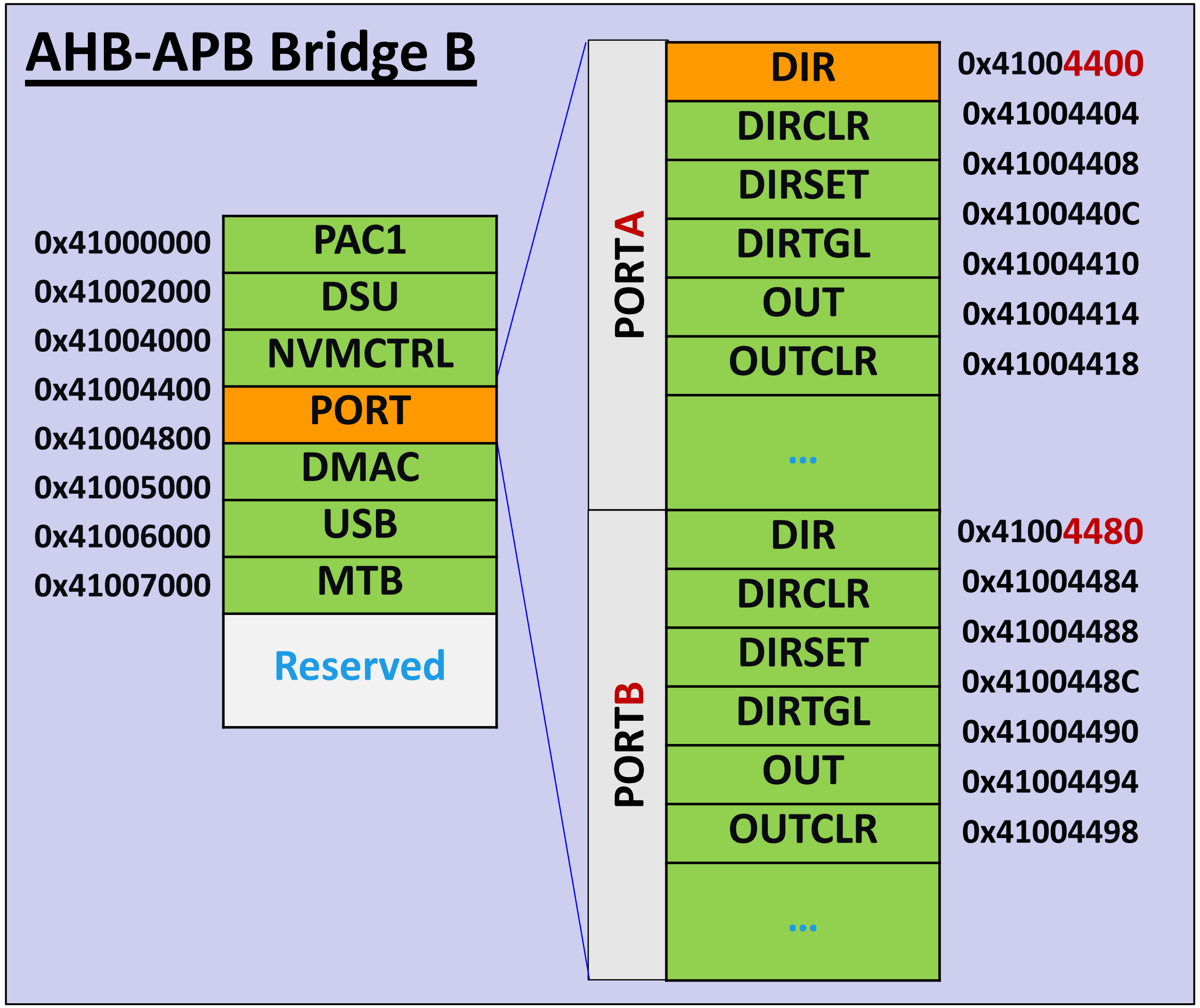


PORT Block Diagram

- **SAMD21G18A PORT Block Diagram**



Register Access



Register Summary from datasheet

Offset	Name	Bit Pos.
0x00	DIR	7:0
0x01		15:8
0x02		23:16
0x03		31:24
0x04	DIRCLR	7:0
0x05		15:8
0x06		23:16
0x07		31:24
0x08	DIRSET	7:0
0x09		15:8
0x0A		23:16
0x0B		31:24
0x0C	DIRTGL	7:0
0x0D		15:8
0x0E		23:16
0x0F		31:24
0x10	OUT	7:0
0x11		15:8
0x12		23:16
0x13		31:24
0x14	OUTCLR	7:0
0x15		15:8
0x16		23:16
0x17		31:24

Register Access

- There are two way to access Register:
  - Access by Register Pointer

```
*(__IO uint32_t *) (0x41004400U) = 0x55; /* 0x41004400U : PORTA DIR */
*(__IO uint32_t *) (0x41004480U) = 0x55; /* 0x41004480U : PORTB DIR */
```

  - Access by Pre-defined Struct

```
PORT_REGS->GROUP[0].PORT_DIR = 0x55; /* Group[0] : PORTA Register Set */
PORT_REGS->GROUP[1].PORT_DIR = 0x55; /* Group[1] : PORTB Register Set */
```

atsamd21g18a.h

```
#define PORT_REGS  ((port_registers_t*)0x41004400)
```

port.h

```
typedef struct
{  /* Port Module */
      port_group_registers_t   GROUP[GROUP_NUMBER];
} port_registers_t;
```

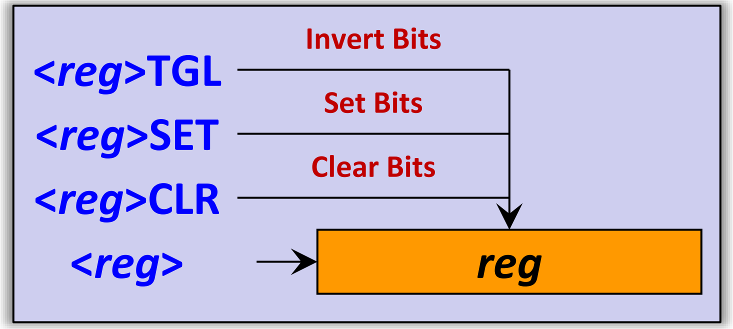
port.h

```
typedef struct
{
  __IO  uint32_t    PORT_DIR;
  __IO  uint32_t    PORT_DIRCLR;
  __IO  uint32_t    PORT_DIRSET;
  __IO  uint32_t    PORT_DIRTGL;
  …
} port_group_registers_t;
```

Atomic Bits Manipulation

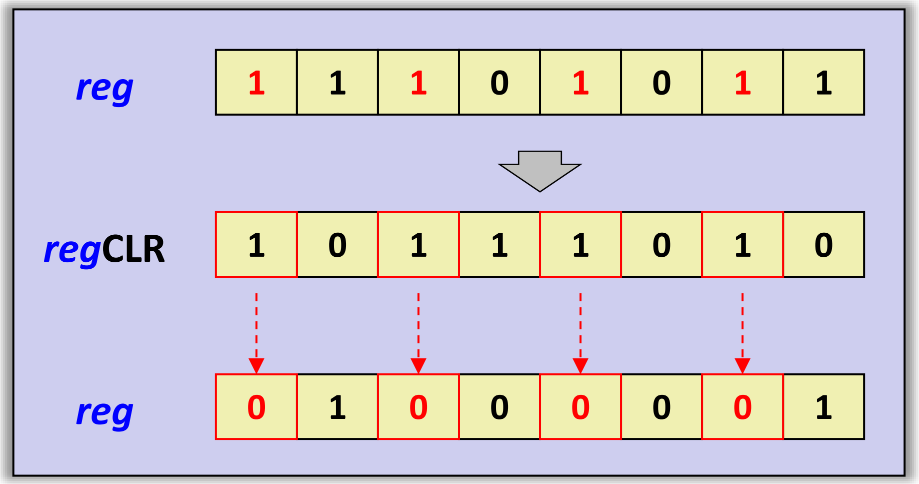
- Some register provide *<reg>*CLR, *<reg>*SET , *<reg>*TGL register for atomic bits manipulation.

- This register manipulated without doing a read-modify-write operation by using below
  - *<reg>* Toggle (*<reg>*TGL)
  - *<reg>* Clear (*<reg>*CLR)
  - *<reg>* Set (*<reg>*SET)

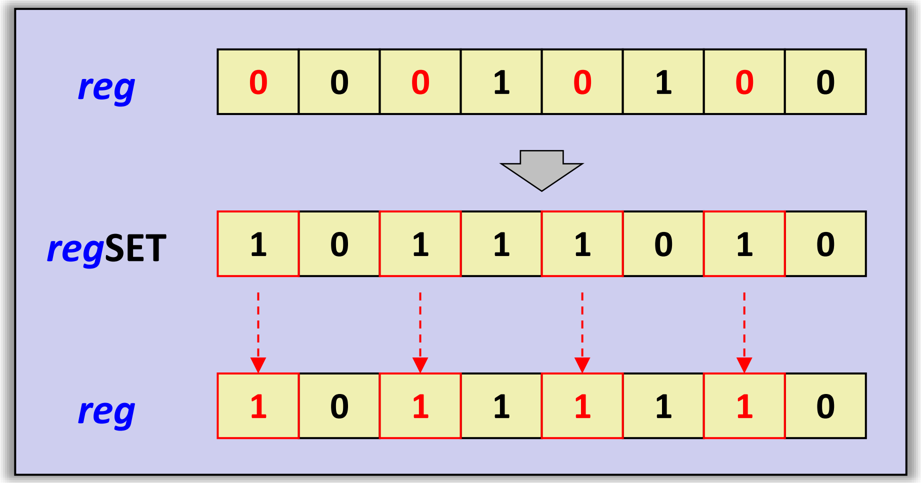


- Setting the manipulation register bits corresponding to the base register will set the bits to 1, clear to 0 or toggle.

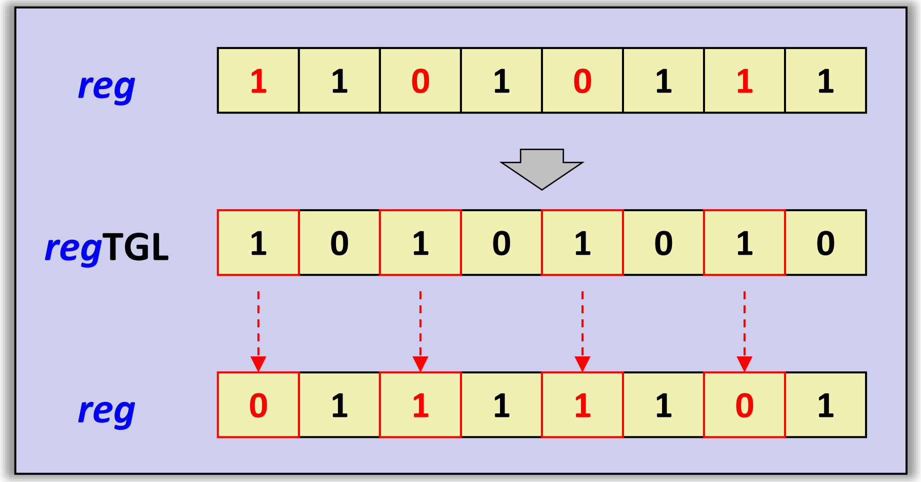
Atomic Bits Manipulation



*reg*CLR : corresponding bits clear to 0



*reg*SET : corresponding bits set to 1



*reg*TGL : corresponding bits toggled
(0 -> 1,   1 -> 0)

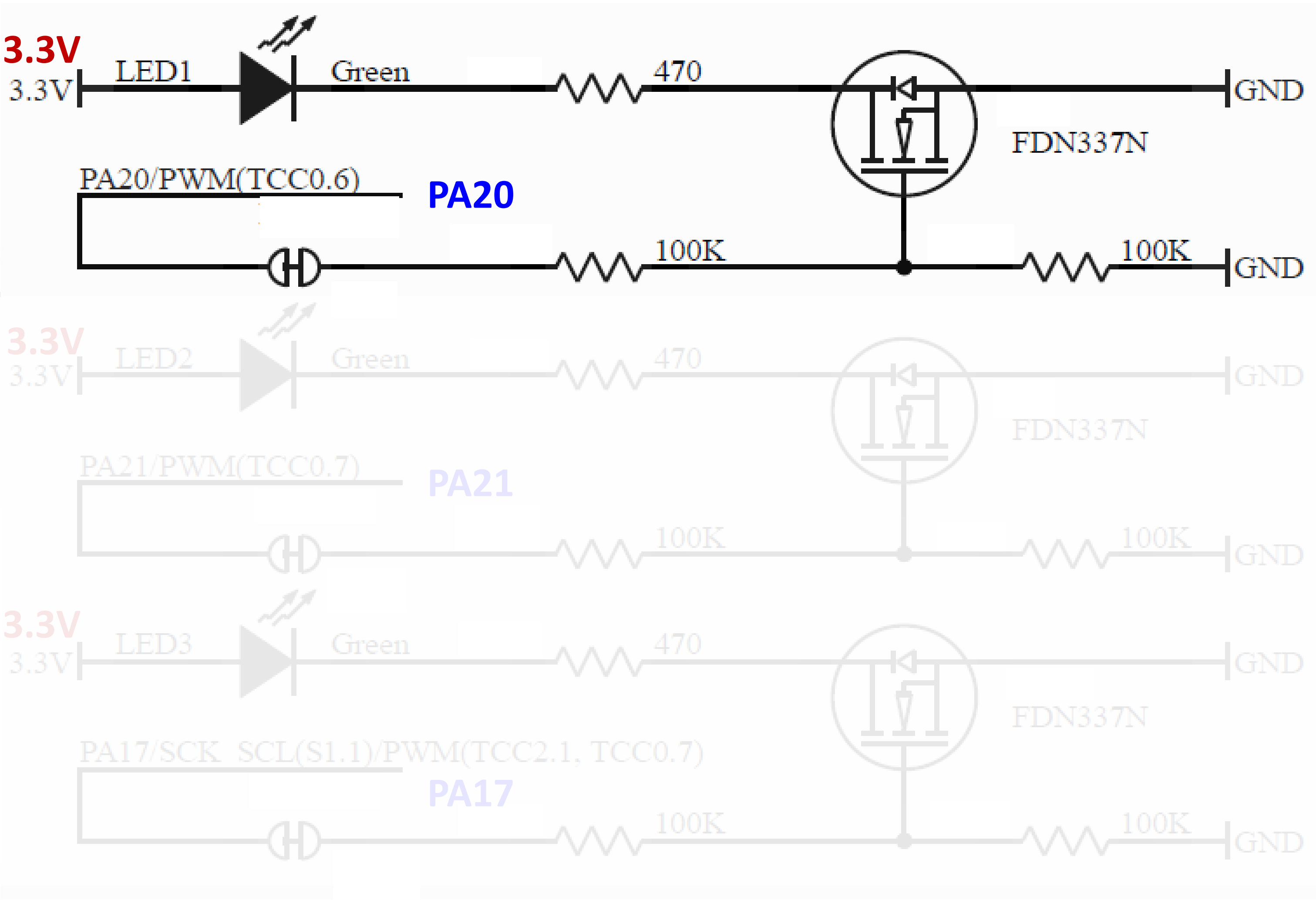
Lab1 PORT Output

- Try to control LED1(PA20) continues toggle.
  (period around 500ms ~ 1S).
- You can use register pointer or pre-defined struct to access PORT.

- **How to start？**

Check Schematic

LED1	PA20
LED2	PA21
LED3	PA17
BT1	PA14
BT2	PA15



MICROCHIP

Lab1 PORT Output

Step 1

a Add code segment to your main loop

```
// TODO 1.01
uint32_t i = 0;

int main (void)
{
   SYS_Initialize ( NULL );

   // TODO 1.02
   *(__IO uint32_t *)(0x41004400U + 0x08U) = 0x00100000;

   while( true )
   {
      // TODO 1.03
      for ( i = 0 ; i < 2000000 ; i++ ){}
      *(__IO uint32_t *)(0x41004400U + 0x1CU) = 0x00100000;
   }
}
```

Access by Register Pointer

Period Control

b Program firmware to target board then observe result.

Lab1 PORT Output

Register Description

0x00100000 (Hex)
0000 0000 0001 0000 0000 0000 0000 0000 b (Binary)

3322 2222 2222 1111 1111 11
1098 7654 3210 9876 5432 1098 7654 3210

```
// TODO 1.01
uint32_t i = 0;

int main (void)
{
   SYS_Initialize ( NULL );

   // TODO 1.02
    *(__IO uint32_t *)(0x41004400U + 0x08U) = 0x00100000;

   while( true )
   {
      // TODO 1.03
      for ( i = 0 ; i < 2000000 ; i++ ) {}
       *(__IO uint32_t *)(0x41004400U + 0x1CU) = 0x00100000;
   }
}
```

LED1 Pin PA20

Offset	Name
0x00	DIR
0x01	
0x02	
0x03	
0x04	DIRCLR
0x05	
0x06	
0x07	
0x08	DIRSET
0x09	
0x0A	
0x0B	
0x1C	OUTTGL
0x1D	
0x1E	
0x1F	

Lab1 PORT Output - Access by Pre-defined Struct

Step 2

a Modify code segment to your main loop

Access by Pre-defined Struct

```
// TODO 1.01
uint32_t i = 0;

int main (void)
{
   SYS_Initialize ( NULL );

   // TODO 1.02
    PORT_REGS->GROUP[0].PORT_DIRSET = 1<<20;

   while( true )
   {
      // TODO 1.03
      for ( i = 0 ; i < 2000000 ; i++ ) {}
      PORT_REGS->GROUP[0].PORT_OUTTGL = 1<<20;
   }
}
```

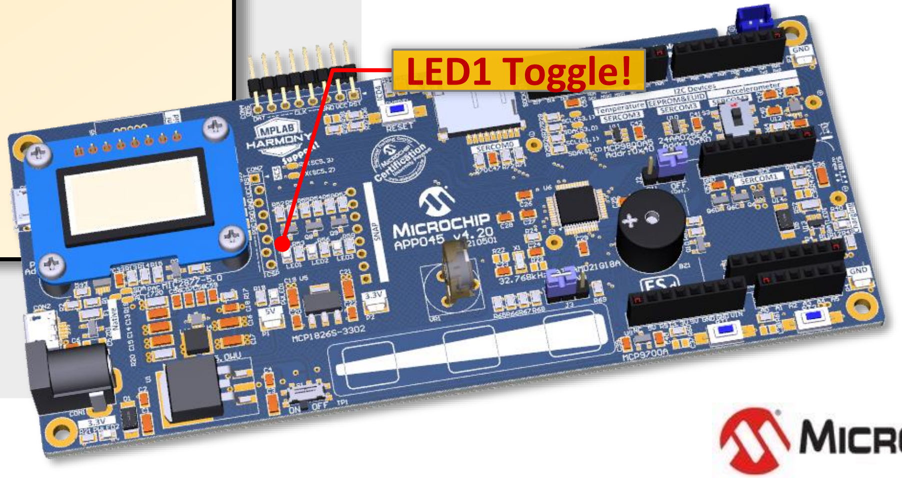
Period Control

Access by Register Pointer

```
// TODO 1.01
uint32_t i = 0;

int main (void)
{
  SYS_Initialize ( NULL );

  // TODO 1.02
  *(__IO uint32_t *)(0x41004400U + 0x08U) = 0x00100000 ;

  while( true )
  {
    // TODO 1.03
    for ( i = 0 ; i < 2000000 ; i++  ){}
    *(__IO uint32_t *)(0x41004400U + 0x1CU) = 0x00100000;
  }
}
```

Period Control

b Program firmware to target board then observe result.

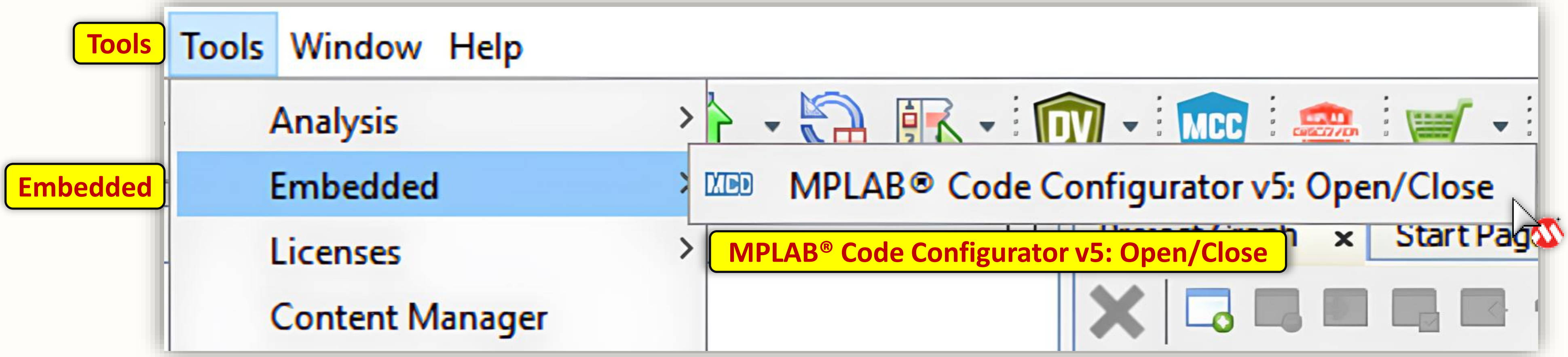


MICROCHIP

Pin Configuration using MCC

Hint

- **Notice ! If you already close MCC, please open it through below steps.**
- **Select to Tools > Embedded > MPLAB® Code Configurator v5: Open/Close**

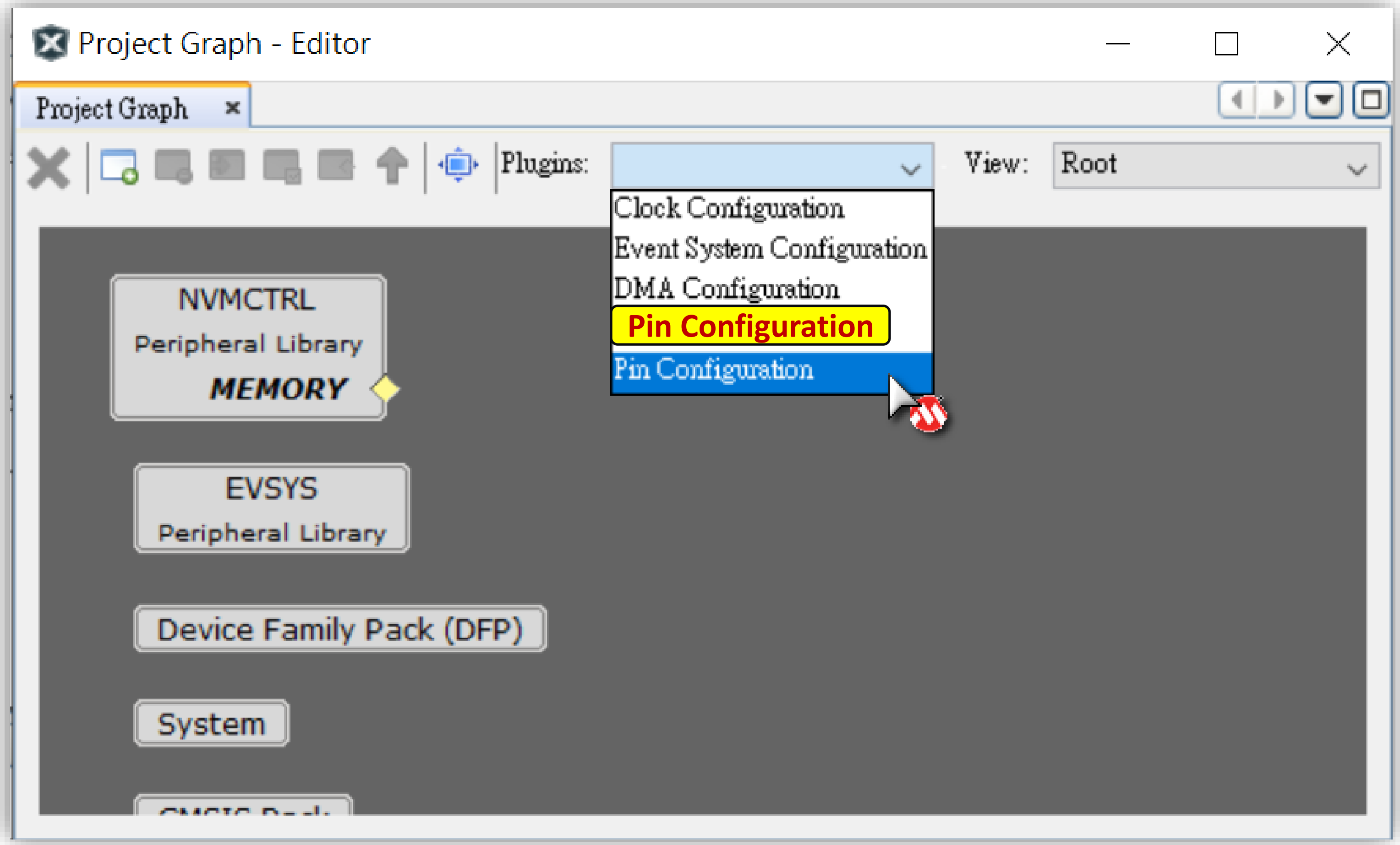


- **Or Click MCC Icon.**

Lab1 PORT Output - Pin Configuration using MCC

Step 3

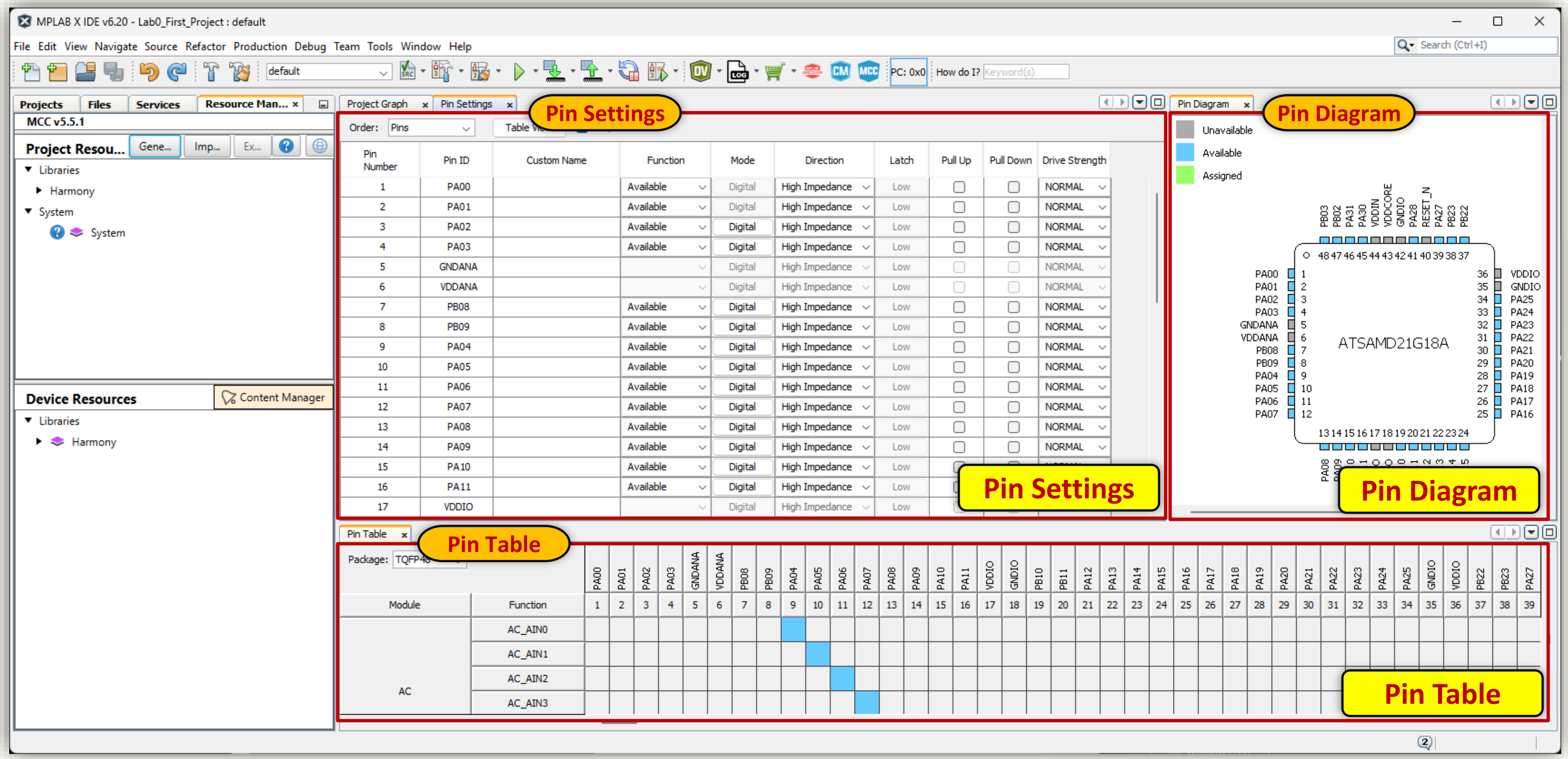
- **Return to MCC Harmony Configurator.**
- **Select Project Graph > Plugins > Pin Configuration**



Pin Configuration

Hint

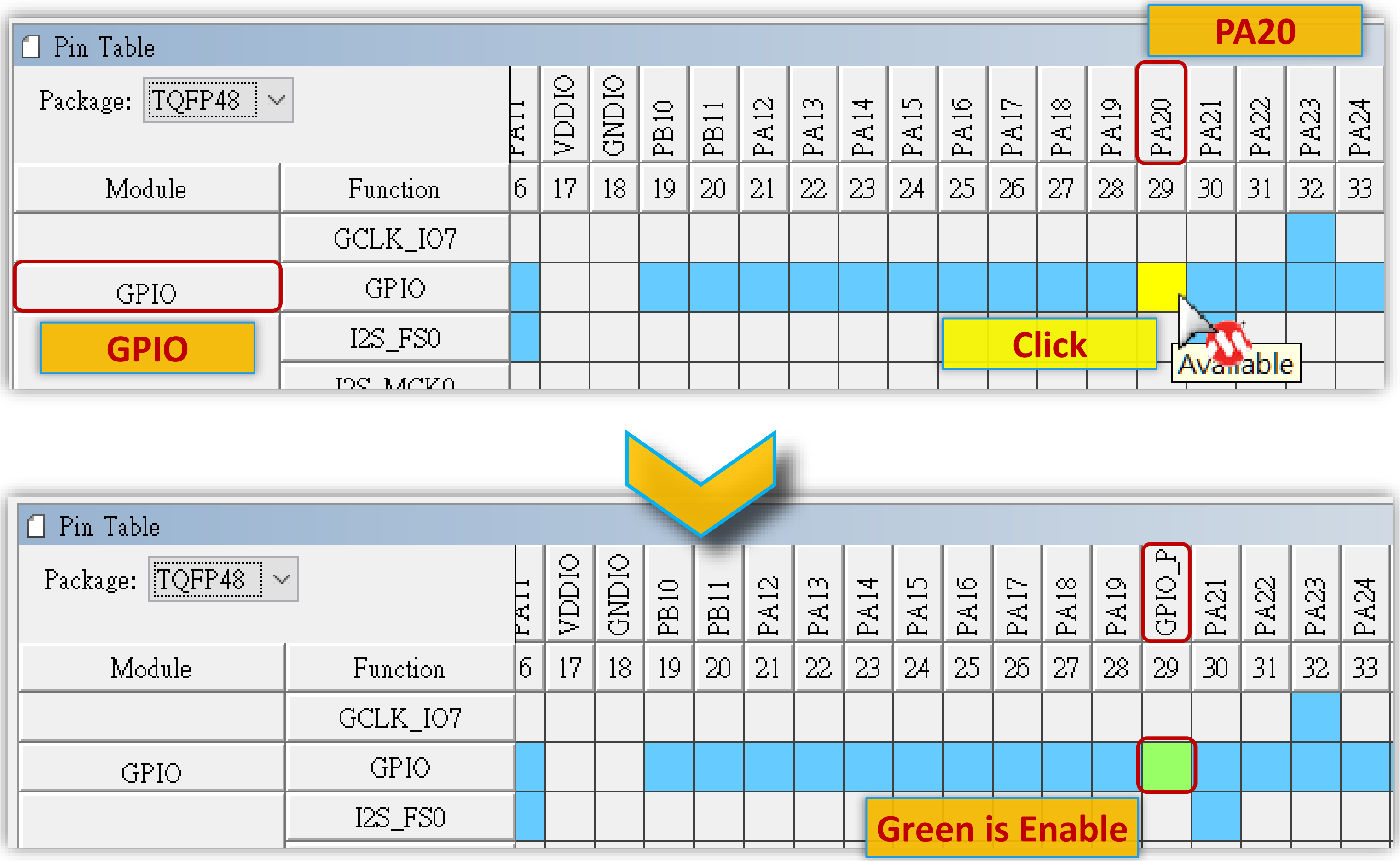
- **Pin Configuration will open three pages, you can refer to the following to dock.**



Lab1 PORT Output - Pin Configuration using MCC

Step 4

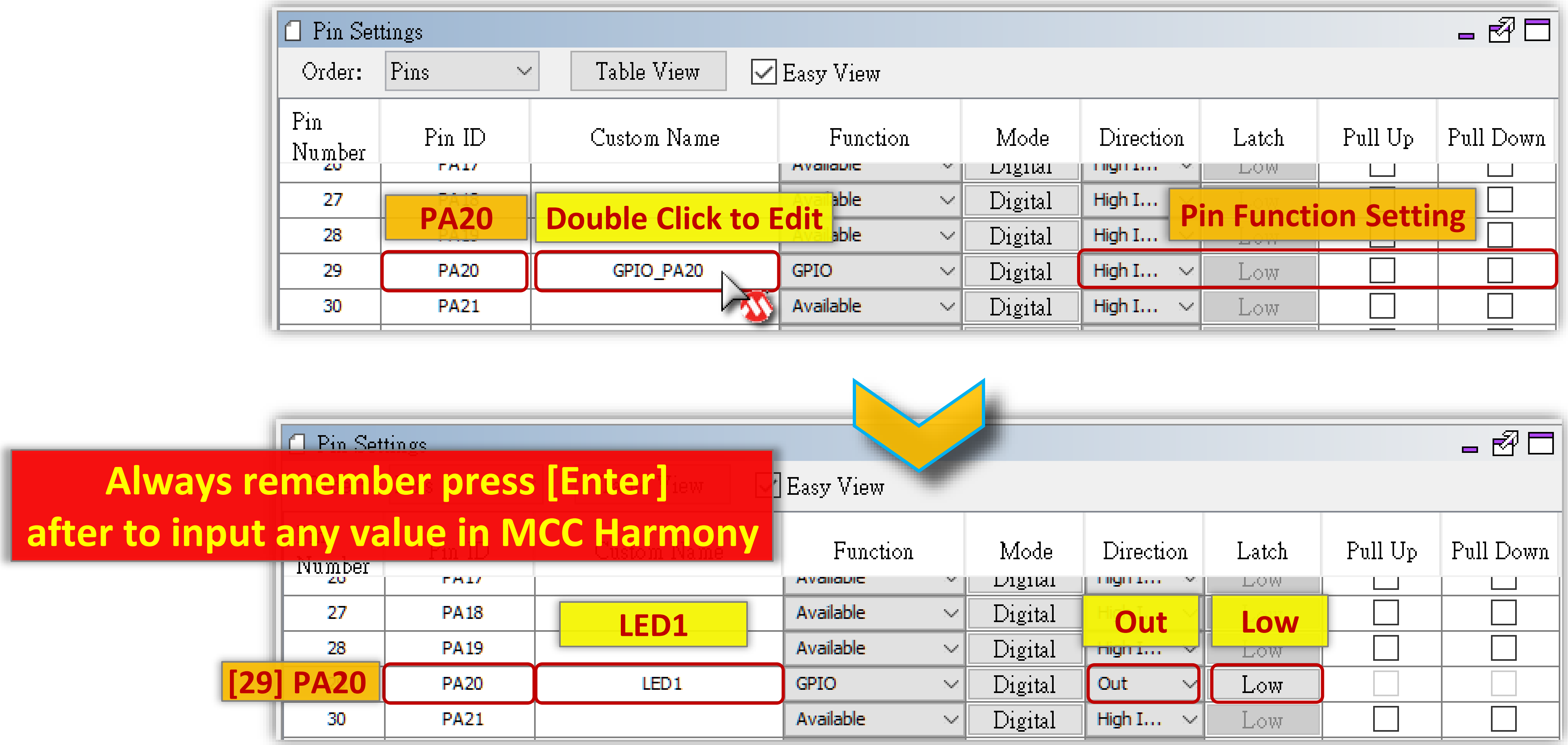
- **Pin Table configure. (Enable or Disable Pin)**



Lab1 PORT Output - Pin Configuration using MCC

Step 5

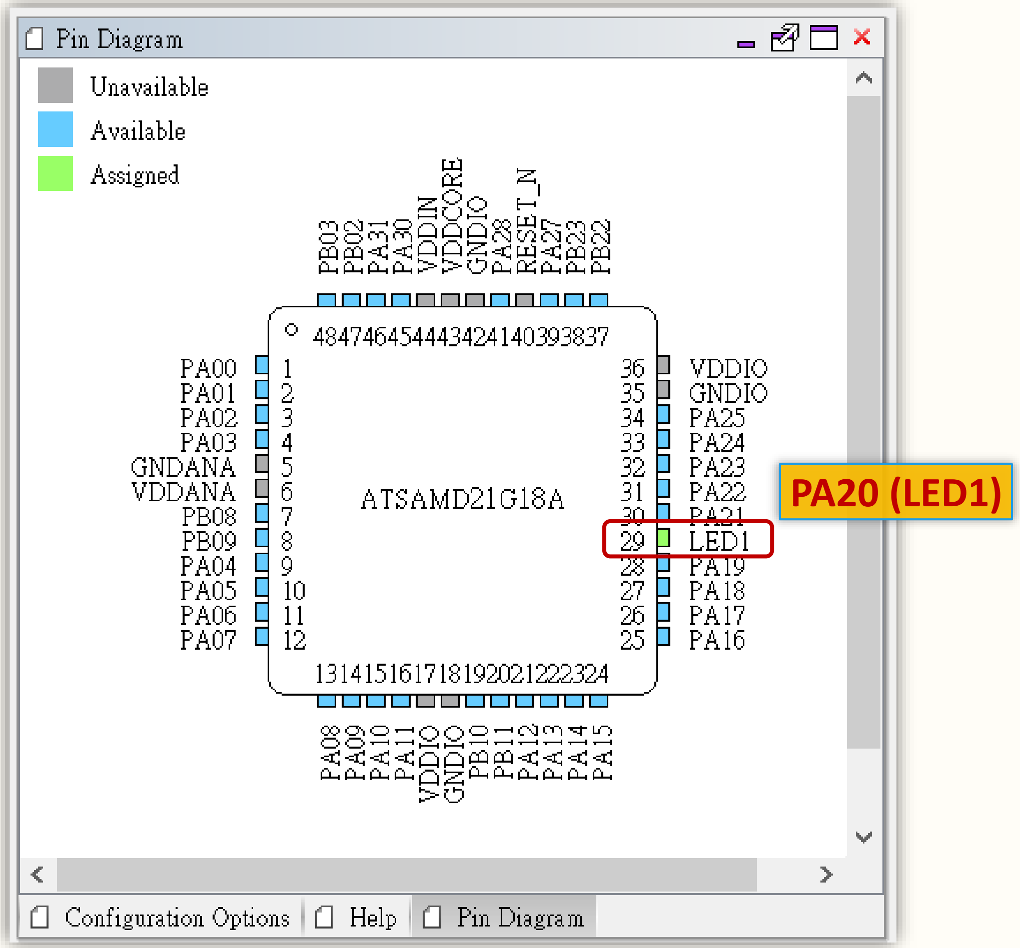
- **Pin Setting**. (Pin Function Setting)



Lab1 PORT Output - Pin Configuration using MCC

Hint

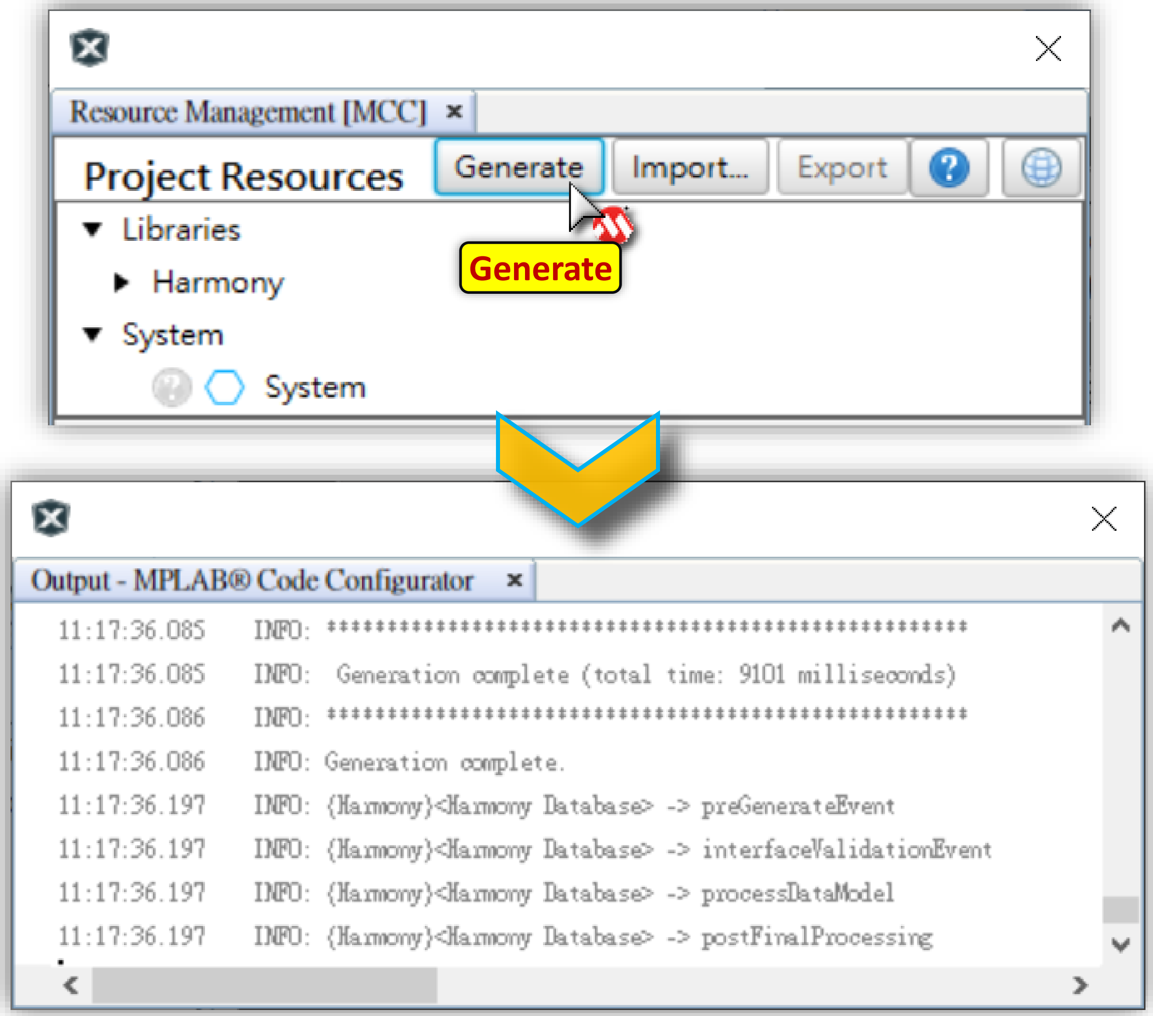
- **Pin Diagram. (Hardware design reference)**



Lab1 PORT Output - Pin Configuration using MCC

Step 6

- Click "Generate" Button to Generate your Code.



MCC Harmony GPIO Macros

Plib_port.h

```
/*** Macros for LED1 pin ***/
#define LED1_Set()              (PORT_REGS->GROUP[0].PORT_OUTSET = 1 << 20)
#define LED1_Clear()            (PORT_REGS->GROUP[0].PORT_OUTCLR = 1 << 20)
#define LED1_Toggle()           (PORT_REGS->GROUP[0].PORT_OUTTGL = 1 << 20)
#define LED1_Get()              (((PORT_REGS->GROUP[0].PORT_IN >> 20)) & 0x01)
#define LED1_OutputEnable()     (PORT_REGS->GROUP[0].PORT_DIRSET = 1 << 20)
#define LED1_InputEnable()      (PORT_REGS->GROUP[0].PORT_DIRCLR = 1 << 20)
#define LED1_PIN                PORT_PIN_PA20
```

Plib_port.c

```
void PORT_Initialize(void)
{
  /************************** GROUP 0 Initialization *************************/
  PORT_REGS->GROUP[0].PORT_DIR = 0x100000;
  PORT_REGS->GROUP[0].PORT_OUT = 0x100000;

  /************************** GROUP 1 Initialization *************************/
}
```

Lab1 PORT Output - Pin Configuration using MCC

```
// TODO 1.01
uint32_t i = 0;

int main (void)
{
  SYS_Initialize ( NULL );

  // TODO 1.02
  PORT_REGS->GROUP[0].PORT_DIRSET = 1<<20;

  while( true )
  {
    // TODO 1.03
    for ( i = 0 ; i < 2000000 ; i++ ) {}
    PORT_REGS->GROUP[0].PORT_OUTTGL = 1<<20;
  }
}
```

Access by Pre-defined Struct

Period Control

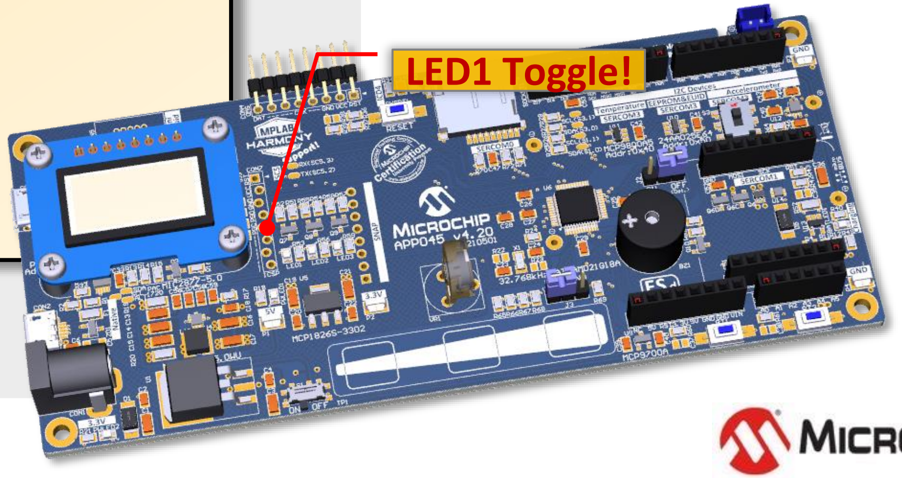
Step 7

a Modify code segment to your main loop

```
// TODO 1.01
uint32_t i = 0;

int main (void)
{
   SYS_Initialize ( NULL );

   // TODO 1.02
   // PORT_REGS->GROUP[0].PORT_DIRSET = 1<<20;

   while( true )
   {
      // TODO 1.03
      for ( i = 0 ; i < 2000000 ; i++ ) {}
      // PORT_REGS->GROUP[0].PORT_OUTTGL = 1<<20;
      LED1_Toggle();
   }
}
```

PORT initialization already done here

Comment out this line

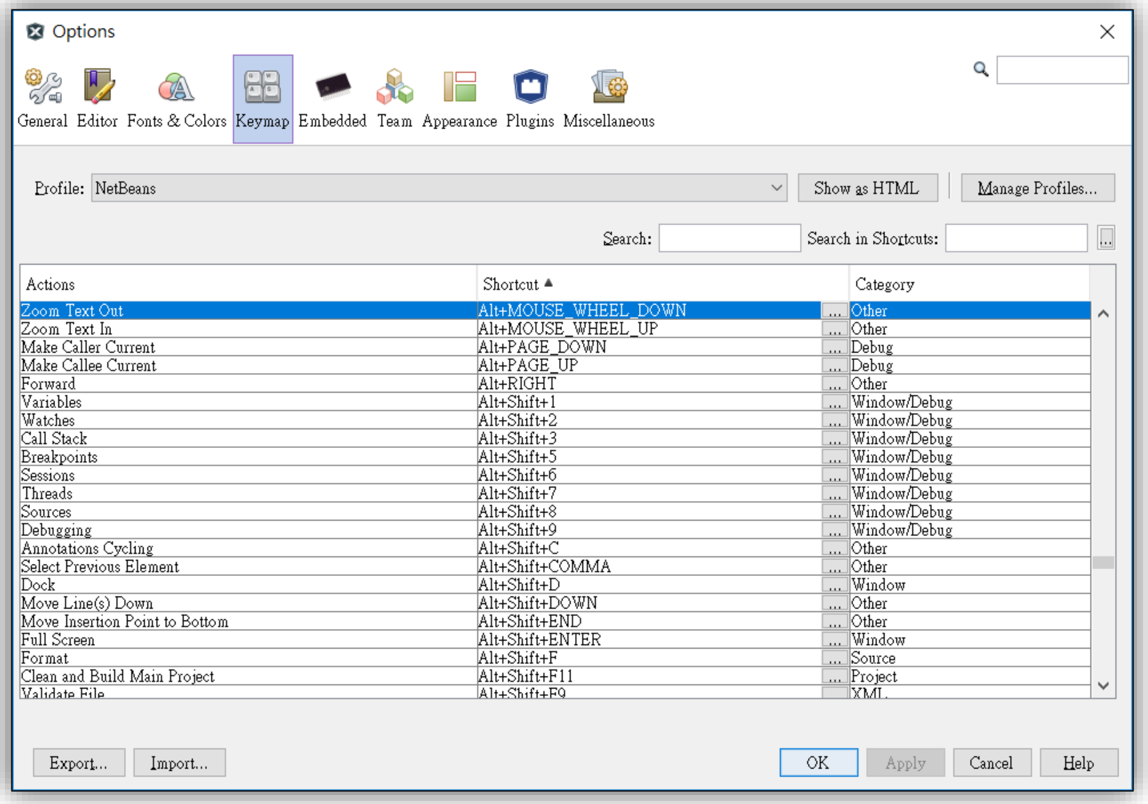


b Program firmware to target board then observe result.

MICROCHIP

MPLAB X IDE Shortcut key

- **MPLAB X IDE Provide a few shortcut key to enhance your performance.**
  - **Ctrl + Alt + ** : Show All Code Completion
    (Alternative shortcut : Ctrl **+ ** , Ctrl + Alt + Space**)**
  - **Ctrl + Mouse Left Click** : Open Macro (Code Trace)
  - **Alt + Left** : Backward (Code Trace)
  - **Alt + Shift + F** : Code Format
  - **Ctrl + Shift + C** :Toggle Comment



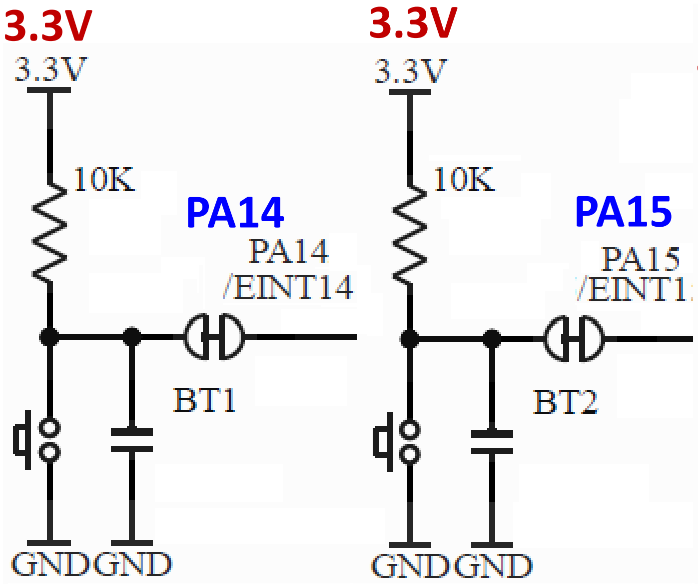
- **More ?**
  - Tools > Options > Keymap

Lab2 PORT Input and Output

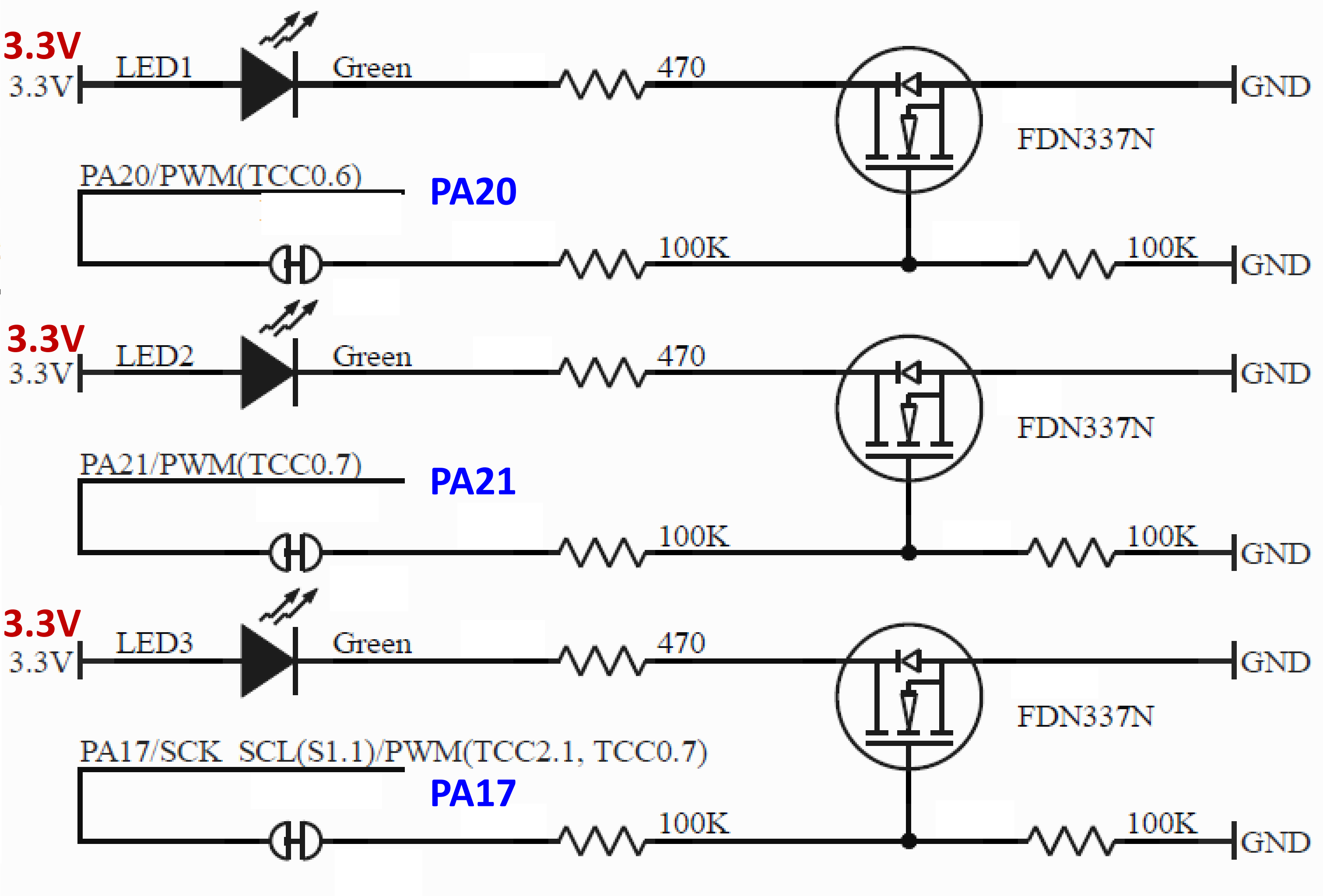
- Based on Lab1, Try to add LED2(PA21) continues toggle (Period around 500ms ~ 1S).
- Get BT1(PA14) status then use to control LED3(PA17).
  Pressed -> LED3 Light
  Released -> LED3 Dark

- Let's go!

Check Schematic



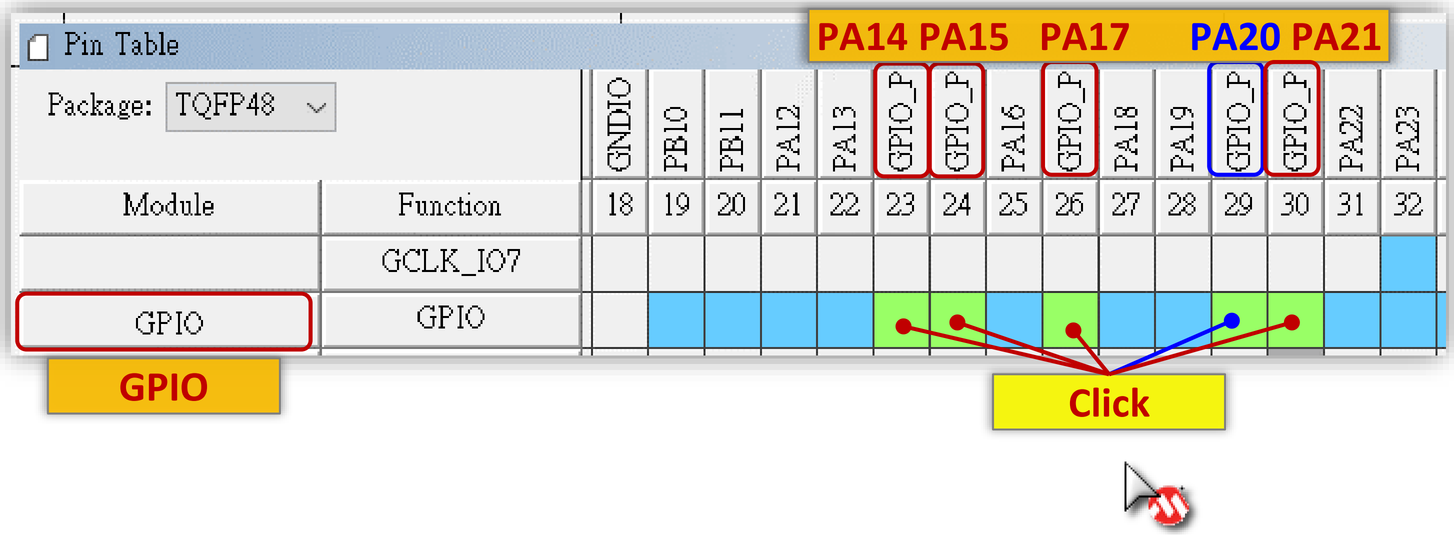
LED1	PA20
LED2	PA21
LED3	PA17
BT1	PA14
BT2	PA15



Lab2 PORT Input and Output

Step 1

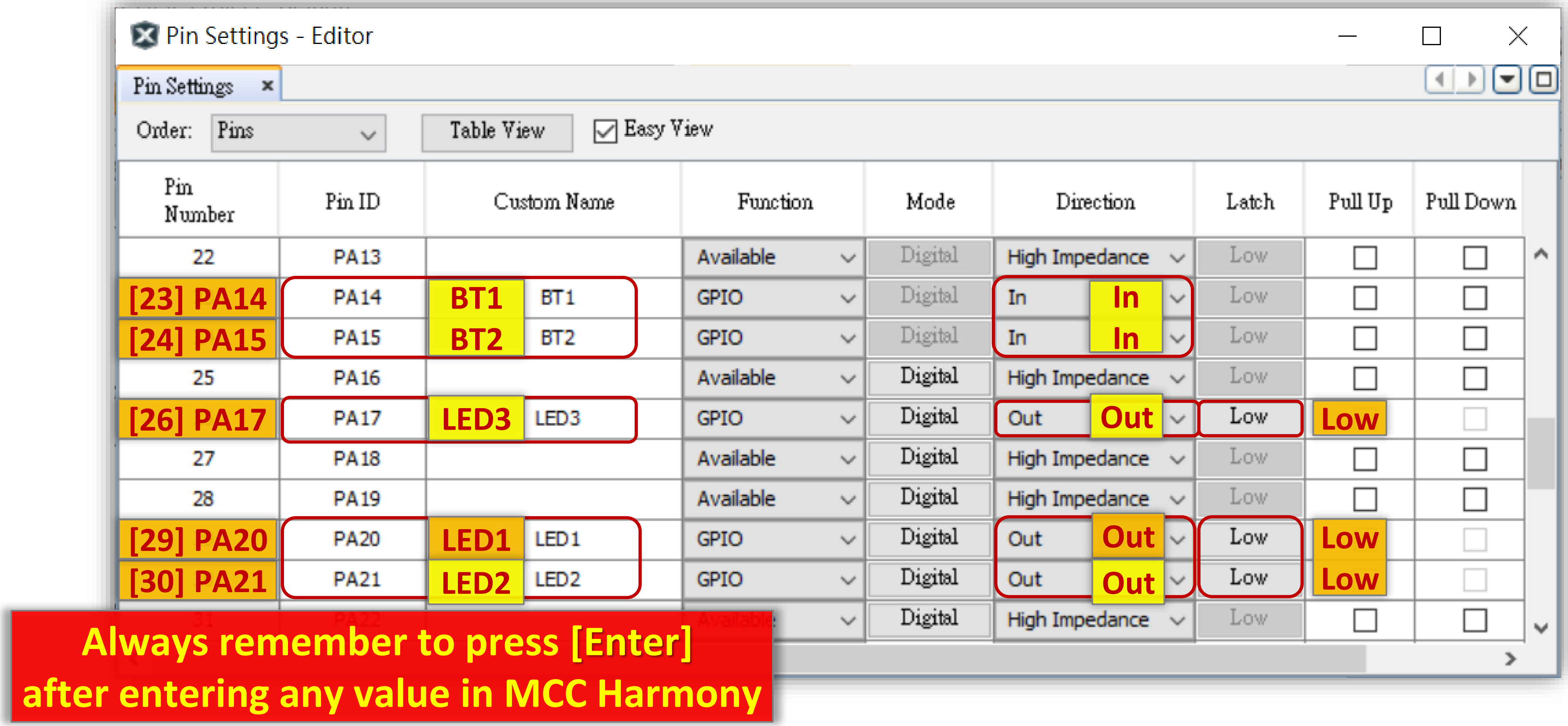
- **Pin Table configure. (Enable or Disable Pin)**



Lab2 PORT Input and Output

Step 2

- **Pin Setting. (Pin Function Setting)**

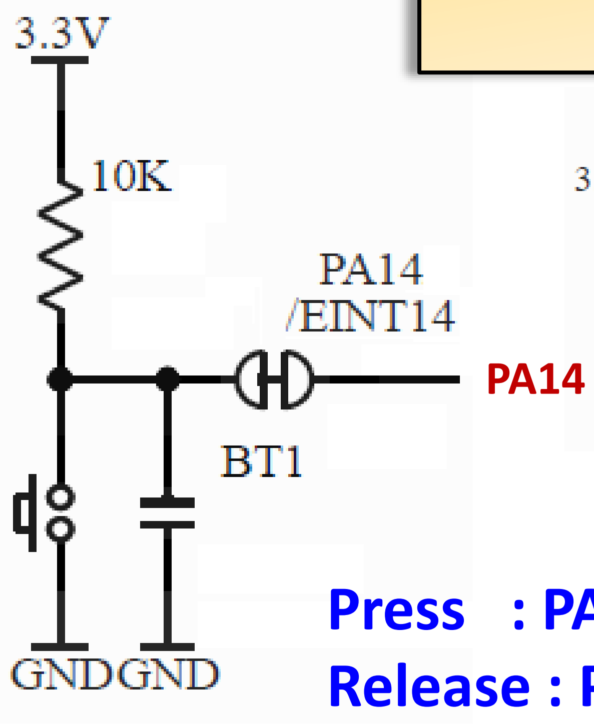


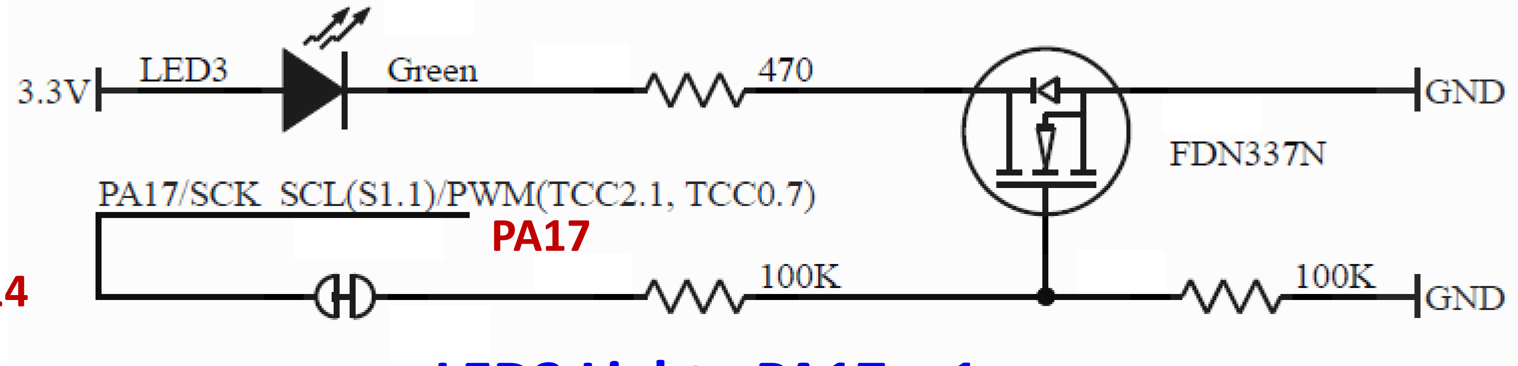
Lab2 PORT Input and Output

Schematic Describe

- Get **BT1(PA14)** status then use to control **LED3(PA17).**
  Pressed -> LED3 Light , Released -> LED3 Dark

```
if ( BT1_Get() ) LED3_Clear();
else             LED3_Set();
```





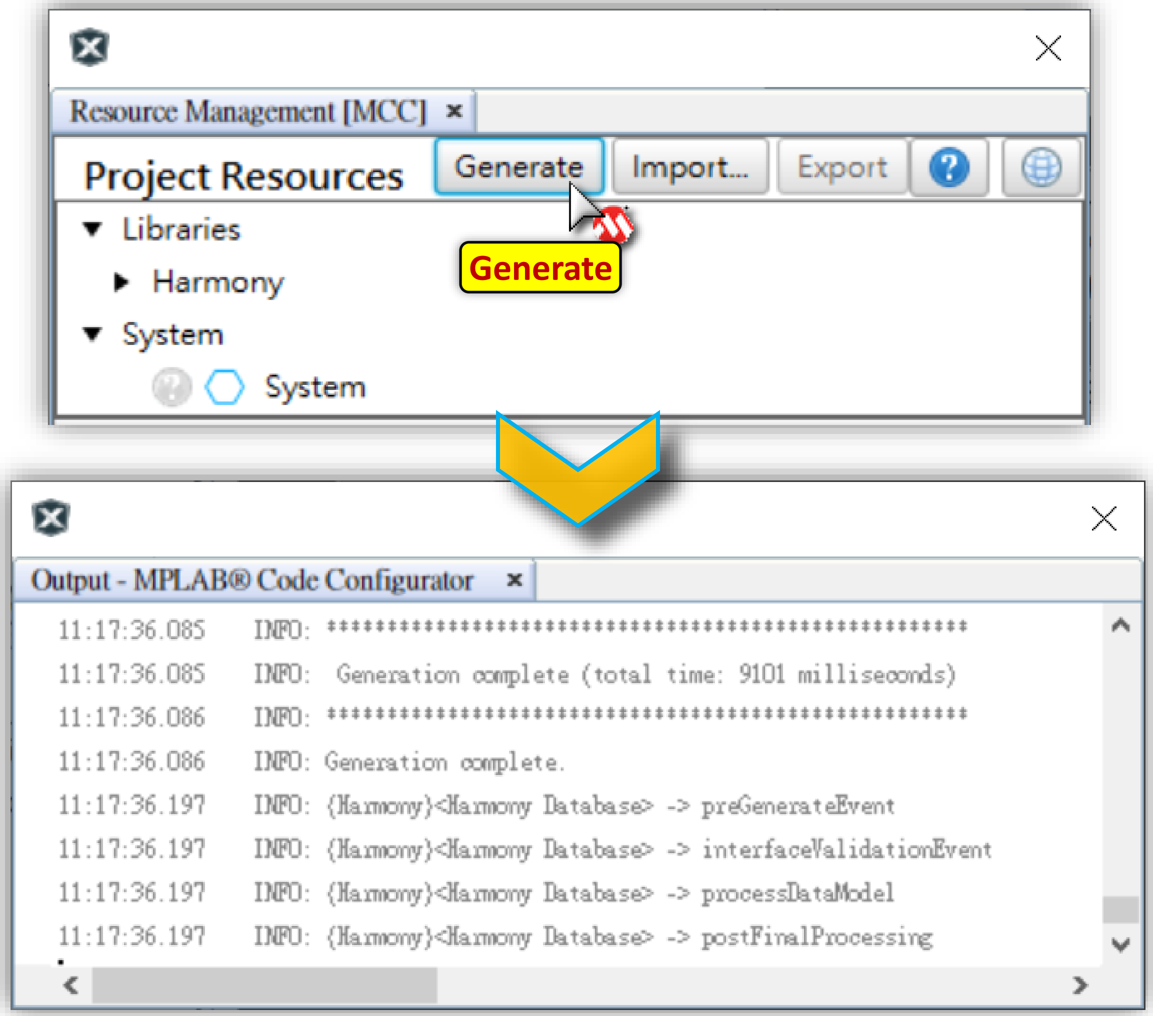
LED3 Light : PA17 = 1
LED3 Dark : PA17 = 0

Press : PA14 = 0
Release : PA14 = 1

Lab2 PORT Input and Output

Step 3

- Click "Generate" Button to Generate your Code.



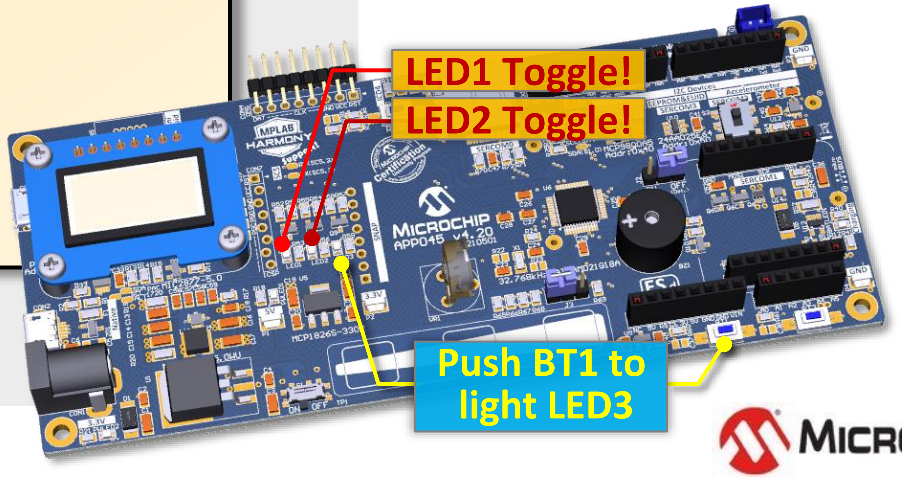
Lab2 PORT Input and Output

Step 4

a **Add code segment to your main loop**

```
uint32_t i = 0;
int main (void)
{
   SYS_Initialize ( NULL );

   while( true )
   {
      for ( i = 0 ; i < 2000000 ; i++ ){}

      // TODO 2.01
       LED1_Toggle();
       LED2_Toggle();

      // TODO 2.02
       if ( BT1_Get() ) LED3_Clear();
       else             LED3_Set();
   }
}
```

b **Program firmware to target board then observe result.**



MICROCHIP

Lab2 PORT Input and Output

Button Response Discuss

- **Button Response Time Issue ?**
  The software delay will influence the polling interval.
  Try to change coding style to state machine mode.

```
while( true )
{
  for ( i = 0; i < 2000000 ; i++ ){}

  LEDs_Toggle();

  if ( Button_Detected() )
    LED_Light();
  else
    LED_Dark();
}
```

```
while( true )
{
  if ( i++ > 2000000 )
  {
    i = 0;

    LEDs_Toggle();
  }

  if ( Button_Detected() )
    LED_Light();
  else
    LED_Dark();
}
```

Lab2 PORT Input and Output

Step 5

a change coding style to state machine mode

```
uint32_t i = 0;
int main (void)
{
   SYS_Initialize ( NULL );

   while( true )
   {
      if ( i++ > 2000000 )
      {
         i = 0;

         // TODO 2.01
         LED1_Toggle();
         LED2_Toggle();
      }

      // TODO 2.02
      if ( BT1_Get() ) LED3_Clear();
      else             LED3_Set();
   }
}
```

How about current LED toggle speed after change to state machine flow control ? Why ?

Timer/Counter, TC Architecture

SAMD21 Timer/Counter

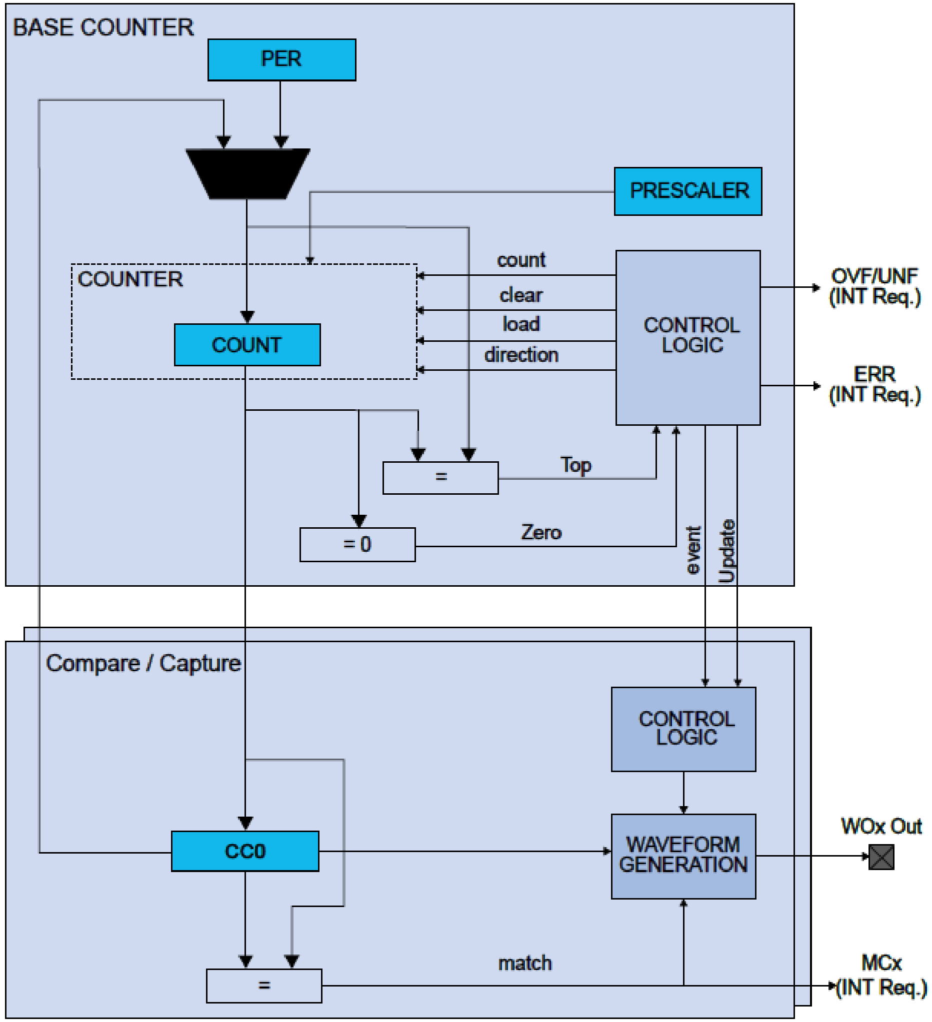
- The SAMD21 Timer/Counter, TC module consists 3 function Counter, Compare and Capture.

- The counter can be set to count events, or it can be configured to count clock pulses. The counter, together with the compare/capture channels, can be configured to timestamp input events, allowing capture of frequency and pulse width.

- It can also perform waveform generation, such as frequency generation and pulse-width modulation (PWM).

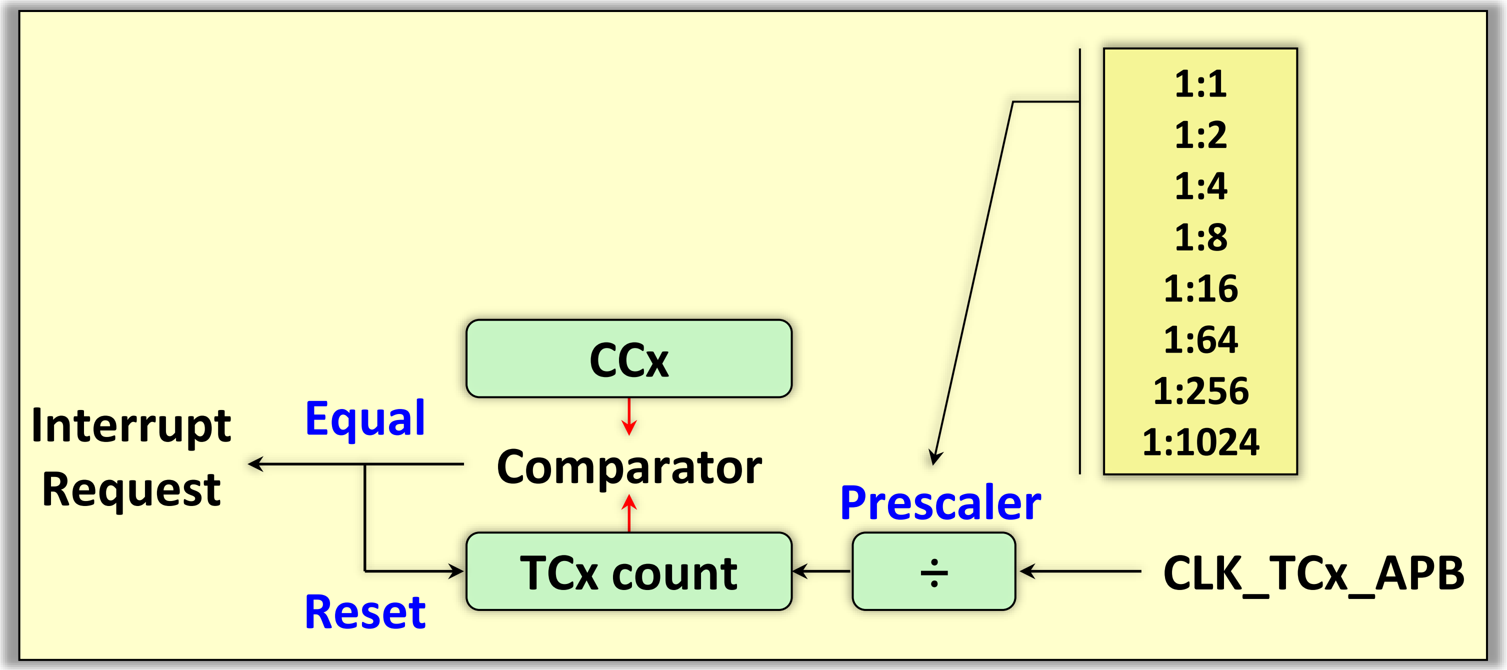
TC Block Diagram

- **SAMD21G18A TC Block Diagram**



SAMD21 Timer/Counter

- Please refer to the block diagram, just a concept. The real and detail block diagram please refer to the datasheet.
- Clock into TCx after prescaler. Comparator continuously compares the values of TCx count and CCx. Asserts a into the interrupt request when equal, and resets TCx count to zero.
- You can fill a value to CCx to determine period you want.



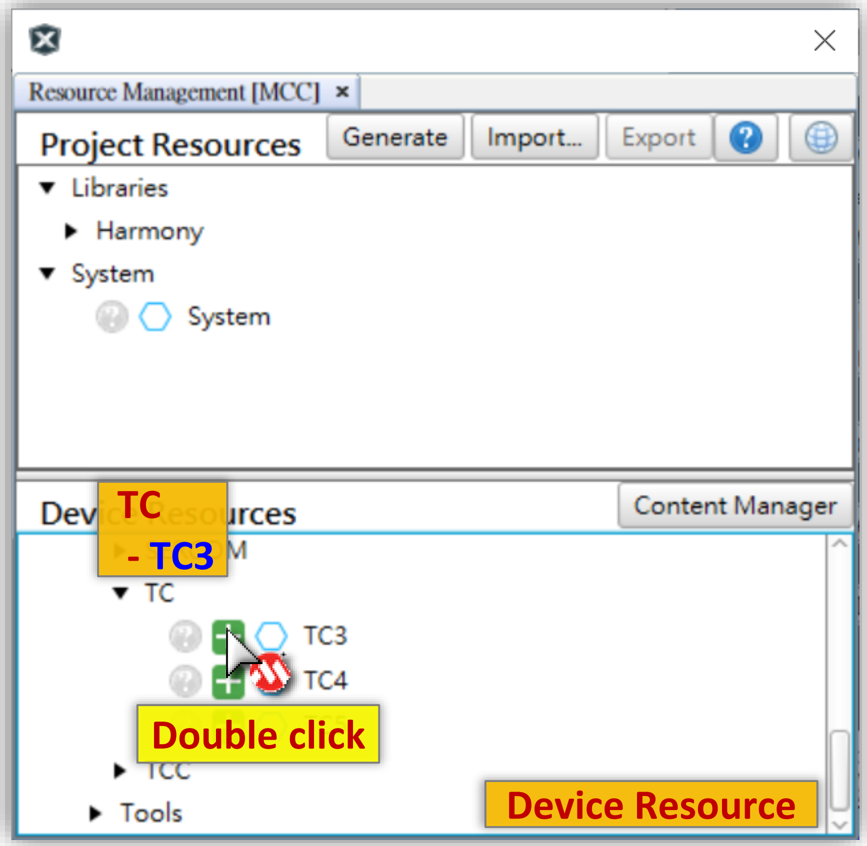
Match Pulse-Width Modulation (MPWM), 16Bit, Up count

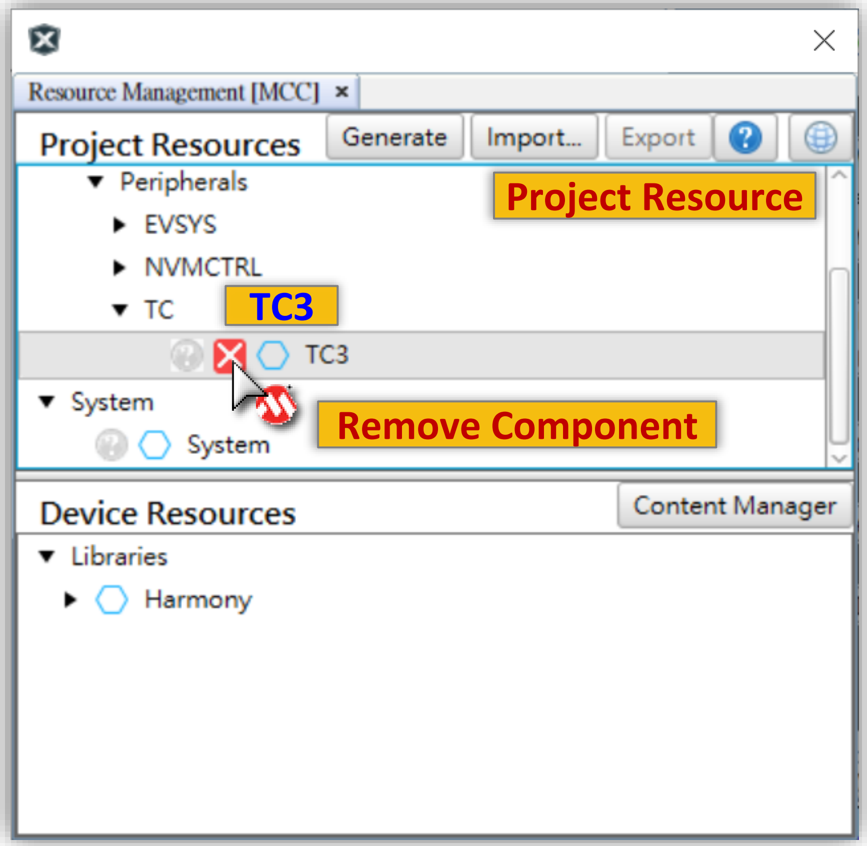
Timer/Counter

- **TC provide 4 difference operation mode for Timer, Counter and Compare.**
  This training only force on MFRQ mode. It's easy to use.
  if you want to know more, please refer appendix Timer/Counter Operation Mode.
  - **Normal Frequency Generation (NFRQ)**
    toggled on each compare match
  - **Match Frequency Generation (MFRQ)**
    toggles on each update condition
  - **Normal Pulse-Width Modulation Operation (NPWM)**
    toggles on each match & update condition
  - **Match Pulse-Width Modulation Operation (MPWM)**
    toggles on each match & update condition

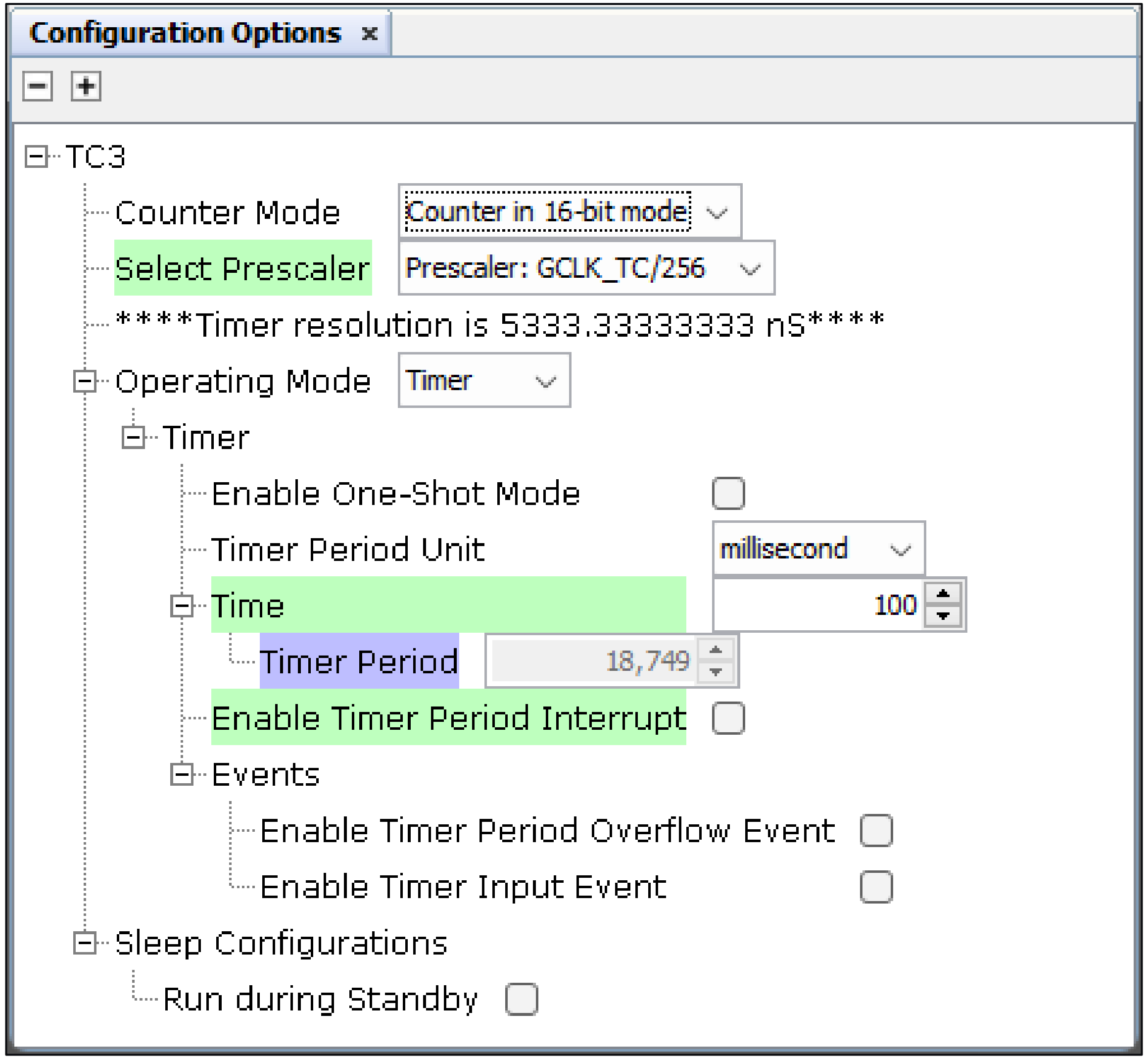
Add TC Function using MCC Harmony

- Find TC3 component in Device Resource window.
- Double click TC3 to add to Project Resource window.
- Click ❎ could remove component from Project Resource window.





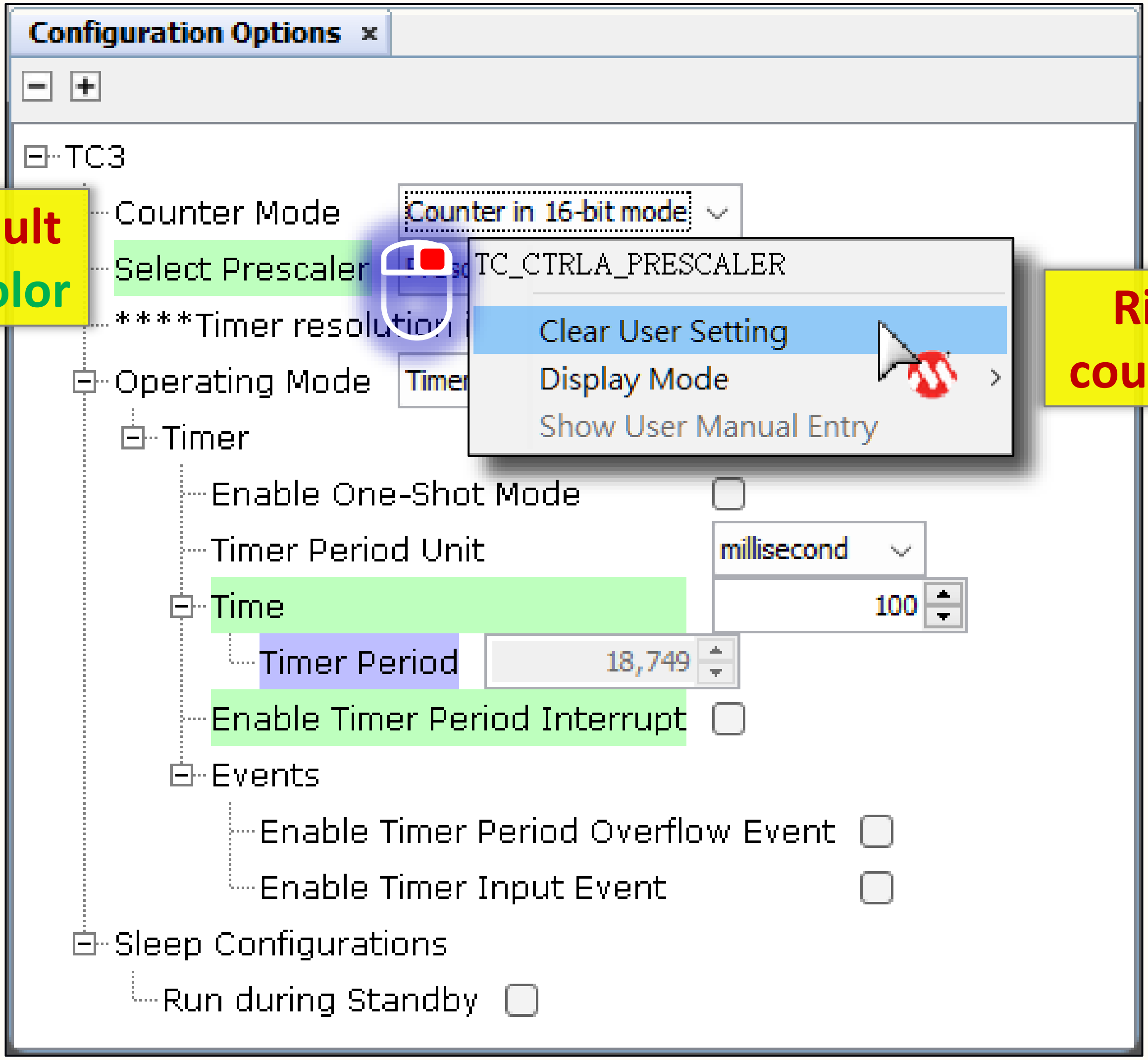
TC Configuration Options Example



TC Configuration Options Example

Hint

Value different to default will change to green color



Right click the value could Clear User Setting

TC Polling Functions

Interface Routines

```
void TC3_TimerInitialize( void );
void TC3_TimerStart( void );
void TC3_TimerStop( void );
uint32_t TC3_TimerFrequencyGet( void );
void TC3_Timer16bitPeriodSet( uint16_t period );
uint16_t TC3_Timer16bitPeriodGet( void );
uint16_t TC3_Timer16bitCounterGet( void );
void TC3_Timer16bitCounterSet( uint16_t count );
bool TC3_TimerPeriodHasExpired ( void );
```

TC Interrupt Flag Event

- **The TC has the following interrupt sources:**
  - **Overflow/Underflow (OVF)**
    Counter overflow/underflow
  - **Match or Capture Channel x (MCx)**
    Compare match or capture
  - **Capture Overflow Error (ERR)**
    New capture interrupt occurs while interrupt flag is set.
  - **Synchronization Ready (SYNCRDY)**
    Read/Write synchronization ready.

- **How to polling TC overflow interrupt flag status ?**

TC Polling Code Example

TC Timer Method (MPWM Mode) Overflow Polling Example

```
…
TC3_TimerStart();

while( true )
{
    if (TC3_TimerPeriodHasExpired ())
    {
        LED1_Toggle();
    }
}
```

```
bool TC3_TimerPeriodHasExpired( void )
{
    bool timer_status;
    timer_status = (bool) ((TC3_REGS->COUNT16.TC_INTFLAG) & TC_INTFLAG_OVF_Msk);
    TC3_REGS->COUNT16.TC_INTFLAG = timer_status;
    return timer_status;
}
```

Lab3 TC Timer Method Polling

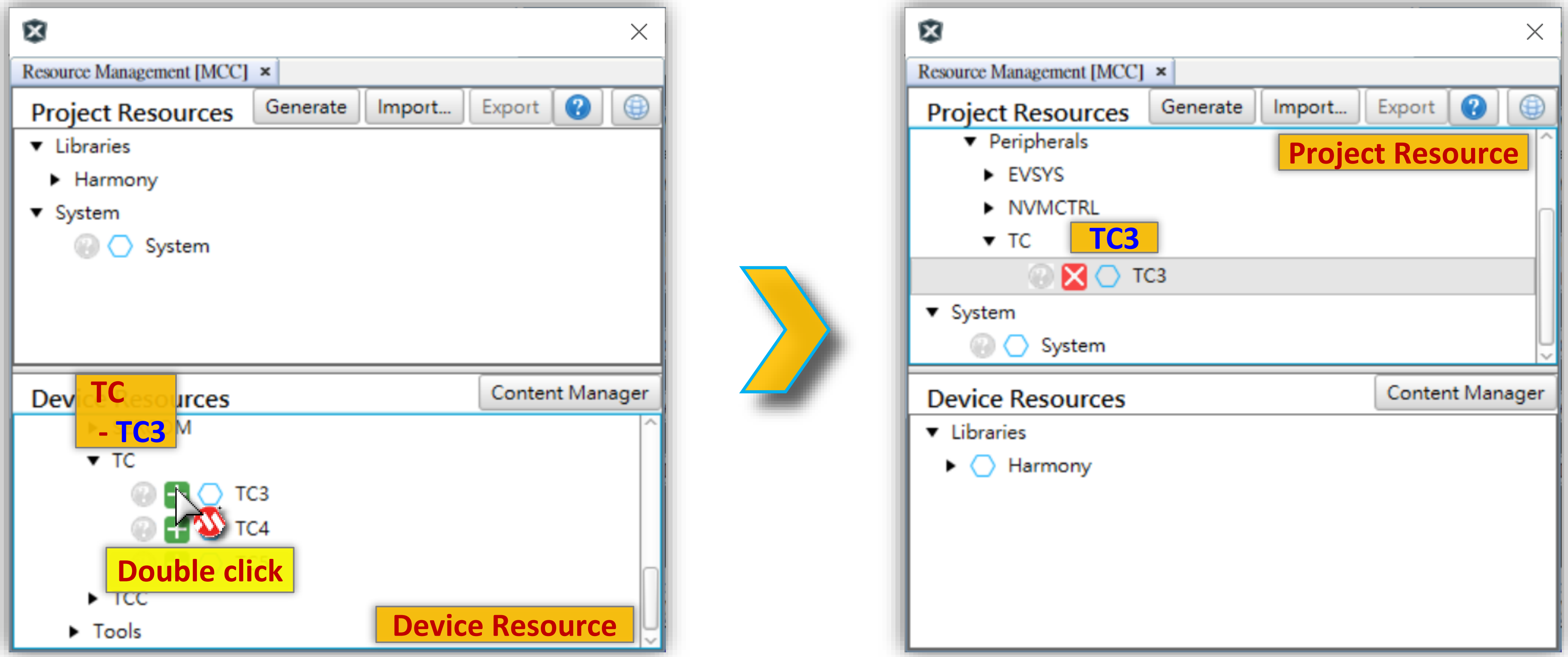
- Try adding TC3 instead of software delay to control LED1(PA20) to continues toggle.
- The TC3 setup to Timer and Polling method (It's implemented by MPWM 16-bit mode), and setup the Timer period to 100ms.

- **Let's go!**

Lab3 TC Timer Method Polling

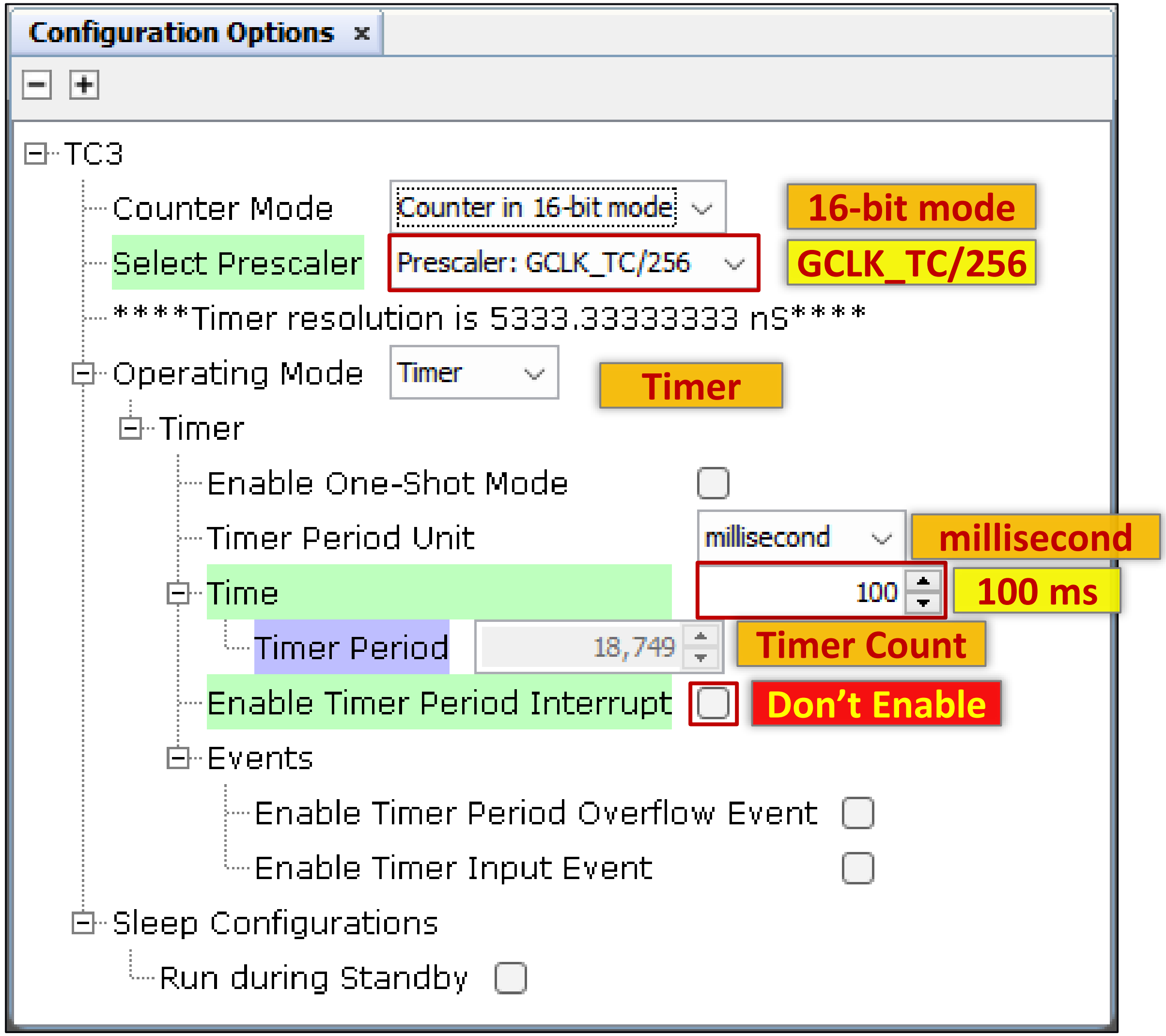
Step 1

- Find TC3 component in Device Resource window.
- Double click TC3 to add to Project Resource window.



Lab3 TC Timer Method Polling

Step 2



System Clock = 48MHz

One second counts of system clock :

48,000,000 counts

One second counts after Prescaler :

48,000,000 / 256 187,500 counts

100ms counts for target :

187,500 / 10 18,750 counts

Counter clear to zero at the next tick :

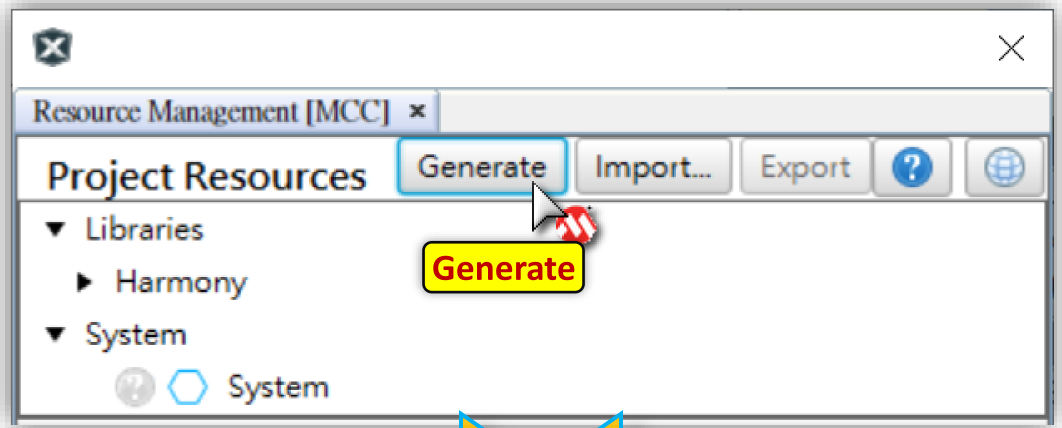
18,749 + 1 18,750 counts

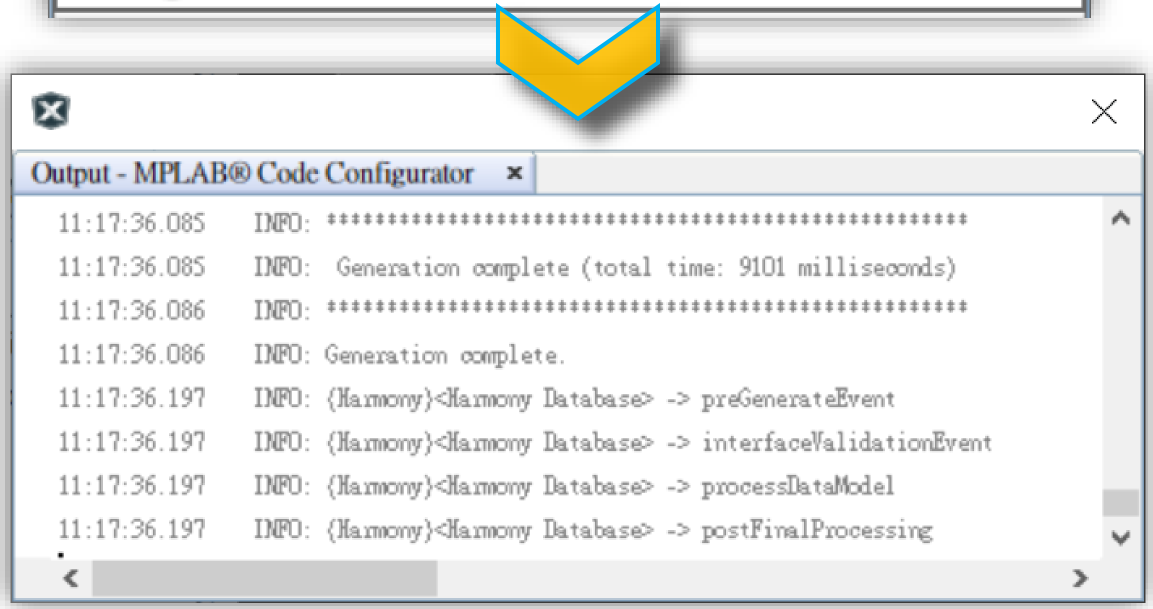
How to calculate count 18749 ?

Lab3 TC Timer Method Polling

Step 3

- Click "Generate" Button to Generate your Code.





Lab3 TC Timer Method Polling

Step 4

- **Remove or comment the following code in the previous Labs first.**

```
int main ( void )
{
    …
    while ( true )
    {
        …

        if ( ++i >= 2000000 )
        {
            i = 0;
            // TODO 3.01
            // LED1_Toggle();
            LED2_Toggle();
        }
        …
    }
    …
}
```

Lab3 TC Timer Method Polling

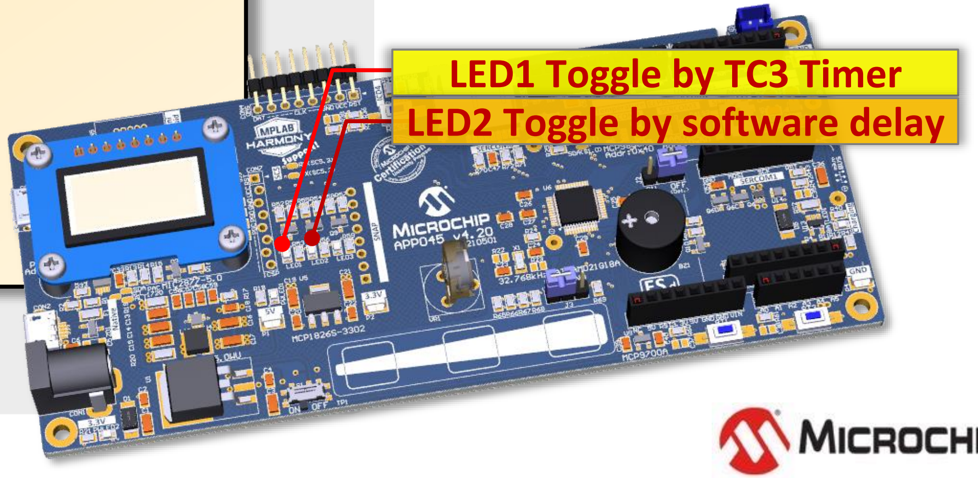
Step 5 & Result

a Add code segment to your main loop

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

    // TODO 3.02
     TC3_TimerStart();

    while( true )
    {
       // TODO 3.03
        if ( TC3_TimerPeriodHasExpired() )
        {
              LED1_Toggle();
        }
       …
    }
}
```

Period Control

b Program firmware to target board then observe result.



Lab4 TCs Timer Method Polling

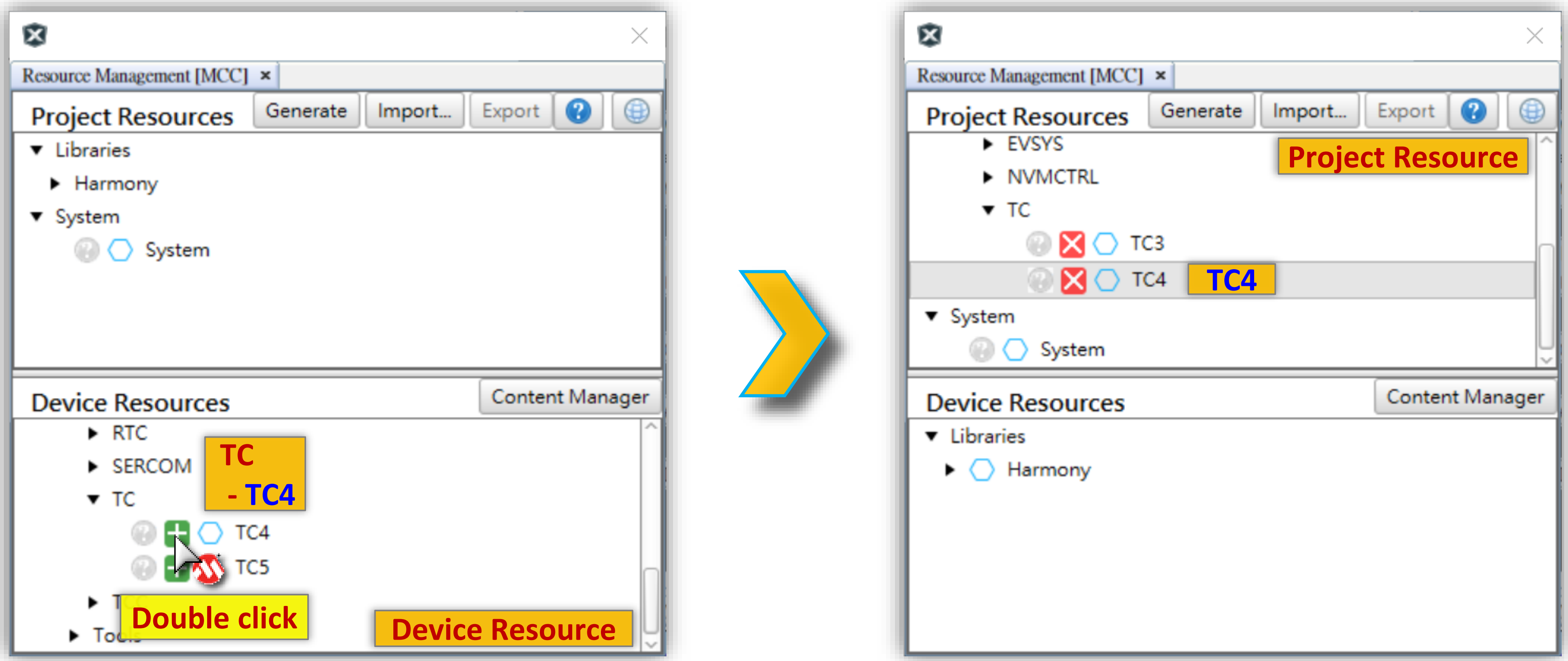
- Try add TC4 to replace software delay for control LED2(PA21) to continues toggle.
- The TC4 setup to Timer and Polling method (It’s implemented by MPWM 16-bit mode), and setup the Timer period to 50ms.

- **Let’s go!**

Lab4 TCs Timer Method Polling

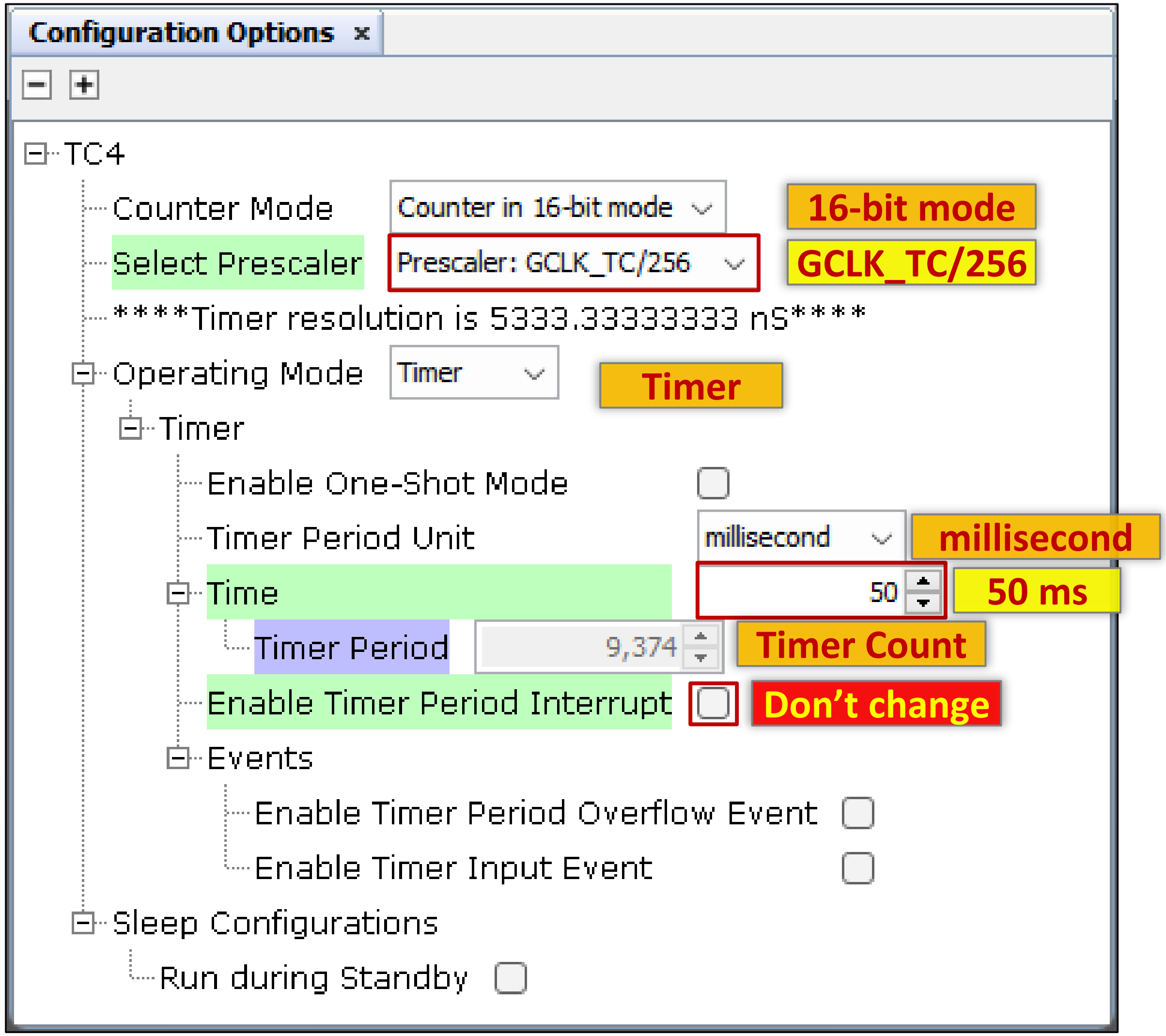
Step 1

- Find TC4 component in Device Resource window.
- Double click TC4 to add to Project Resource window.



Lab4 TCs Timer Method Polling

Step 2



System Clock = 48MHz

One second counts of system clock :

48,000,000 counts

One second counts after Prescaler :

48,000,000 / 256 187,500 counts

50ms counts for target :

187,500 / 20 9,375 counts

Counter clear to zero at the next tick :

9,374 + 1 9,375 counts

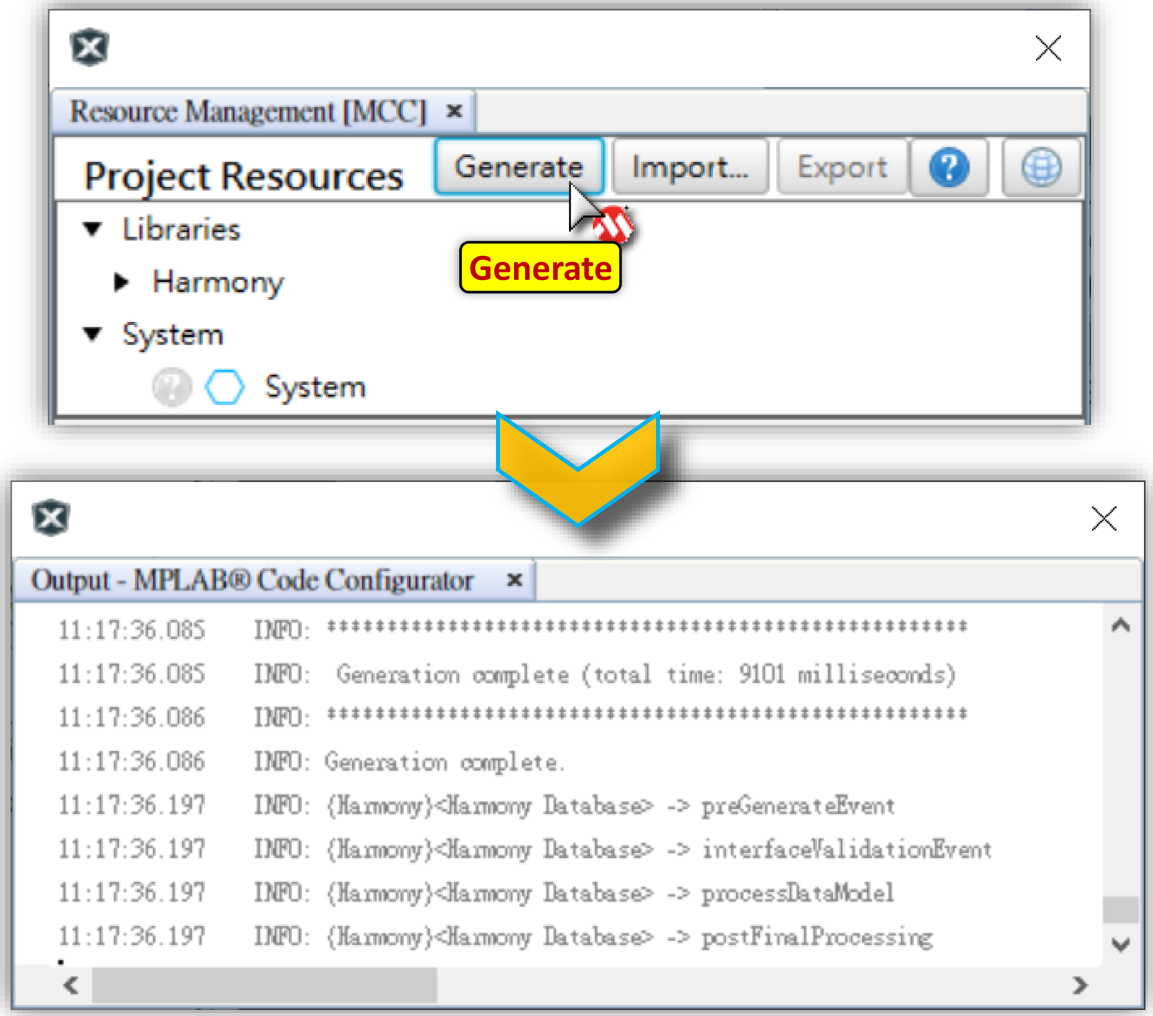
How to calculate count 9374 ?

MICROCHIP

Lab4 TCs Timer Method Polling

Step 3

- Click "Generate" Button to Generate your Code.



Lab4 TCs Timer Method Polling

Step 4

- **Remove or comment the following code in the previous Labs first.**

```
int main ( void )
{
    …
    while ( true )
   {
        …
        // TODO 4.01
        // if ( ++i >= 2000000 )
        // {
        //      i = 0;
        //      LED1_Toggle();
        //      LED2_Toggle();
        //  }
        …
   }
   …
}
```

Lab4 TCs Timer Method Polling

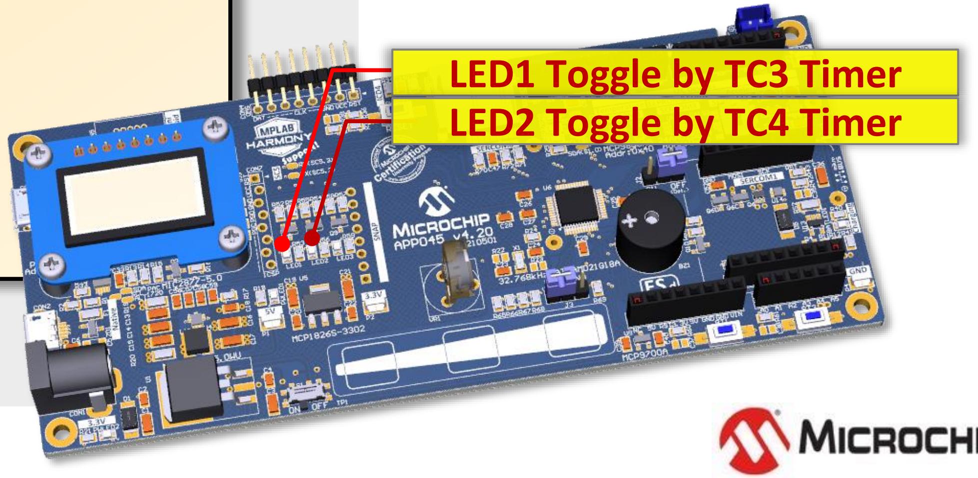
Step 4 & Result

a Add code segment to your main loop

```
int main (void)
{
   SYS_Initialize ( NULL );
   …
   TC3_TimerStart();
   // TODO 4.02
   TC4_TimerStart ();

   while( true )
   { …
     // TODO 4.03
     if ( TC4_TimerPeriodHasExpired() )
     {
          LED2_Toggle();
     }
     …
   }
}
```

Period Control

b Program firmware to target board then observe result.



MICROCHIP

NVIC Architecture

Nested Vector Interrupt Controller Architecture

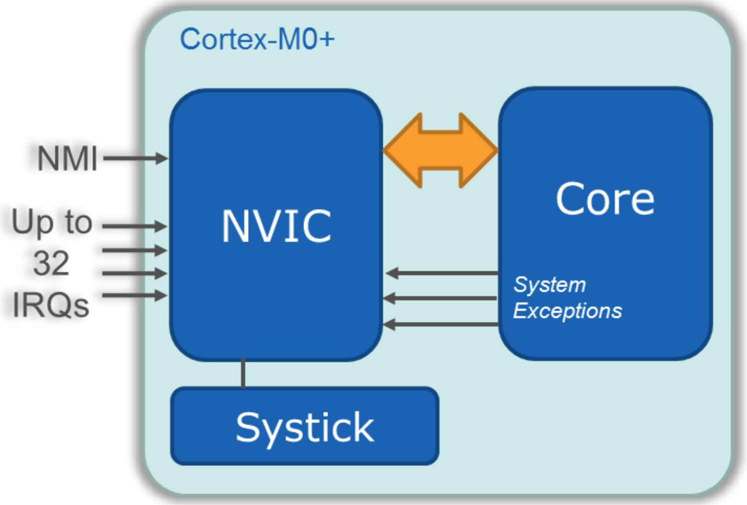
Nested Vector Interrupt Controller

ARM® Cortex® M0+

- NVIC provides an interface between interrupt sources (peripherals and external pins) and the core.

- The **priority (Level 0 ~ 3)** for each interrupt source is programmable, **Level 0 is highest priority**.

- Each interrupt lines has connected to one peripheral instance and **have one or more interrupt flags.**



- An interrupt request is generated from the peripheral when the **interrupt flag** has been set and the corresponding **interrupt is enabled**.

Natural Priority

- **If two pending interrupts share the same priority, priority is given to the interrupt with the lowest exception number (lowest interrupt vector address).**

Exception number	IRQ number	Vector	Offset
16+n	n	IRQn	0x40+4n
.		.	.
.		.	.
.		.	.
18	2	IRQ2	0x48
17	1	IRQ1	0x44
16	0	IRQ0	0x40
15	-1	SysTick, if implemented	0x3C
14	-2	PendSV	0x38
13		Reserved	
12			
11	-5	SVCall	0x2C
10			
9			
8			
7		Reserved	
6			
5			
4			0x10
3	-13	HardFault	0x0C
2	-14	NMI	0x08
1		Reset	0x04
		Initial SP value	0x00

Each peripheral has been assigned to one of Vector table and based to address 0x00000000

TC3 (18)

TC4 (19)

Peripheral Source	NVIC Line
EIC NMI – External Interrupt Controller	NMI
PM – Power Manager	0
SYSCTRL – System Control	1
WDT – Watchdog Timer	2
RTC – Real Time Counter	3
EIC – External Interrupt Controller	4
NVMCTRL – Non-Volatile Memory Controller	5
DMAC - Direct Memory Access Controller	6
USB - Universal Serial Bus	7
EVSYS – Event System	8
SERCOM0 – Serial Communication Interface 0	9
SERCOM1 – Serial Communication Interface 1	10
SERCOM2 – Serial Communication Interface 2	11
SERCOM3 – Serial Communication Interface 3	12
SERCOM4 – Serial Communication Interface 4	13
SERCOM5 – Serial Communication Interface 5	14
TCC0 – Timer Counter for Control 0	15
TCC1 – Timer Counter for Control 1	16
TCC2 – Timer Counter for Control 2	17
TC3 – Timer Counter 3	18
TC4 – Timer Counter 4	19
TC5 – Timer Counter 5	20
TC6 – Timer Counter 6	21
TC7 – Timer Counter 7	22
ADC – Analog-to-Digital Converter	23
AC – Analog Comparator	24
DAC – Digital-to-Analog Converter	25
PTC – Peripheral Touch Controller	26
I2S - Inter IC Sound	27

SAMD21G18A IVT Define

Interrupt.c

```
__attribute__ ((section(".vectors")))
const DeviceVectors exception_table=
{
    /* Configure Initial Stack Pointer, using linker-generated symbols */
    .pvStack = (void*) (&_stack),

    .pfnReset_Handler            = ( void * ) Reset_Handler,
    .pfnNonMaskableInt_Handler = ( void * ) NonMaskableInt_Handler,
    .pfnHardFault_Handler        = ( void * ) HardFault_Handler,
    .pfnSVCall_Handler           = ( void * ) SVCall_Handler,
    .pfnPendSV_Handler           = ( void * ) PendSV_Handler,
    .pfnSysTick_Handler          = ( void * ) SysTick_Handler,
    .pfnNVMCTRL_Handler          = ( void * ) NVMCTRL_Handler,
     …
    .pfnSERCOM5_Handler          = ( void * ) SERCOM0_Handler,
     …
    .pfnTCC2_Handler             = ( void * ) TCC2_Handler,
    .pfnTC3_Handler              = ( void * ) TC3_TimerInterruptHandler,
    .pfnTC4_Handler              = ( void * ) TC4_TimerInterruptHandler,
    …
    .pfnADC_Handler              = ( void * ) ADC_Handler,
    …
};
```

ISR Function Example

- So, You need Implement ISR (Interrupt Service Routine) by yourself like below ?

plib_tc3.c

```
#include "plib_tc3.h"
…

void TC3_TimerInterruptHandler( void )
{
    TC_TIMER_STATUS status;
    status = (TC_TIMER_STATUS) (TC3_REGS->COUNT16.TC_INTFLAG);
    /* Clear interrupt flags */
    TC3_REGS->COUNT16.TC_INTFLAG = TC_INTFLAG_Msk;

    // Getting starter add your code here now. !?


}
```

Now ? Hands-on Lab ?

No!

Callback Function

- MCC module provide the interrupt mode driver.
- All flag and event handling by MCC interrupt interface routine automatically, user just focus on application. It's more powerful and easy to understand than operating a NVIC ISR function directly.
- The interrupt mode provide "Callback" function to hook up your customer function to specific interrupt source and event.
- For Example,
  You can hook the LED toggle callback function in TC overflow event. Callback function will be executed automatically while TC overflow event has occurred.

MCC Harmony Interrupt Example

- **Register a custom "Callback" function and write the action you need to execute.**
  - **The action will be automatically executed when interrupt occurs !**

Callback Function

Write the action you need to execute.

```c
void TC3_TimerExpired(...)
{
    if (status & TC_INTFLAG_OVF_Msk)
    {
        LED1_Toggle();
    }
}
```

Callback Register

Register the "Callback" function you defined.

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

    TC3_TimerCallbackRegister(TC3_TimerExpired, ...);
    TC3_TimerStart();

    while ( true )
    {
        SYS_Tasks ( );
    }

    return ( EXIT_FAILURE );
}
```

Call Callback

Provided by MCC Harmony, automatically executed when interrupt occurs.

```c
/* Timer Interrupt handler */
void __attribute__((used)) TC3_TimerInterruptHandler( void )
{
    TC_TIMER_STATUS status;
    status = (TC_TIMER_STATUS) (TC3_REGS->COUNT16.TC_INTFLAG);
    /* Clear interrupt flags */
    TC3_REGS->COUNT16.TC_INTFLAG = TC_INTFLAG_Msk;
    if(TC3_CallbackObject.callback != NULL)
    {
        uintptr_t context = TC3_CallbackObject.context;
        TC3_CallbackObject.callback(status, context);
    }
}
```

Automatically Generated

MICROCHIP

TC Callback Hook Example

```
void TCn_TimerExpired(TC_TIMER_STATUS status, uintptr_t context)
{
   if( status & TC_INTFLAG_OVF_Msk )
      LED_Toggle();
}

TCn_TimerCallbackRegister (TCn_TimerExpired , (uintptr_t )NULL );
TCn_TimerStart ();
```

Hook (Register)

Callback

Application

System

```
/* Register callback function */
void TCn_TimerCallbackRegister( TC_TIMER_CALLBACK callback, uintptr_t context )
{
   TCn_CallbackObject.callback = callback;

   TCn_CallbackObject.context = context;
}
```

Function Pointer

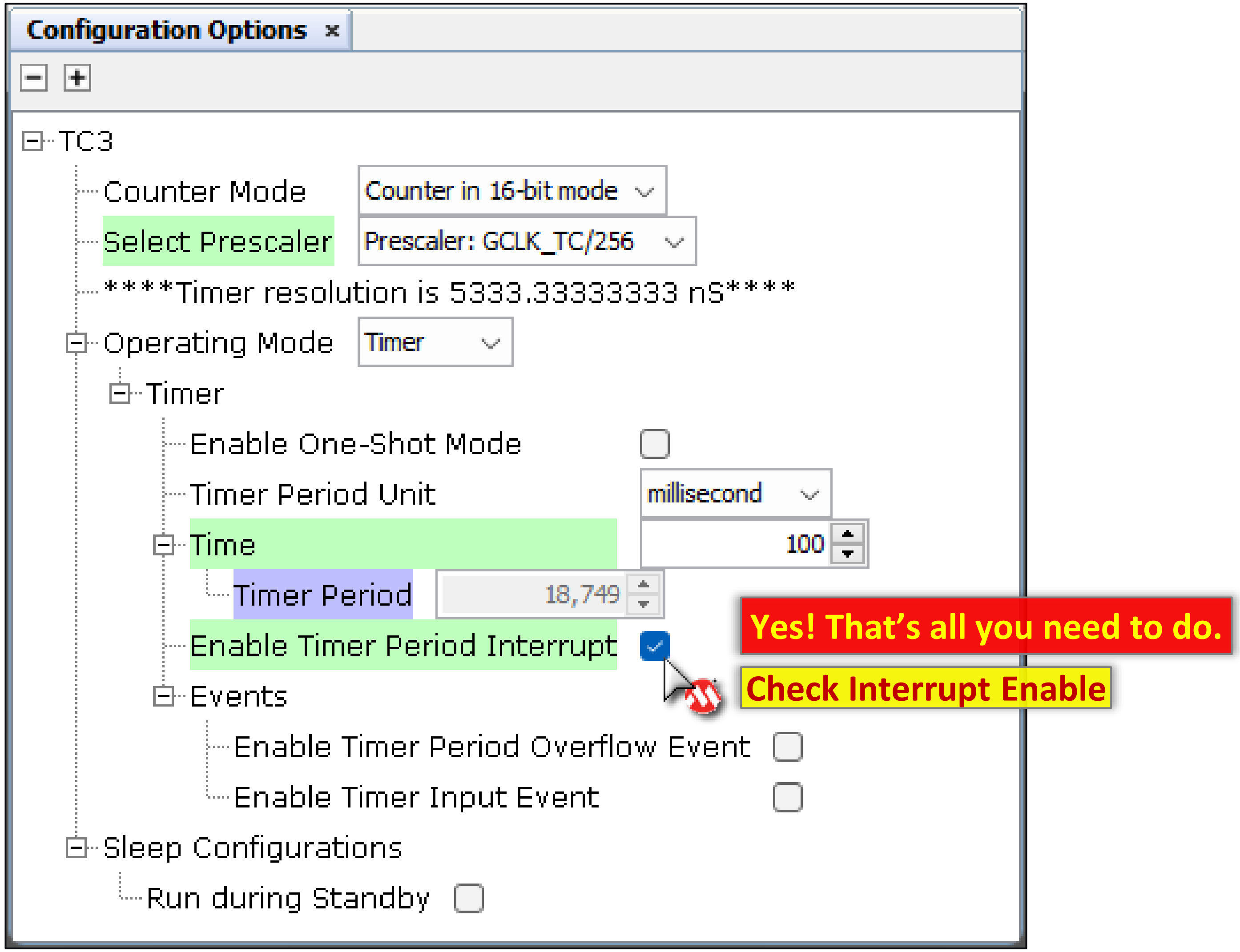
```
/* Timer Interrupt handler */
void TCn_TimerInterruptHandler ( void )
{
   TC_TIMER_STATUS status;
   status = (TC_TIMER_STATUS) TCn_REGS->COUNT16.TC_INTFLAG;
   /* Clear interrupt flags */
   TCn_REGS->COUNT16.TC_INTFLAG = TC_INTFLAG_Msk;
   if( TCn_CallbackObject.callback != NULL )
   {
     TCn_CallbackObject.callback( status, TCn_CallbackObject.context );
   }
}
```

Interrupt

MICROCHIP

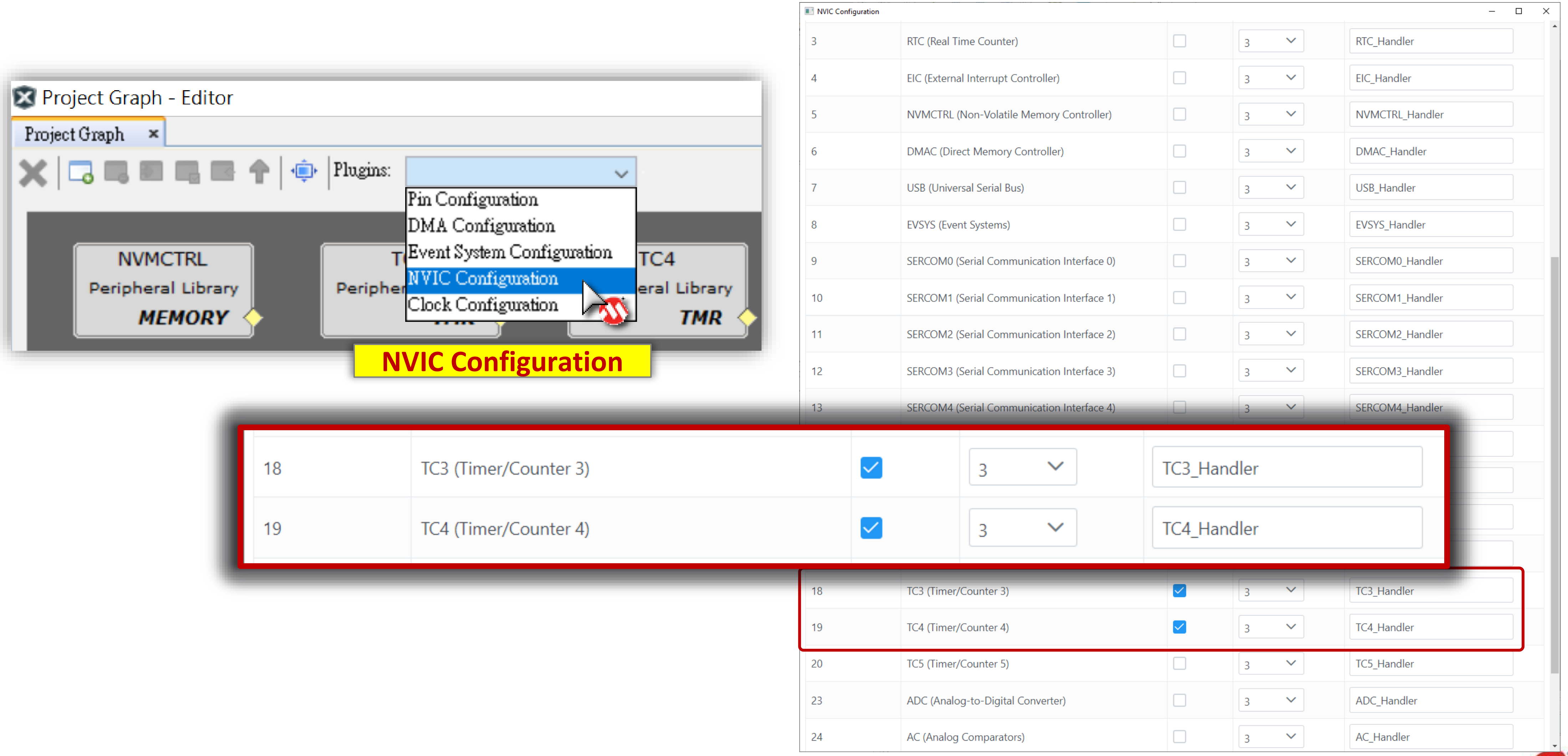
MCC Harmony Interrupt Example

- **Enable Interrupt in TC configuration windows.**
  - **Only one click !**



NVIC Priority Settings in MCC Harmony

- Open NVIC Settings windows in Project Graph > Plugins > NVIC Configuration.



TC Interrupt Functions

Interface Routines

```
void TC3_TimerInitialize( void );
void TC3_TimerStart( void );
void TC3_TimerStop( void );
uint32_t TC3_TimerFrequencyGet( void );
void TC3_Timer16bitPeriodSet( uint16_t period );
uint16_t TC3_Timer16bitPeriodGet( void );
uint16_t TC3_Timer16bitCounterGet( void );
void TC3_Timer16bitCounterSet( uint16_t count );
void TC3_TimerCallbackRegister( TC_Timer_CALLBACK callback, uintptr_t context );
```

A new function beyond polling mode.

Lab5 TC Timer Method Interrupt

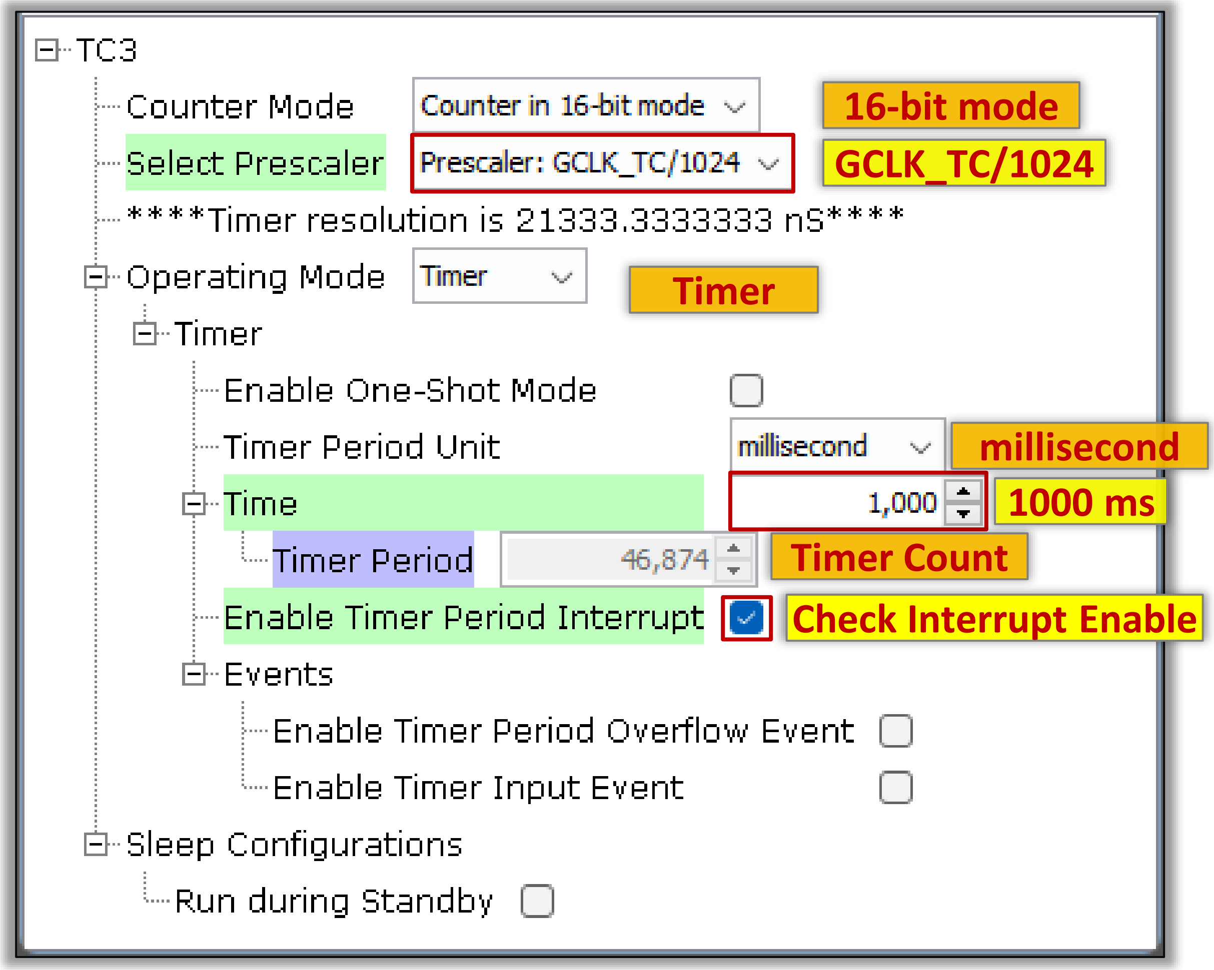
- **Try to configure TC3 to use Interrupt method.**
  LED1 will toggle in Timer Callback while Timer Period Expired interrupt.
- **Modify TC3 Timer Period to 1000ms in MCC for easy observe the toggle.**
- **Modify code segment in main() and follow below sequence.**
  - Firstly, remove Timer polling LED1 toggle from main loop.
  - Secondary, Create a TC3 Timer Callback function for LED1 toggle
  - Finally, hook(register) up TC3 Timer Callback function to TC3 Timer Interrupt event.

- **Let’s go!**

Lab5 TC Timer Method Interrupt

Step 1

- Configure TC3 Period to 1000 ms and Enable Period Interrupt in configuration.



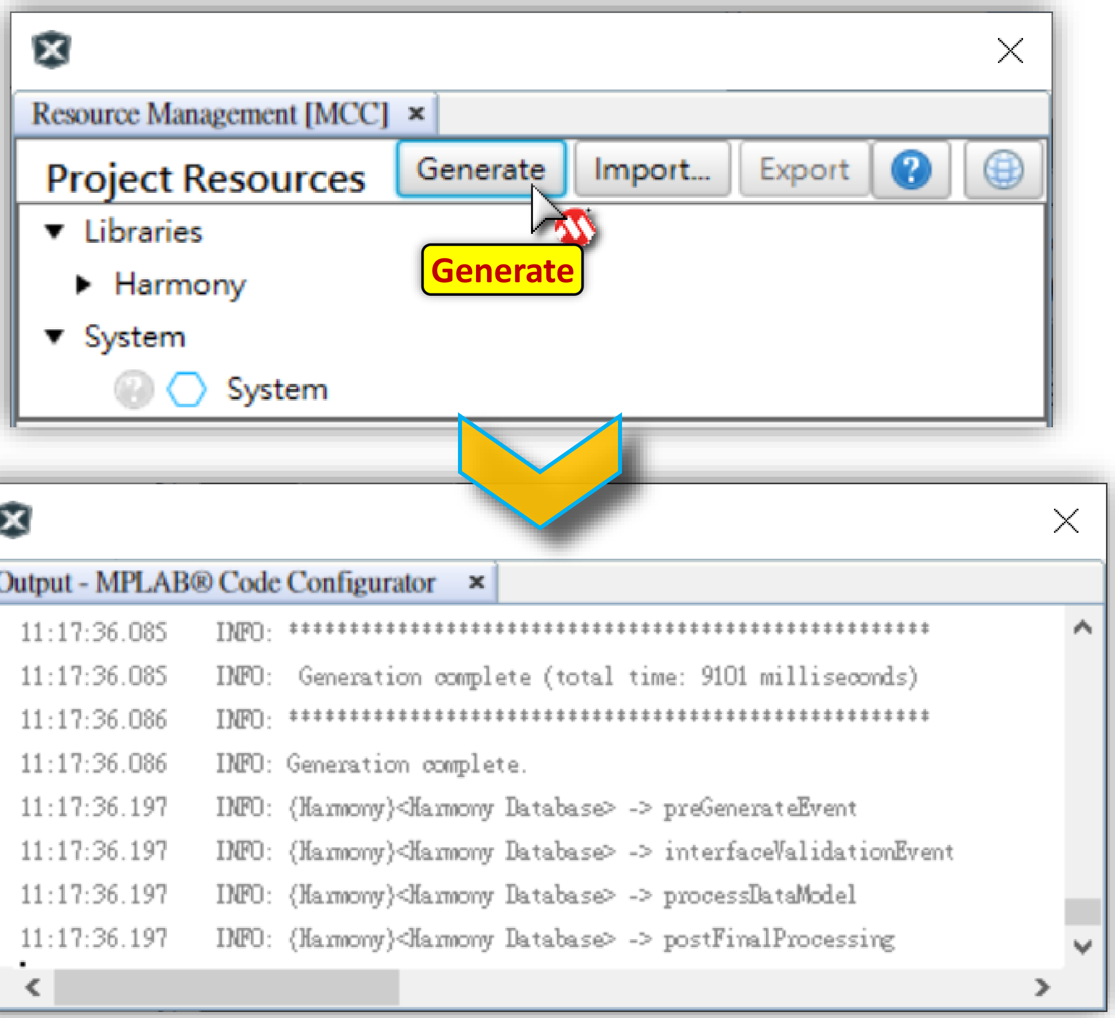
System Clock = 48MHz
One second counts of system clock :
48,000,000 counts
One second counts after Prescaler :
48,000,000 / 1024 46,875 counts
Counter clear to zero at the next tick :
46,874 + 1 46,875 counts

How to calculate count 46874 ?

Lab5 TC Timer Method Interrupt

Step 2

- Click "Generate" Button to Generate your Code.



Lab5 TC Timer Method Interrupt

Step 3

- **Remove or comment the following code in the previous Labs first.**

```
int main ( void )
{
    …
    while ( true )
    {
        …
        // TODO 5.01
        //  if ( TC3_TimerPeriodHasExpired() )
        //  {
        //      LED1_Toggle();
        //  }
        …
    }
    …
}
```

Lab5 TC Timer Method Interrupt

Step 4

a Add code segment.

```
// TODO 5.02
void TC3_TimerExpired(TC_TIMER_STATUS status, uintptr_t context)
{
  if( status & TC_INTFLAG_OVF_Msk )
  {
    LED1_Toggle();
  }
}

int main (void)
{
  SYS_Initialize ( NULL );
  …
  // TODO 5.03
  TC3_TimerCallbackRegister( TC3_TimerExpired, (uintptr_t )NULL );
  TC3_TimerStart();

  while( true )
  { …
  }
}
```

b Program firmware to target board then observe result.

Lab6 TCs Timer Method Interrupt

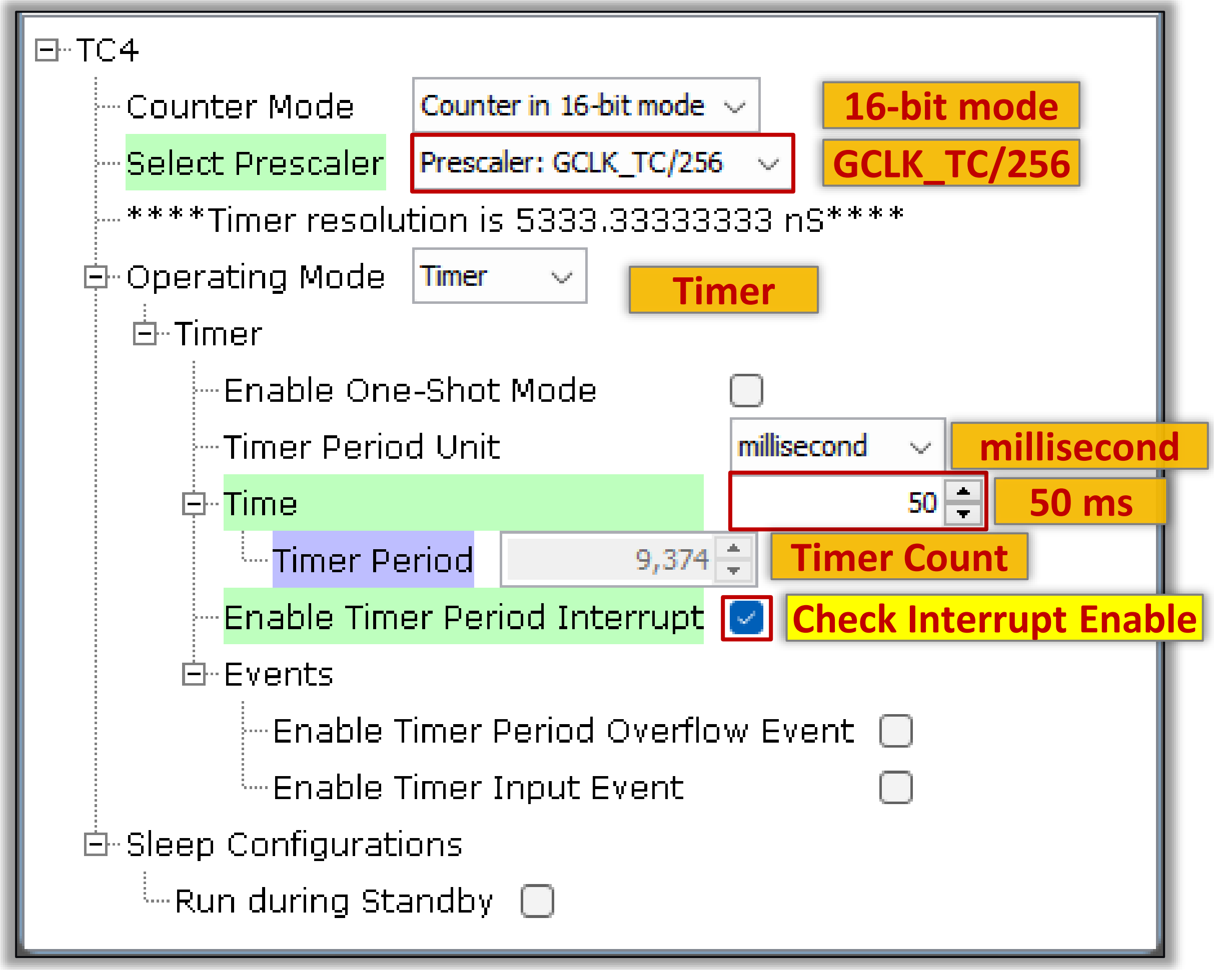
- **Try to configure TC4 to use Interrupt method.**
  LED2 will toggle in Timer Callback while Timer Period Expired interrupt.
- **Keep TC4 Timer Period as 50ms in MCC that we did in previous Lab.**
- **Modify code segment in main() and follow below sequence.**
  - Firstly, remove Timer polling LED2 toggle from main loop.
  - Secondary, Create a TC4 Timer Callback function for LED2 toggle
  - Finally, hook(register) up TC4 Timer Callback function to TC4 Timer Interrupt event.

- **Let’s go!**

Lab6 TCs Timer Method Interrupt

Step 1

- **Enable Period Interrupt** of TC4 in configuration windows.



System Clock = 48MHz

One second counts of system clock :

48,000,000 counts

One second counts after Prescaler :

48,000,000 / 256 187,500 counts

50ms counts for target :

187,500 / 20 9,375 counts

Counter clear to zero at the next tick :

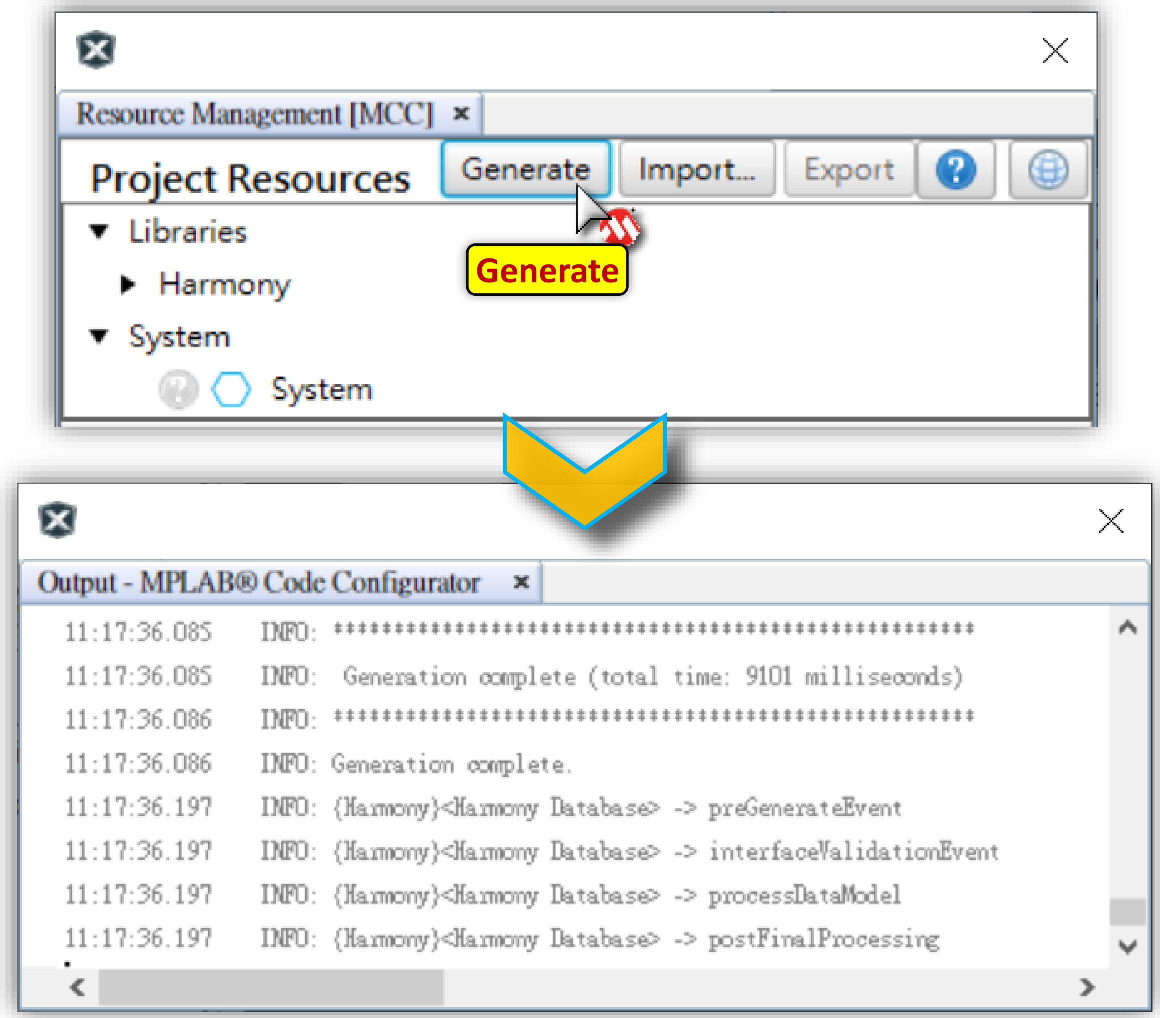
9,374 + 1 9,375 counts

How to calculate count 9374 ?

Lab6 TCs Timer Method Interrupt

Step 2

- Click "Generate" Button to Generate your Code.



Lab6 TCs Timer Method Interrupt

Step 3

- **Remove or comment the following code in the previous Labs first.**

```
int main ( void )
{
    …
    while ( true )
    {
        …
        // TODO 6.01
        //  if ( TC4_TimerPeriodHasExpired() )
        //  {
        //      LED2_Toggle();
        //  }
        …
    }
    …
}
```

Lab6 TCs Timer Method Interrupt

Step 4

a. Add code segment to your main loop

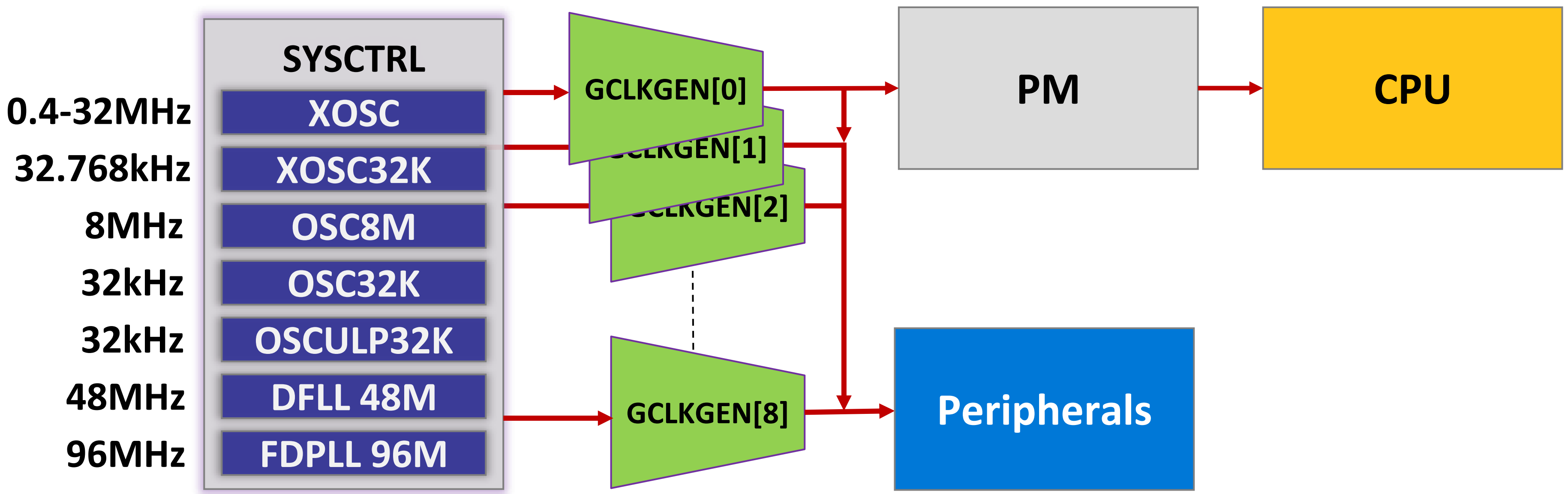
```
// TODO 6.02
void TC4_TimerExpired(TC_TIMER_STATUS status, uintptr_t context)
{
  if( status & TC_INTFLAG_OVF_Msk )
  {
    LED2_Toggle();
  }
}

int main (void)
{
  SYS_Initialize ( NULL );
  …
  // TODO 6.03
  TC4_TimerCallbackRegister( TC4_TimerExpired, (uintptr_t )NULL );
  TC4_TimerStart();

  while( true )
  { …
  }
}
```

b. Program firmware to target board then observe result.

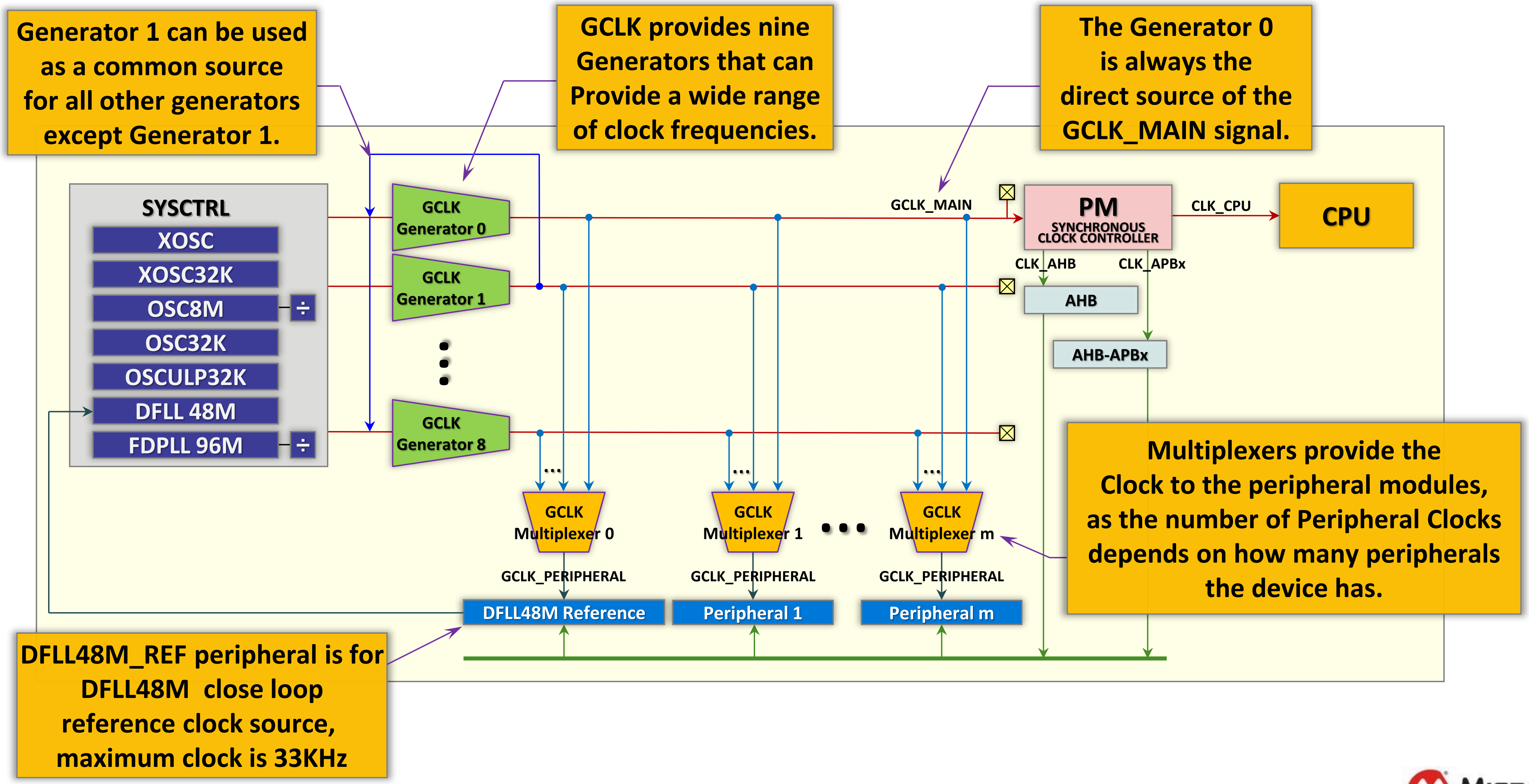
Clock System Architecture

Clock System

- Review previously section, SAMD21's clock system provides clock sources to the Generic Clock Controller. The available clock sources are XOSC, XOSC32K, OSC32K, OSCULP32K, OSC8M,DFLL48M and FDPLL96M.
- The bus clock can be enabled and disabled in the Power Manager(PM) include CPU Clock(GCLK_MAIN).

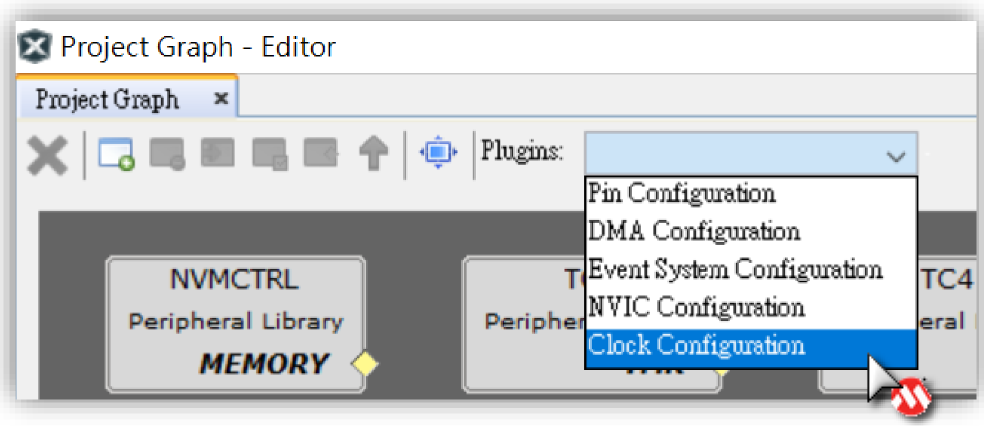


Clock System Block Diagram

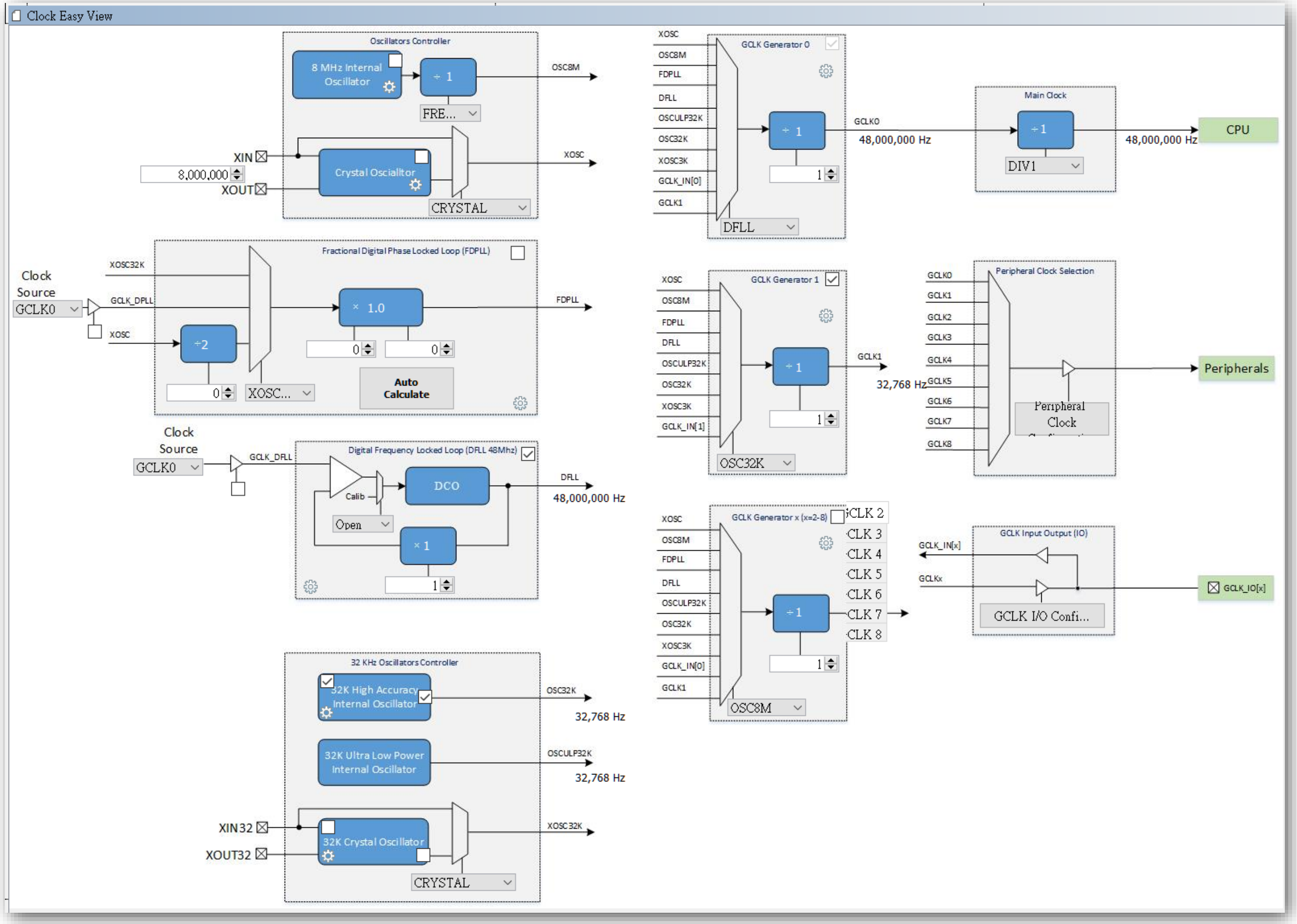


Clock Setting Use MCC Harmony

- Open Clock Easy View windows in Project Graph > Plugins > Clock Configuration.

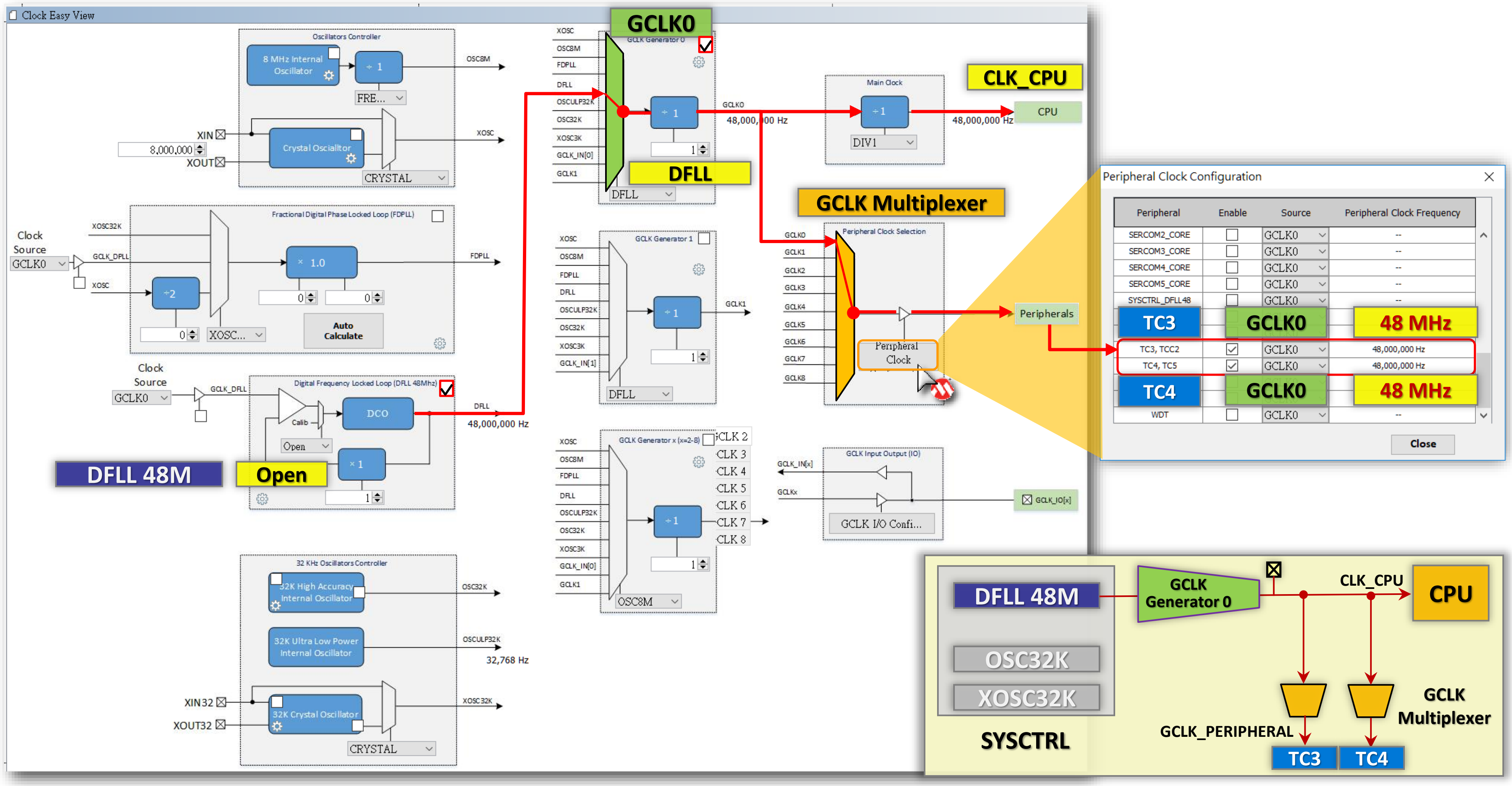


Clock Configuration



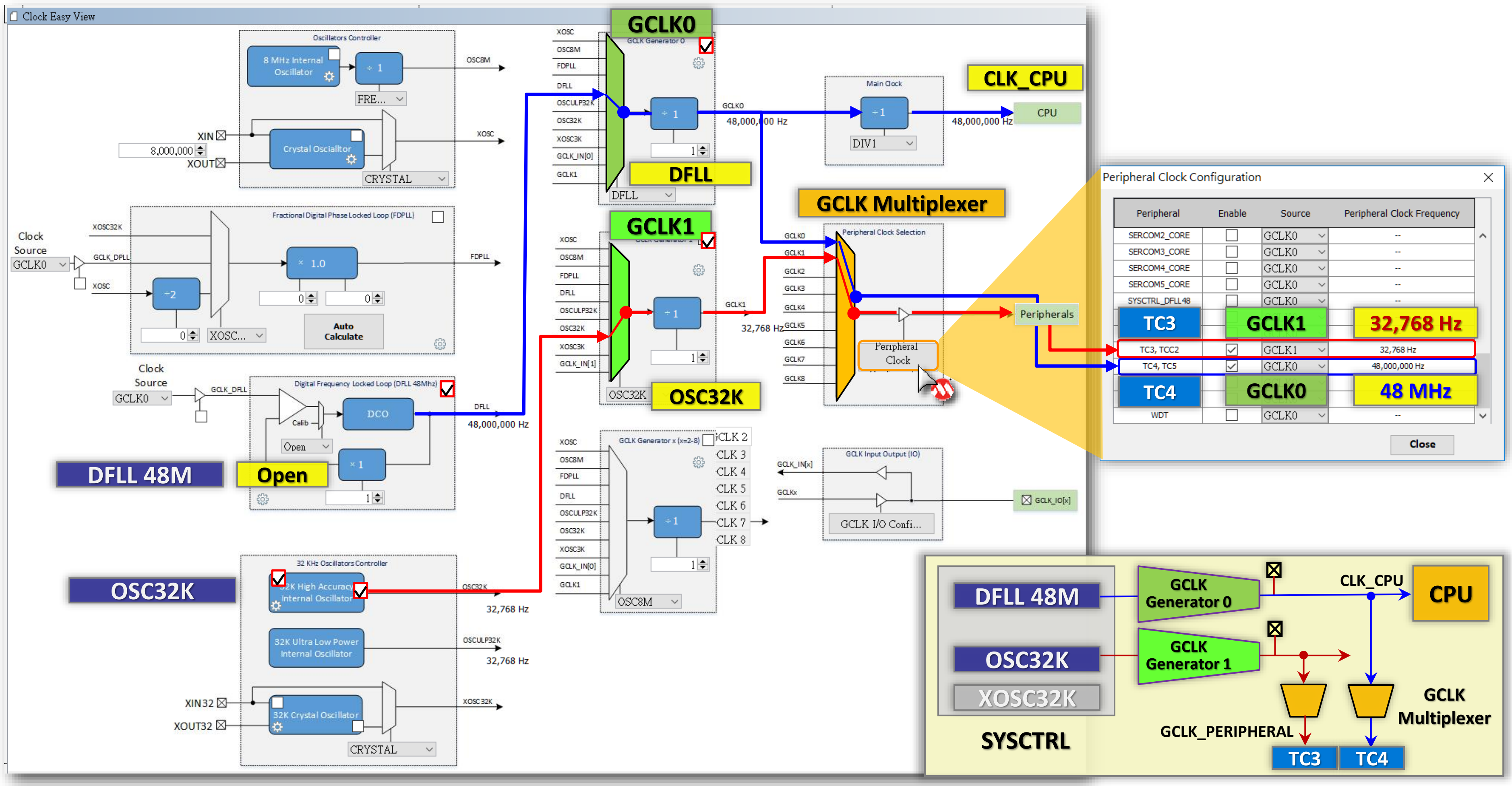
Clock Setting Use MCC Harmony

- **Default clock setting of new project (DFLL48M open loop).**



Clock Setting Use MCC Harmony

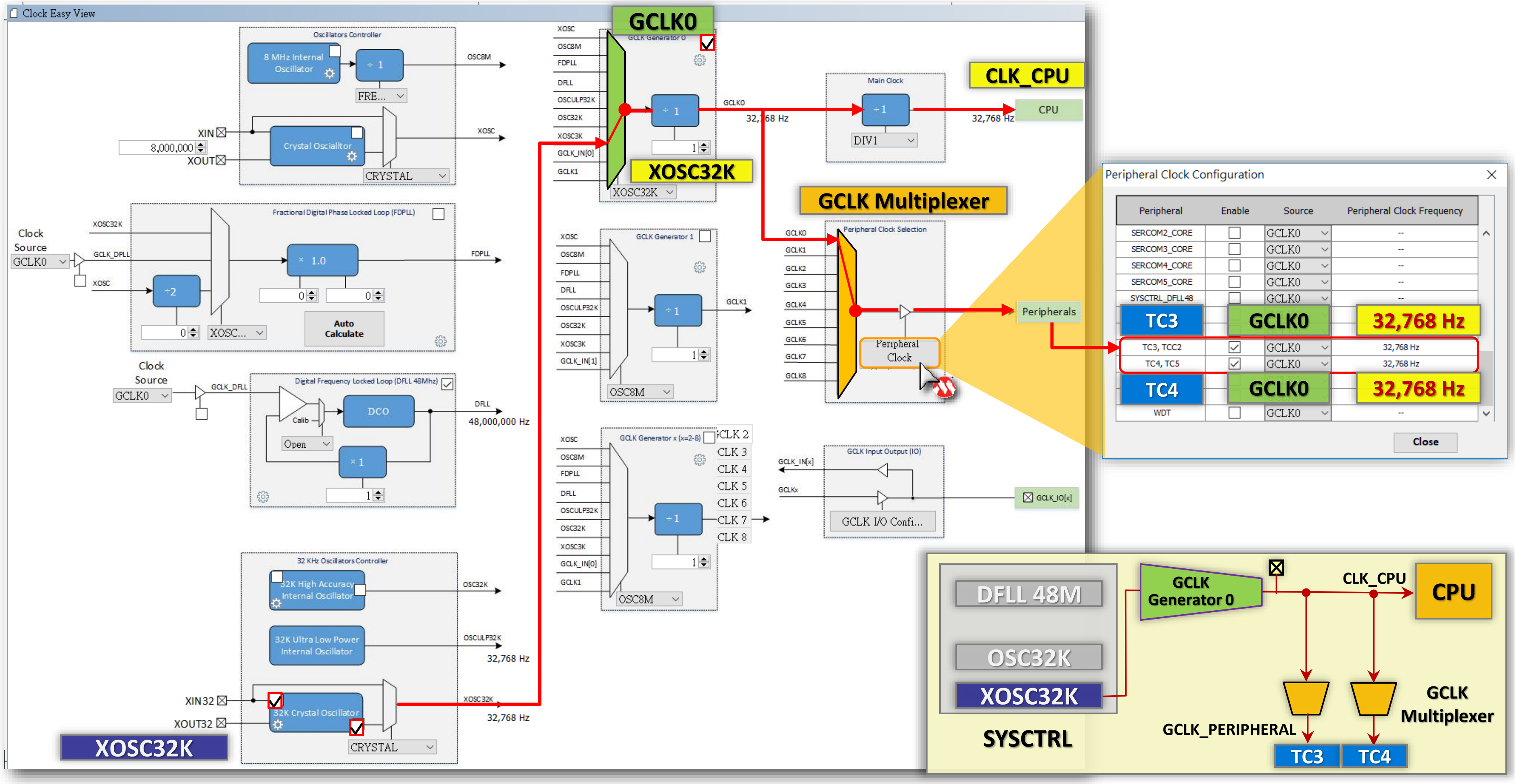
- **For Example : Internal 32k RC -> GCLK Generator 1 -> TC3**



MICROCHIP

Clock Setting Use MCC Harmony

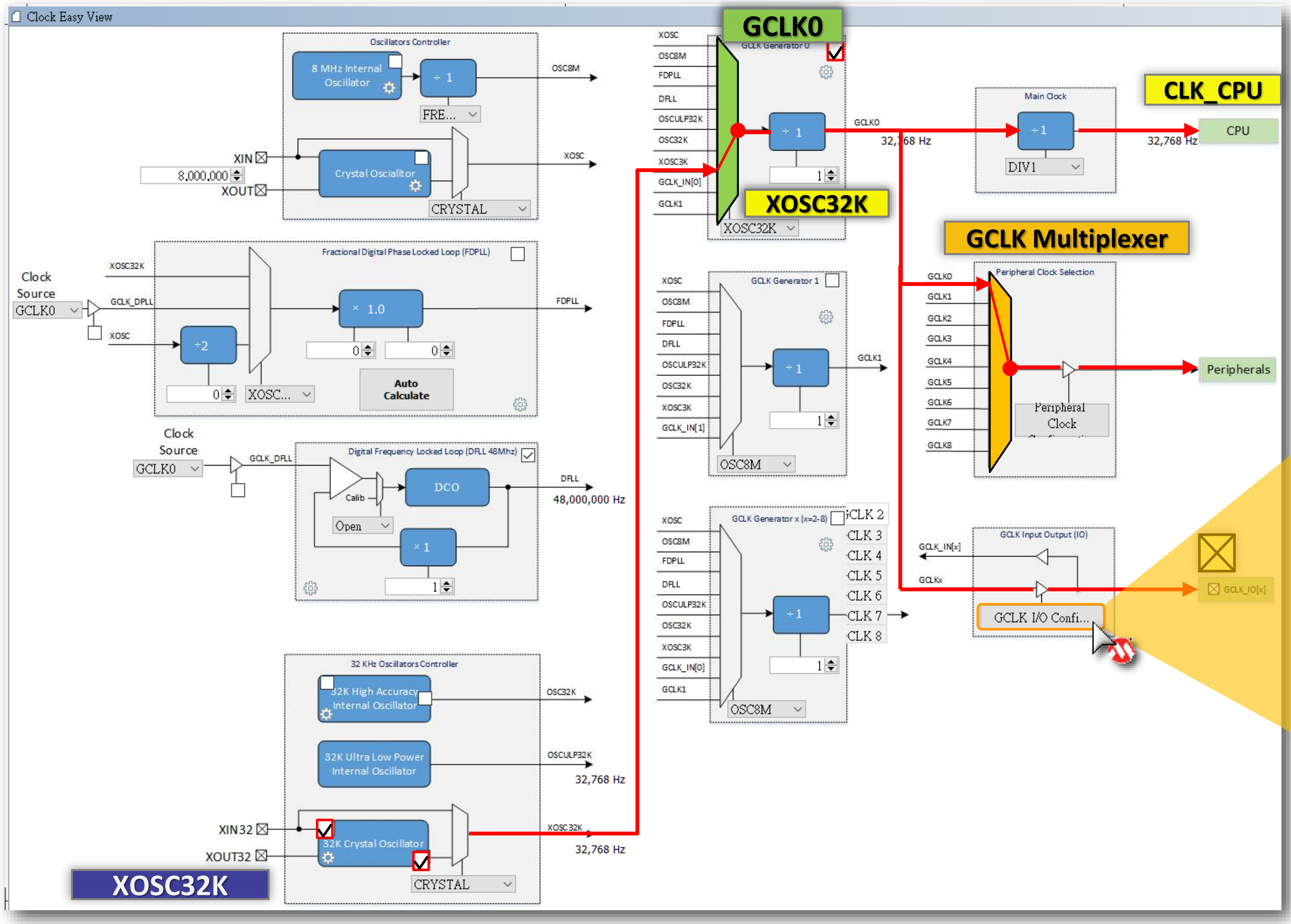
- **For Example : External 32.768K -> Generator 0 -> CPU & TC3 & TC4**

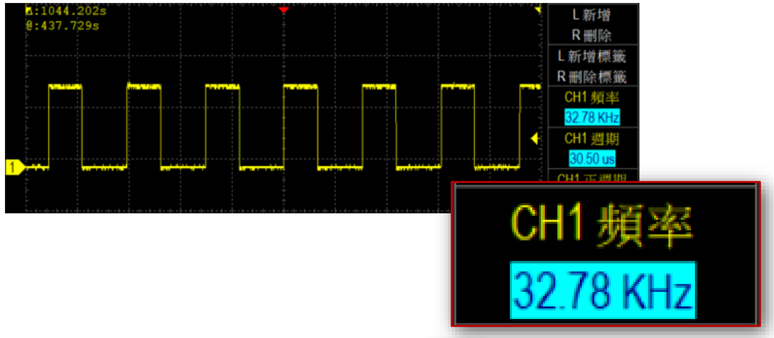


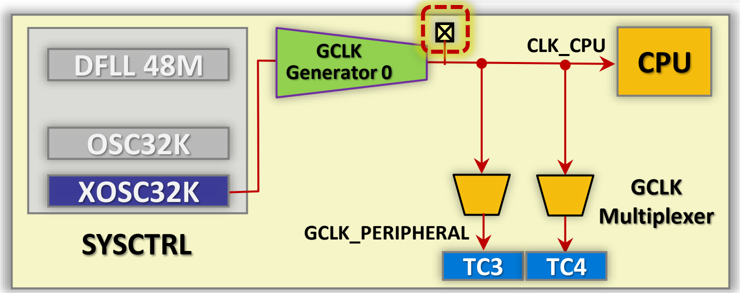
MICROCHIP

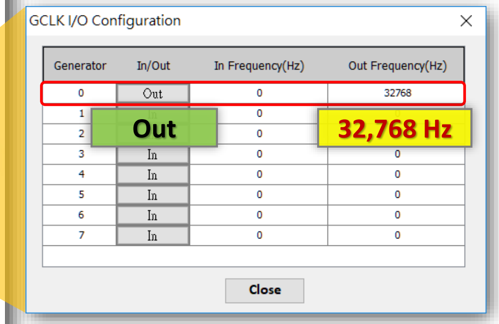
Clock Setting Use MCC Harmony

- **For Example : Configure GCLK Generator 0 as a clock output.**









Generator	In/Out	In Frequency(Hz)	Out Frequency(Hz)
0	Out	0	32768
1		0	
2		0	
3	In	0	0
4	In	0	0
5	In	0	0
6	In	0	0
7	In	0	0

Lab7 System Clock XOSC32K

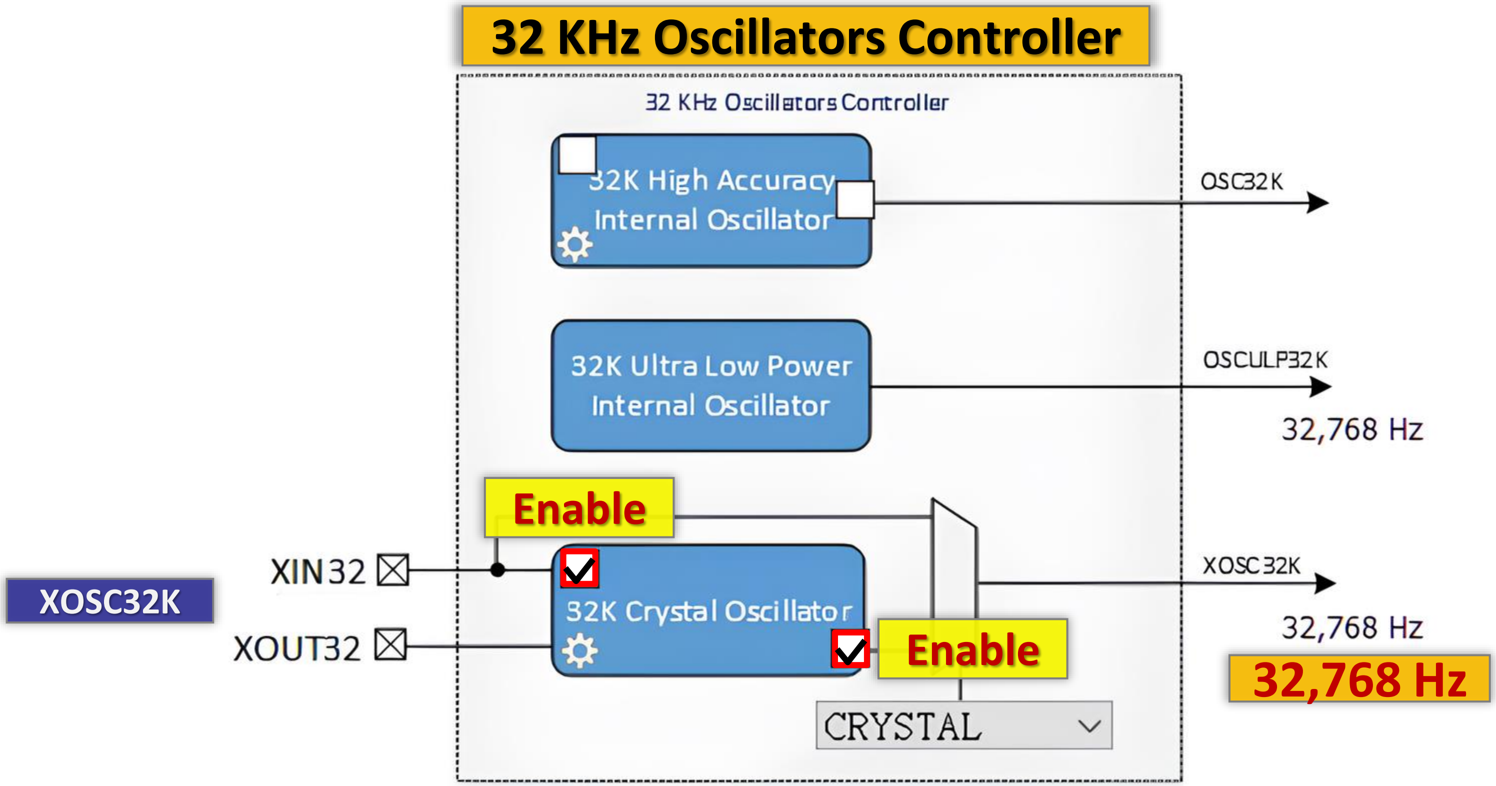
- Try to change Main Clock from default source to External 32.768K.
- Setting XOSC32K then enable it, first.
- Connect XOSC32K to Generator 0.
- Connect TC3, TC4 to Generator 0.
- Try to configure GCLK Generator 0 as a clock output.

- **Let’s go!**

Lab7 System Clock XOSC32K

Step 1

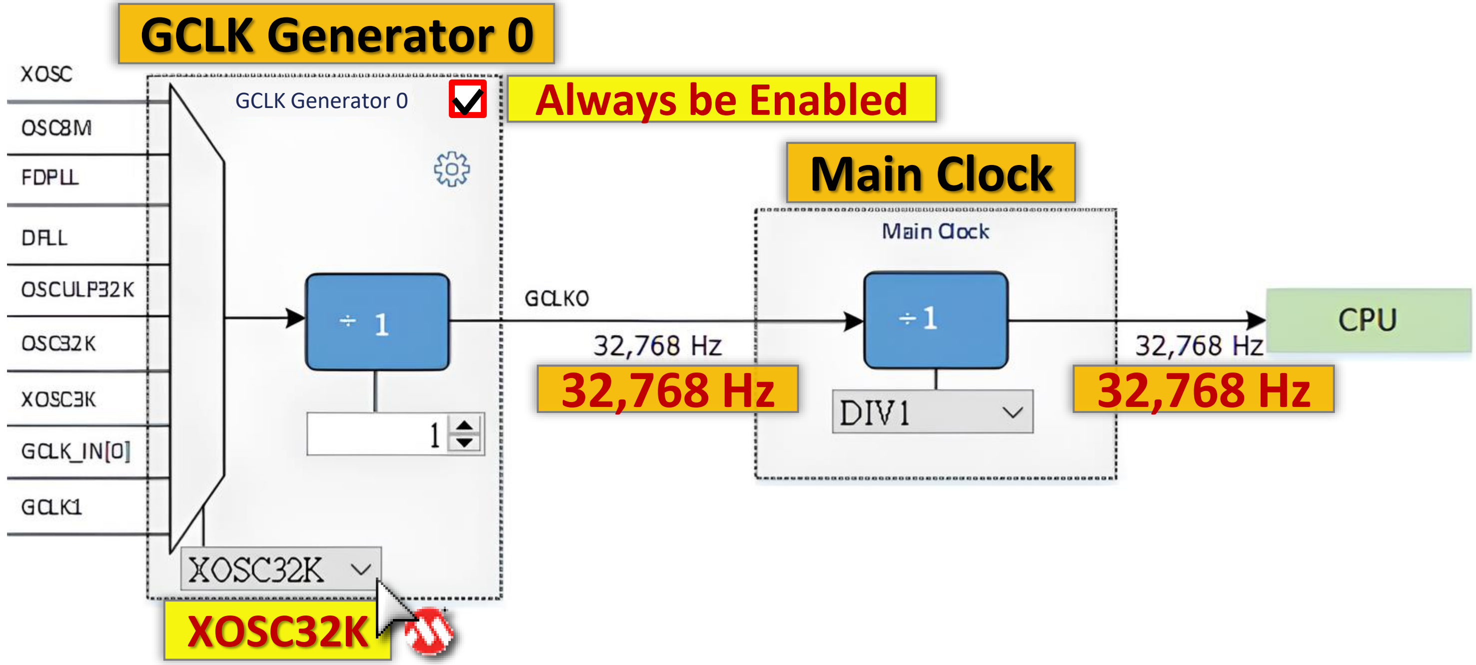
- **Enable external XOSC32K source.**



Lab7 System Clock XOSC32K

Step 2

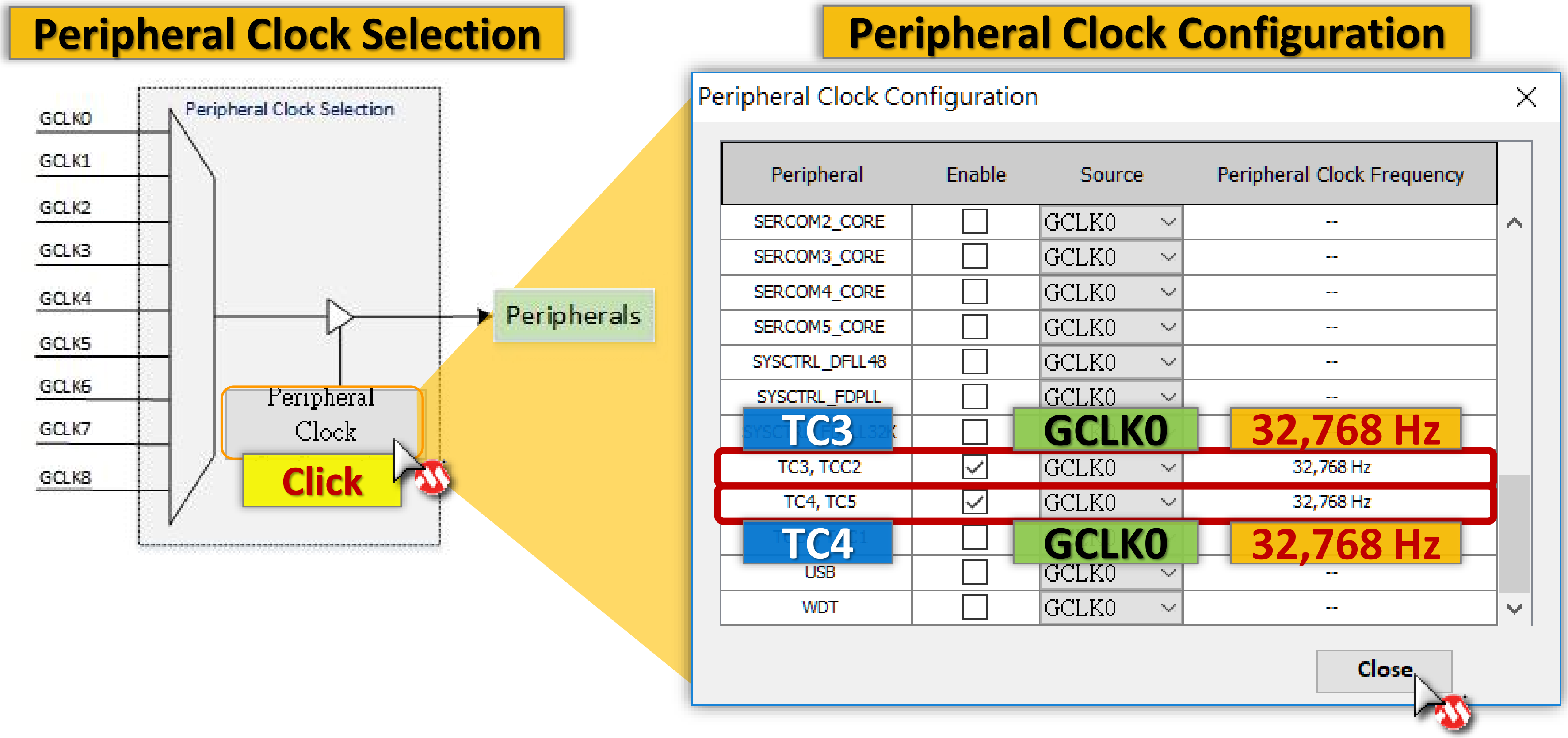
- **Configure the GCLK Generator 0 clock source to change the CPU clock.**



Lab7 System Clock XOSC32K

Step 3

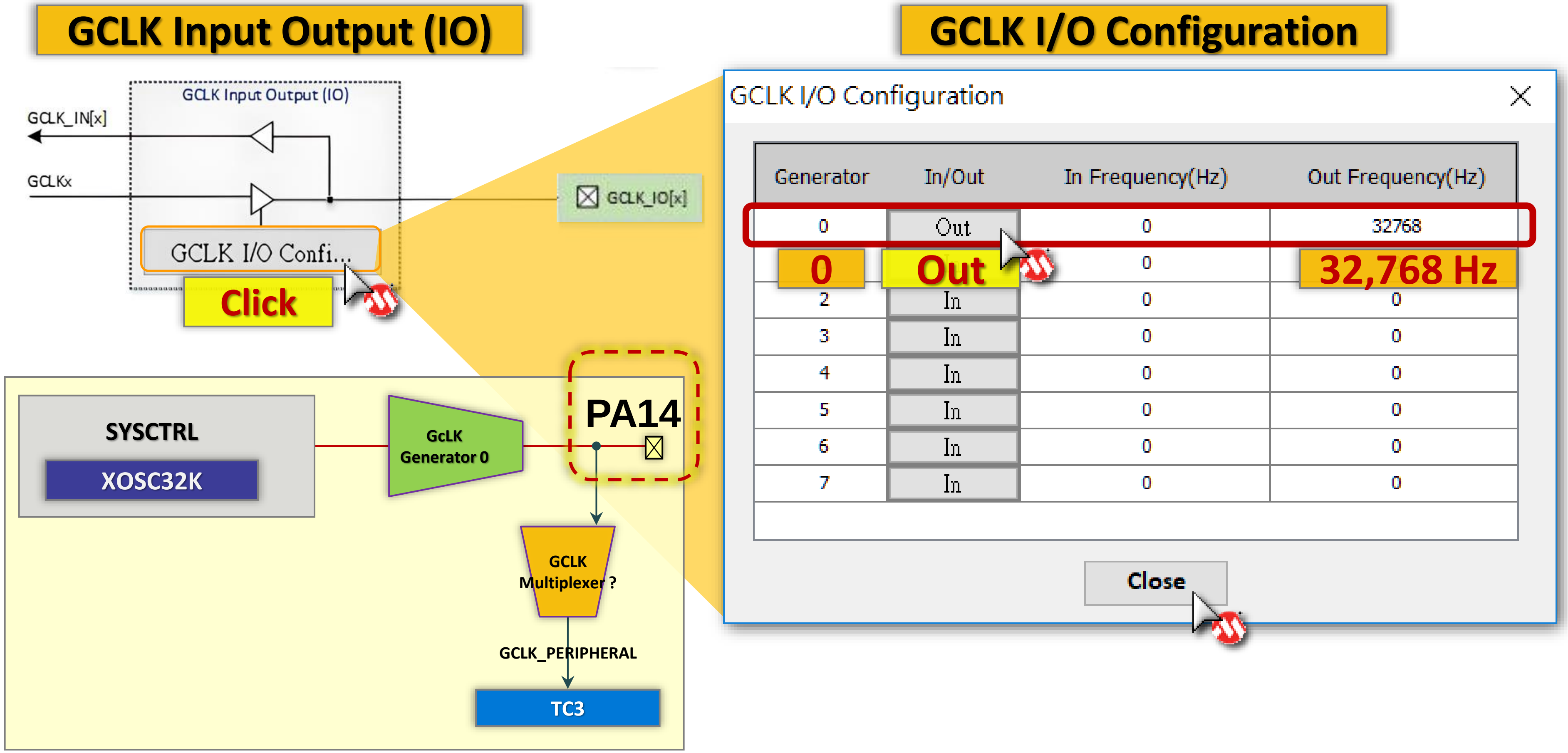
- **Observe Peripheral Clock of TC3 and TC4.**



Lab7 System Clock XOSC32K

Step 4

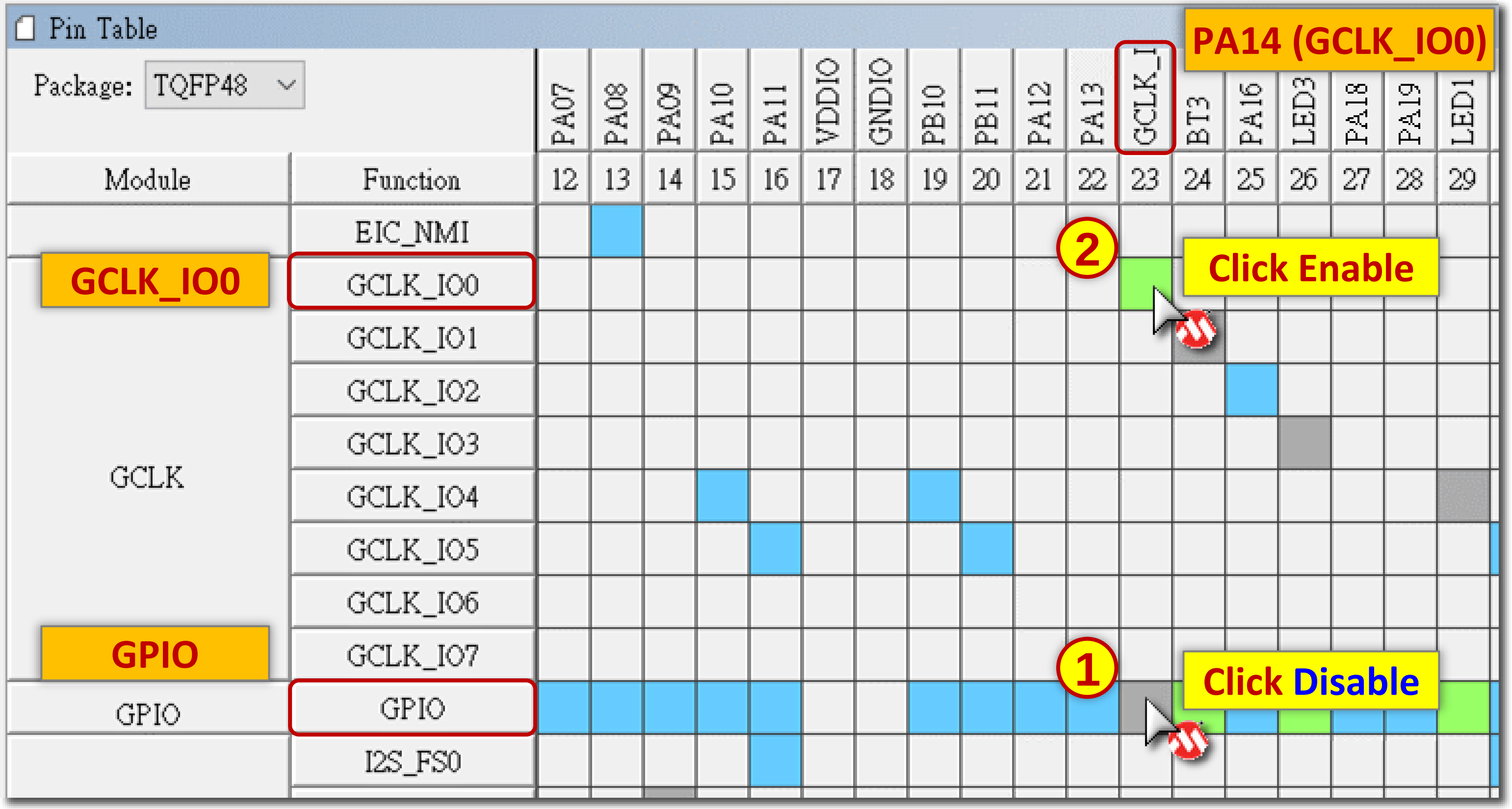
- **Try to configure GCLK Generator 0 as a clock output.**



Lab7 System Clock XOSC32K

Step 5

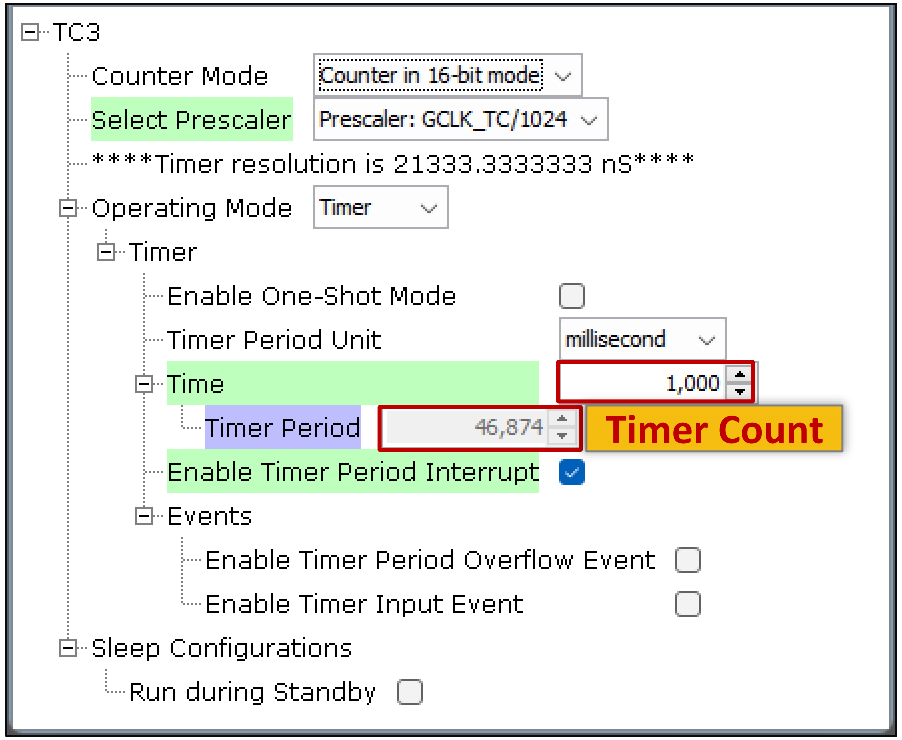
- Please disable BT1(PA14) GPIO pin function firstly and then reassign PA14 as GCLK_IO0 output pin.



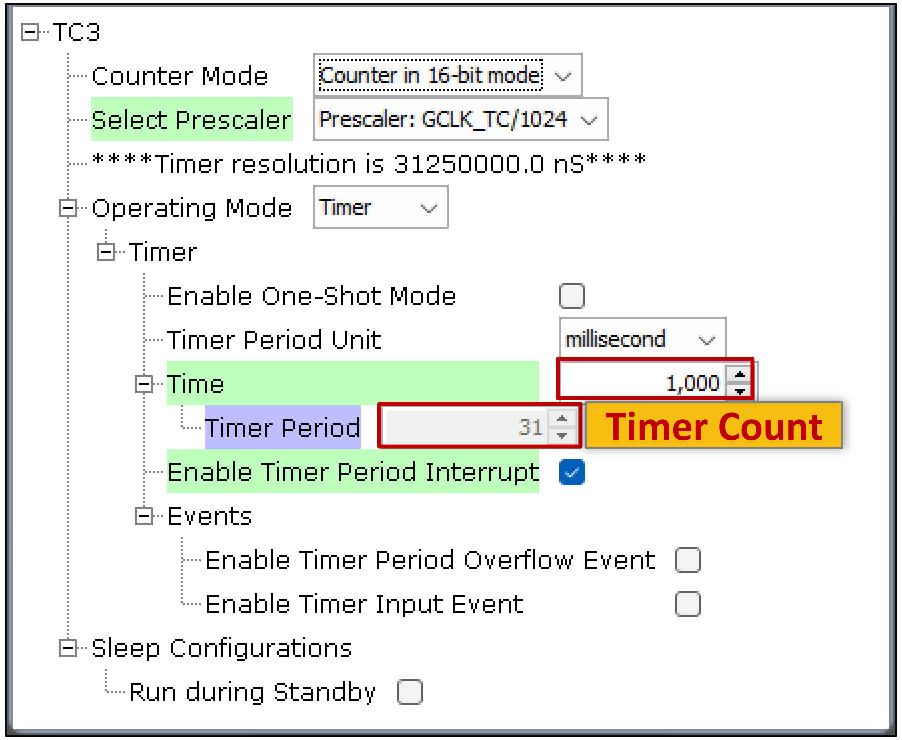
Lab7 System Clock XOSC32K

What’s changed if to change main clock from 48MHz to 32.768KHz ?

- **Let’s check TC3 configuration to see the difference after change the clock source.**



Clock Source : DFLL48M



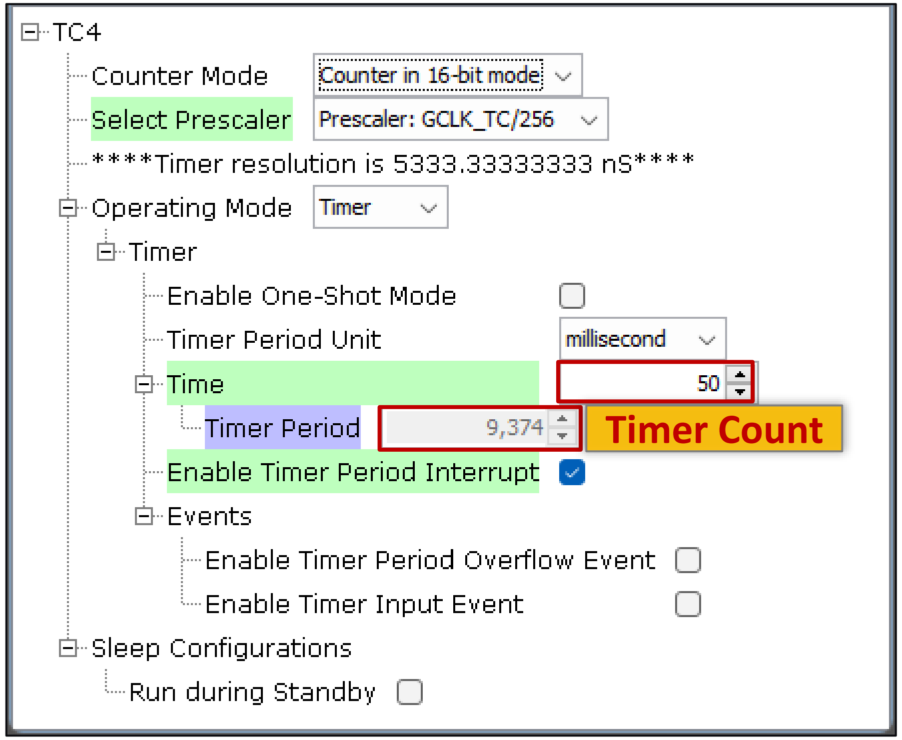
Clock Source : XOSC32K

MICROCHIP

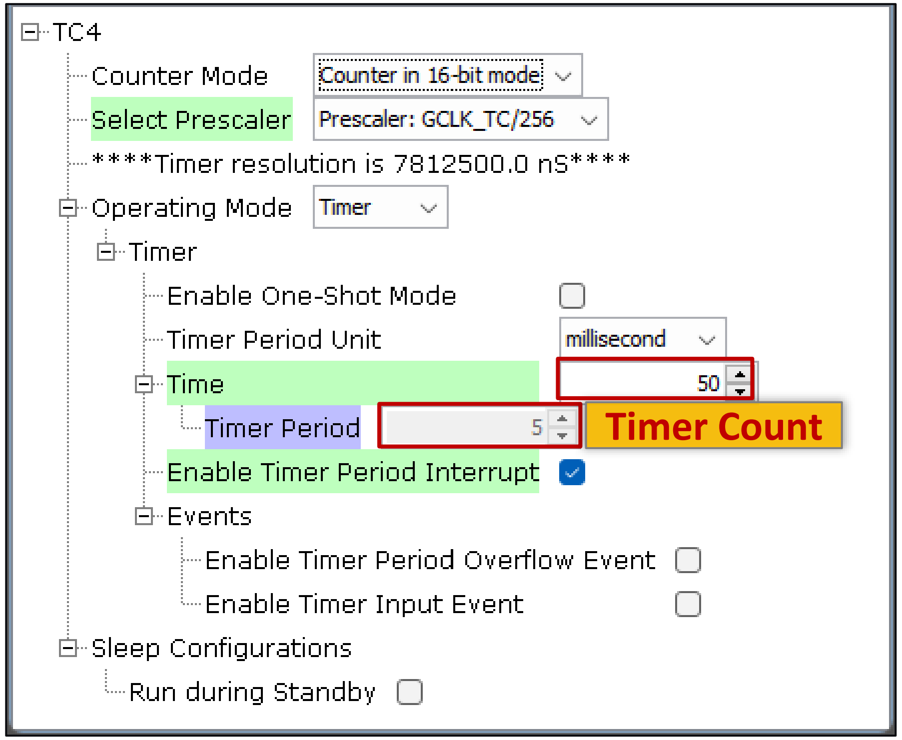
Lab7 System Clock XOSC32K

What's changed if to change main clock from 48MHz to 32.768KHz ?

- **Let's check TC4 configuration to see the difference after change the clock source.**



Clock Source : DFLL48M

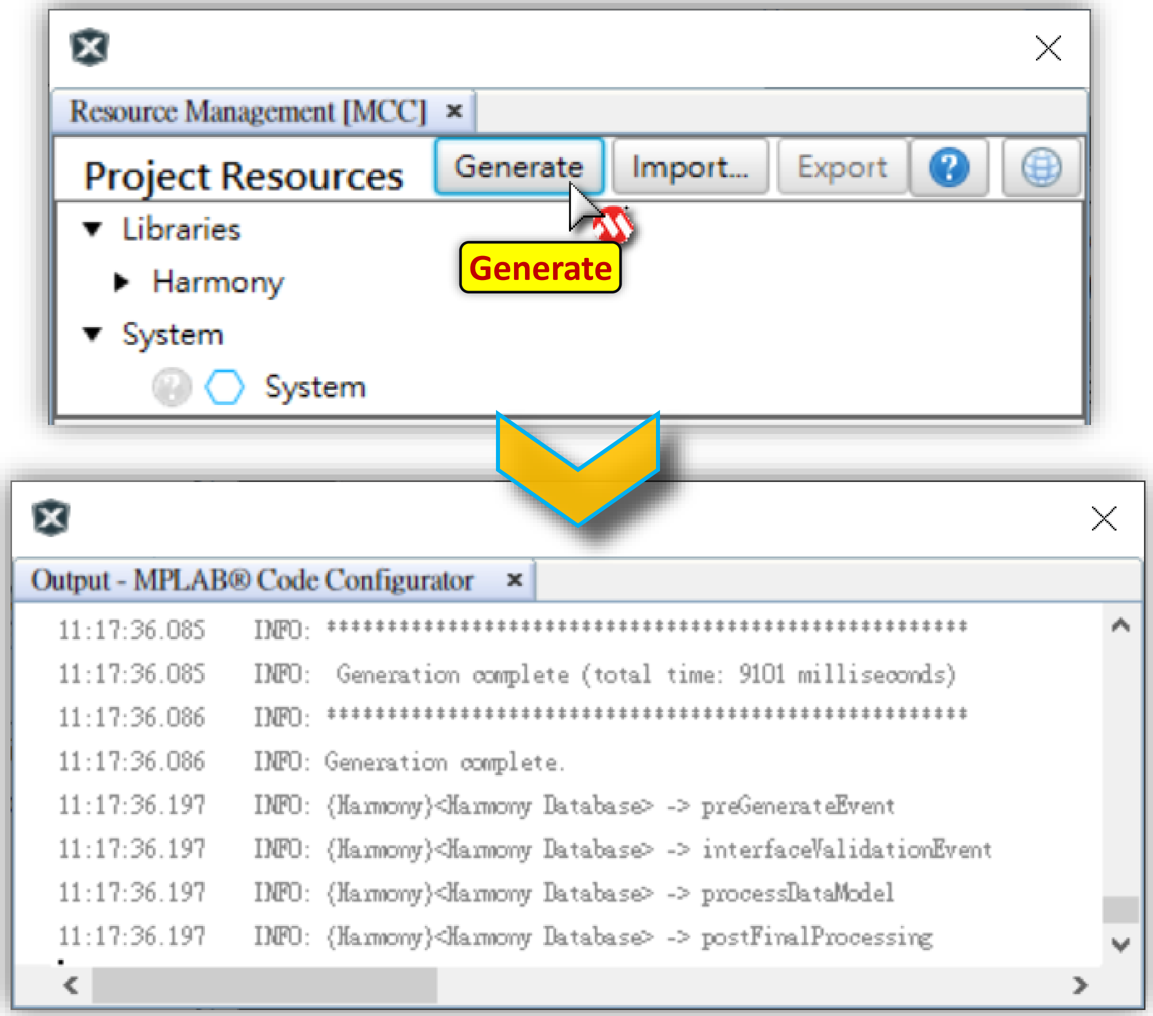


Clock Source : XOSC32K

Lab7 System Clock XOSC32K

Step 6

- Click "Generate" Button to Generate your Code.



Lab7 System Clock XOSC32K

Step 7

a **Comment out BT related GPIO code segment.**

```
…

int main (void)
{
    SYS_Initialize ( NULL );

   while( true )
   {
       …

       // TODO 7.01
       //   if ( BT1_Get() ) LED3_Clear();
       //   else              LED3_Set();
   }
}
```

Flash Access Time and Wait State

- The SAMD21 main flash Random Read time around 40nS. That means **you must adjust Wait State to proper value if main clock over than 24MHz**. The actual value shows in below, it will depend on system supply voltage.

V_{DD} range	NVM Wait States	Maximum Operating Frequency	Units
1.62V to 2.7V	0	14	MHz
	1	28	
	2	42	
	3	48	
2.7V to 3.63V	0	24	
	1	48	

CPU_CLK = 32.768KHz

VDD = 3.3V

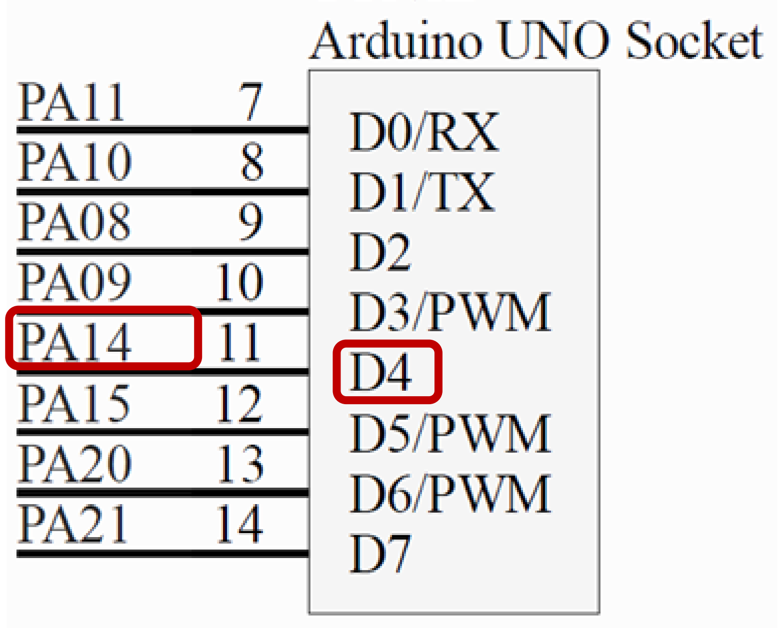
MCC Harmony will auto adjust proper NVM Wait State.
(In case of 3.3V voltage supply only).

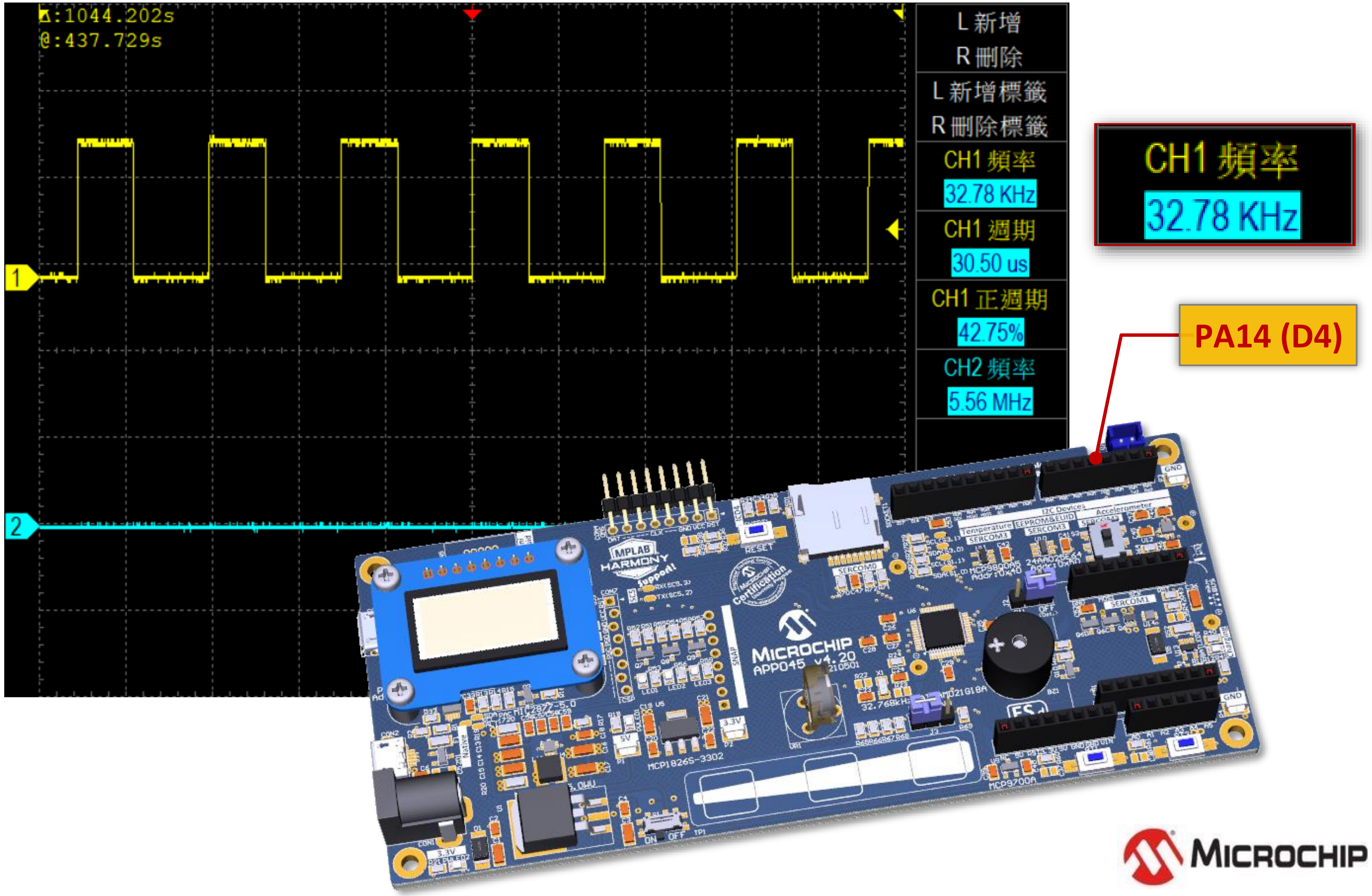
plib_nvmctrl.c

```
void NVMCTRL_Initialize(void)
{
   NVMCTRL_REGS->NVMCTRL_CTRLB =
         NVMCTRL_CTRLB_READMODE_NO_MISS_PENALTY |
         NVMCTRL_CTRLB_SLEEPPRM_WAKEONACCESS |
         NVMCTRL_CTRLB_RWS(0) |
         NVMCTRL_CTRLB_MANW_Msk;
}
```

Lab7 System Clock XOSC32K

Try to observe the clock output



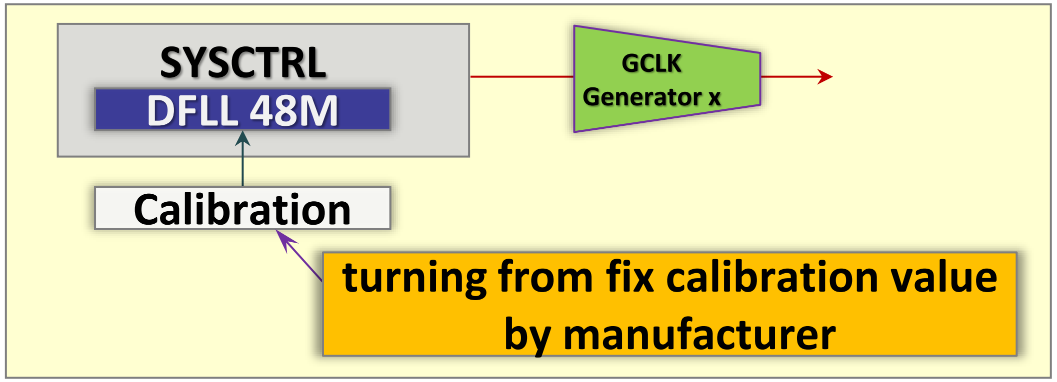


Clock source from DFLL48M

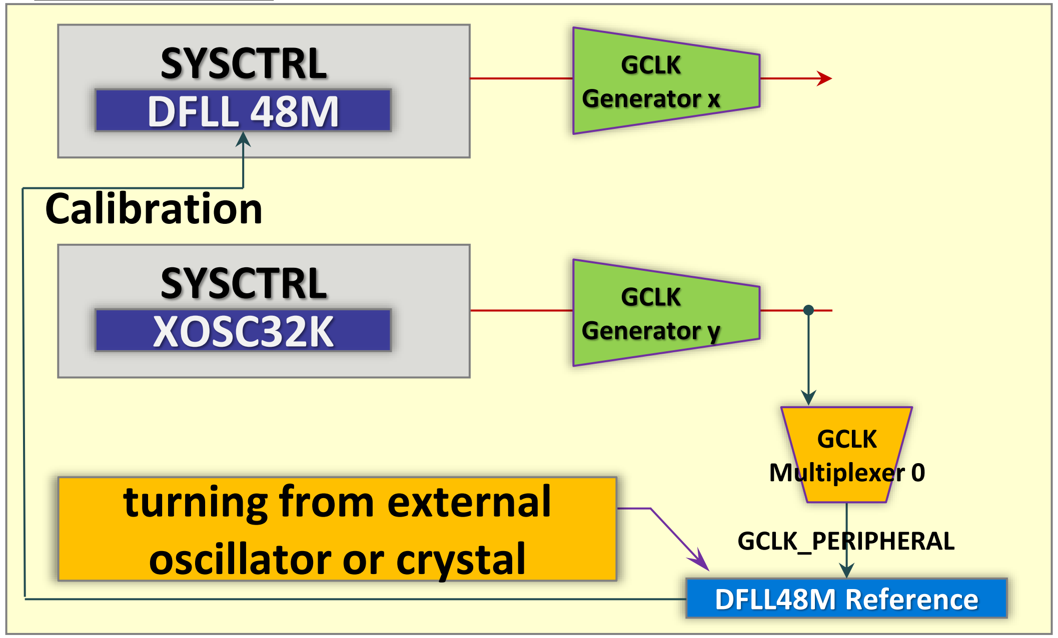
- There are 2 different mode for DFLL48M clock source
  - Open Loop : Project Default Use
    Clock turning from fix calibration value by manufacturer.

  - Close Loop :
    Clock turning from external oscillator or crystal.

Open Loop



Close Loop



Lab8 System Clock DFLL48M

Close Loop

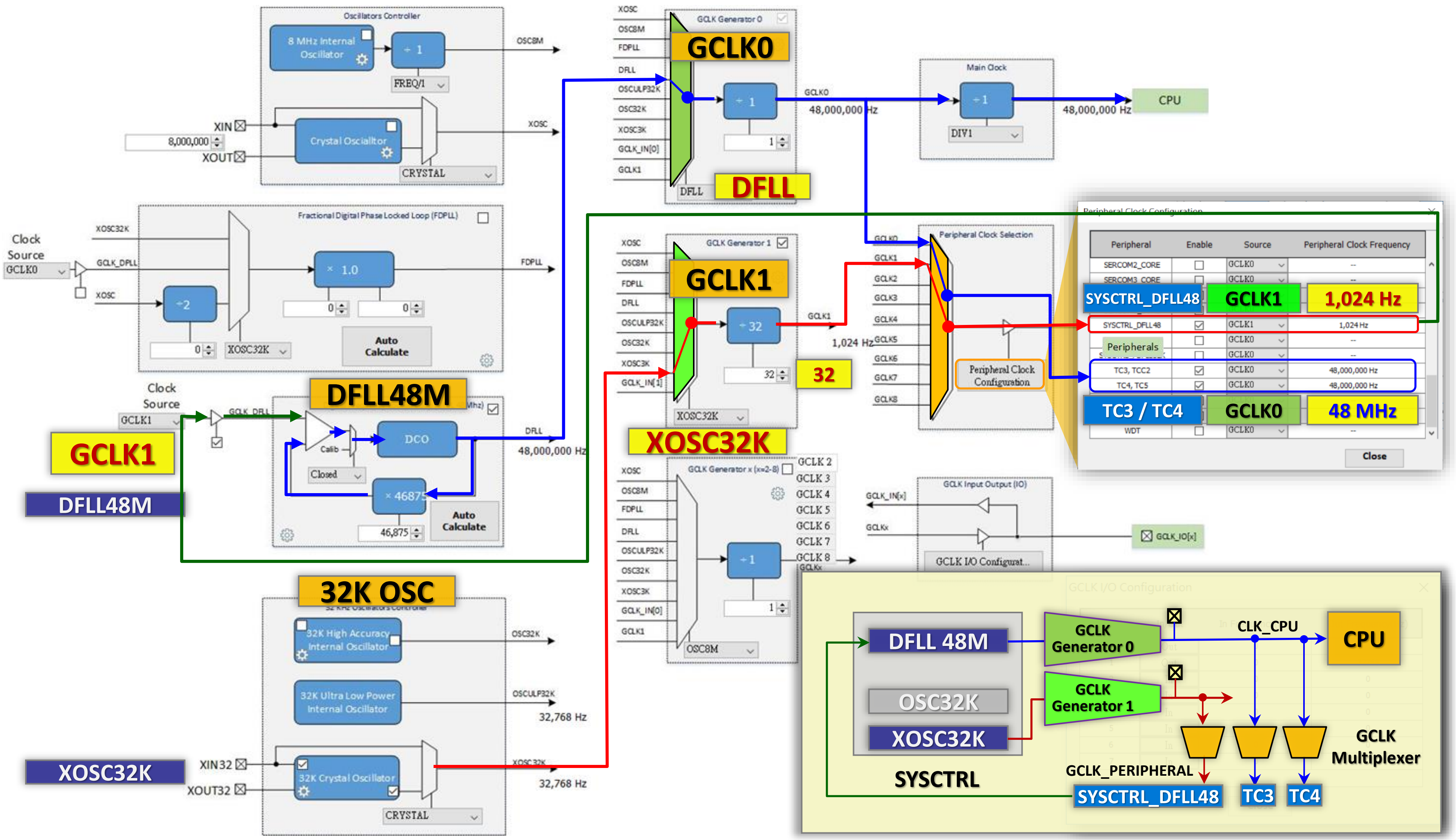
Configure DFLL48M (Close loop) as source of main clock (GCLK0).

- Enable XOSC32K clock source.
- Enable GCLK Generator 1 and assign XOSC32K as source of it.
- Enable DFLL48M and assign GCLK1 as source or it.
- Select Close Loop calibration of DFLL48M.
- Click Auto Calculate to generate multiplier factor of DFLL48M.
- Assign DFLL48M as clock source of GCLK Generator 0.
- CPU wait state will auto judge by MCC Harmony.

- **Let's go!**

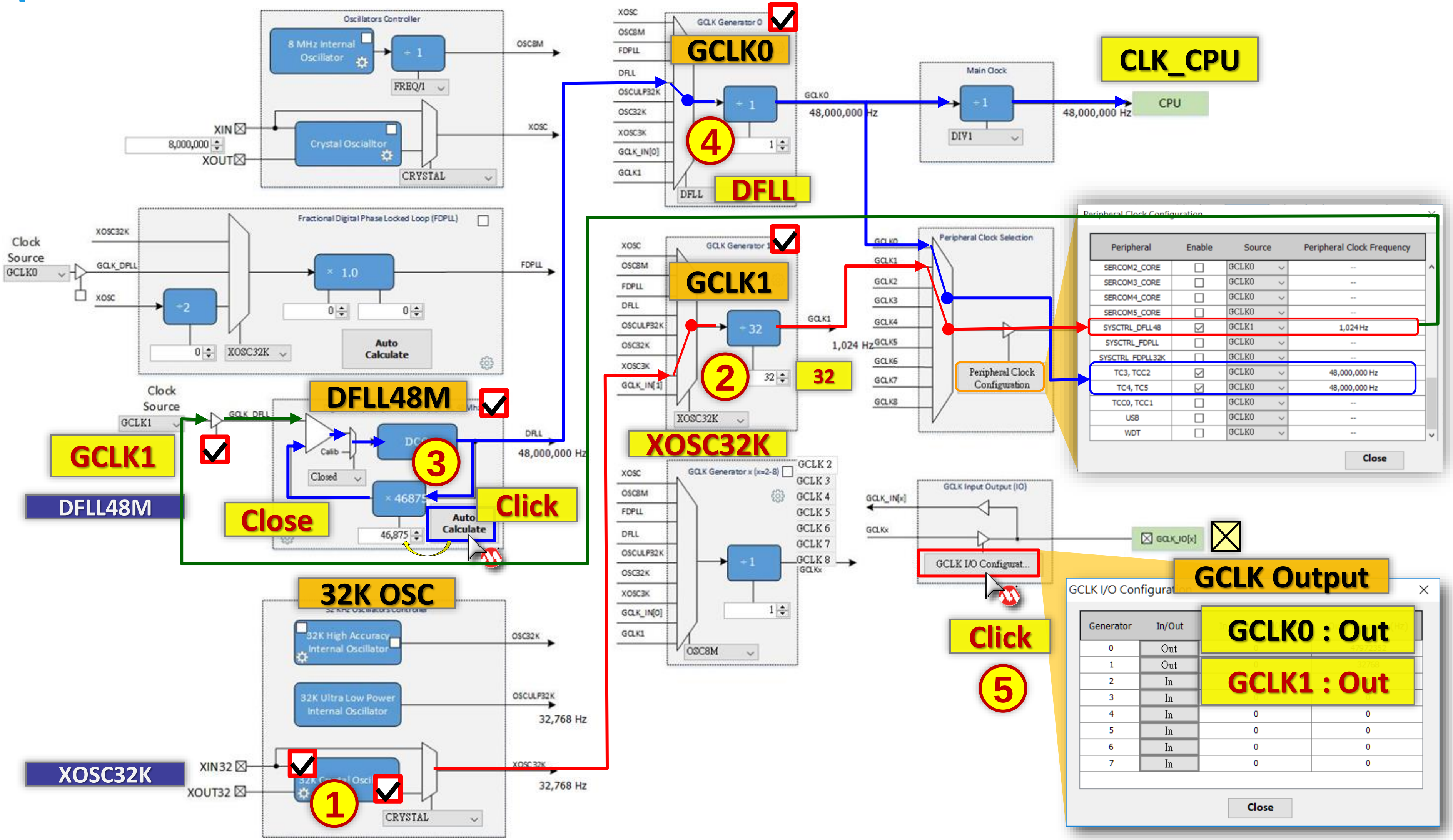
Lab8 System Clock DFLL48M

Overview



Lab8 System Clock DFLL48M

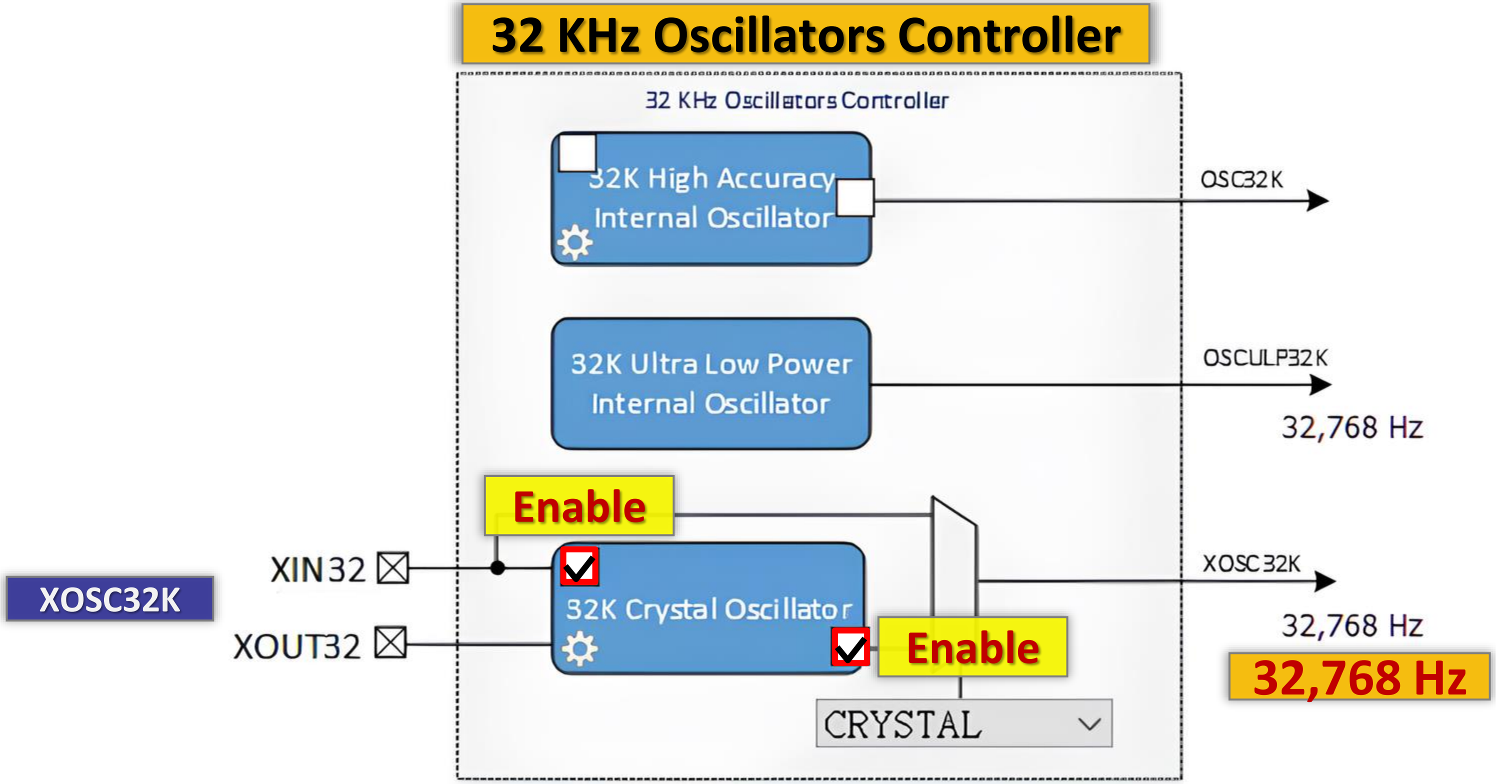
Setup Steps Overview



Lab8 System Clock DFLL48M

Step 1

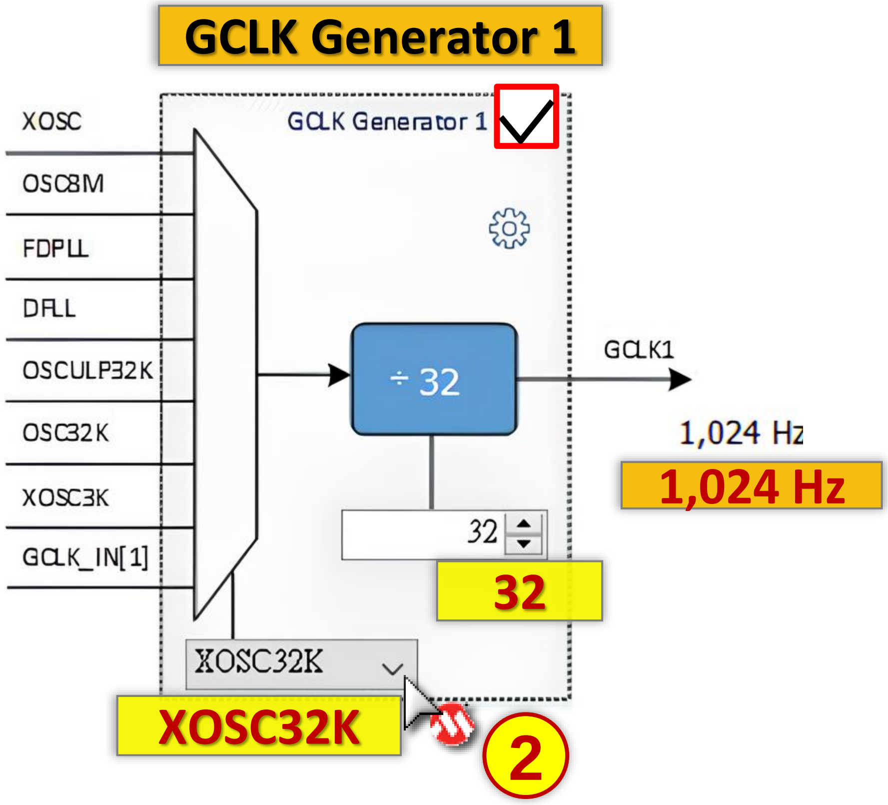
- **Enable external XOSC32K source.**



Lab8 System Clock DFLL48M

Step 2

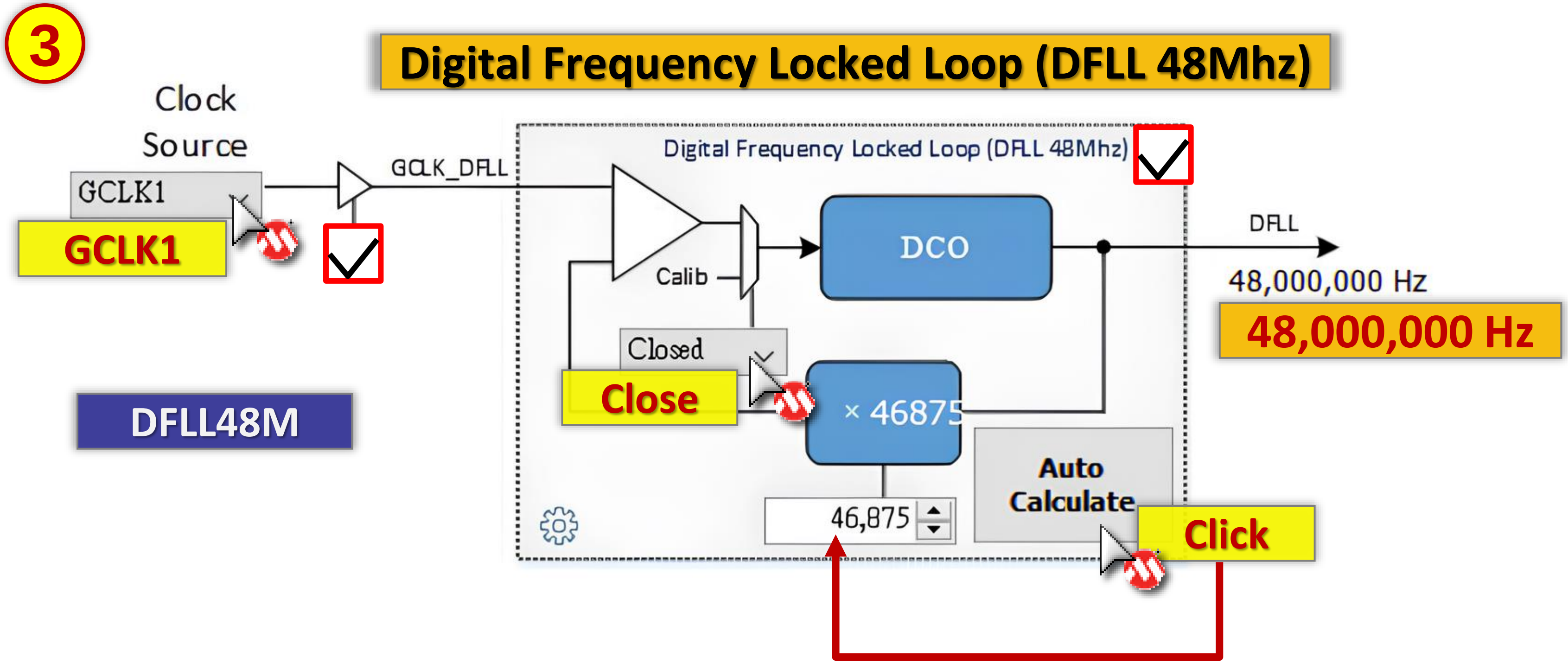
- **Configure the GCLK Generator 1 clock source and divider to generate 1024 Hz.**



Lab8 System Clock DFLL48M

Step 3

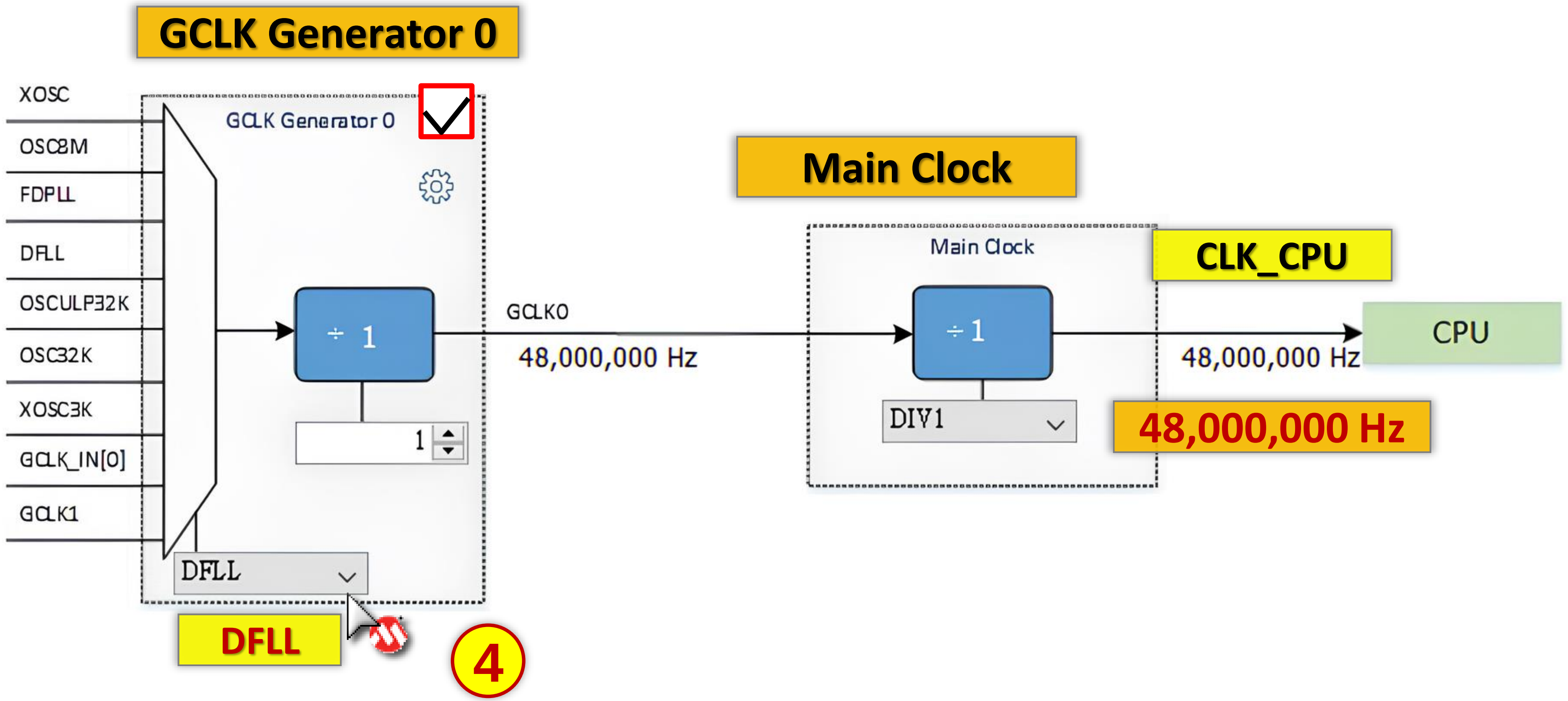
- **Configure DFLL48M (Close loop) and generate 48 MHz.**



Lab8 System Clock DFLL48M

Step 4

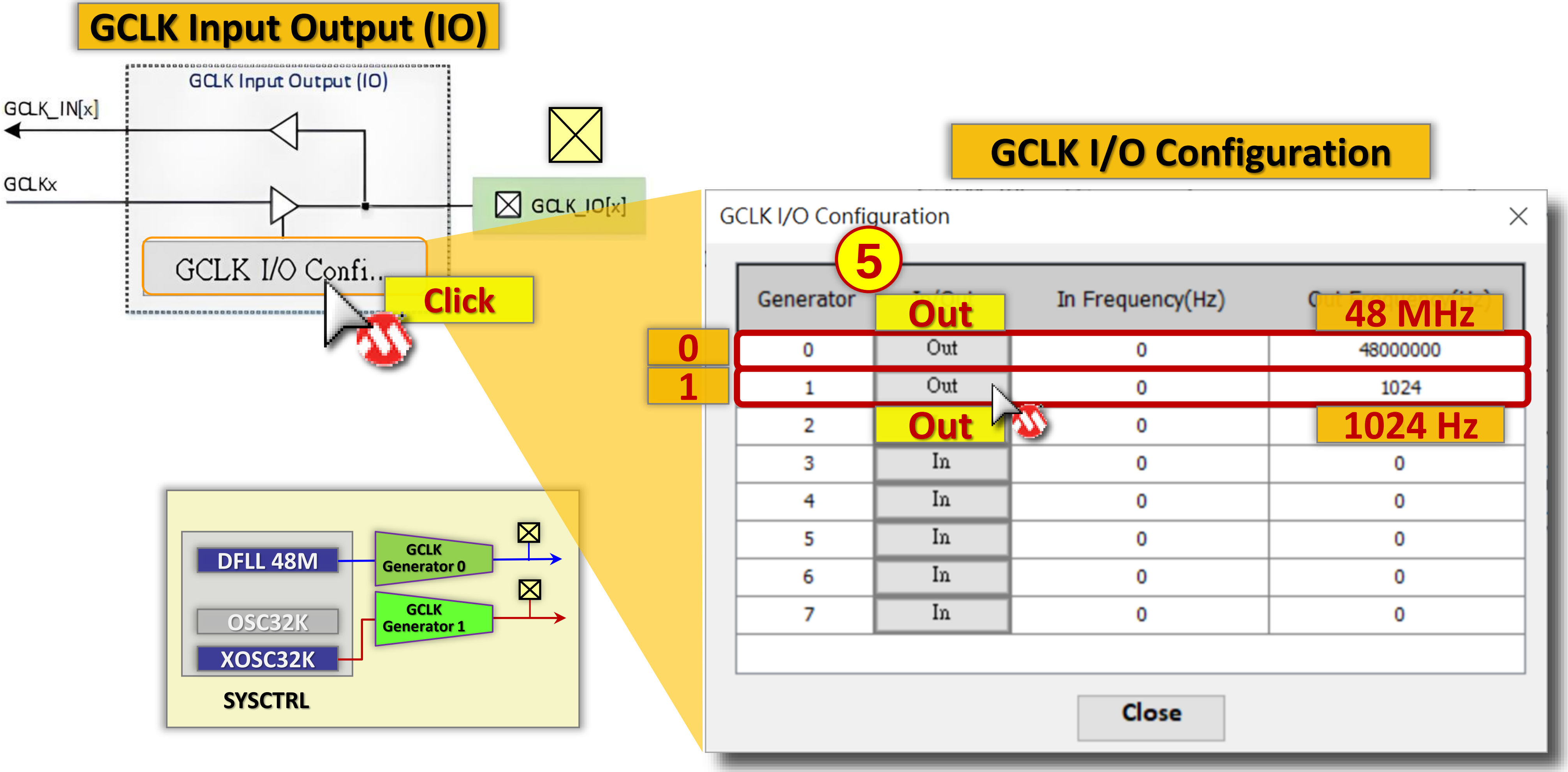
- **Configure the GCLK Generator 0 clock source to change the CPU clock to 48 MHz.**



Lab8 System Clock DFLL48M

Step 5

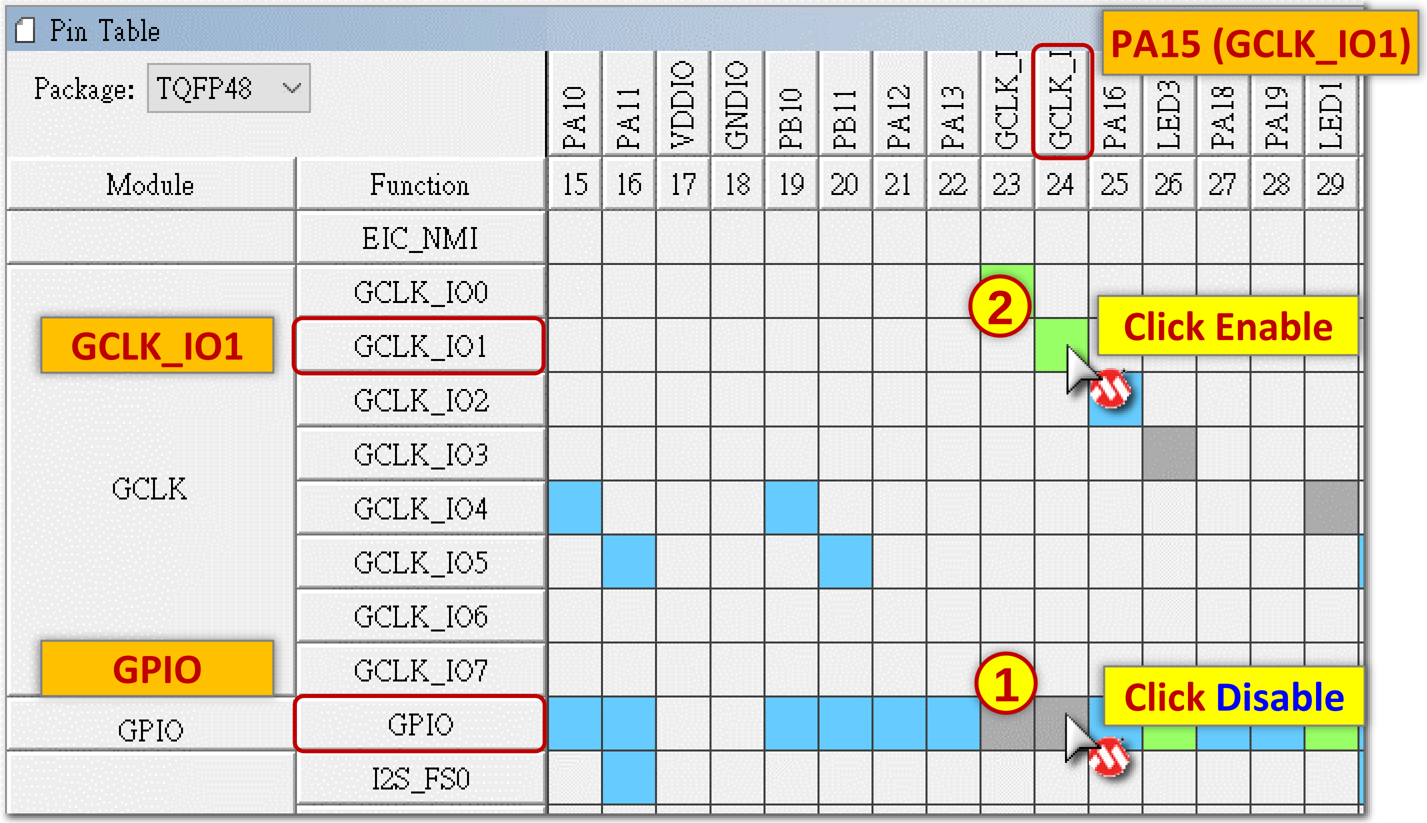
- Try to configure GCLK Generator 0 and Generator 1 as a clock output.



Lab8 System Clock DFLL48M

Step 6

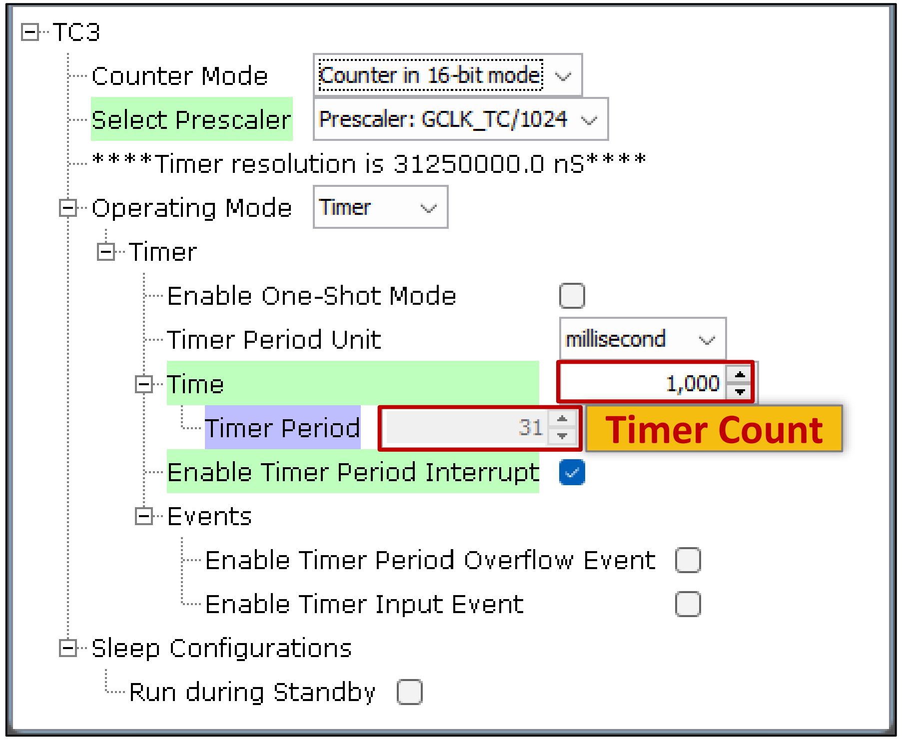
- Please disable GPIO PA15 pin function firstly and then reassign PA15 as GCLK_IO1 output pin.



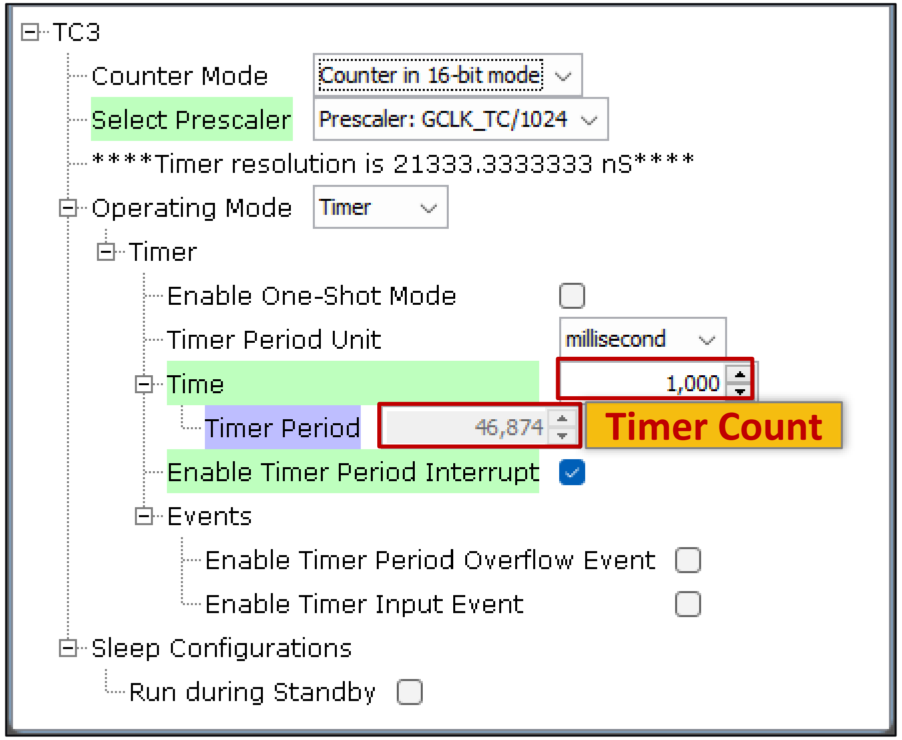
Lab8 System Clock DFLL48M

What’s changed if to change main clock from 32.768KHz to DFLL Close Loop ?

- Let’s check **TC3** configuration to see the difference after change the clock source.



Clock Source : XOSC32K

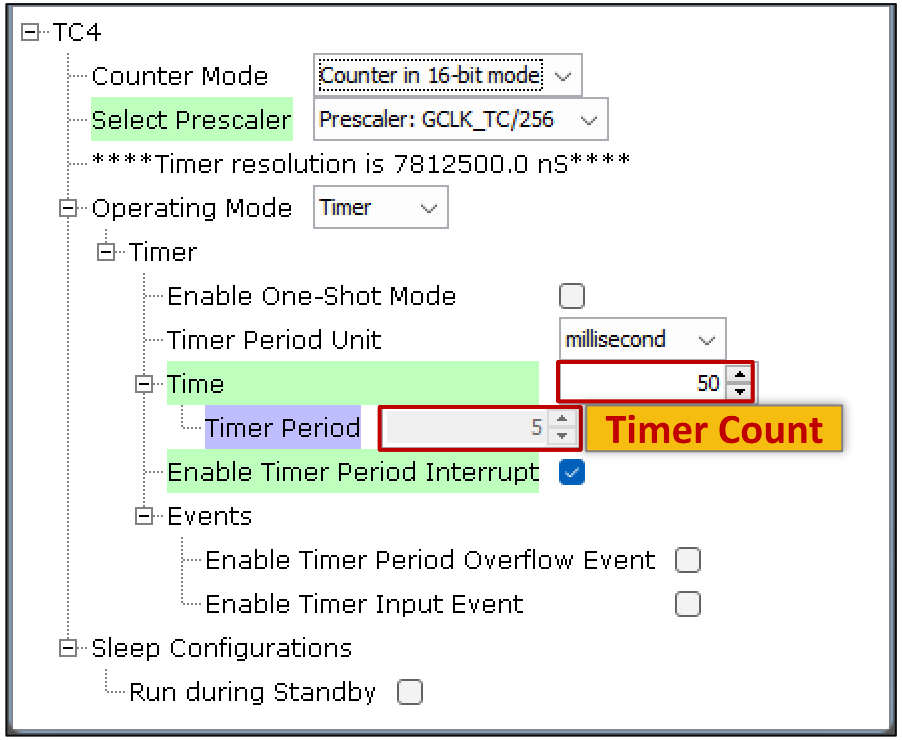


Clock Source : DFLL48M

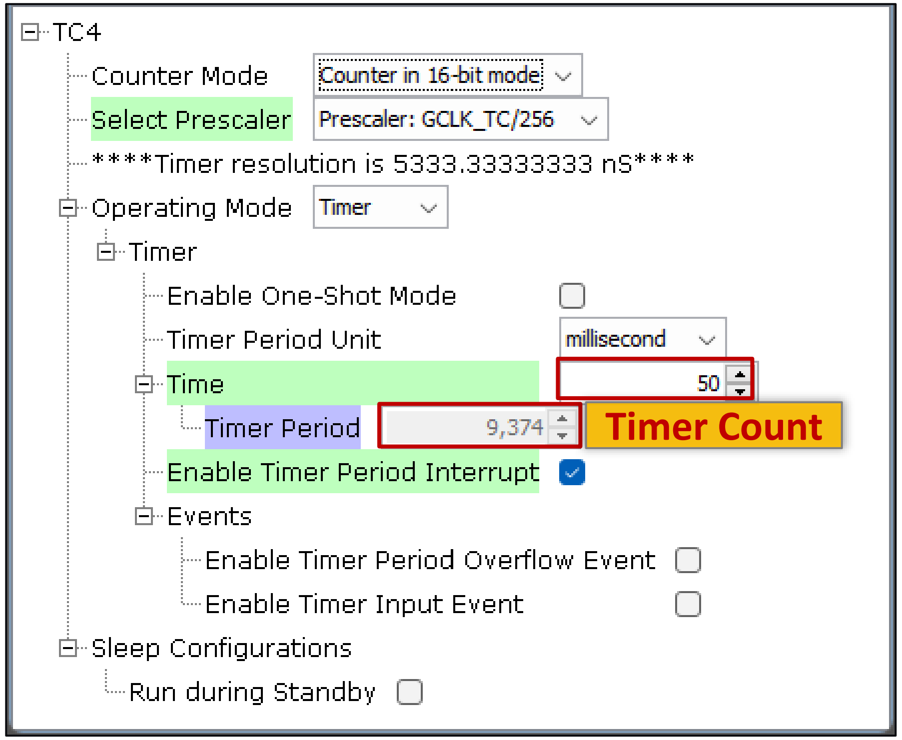
Lab8 System Clock DFLL48M

What's changed if to change main clock from 32.768KHz to DFLL Close Loop ?

- **Let's check TC4 configuration to see the difference after change the clock source.**



Clock Source : XOSC32K

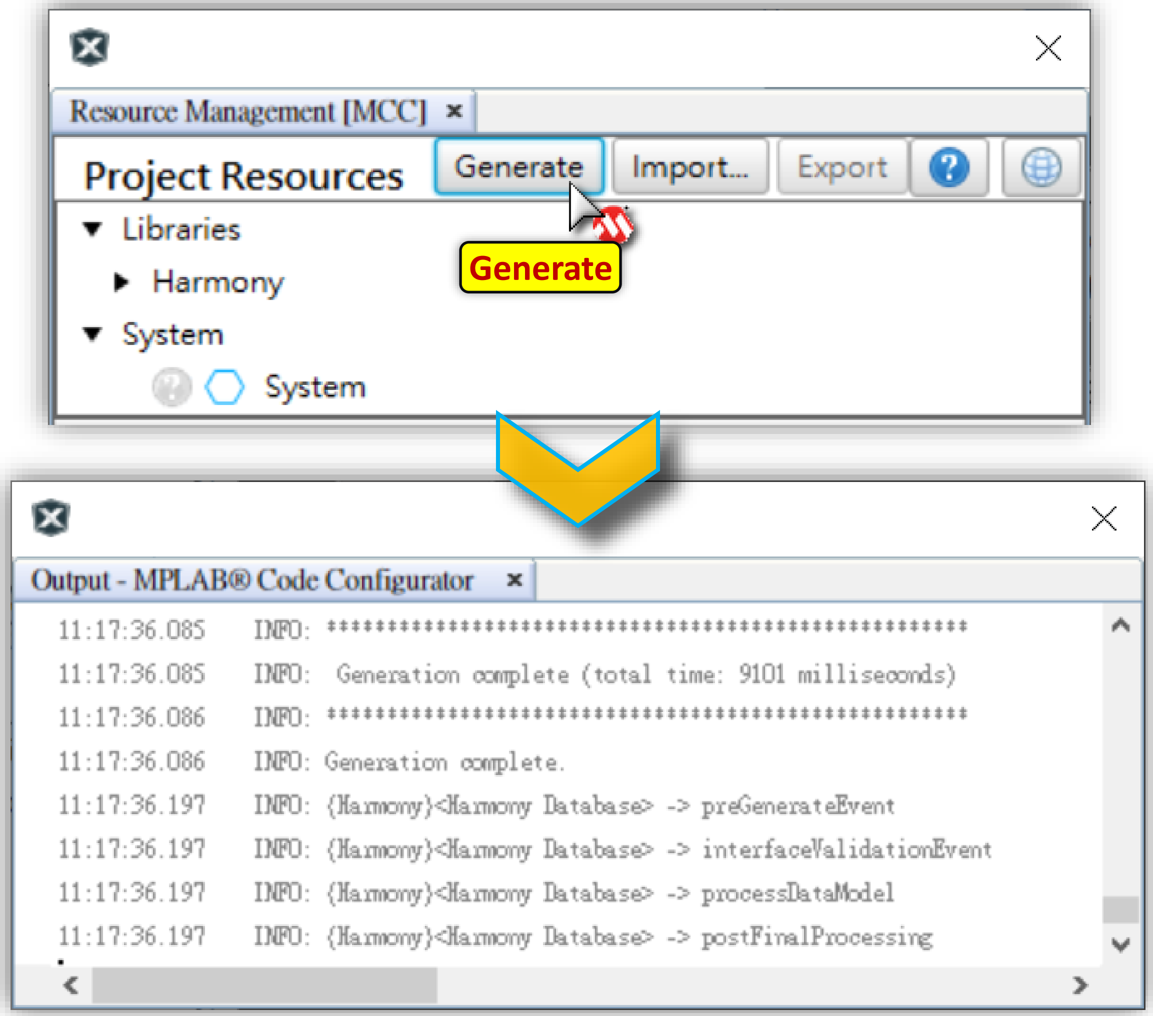


Clock Source : DFLL48M

Lab8 System Clock DFLL48M

Step 7

- Click "Generate" Button to Generate your Code.



Lab8 System Clock DFLL48M

Step 8

a Compiler and programming to your EVB to check the result.

```
...

int main (void)
{
    SYS_Initialize ( NULL );

    wh
    {
        ...

        // TODO 7.01
        // if ( BT2_Get() ) LED3_Clear();
        // else              LED3_Set();
    }
}
```

No code need to change or modify.

Flash Access Time and Wait State

- MCC Harmony will auto adjust proper NVM Wait State, also.

```
plib_nvmctrl.c
void NVMCTRL_Initialize(void)
{
   NVMCTRL_REGS->NVMCTRL_CTRLB =
          NVMCTRL_CTRLB_READMODE_NO_MISS_PENALTY |
          NVMCTRL_CTRLB_SLEEPPRM_WAKEONACCESS |
          NVMCTRL_CTRLB_RWS(1) |
          NVMCTRL_CTRLB_MANW_Msk;
}
```

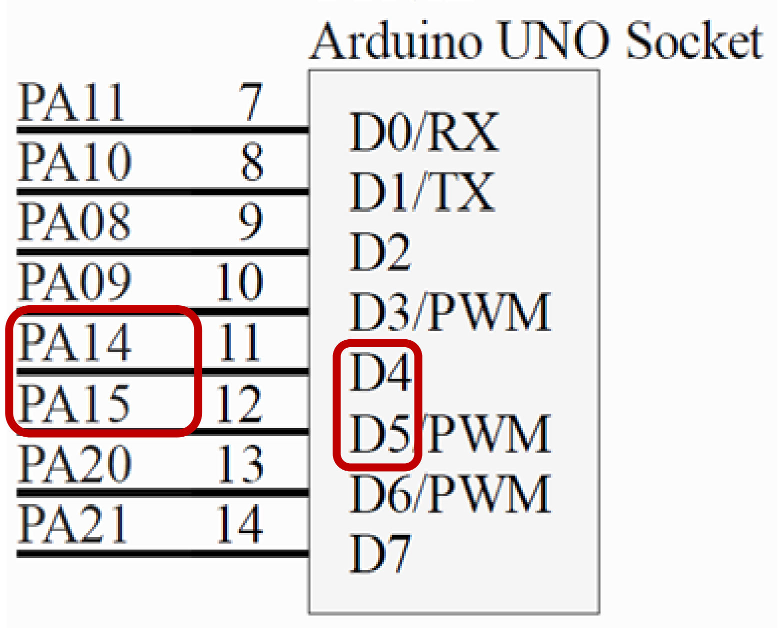
V_{DD} range	NVM Wait States	Maximum Operating Frequency	Units
1.62V to 2.7V	0	14	MHz
	1	28	
	2	42	
	3	48	
2.7V to 3.63V	0	24	
	1	48	

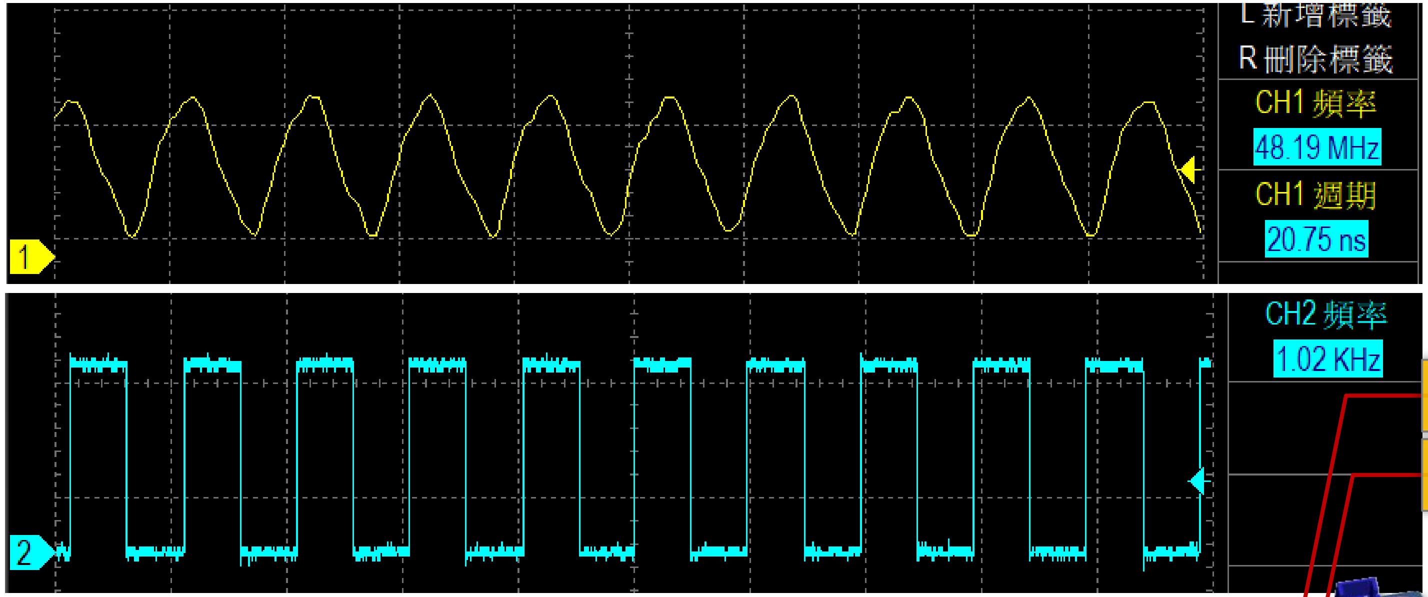
VDD = 3.3V

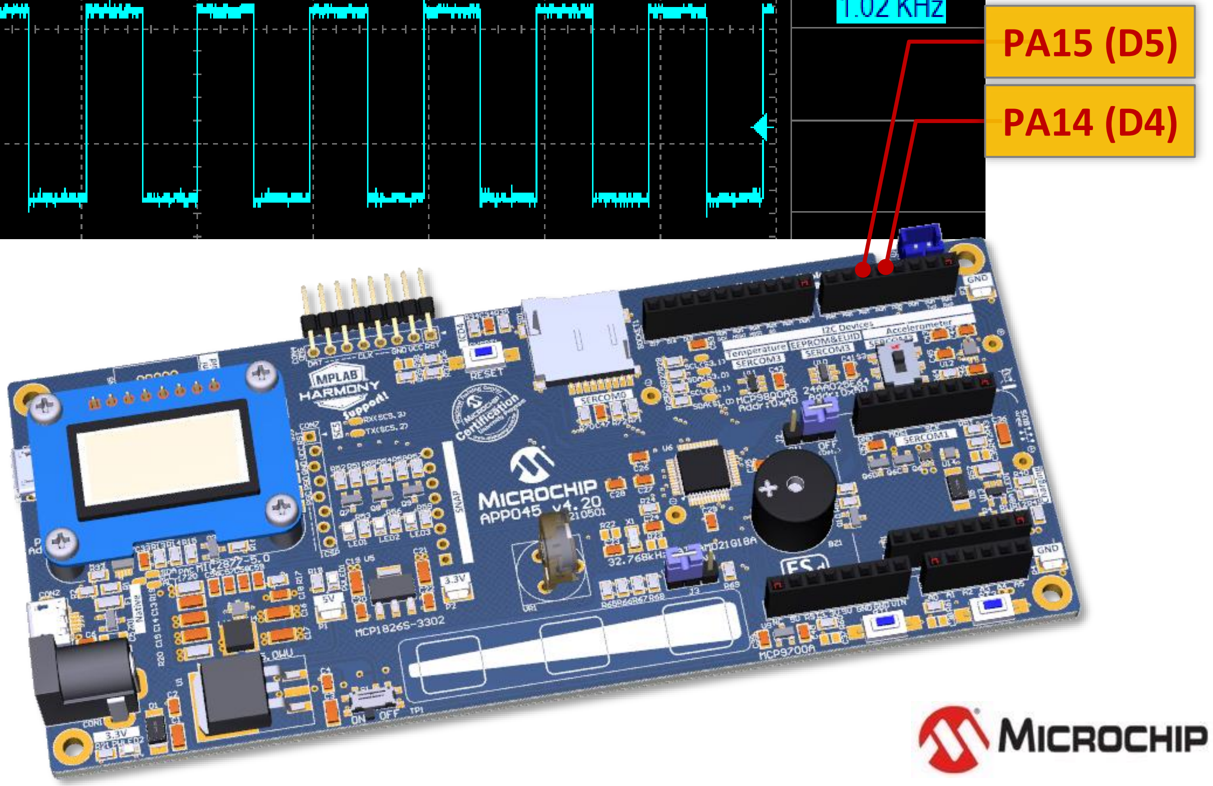
CPU_CLK = 48MHz

Lab8 System Clock DFLL48M

Try to observe the clock output

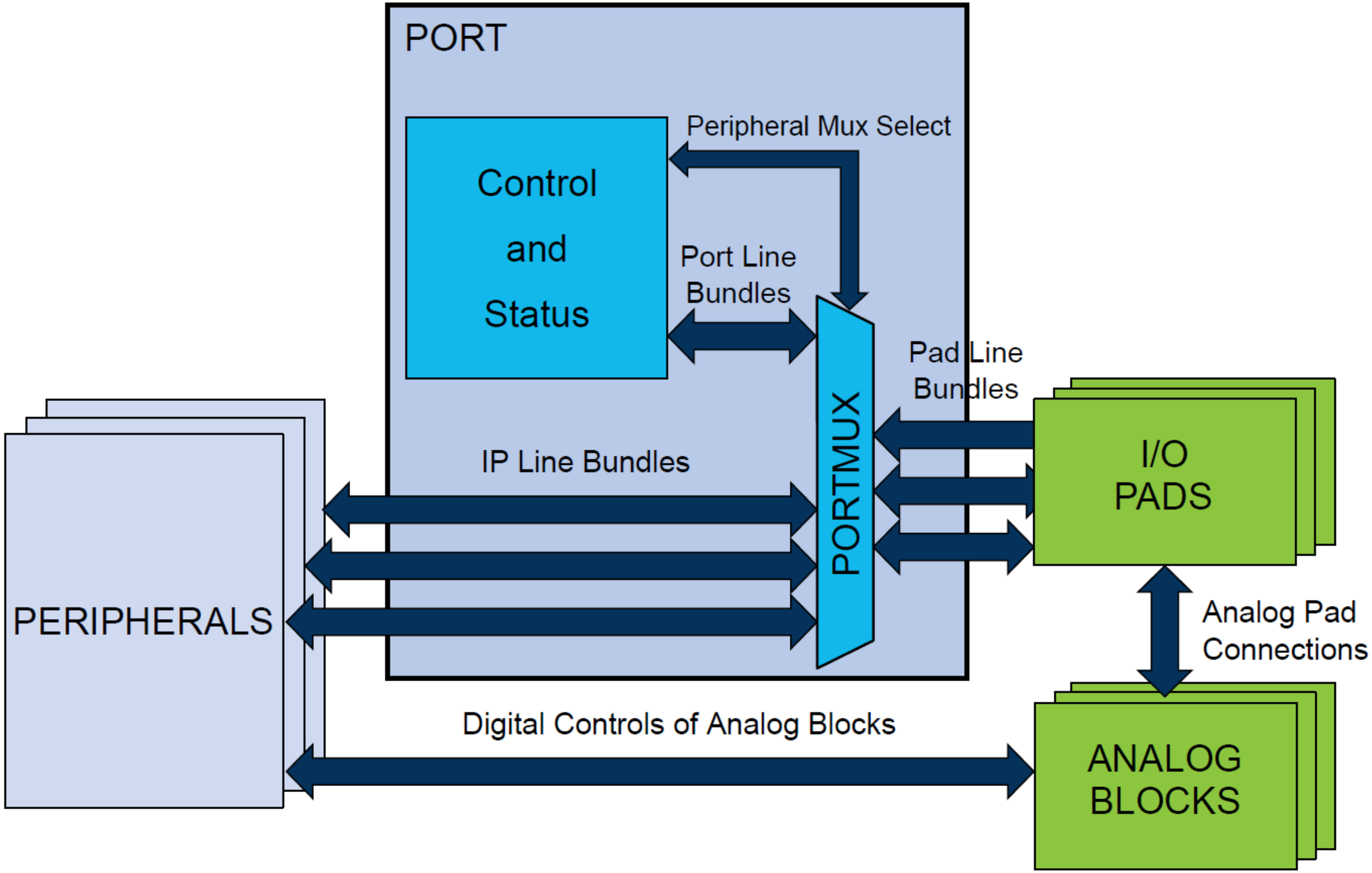






Pin Multiplexer (PINMUX)

What's PINMUX ?



What's PINMUX ?

- **Each pin is by default controlled by the PORT as a general purpose I/O and alternatively. It can be assigned to one of the peripheral functions group.**

Function Groups (A ~ H)

			PMUX		0	1						2	3	4	5	6	7
Pin[1]			I/O Pin	Supply	A	B[2][3]						C	D	E	F	G	H
SAMD21E	SAMD21G	SAMD21J			EIC	REF	ADC	AC	PTC	DAC	SERCOM[2][3]	SERCOM-ALT	TC[4]/TCC	TCC	COM	AC/ GCLK
15	23	31	PA14	VDDIO	EXTINT[14]						SERCOM2/ PAD[2]	SERCOM4/ PAD[2]	TC3/WO[0]	TCC0/ WO[4]		GCLK_IO[0]
16	24	32	PA15	VDDIO	EXTINT[15]						SERCOM2/ PAD[3]	SERCOM4/ PAD[3]	TC3/WO[1]	TCC0/ WO[5]		GCLK_IO[1]

Pin Name

MICROCHIP

PINMUX PLIB Analysis

main() @ main.c

```
SYS_Initialize ( NULL );
```

SYS_Initialize() @ initialization.c

```
PORT_Initialize();
```

Offset	Name	7	6	5	4	3	2	1	0
0x30	PMUX0	PMUXO[3:0]				PMUXE[3:0]			
PMUX		PMUXO (ODD)				PMUXE (EVEN)			
0x3F	PMUX15	PMUXO[3:0]				PMUXE[3:0]			
0x40	PINCFG0		DRVSTR				PULLEN	INEN	PMUXEN
PINCFG									
0x5F	PINCFG31		DRVSTR				PULLEN	INEN	PMUXEN

INEN PMUXEN

PORT_Initialize() @ plib_port.c

```
PORT_REGS->GROUP[0].PORT_DIR = 0x120200;
PORT_REGS->GROUP[0].PORT_OUT = 0x200;
PORT_REGS->GROUP[0].PORT_PINCFG[14] = 0x3;
PORT_REGS->GROUP[0].PORT_PINCFG[15] = 0x3;

PORT_REGS->GROUP[0].PORT_PMUX[7] = 0x77;
```

PA14 (PINMUX Enable) **INEN = 1 (Input buffer Enable)**

PA15 (PINMUX Enable) **INEN = 1 (Input buffer Enable)**

PMUX[7] : PA15(GCLK_IO[1]) , PA14(GCLK_IO[0])

PINMUX PLIB Analysis

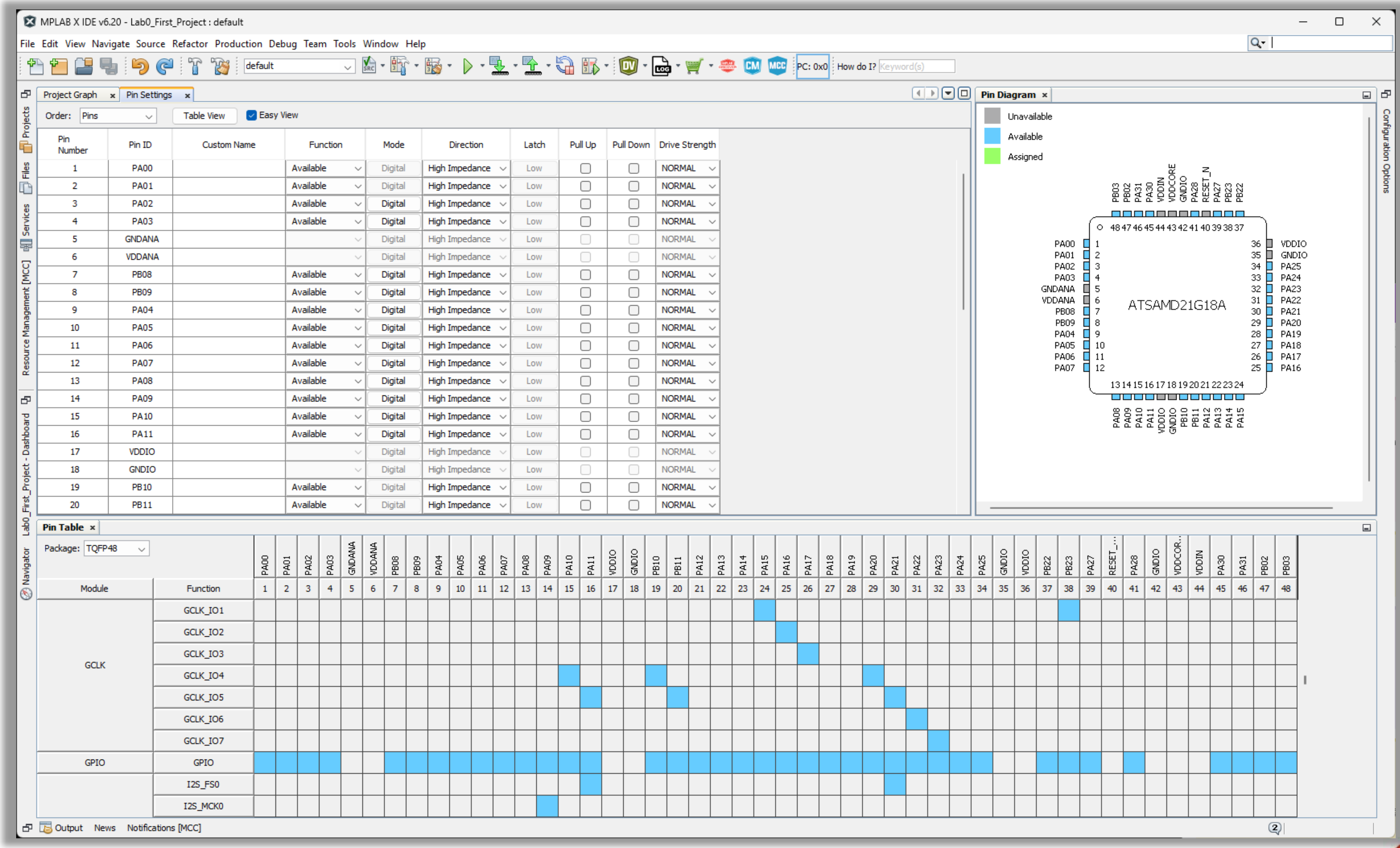
				PMUX	0	1					2	3	4	5	6	7
Pin(1)			I/O Pin	Supply	A	B(2)(3)					C	D	E	F	G	H
SAMD21E	SAMD21G	SAMD21J			EIC	REF	ADC	AC	PTC	DAC	SERCOM(2)(3)	SERCOM-ALT	TC(4)/TCC	TCC	COM	AC/ GCLK
15	PA14 (7)		PA14	VDDIO	EXTINT[14]						SERCOM2/ PAD[2]	SERCOM4/ PAD[2]	TC3/WO[0]	TCC(WO[4	GCLK_IO[0]	GCLK_IO[0]
16	PA15 (7)		PA15	VDDIO	EXTINT[15]						SERCOM2/ PAD[3]	SERCOM4/ PAD[3]	TC3/WO[1]	TCC(WO[5	GCLK_IO[1]	GCLK_IO[1]

PORT_Initialize() @ plib_port.c

```
PORT_REGS->GROUP[0].PORT_DIR = 0x120200;
PORT_REGS->GROUP[0].PORT_OUT = 0x200;
PORT_REGS->GROUP[0].PORT_PINCFG[14] = 0x3;
PORT_REGS->GROUP[0].PORT_PINCFG[15] = 0x3;

PORT_REGS->GROUP[0].PORT_PMUX[7] = 0x77;
```

PMUX[7] : PA15(GCLK_IO[1]) , PA14(GCLK_IO[0])

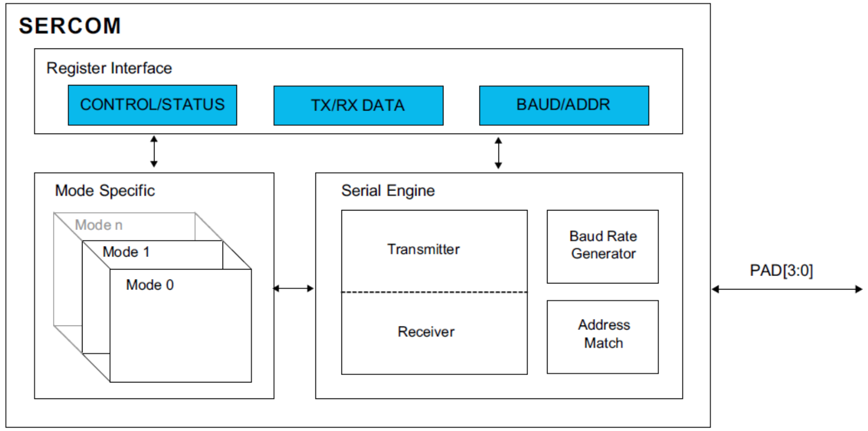
PINMUX Configuration Example



SERCOM – UART Architecture

SERCOM Introduction

- Serial Communication Interface, SERCOM be configured to support a number of modes :
  - I2C
  - SPI
  - USART : RS232, RS485, IrDA
- There are up to six instances of the serial communication interface SERCOM peripheral.

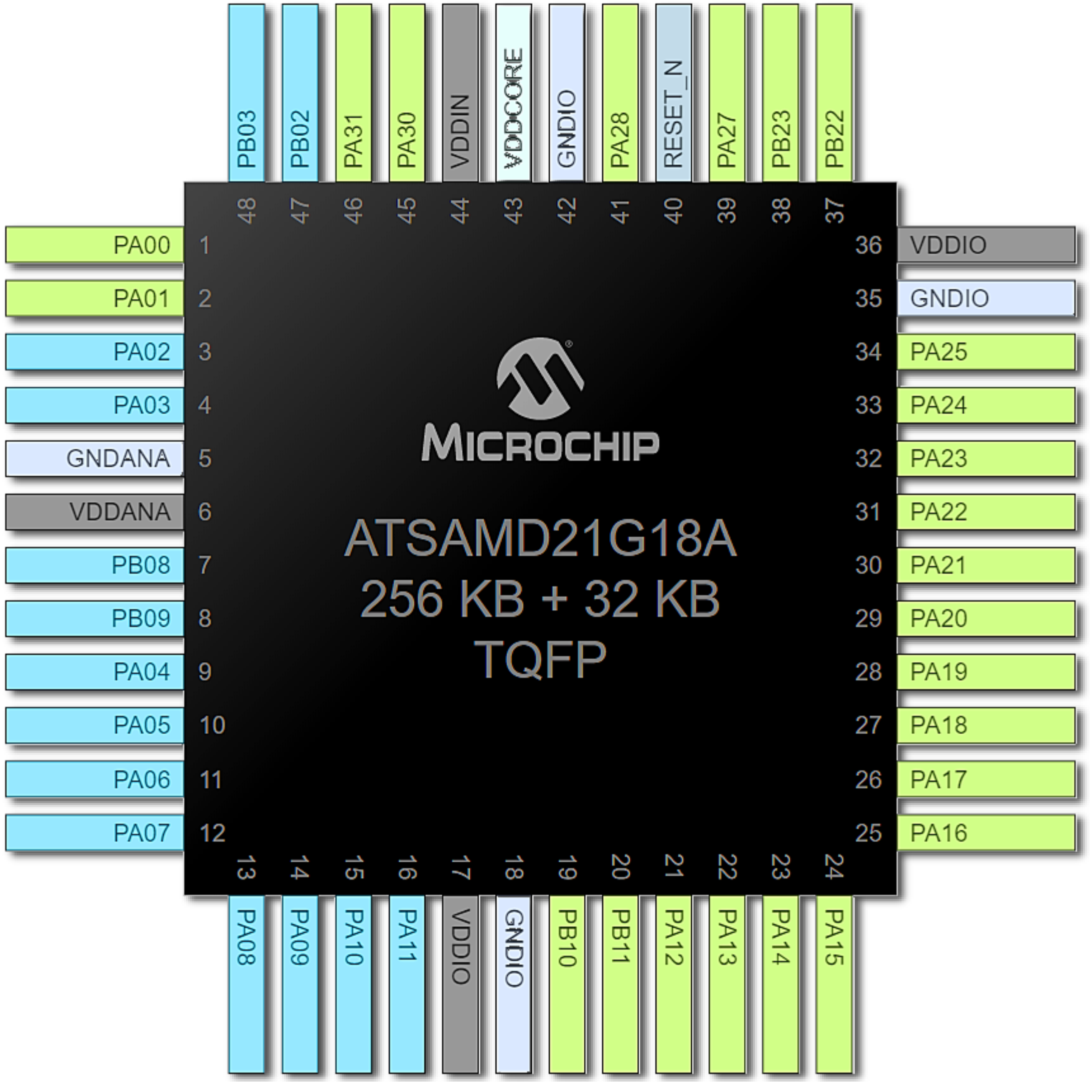


- The SERCOM serial engine consists of a transmitter and receiver, baud-rate generator and address matching functionality. It can use the internal generic clock or an external clock to operate in sleep.

SERCOM Introduction

- **Pin Multiplex**

	PAD[0]	PAD[1]	PAD[2]	PAD[3]	PMUX
SERCOM0	PA04	PA05	PA06	PA07	D
	PA08	PA09	PA10	PA11	C
SERCOM1	PA00	PA01	PA30	PA31	D
	PA16	PA17	PA18	PA19	C
SERCOM2	PA08	PA09	PA10	PA11	D
	PA12	PA13	PA14	PA15	C
SERCOM3	PA16	PA17	PA18	PA19	D
			PA20	PA21	D
	PA22	PA23	PA24	PA25	C
SERCOM4	PA12	PA13	PA14	PA15	D
	PB08	PB09	PB10	PB11	D
SERCOM5			PA20	PA21	C
	PA22	PA23	PA24	PA25	D
	PB02	PB03	PB22	PB23	D



MICROCHIP

SERCOM Introduction

- **Pin Multiplex in Harmony (Clear)**

Pin Table

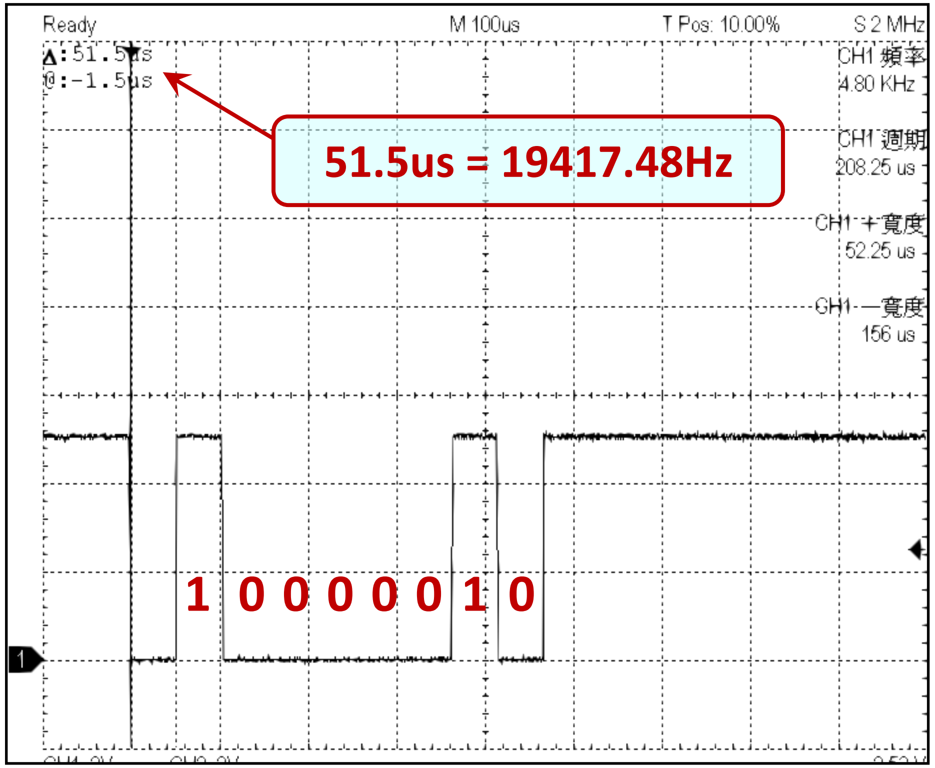
Package: TQFP48		1	2	3	4	9	10	11	12	13	14	15	16	21	22	23	24	25	26	27	28	29	30	31	32	33	34	39	41	45	46	47	48	7	8	19	20	37	38
		A																														B							
Module	Function	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	27	28	30	31	2	3	8	9	10	11	22	23
SERCOM0	SERCOM0_PAD0					■				■																													
	SERCOM0_PAD1						■				■																												
	SERCOM0_PAD2							■				■																											
	SERCOM0_PAD3								■				■																										
SERCOM1	SERCOM1_PAD0	■																■																					
	SERCOM1_PAD1		■																■																				
	SERCOM1_PAD2																			■										■									
	SERCOM1_PAD3																				■										■								
SERCOM2	SERCOM2_PAD0									■				■																									
	SERCOM2_PAD1										■				■																								
	SERCOM2_PAD2											■				■																							
	SERCOM2_PAD3												■				■																						
SERCOM3	SERCOM3_PAD0																	■						■															
	SERCOM3_PAD1																		■						■														
	SERCOM3_PAD2																			■		■				■													
	SERCOM3_PAD3																				■		■				■												
SERCOM4	SERCOM4_PAD0													■																				■					
	SERCOM4_PAD1														■																				■				
	SERCOM4_PAD2															■																				■			
	SERCOM4_PAD3																■																				■		
SERCOM5	SERCOM5_PAD0																							■								■							
	SERCOM5_PAD1																								■								■						
	SERCOM5_PAD2																					■				■												■	
	SERCOM5_PAD3																						■				■												■

SERCOM Introduction

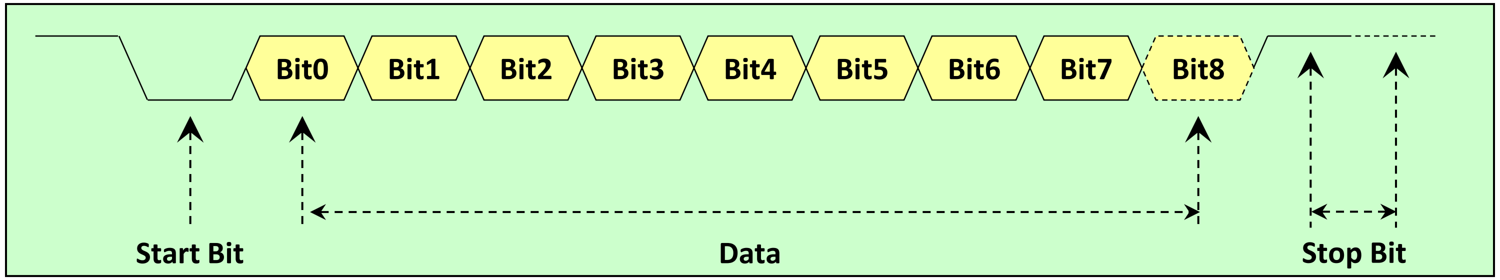
- **For I2C**
  - Fast Mode (400kHz)
  - Master or slave operation
  - SMBUS compatible
  - Wake on address match
- **For SPI**
  - Up to 24Mb/s
  - Supports all four SPI modes
  - Full-duplex operation
  - Single data direction mode frees unused MISO or MOSI pin
- **For USART**
  - Up to 24Mb/s
  - Half-/Full-duplex operation
  - Sync and Async operation

What’s UART ?

- UART : Universal Asynchronous Receiver Transmitter.
- The module data exchange through serial communication.
- The data formatting include
  1 start bit,
  5 to 9 data bits and
  1 or 2 stop bits.

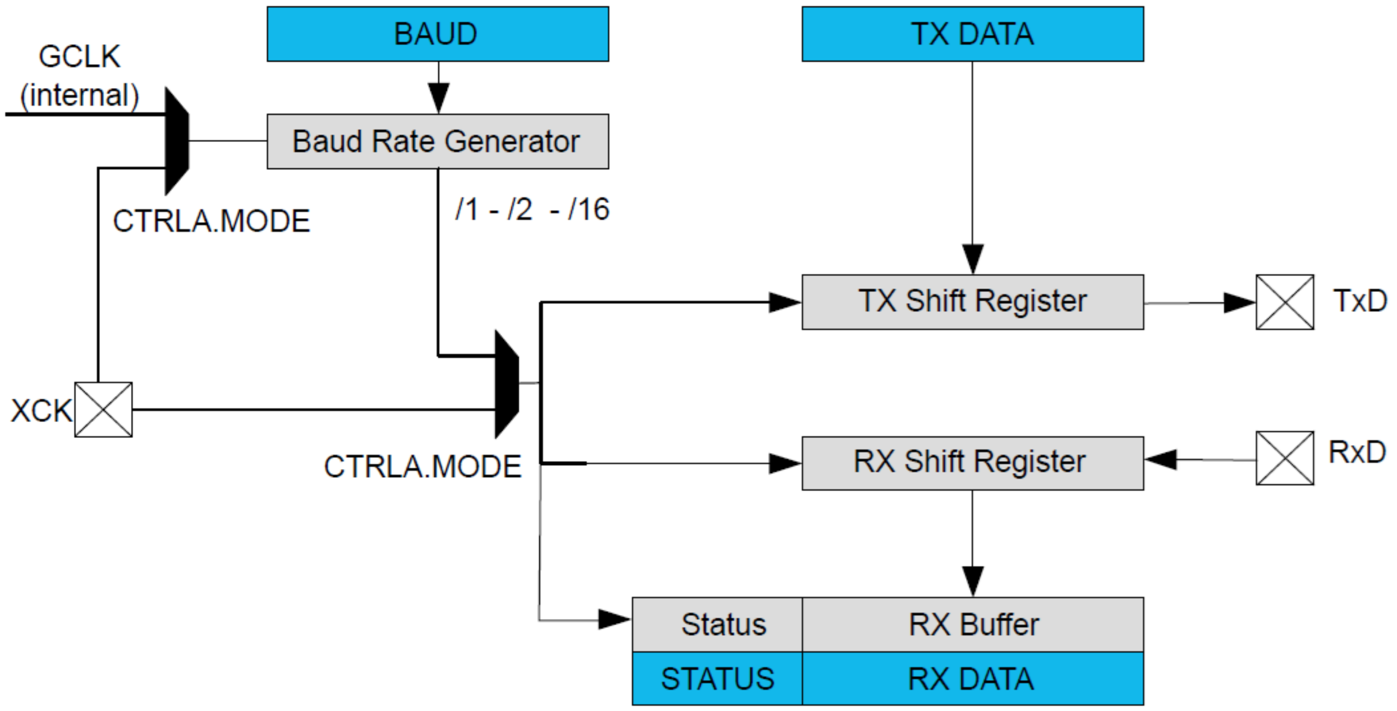


UART Transmit Example :
’A’(0x41), 19200 bps



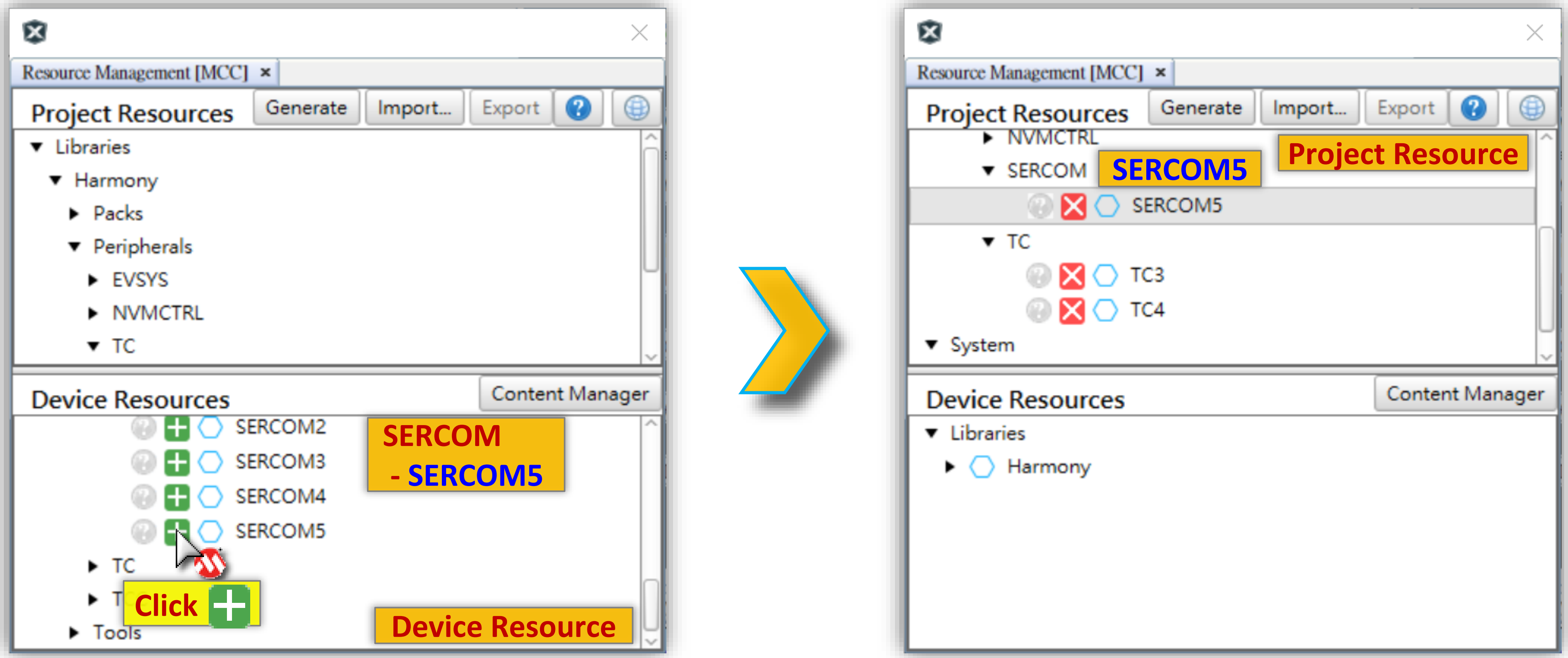
SERCOM - USART

- Supports serial frames with 5, 6, 7, 8 or 9 data bits and 1 or 2 stop bits, Parity Check, RTS and CTS flow control.
- Full-duplex operation, RS-232, RS-485 and IrDA Supported.
- Buffer overflow and frame error detection.
- Internal or external clock source for asynchronous and synchronous operation.



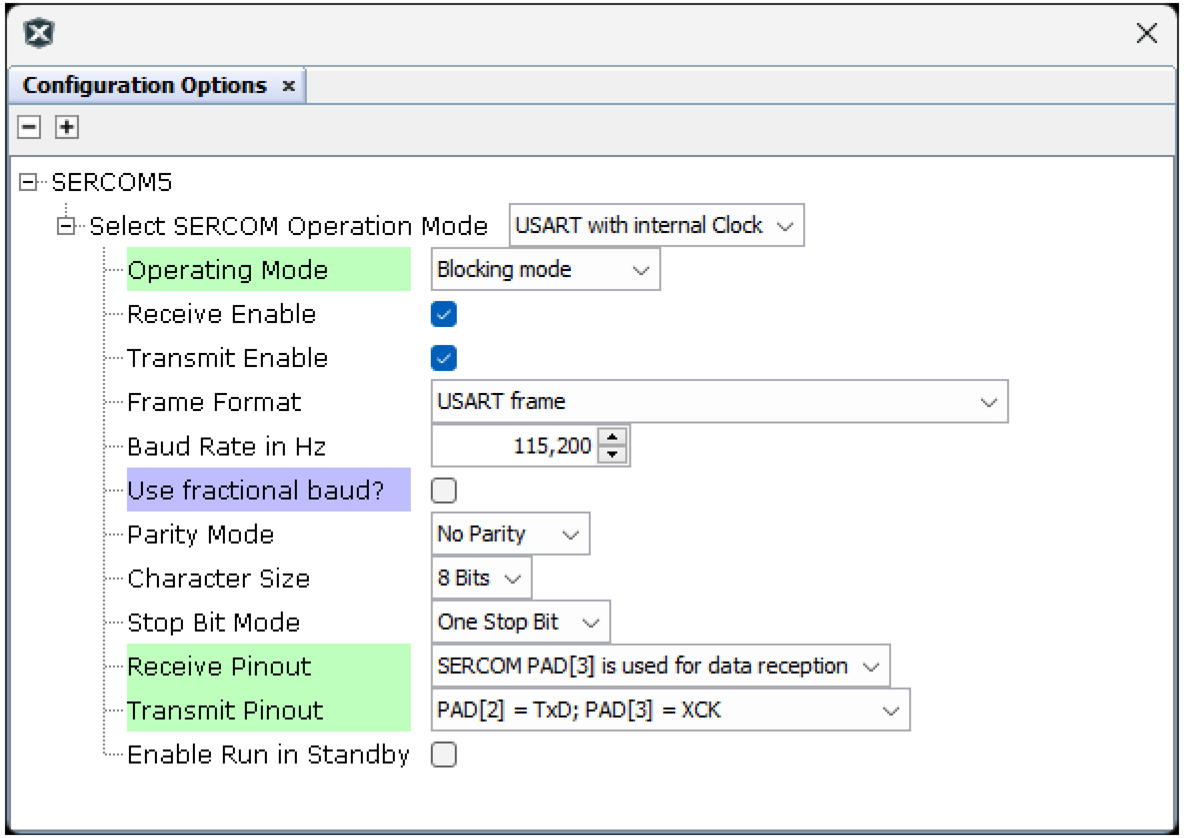
Add SERCOM Function using MCC Harmony

- Find **SERCOM5** component in Device Resource window.
- Click  to add **SERCOM5** to Project Resource window.

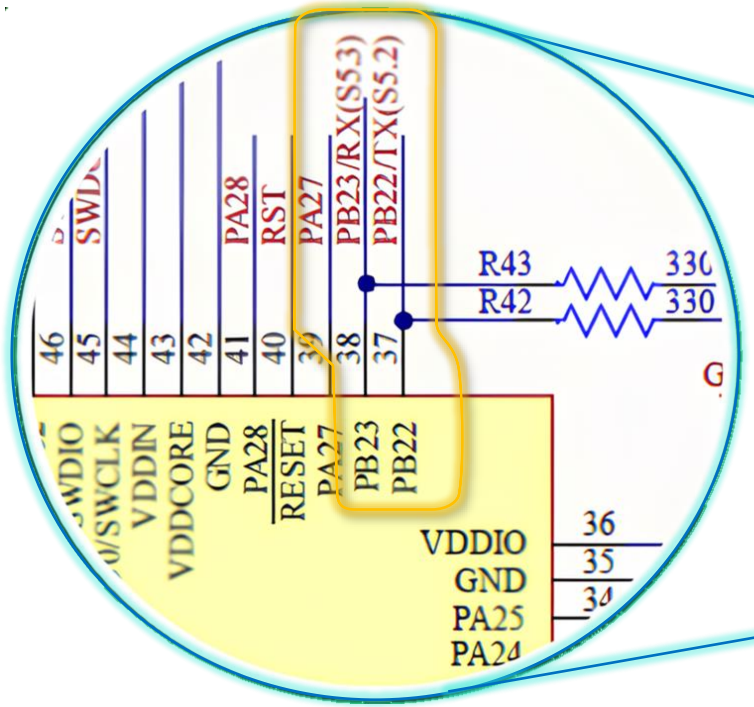


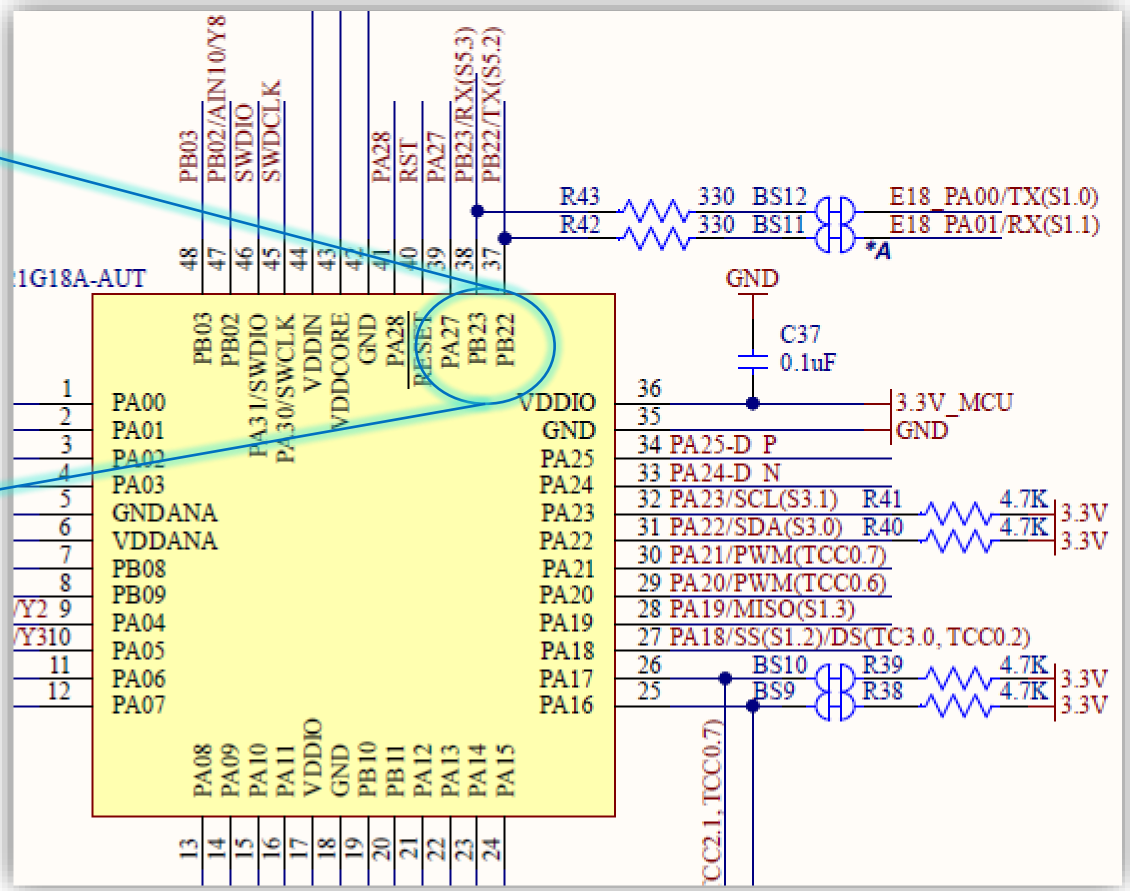
SERCOM USART Polling

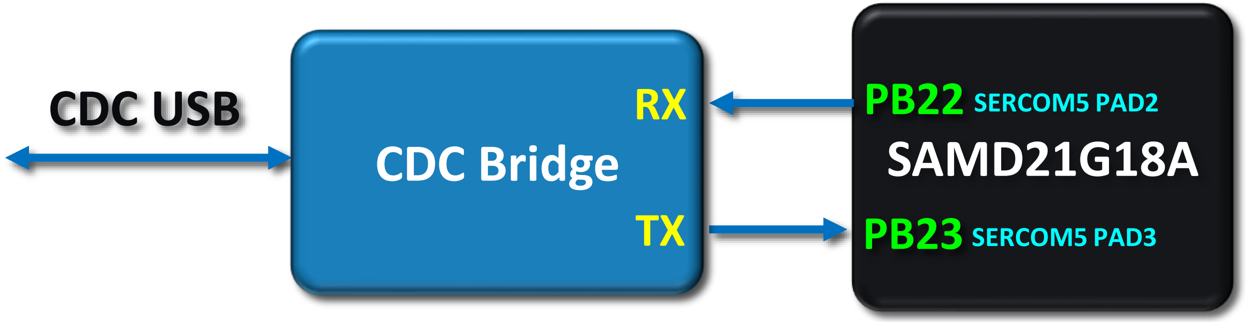
Configuration Options Example



APP045 SERCOM USART Schematic





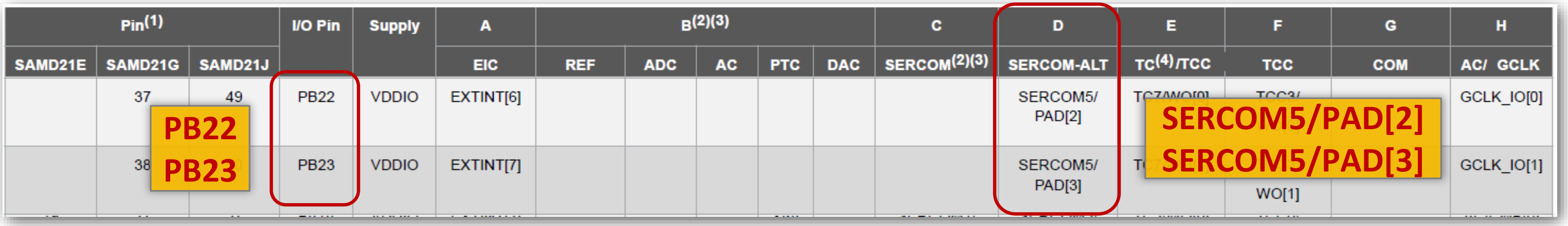


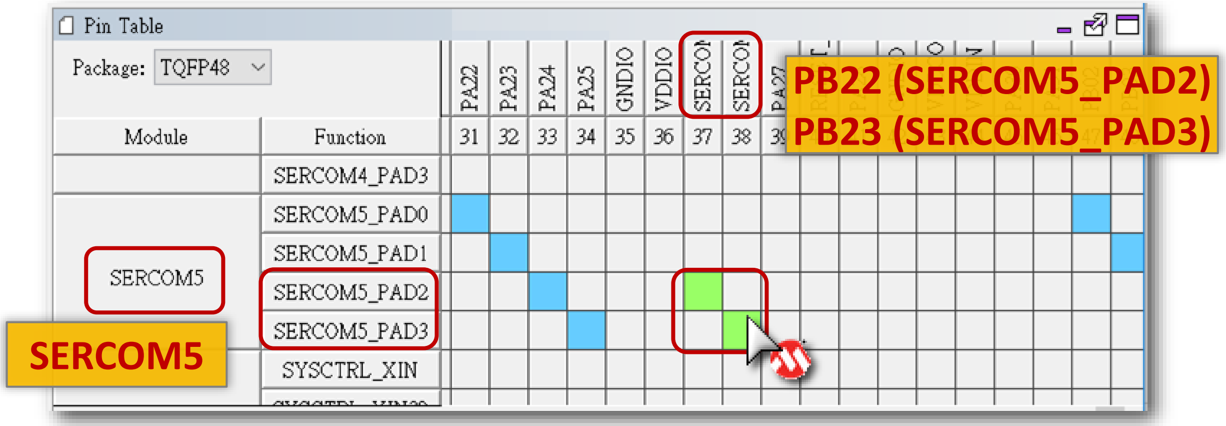
MICROCHIP

SERCOM USART

PINMUX Configuration Example

- **SERCOM allow limited alternate pad index assignment.**
- **For example : PB22 as SERCOM5_PAD2**
  PB23 as SERCOM5_PAD3

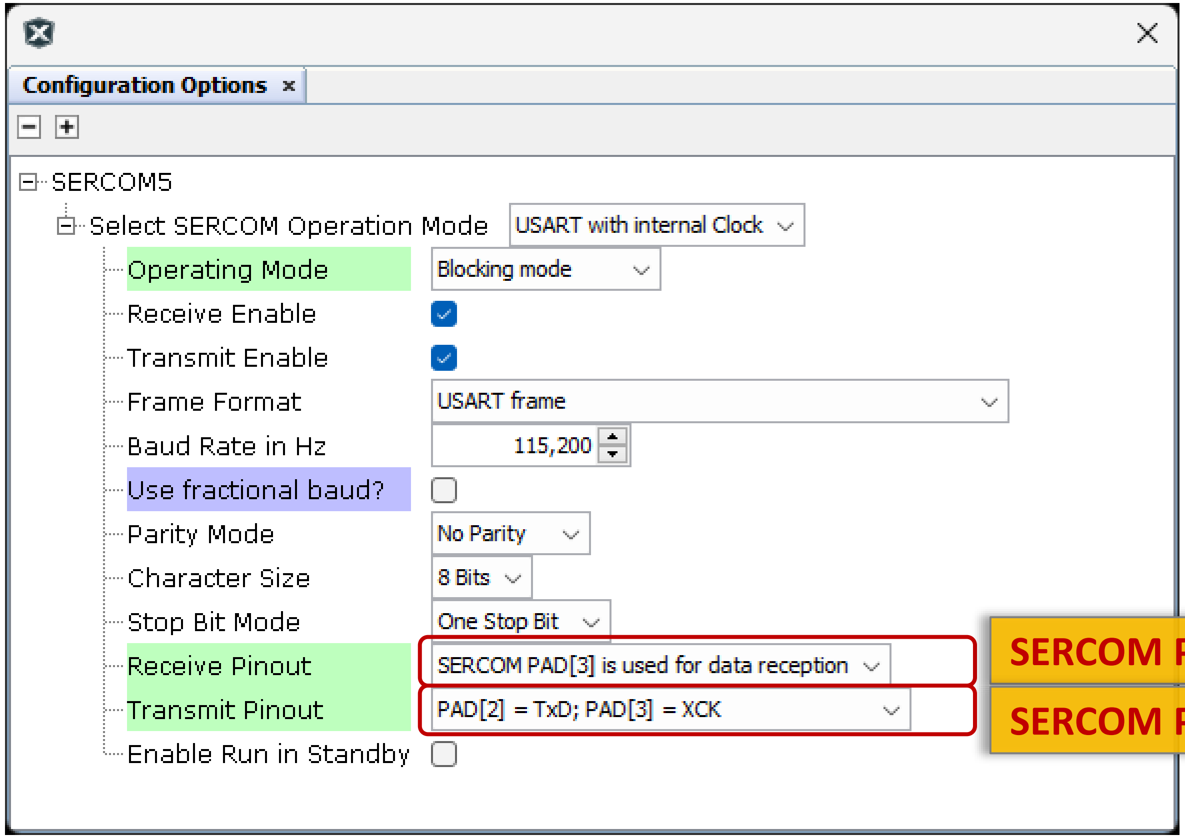




SERCOM USART

PINMUX Configuration Example

- Remember arrange pads function in SERCOM module.
- For example : SERCOM5 PAD[2] as USART TXD (PB22)
  SERCOM5 PAD[3] as USART RXD (PB23)



SERCOM PAD[3] is used for Data Receive

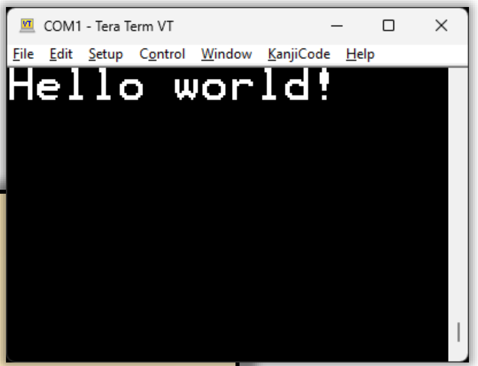
SERCOM PAD[2] is used for Data Transmit

SERCOM USART Polling

Interface Routines

```
void SERCOM5_USART_Initialize( void );
bool SERCOM5_USART_SerialSetup( USART_SERIAL_SETUP * serialSetup, uint32_t clkFrequency );
bool SERCOM5_USART_Write( void *buffer, const size_t size );
bool SERCOM5_USART_TransmitterIsReady( void );
bool SERCOM5_USART_TransmitComplete( void );
void SERCOM5_USART_WriteByte( int data );
bool SERCOM5_USART_Read( void *buffer, const size_t size );
bool SERCOM5_USART_ReceiverIsReady( void );
int SERCOM5_USART_ReadByte( void );
USART_ERROR SERCOM5_USART_ErrorGet( void );
uint32_t SERCOM5_USART_FrequencyGet( void );
```

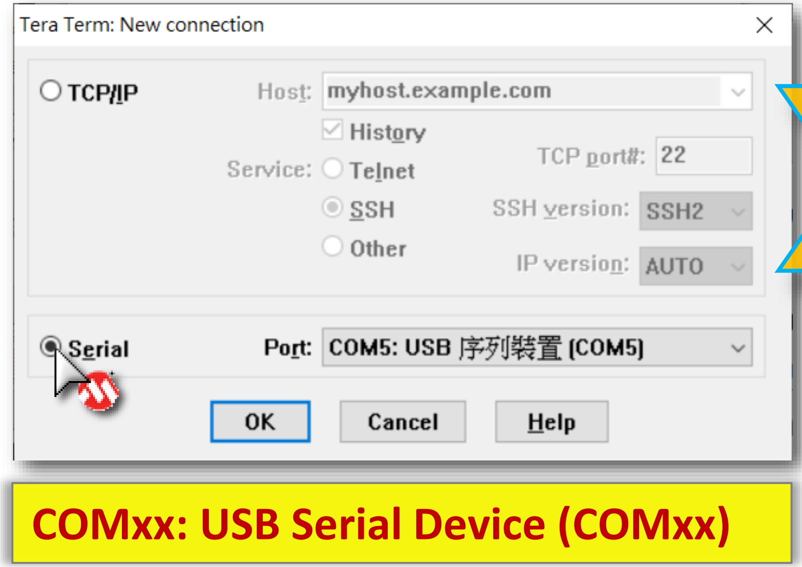
SERCOM USART Tx Polling Code Example

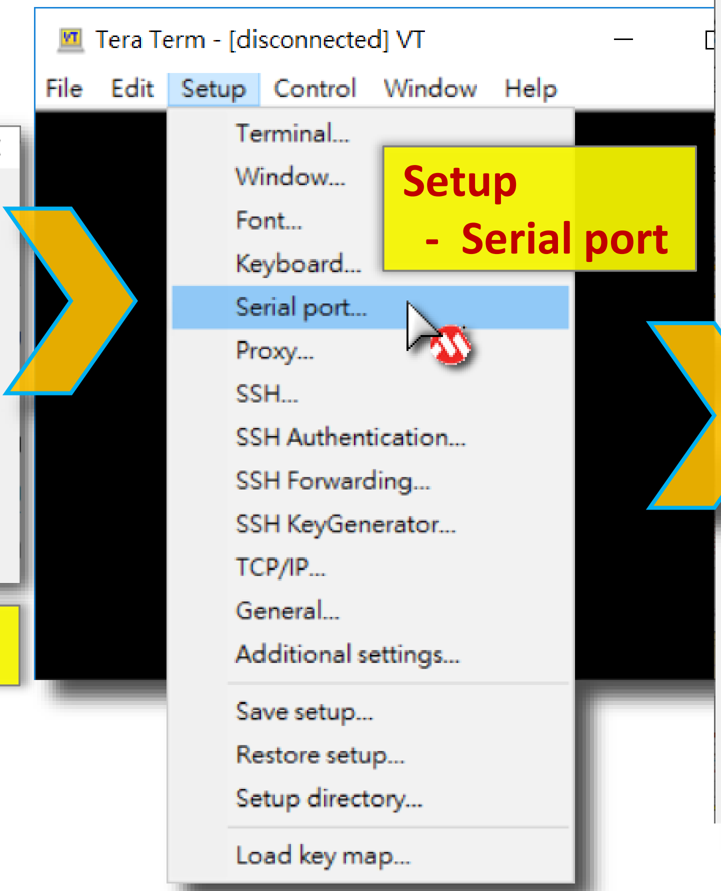


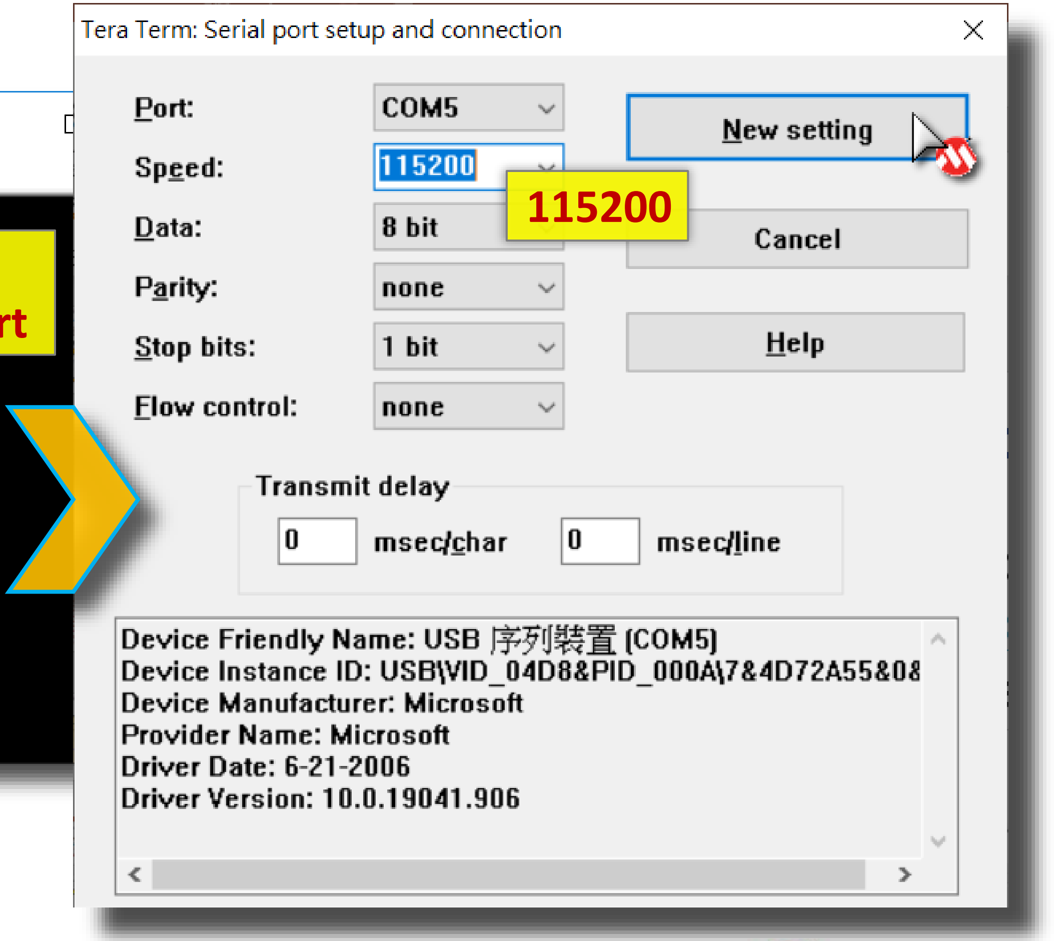

```
…

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

   while( true )
   {
      …
      SERCOM5_USART_Write( "Hello world!\r\n", 14 );
   }
}
```

Terminal Console Software

- Tera Term is a terminal console software. (https://teratermproject.github.io/)
- Select COMxx: USB Serial Device (COMxx)
- Set 115200-N-8-1.







MICROCHIP

Lab9 SERCOM - UART Tx Polling

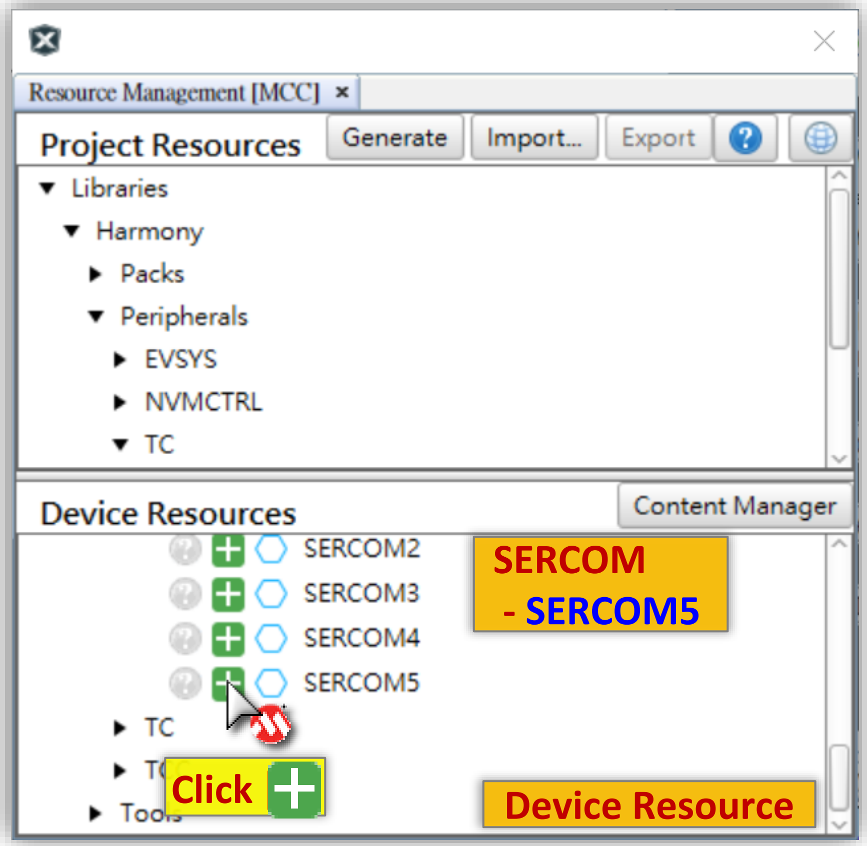
- Try to add SERCOM-USART function to project.
- UART set to 115200-N-8-1, No flow control.
- Send string from UART to PC of one second interval repeated. (Hello World!)
- Character \r for carriage Return (0x0D)
- Character \n for New line (0x0A)

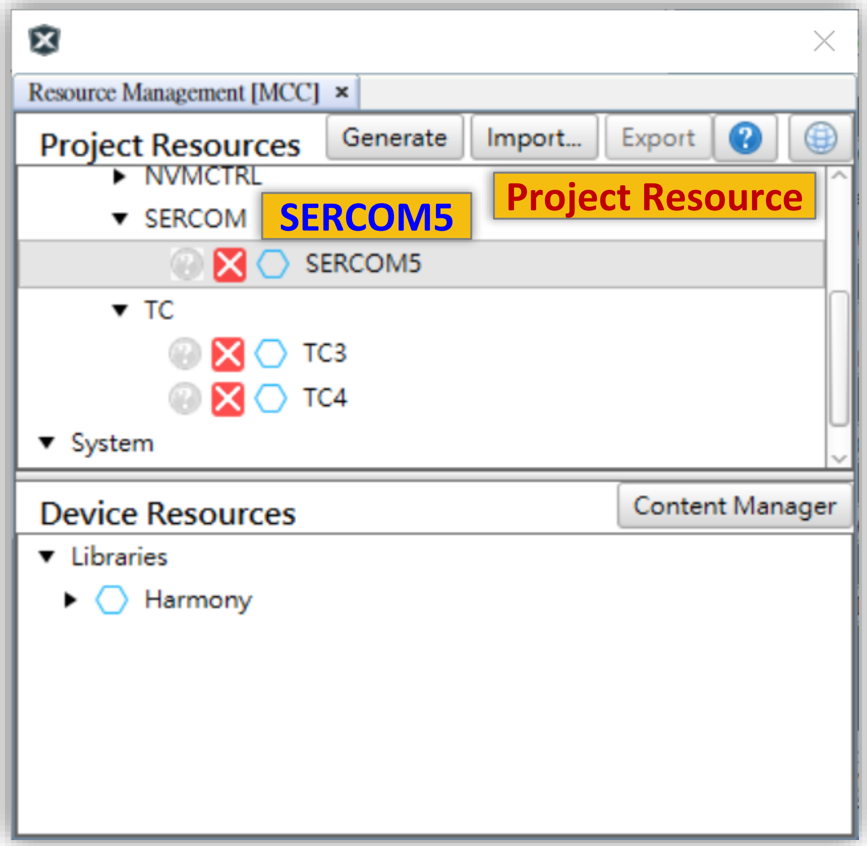
- **Let's go!**

Lab9 SERCOM - UART Tx Polling

Step 1

- Find **SERCOM5** component in Device Resource window.
- Click [+] to add **SERCOM5** to Project Resource window.

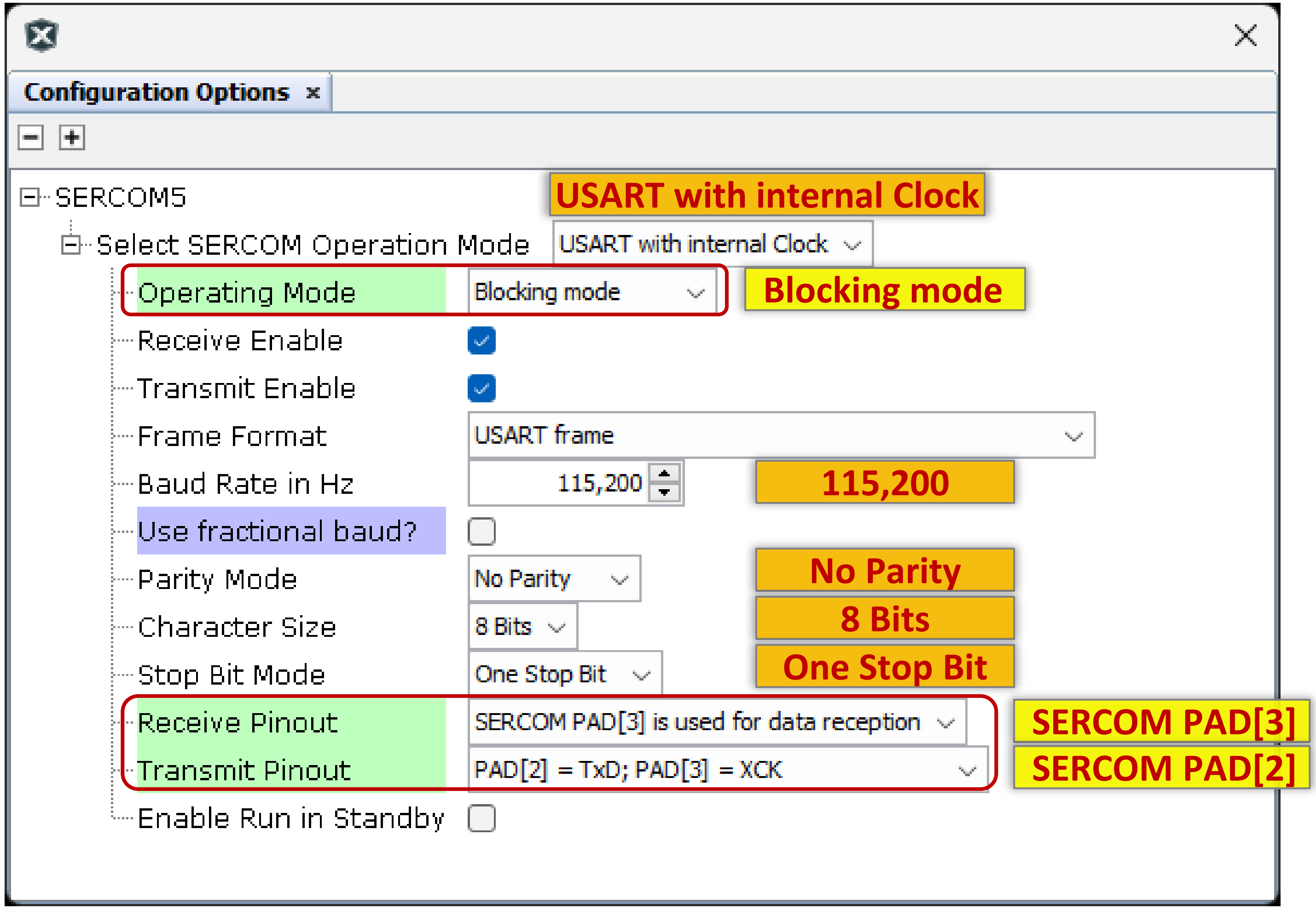




MICROCHIP

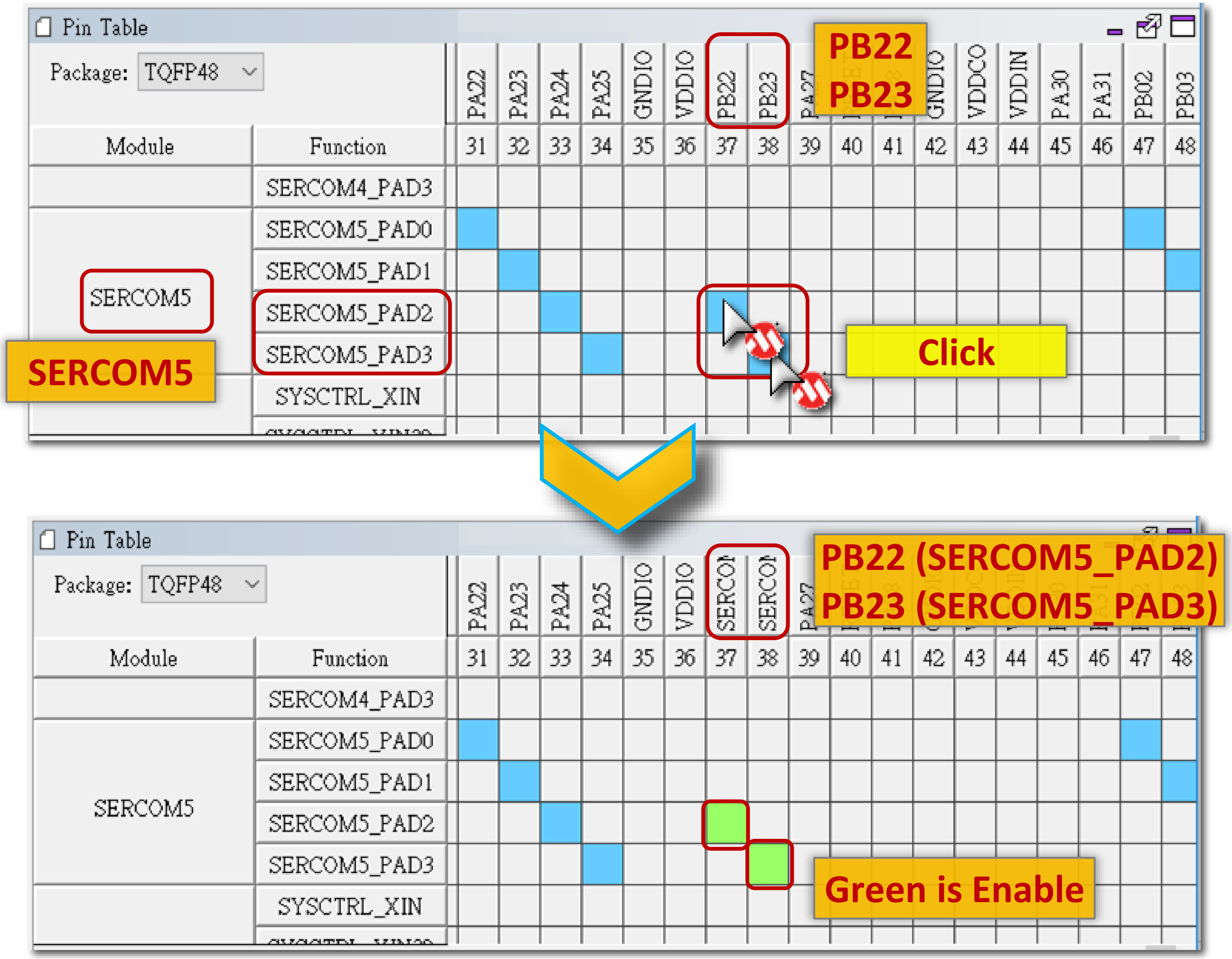
Lab9 SERCOM - UART Tx Polling

Step 2



Lab9 SERCOM - UART Tx Polling

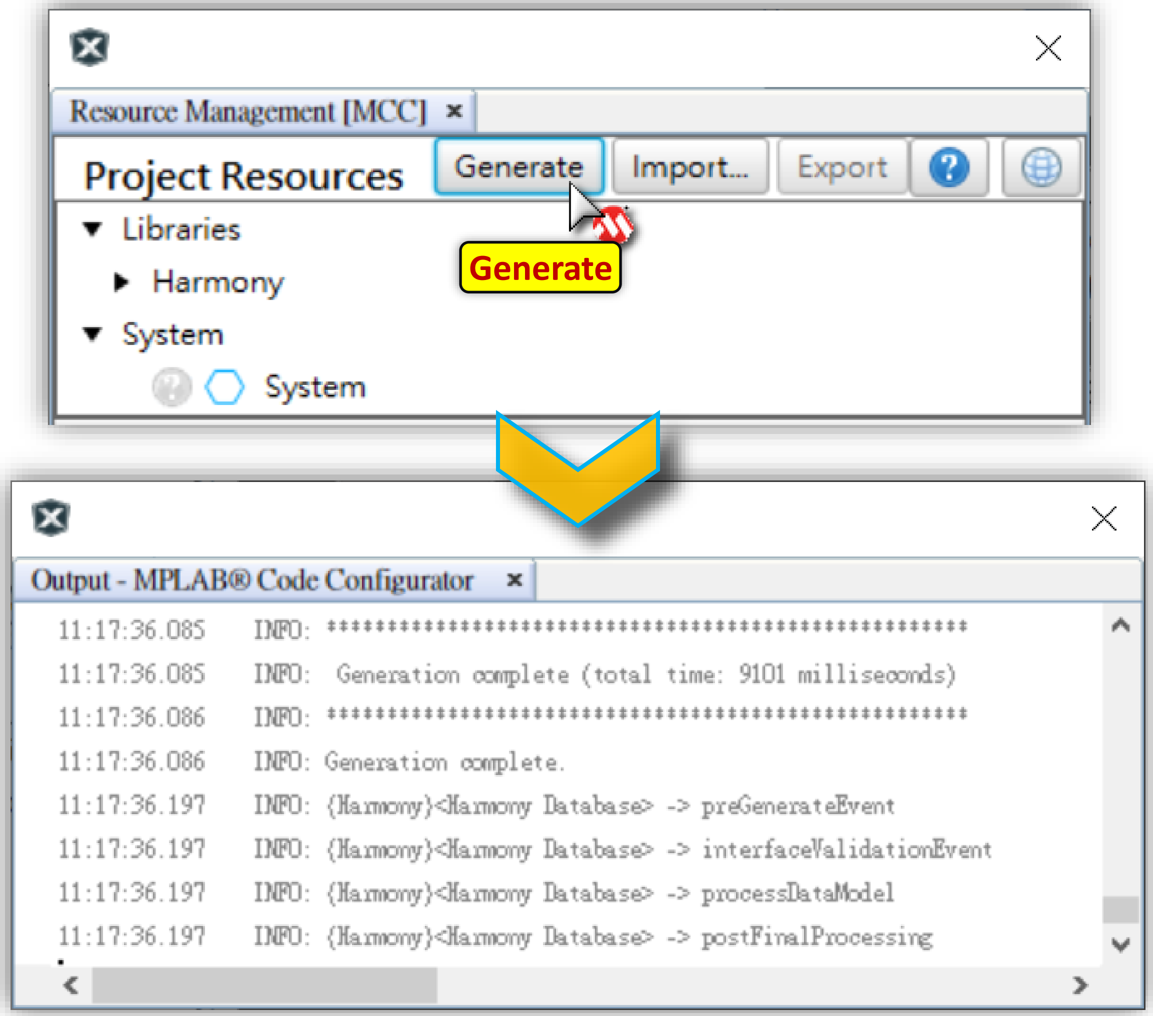
Step 3



Lab9 SERCOM - UART Tx Polling

Step 4

- Click "Generate" Button to Generate your Code.



Lab9 SERCOM - UART Tx Polling

Step 5

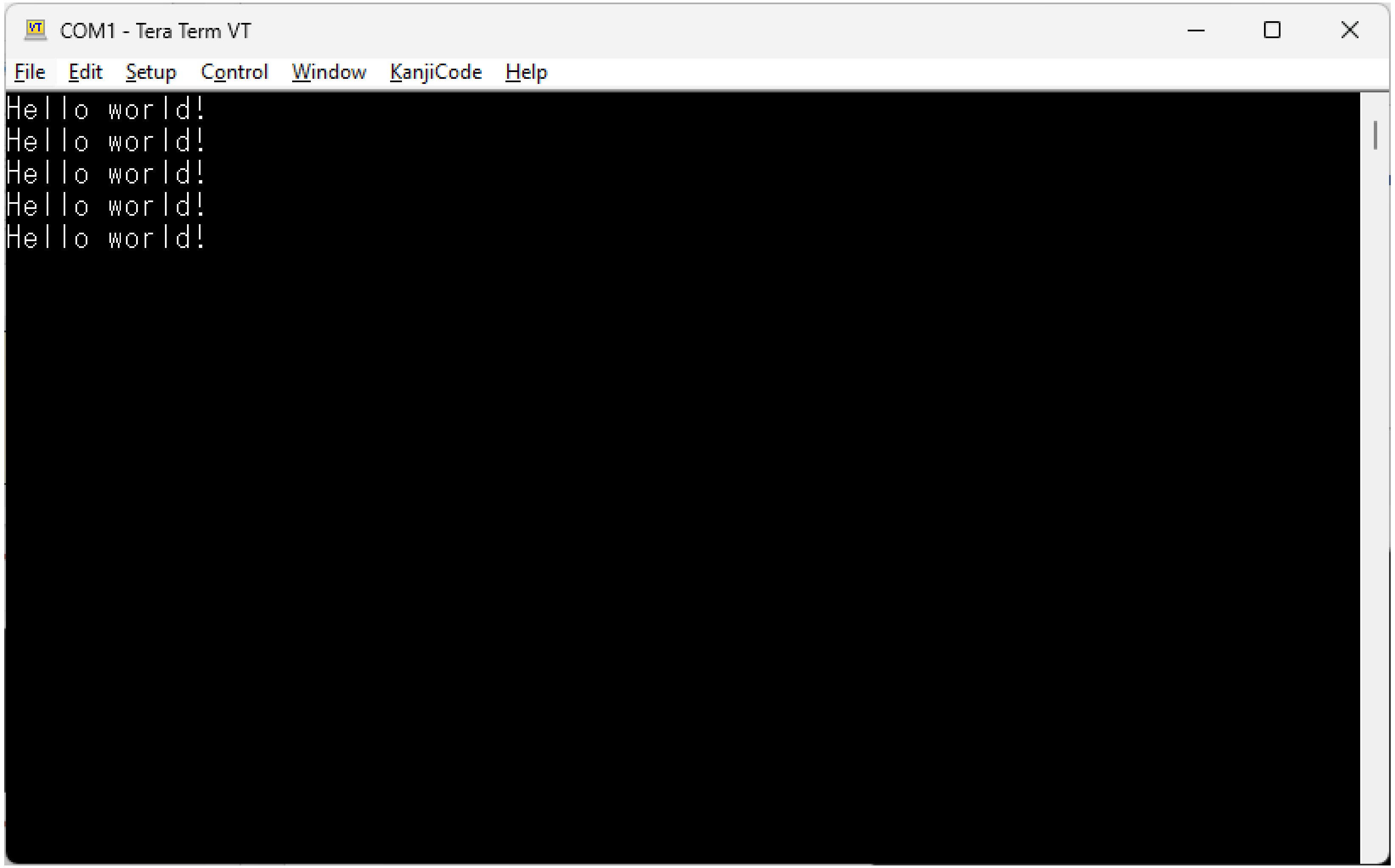
a Add code segment to your main loop

```
void TC3_TimerExpired(TC_TIMER_STATUS status, uintptr_t context)
{
        if( status & TC_INTFLAG_Msk )
        {
            LED1_Toggle();

            // TODO 9.01
             SERCOM5_USART_Write( "Hello world!\r\n", 14 );
        }
}

int main (void)
{
    SYS_Initialize ( NULL );

    …

}
```

b Program firmware to target board then observe result.

Lab9 SERCOM - UART Tx Polling

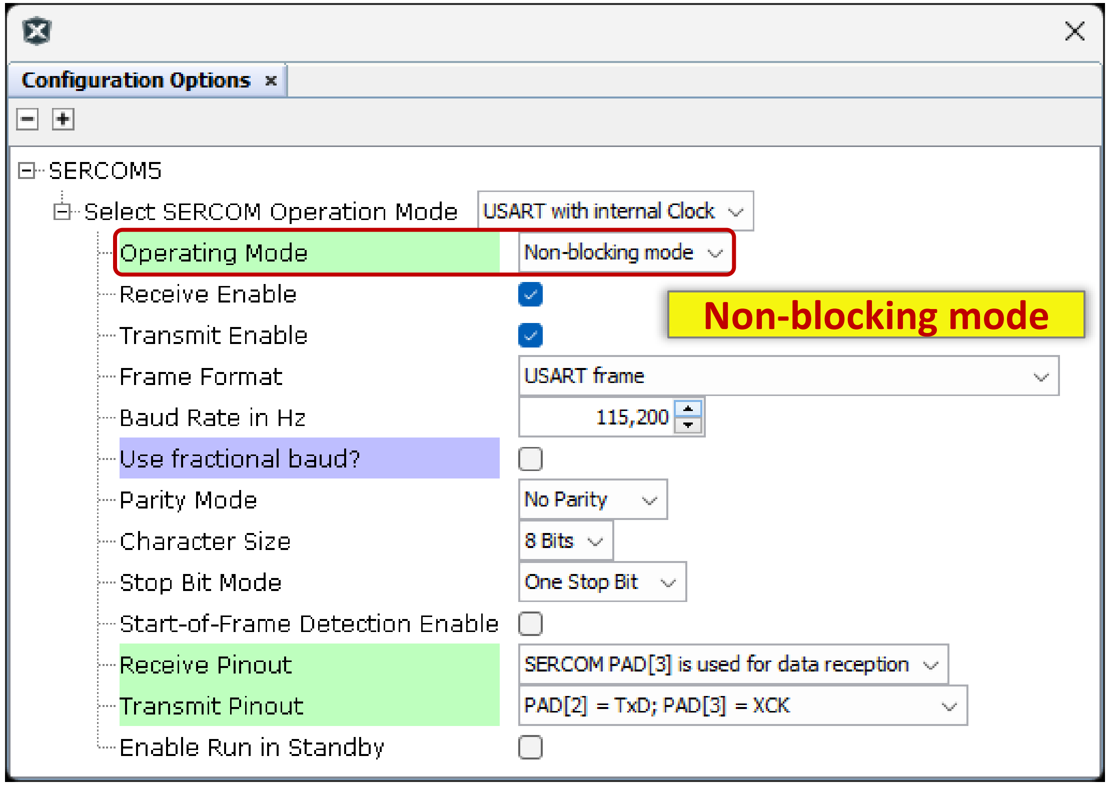
Result

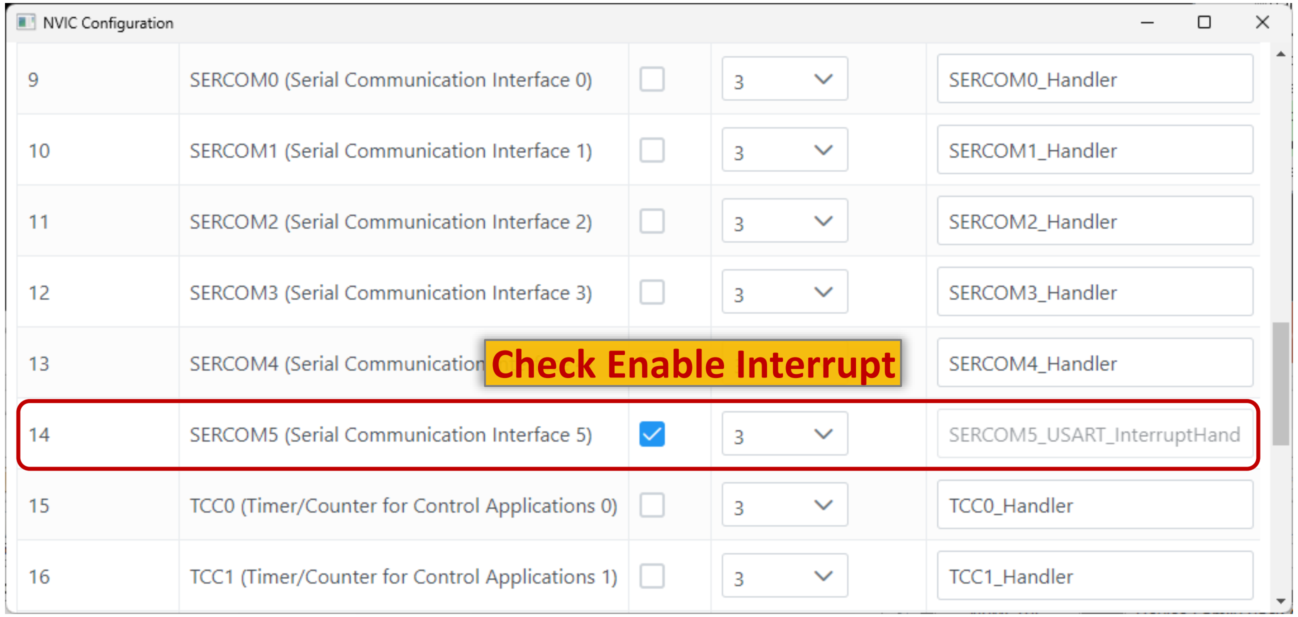


SERCOM USART Interrupt

Configuration Options Example

- **SERCOM provide Non-blocking (interrupt) mode, also.**





SERCOM USART Interrupt

Interface Routines

```
void SERCOM5_USART_Initialize( void );
bool SERCOM5_USART_SerialSetup( USART_SERIAL_SETUP * serialSetup, uint32_t clkFrequency );
bool SERCOM5_USART_Write( void *buffer, const size_t size );
bool SERCOM5_USART_WriteIsBusy( void );
void SERCOM5_USART_WriteCallbackRegister( SERCOM_USART_CALLBACK callback, uintptr_t context );
bool SERCOM5_USART_Read( void *buffer, const size_t size );
bool SERCOM5_USART_ReadIsBusy( void );
void SERCOM5_USART_ReadCallbackRegister( SERCOM_USART_CALLBACK callback, uintptr_t context );
USART_ERROR SERCOM5_USART_ErrorGet( void );
uint32_t SERCOM5_USART_FrequencyGet( void );
```

USART Interrupt Code Example

```
uint8_t USART5_ReceiveData[1];
void USART5_Receive( uintptr_t context )
{
   SERCOM5_USART_Write( USART5_ReceiveData, 1 );
   SERCOM5_USART_Read( USART5_ReceiveData, 1 );
}

void USART5_ Transmit( uintptr_t context ) {  }

int main (void)
{
   SYS_Initialize ( NULL );
   …
   SERCOM5_USART_ReadCallbackRegister( USART5_Receive, (uintptr_t )NULL );
   SERCOM5_USART_WriteCallbackRegister( USART5_Transmit, (uintptr_t )NULL );
   SERCOM5_USART_Read( USART5_ReceiveData, 1 );
   while( true )
   { …
     SERCOM5_USART_Write( "Hello world!\r\n", 14 );
   }
}
```

One-Shot Event

Callback Register

About String Functions

- UART generally to used for text string communication, we can use the string functions provided by C to convert multiple variables to an output string.
- Here introduce two string functions by C Standard Library, sprintf() and strlen().
- int sprintf(char *s, const char *format, ...);
  Formatting values to target char type string buffer.
  * Must include <stdio.h>
- size_t strlen(const char *s);
  Calculate string length, not include the end of string 0x00('\0').
  * Must include <string.h>
- For example :

```
#include <stdio.h>
#include <string.h>

uint8_t USARTRTxBuffer[100];
uint16_t USART5_ReceiveData;
sprintf((char *) USARTRTxBuffer, "Received Data : %1c\r\n", USART5_ReceiveData);
SERCOM5_USART_Write( USARTRTxBuffer, strlen((char *)USARTRTxBuffer));
```

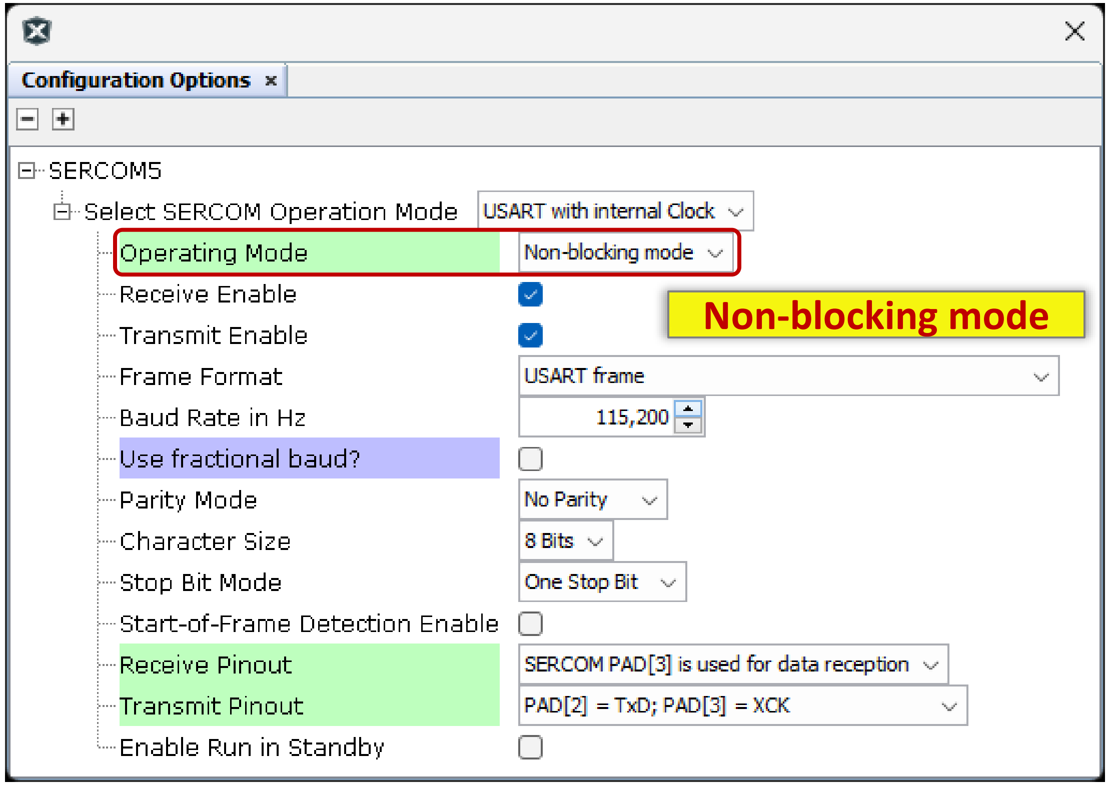
Lab10 SERCOM - UART Tx/Rx Interrupt

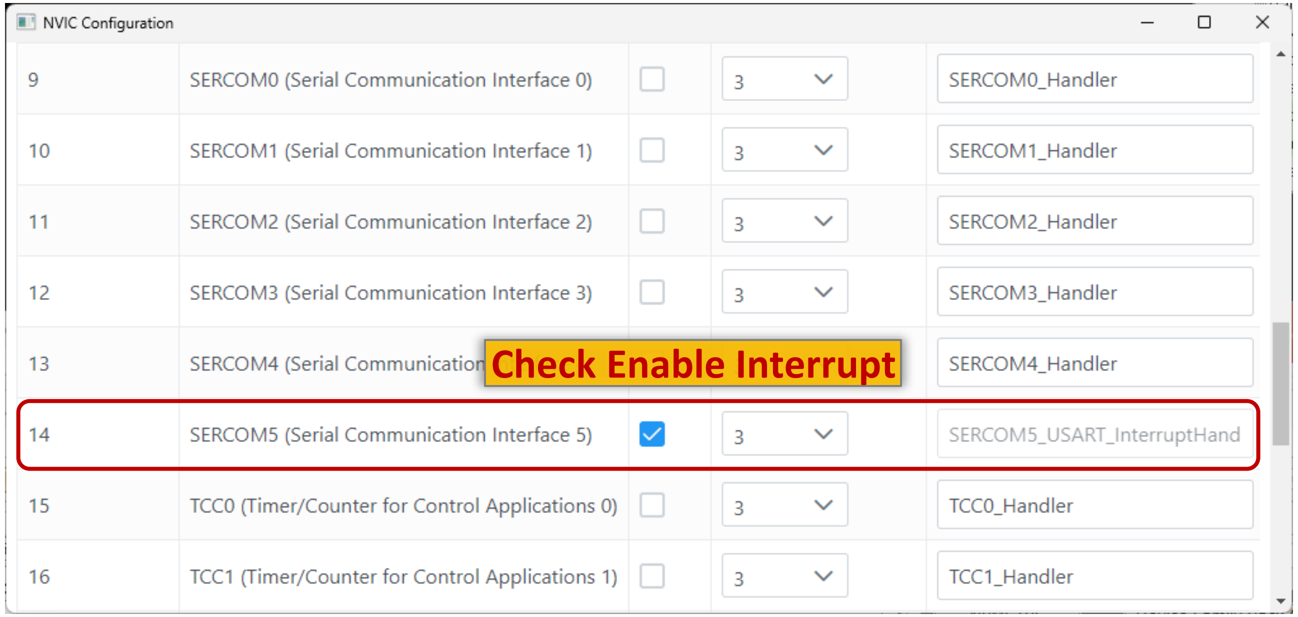
- Try to enable SERCOM-USART Interrupt Callback driver.
- Modify polling transmit function to asynchronous function.
- Add new USART receive function base on interrupt callback function.
- Try to receive character from PC and loopback to TeraTerm as string format "Received Data : x" .
- SERCOM - UART settings are the same as previous Lab.

- **Let's go!**

Lab10 SERCOM - UART Tx/Rx Interrupt

Step 1

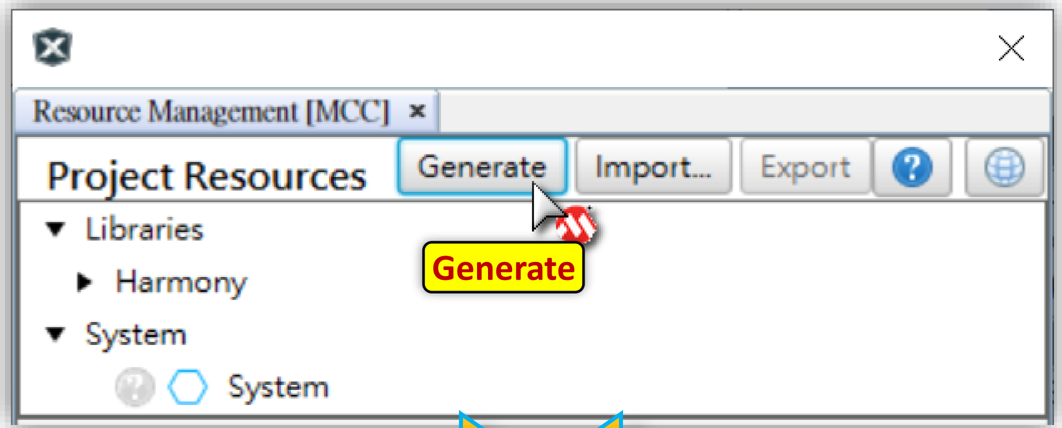


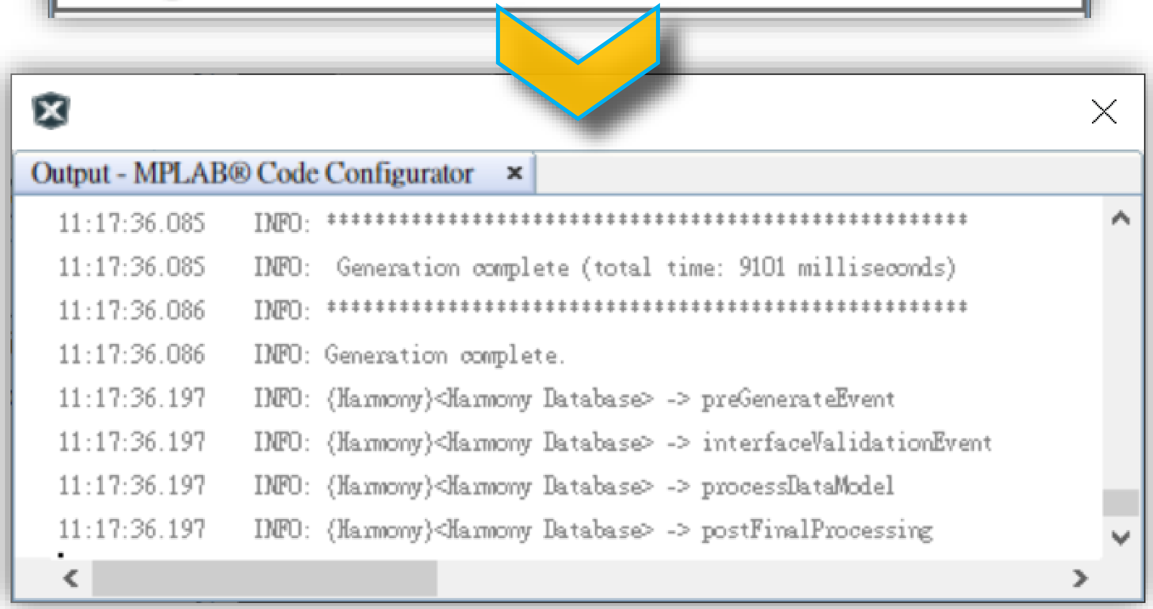


Lab10 SERCOM - UART Tx/Rx Interrupt

Step 2

- Click "Generate" Button to Generate your Code.





Lab10 SERCOM - UART Tx/Rx Interrupt

Step 3

a Create UART TxRx Callback function, related buffer and flags.

```
// TODO 10.01
#include <stdio.h>
#include <string.h>

// TODO 10.02
uint8_t USARTRTxBuffer[100];
uint8_t USART5_ReceiveData[1];
volatile uint8_t USART5_IsReceived = 0;
volatile uint8_t USART5_IsTransmitted = 1;

void USART5_Transmit( uintptr_t context )
{
  USART5_IsTransmitted = 1;
}

void USART5_Receive( uintptr_t context )
{
  USART5_IsReceived = 1;
}

int main (void)
{   …  }
```

Lab10 SERCOM - UART Tx/Rx Interrupt

Step 4

a **Modify TC3 Callback for UART Tx**

```
void TC3_TimerExpired(TC_TIMER_STATUS status, uintptr_t context)
{
      if( status & TC_INTFLAG_OVF_Msk )
      {
         LED1_Toggle();

         // TODO 10.03
         // SERCOM5_USART_Write( "Hello world!\r\n", 14 );
         sprintf((char *) USARTRTxBuffer, "Hello world!\r\n" );
         SERCOM5_USART_Write( USARTRTxBuffer, strlen((char*)USARTRTxBuffer) );
      }
}

int main (void)
{
   SYS_Initialize ( NULL );

   ...
}
```

b **Program firmware to target board then observe result.**

Lab10 SERCOM - UART Tx/Rx Interrupt

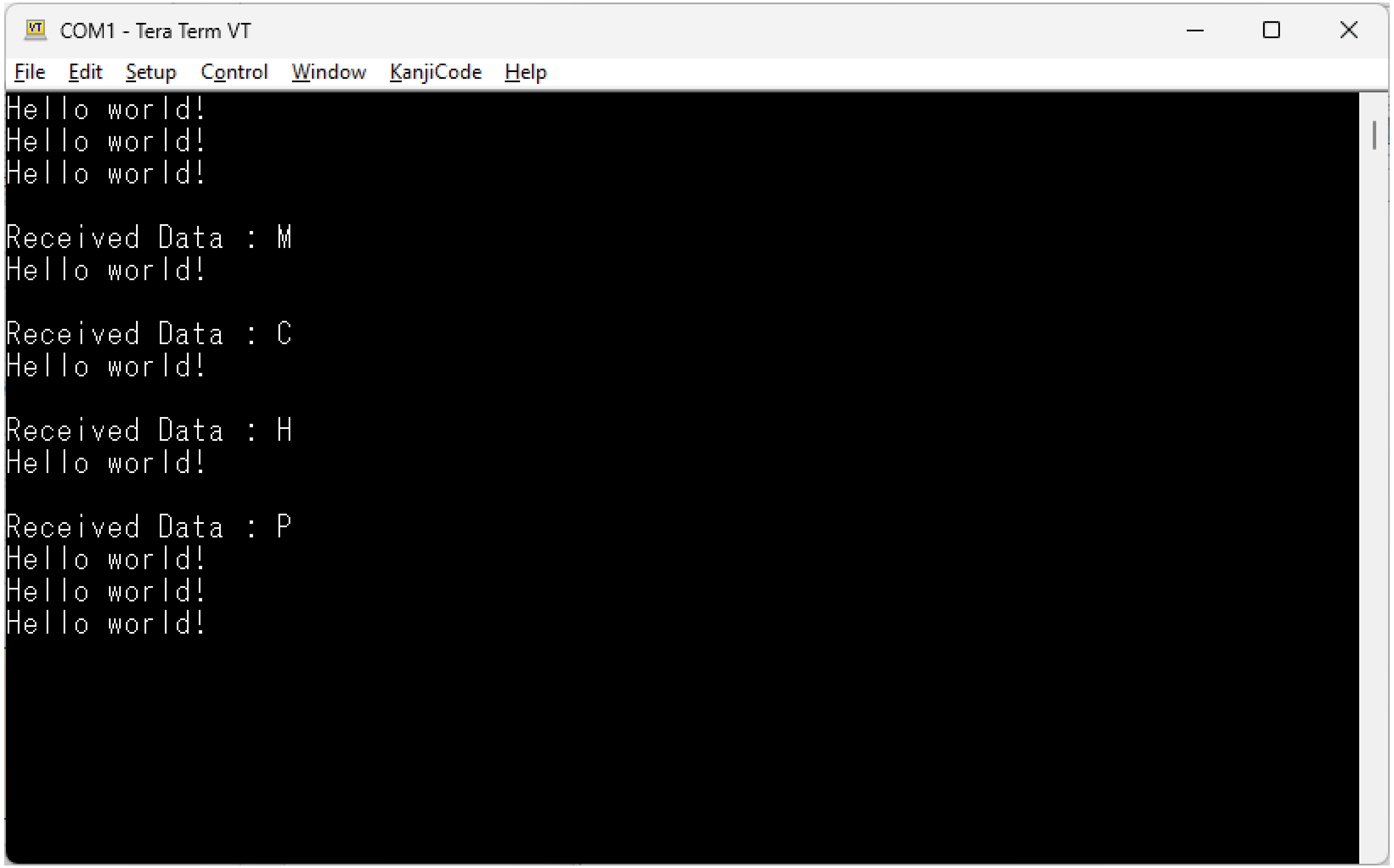
Step 5

a Add code segment to your main loop

```
int main (void)
{
  …
  // TODO 10.04
  SERCOM5_USART_ReadCallbackRegister( USART5_Receive, (uintptr_t )NULL );
  SERCOM5_USART_WriteCallbackRegister( USART5_Transmit, (uintptr_t )NULL );
  SERCOM5_USART_Read( USART5_ReceiveData, 1 );
  while( true )
  {
    …
    // TODO 10.05
    if( USART5_IsReceived )
    {
      USART5_IsReceived = 0;
      sprintf((char *) USARTRTxBuffer,
                      "\r\nReceived Data : %1c\r\n", USART5_ReceiveData[0] );
      SERCOM5_USART_Write( USARTRTxBuffer, strlen((char*)USARTRTxBuffer) );
      while( SERCOM5_USART_WriteIsBusy() );
      SERCOM5_USART_Read( USART5_ReceiveData, 1 );
    }
  }
}
```

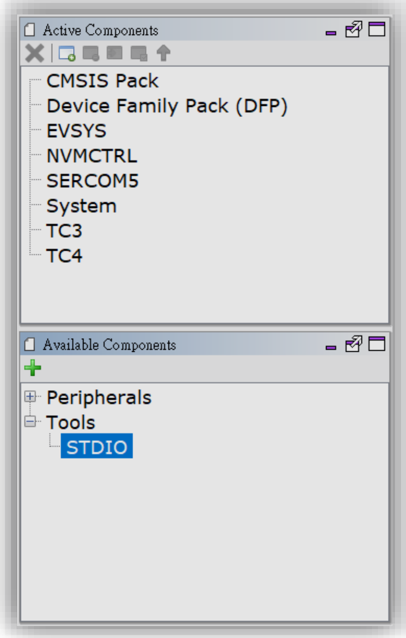
Lab10 SERCOM - UART Tx/Rx Interrupt

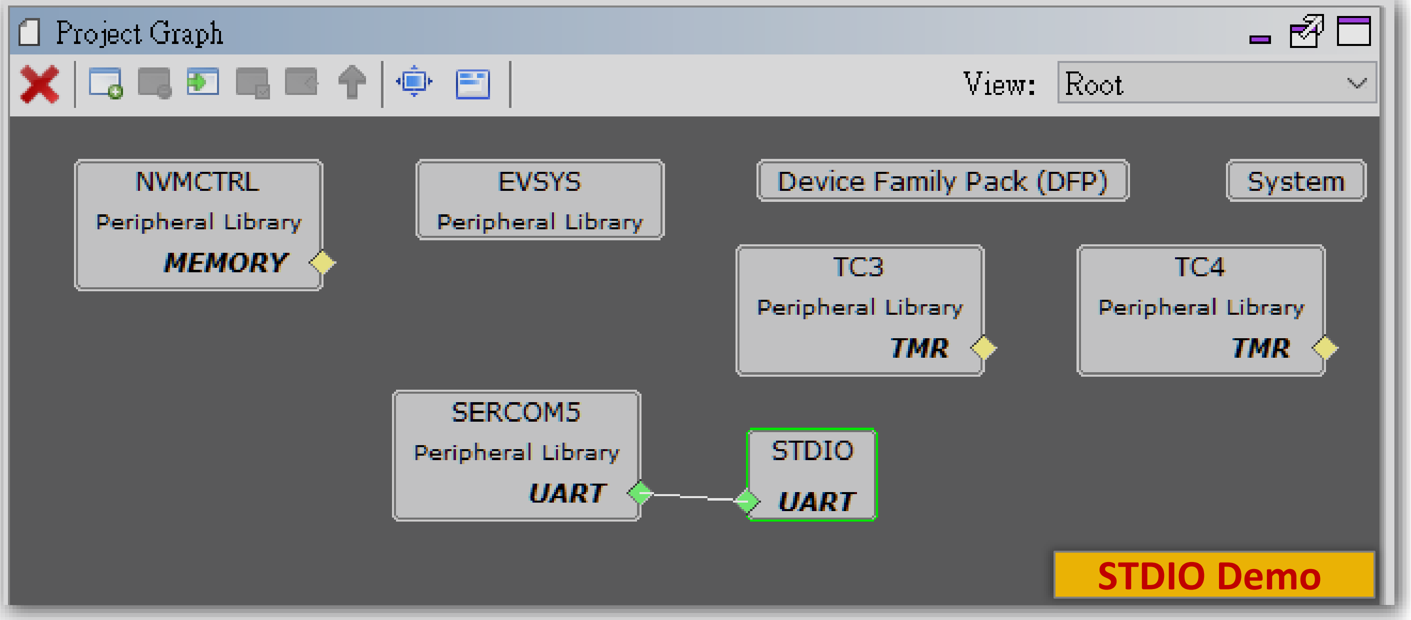
Result



The STDIO limitation

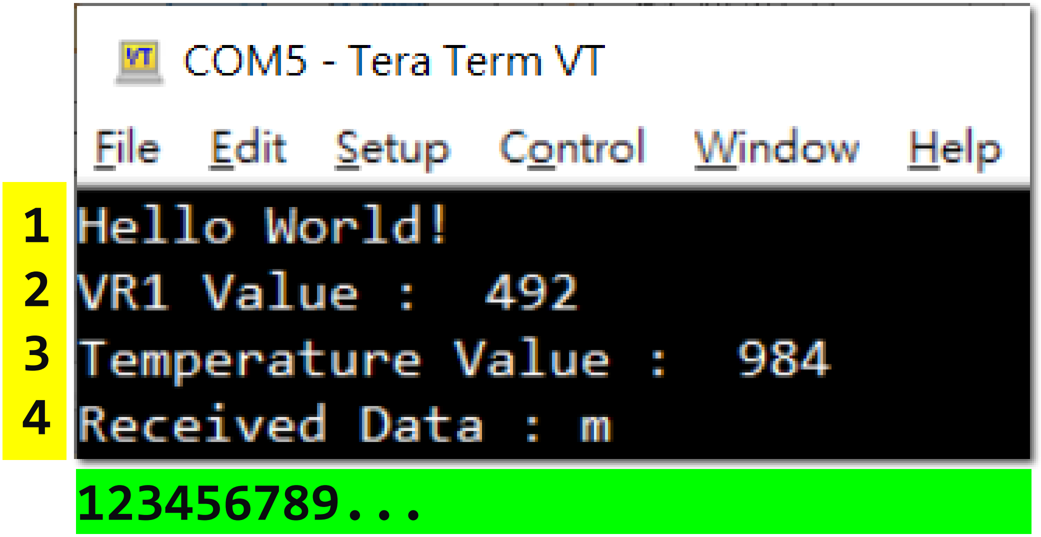
- Harmony supports standard input/output (STDIO), and it is easy to use printf() function to output UART messages. But the mechanism of STDIO module is based on polling mode (blocking mode). It will conflict when selecting SERCOM-UART Non-blocking mode and trying to use printf().
- Let's try to implement our myprintf() function to use interrupt method but have convenient use as the C printf() function.





VT100 Console Control

- Some terminals support VT100 commands and can perform some actions.
- We can use VT100 command to control Tera Term UART output as




```
myprintf("\033[2J");    // Clear screen
myprintf("\033[?25l"); // Hide cursor
myprintf("\033[y;xH"); // Set cursor to (x,y)
                       // Left-up corner is from (1,1)
```

Lab11 SERCOM - UART Tx/Rx Interrupt with myprintf

- Try add myprintf() function code for user code.
- After done this, you could use myprintf() function to instead of USART interface routines.
- Try adding the VT100 command to make the displayed information easier to view.
  SERCOM - UART setting same as the previous Lab.

- **Let's go!**

Lab11 SERCOM - UART Tx/Rx Interrupt with myprintf

Step 1

a Add code segment to user code.

```
// TODO 11.01
#include <stdarg.h>

// TODO 11.02
#define SYS_CONSOLE_PRINT_BUFFER_SIZE 200
static char consolePrintBuffer[SYS_CONSOLE_PRINT_BUFFER_SIZE];
volatile uint8_t TC3_HasExpired= 0;
volatile uint8_t TC4_HasExpired= 0;
…
// TODO 11.03
void myprintf(const char *format, ...)
{
    size_t len = 0;
    va_list args = {0};
    va_start(args, format);
    len = vsnprintf(consolePrintBuffer, SYS_CONSOLE_PRINT_BUFFER_SIZE, format , args);
    va_end(args);

    if ((len > 0) && (len < SYS_CONSOLE_PRINT_BUFFER_SIZE))
    {
        consolePrintBuffer[len] = '\0';
        SERCOM5_USART_Write(consolePrintBuffer, len);
        while (SERCOM5_USART_WriteIsBusy());
    }
}
```

Lab11 SERCOM - UART Tx/Rx Interrupt with myprintf

Step 2

a Add software flag control of TC3 and TC4.

```
...
void TC3_TimerExpired(TC_TIMER_STATUS status, uintptr_t context)
{
   if( status & TC_INTFLAG_OVF_Msk )
   {
      // TODO 11.04
       TC3_HasExpired= 1;
     //  LED1_Toggle();
     //  sprintf((char *) USARTRTxBuffer, "Hello world!\r\n" );
     //  SERCOM5_USART_Write( USARTRTxBuffer, strlen((char*)USARTRTxBuffer) );
   }
}
void TC4_TimerExpired(TC_TIMER_STATUS status, uintptr_t context)
{
   if( status & TC_INTFLAG_OVF_Msk )
   {
      // TODO 11.05
       TC4_HasExpired= 1;
     //  LED2_Toggle();
   }
}
...
```

Lab11 SERCOM - UART Tx/Rx Interrupt with myprintf

Step 3

a Add code segment to your main while loop.

```
int main ( void )
{
  …
  // TODO 11.06
  myprintf("\033[2J");
  myprintf("\033[?25l");
  while( true )
  {
    // TODO 11.07
    if (TC3_HasExpired)
    {
      TC3_HasExpired= 0;
      LED1_Toggle();
      myprintf( "\033[1;1HHello World!" );
    }

    if (TC4_HasExpired)
    {
      TC4_HasExpired= 0;
      LED2_Toggle();
    }
  }
}
```

Lab11 SERCOM - UART Tx/Rx Interrupt with myprintf

Step 4

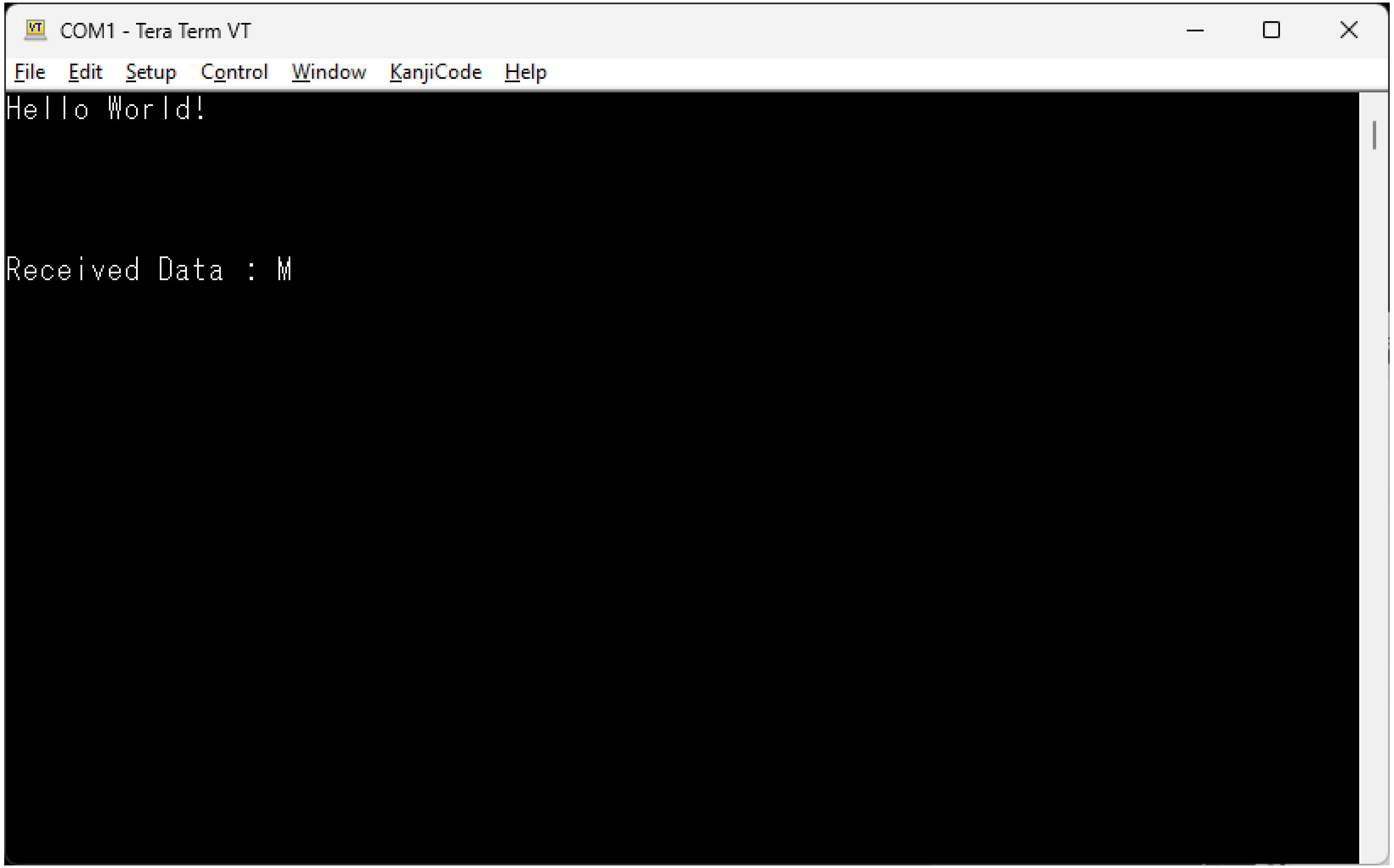
a Add code segment to your main while loop.

```
int main ( void )
{
   …
   while( true )
   {
      …

      if( USART5_IsReceived )
      {
          USART5_IsReceived = 0;

          // TODO 11.08
         // sprintf((char *) USARTRTxBuffer, "\r\nReceived Data : %1c\r\n", USART5_ReceiveData[0] );
         // SERCOM5_USART_Write( USARTRTxBuffer, strlen((char*)USARTRTxBuffer) );
         // while( SERCOM5_USART_WriteIsBusy() );
          myprintf("\033[6;1HReceived Data : %1c", USART5_ReceiveData[0] );
          SERCOM5_USART_Read( USART5_ReceiveData, 1 );
      }
    }
}
```

MICROCHIP

Lab11 SERCOM - UART Tx/Rx Interrupt with myprintf

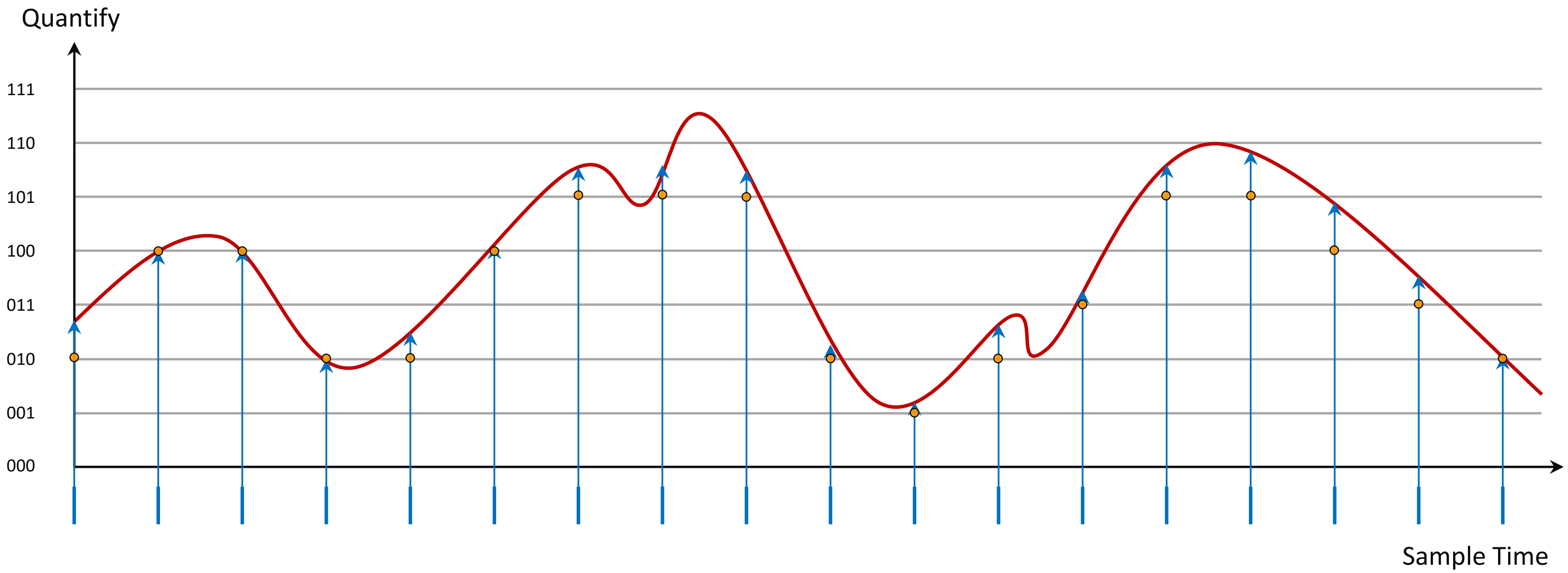
Result



High Speed ADC Architecture

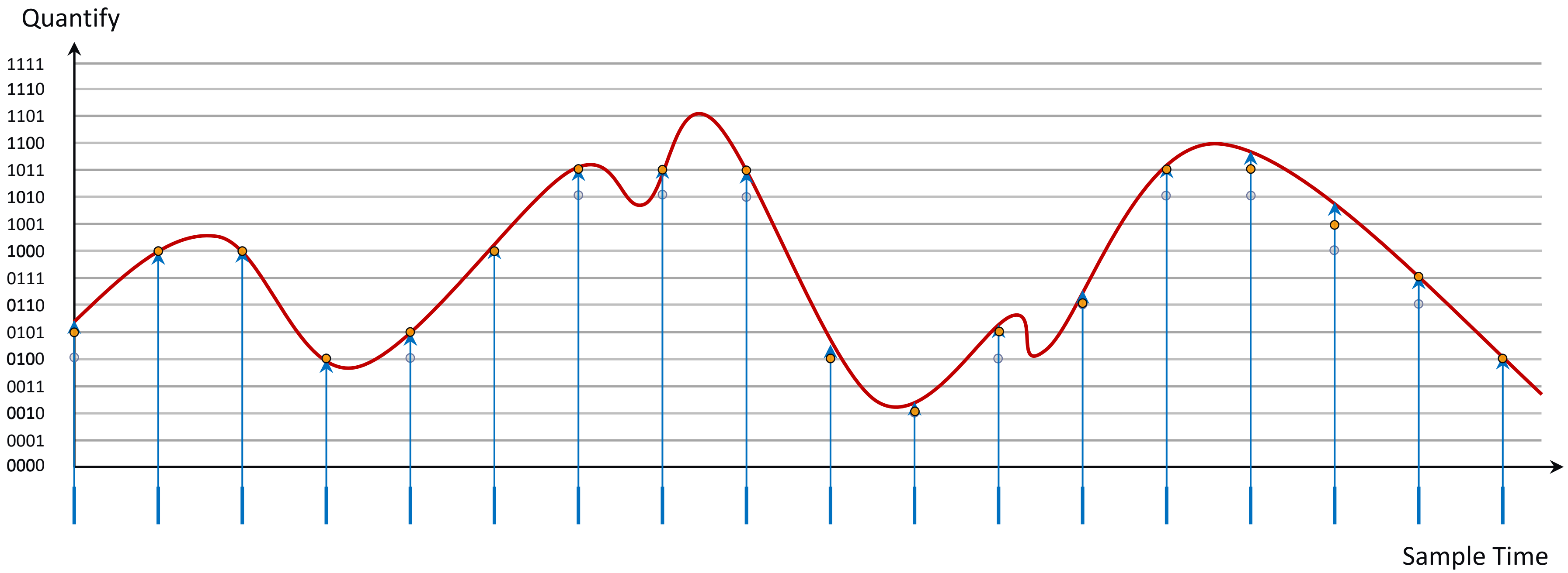
What's ADC ?

- **The Analog-to-Digital Converter (ADC) converts a signal from the time continuous /amplitude continuous domain into a time discrete/amplitude discrete domain.**



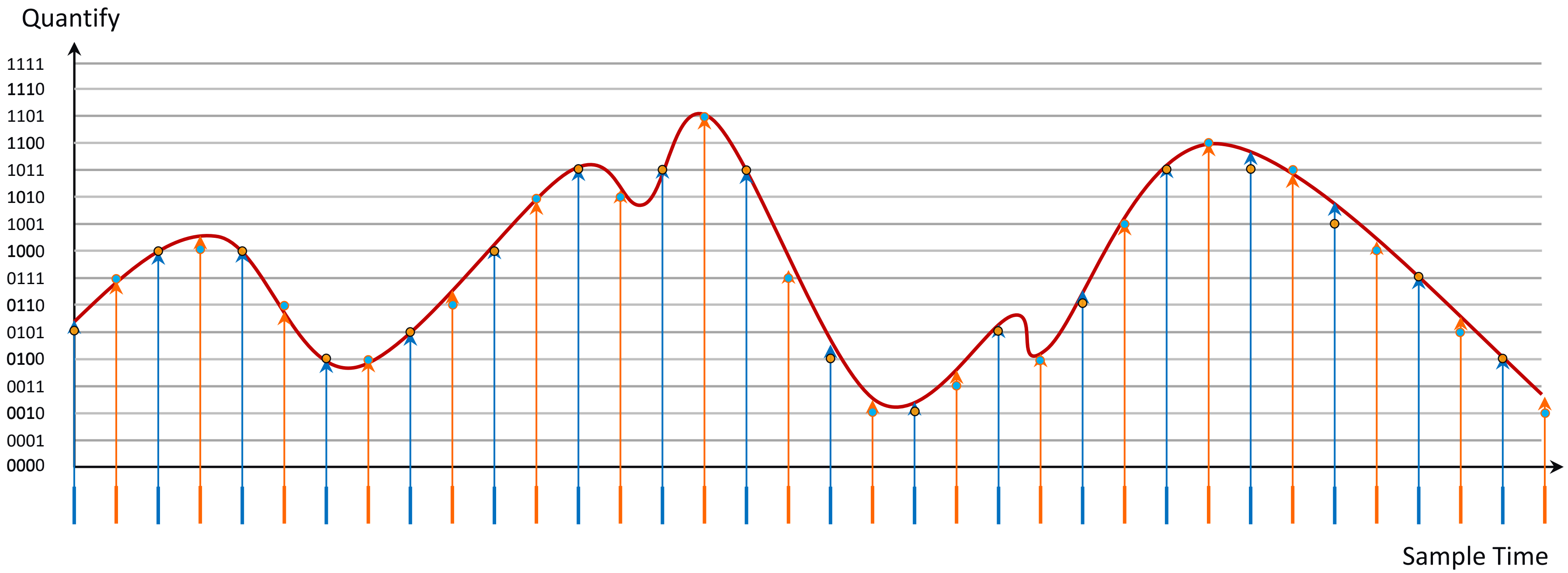
Resolution

- **The resolution of the converter indicates the number of discrete values. The resolution determines the magnitude of the quantization error. (wiki)**



Sample Rate

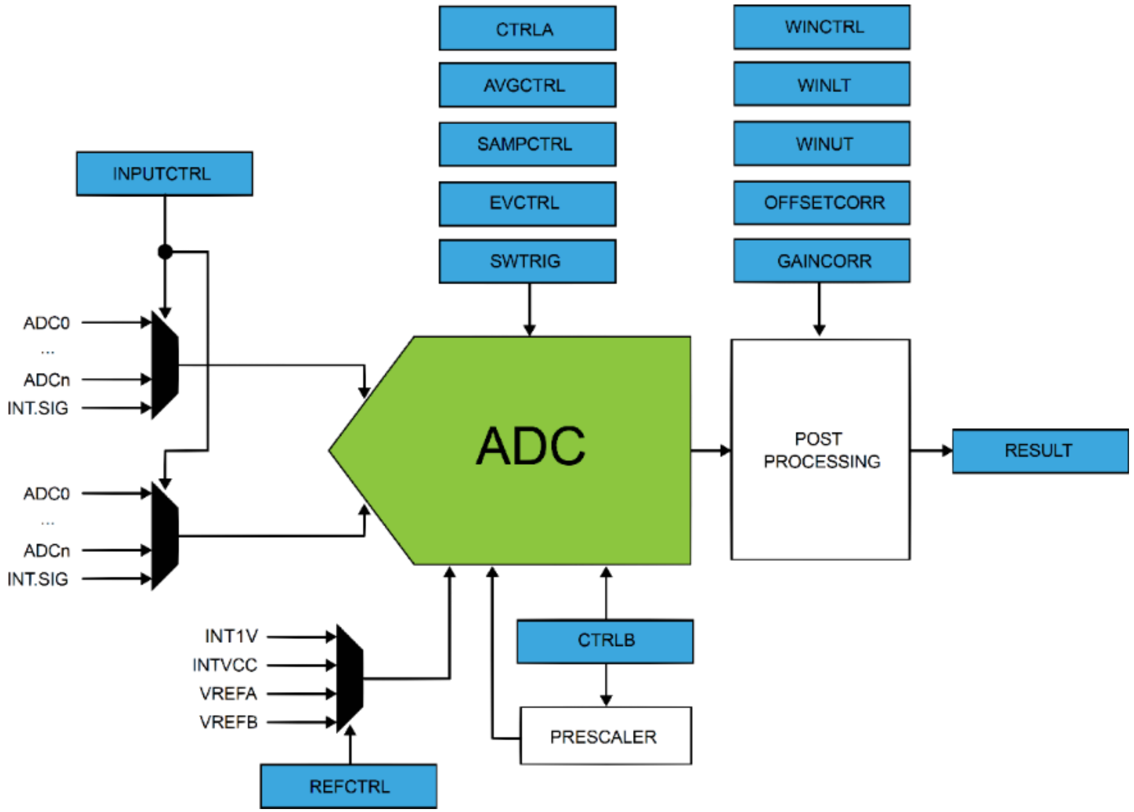
- The analog signal is required to define the rate at which new digital values are sampled from the analog signal.
- This faithful reproduction is only possible if the sampling rate is higher than twice the highest frequency of the signal. (wiki)



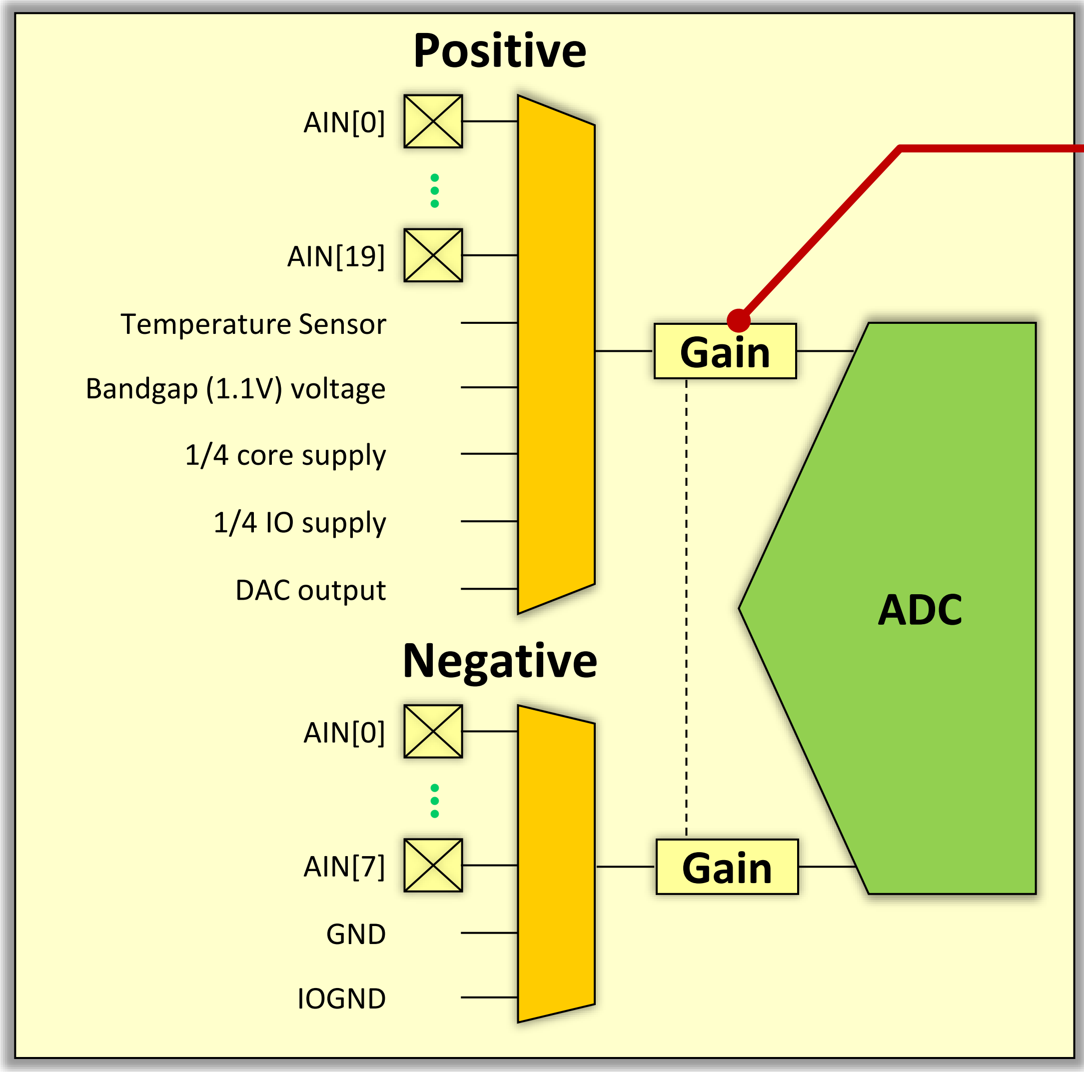
ADC Block Diagram

- Up to 32 analog input, 8, 10 or 12 bits resolution.
- Differential and single-ended inputs.
- 1/2x to 16x gain. Up to 350ksps.
- Support Event-triggered and pin scan.
- Support averaging and oversampling.
- External or Internal reference voltage.
- Oversample accumulation average and window monitor feature.

$$Reusult_{AD} = \left(\frac{AN_{+} - AN_{-}}{V_{Ref} - AN_{-}} \times 2^{n}\right) - 1$$

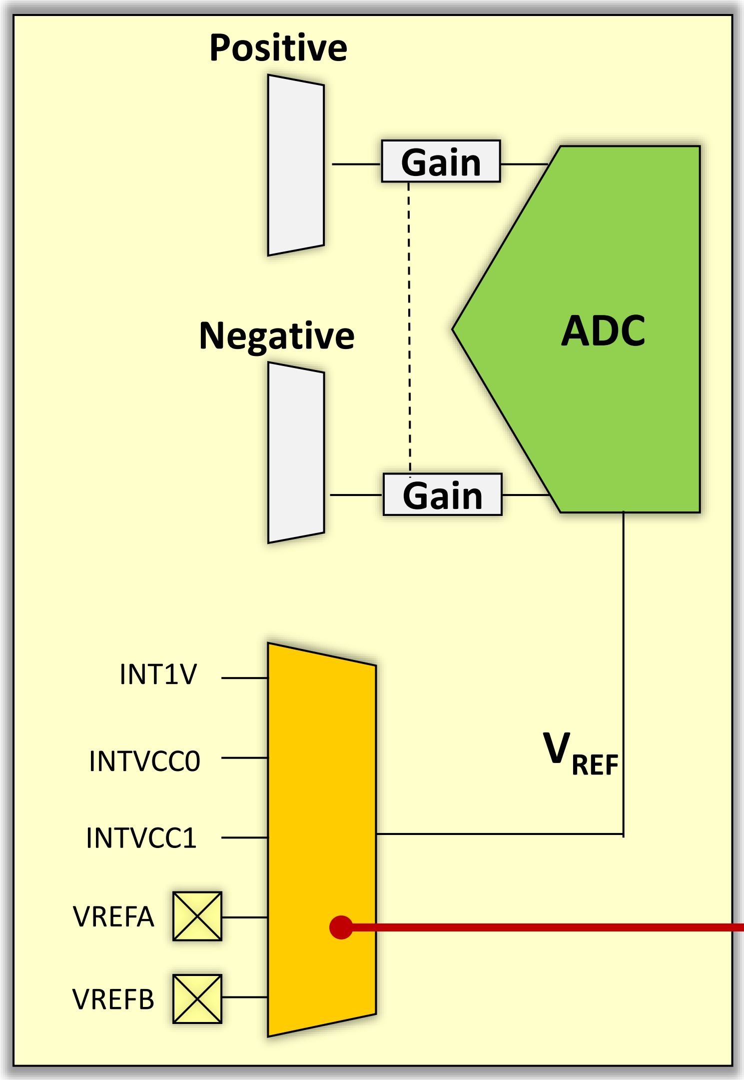


ADC Channel Selection & Gain



GAIN[3:0]	Name
0x0	1X
0x1	2X
0x2	4X
0x3	8X
0x4	16X
0x5-0xE	Reserved
0xF	1/2x

ADC Reference Voltage Selection



VREFA and VREFB range

1.0V ~ (V_{DDANA} - 0.6V)

Conversion range:

– Single-ended mode

0V ~ V_{REF}/Gain

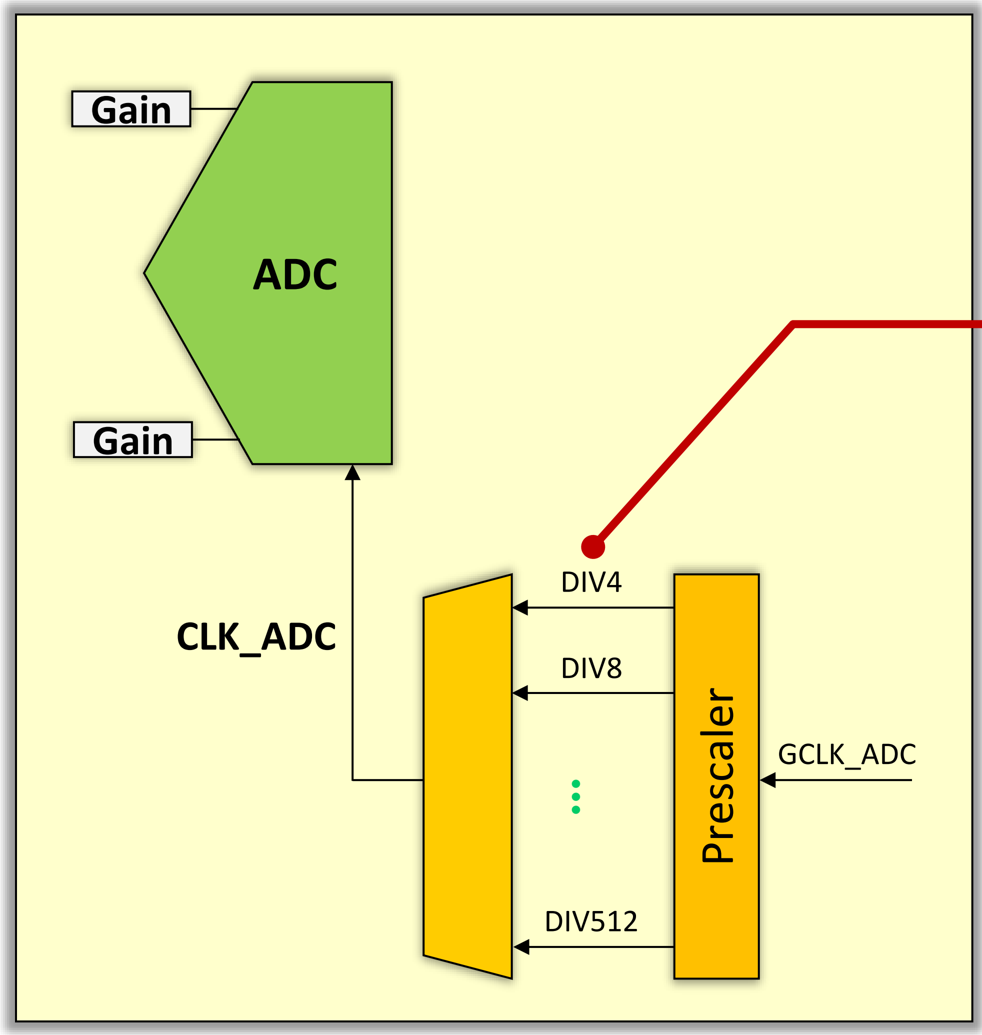
– Differential mode

-V_{REF}/Gain ~ V_{REF}/Gain

REFSEL[3:0]	Name	Description
0x0	INT1V	1.0V voltage reference
0x1	INTVCC0	1/1.48 VDDANA
0x2	INTVCC1	1/2 VDDANA (only for VDDANA > 2.0V)
0x3	VREFA	External reference
0x4	VREFB	External reference
0x5-0xF		Reserved

VDDANA = 3.3V
½ VDDANA = 1.65V

ADC Prescaler



PRESCALER[2:0]	Name
0x0	DIV4
0x1	DIV8
0x2	DIV16
0x3	DIV32
0x4	DIV64
0x5	DIV128
0x6	DIV256
0x7	DIV512

MICROCHIP

ADC Trigger Source

- **There are three way to be started data conversion.**
  - **Software Trigger**

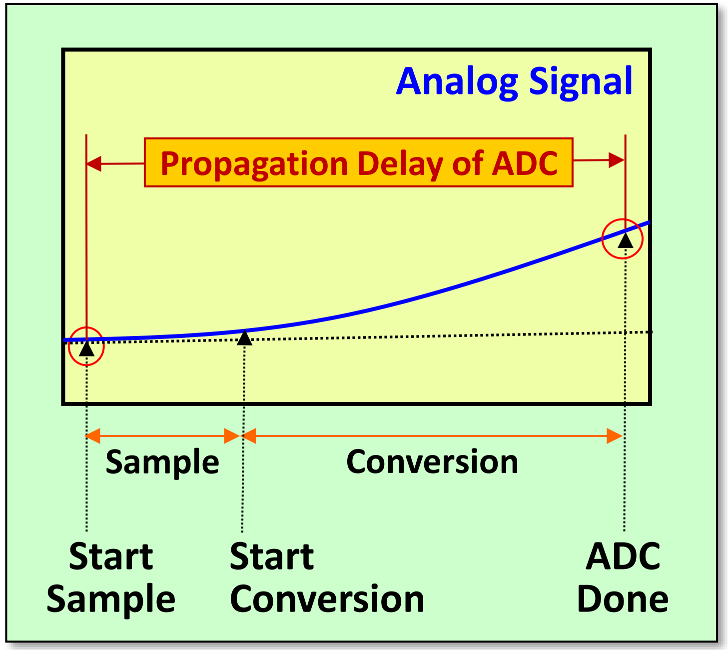
    One shot conversion control via software.
  - **Free Run**

    A free-running mode can be used to continuously convert an input channel. When using free-running mode the first conversion must be started, while subsequent conversions will start automatically at the end of previous conversions.
  - **HW Event Trigger**

    Through Event System peripheral to automatically conversion.

ADC Characteristics

- **Maximum Sampling Rate :**
  Single-shot : **2.1M/Round(6+0.5) = 300k samples/second.**
  Free-running : **2.1M/6 = up to 350k samples/second.**

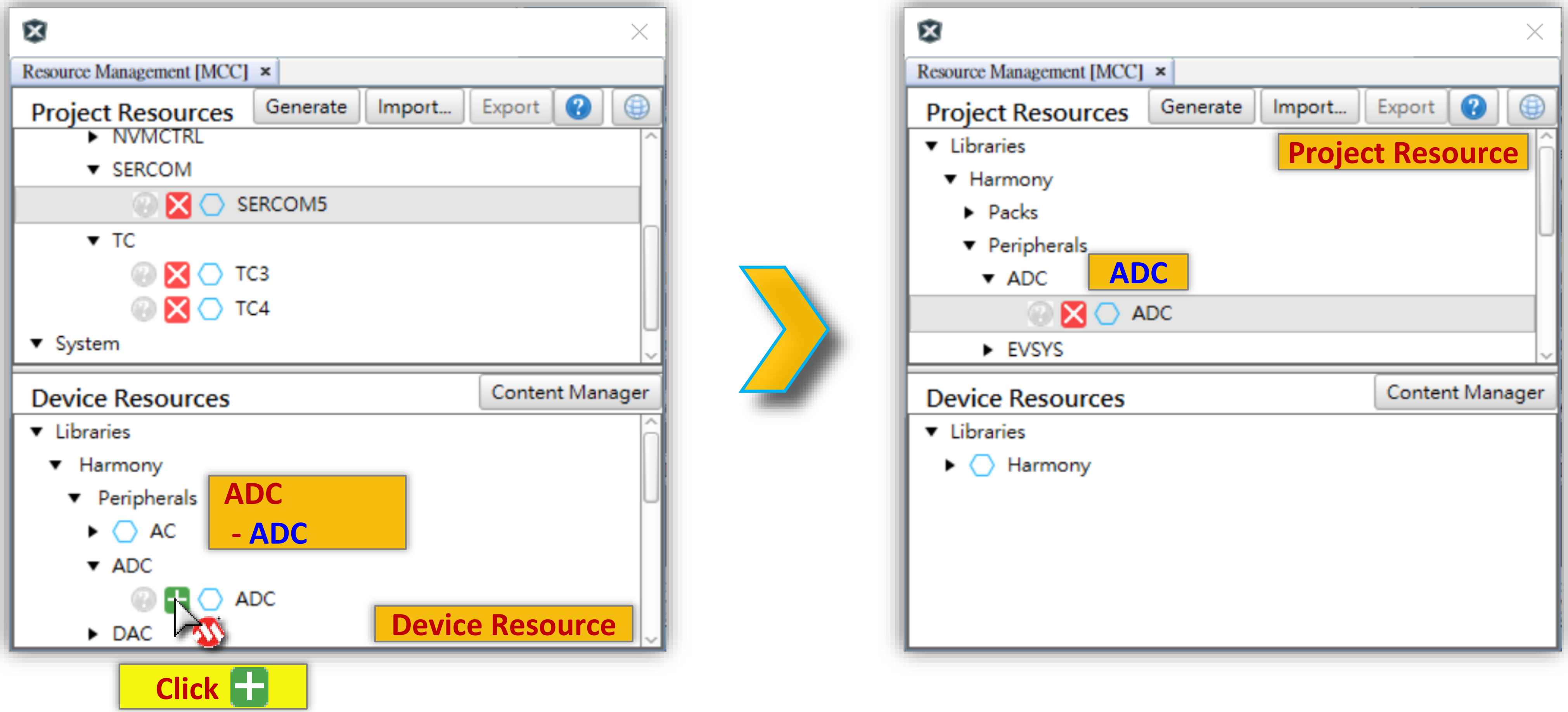


Symbol	Parameter	Conditions	Min.	Max.	Units
RES	Resolution		8	12	bits
f_{CLK_ADC}	ADC Clock frequency		30	2100	kHz
	Sample rate	Single shot	5	300	ksps
		Free running	5	350	ksps
	Sampling time		0.5	-	cycles
	Conversion time	1x Gain	6	-	cycles

MICROCHIP

Add ADC Function using MCC Harmony

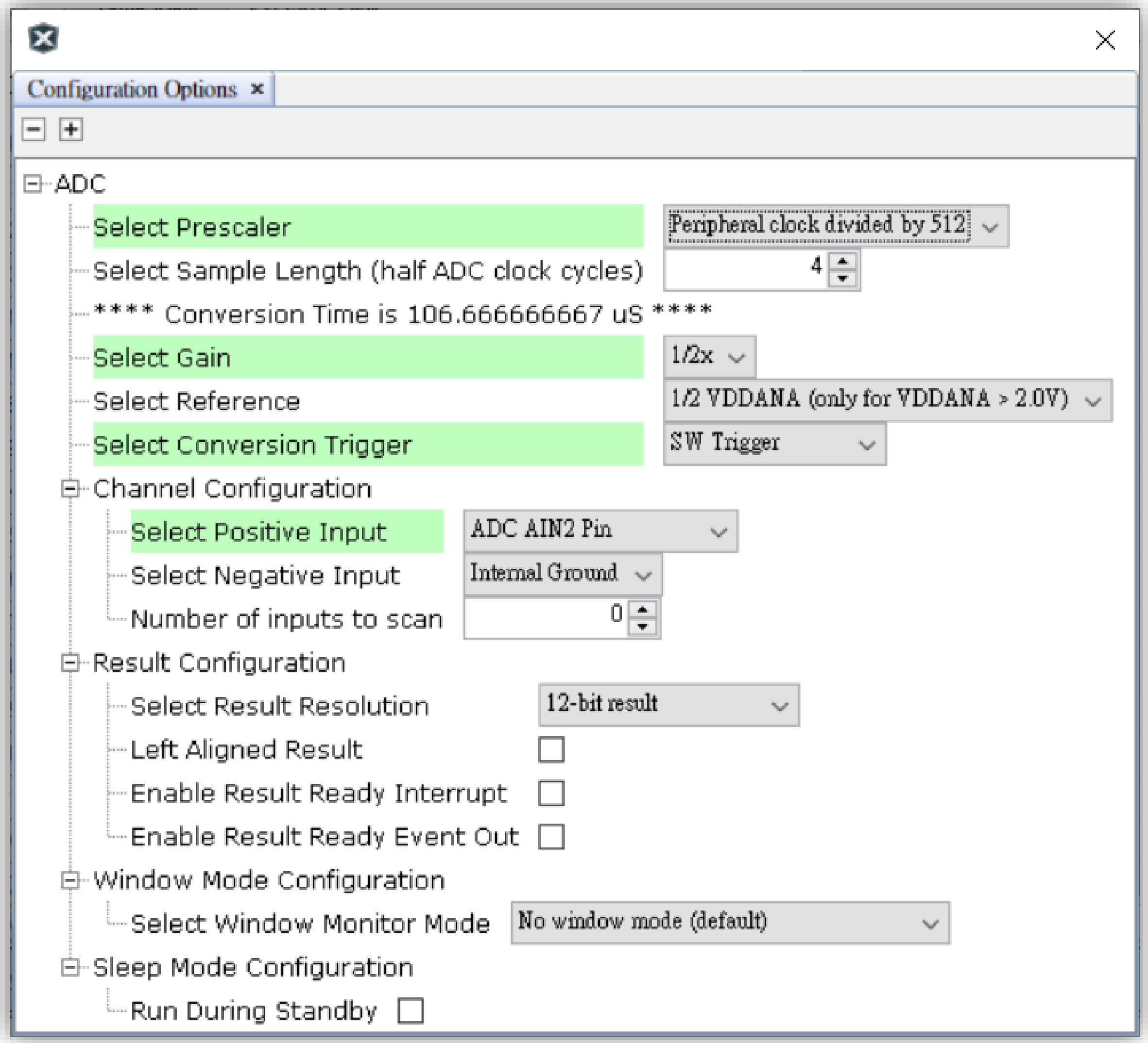
- Find ADC component in Device Resource window.
- Click ➕ to add ADC to Project Resource window.



MICROCHIP

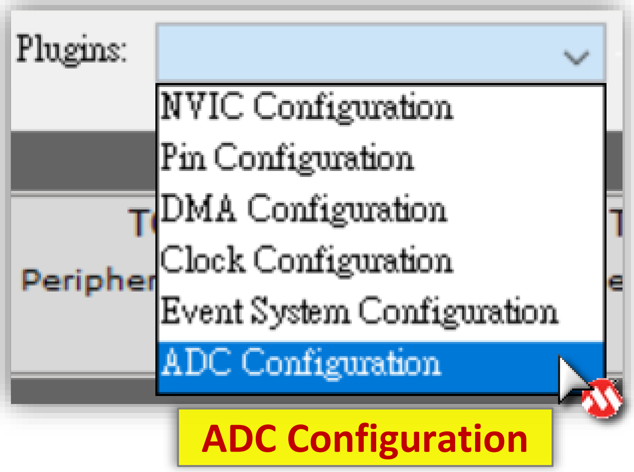
ADC Polling

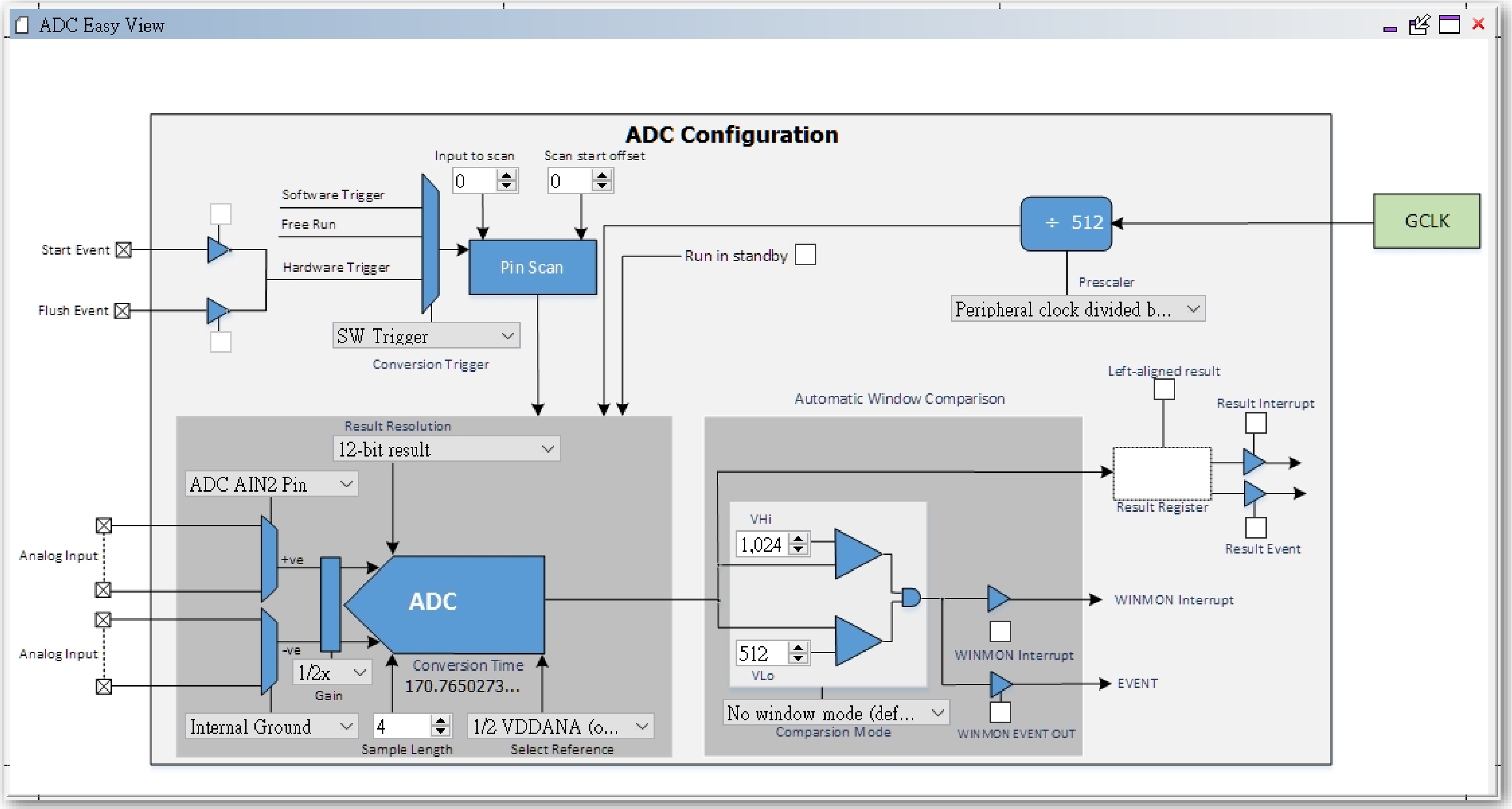
Configuration Options Example



ADC Polling

Easy View Configuration Example





ADC Polling

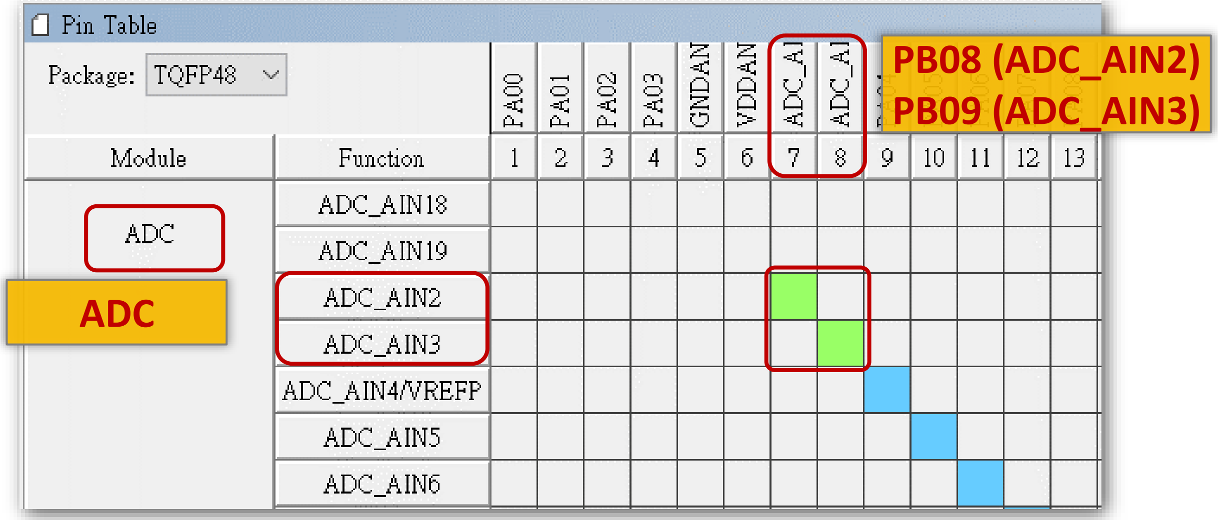
PINMUX Configuration Example

- Analog pins will all belong to PINMUX function group B.
- For example : PB08 as AIN[2] (Analog input Channel 2)
  PB09 as AIN[3] (Analog input Channel 3)

Pin[1]			I/O Pin	Supply	A	B[2][3]					C	D	E	F	G	H
SAMD21E	SAMD21G	SAMD21J			EIC	REF	ADC	AC	PTC	DAC	SERCOM[2][3]	SERCOM-ALT	TC[4] /TCC	TCC	COM	AC/ GCLK
	7	11	PB08	VDDANA	EXTINT[8]		AIN[2]		Y[14]			SERCOM4/ PAD[0]	TC4/WO[0]	TCC3/ WO[6]		
	8		PB09	VDDANA	EXTINT[9]		AIN[3]					SERCOM4/ PAD[1]	TC4/WO[1]	TCC3/ WO[7]		

PB08
PB09

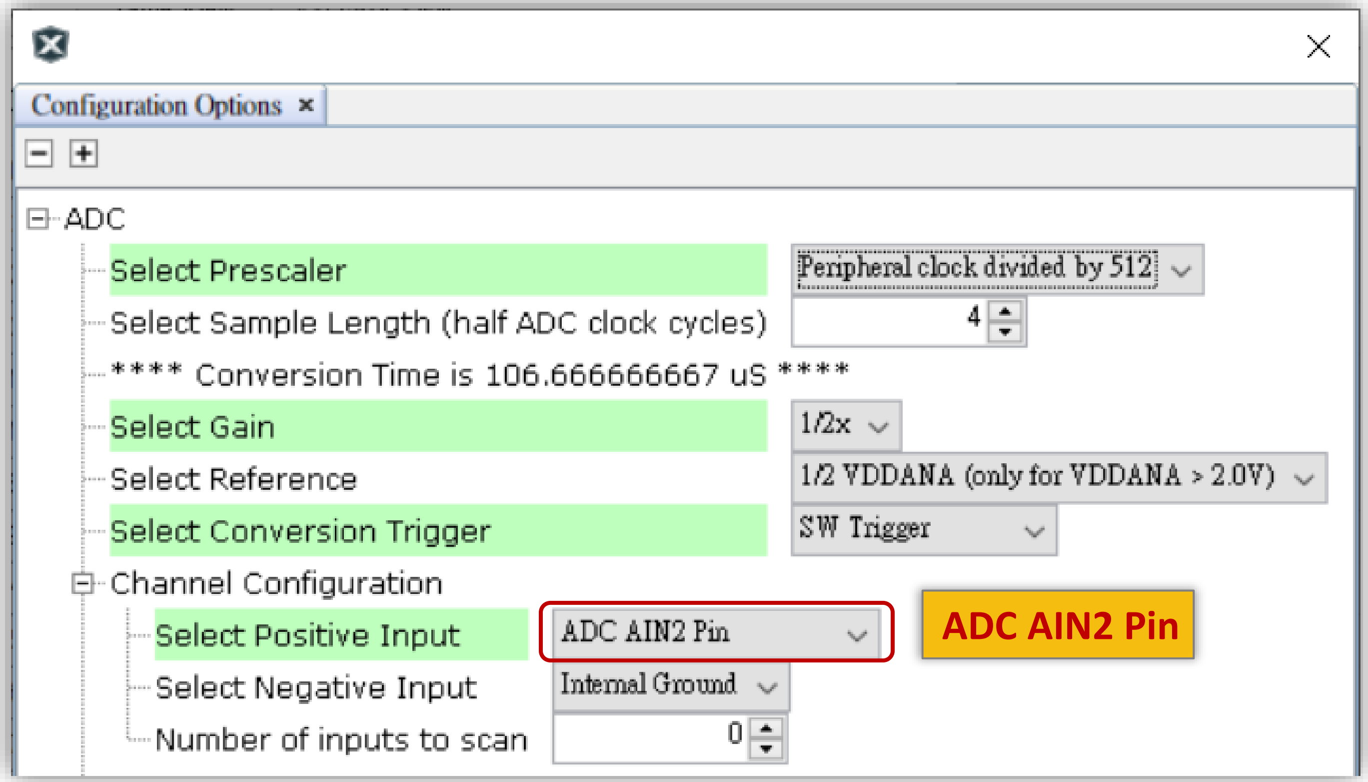
AIN[2]
AIN[3]



ADC Polling

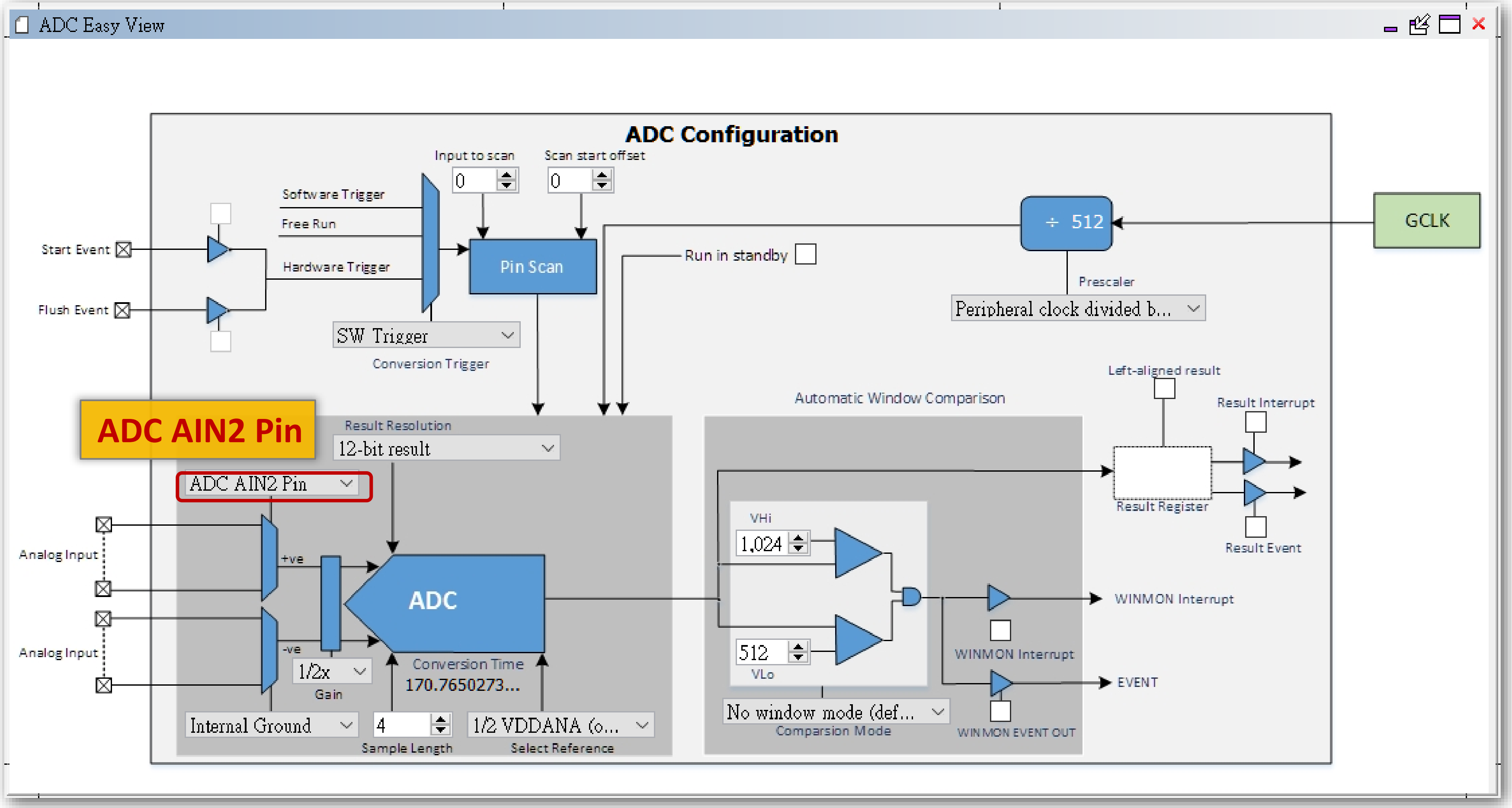
PINMUX Configuration Example

- Remember assign Positive Input pin in ADC module.
- For example : ADC AIN2 Pin as Positive Input Pin (PB08)



ADC Polling

Easy View PINMUX Configuration Example



MICROCHIP

ADC Polling

Interface Routines

```
void ADC_Initialize( void );
void ADC_Enable( void );
void ADC_Disable( void );
void ADC_ChannelSelect( ADC_POSINPUT positiveInput, ADC_NEGINPUT negativeInput );
void ADC_ConversionStart( void );
uint16_t ADC_ConversionResultGet( void );
bool ADC_ConversionStatusGet( void );
void ADC_ComparisonWindowSet(uint16_t low_threshold, uint16_t high_threshold);
void ADC_WindowModeSet(ADC_WINMODE mode);
```

ADC Polling

Software Trigger Code Example

```
…

int main (void)
{
    SYS_Initialize ( NULL );
    …
    ADC_Enable();
    while( true )
    {
        …
        ADC_ConversionStart();
        while( !ADC_ConversionStatusGet() );
        myprintf( "%d\r\n", ADC_ConversionResultGet() );
    }
}
```

ADC Factory Calibration

- **ADC Factory calibration value will auto load and applicate in MCC Harmony generated ADC initial function.**

plib_adc.c

```
void ADC_Initialize( void )
{ …
  /* Write linearity calibration and bias calibration */
  ADC_REGS->ADC_CALIB =
    (uint32_t)( ADC_CALIB_LINEARITY_CAL( adc_linearity0 | (adc_linearity1 << 5) ) ) |
    ADC_CALIB_BIAS_CAL( ( ( (*(uint64_t*)OTP4_ADDR + 1) & ADC_BIASCAL_Msk ) >> ADC_BIASCAL_POS) );
}
```

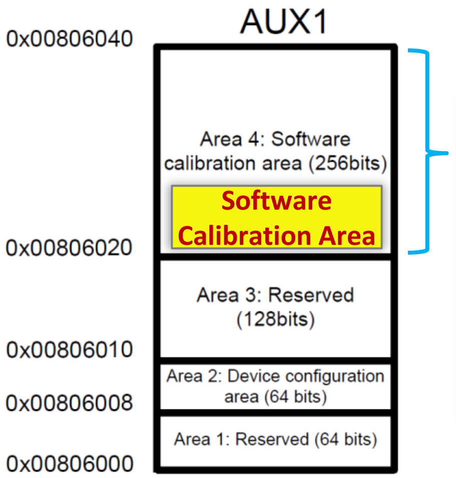


Table 10-5. NVM Software Calibration Area Mapping

Bit Position	Name	Description
34:27	ADC LINEARITY ADC_LINEARITY	ADC Linearity Calibration. Should be written to ADC CALIB register.
37:35	ADC BIASCAL ADC_BIASCAL	ADC Bias Calibration. Should be written to ADC CALIB register.

MICROCHIP

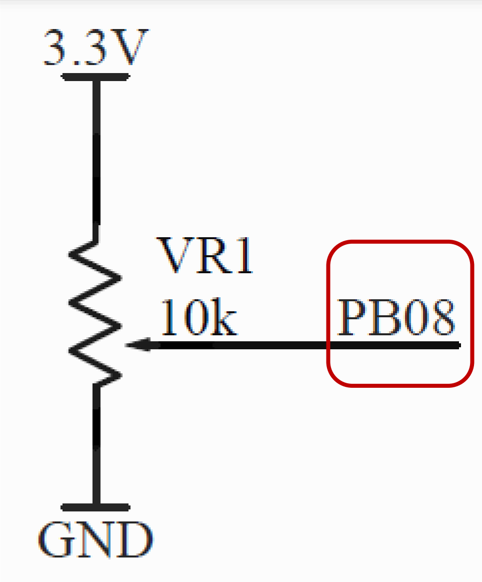
Lab12 ADC Polling Software Trigger

- Try to add ADC function to project.
- Initialize the ADC and get the conversion value from VR1 (AIN[2]) periodically, and then send it to PC by using UART .
- ADC set to Software Trigger, 12 bits resolution, Single-End mode.

- Let’s go！

Lab12 ADC Polling Software Trigger

Pin(1)			I/O Pin	Supply	A	B(2)(3)				
SAMD21E	SAMD21G	SAMD21J			EIC	REF	ADC	AC	PTC	DAC
	7	11	PB08	VDDANA	EXTINT[8]		AIN[2]		Y[14]	
	8	12	PB09	VDDANA	EXTINT[9]		AIN[3]		Y[15]	

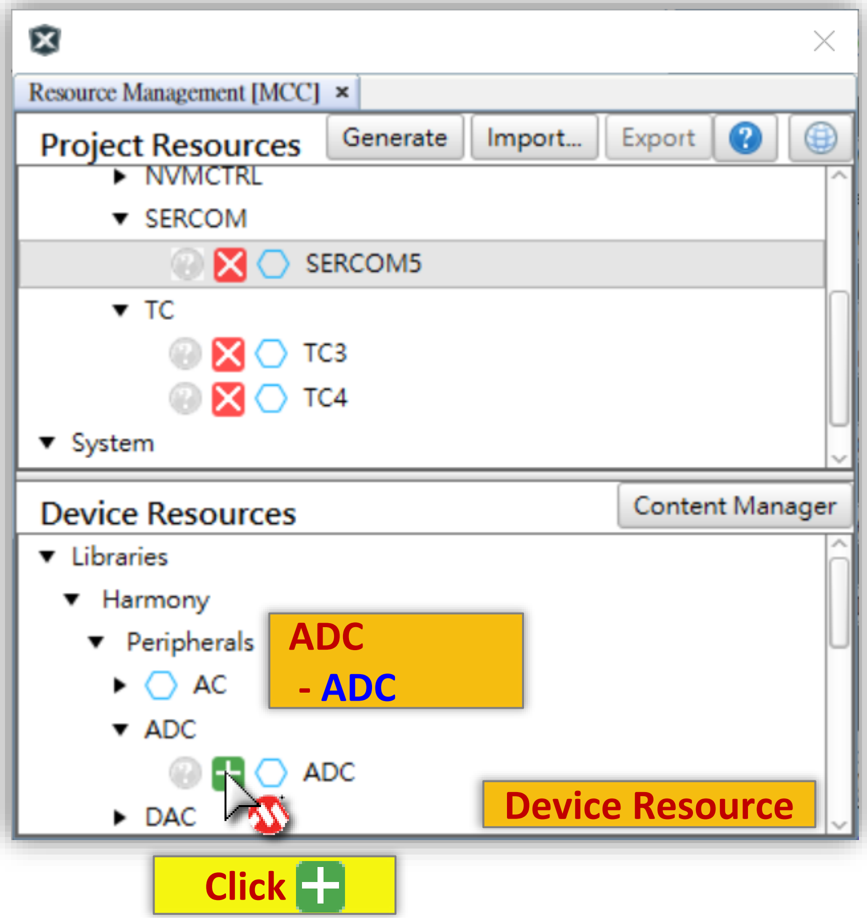


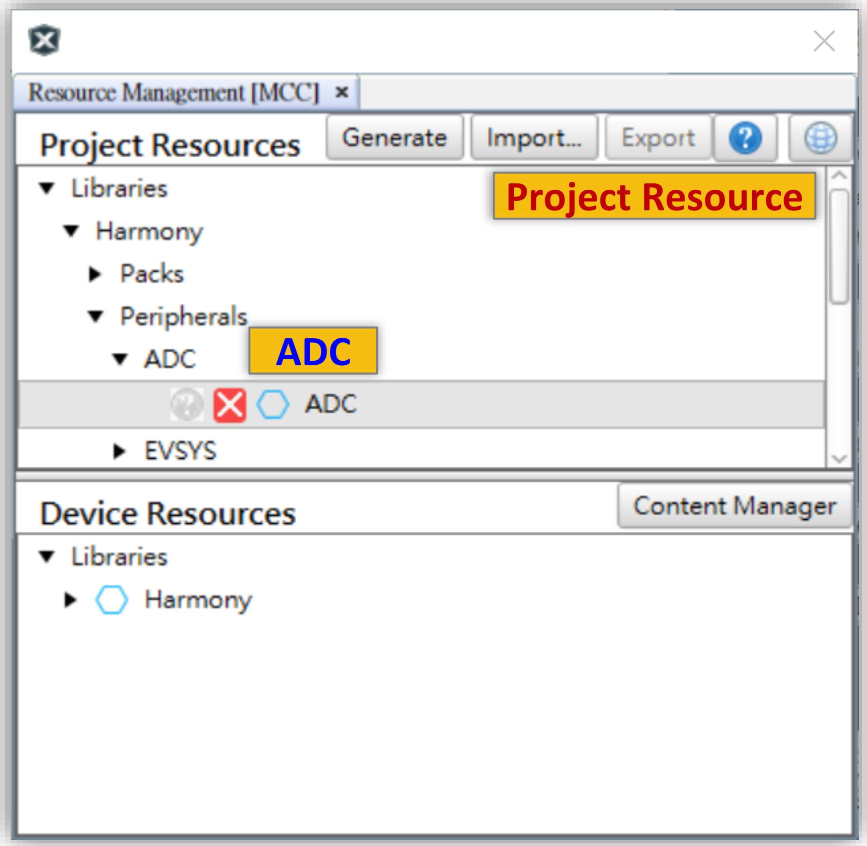
MICROCHIP

Lab12 ADC Polling Software Trigger

Step 1

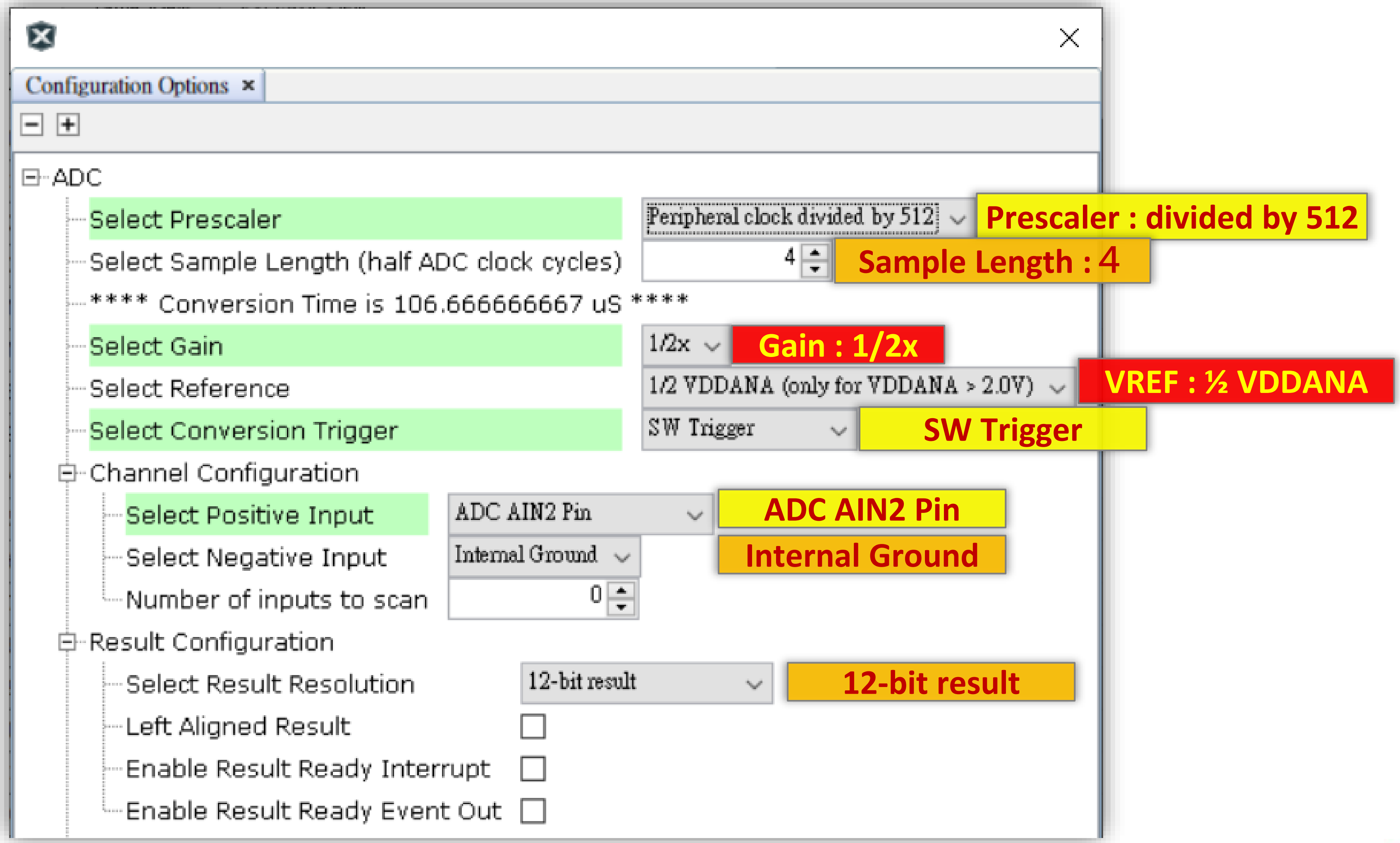
- Find ADC component in Device Resource window.
- Click + to add ADC to Project Resource window.





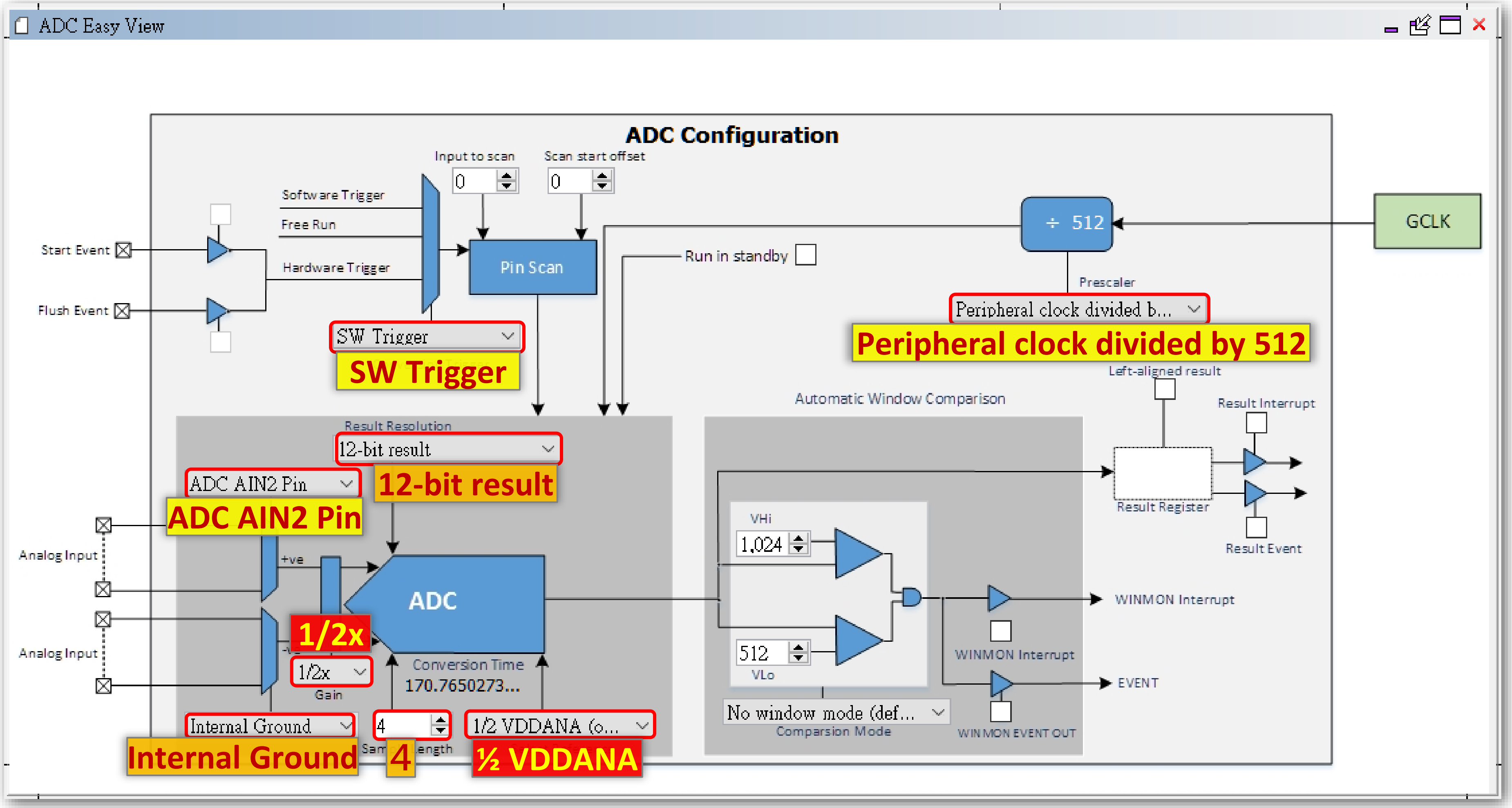
Lab12 ADC Polling Software Trigger

Step 2 (Use Configuration Options)



Lab12 ADC Polling Software Trigger

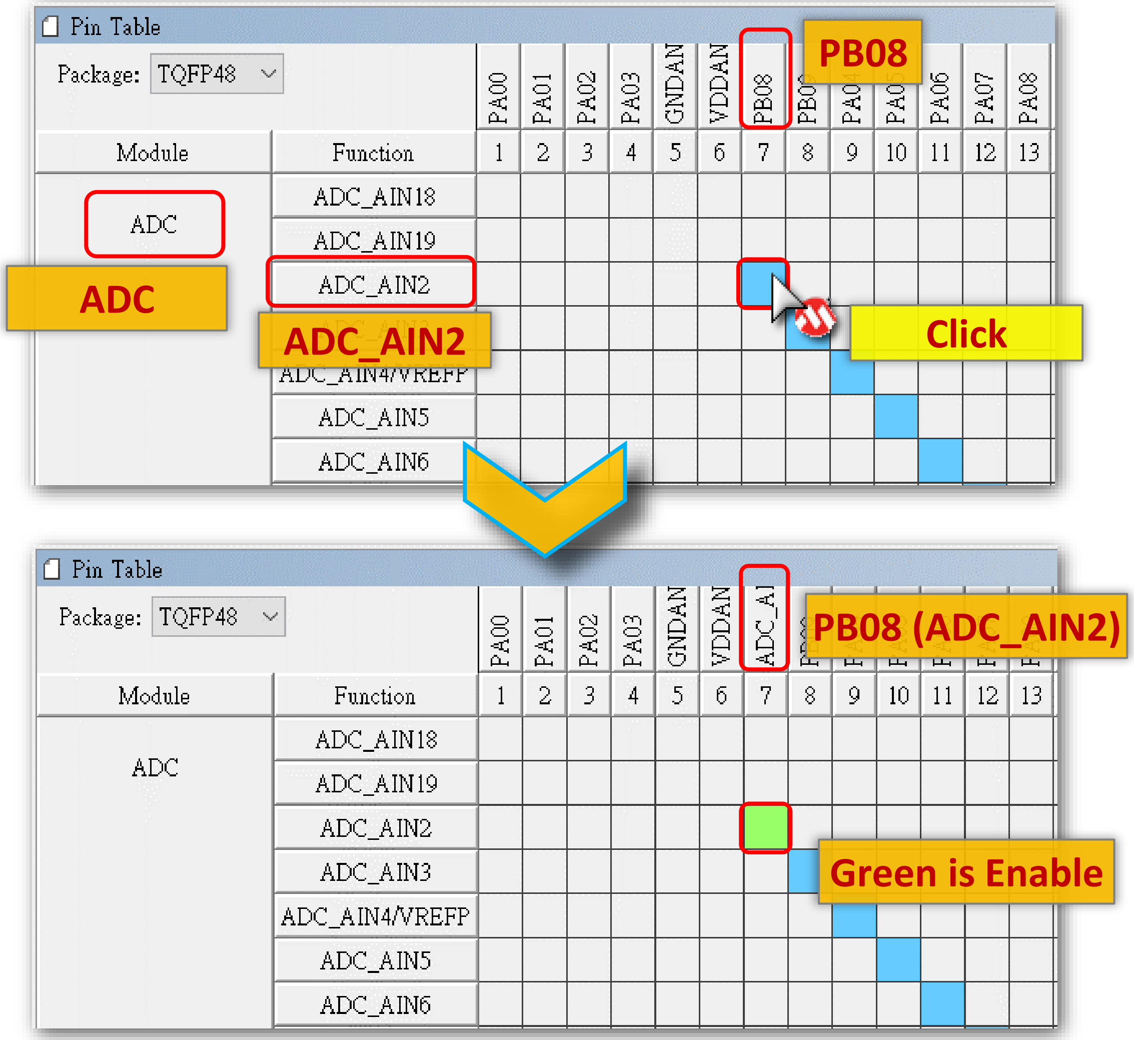
Step 2 (Use Easy View)



MICROCHIP

Lab12 ADC Polling Software Trigger

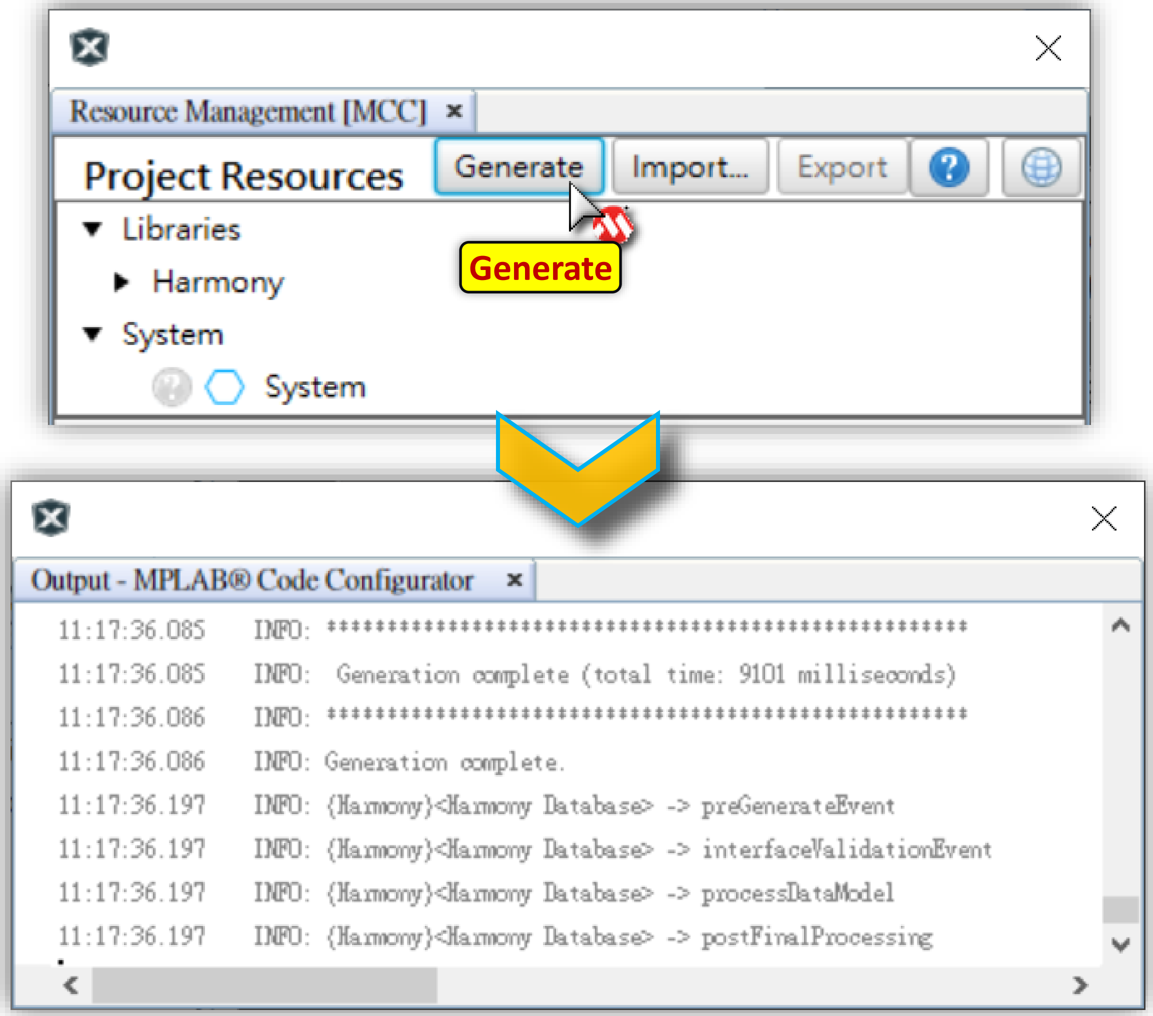
Step 3



Lab12 ADC Polling Software Trigger

Step 4

- Click "Generate" Button to Generate your Code.



Lab12 ADC Polling Software Trigger

Step 5

a Add code segment to your main.c

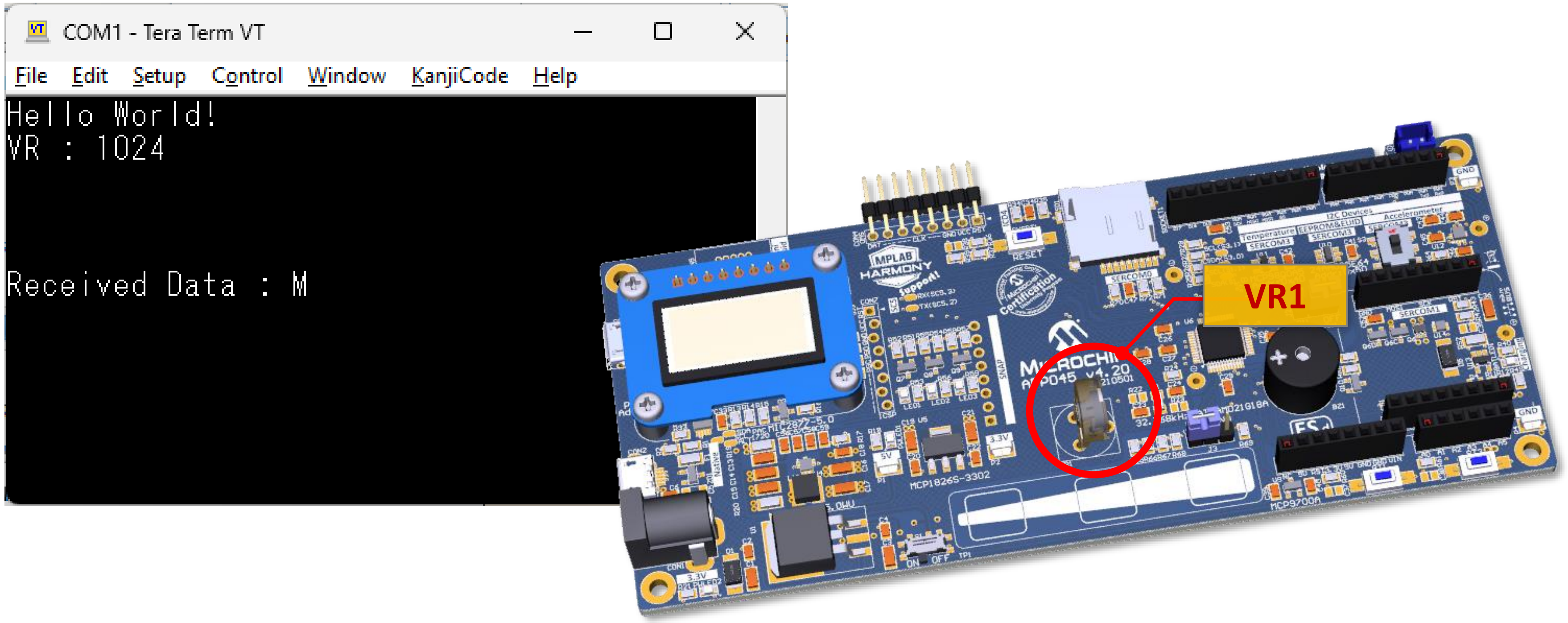
```
// TODO 12.01
uint16_t ADC_Result;

int main (void)
{ ...
   // TODO 12.02
   ADC_Enable();

   while( true )
   { ...
      if( TC4_HasExpired)
      { ...

         // TODO 12.03
         ADC_ConversionStart();
         while( !ADC_ConversionStatusGet() );
         ADC_Result = ADC_ConversionResultGet();
         myprintf("\033[2;1HVR : %4d", ADC_Result );
      }
      ...
   }
}
```

b Program firmware to target board then observe result.

Lab12 ADC Polling Software Trigger

Result

- "Hello World!" string and VR1 ADC value will output to terminal with one second period.



ADC Interrupt and Pin Scan

ADC Pin Scan

- The ADC support analog channel switch call Pin Scan function.
- Pin scan switch analog channel sequential after a conversion automatically.
- Start Pin : Positive Input + Scan Start Offset
- End Pin : Start Pin + Number of Inputs to Scan

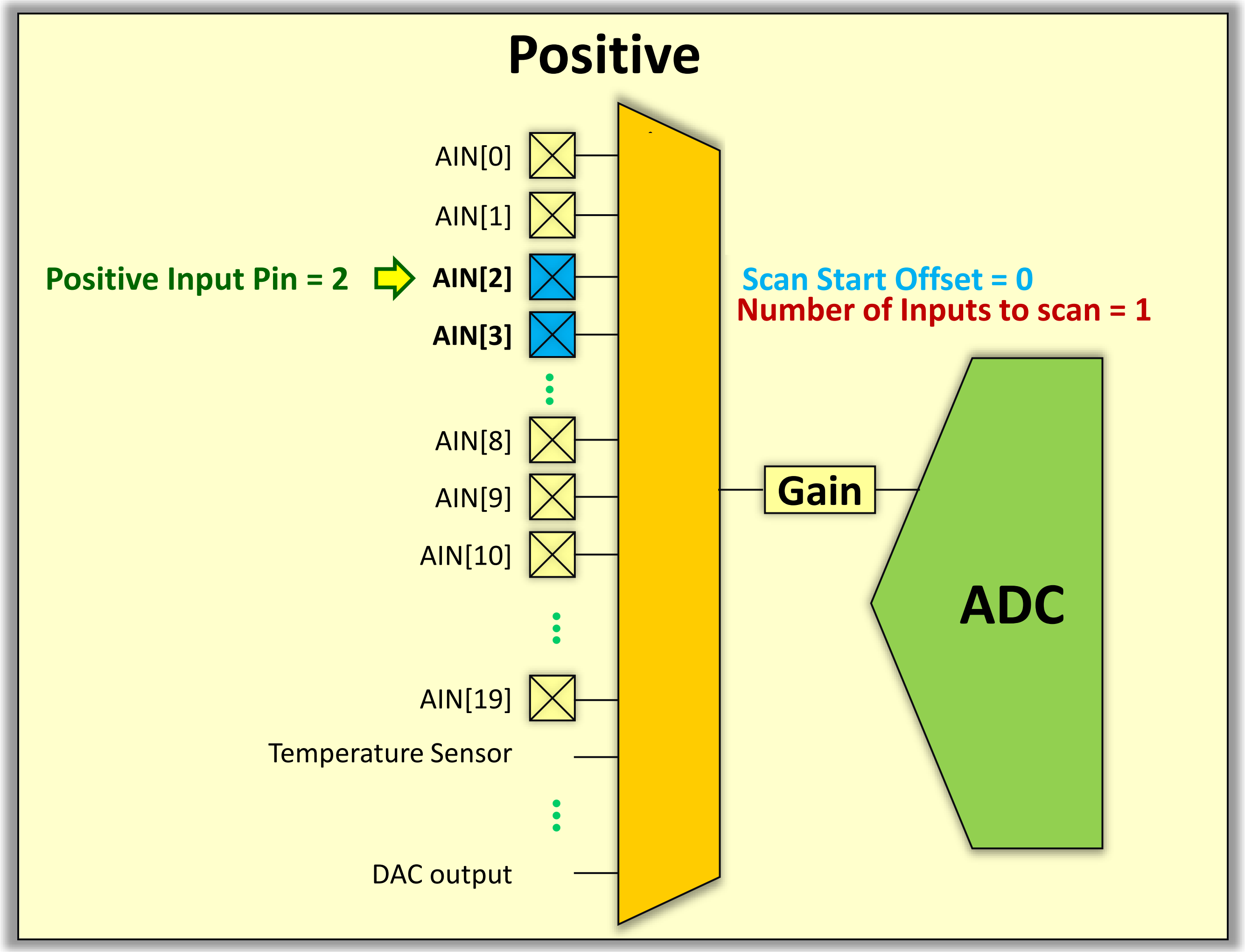
For Example : To do ADC Pin Scan from AIN[2] to AIN[3].

Positive Input Pin = AIN[0] Scan Start Offset = 2 Start Pin = AIN[0+2] = AIN[2]	OR	Positive Input Pin = AIN[2] Scan Start Offset = 0 Start Pin = AIN[2+0] = AIN[2]

Number of Inputs to scan = 1

End Pin = AIN[2+1] = AIN[3]

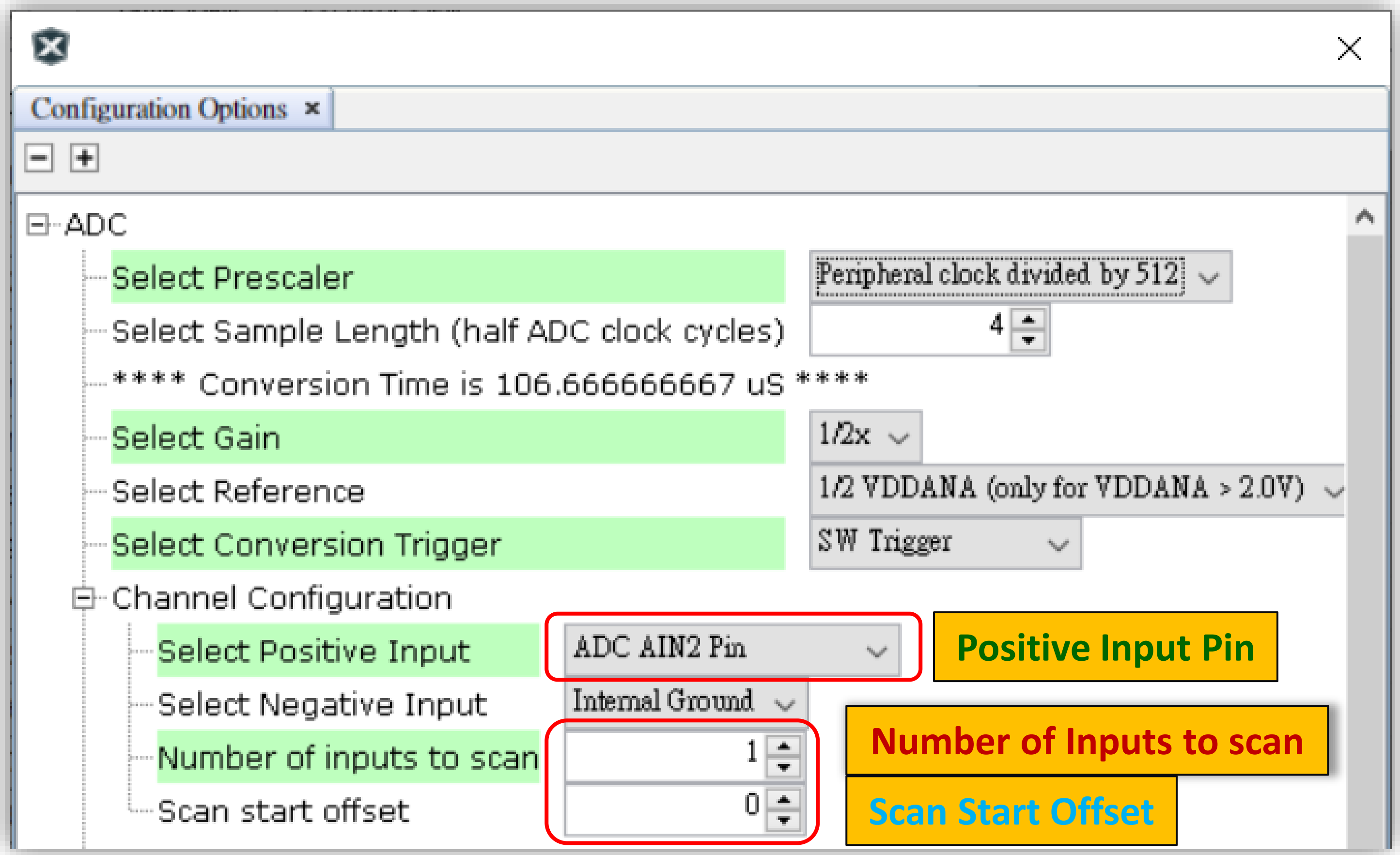
ADC Pin Scan



Positive
AIN[0]
AIN[1]
Positive Input Pin = 2
AIN[2]
AIN[3]
AIN[8]
AIN[9]
AIN[10]
AIN[19]
Temperature Sensor
DAC output
Scan Start Offset = 0
Number of Inputs to scan = 1
Gain
ADC

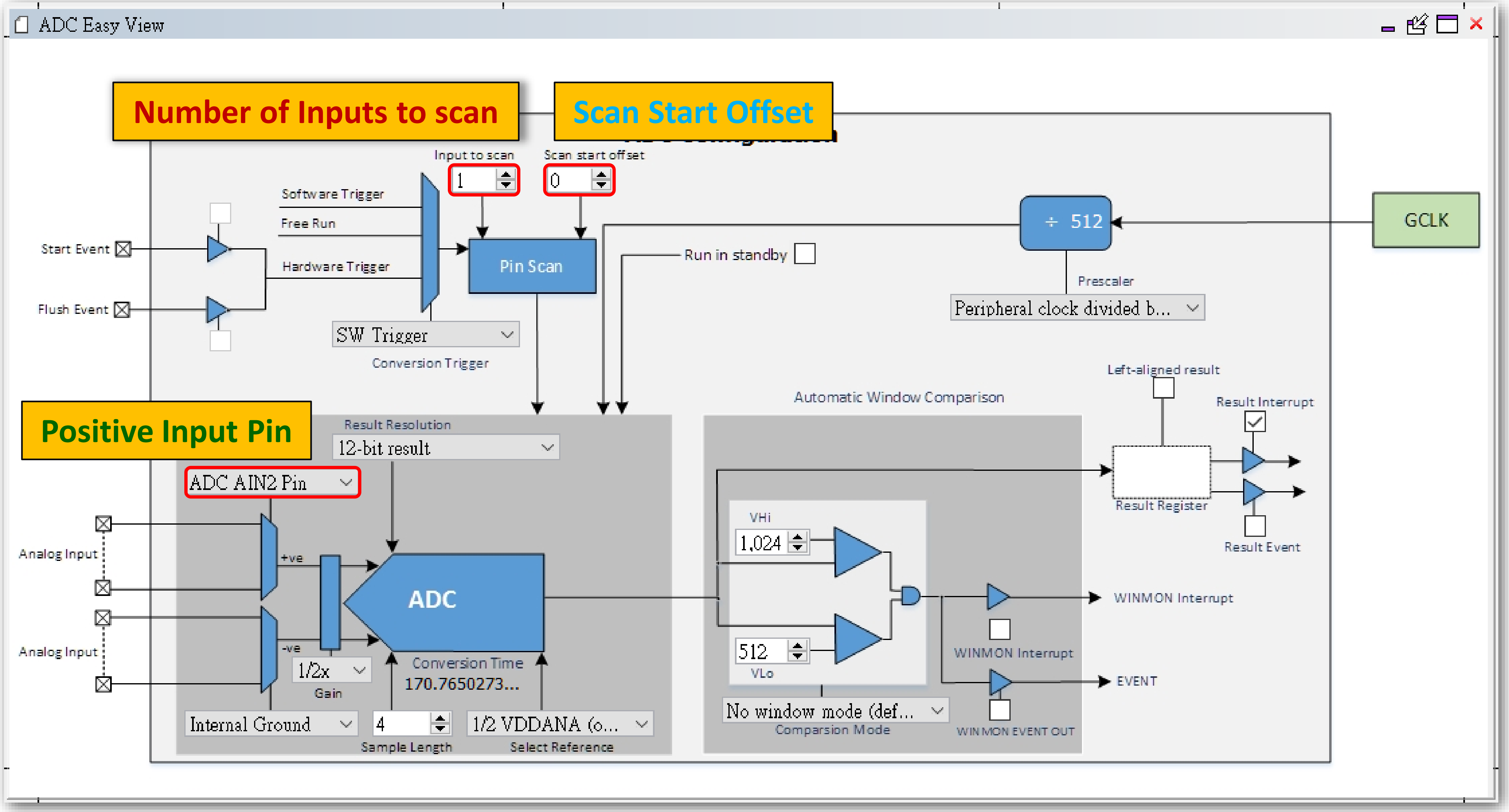
ADC Pin Scan

Configuration Options Example



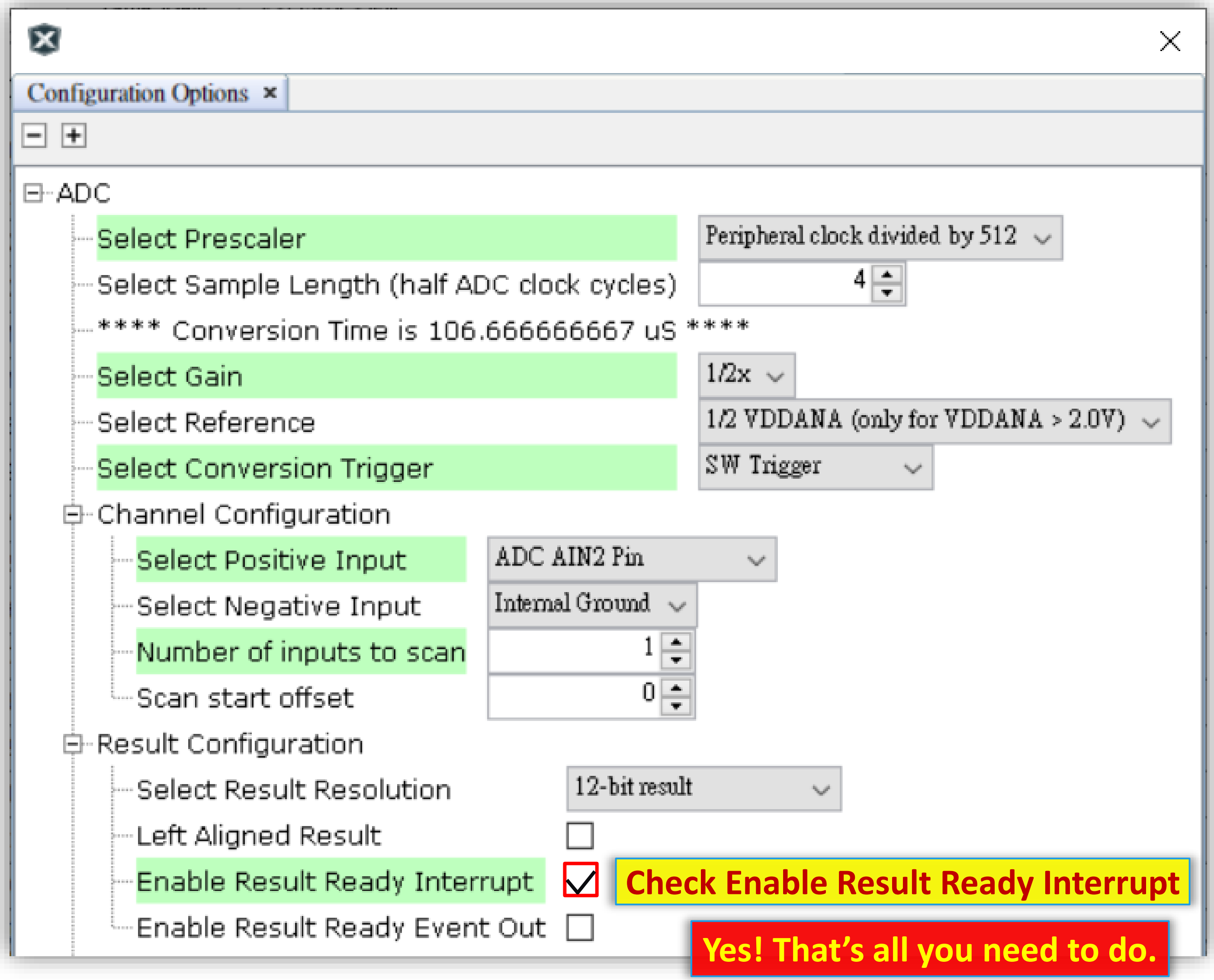
ADC Pin Scan

Easy View Configuration Example



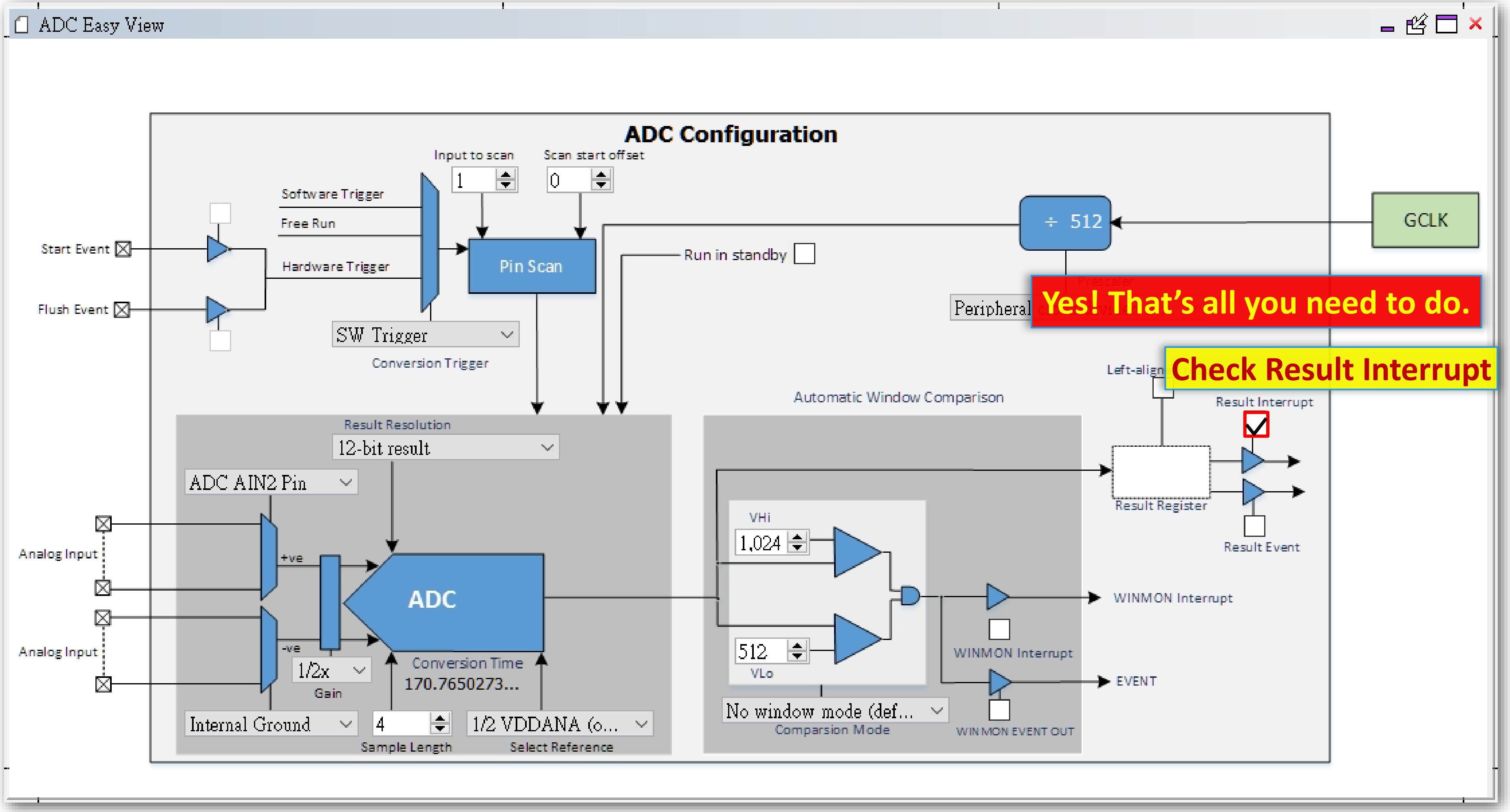
ADC Interrupt

Configuration Options Example



ADC Interrupt

Easy View Configuration Example



ADC Interrupt

Interface Routines

```
void ADC_Initialize( void );
void ADC_Enable( void );
void ADC_Disable( void );
void ADC_ChannelSelect( ADC_POSINPUT positiveInput,  ADC_NEGINPUT negativeInput );
void ADC_ConversionStart( void );
uint16_t ADC_ConversionResultGet( void );
void ADC_ComparisonWindowSet(uint16_t low_threshold, uint16_t high_threshold);
void ADC_WindowModeSet(ADC_WINMODE mode);
void ADC_CallbackRegister( ADC_CALLBACK callback, uintptr_t context );
```

ADC Interrupt

Software Trigger Code Example

```
 …
void ADC_Complete( ADC_STATUS status, uintptr_t context )
{
   if( status & ADC_INTFLAG_RESRDY_Msk )
   {
      myprintf( "%d\r\n", ADC_ConversionResultGet() );
   }
   ADC_ConversionStart();
}

int main (void)
{  …
   ADC_CallbackRegister( ADC_Complete, (uintptr_t )NULL );
   ADC_Enable();
   ADC_ConversionStart();
   while( true )
   {   …
   }
}
```

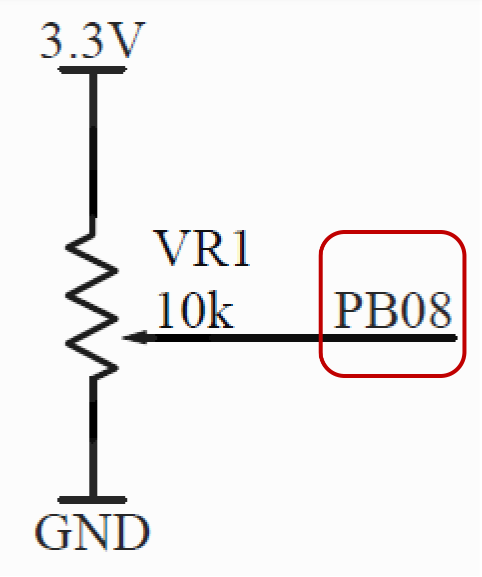
Lab13 ADC Pin Scan

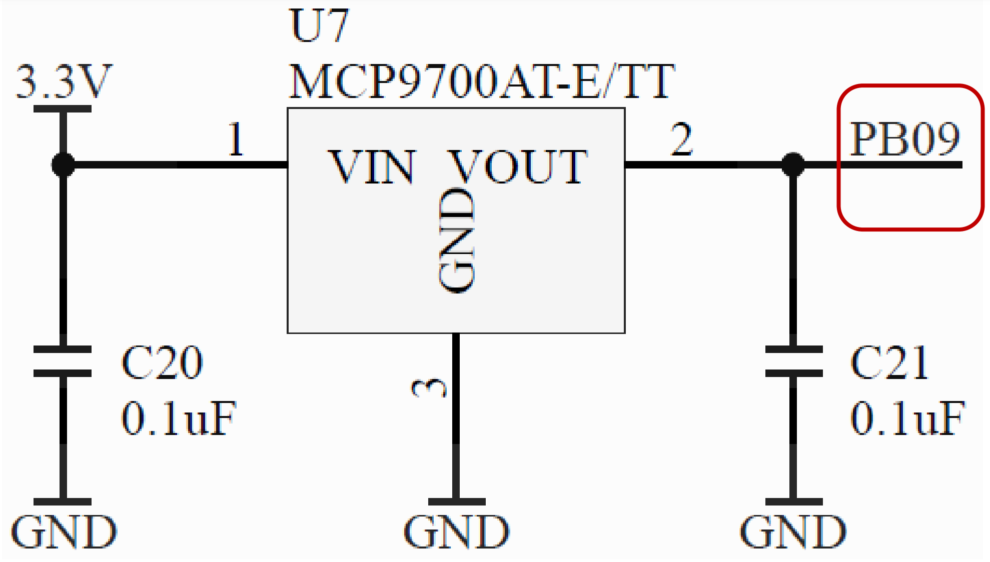
- Change ADC polling to Interrupt callback mode and enable pin scan function.
- Get the ADC conversion value from VR1(AIN[2]) and Analog Temperature Sensor(AIN[3]) with one second period and then use UART to output ADC values to PC.

- Let’s go!

Lab13 ADC Pin Scan

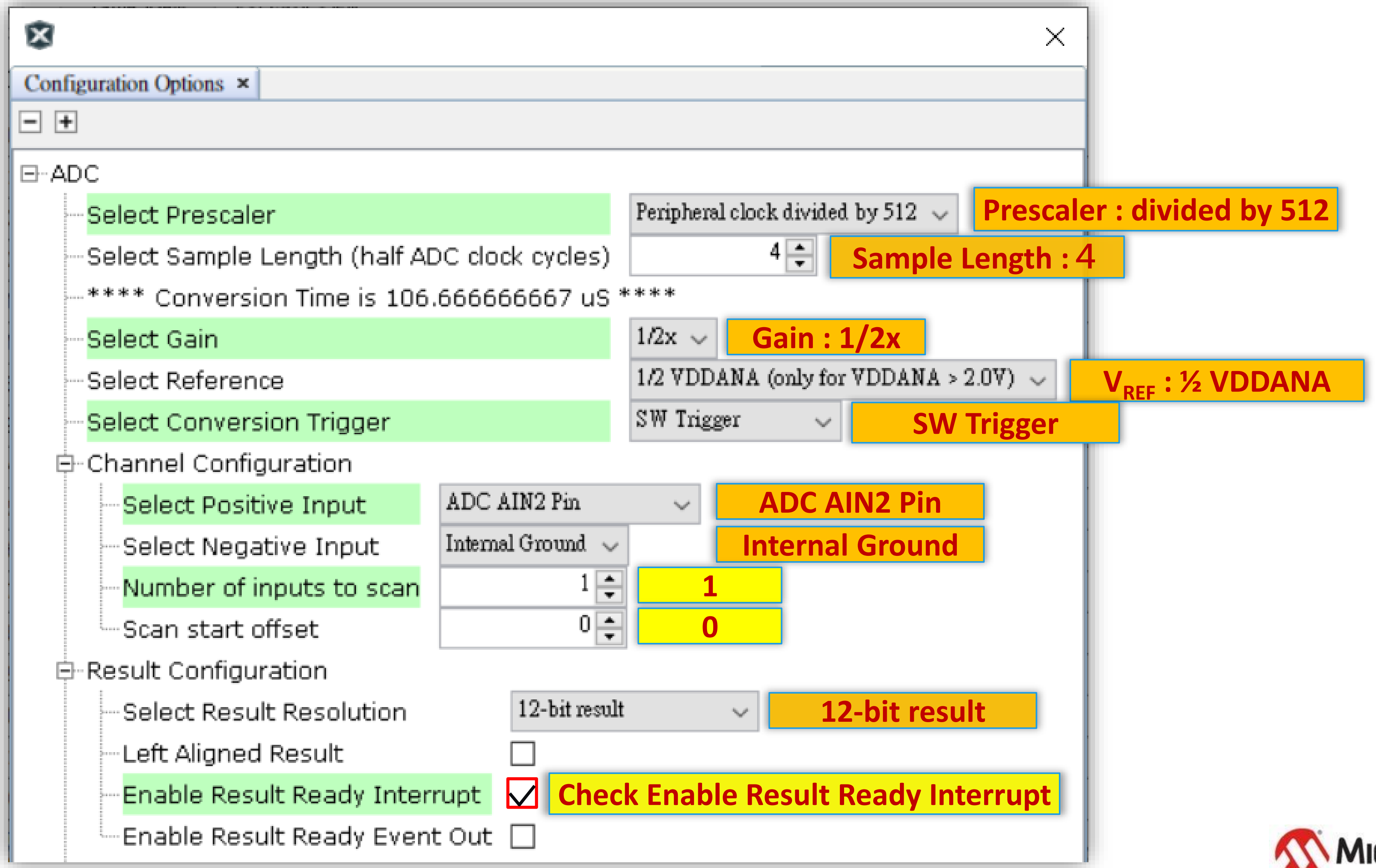
Pin(1)			I/O Pin	Supply	A	B(2)(3)				
SAMD21E	SAMD21G	SAMD21J			EIC	REF	ADC	AC	PTC	DAC
	7	11	PB08	VDDANA	EXTINT[8]		AIN[2]		Y[14]	
	8	12	PB09	VDDANA	EXTINT[9]		AIN[3]		Y[15]	





Lab13 ADC Pin Scan

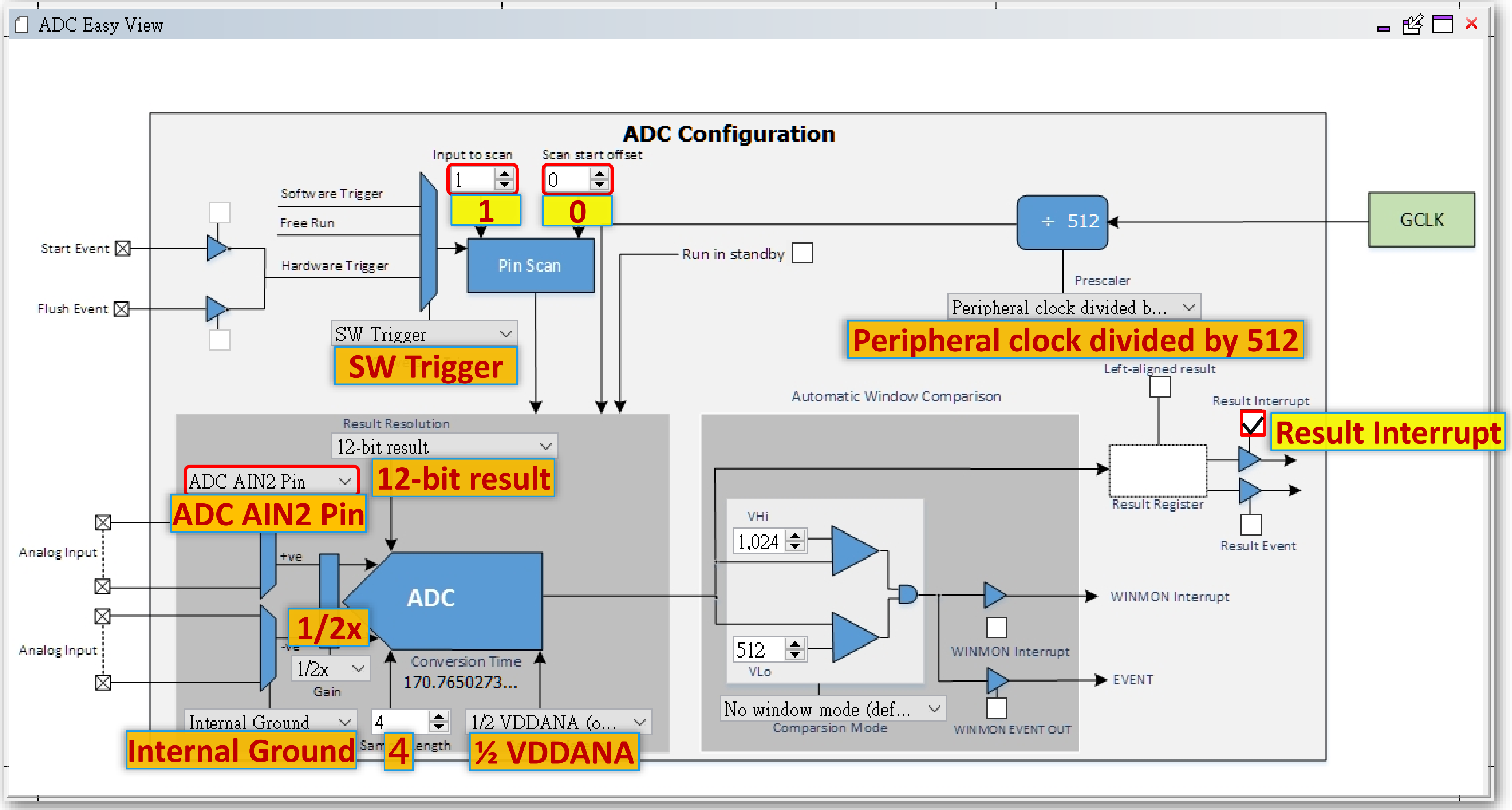
Step 1 (Use Configuration Options)



MICROCHIP

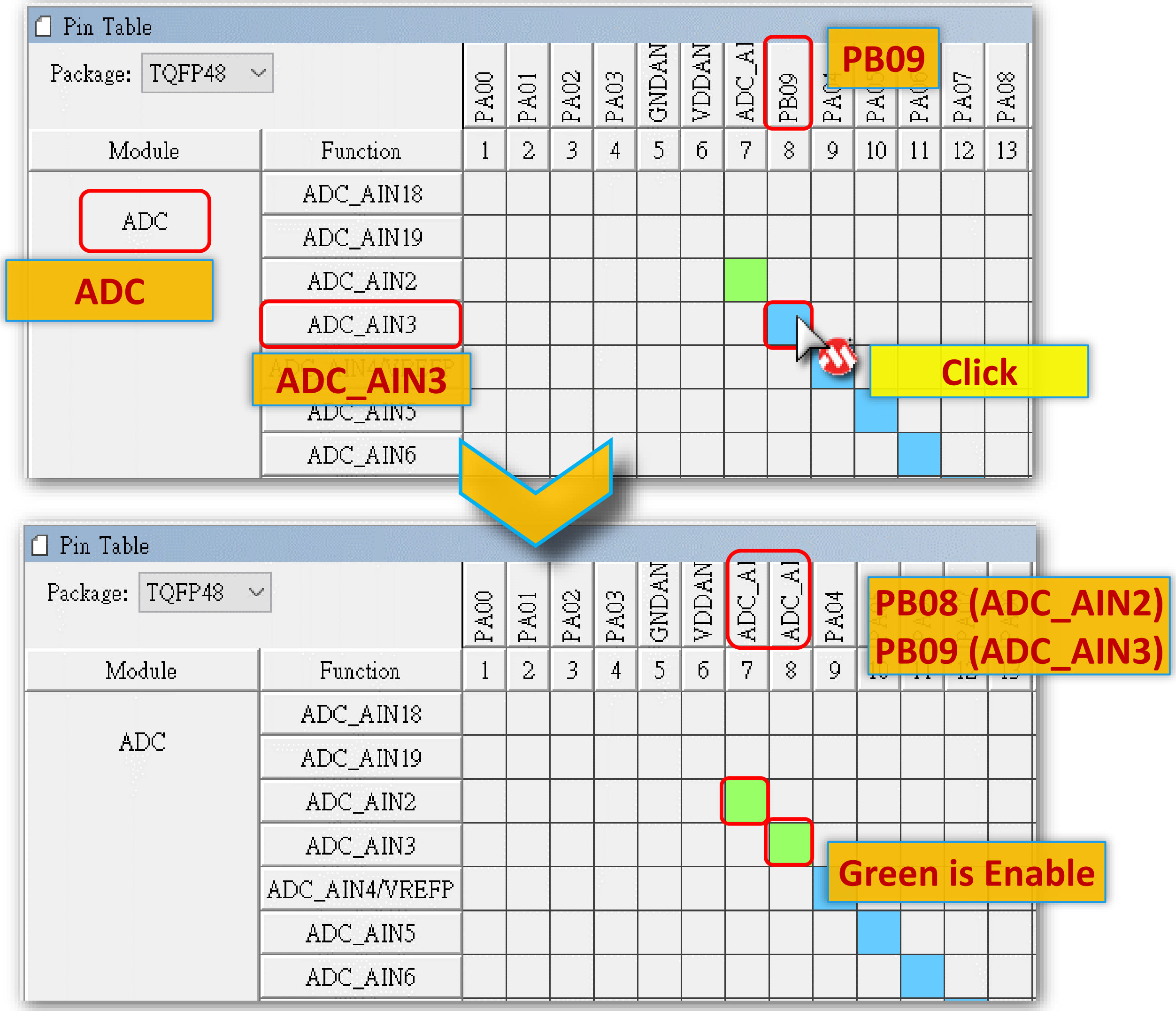
Lab13 ADC Pin Scan

Step 1 (Use Easy View)



Lab13 ADC Pin Scan

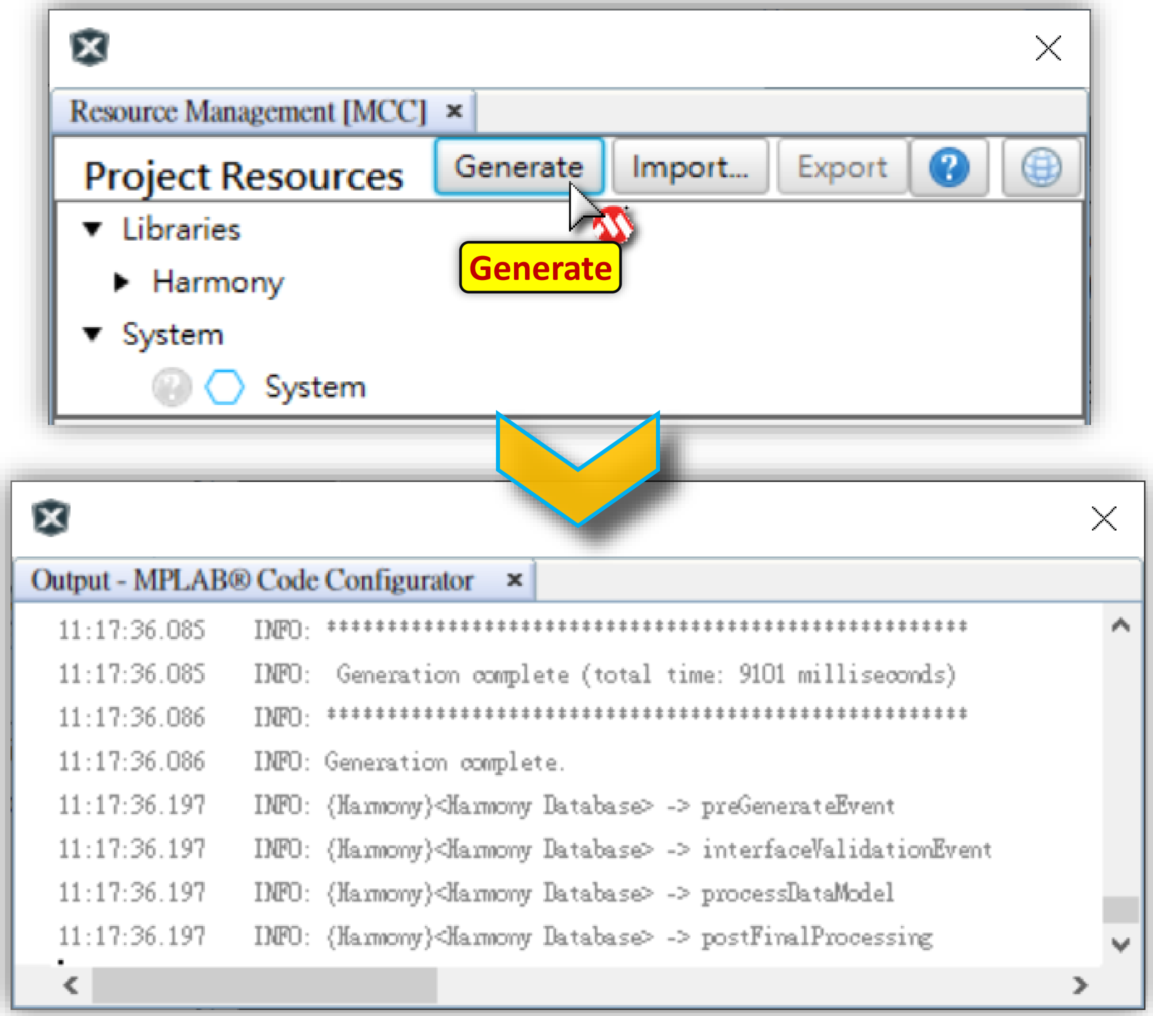
Step 2



Lab13 ADC Pin Scan

Step 3

- Click "Generate" Button to Generate your Code.



Lab13 ADC Pin Scan

Step 4

a Add code segment to your main.c

```
// TODO 13.01
// uint16_t ADC_Result;
uint16_t ADC_Result[2];
volatile uint8_t ADC_IsCompleted = 0;
volatile int8_t  ADC_ChannelIdx = 0;
…
// TODO 13.02
void ADC_Complete( ADC_STATUS status, uintptr_t context )
{
  if( status & ADC_INTFLAG_RESRDY_Msk )
  {
    ADC_Result[ADC_ChannelIdx] = ADC_ConversionResultGet();
    ADC_ChannelIdx = ( ADC_ChannelIdx + 1 ) % 2;
    if( ADC_ChannelIdx == 0 )
    {
        ADC_IsCompleted = 1;
    }
  }
}
…

int main (void)
{  …
```

Lab13 ADC Pin Scan

Step 5

a Add code segment to your main.c

```
…

int main (void)
{
   …
   // TODO 13.03
   ADC_CallbackRegister( ADC_Complete, (uintptr_t )NULL );
   ADC_Enable();

   while( true )
   {
      …
      if( TC4_HasExpired)
      {
         …
         // TODO 13.04
         ADC_ConversionStart();
         //  while (!ADC_ConversionStatusGet());
         //  ADC_Result = ADC_ConversionResultGet();
         //  myprintf("\033[2;1HVR : %4d", ADC_Result);
      }
      …
   }
}
```

Lab13 ADC Pin Scan

Step 6

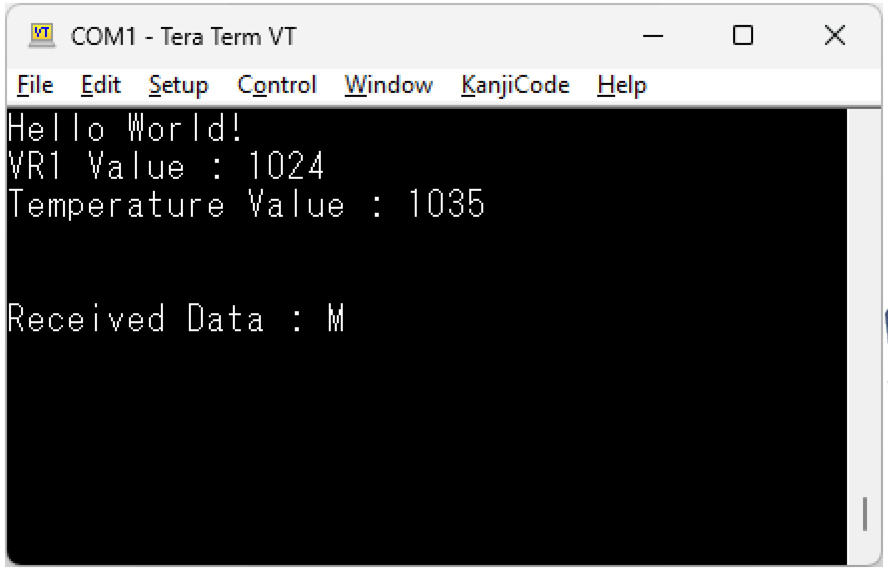
a Add code segment to your main.c

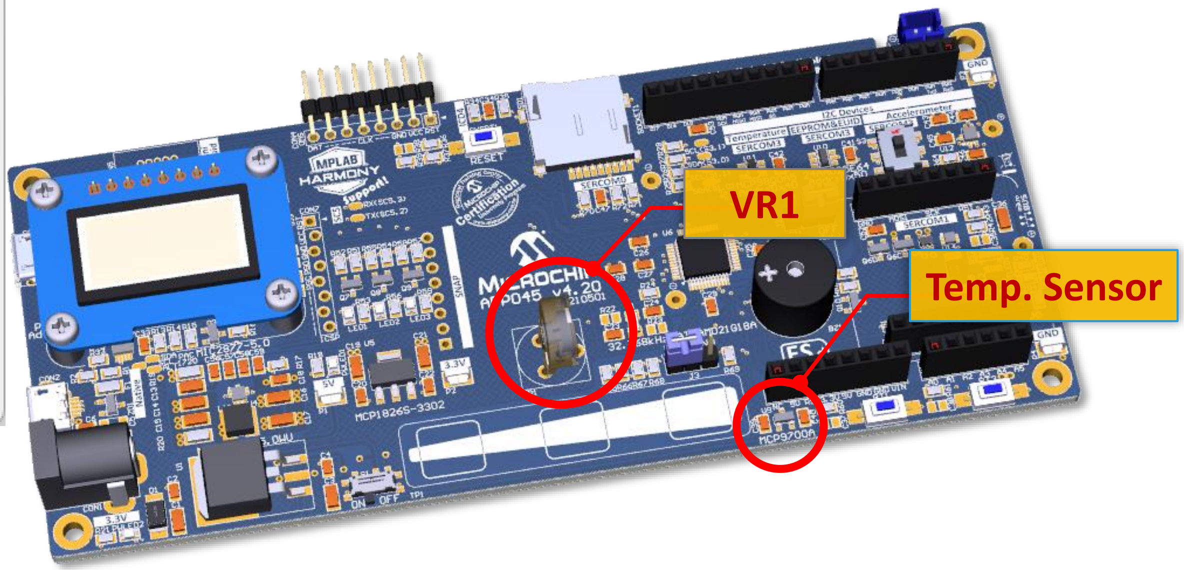
```
…

int main (void)
{
   …

   while( true )
   { …

      if( USART5_IsReceived )
      { … }

      // TODO 13.05
      if( ADC_IsCompleted )
      {
        ADC_IsCompleted = 0;

        myprintf( "\033[2;1HVR1 Value : %4d", ADC_Result[0] );
        myprintf( "\033[3;1HTemperature Value : %4d", ADC_Result[1] );
      }
   }
}
```

Lab13 ADC Pin Scan

Result

- "Hello World!" string and VR1 / Temperature Sensor ADC value will output to terminal with one second period.

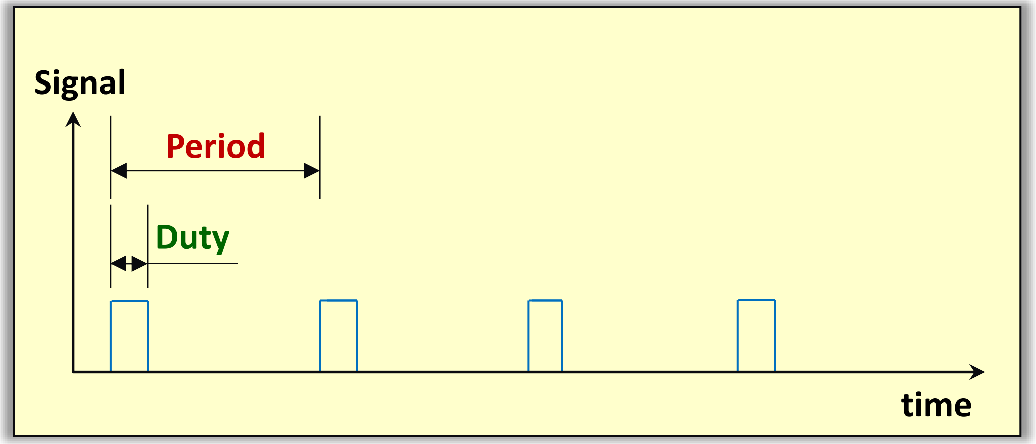




TCC – PWM Architecture

What's PWM

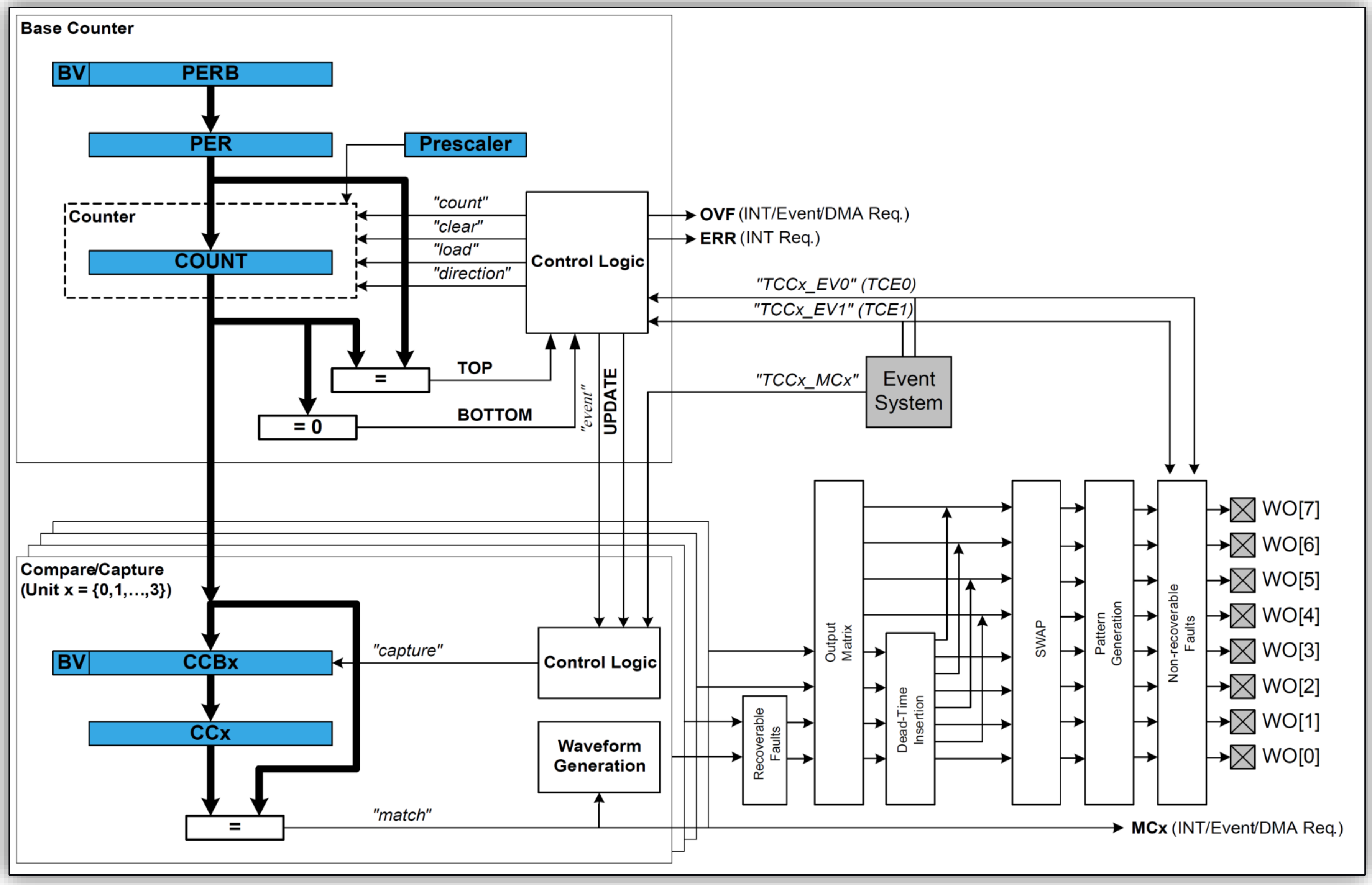
- The Pulse-width modulation (PWM) Waveform is a modulation technique used to encode a message into a pulsing signal.
- Its main use is to allow the control of the power supplied to electrical devices, ex. motor control, ballast, LED, H-bridge, power converters, and other types of power control applications.
- There have two important terms
  - **Period** :
    The single square wave duration.
  - **Duty** :
    The proportion of active time to the regular interval



SAMD21 TCC

- **Timer/Counter for Control, TCC is the TC Enhanced version.**
- **TCC provide below feature**
  - Up to four compare/capture channels
  - Waveform (PWM, Frequency generation)
  - Input capture (Event, Frequency, PWM Capture)
  - Fault protection for safe disabling
  - 3 input Event, 3 output Event
  - Support Interrupt, DMA, Event

TCC Block Diagram



MICROCHIP

TCCx, Channel and WO[x]

- **TCCx : Timer Counter for Control peripheral**
  - TCC0, TCC1, TCC2

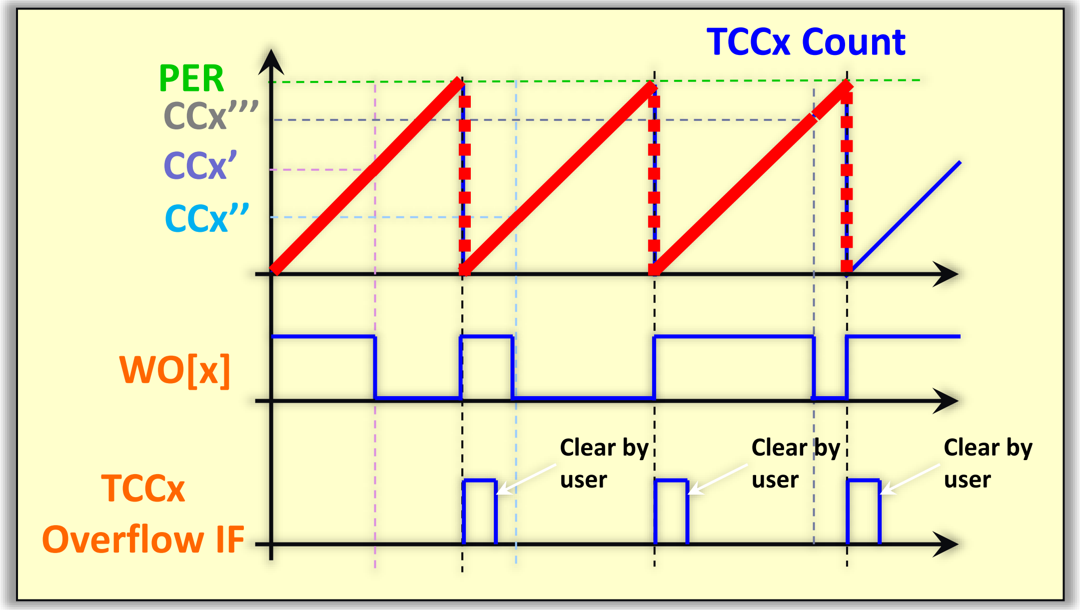
- **Channel : Compare Channel CCx**
  - TCC0 : 4 channel
  - TCC1 : 2 channel
  - TCC2 : 2 channel

- **WO[x] : Waveform Output Pins**
  - TCC0 : 8 pins → WO[0] ~ WO[7] → Channel [0, 1, 2, 3, **0, 1, 2, 3**]
  - TCC1 : 4 pins → WO[0] ~ WO[3] → Channel [0, 1, **0, 1**]
  - TCC2 : 2 pins → WO[0] ~ WO[1] → Channel [0, 1]

Normal Pulse-Width Modulation

NPWM Mode

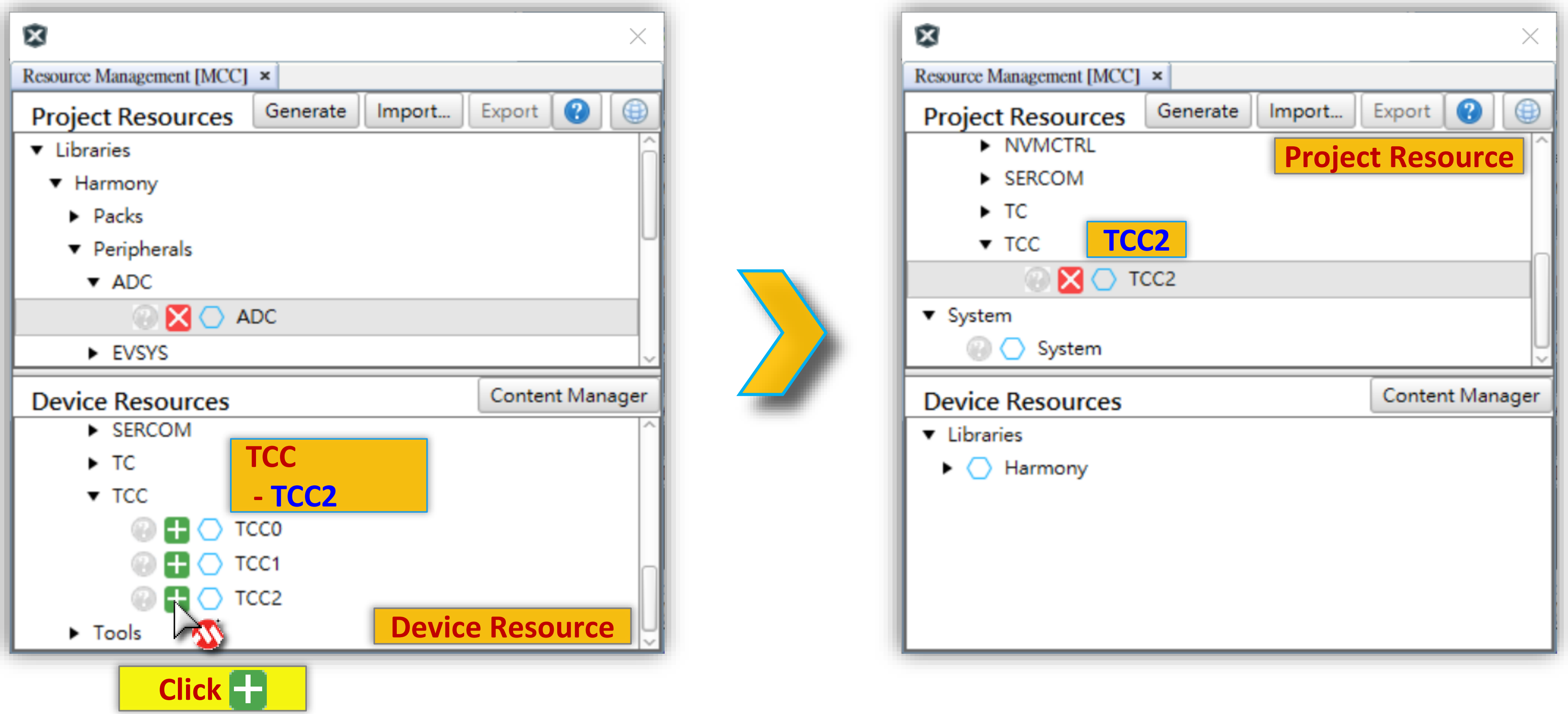
- For Normal Pulse-Width Modulation , the period time (T) is controlled by the PER register. WO[x] is set at start and cleared on compare match.



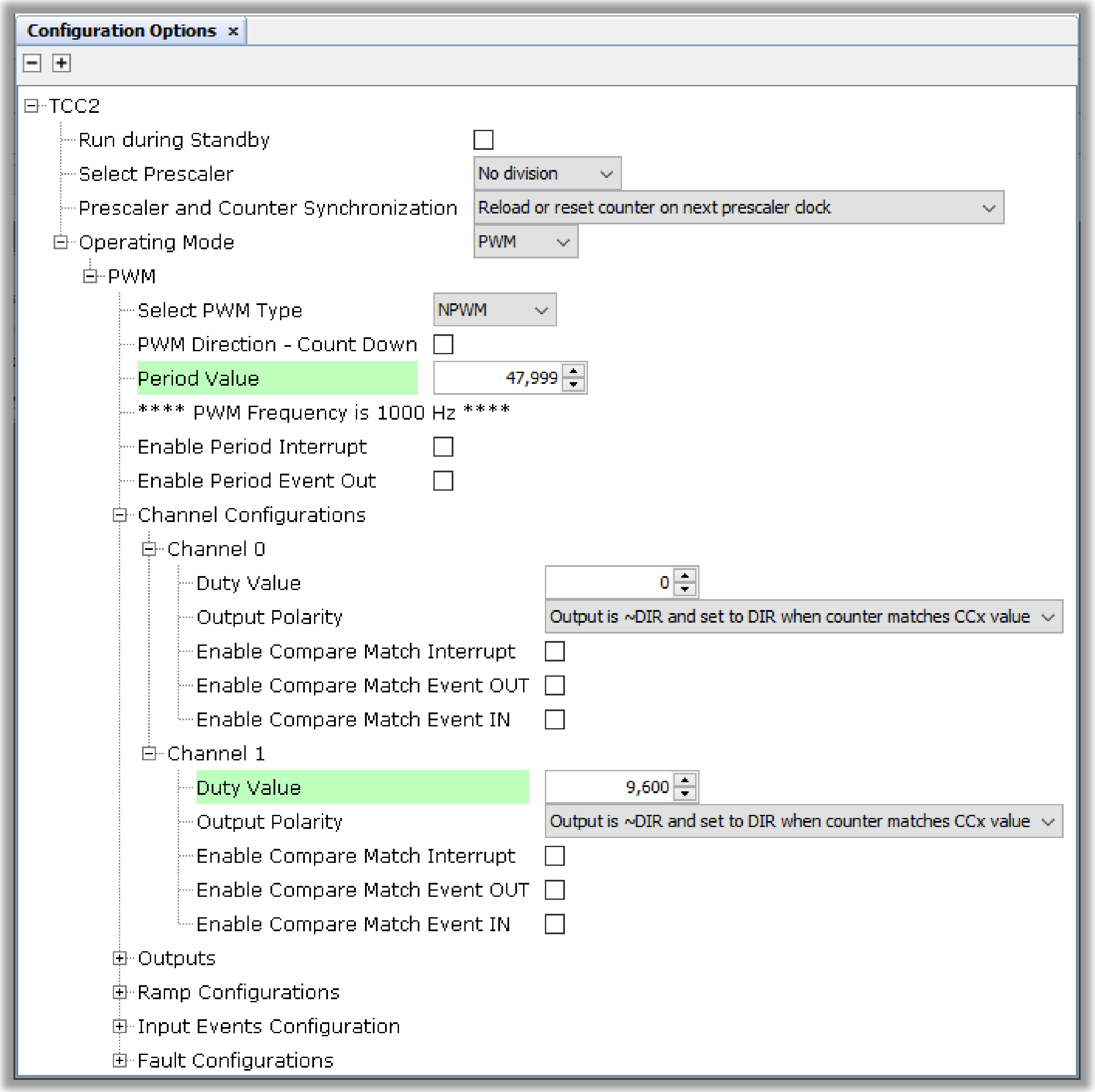
Rule
1.TCCx Counter == CCx -> WO[x] Clear to low
2.TCCx Counter == PER -> WO[x] Set to high

Add TCC Function using MCC Harmony

- Find TCC component in Device Resource window.
- Click to add TCC2 to Project Resource window.



TCC Configuration Options Example

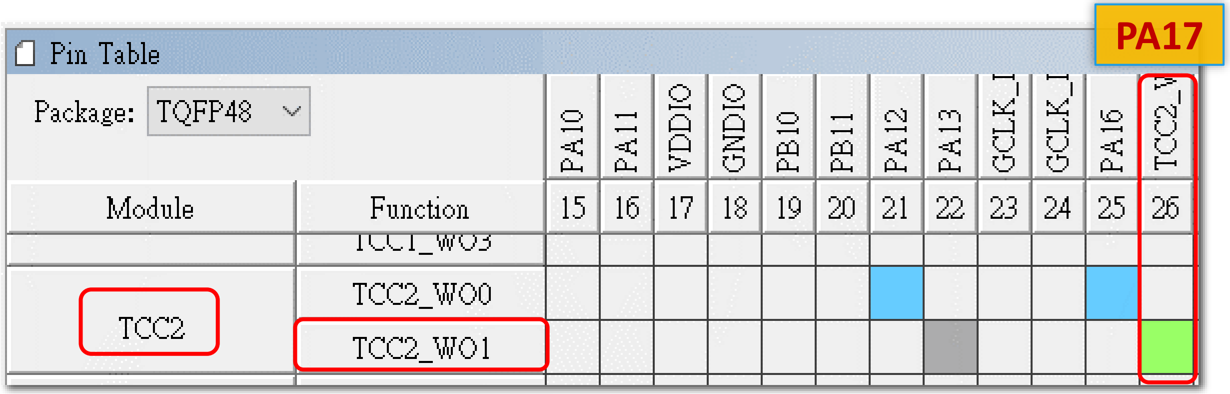


TCC PMUX Configuration Example

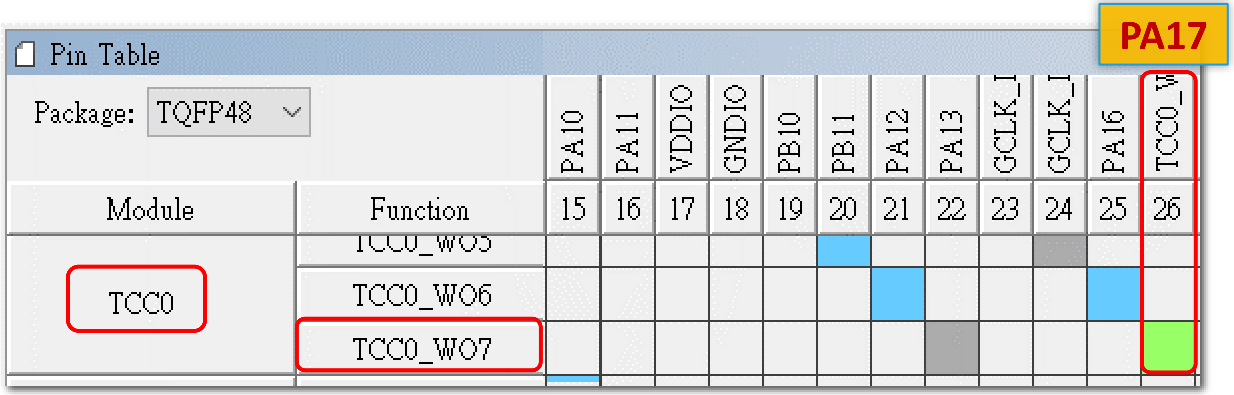
- TCC could output waveform to corresponds pins of different channel.
- For example :

  PA17 as TCC2/WO[1] (TCC2 Channel 1) or
  TCC0/WO[7] (TCC0 Channel 3)

Pin[1]			I/O Pin	Supply	A	B[2][3]					C	D	E	F	G	H
SAMD21E	SAMD21G	SAMD21J			EIC	REF	ADC	AC	PTC	DAC	SERCOM[2][3]	SERCOM-ALT	TC[4] /TCC	TCC	COM	AC/ GCLK
18	26	PA17	PA17	VDDIO	EXTINT[1]				X[5]		TCC2/WO[1]		TCC2/WO[1]	TCC0/WO[7]	TCC0/WO[7]	



TCC2 TCC2/WO[1]



TCC0 TCC0/WO[7]

MICROCHIP

TCC Polling

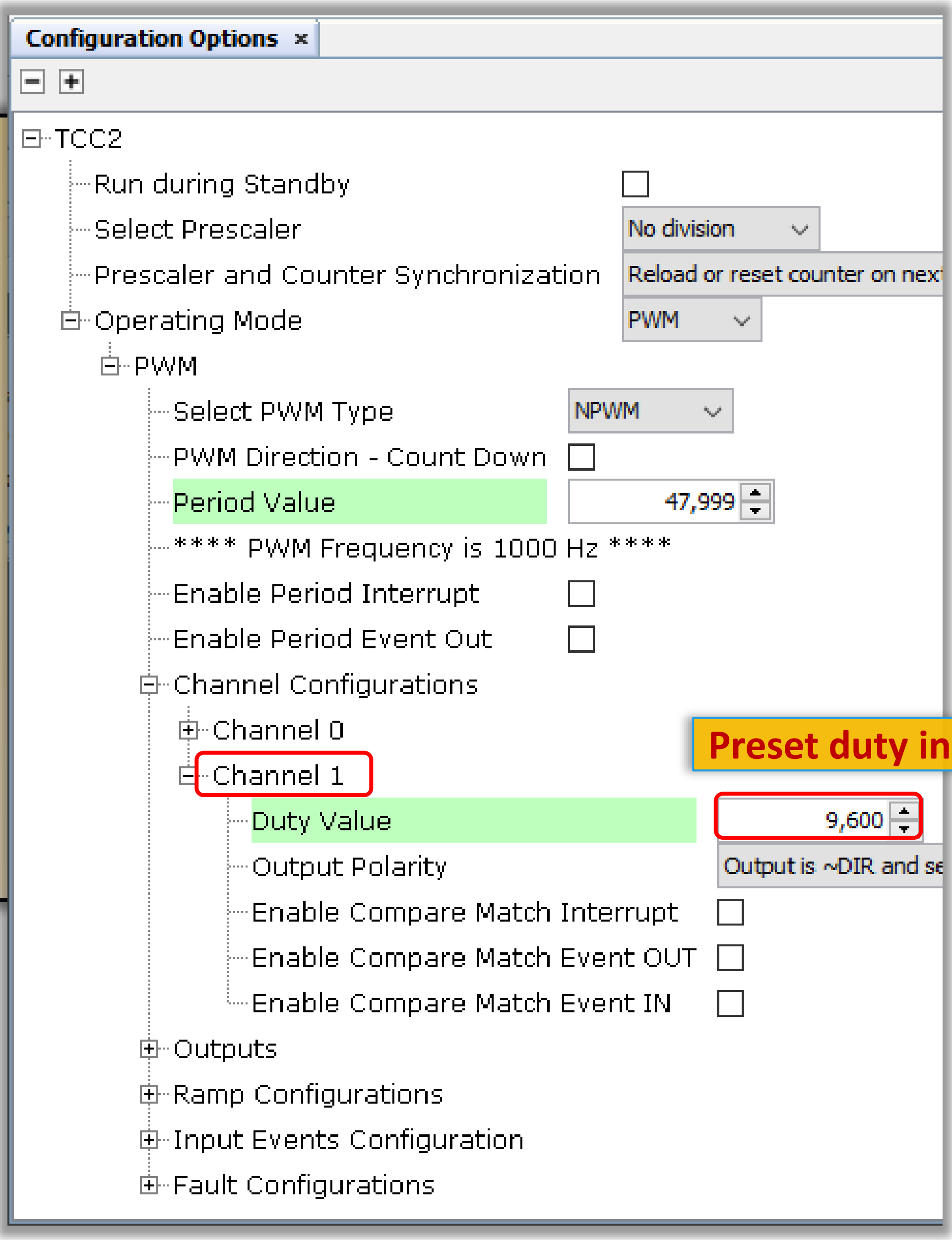
Interface Routines

```
void TCC2_PWMInitialize(void);
void TCC2_PWMStart(void);
void TCC2_PWMStop(void);
void TCC2_PWMForceUpdate(void);
void TCC2_PWMPeriodInterruptEnable(void);
void TCC2_PWMPeriodInterruptDisable(void);
uint32_t TCC2_PWMInterruptStatusGet(void);
void TCC2_PWM16bitPeriodSet(uint16_t period);
uint16_t TCC2_PWM16bitPeriodGet(void);
void TCC2_PWM16bitCounterSet(uint16_t count);
__STATIC_INLINE void TCC2_PWM16bitDutySet(TCC2_CHANNEL_NUM channel, uint16_t duty);
```

TCC Polling Code Example

```
…

int main (void)
{
   SYS_Initialize ( NULL );
   …
   TCC2_PWMStart();
   TCC2_PWM16bitDutySet( 1, 9600 );

   while( true )
   {
      …
   }
}
```

Runtime update duty Channel 1, Duty Value 9600



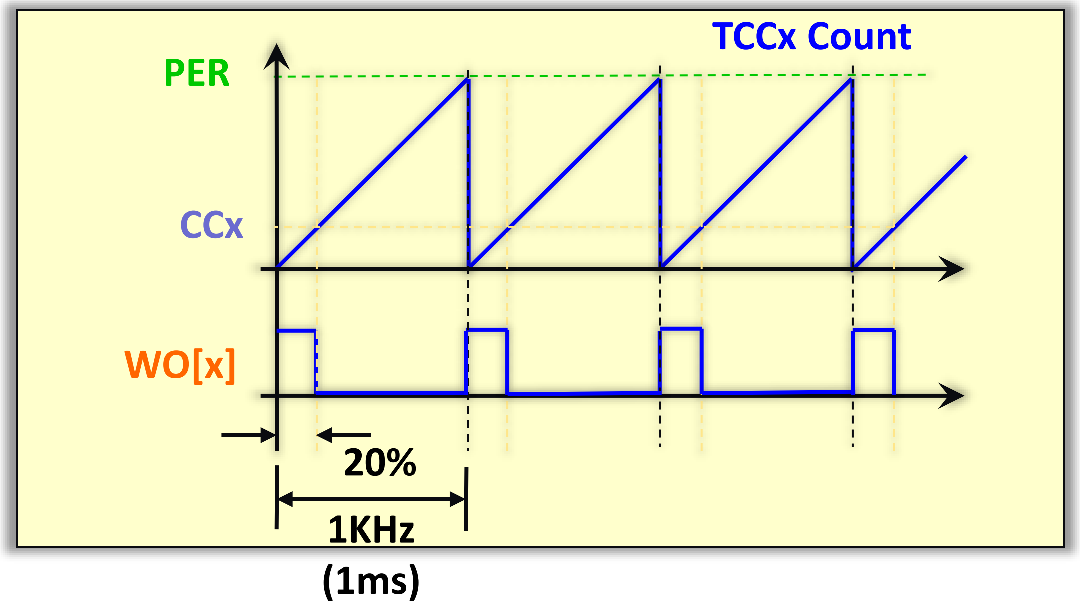
Preset duty in MCC Harmony

MICROCHIP

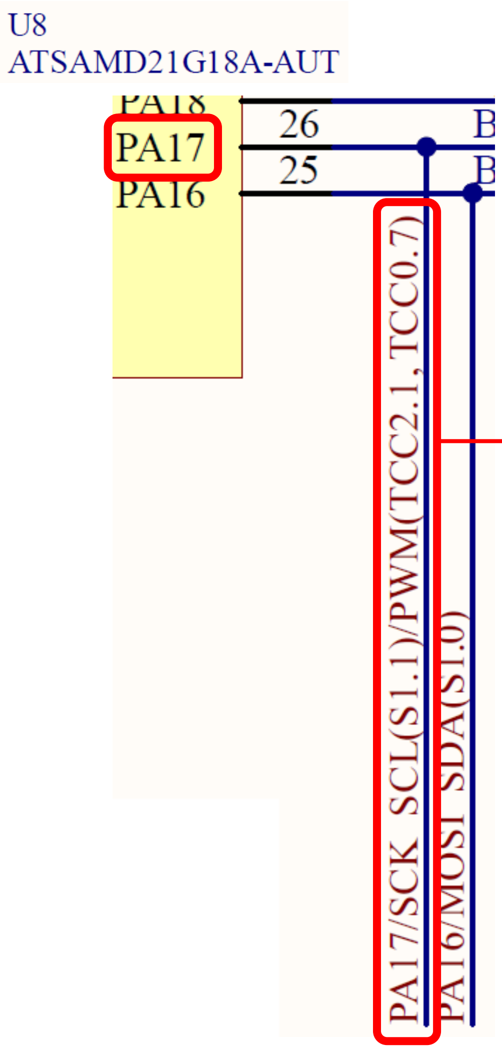
Lab14 TCC NPWM

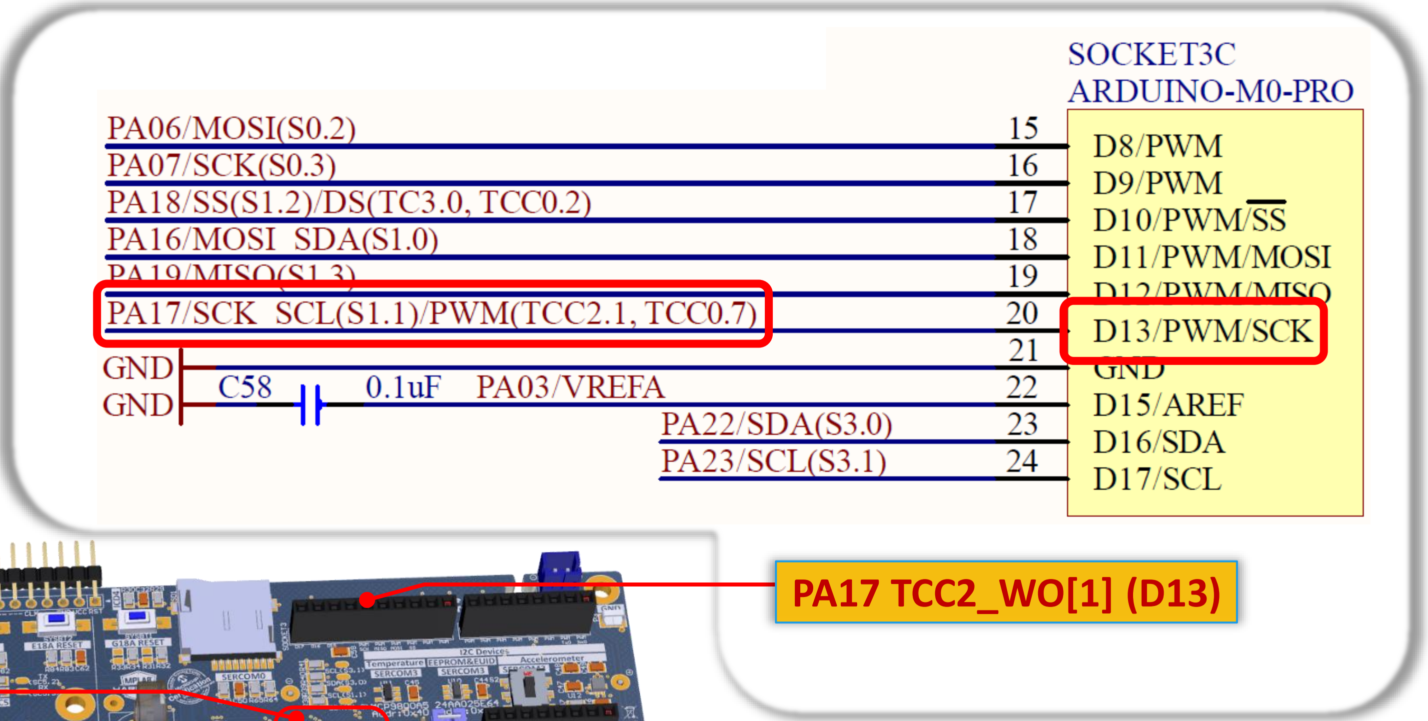
- Add TCC2 NPWM Mode (Single Slope PWM) function to your project.
- Disable LED3(PA17) GPIO function firstly.
- Configure TCC2 to output PWM waveform to LED3(PA17).
- PWM frequency is 1KHz with 20% Duty.

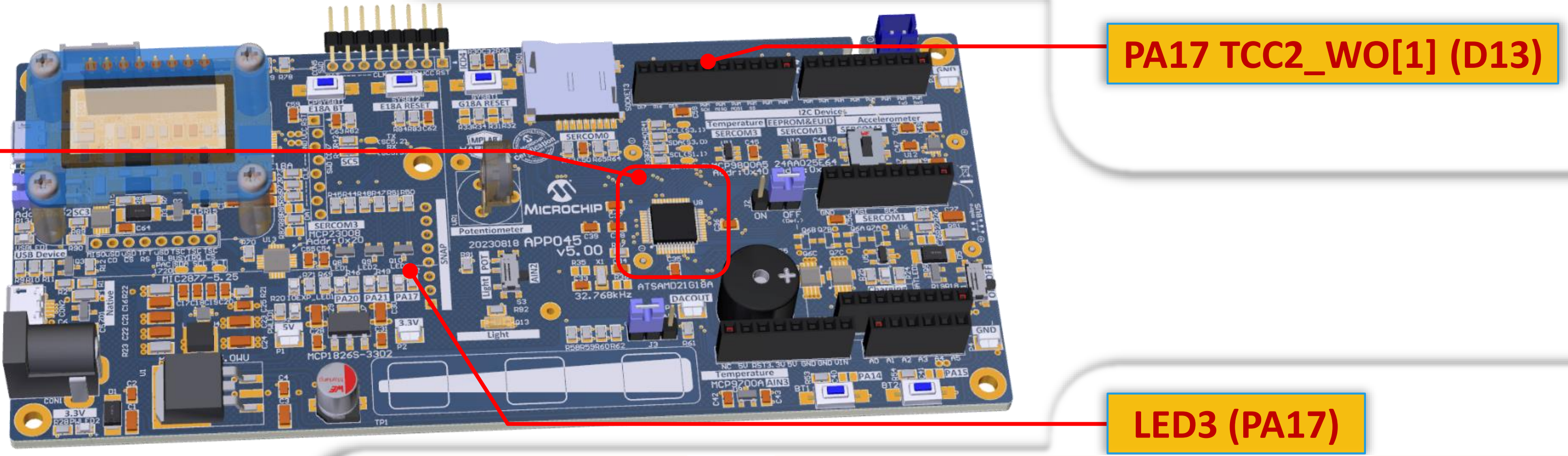
- **Let's go!**

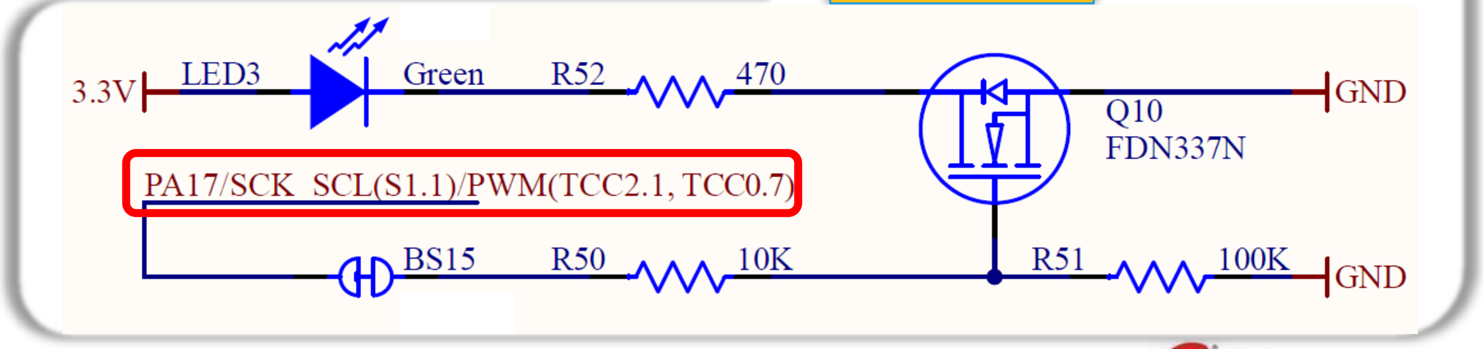


Lab14 TCC NPWM





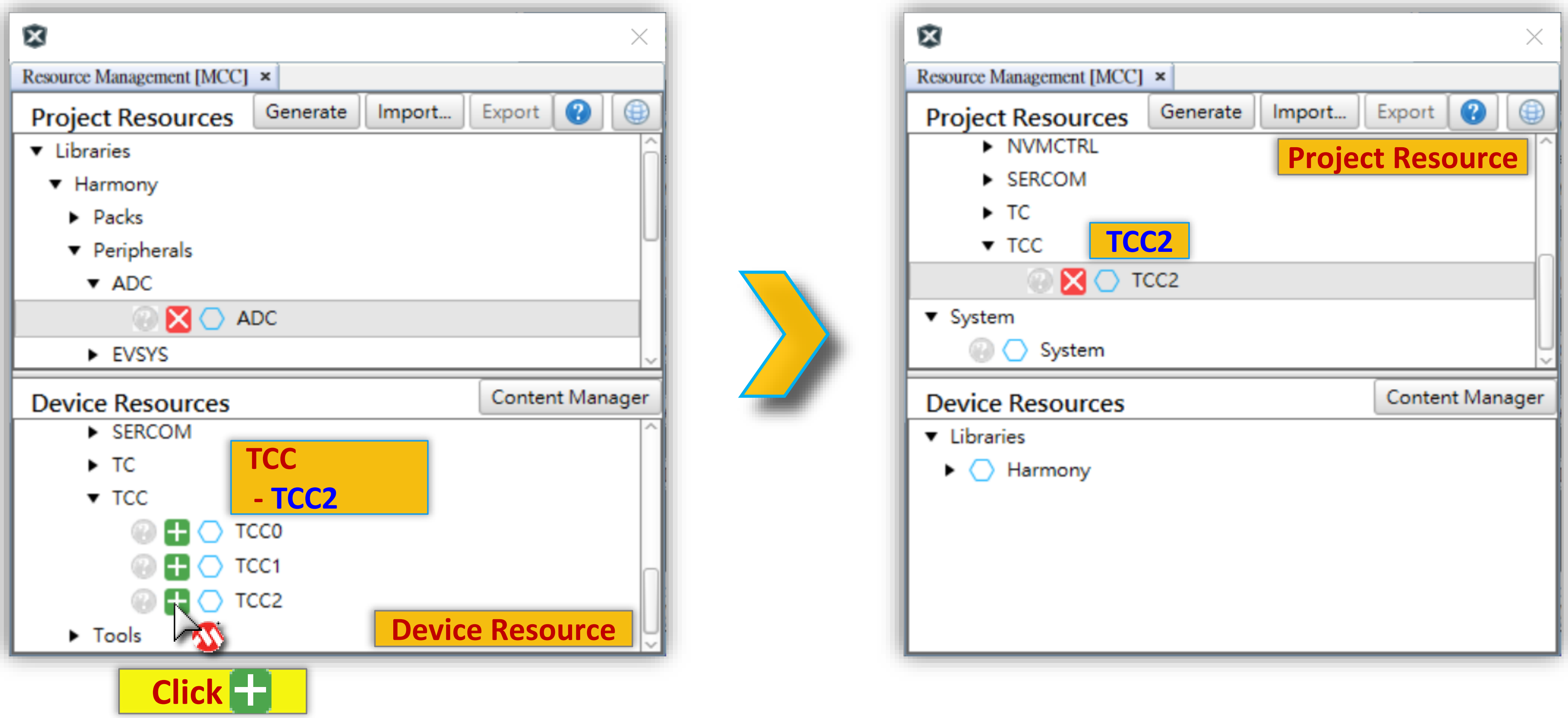




Lab14 TCC NPWM

Step 1

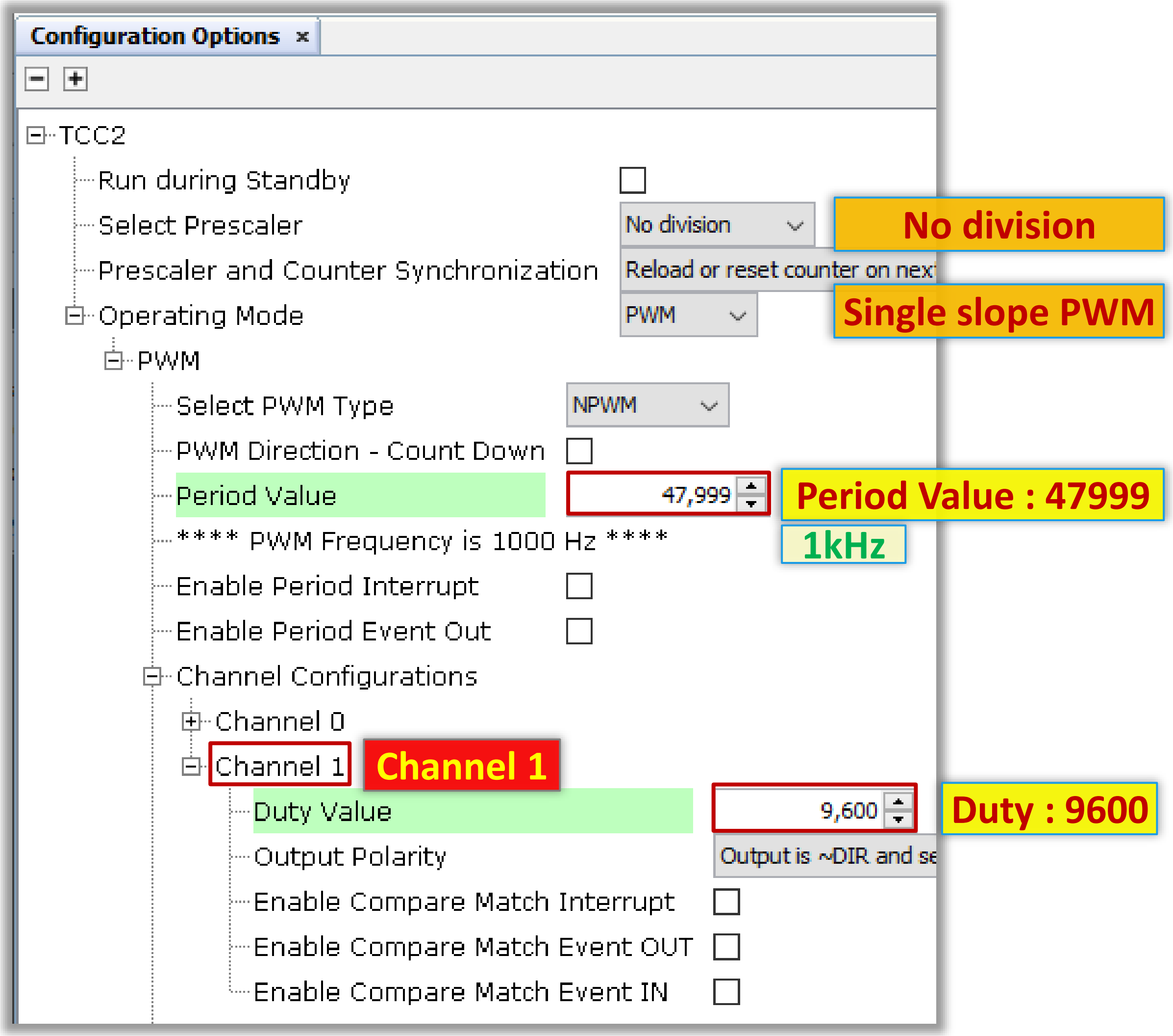
- Find TCC component in Available Components window.
- Click to add TCC2 to Active Components window.



MICROCHIP

Lab14 TCC NPWM

Step 2



System Clock = 48MHz

One second counts of system clock :

48,000,000 counts

One second counts after Prescaler :

48000000 / 1 48,000,000 counts

1ms counts for PWM period target :

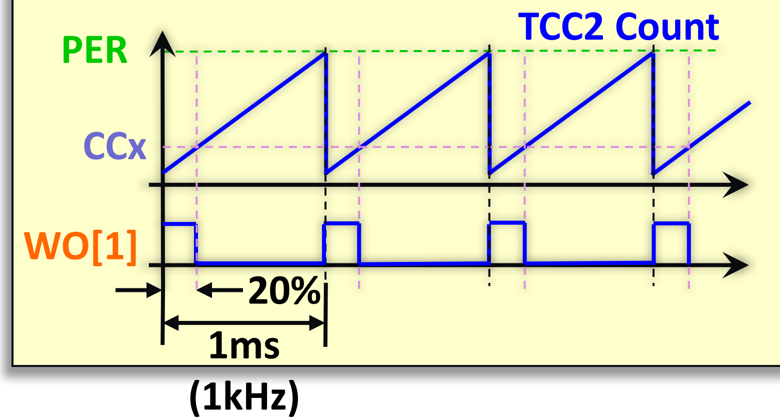
48000000 / 1000 48,000 counts

Counter clear to zero at the next tick :

47,999 + 1 48,000 counts

20% counts for PWM duty target :

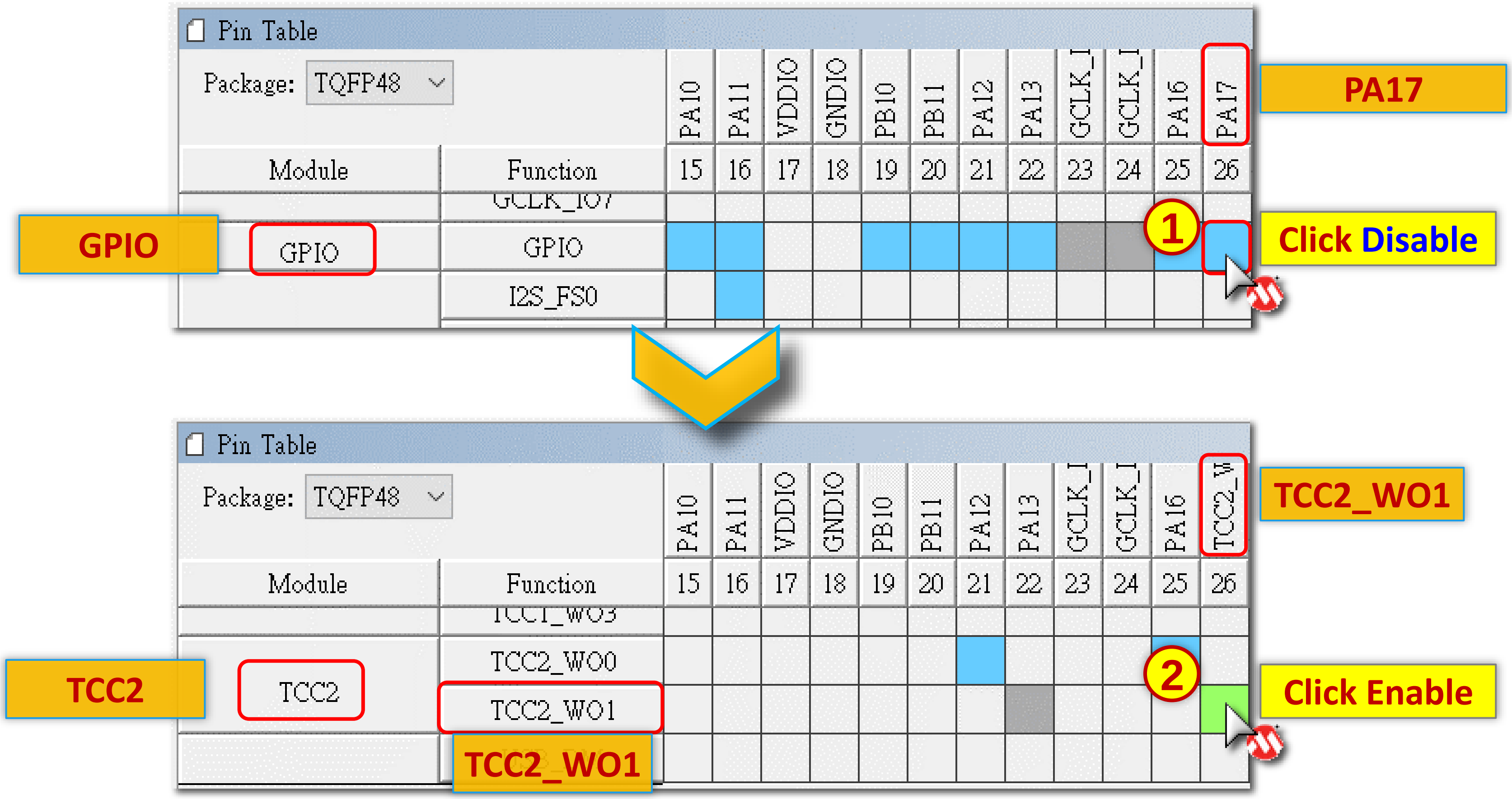
48000 * 20% 9,600 counts



Lab14 TCC NPWM

Step 3

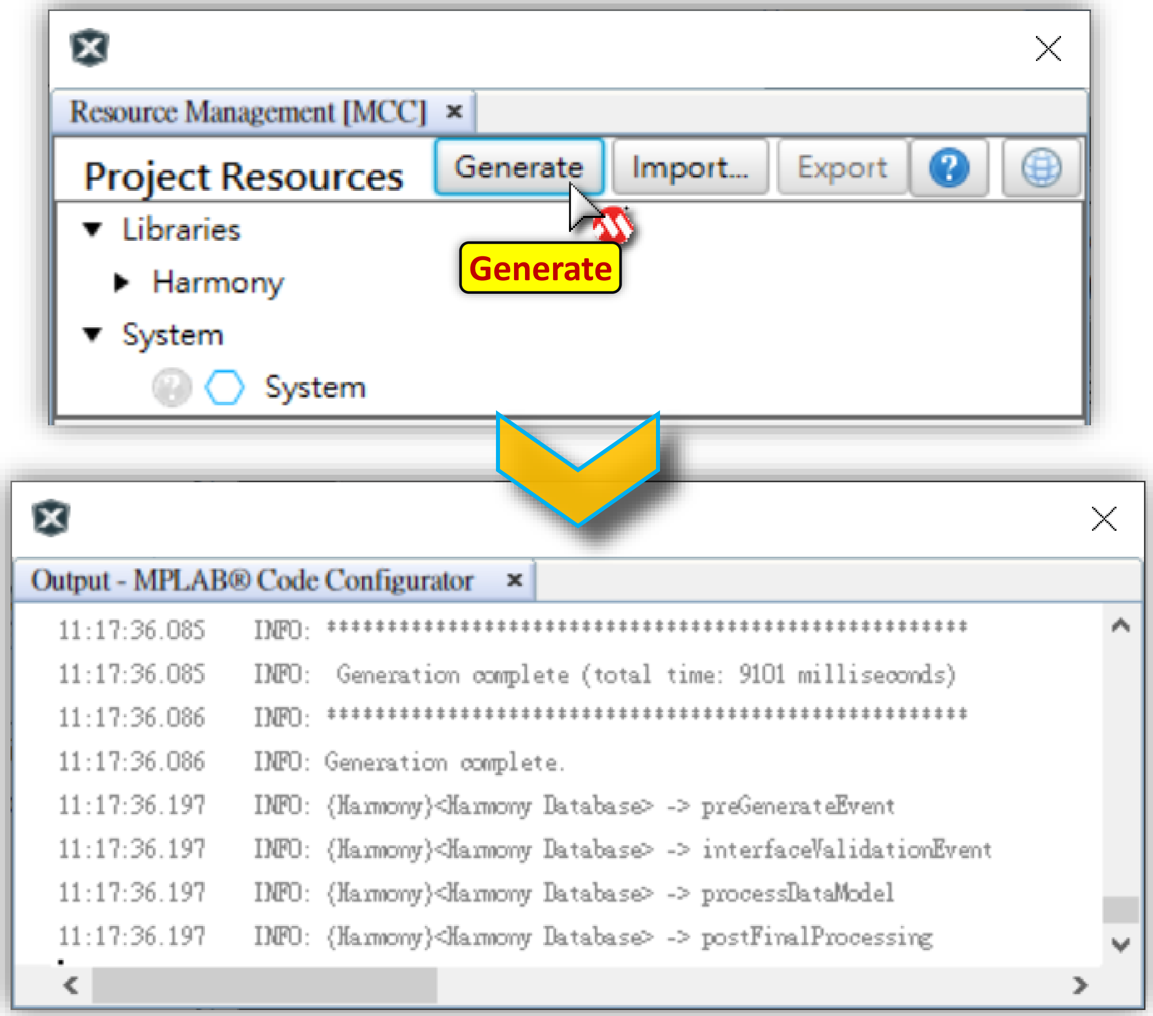
- Disable LED3(PA17) GPIO function firstly.
- Enable TCC2 Waveform Output 1 (TCC2_WO1).



Lab14 TCC NPWM

Step 4

- Click "Generate" Button to Generate your Code.



Lab14 TCC NPWM

Step 5

a Add code segment to your main.c

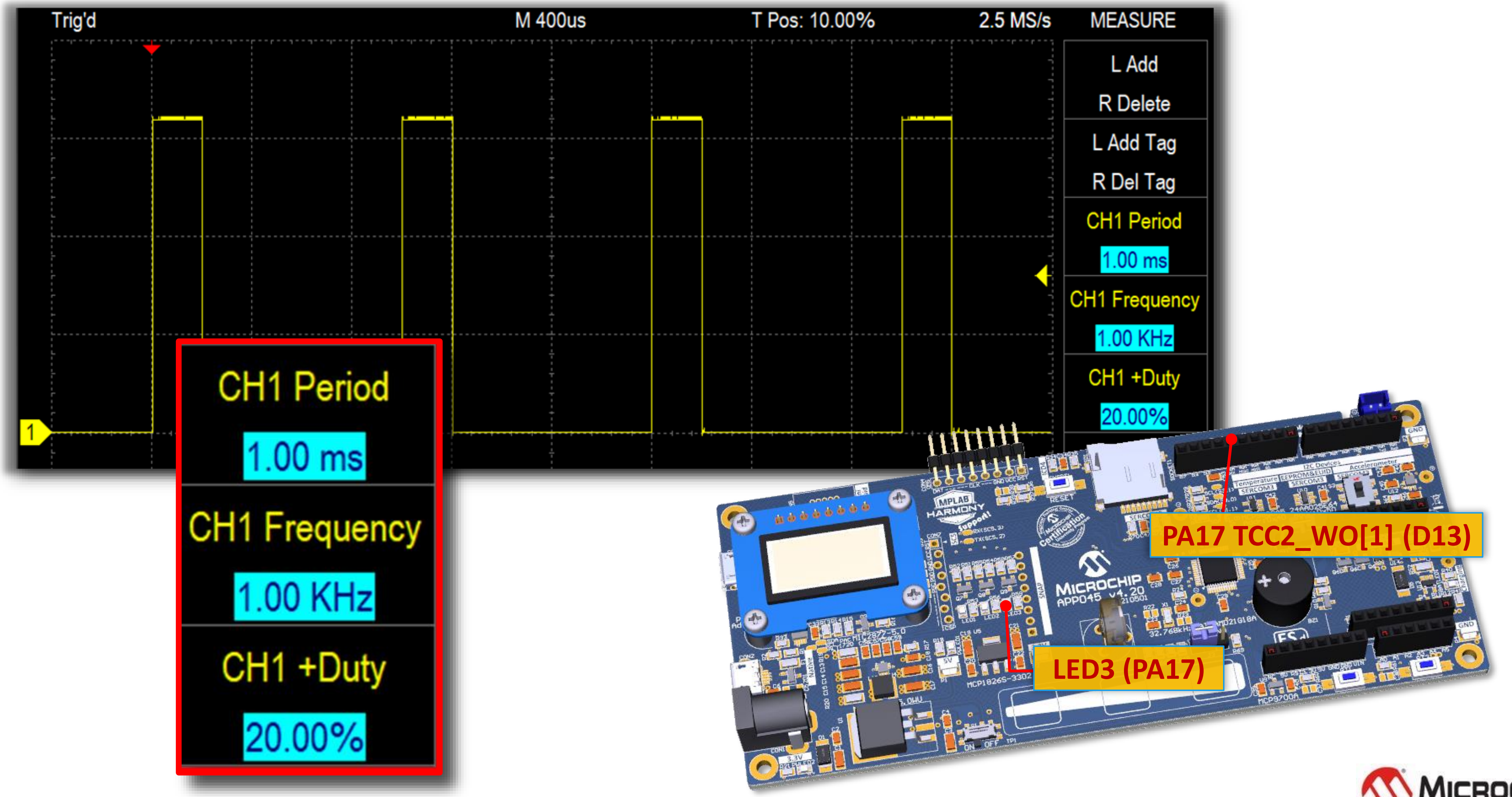
```
…

int main (void)
{
    …

    // TODO 14.01
    TCC2_PWMStart();

    while( true )
    {
        …
    }

}
```

b Program firmware to target board then observe result.

MICROCHIP

Lab14 TCC NPWM

Result



100% Duty Discuss

- When you want to set the PWM duty to 100%, the value of CCx must be set to the same as the actual PWM period count.
  - Period: PER = 48000-1 = 47999 (1kHz)
  - Duty: CCx = 48000 (100% Duty)
- TCCx Counter will not equal CCx, because the count value will be reset when equals PER. Observe the rules, this state of "Output Set to low" is not executed.
  - 1. TCCx Counter = CCx -> Output Set to low (Another rule is executed first)
  - 2. TCCx Counter = PER -> Output Set to high (Counter will be reset at the same time)

- Based on known issue, we can refer to the following example :

```
uint8_t PWM_Duty = 100;

TCC2_PWM16bitDutySet( 1, (PER + 1) * PWM_Duty );
```

Lab15 TCC NPWM ADC Duty Control

- **Use VR1 to adjust TCC2 PWM duty for LED3 dimming control.**
  - VR1 ADC Value = 0 -> TCC2 PWM Duty 0%
  - VR1 ADC Value = 4095 -> TCC2 PWM Duty 100%

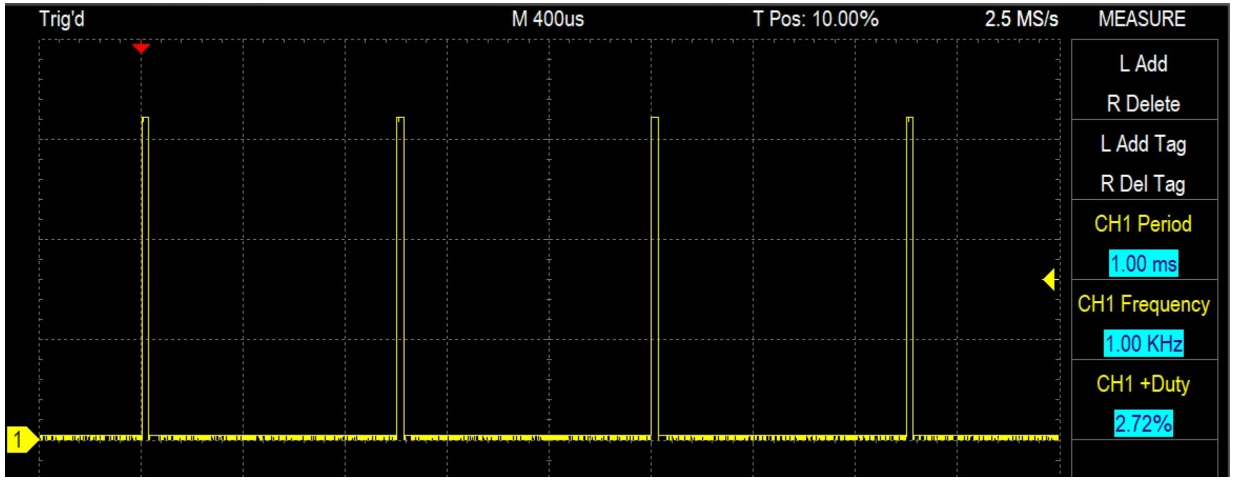
```
if( ADC_IsCompleted )
{
   ADC_IsCompleted = 0;
   …

   // TODO 15.01
   TCC2_PWM16bitDutySet( 1, (uint32_t)ADC_Result[0] * (TCC2_PWM16bitPeriodGet() + 1) / 4095 );
}
```

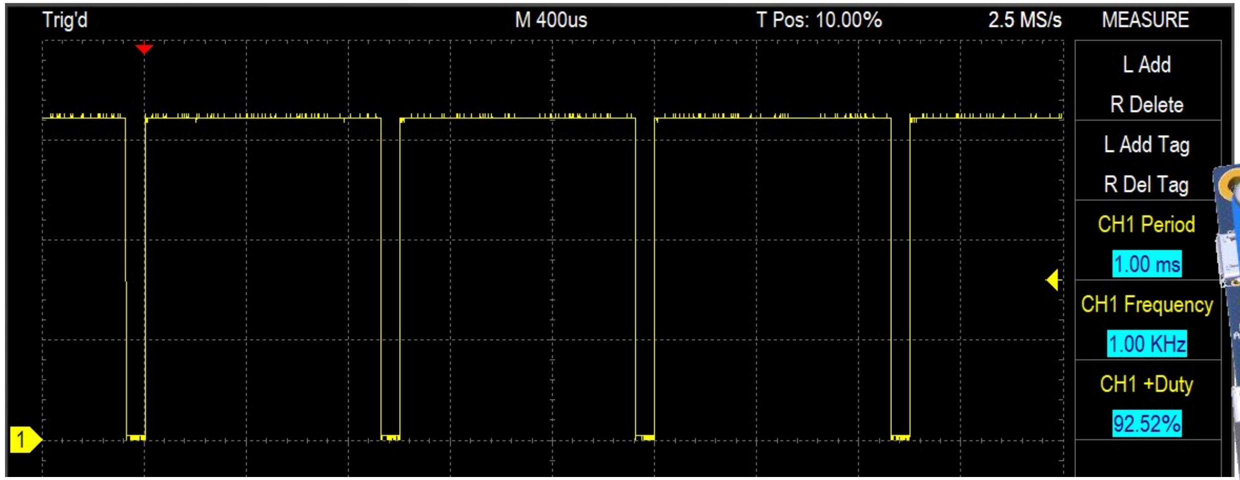
TCC2_ PWM_Peroid = 48000
ADC_Result[0] = 0 ~ 4095
ADC_Result[0] / 4095 = 0.0 ~ 1.0
ADC_Result[0] * TCC2_ PWM_Peroid / 4095 = 0 ~ 48000

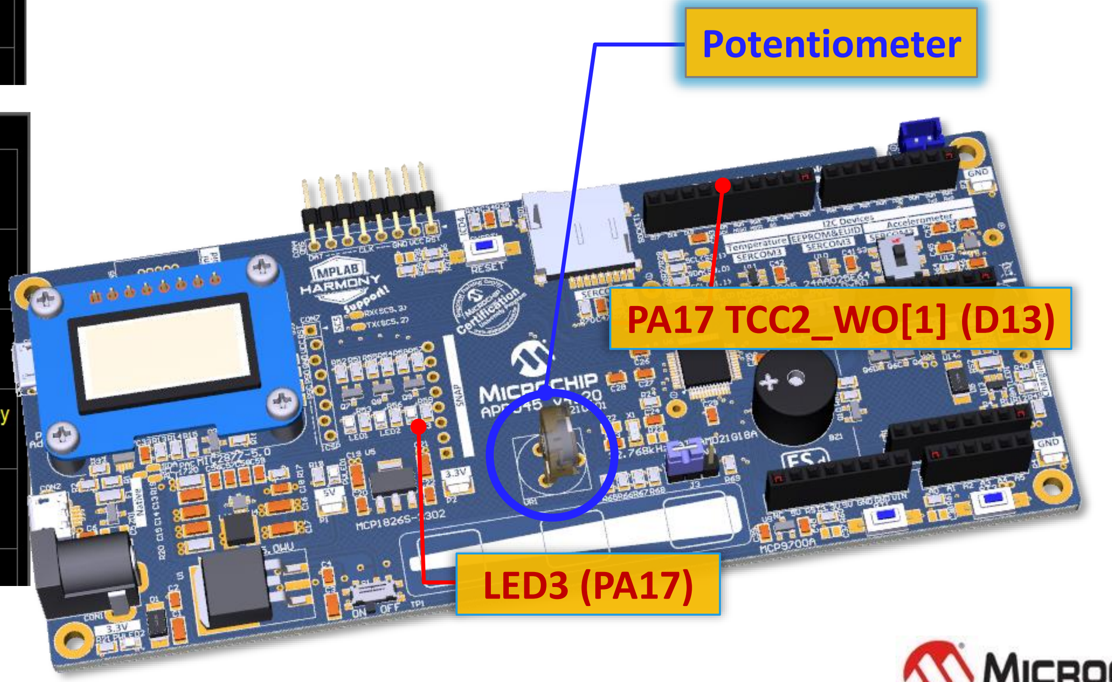
MICROCHIP

Lab15 TCC NPWM ADC Duty Control

Result







Lab16 TCC NPWM Auto Duty Control

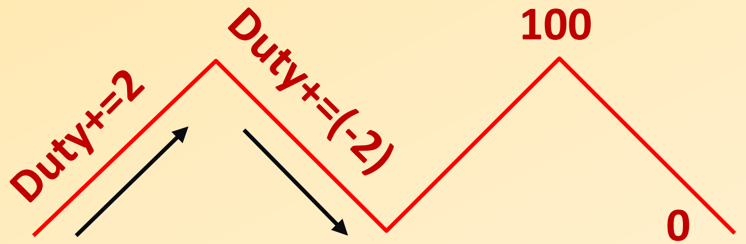
- **Remove or comment the following code in the previous Labs first.**

```
if (ADC_IsCompleted)
{
        …
        // TODO 16.01
        //  TCC2_PWM16bitDutySet( 1, (uint32_t)ADC_Result[0] * (TCC2_PWM16bitPeriodGet() + 1) / 4095 );
}
```

Lab16 TCC NPWM Auto Duty Control

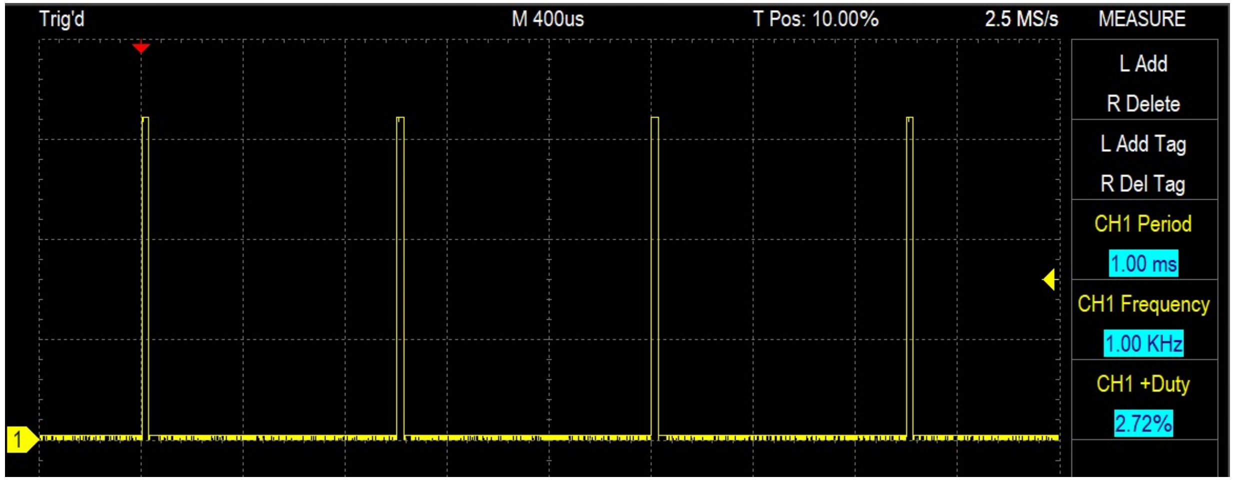
- Use **Timer(TC4)** to control PWM duty for **breathing light effect** on **LED3**. Duty cycle changes in steps of 2% every 50ms intervals.
  (Duty 0% -> Duty 100% -> Duty 0% -> … repeat)

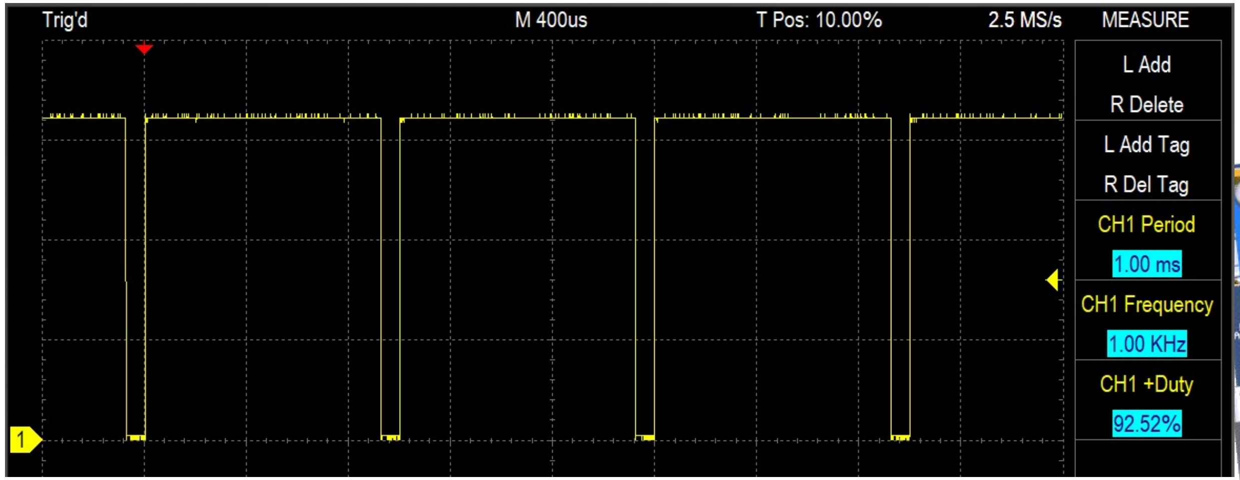
```
// TODO 16.02
uint8_t Duty = 50;
int8_t DutyDistance = 2;

int main (void)
{ …
   if (TC4_HasExpired)
   {
      TC4_HasExpired = 0;
      …
      // TODO 16.03
      Duty += DutyDistance;
      if (Duty >= 100 || Duty <= 0)
        DutyDistance = -DutyDistance;

      TCC2_PWM16bitDutySet( 1, ((uint32_t)Duty * (TCC2_PWM16bitPeriodGet() + 1)) / 100 );
      …
```

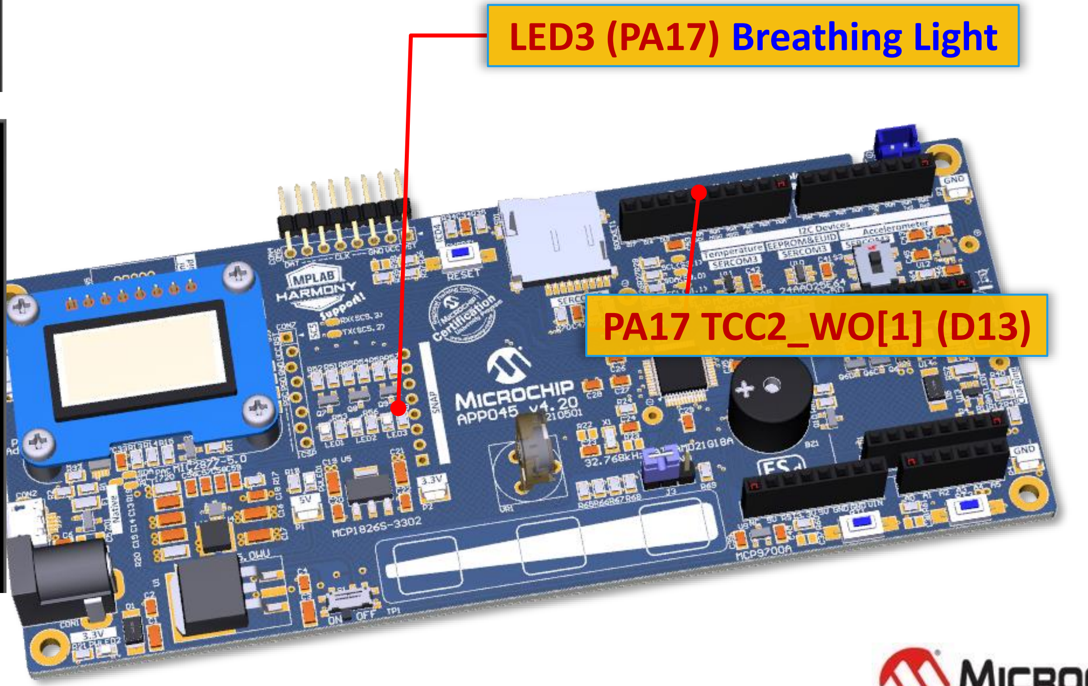


Lab16 TCC NPWM Auto Duty Control

Result





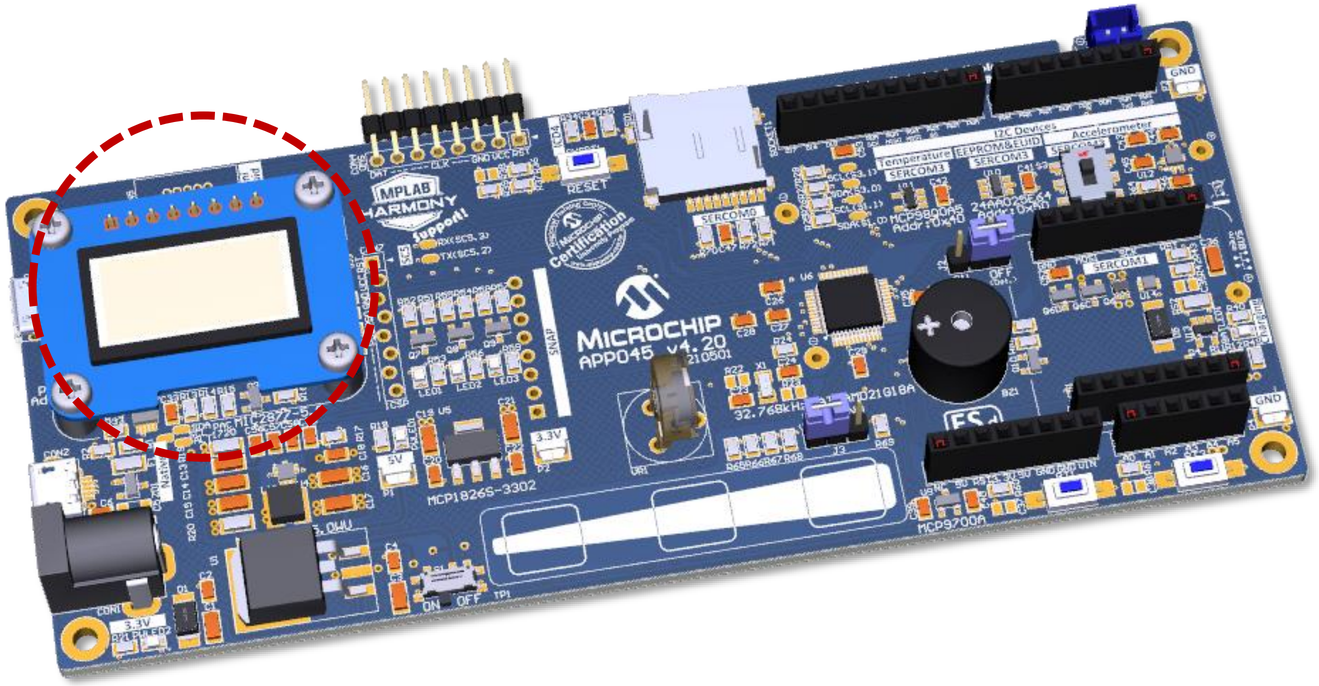


SERCOM-SPI (OLED SSD1306)

OLED Graphic Display

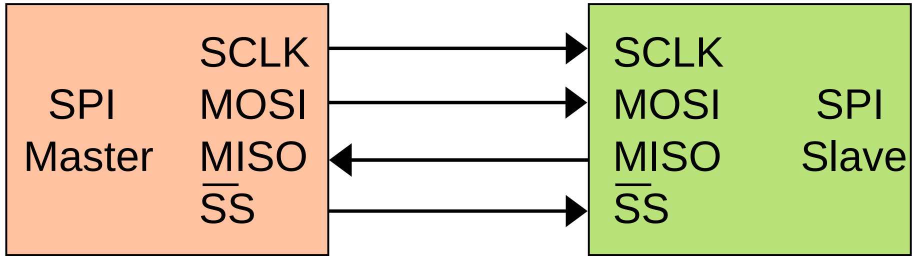
- **SSD1306**

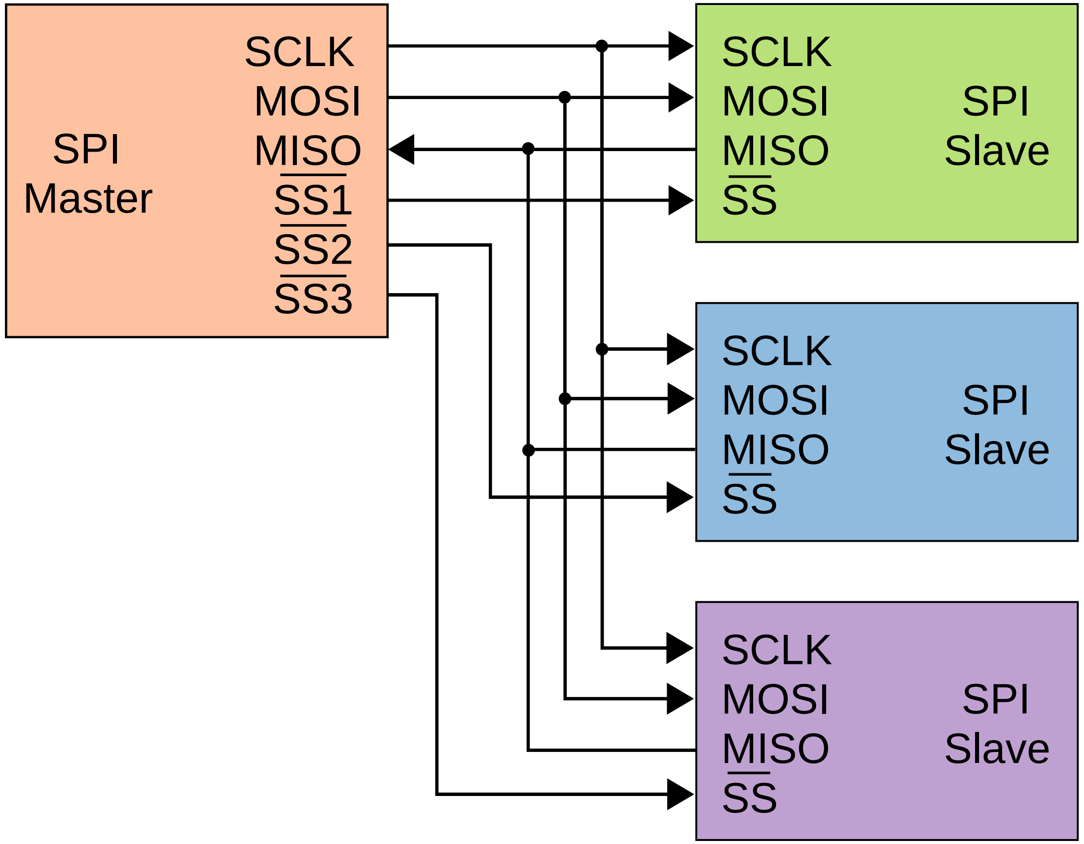
  128 x 64 Dot Matrix OLED/PLED Segment/Common Driver with Controller and SPI communication interface.



What's SPI

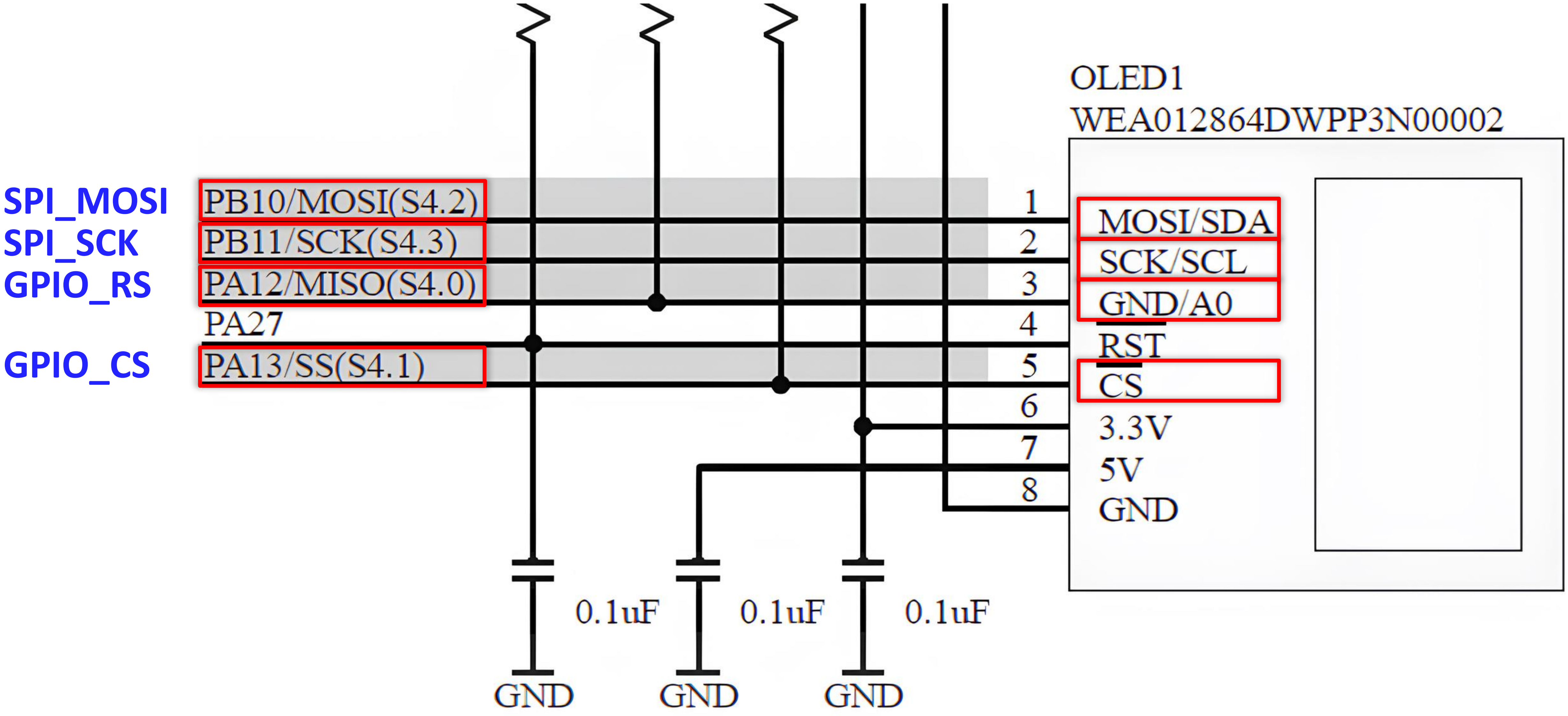
- **SPI, Serial Peripheral Interface**
- **It's a serial communication interface.**
  Synchronous, inside-board level, short distance communication.





https://zh.wikipedia.org/wiki/%E5%BA%8F%E5%88%97%E5%91%A8%E9%82%8A%E4%BB%8B%E9%9D%A2

APP045 OLED SSD1306 Schematic



Lab17 SERCOM - SPI SSD1306 OLED

- Add SERCOM4-SPI module to your project.
- Add OLED control function to shows bitmap logo on your OLED display.
- Pre-build Logo Image and Font header file at

.\Appendix\LogoFont\Microchip_Logo.h

.\Appendix\LogoFont\Alphabet_Fonts.h

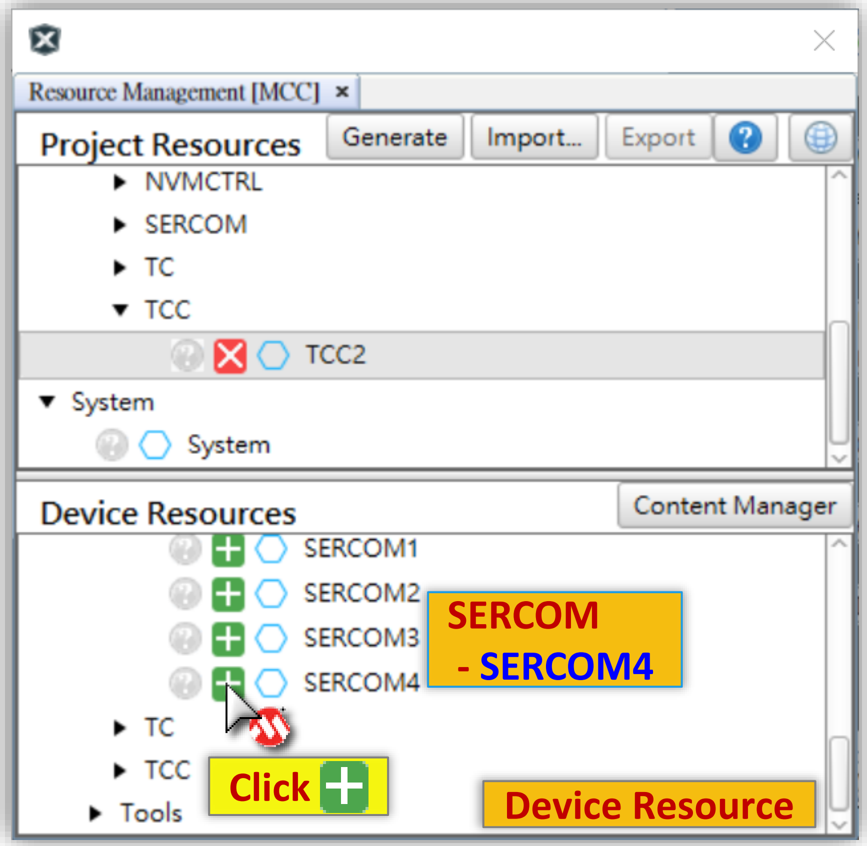
- Let’s go!

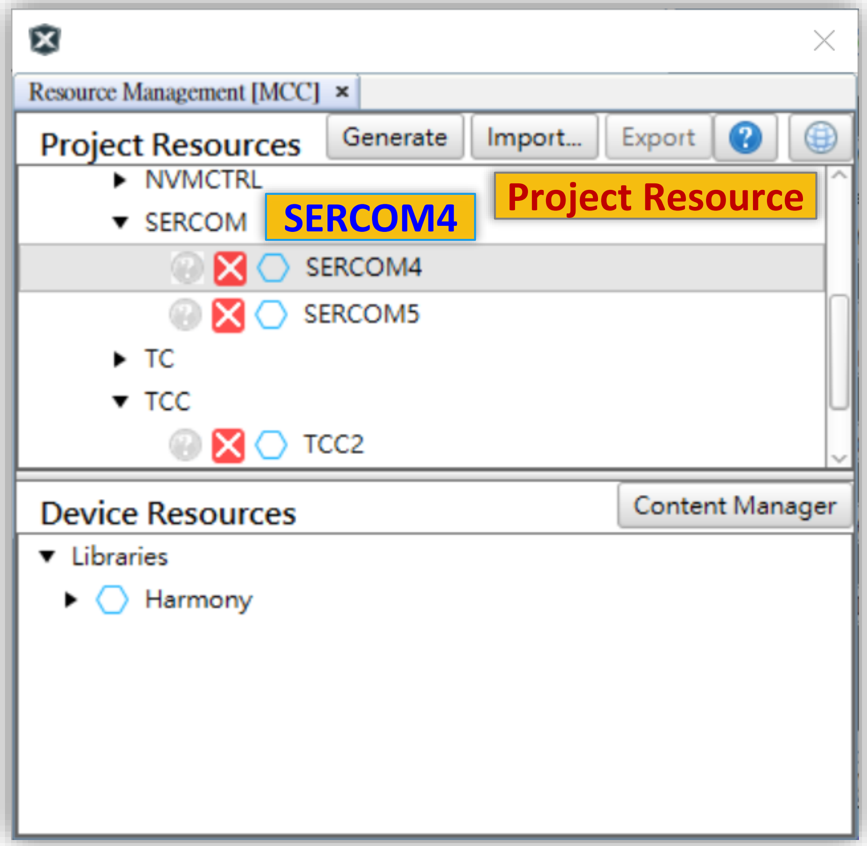
MICROCHIP

Lab17 SERCOM - SPI SSD1306 OLED

Step 1

- Find SERCOM4 component in Device Resource window.
- Click ➕ to add SERCOM4 to Project Resource window.

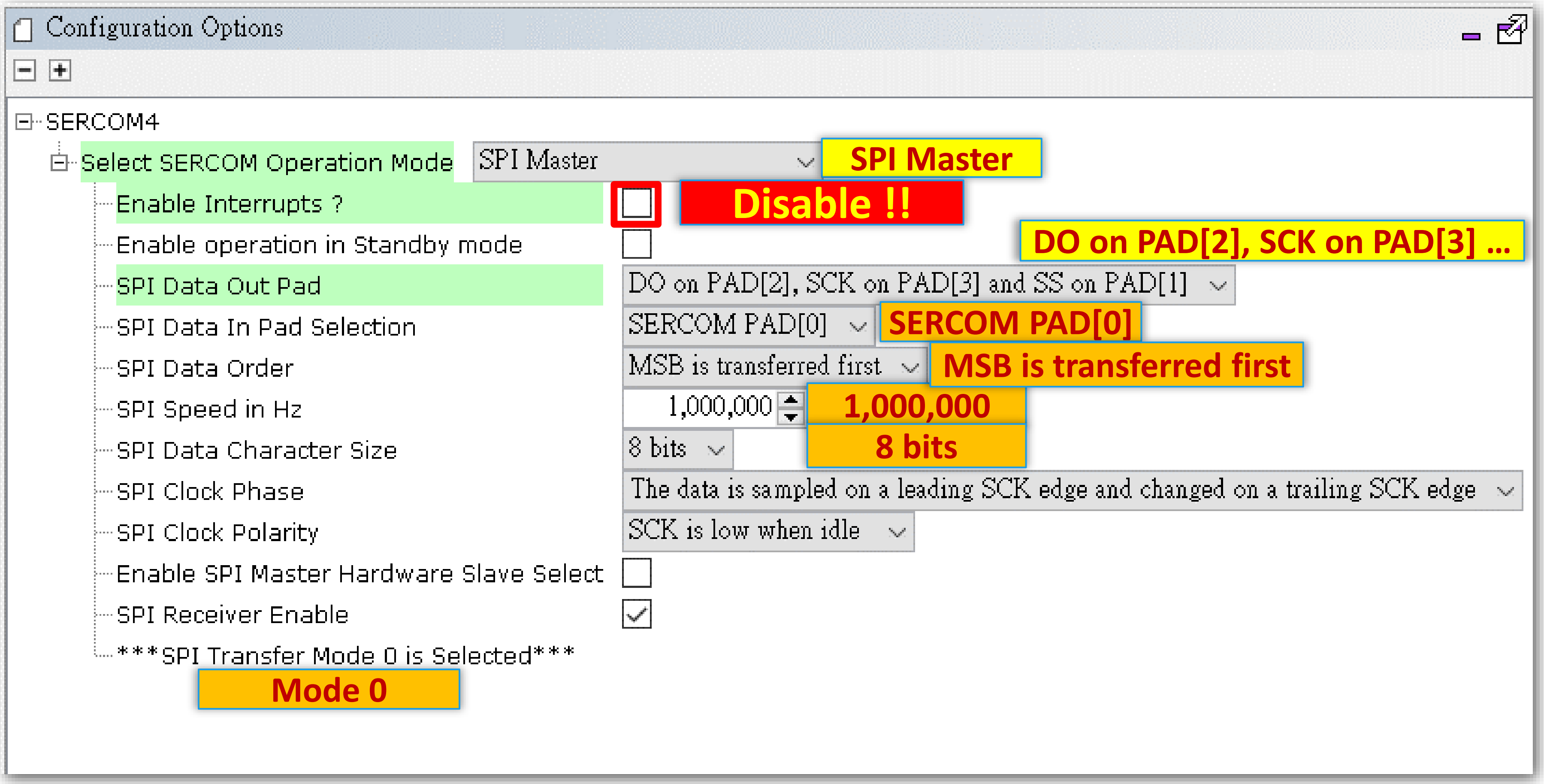




Lab17 SERCOM - SPI SSD1306 OLED

Step 2

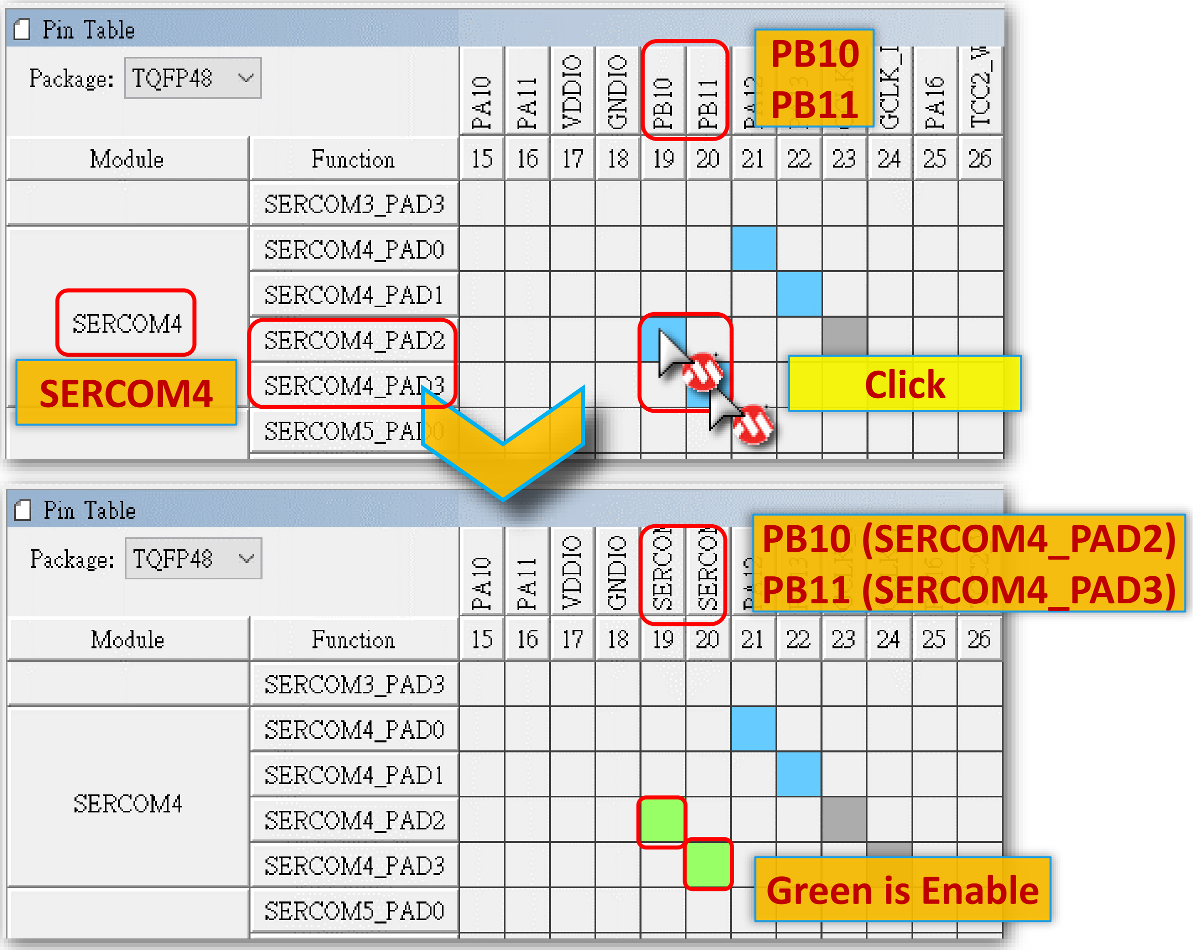
- **Setup SERCOM4 SPI Configuration.**



Lab17 SERCOM - SPI SSD1306 OLED

Step 3

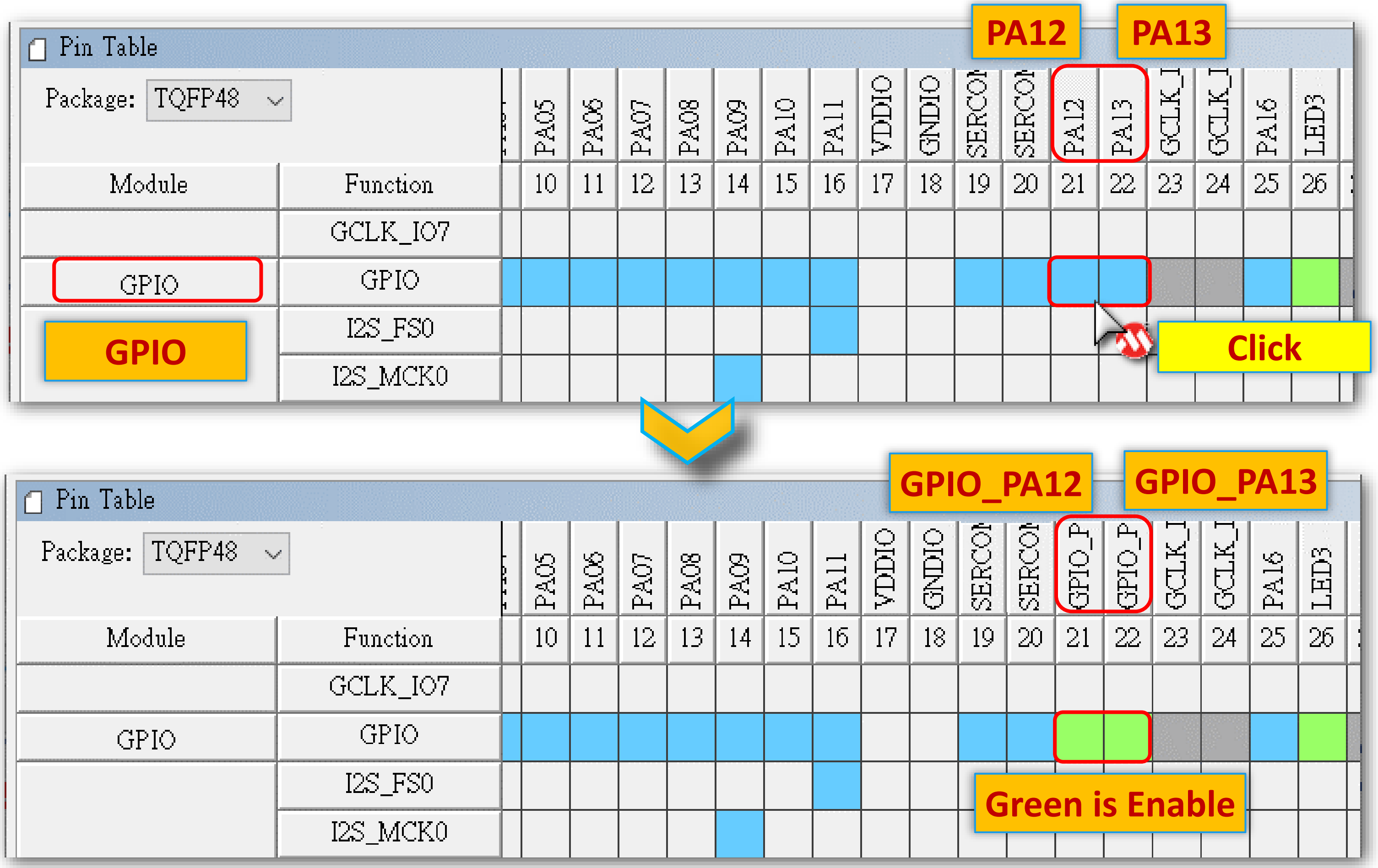
- **Pin Table Configuration of SERCOM4**



Lab17 SERCOM - SPI SSD1306 OLED

Step 4

- **Pin Table Configuration of GPIO**



Lab17 SERCOM - SPI SSD1306 OLED

Step 5

- **Pin Setting of GPIO for direction and custom name.**

Pin Settings

Order: Pins | Table View | ☑ Easy View

Pin Number	Pin ID	Custom Name	Function	Mode	Direction	Latch	Pull Up	Pull Down	Drive Strength
20				Digital	Hi		☐	☐	NORMAL
21	PA12	SSD1306_RS_PIN	GPIO	Digital	Out	High			NORMAL
22	PA13	SSD1306_CS_PIN	GPIO	Digital	Out	High			NORMAL
23				Digital	Hi		☐	☐	NORMAL
24	PA15			Digital	High Impedance	n/a	☐	☐	NORMAL
25	PA16		Available	Digital	High Impedance	Low	☐	☐	NORMAL
26	PA17	LED3	GPIO	Digital	Out	Low			NORMAL
27	PA18		TC3_WO0	Digital	High Impedance	n/a	☐	☐	NORMAL
28	PA19		TC3_WO1	Digital	High Impedance	n/a	☐	☐	NORMAL
29	PA20	LED1	GPIO	Digital	Out	Low			NORMAL
30	PA21	LED2	GPIO	Digital	Out	Low			NORMAL

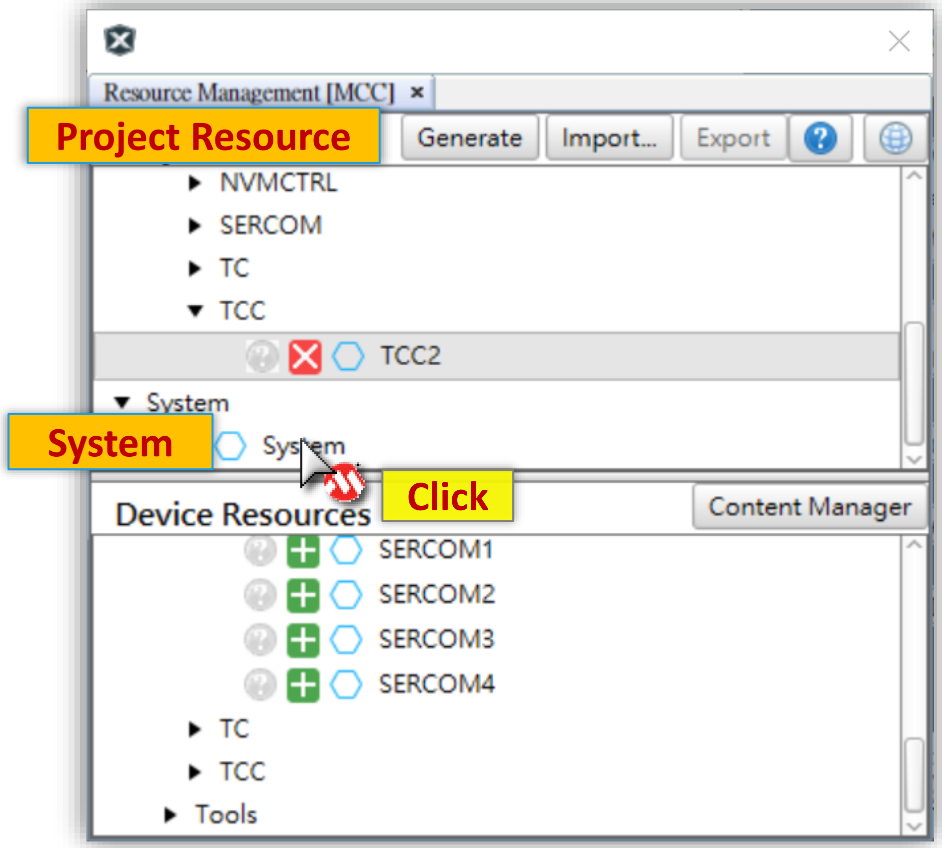
PA12 | SSD1306_RS | Out | High

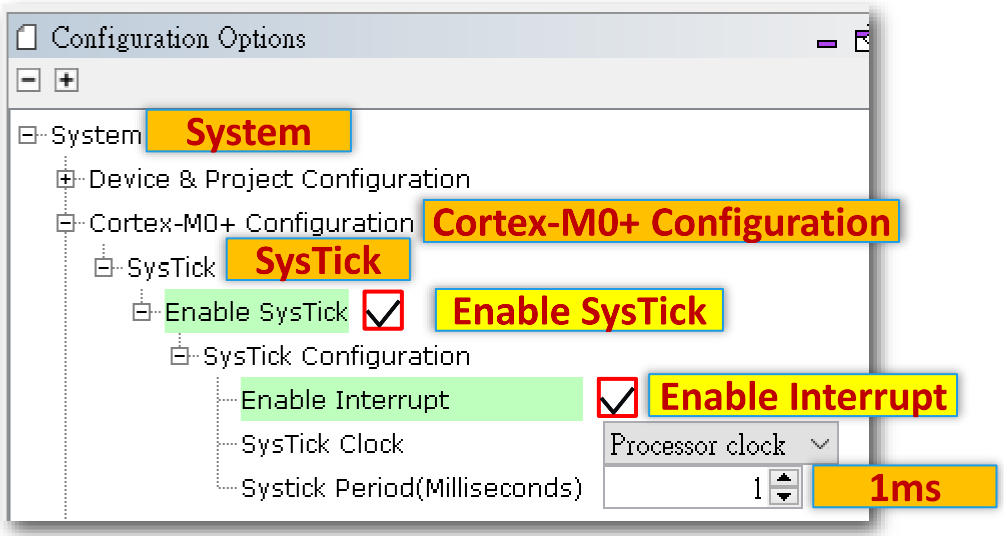
PA13 | SSD1306_CS | Out | High

Lab17 SERCOM - SPI SSD1306 OLED

Step 6

- Select exist module System in Project Resource.
- Enable SysTick and Enable Interrupt for Delay Function.

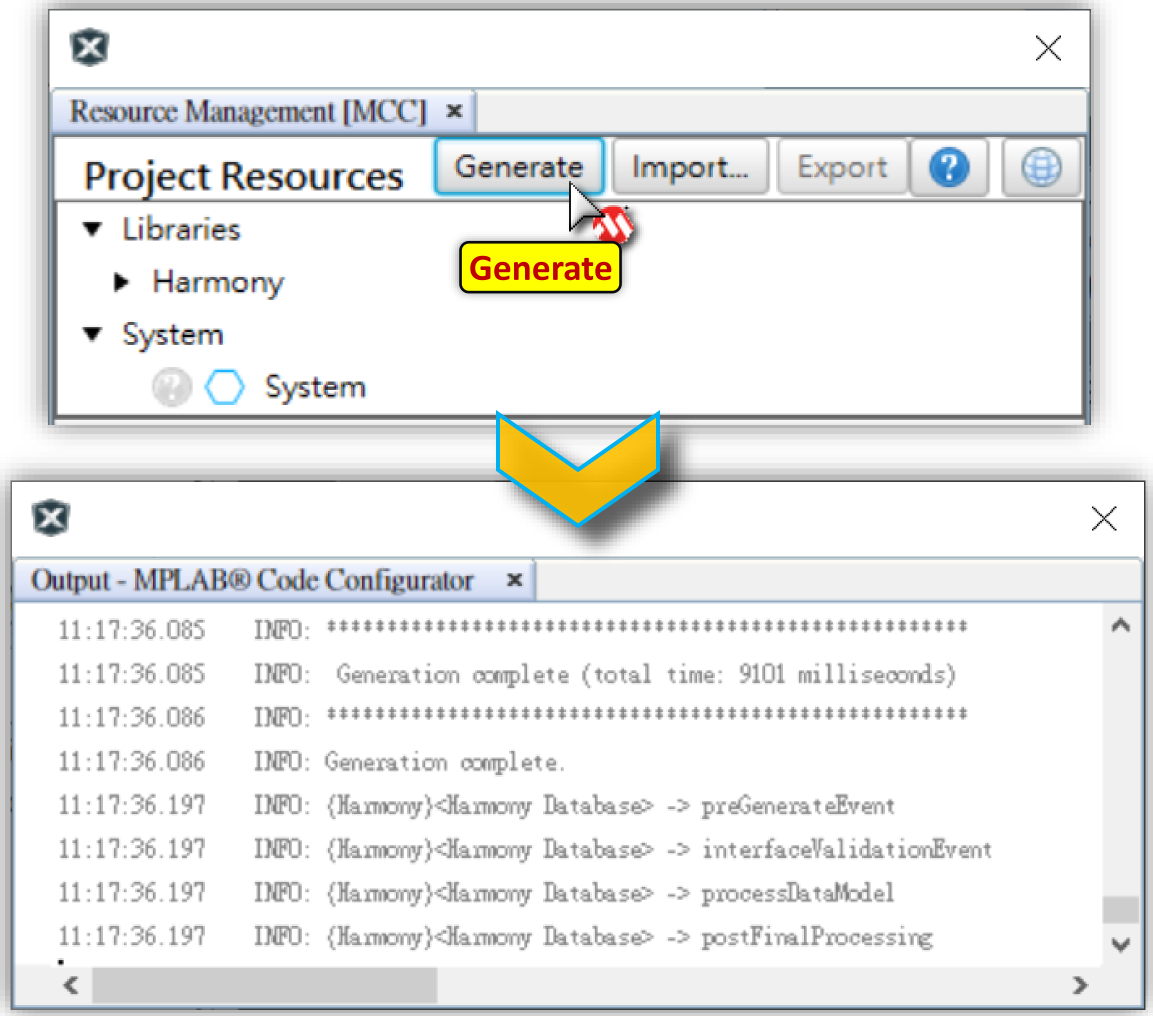




Lab17 SERCOM - SPI SSD1306 OLED

Step 7

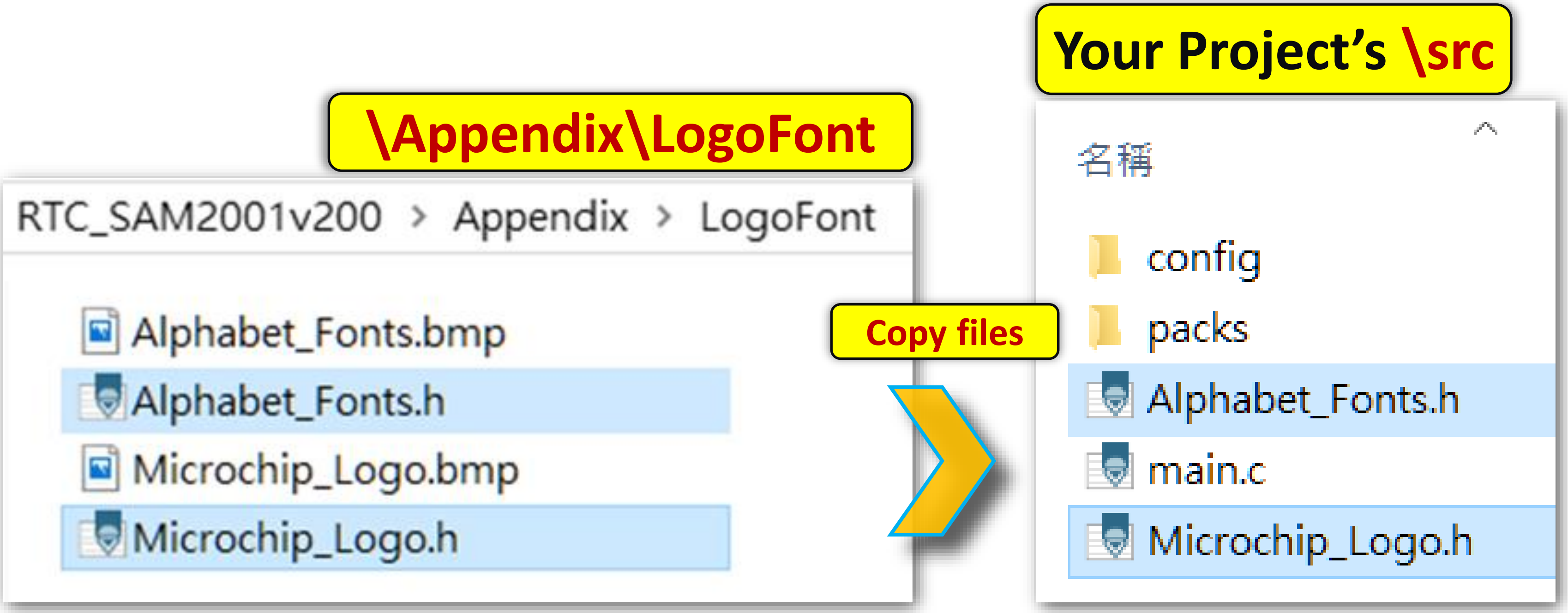
- Click "Generate" Button to Generate your Code.



Lab17 SERCOM - SPI SSD1306 OLED

Step 8

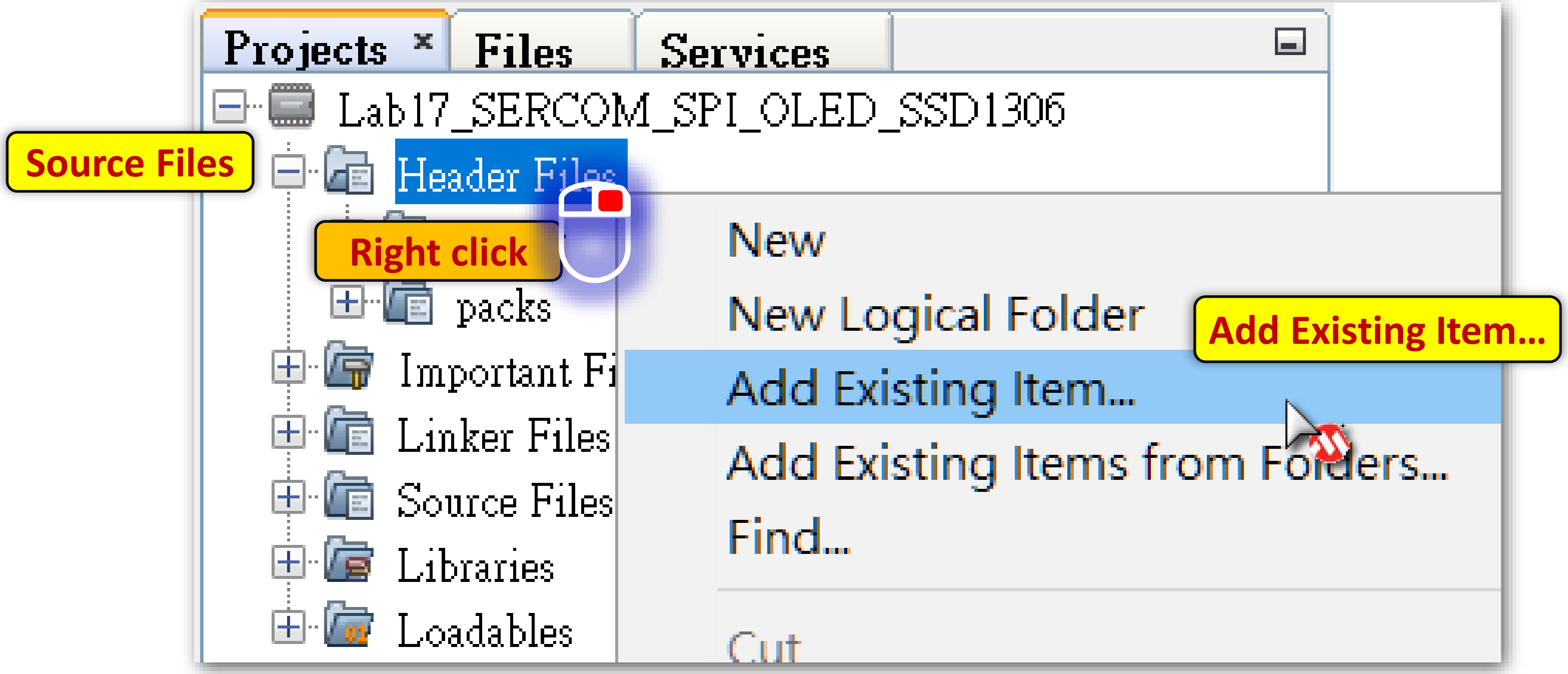
- Copy below files to your project's \src folder
  Logo image file : Microchip_Logo.h
  Font image file : Alphabet_Fonts.h



Lab17 SERCOM - SPI SSD1306 OLED

Step 9

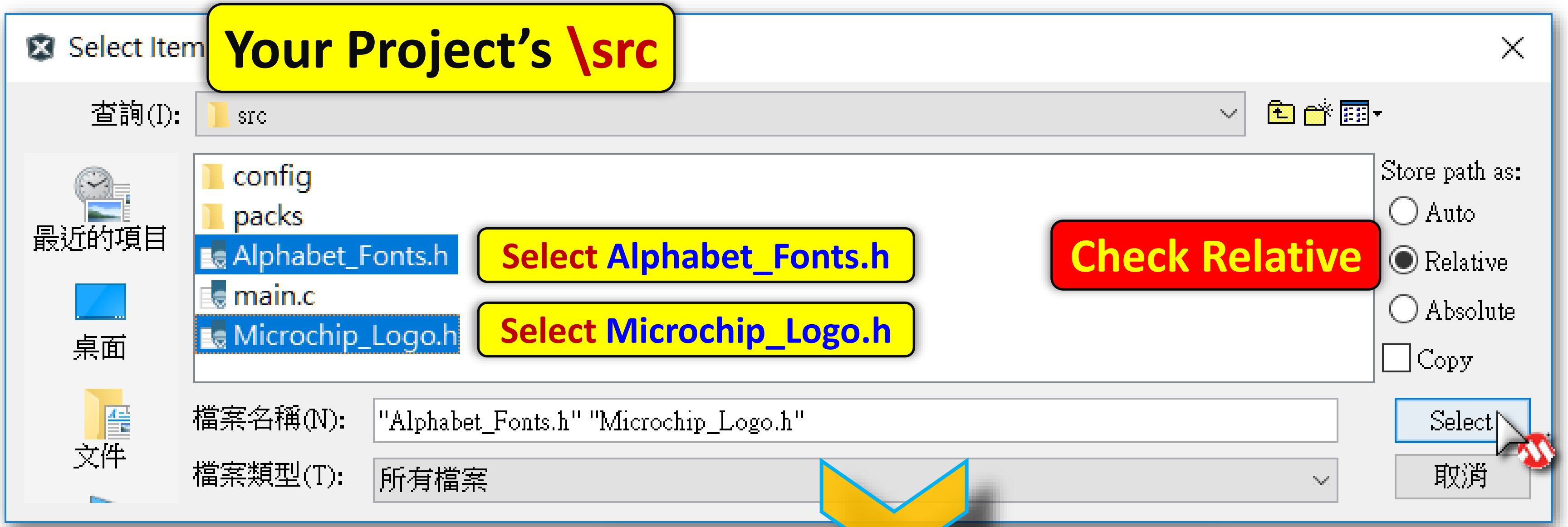
- **Add pre-build logo image header files into project**
  Right click Header Files > Add Existing item…



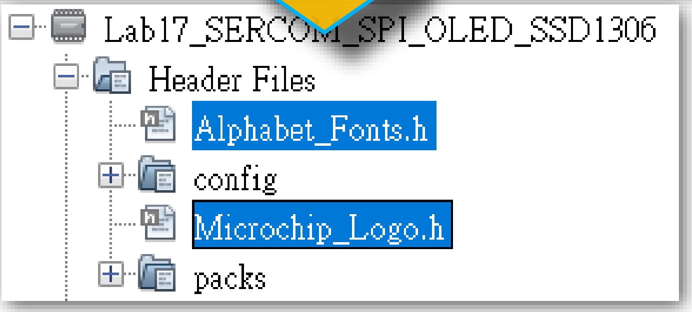
Lab17 SERCOM - SPI SSD1306 OLED

Step 10

- Select folder in your project's \src
- Add Alphabet_Fonts.h and Microchip_Logo.h into project.



Alphabet_Fonts.h
Microchip_Logo.h
added in "Header Files"



MICROCHIP

Lab17 SERCOM - SPI SSD1306 OLED

Step 11

a Add code segment to your project

```
// TODO 17.01
#include "Microchip_Logo.h"
#include "Alphabet_Fonts.h"
…
// TODO 17.02
#define LCM_WIDTH 128
#define LCM_HEIGHT 64
#define FONT_WIDTH  7
#define FONT_HEIGHT 8
uint8_t LogoX, LogoY;
unsigned char LCM_Blank[1024];
char Alphabet[95] = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijk\
                     lmnopqrstuvwxyz~!@#$%^&()[]{}_\\|;:,.`'\"<>+-*/=? ";

const uint8_t LCM_InitCMD[] = {
  0xAE, 0xD5, 0x80, 0xA8, 0x3F, 0xD3, 0x00, 0x40, 0x8D, 0x14, 0x20, 0x02, 0xA1,
  0xC8, 0xDA, 0x12, 0x81, 0xCF, 0xD9, 0xF1, 0xDB, 0x40, 0xA4, 0xA6, 0xAF
};
…
```

MICROCHIP

Lab17 SERCOM - SPI SSD1306 OLED

Step 12

a Add code segment to your project

```
// TODO 17.03
...
void LCM_SetCursor( uint8_t x, uint8_t y )
{
  uint8_t OUT[3] = { 0x0F & x, 0x10 + (x >> 4), 0xB0 + y };
  SSD1306_RS_PIN_Clear();
  SERCOM4_SPI_Write(( void* )OUT, 3);
}

void LCM_DrawBitmap( uint8_t sx, uint8_t sy, uint8_t width,
                                 uint8_t height, const unsigned char *byte )
{
  for( int y = 0; y < (height/8); y++ )
  {
    LCM_SetCursor(sx, sy + y);
    SSD1306_RS_PIN_Set();
    SERCOM4_SPI_Write(( void* )byte, (width - sx));
    byte += width;
  }
}
...
```

Lab17 SERCOM - SPI SSD1306 OLED

Step 13

a Add code segment to your project

```
// TODO 17.03 (continue)
…
void LCM_DrawFont( uint8_t sx, uint8_t row, char ch )
{
  int chIdx = 0;

  while( ch != Alphabet[chIdx] ) { chIdx++; }
  LCM_SetCursor(sx, row);
  SSD1306_RS_PIN_Set();
  SERCOM4_SPI_Write( ( void* )(Alphabet_Font+(chIdx*FONT_WIDTH)),
                     FONT_WIDTH );
}

void LCM_DrawString( uint8_t sx, uint8_t row, char *str )
{
  for( int x=0 ; x<strlen(str); x++ )
  {
    LCM_DrawFont( sx+(x*FONT_WIDTH), row, str[x] );
  }
}
…
```

Lab17 SERCOM - SPI SSD1306 OLED

Step 14

a Add code segment to your project

```
…
int main (void)
{ …
   TCC2_PWMStart();

   // TODO 17.04
   SYSTICK_TimerStart();
   SSD1306_CS_PIN_Clear();
   SSD1306_RS_PIN_Clear();
   SERCOM4_SPI_Write(( void* )LCM_InitCMD, sizeof (LCM_InitCMD));
   memset(LCM_Blank, 0, 1024);
   LCM_DrawBitmap(0, 0, LCM_WIDTH, LCM_HEIGHT, LCM_Blank);
   LogoX = 127;
   LogoY = 0;
   for( i = 0; i < 128; i += 2 )
   {
     LCM_DrawBitmap(LogoX - i, LogoY, LCM_WIDTH, LCM_HEIGHT, Microchip_Logo);
     SYSTICK_DelayMs((i * i) / 400);
   }
   …
```

MICROCHIP

Lab17 SERCOM - SPI SSD1306 OLED

Step 15

a Add code segment to your project

```
int main (void)
{ …
  // TODO 17.04 (continue)
   …
   SYSTICK_DelayMs(1000);
   memset(LCM_Blank, 0, 1024);
   LCM_DrawBitmap(0, 0, LCM_WIDTH, LCM_HEIGHT, LCM_Blank);
   sprintf( (char *)USARTRTxBuffer, "SysClock %ld.%-ldMHz",
                               SYSTICK_TimerFrequencyGet()/1000000,
                              (SYSTICK_TimerFrequencyGet()%1000000)/1000 );
   LCM_DrawString( 0, 0, (char *)USARTRTxBuffer );
   sprintf( (char *)USARTRTxBuffer, "TC3 period %ldms",
                 (TC3_Timer16bitPeriodGet()+1)*1000/TC3_TimerFrequencyGet() );
   LCM_DrawString( 0, 1, (char *)USARTRTxBuffer );
   sprintf( (char *)USARTRTxBuffer, "TC4 period %ldms",
                 (TC4_Timer16bitPeriodGet()+1)*1000/TC4_TimerFrequencyGet() );
   LCM_DrawString( 0, 2, (char *)USARTRTxBuffer );
   sprintf( (char *)USARTRTxBuffer, "Received Data :" );
   LCM_DrawString( 0, 7, (char *)USARTRTxBuffer );
   …
```

Lab17 SERCOM - SPI SSD1306 OLED

Step 16

a Add code segment to your project

```
while ( true )
{ …
   if (TC3_ HasExpired)
   { …
      LED1_Toggle();
      …
       // TODO 17.05
       sprintf( (char *)USARTRTxBuffer, "Hello World!");
       LCM_DrawString( 0, 3, (char *)USARTRTxBuffer );
}

   if (TC4_ HasExpired)
   { …
       LED2_Toggle();
      …
      // TODO 17.06
       sprintf( (char *)USARTRTxBuffer, "PWM Duty : %3d%%", Duty);
       LCM_DrawString( 0, 4, (char *)USARTRTxBuffer );
      }
}
```

Lab17 SERCOM - SPI SSD1306 OLED

Step 17

a Add code segment to your project

```
while ( true )
{ …
  if (USART5_IsReceived)
  { …
    // TODO 17.07
    sprintf( (char *)USARTRTxBuffer, "Received Data : %1c",
                                                        USART5_ReceiveData[0] );
    LCM_DrawString( 0, 7, (char *)USARTRTxBuffer );
    …
  }

  if (ADC_IsCompleted)
  { …
    // TODO 17.08
    sprintf( (char *)USARTRTxBuffer, "VR1  : %4d", ADC_Result[0]);
    LCM_DrawString( 0, 5, (char *)USARTRTxBuffer );
    sprintf( (char *)USARTRTxBuffer, "Temp : %4d", ADC_Result[1]);
    LCM_DrawString( 0, 6, (char *)USARTRTxBuffer );
    …
  }
}
```

OLED Functions

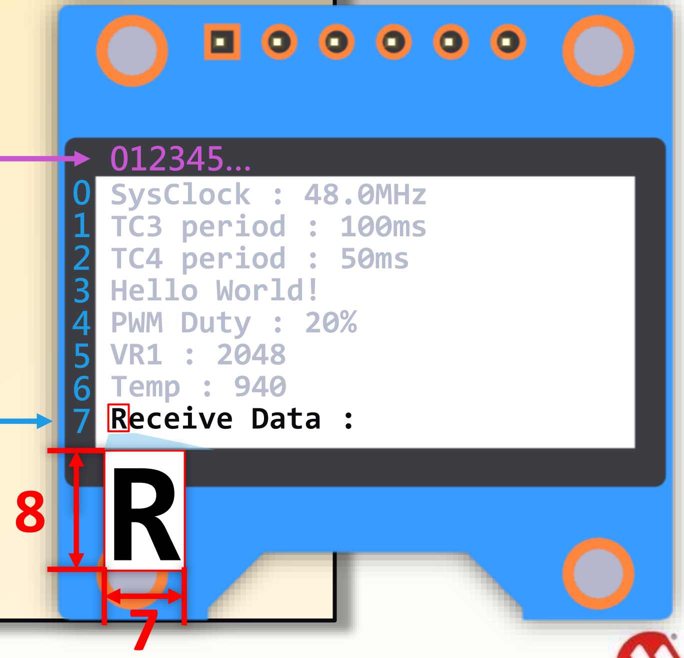
Interface Routines

```
// Delay some time (ms)
void SYSTICK_DelayMs( uint32_t delay_ms);
// Set cursor position
void LCM_SetCursor( uint8_t x, uint8_t y );
// Draw a font
void LCM_DrawFont( uint8_t sx, uint8_t row, char ch ) ;
// Draw a string
void LCM_DrawString( uint8_t sx, uint8_t row, char *str ) ;
// Draw an image
void LCM_DrawBitmap( uint8_t sx, uint8_t sy, uint8_t width,
                     uint8_t height, const unsigned char *byte ) ;
```

OLED Draw Font

Hint

```
#include "Alphabet_Fonts.h"
…
#define FONT_WIDTH  7
#define FONT_HEIGHT 8
char Alphabet[95] = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijk\
                     lmnopqrstuvwxyz~!@#$%^&()[]{}_\\|;:,.`'\"<>+-*/=? ";
…
void LCM_DrawFont( uint8_t sx, uint8_t row, char ch ) {…}
…
void LCM_DrawString( uint8_t sx, uint8_t row, char *str ) {…}
…

int main (void)
{
  …
  sprintf( (char *)USARTRTxBuffer, "Received Data :" );
  LCM_DrawString( 0, 7, (char *)USARTRTxBuffer );

  …
}
```

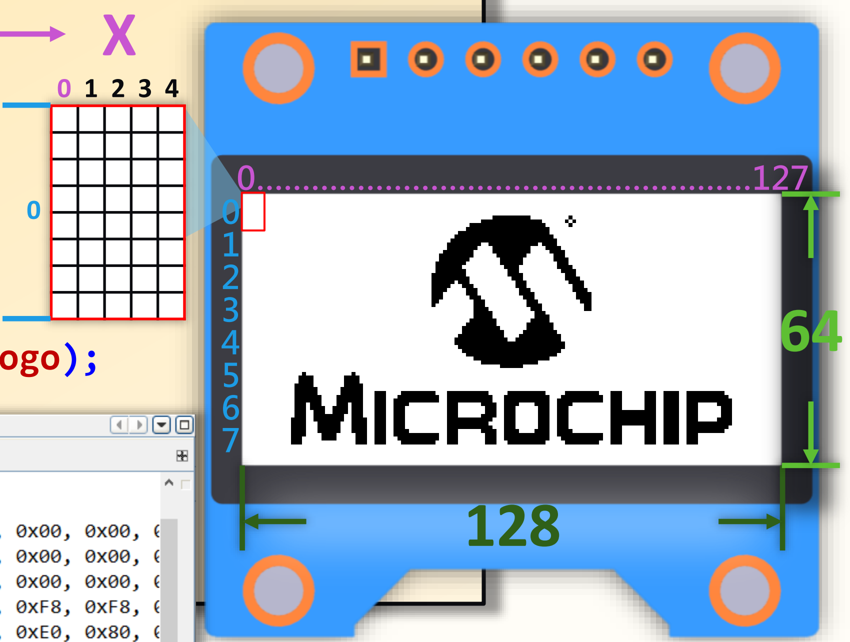


OLED Draw Image

Hint

```
#include "Microchip_Logo.h"
…
#define LCM_WIDTH 128
#define LCM_HEIGHT 64
…
void LCM_DrawBitmap( uint8_t sx, uint8_t sy, uint8_t width,
                                  uint8_t height, const unsigned char *byte ) {…}
…

int main (void)
{
  …
  LCM_DrawBitmap(0, 0, LCM_WIDTH, LCM_HEIGHT, Microchip_Logo);

  …
}
```

Y_row 0



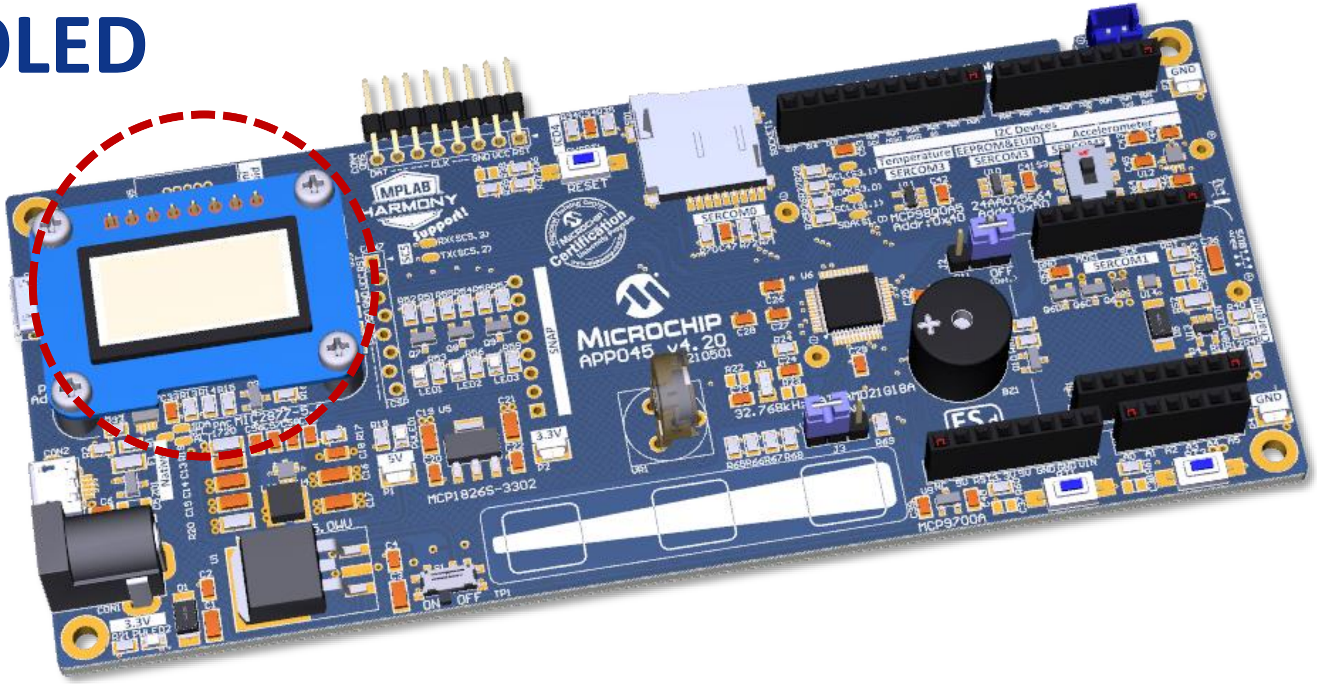
Microchip_Logo.h

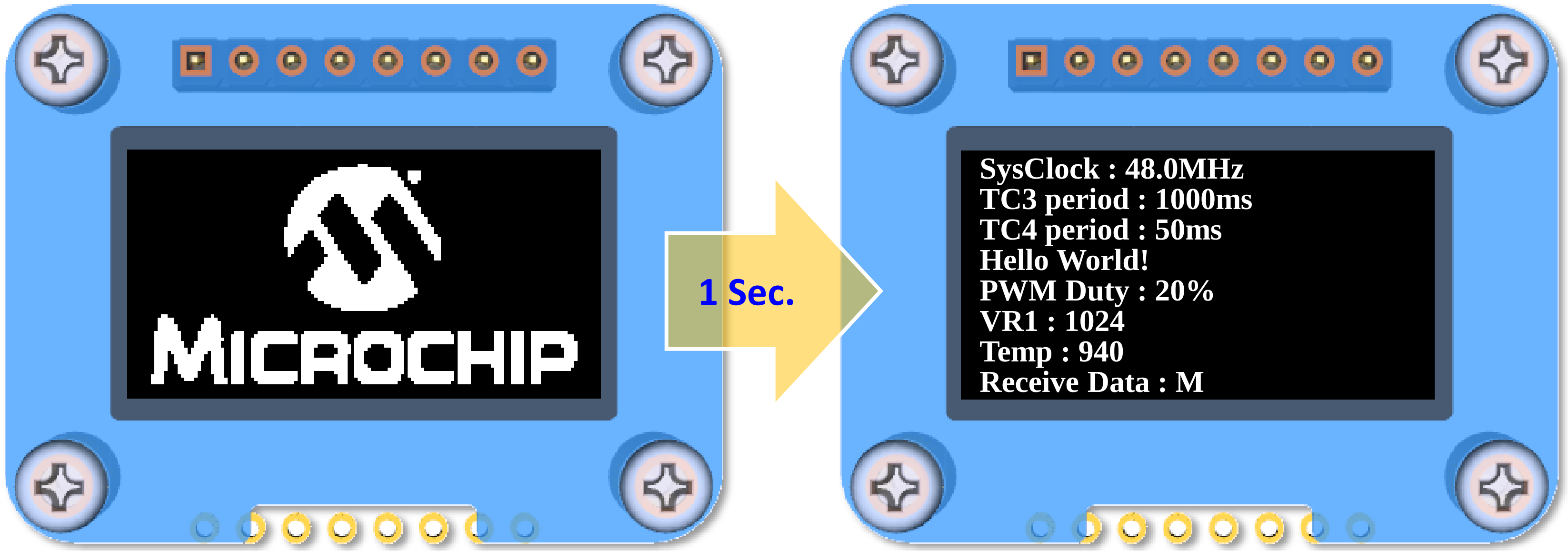
Source History

```
const unsigned char Microchip_Logo [] = {
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x80, 0xC0, 0xC0, 0xE0, 0xF0, 0xF0, 0xF0, 0xF8, 0xF8, 0xF8,
0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xF8, 0xF8, 0xF8, 0xF0, 0xE0, 0x80,
0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
```

Lab17 SERCOM - SPI SSD1306 OLED

Result





Lab17.1 SERCOM - SPI SSD1306 OLED

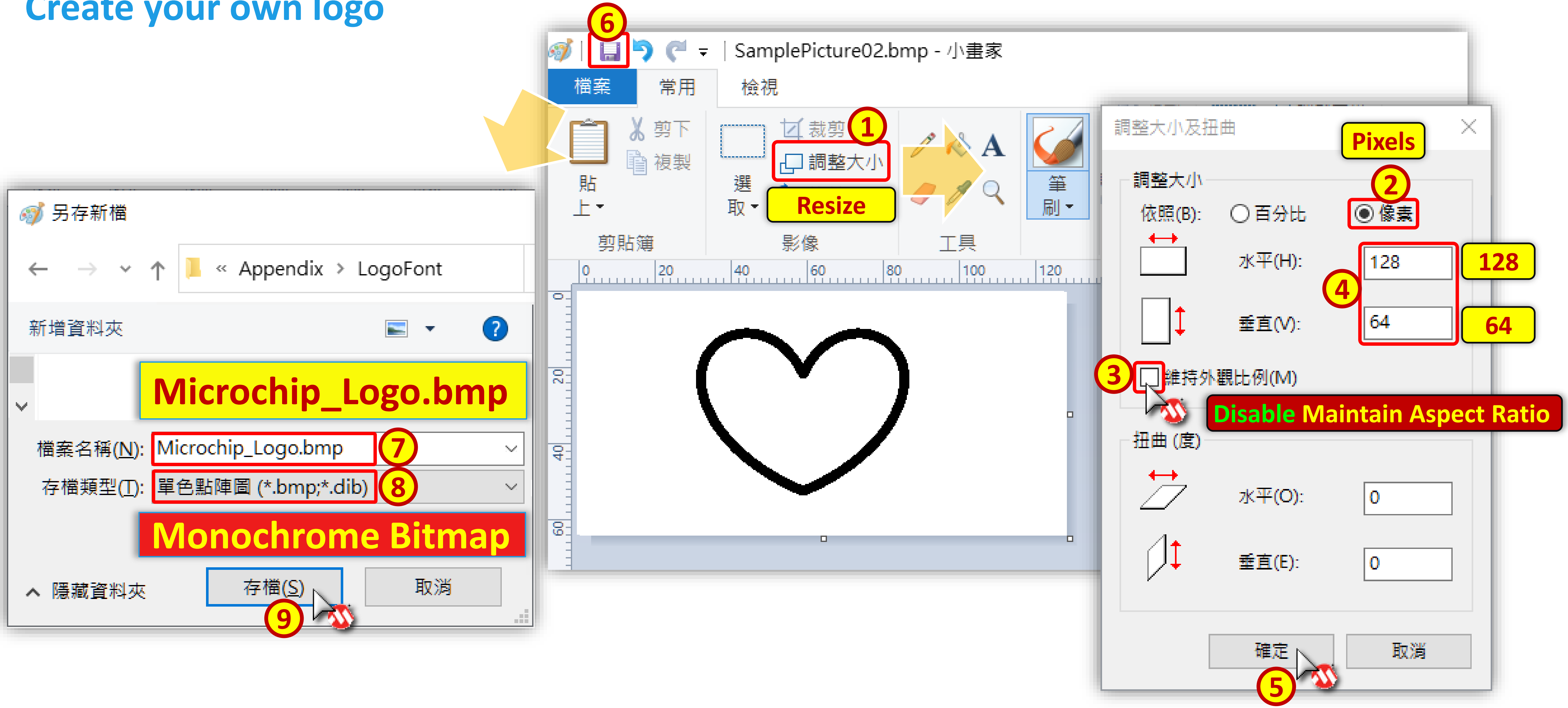
- To create your personally welcome page (logo) !
- Create your own Logo and replace original one.
- Logo image create application at

  .\Tools\LCDAssistant.exe

- Let’s go!

Lab17.1 SERCOM - SPI SSD1306 OLED

Create your own logo

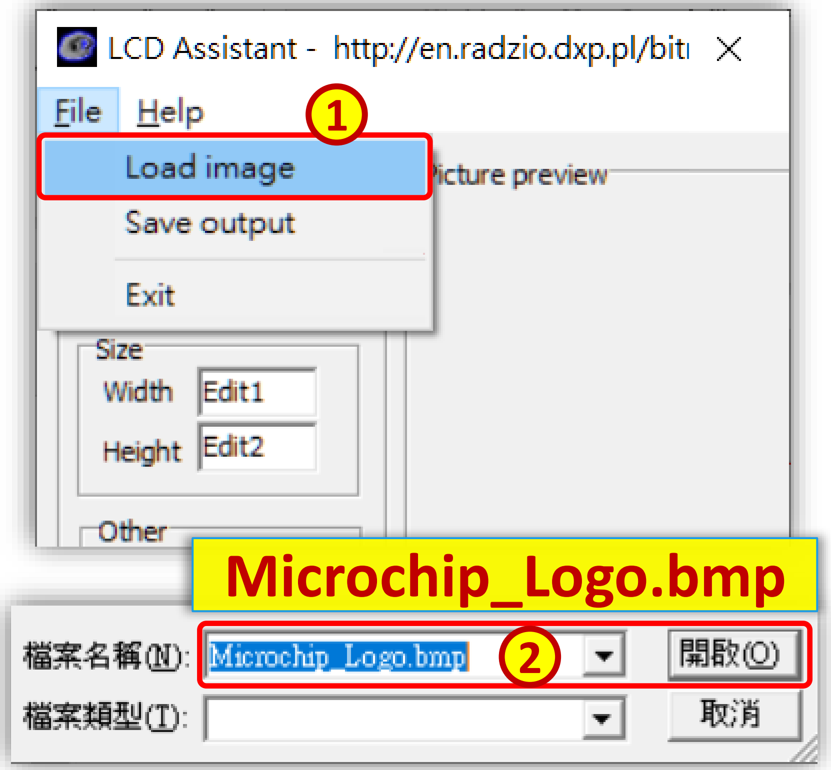


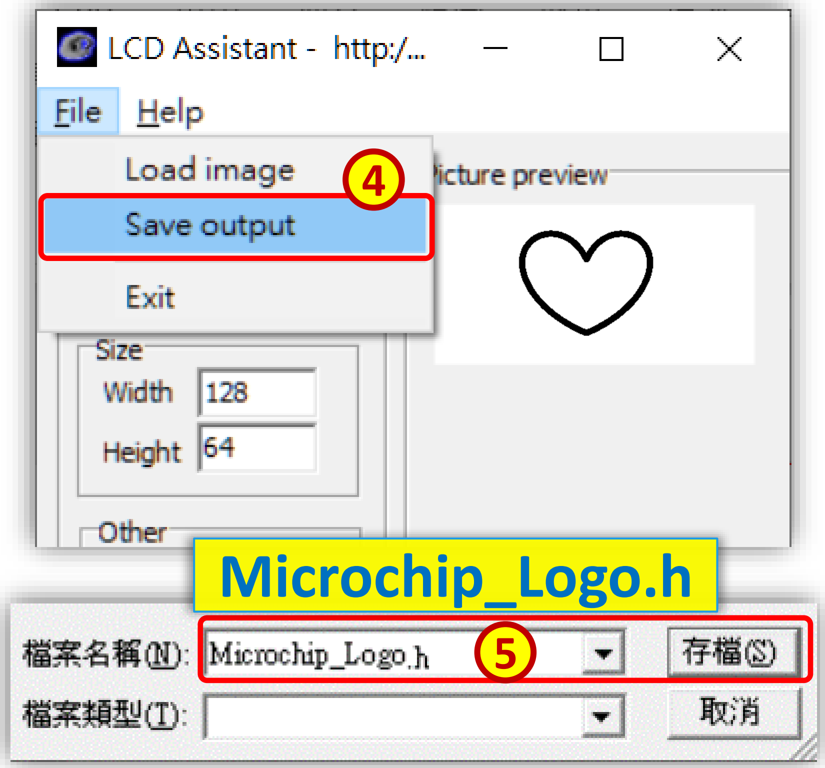
MICROCHIP

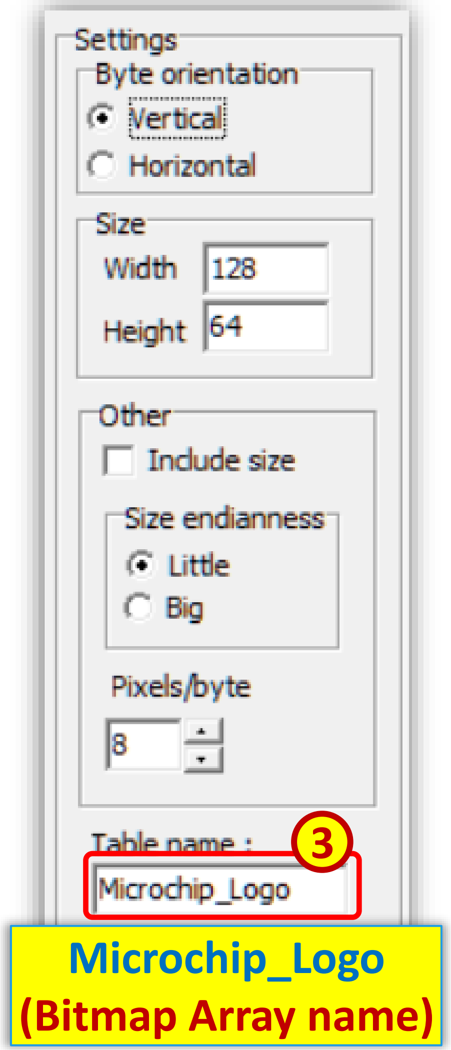
Lab17.1 SERCOM - SPI SSD1306 OLED

Create your own logo

- Execute \Tools\LCDAssistant.exe
  Load Microchip_Logo.bmp and Save output to Microchip_Logo.h







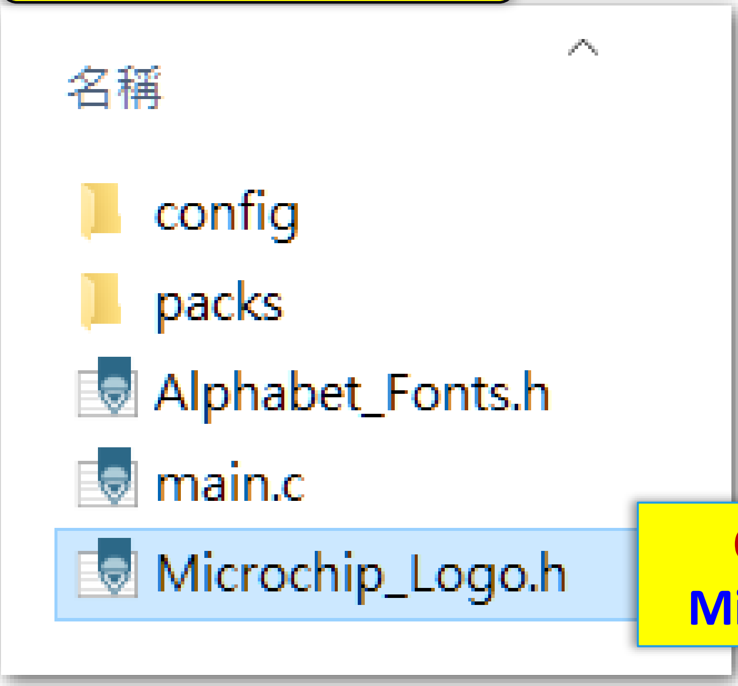
MICROCHIP

Lab17.1 SERCOM - SPI SSD1306 OLED

Create your own logo

- Copy new Microchip_Logo.h to overwrite old one in folder your project's \src

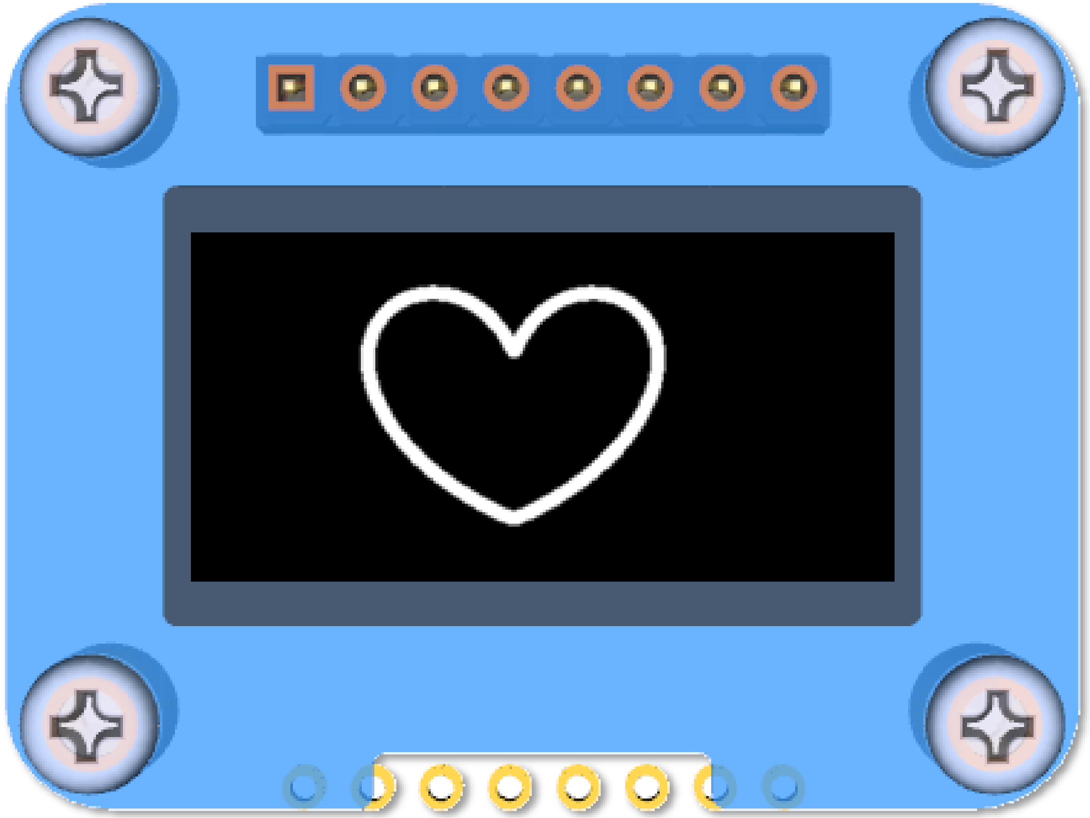
Your Project's \src



Overwrite old Microchip_Logo.h

Lab17.1 SERCOM - SPI SSD1306 OLED

Create your own logo result



Microchip Trademarks

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

Trademark	Description/Appropriate Nouns	Notes
ClockWorks®	configurator, device, family, product, series	US only
CodeGuard™	security	
CryptoAuthentication™	development library, device, family, IC, library	
CryptoAutomotive™	development kit, security IC, technology	
CryptoCompanion™	chip	
CryptoController™	solution, TPM, trusted platform module	
CryptoMemory®	cryptographic security IC, IC	
CryptoRF®	device, transponder	
dsPIC®	DSC, digital signal controller	
dsPIC®	dsPIC® logo	
dsPICDEM™	demonstration board, development board	
dsPICDEM.net™	board	
Dynamic Averaged Matching™ (DAM™)	architecture	
ECAN™	solution, technology	
Espresso T1S™	device	
EtherGREEN™	family, platform, technology	
EtherSynch®	family, platform, technology	US only
EyeOpen™	cable compensation	
Flashtec®	controller, family, product	US only
flexPWR®	technology	
Frequency on Demand®		Registered in other countries (not US)
GestIC®	controller, sensing, technology	
GridTime™	device, GNSS time server, time server	
HELDO®	family, regulator	
Hyper Speed Control®	architecture, family	US only
HyperLight Load®	mode	US only
In-Circuit Serial Programming™ (ICSP™)	programming capability	
IdealBridge™	dual MOSFET-bridge rectifier, rectifier	
IGaT™	board, demonstration board	
IGLOO®	device, family, FPGA, series	
INICnet™	sniffer, technology	

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

Trademark	Description/Appropriate Nouns	Notes
IntDrive™	driver board, integrated solution, power module	
Intelligent Paralleling™	technology	
IntelliMOS®	family, power stage family	US only
Inter-Chip Connectivity™	technology	
JitterBlocker™	attenuator, family	
JukeBlox®	platform, technology	
jukeblox®	JukeBlox® registered word mark with a design	Image is not registered; this is a graphic treatment with the registered word mark
KeeLoq®	protocol, security IC, technology	
Kleer®	technology	
KLEER®	Kleer® registered word mark with a design	Image is not registered; this is a graphic treatment with the word mark
Knob-on-Display™ (KoD™)	device, family, HMI, knob designer, position wizard, solution, technology, touch controller, user interface, widget configurator	
LANCheck®	online design review, online review, review	
Libero®	design file, designer, IDE, SoC design suite, software, tool	US only
LinkMD®	cable diagnostics, diagnostics engine, family, technology	
MarginLink™	signal integrity testing	
maxCrypto™	controller-based encryption, encryption	
maXStylus®	solution	
maXTouch®	family, studio, technology, touchscreen controller	
maxView™	driver, storage manager, tool	
MediaLB®	bus, controller, device	
megaAVR®	device, MCU, microcontroller	
memBrain™	neuromorphic memory product, product, solution, technology	
	Microchip logo	Please do not use this logo; always use the Microchip name and logo.

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

Trademark	Description/Appropriate Nouns	Notes
MICROCHIP	Microchip name and logo	This is the vertical representation; the horizontal representation is also available.
Microsemi	Microsemi logo	Do not use
Microsemi®		Do not use
Mindi™	analog simulator, analog simulator tool	
MiWi™	application, coordinator, mesh, network, protocol, software, stack, wireless networking protocol	
MOST®	board, cooperation, design, device, evaluation kit, NetServices, network, platform, product, specification, system, technology	
MOST®	MOST® logo	
motorBench®	development suite	US only
MPASM™	assembler	
MPF™	device, product	
MPLAB®	**Nouns:** C compiler, compiler, development ecosystem, ICD, ICE, IDE, IPE, in-circuit debugger, in-circuit emulator, integrated development environment, integrated programming environment, library, programmer, server, starter kit, universal device programmer, XC compiler **Tool Names:** AI Edge, Analog Designer, Analysis Tool Suite, Code Configurator, Connect Configurator, Data Visualizer, Discover, Extensions, Harmony, Mindi™, Network Creator, PICkit™, Programmer-to-Go, Programmer-to-Go Mobile App, PTG, SiC Power Simulator, Snap, Xpress	
MPLAB® CERTIFIED	MPLAB® Certified logo	
MPLIB™	librarian, object librarian	

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

Trademark	Description/Appropriate Nouns	Notes
MPLINK™	linker, object linker	
mSiC™	bare die, diode, gate driver, module, MOSFET, product, solution, technology	
mTouch®	development kit, evaluation kit, library, sensing solution, solution, technology	US only
MultiTRAK™	technology	
NetDetach™	technology	
Omniscient Code Generation™ (OCG)	compilation technology	
OptoLyzer®	network analysis tool, studio, tool	
PIC®	assembler, device, MCU, microcontroller	
PIC32®	PIC32 logo	
PIC64™	PIC64 logo	
PICDEM™	demonstration board	
PICDEM.net™	Internet/Ethernet demonstration board	
PICkit™	debugger, development tool, in-circuit debugger, on-board, programmer, starter kit, tool	
picoPower®	technology	
PICSTART®	development programmer	
PICtail™	board, daughter board	
PolarFire®	device, family, FPGA, kit, SoC, System-on-Chip	
PowerSmart™	designer, development suite, digital control library	
Precision Edge®	family, product family	US only
ProASIC®	architecture, board, family, FPGA, kit	US only
ProASIC Plus®	architecture, device, family, FPGA, switch	US only
Prochip Designer®	software suite	
PureSilicon™	oscillator, technology	
QMatrix™	device, sensor	
QTouch®	board, composer, configurator, device, development platform, library, project builder, sensor, solution, technology, tool	
Quiet-Wire®	enhanced EMI technology, family, technology	

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

Trademark	Description/Appropriate Nouns	Notes
Ripple Blocker™	attenuator, family of linear regulators, product line, technology	
REAL ICE™	in-circuit emulator	
RTAX™	FPGA	
RTG4™	FPGA	
SAM-BA®	applet, application, host, in-system programmer, ISP, monitor, software	
SAM-ICE™	emulator	
SenGenuity®	product, sensor	
Serial Quad I/O™ (SQI™)	family, Flash device, product	
Silicon Storage Technology®		Japan only
simpleMAP™	protocol	
SimpliPHY™	device, family, product	
SmartBuffer™	device, IC	
SmartFusion®	evaluation kit, FPGA, SoC, System-on-Chip	US only
SmartHLS™	compiler, compiler software, IDE, software	
SMART-I.S.™	technology	
SpyNIC®	device	
SST®	Acronym for Silicon Storage Technology	
SST® A Microchip Technology Company	SST logo	
STK®	starter kit	
storClad™	encryption technology, technology	
SuperFlash®	device, kit, macro, memory, technology	
SuperSwitcher™	buck regulator, family, regulator	
SuperSwitcher II™	family	
Switchtec™	family, product line, switch, technology	
Symmcom®		China only
Symmetricom®		
SynchroPHY™	device, family, product	
SyncServer®	clock, device, instrument, oscillator, server	

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

Trademark	Description/Appropriate Nouns	Notes
SyncWorld®	ecosystem program, timing	US only
Tachyon®	controller, family, platform, product, technology	
The Embedded Control Solutions Company®		US only; should be in italics
TimeCesium®	primary frequency reference, product	US only
TimeHub®	platform, system	US only
TimePictra®	platform, software suite, synchronization management system	US only
TimeProvider®	clock, enhanced PRTC, ePRTC, extension, system, grandmaster, kit, module, product, series, server, shelf	US only
TimeSource®	PRC, PRS, primary reference clock, primary reference source, shelf, system	
tinyAVR®	MCU, microcontroller	
Total Endurance™	software model	
TSHARC™	controller, touchscreen controller	
Turing™	hardware security module, programmer	
UNI/O®	bus, communication protocol, firmware, hardware port, memory chip	
USBCheck™	design review	
VariSense™	technology	
VectorBlox™	accelerator, intellectual property, IP, platform, SDK, software development kit, solution, tool, tool chain, tool flow	
Vectron®	oscillators, products	
VelocityDRIVE™	CT, configuration tool, product, solution, software platform, SP	
VeriPHY™	cable diagnostics algorithm, cable diagnostics software suite	
ViewSpan™	technology	
WiperLock™	technology	
XMEGA®	Custom Logic (XCL), device, family, MCU, microcontroller, series	
XpressConnect™	retimer	
ZENA™	analyzer, software, wireless adapter	
ZL®		US only

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

Trademark	Description/Appropriate Nouns	Notes
	Adaptec "a" logo	Registered in other countries (not US)
Adaptec®	adapter, cable, controller, driver, family, product, series, software, solution	
Adjacent Key Suppression™	technology	
AgileSwitch®	device, digital programmable gate driver, driver, family, gate driver	US only
AKS™	technology	
Analog-for-the-Digital Age™	AIPD family tagline	
Any Capacitor™	stable	
AnyIn™	capability, feature, structure	
AnyOut™	capability, feature	
Augmented Switching™	technology	
AVR®	architecture, CPU, device, family, MCU, microcontroller	
AVR®	AVR® logo	
AVR Freaks®	forum	
BesTime®	technology	
BitCloud®	SDK, software development kit	
BlueSky®	firewall, subscription service, technology	
BodyCom™	development kit, system, technology	
chipKIT®	I/O shield, board, hardware, network, platform, software	
chipKIT®	chipKIT logo	
Clockstudio™	application, program, software, utility	

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

MICROCHIP