



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

Presented By: Calvin Ho
25004-MCU1



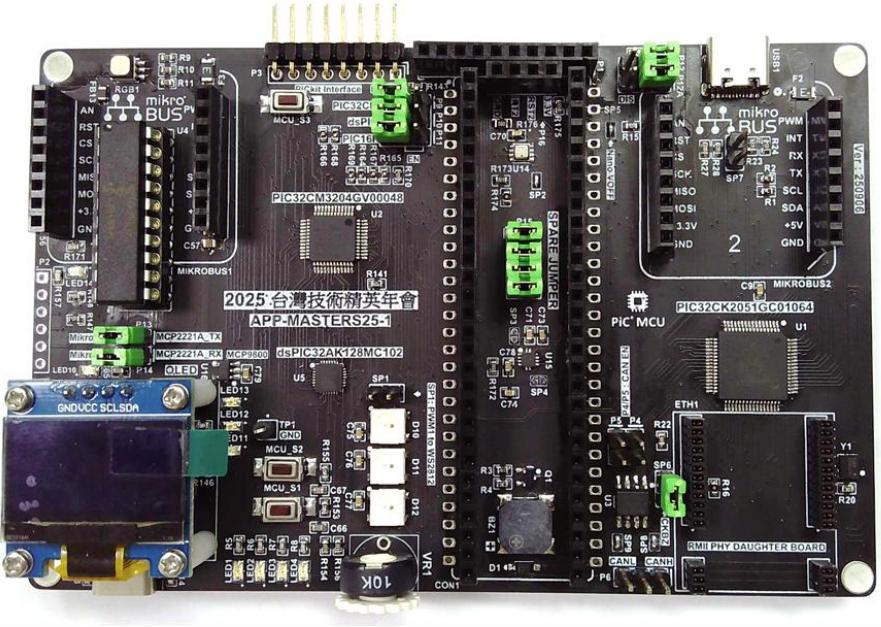
Class Objectives

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





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

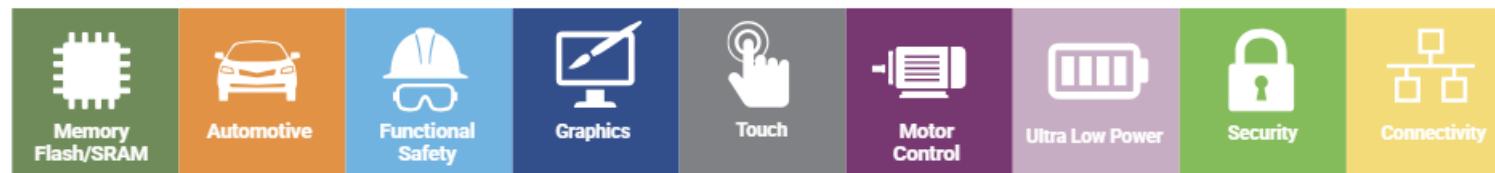
PIC32A MCUs

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.

Legacy 32-bit MCUs With Arm® Cortex® Cores

	Memory Flash / SRAM	Automotive	Connectivity	Functional Safety	Graphics	Motor Control	Security	Touch	Ultra-Low Power
Performance ↑	SAM V7x Arm® Cortex®-M7, 300 MHz	512–2048 KB/ 256–384 KB	●						
	SAM E7x Arm Cortex-M7, 300 MHz	512–2048 KB/ 256–384 KB		●	●	●	●	●	
	SAM S7x Arm Cortex-M7, 300 MHz	512–2048 KB/ 256–384 KB		●	●	●	●	●	
	SAM E5x Arm Cortex-M4F, 120 MHz	256–1024 KB/ 128–256 KB	●	●	●	●	●	●	
	SAM D5x Arm Cortex-M4F, 120 MHz	256–1024 KB/ 128–256 KB	●	●	●	●	●	●	
	SAM G Arm Cortex-M4F, 120 MHz	256–512 KB/ 64–176 KB							●
	SAM 4 Arm Cortex-M4F, 48–120 MHz	128–2048 KB/ 32–160 KB		SAM 4E/4S					SAM 4L
	SAM D Arm Cortex-M0+, 48 MHz	8–256 KB/ 2–32 KB	●	●	●	●	●	●	
	SAM C Arm Cortex-M0+, 48–64 MHz	32–256 KB/ 4–32 KB	●	●	●	●	●	●	
	SAM L21/L22 Arm Cortex-M0+, 32–48 MHz	32–256 KB/ 4–40 KB		●			●	●	●
	SAM L10/L11 Arm Cortex M-23, 32 MHz	16–64 KB/ 4–16 KB	●				●	●	●

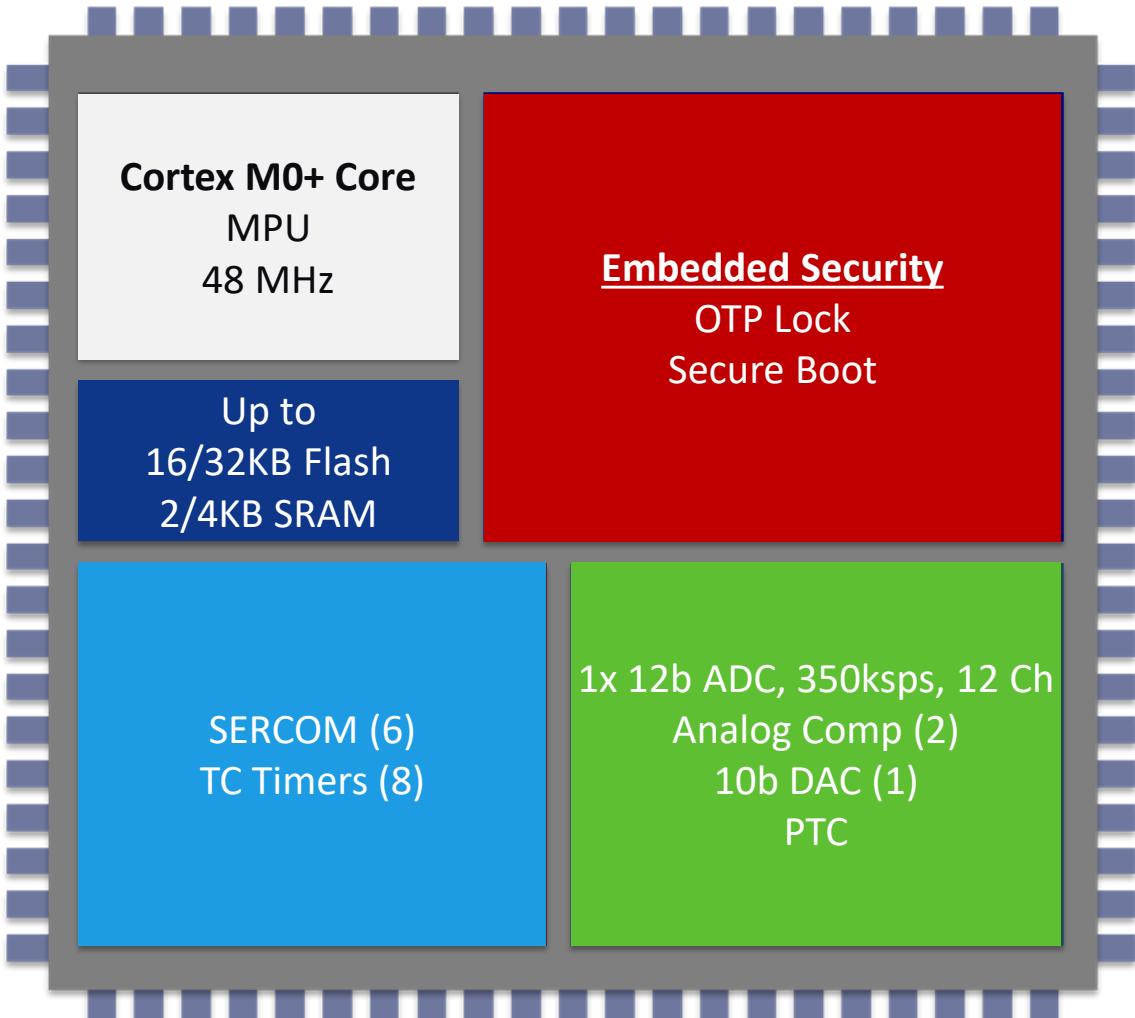
New 32-bit MCUs With Arm® Cortex® Cores



PIC32CZ CA Arm® Cortex® -M7, 300 MHz	2048-8192 KB/ 512-1024 KB	●	●	●	●		●	●
PIC32CK Arm® Cortex® -M33, 120 MHz	1024-2048 KB/ 256-512 KB	●	●		●		●	●
PIC32CX SG Arm® Cortex® -M4F, 120 MHz	1024 KB 256 KB	●	●	●	●	●	●	●
PIC32CM JH Arm® Cortex® -M0+, 48 MHz	128-512 KB/ 16-64 KB	●	●		●	●	●	●
PIC32CM LX Arm® Cortex® -M23, 48 MHz	128-512 KB/ 16-64 KB				●		●	●
PIC32CM MC Arm® Cortex® -M0+, 48 MHz	64-128 KB 8-16 KB		●		●			
PIC32CM GV00 Arm® Cortex® -M0+, 48 MHz	16-32 KB/ 2-4 KB			●				
PIC32CM JH00 Arm® Cortex® -M0+, 48 MHz	32-64 KB/ 4-8 KB			●				

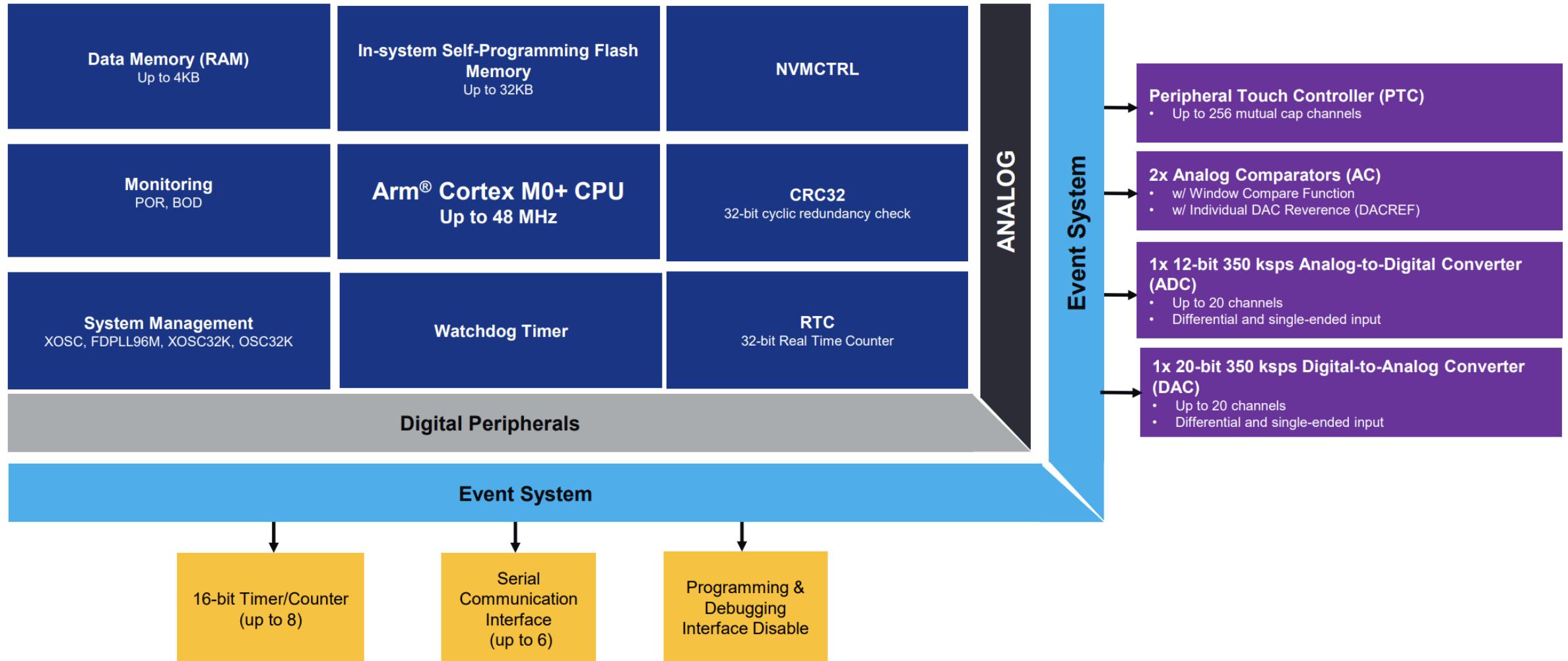
PIC32CM GV – Value Line Cortex-M0+

Low-Cost Pin compatible SAMD2x



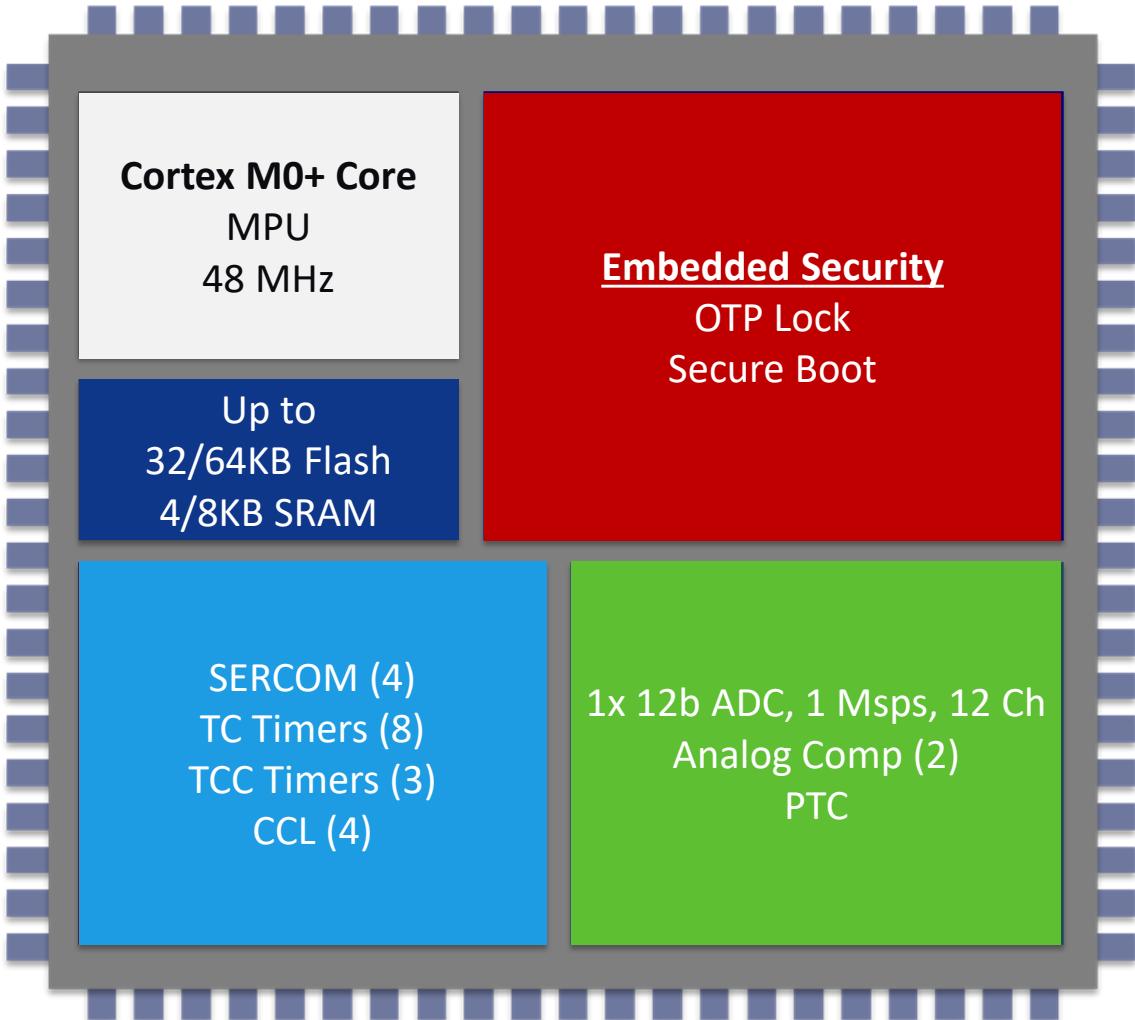
- **New 3.3V Value Line Series:**
 - Pin and binary compatible with SAMD20
 - Up to 50% savings compared to SAMD20
 - New Curiosity Nano Board
 - ASIL-B Compliant Hardware Design
- **Key Features:**
 - Single, multi-channel 12B ADC
 - Secure Boot function via OTP Lock
 - One Time Programmable Option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

PIC32CM GV00 Block Diagram



PIC32CM JH00 5V Cortex-M0+ Family

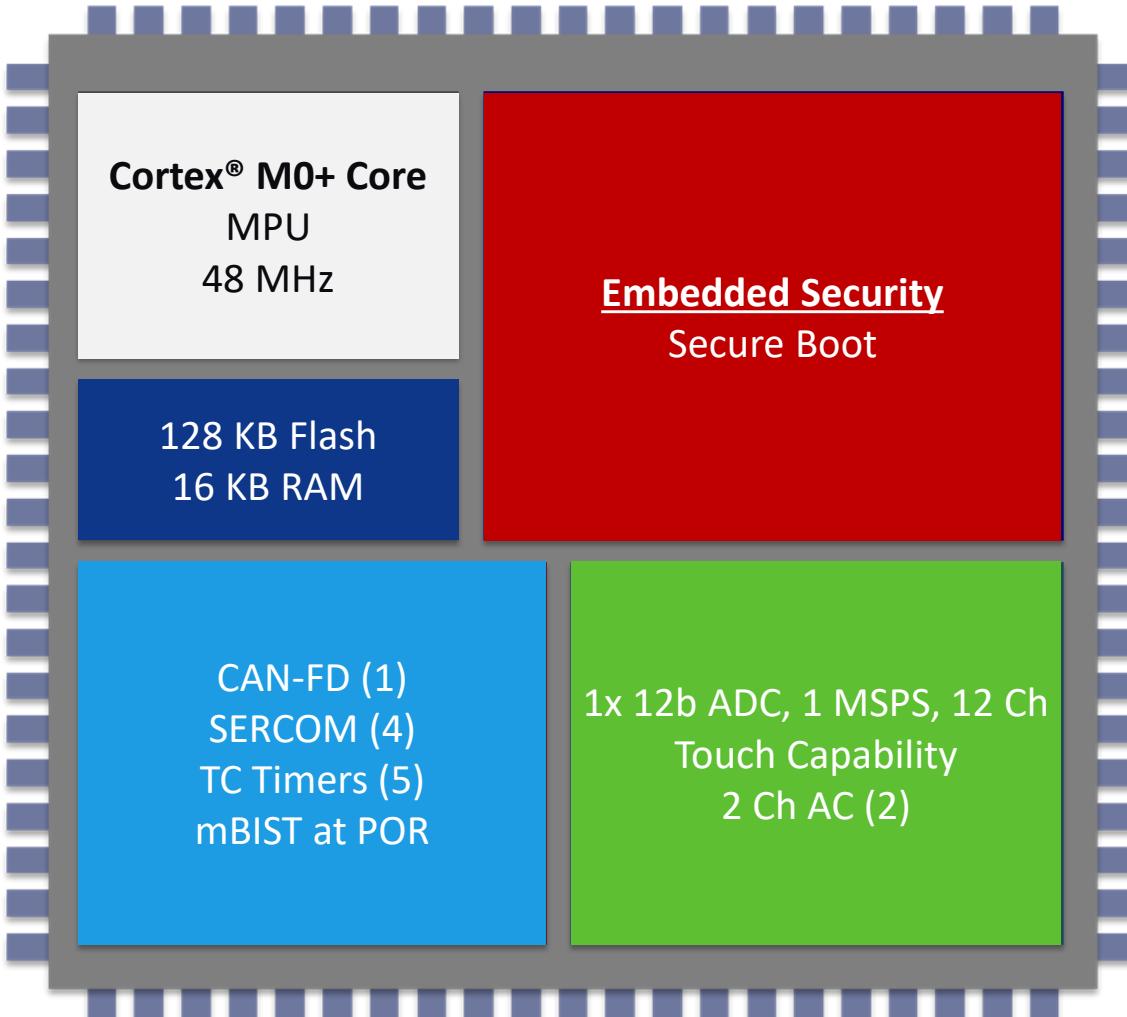
New Low Cost 64KB flash – Pin compatible SAMC20



- **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**

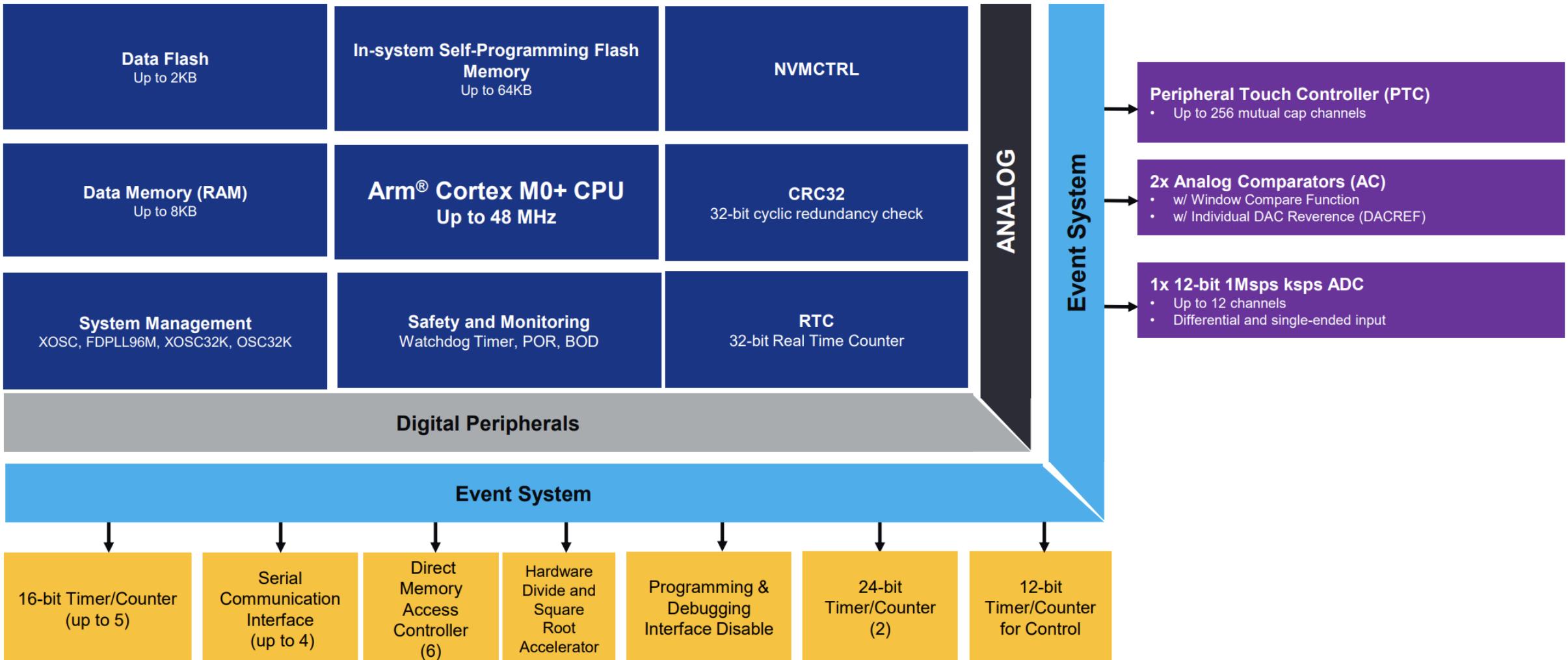
PIC32CM JH01 5V Cortex-M0+ Family

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**

PIC32CM JH00 Block Diagram





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

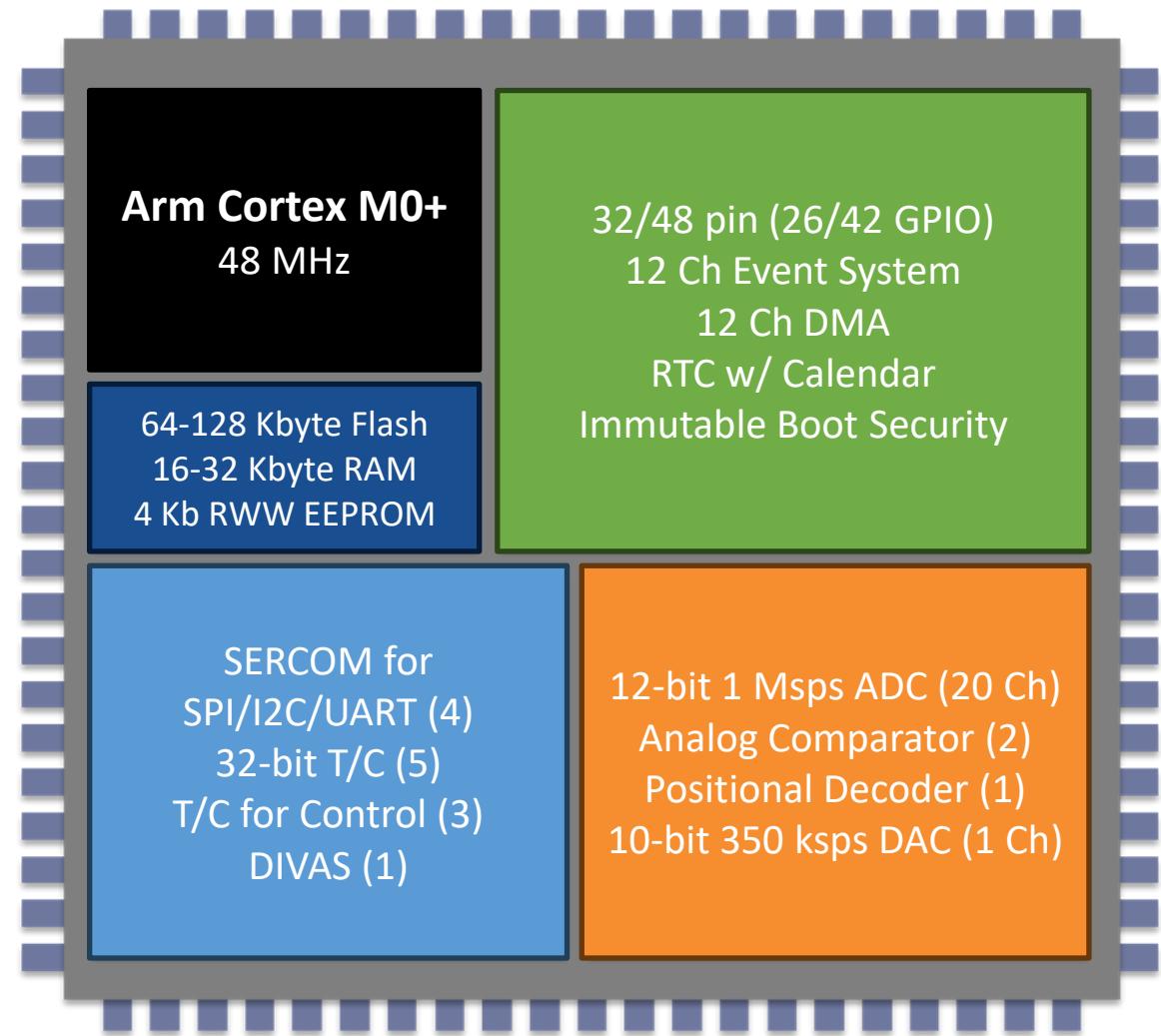
*unlock only after a full device erase

CP = Code protection

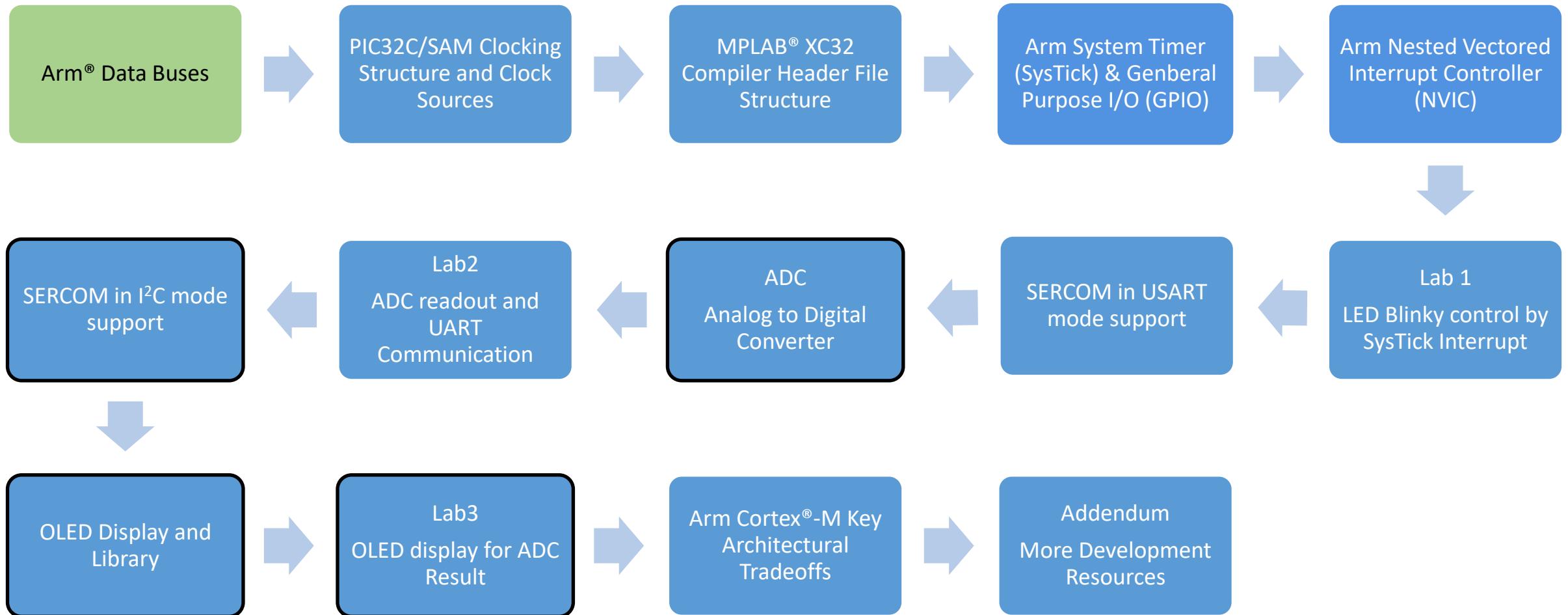
CPD = Code Protection of Data EEPROM

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



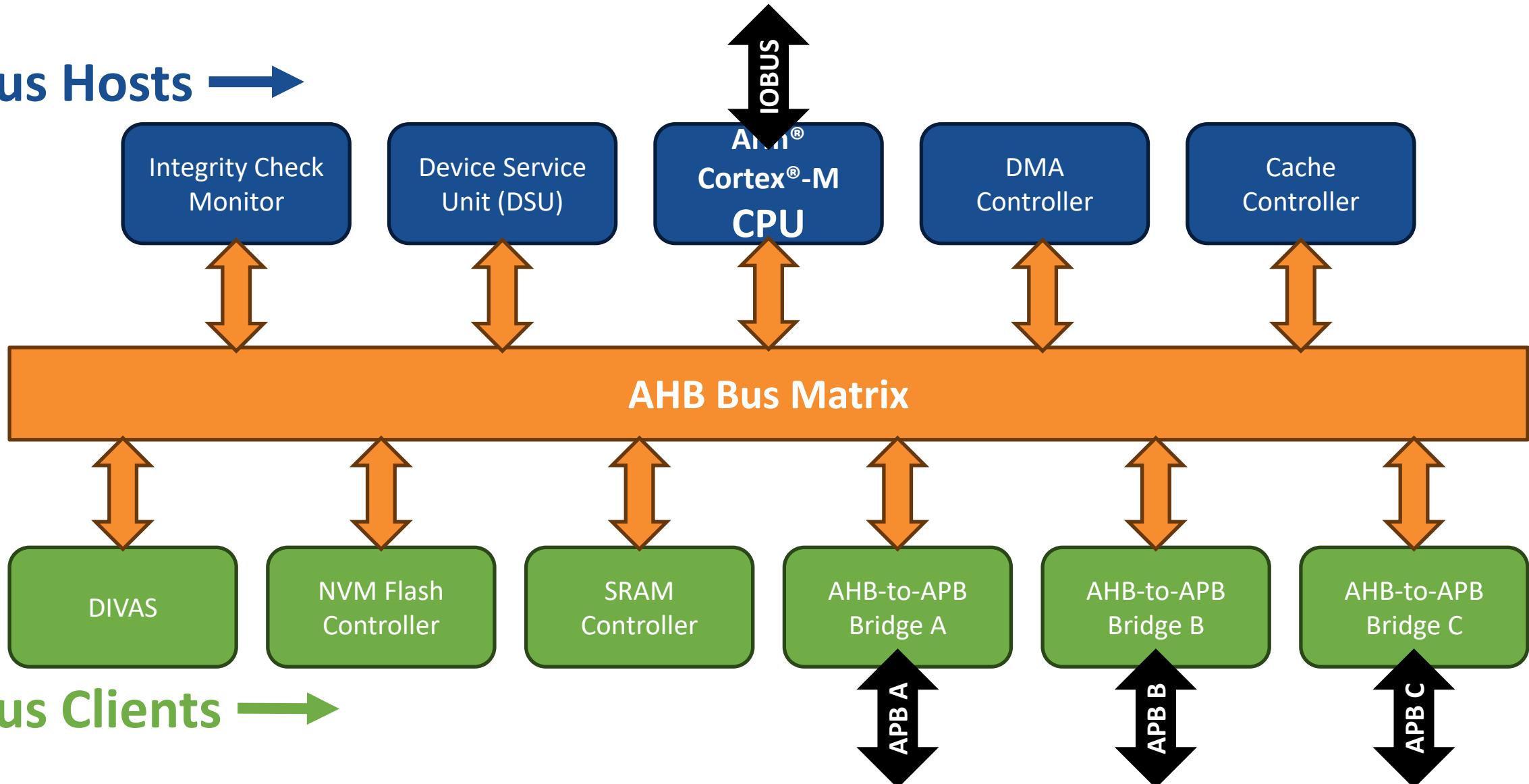


Arm® Data Buses

Advanced High-Performance Bus (AHB)
Advanced Peripheral Bus (APB)
Single Cycle I/O Bus (IOPBUS)

Advanced High-Performance Bus (AHB)

Bus Hosts →



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)

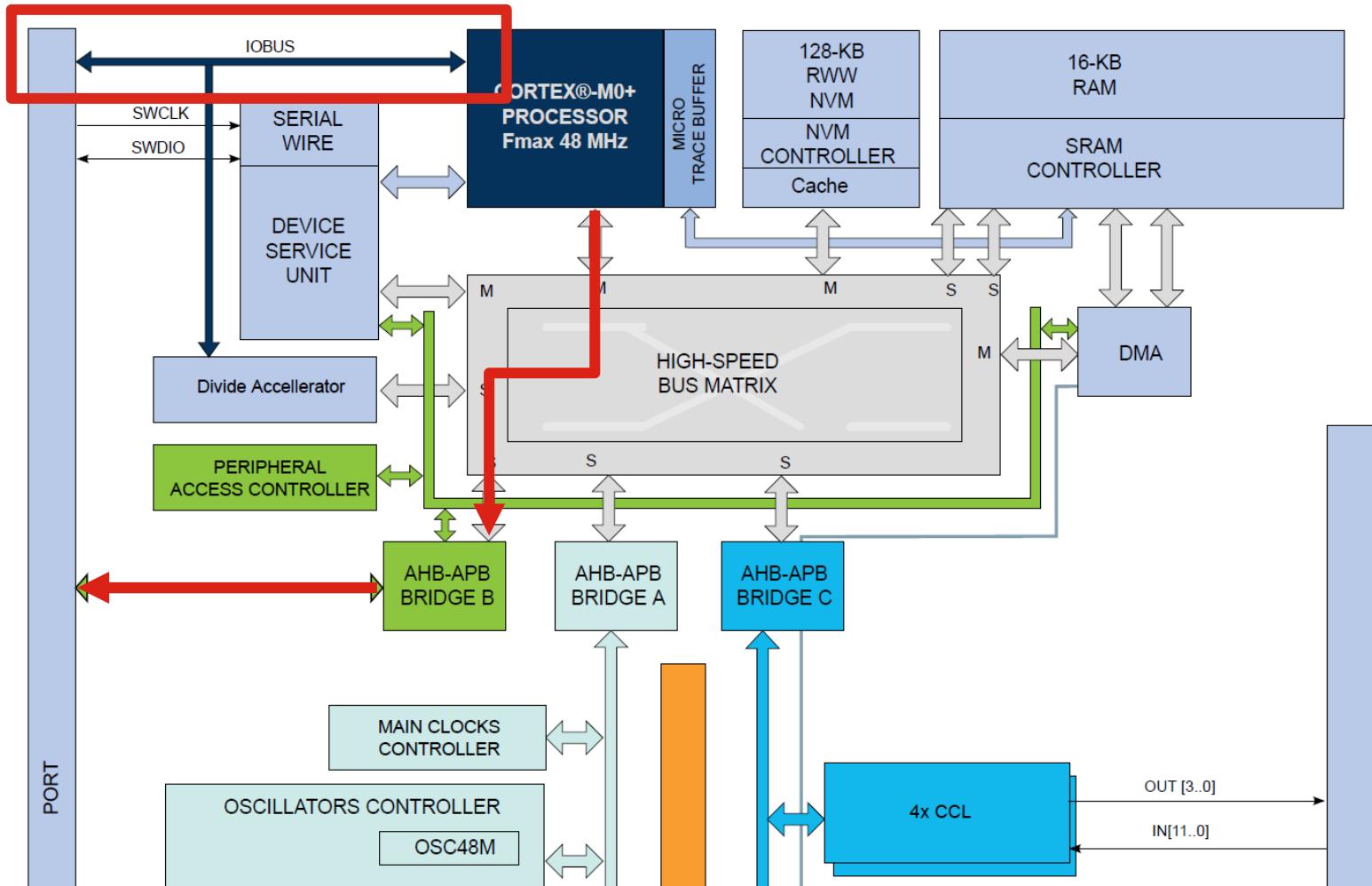
Available on Arm® Cortex®-M0+ and Cortex-M23

Separate bus interface

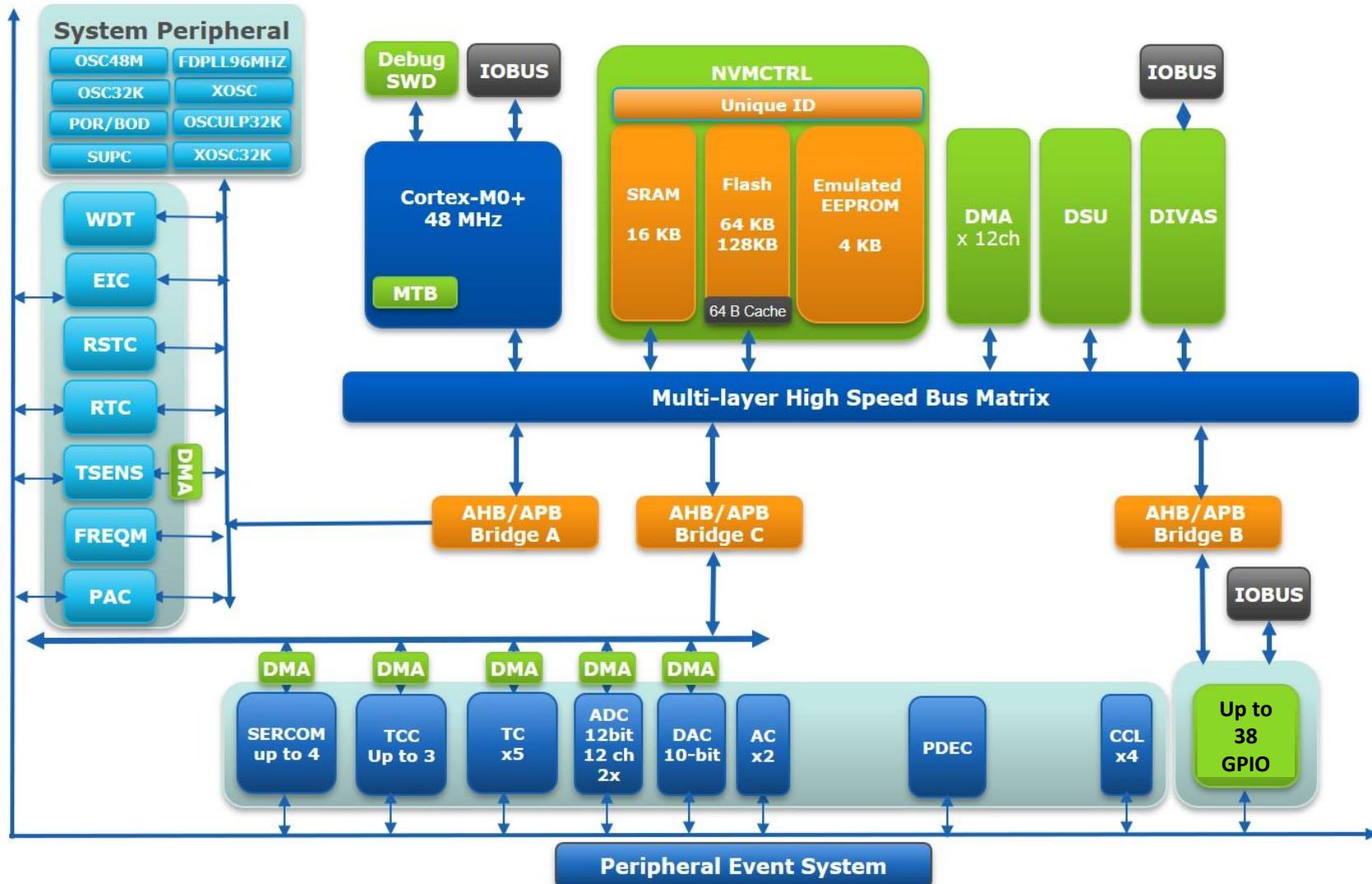
GPIO access without bus transfer

Memory mapped to a separate region

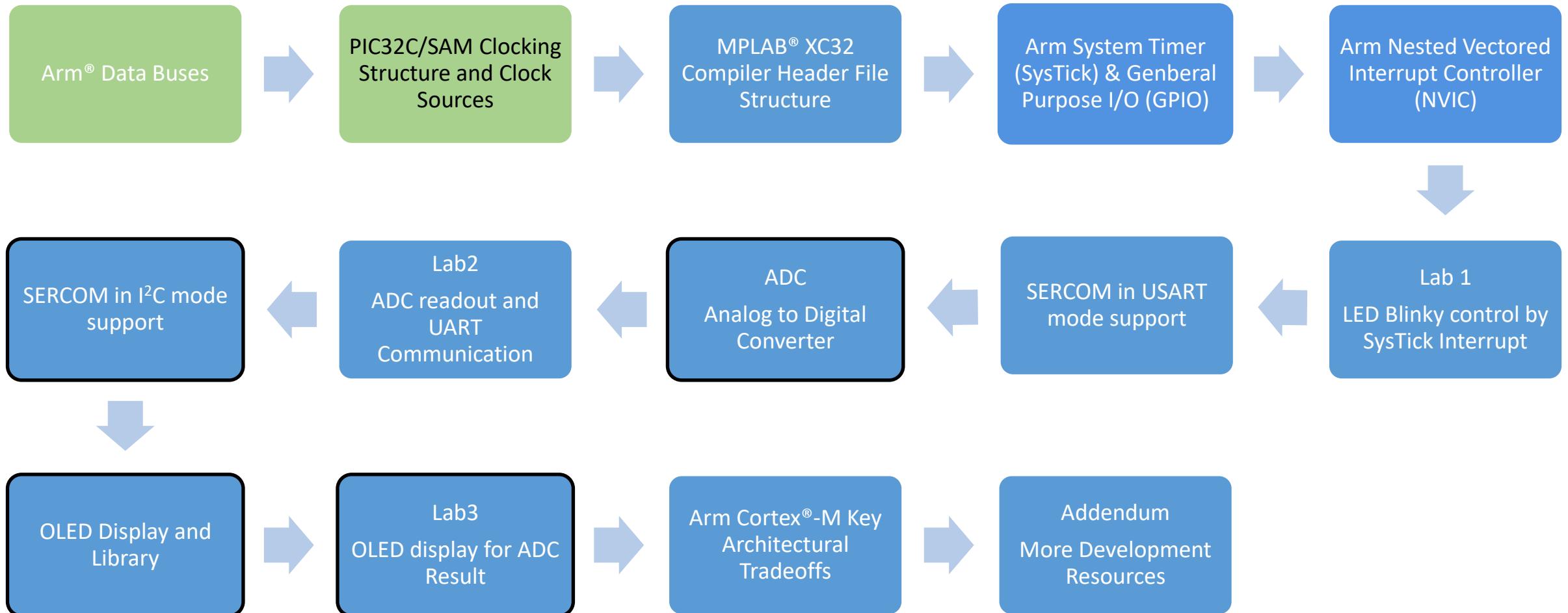
No resynchronization required



Bus Layout in PIC32CM MC00 Family



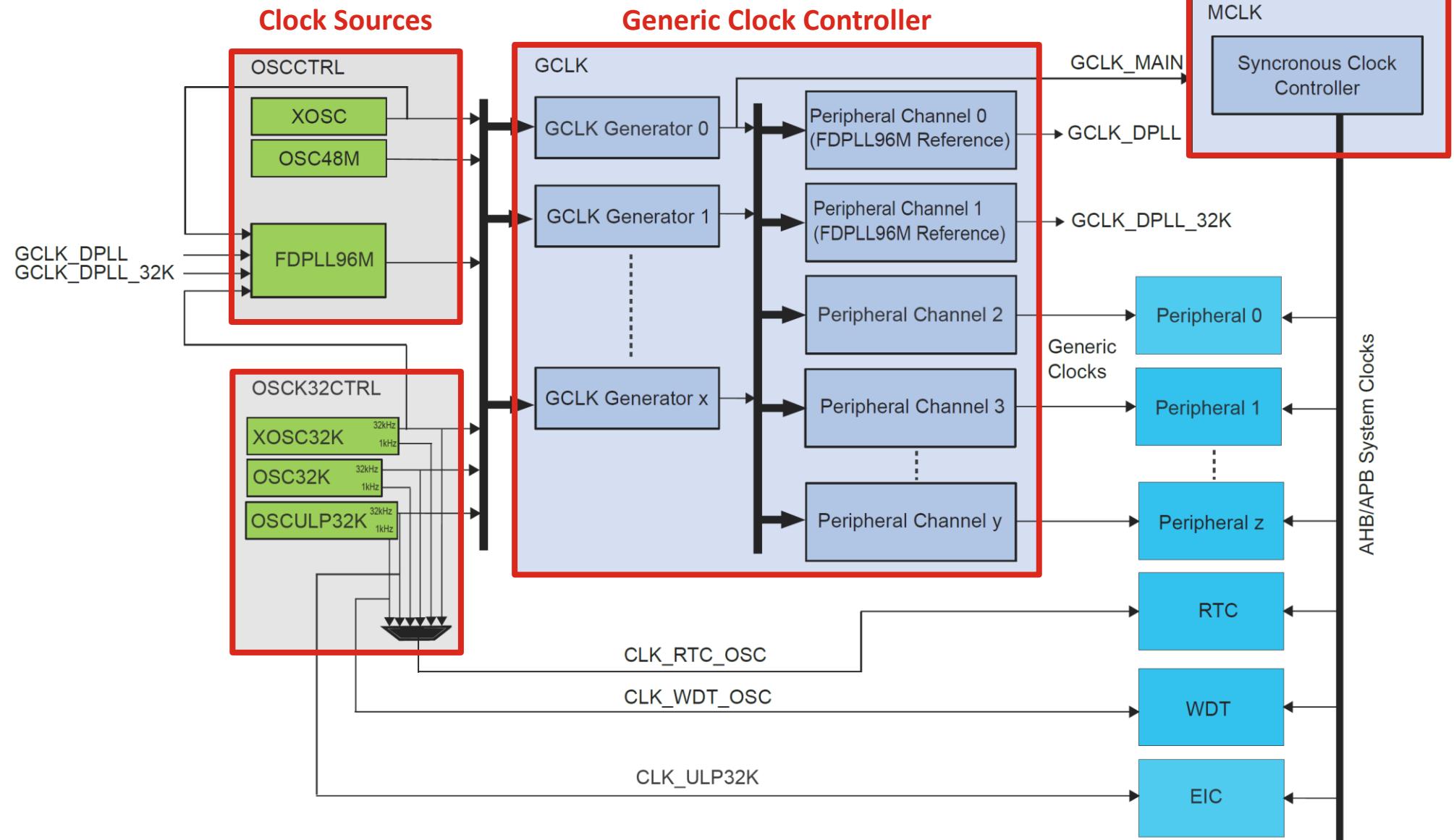
Class Outline





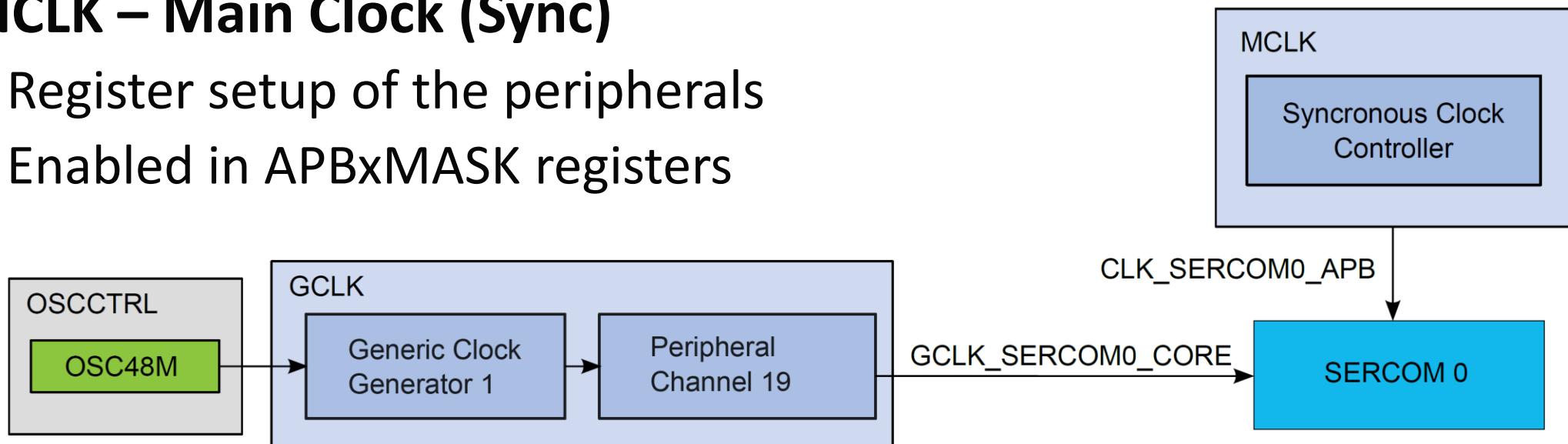
PIC32C and SAM Clock Structure

Clock System

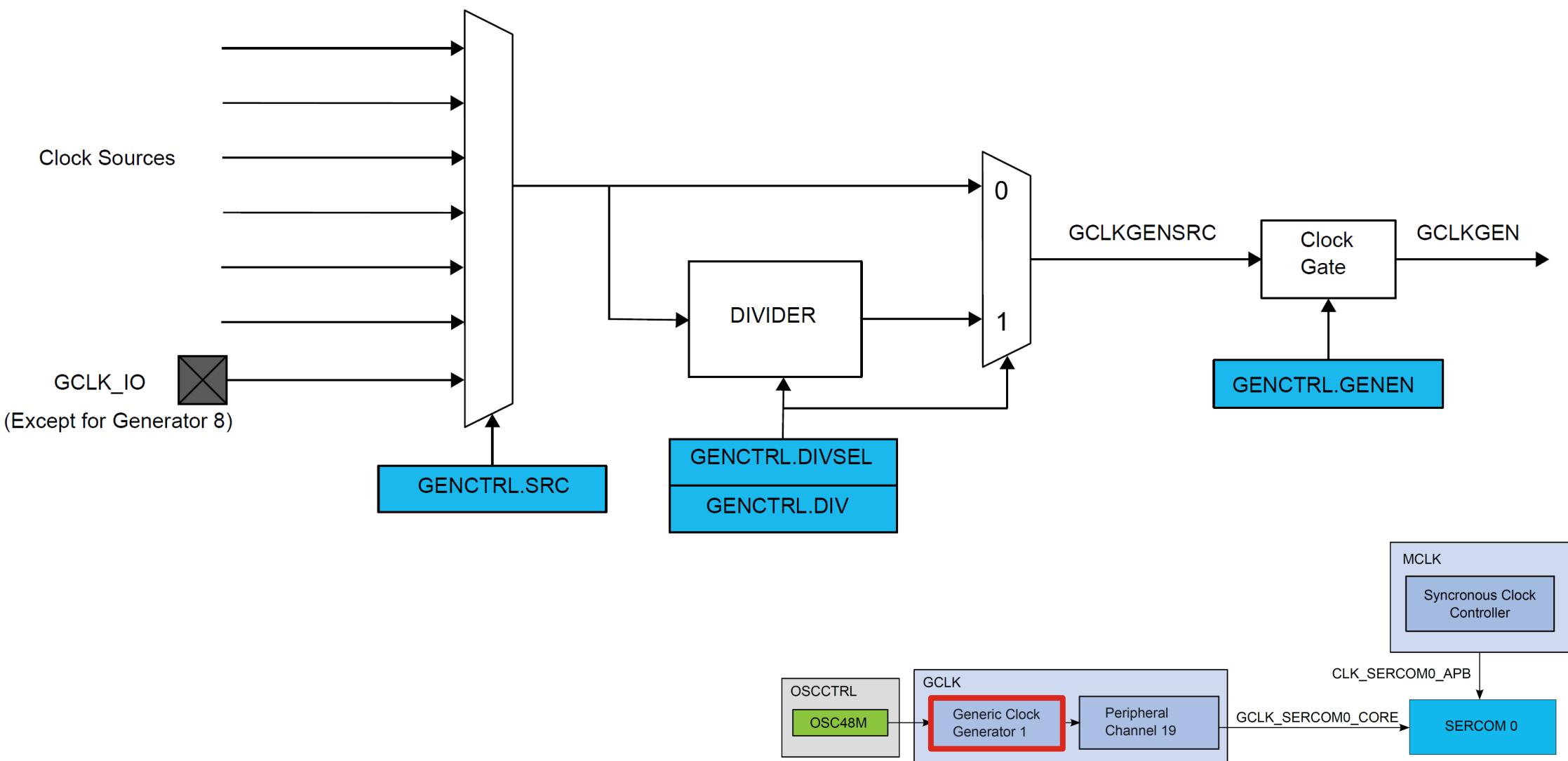


Clock Bus Domains and Peripheral Clock Setup

- **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

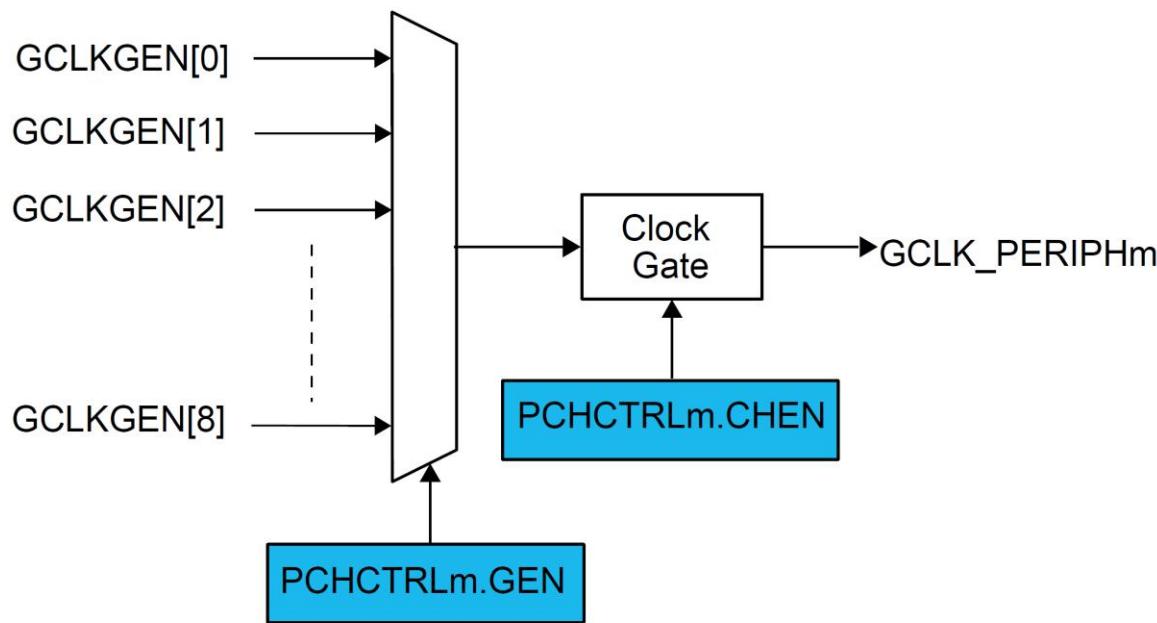


GCLK Generator Setup – GENCTRLn Register

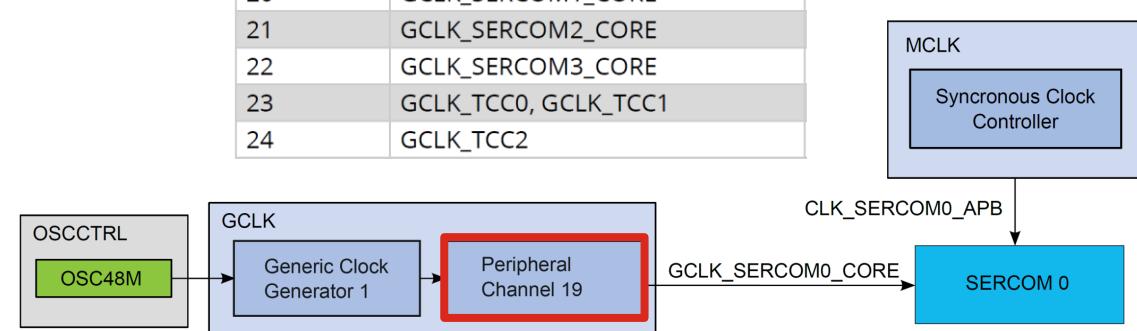


GCLK Peripheral Channel Setup – PCHCTRLm Register

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
Access						PDEC		
Reset						R/W		
						0		
Bit	23	22	21	20	19	18	17	16
Access	CCL		DAC	AC	SDADC	ADC1	ADC0	TC4
Reset	R/W		R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Access	TC3	TC2	TC1	TC0	TCC2	TCC1	TCC0	
Reset	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	
Bit	7	6	5	4	3	2	1	0
Access			SERCOM3	SERCOM2	SERCOM1	SERCOM0	EVSYS	
Reset			R/W	R/W	R/W	R/W	R/W	
			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

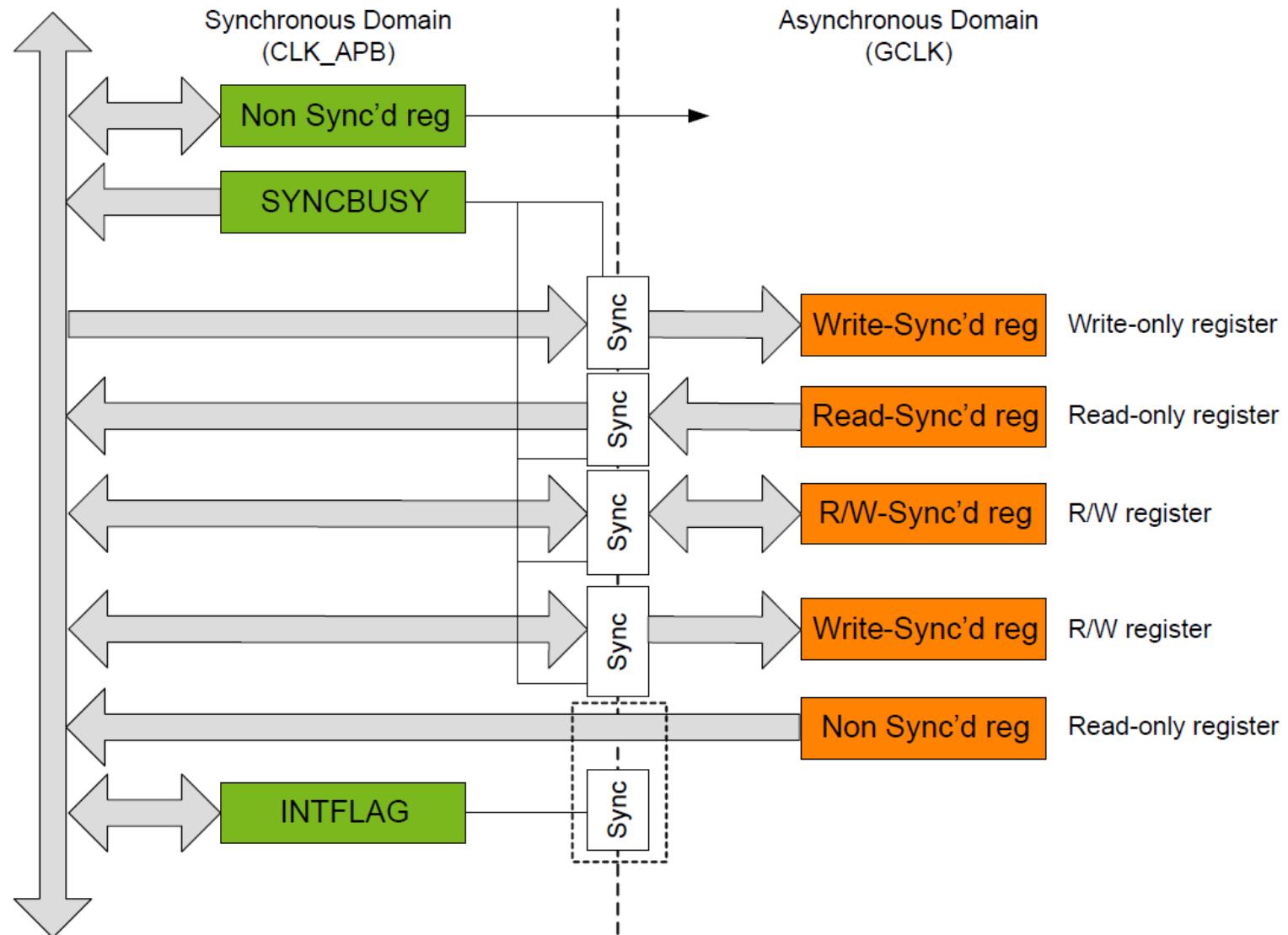
Value	Description
0	The APBC clock for the SDADC is stopped.
1	The APBC clock for the SDADC is enabled.

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				CLEAR[7:0]				
0x0C	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

Oscillator Control Blocks

Osc Controller (OSCCTRL)

XOSC
(External)

OSC48M/DFLL48M
(Internal)

FDPLL96M

32kHz Osc Controller (OSC32KCTRL)

XOSC32K
(External)

OSC32K
(Internal)

OSCULP32K

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

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

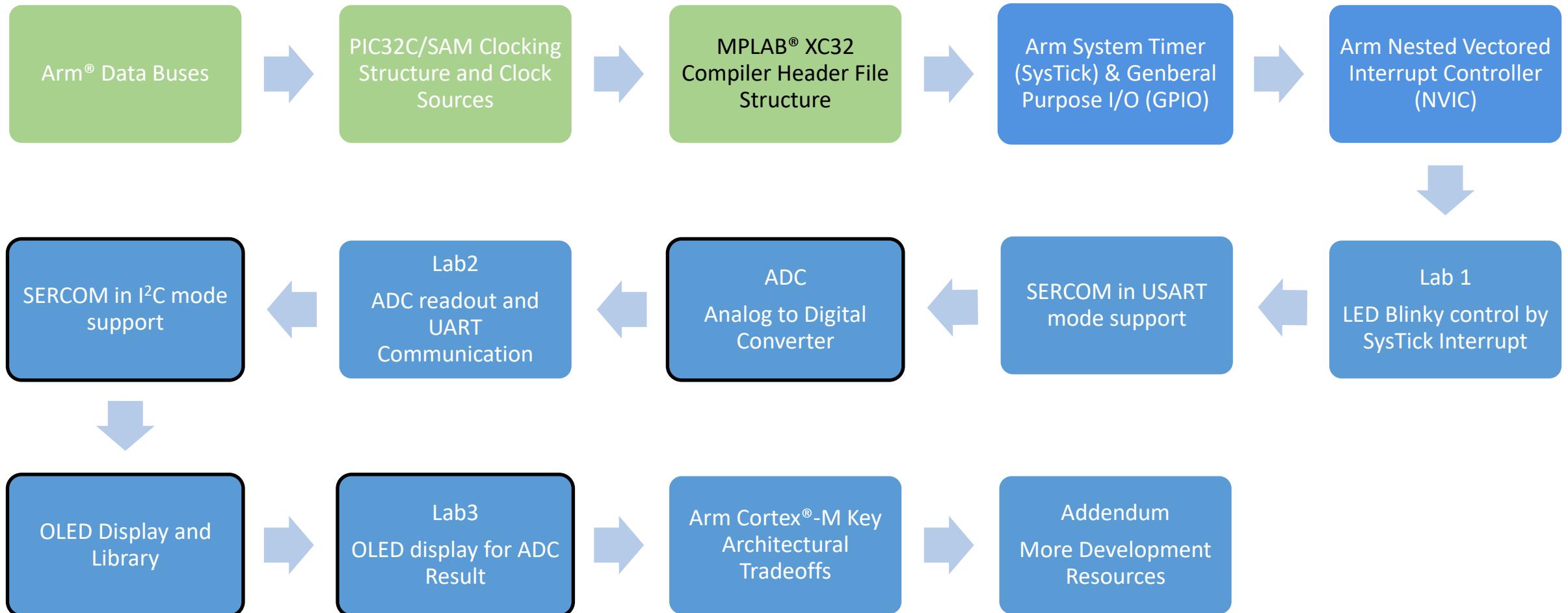
Configure DFLL
in open loop
mode

Set
multiplication
factor for closed
loop mode

Output
frequency is
continuously
regulated against
reference clock

$$\text{DFLL48 Frequency} = \text{Multiplier} \times \text{Reference Clock Frequency}$$

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`>

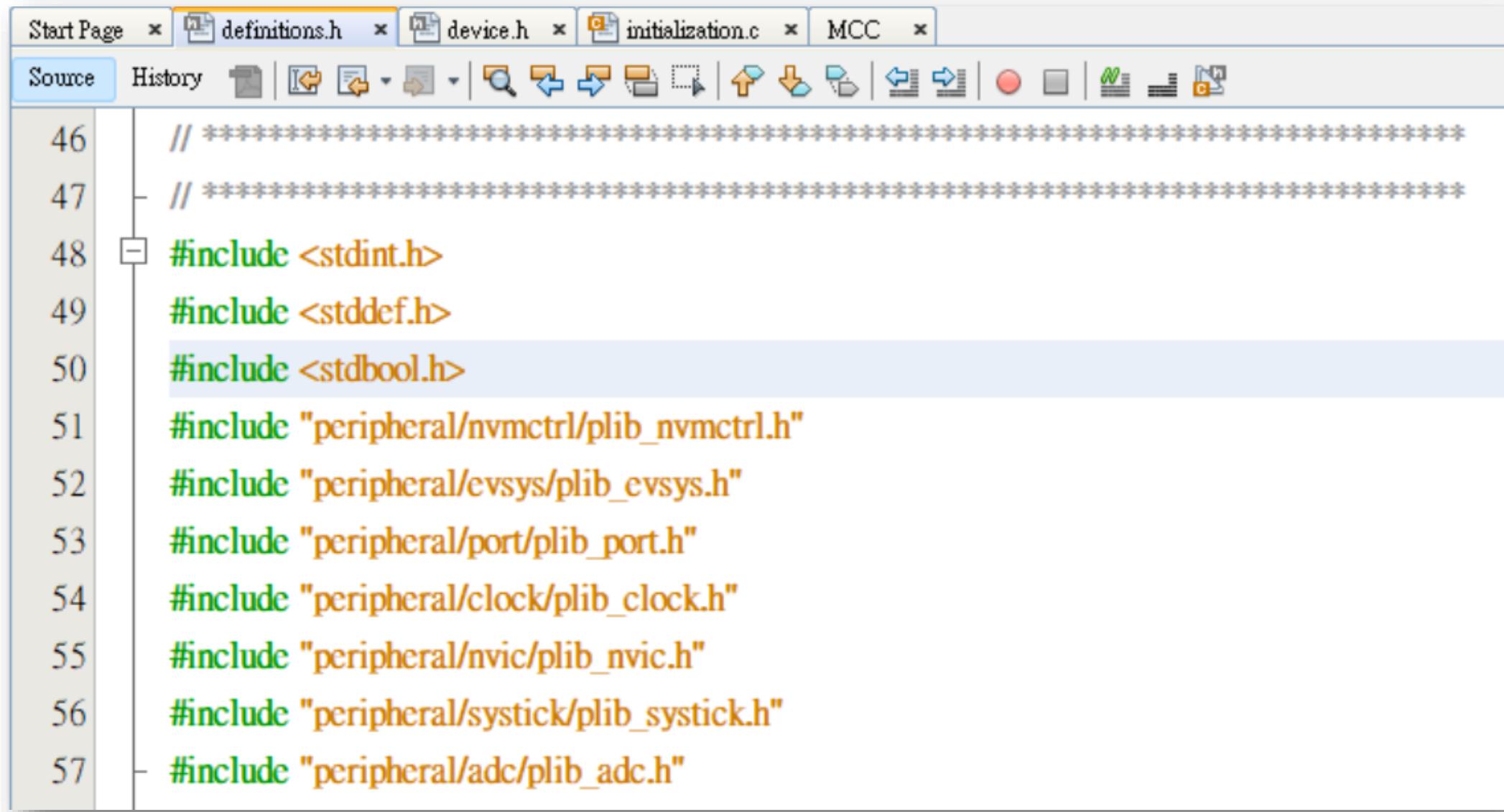


The `pic32cm3204gv00048.h` file will include other header files that provide peripheral information



`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



The screenshot shows a code editor window with the tab bar at the top containing "Start Page", "definitions.h", "device.h", "initialization.c", and "MCC". Below the tab bar is a toolbar with various icons for file operations like new, open, save, cut, copy, paste, and search. The main area of the editor displays the content of the "definitions.h" file. The code consists of several lines of C preprocessor directives (">#include) followed by comments. Lines 46 and 47 contain multi-line comments starting with double slashes. Lines 48 through 57 each begin with a "#include" directive followed by a specific header file path.

```
// *****
// *****
#include <stdint.h>
#include <stddef.h>
#include <stdbool.h>
#include "peripheral/nvmctrl/plib_nvmctrl.h"
#include "peripheral/cvsys/plib_cvsys.h"
#include "peripheral/port/plib_port.h"
#include "peripheral/clock/plib_clock.h"
#include "peripheral/nvic/plib_nvic.h"
#include "peripheral/systick/plib_systick.h"
#include "peripheral/adc/plib_adc.h"
```

Example of pic32cm3204gv00048.h

```
Start Page x definitions.h x device.h x initialization.c x MCC x pic32cm3204gv00048.h x
Source History |                   
218 // ****
219 /* SOFTWARE PERIPHERAL API DEFINITIONS FOR PIC32CM3204GV00048 */
220 /* **** */
221 #include "component/ac.h"
222 #include "component/adc.h"
223 #include "component/dac.h"
224 #include "component/dsu.h"
225 #include "component/eic.h"
226 #include "component/evsys.h"
227 #include "component/fuses.h"
228 #include "component/gclk.h"
229 #include "component/nvmctrl.h"
230 #include "component/pac.h"
```

Example of initialization.c

```
Start Page x definitions.h x device.h x initialization.c x MCC x pic32cm3204gv00048.h x
Source History |           |     
45 // ****
46 #include "definitions.h"
47 #include "device.h"
48 // ****
49 // ****
50 // Section: Configuration Bits
51 // ****
52 // ****
53 #pragma config NVMCTRL_BOOTPROT = SIZE_0BYTES
54 #pragma config NVMCTRL_EEPROM_SIZE = SIZE_0BYTES
55 #pragma config BOD33USERLEVEL = 0x7U // Enter Hexadecimal value
56 #pragma config BOD33_EN = ENABLED
57 #pragma config BOD33_ACTION = RESET
```

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

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;
```

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

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_XOSC32KRDY_Pos      _UINT32_(1)
#define OSC32KCTRL_STATUS_XOSC32KRDY_Msk     (_OSC32KCTRL_STATUS_XOSC32KRDY_Msk &
                                                (_UINT32_(value) << OSC32KCTRL_STATUS_XOSC32KRDY_Pos))
#define OSC32KCTRL_STATUS_XOSC32KRDY_Pos      _UINT32_(2)
#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_XOSC32KRDY_Pos      _UINT32_(3)
#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_CLKFAIL_Pos         _UINT32_(0)
#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_CLKFAIL_Pos         _UINT32_(1)
#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_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_CLKFAIL_Pos         _UINT32_(3)
#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_(0)
#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_CLKSW_Pos          _UINT32_(1)
#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_CLKSW_Pos          _UINT32_(2)
#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_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)
```

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

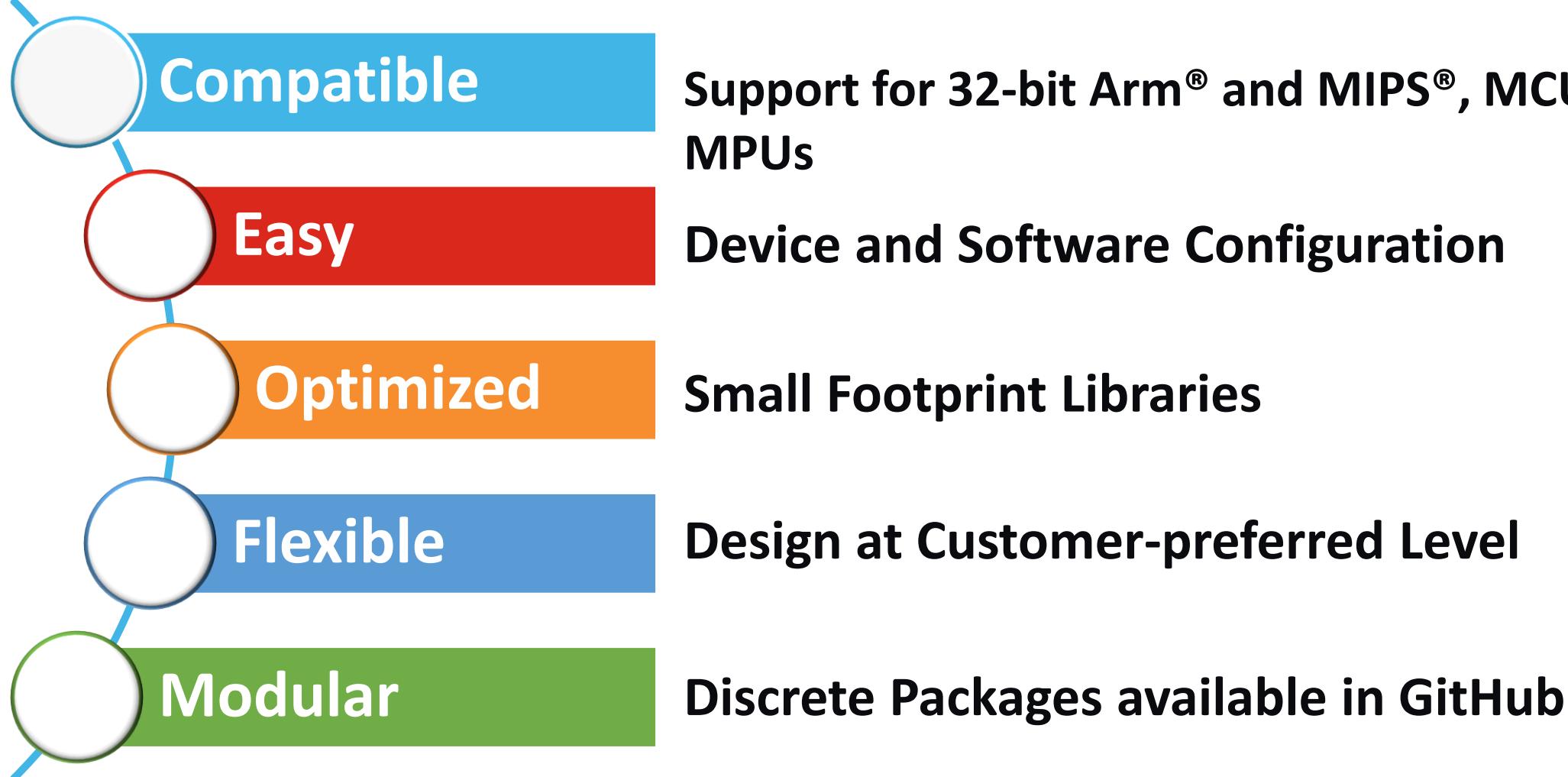
Example

```
/* ===== 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 */

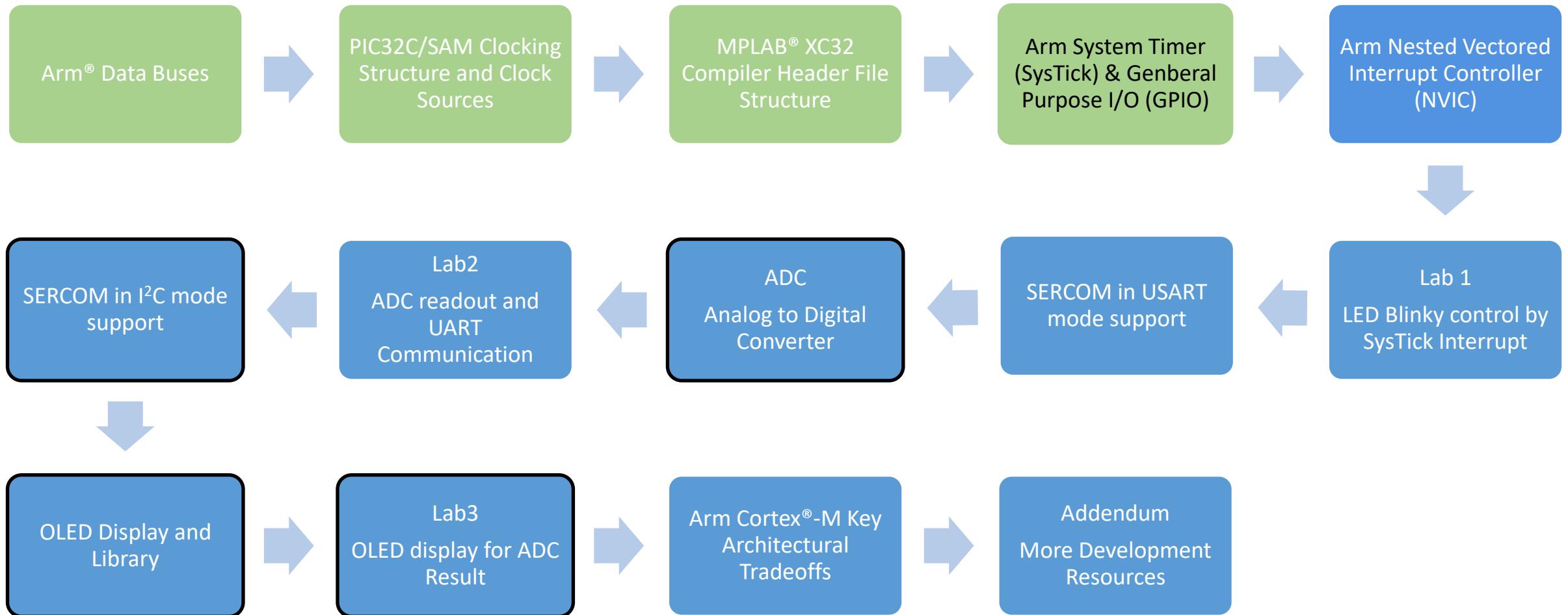
/* ===== Peripheral I/O masks ===== */
#define PORT_PA00          (_UINT32_(1) << 0)    /**< PORT mask for PA00 */
#define PORT_PA01          (_UINT32_(1) << 1)    /**< PORT mask for PA01 */
#define PORT_PA02          (_UINT32_(1) << 2)    /**< PORT mask for PA02 */

/* ===== 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)
```

MCC Harmony v3



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

SysTick – Initialize from System

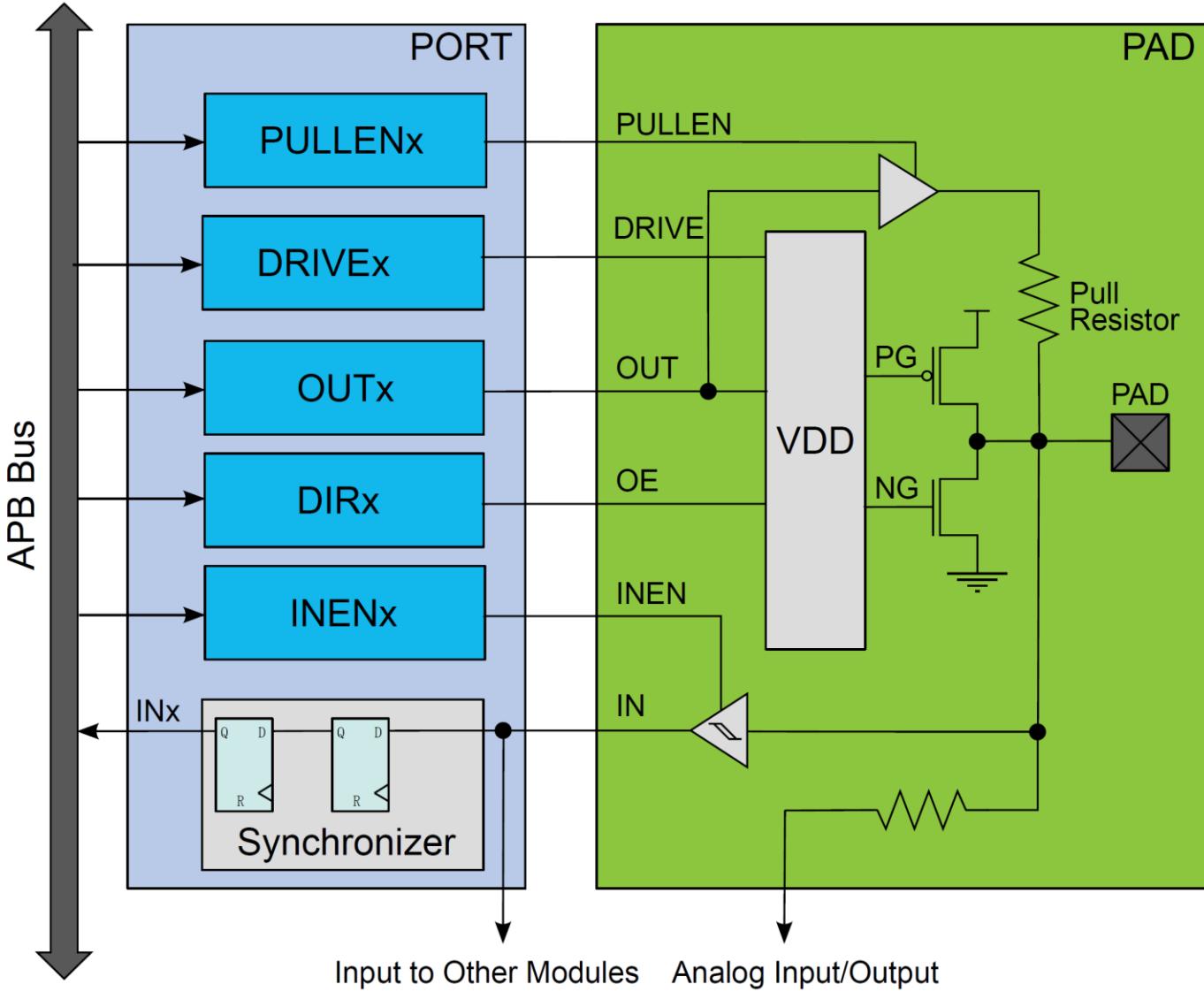
The screenshot shows the Microchip MCC (Memory Configuration Center) software interface. The top menu bar includes tabs for initialization.c, pic32cm2204gv00048.h, main.c, nvmctrl.h, adc.h, MCC (highlighted), Pin Diagram, Pin Table, and Pin Settings. The left sidebar has sections for Project Resource (Generate, Content Manager, Import, Export) and Device Resource (Libraries, Harmony, System). The Project Graph panel shows CMSIS Pack, System (selected), and Device Family Pack (DFP). Within the System tab, there are nodes for EVSYS Peripheral Library and NVMCTRL Peripheral Library (MEMORY). The right panel displays the System configuration tree, which is expanded to show the following settings:

- System
 - Device & Project Configuration
 - Cortex-M0+ Configuration
 - SysTick
 - Enable SysTick (checked)
 - SysTick Configuration
 - Enable Interrupt (checked)
 - SysTick Clock (Processor clock)
 - Systick Period(Milliseconds) (100)
 - Ports
 - Use PORT PLIB ? (checked)
 - Clock
 - Clock Source Configuration
 - Generic Clock Generator
 - (GCLK) Configuration
 - Synchronous Clock Configuration
 - Peripheral Clock Configuration



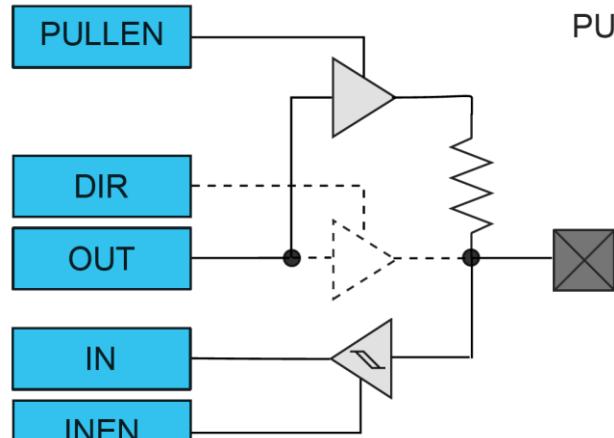
PIC32C and SAM General Purpose I/O (GPIO)

GPIO Structure



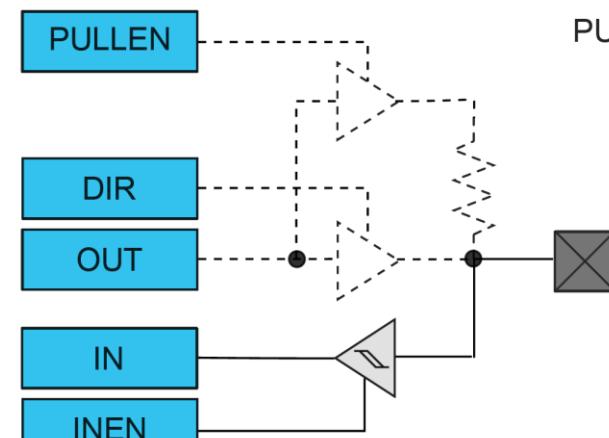
Pin Configuration Registers

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

PULLEN INEN DIR
1 1 0



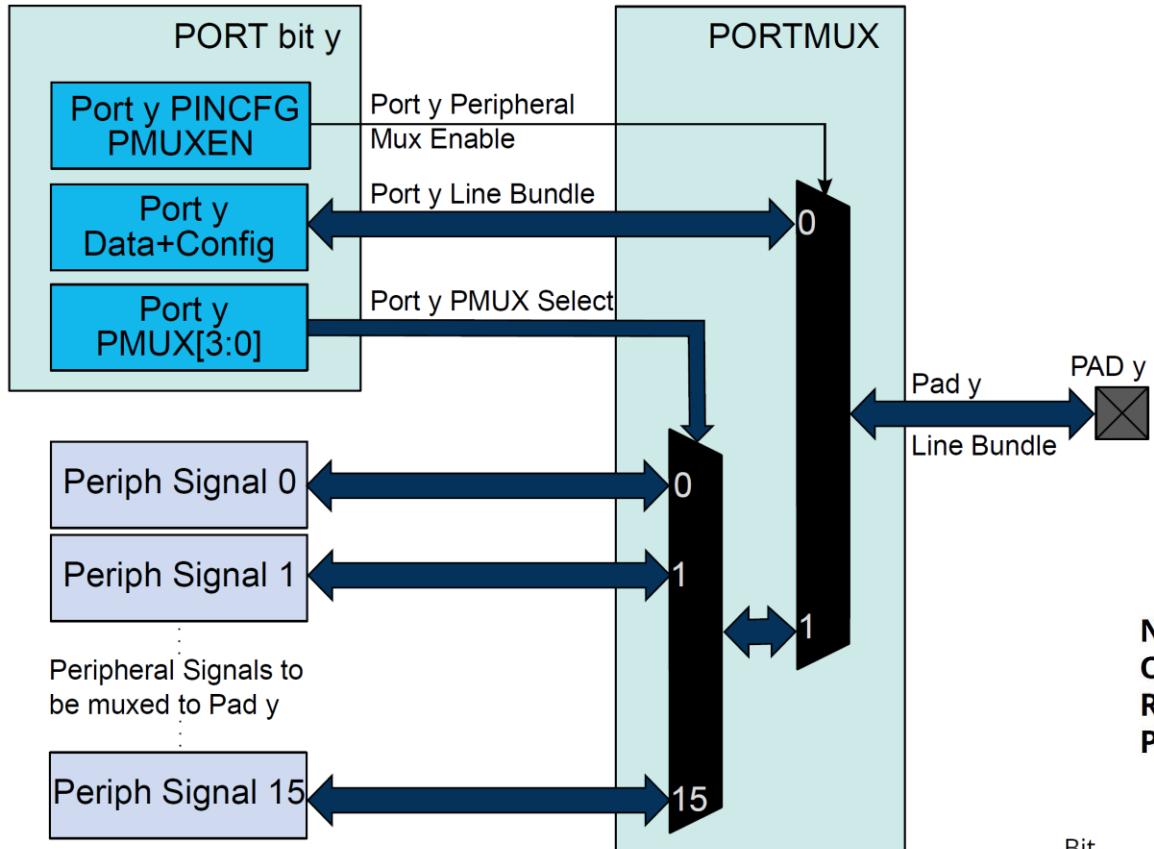
Standard Input

PULLEN INEN DIR
0 1 0

Registers Used for GPIO

Offset	Name	Bit Pos.	7	6	5	4	3			
0x00	DIR	7:0					DIR[7:0]	0x10	OUT	7:0
		15:8					DIR[15:8]			15:8
		23:16					DIR[23:16]			23:16
		31:24					DIR[31:24]			31:24
0x04	DIRCLR	7:0					DIRCLR[7:0]	0x14	OUTCLR	7:0
		15:8					DIRCLR[15:8]			15:8
		23:16					DIRCLR[23:16]			23:16
		31:24					DIRCLR[31:24]			31:24
0x08	DIRSET	7:0					DIRSET[7:0]	0x18	OUTSET	7:0
		15:8					DIRSET[15:8]			15:8
		23:16					DIRSET[23:16]			23:16
		31:24					DIRSET[31:24]			31:24
0x0C	DIRTGL	7:0					DIRTGL[7:0]	0x1C	OUTTGL	7:0
		15:8					DIRTGL[15:8]			15:8
		23:16					DIRTGL[23:16]			23:16
		31:24					DIRTGL[31:24]			31:24
0x20	IN								IN	7:0
										15:8
										23:16
										31:24

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
Access	PMUXO[3:0]				PMUXE[3:0]			
Reset	RW	RW	RW	RW	RW	RW	RW	RW

Connecting GPIO to Peripherals – Muxing Registers

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

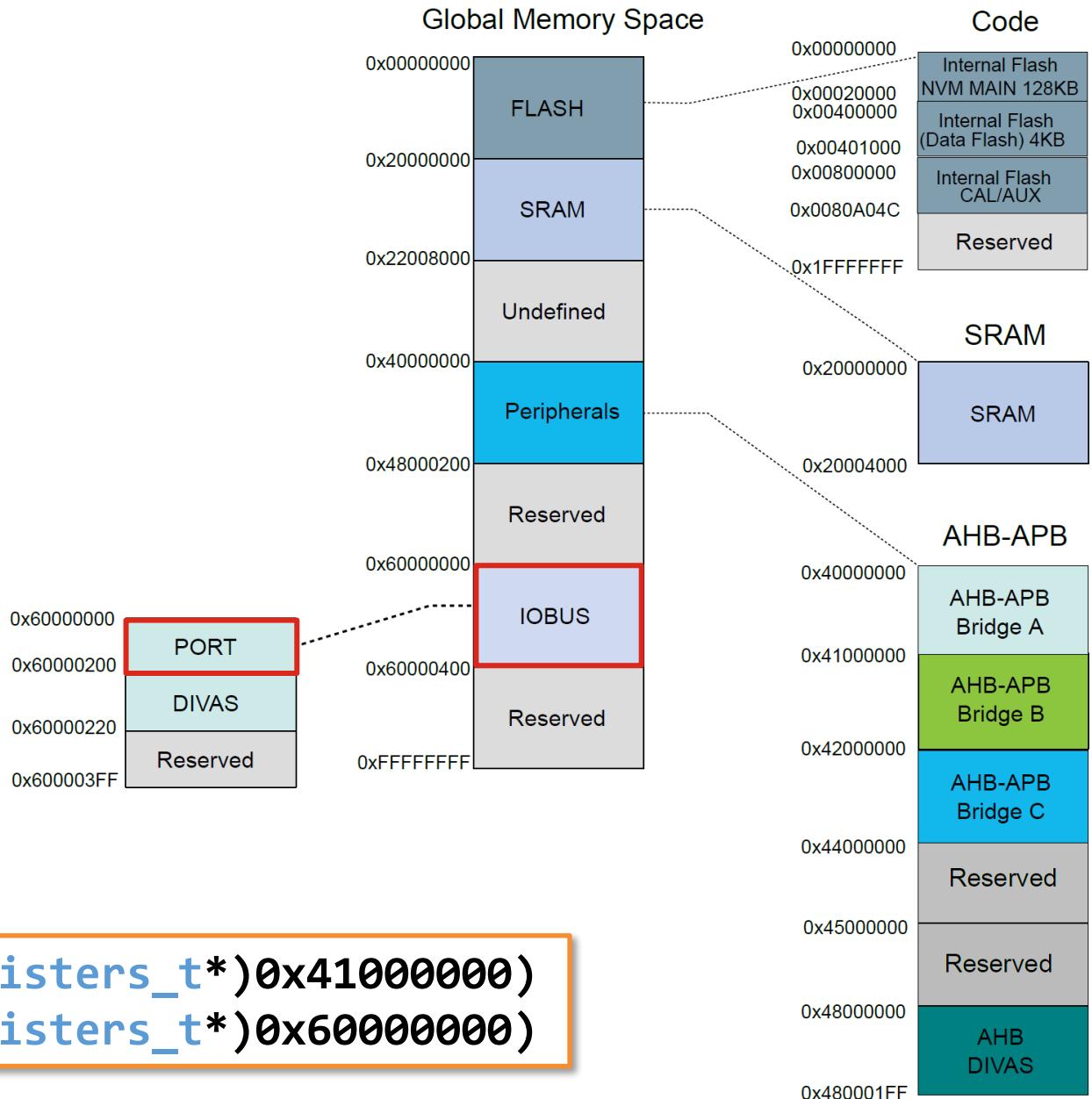
48-pin QFN	Pin name	A		B				SERCOM	SERCOM-ALT	TC/πCC	TCC	PDEC	AC/ GCLK	CCL	CCL/PDEC	Supply	Reset State
		EIC	REF	ADC0	ADC1	SDADC	AC										
1	PA00/ XIN32	EXTINT[0]							SERCOM1/ PAD[0]	TCC2/ WO[0]						VDDANA	I/O, HI-Z
2	PA01/ XOUT32	EXTINT[1]							SERCOM1/ PAD[1]	TCC2/ WO[1]						VDDANA	I/O, HI-Z
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

Port Access Through Single-Cycle IOBUS

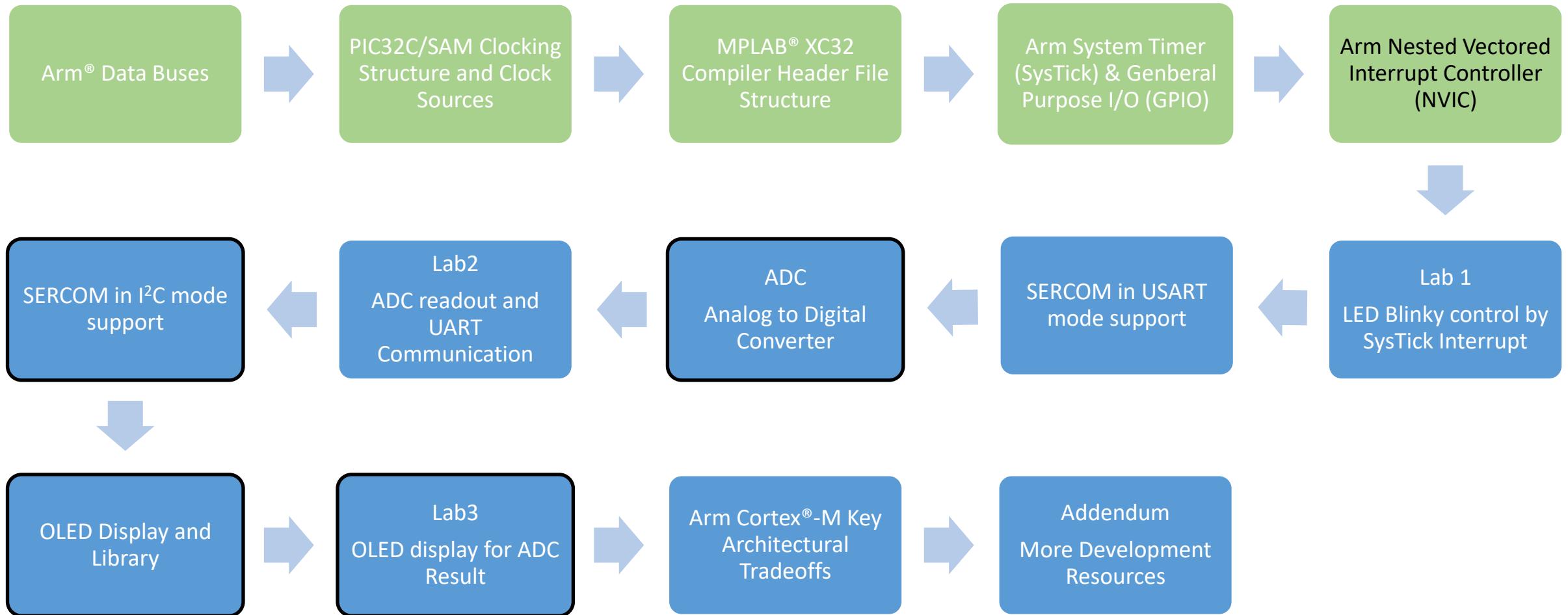
Port access priority:

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

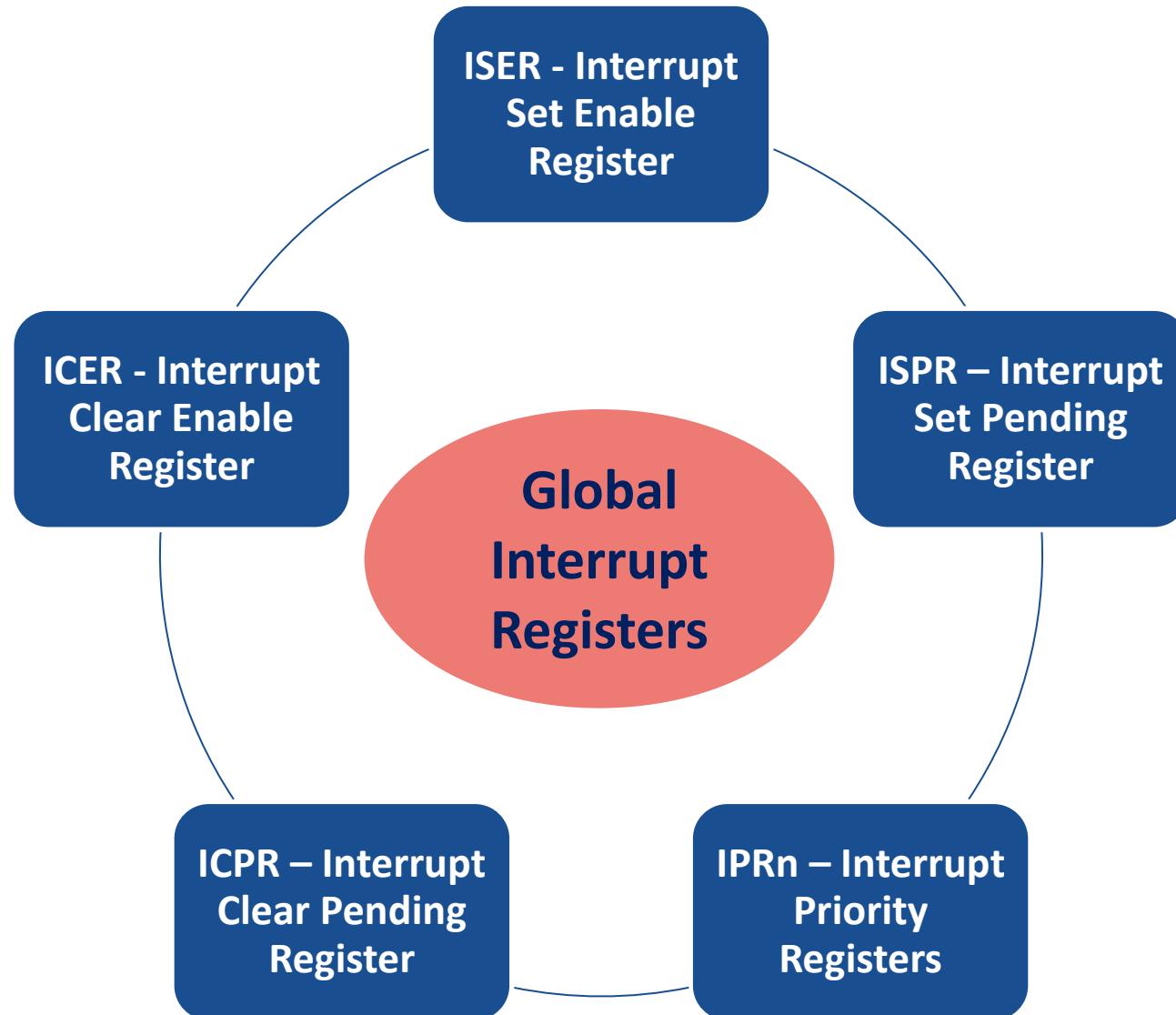


Up to 32 interrupt sources, each with four levels of priority on
Arm® Cortex®-M0+ devices (more on Cortex-M23/M4/M33)



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)	0
Main Clock (MCLK)	
Oscillators Controller (OSCCTRL)	
32 kHz Oscillators Controller (OSC32KCTRL)	
Supply Controller (SUPC)	
Protection Access Controller (PAC)	
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

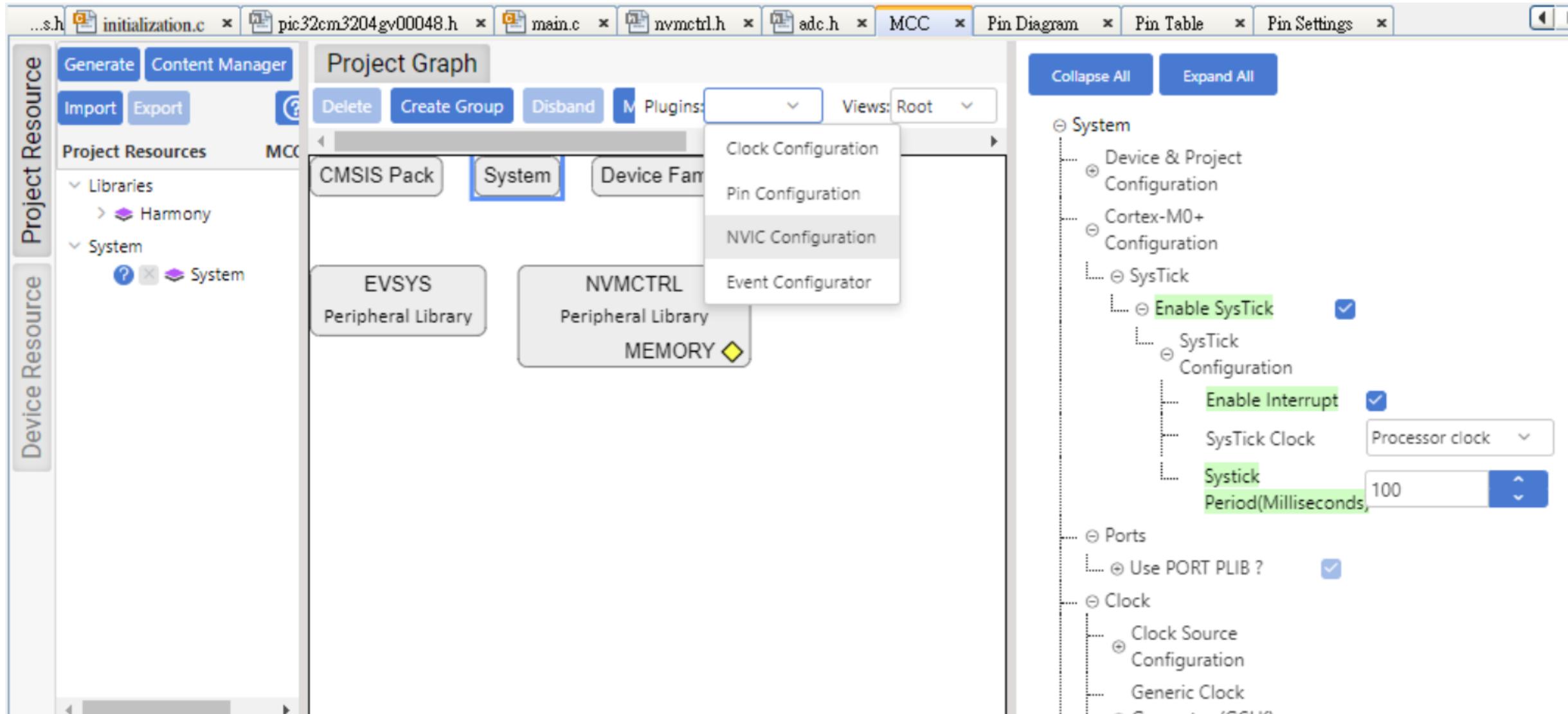
- Interrupt names (**IRQn**) are located in the device header file (**pic32cm1216mc00032.h**)
- To enable TC3 interrupt:

```
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
Access			MC1 R/W	MC0 R/W			ERR R/W	OVF R/W
Reset			0	0			0	0

NVIC Configuration Plugin in MCC Harmony

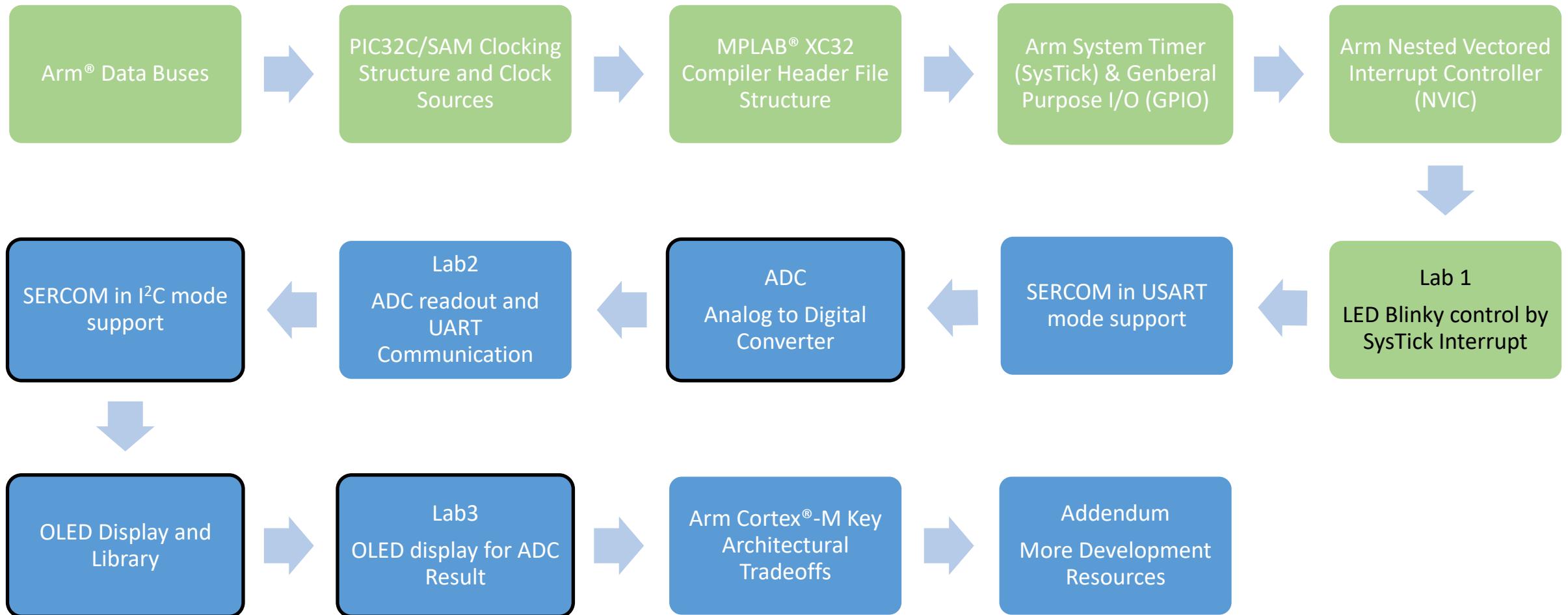


NVIC Configuration in MCC Harmony

The screenshot shows the NVIC Configuration window in MCC Harmony. The table lists 16 interrupt vectors, their descriptions, enable status, priority, and handler names. Most vectors have priority set to 0 or 3, while the Power Manager (vector 0) has priority 3.

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



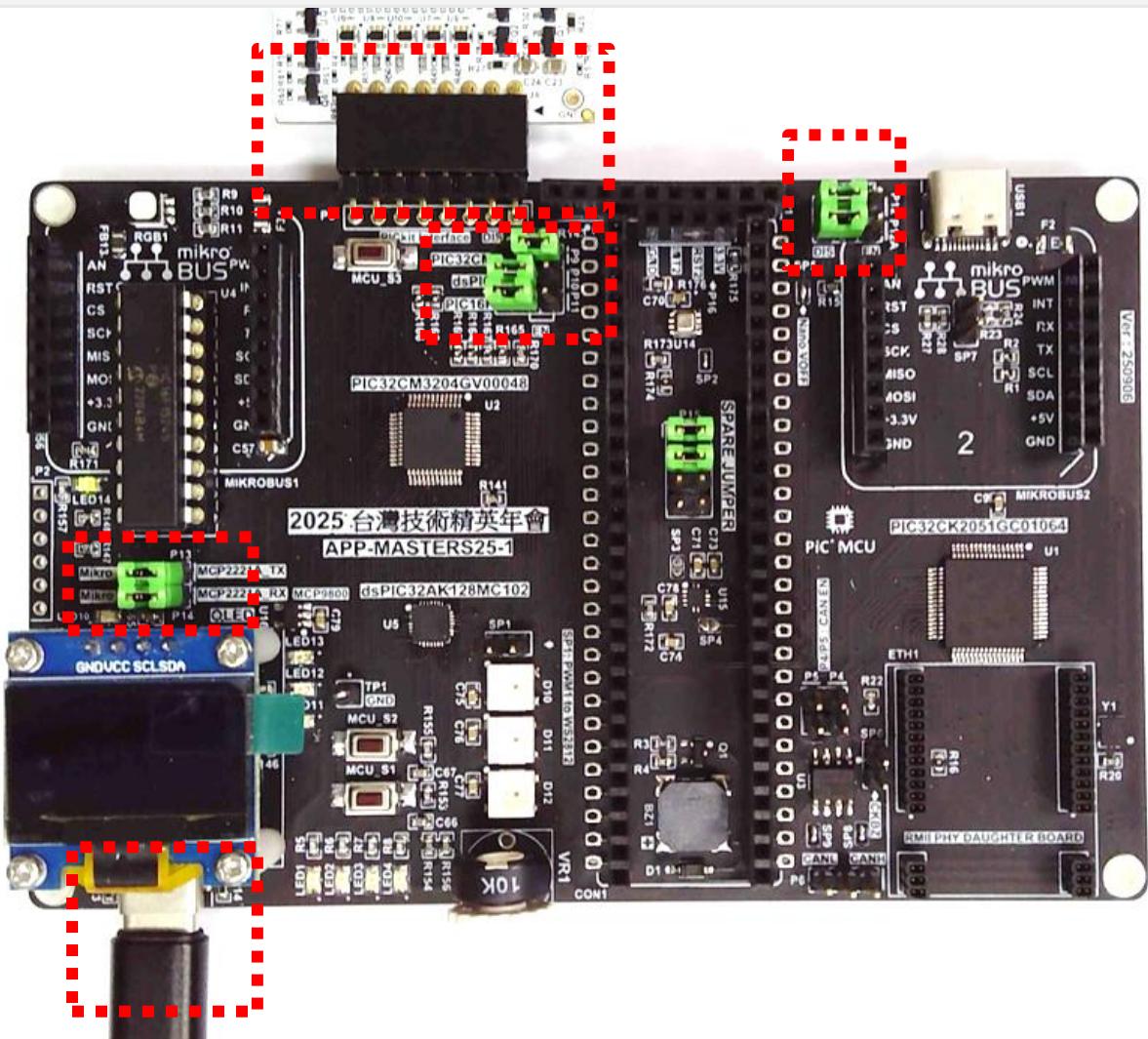
設定 SysTick 來獲得一個 100ms 的時間來源並使用其中斷來驅動 LED1



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

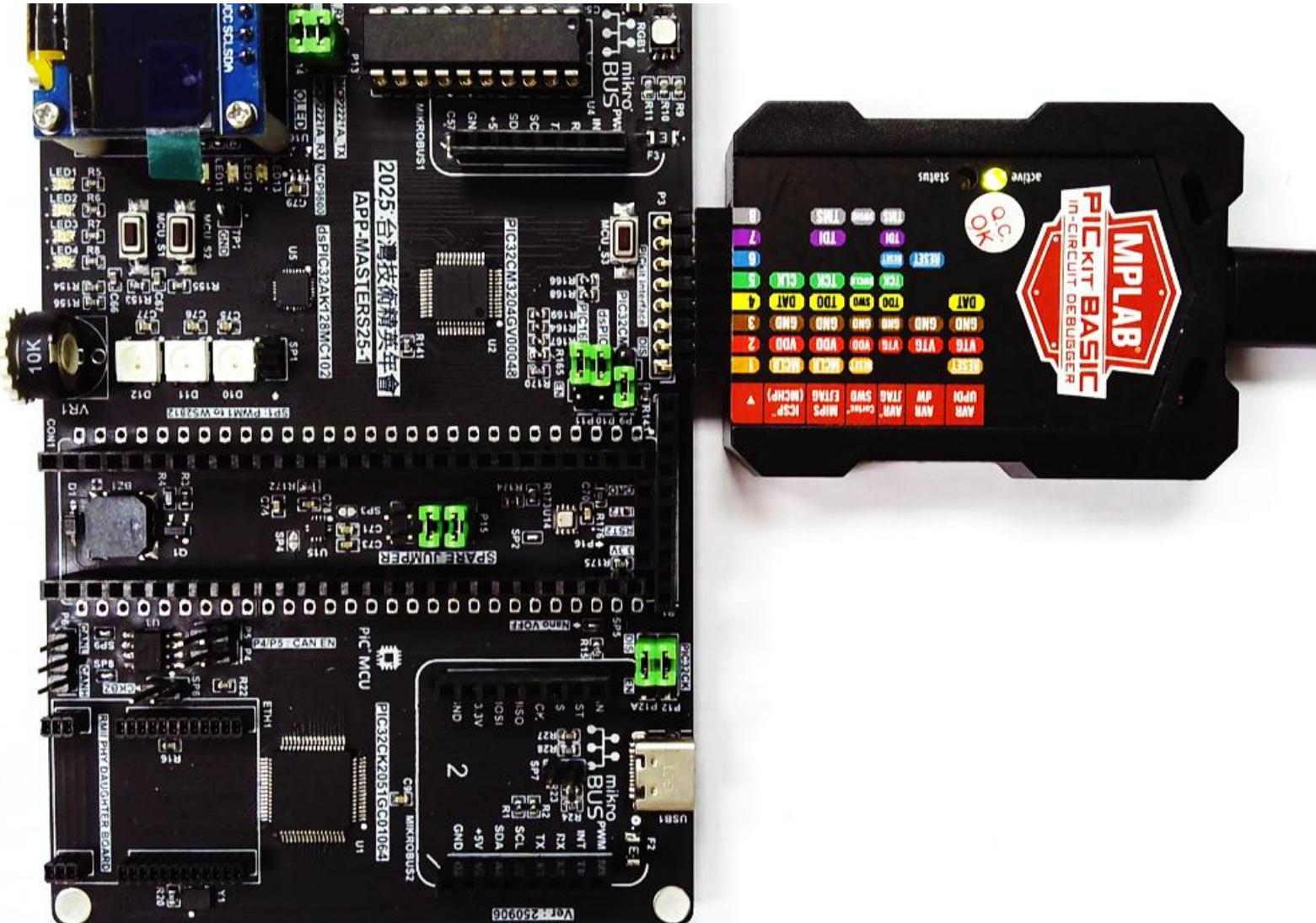
APP-MASTERS25-1 實驗板的 Jumper 設定以及燒錄器連接方式



- 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

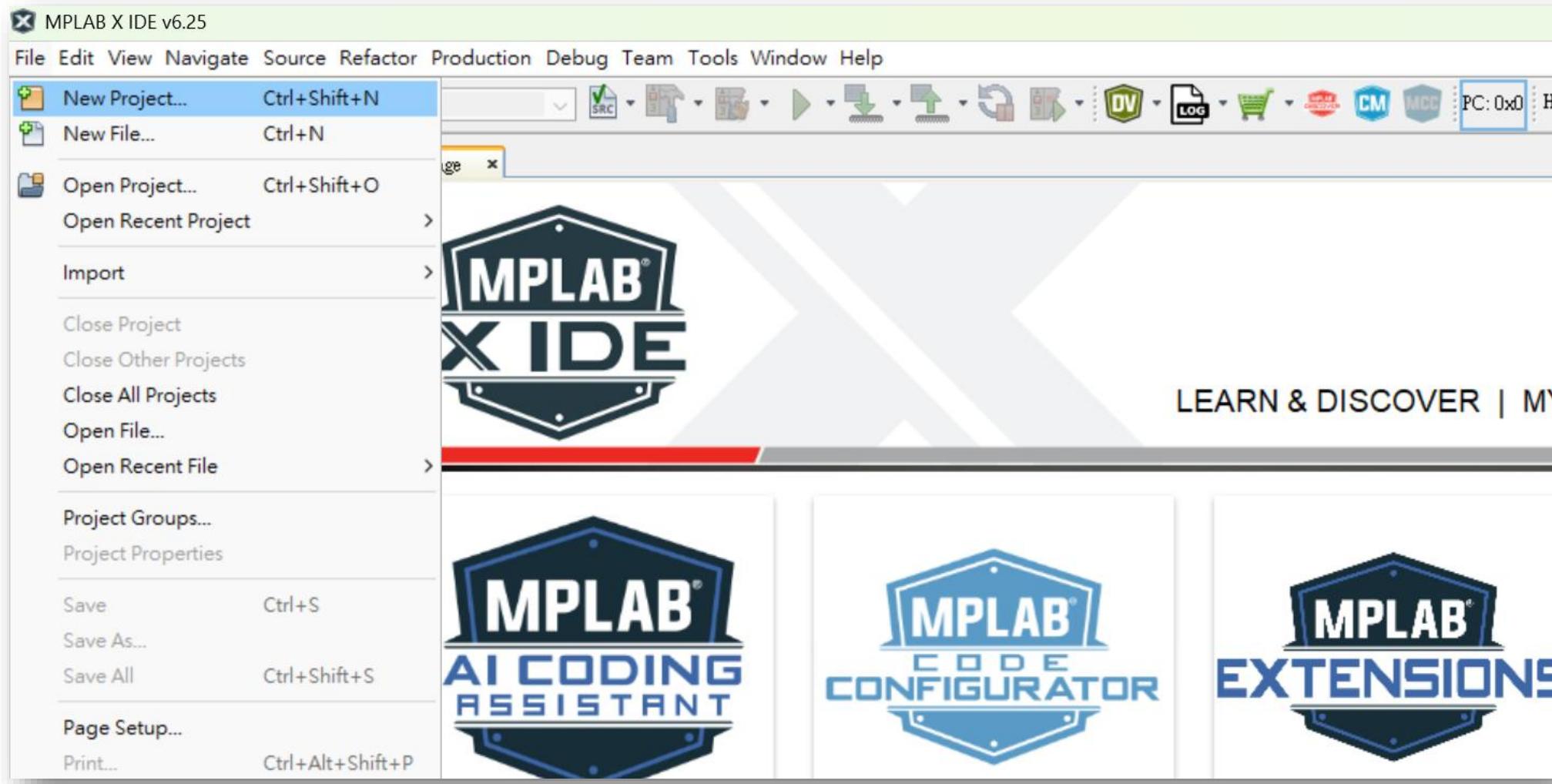
Lab1 – MPLAB® X IDE project creation & Systick

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



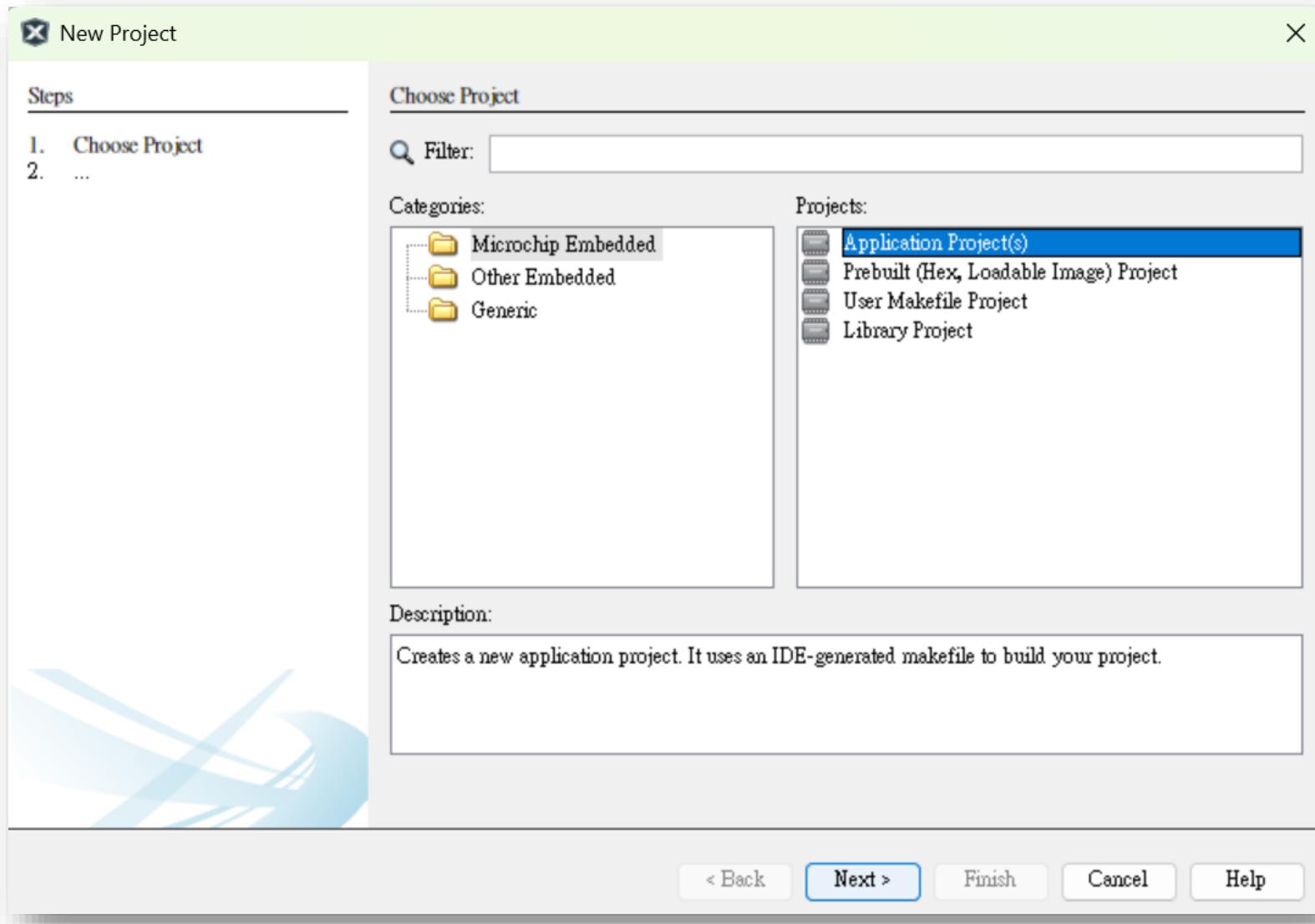
Lab1 – MPLAB® X IDE project creation & Systick

File -> New Project



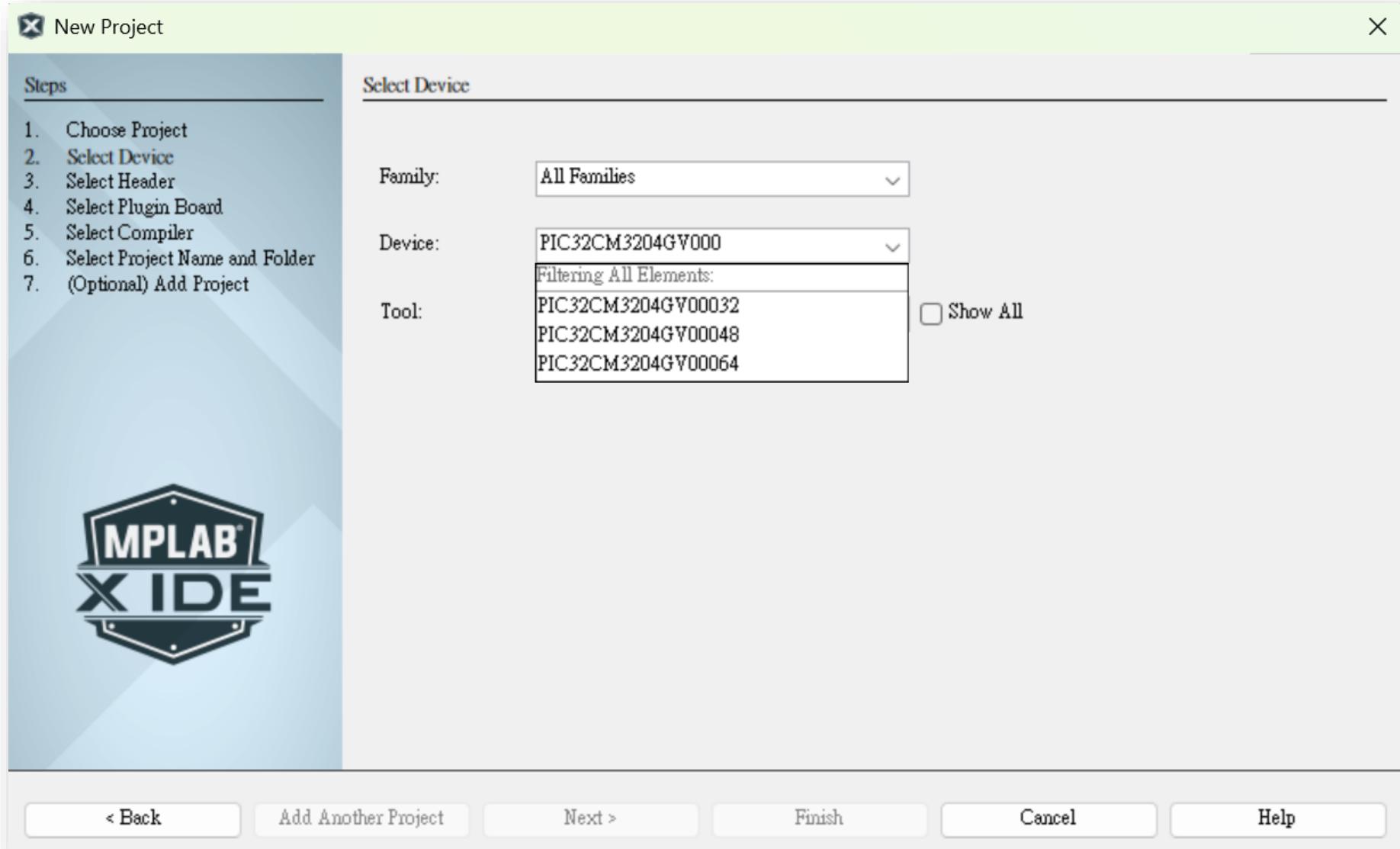
Lab1 – MPLAB® X IDE project creation & Systick

Select “Application Project(s)



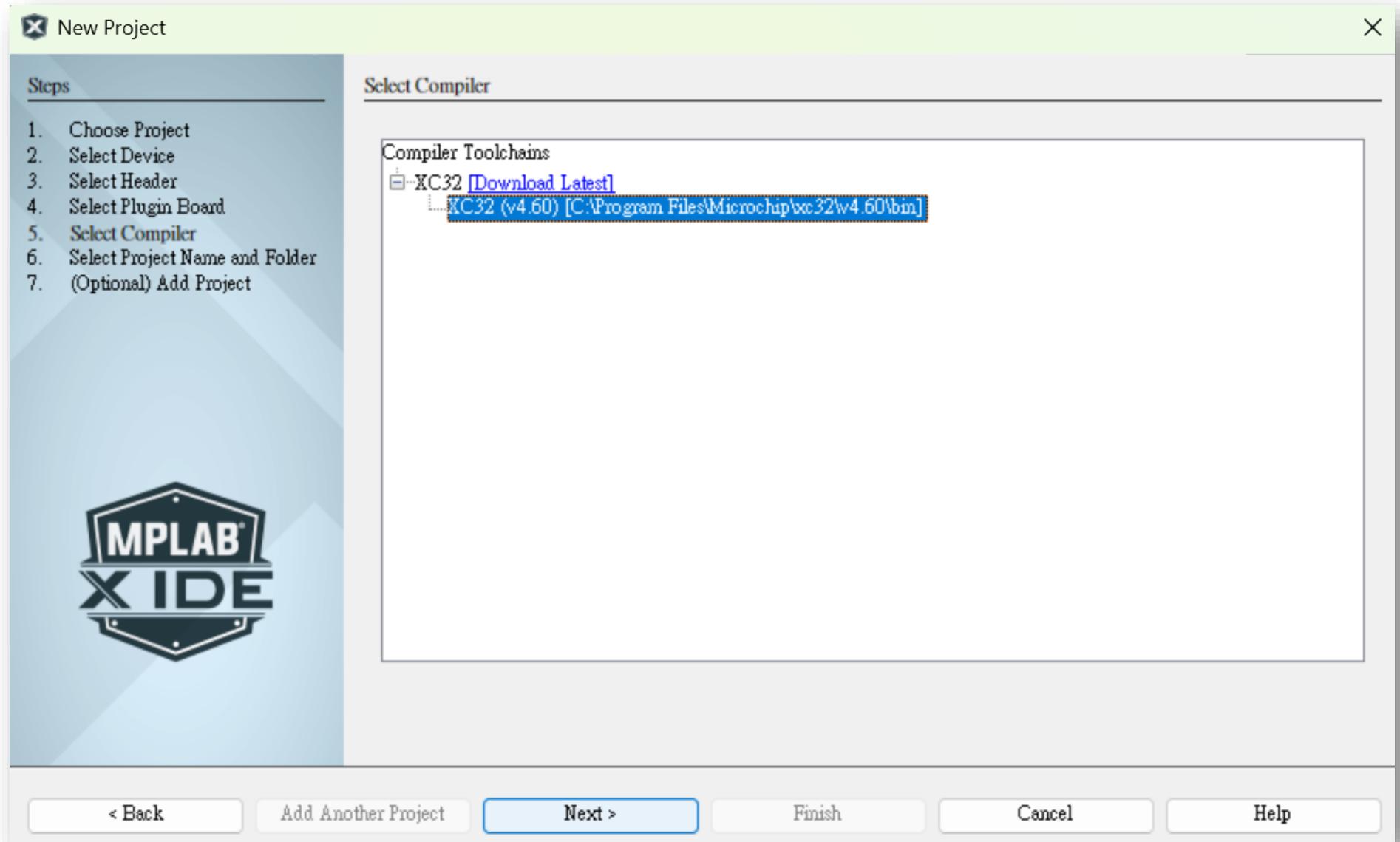
Lab1 – MPLAB® X IDE project creation & Systick

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



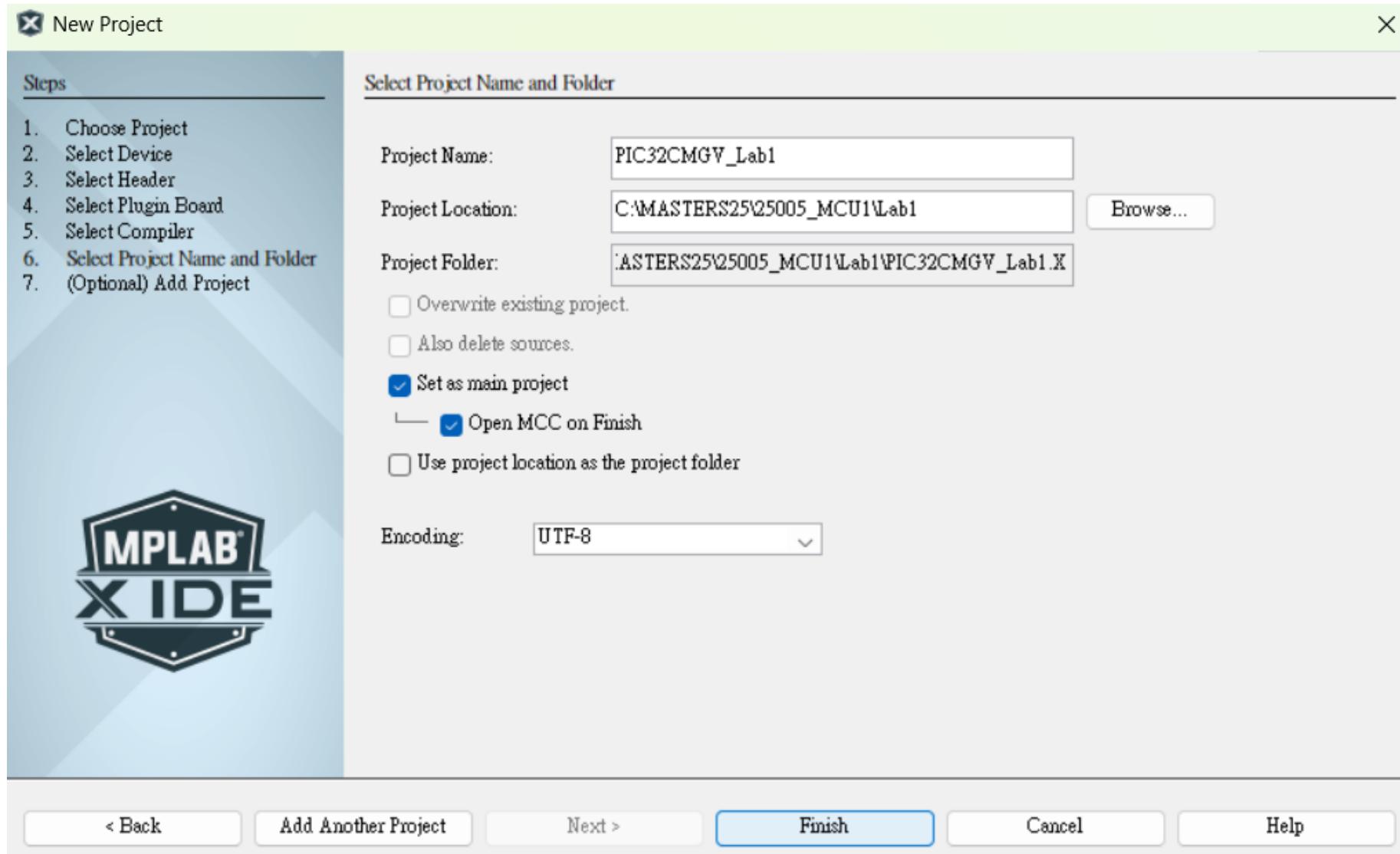
Lab1 – MPLAB® X IDE project creation & Systick

Select Compiler Toolchains – 使用 XC32 v4.60 版



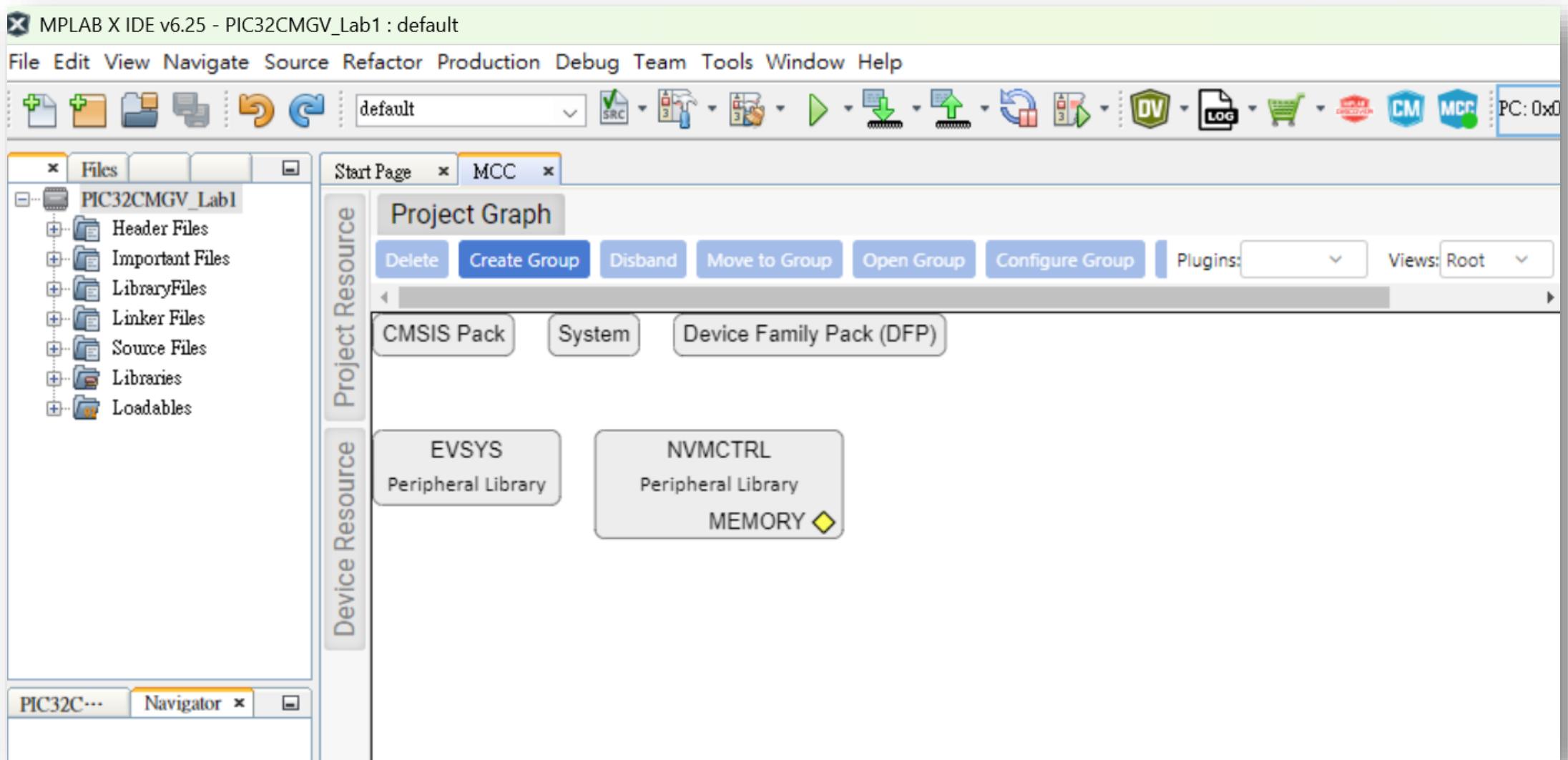
Lab1 – MPLAB® X IDE project creation & Systick

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



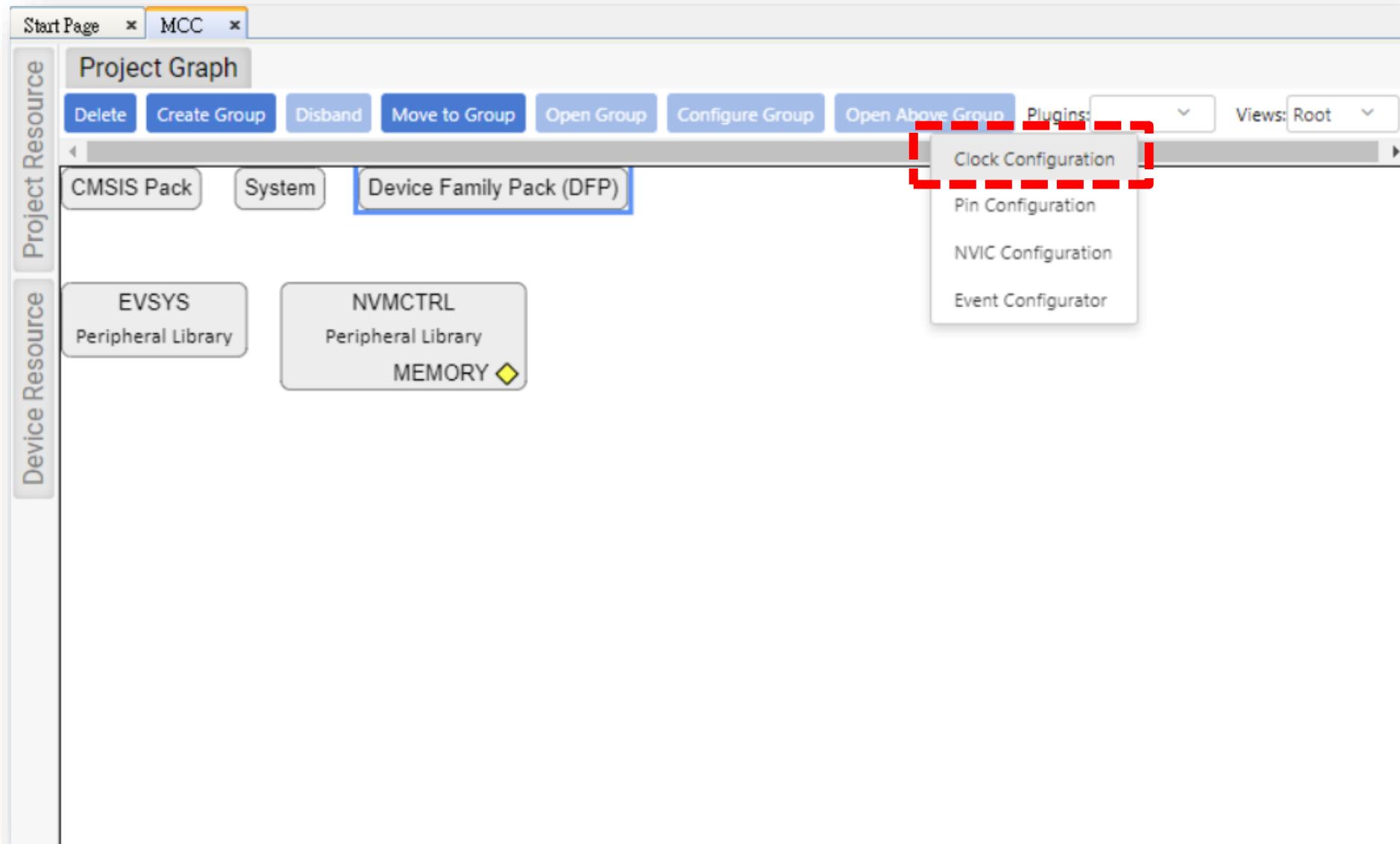
Lab1 – MPLAB® X IDE project creation & Systick

按下 Finish 之後 MCC 會被自動開啟並下載必要的軟體元件



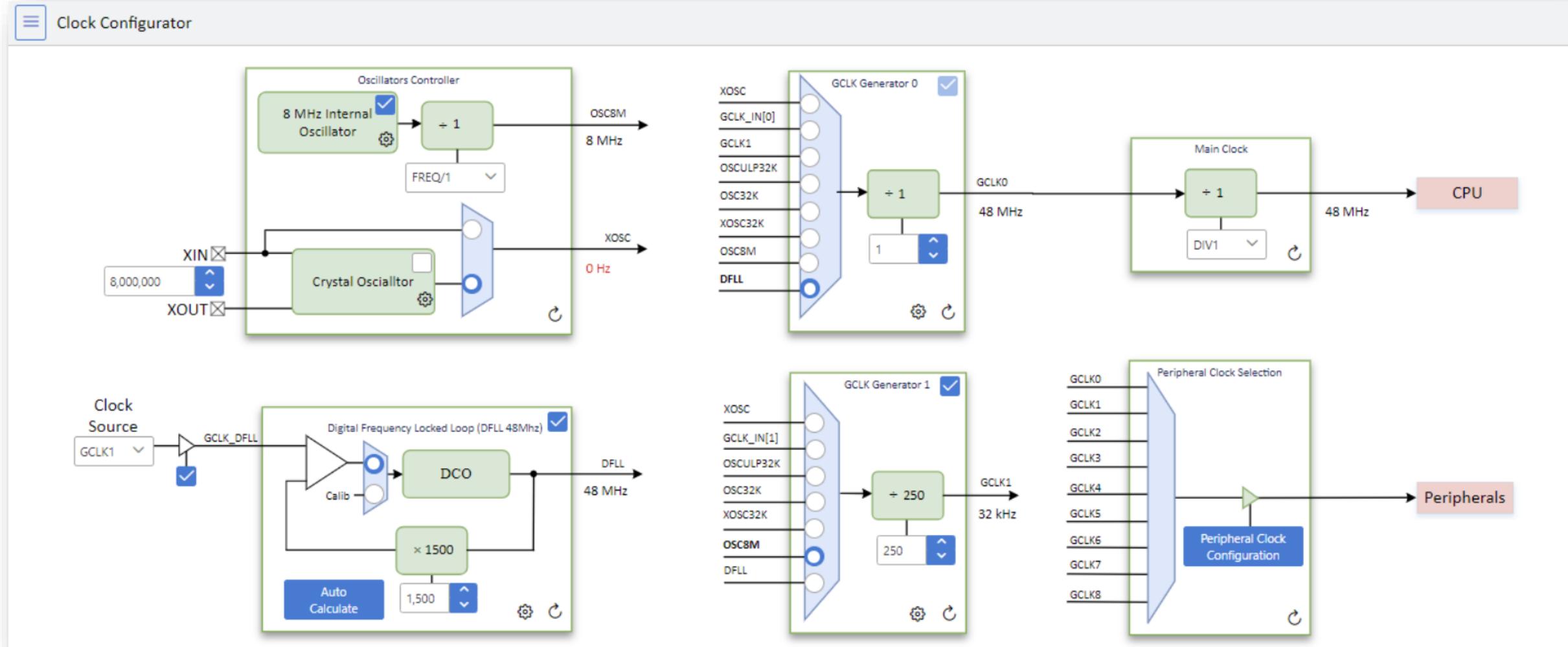
Lab1 – MPLAB® X IDE project creation & Systick

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



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

The screenshot shows the MPLAB X IDE interface with the "Clock Configurator" window open. The window title is "Clock Configurator" and the sub-tab is "Peripheral Clock Configuration". The table lists various peripherals and their clock settings:

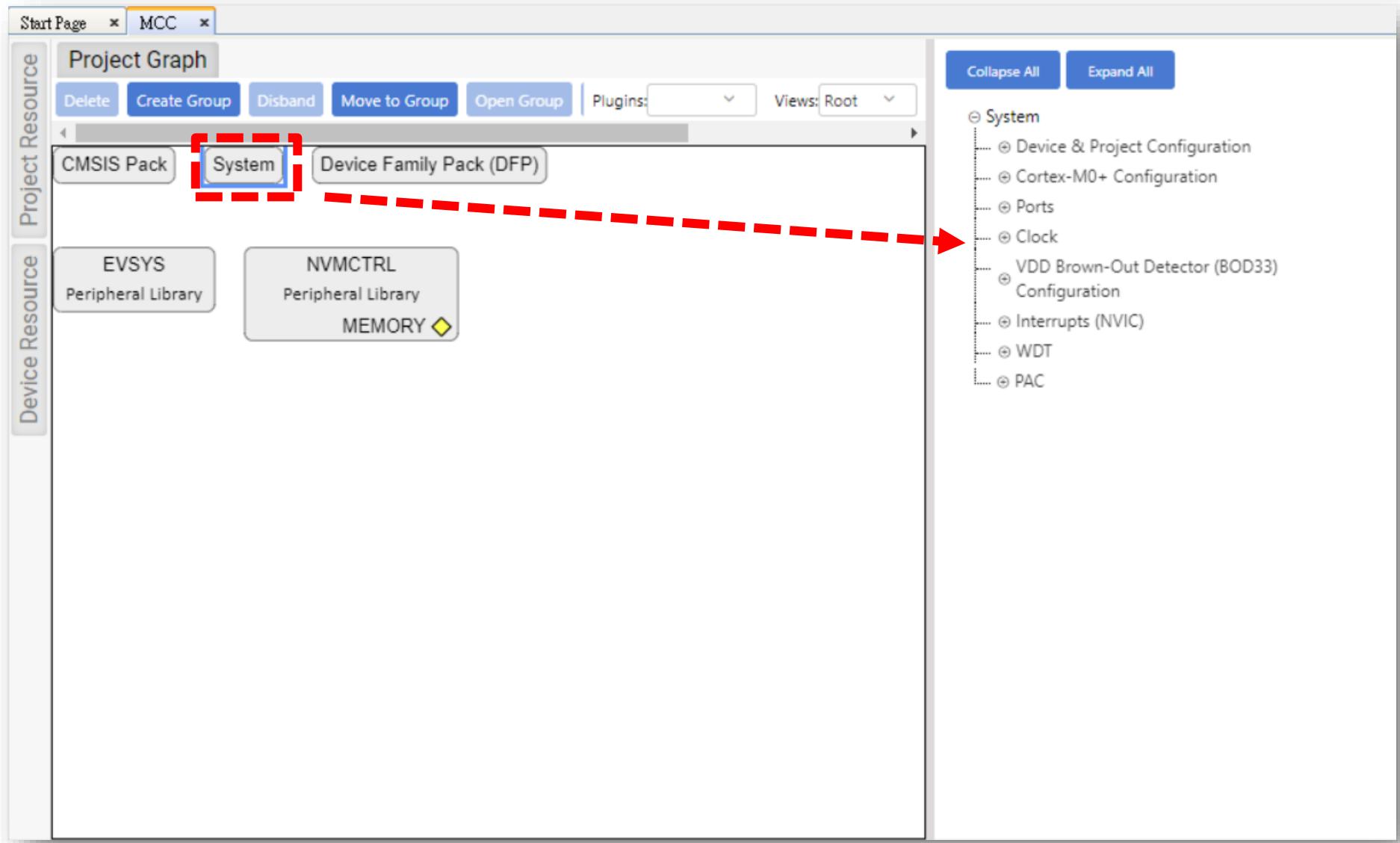
Peripheral	Enable	Source	Peripheral Clock Frequency
AC_ANA	<input type="checkbox"/>	GCLK0	--
AC_DIG	<input type="checkbox"/>	GCLK0	--
ADC	<input type="checkbox"/>	GCLK0	--
DAC	<input type="checkbox"/>	GCLK0	--
EIC	<input type="checkbox"/>	GCLK0	--
EVSYS_0	<input type="checkbox"/>	GCLK0	--
EVSYS_1	<input type="checkbox"/>	GCLK0	--
EVSYS_2	<input type="checkbox"/>	GCLK0	--
EVSYS_3	<input type="checkbox"/>	GCLK0	--
EVSYS_4	<input type="checkbox"/>	GCLK0	--
EVSYS_5	<input type="checkbox"/>	GCLK0	--

A red arrow points from the "Peripheral Clock Selection" section of the configuration dialog to the "Peripheral Clock Configuration" section, indicating the connection between the two.

On the right side of the interface, there is a clock tree diagram. It starts with a "Main Clock" source, which is divided by a "DIV1" block to produce a "48 MHz" signal for the "CPU". This same "48 MHz" signal also feeds into a "Peripheral Clock Selection" block. From this block, individual clock signals are distributed to the "Peripherals" listed in the configuration table.

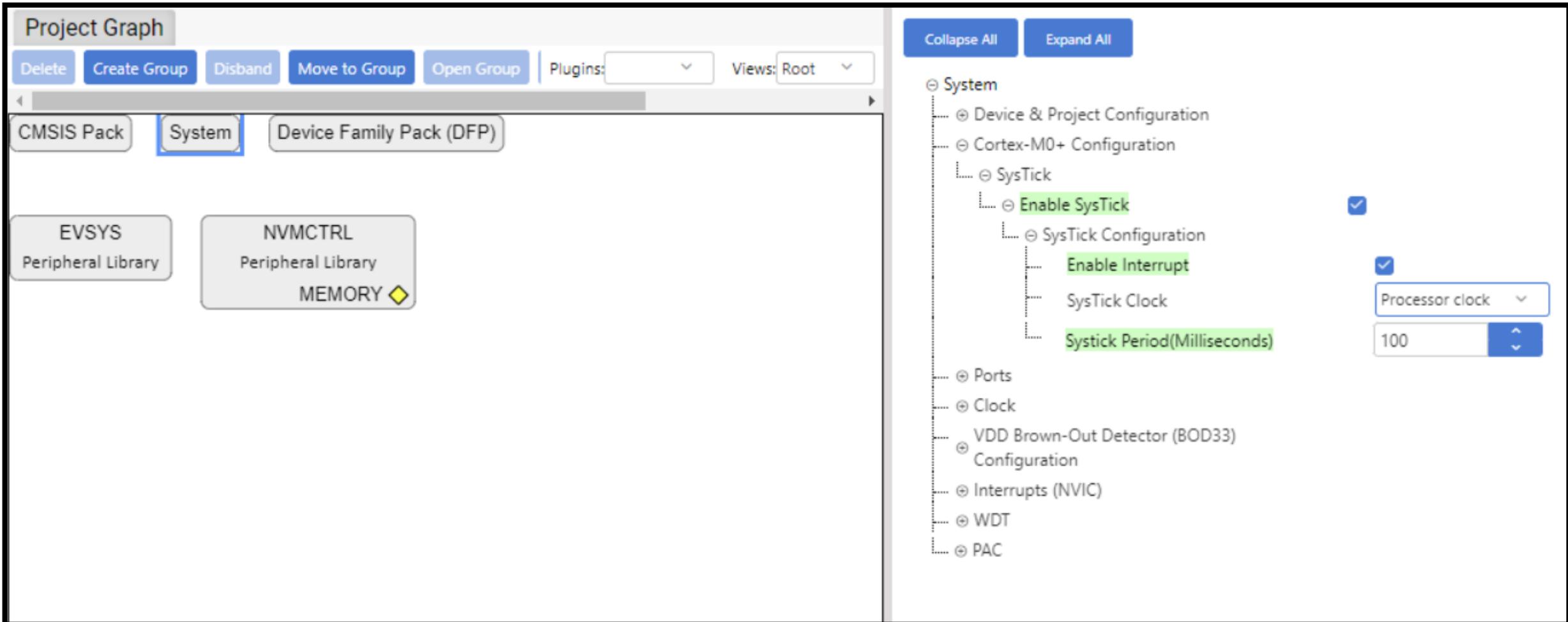
Lab1 – MPLAB® X IDE project creation & Systick

點選 System 來設定 SysTick 以及觀察其他重要選項



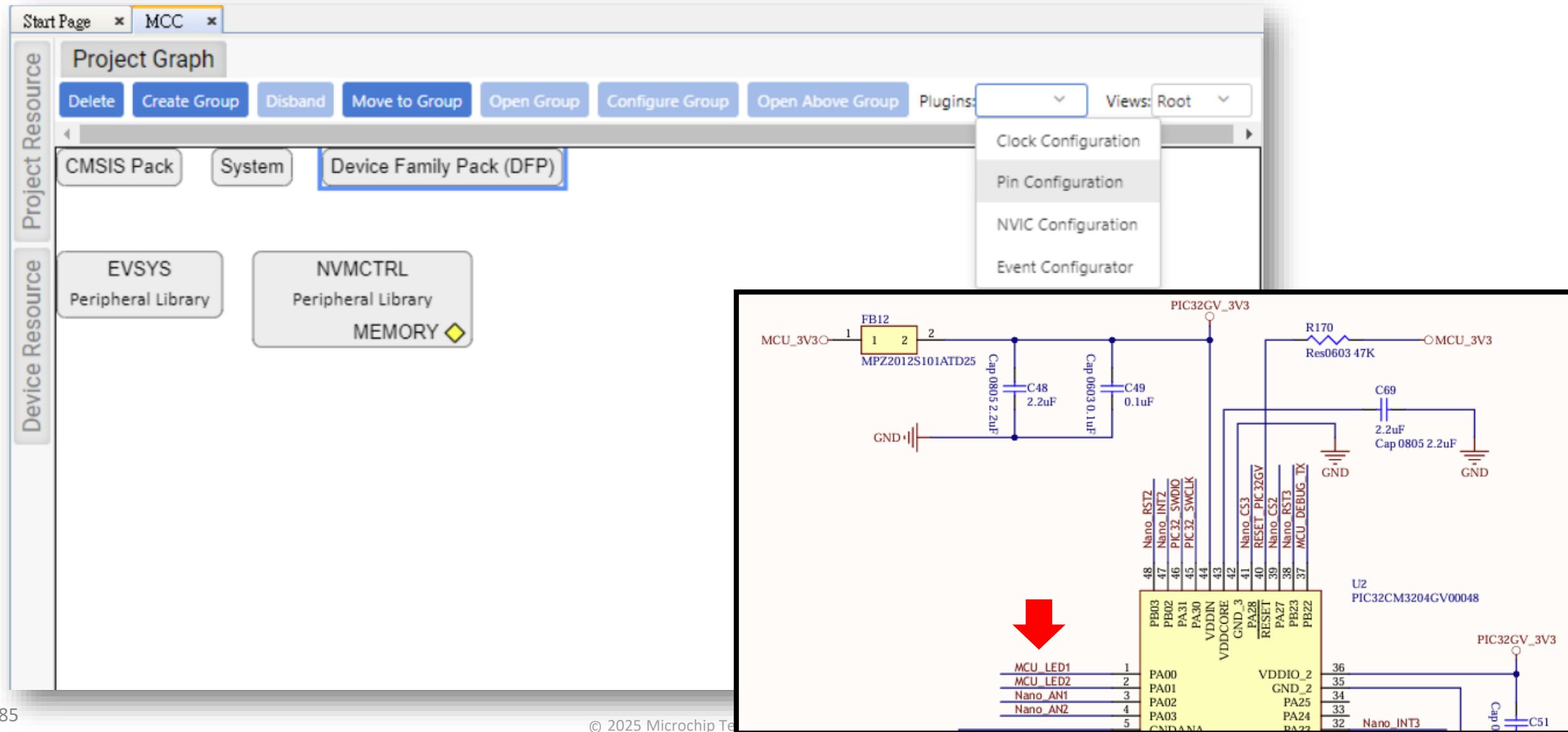
Lab1 – MPLAB® X IDE project creation & Systick

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

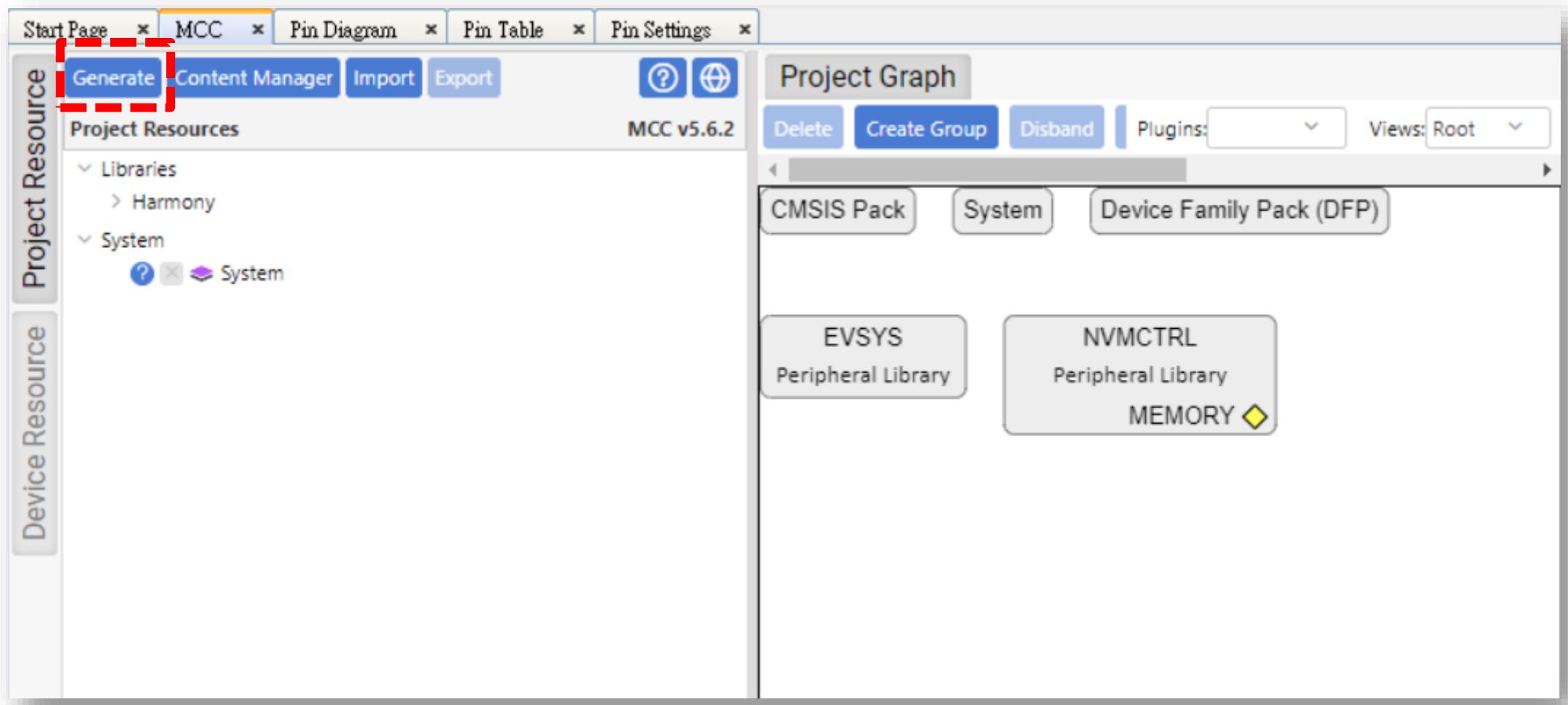
- Function
- Direction
- Custom Name (MCC 在產生 PLIB API 時會使用此名稱)

The screenshot shows the 'Pin Settings' tab in the MPLAB X IDE. The table lists pins from 1 to 7, with PA00 selected. The columns represent Pin Number, Pin ID, Custom Name, Function, Mode, Direction, Latch, Pull Up, Pull Down, and Drive Strength. PA00 is configured as a Digital output (Out) with Low as the default value for Latch, Pull Up, and Pull Down. The 'Custom Name' field contains 'LED1'. The 'Function' dropdown for PA00 shows 'GPIO' as the current selection, with other options like 'Available', 'EIC_EXTINT0', 'GPIO', 'SERCOM1_PAD0', 'SYSCTRL_XIN32', and 'TC2_WO0' listed below it.

Pin Number	Pin ID	Custom Name	Function	Mode	Direction	Latch	Pull Up	Pull Down	Drive Strength
1	PA00	LED1	GPIO	Digital	Out	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
2	PA01		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
3	PA02		EIC_EXTINT0	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
4	PA03		GPIO	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
5	GNDANA		SERCOM1_PAD0	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
6	VDDANA		SYSCTRL_XIN32	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
7	PB08		TC2_WO0	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
			Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL

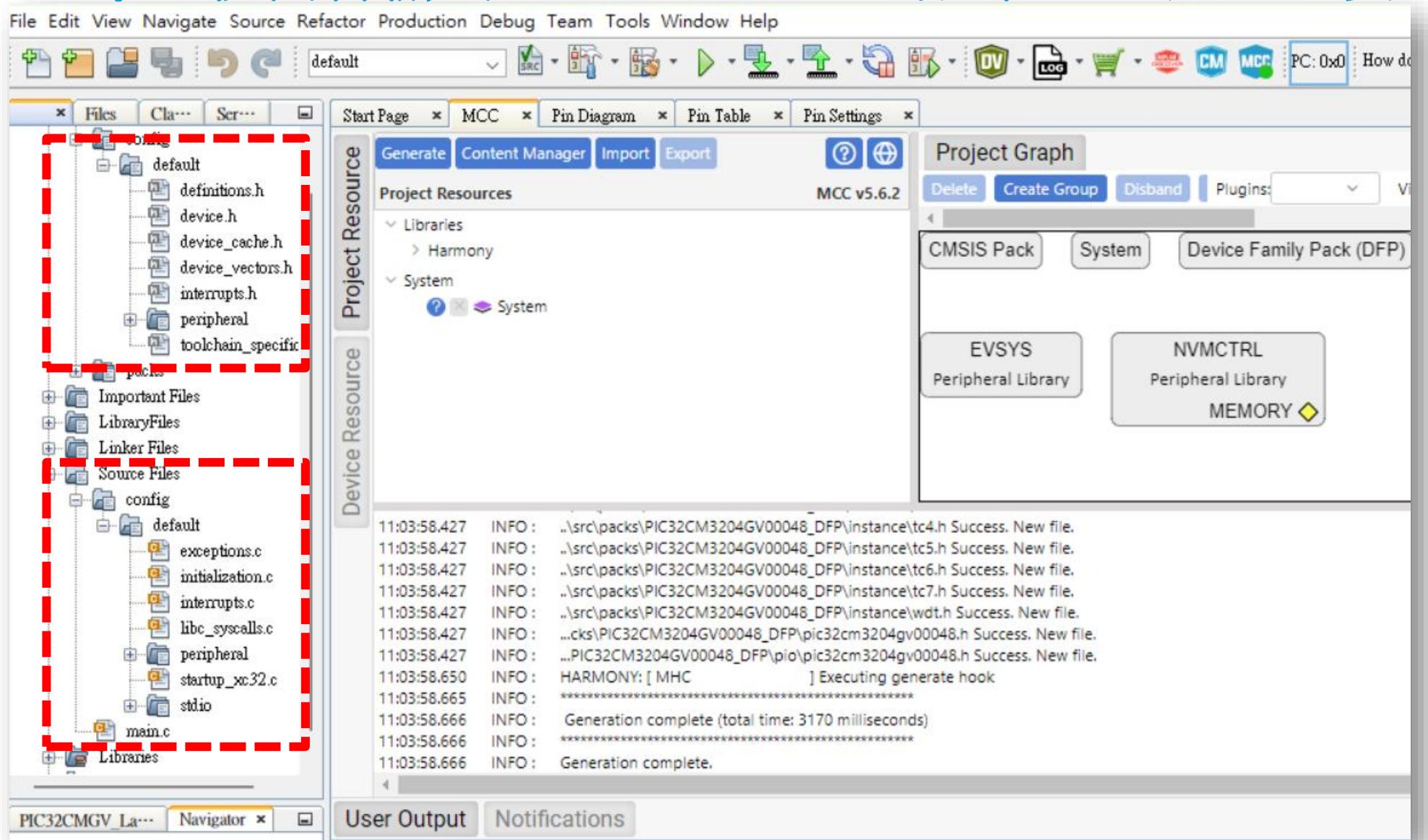
Lab1 – MPLAB® X IDE project creation & Systick

回到 MCC 頁面，點選 Project Resource 中的 **Generate**



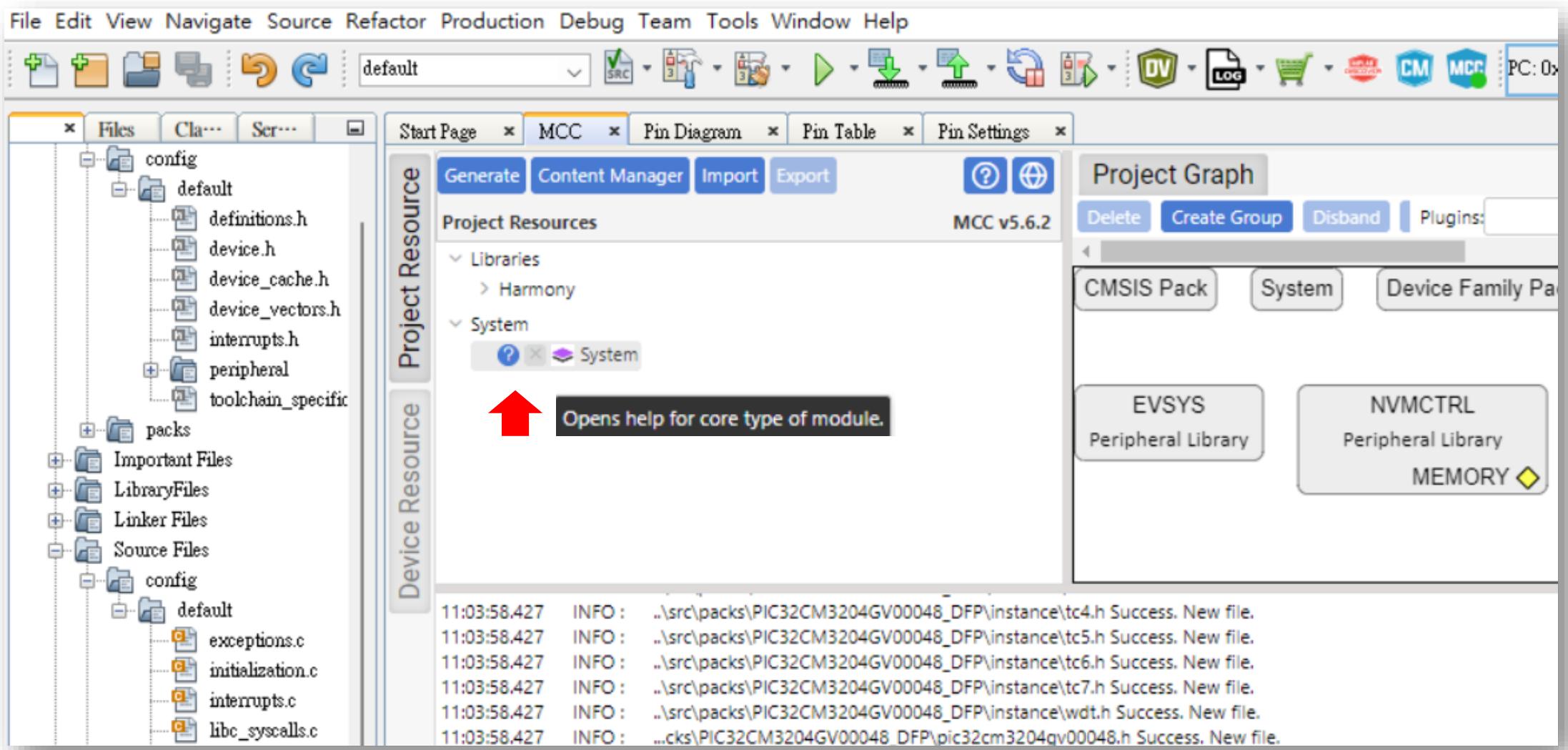
Lab1 – MPLAB® X IDE project creation & Systick

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



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

The screenshot shows a web browser window displaying the Microchip MPLAB Harmony Peripheral Libraries documentation. The URL in the address bar is [microchip.com](#). The page title is "MPLAB® Harmony Peripheral Libraries". The left sidebar contains a navigation tree with the following items:

- > 2.109 Single-Edge Nibble Transmission (SENT)
- < > 2.110 Serial Communication Interface (SERCOM)
- > 2.111 Shutdown Controller (SHDWC)
- > 2.112 I2C SMBUS
- > 2.113 Static Memory Controller (SMC)
- > 2.114 Serial Peripheral Interface (SPI)
- > 2.115 Serial Quad Interface (SQI)
- 2.116 STDIO
- > 2.117 Supply Controller (SUPC)
- 2.118 System** (selected)
- > 2.119 System timer (SysTick)
- > 2.120 Timer Counter (TC)

The main content area displays a table of peripheral libraries:

<ul style="list-style-type: none">Cache Controller (Cache)Cortex-M Cache Controller (CMCC)	Cache controller peripheral library for the target device.
<ul style="list-style-type: none">Direct Memory Access Controller (DMA)Direct Memory Access Controller (DMAC)Extensible DMA Controller (XDMAC)	DMA controller peripheral library for the target device.
<ul style="list-style-type: none">MIPS Core Timer (CORETIMER)System timer (SysTick) (highlighted with a red box)ARM Cortex A Generic timer (GENERIC_TIMER)	Core specific timer peripheral library for the target device.
<ul style="list-style-type: none">Watchdog Timer (WDT)Dual Watchdog Timer (DWDT)Dead Man Timer (DMT)	Watchdog timer peripheral library for the target device.

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.

```
/* 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();
}
```

Copy 

Lab1 – MPLAB® X IDE project creation & Systick

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

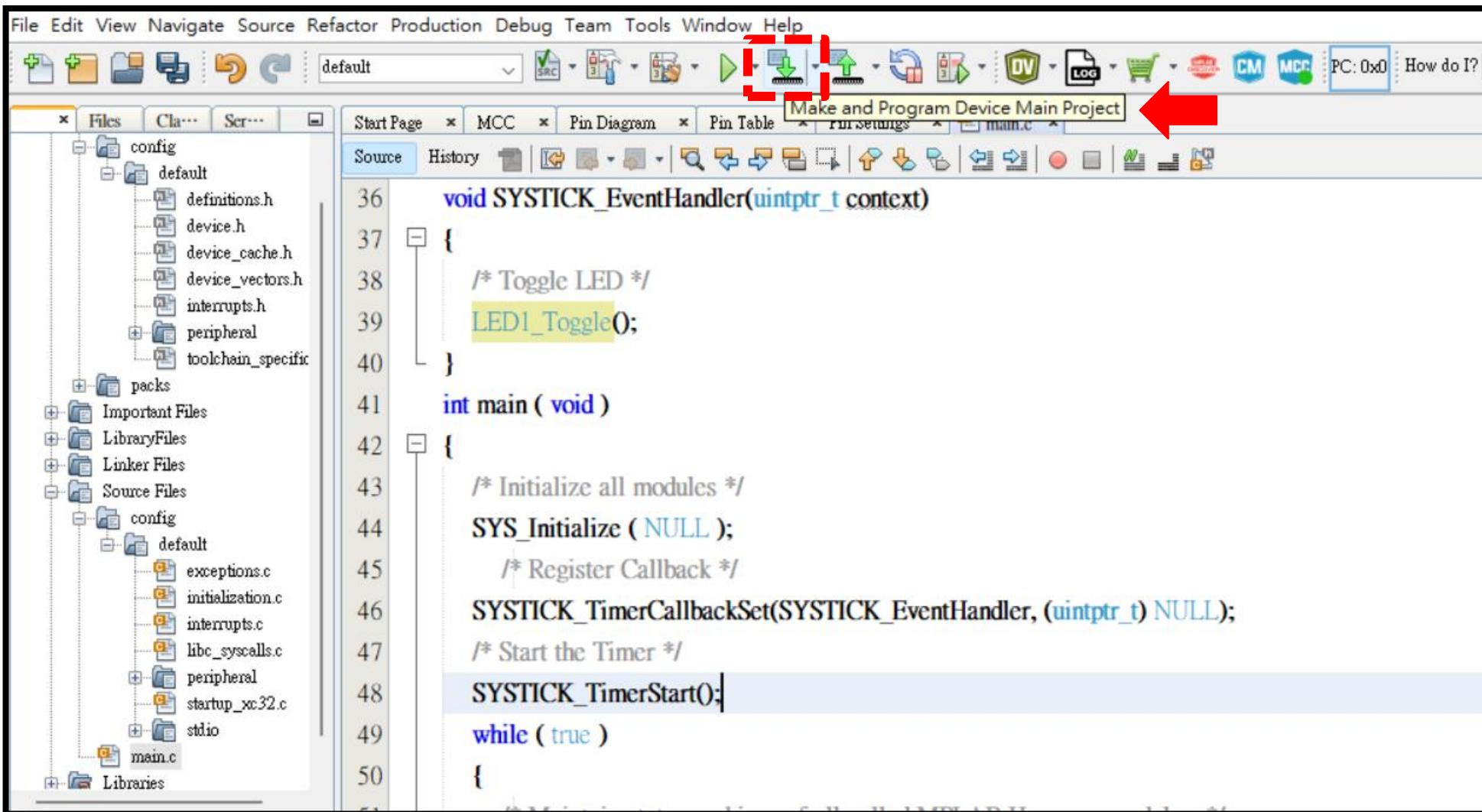
The screenshot shows the MPLAB X IDE interface. The left pane displays the project structure under 'Source Files' and 'config'. The right pane shows the source code for 'main.c'. The code implements a SYSTICK interrupt handler to toggle an LED and starts the timer in the main function.

```
void SYSTICK_EventHandler(uintptr_t context)
{
    /* Toggle LED */
    LED1_Toggle0;
}

int main ( void )
{
    /* Initialize all modules */
    SYS_Initialize ( NULL );
    /* Register Callback */
    SYSTICK_TimerCallbackSet(SYSTICK_EventHandler, (uintptr_t) NULL);
    /* Start the Timer */
    SYSTICK_TimerStart();
    while ( true )
    {
        /* Maintain state machines of all polled MPLAB Harmony modules. */
        SYS_Tasks ();
    }
}
```

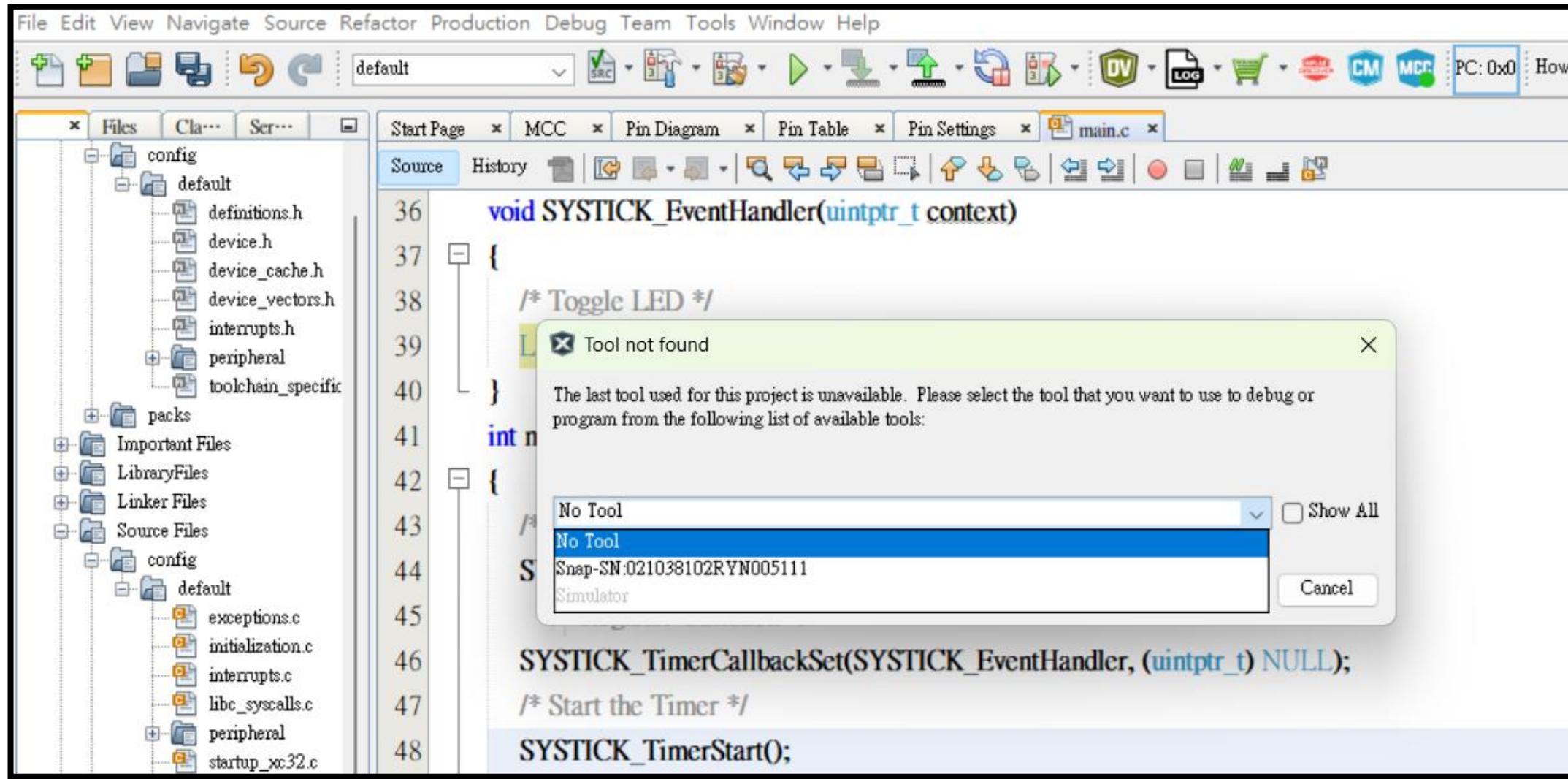
Lab1 – MPLAB® X IDE project creation & Systick

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

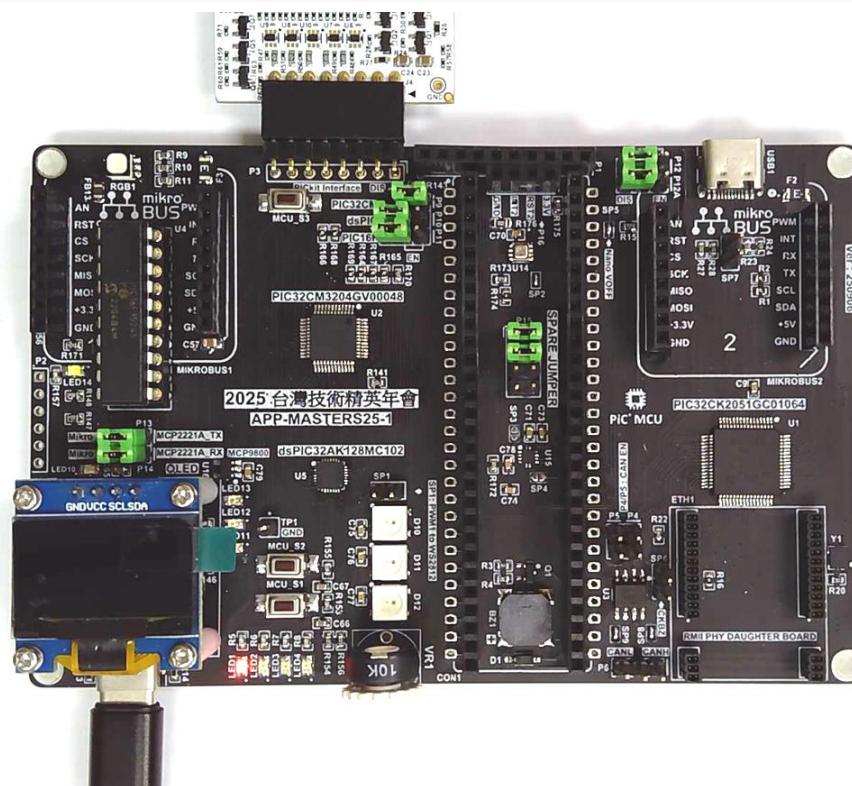


Lab1 – MPLAB® X IDE project creation & Systick

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

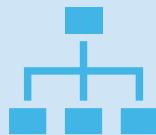
Due to the large memory ranges on this device, only the areas of memory that have been loaded with code (via the build process
Programming/Verify complete
```



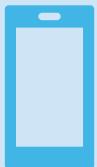
Lab 1 Summary



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

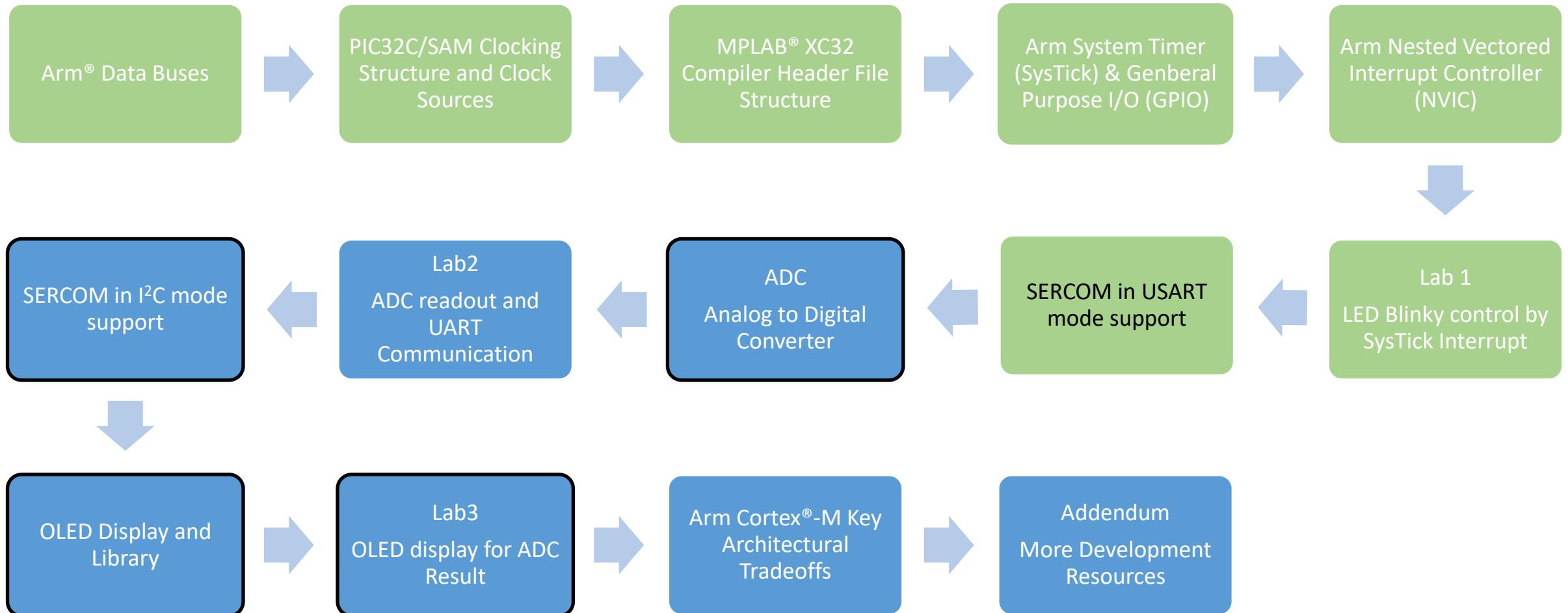


Check and Config Clock – Make sure CPU runs at 48 Mhz
Config SysTick and learn how to implement “Interrupt”
Config I/O Pins and use Custom Name for easy access



Program the device by using Microchip Tool
Viewed the main device and component header files

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

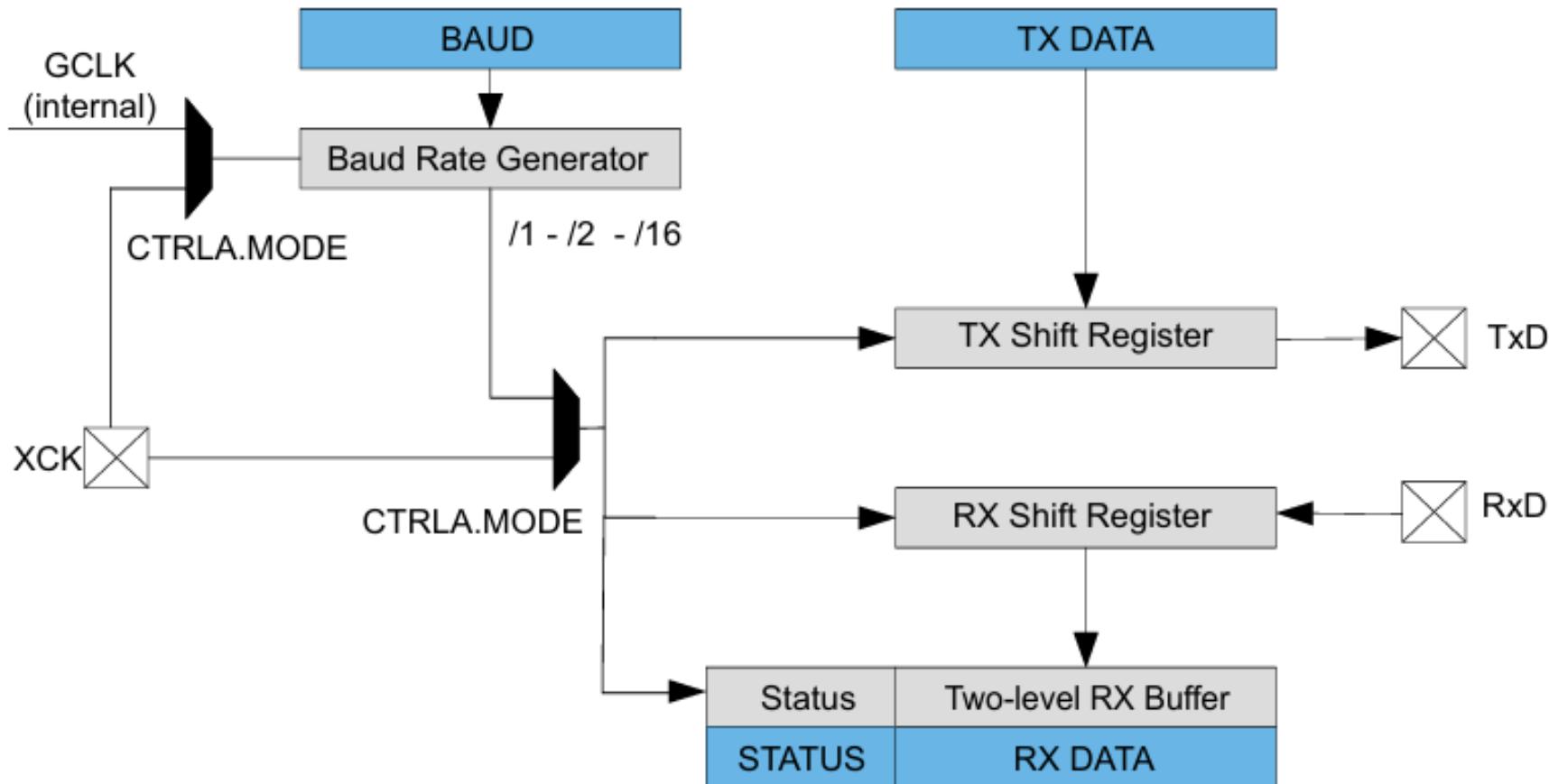
SERCOM Synchronous and Asynchronous Receiver and Transmitter

- **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

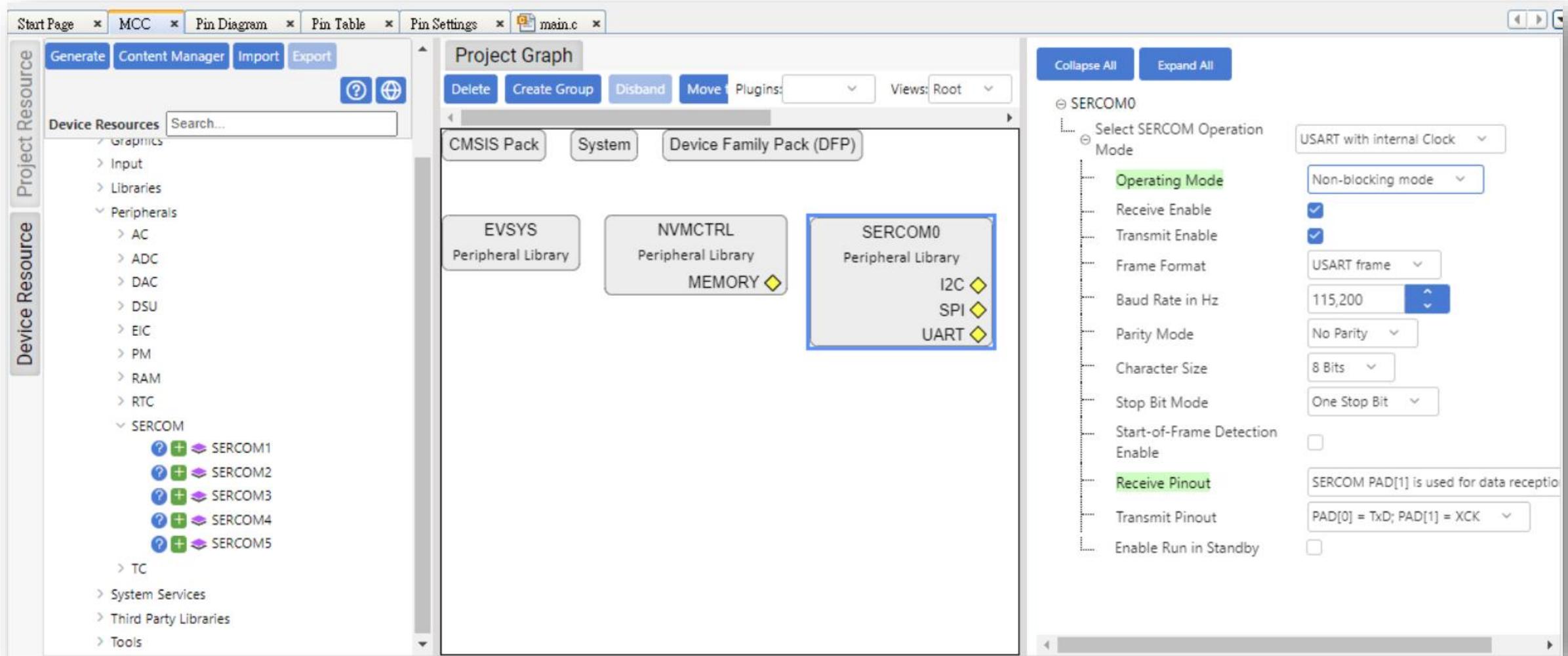
SERCOM - USART mode

SERCOM Synchronous and Asynchronous Receiver and Transmitter

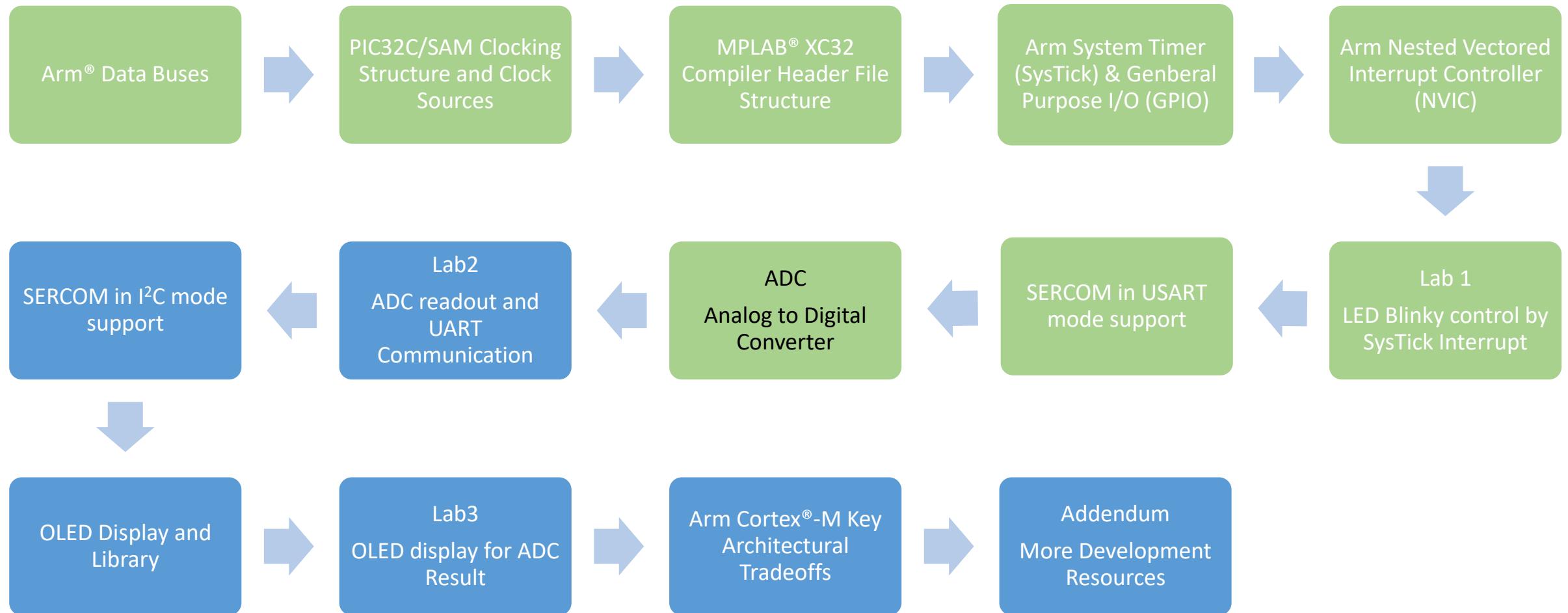


SERCOM - USART mode

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



Class Outline





ADC

Analog-to-Digital Converter

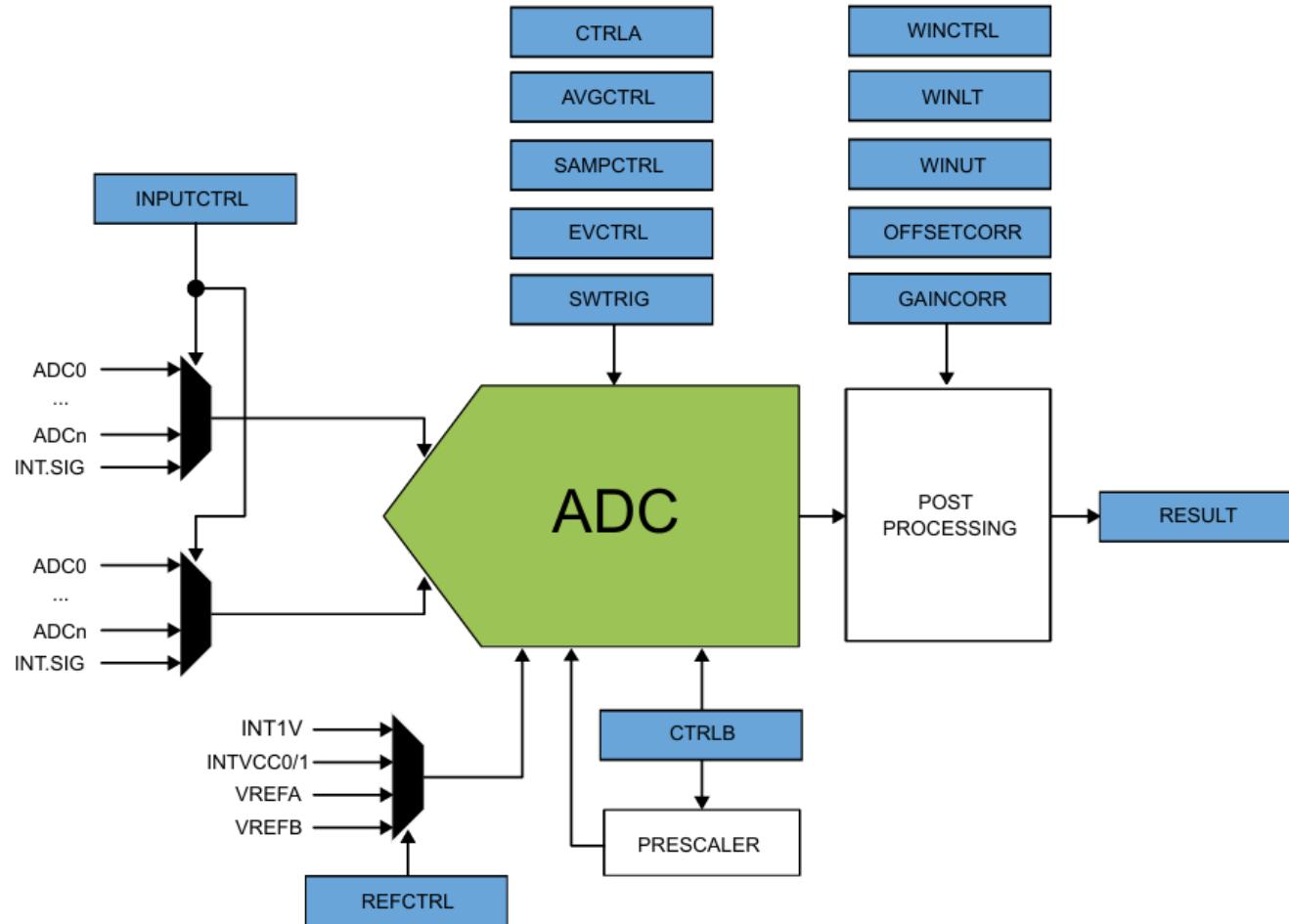
ADC - Analog-to-Digital Converter

- 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

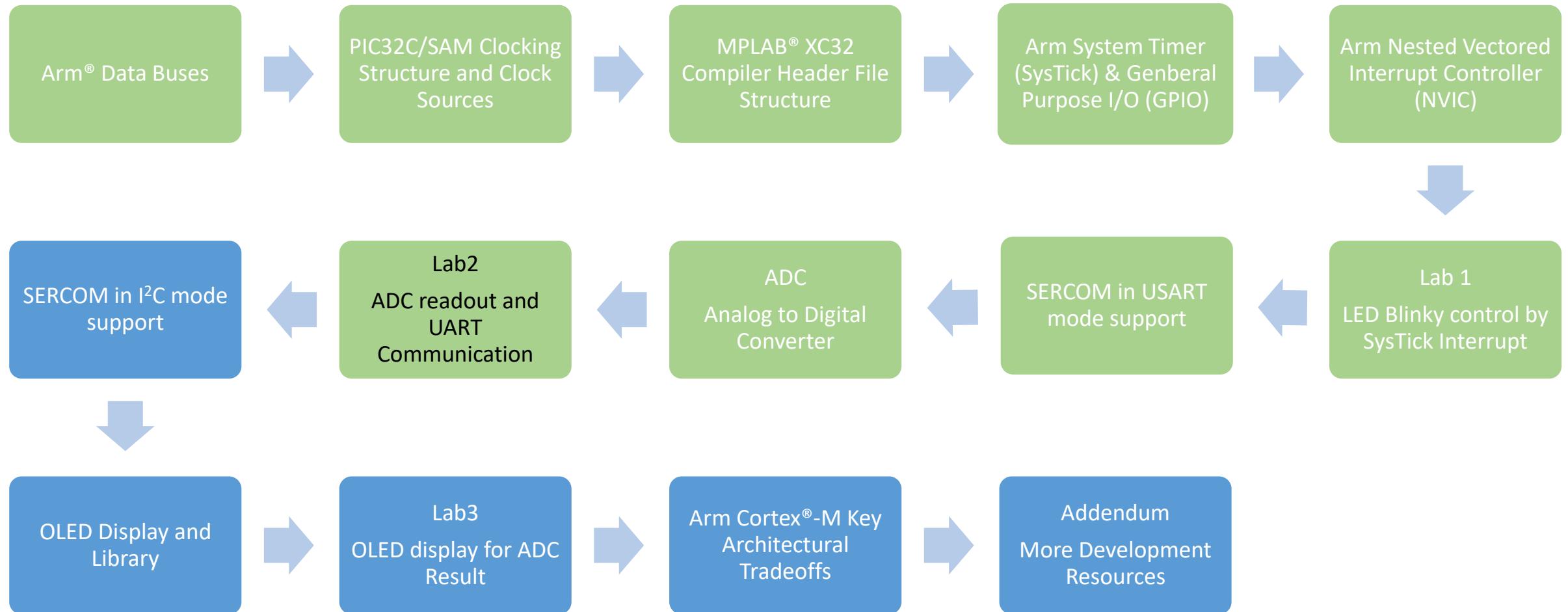
ADC

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

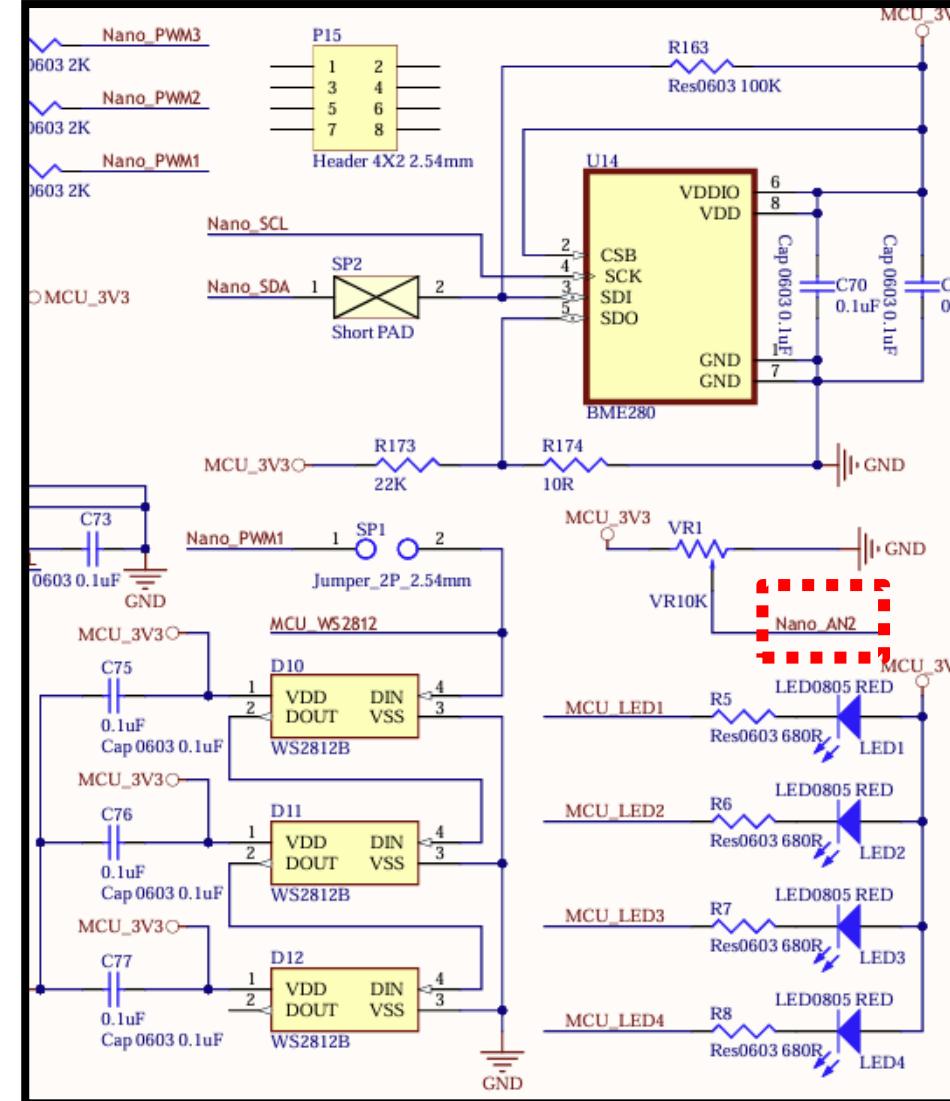
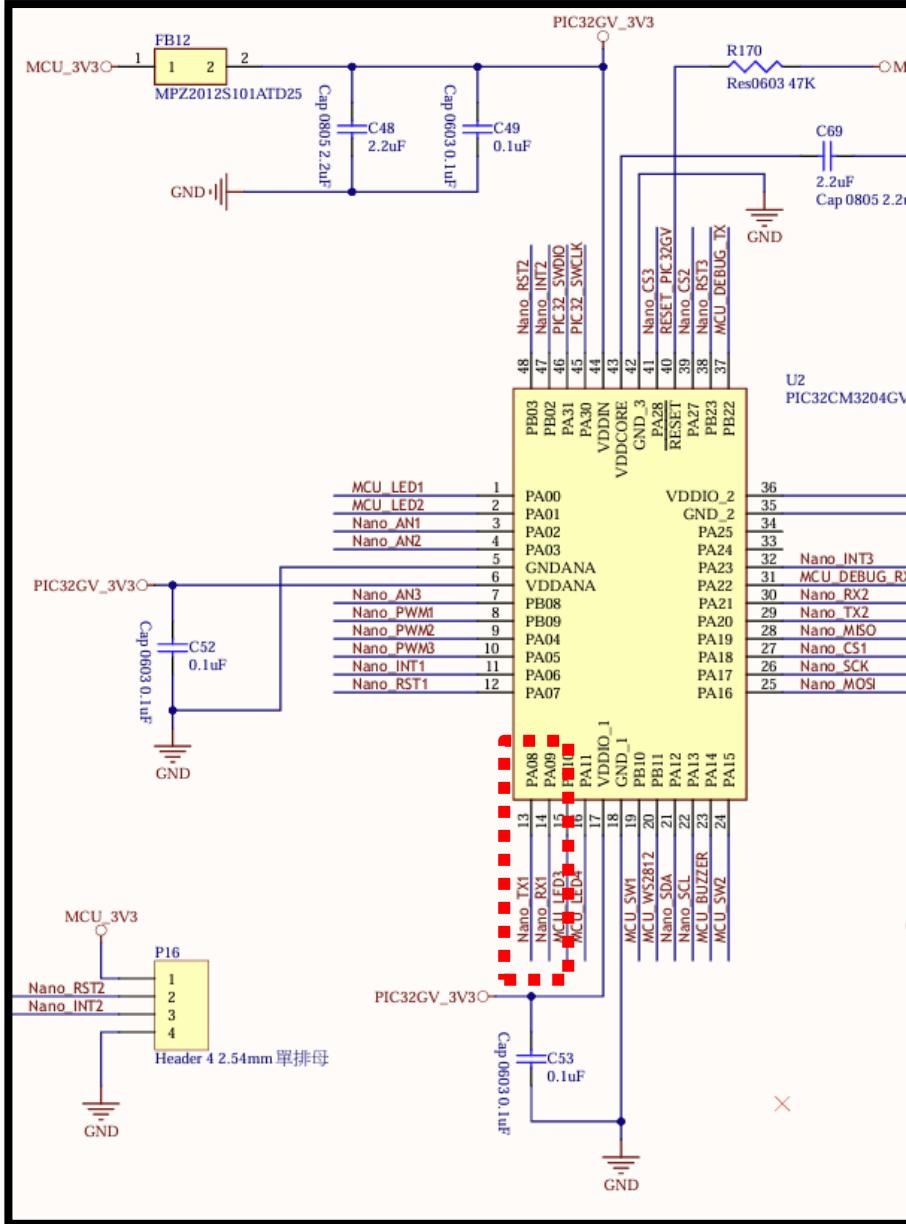


Review the generated
PLIBs



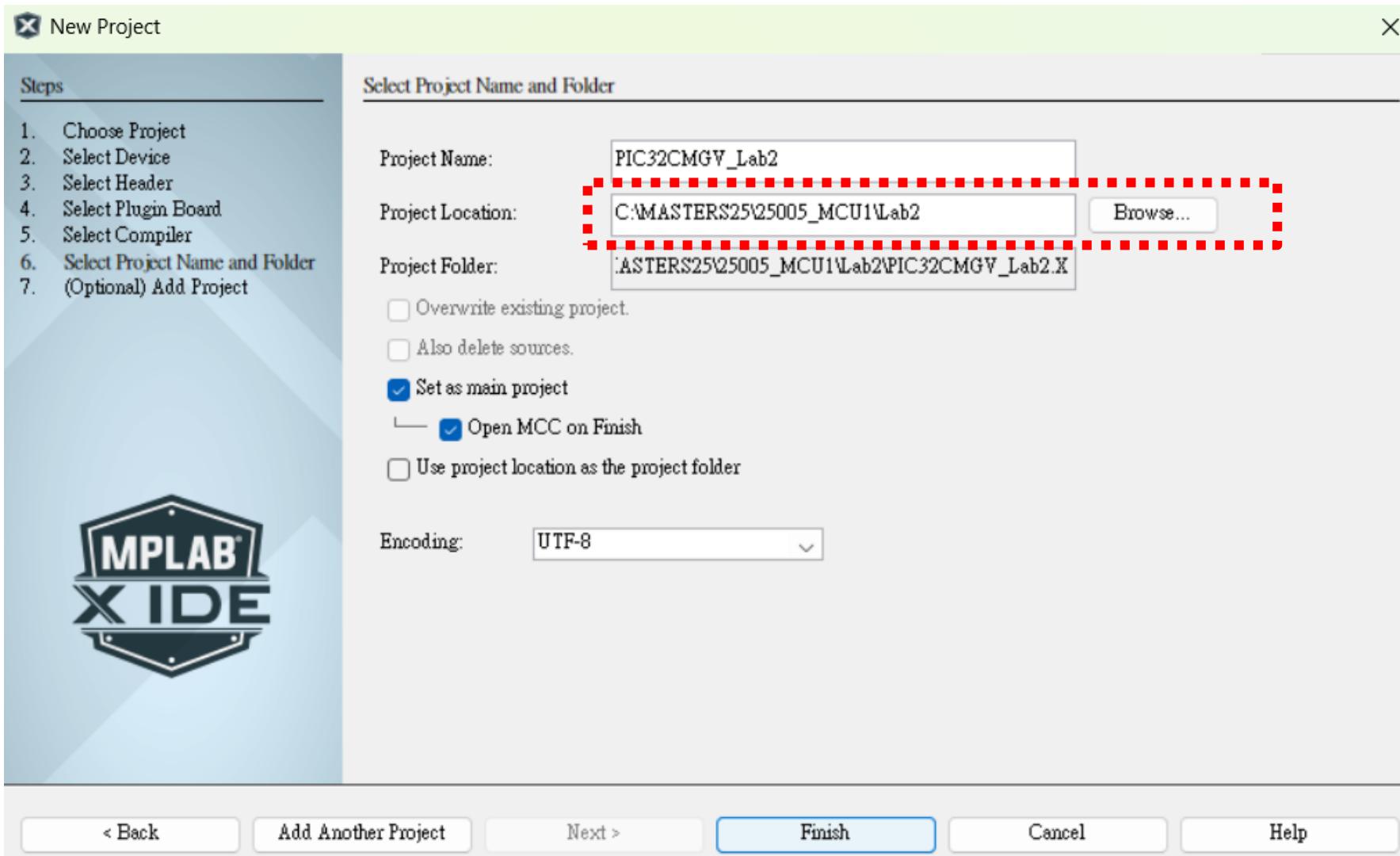
Send VR(ADC) result
through UART to
MPLAB® Data Visualizer

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



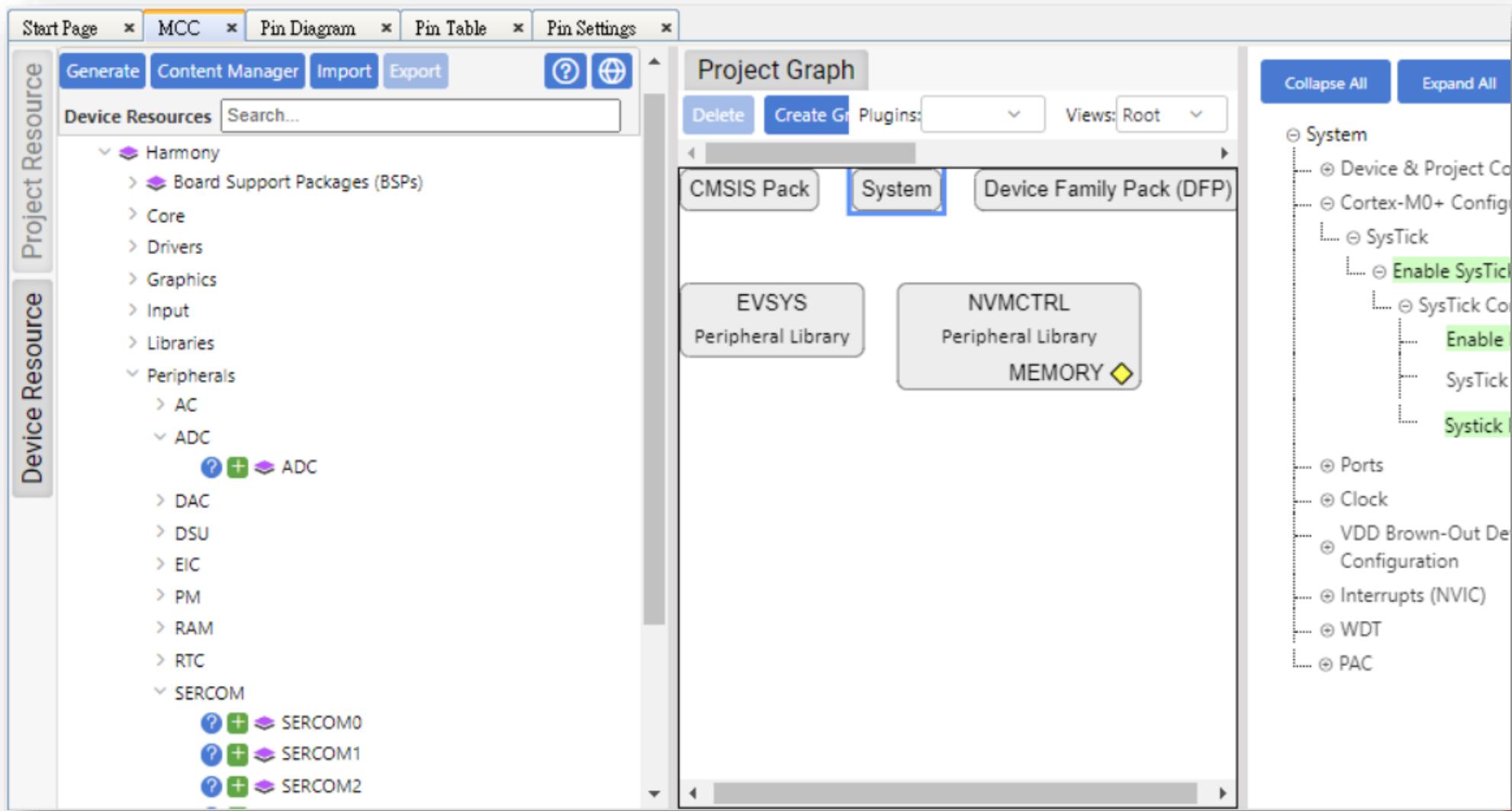
Lab2 – Interface to the World

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

Pin Settings									
Order:	Pins	Table View	<input checked="" type="checkbox"/> 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	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
2	PA01		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
3	PA02		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
4	PA03	VR1	ADC_AIN1	Analog	High Impedance	n/a	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
5	GNDANA			Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
6	VDDANA			Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
7	PB08		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
8	PB09		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
9	PA04		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
10	PA05		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
11	PA06		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
12	PA07		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
13	PA08		SERCOM0_PAD0	Digital	High Impedance	n/a	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
14	PA09		SERCOM2_PAD1	Digital	High Impedance	n/a	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
15	PA10		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
16	PA11		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
17	VDDIO			Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
18	GNDIO			Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
19	PB10		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL

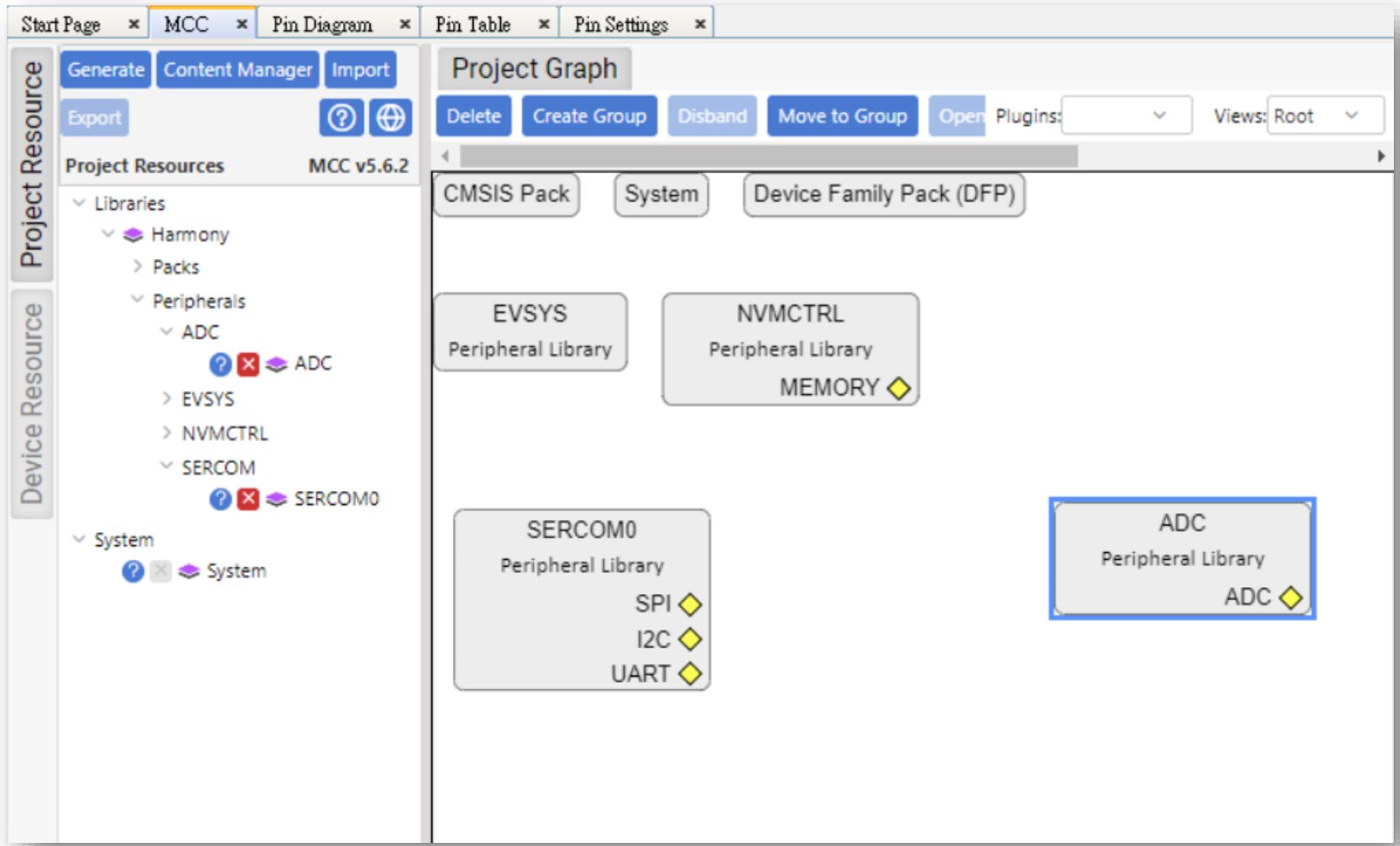
Lab2 – Interface to the World

在 Device Resource 中加入 ADC & SERCOM0



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)

The screenshot shows the Microchip Studio interface with the Project Graph on the left and the ADC configuration on the right.

Project Graph:

- Buttons: Delete, Create Group, Disband, Move to Group, Open, Plugins: [dropdown], Views: Root
- Groups:
 - CMSIS Pack
 - System
 - Device Family Pack (DFP)
 - EVSYS Peripheral Library
 - NVMCTRL Peripheral Library
 - MEMORY ◀
 - SERCOM0 Peripheral Library
 - SPI ◀
 - I2C ◀
 - UART ◀
 - ADC Peripheral Library
 - ADC ◀

ADC Configuration:

- ADC** settings:
 - Select Prescaler: Peripheral clock divided by 32
 - Select Sample Length (half ADC clock cycles): 4
 - Conversion Time: **** Conversion Time is 6.66666666667 uS ****
 - Select Gain: 1/2x
 - Select Reference: 1/2 VDDANA (only for VDDANA > 2.0V)
 - Select Conversion Trigger: SW Trigger
- Channel Configuration**:
 - Select Positive Input: ADC AIN1 Pin
 - Select Negative Input: Internal ground
 - Number of inputs to scan: 0
- Result Configuration**:
 - Select Result Resolution: 12-bit result
 - Left Aligned Result:
 - Enable Result Ready Interrupt:
 - Enable Result Ready Event:

Lab2 – Interface to the World

UART 的規劃

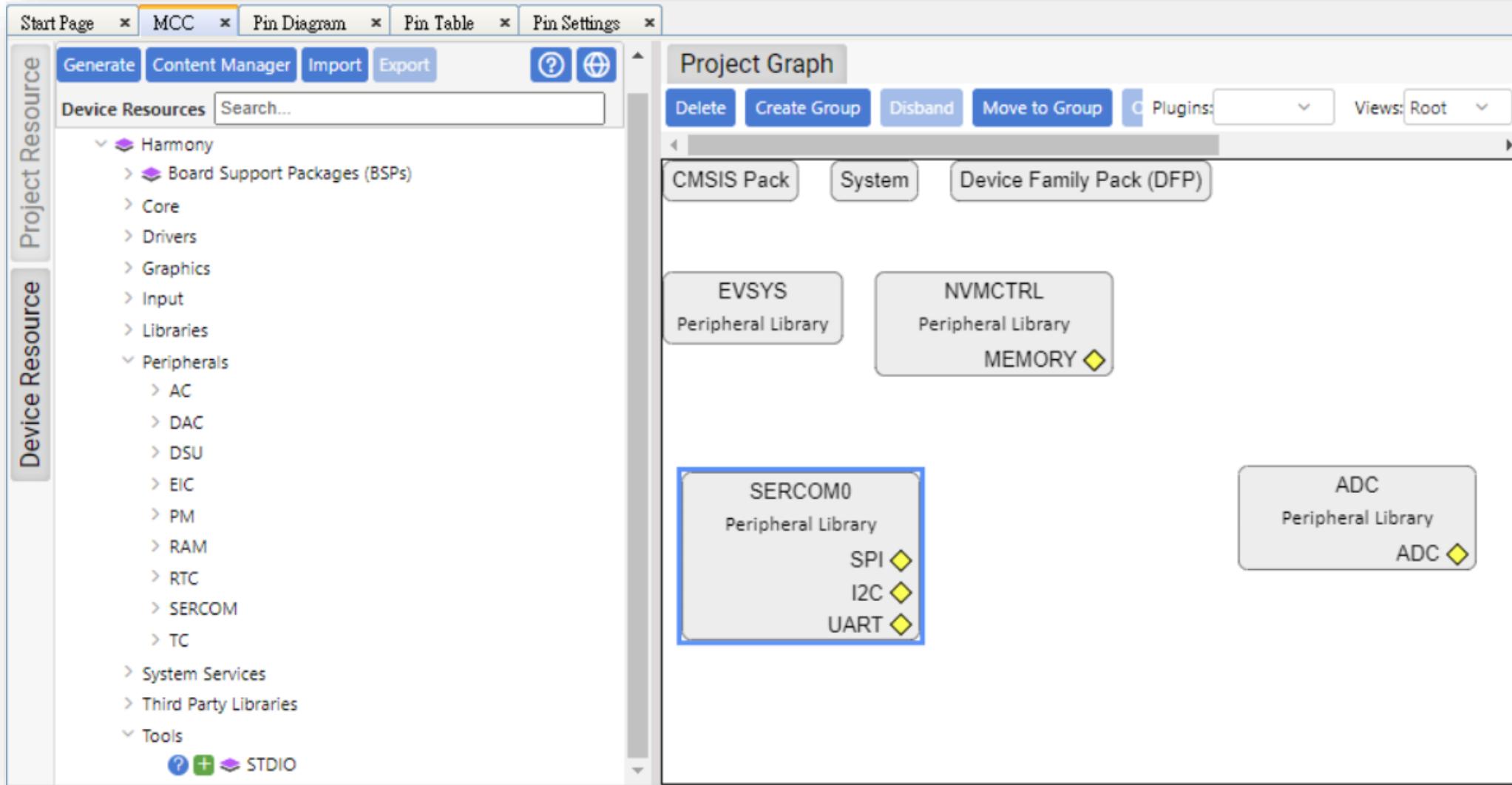
The screenshot shows the Microchip Studio Project Graph interface. On the left, the 'Project Graph' tab is selected, displaying various peripheral library components: CMSIS Pack, System, Device Family Pack (DFP), EVSYS Peripheral Library, NVMCTRL Peripheral Library, MEMORY, SERCOM0 Peripheral Library (which is highlighted with a blue border), ADC Peripheral Library, and a component labeled 'ADC' with a yellow diamond icon.

On the right, the configuration pane is open for the SERCOM0 peripheral. It includes 'Collapse All' and 'Expand All' buttons at the top. The 'Select SERCOM Operation Mode' section is expanded, showing the following settings:

- Operating Mode: Non-blocking mode (checked)
- Receive Enable: checked
- Transmit Enable: checked
- Frame Format: USART frame
- Baud Rate in Hz: 115,200 (selected)
- Parity Mode: No Parity
- Character Size: 8 Bits
- Stop Bit Mode: One Stop Bit
- Start-of-Frame Detection: Enable (unchecked)
- Receive Pinout: SERCOM PAD[1] is used for data reception
- Transmit Pinout: PAD[0] = TxD; PAD[1] = XCK
- Enable Run in Standby: (unchecked)

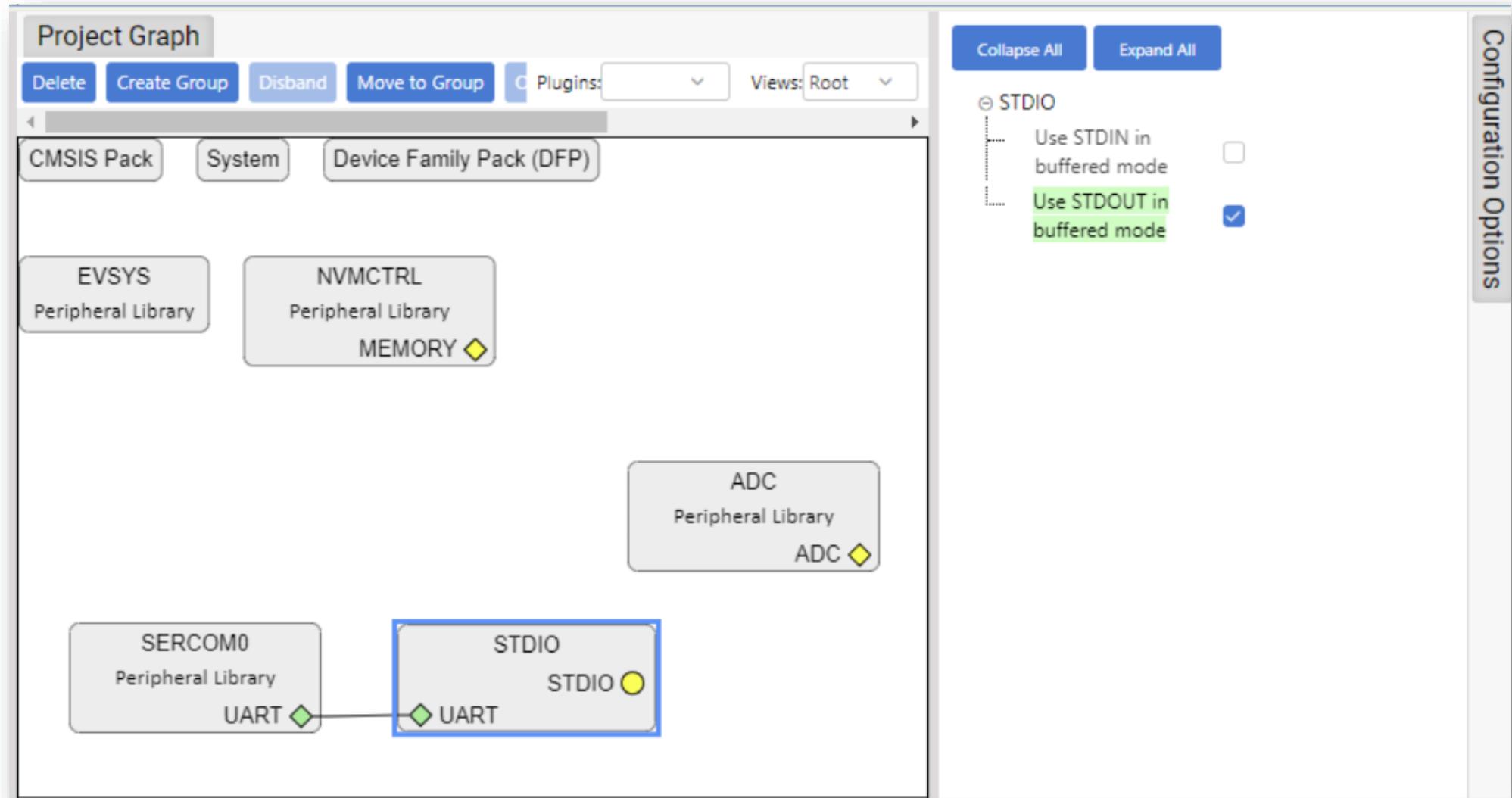
Lab2 – Interface to the World

將 STDIO 加入以便於將 UART 的資料導向 stdout



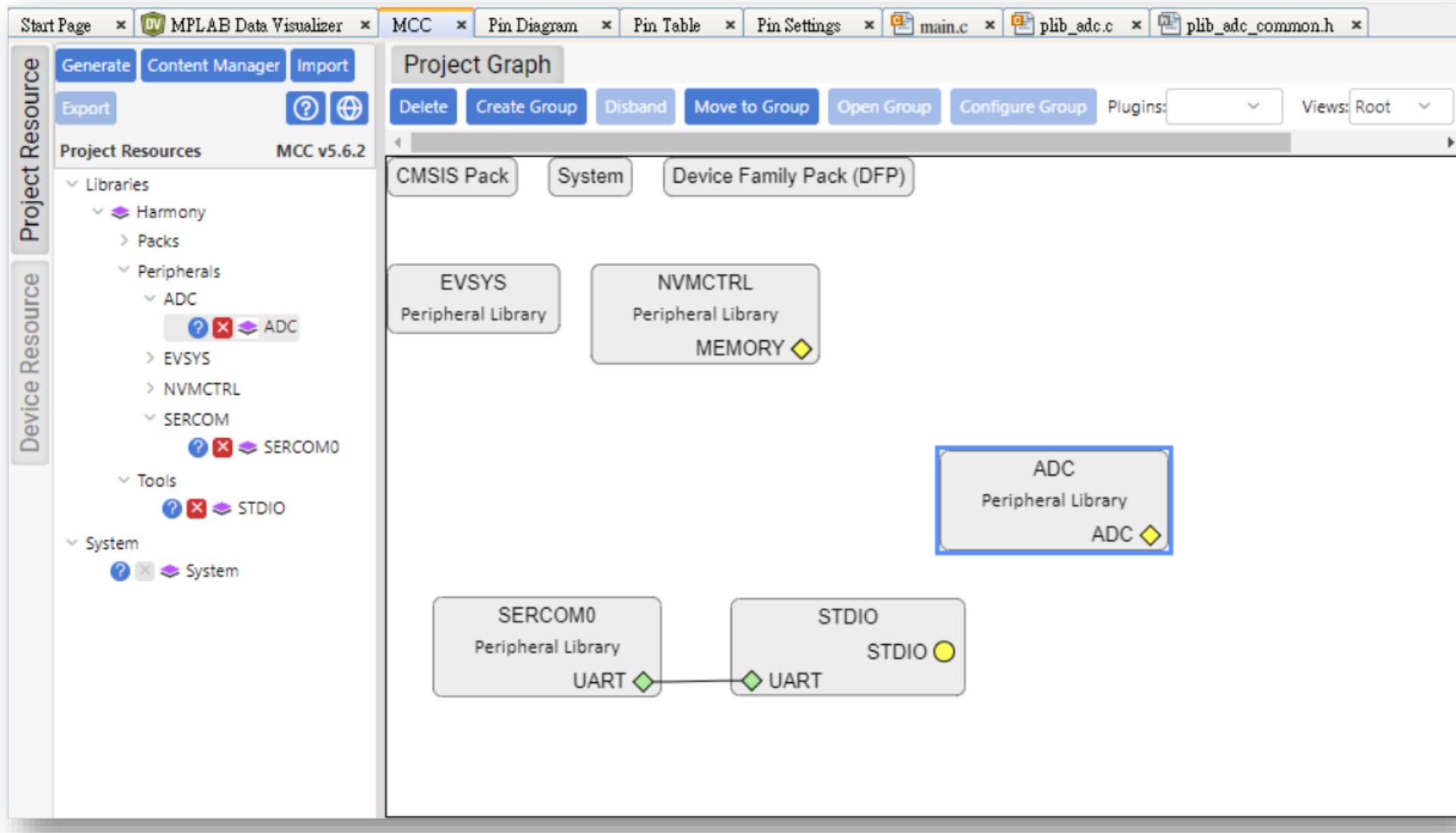
Lab2 – Interface to the World

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



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



MPLAB® Harmony Peripheral Libraries

[Home](#) / [2 API Documentation](#) / [2.3 Analog Digital Converter \(ADC\)](#)

1 MPLAB® Harmony Peripheral Libraries

- < [2 API Documentation](#)
 - > [2.1 Analog Comparators \(AC\)](#)
 - > [2.2 Analog Comparator Controller \(ACC\)](#)
 - ✓ [2.3 Analog Digital Converter \(ADC\)](#)
 - [2.3.1 ADCx_CallbackRegister Function](#)
 - [2.3.2 ADCx_ChannelSelect Function](#)
 - [2.3.3 ADCx_ComparisonWindowSet Function](#)
 - [2.3.4 ADCx_ConversionResultGet Function](#)
 - [2.3.5 ADCx_ConversionSequencelsFinished Function](#)
 - [2.3.6 ADCx_ConversionStart Function](#)
 - [2.3.7 ADCx_ConversionStatusGet Function](#)
 - [2.3.8 ADCx_Disable Function](#)
 - [2.3.9 ADCx_Enable Function](#)
 - [2.3.10 ADCx_Initialize Function](#)
 - [2.3.11 ADCx_InterruptsClear Function](#)
 - [2.3.12 ADCx_InterruptsDisable Function](#)
 - [2.3.13 ADCx_InterruptsEnable Function](#)

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

Lab2 – Interface to the World

ADCx_ConversionResultGet Function 的使用範例

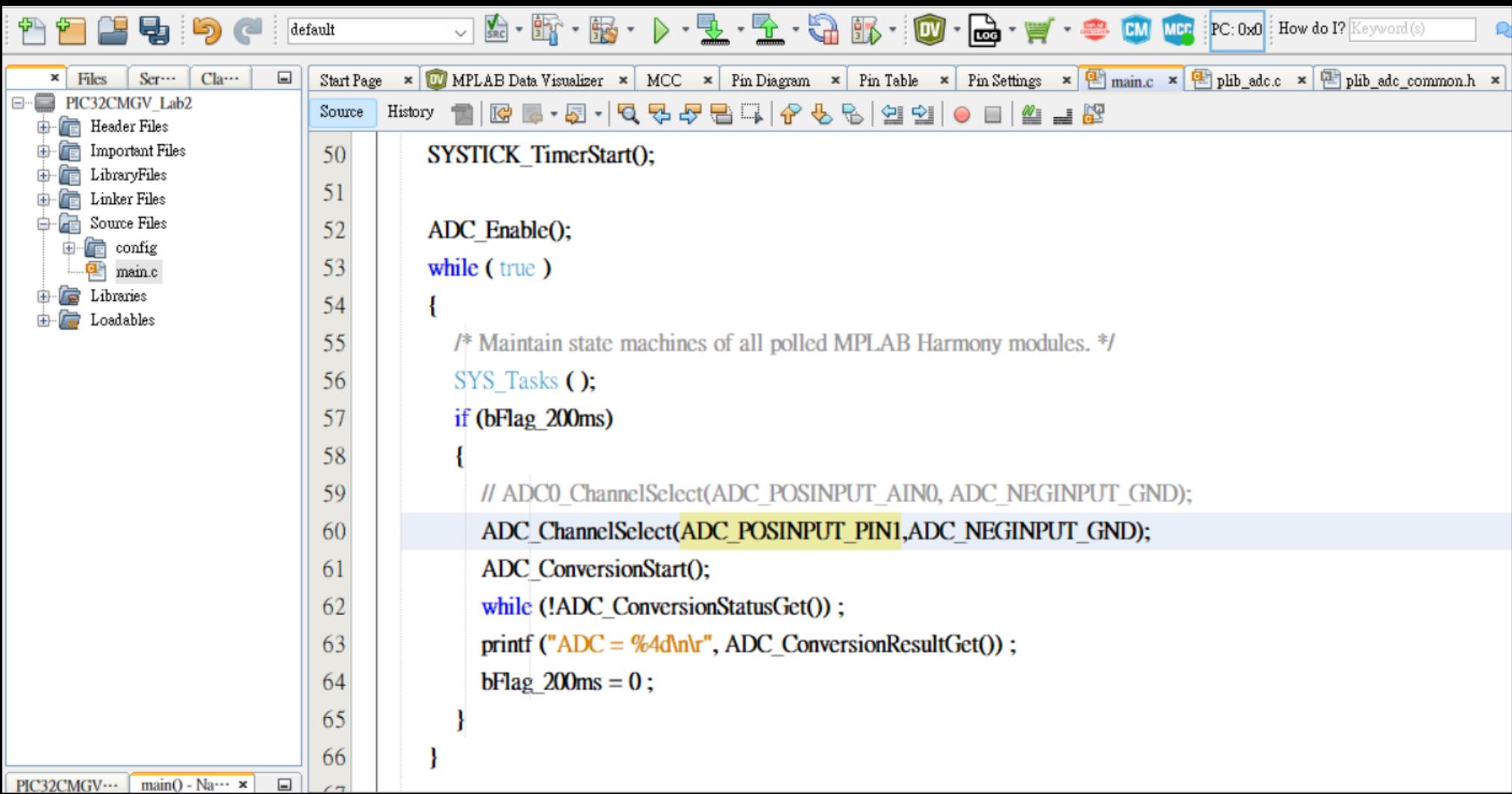
- 因為 PIC32CM3204 GV 系列只有一個 ADC，所以產生的 API 名稱為 **ADC_ConversionResultGet()**

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

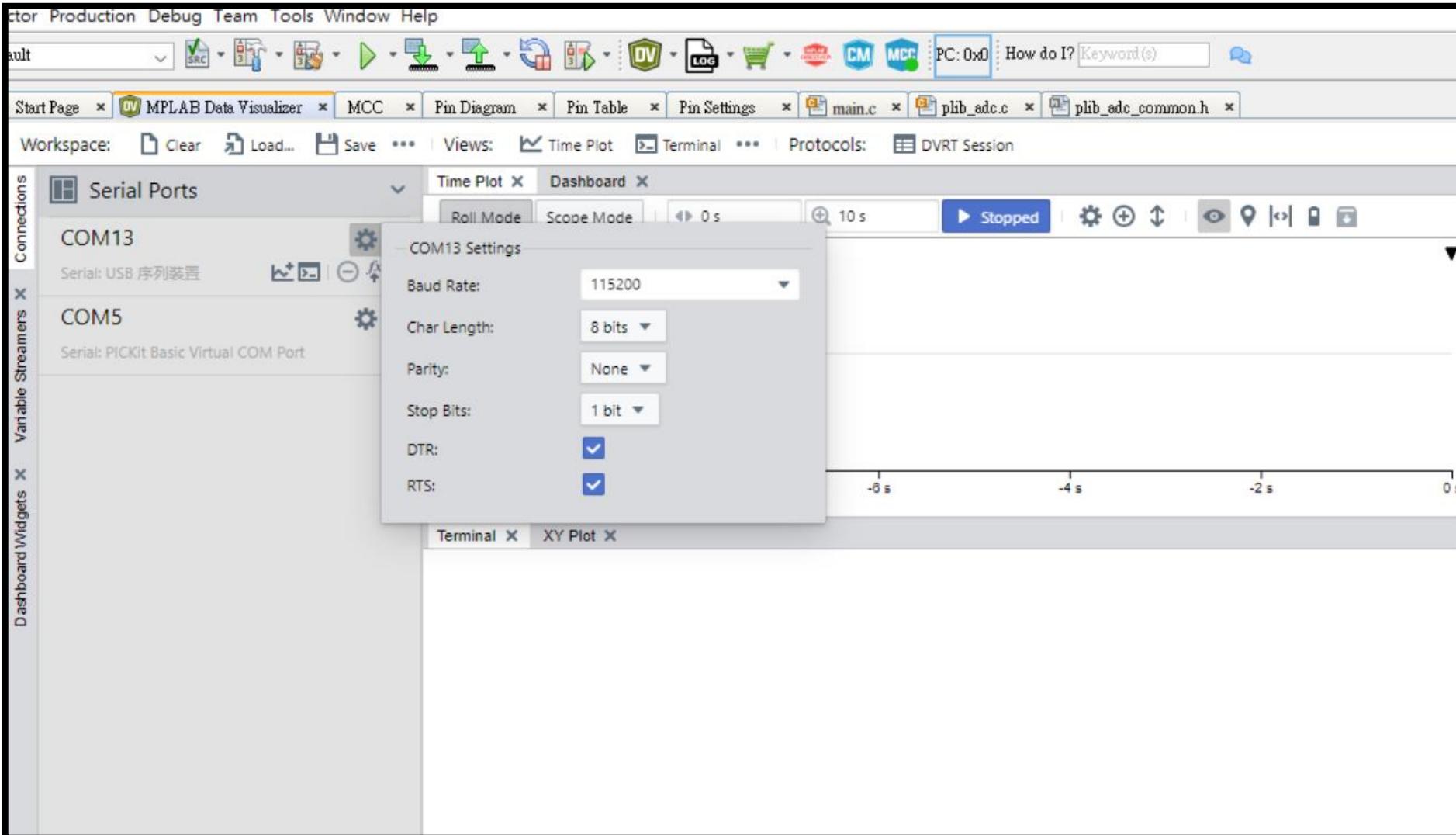


The screenshot shows the MPLAB X IDE interface with the main.c file open in the editor. The code implements a periodic ADC conversion task using the SysTick timer. It includes calls to SYSTICK_TimerStart(), ADC_Enable(), SYS_Tasks(), and ADC_ConversionStart(). A conditional block checks for the bFlag_200ms flag and performs an ADC conversion if it is set.

```
50     SYSTICK_TimerStart();
51
52     ADC_Enable();
53     while ( true )
54     {
55         /* Maintain state machines of all polled MPLAB Harmony modules. */
56         SYS_Tasks ();
57         if (bFlag_200ms)
58         {
59             // ADC0_ChannelSelect(ADC_POSINPUT_AIN0, ADC_NEGINPUT_GND);
60             ADC_ChannelSelect(ADC_POSINPUT_P1, ADC_NEGINPUT_GND);
61             ADC_ConversionStart();
62             while (!ADC_ConversionStatusGet());
63             printf ("ADC = %4d\n", ADC_ConversionResultGet());
64             bFlag_200ms = 0;
65         }
66     }
67 }
```

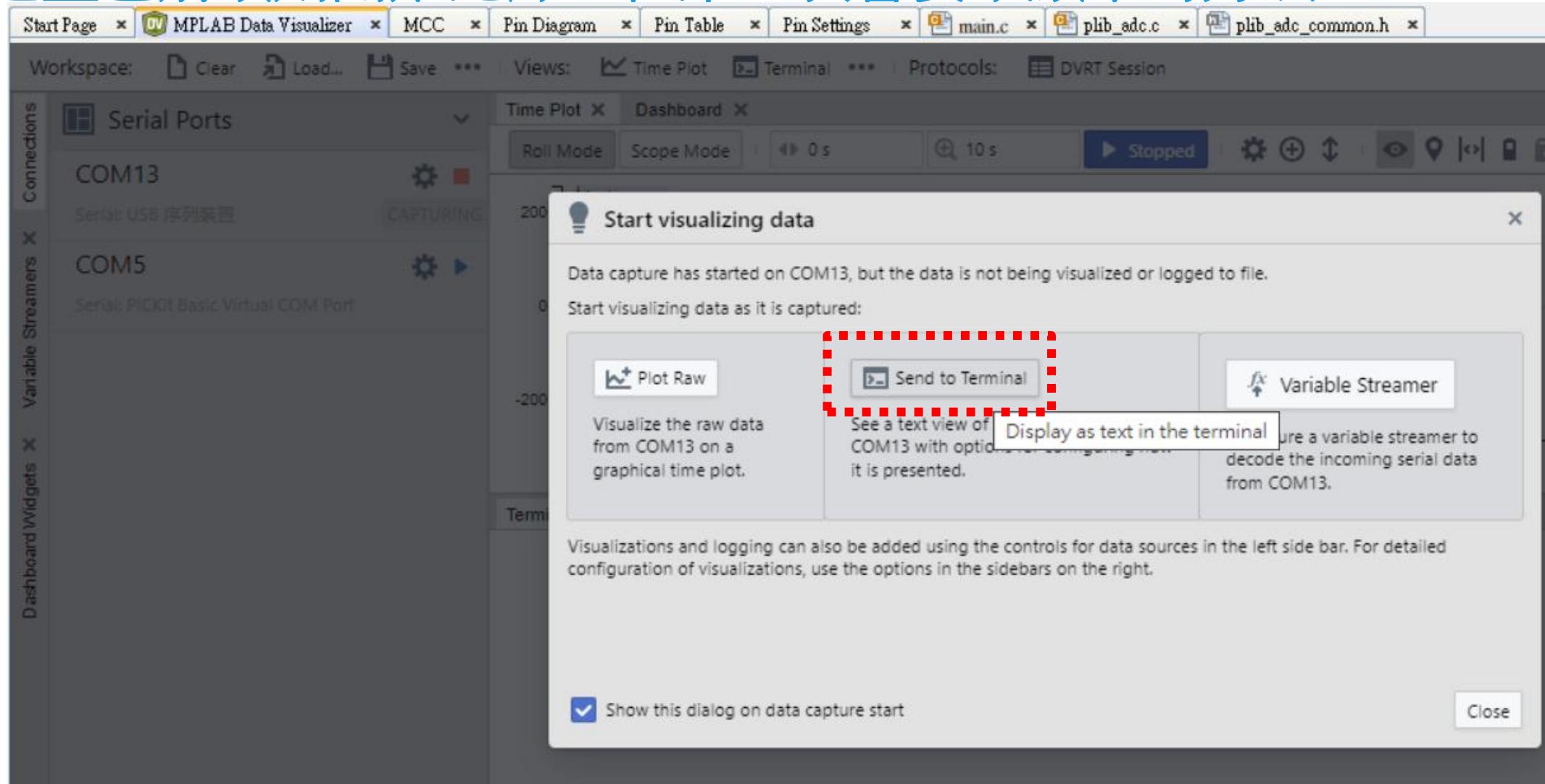
Lab2 – Interface to the World

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



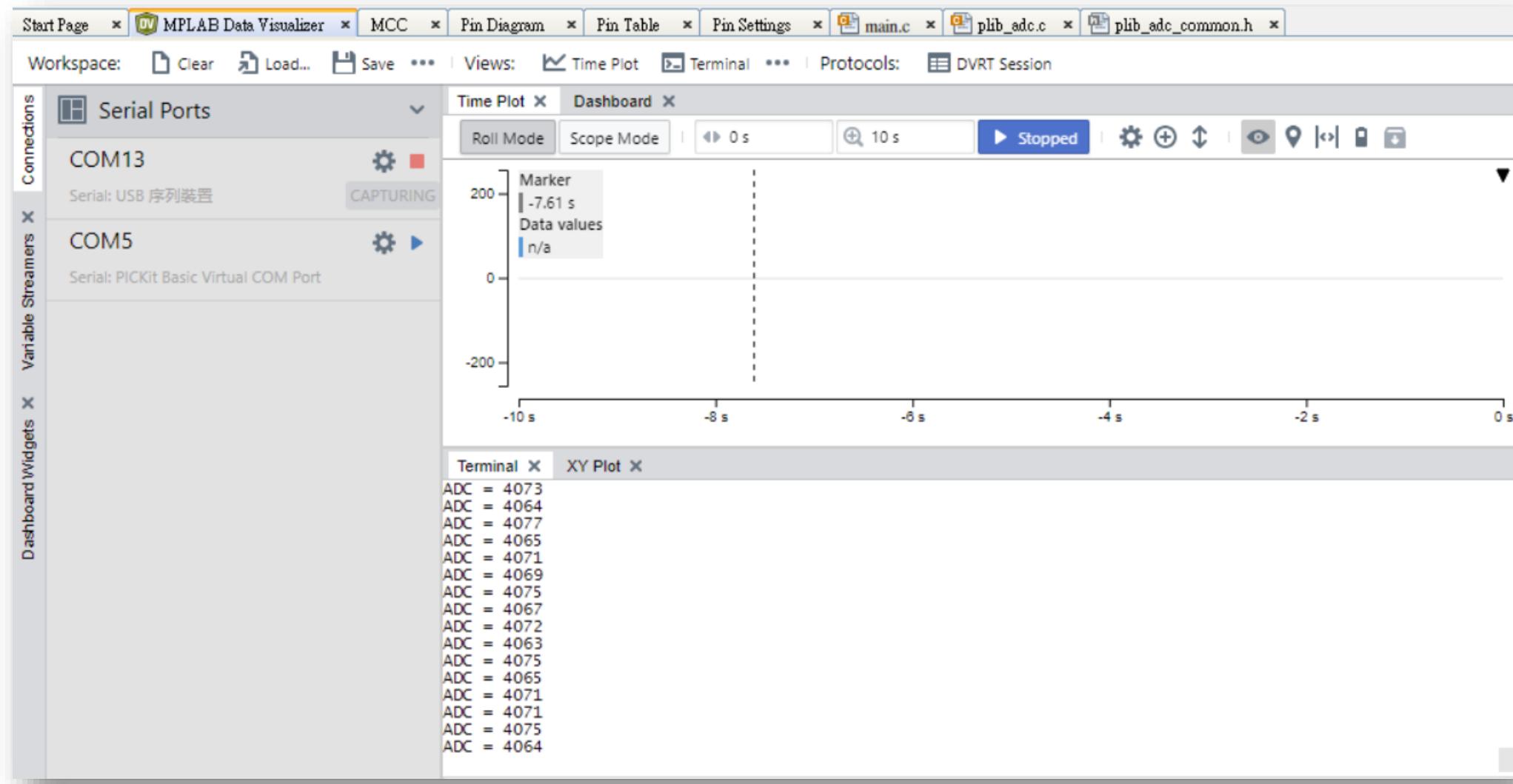
Lab2 – Interface to the World

點選藍色箭頭則開始記錄，但第一次會要求顯示的方法

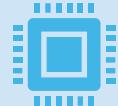


Lab2 – Interface to the World

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



Lab 2 Summary



Properly set up SERCOM to support UART Function



Set up ADC module with proper Gain and Reference

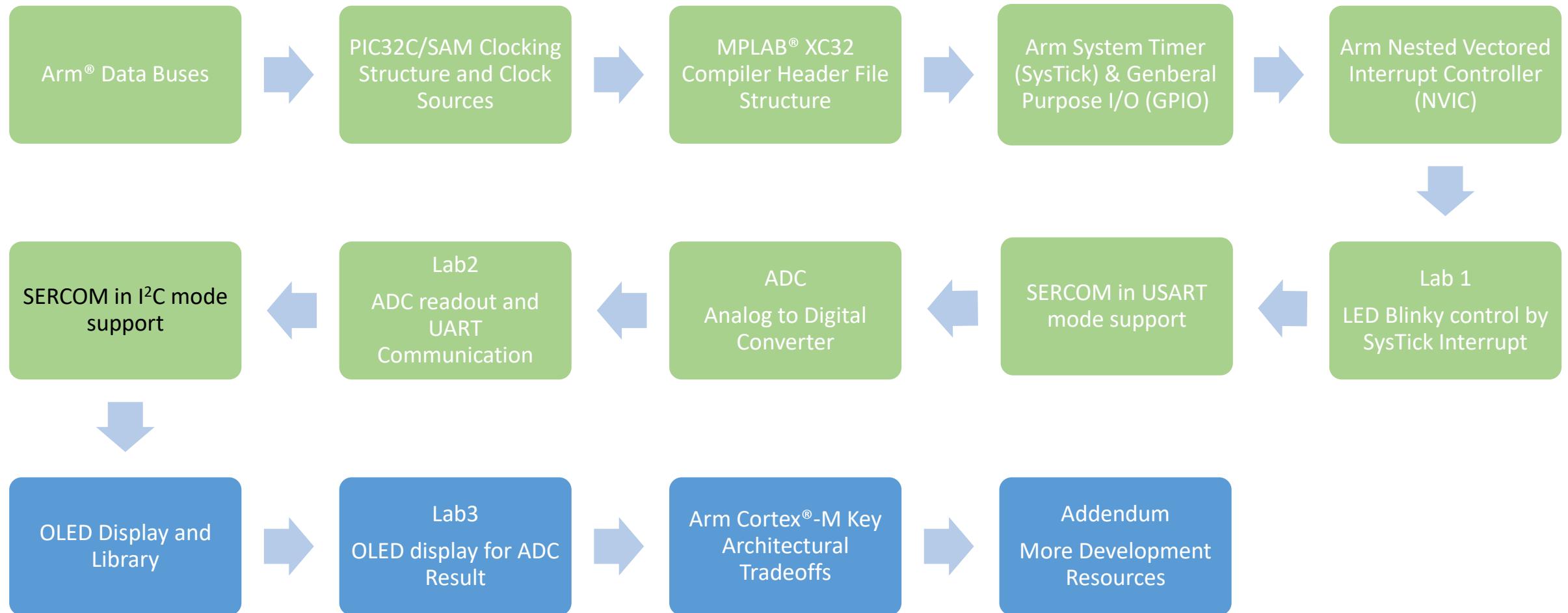


Reviewed generated PLIBs and added code



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

Class Outline





SERCOM in I2C mode

Serial Communication Interface

What is I²C

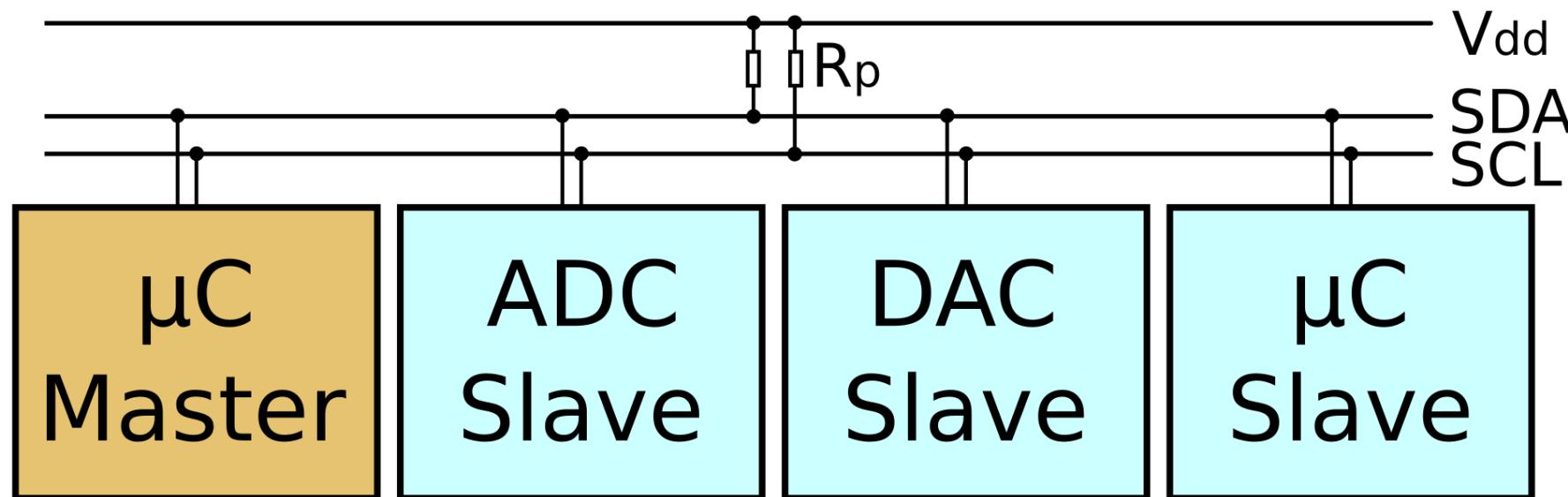
來自維基百科的資訊

- 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]。

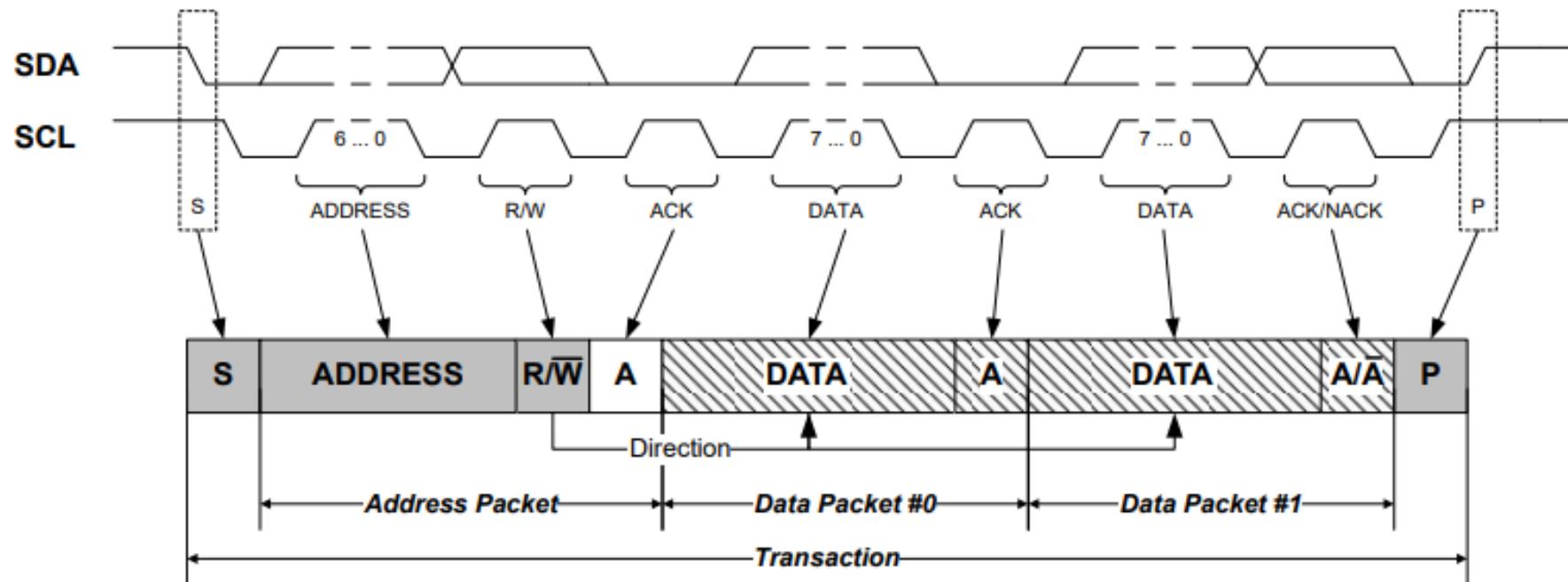
I²C 基礎知識

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

	START condition
	repeated START condition
	STOP condition

Data Package Direction

	Host Read
	Host Write

Acknowledge

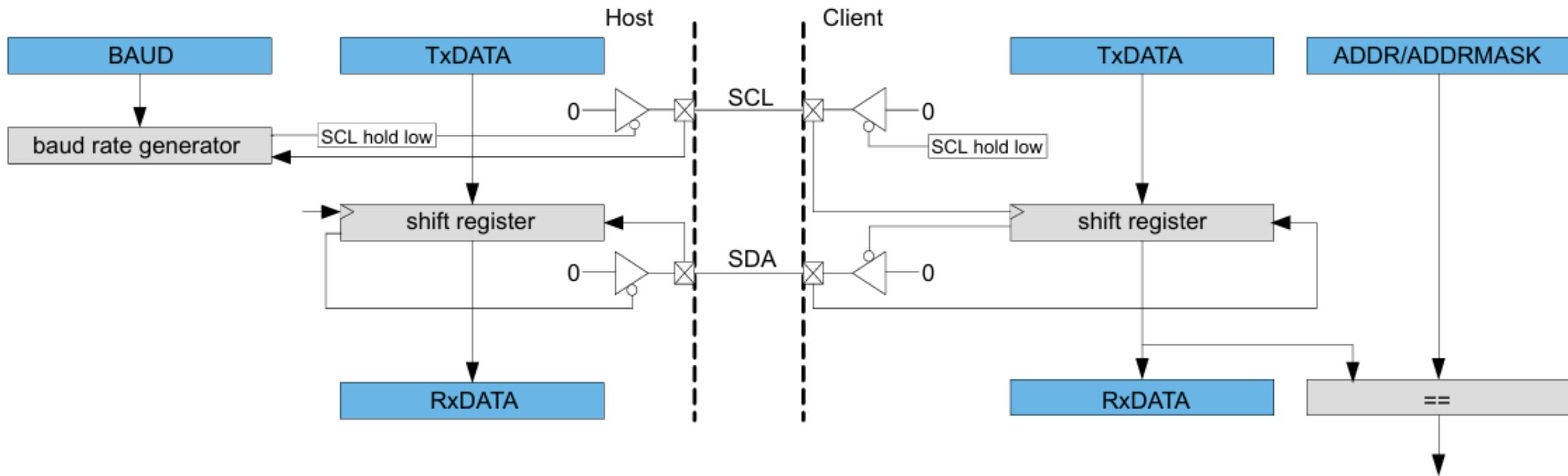
	Acknowledge (ACK)
	Not Acknowledge (NACK)

PIC32CM3204GV SERCOM 在 I2C mode 的主要功能

- Host or Client operation
- Philips I2C compatible
- SMBusTM 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

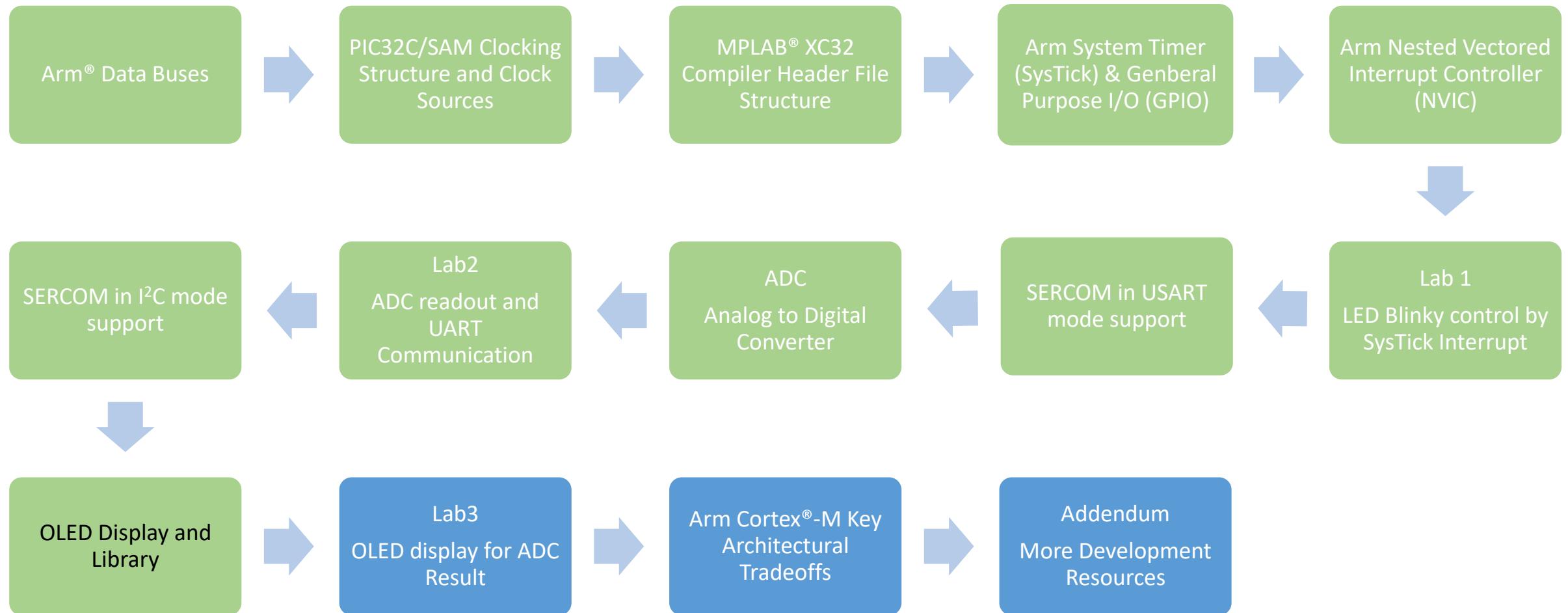
PIC32CM3204GV SERCOM 在 I²C mode 的方塊圖

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





OLED Display

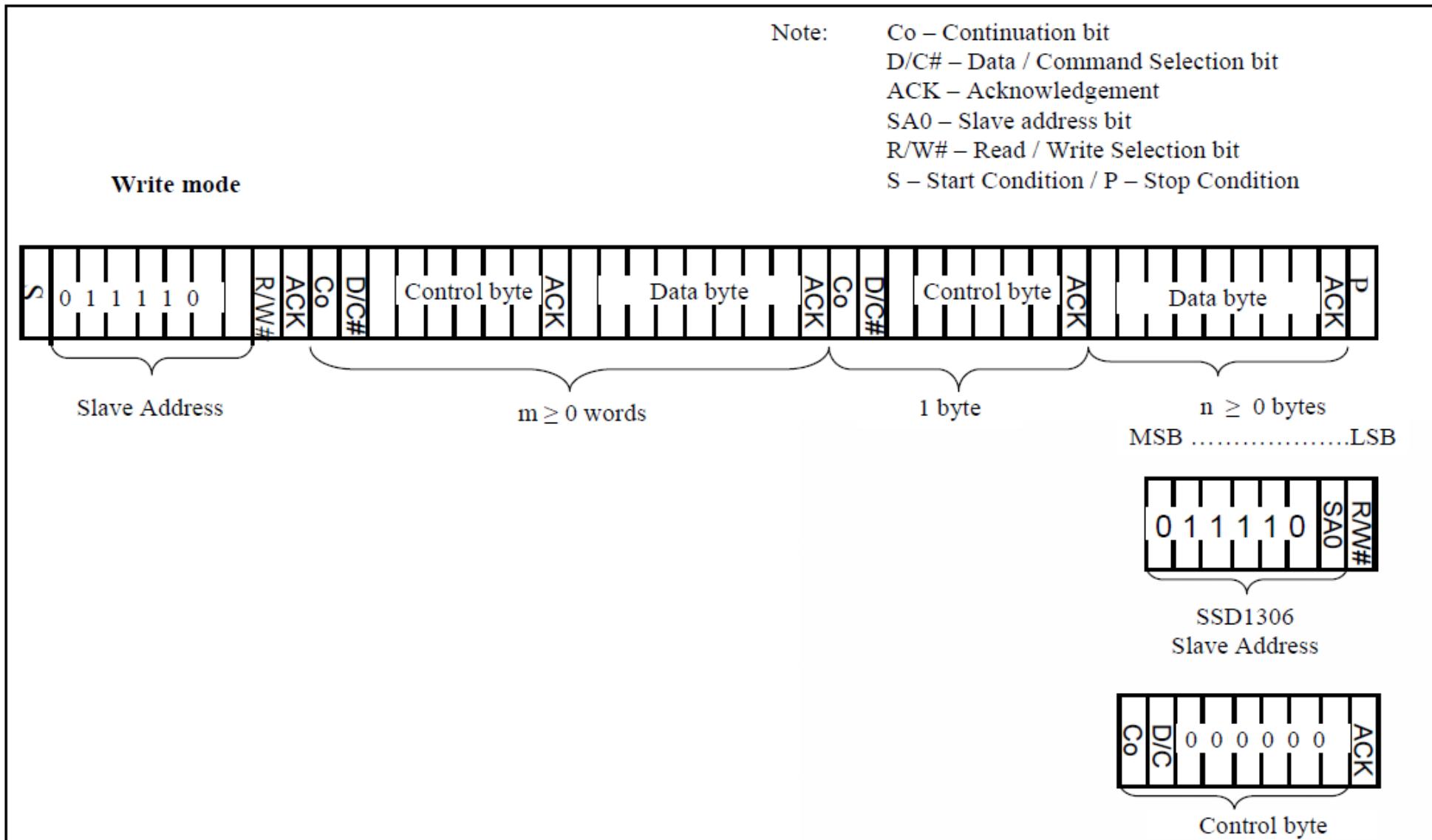
OLED Display簡介

- 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#

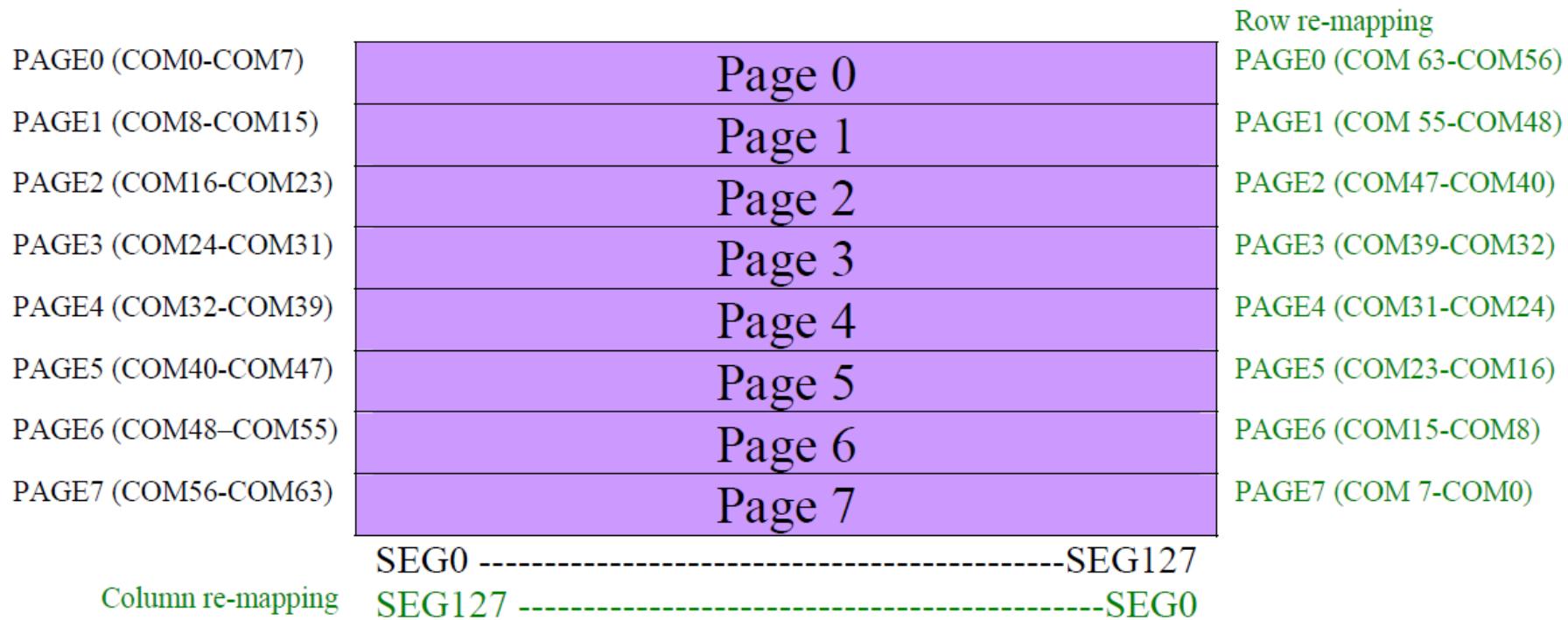
SSD1306 I²C Mode的資料格式



SSD1306內部的Display RAM

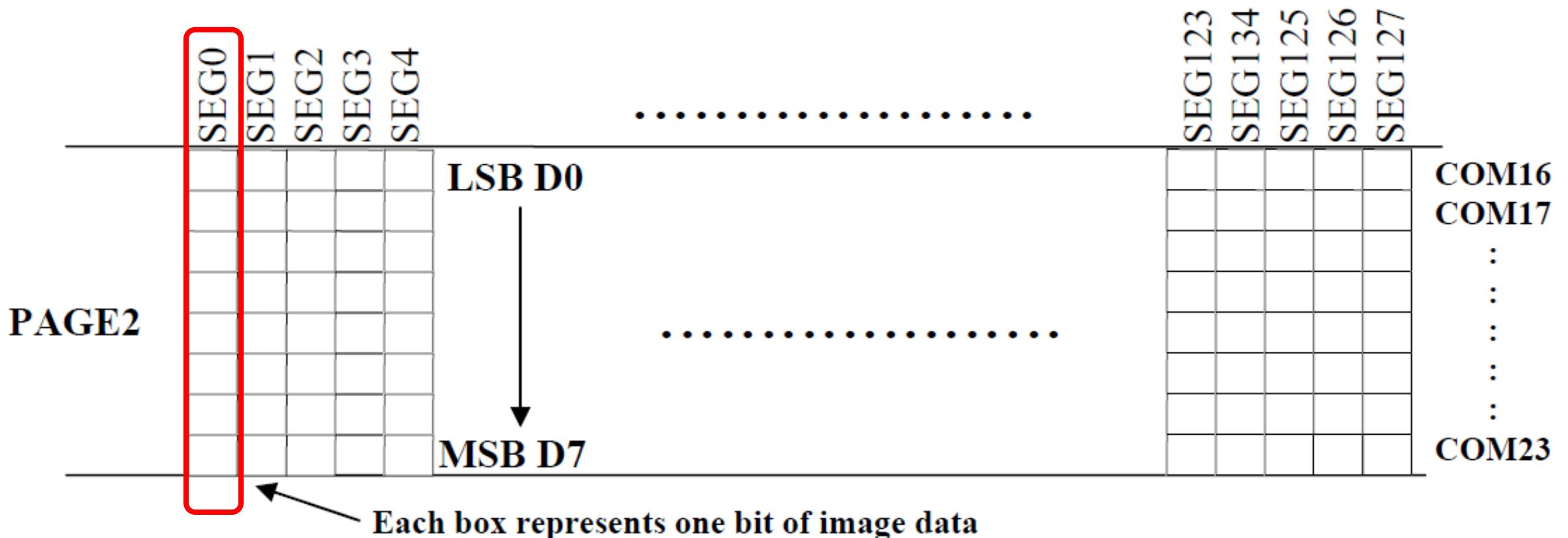
The GDDRAM is a bit mapped static RAM holding the bit pattern to be displayed. The size of the RAM is 128 x 64 bits and the RAM is divided into eight pages, from PAGE0 to PAGE7, which are used for monochrome 128x64 dot matrix display, as shown in Figure 8-13.

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！

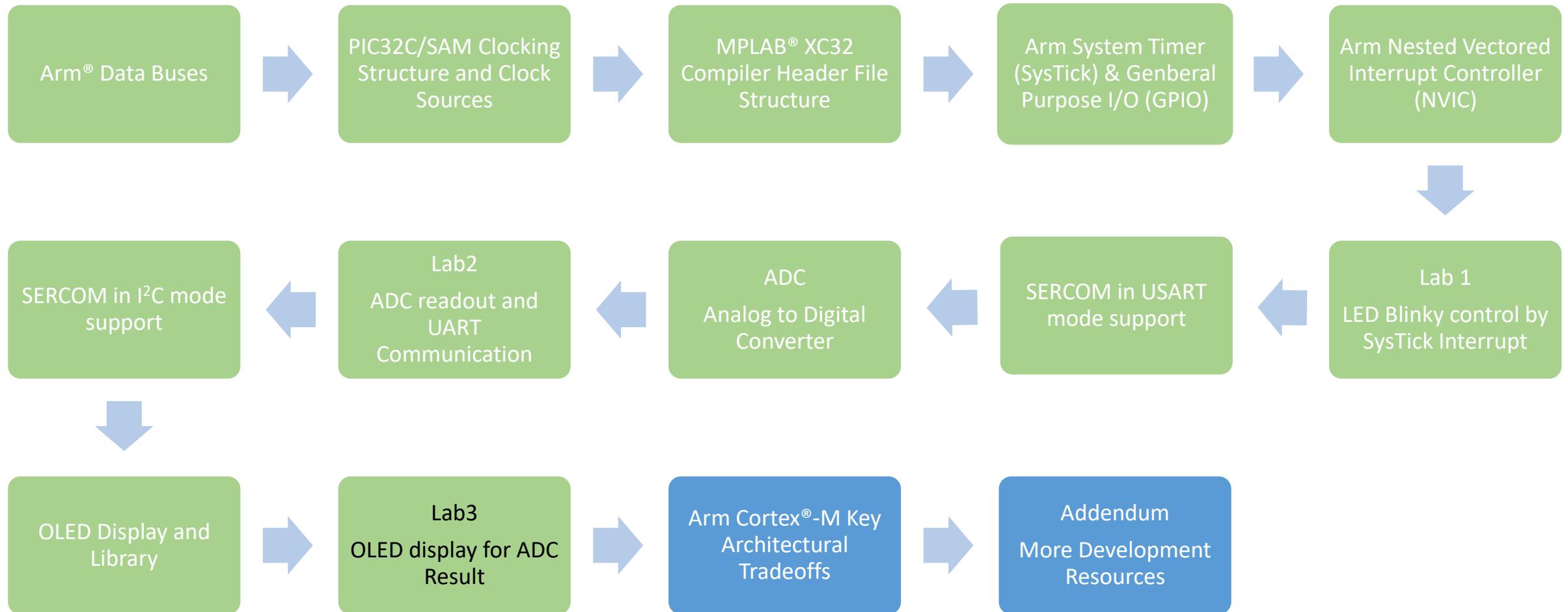


SSD1306 OLED 的操作程式

- 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 I²C
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

 **New Project** X

Steps

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

Select Project Name and Folder

Project Name:

Project Location: Browse...

Project Folder:

Overwrite existing project.

Also delete sources.

Set as main project

Open MCC on Finish

Use project location as the project folder

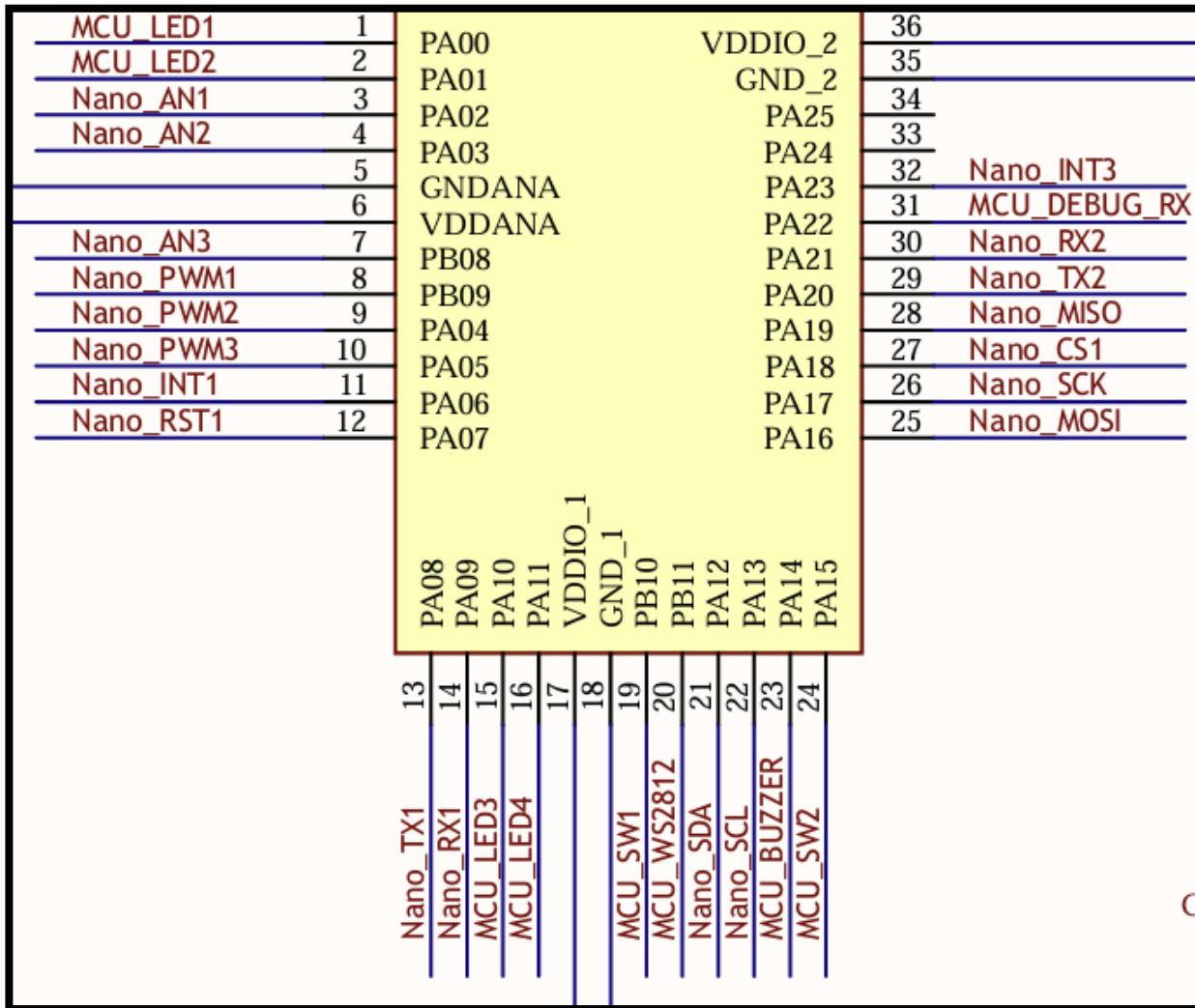
Encoding:

< Back Add Another Project Next > Finish Cancel Help



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)

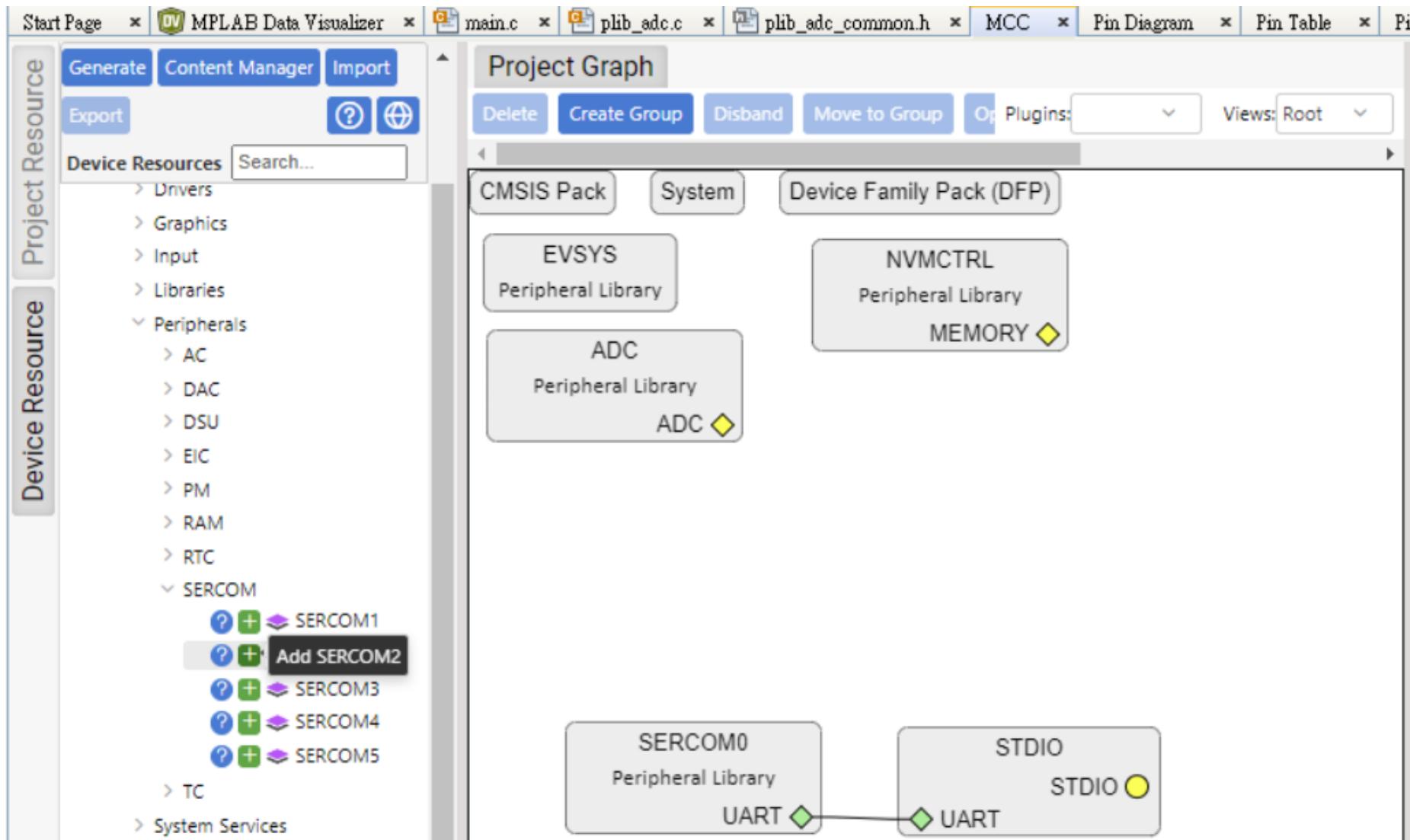
Lab 3 – OLED Display for ADC Result

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

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

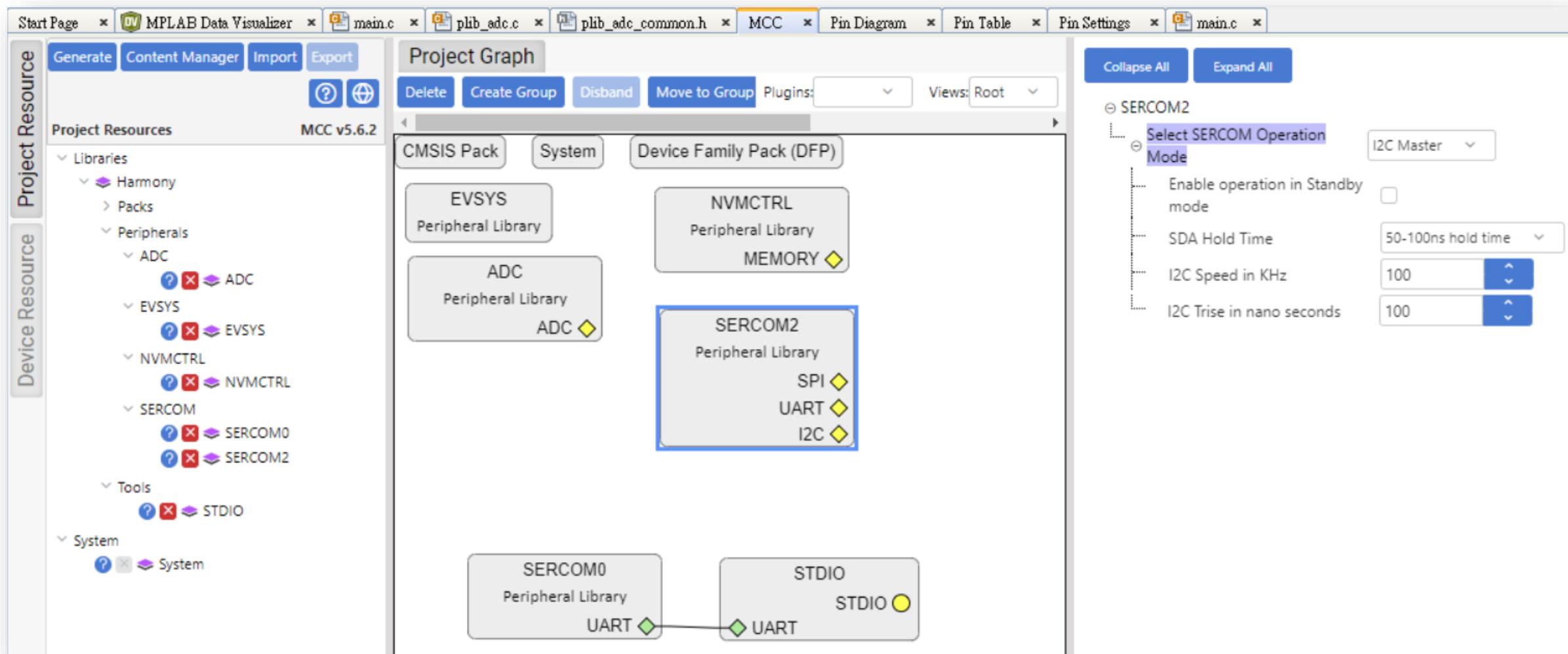
Lab 3 – OLED Display for ADC Result

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

The screenshot shows the MPLAB X IDE interface. The title bar reads "MPLAB X IDE v6.25 - PIC32CMGV_Lab3 : default". The menu bar includes File, Edit, View, Navigate, Source, Refactor, Production, Debug, Team, Tools, Window, Help. The toolbar has various icons for file operations like Open, Save, Build, and Run. The project tree on the left shows a project named "PIC32CMGV_Lab3" with subfolders Header Files, Important Files, LibraryFiles, Linker Files, Source Files (containing config, main.c, and OLED128x64.c), Libraries, and Loadables. The main code editor window displays C code. The code starts with a function definition for "SYSTICK_EventHandler" which toggles an LED and sets a flag. It then defines the "main" function which initializes modules and registers a callback. The status bar at the bottom shows the current file is "OLED_Put16x16Ch" and the line is "OLED_Init0;".

```
43
44     void SYSTICK_EventHandler(uintptr_t context)
45     {
46         /* Toggle LED */
47         LED1_Toggle();
48         bFlag_200ms = 1;
49     }
50
51     int main ( void )
52     {
53         /* Initialize all modules */
54         SYS_Initialize ( NULL );
55         /* Register Callback */
```

Lab 3 – OLED Display for ADC Result

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

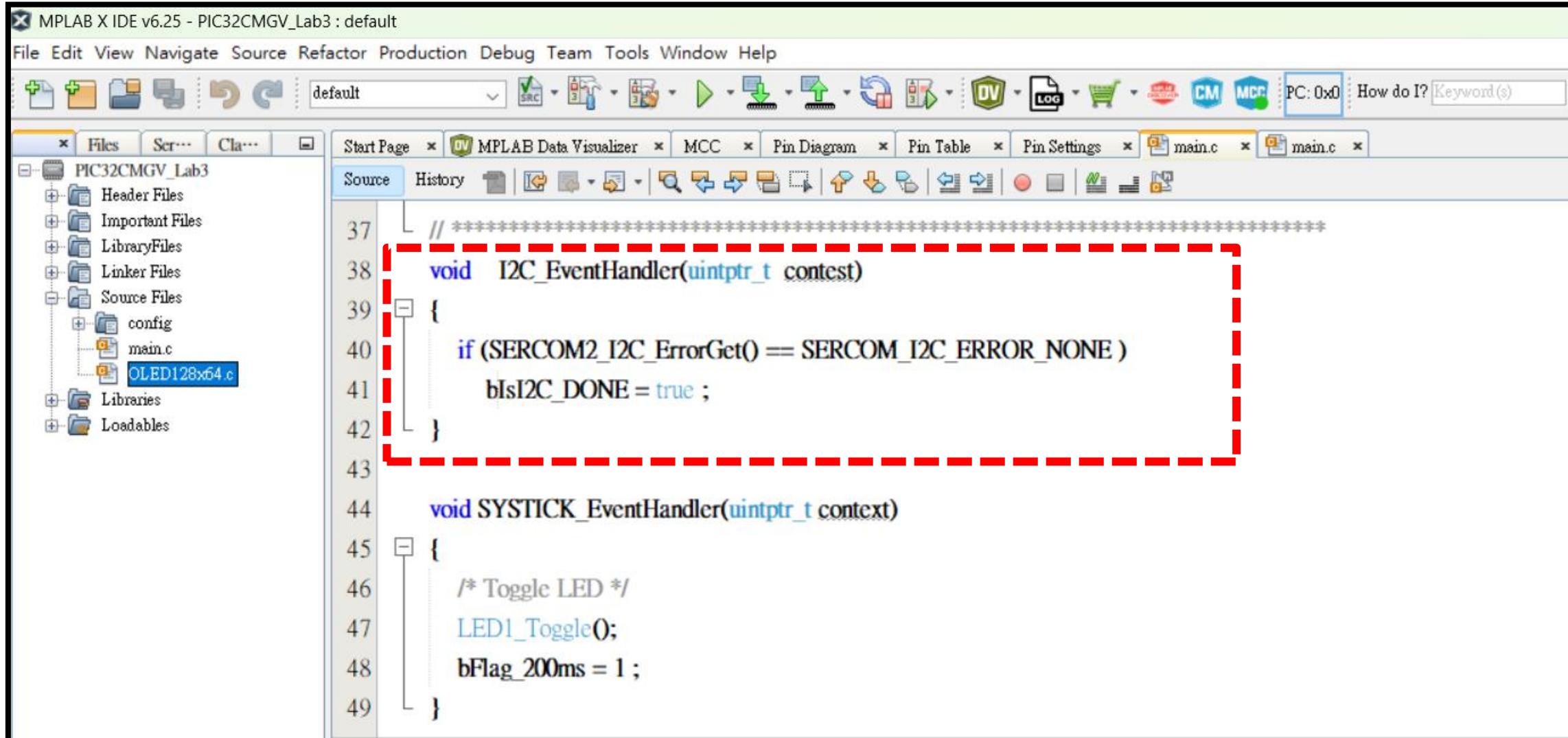
The screenshot shows the MPLAB X IDE interface with the project 'PIC32CMGV_Lab3' open. The 'main.c' file is selected in the tab bar. The code editor displays the following C code:

```
22 // ****
23 // ****
24 #include <stddef.h>           // Defines NULL
25 #include <stdbool.h>          // Defines true
26 #include <stdlib.h>            // Defines EXIT_FAILURE
27 #include "definitions.h"       // SYS function prototypes
28 #include "OLEDI28x64.h"         // Line 28 is highlighted with a red dashed box
29
30 volatile bool    bFlag_200ms = 0;
31 volatile bool    bIsI2C_DONE = true;
32 uint8_t          ASCII_Buffer[24];
33 // ****
34 // ****
```

A red dashed rectangular box highlights the line '#include "OLEDI28x64.h"' and the declarations for 'bFlag_200ms', 'bIsI2C_DONE', and 'ASCII_Buffer'. The file tree on the left shows the project structure, including 'Source Files' containing 'config', 'main.c', and 'OLEDI28x64.c'.

Lab 3 – OLED Display for ADC Result

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



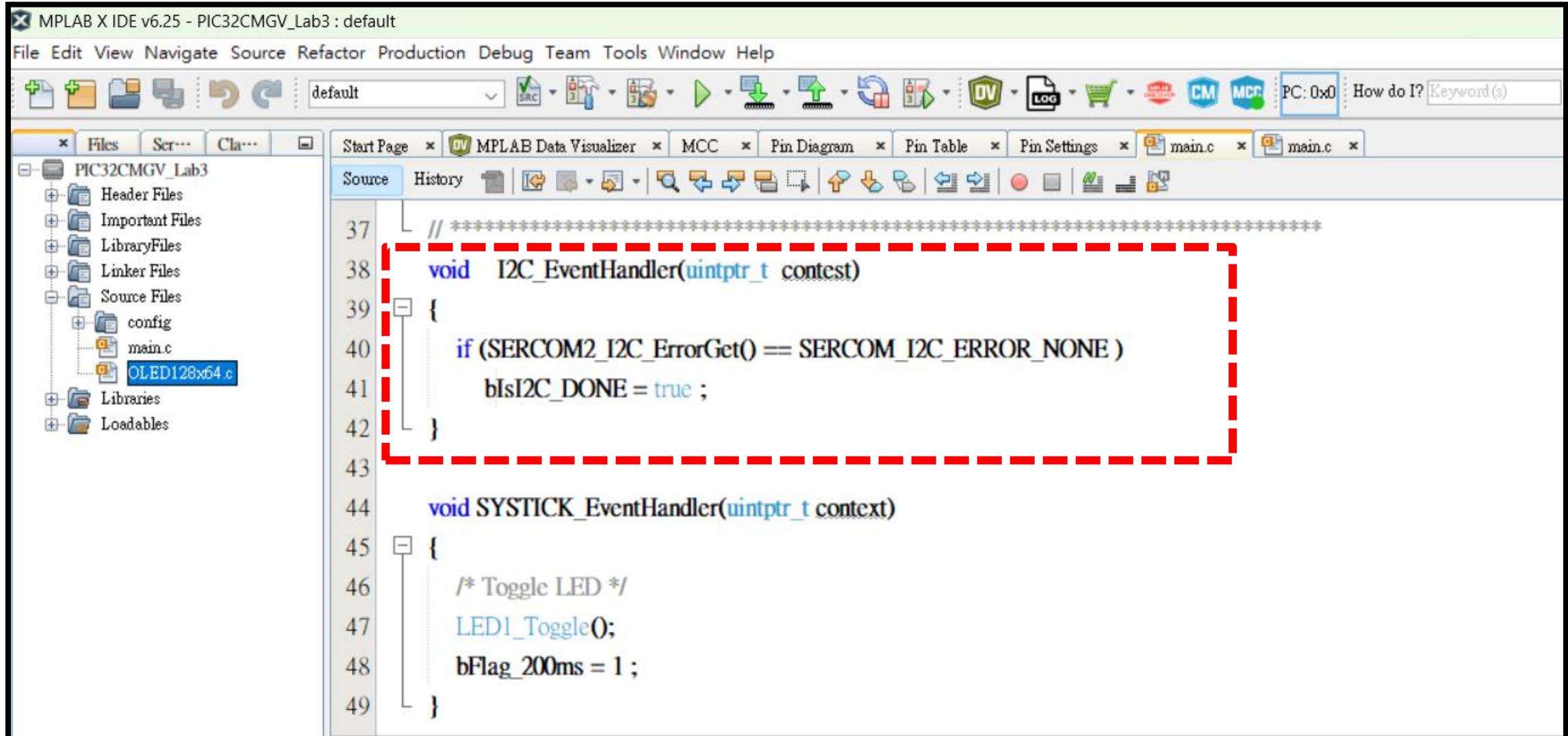
The screenshot shows the MPLAB X IDE interface with the project "PIC32CMGV_Lab3" open. The main window displays the "main.c" source code. Two sections of code are highlighted with red dashed boxes:

```
// *****
void I2C_EventHandler(uintptr_t context)
{
    if (SERCOM2_I2C_ErrorGet() == SERCOM_I2C_ERROR_NONE )
        bIsI2C_DONE = true ;
}

void SYSTICK_EventHandler(uintptr_t context)
{
    /* Toggle LED */
    LED1_Toggle();
    bFlag_200ms = 1 ;
}
```

Lab 3 – OLED Display for ADC Result

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



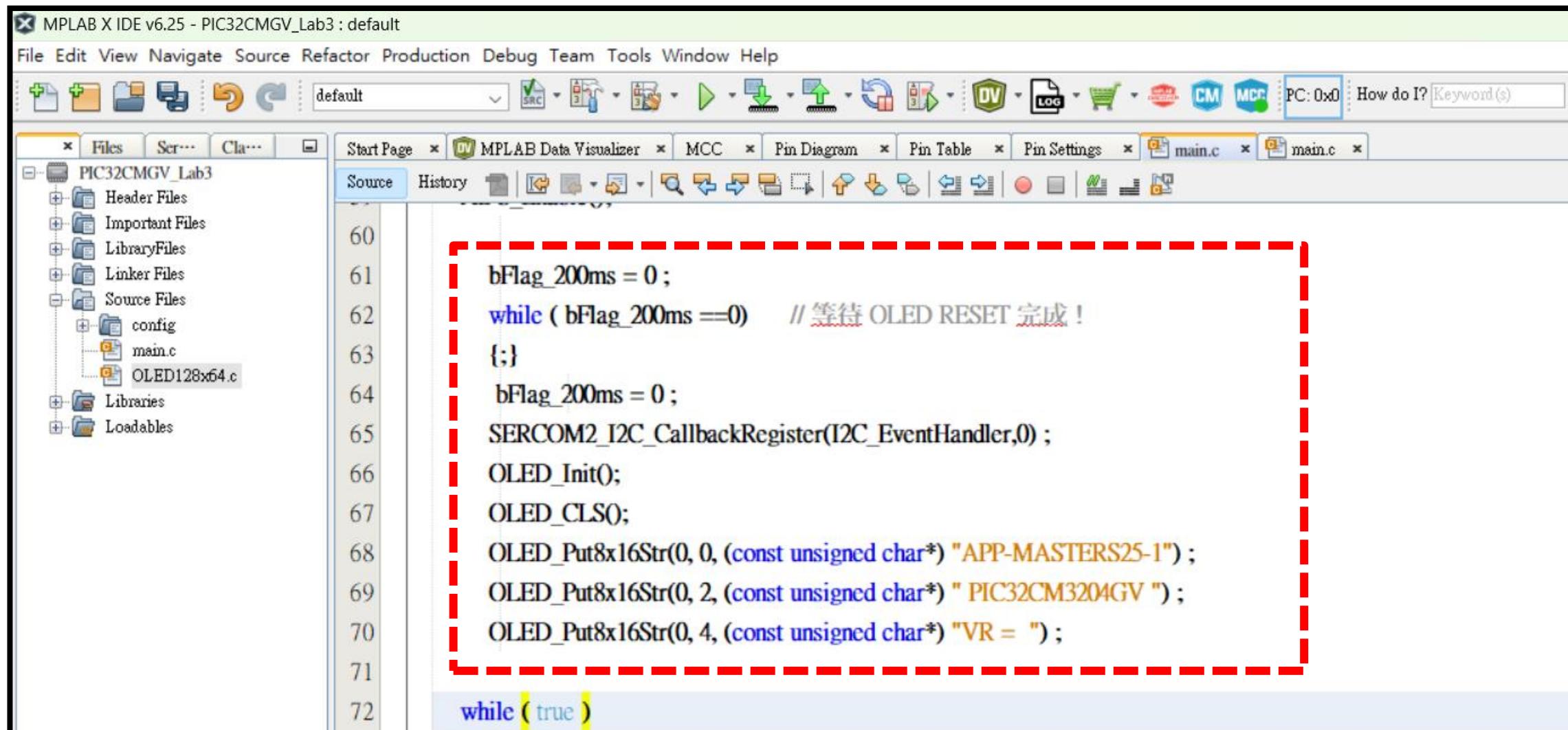
The screenshot shows the MPLAB X IDE interface with the project "PIC32CMGV_Lab3" open. The main window displays the "main.c" source code. Two sections of code are highlighted with red dashed boxes:

```
// *****
void I2C_EventHandler(uintptr_t context)
{
    if (SERCOM2_I2C_ErrorGet() == SERCOM_I2C_ERROR_NONE )
        bIsI2C_DONE = true ;
}

void SYSTICK_EventHandler(uintptr_t context)
{
    /* Toggle LED */
    LED1_Toggle();
    bFlag_200ms = 1 ;
}
```

Lab 3 – OLED Display for ADC Result

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



The screenshot shows the MPLAB X IDE interface with the title bar "MPLAB X IDE v6.25 - PIC32CMGV_Lab3 : default". The menu bar includes File, Edit, View, Navigate, Source, Refactor, Production, Debug, Team, Tools, Window, and Help. The toolbar has various icons for file operations like Open, Save, and Build. The project tree on the left shows the folder structure for "PIC32CMGV_Lab3", including Header Files, Important Files, LibraryFiles, Linker Files, Source Files (containing config, main.c, and OLED128x64.c), Libraries, and Loadables. The main code editor window displays the "main.c" file. A red dashed rectangle highlights the following code segment:

```
60
61     bFlag_200ms = 0 ;
62     while ( bFlag_200ms ==0) // 等待 OLED RESET 完成 !
63     {};
64     bFlag_200ms = 0 ;
65     SERCOM2_I2C_CallbackRegister(I2C_EventHandler,0) ;
66     OLED_Init();
67     OLED_CLSO;
68     OLED_Put8x16Str(0, 0, (const unsigned char*) "APP-MASTERS25-1") ;
69     OLED_Put8x16Str(0, 2, (const unsigned char*) " PIC32CM3204GV ") ;
70     OLED_Put8x16Str(0, 4, (const unsigned char*) "VR = ") ;
71
72     while ( true )
```

Lab 3 – OLED Display for ADC Result

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

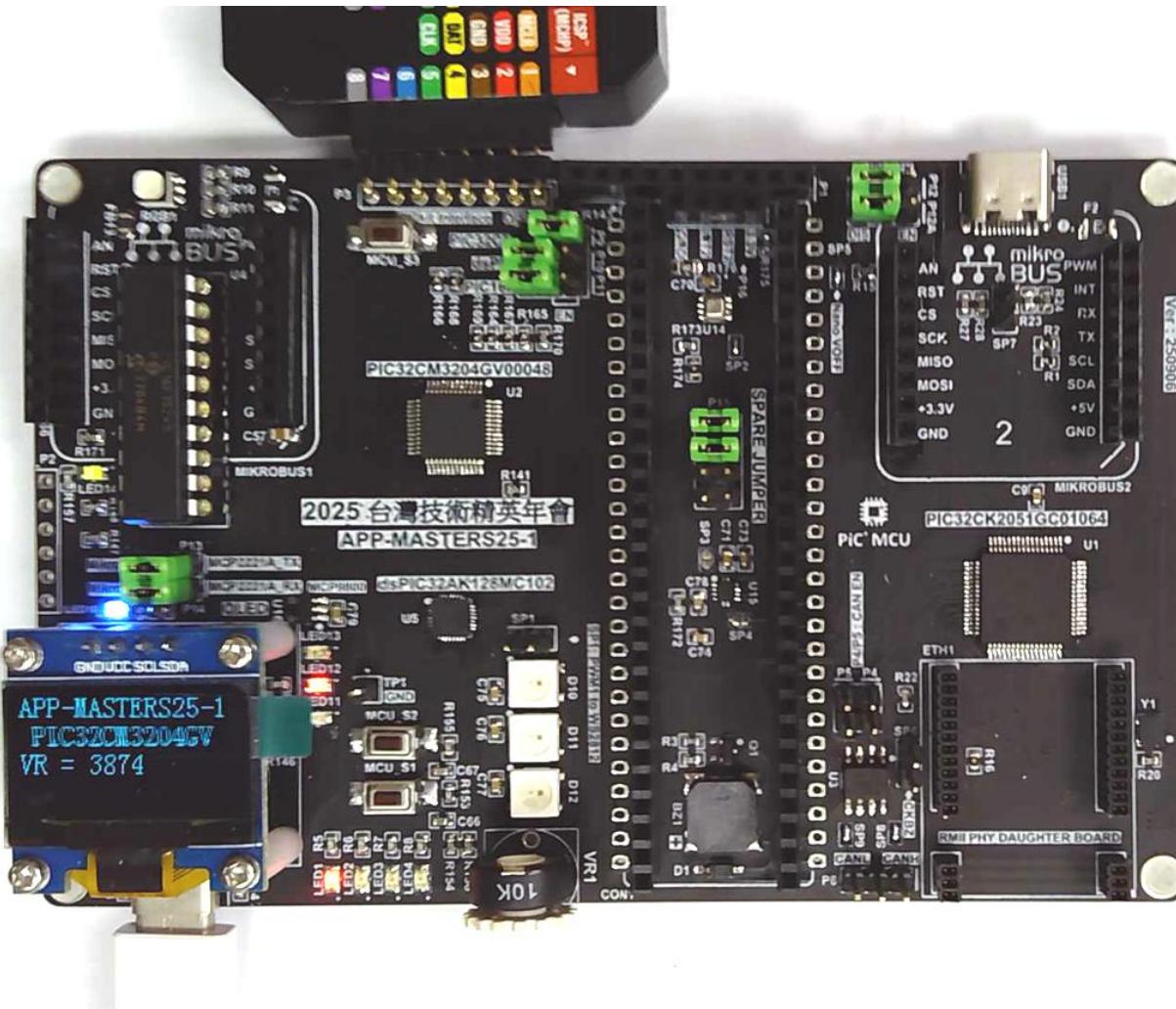
The screenshot shows the MPLAB X IDE interface with the title bar "MPLAB X IDE v6.25 - PIC32CMGV_Lab3 : default". The menu bar includes File, Edit, View, Navigate, Source, Refactor, Production, Debug, Team, Tools, Window, and Help. The toolbar has various icons for file operations like Open, Save, Build, and Run. The project tree on the left shows "PIC32CMGV_Lab3" with subfolders Header Files, Important Files, LibraryFiles, Linker Files, Source Files, Libraries, and Loadables. The main editor window displays the "main.c" file with the following code:

```
75 SYS_Tasks ();
76 if (bFlag_200ms)
77 {
78     // ADC0_ChannelSelect(ADC_POSINPUT_AIN0, ADC_NEGINPUT_GND);
79     ADC_ChannelSelect(ADC_POSINPUT_PIN1, ADC_NEGINPUT_GND);
80     ADC_ConversionStart();
81     while (!ADC_ConversionStatusGet());
82     printf ("ADC = %4d\n", ADC_ConversionResultGet());
83     sprintf((char*)ASCII_Buffer, "%4d", ADC_ConversionResultGet());
84     OLED_Put8x16Str(40, 4, ASCII_Buffer);
85     bFlag_200ms = 0;
86 }
87 }
```

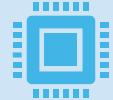
A red dashed box highlights the section of code from line 82 to line 84, which prints the ADC result to a buffer and then displays it on the OLED screen.

Lab 3 – OLED Display for ADC Result

Lab3 實際執行結果



Lab 3 Summary



Properly set up SERCOM to implement I²C MASTER Function



Add Library into your project

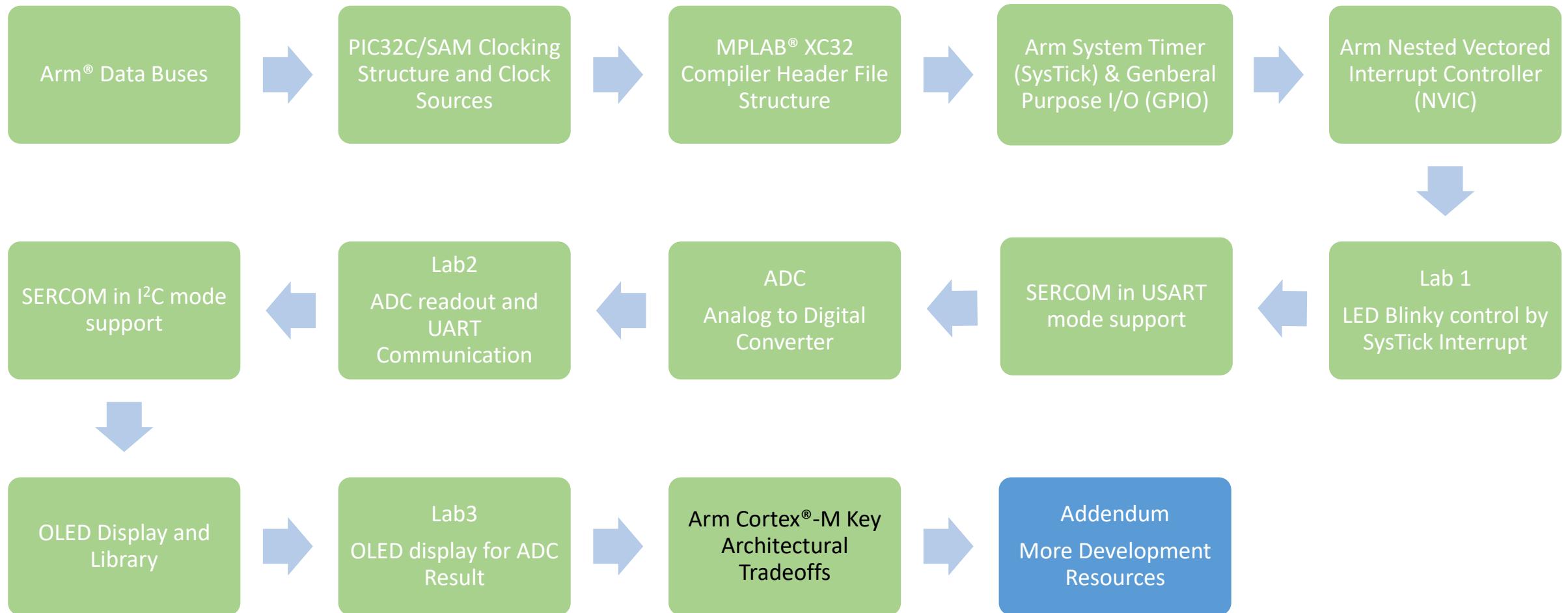


Config OLED Display by using Library



Display ADC read result to Line-3 of OLED

Class Outline





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

Instruction Set Architecture (ISA) Relationship

Arm® Cortex®-M0+

Arm Cortex-M23

Arm Cortex-M3/M4/M7

Arm Cortex-M33

ARMv7E-M

ARMv6-M

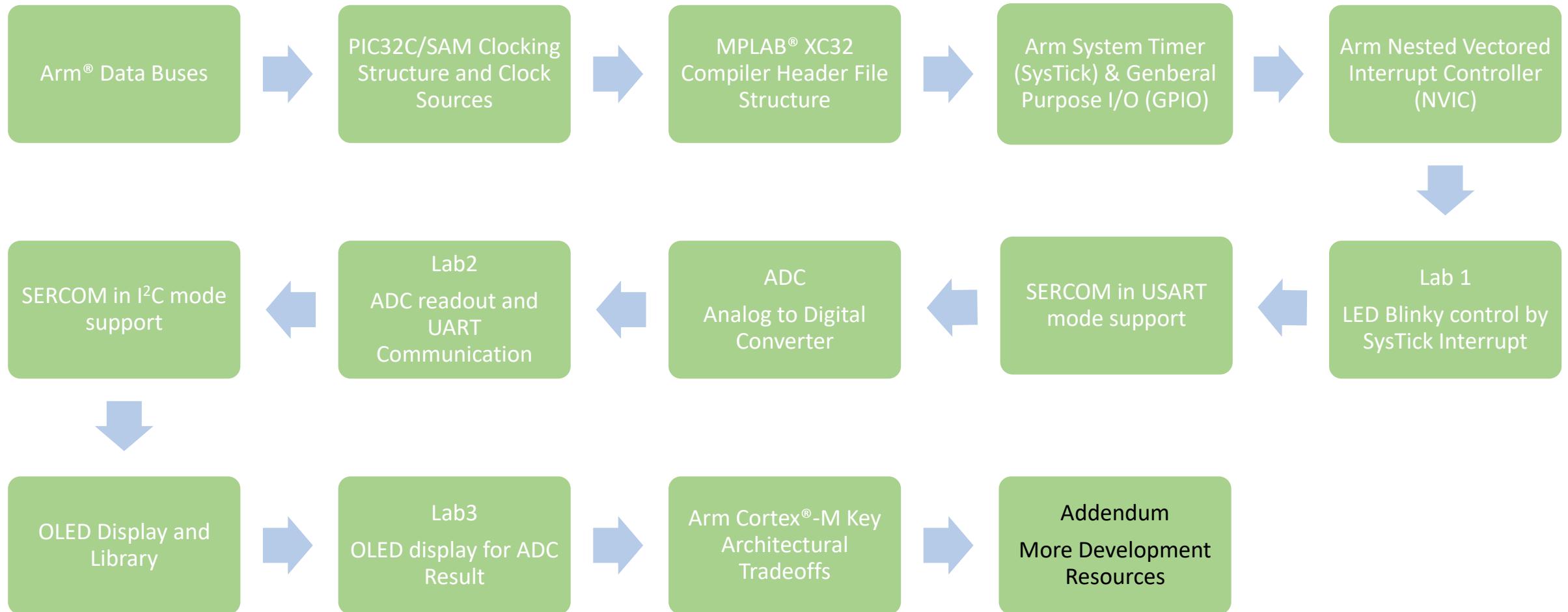
ARMv8-M
Main

ARMv8-M
Baseline

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





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

The screenshot shows the Microchip MPLAB® DISCOVER platform interface. The top navigation bar includes the Microchip logo, the MPLAB DISCOVER logo, a search bar containing 'pic32cm', and various icons for help and download. The left sidebar contains a navigation menu with categories like Code Examples (447), Documentation (461), Learning Resources (18), and Evaluation Boards (19). The main content area displays a search result for 'pic32cm' with 945 items found, showing 1-20 of them. The results are listed in a table with columns for Name and Category, all categorized under 'MCU/MPU Examples'. The results include examples for PIC32CM LE and GC families.

Name	Category
USB Host CDC Basic Example (cdc_basic) Example for PIC32CM LE Family	MCU/MPU Examples
USB Host CDC Basic Example (cdc_basic) Example for PIC32CM GC Family	MCU/MPU Examples
USB Host CDC MSD Example (cdc_msd) Example for PIC32CM GC Family	MCU/MPU Examples
USB Host HID Basic Keyboard Example (hid_basic_keyboard) Example for PIC32CM LE Family	MCU/MPU Examples
USB Host HID Basic Keyboard Example (hid_basic_keyboard) Example for PIC32CM GC Family	MCU/MPU Examples
USB Host HID Basic Mouse USART Example (hid_basic_mouse_usart) Example for PIC32CM GC Family	MCU/MPU Examples
USB Host MSD Basic Example (msd_basic) Example for PIC32CM LE Family	MCU/MPU Examples
USB Host MSD Basic Example (msd_basic) Example for PIC32CM GC Family	MCU/MPU Examples
USB Host MSD Basic Example (msd_basic) Example PIC32CM GC Family	MCU/MPU Examples
USB Host Vendor Example (vendor) Example for PIC32CM GC Family	MCU/MPU Examples

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 POSSIBILITY 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:

© 2025 Microchip Technology Incorporated, All Rights Reserved.