10TH ANNIVERSARY MASTERS 25
EMPOWERING INNOVATION
十年精英聚首，共創技術未來

Getting Started With PIC32C and SAM Arm® Cortex®-M Microcontrollers

Presented By: Calvin Ho

25004-MCU1

MICROCHIP

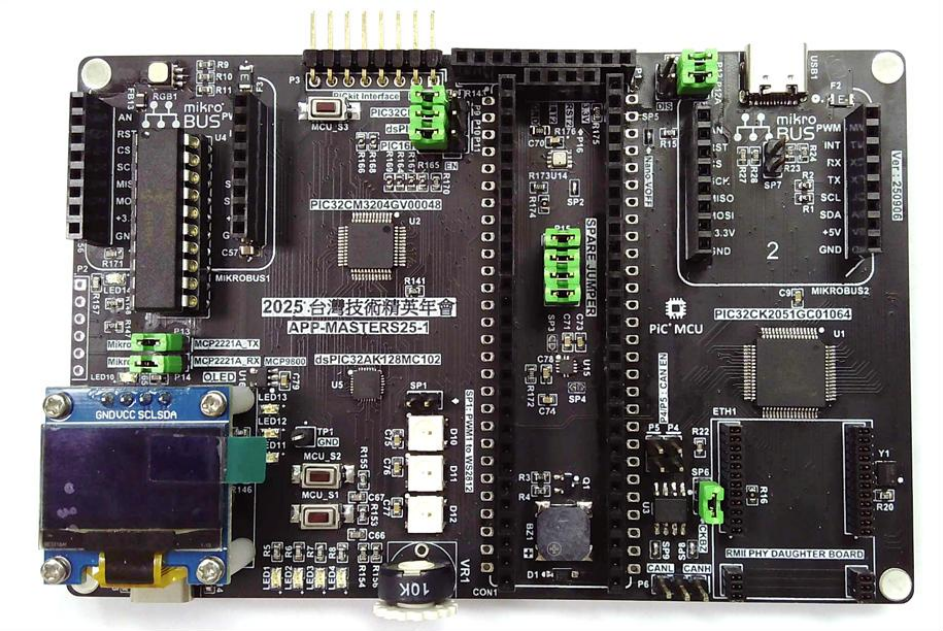
Class Objectives

Learn basic concepts of **Arm® Cortex®-M0+**, M23, M33, and M4 architectures such as **bus matrix** and **nested vectored interrupts**

Learn system architecture of Arm Cortex-M based **PIC32C** and SAM microcontrollers such as **clock system**, **GPIO** and **bus synchronization**

Learn how to use **MPLAB® X IDE** and **MCC Harmony** to easily generate code, understand **MPLAB XC32 compiler** register accesses, and program and debug code using APP-MASTERS25-1 evaluation board

Development Tools Used In This Class



Hardware

- APP-MASTERS25-1
  台灣技術精英年會主題實驗板
- MPLAB® SNAP 燒錄/除錯器

Software

- MPLAB® X IDE v6.25
- MPLAB XC32 Compiler v4.60
- MPLAB® Code Configurator (MCC) Harmony v5.6

MPLAB® X IDE

MICROCHIP

Microchip 32-bit MCU Portfolio

MCU1 課程重點 – MCUs With Arm® Cortex® Cores

32-bit MCU Cores

MCUs With Arm® Cortex® Cores

PIC32C and SAM MCUs

PIC32C devices deliver advanced performance, low power consumption and a rich development ecosystem, making them well suited for scalable, real-time embedded applications across a wide range of industries.

MCUs With MIPS32® Core

PIC32M MCUs

PIC32M devices are powered by MIPS32 cores, which provide a high-performance, power-efficient processing architecture that enables fast instruction execution. These MCUs target embedded applications requiring real-time control and multitasking.

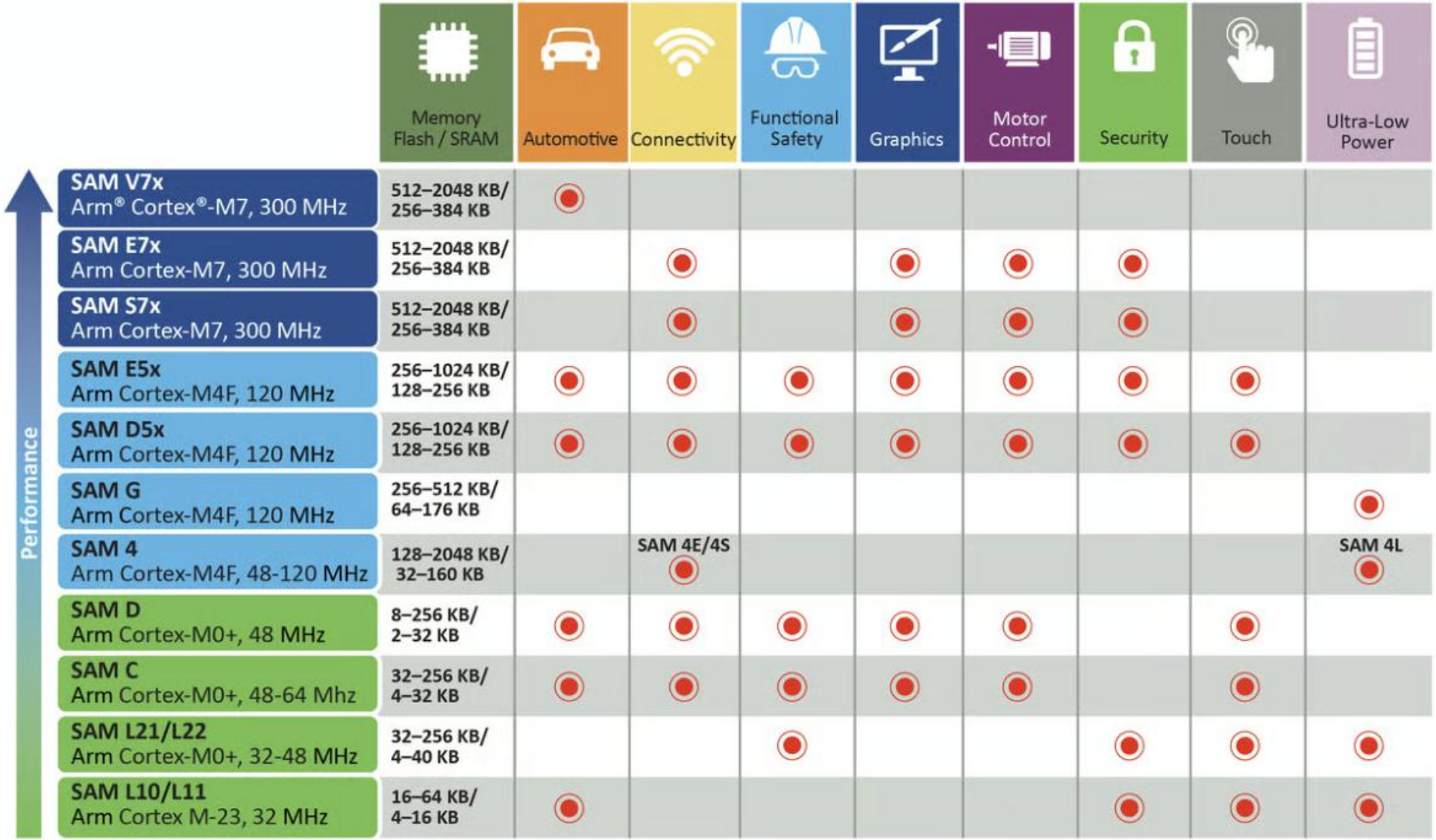
MCUs With a 32-bit CPU

PIC32A MCUs

PIC32A devices are powered by a 200 MHz 32-bit CPU that offers a Double-Precision Floating Point Unit (DP-FPU) and high-speed analog providing efficient processing, low-power operation and robust peripheral integration to offer a versatile solution for high-performance embedded applications.

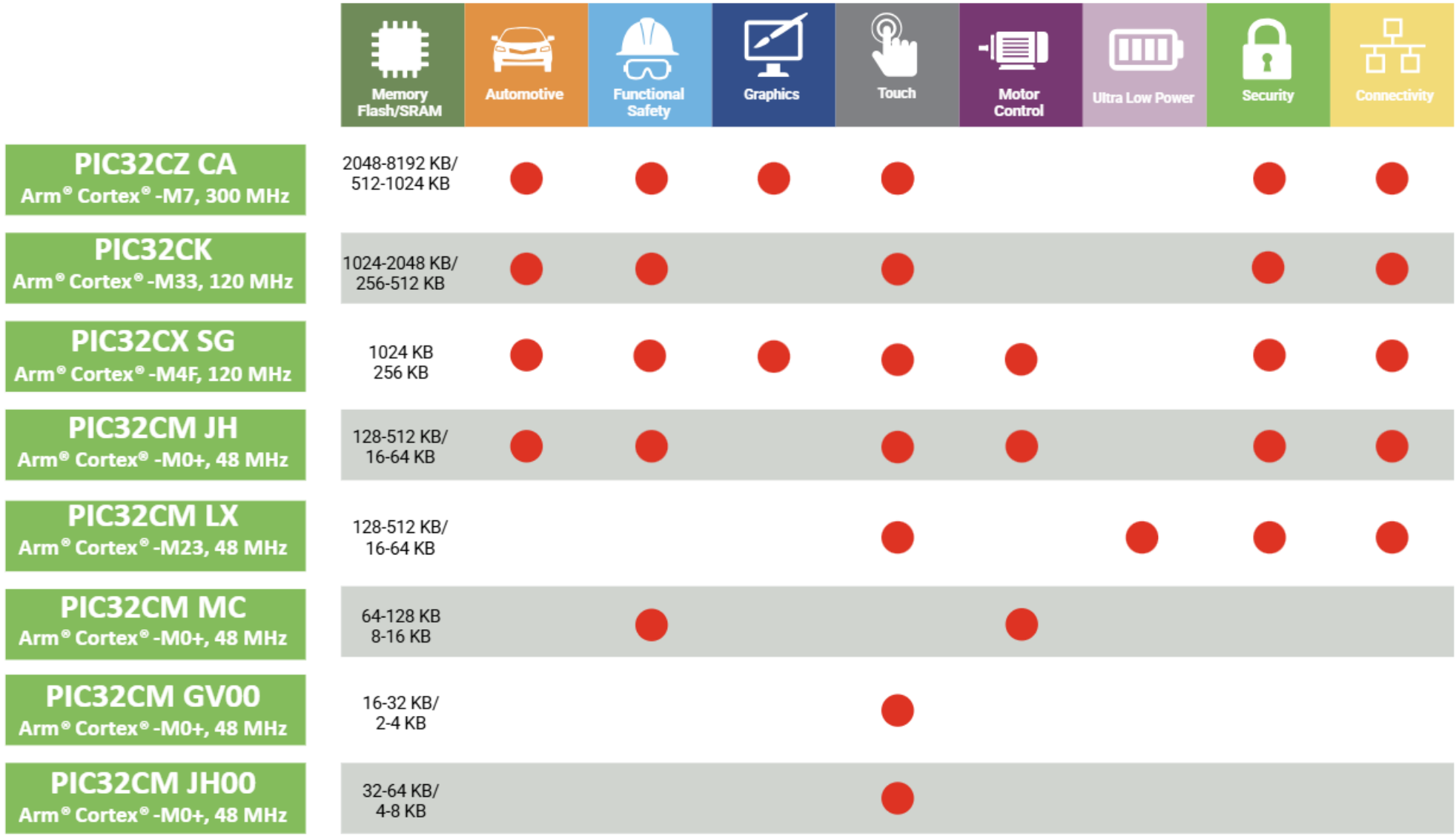
MICROCHIP

Legacy 32-bit MCUs With Arm® Cortex® Cores



MICROCHIP

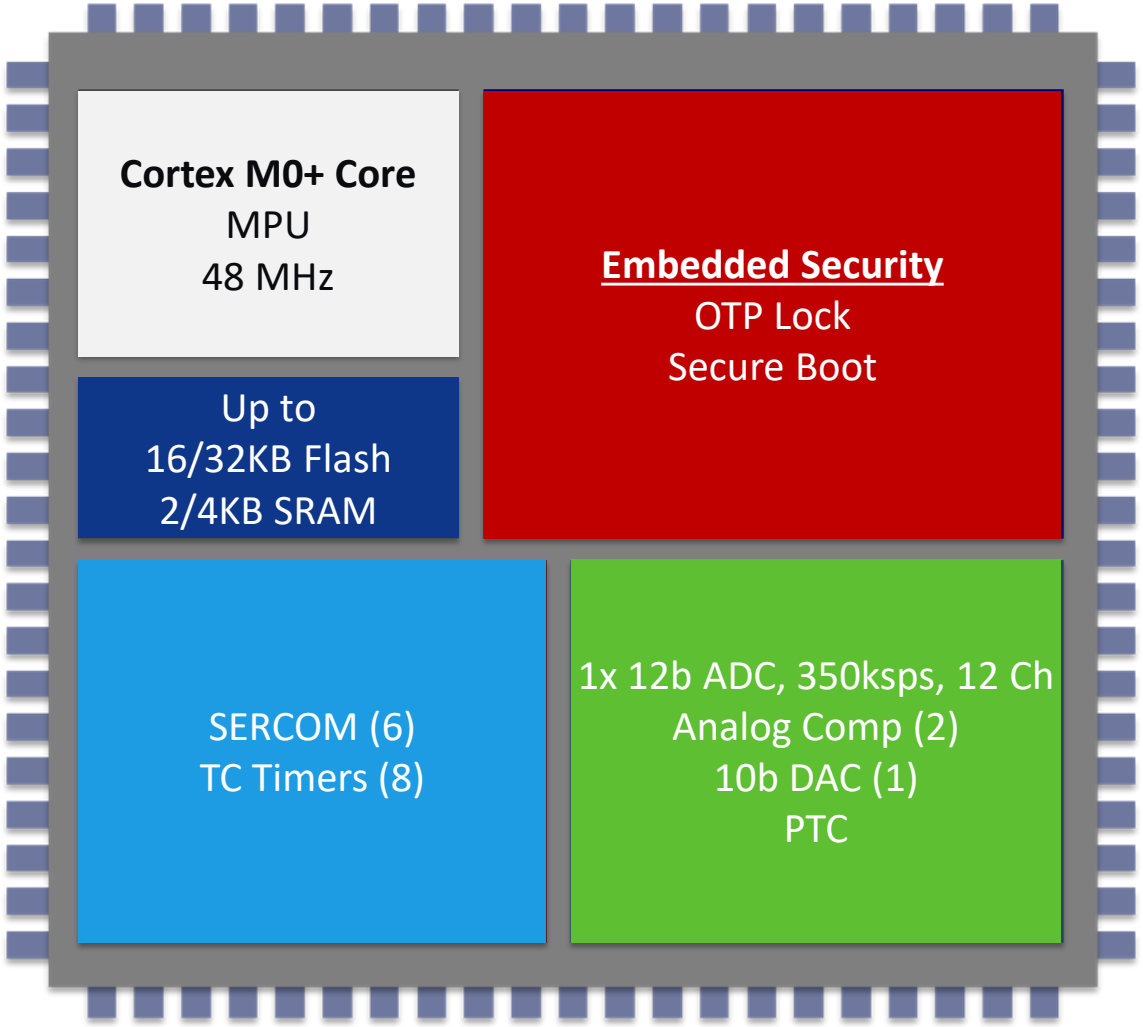
New 32-bit MCUs With Arm® Cortex® Cores



MICROCHIP

PIC32CM GV – Value Line Cortex-M0+

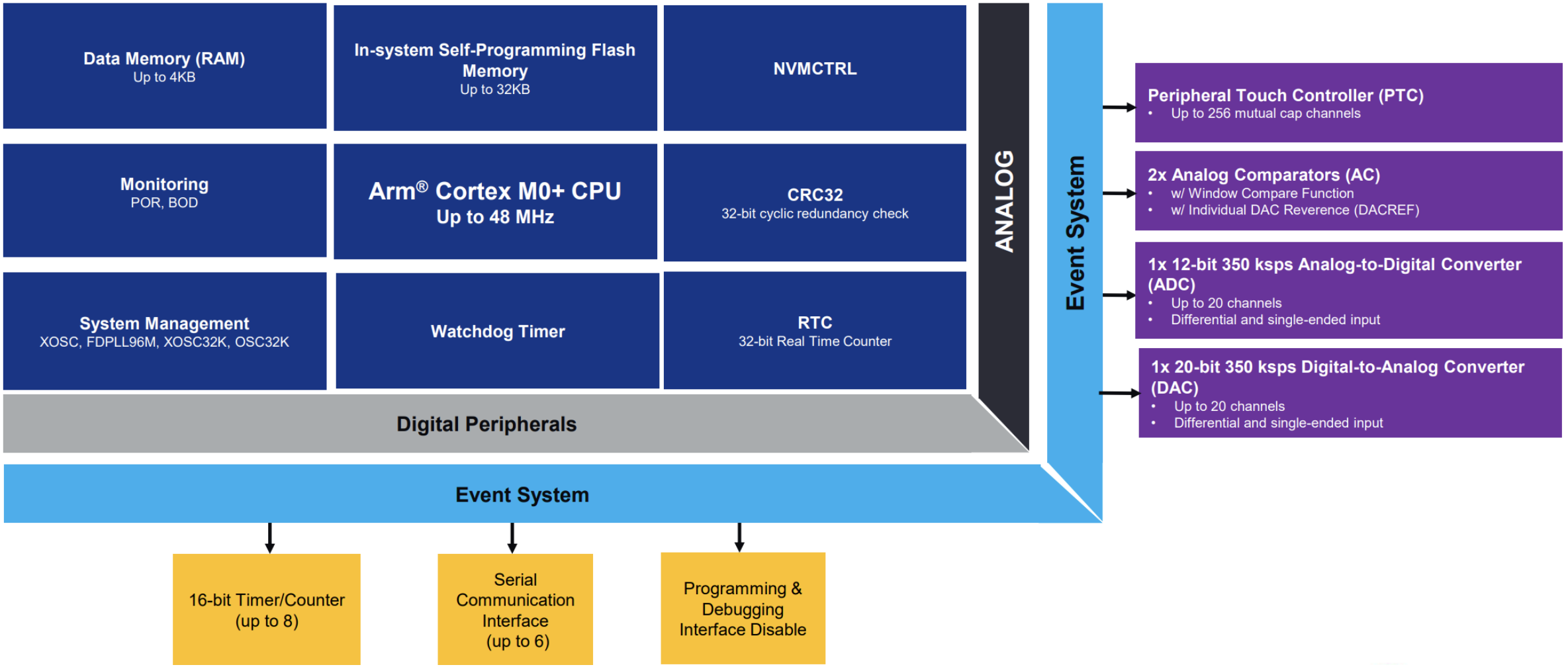
Low-Cost Pin compatible SAMD2x



- **New 3.3V Value Line Series:**
  - Pin and binary compatible with SAMD20
  - Up to 50% savings compared to SAMD20
  - New Curiosity Nano Board
  - ASIL-B Compliant Hardware Design
- **Key Features:**
  - Single, multi-channel 12B ADC
  - Secure Boot function via OTP Lock
  - One Time Programmable Option
  - Peripheral Touch Controller
- **Target Markets:**
  - Automotive, White Goods, and Industrial Markets

MICROCHIP

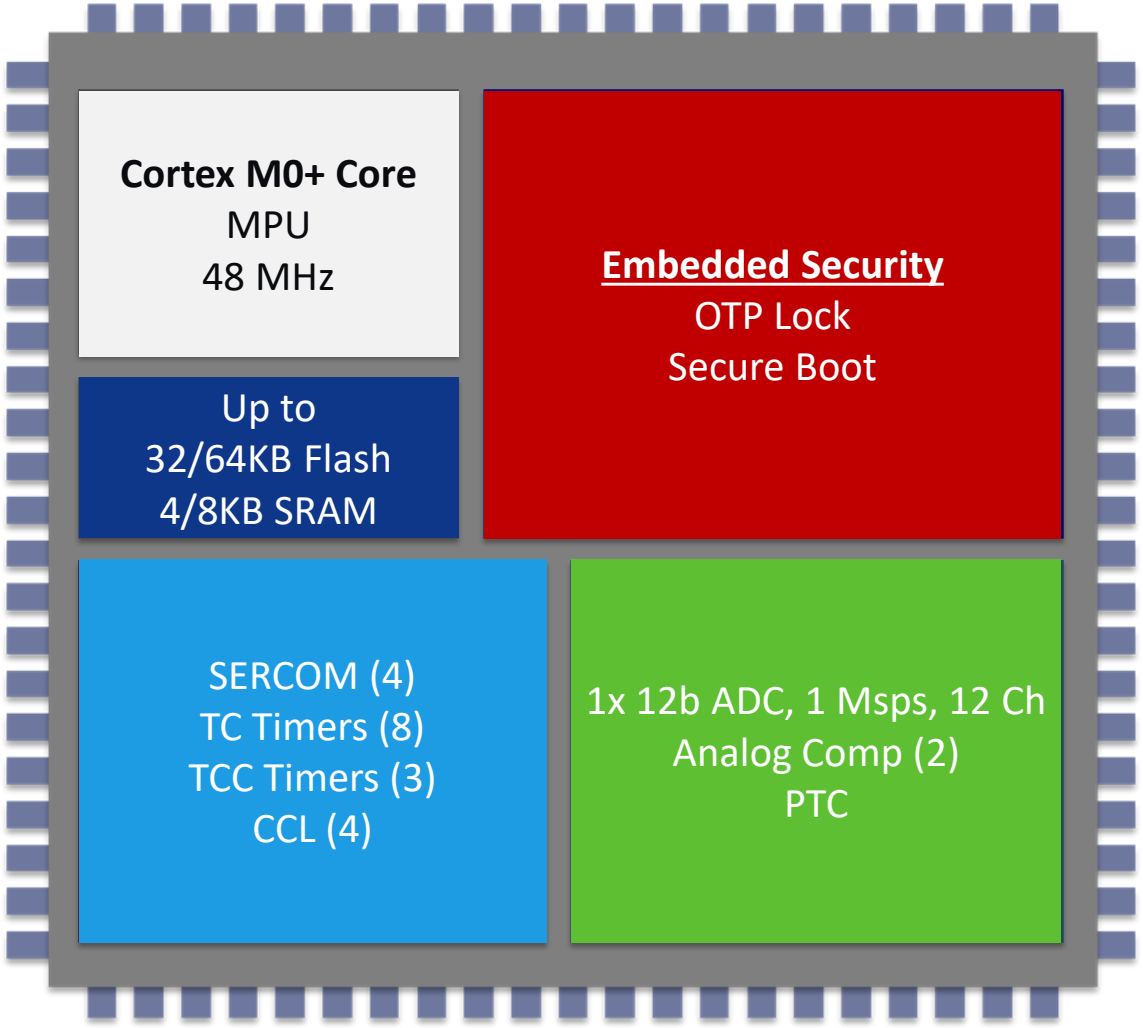
PIC32CM GV00 Block Diagram



MICROCHIP

PIC32CM JH00 5V Cortex-M0+ Family

New Low Cost 64KB flash – Pin compatible SAMC20

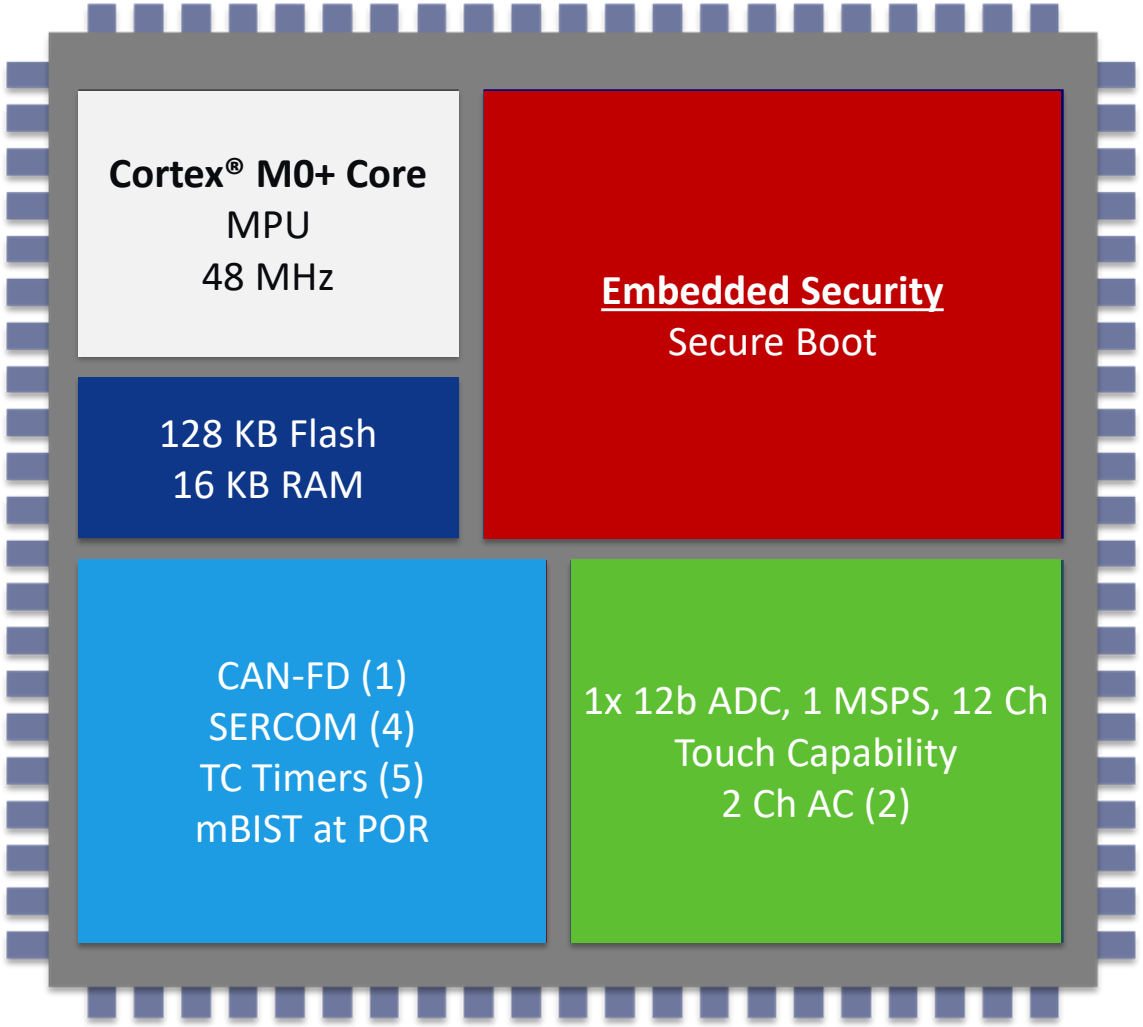


- **New 5V Low-Cost Additions!**
  - 32-64KB flash version pin and binary compatible with SAM C20 and ASIL-B Compliant Hardware Design
  - Up to 60% savings compared to SAM C2x
  - New Curiosity Nano Board
- **Key Features:**
  - True 5V
  - Single, multi-channel 12B ADC
  - Secure Boot function via OTP Lock
  - Peripheral Touch Controller up to 256 channels
- **Target Markets:**
  - Automotive, White Goods, and Industrial Markets
- **Launch: CQ2-25**

MICROCHIP

PIC32CM JH01 5V Cortex-M0+ Family

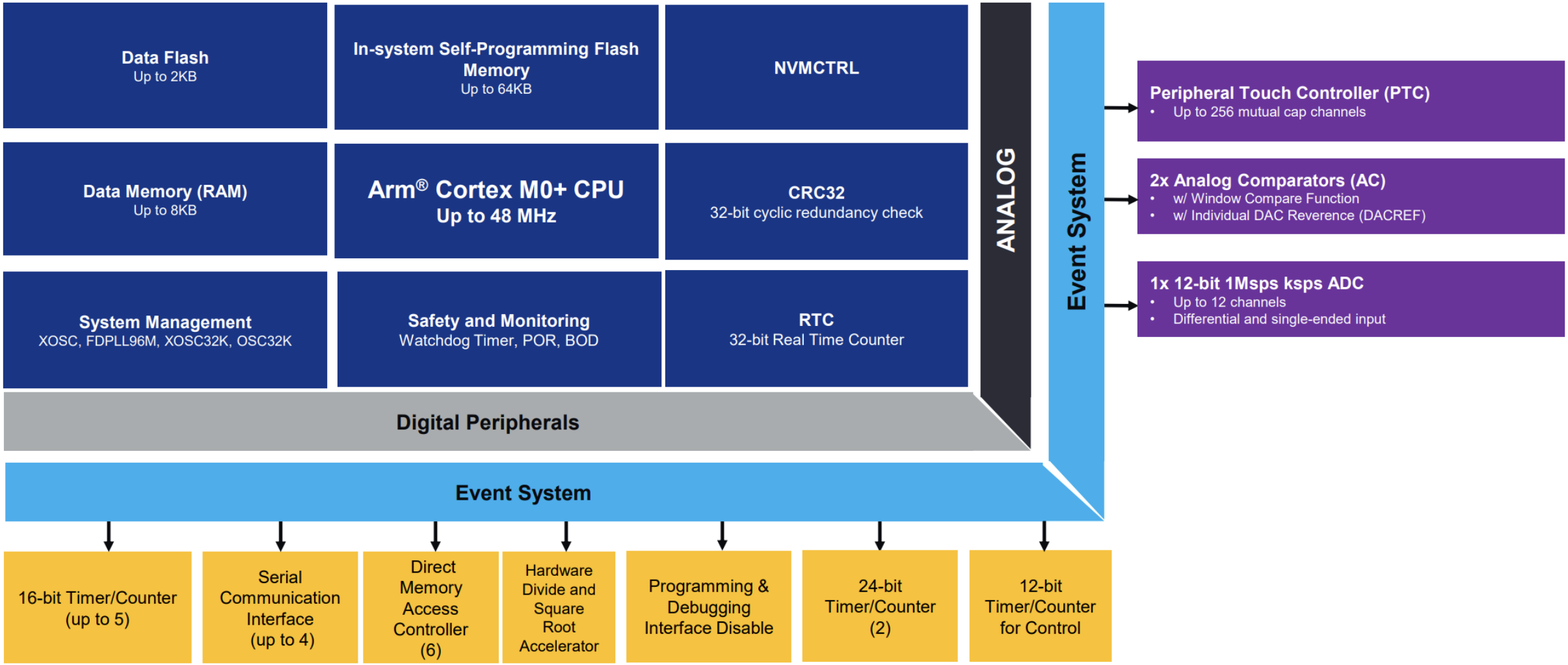
New 128KB Family Member Release – Compare to SAMC2x



- **New Family Additions:**
  - Cost reduced version of strong PIC32CM JH Family
  - Adds 32 & 48-pin, 128KB flash with single CAN-FD
- **Key Features:**
  - True 5V with CAN-FD
  - mBIST at Power-on Reset
  - Secure Boot
  - Touch capability
- **Target Markets:**
  - Automotive, White Goods, and Industrial Markets
- **Expands:**
  - 512KB & 256KB Family Launched Fall 2022
- **Launch: CQ2-25**

MICROCHIP

PIC32CM JH00 Block Diagram



MICROCHIP

Programming and Debug Interface Disable (PDID)

New regulations require prevention of physical attacks to repurpose electronics for malicious activities by hooking on to exposed programming/debugging pins.

- Additional security feature where user can choose to disable all future device erase and programming through normal program/debugger interface
- This feature, together with lock-bits (restricted read access), means device will not be reprogrammable or reconfigurable from debugger/programmer
- All further updates to the flash must be done through a (customer's) **immutable bootloader**
- ***No failure analysis on NVM possible after this feature is enabled***
- **Customers will need to contact Microchip Sales to use/learn more about this feature!**

Programming / debugger access on different security levels

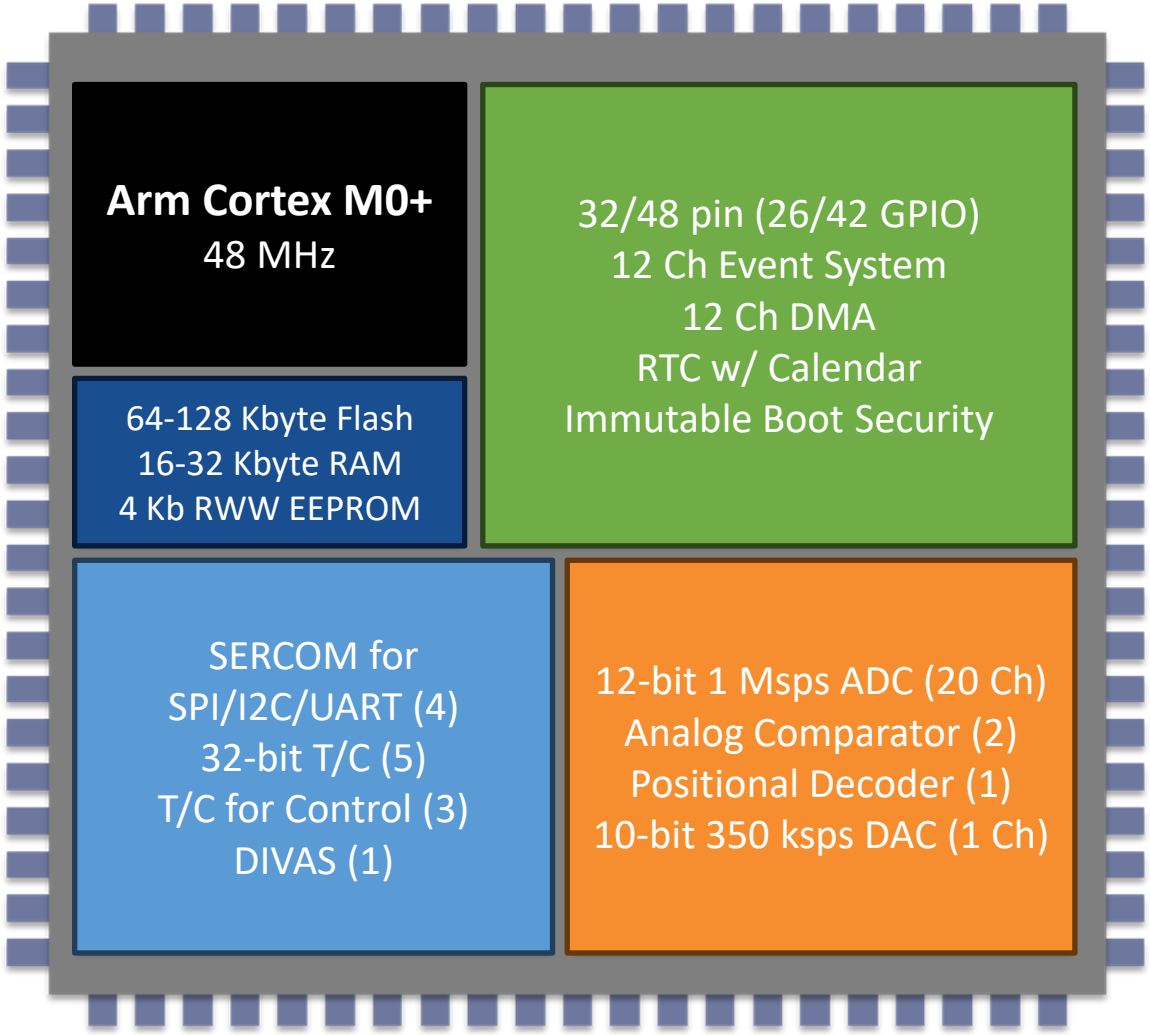
Security Feature	Flash read	Flash write	Identify device	FA
No security	✓	✓	✓	Normal
Lock bits set / CP/CPD	⛔🔒*	🔒*	✓	Normal
PDID enabled	⛔	⛔	✓	**Limited**

*unlock only after a full device erase
CP = Code protection
CPD = Code Protection of Data EEPROM

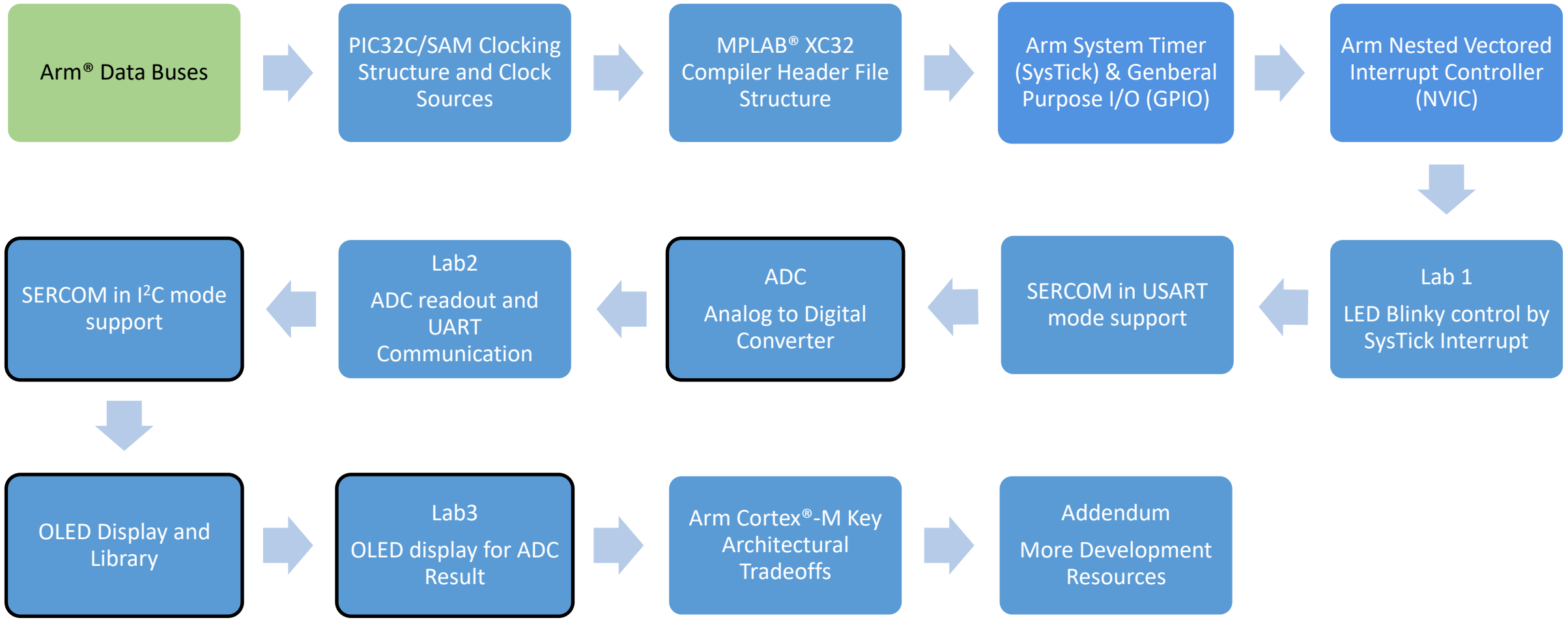
MICROCHIP

PIC32CM MC00 Family Features Overview

- **Arm® Cortex®-M0+ Core**
- **Nested Vectored Interrupt Controller**
  - 32 interrupts and one NMI
- **Single cycle 32-bit multiplier**
- **Single cycle I/O port**
- **Secure Features**
  - Unprivileged/Privileged support
  - 8-region MPU
- **32-bit Instruction Fetch Width**
  - Up to two 16-bit Thumb instruction fetches concurrently



Class Outline



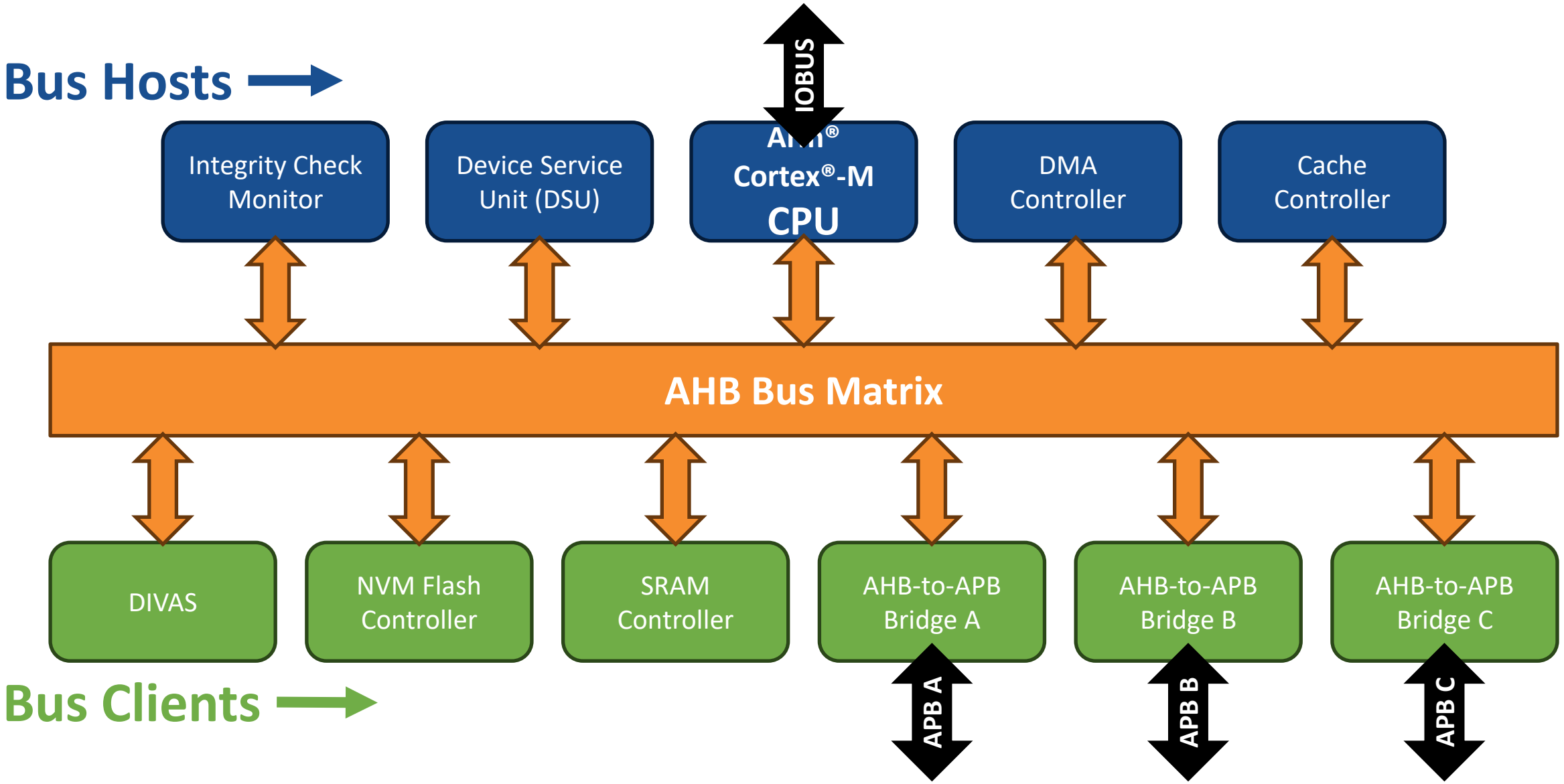
Arm® Data Buses

Advanced High-Performance Bus (AHB)

Advanced Peripheral Bus (APB)

Single Cycle I/O Bus (IOBUS)

Advanced High-Performance Bus (AHB)

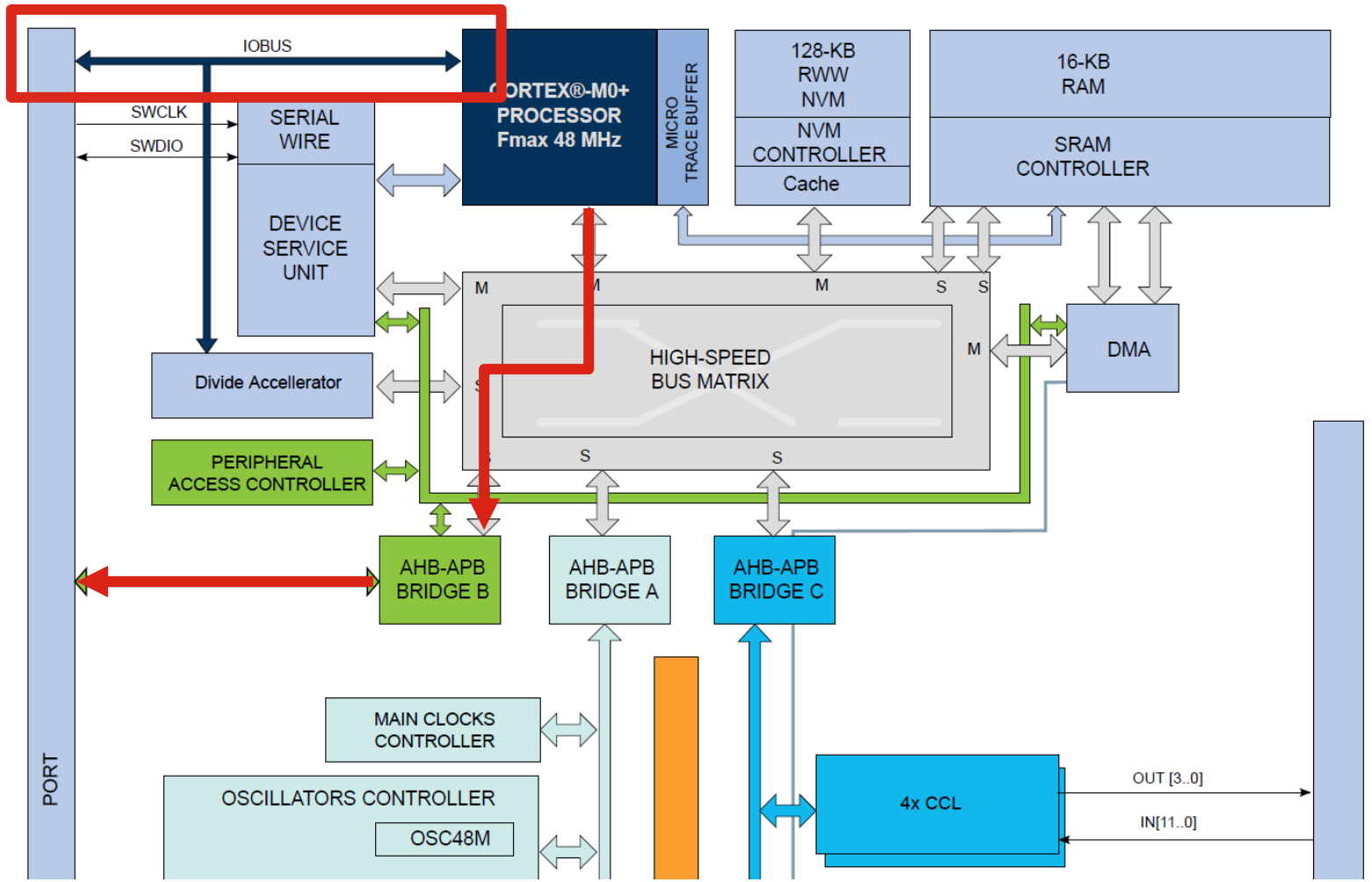


Advanced Peripheral Bus (APB)

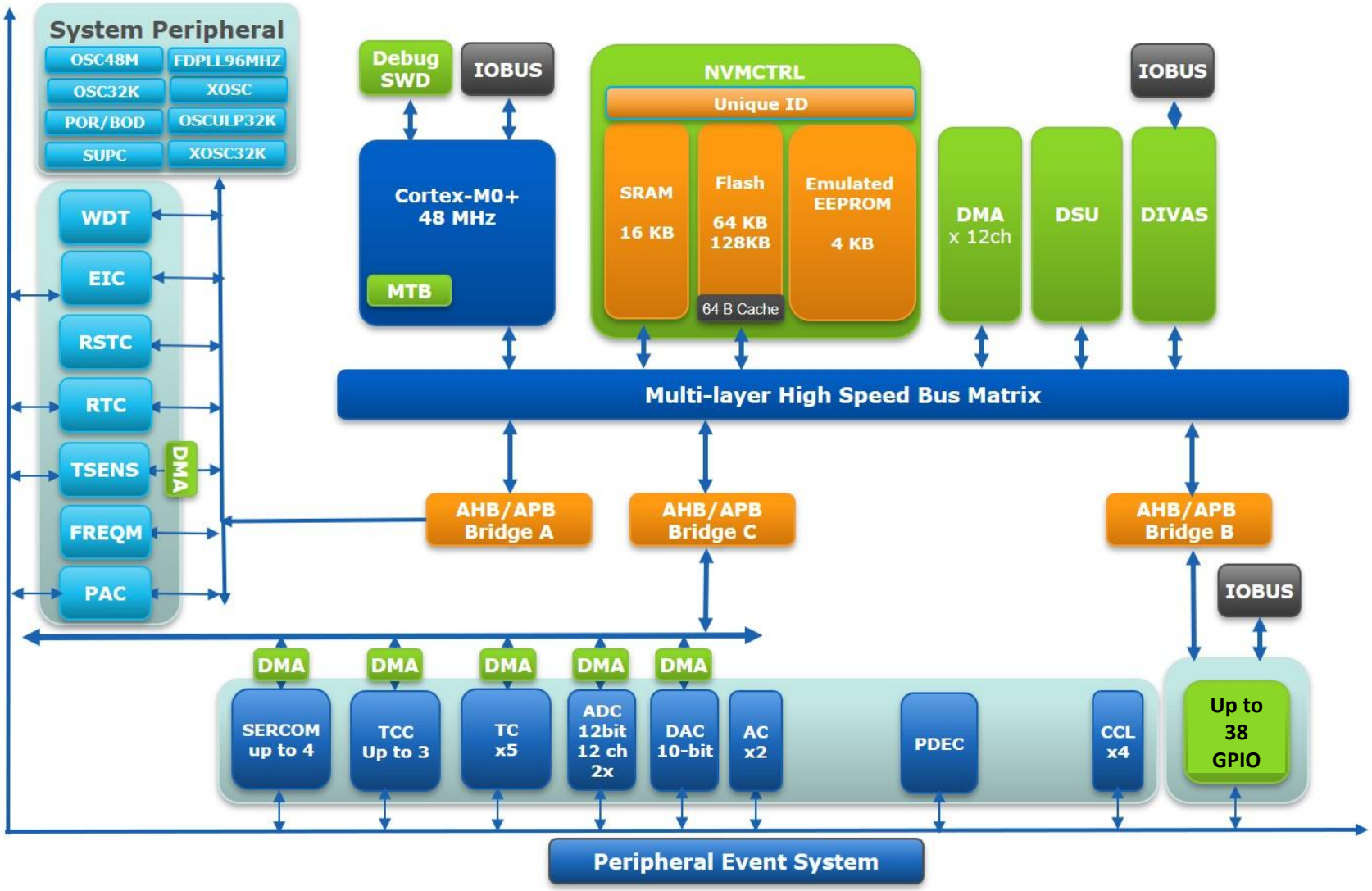
Low Power	Can run without intervention from AHB
Low Complexity Data Transfers	Non-pipelined transfers
Client to AHB Bus	AHB Bus controls data transfers to control registers
Flexible Asynchronous Clock	Can run at different speed than AHB
Bridge Connections	Connects via bridges to AHB

Single Cycle I/O Bus (IOBUS)

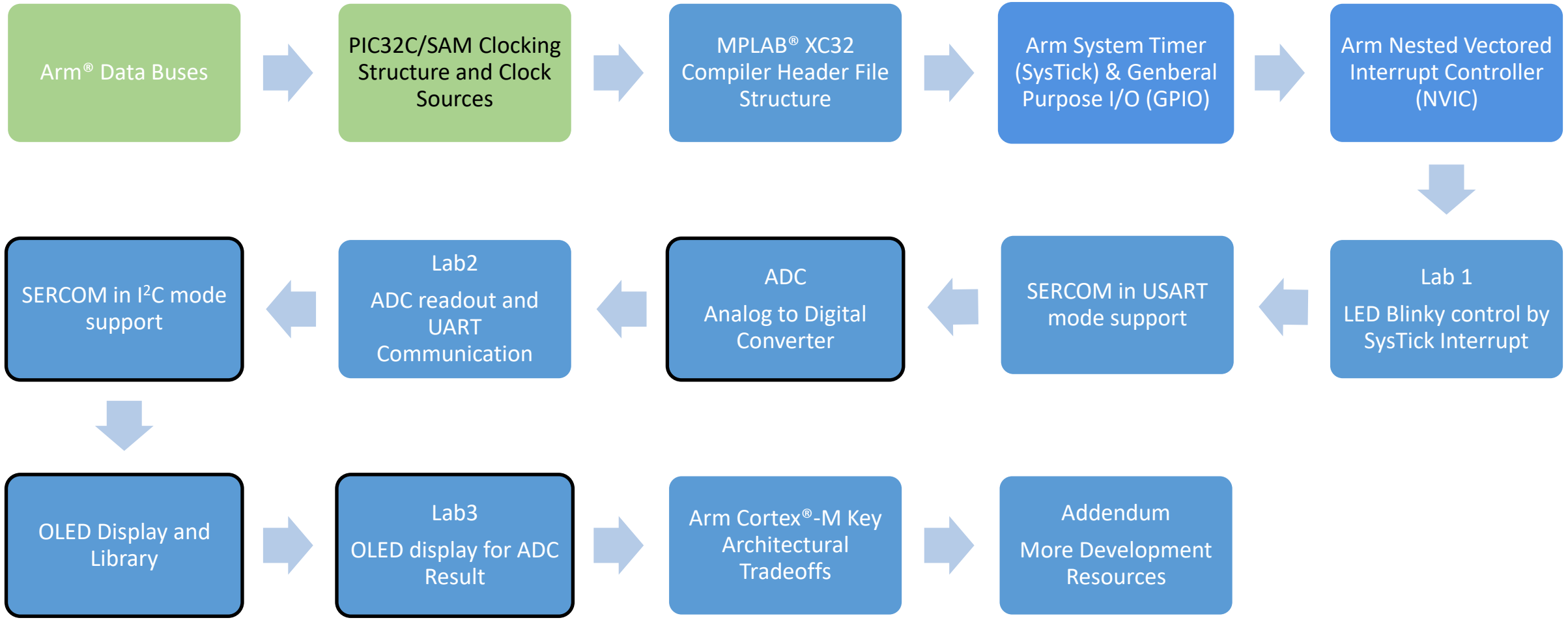
- Available on Arm® Cortex®-M0+ and Cortex-M23
- Separate bus interface
- GPIO access without bus transfer
- Memory mapped to a separate region
- No resynchronization required



Bus Layout in PIC32CM MC00 Family

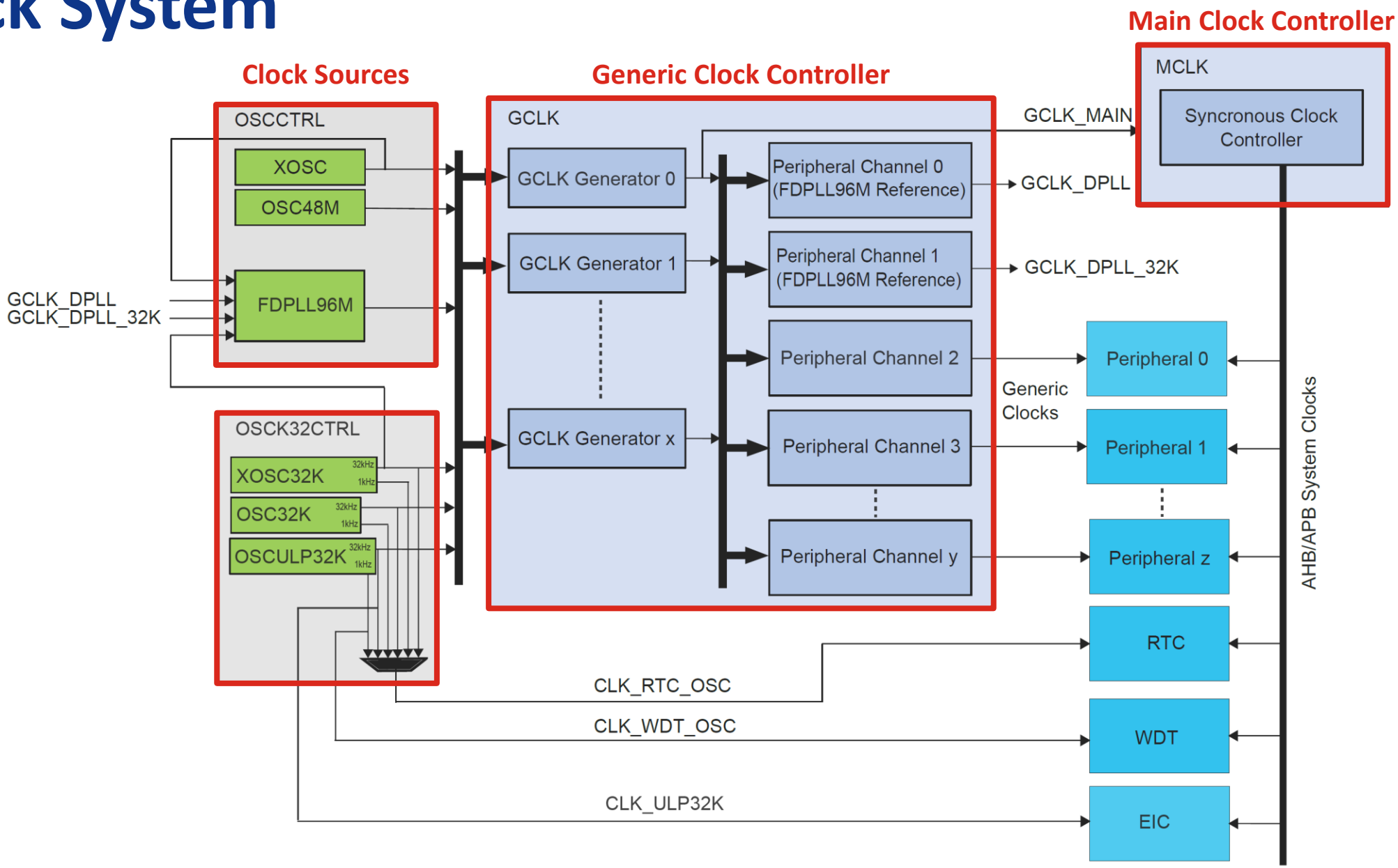


Class Outline



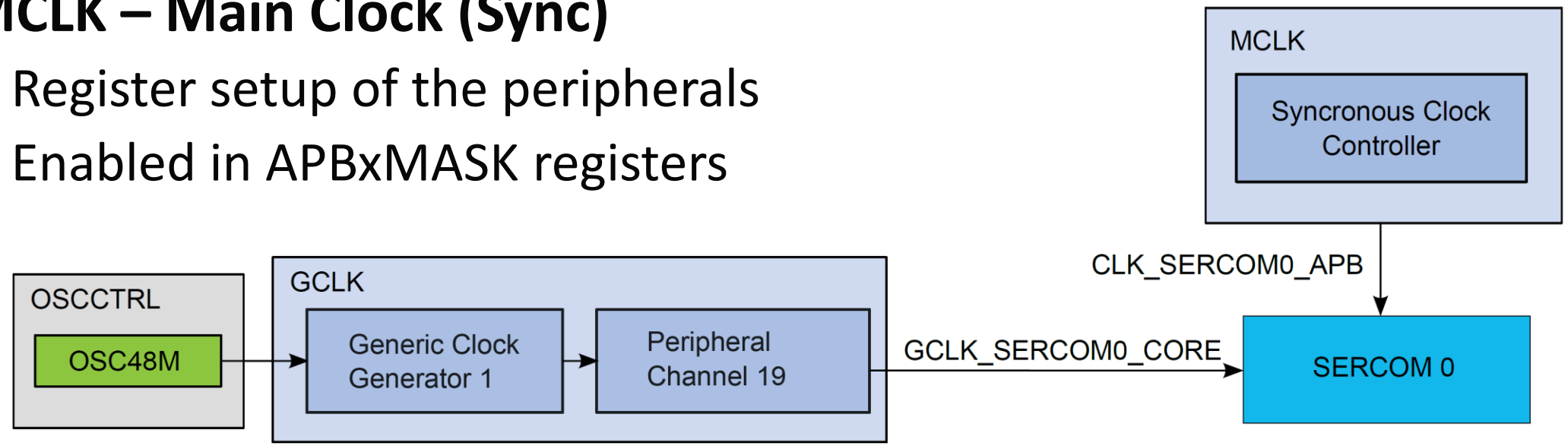
PIC32C and SAM Clock Structure

Clock System

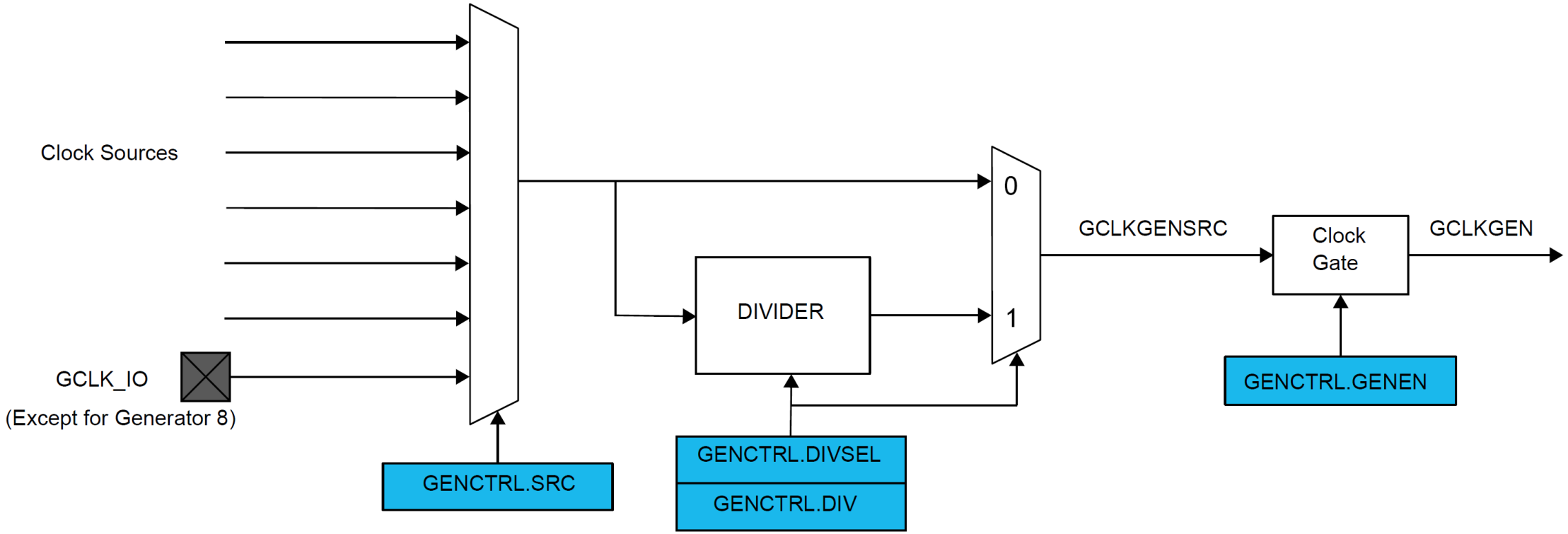


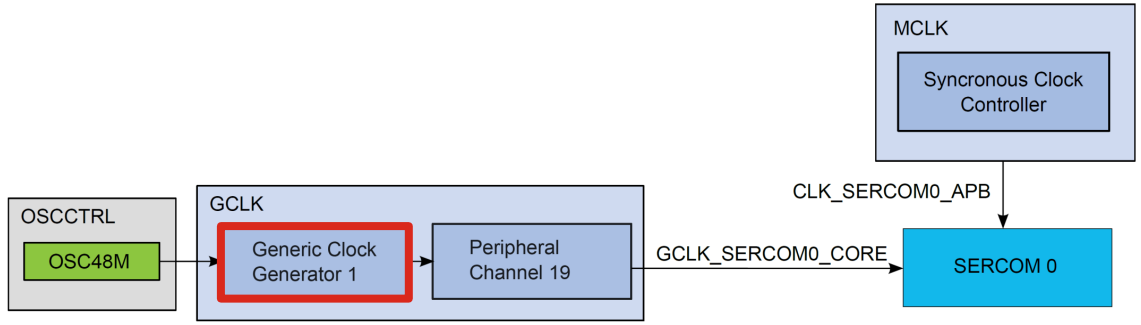
Clock Bus Domains and Peripheral Clock Setup

- **GCLK – Generic Clock (Async)**
  - Operations of the peripheral
  - Enabled in GENCTRLn and PCHCTRLn registers
  - Can use different clock from sync for power savings
- **MCLK – Main Clock (Sync)**
  - Register setup of the peripherals
  - Enabled in APBxMASK registers



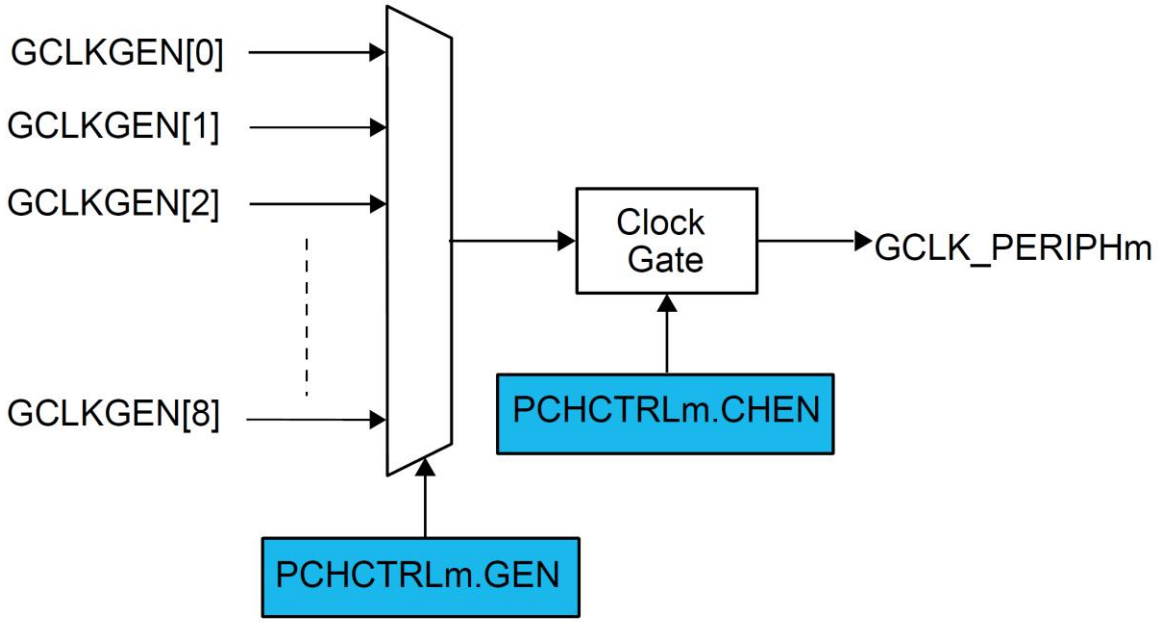
GCLK Generator Setup – GENCTRLn Register



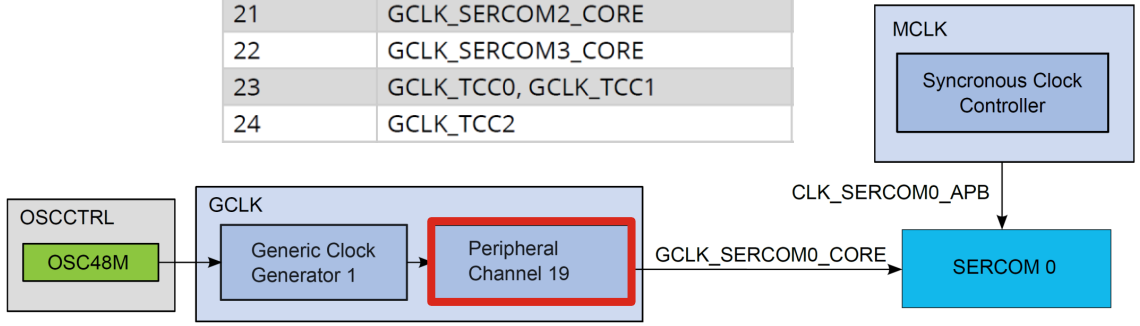


GCLK Peripheral Channel Setup – PCHCTRLm Register

Name: PCHCTRLm
Offset: 0x80 + m*0x04 [m=0..34]
Reset: 0x00000000
Property: PAC Write-Protection



index(m)	Name
0	GCLK_DPLL
1	GCLK_DPLL_32K
2	GCLK_EIC
3	GCLK_FREQM_MSR
4	GCLK_FREQM_REF
5	GCLK_TSENS
6	GCLK_EVSYS_CHANNEL_0
7	GCLK_EVSYS_CHANNEL_1
8	GCLK_EVSYS_CHANNEL_2
9	GCLK_EVSYS_CHANNEL_3
10	GCLK_EVSYS_CHANNEL_4
11	GCLK_EVSYS_CHANNEL_5
12	GCLK_EVSYS_CHANNEL_6
13	GCLK_EVSYS_CHANNEL_7
14	GCLK_EVSYS_CHANNEL_8
15	GCLK_EVSYS_CHANNEL_9
16	GCLK_EVSYS_CHANNEL_10
17	GCLK_EVSYS_CHANNEL_11
18	GCLK_SERCOM[0:3]_SLOW
19	GCLK_SERCOM0_CORE
20	GCLK_SERCOM1_CORE
21	GCLK_SERCOM2_CORE
22	GCLK_SERCOM3_CORE
23	GCLK_TCC0, GCLK_TCC1
24	GCLK_TCC2



MCLK/APB Clock Setup – APBxMASK Register

Name: APBCMASK
Offset: 0x1C
Reset: 0x00000000
Property: PAC Write-Protection

Bit	31	30	29	28	27	26	25	24
						PDEC		
Access						R/W		
Reset						0		

Bit	23	22	21	20	19	18	17	16
	CCL		DAC	AC	SDADC	ADC1	ADC0	TC4
Access	R/W		R/W	R/W	R/W	R/W	R/W	R/W
Reset	0		0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TC3	TC2	TC1	TC0	TCC2	TCC1	TCC0	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0
				SERCOM3	SERCOM2	SERCOM1	SERCOM0	EVSYS
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit 26 – PDEC PDEC APBC Clock Enable

Value	Description
0	The APBC clock for the PDEC is stopped.
1	The APBC clock for the PDEC is enabled.

Bit 23 – CCL CCL APBC Clock Enable

Value	Description
0	The APBC clock for the CCL is stopped.
1	The APBC clock for the CCL is enabled.

Bit 21 – DAC DAC APBC Mask Clock Enable

Value	Description
0	The APBC clock for the DAC is stopped.
1	The APBC clock for the DAC is enabled.

Bit 20 – AC AC APBC Mask Clock Enable

Value	Description
0	The APBC clock for the AC is stopped.
1	The APBC clock for the AC is enabled.

Bit 19 – SDADC SDADC APBC Clock Enable

Value	Description
0	The APBC clock for the SDADC is stopped.
1	The APBC clock for the SDADC is enabled.

Clock Synchronization

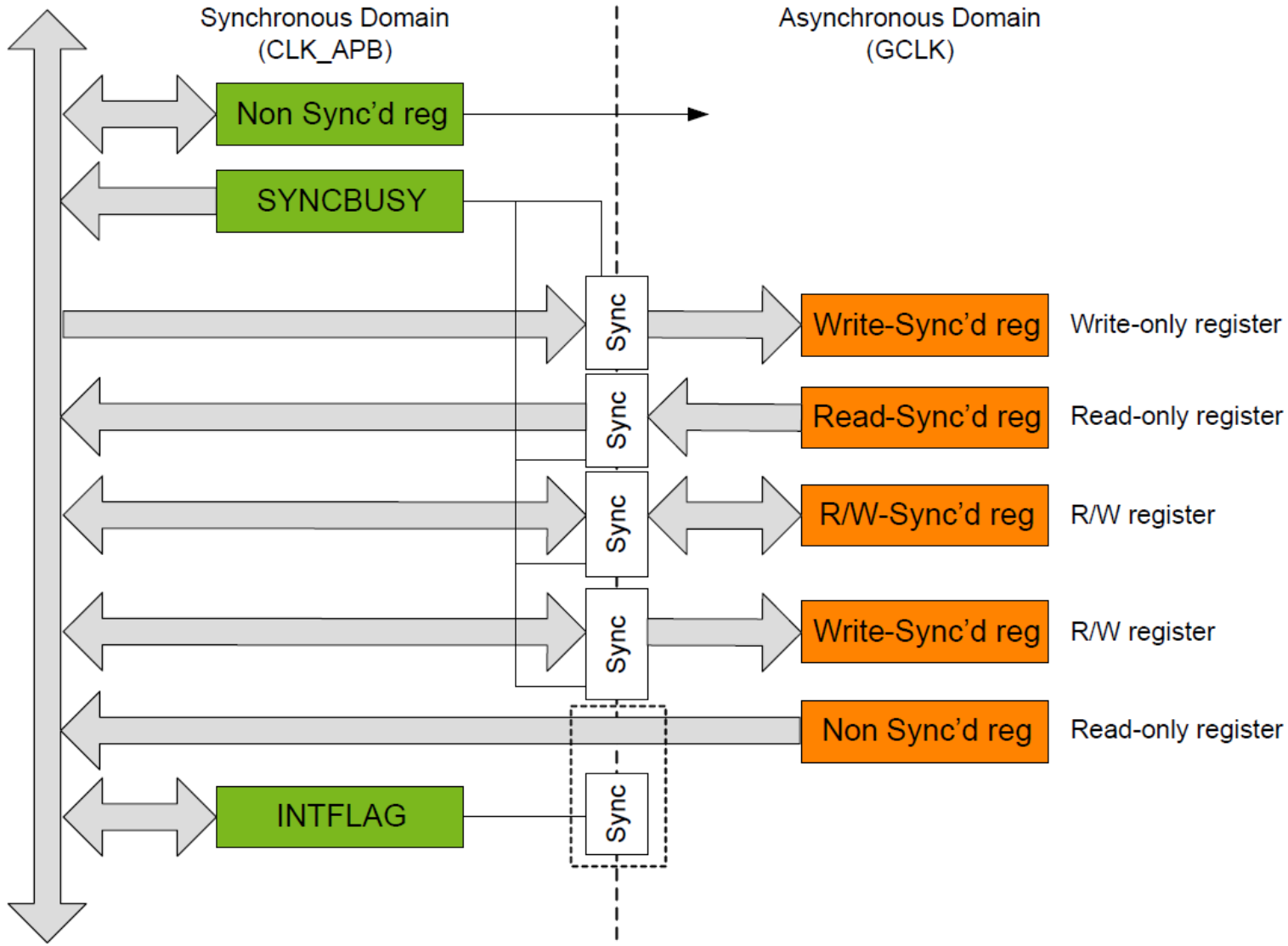
Synchronization Delay

Min:

$5\ P_{GCLK} + 2\ P_{APB}$

Max:

$6\ P_{GCLK} + 3\ P_{APB}$



SYNCBUSY Example – WDT Peripheral

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0	ALWAYSON					WEN	ENABLE	
0x01	CONFIG	7:0	WINDOW[3:0]					PER[3:0]		
0x02	EWCTRL	7:0						EWOFFSET[3:0]		
0x03	Reserved									
0x04	INTENCLR	7:0								EW
0x05	INTENSET	7:0								EW
0x06	INTFLAG	7:0								EW
0x07	Reserved									
0x08	SYNCBUSY	7:0				CLEAR	ALWAYSON	WEN	ENABLE	
		15:8								
		23:16								
		31:24								
0x0C	CLEAR	7:0	CLEAR[7:0]							

Name: CLOCK
Offset: 0x18
Reset: 0x00000000
Property: PAC Write-Protection, Write-Synchronized, Read-Synchronized

Bit 4 – CLEAR CLEAR Synchronization Busy

Value	Description
0	Write synchronization of the CLEAR register is complete.
1	Write synchronization of the CLEAR register is ongoing.

Clock Synchronization Summary

- **Write Sync Registers**
  - Write when sync bit is clear
  - Otherwise – write is discarded
- **Read Sync Registers**
  - Read when sync bit is clear
  - Otherwise – stale value

```
/* Wait for synchronization */
while(WDT_REGS->WDT_SYNCBUSY != 0U);
```

PIC32C and SAM Clock Sources

Oscillator Control Blocks

Osc Controller (OSCCTRL)

- XOSC (External)
- OSC48M/DFLL48M (Internal)
- FDPLL96M

32kHz Osc Controller (OSC32KCTRL)

- XOSC32K (External)
- OSC32K (Internal)
- OSCULP32K

MICROCHIP

Oscillator Control Blocks

Osc Controller (OSCCTRL)

- XOSC (External)
- OSC48M/DFL48M (Internal)
- FDPLL96M

32kHz Osc Controller (OSC32KCTRL)

- XOSC32K (External)
- OSC32K (Internal)
- OSCULP32K

MICROCHIP

48 MHz Internal Oscillator — OSC48M and DFLL48M

Internal oscillator with no external components

48 MHz output frequency

OSC48M/DFLL48M can operate stand-alone in open loop mode

DFLL48M can operate in closed loop mode for more accuracy

Can be a source for all clock generators

MICROCHIP

OSC48M Setup

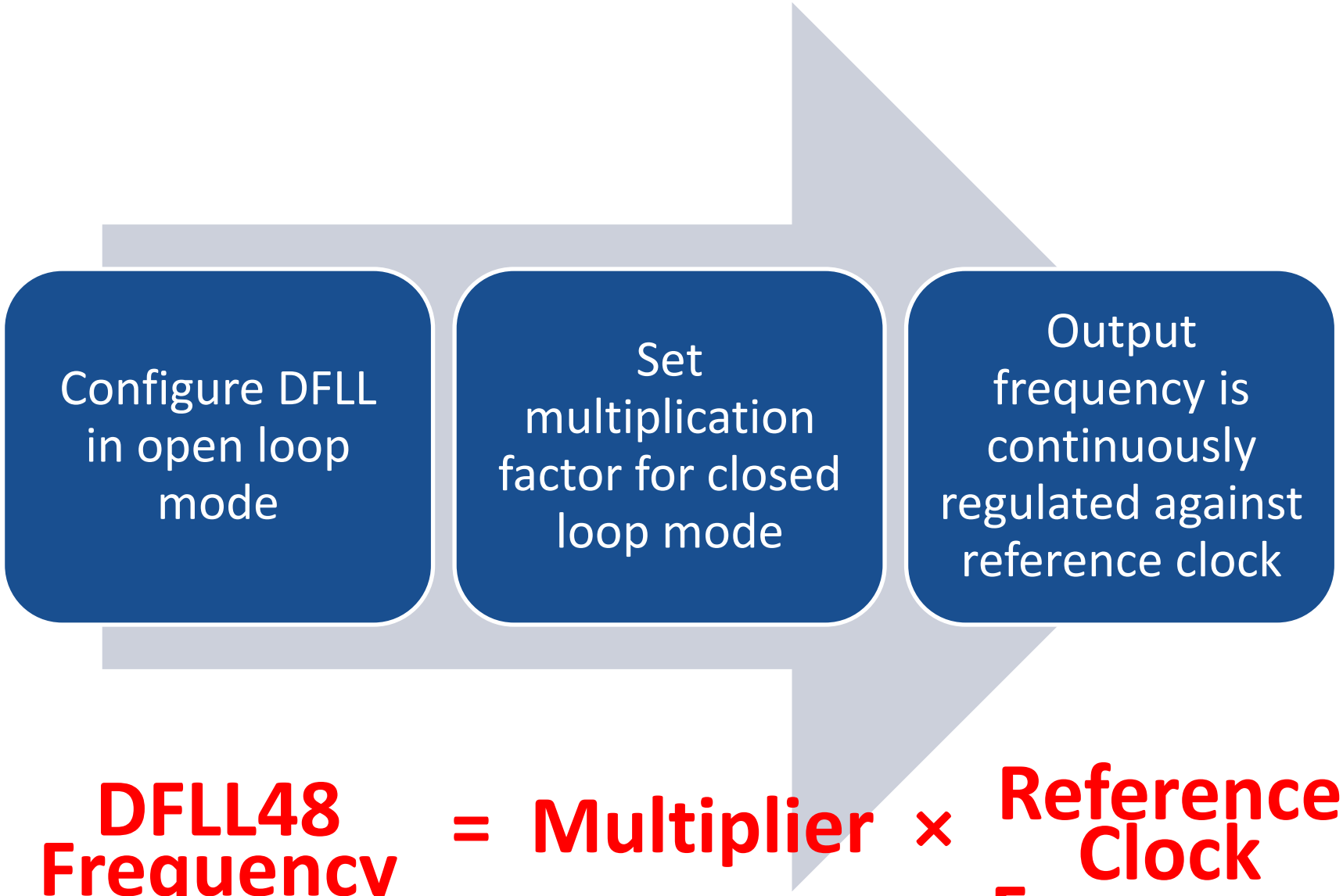
Enable by writing to OSC48MCTRL.ENABLE

Select frequency through OSC48MDIV register

Ready status reflected in STATUS.OSC48MRDY

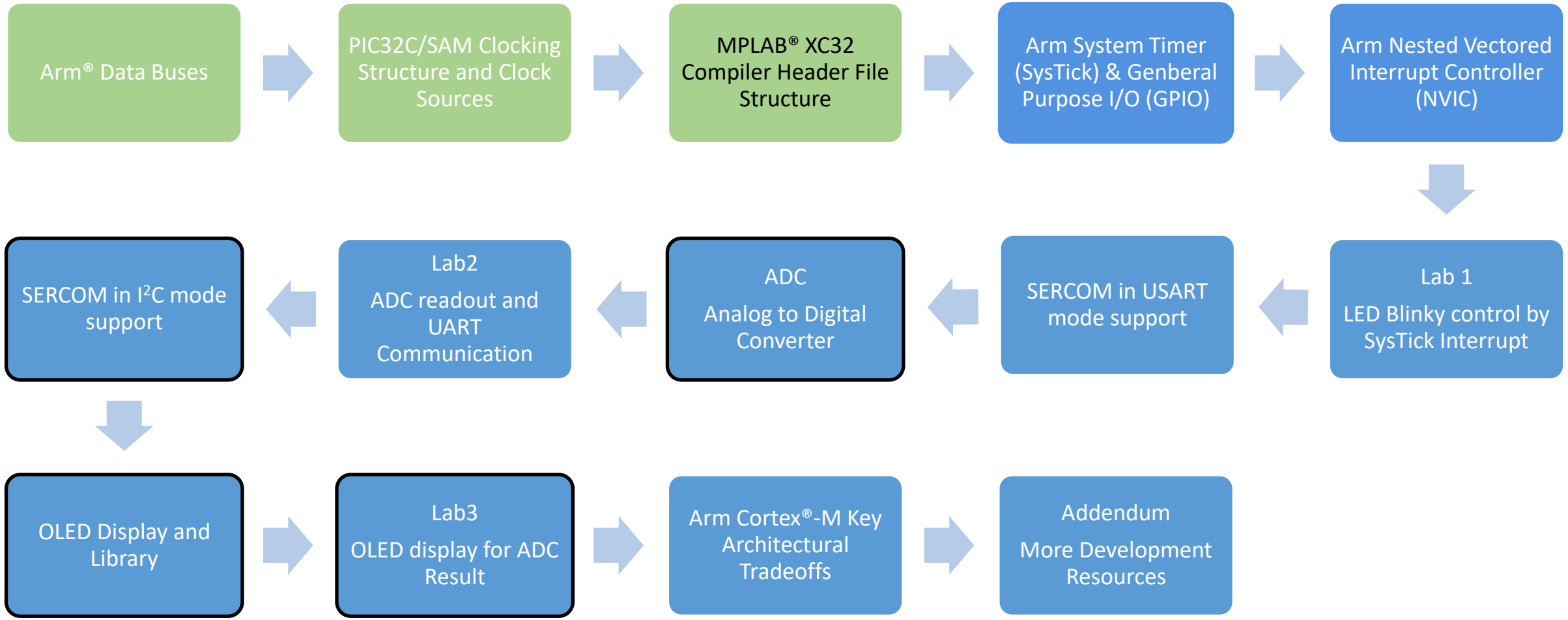
Default clock after reset at 4 MHz

DFLL48M Operation



DFLL48 Frequency = Multiplier × Reference Clock Frequency

Class Outline

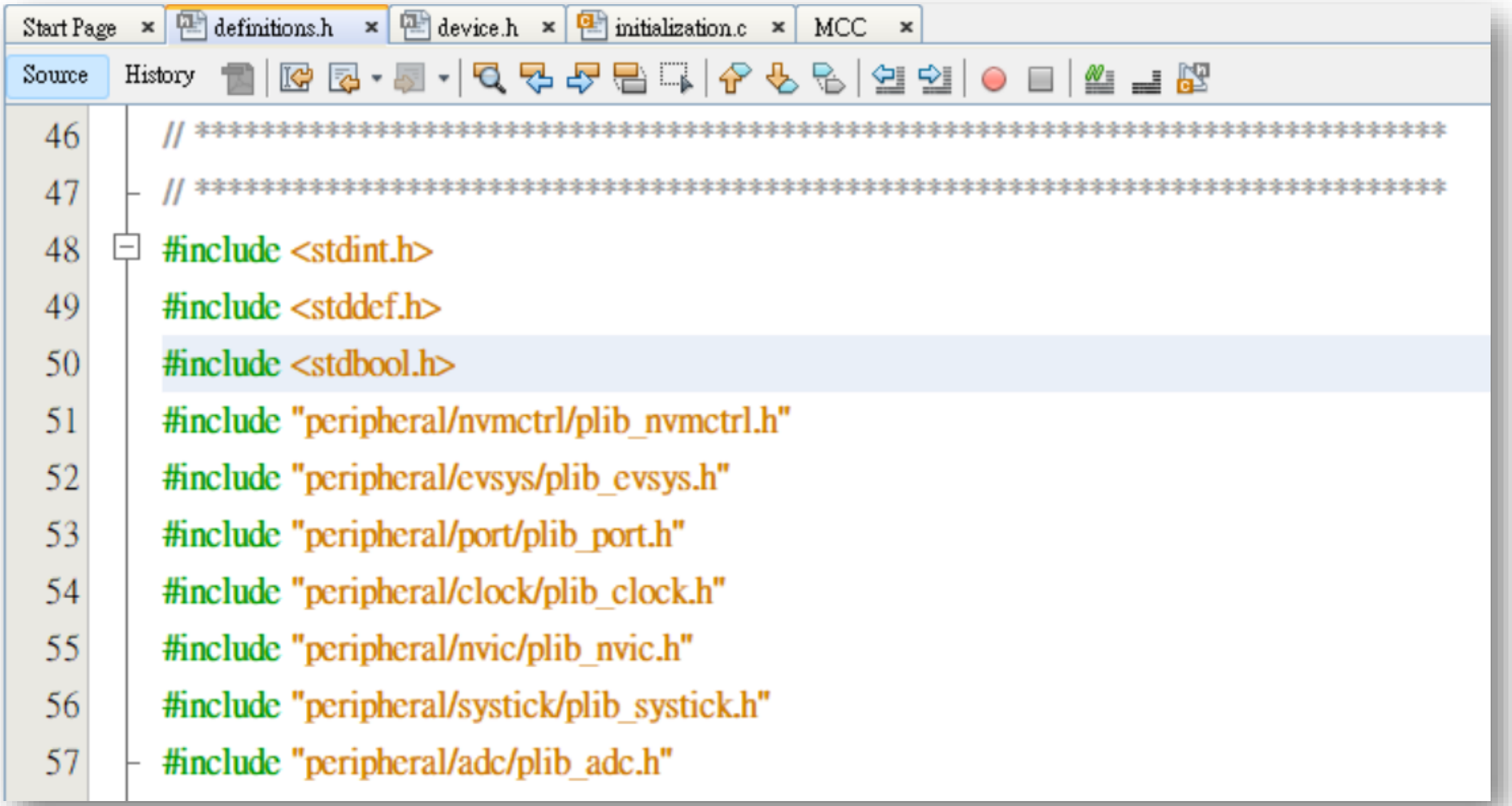


MPLAB® XC32 Compiler Header File Structure

MPLAB® XC32 Compiler Header Files

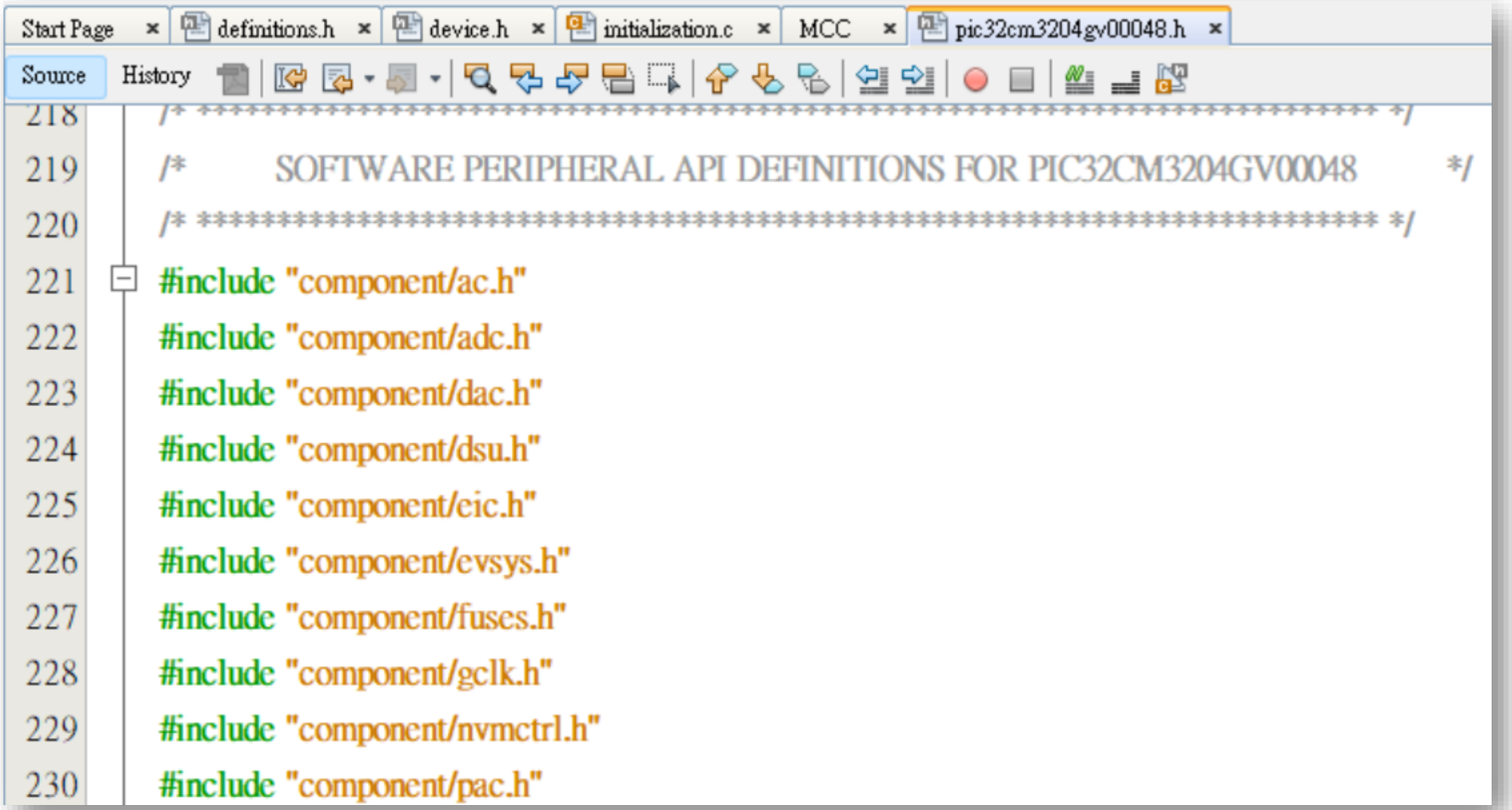
- <definitions.h> which includes necessary PLIB and device header files
- For the PIC32CM3204gv00048, the file pic32cm3204gv00048.h is included by <device.h>
- The pic32cm3204gv00048.h file will include other header files that provide peripheral information
- Initialization.c Include both definitions.h and device.h to allow the proper common header file to be included in the project

Example of definitions.h

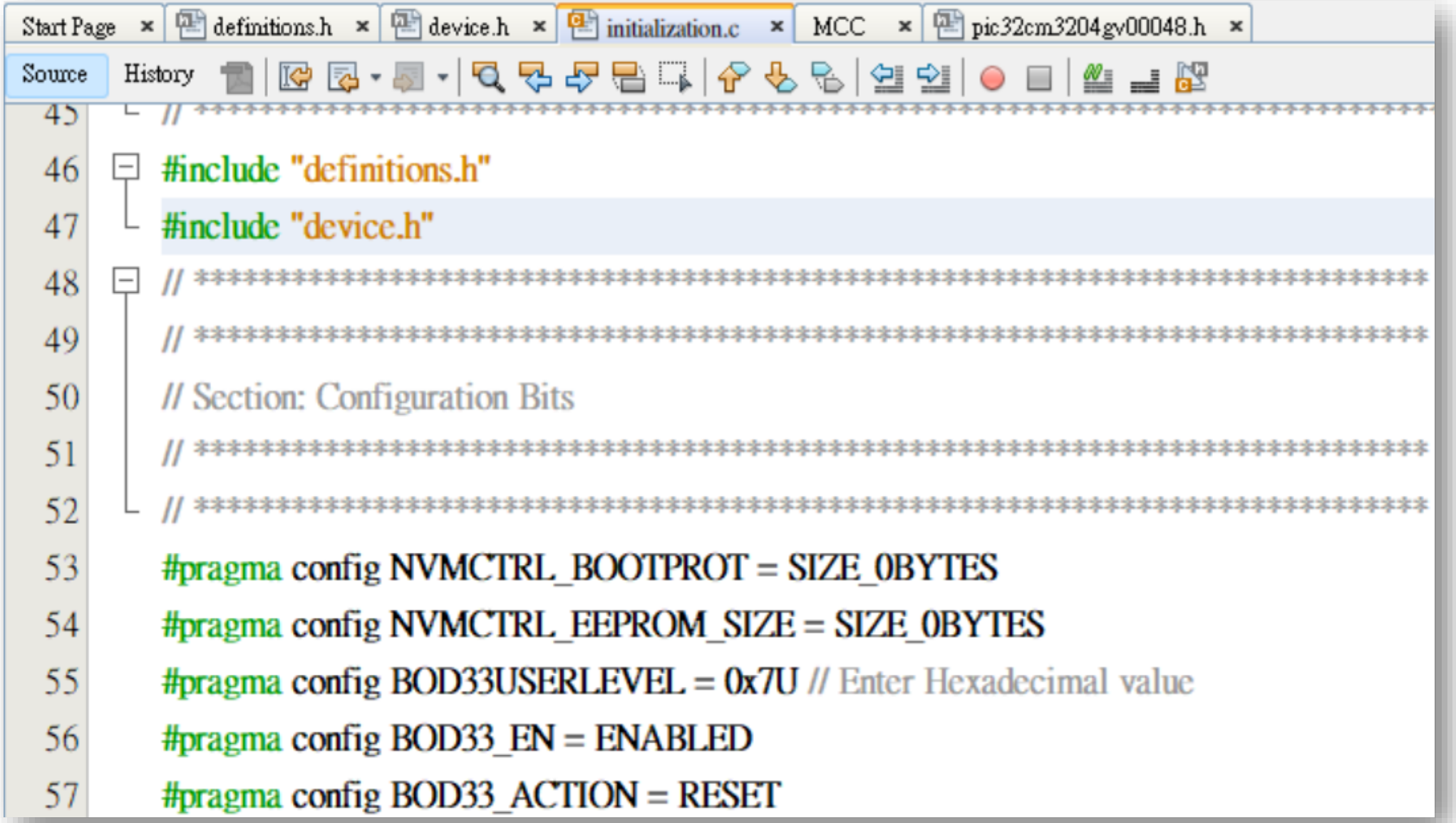


MICROCHIP

Example of pic32cm3204gv00048.h



Example of initialization.c



Device Header File

- **pic32cm3204mc00048.h contains the main block addresses for all peripherals**

Example

```
#if !(defined(__ASSEMBLER__) || defined(__IAR_SYSTEMS_ASM__))
#define AC_REGS                          ((ac_registers_t*)0x42004400)          /* AC Registers Address          */
#define ADC_REGS                         ((adc_registers_t*)0x42004000)         /* ADC Registers Address         */
#define DAC_REGS                         ((dac_registers_t*)0x42004800)         /* DAC Registers Address         */
#define DSU_REGS                         ((dsu_registers_t*)0x41002000)         /* DSU Registers Address         */
#define EIC_REGS                         ((eic_registers_t*)0x40001800)         /* EIC Registers Address         */
#define EVSYS_REGS                       ((evsys_registers_t*)0x42000400)       /* EVSYS Registers Address       */
#define GCLK_REGS                        ((gclk_registers_t*)0x40000c00)        /* GCLK Registers Address        */
```

- **These #defines are typecast as pointers and will be used to point to registers within their respective blocks**

MICROCHIP

Peripheral Header Files

Each peripheral has their own header file for all registers in the specific peripheral ("component/gclk.h" as example)

Example

```
typedef struct
{  /* Generic Clock Generator */
  __IO  uint8_t                  GCLK_CTRL;          /* Offset: 0x00 (R/W  8) Control */
  __I   uint8_t                  GCLK_STATUS;        /* Offset: 0x01 (R/   8) Status */
  __IO  uint16_t                 GCLK_CLKCTRL;       /* Offset: 0x02 (R/W  16) Generic Clock Control */
  __IO  uint32_t                 GCLK_GENCTRL;       /* Offset: 0x04 (R/W  32) Generic Clock Generator Control */
  __IO  uint32_t                 GCLK_GENDIV;        /* Offset: 0x08 (R/W  32) Generic Clock Generator Division */
} gclk_registers_t;
```


MICROCHIP

Peripheral Header Files

Structure members are listed in order of their placement in memory

Structure members are in offset positions from the block address

Block address pointer can access any register in the block

Example

OSC32KCTRL_REGS->OSC32KCTRL_STATUS
accesses the OSC32KCTRL_STATUS register within the OSC32KCTRL block

MICROCHIP

Peripheral Access Example

Use the definition in their own header file for all registers in the peripheral

Example

```
static void GCLK1_Initialize(void)
{
    GCLK_REGS->GCLK_GENCTRL = GCLK_GENCTRL_SRC(6U) | GCLK_GENCTRL_GENEN_Msk | GCLK_GENCTRL_ID(1U);

    GCLK_REGS->GCLK_GENDIV = GCLK_GENDIV_DIV(250U) | GCLK_GENDIV_ID(1U);
    while((GCLK_REGS->GCLK_STATUS & GCLK_STATUS_SYNCBUSY_Msk) == GCLK_STATUS_SYNCBUSY_Msk)
    {
        /* wait for the Generator 1 synchronization */
    }
}
```

Peripheral Header Files

Each peripheral (component) header file also contains bit field values arranged by registers

Example

```
/* -- OSC32KCTRL_STATUS : (OSC32KCTRL Offset: 0x0C) ( R/ 32) Power and Clocks Status -- */
#define OSC32KCTRL_STATUS_RESETVALUE                          _UINT32_(0x00)

#define OSC32KCTRL_STATUS_XOSC32KRDY_Pos                      _UINT32_(0)
#define OSC32KCTRL_STATUS_XOSC32KRDY_Msk                      (_UINT32_(0x1) << OSC32KCTRL_STATUS_XOSC32KRDY_Pos)
#define OSC32KCTRL_STATUS_XOSC32KRDY(value)                   (OSC32KCTRL_STATUS_XOSC32KRDY_Msk &
                                                              (_UINT32_(value) << OSC32KCTRL_STATUS_XOSC32KRDY_Pos))
#define OSC32KCTRL_STATUS_OSC32KRDY_Pos                       _UINT32_(1)
#define OSC32KCTRL_STATUS_OSC32KRDY(value)                    (OSC32KCTRL_STATUS_OSC32KRDY_Msk &
                                                              (_UINT32_(value) << OSC32KCTRL_STATUS_OSC32KRDY_Pos))
#define OSC32KCTRL_STATUS_CLKFAIL_Pos                         _UINT32_(2)
#define OSC32KCTRL_STATUS_CLKFAIL_Msk                         (_UINT32_(0x1) << OSC32KCTRL_STATUS_CLKFAIL_Pos)
#define OSC32KCTRL_STATUS_CLKFAIL(value)                      (OSC32KCTRL_STATUS_CLKFAIL_Msk &
                                                              (_UINT32_(value) << OSC32KCTRL_STATUS_CLKFAIL_Pos))
#define OSC32KCTRL_STATUS_CLKSW_Pos                           _UINT32_(3)
#define OSC32KCTRL_STATUS_CLKSW_Msk                           (_UINT32_(0x1) << OSC32KCTRL_STATUS_CLKSW_Pos)
#define OSC32KCTRL_STATUS_CLKSW(value)                        (OSC32KCTRL_STATUS_CLKSW_Msk &
                                                              (_UINT32_(value) << OSC32KCTRL_STATUS_CLKSW_Pos))
#define OSC32KCTRL_STATUS_Msk                                 _UINT32_(0x0000000F)
```


MICROCHIP

Port and Pin Header File

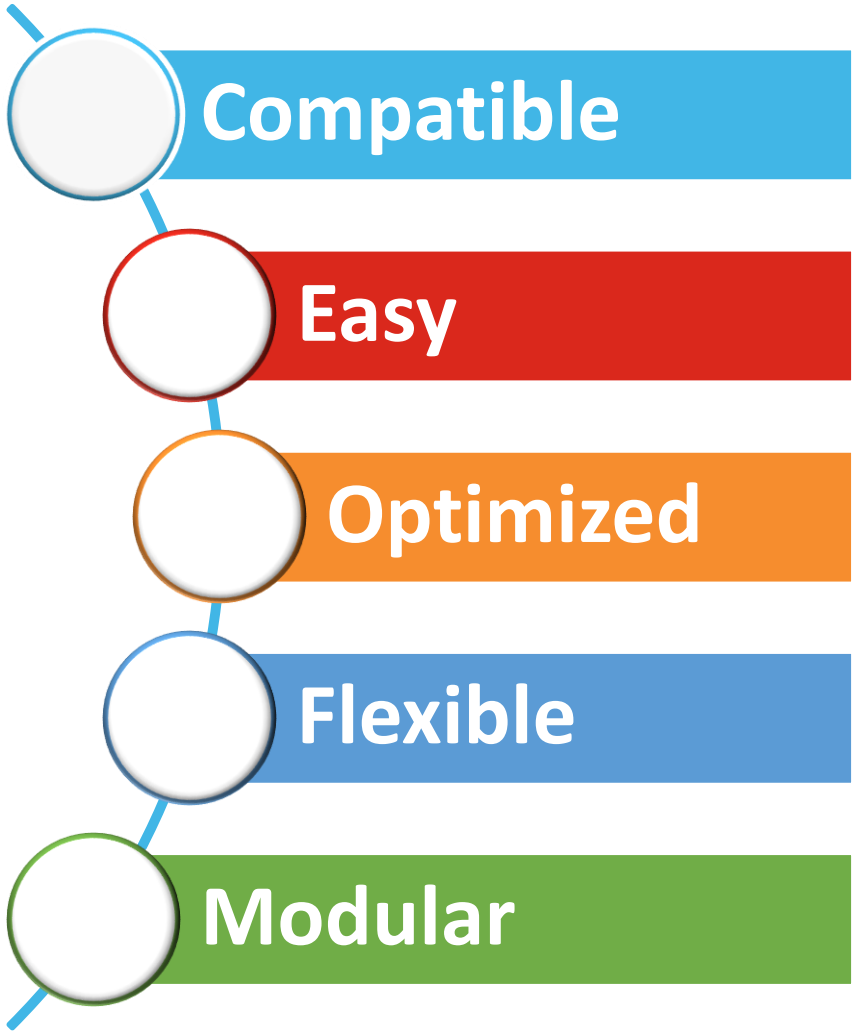
- An additional `pic32cm3204gv00048.h` file is located under the “pio” folder in the compiler directory
- This file contains pin names and pin functionality setup constants for pin multiplexing

Example

```
/* ========== Peripheral I/O pin numbers ========== */
#define PIN_PA00                            (    0)              /**< Pin Number for PA00 */
#define PIN_PA01                            (    1)              /**< Pin Number for PA01 */
#define PIN_PA02                            (    2)              /**< Pin Number for PA02 */

/* ========== Peripheral I/O masks ========== */
#define PORT_PA00                           (_UINT32_(1) << 0)    /**< PORT mask for PA00 */
#define PORT_PA01                           (_UINT32_(1) << 1)    /**< PORT mask for PA01 */
#define PORT_PA02                           (_UINT32_(1) << 2)    /**< PORT mask for PA02 */

/* ========== PORT definition for ADC0 peripheral ========== */
#define PIN_PA02B_ADC0_AIN0                 _UINT32_(2)
#define MUX_PA02B_ADC0_AIN0                 _UINT32_(1)
#define PINMUX_PA02B_ADC0_AIN0              ((PIN_PA02B_ADC0_AIN0 << 16) | MUX_PA02B_ADC0_AIN0)
#define PORT_PA02B_ADC0_AIN0                (_UINT32_(1) << 2)
```


MICROCHIP

MCC Harmony v3



Support for 32-bit Arm® and MIPS®, MCUs and MPUs

Device and Software Configuration

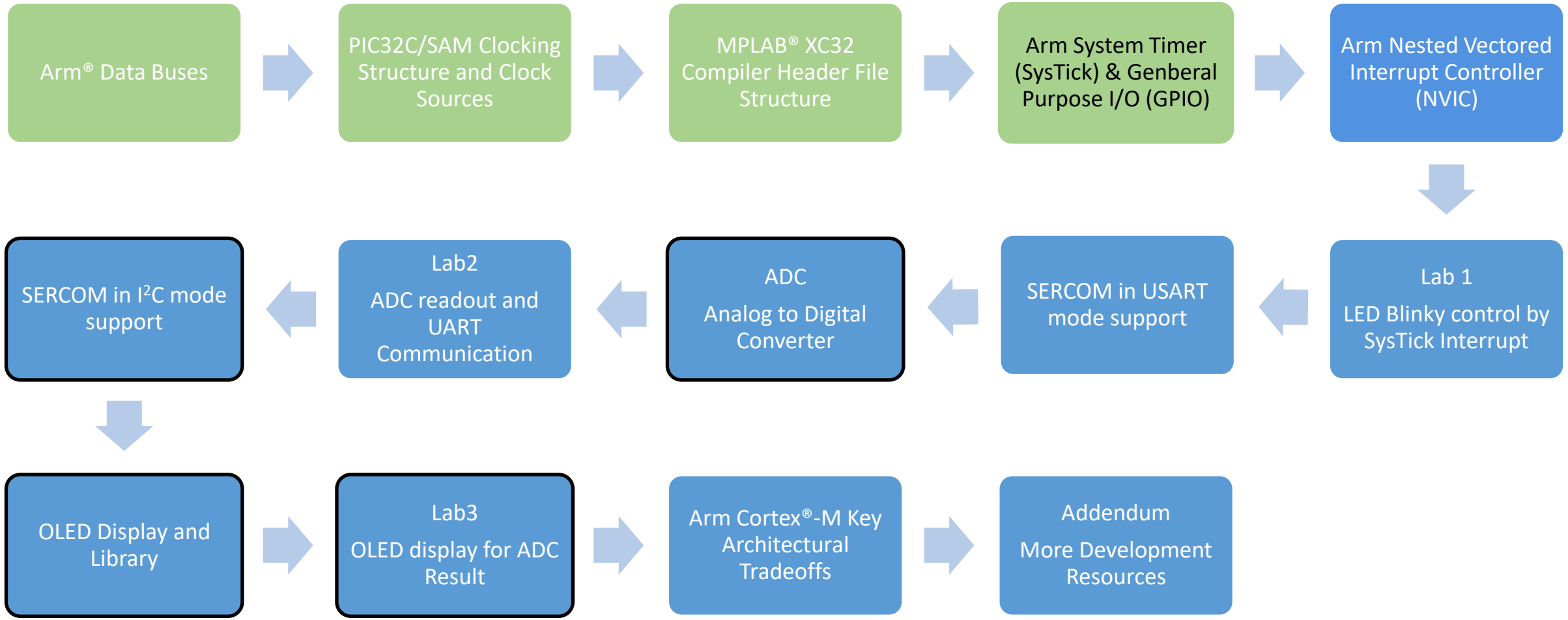
Small Footprint Libraries

Design at Customer-preferred Level

Discrete Packages available in GitHub

MICROCHIP

Class Outline



SysTick Arm® System Timer

SysTick

24-bit core system timer

Clear-on-write
Decrementing
Wrap-on-zero

Clocked by MCLK
Synchronous to processor

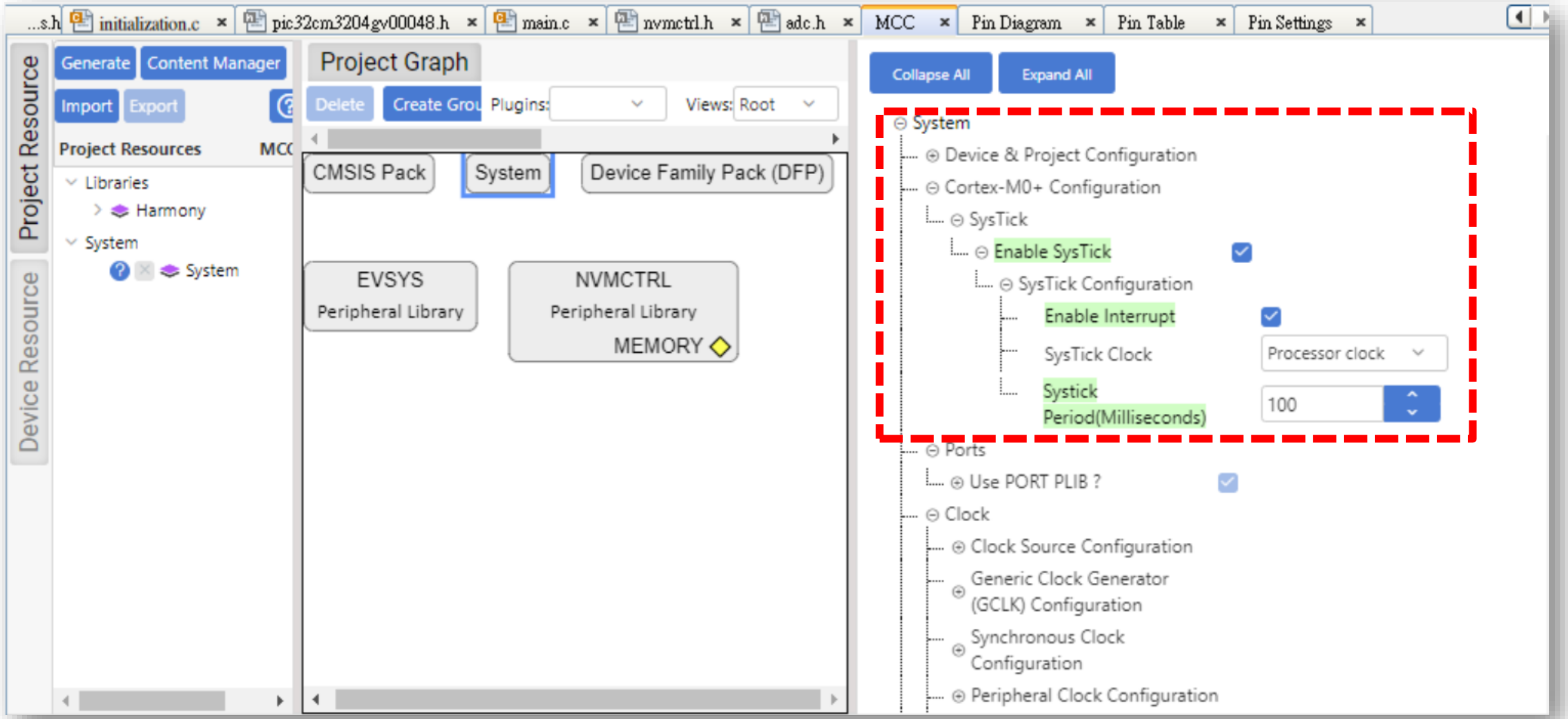
Calibration value of 10 ms rollover

Can interrupt on wrap-on-zero

Auto loads from register

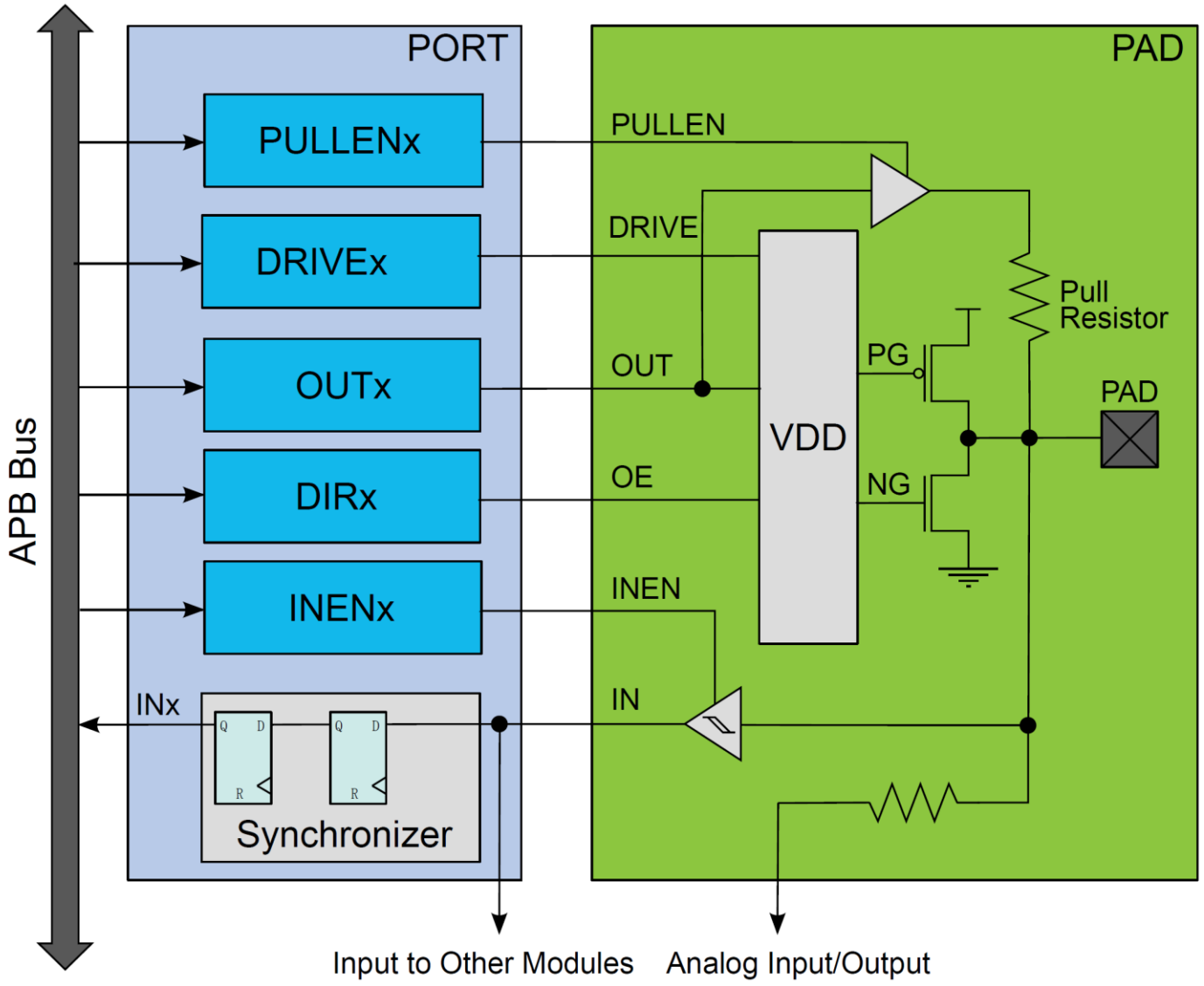
MICROCHIP

SysTick – Initialize from System



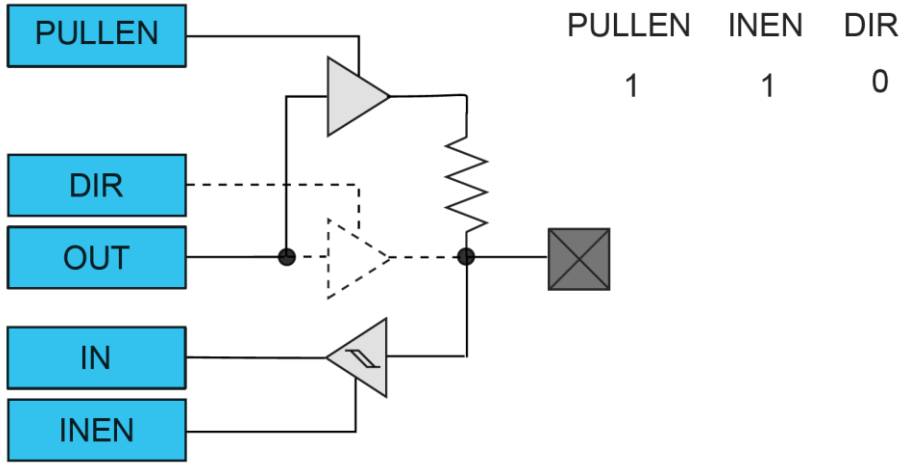
PIC32C and SAM General Purpose I/O (GPIO)

GPIO Structure

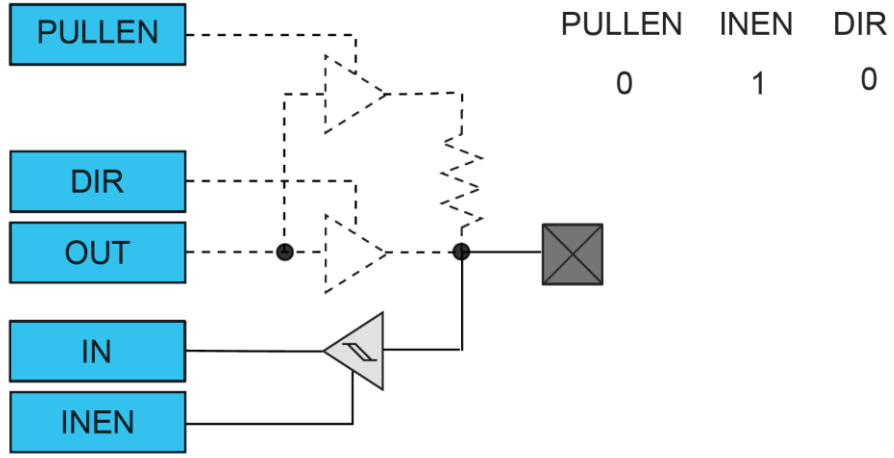


Pin Configuration Registers

DIR	INEN	PULLEN	OUT	Configuration
0	0	0	X	Reset or analog I/O: all digital disabled
0	0	1	0	Pull-down; input buffer disabled
0	0	1	1	Pull-up; input buffer disabled
0	1	0	X	Input
0	1	1	0	Input with pull-down
0	1	1	1	Input with pull-up
1	0	X	X	Output; input buffer disabled
1	1	X	X	Output; input enabled



Input with Pull



Standard Input

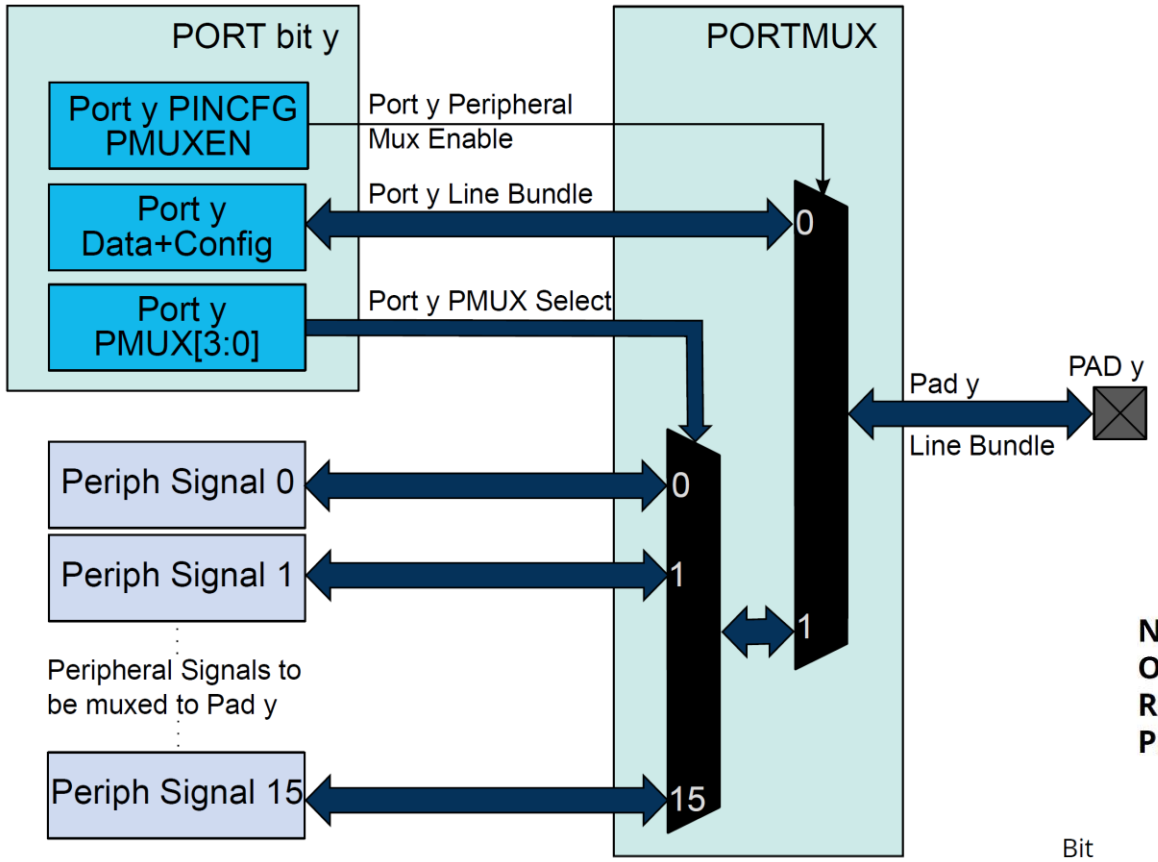
Registers Used for GPIO

Offset	Name	Bit Pos.	7	6	5	4	3
0x00	DIR	7:0				DIR[7:0]	
		15:8				DIR[15:8]	
		23:16				DIR[23:16]	
		31:24				DIR[31:24]	
0x04	DIRCLR	7:0				DIRCLR[7:0]	
		15:8				DIRCLR[15:8]	
		23:16				DIRCLR[23:16]	
		31:24				DIRCLR[31:24]	
0x08	DIRSET	7:0				DIRSET[7:0]	
		15:8				DIRSET[15:8]	
		23:16				DIRSET[23:16]	
		31:24				DIRSET[31:24]	
0x0C	DIRTGL	7:0				DIRTGL[7:0]	
		15:8				DIRTGL[15:8]	
		23:16				DIRTGL[23:16]	
		31:24				DIRTGL[31:24]	

0x10	OUT	7:0
		15:8
		23:16
		31:24
0x14	OUTCLR	7:0
		15:8
		23:16
		31:24
0x18	OUTSET	7:0
		15:8
		23:16
		31:24
0x1C	OUTTGL	7:0
		15:8
		23:16
		31:24
0x20	IN	7:0
		15:8
		23:16
		31:24

MICROCHIP

Connecting GPIO to Peripherals



- PINCFGn.PMUXEN
  - Allows peripheral to drive IO
- PMUXm.PMUX(O/E)
  - Chooses which peripheral signal goes to which odd/even numbered pin

Name: PMUX
Offset: 0x30 + n*0x01 [n=0..15]
Reset: 0x00 except group 0 PMUX15 = 0x06
Property: PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
	PMUXO[3:0]				PMUXE[3:0]			
Access	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0

Connecting GPIO to Peripherals – Muxing Registers

PMUX.PMUXO/E value: 0 = A, 1 = B, …, 9 = J

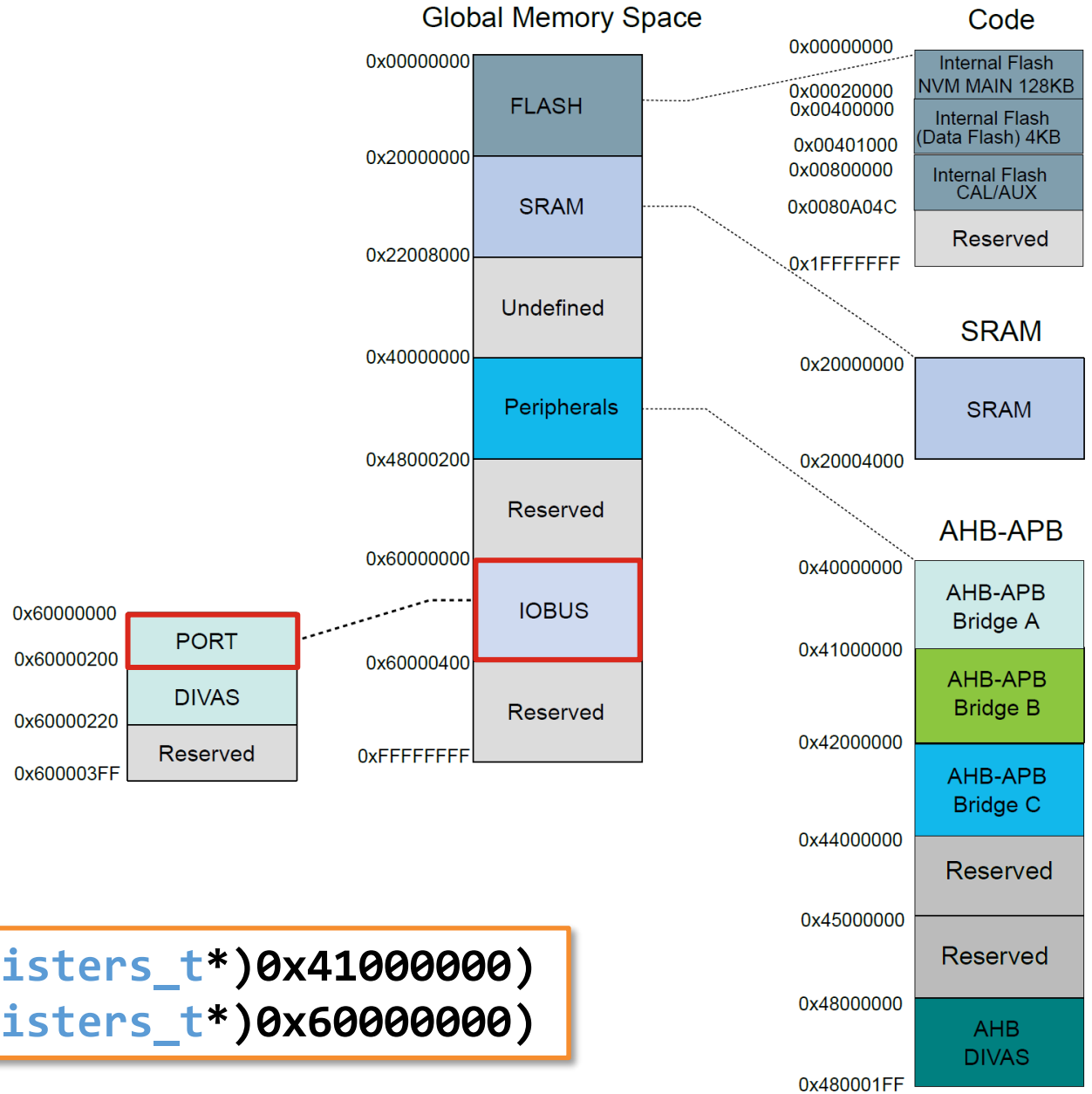
48-pin QFN	Pin name	A	B						C	D	E	F	G	H	I	J	Supply	Reset State
		EIC	REF	ADC0	ADC1	SDADC	AC	DAC	SERCOM	SERCOM-ALT	TC/TCC	TCC	PDEC	AC/ GCLK	CCL	CCL/PDEC		
1	PA00/ XIN32	EXTINT[0]								SERCOM1/ PAD[0]	TCC2/ WO[0]						VDDANA	I/O, HI-Z
2	PA01/ XOUT32	EXTINT[1]								SERCOM1/ PAD[1]	TCC2/ WO[1]						VDDANA	I/O, HI-Z
3	PA02	EXTINT[2]		AIN[0]				VOUT									VDDANA	I/O, HI-Z
4	PA03	EXTINT[3]	VREFA	AIN[1]													VDDANA	I/O, HI-Z
5	GNDANA																GNDANA	
6	VDDANA																VDDANA	
7	PB08	EXTINT[8]		AIN[2]	AIN[4]	INN[1]					TC0/ WO[0]				CCL/ IN[8]		VDDANA	I/O, HI-Z
8	PB09	EXTINT[9]		AIN[3]	AIN[5]	INP[1]					TC0/ WO[1]				CCL/ OUT[2]		VDDANA	I/O, HI-Z
9	PA04	EXTINT[4]	VREFB	AIN[4]			AIN[0]			SERCOM0/ PAD[0]	TCC0/ WO[0]				CCL/ IN[0]		VDDANA	I/O, HI-Z
10	PA05	EXTINT[5]		AIN[5]			AIN[1]			SERCOM0/ PAD[1]	TCC0/ WO[1]				CCL/ IN[1]		VDDANA	I/O, HI-Z
11	PA06	EXTINT[6]		AIN[6]		INN[0]	AIN[2]			SERCOM0/ PAD[2]	TCC1/ WO[0]				CCL/ IN[2]		VDDANA	I/O, HI-Z
12	PA07	EXTINT[7]		AIN[7]		INP[0]	AIN[3]			SERCOM0/ PAD[3]	TCC1/ WO[1]				CCL/ OUT[0]	CCL/ OUT[3]	VDDANA	I/O, HI-Z
13	PA08(1)	NMI		AIN[8]	AIN[10]				SERCOM0/ PAD[0]	SERCOM2/ PAD[0]	TCC0/ WO[0]	TCC1/ WO[2]			CCL/ IN[3]	PDEC[0]	VDDIO	I/O, HI-Z

MICROCHIP

Port Access Through Single-Cycle IOBUS

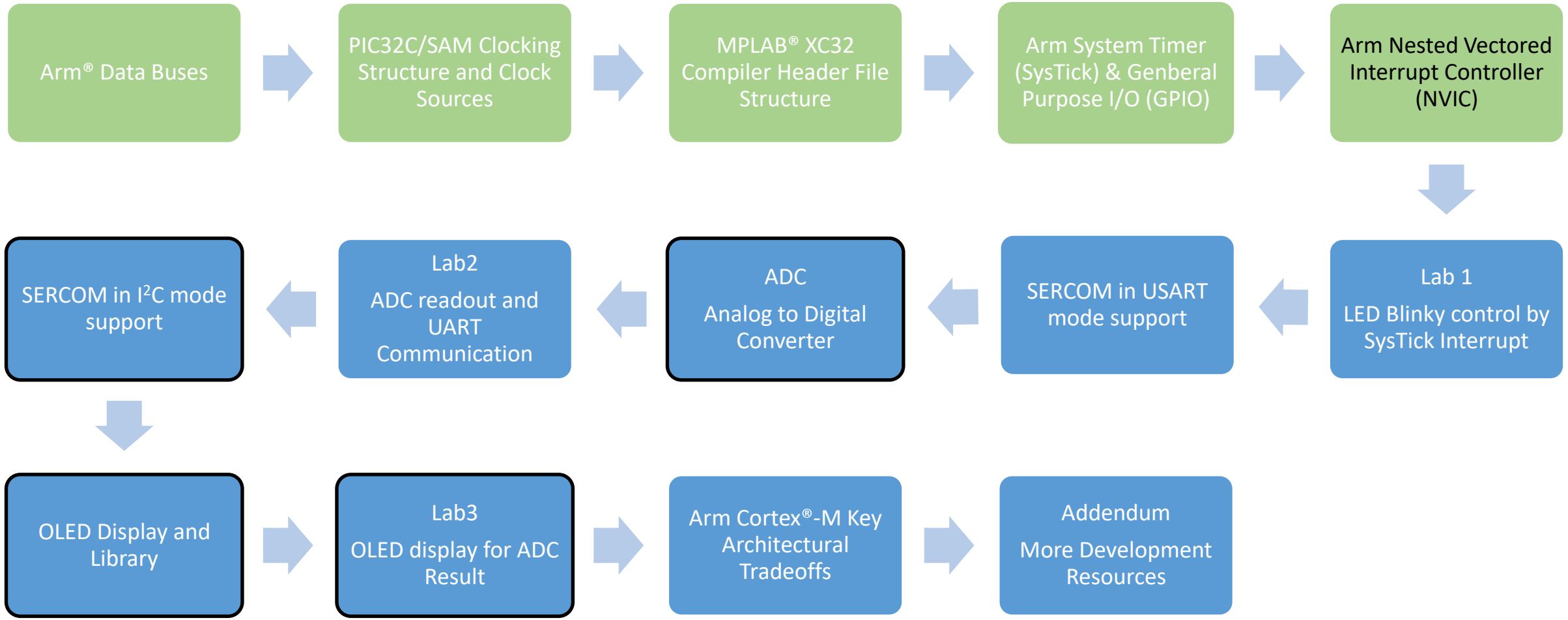
Port access priority:

1. Arm® CPU **IOBUS** (No wait tolerated).
2. Arm CPU **AHB/APB bridge.**
3. **Event System** through asynchronous input events.




```
#define PORT_REGS          ((port_registers_t*)0x41000000)
#define PORT_IOBUS_REGS    ((port_registers_t*)0x60000000)
```

Class Outline



Arm® Nested Vector Interrupt Controller (NVIC)

Nested Vector Interrupt Controller (NVIC) Structure

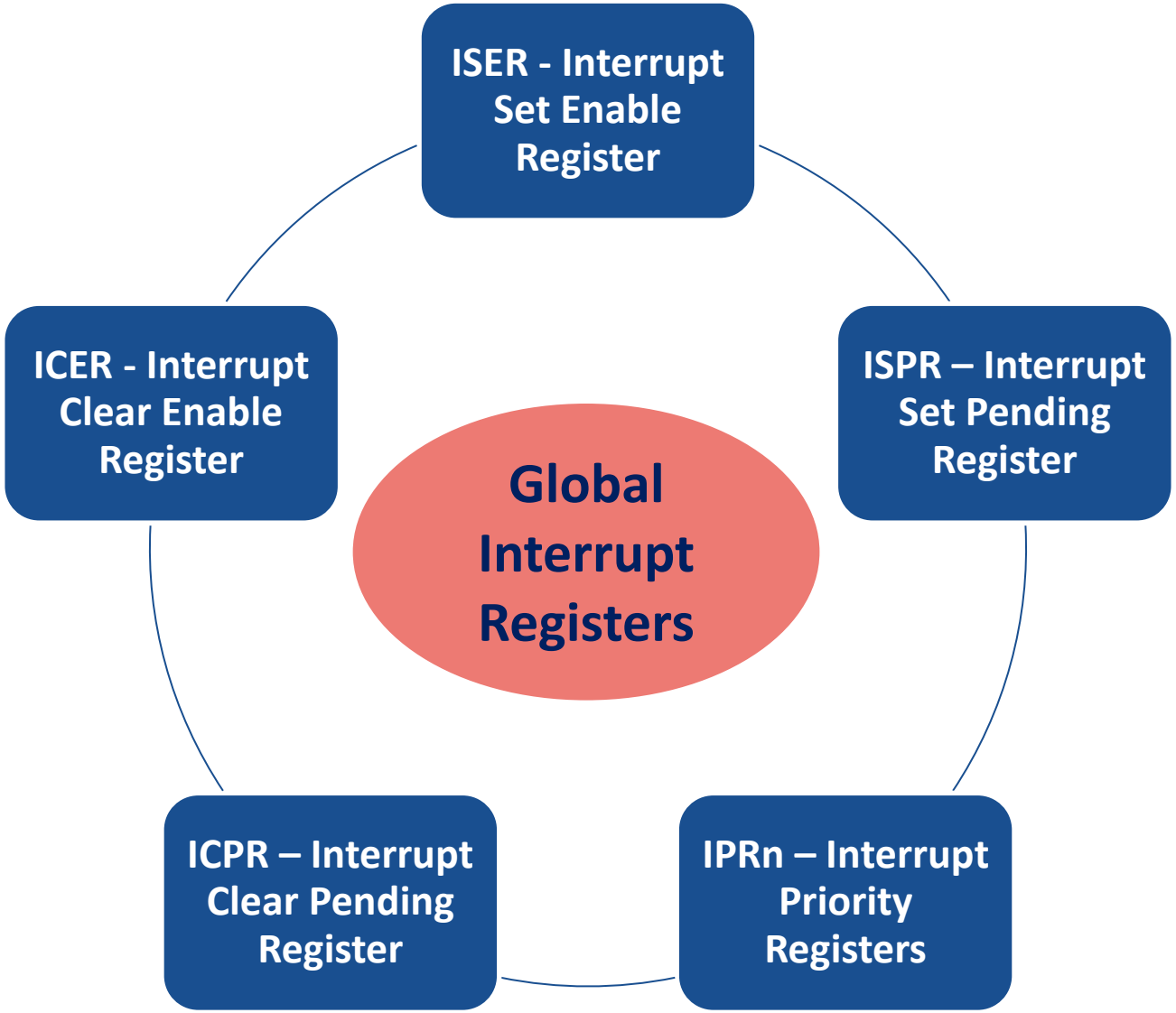
Each peripheral has an assigned vector interrupt

Relocatable vector table via **SCB->VTOR** register

Up to 32 interrupt sources, each with four levels of priority on Arm® Cortex®-M0+ devices (more on Cortex-M23/M4/M33)

Deterministic latency for highest priority interrupt
(M0+ 15 instructions; M23 15/27 instructions; M4 12 instructions)

Arm® NVIC Registers



Peripheral Interrupt Line Mapping

- Arm® Cortex®-M0+ has 32 NVIC lines
- Each interrupt line is connected to one peripheral instance
- Each peripheral can have one or more interrupt flags located in the peripheral's INTFLAG register
- Other Arm Cortex-M cores have up to 240/480 lines, which allows each peripheral to have multiple lines mapped directly to NVIC

Peripheral Source	NVIC Line
External Interrupt Controller (EIC NMI)	NMI
Power Manager (PM) Main Clock (MCLK) Oscillators Controller (OSCCTRL) 32 kHz Oscillators Controller (OSC32KCTRL) Supply Controller (SUPC) Protection Access Controller (PAC)	0
Watchdog Timer (WDT)	1
Real Time Clock (RTC)	2
External Interrupt Controller (EIC)	3
Frequency Meter (FREQM)	4
Temperature Sensor (TSENS)	5
Non-Volatile Memory Controller (NVMCTRL)	6
Direct Memory Access Controller (DMAC)	7
Event System (EVSYS)	8
Serial Communication Controller 0 (SERCOM0)	9
Serial Communication Controller 1 (SERCOM1)	10
Serial Communication Controller 2 (SERCOM2)	11
Serial Communication Controller 3 (SERCOM3)	12
Timer Counter for Control 0 (TCC0)	13
Timer Counter for Control 1 (TCC1)	14
Timer Counter for Control 2 (TCC2)	15
Timer Counter 0 (TC0)	16
Timer Counter 1 (TC1)	17
Timer Counter 2 (TC2)	18
Timer Counter 3 (TC3)	19
Timer Counter 4 (TC4)	20
Analog-to-Digital Converter 0 (ADC0)	21
Analog-to-Digital Converter 1 (ADC1)	22
Analog Comparator (AC)	23
Digital-to-Analog Converter (DAC)	24
SDADC	25
Position Decoder (PDEC)	26
Reserved	27-31

NVIC Specific Header Files

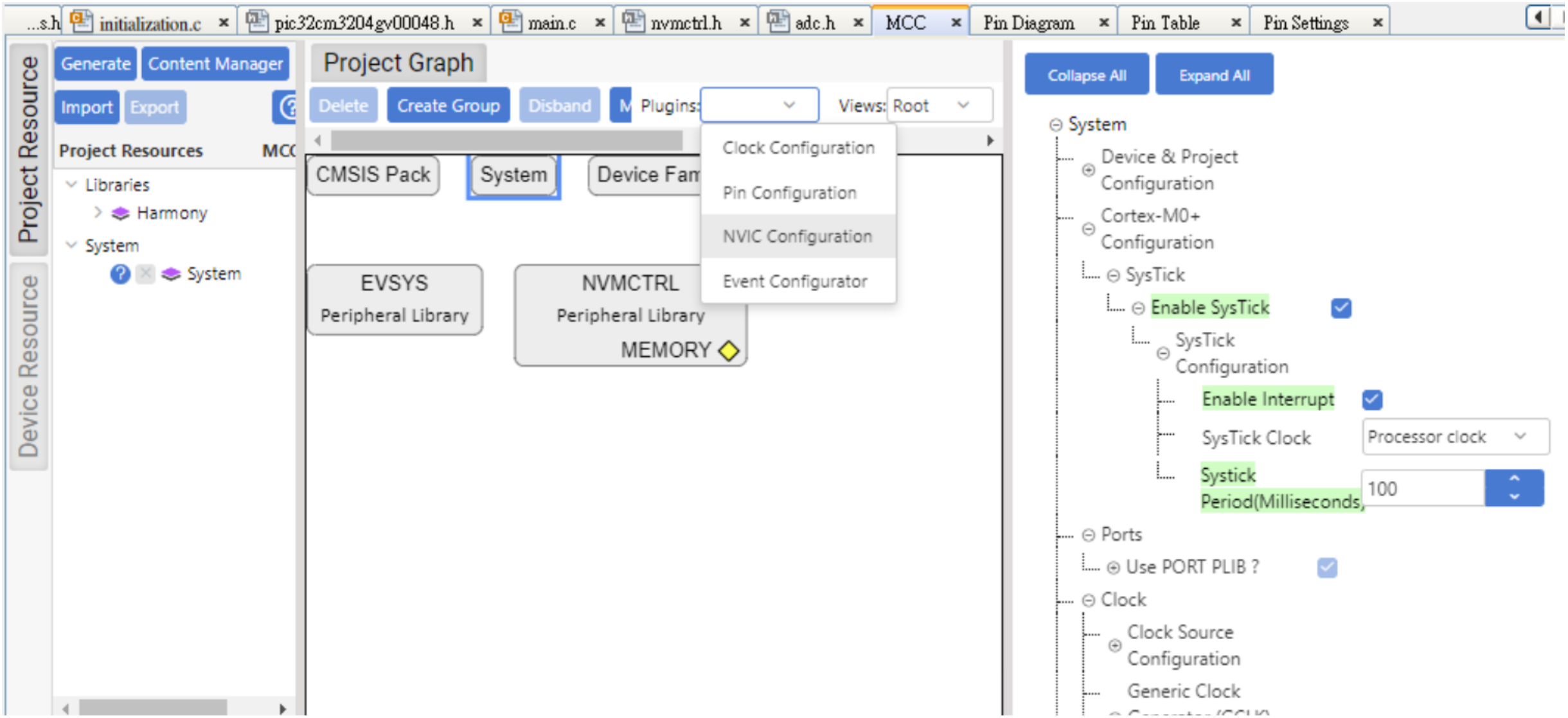
- **Interrupt names (IRQn) are located in the device header file (`pic32cm1216mc00032.h`)**
- **To enable TC3 interrupt:**

```
NVIC_EnableIRQ( TC3_IRQn );
```

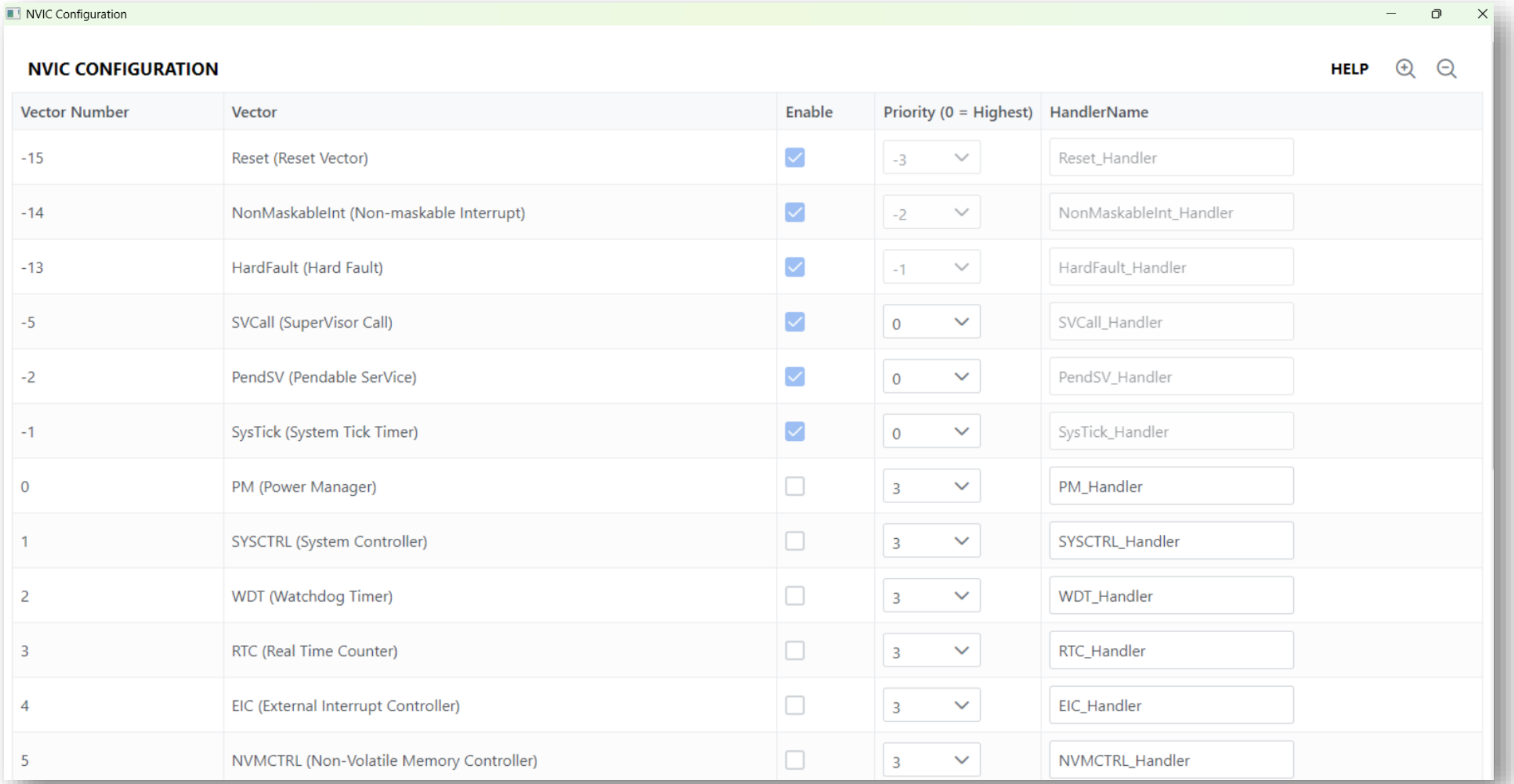
- **Must also enable TC3 peripheral interrupt in the TC3 INTENSET register (“peripheral” interrupt enable):**

Bit	7	6	5	4	3	2	1	0
			MC1	MC0			ERR	OVF
Access			R/W	R/W			R/W	R/W
Reset			0	0			0	0

NVIC Configuration Plugin in MCC Harmony

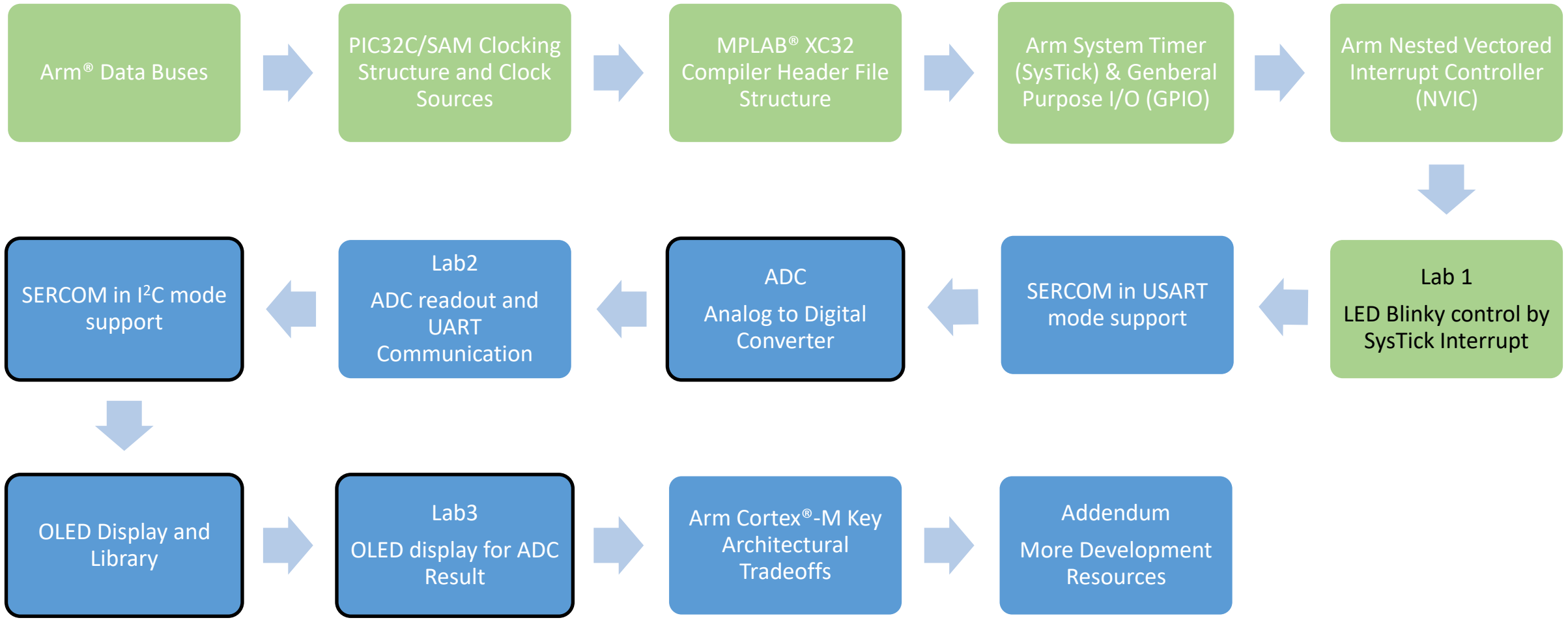


NVIC Configuration in MCC Harmony



Vector Number	Vector	Enable	Priority (0 = Highest)	HandlerName
-15	Reset (Reset Vector)	☑	-3	Reset_Handler
-14	NonMaskableInt (Non-maskable Interrupt)	☑	-2	NonMaskableInt_Handler
-13	HardFault (Hard Fault)	☑	-1	HardFault_Handler
-5	SVCall (SuperVisor Call)	☑	0	SVCall_Handler
-2	PendSV (Pendable SerVice)	☑	0	PendSV_Handler
-1	SysTick (System Tick Timer)	☑	0	SysTick_Handler
0	PM (Power Manager)	☐	3	PM_Handler
1	SYSCTRL (System Controller)	☐	3	SYSCTRL_Handler
2	WDT (Watchdog Timer)	☐	3	WDT_Handler
3	RTC (Real Time Counter)	☐	3	RTC_Handler
4	EIC (External Interrupt Controller)	☐	3	EIC_Handler
5	NVMCTRL (Non-Volatile Memory Controller)	☐	3	NVMCTRL_Handler

Class Outline



Lab 1 – Introduction Lab

MPLAB® X IDE project creation & Systick

Lab 1 Objectives

CREATE A PROJECT IN MPLAB® X IDE

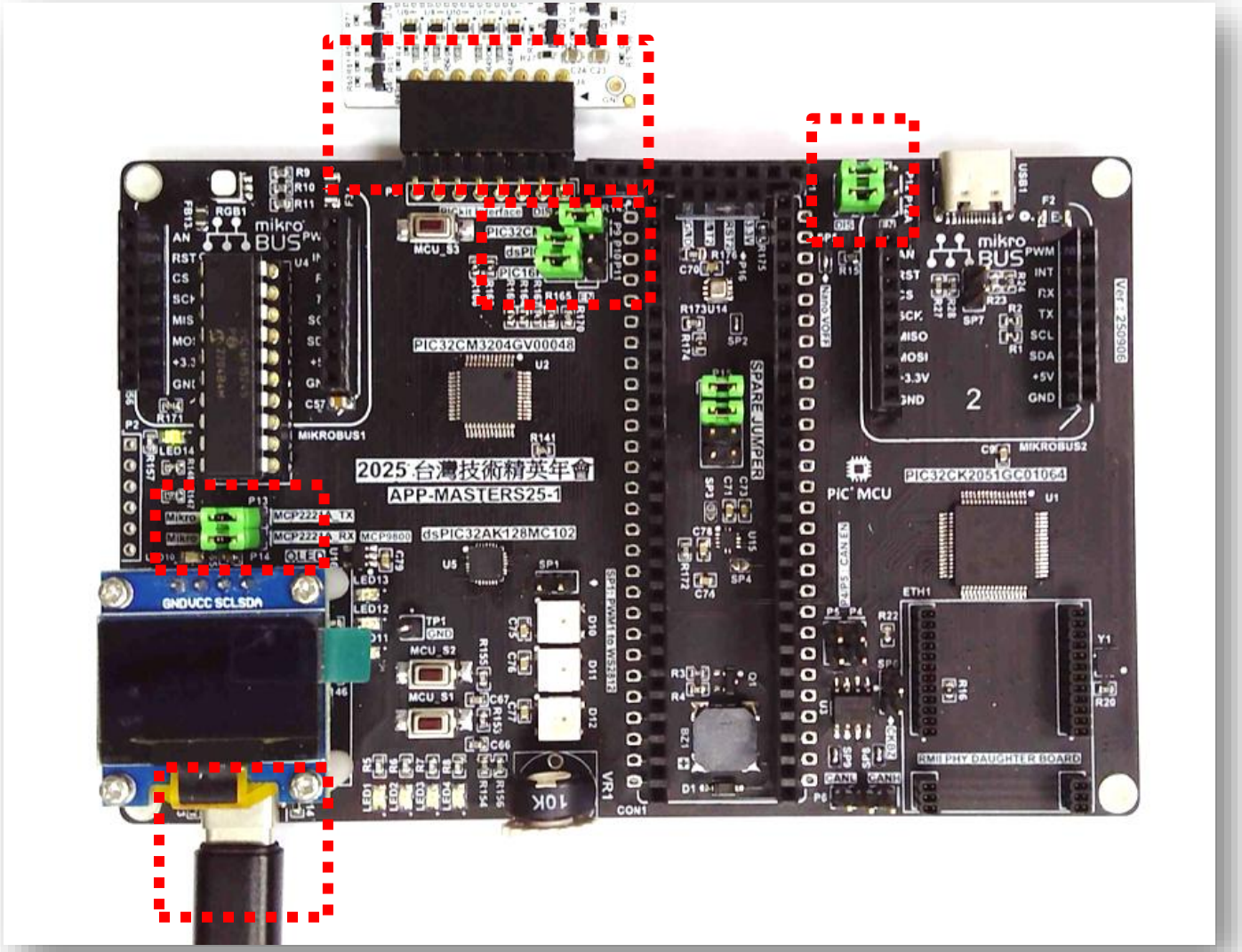
SET UP CLOCKS USING MCC HARMONY V3

設定 SysTick 來獲得一個 100ms 的時間來源並使用其中斷來驅動 LED1

練習燒錄成是並且觀察實際執行結果與相關的檔案

Lab1 – MPLAB® X IDE project creation & Systick

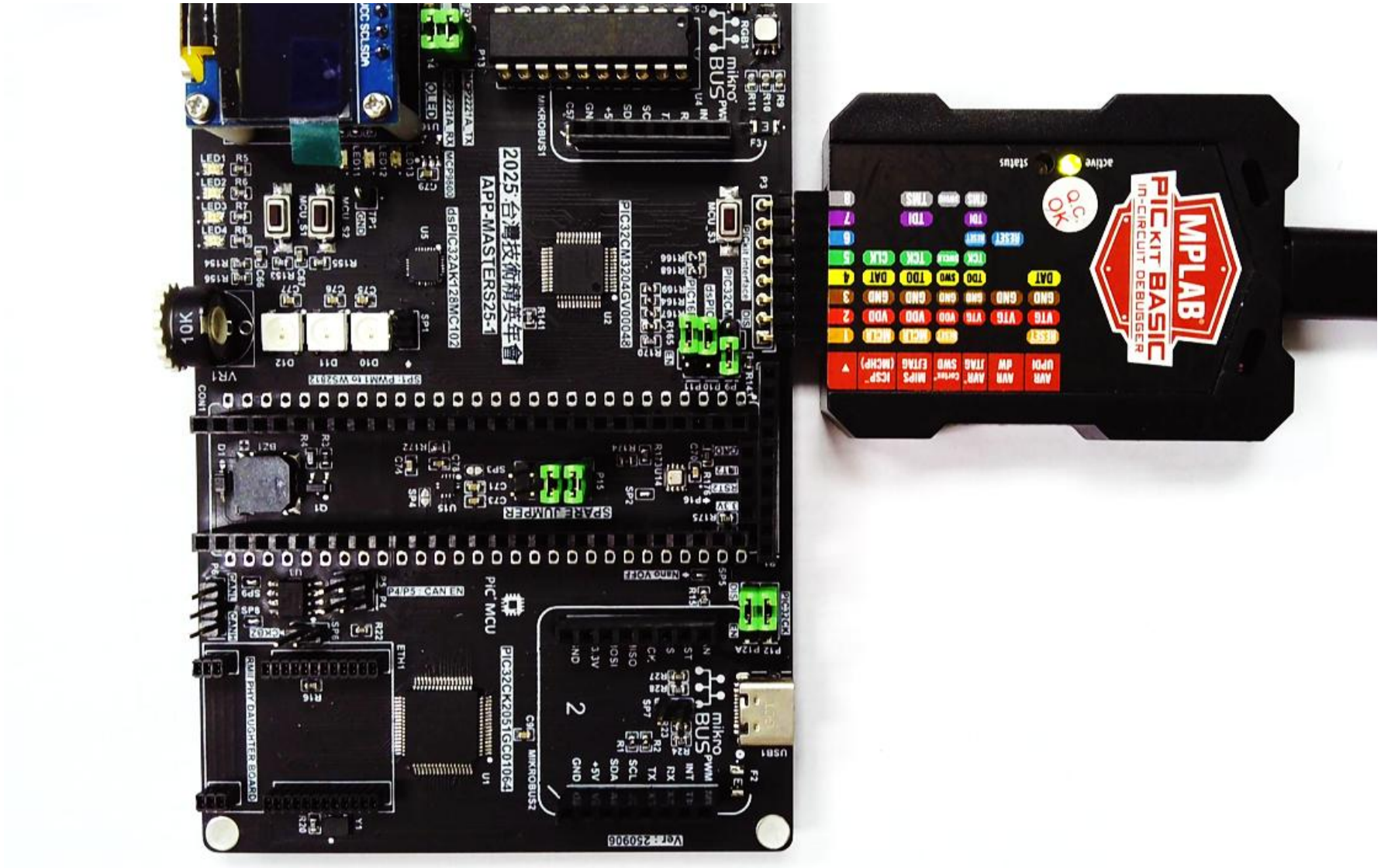
APP-MASTERS25-1 實驗板的 Jumper 設定以及燒錄器連接方式



- **USB2 Connector 以 Type-C Cable 連接到電腦**
- **P3 : 連結到 Programmer**
  - SNAP, PICkit-Basic, PICkit-5
- **P9 : 插接 1&2 來 Enable PIC32CM3204GV00048**
- **P10, P11, P12, P12A 都插接在 2&3 的位置來 DISABLE 其它的 MCU**
- **P13 & P14 要 Close，MCP2221A 才能獲得 MCU 的 UART 信號**

Lab1 – MPLAB® X IDE project creation & Systick

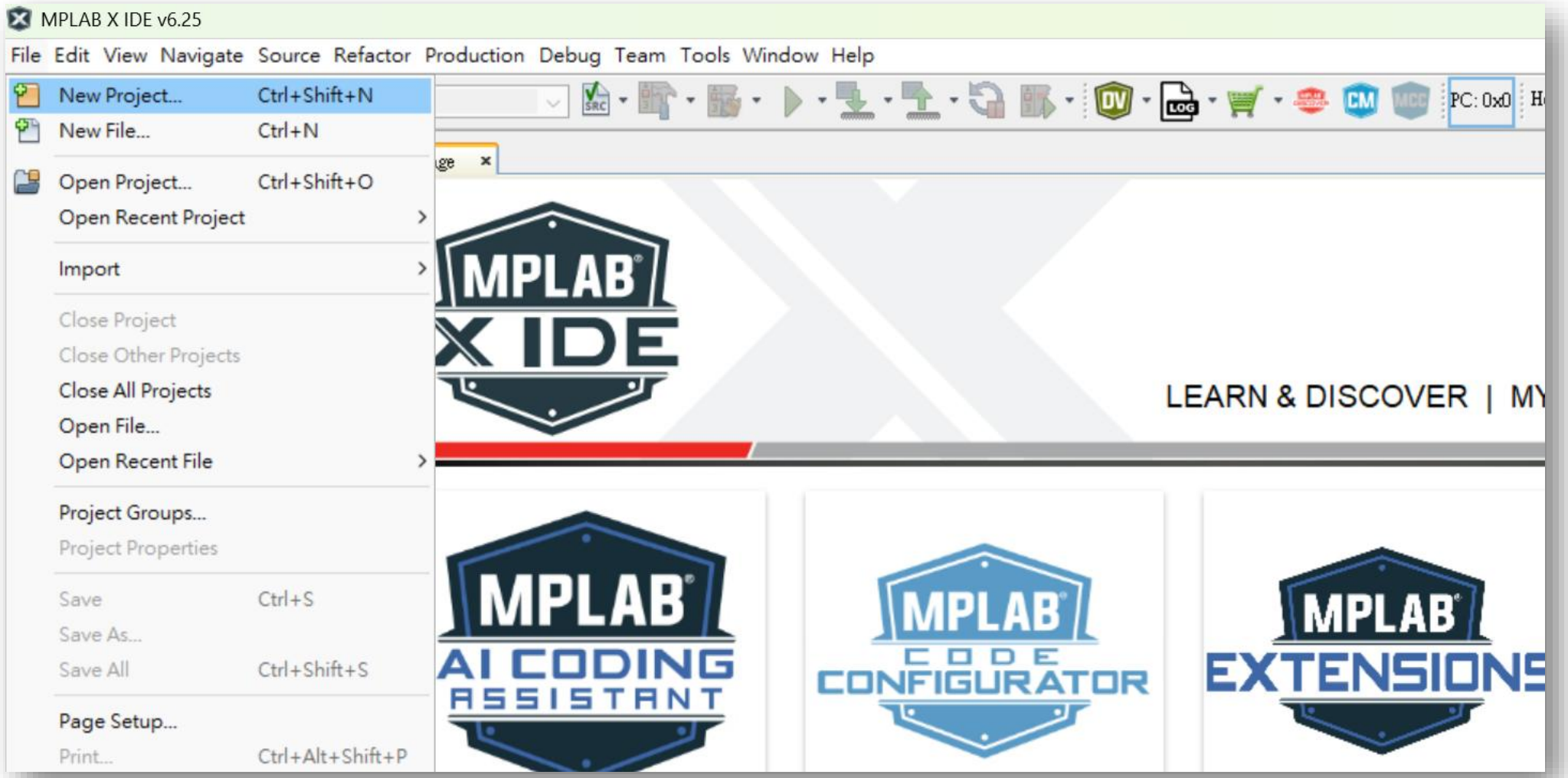
如果您有自己購買的 PICkit BASIC or PICkit-5 也能使用



MICROCHIP

Lab1 – MPLAB® X IDE project creation & Systick

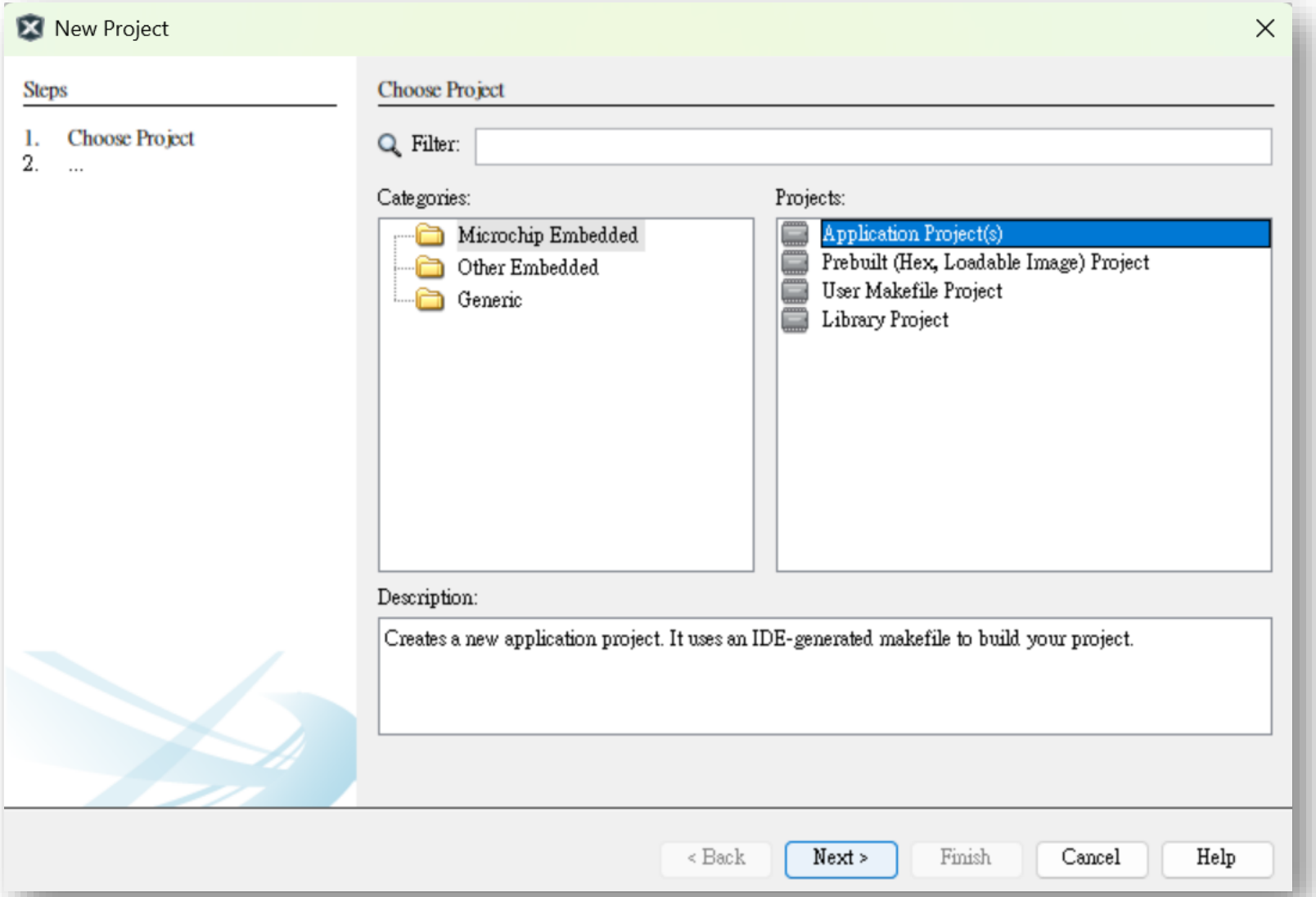
File -> New Project



MICROCHIP

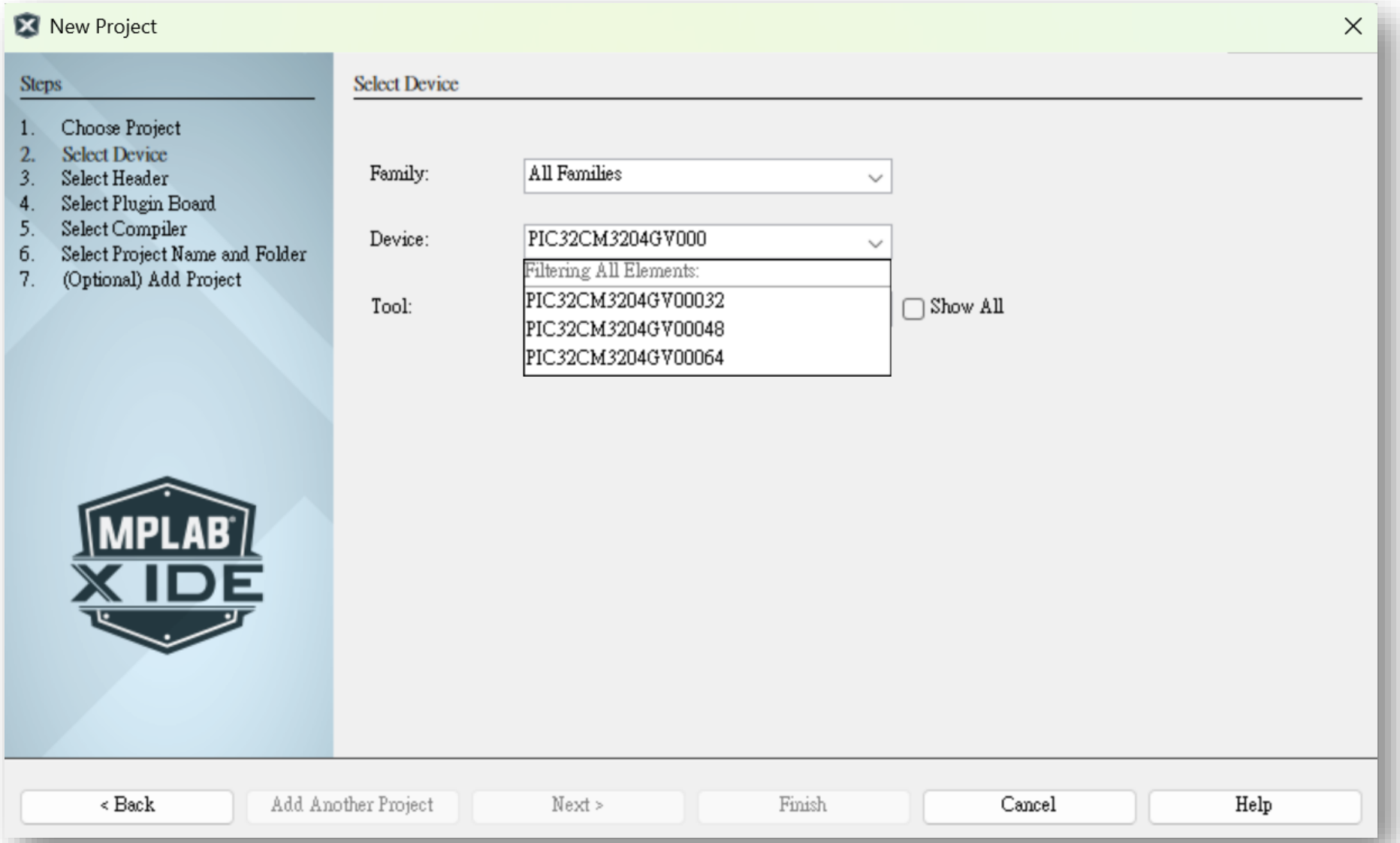
Lab1 – MPLAB® X IDE project creation & Systick

Select “Application Project(s)



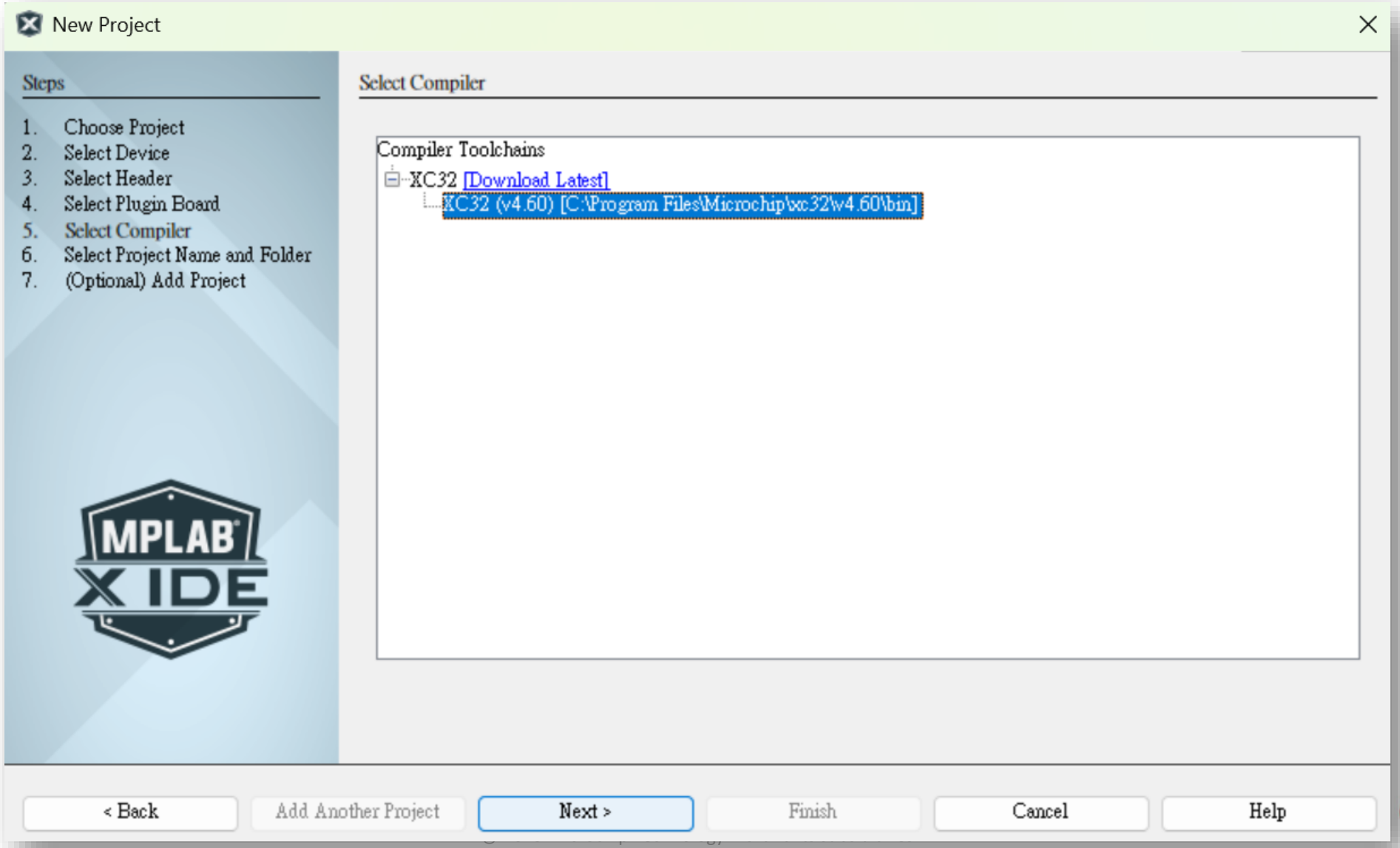
Lab1 – MPLAB® X IDE project creation & Systick

Select Device as PIC32CM3204GV00048 – You can select Tool later ..



Lab1 – MPLAB® X IDE project creation & Systick

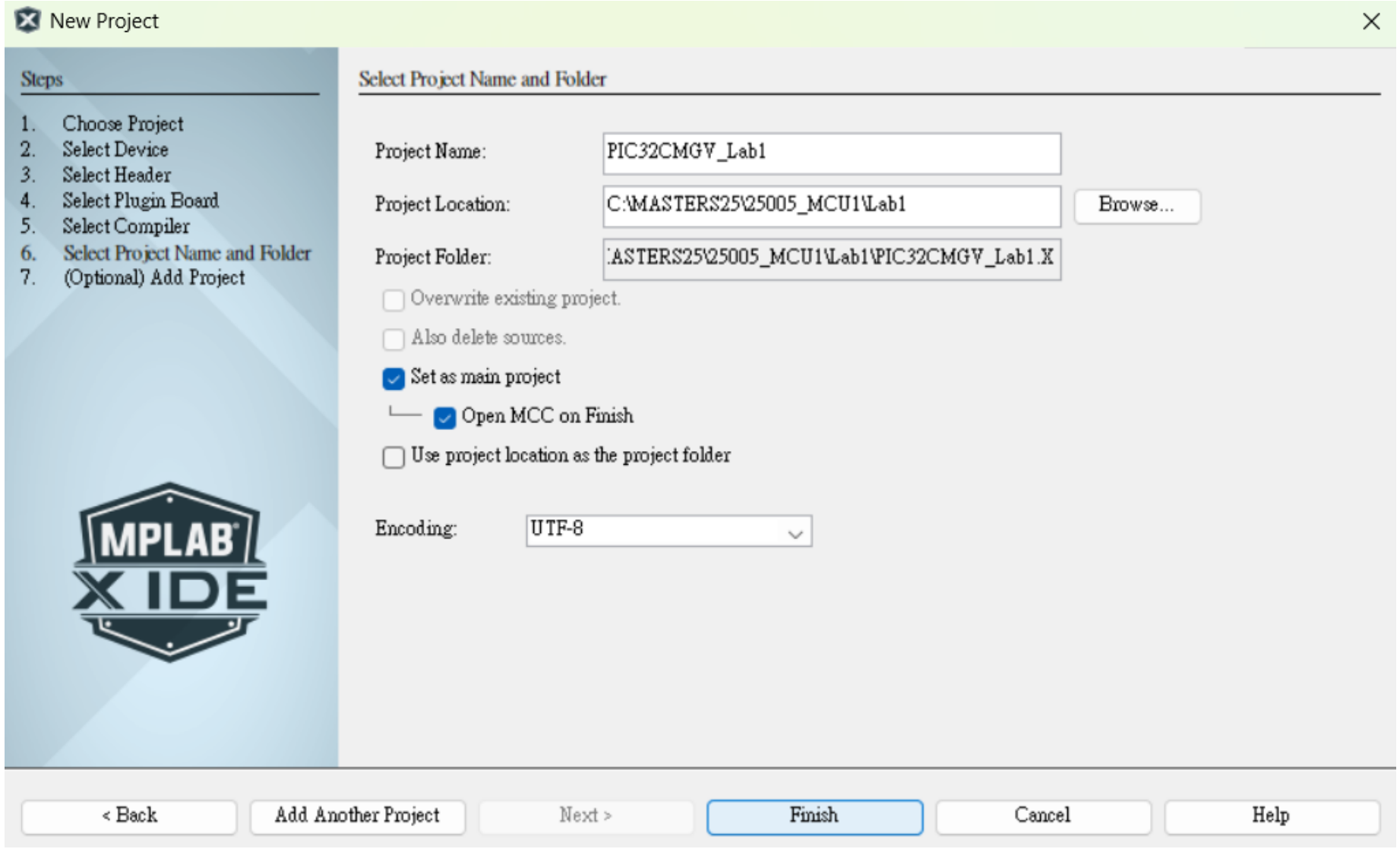
Select Compiler Toolchains – 使用 XC32 v4.60 版



MICROCHIP

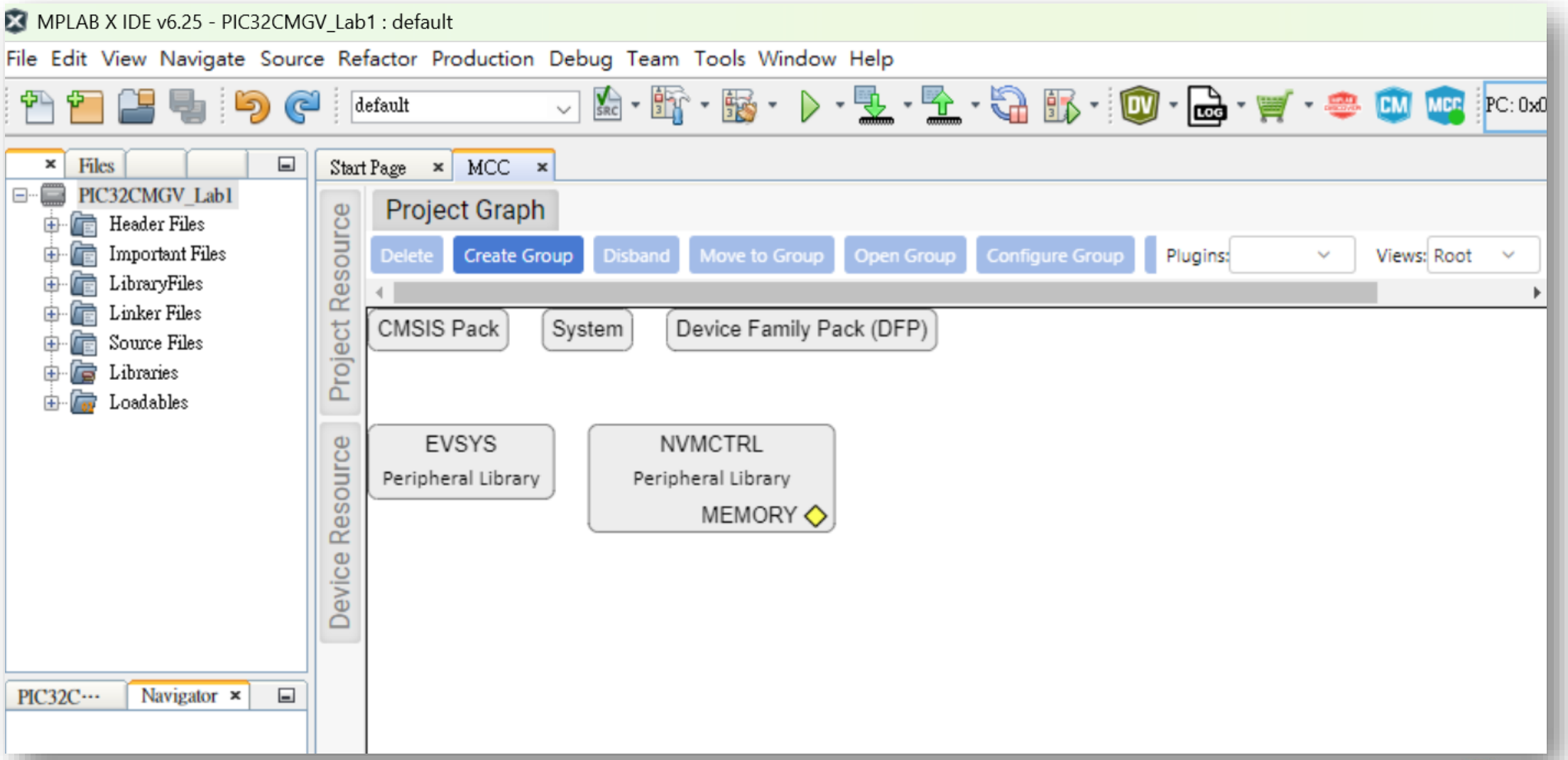
Lab1 – MPLAB® X IDE project creation & Systick

確定 Project Location & Name, 並且將編碼方式設定為 UTF-8 以便正確顯示中文



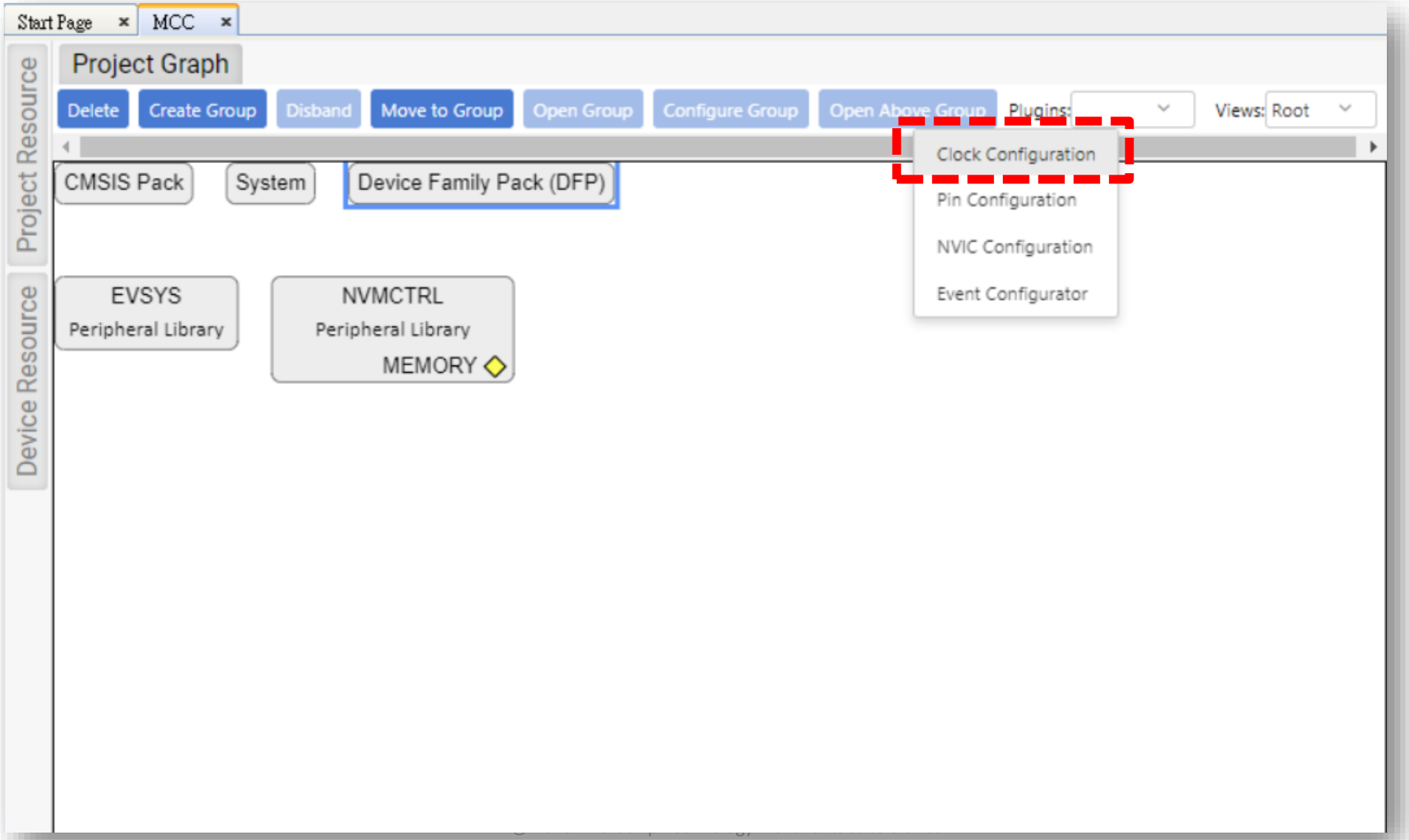
Lab1 – MPLAB® X IDE project creation & Systick

按下 Finish 之後 MCC 會被自動開啟並下載必要的軟體元件



Lab1 – MPLAB® X IDE project creation & Systick

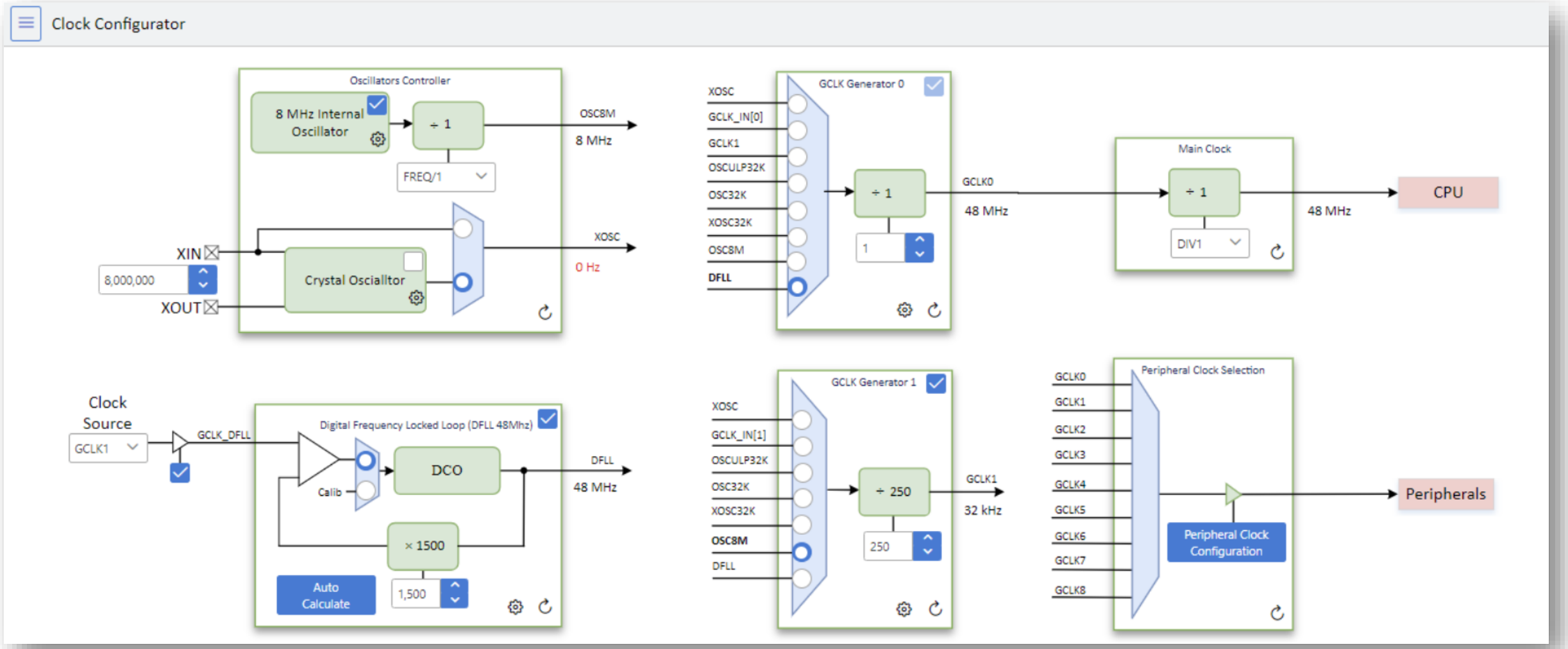
使用 “Clock Configuration” Plugin 來觀察及設定 Clock



MICROCHIP

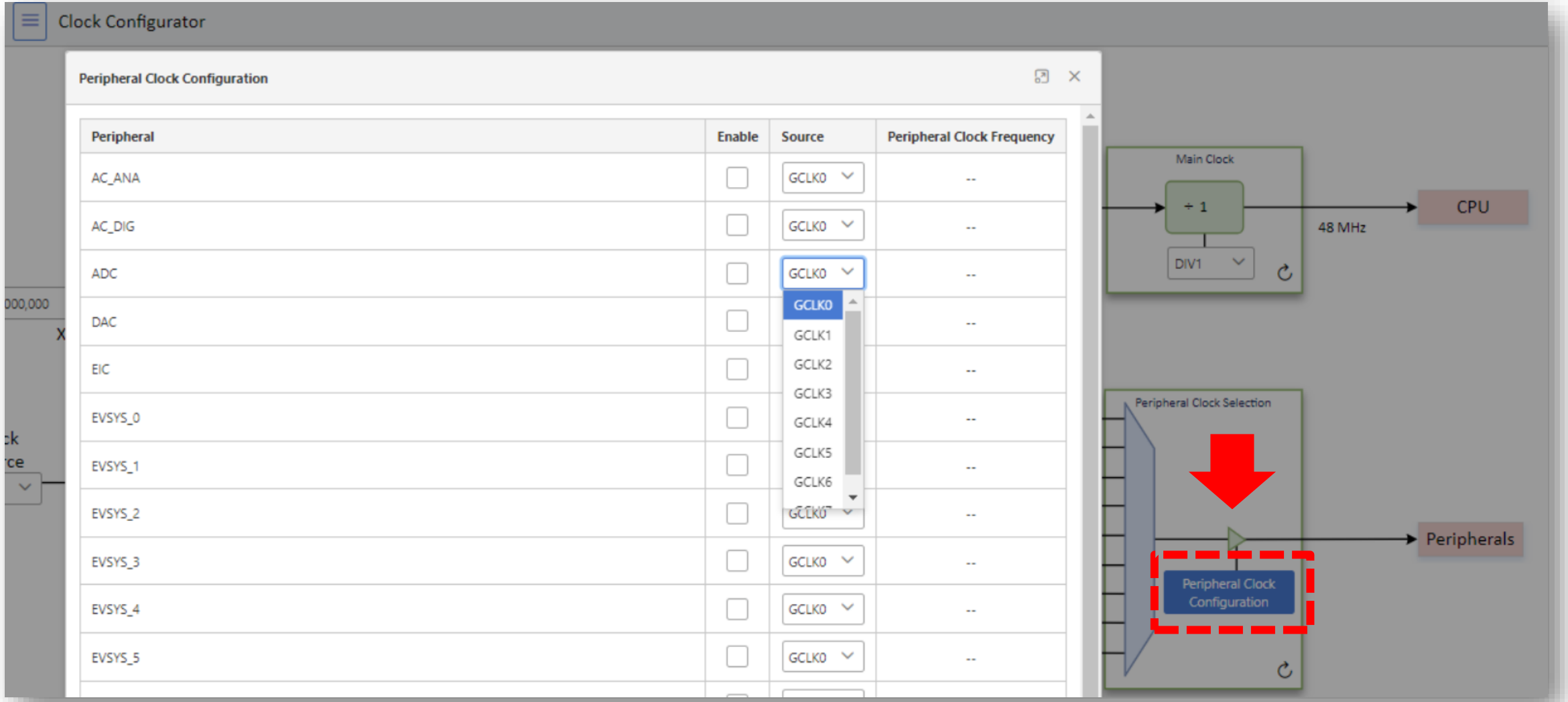
Lab1 – MPLAB® X IDE project creation & Systick

預設的 CPU Clock 已經是 48 Mhz，如有必要可議根據需求再調整



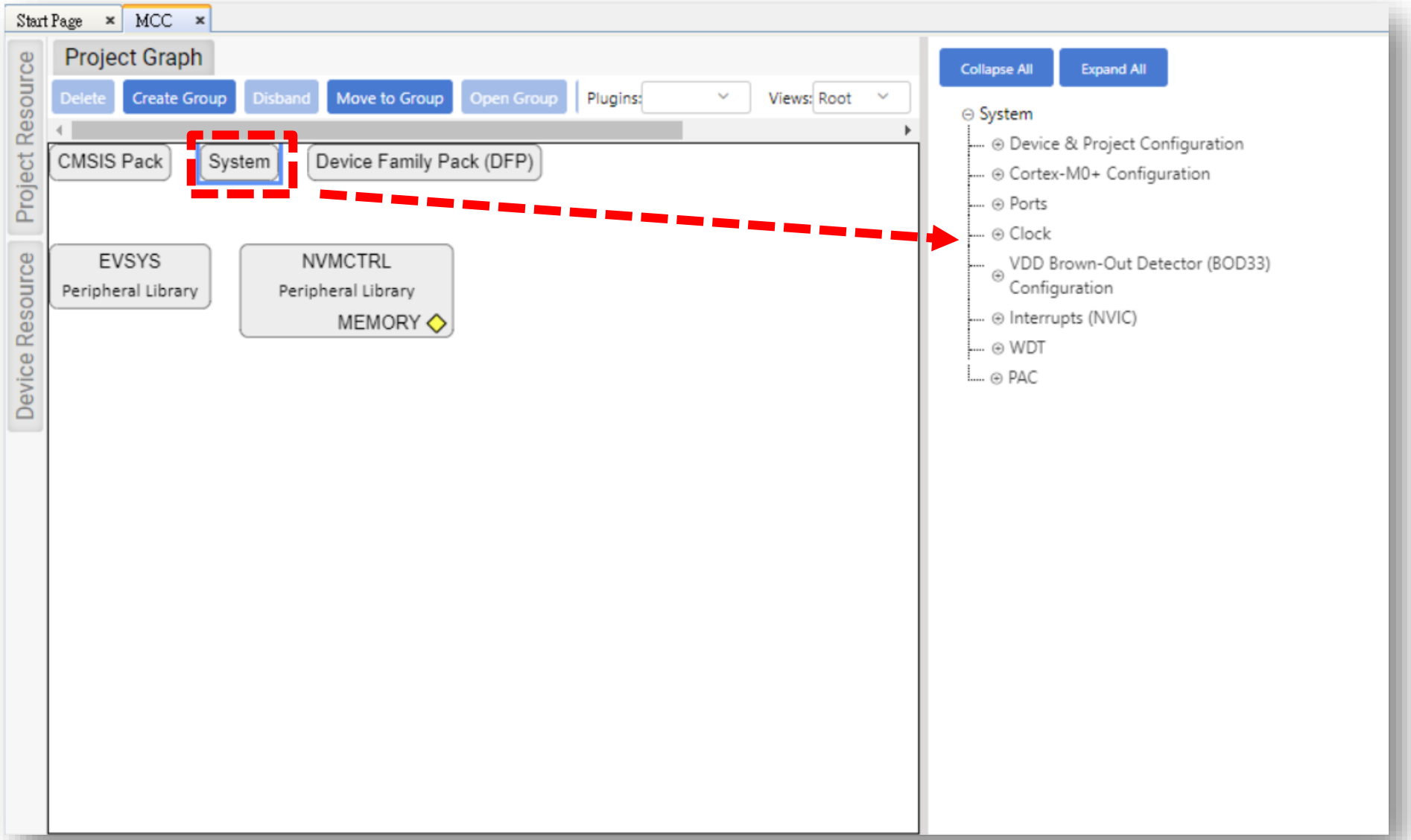
Lab1 – MPLAB® X IDE project creation & Systick

個別的周邊可以選用不同有被 Activated 的 Clock



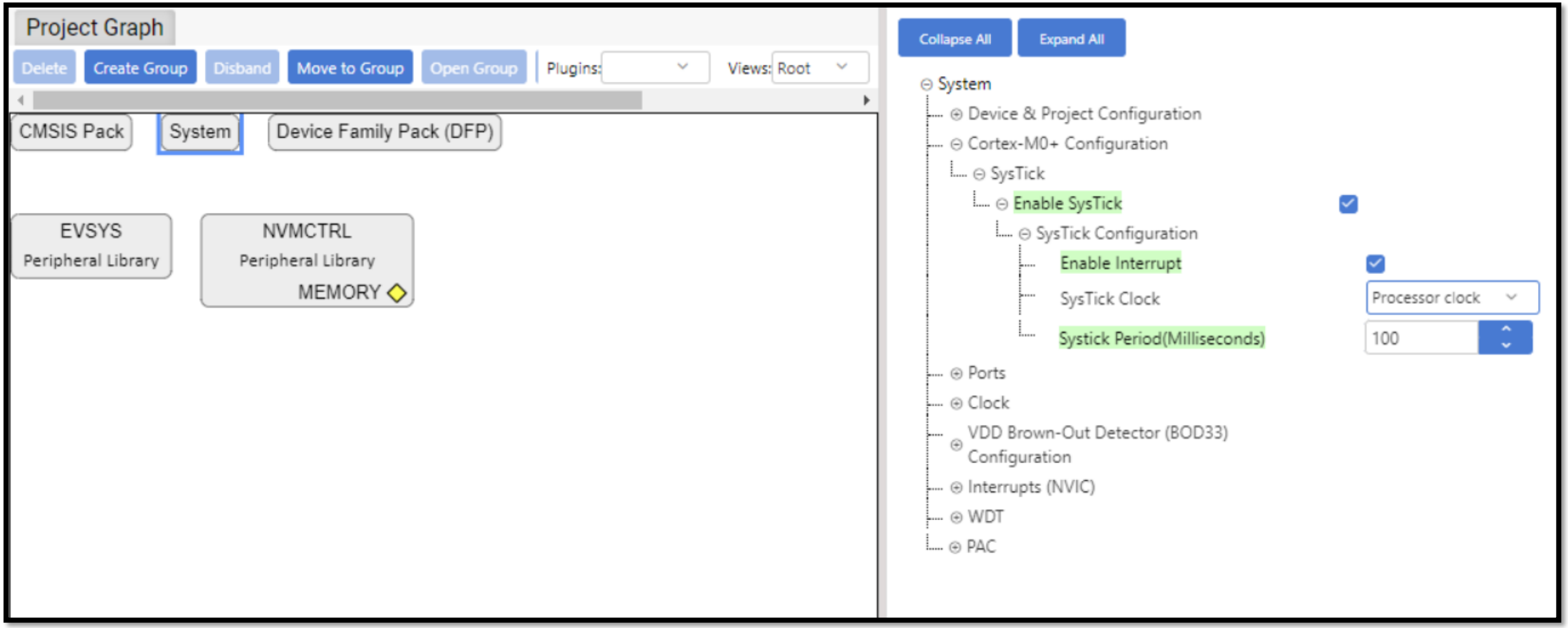
Lab1 – MPLAB® X IDE project creation & Systick

點選 System 來設定 SysTick 以及觀察其他重要選項



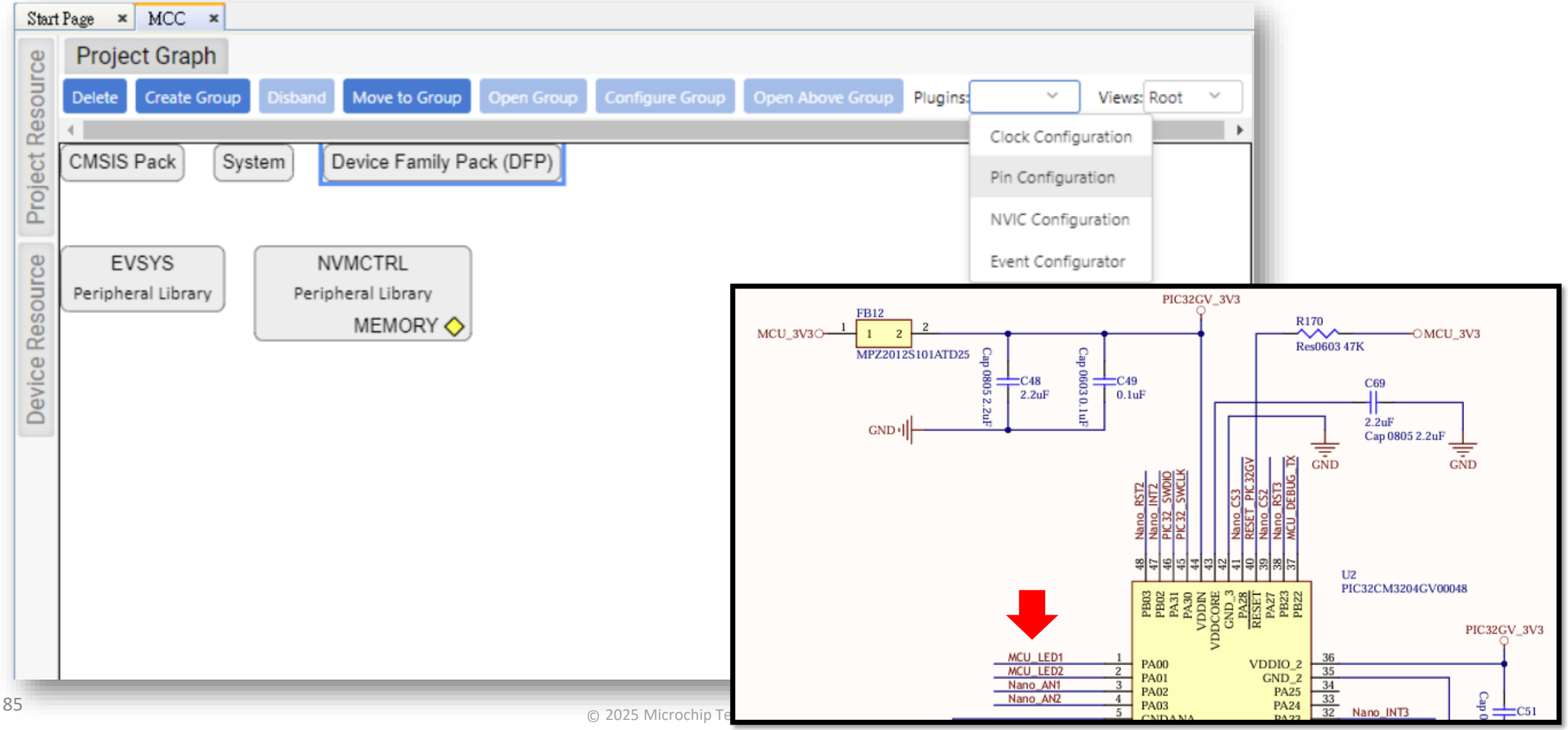
Lab1 – MPLAB® X IDE project creation & Systick

Enable SysTick 、啟用中斷並將 Systick Period 設定為 100 ms



Lab1 – MPLAB® X IDE project creation & Systick

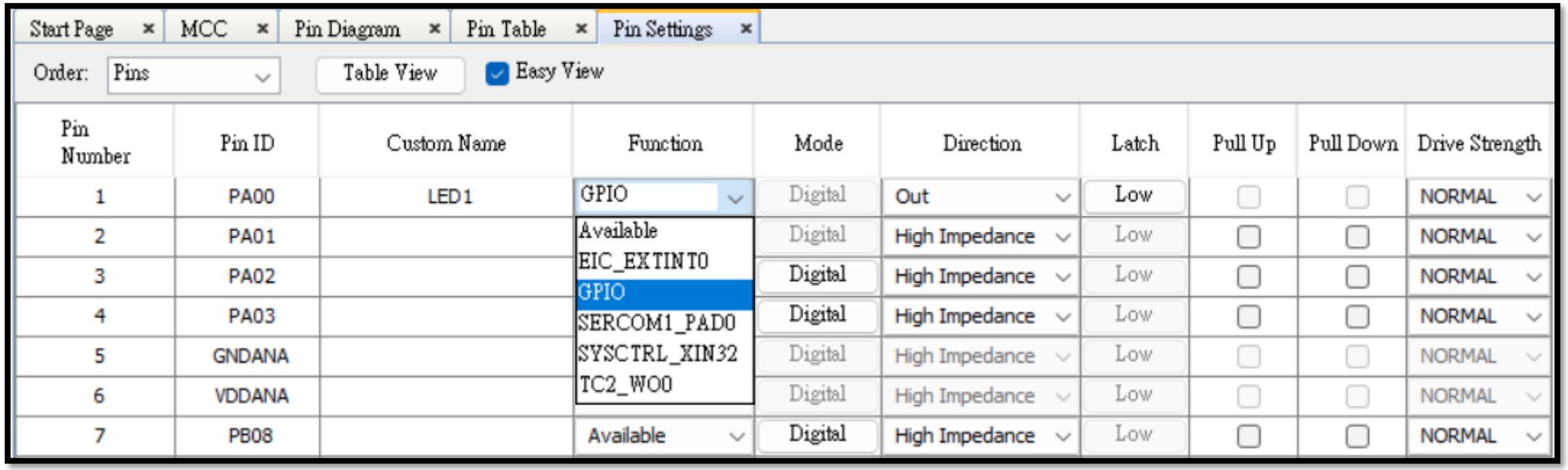
I/O Pin 的設定：選擇 Pin Configuration 這個 Plugin (LED1 @PA00)



Lab1 – MPLAB® X IDE project creation & Systick

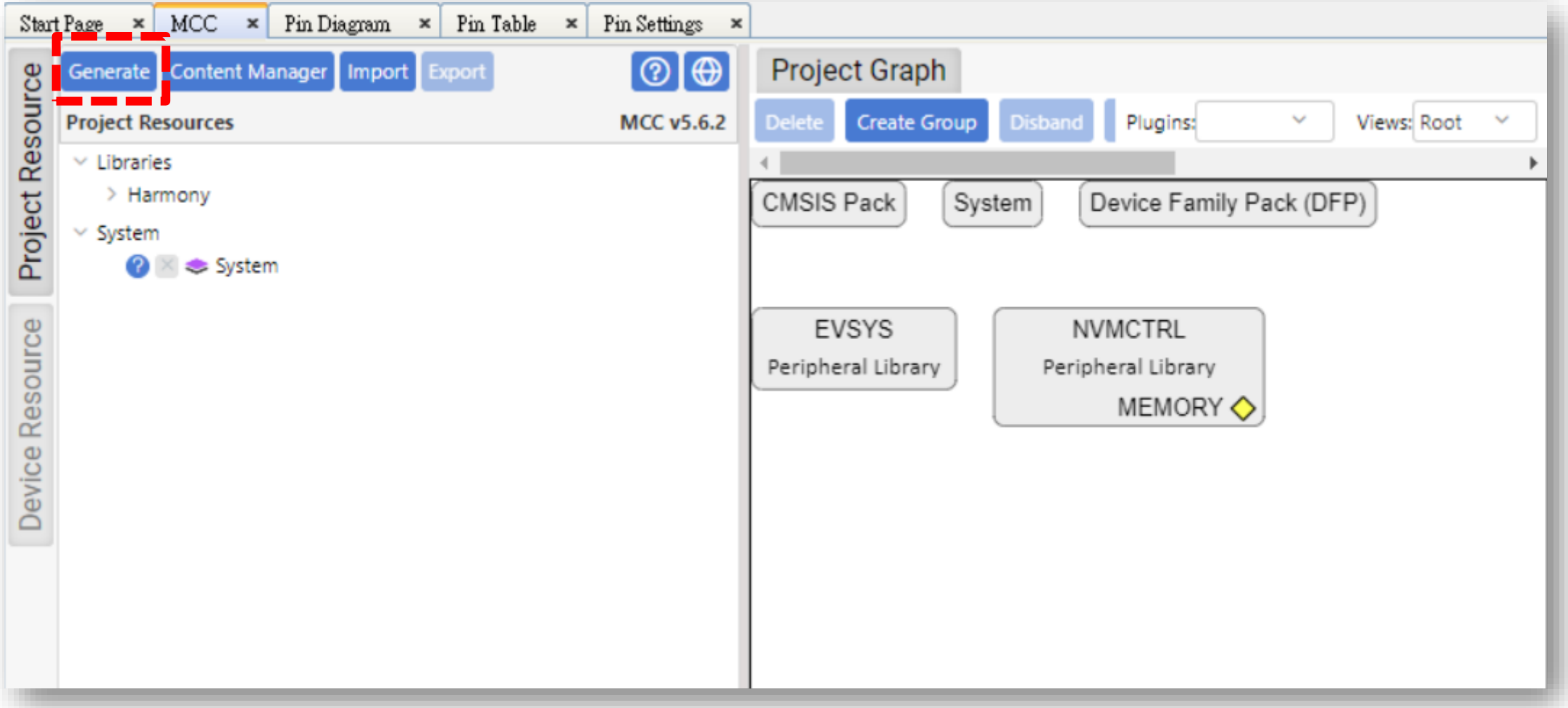
將 PA00 設定為可以做 Out 的 GPIO

- **Function**
- **Direction**
- **Custom Name (MCC 在產生 PLIB API 時會使用此名稱)**



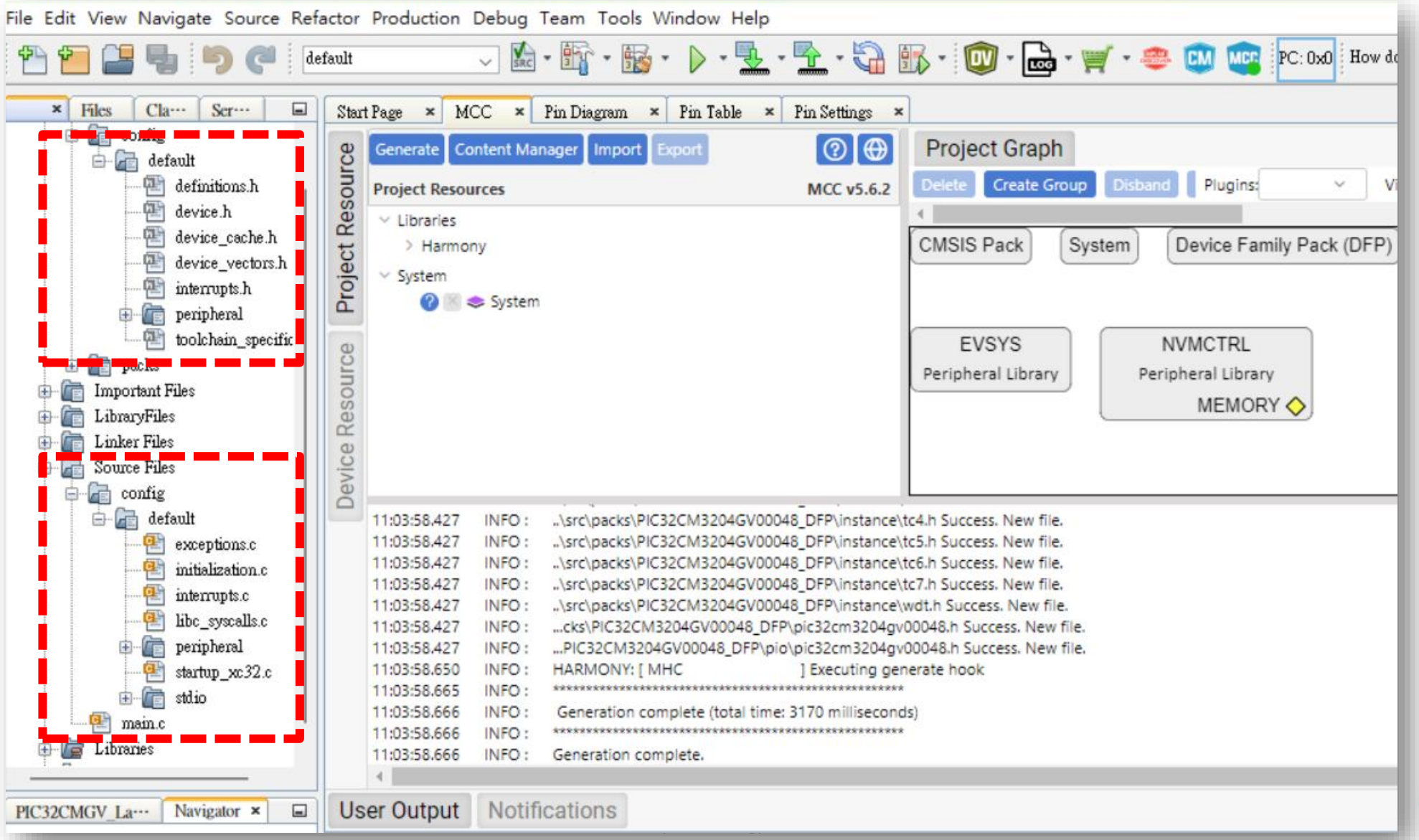
Lab1 – MPLAB® X IDE project creation & Systick

回到 MCC 頁面，點選 Project Resource 中的 Generate



Lab1 – MPLAB® X IDE project creation & Systick

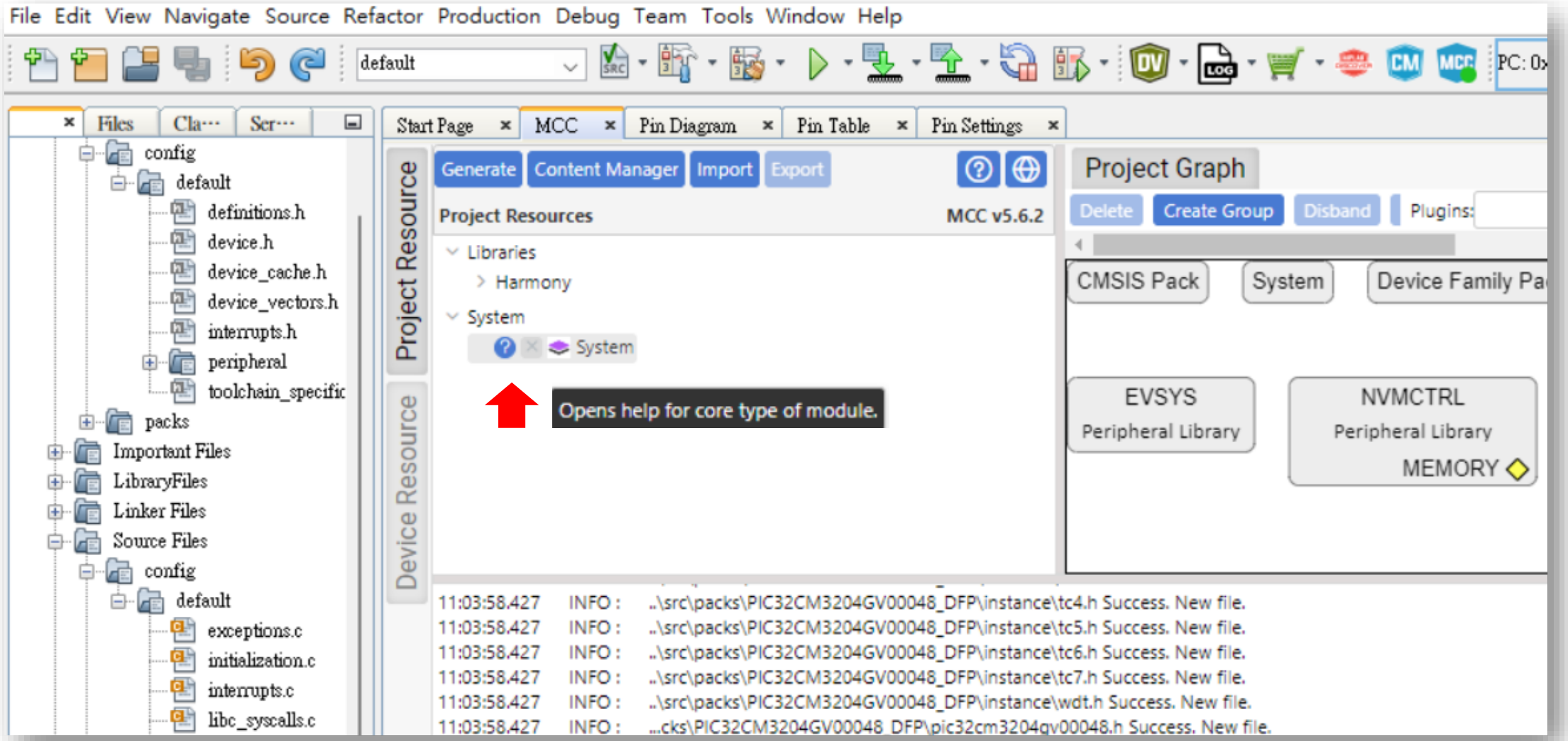
User Output 視窗會輸出 Generate Code 的過程並且產生必要的檔案



MICROCHIP

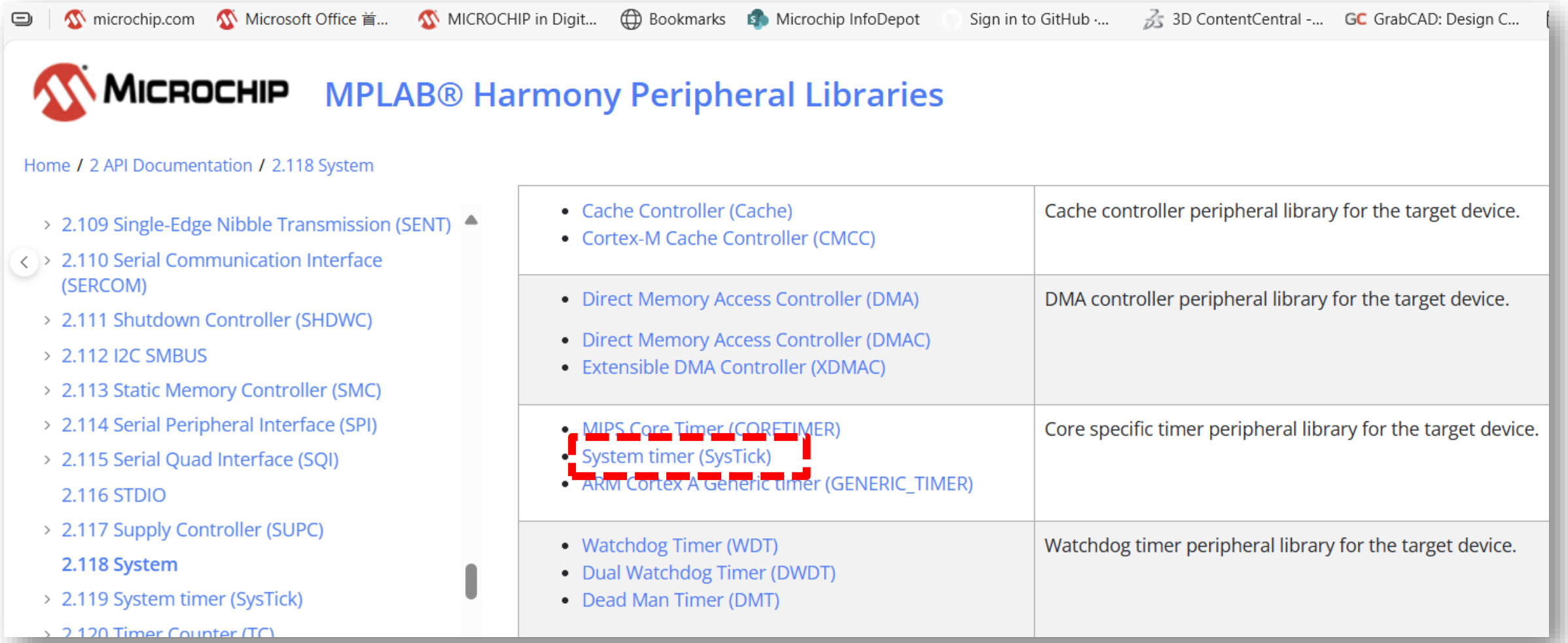
Lab1 – MPLAB® X IDE project creation & Systick

點選 System 模組左邊的 ❓ 即可連結到線上說明



Lab1 – MPLAB® X IDE project creation & Systick

點選 System timer 項目即可找到 SysTick 的應用說明



Lab1 – MPLAB® X IDE project creation & Systick

Lab1 希望使用中斷，所以參考 callback method

Callback method

This example demonstrates how to use SysTick to generate periodic callback.

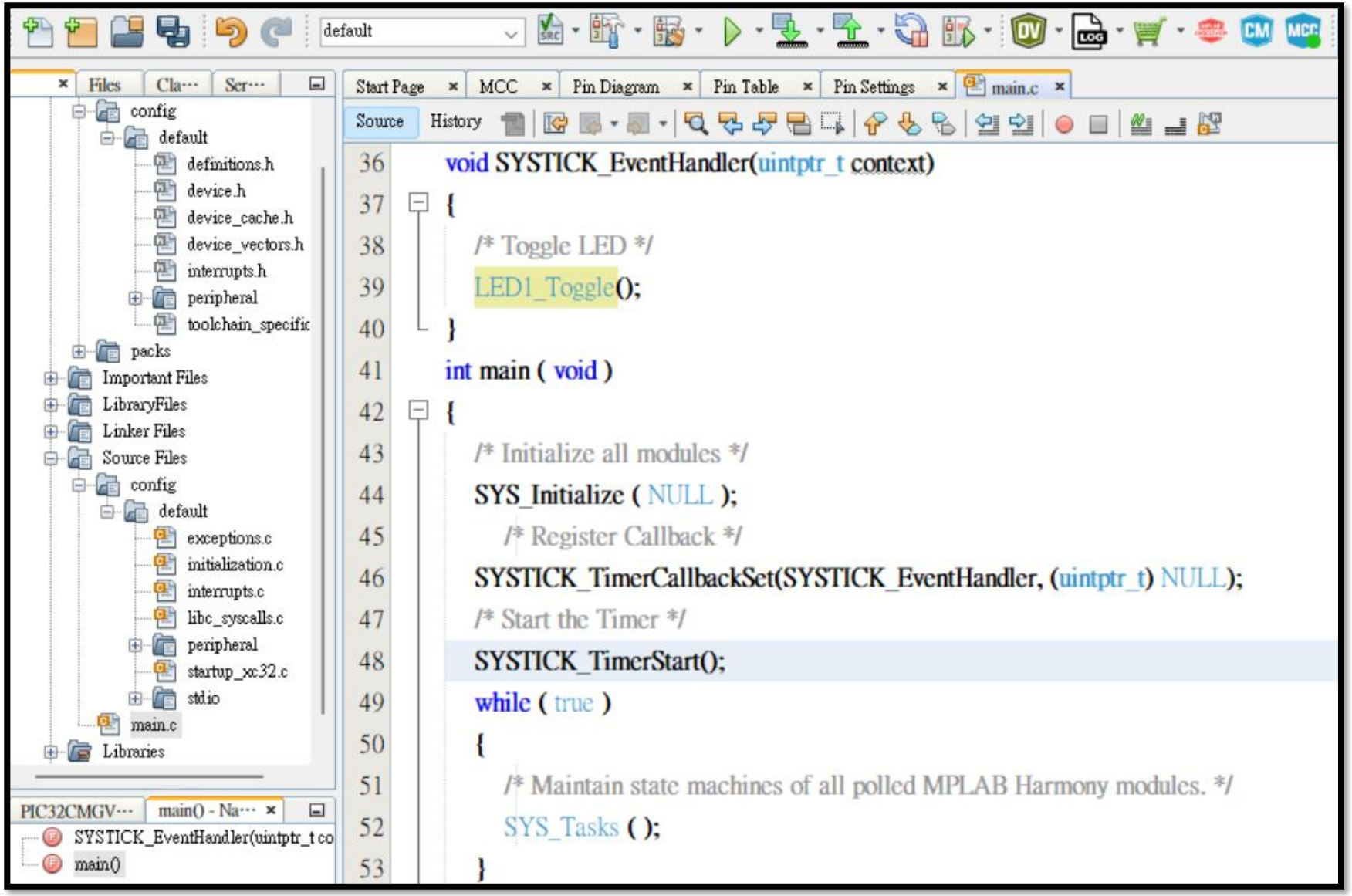
Copy

```
/* This function is called after Timer expires */
void SYSTICK_EventHandler(uintptr_t context)
{
    /* Toggle LED */
    LED_Toggle();
}

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

    /* Register Callback */
    SYSTICK_TimerCallbackSet(SYSTICK_EventHandler, (uintptr_t) NULL);

    /* Start the Timer */
    SYSTICK_TimerStart();
}
```


MICROCHIP

Lab1 – MPLAB® X IDE project creation & Systick

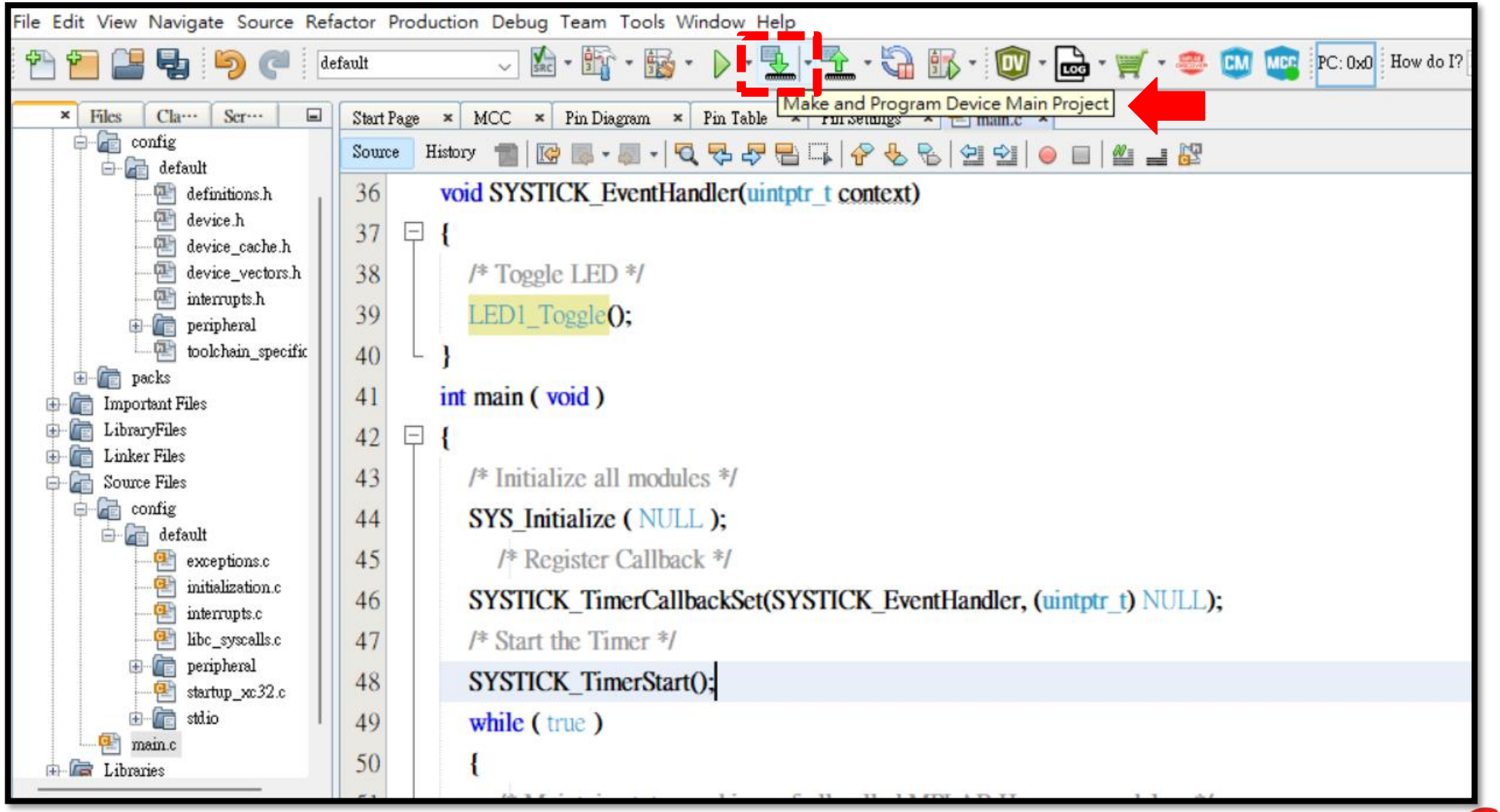
修改後的 main.c -> 請注意 LED1 的 Custom Name 已被用上



MICROCHIP

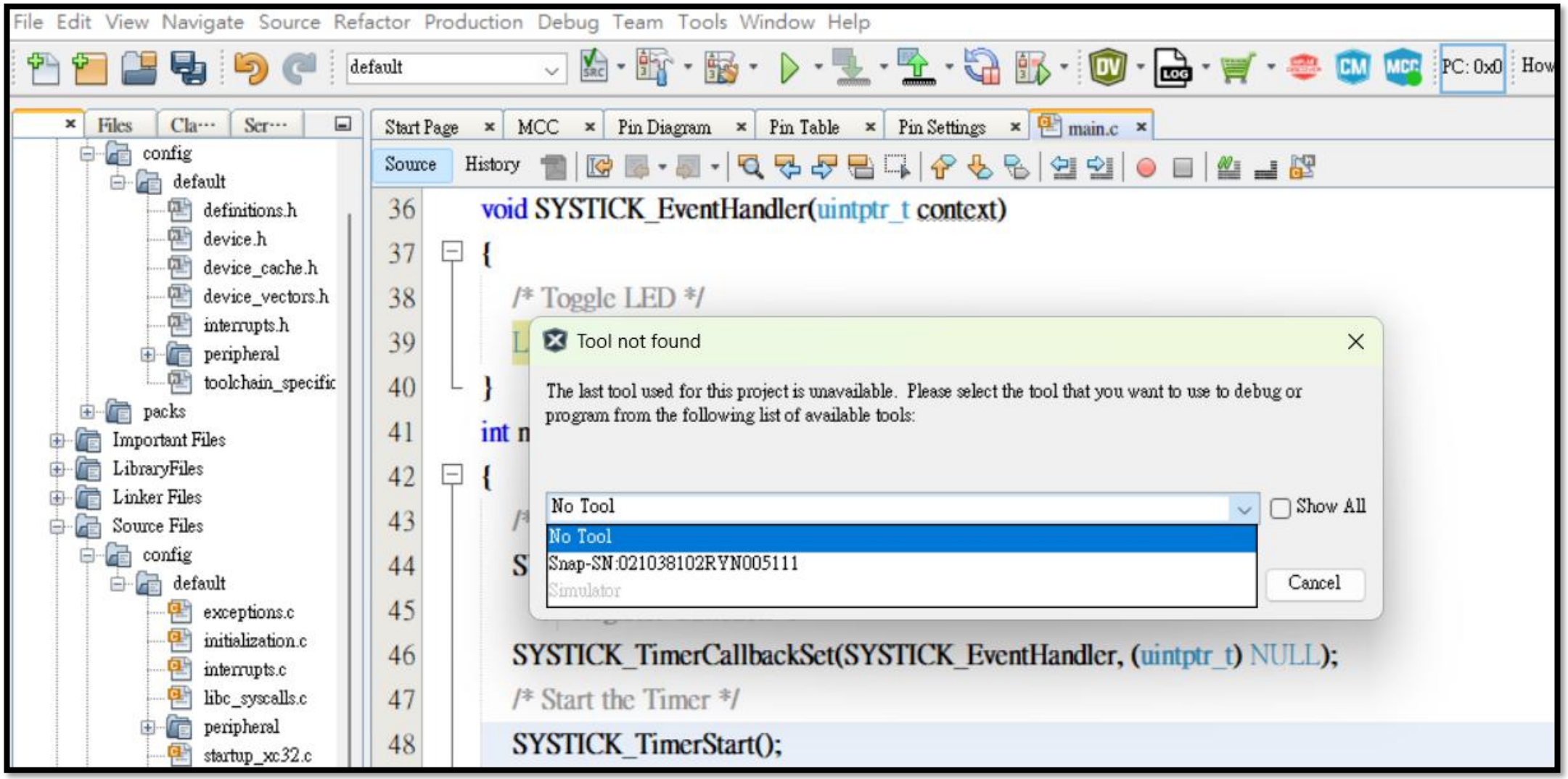
Lab1 – MPLAB® X IDE project creation & Systick

點選 Make and Program Device Main Project 來進行燒錄作業



Lab1 – MPLAB® X IDE project creation & Systick

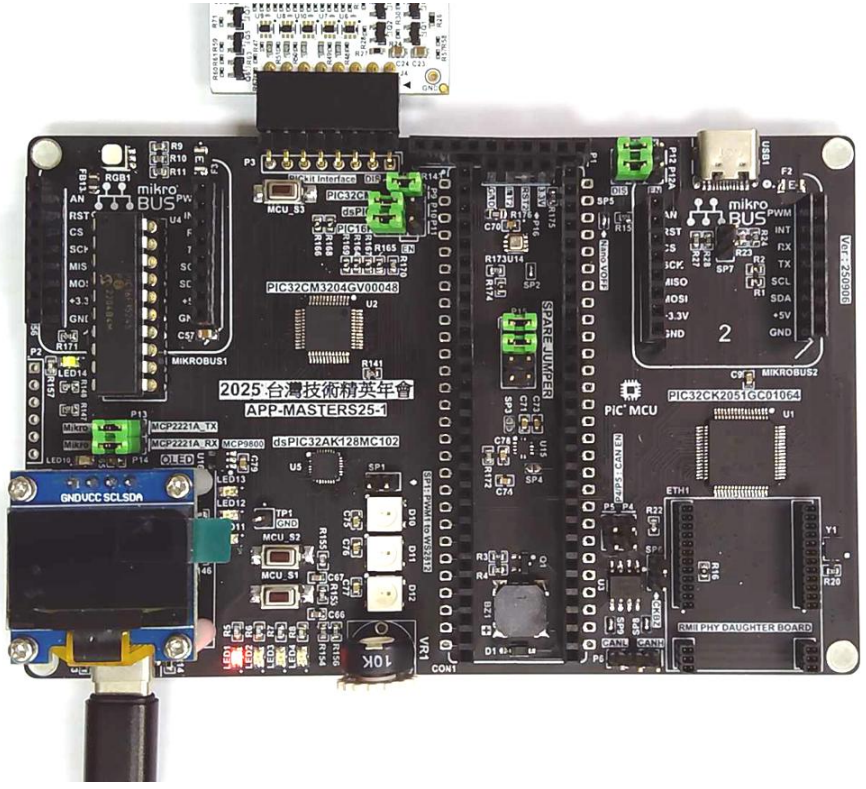
如果連接的 Tool 之前未被辨識過，MPLAB XIDE 會讓適用者確認並選擇



Lab1 – MPLAB® X IDE project creation & Systick

燒錄完成後的執行結果 – LED1 每 100 ms 轉態一次

```
Output ×
Scripting × | MPLAB® Code Configurator × | PIC32CMGV_Lab1 (Build, Load, ...) × | Snap × | Snap-PIC32CMGV_Lab1 ×
The following memory area(s) will be programmed:
program memory: start address = 0x0, end address = 0x4ff
configuration memory
User Id Memory

Due to the large memory ranges on this device, only the areas of memory that have been loaded with code (via the build process
Programming/Verify complete
```



MICROCHIP

Lab 1 Summary

Successfully created a MPLAB® X IDE project using MCC Harmony v3

Check and Config Clock – Make sure CPU runs at 48 Mhz

Config SysTick and learn how to implement “Interrupt”

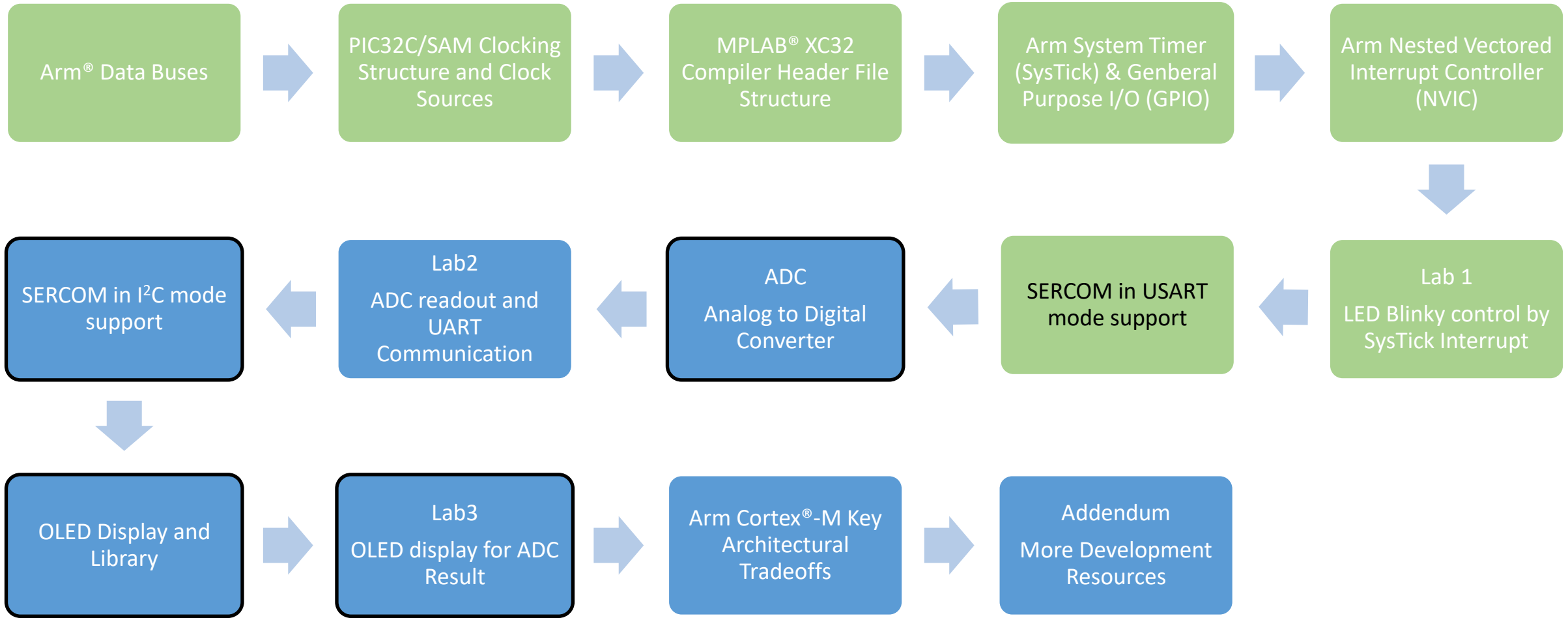
Config I/O Pins and use Custom Name for easy access

Program the device by using Microchip Tool

Viewed the main device and component header files

MICROCHIP

Class Outline



SERCOM in USART mode

Serial Communication Interface

SERCOM – Serial Communication Interface

Overview

- **Features**
  - Interface for configuring into one of the following:
    - I2C – Two-wire serial interface SMBus™ compatible
    - SPI – Serial peripheral interface
    - USART – Universal synchronous and asynchronous serial receiver and transmitter
  - Single transmit buffer and double receive buffer
  - Baud-rate generator
  - Address match/mask logic
  - Operational in all sleep mode
  - **When an instance of SERCOM is configured and enabled, all of the resources of that SERCOM instance will be dedicated to the selected mode.**

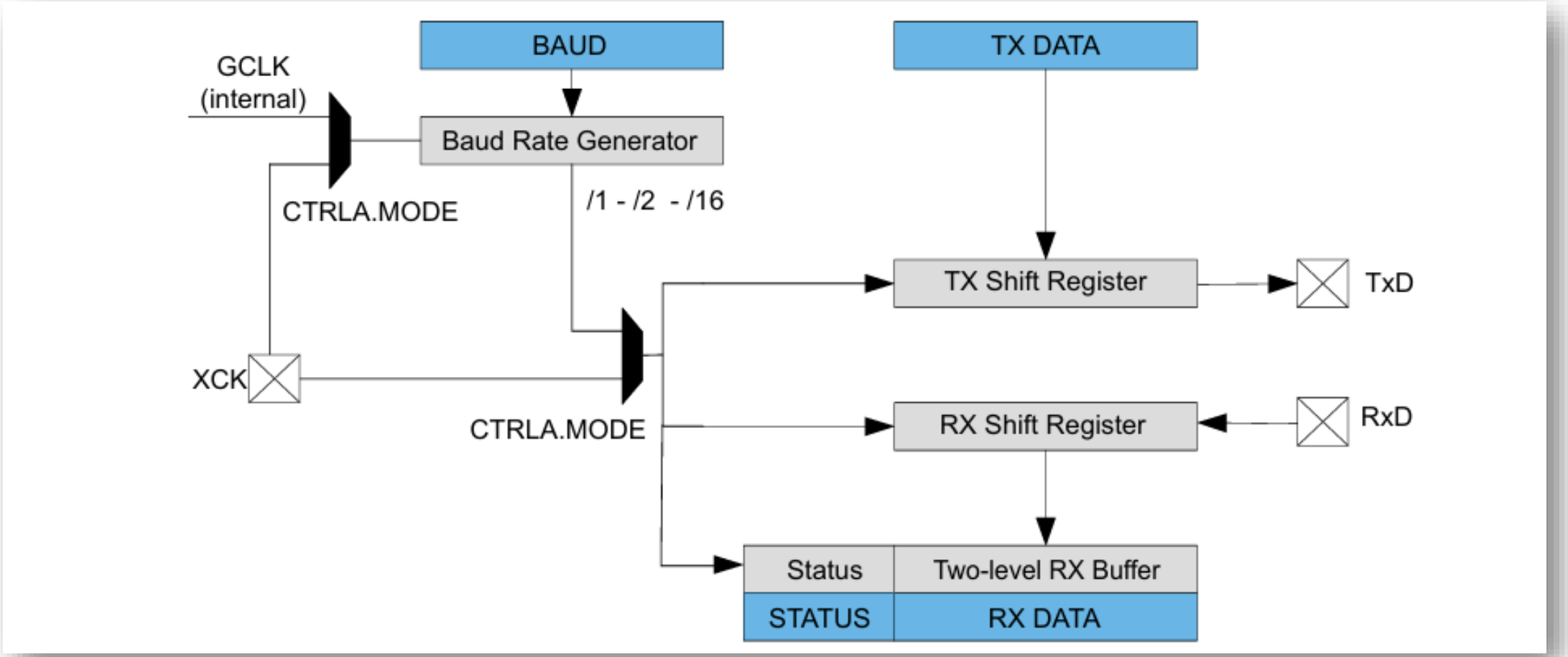
SERCOM - USART mode

SERCOM Synchronous and Asynchronous Receiver and Transmitter

- **USART Features**
  - Full-duplex operation
  - Asynchronous (with clock reconstruction) or synchronous operation
  - Internal or external clock source for asynchronous and synchronous operation
  - Baud-rate generator
  - Supports serial frames with 5, 6, 7, 8 or 9 data bits and 1 or 2 stop bits
  - Odd or even parity generation and parity check
  - Selectable LSB- or MSB-first data transfer
  - Buffer overflow and frame error detection
  - Noise filtering, including false start-bit detection and digital low-pass filter
  - Can operate in all sleep modes
  - Operation at speeds up to half the system clock for internally generated clocks
  - Operation at speeds up to the system clock for externally generated clocks
  - Start-of-frame detection

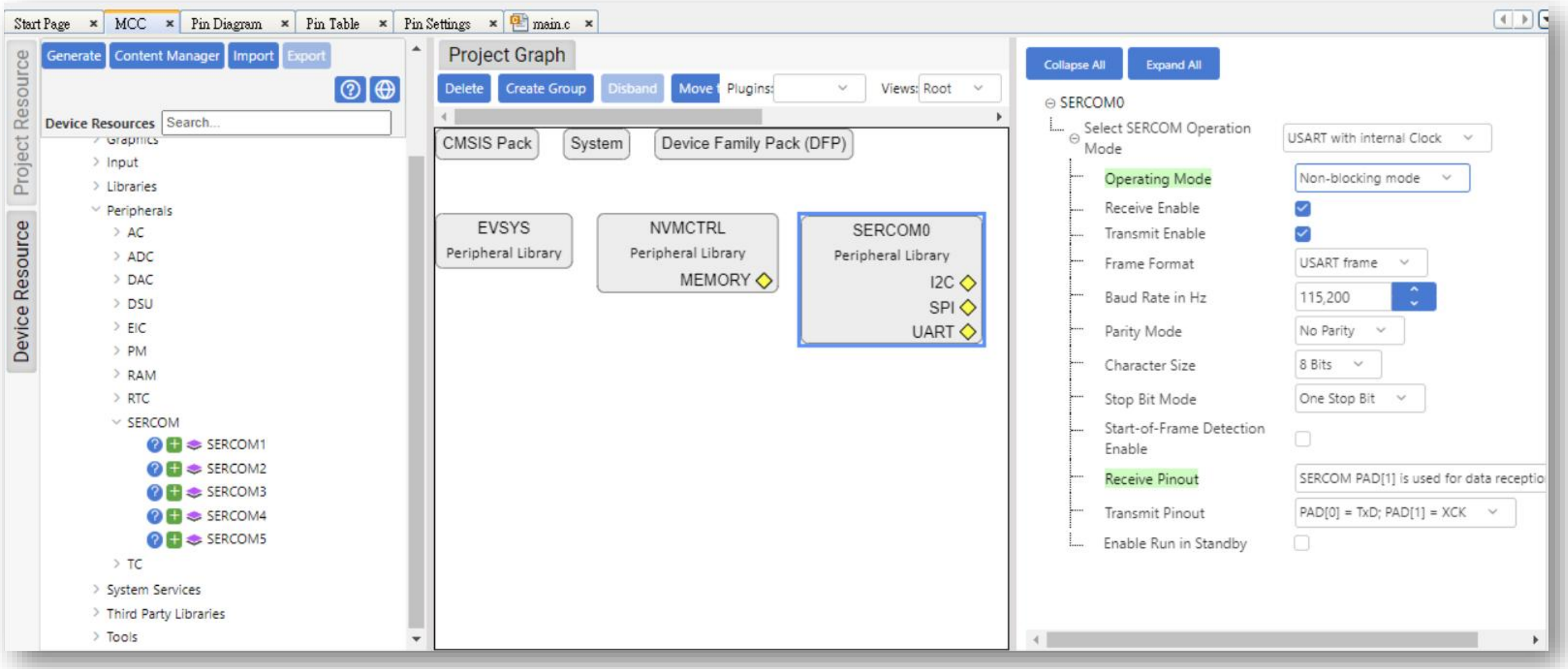
SERCOM - USART mode

SERCOM Synchronous and Asynchronous Receiver and Transmitter



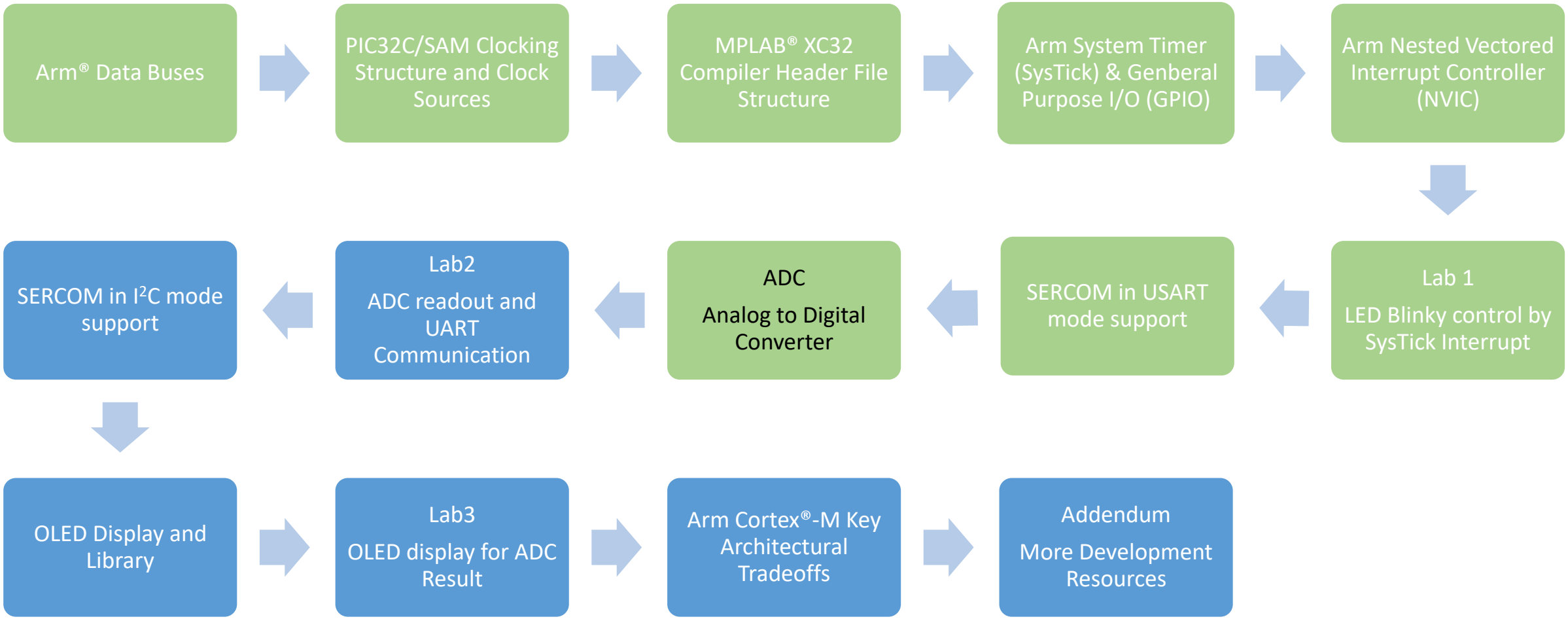
SERCOM - USART mode

規劃 SERCOM 時要特別注意 Transmit & Receive 的腳位



MICROCHIP

Class Outline



MICROCHIP

ADC

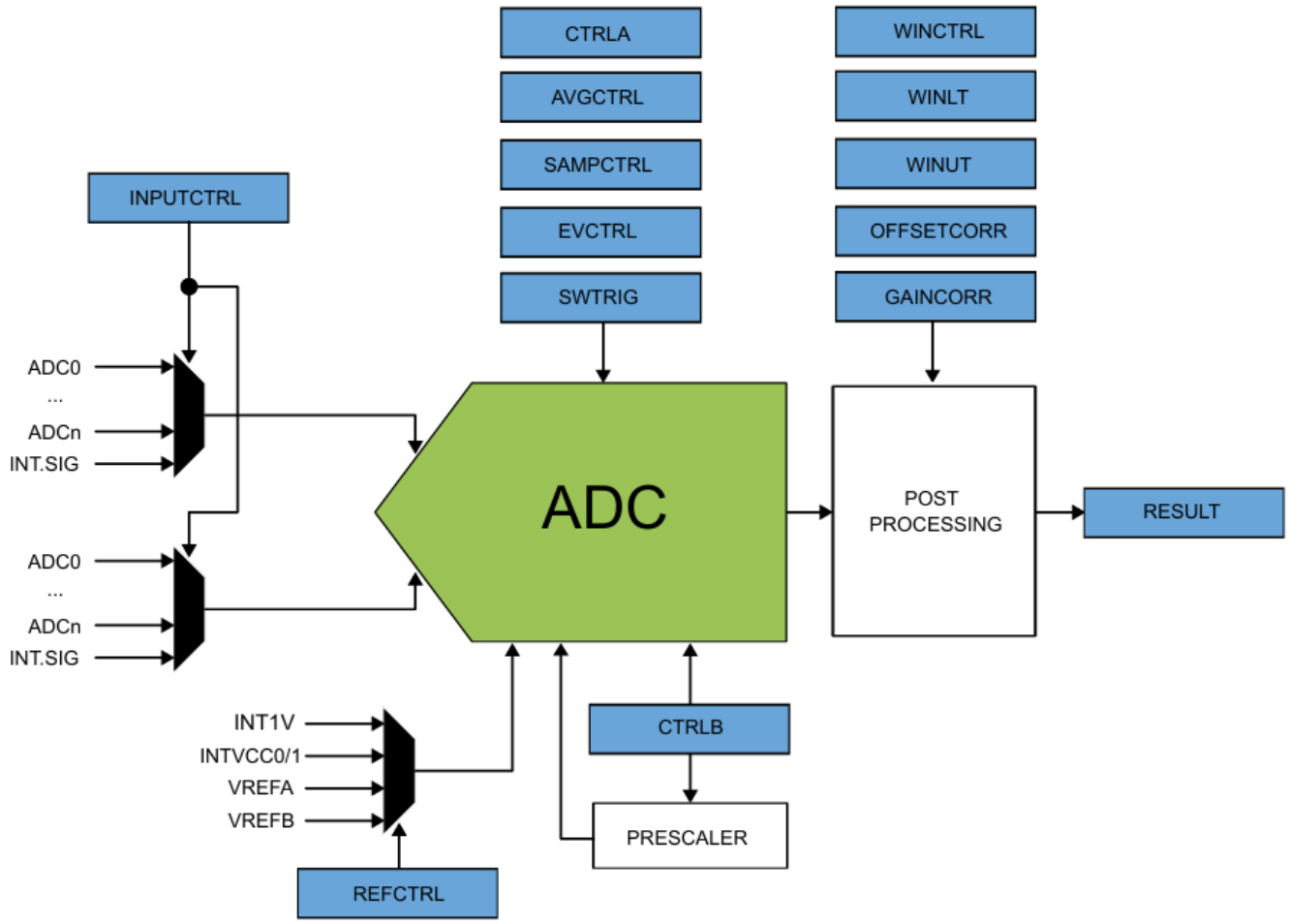
Analog-to-Digital Converter

ADC - Analog-to-Digital Converter

- **Features**
  - 8-bit, 10-bit, or 12-bit resolution
  - Up to 350,000 samples per second (350 ksps)
  - Differential and single-ended inputs:
    - Up to 32 analog input
      - 25 positive and 10 negative, including internal and external
    - Five internal inputs :
      - Bandgap、Temperature sensor、DAC、caled core supply、 Scaled I/O supply
    - 1/2x to 16x gain
    - Single, Continuous and Pin-scan Conversion Options
    - Windowing Monitor with Selectable Channel
    - Built-in Internal Reference and External Reference Options: Four bits for reference selection
    - Event-triggered Conversion for Accurate Timing (One Event Input)
    - Hardware Gain and Offset Compensation
    - Averaging and Oversampling with Decimation to Support, up to 16-bit Result
    - Selectable Sampling Time

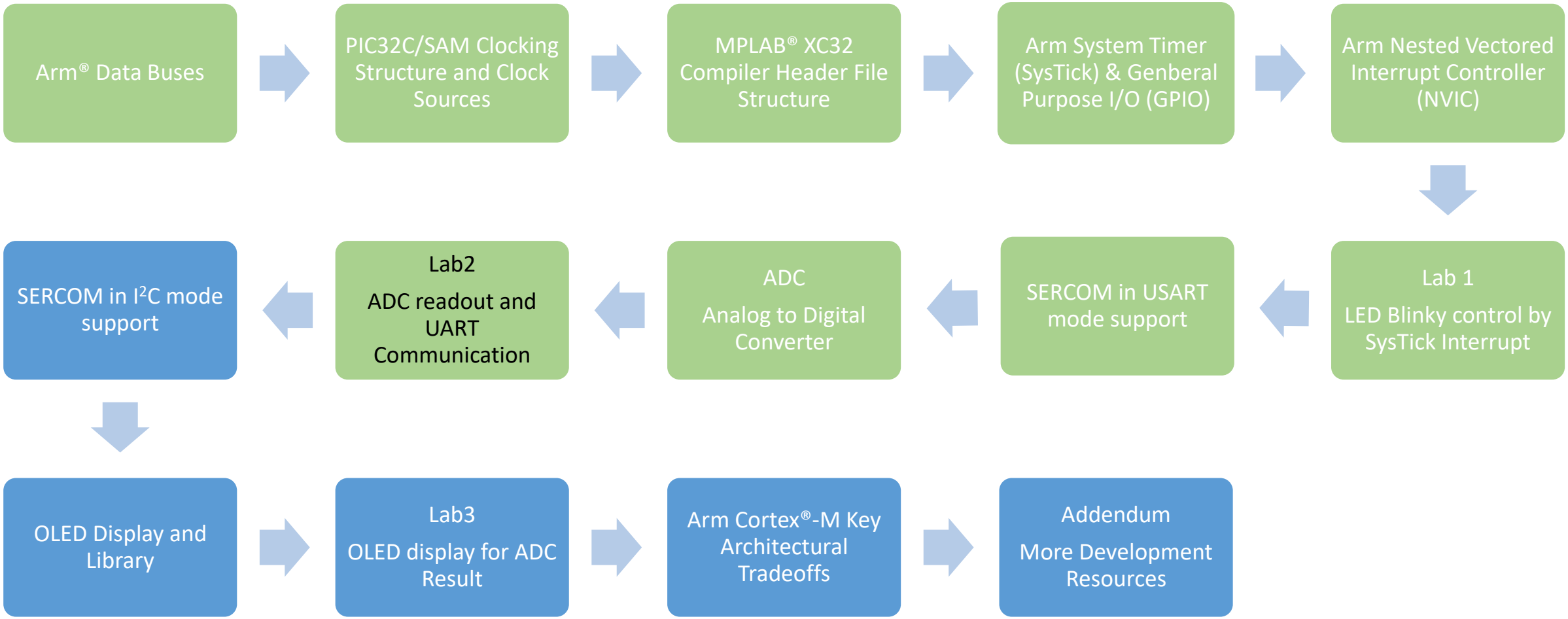
ADC

Block Diagram



Note: INT1V is the buffered internal reference of 1.0V, derived from the internal 1.1V bandgap reference.

Class Outline



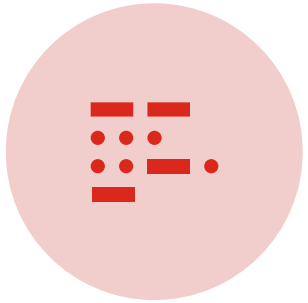
Lab 2 – Interface to the world

ADC read and UART communication

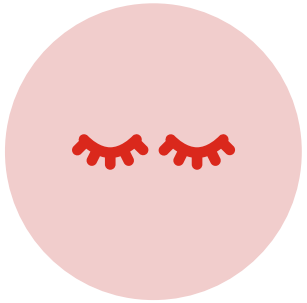
Lab 2 Objectives

Continue setup from Lab 1

Set up UART (115200,N,8,1) & ADC



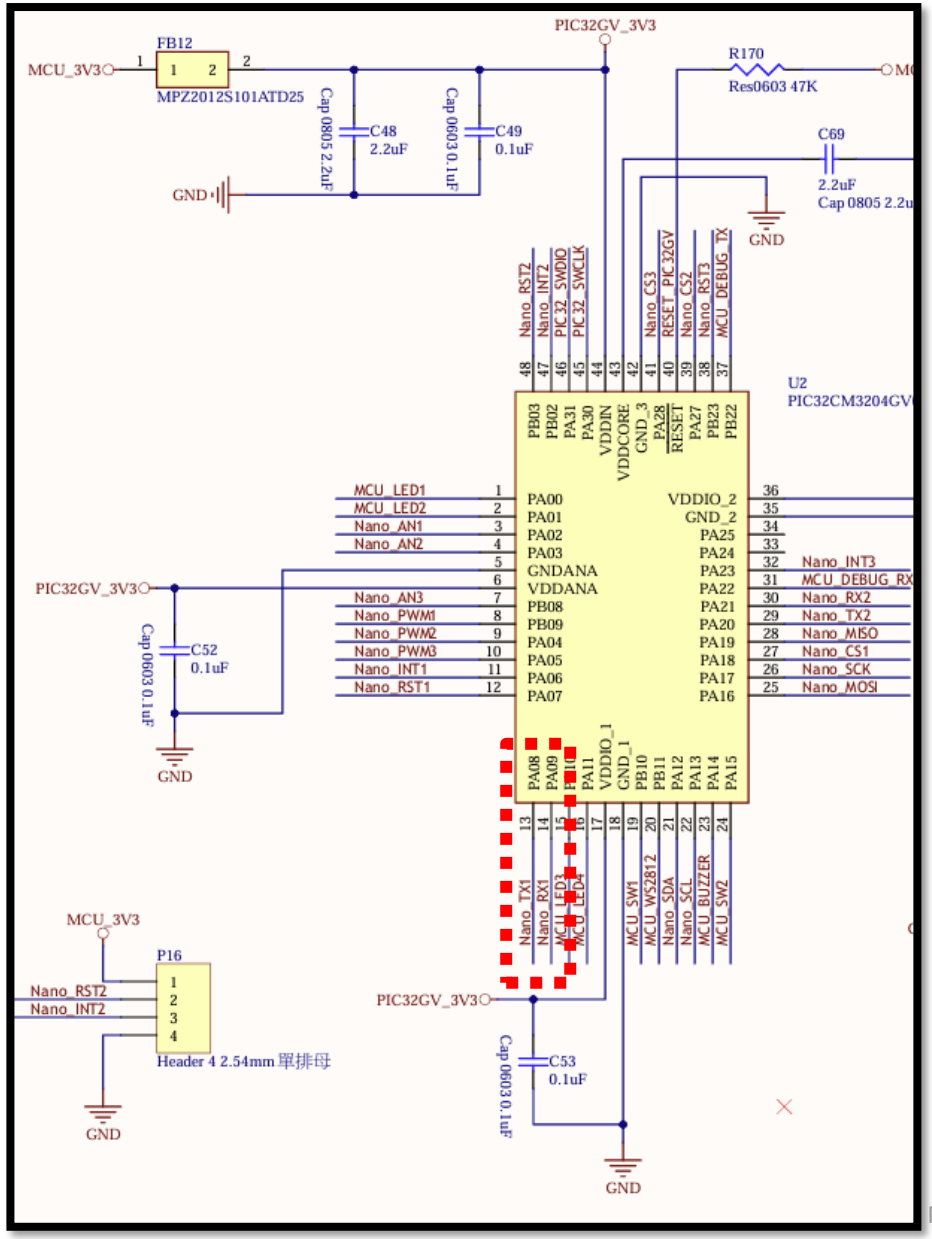
Review the generated PLIBs

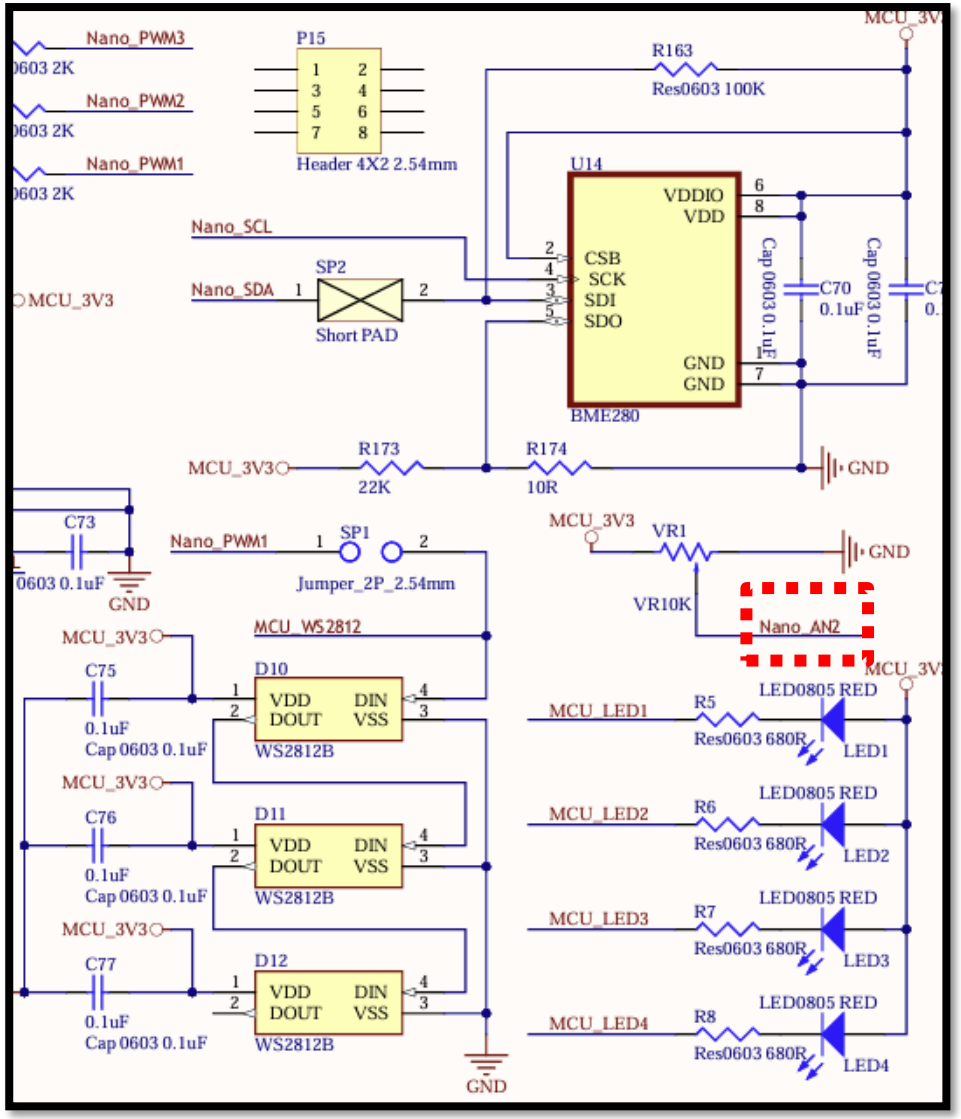


Send VR(ADC) result through UART to

MPLAB® Data Visualizer

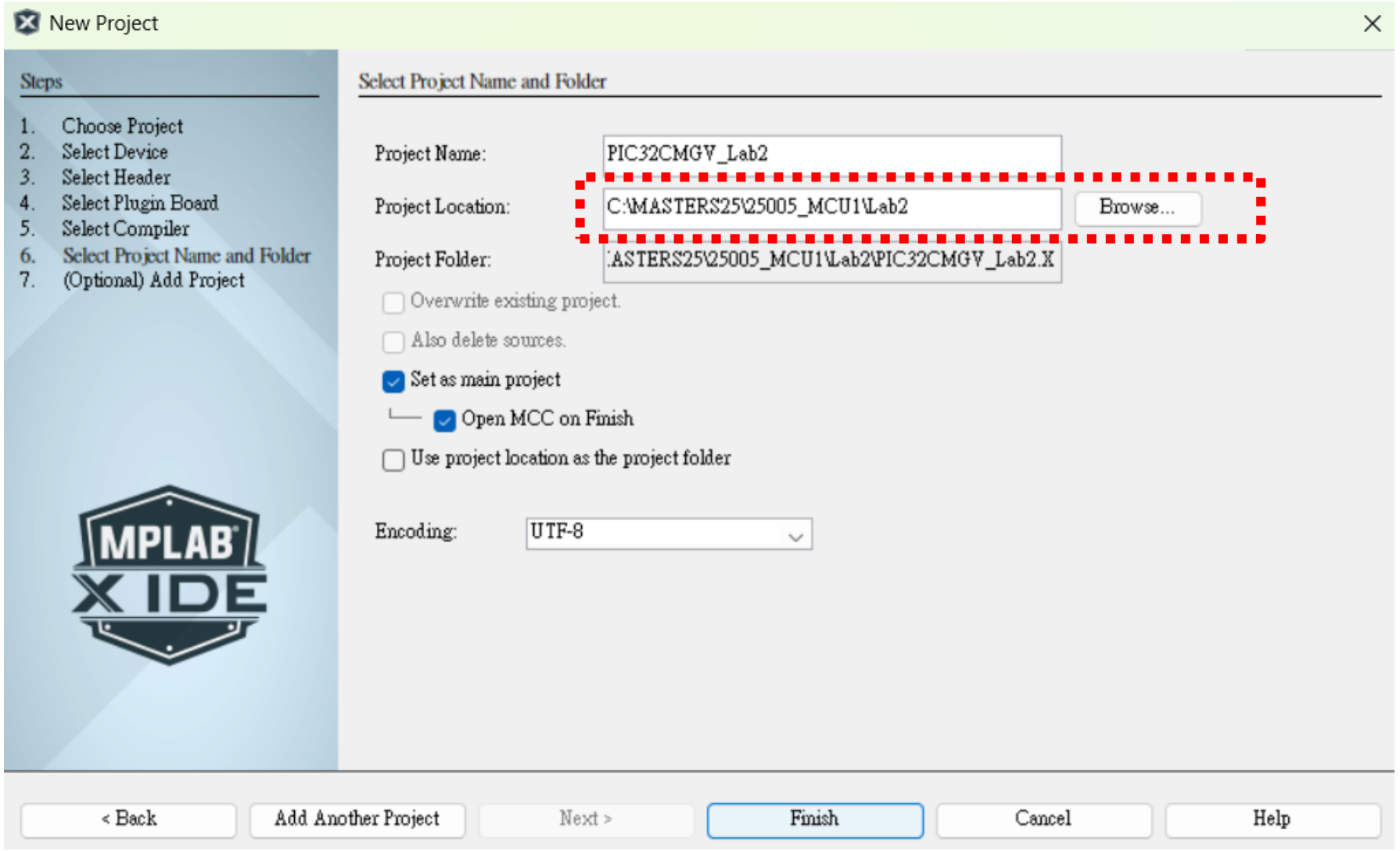
Lab2 – Interface to the World -> 使用到的接腳





Lab2 – Interface to the World

建立 Lab2 保留 Lab1 的部分, 讓 SysTick 以 200ms 的時間來中斷 MCU



MICROCHIP

Lab2 – Interface to the World

增加對 VR (PA03) 以及 UART (PA08&PA09) Pin 腳的設定

Start Page | MCC | Pin Diagram | Pin Table | Pin Settings

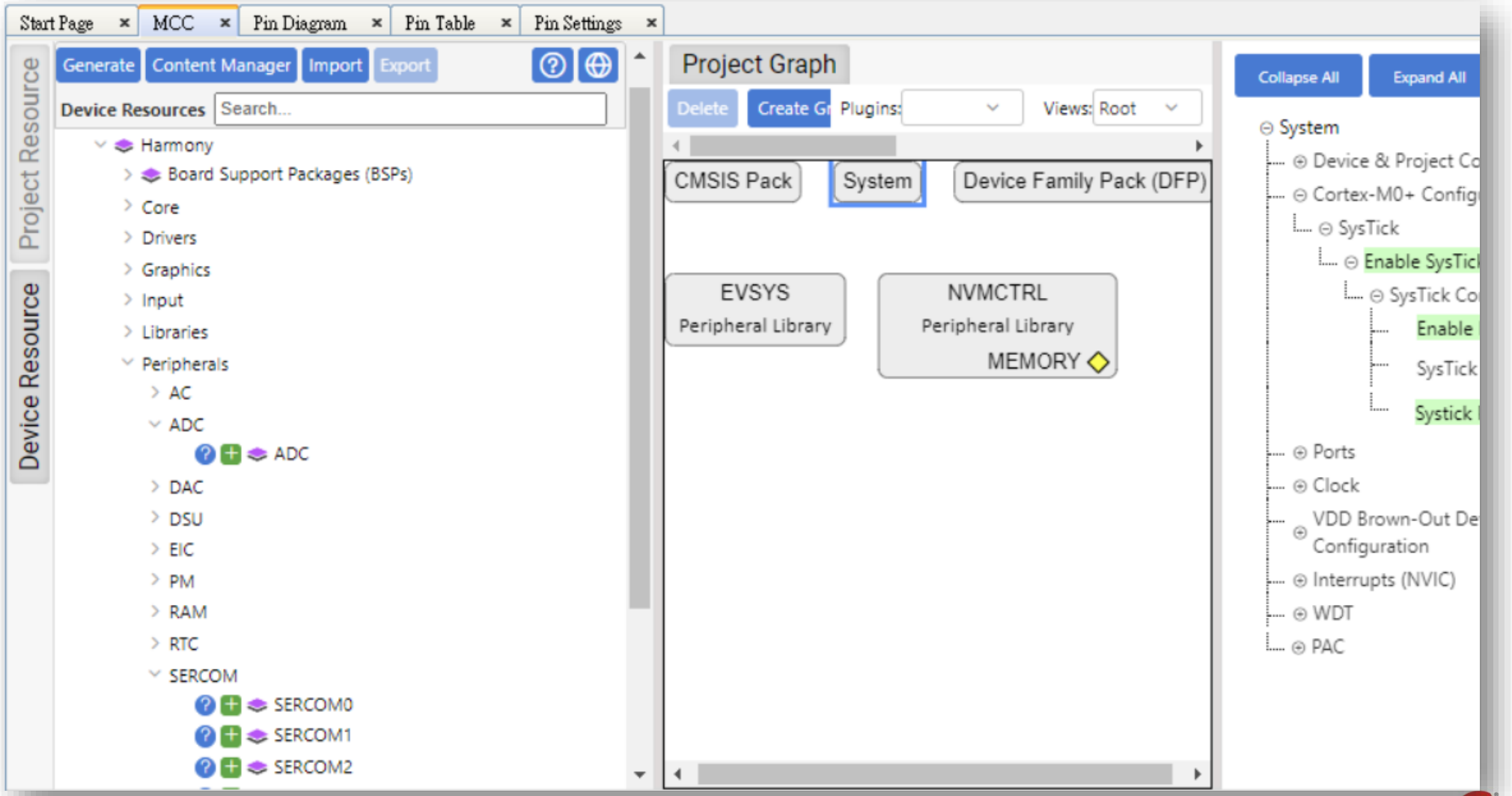
Order: Pins | Table View | Easy View

Pin Number	Pin ID	Custom Name	Function	Mode	Direction	Latch	Pull Up	Pull Down	Drive Strength
1	PA00	LED1	GPIO	Digital	Out	Low			NORMAL
2	PA01		Available	Digital	High Impedance	Low			NORMAL
3	PA02		Available	Digital	High Impedance	Low			NORMAL
4	PA03	VR1	ADC_AIN1	Analog	High Impedance	n/a			NORMAL
5	GNDANA			Digital	High Impedance	Low			NORMAL
6	VDDANA			Digital	High Impedance	Low			NORMAL
7	PB08		Available	Digital	High Impedance	Low			NORMAL
8	PB09		Available	Digital	High Impedance	Low			NORMAL
9	PA04		Available	Digital	High Impedance	Low			NORMAL
10	PA05		Available	Digital	High Impedance	Low			NORMAL
11	PA06		Available	Digital	High Impedance	Low			NORMAL
12	PA07		Available	Digital	High Impedance	Low			NORMAL
13	PA08		SERCOM0_PAD0	Digital	High Impedance	n/a			NORMAL
14	PA09		SERCOM2_PAD1	Digital	High Impedance	n/a			NORMAL
15	PA10		Available	Digital	High Impedance	Low			NORMAL
16	PA11		Available	Digital	High Impedance	Low			NORMAL
17	VDDIO			Digital	High Impedance	Low			NORMAL
18	GNDIO			Digital	High Impedance	Low			NORMAL
19	PB10		Available	Digital	High Impedance	Low			NORMAL

MICROCHIP

Lab2 – Interface to the World

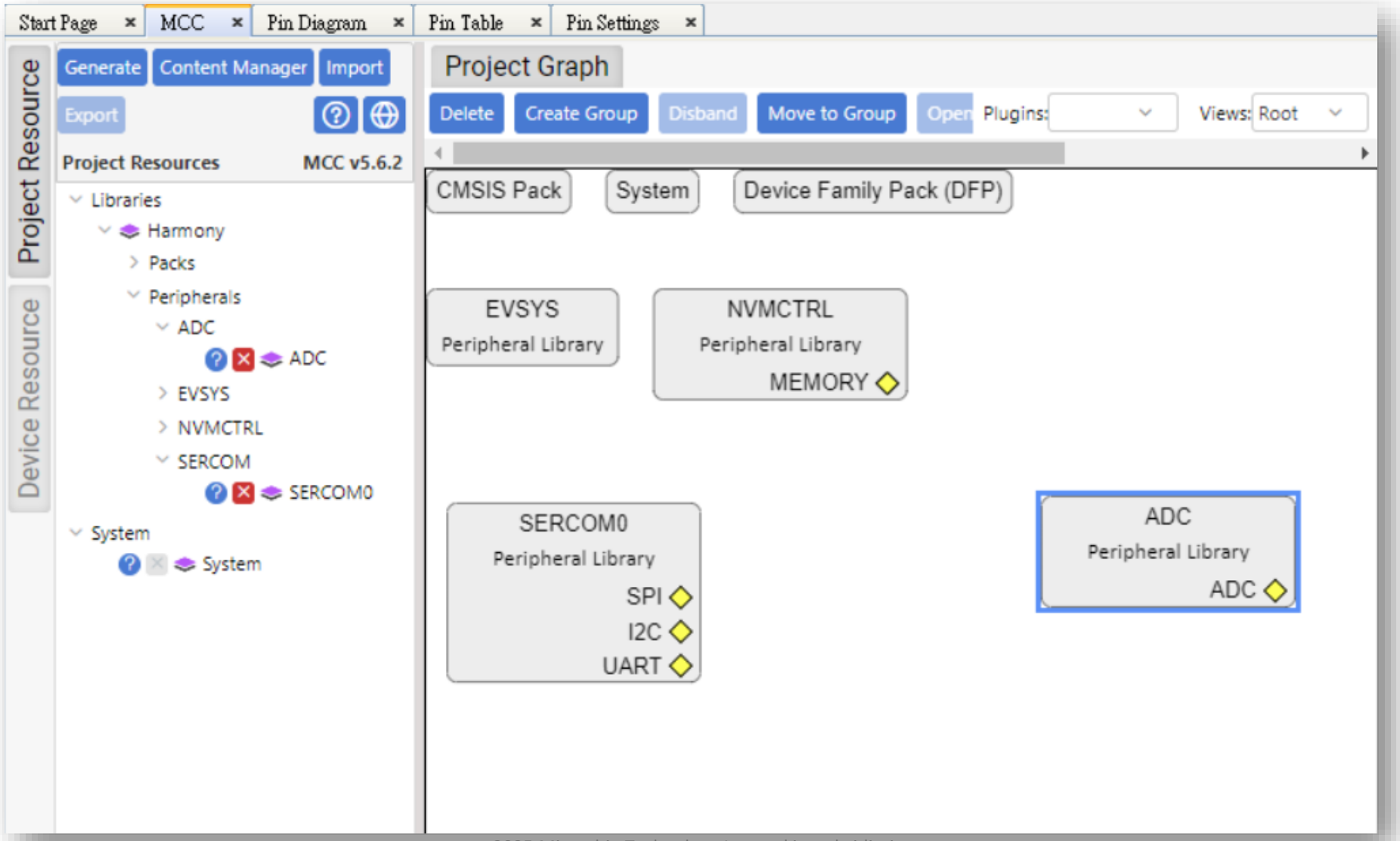
在 Device Resource 中加入 ADC & SERCOM0



MICROCHIP

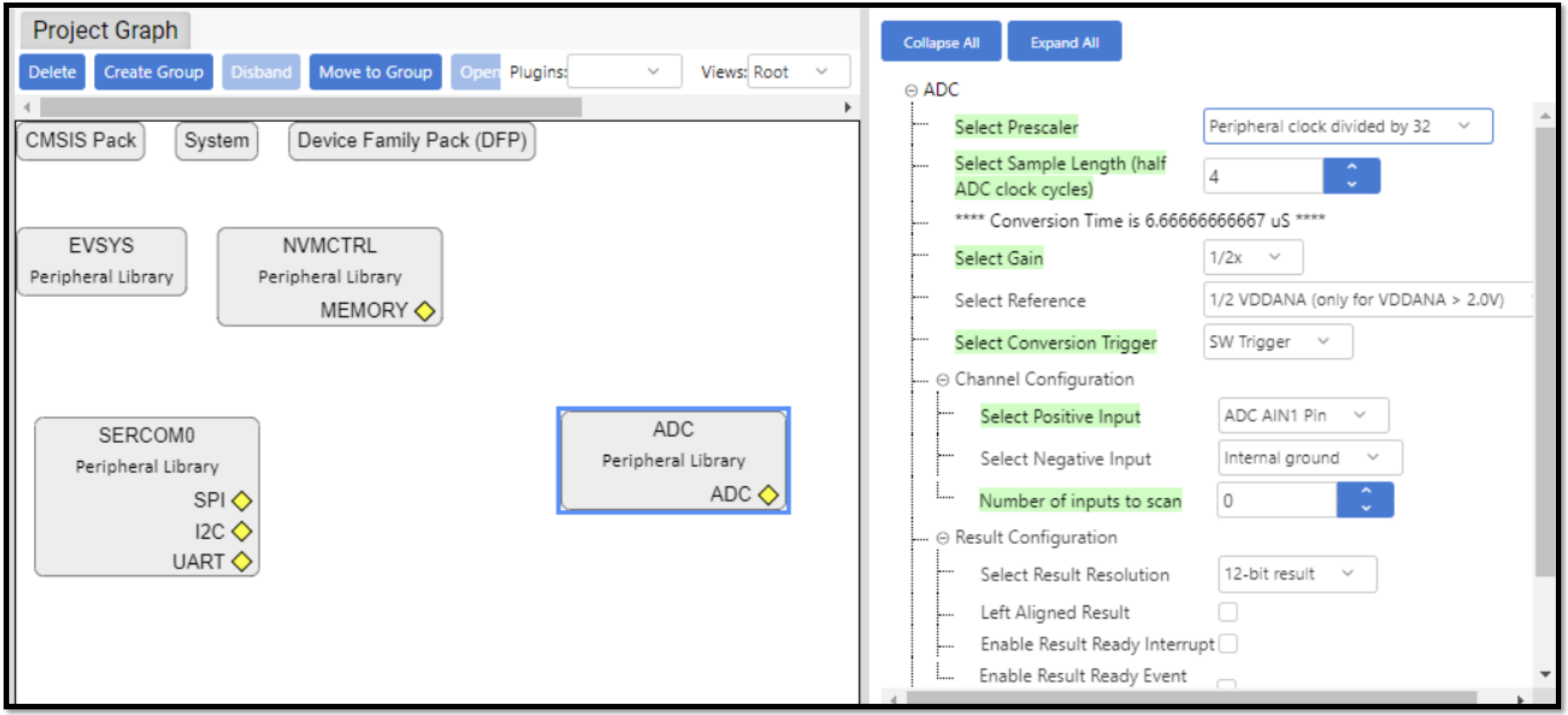
Lab2 – Interface to the World

在 Project Resource 中以及 MCC 視窗可以看到被加入的物件



Lab2 – Interface to the World

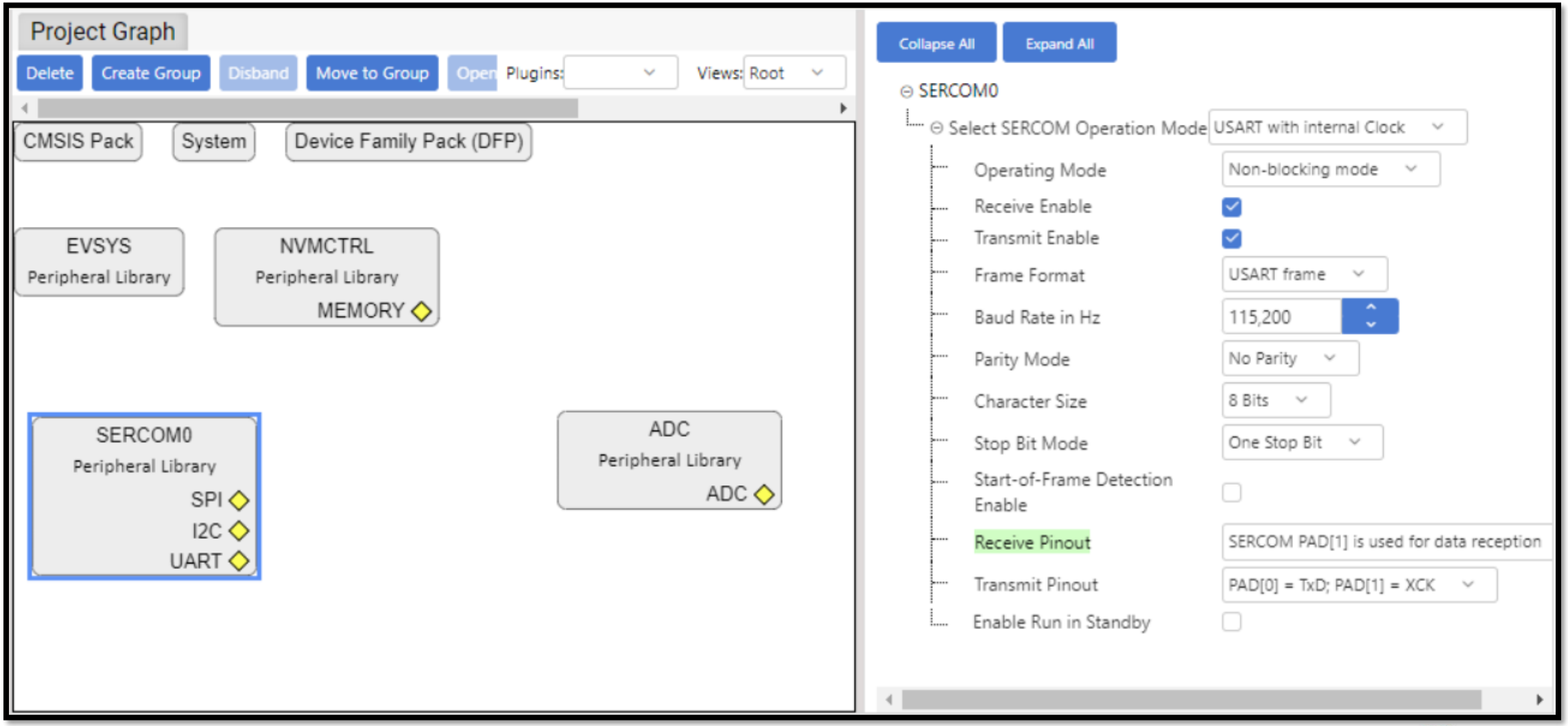
ADC 的規劃 – 針對單一 channel 的規劃 (VR的上下限為 0~3.3V)



MICROCHIP

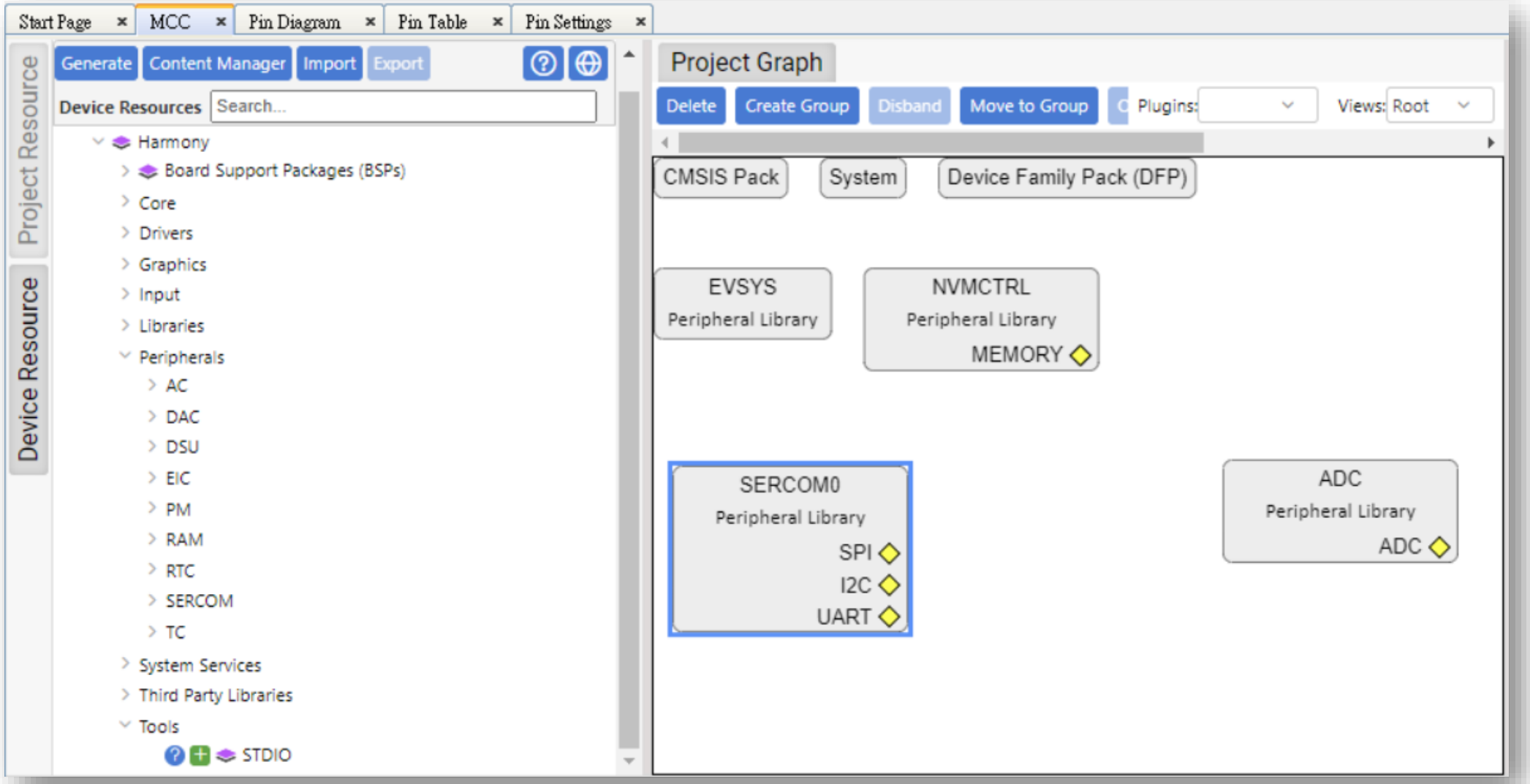
Lab2 – Interface to the World

UART 的規劃



Lab2 – Interface to the World

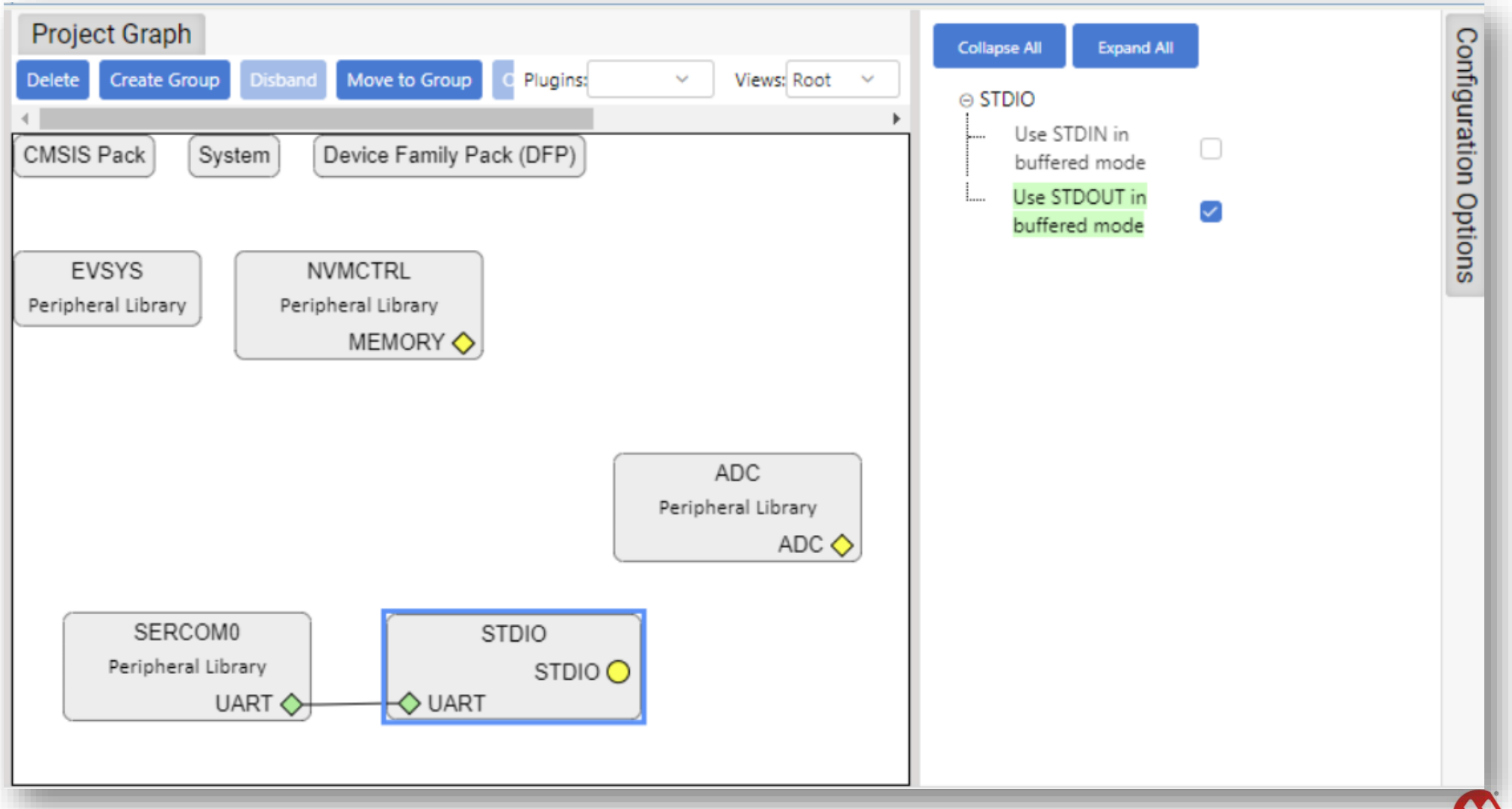
將 STDIO 加入以便於將 UART 的資料導向 stdout



MICROCHIP

Lab2 – Interface to the World

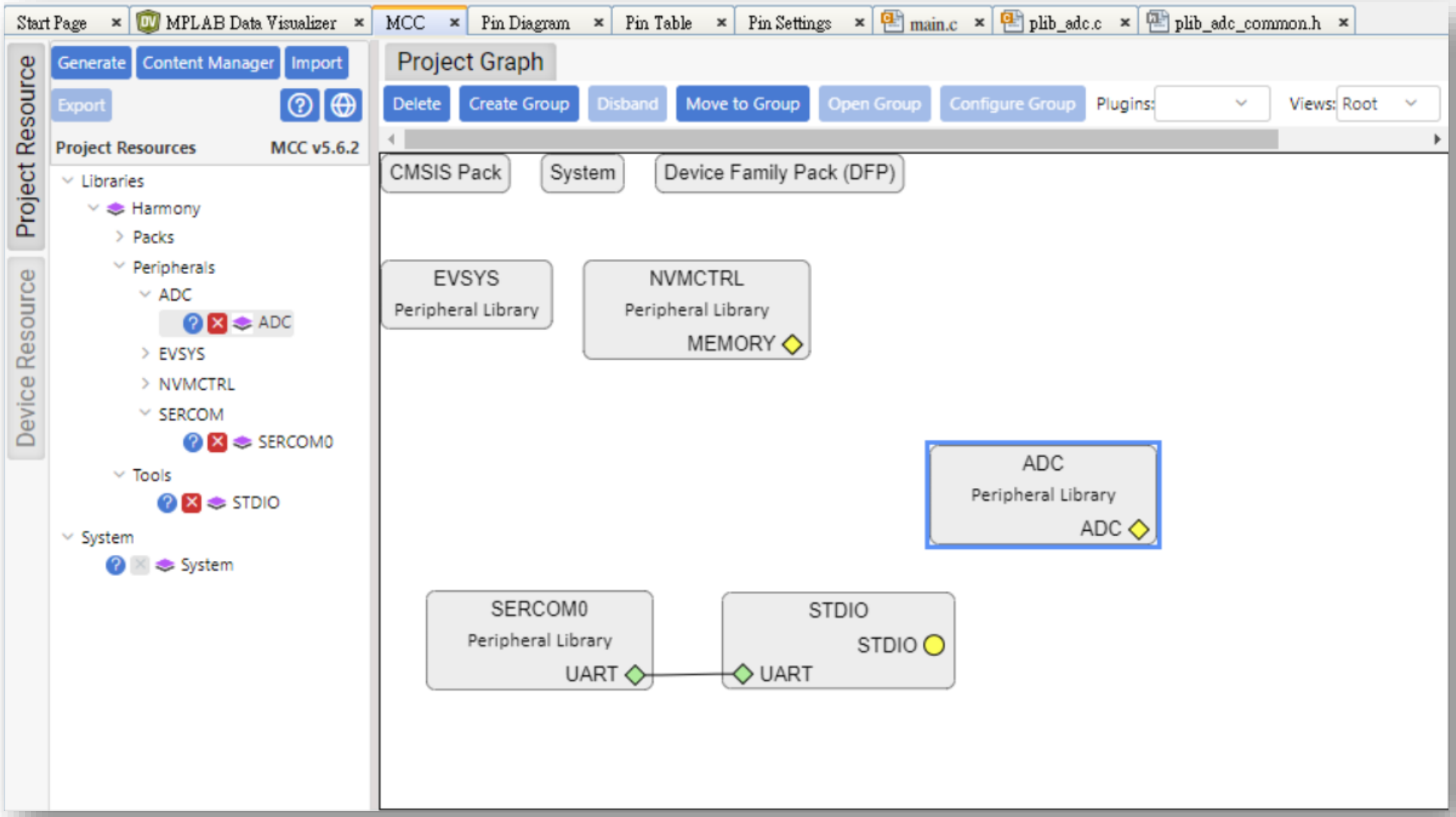
將 SERCOM0 以滑鼠與 STDIO 做連結即可使用 printf 來輸出資料



MICROCHIP

Lab2 – Interface to the World

點選 ADC module 左邊的問號 ADC 一樣可以連結到線上的說明文件



Lab2 – Interface to the World

ADC Module 的線上說明文件

MICROCHIP MPLAB® Harmony Peripheral Libraries

Home / 2 API Documentation / 2.3 Analog Digital Converter (ADC)

2.3 Analog Digital Converter (ADC)

This Plib implements software abstraction for ADC Peripheral.

Library Interface

Analog Digital Converter peripheral library provides the following interfaces:

Functions

Name	Description
ADCx_Initialize	Initializes ADC peripheral
ADCx_Enable	Enable (turn ON) ADC module
ADCx_Disable	Disable ADC module
ADC_SamplingStart	Starts the sampling
ADCx_ChannelSelect	Selects ADC input channel
ADCx_ConversionStart	Starts the ADC conversion of all the enabled channels with the

MICROCHIP

Lab2 – Interface to the World

ADCx_ConversionResultGet Function 的使用範例

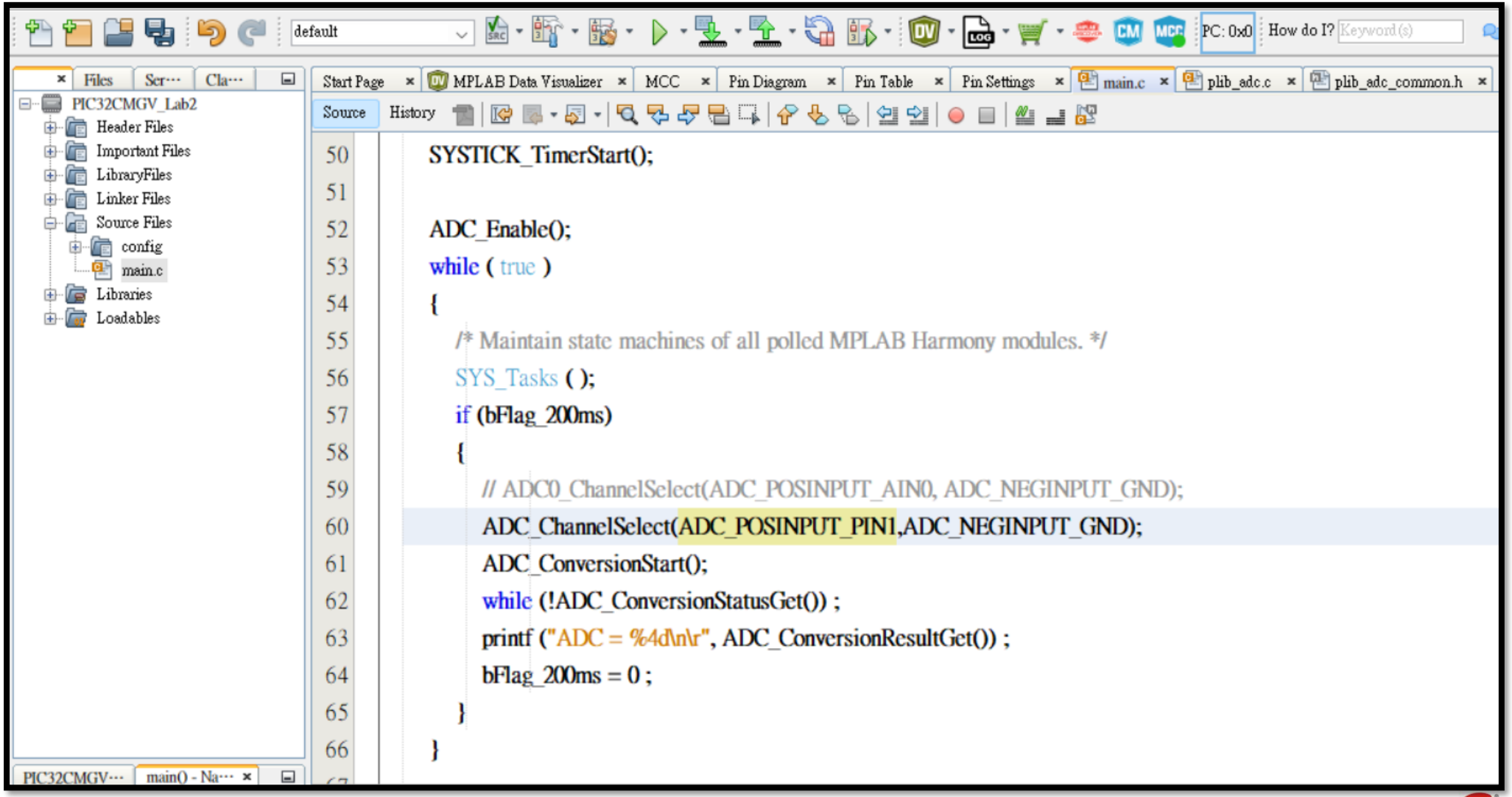
- 因為 **PIC32CM3204 GV** 系列只有一個 **ADC**，所以產生的 **API** 名稱為 **ADC_ConversionResultGet()**

Example

```
uint16_t adcResult = 0;
ADC0_Initialize();
ADC0_ConversionStart();
while(!ADC0_ConversionStatusGet());
adcResult = ADC0_ConversionResultGet();
```

Lab2 – Interface to the World

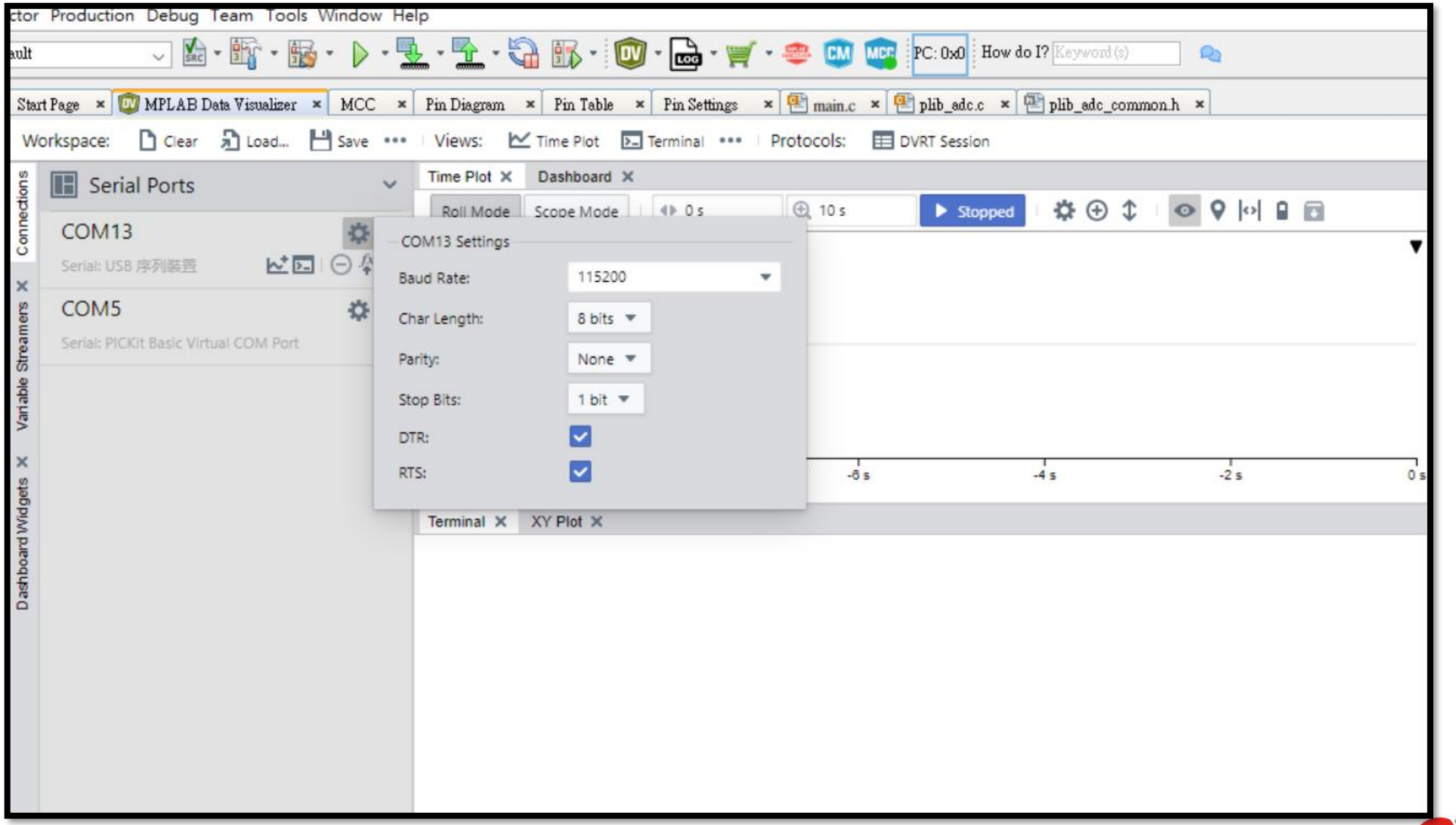
修改後的 main.c 主體，注意 bFlag_200ms 由 SysTick 的中斷設定



MICROCHIP

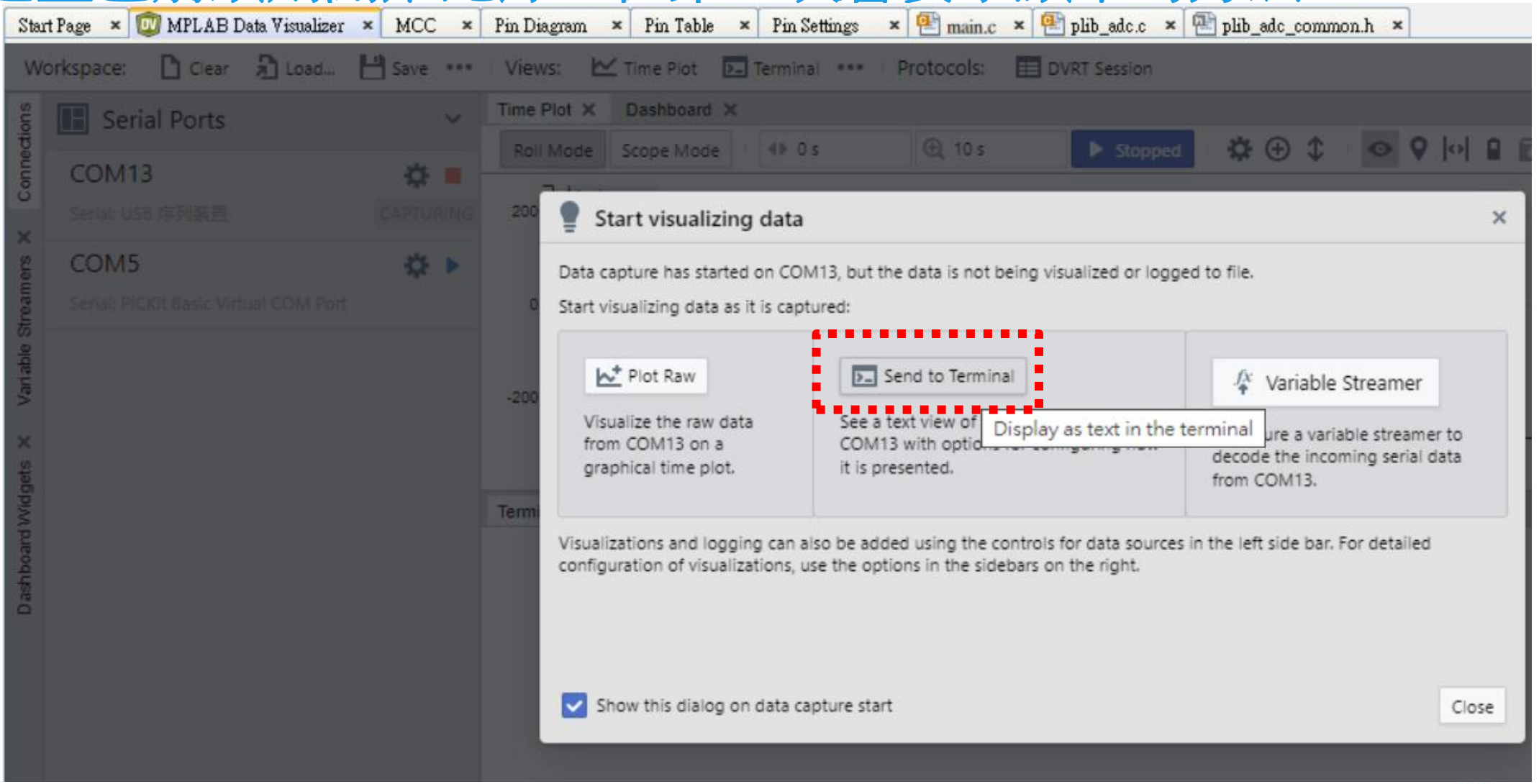
Lab2 – Interface to the World

點選 Data Visualizer，選擇正確的 COM 並確定通信速率及格式正確



Lab2 – Interface to the World

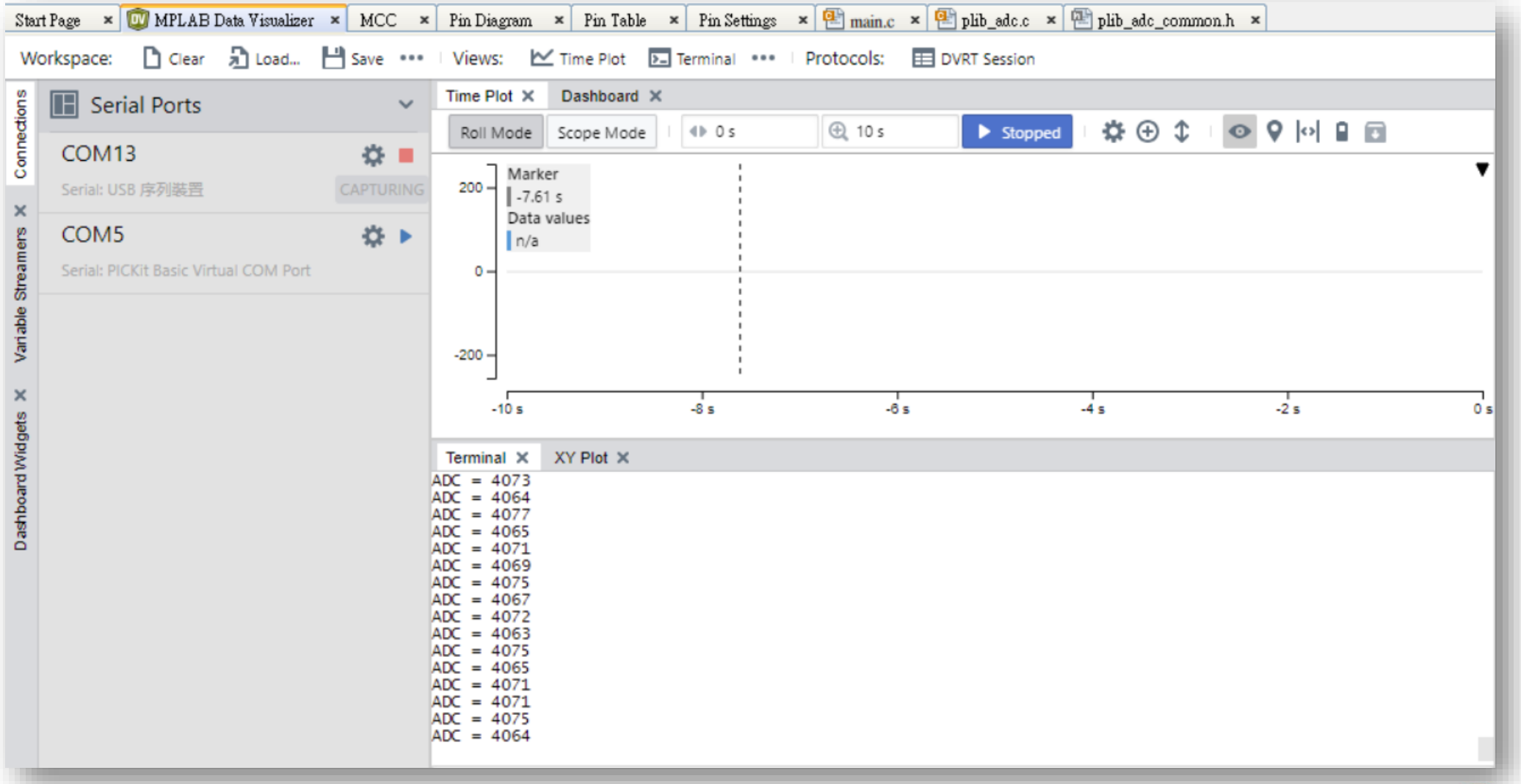
點選藍色箭頭則開始記錄，但第一次會要求顯示的方法



MICROCHIP

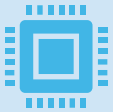
Lab2 – Interface to the World

ADC 的讀值被 Serial Port 輸出的結果 (轉動 VR 會看到變化)



MICROCHIP

Lab 2 Summary



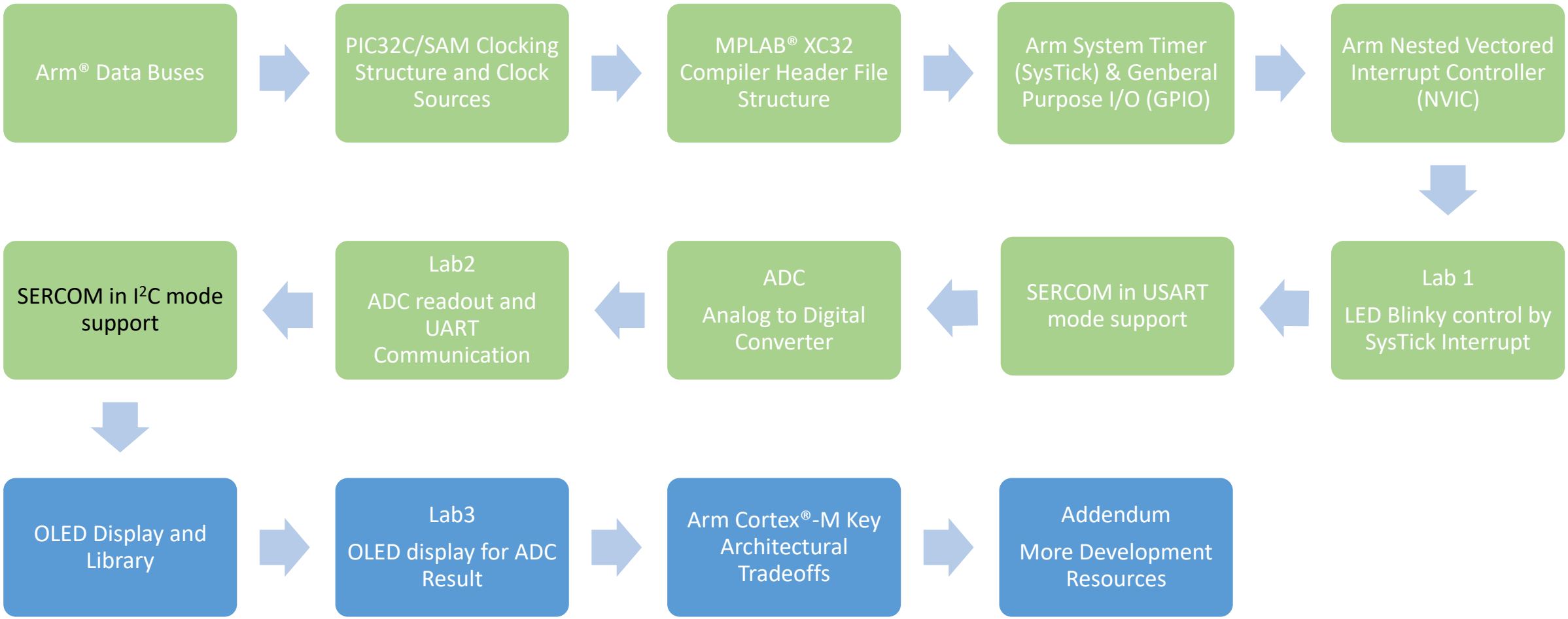
Properly set up SERCOM to support UART Function

Set up ADC module with proper Gain and Reference

Reviewed generated PLIBs and added code

Send VR(ADC) result through UART to MPLAB® Data Visualizer

Class Outline



SERCOM in I2C mode

Serial Communication Interface

What is I²C

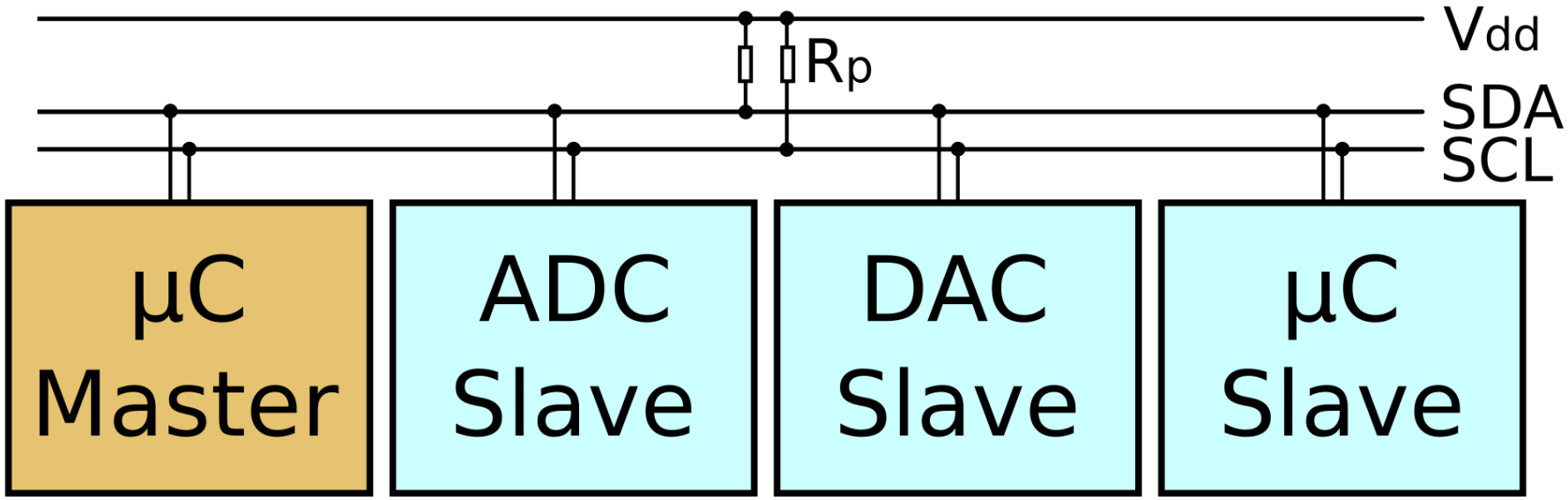
來自維基百科的資訊

- **I²C（Inter-Integrated Circuit）字面上的意思是積體電路之間，它其實是I²C Bus簡稱，所以中文應該叫積體匯流排電路，它是一種串列通訊匯流排，使用多主從架構，由飛利浦公司在1980年代為了讓主機板、嵌入式系統或手機用以連接低速週邊裝置而發展。I²C的正確讀法為「I平方C」（"I-squared-C"），而「I二C」（"I-two-C"）則是另一種錯誤但被廣泛使用的讀法。自2006年10月1日起，使用I²C協定已經不需要支付專利費，但製造商仍然需要付費以取得I²C從屬裝置位址[1]。**

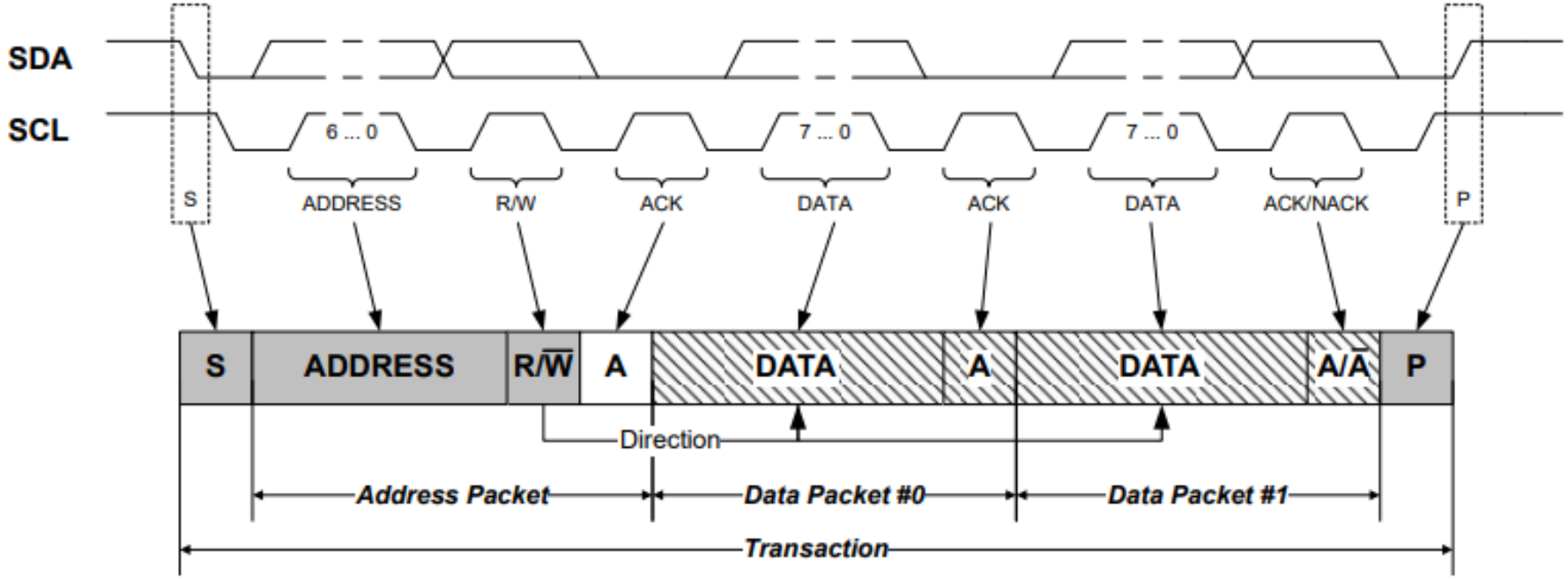
I2C 基礎知識

來自維基百科：

- **I²C只使用兩條雙向汲極開路（Open Drain）線，其中一條線為傳輸資料的串列資料線（SDA, Serial DAta line），另一條線是啟動或停止傳輸以及傳送時鐘序列的串列時脈（SCL, Serial CLock line）線，這兩條線上都有上拉電阻[2]。I²C允許相當大的工作電壓範圍，但典型的電壓準位為+3.3V或+5v。**



標準 I²C 介面傳輸的 Topology (7-Bit Addr.)



Bus Driver

- Host driving bus
- Client driving bus
- Either Host or Client driving bus

Special Bus Conditions

- S — START condition
- Sr — repeated START condition
- P — STOP condition

Data Package Direction

- R ('1') — Host Read
- W ('0') — Host Write

Acknowledge

- A ('0') — Acknowledge (ACK)
- Ā ('1') — Not Acknowledge (NACK)

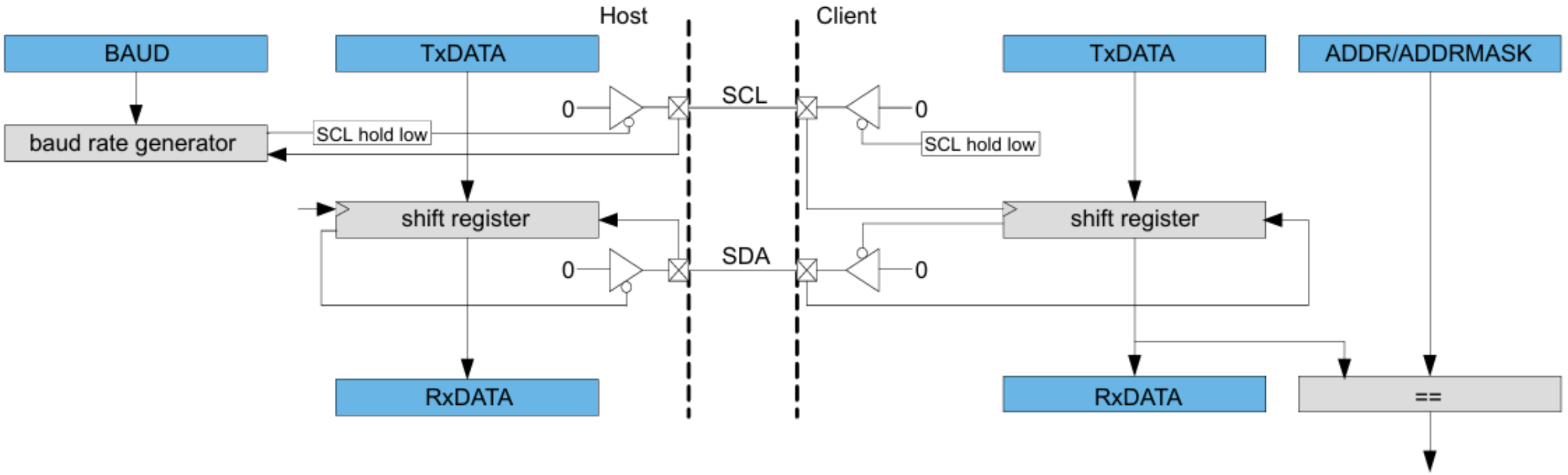
MICROCHIP

PIC32CM3204GV SERCOM 在 I2C mode 的主要功能

- **Host or Client operation**
- **Philips I2C compatible**
- **SMBus™ compatible**
- **Support of 100 kHz and 400 kHz I2C mode low system clock frequencies**
- **Physical interface includes:– Slew-rate limited outputs– Filtered inputs**
- **Client operation:**
  - Operation in all sleep modes
  - Wake-up on address match
  - 7-bit Address match in hardware for:
    - Unique address and/or 7-bit general call address
    - Address range
    - Two unique addresses

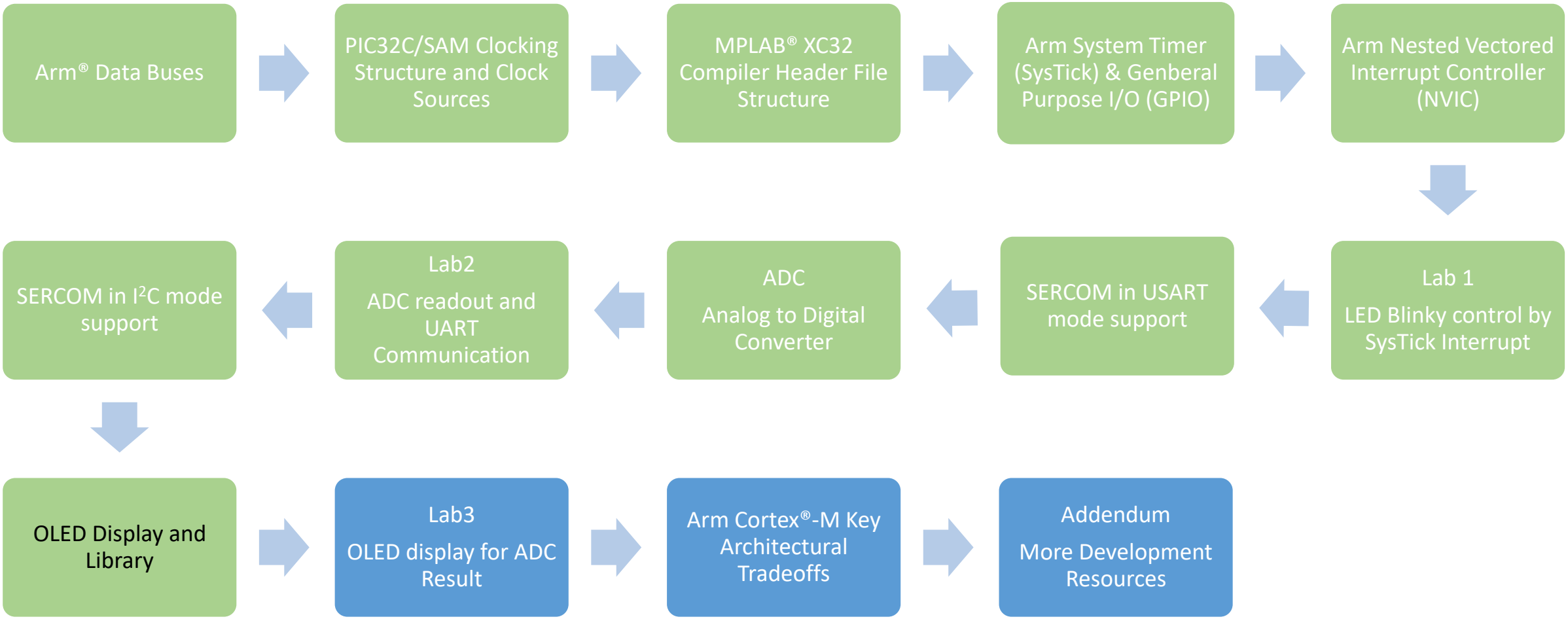
PIC32CM3204GV SERCOM 在 I2C mode 的方塊圖

Figure 25-1. I²C Single-Host Single-Client Interconnection



Signal Name	Type	Description
PAD[0]	Digital I/O	SDA
PAD[1]	Digital I/O	SCL
PAD[2]	Digital I/O	SDA_OUT (4-wire operation)
PAD[3]	Digital I/O	SCL_OUT (4-wire operation)

Class Outline



MICROCHIP

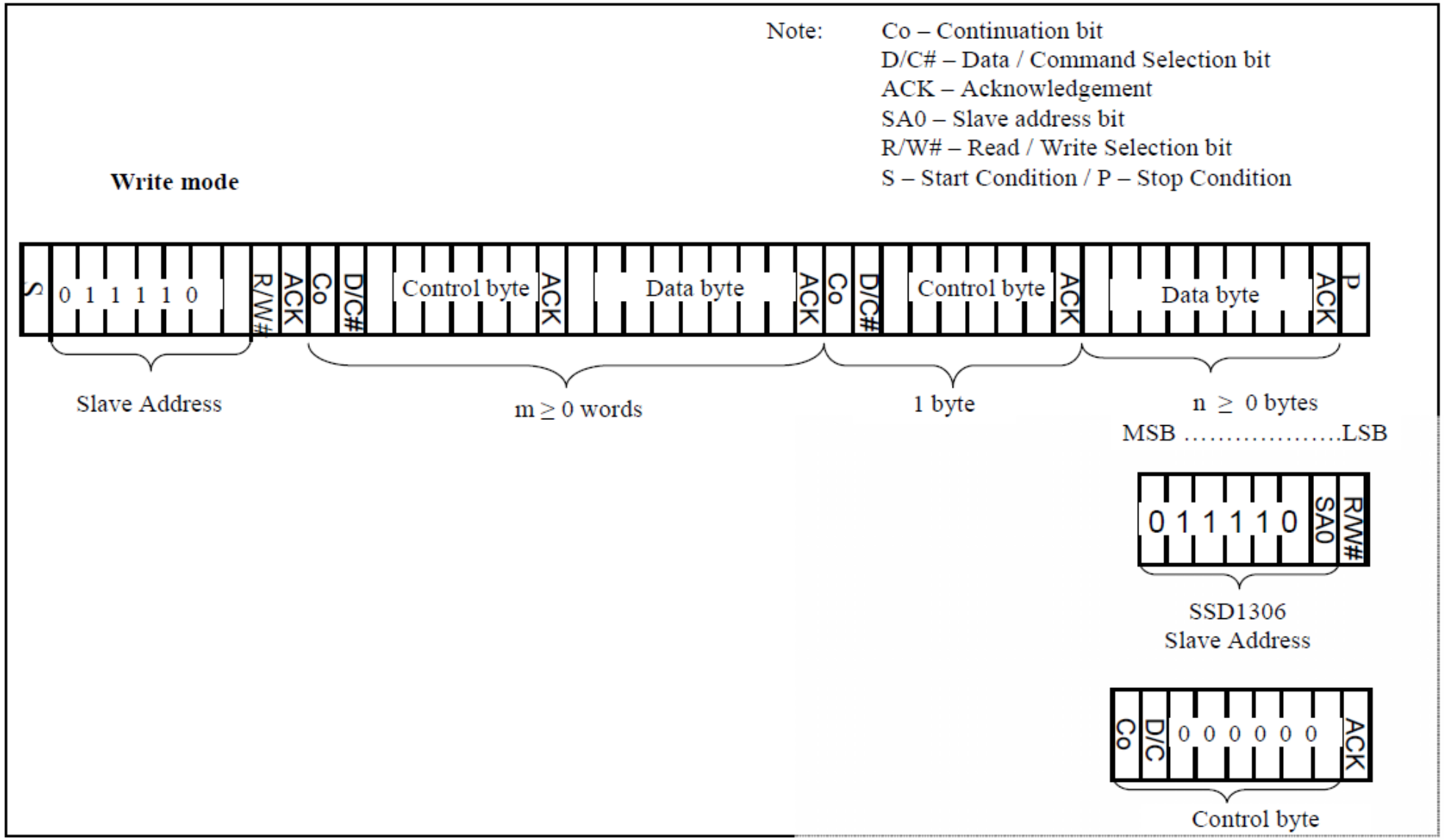
OLED Display

OLED Display簡介

- 使用驅動IC： SSD1306
- 0.96吋，解析度128 * 64
- 資料介面I²C
- VCC電源（3~5.5V）
- I²C位址 (0x3C)

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0	1	1	1	1	0	SA0	R/W#

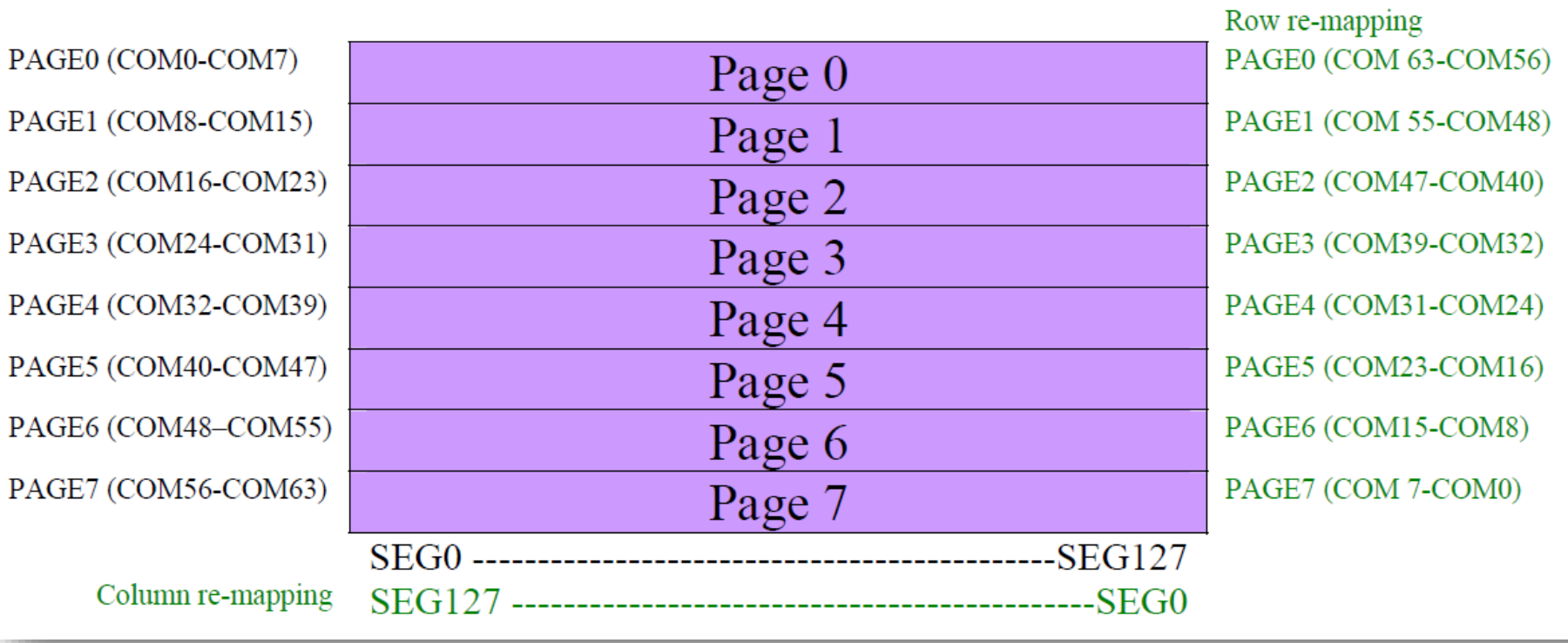
SSD1306 I²C Mode的資料格式



SSD1306内部的Display RAM

The GDDRAM is a bit mapped static RAM holding the bit pattern to be displayed. The size of the RAM is 128 x 64 bits and the RAM is divided into eight pages, from PAGE0 to PAGE7, which are used for monochrome 128x64 dot matrix display, as shown in Figure 8-13.

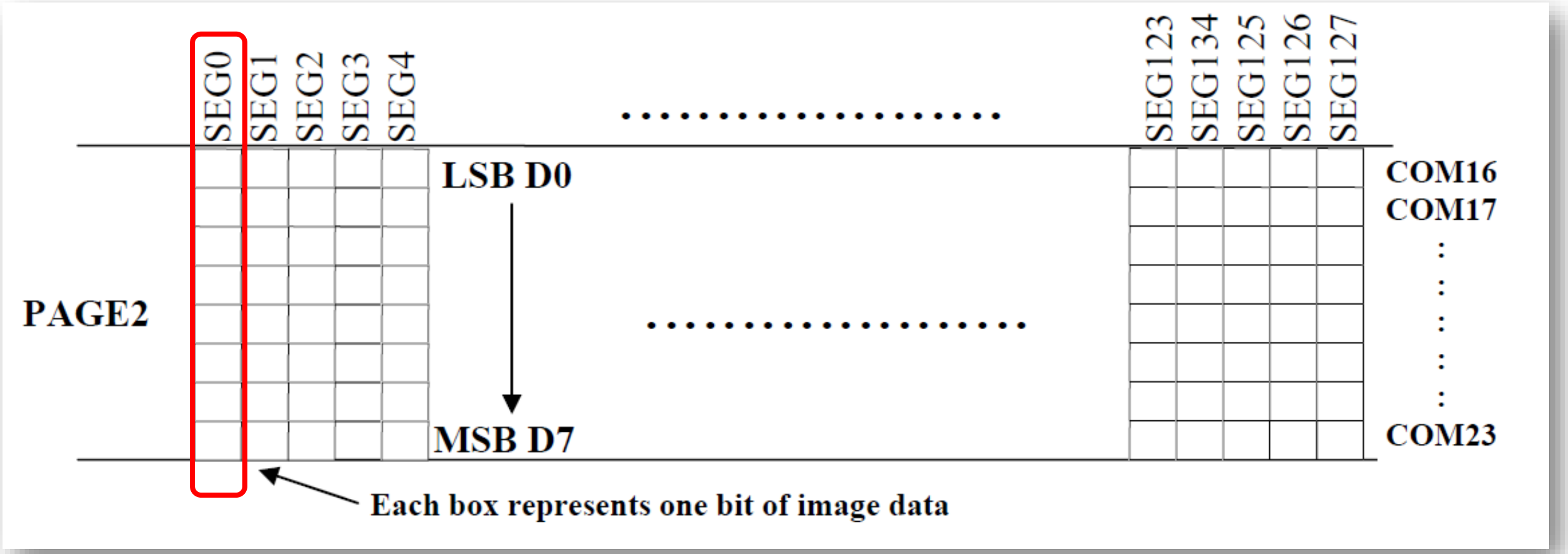
Figure 8-13 : GDDRAM pages structure of SSD1306



SSD1306 GDDRAM的放大圖

每一個Page有128個Bytes的資料，控制128*8個點

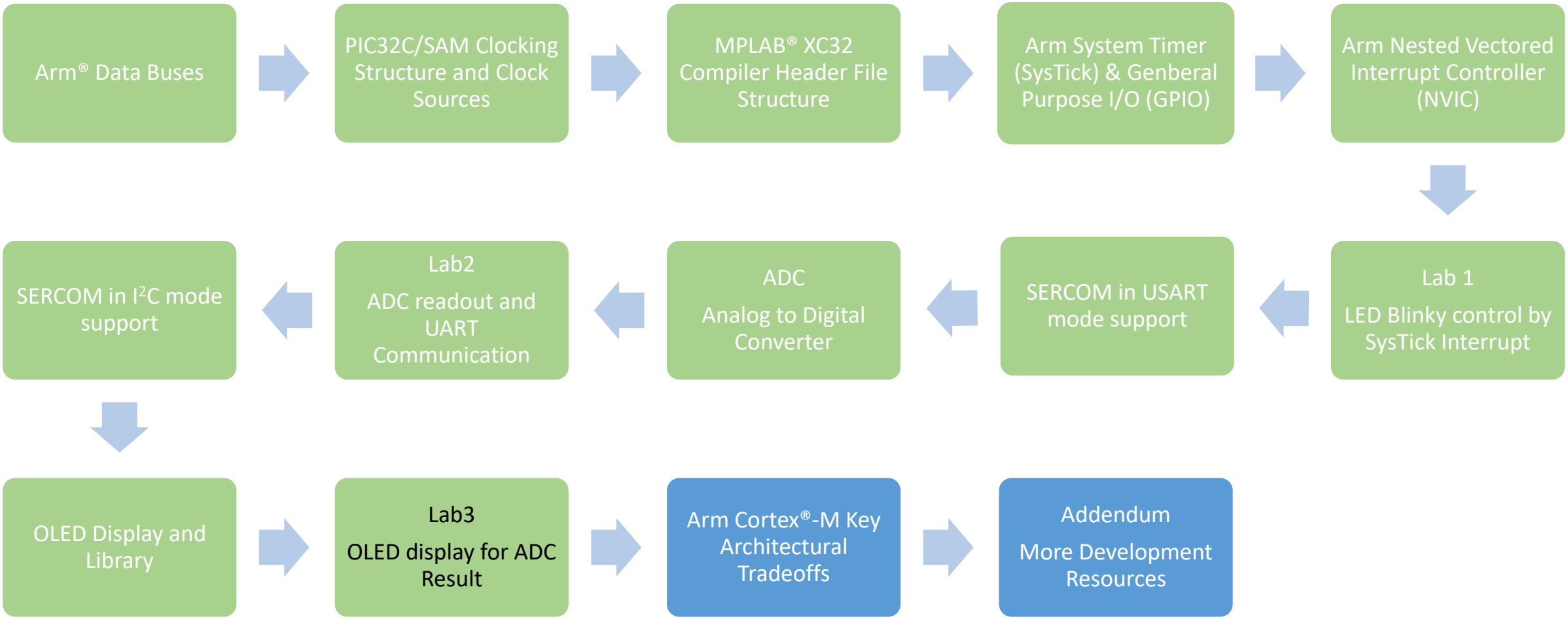
總共有8個Pages = 128 * 64點！



SSD1306 OLED 的操作程式

- 這是一個廣泛被使用的模組，顧可以輕易地找到開源的範例程式
- 教學課程中只以 **OLED** 做文字資料顯示，故只提供初始化以及文字處理的副程式。
- **OLED** 的副程式以及宣告已經 **copy to \MASTERS25\25005_MCU1\OLED**

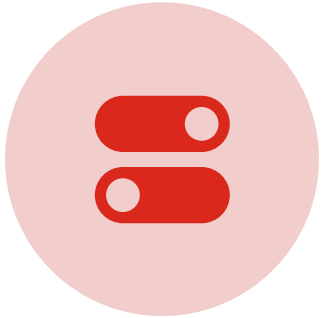
Class Outline



Lab 3 – OLED Display for ADC Result

Lab 3 Objectives

Continue setup from Lab 2 (Reserve all function)

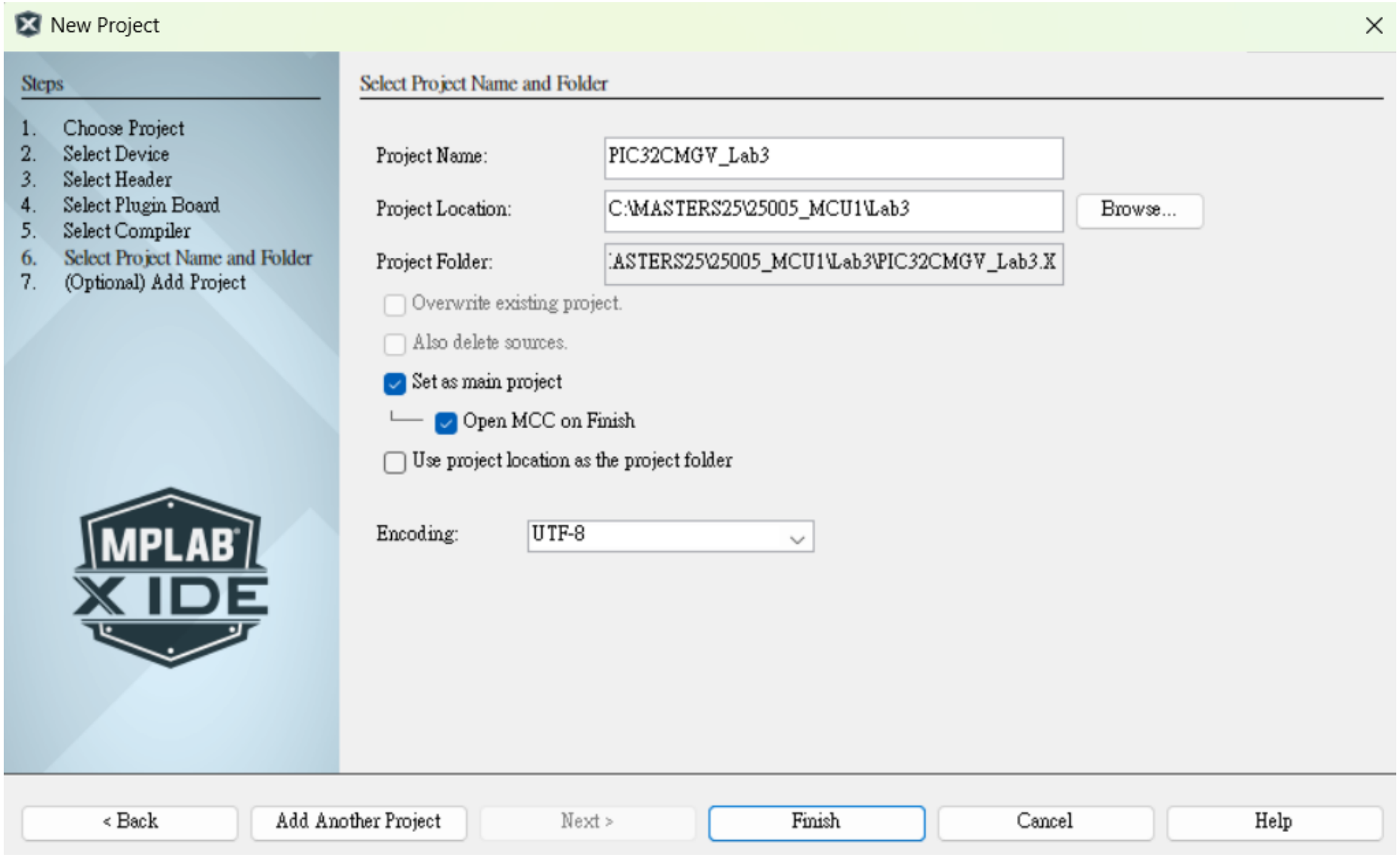


Set up SERCOM for I^2C Master support

Initialize and operate OLED display by using Library

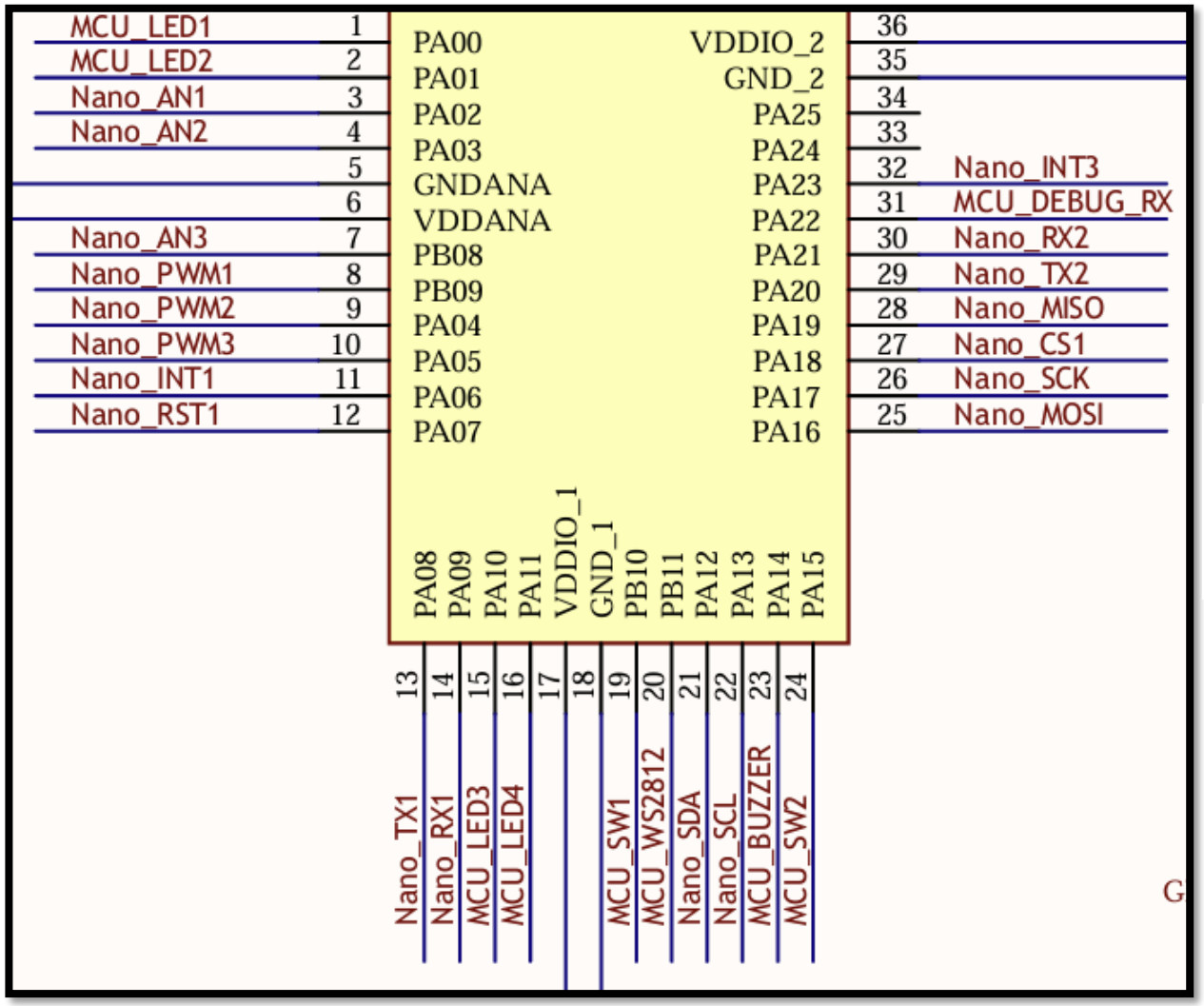
Additionally update ADC result to OLED periodically (200ms)

Lab 3 – OLED Display for ADC Result



Lab 3 – OLED Display for ADC Result

SDA & SCL are located @ PA12 & PA13



Signal Name	Type	Description
PAD[0]	Digital I/O	SDA
PAD[1]	Digital I/O	SCL
PAD[2]	Digital I/O	SDA_OUT (4-wire operation)
PAD[3]	Digital I/O	SCL_OUT (4-wire operation)

Lab 3 – OLED Display for ADC Result

Set SERCOM2 for I2C function (PAD0 = SDA, PAD1=SCL)

Start Page | MPLAB Data Visualizer | main.c | plib_adc.c | plib_adc_common.h | MCC | Pin Diagram | Pin Table | Pin Setting

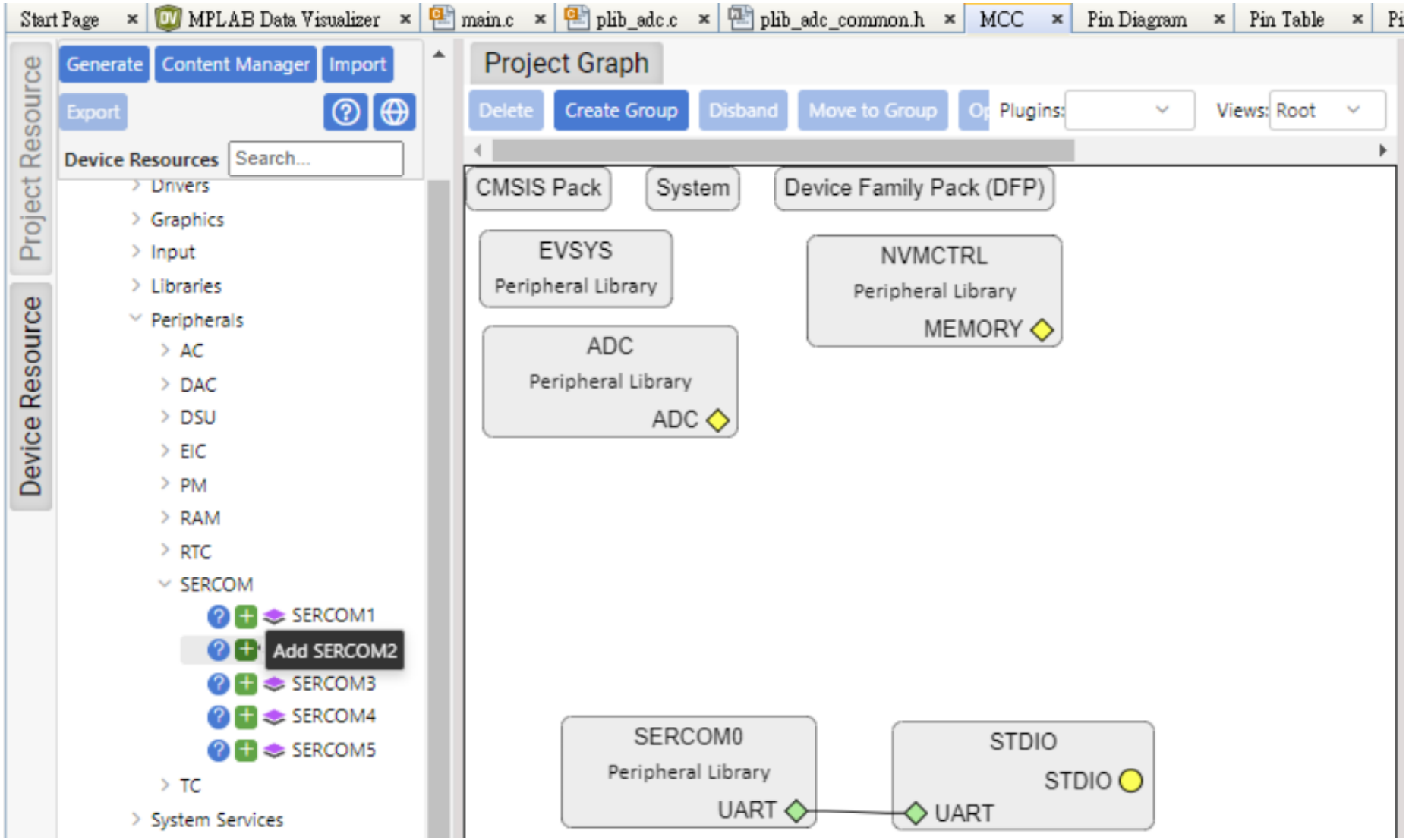
Order: Pins | Table View | Easy View

Pin Number	Pin ID	Custom Name	Function	Mode	Direction	Latch	Pull Up	Pull Down	Drive Strength
10	PA05		Available	Digital	High Impedance	Low	☐	☐	NORMAL
11	PA06		Available	Digital	High Impedance	Low	☐	☐	NORMAL
12	PA07		Available	Digital	High Impedance	Low	☐	☐	NORMAL
13	PA08		SERCOM0_PAD0	Digital	High Impedance	n/a	☐	☐	NORMAL
14	PA09		SERCOM0_PAD1	Digital	High Impedance	n/a	☐	☐	NORMAL
15	PA10		Available	Digital	High Impedance	Low	☐	☐	NORMAL
16	PA11		Available	Digital	High Impedance	Low	☐	☐	NORMAL
17	VDDIO			Digital	High Impedance	Low	☐	☐	NORMAL
18	GNDIO			Digital	High Impedance	Low	☐	☐	NORMAL
19	PB10		Available	Digital	High Impedance	Low	☐	☐	NORMAL
20	PB11		Available	Digital	High Impedance	Low	☐	☐	NORMAL
21	PA12		SERCOM2_PAD0	Digital	High Impedance	n/a	☐	☐	NORMAL
22	PA13		SERCOM2_PAD1	Digital	High Impedance	n/a	☐	☐	NORMAL
23	PA14		Available	Digital	High Impedance	Low	☐	☐	NORMAL
24	PA15		Available	Digital	High Impedance	Low	☐	☐	NORMAL
25	PA16		Available	Digital	High Impedance	Low	☐	☐	NORMAL
26	PA17		Available	Digital	High Impedance	Low	☐	☐	NORMAL
27	PA18		Available	Digital	High Impedance	Low	☐	☐	NORMAL

MICROCHIP

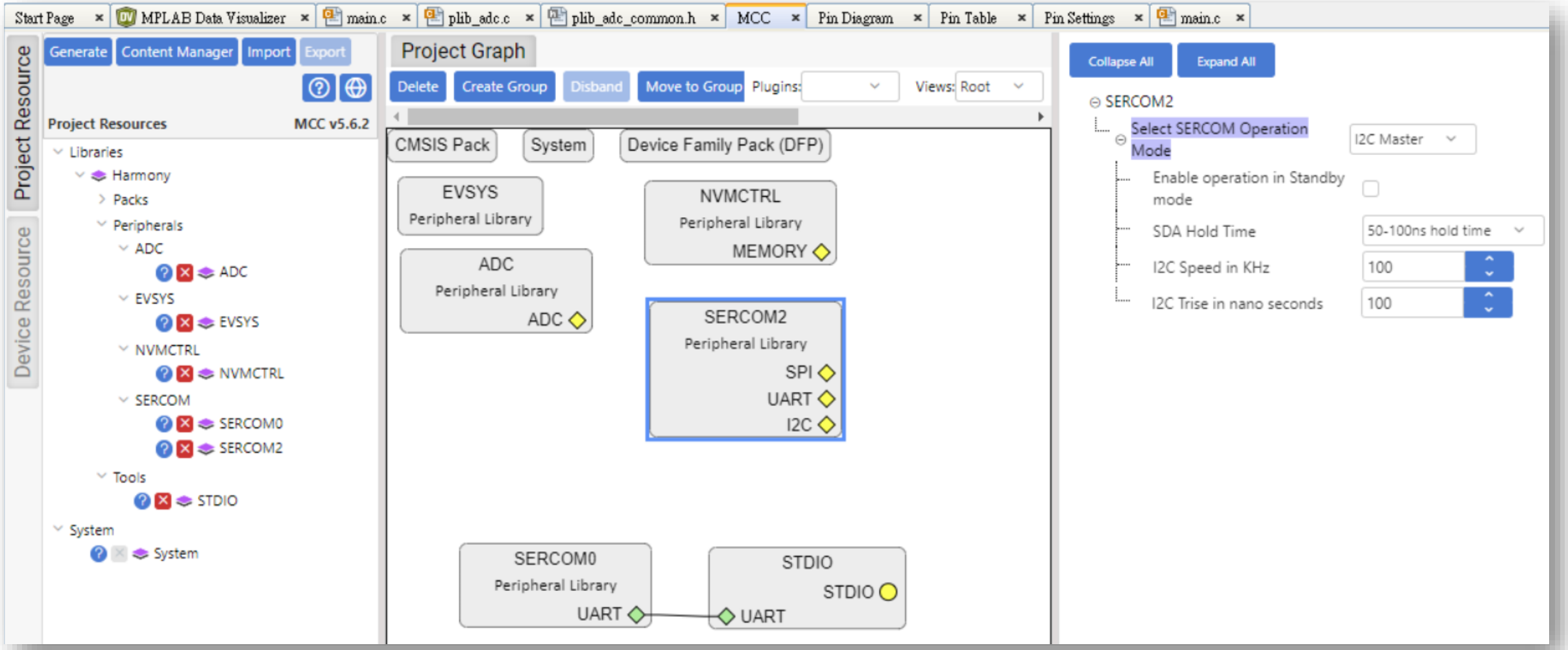
Lab 3 – OLED Display for ADC Result

加入 SERCOM2



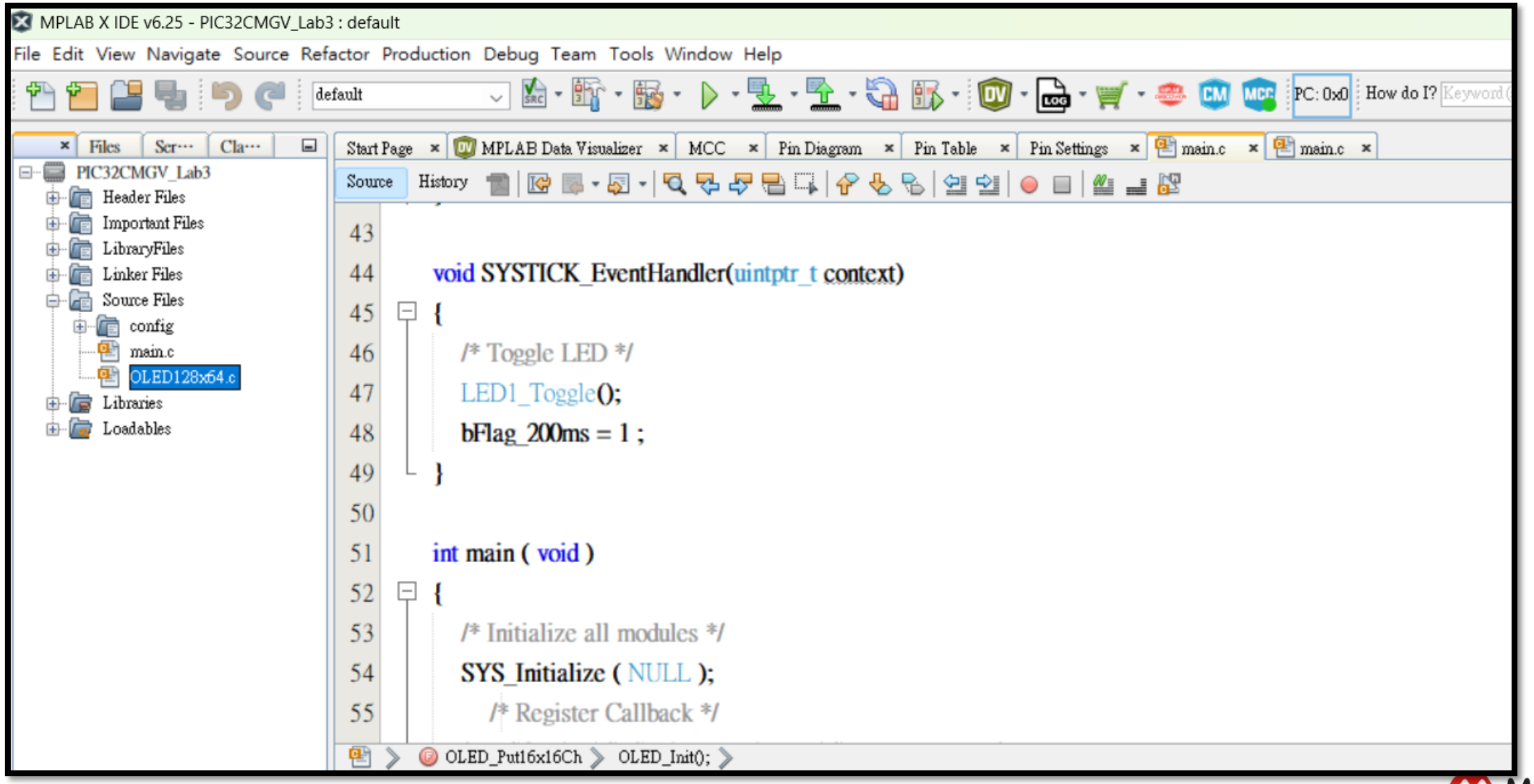
Lab 3 – OLED Display for ADC Result

Config SERCOM2 as I2C Master



Lab 3 – OLED Display for ADC Result

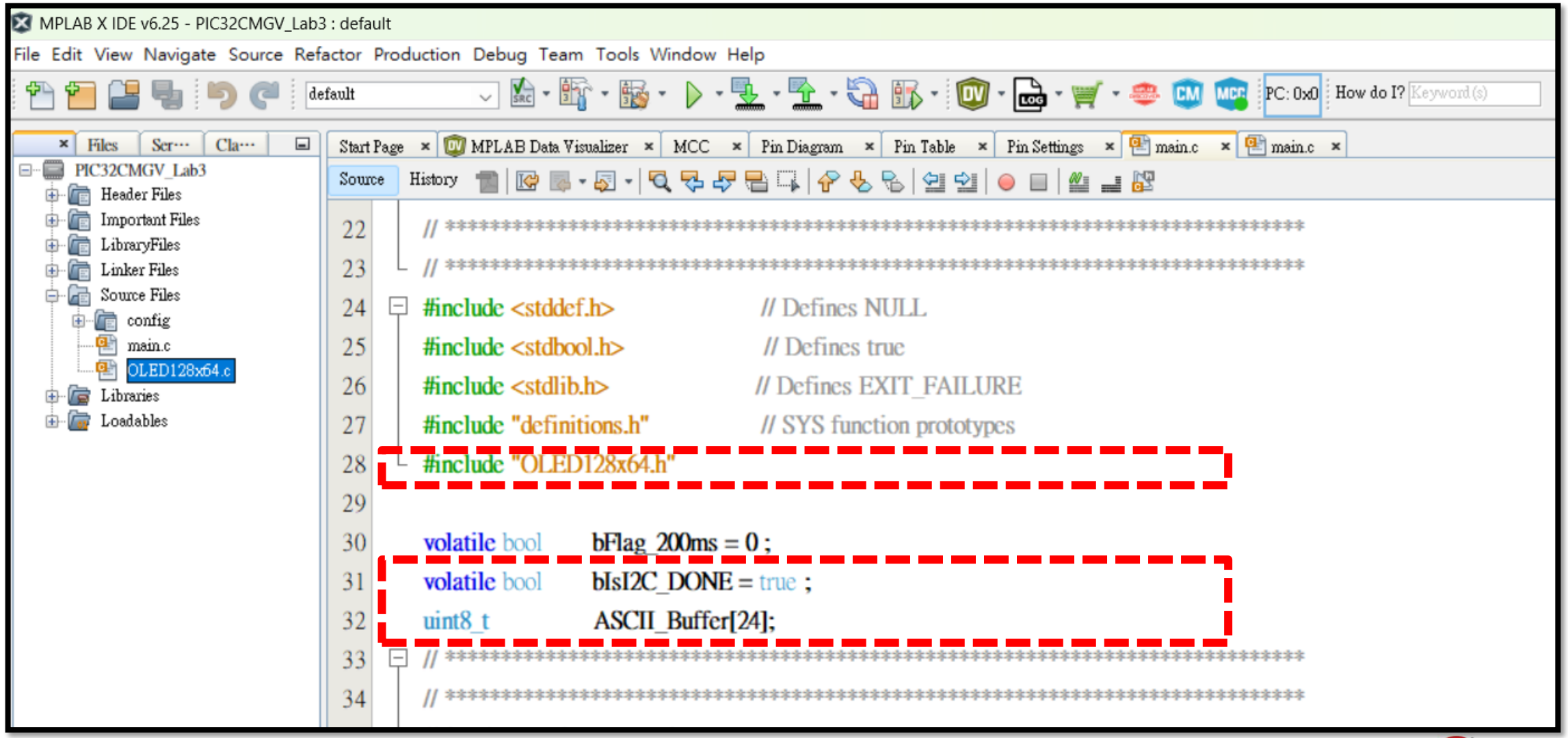
Copy all library files to \src folder and add OLED128x64.c to project



MICROCHIP

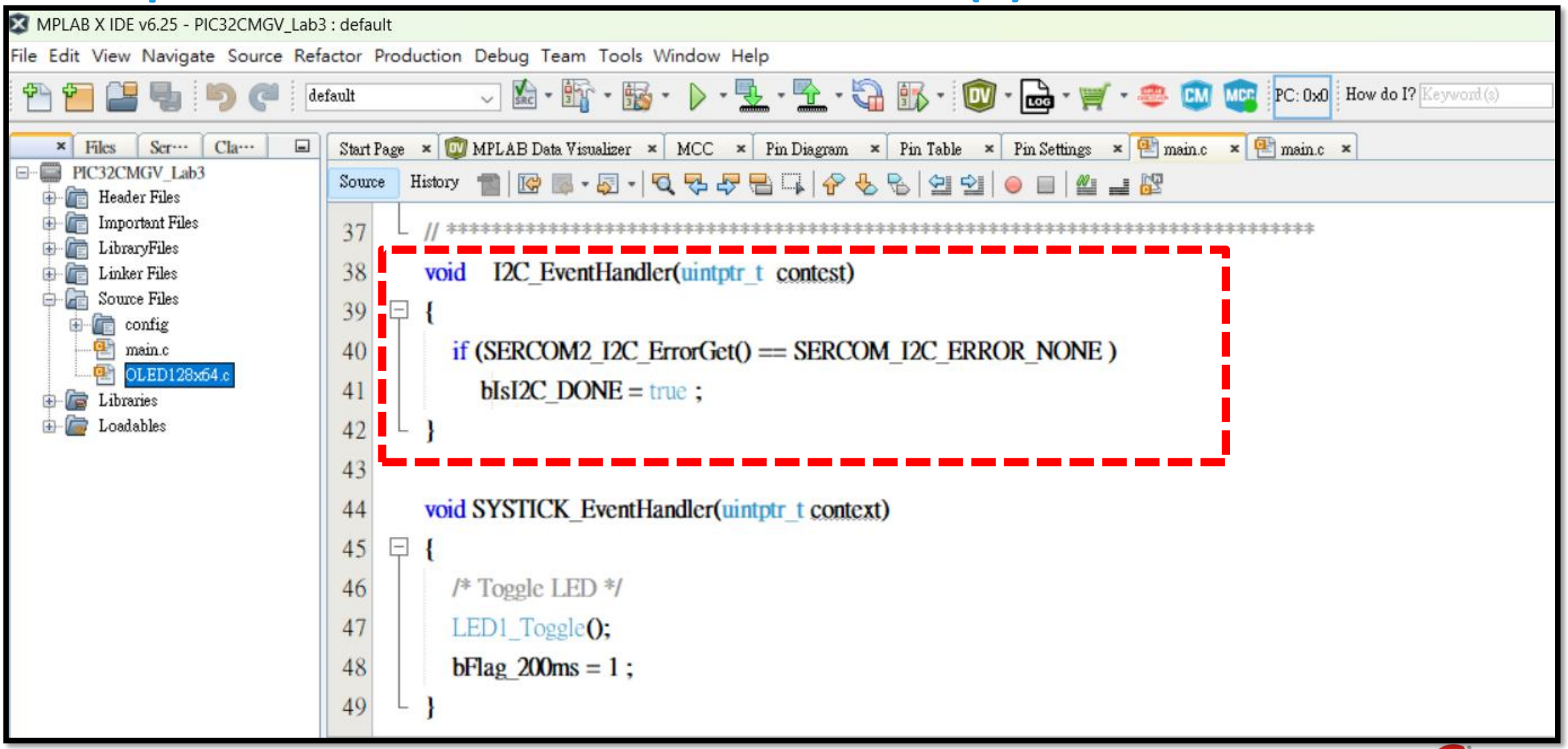
Lab 3 – OLED Display for ADC Result

Add required code and definition to main.c (1)



Lab 3 – OLED Display for ADC Result

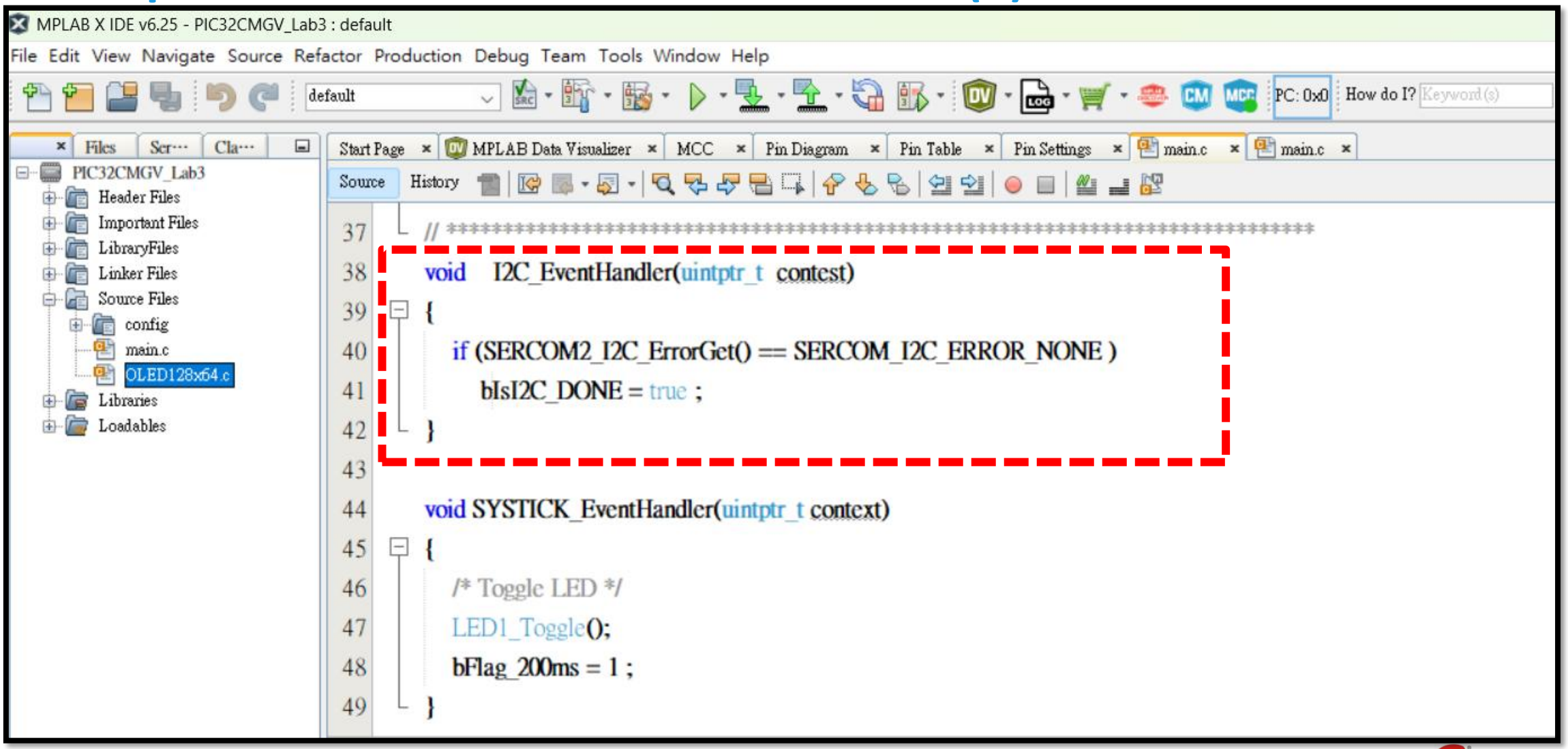
Add required code and definition to main.c (2)



MICROCHIP

Lab 3 – OLED Display for ADC Result

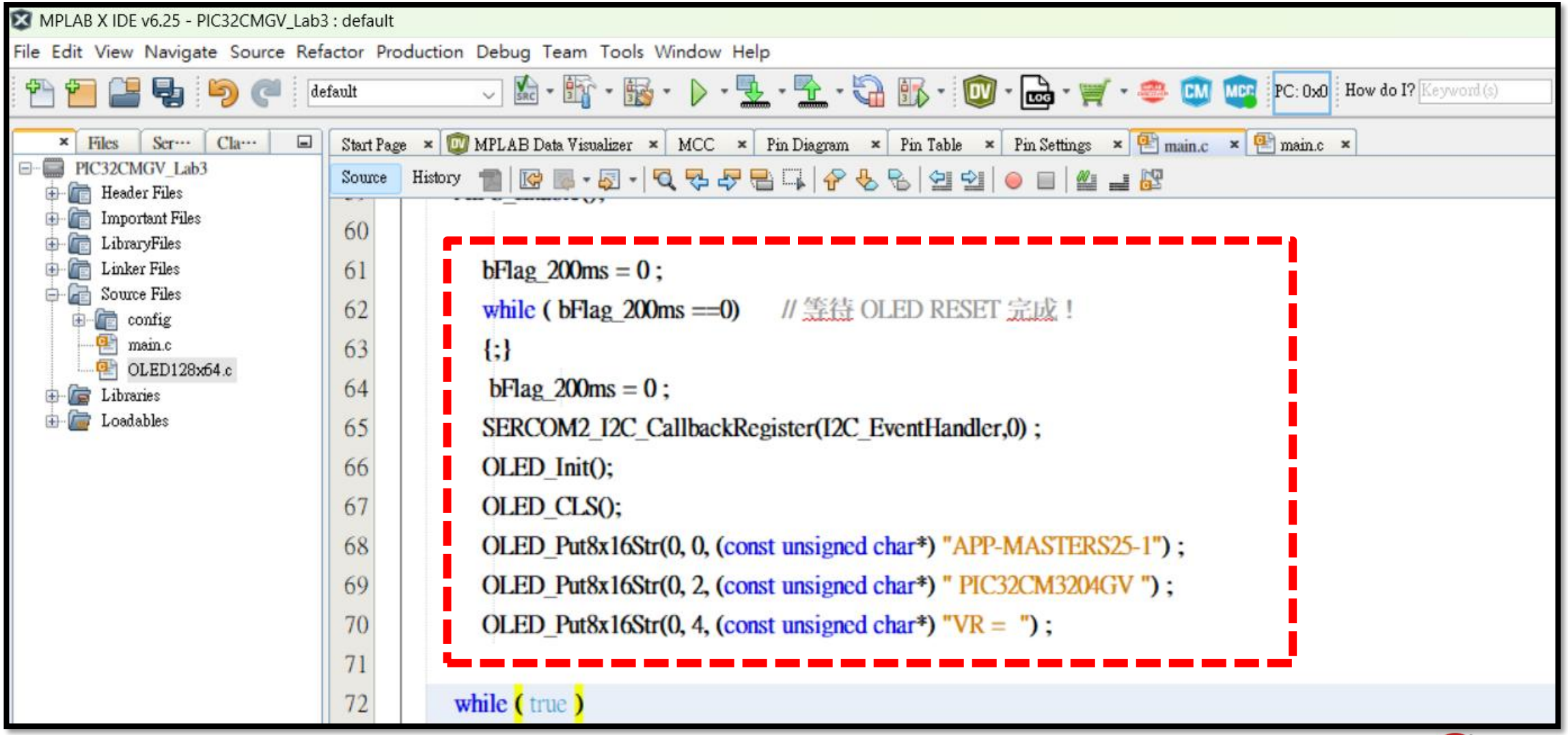
Add required code and definition to main.c (3)



MICROCHIP

Lab 3 – OLED Display for ADC Result

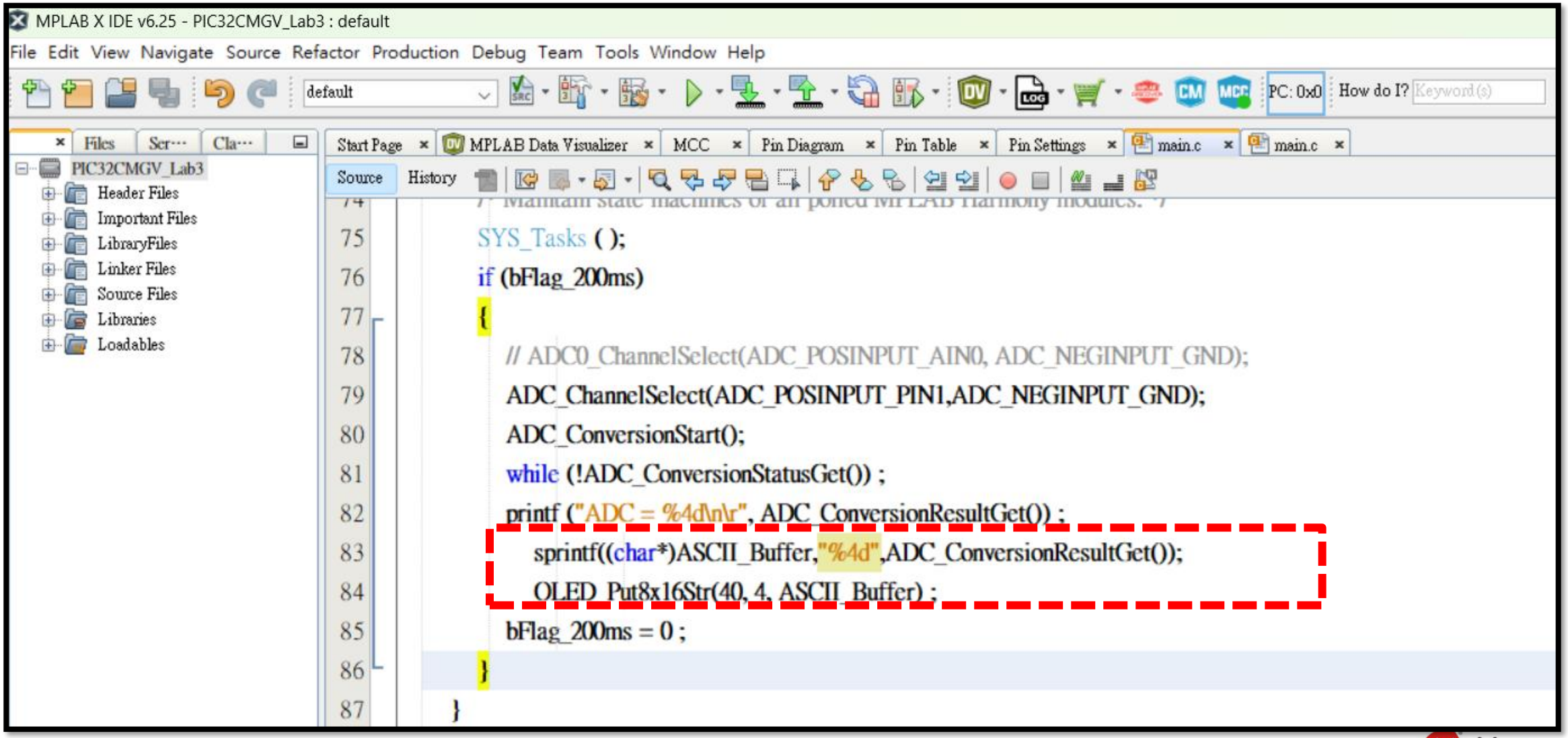
Add required code and definition to main.c (4)



MICROCHIP

Lab 3 – OLED Display for ADC Result

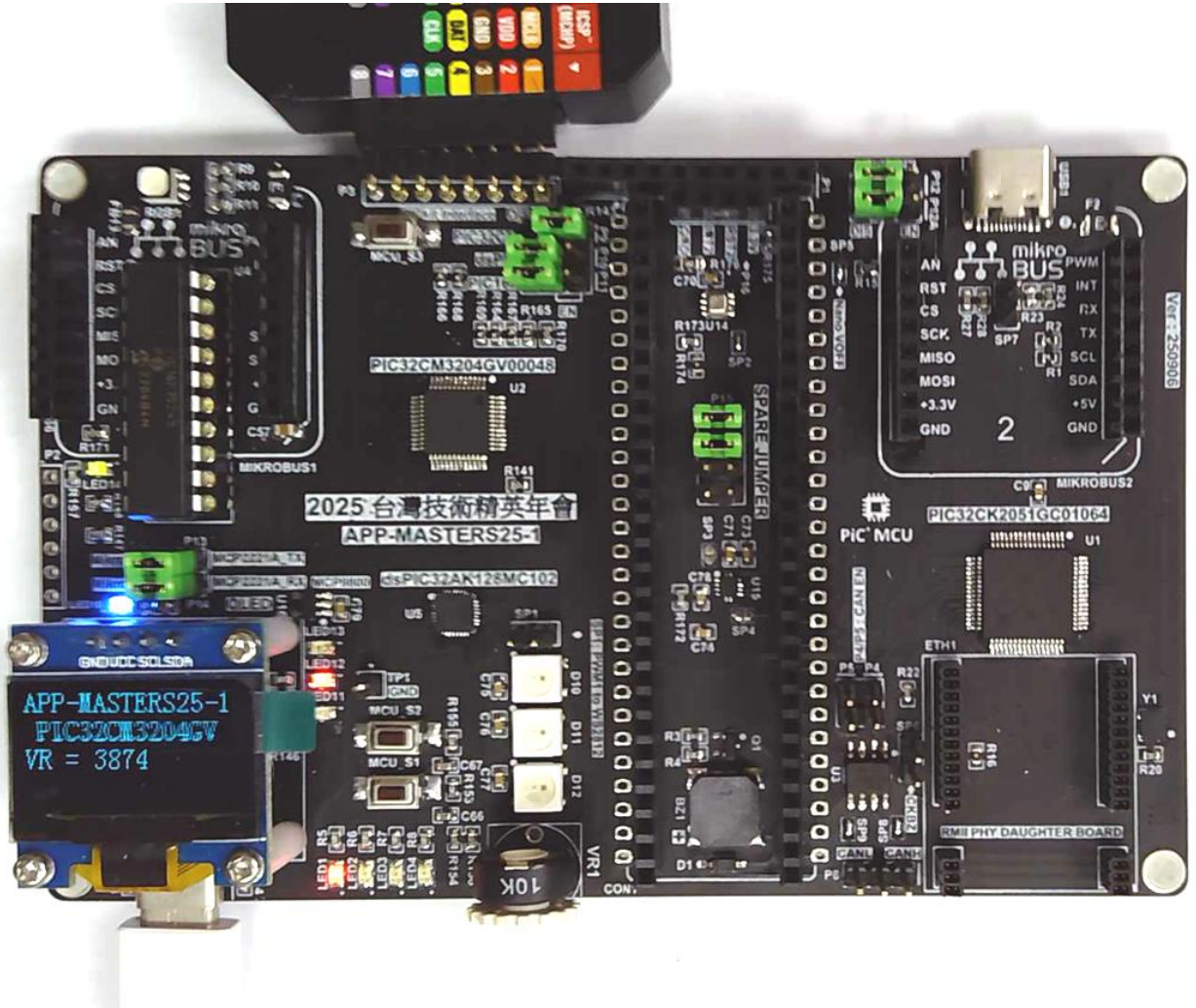
Add required code and definition to main.c (5)



MICROCHIP

Lab 3 – OLED Display for ADC Result

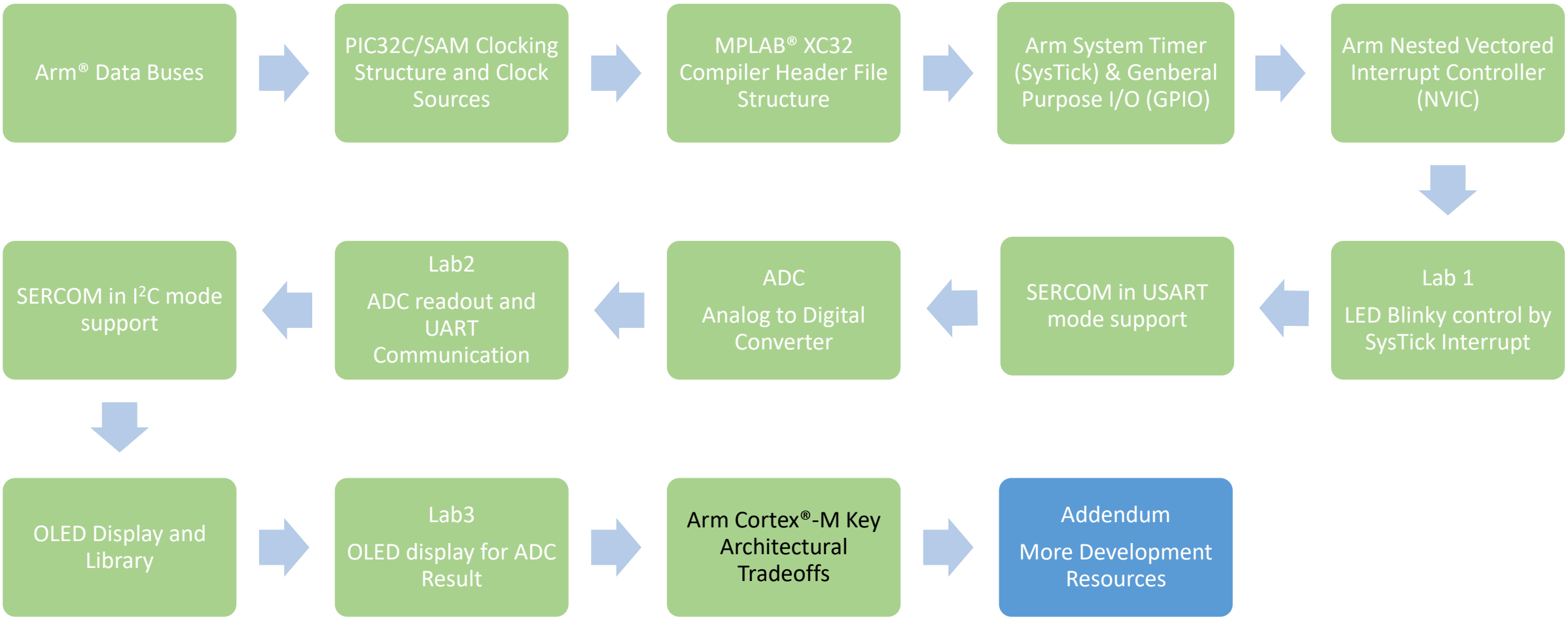
Lab3 實際執行結果



Lab 3 Summary

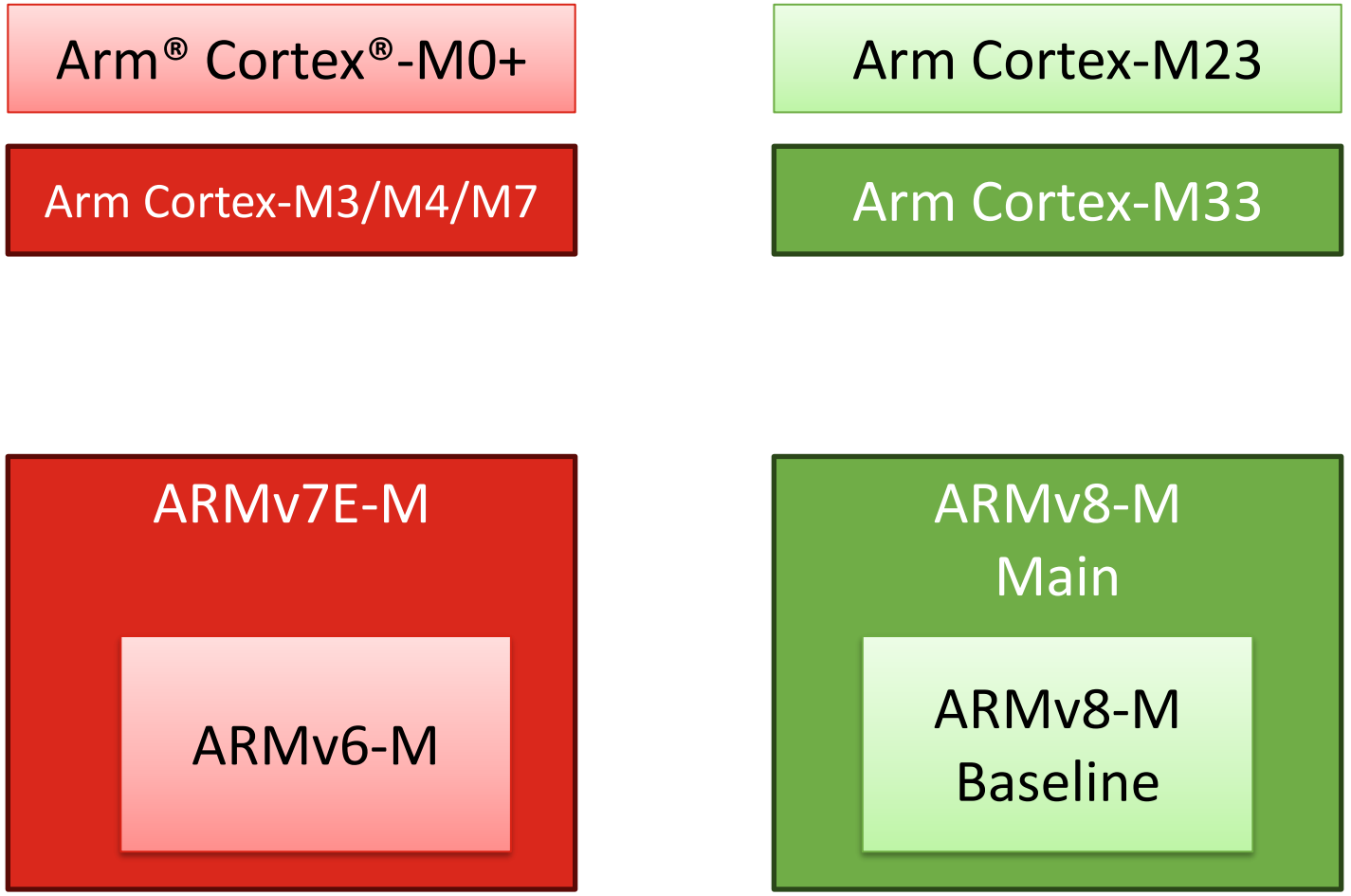
- Properly set up SERCOM to implement I^2C MASTER Function
- Add Library into your project
- Config OLED Display by using Library
- Display ADC read result to Line-3 of OLED

Class Outline



Arm® Cortex®-M23/M4/M33 Key Architectural Tradeoffs

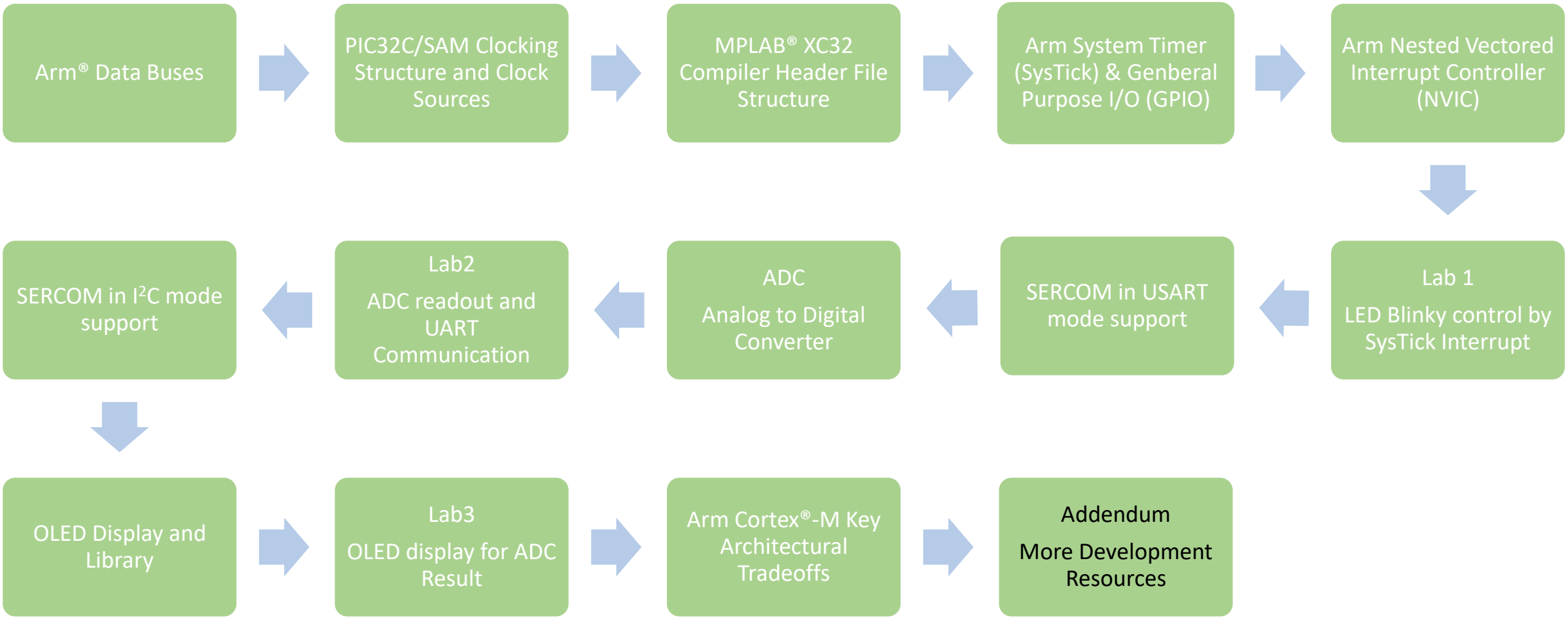
Instruction Set Architecture (ISA) Relationship



Key Architectural Tradeoffs

Arm® Cortex®-M Core	M0+	M23	M4F	M33	M7
Architecture	Von Neumann	Von Neumann	Harvard	Harvard	Harvard
ISA	ARMv6-M	ARMv8-M baseline	ARMv7E-M	ARMv8-M main	ARMv7E-M
Pipeline	2-stage	2-stage	3-stage	3-stage	6-stage
TrustZone®	No	Yes (option)	No	Yes (option)	Yes (option)
DSP Extension	No	No	Yes/SP-FPU	Yes/DP-FPU	Yes/DP-FPU
Hardware Divide	No	Yes	Yes	Yes	Yes
DMIPS/MHz	0.95	0.98	1.25	1.5	2.14
Max Interrupts	32	240	240	480	240
Bus Protocol	AHB Lite	AHB5	AHB Lite	AHB	AXI
Example Microchip Parts	PIC32CM JH/MC SAM C21/D21	PIC32CM Lx	PIC32CX SG/BZ SAM D5x/E5x	PIC32CK SG/GC	PIC32CZ CA SAM E70/V71

Class Outline



MICROCHIP

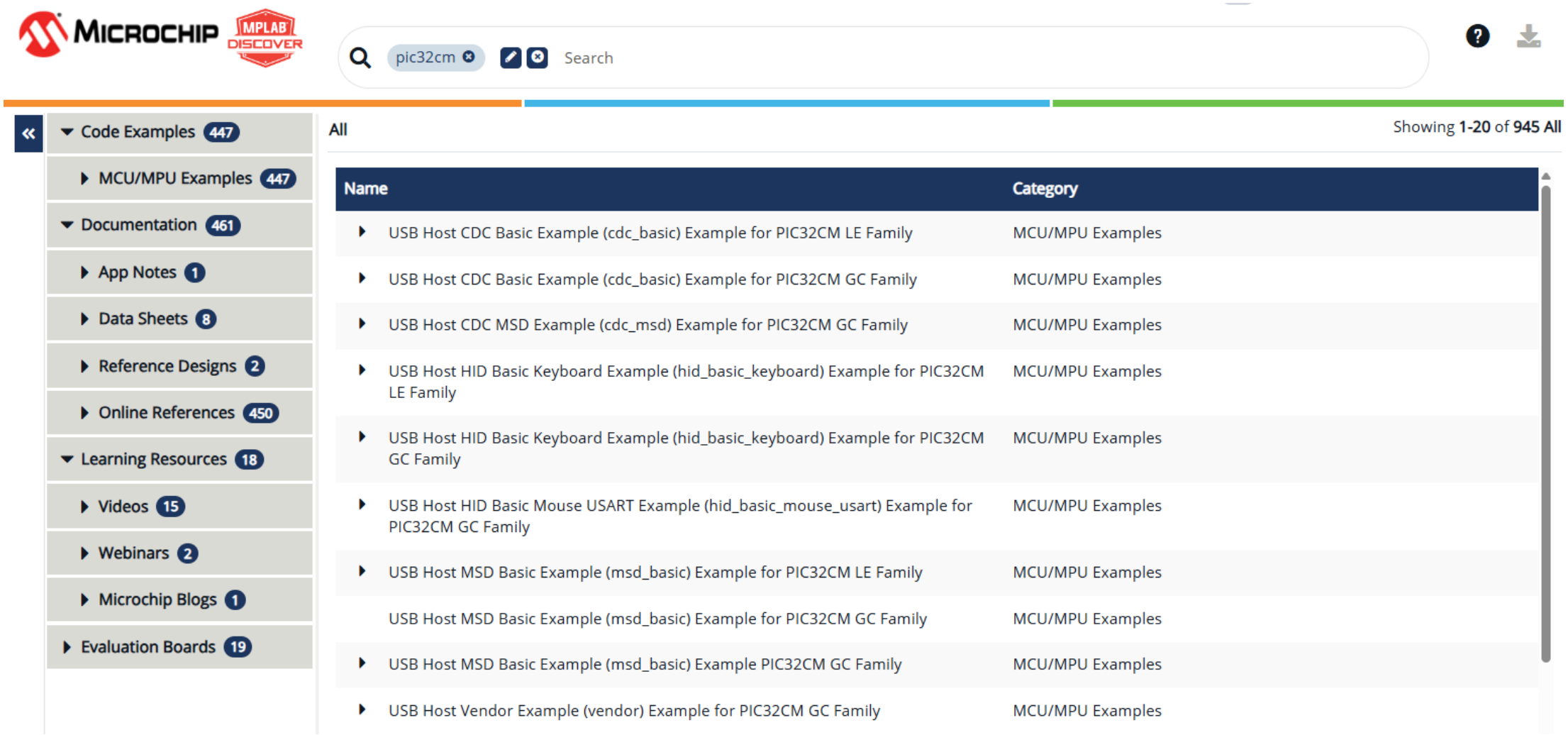
Class Summary

What's Next – Suggested MASTERs 2025 Classes

25005 MCU2	• Maximizing Microchip's Arm® Microcontroller Performance with the Direct Memory Access and Event System Peripherals
25009 MCU6	• Low-Power Design with Microchip's Arm Cortex® Microcontrollers: Essential Techniques
25023 FRM4	• Quick Start with MPLAB® Harmony for 32-bit MCUs/MPUs

What's Next – Code Examples and App Notes

discover.microchip.com



MICROCHIP

What's Next – Microchip University

mu.microchip.com

ARM® Cortex®-M Architecture Overview

This course covers the architectural differences of the Cortex-M cores and will help you select which device is the best choice for your own application. (Sept 2022)

142 min

Developing Bootloaders for ARM® Cortex® M0+ MCUs

This class will review the basics of firmware bootloaders and show you how to deploy a bootloader-enabled application on a SAM D21 MCU, using the MPLAB® X IDE and the MPLAB Harmony bootloader library. (October 2023)

86 min

Getting Started with Writing Code for the Microchip ARM® Cortex® Microcontrollers

Learn how to get started writing code and operating peripherals on the Microchip SAM ARM Cortex M0+, M23 and M4 microcontrollers. (Sept 2021)

184 min

Low-Power System Design with Microchip's 32-bit Cortex M0+/M23 ARM® Microcontrollers

Low power development fundamentals for the 32-bit ATSAML10 MCU with power measurement demonstrations using the SAML10-Xplained Pro Demo Board and Data Visualizer. (Sept 2021)

126 min

The End

LEGAL NOTICE

MASTERs 2025

SOFTWARE:

Microchip software provided to you during the MASTERs 2025 conference is provided "AS IS", without warranty. Microchip assumes no liability or responsibility for your use of any Microchip software. You must use Microchip software exclusively with Microchip products. Further, use of Microchip software is subject to copyright and the disclaimers and license terms accompanying such software, whether set forth at the install of each program or posted in a header or text file.

Notwithstanding the above, certain components of software offered by Microchip and 3rd parties may be covered by "open source" software licenses, including licenses that require that the distributor of such software make the software available in source code format. To the extent required by any such open-source software license, the terms of such license govern.

Any transfer of Microchip product, technology, or software may be subject to export controls. Microchip encourages any entity transferring product, technology, or software to seek appropriate legal advice and/or consult the applicable governmental agencies prior to transfer. The information provided herein is subject to change at any time and without notice.

NOTICE & DISCLAIMER:

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

MICROCHIP DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, OR STATUTORY, INCLUDING ANY IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY DIRECT OR INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND RELATED TO THESE MATERIALS OR ACCOMPANYING INFORMATION PROVIDED TO YOU BY MICROCHIP OR OTHER THIRD PARTIES, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBLITY OF SUCH DAMAGES OR THE DAMAGES ARE FORESEEABLE. PLEASE BE AWARE THAT THE IMPLEMENTATION OF INTELLECTUAL PROPERTY PRESENTED IN MASTERs 2025 MAY REQUIRE A LICENSE FROM MICROCHIP OR THIRD PARTIES.

TRADEMARKS:

The Microchip name and logo and the Microchip logo are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries. Information on and a listing of Microchip trademarks is available at https://www.microchip.com/en-us/about/legal-information/microchip-trademarks.

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

COPYRIGHT: