

**SAMD21 ARM Cortex-M0+
Microcontroller & Harmony v3
SAM2001_{ADV} v1.01**



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



SMART | CONNECTED | SECURE

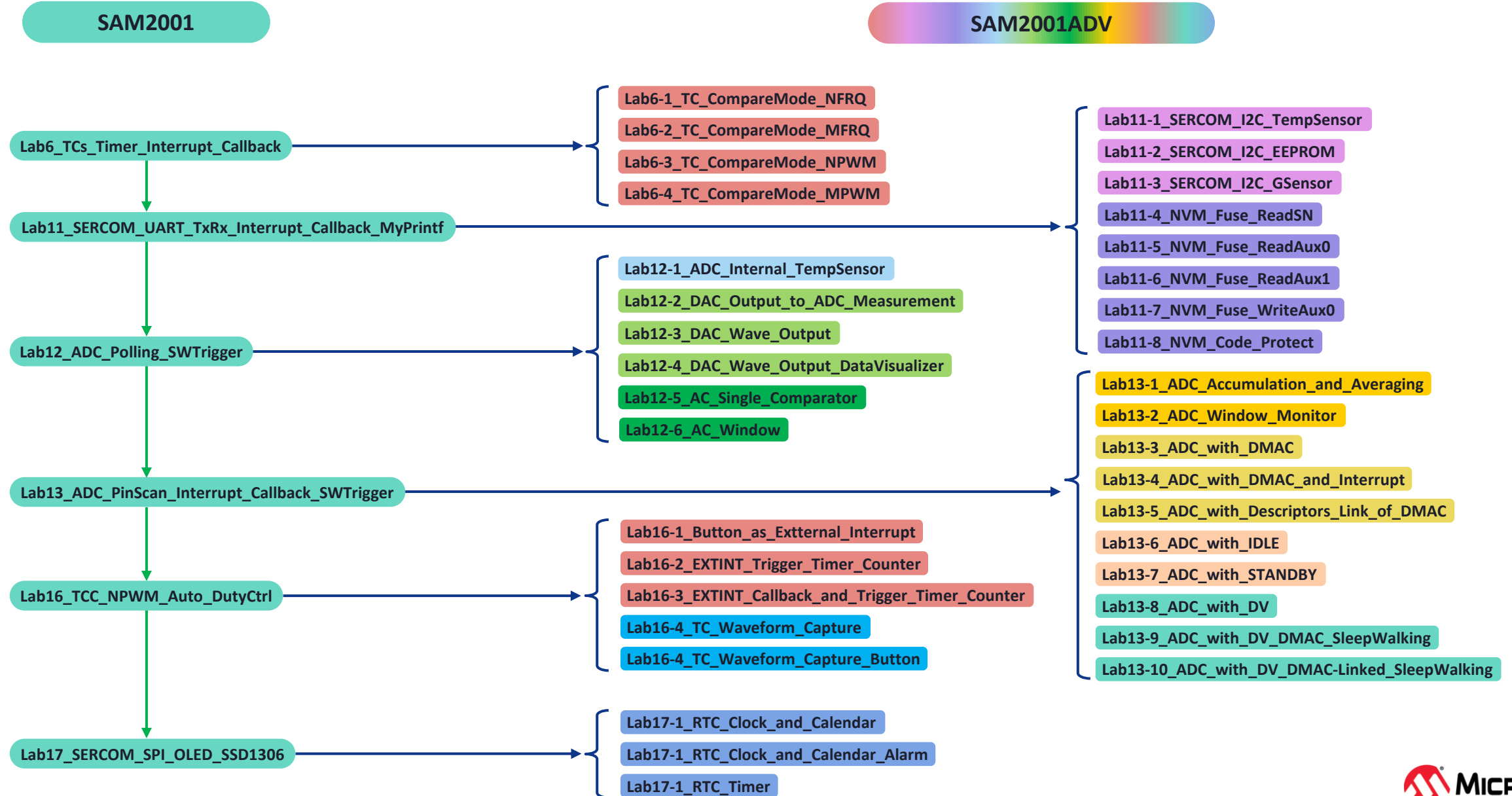
**CAE Taiwan Team
Microchip Technology Inc.**

2022 December

Class Objectives

- **When you finish this course, you will be able to:**
 - Be familiar with MPLAB X IDE and Harmony 3
 - Understand how to configure and use the SAMD21 series peripherals.
- **There have two level course included.**
 - Major course : General study of basic features.
 - Appendix : Advance features and appendix.

Lab Tree Branch from SAM2001 to SAM2001ADV



Advance Course (1/4)

- [SAMD21 EVB APP045 v4.20 Introduction](#)
- [Timer/Counter Operation Mode](#)
 -  [Lab6-1 TC Operation Mode - NFRQ](#)
 -  [Lab6-2 TC Operation Mode - MFRQ](#)
 -  [Lab6-3 TC Operation Mode - NPWM](#)
 -  [Lab6-4 TC Operation Mode - MPWM](#)
- [SERCOM – I2C Architecture](#)
 -  [Lab11-1 SERCOM – I2C Access Temperature Sensor](#)
 -  [Lab11-2 SERCOM – I2C Access Serial EEPROM](#)
 -  [Lab11-3 SERCOM – I2C Access G-Sensor](#)
- [ADC Accumulation and Averaging](#)
 -  [Lab13-1 ADC Accumulation and Averaging](#)
- [ADC Window Monitor](#)
 -  [Lab13-2 ADC Window Monitor](#)

Advance Course (2/4)

- Internal Temperature Sensor

-  Lab12-1 Internal Temperature Sensor through ADC

- RTC (Real Time Controller)

-  Lab17-1 RTC Clock and Calendar

- DAC Architecture

-  Lab12-2 DAC Output to ADC Measurement

-  Lab12-3 DAC Wave Output

- MPLAB® Data Visualizer

-  Lab12-4 DAC Wave Output DataVisualizer

- Analog Comparators (AC) Architecture

-  Lab12-5 AC Single Comparator

-  Lab12-6 AC Window

- External Interrupt Controller (EIC)

-  Lab16-1 Button as External Interrupt

Advance Course (3/4)

- Event System (EVSYS)

-  Lab16-2 EXTINT Trigger Timer Counter

-  Lab16-3 EXTINT Callback and Trigger Timer Counter

- TC Waveform Capture (Integrate TC, EIC and Event System)

-  Lab16-4 TC Waveform Capture

- Direct Memory Access Controller (DMAC) - Single Transfer Block

-  Lab13-3 ADC with DMAC

-  Lab13-4 ADC with DMAC and Interrupt

- Direct Memory Access Controller (DMAC) - Linked List Transfer

-  Lab13-5 ADC with Descriptors Link of DMAC

- Power Debugger

- Sleep Mode

-  Lab13-6 ADC with IDLE

-  Lab13-7 ADC with STANDBY

Advance Course (4/4)

- Application : Sleepwalking

-  Lab13-8_ADC_with_DV

-  Lab13-9_ADC_with_DV_DMALinked_SleepWalking

-  Lab13-10_ADC_with_DV_DMALinked_SleepWalking

- Auxiliary Space of NVM and Fuse Maintains

-  Lab11-4 Read Unique Serial Number

-  Lab11-5 Read NVM AUX0 (User Row)

-  Lab11-6 Read NVM AUX1 (Software Calibration)

-  Lab11-7 Write NVM AUX0 (User Row)

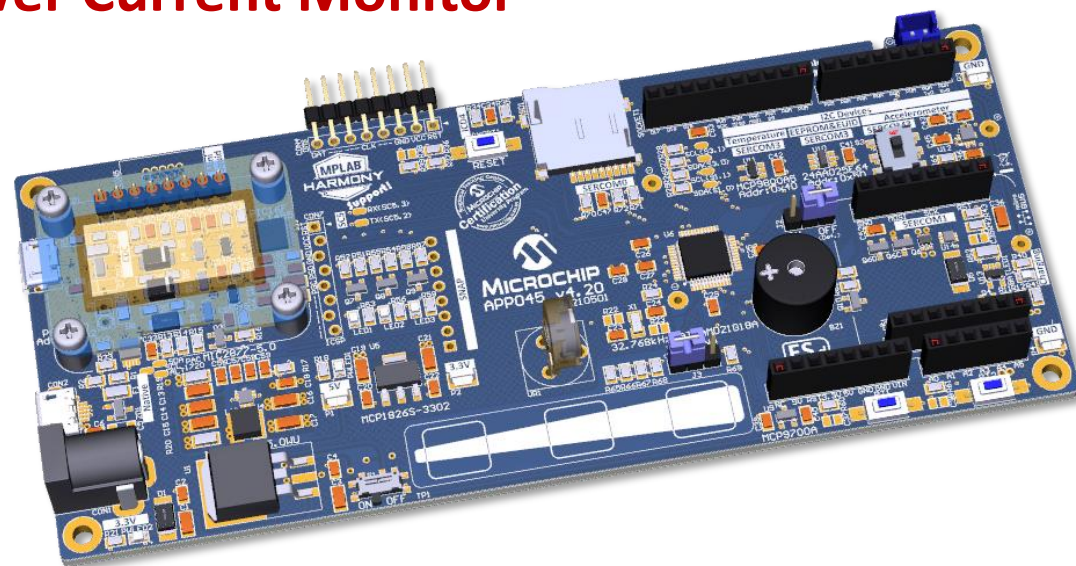
-  Lab11-8 Code protect by NVM Security Bit

SAMD21 EVB APP045 v4

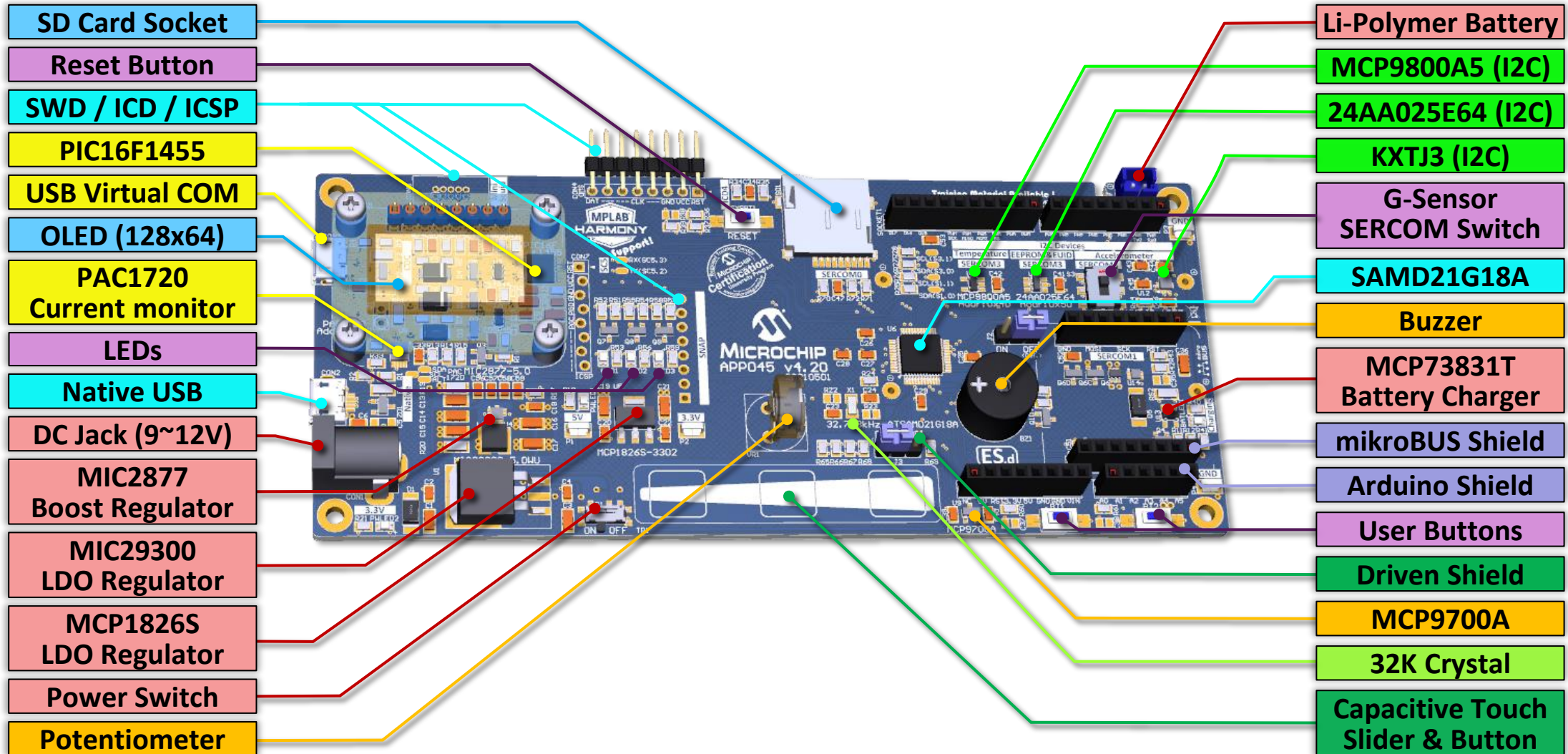
Introduction

APP045 v4 Features

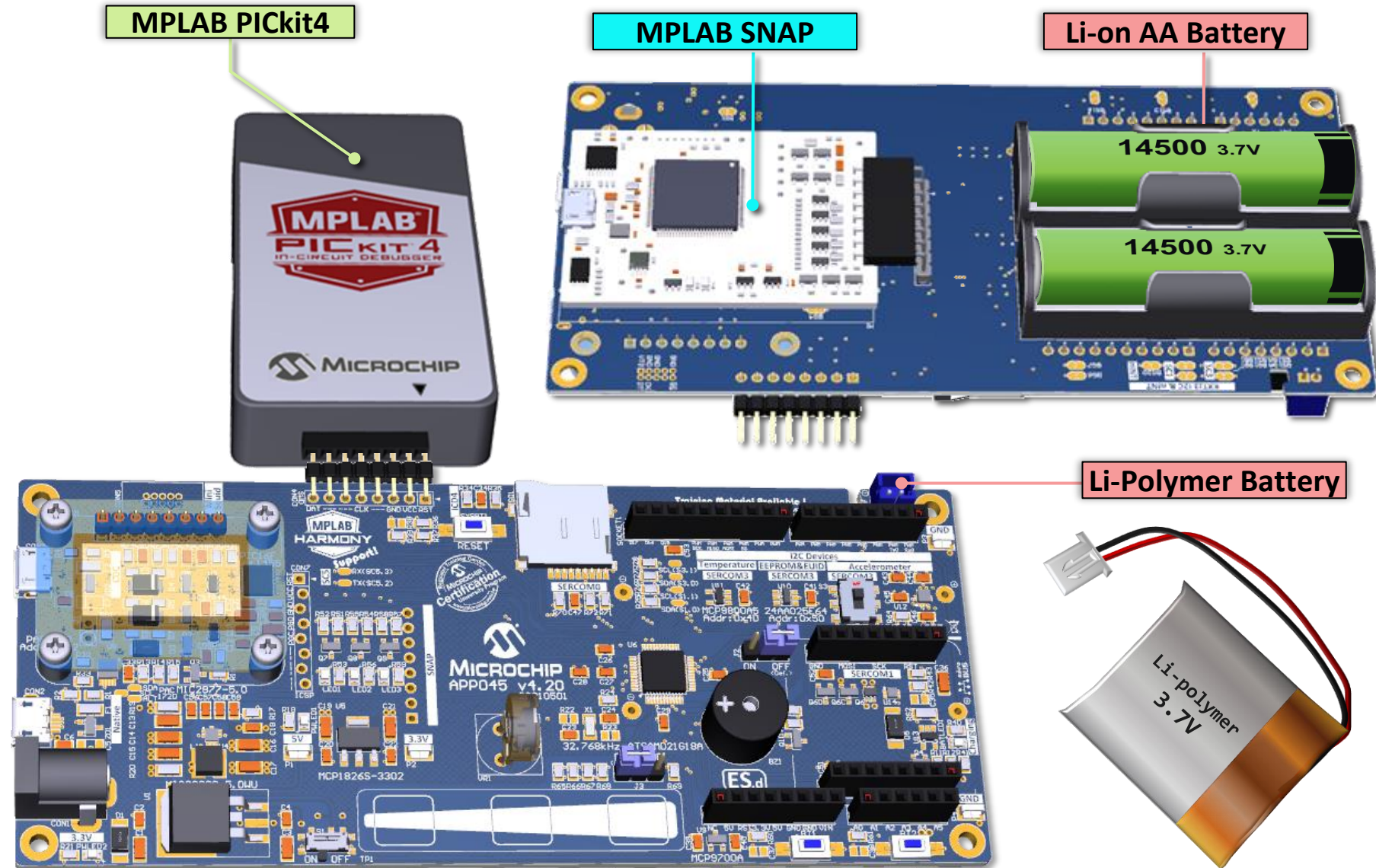
- On board **SAMD21G18A** 32Bit ARM Cortex M0+ Microcontroller
- Support plugged **MPLAB SNAP** external programmer/debugger
- Onboard **USB CDC Virtual COM Port** and **Power Current Monitor**
- Multiple Power Supply Source
- **Support Li-on Battery, boost and charger**
- 2 x Buttons, 3 x LEDs
- 1 x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



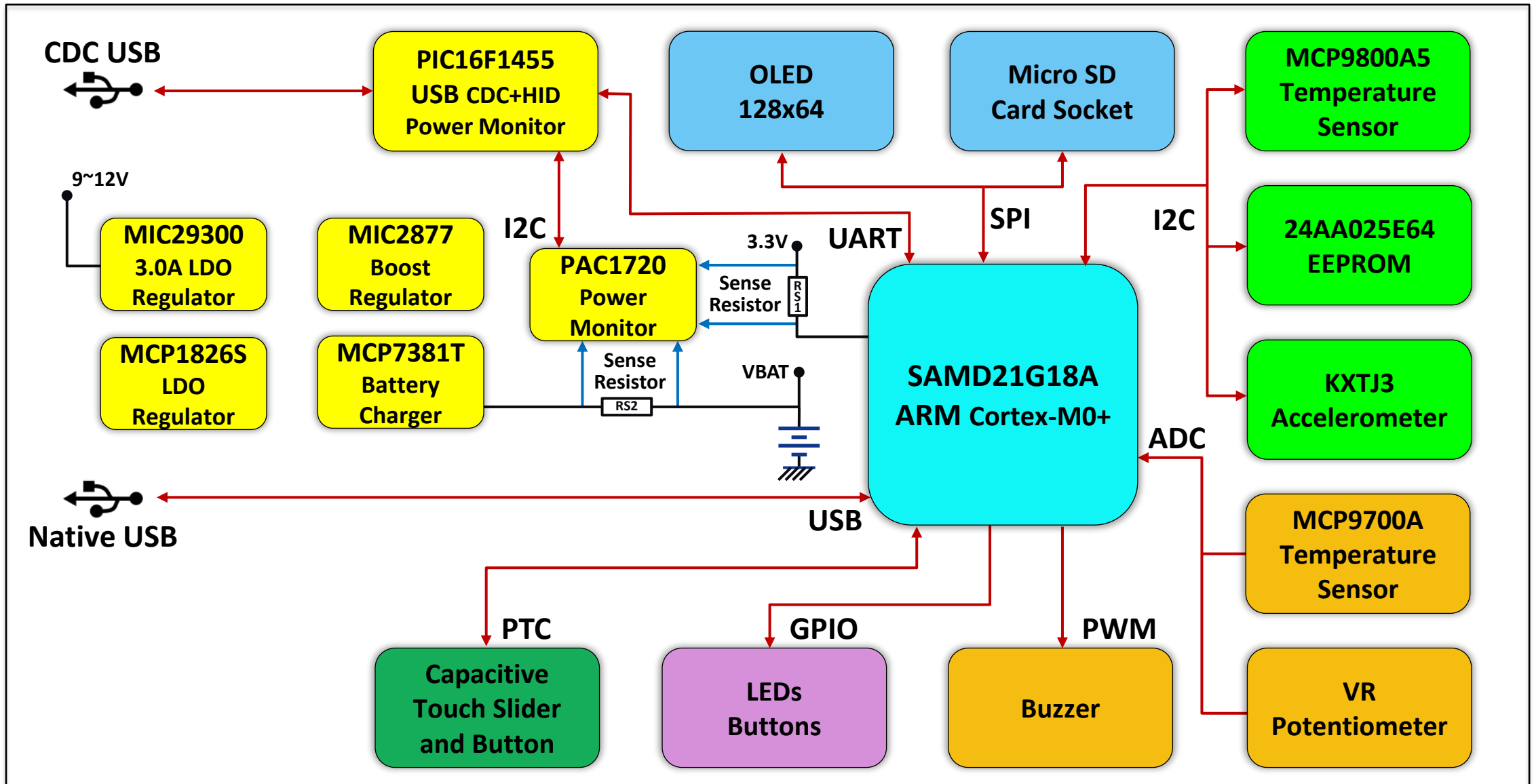
APP045 v4 Overview



APP045 v4 Programmer and Debugger



APP045 v4 Block Diagram



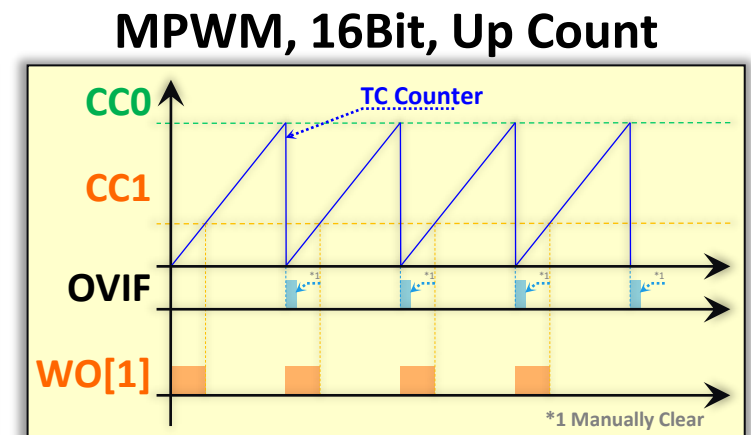
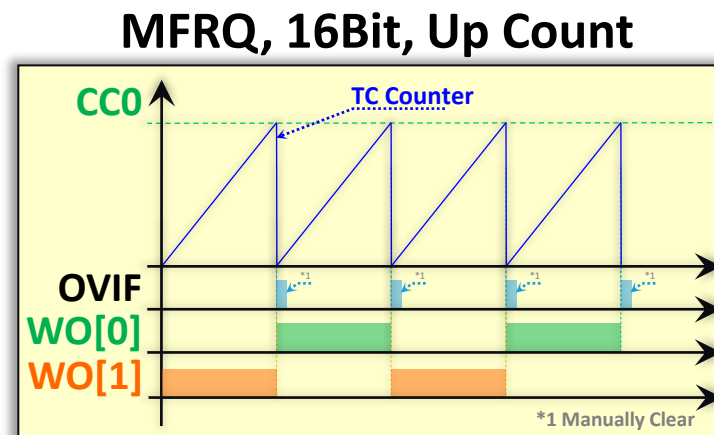
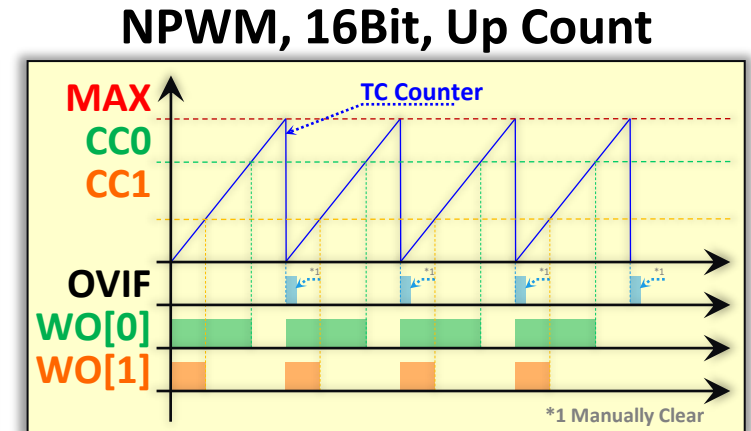
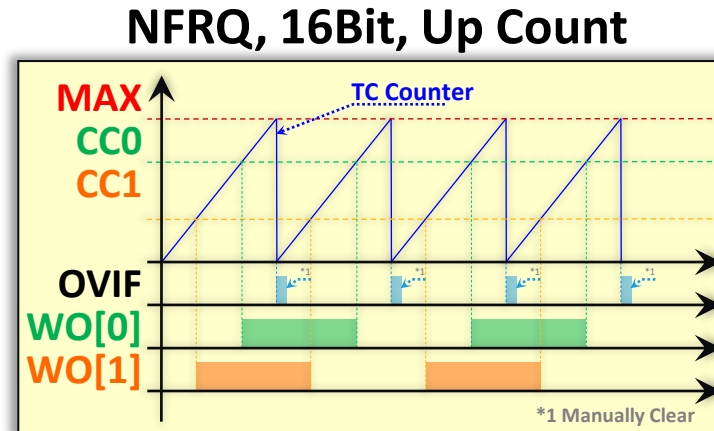
Timer/Counter Operation Mode

Review the Timer/Counter

- **The TC module consists 3 function Counter, Compare and Capture. The counter can be set to count events, or it can be configured to count clock pulses.**
- **TC support three counter resolutions, 32-bits, 16-bits and 8-bits**
- **The counter can be set to up or down count.**
- **Timer/Counter (TC) provide 4 waveform generation operation.**
 - Normal Frequency Generation (NFRQ)
 - Match Frequency Generation (MFRQ)
 - Normal Pulse-Width Modulation Operation (NPWM)
 - Match Pulse-Width Modulation Operation (MPWM)

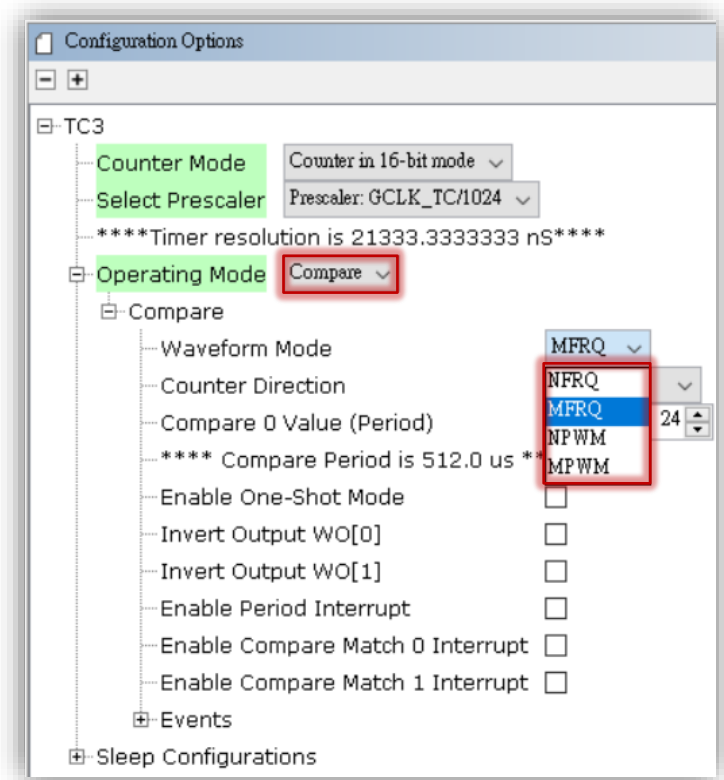
Compare 4 waveform generation mode

- There 4 difference waveform generation modes, the figure show the behavior at 4 mode.
Based on up count and 16-bits setting.
- We will focus the output waveform for 4 difference mode at this section.



Waveform Generating of Timer/Counter

- The mode is determining the waveform output style from WO[n] pins.
- It's selected by the waveform generation operation bit group in the Control A register. (CTRLA.WAVEGEN[1:0])
- At MHC, It's selected by “Operating Mode”, You can determine the mode under “Compare” mode.
 - Normal Frequency Generation (NFRQ)
 - Match Frequency Generation (MFRQ)
 - Normal Pulse-Width Modulation Operation (NPWM)
 - Match Pulse-Width Modulation Operation (MPWM)



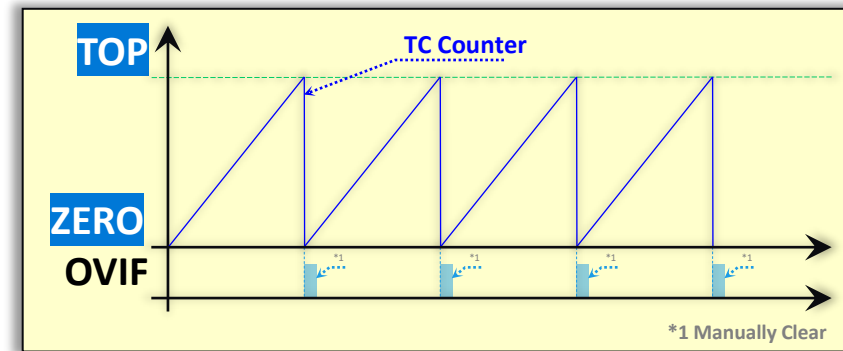
UP/DOWN Count and Counter Update

- Let me discuss the up and down counter firstly.
 - TC counter can be set to up or down count, when the value is reached, TC counter was updated to **TOP** or **ZERO**.
 - For UP Count Mode
 - Timing : when Counter = **TOP**
 - Result : was updated to **ZERO**
 - * If (Counter == TOP) Counter = 0;
 - For DOWN Count Mode
 - Timing : Counter = **ZERO**
 - Result : was updated to **TOP**
 - * If (Counter == ZERO) Counter = TOP;
- UP Count Mode**

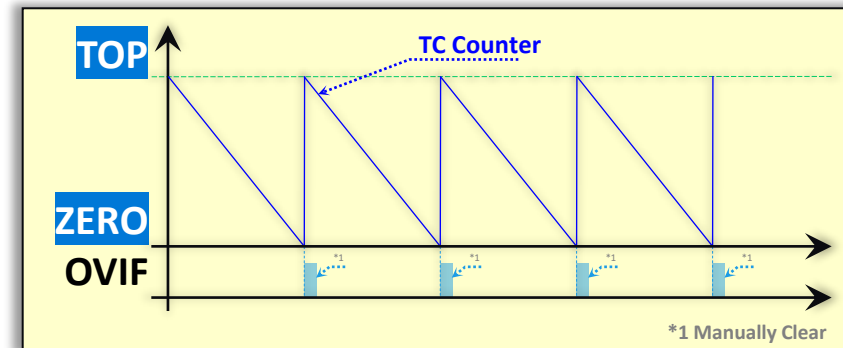
The diagram illustrates the UP Count Mode. The vertical axis represents the counter value, with markers for **TOP** (indicated by a blue box) and **ZERO** (indicated by a blue box). A horizontal dashed line marks the **TOP** level. The horizontal axis represents time. The counter value starts at **ZERO** and increases linearly until it reaches **TOP**. At this point, the counter resets to **ZERO** and begins a new cycle. This process repeats four times. A blue dotted line labeled "TC Counter" follows the sawtooth pattern. Small blue arrows labeled "OVIF" point to the reset points at **ZERO**. A note at the bottom right states "*1 Manually Clear".
- DOWN Count Mode**

The diagram illustrates the DOWN Count Mode. The vertical axis represents the counter value, with markers for **TOP** (indicated by a blue box) and **ZERO** (indicated by a blue box). A horizontal dashed line marks the **TOP** level. The horizontal axis represents time. The counter value starts at **TOP** and decreases linearly until it reaches **ZERO**. At this point, the counter resets to **TOP** and begins a new cycle. This process repeats four times. A blue dotted line labeled "TC Counter" follows the sawtooth pattern. Small blue arrows labeled "OVIF" point to the reset points at **TOP**.

UP Count Mode



DOWN Count Mode



Definition for TOP or ZERO

- **TOP**

- The **TOP** determine by the timer counter mode bit group in the Control A register. And

waveform generation operation bit group in the Control A register.

(CTRLA.MODE[1:0]) and (CTRLA.WAVEGEN[1:0])

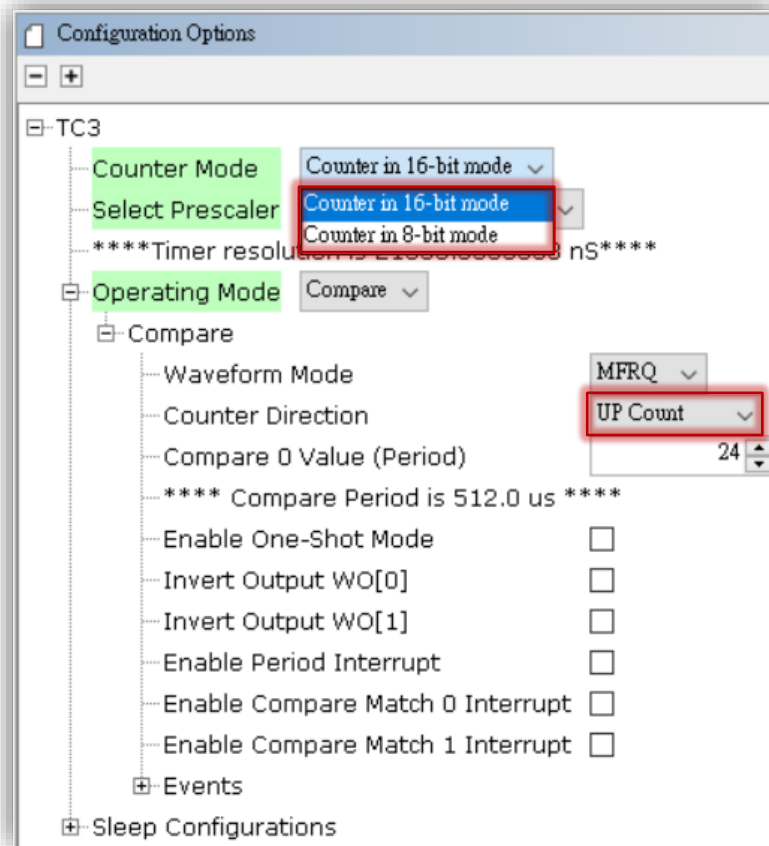
- **ZERO**

- The **ZERO** always “0”.

	TOP			ZERO
	8-bits	16-bits	32-bits	
NFRQ	PER	$\text{MAX}(2^{16}-1)$	$\text{MAX}(2^{32}-1)$	0
MFRQ	$\text{CCO}_{(8\text{-bits})}$	$\text{CCO}_{(16\text{-bits})}$	$\text{CCO}_{(32\text{-bits})}$	0
NPWM	PER	$\text{MAX}(2^{16}-1)$	$\text{MAX}(2^{32}-1)$	0
MPWM	$\text{CCO}_{(8\text{-bits})}$	$\text{CCO}_{(16\text{-bits})}$	$\text{CCO}_{(32\text{-bits})}$	0

Mode Selecting at MHC

- At MHC, It's selected by “Operating Mode” under “Compare” mode and “Counter Mode”.



Waveform at difference mode

- Refer below figures, this show the waveform was changed timing at difference modes.

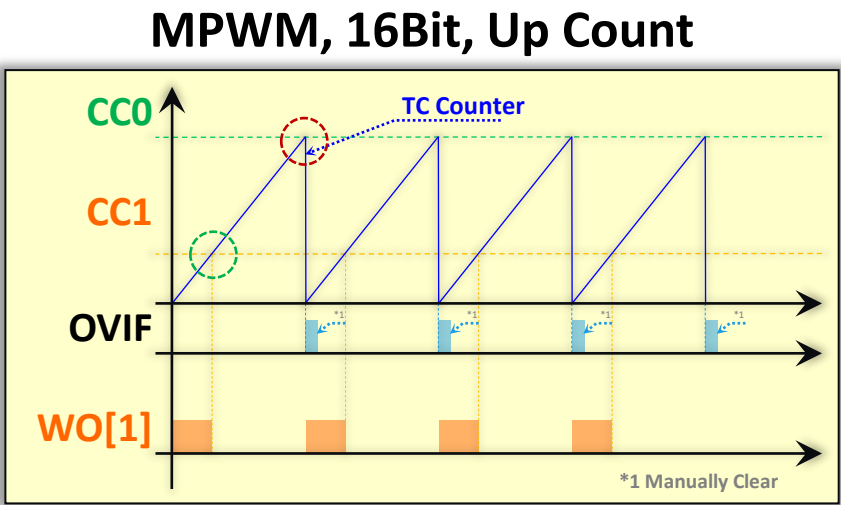
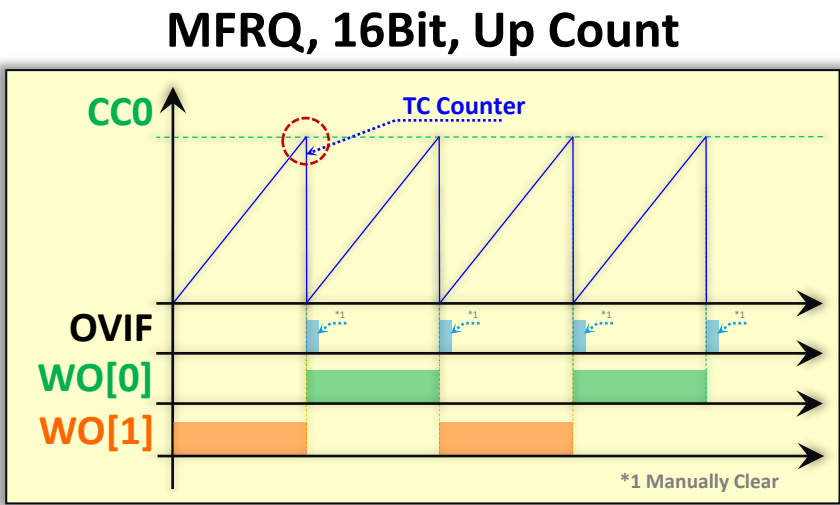
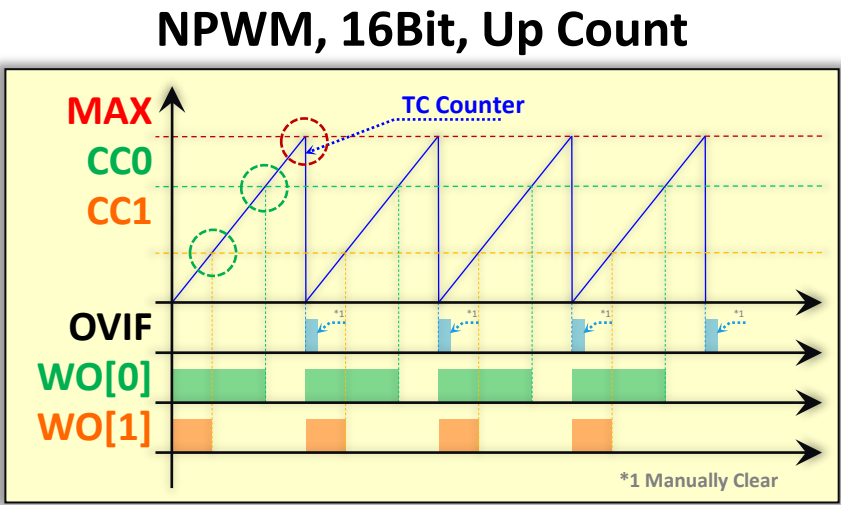
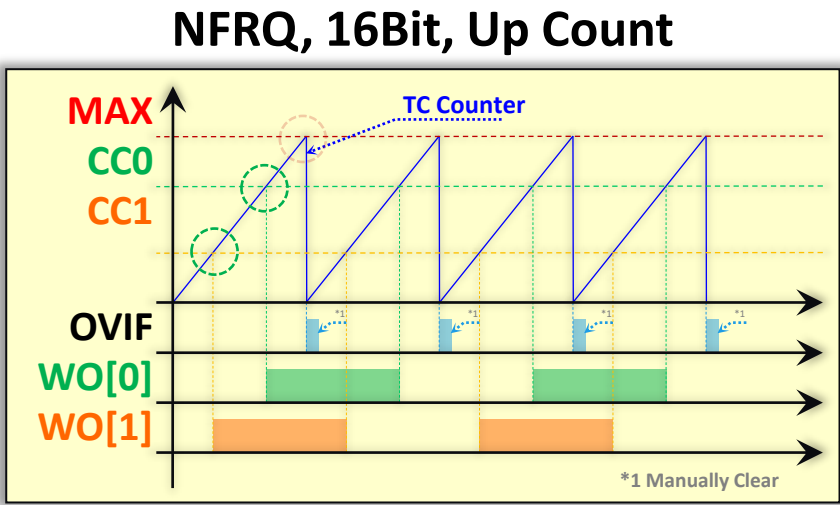
UP Count Mode

	Output		Interrupt
	Match	Update	
NFRQ	Toggle	Stable	TOP
MFRQ	Stable	Toggle	TOP
NPWM	Clear	Set	TOP
MPWM	Clear	Set	TOP

DOWN Count Mode

	Output		Interrupt
	Match	Update	
NFRQ	Toggle	Stable	ZERO
MFRQ	Stable	Toggle	ZERO
NPWM	Set	Clear	ZERO
MPWM	Set	Clear	ZERO

Waveform at difference mode - cont.



○ Match
○ Update

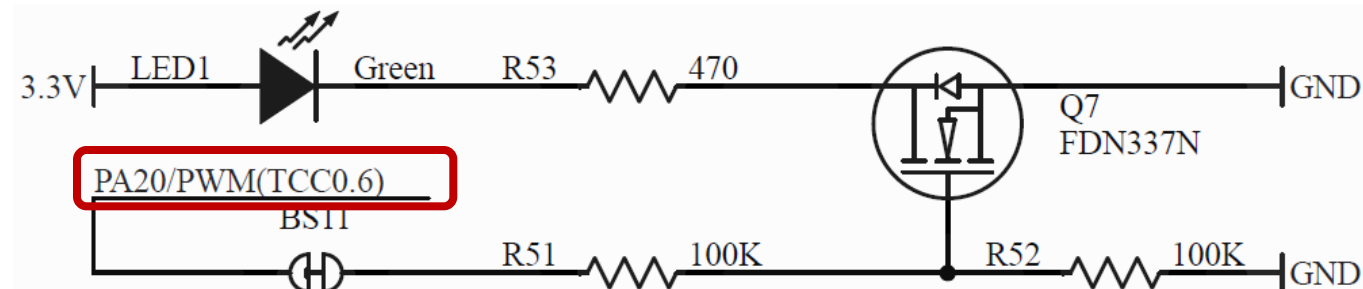
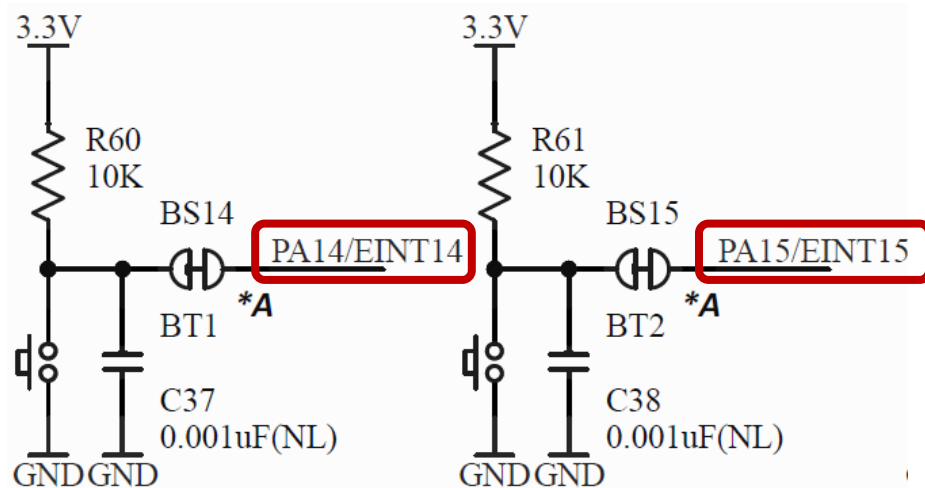
UP Count Mode

	Output		Interrupt
	Match	Update	
NFRQ	Toggle	Stable	TOP
MFRQ	Stable	Toggle	TOP
NPWM	Clear	Set	TOP
MPWM	Clear	Set	TOP

Lab6-1 TC Operation Mode - NFRQ

- Please continue from [SAM2001 Lab6 \(TCs Timer Method Interrupt Callback\)](#)
- Try to config TC3 to Compare - NFRQ (Normal Frequency Generation) mode and enable waveform output to PA14 and PA15.
 - WO[0] -> PA14, WO[1] -> PA15
 - GCLK0 -> 48MHz
 - CC0 -> 16,383
 - CC1 -> 32,767
- **Let's go !**

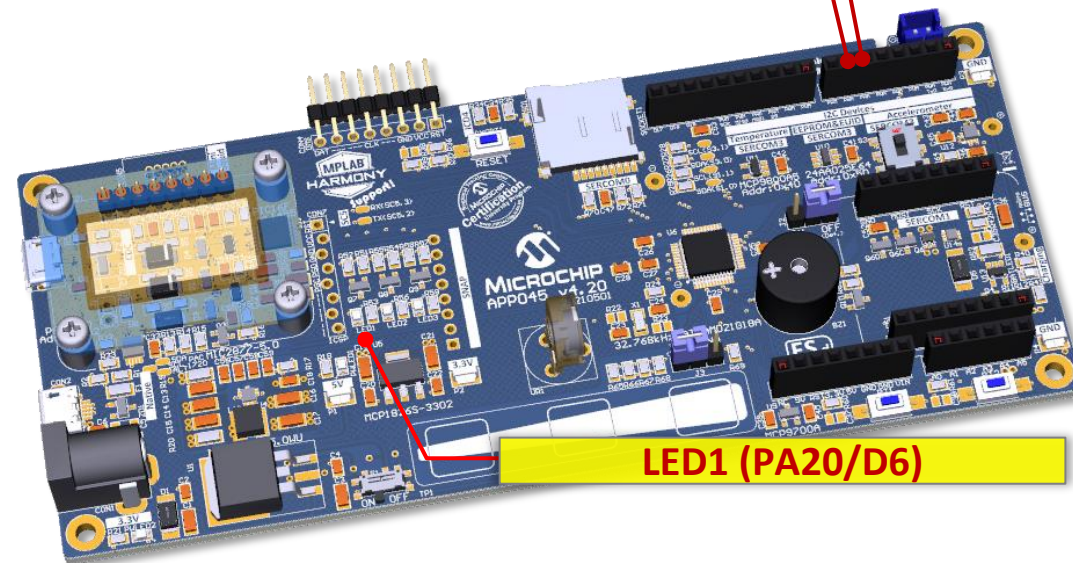
Lab6-1 TC Operation Mode - NFRQ



WO[0] (PA14/D4)

WO[1] (PA15/D5)

SOCKET1B		ARDUINO-M0-PRO	
PA11/RX(S2.3)	7	D0/PWM/RX	
PA10/TX(S2.2)	8	D1/PWM/TX	
PA08/MISO(S0.0)	9	D2/PWM	
PA09/SS(S0.1)	10	D3/PWM	
PA14/EINT14	11	D4/PWM	
PA15/EINT15	12	D5/PWM	
PA20/PWM(TCC0.6)	13	D6/PWM	
PA21/PWM(TCC0.7)	14	D7/PWM	



✖ Lab6-1 TC Operation Mode - NFRQ

Step 1

- Configure **TC3 Period to NFRQ mode and Enable Interrupts** in configuration.

The screenshot shows the configuration window for TC3. The following settings are highlighted with yellow callouts:

- Counter Mode:** Counter in 16-bit mode (16-bit mode)
- Select Prescaler:** Prescaler: GCLK_TC (GCLK_TC)
- Operating Mode:** Compare (Compare)
- Waveform Mode:** NFRQ (NFRQ)
- Counter Direction:** UP Count (UP Count)
- Compare 0 Value:** 16,383 (CC0 : 16,383)
- Compare 1 Value:** 32,767 (CC1 : 32,767)
- Enable Period Interrupt:** ☒ (Check Interrupt Enable)
- Enable Compare Match 0 Interrupt:** ☒ (Check Interrupt Enable)
- Enable Compare Match 1 Interrupt:** ☒ (Check Interrupt Enable)

Additional visible settings include:

- Timer resolution: 20.8333333333 ns
- Period Value: 0xffff
- Compare Period: 1365.3125 us
- Enable One-Shot Mode: ☐
- Invert Output WO[0]: ☐
- Invert Output WO[1]: ☐
- Events: ☐
- Sleep Configurations: ☐

⚙️ Lab6-1 TC Operation Mode - NFRQ

Step 2

- Disable BT1(PA14) and BT2(PA15) GPIO function firstly.
- Enable TC3 Waveform Output 0 (W0[0], TC3_WO0) and TC3 Waveform Output 1 (W0[1], TC3_WO1) to PA14, PA15

GPIO

Module	Function	PA11	VDDIO	GNDDIO	PB10	PB11	PA12	PA13	BT1	BT2
	GCLK_IO7	6	17	18	19	20	21	22	23	24
GPIO	GPIO									
	I2S_FS0									
	I2S_MC0									

BT1, BT2

Click Disable

TC3

Module	Function	PA11	VDDIO	GNDDIO	PB10	PB11	PA12	PA13	TC3_WO0	TC3_WO1
	SYSCTRL_XOUT32	6	17	18	19	20	21	22	23	24
TC3	TC3_WO0									
	TC3_WO1									

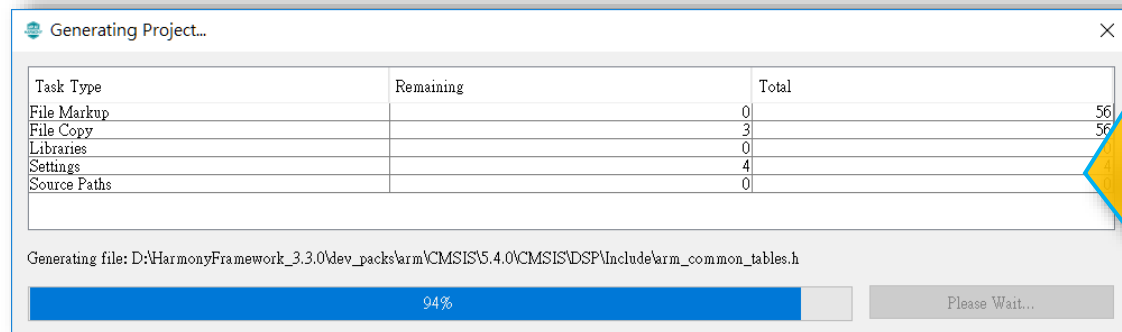
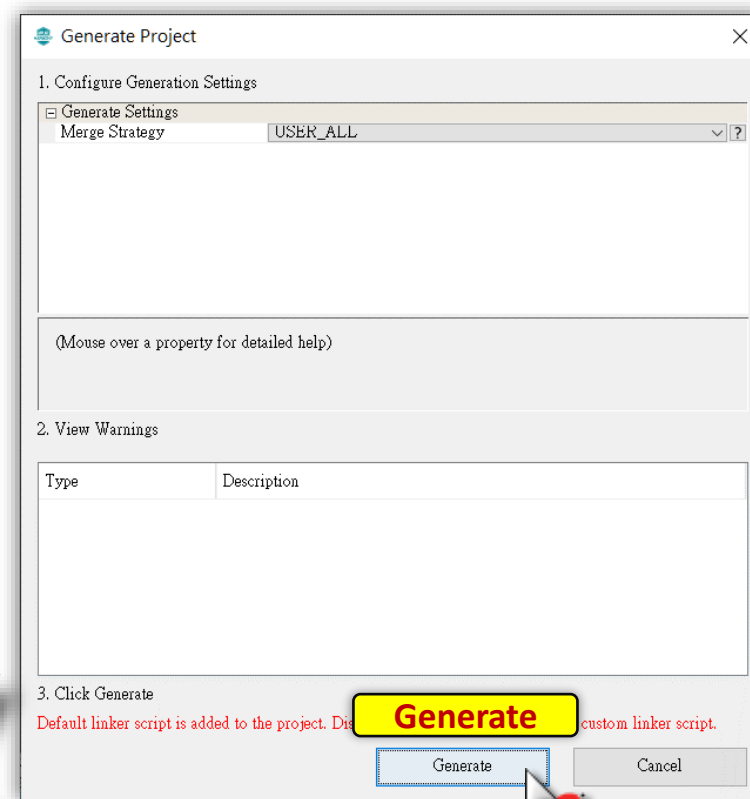
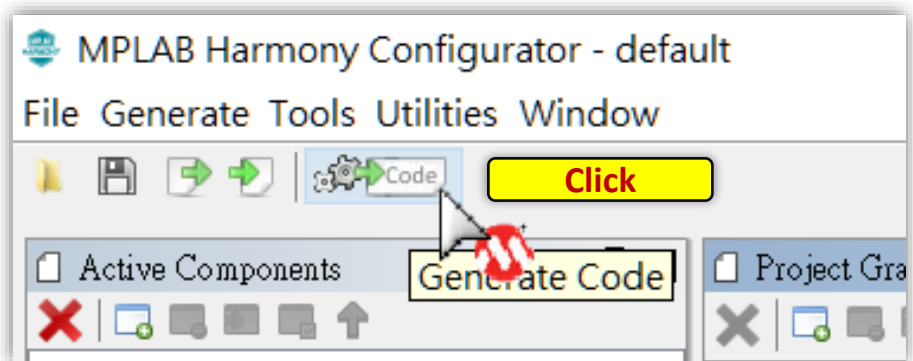
TC3_WO[0,1]

Click Enable

⚙️ Lab6-1 TC Operation Mode - NFRQ

Step 3

- Click  to Generate Code.



Lab6-1 TC Operation Mode - NFRQ

Step 4

```
...
uint32_t i = 0;

// TODO 6-1.01
// void TC3_TimerExpired(TC_TIMER_STATUS status, uintptr_t context)
// {
//     if( status & TC_INTFLAG_OVF_Msk)
//         LED1_Toggle();
// }

void TC3_CompareUpdated(TC_COMPARE_STATUS status, uintptr_t context)
{
    if (status & TC_COMPARE_STATUS_OVERFLOW)
    {
        LED1_Set();
        for (i = 0; i < 50; i++);
        LED1_Clear();
    }
}

void TC4_TimerExpired(TC_TIMER_STATUS status, uintptr_t context) { ... }

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


Lab6-1 TC Operation Mode - NFRQ

Step 5

```
int main (void)
{

    SYS_Initialize(NULL);

    // TODO 6-1.02
    // TC3_TimerCallbackRegister( TC3_TimerExpired, (uintptr_t )NULL );
    // TC3_TimerStart();
    TC3_CompareCallbackRegister(TC3_CompareUpdated, (uintptr_t) NULL);
    TC3_CompareStart();

    TC4_TimerCallbackRegister(TC4_TimerExpired, (uintptr_t) NULL);
    TC4_TimerStart();

    while ( true )
    {
        SYS_Tasks();

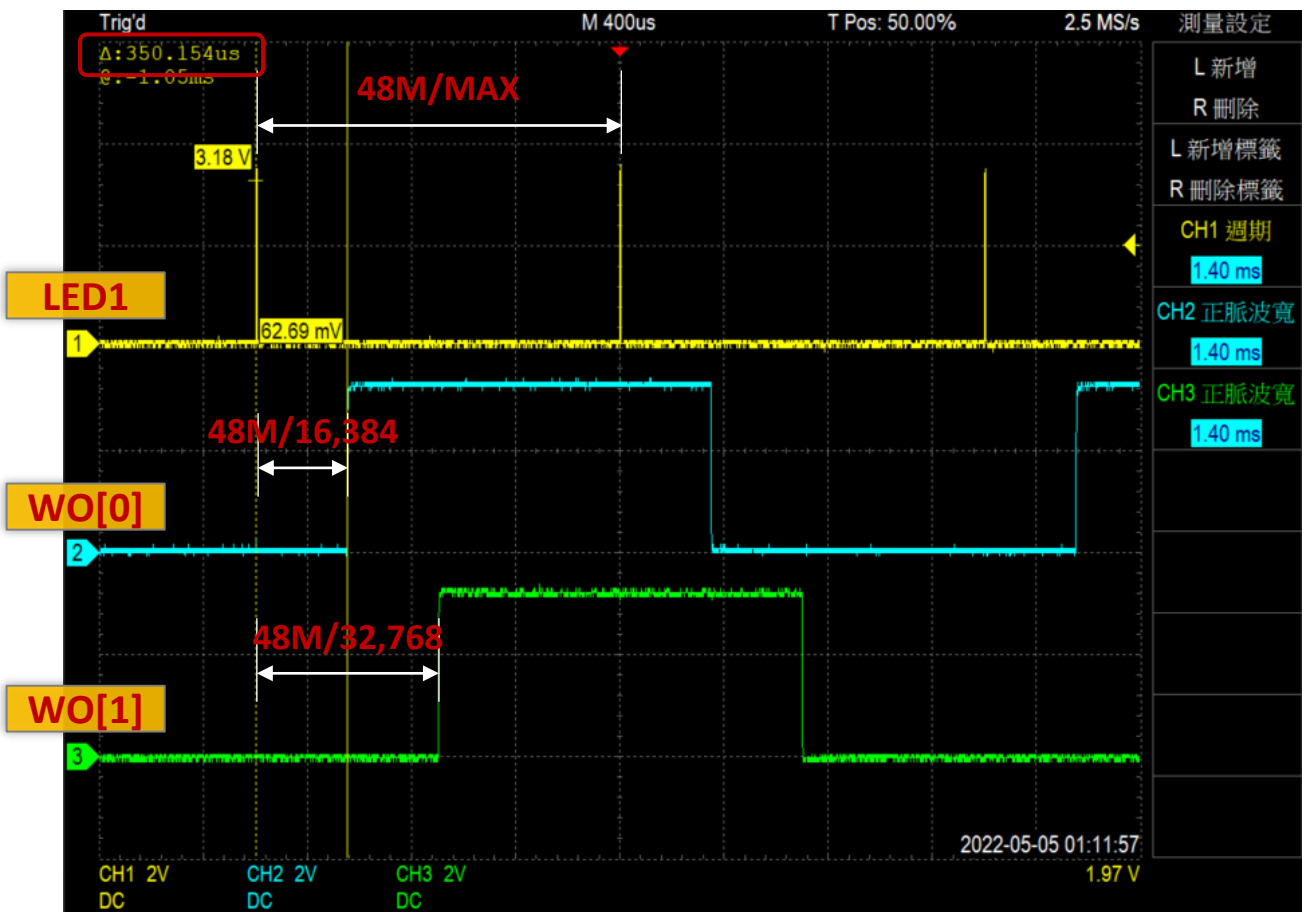
        // TODO 6-1.03
        // if (BT1_Get()) LED3_Clear();
        // else LED3_Set();
    }

    return ( EXIT_FAILURE);
}
```

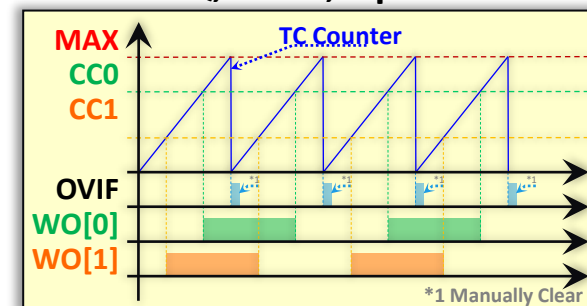
Lab6-1 TC Operation Mode - NFRQ

Result

- Measure the waveform of LED1, WO[0] and WO[1].

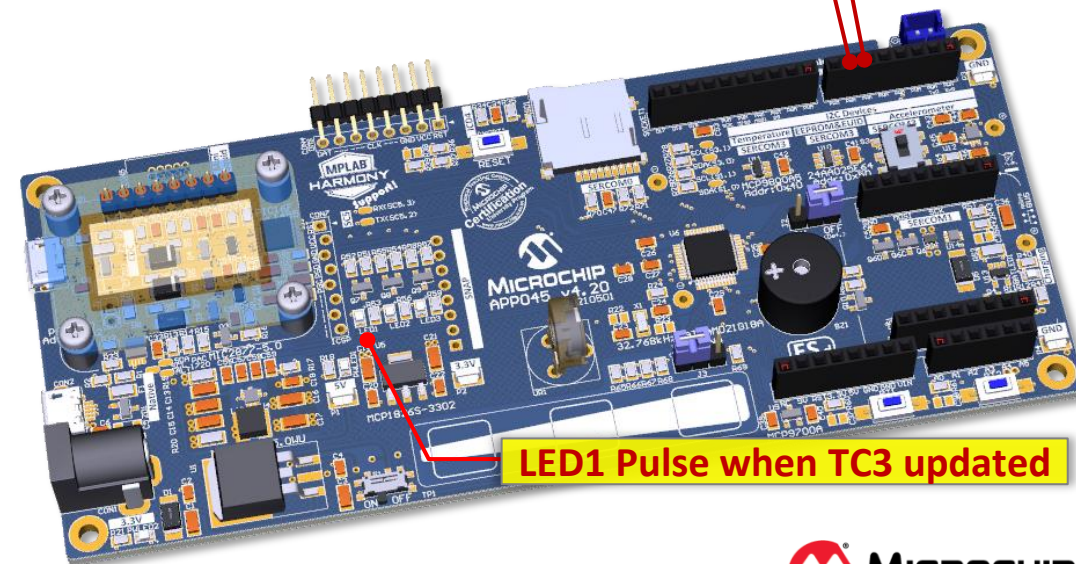


NFRQ, 16Bit, Up Count



WO[0] (PA14/D4)

WO[1] (PA15/D5)



Lab6-2 TC Operation Mode - MFRQ

- Please continue from [SAM2001 Lab6-1 \(TC Compare Mode - NFRQ\)](#)
- Try to config TC3 to
Compare - MFRQ (Match Frequency Generation) mode
 - CC0 -> 4,799
- **Let's go !**

⚙️ Lab6-2 TC Operation Mode - MFRQ

Step 1

- Configure **TC3 Period to MFRQ mode** in configuration.

The screenshot shows the configuration window for TC3. The following settings are highlighted with callouts:

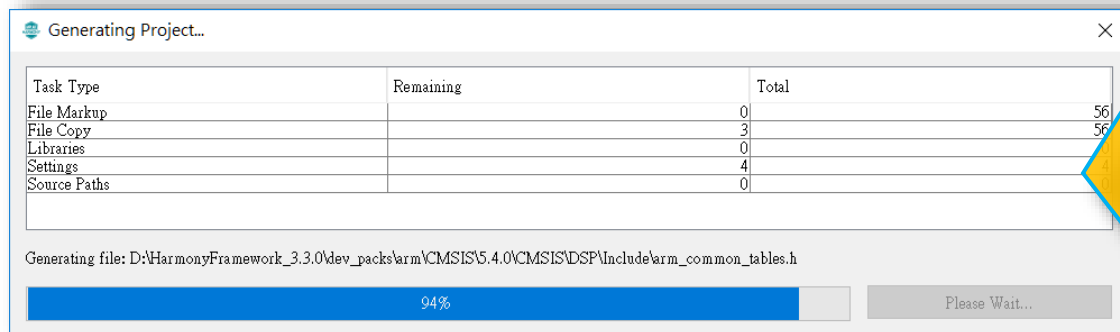
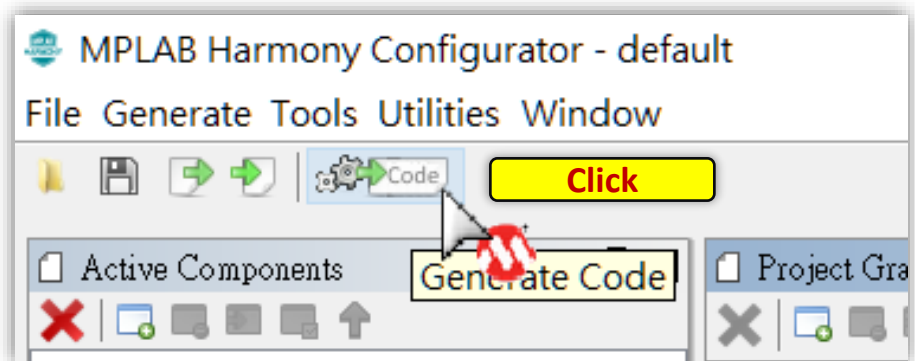
- Counter Mode:** Counter in 16-bit mode (16-bit mode)
- Select Prescaler:** Prescaler: GCLK_TC (GCLK_TC)
- Operating Mode:** Compare (Compare)
- Waveform Mode:** MFRQ (MFRQ)
- Counter Direction:** UP Count (UP Count)
- Compare 0 Value (Period):** 4,799 (CC0 : 4,799)
- Interrupts:** Enable Period Interrupt, Enable Compare Match 0 Interrupt, and Enable Compare Match 1 Interrupt are all checked (Check Interrupt Enable)

Additional text in the window includes: "****Timer resolution is 20.8333333333 ns" and "**** Compare Period is 99.9791666667 us ****".

⚙️ Lab6-2 TC Operation Mode - MFRQ

Step 2

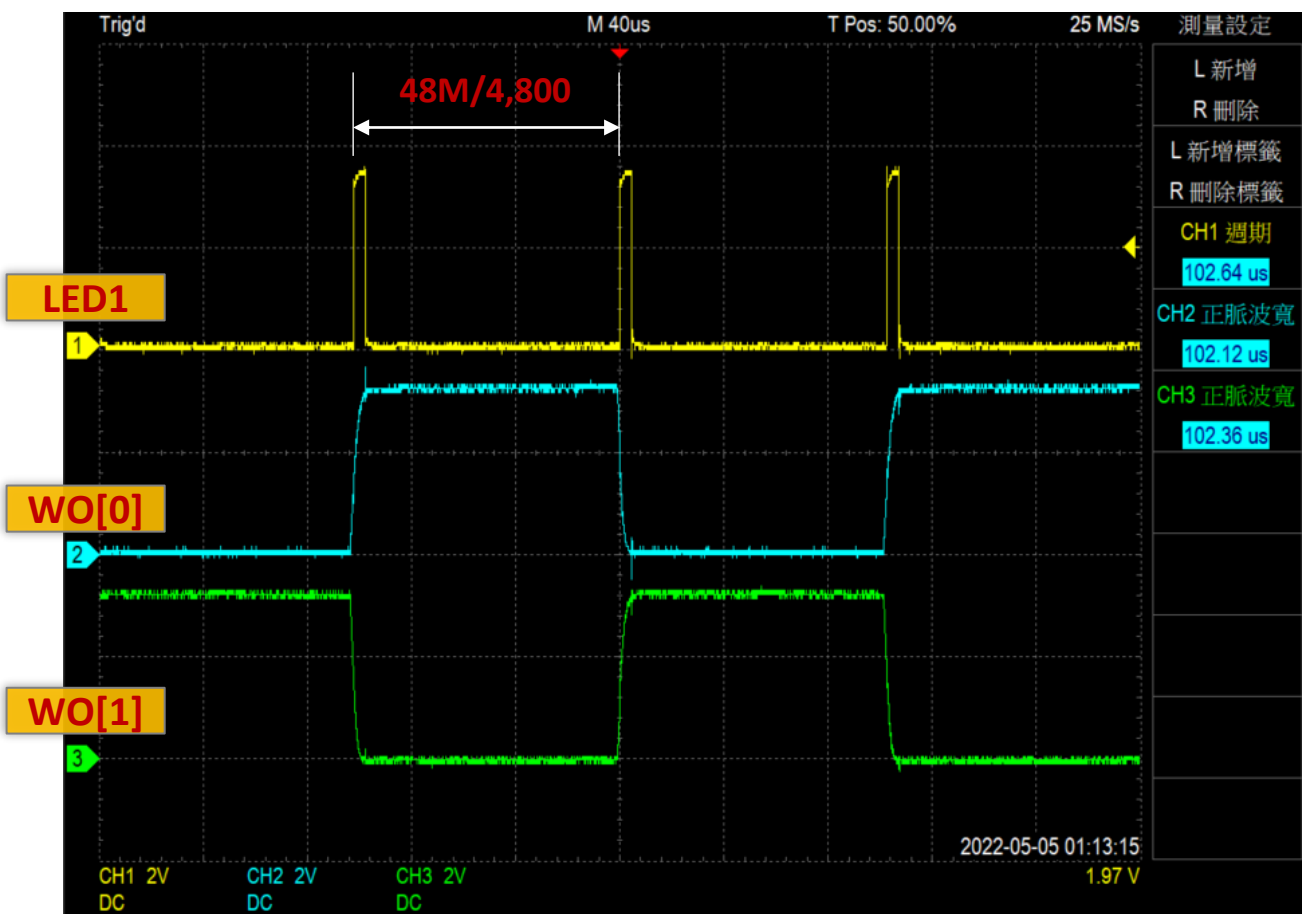
- Click  to Generate Code.



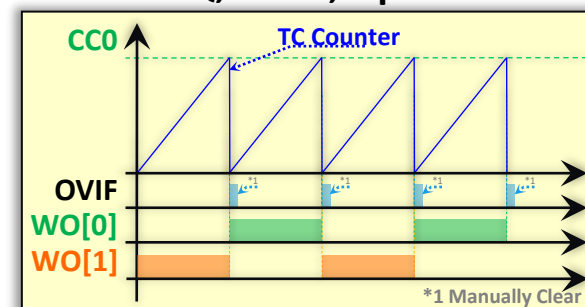
✂ Lab6-2 TC Operation Mode - MFRQ

Result

- Measure the waveform of LED1, WO[0] and WO[1].

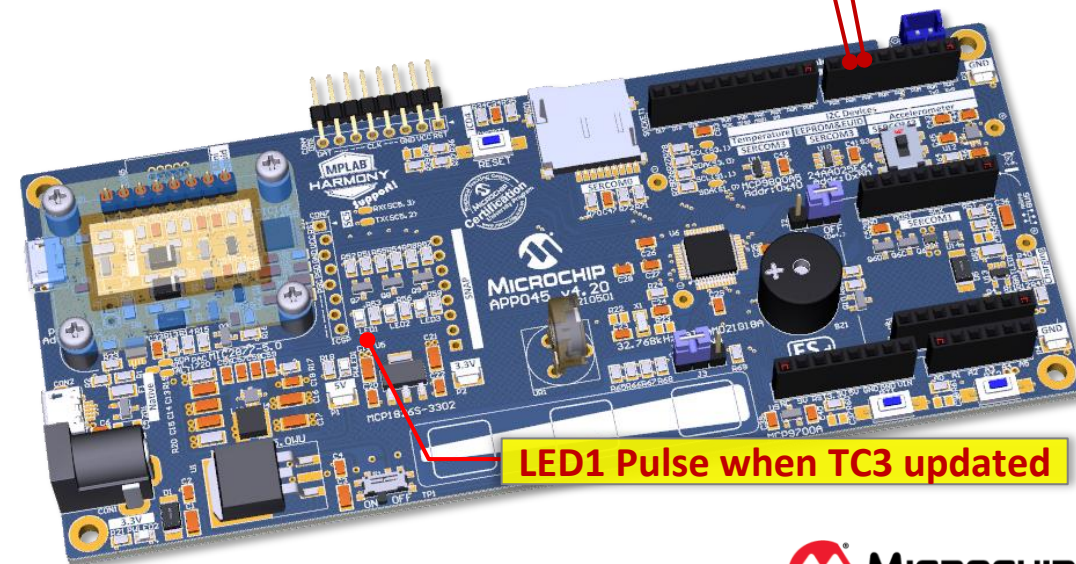


MFRQ, 16Bit, Up Count



WO[0] (PA14/D4)

WO[1] (PA15/D5)



Lab6-3 TC Operation Mode - NPWM

- Please continue from [SAM2001 Lab6-1 \(TC Compare Mode - NFRQ\)](#)
- Try to config TC3 to
Compare - NPWM (Normal Pulse-Width Modulation Operation) mode
 - CC0 -> 16,383
 - CC1 -> 32,767
- **Let's go !**

⚙️ Lab6-3 TC Operation Mode - NPWM

Step 1

- Configure **TC3 Period to NPWM mode and Enable Interrupts** in configuration.

TC3

Counter Mode: Counter in 16-bit mode **16-bit mode**

Select Prescaler: Prescaler: GCLK_TC **GCLK_TC**

****Timer resolution is 20.8333333333 ns****

Operating Mode: Compare **Compare**

Compare

Waveform Mode: NPWM **NPWM**

Counter Direction: UP Count **UP Count**

**** Period Value is 0xffff ****

Compare 0 Value: 16,383 **CC0 : 16,383**

Compare 1 Value: 32,767 **CC1 : 32,767**

**** Compare Period is 1365.3125 us ****

Enable One-Shot Mode: ☐

Invert Output WO[0]: ☐

Invert Output WO[1]: ☐

Enable Period Interrupt: ☒ **Check Interrupt Enable**

Enable Compare Match 0 Interrupt: ☒

Enable Compare Match 1 Interrupt: ☒

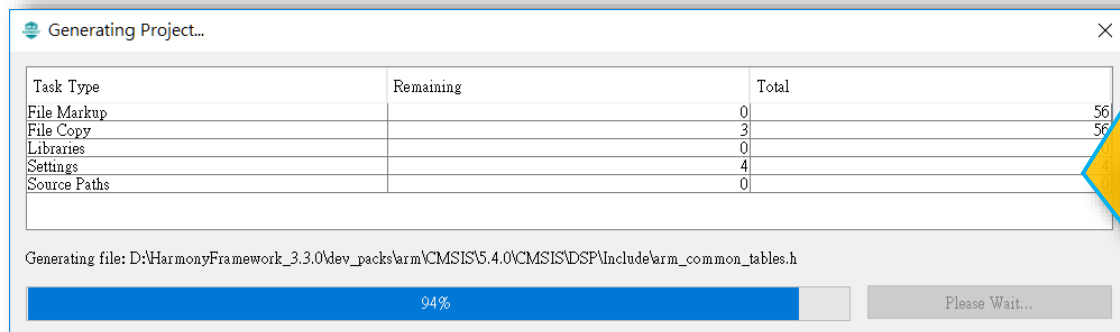
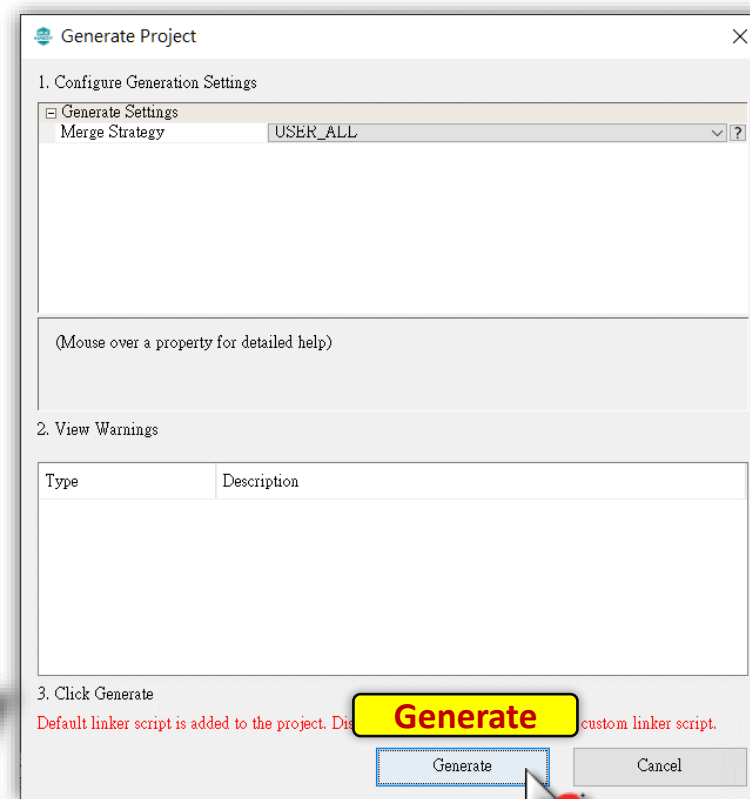
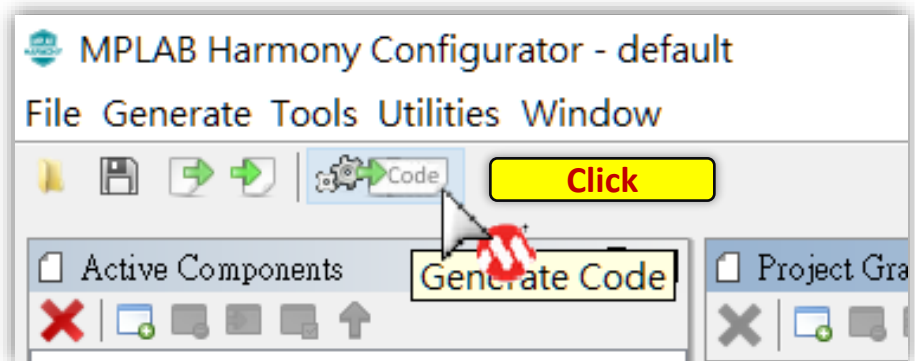
Events

Sleep Configurations

⚙️ Lab6-3 TC Operation Mode - NPWM

Step 2

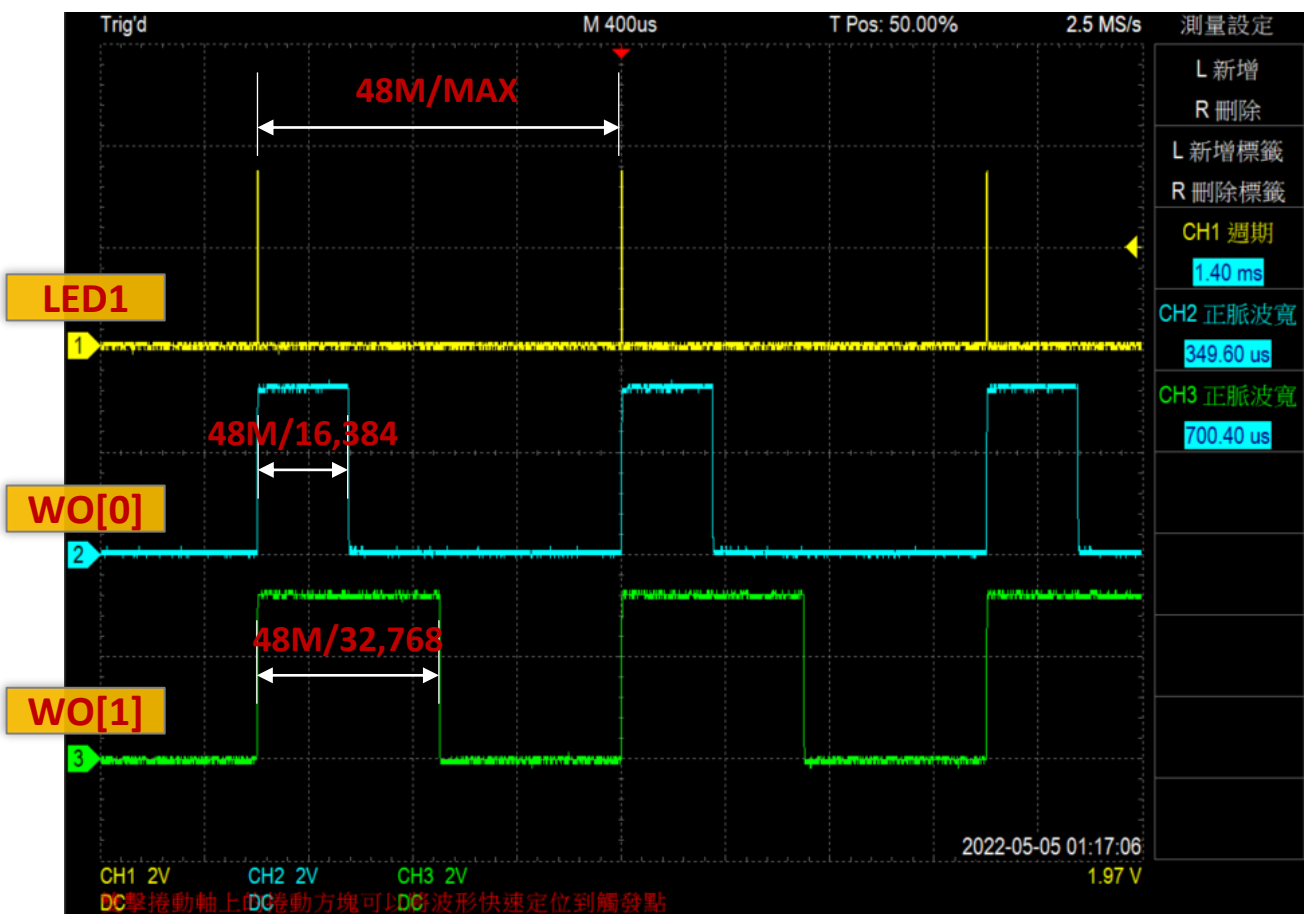
- Click  to Generate Code.



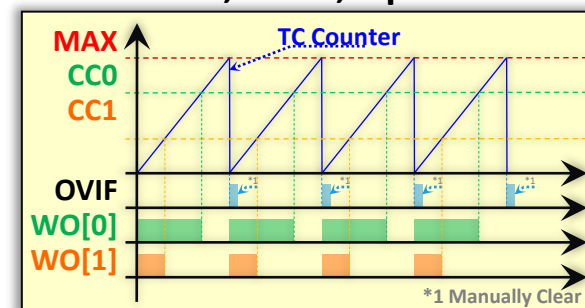
Lab6-3 TC Operation Mode - NPWM

Result

- Measure the waveform of LED1, WO[0] and WO[1].

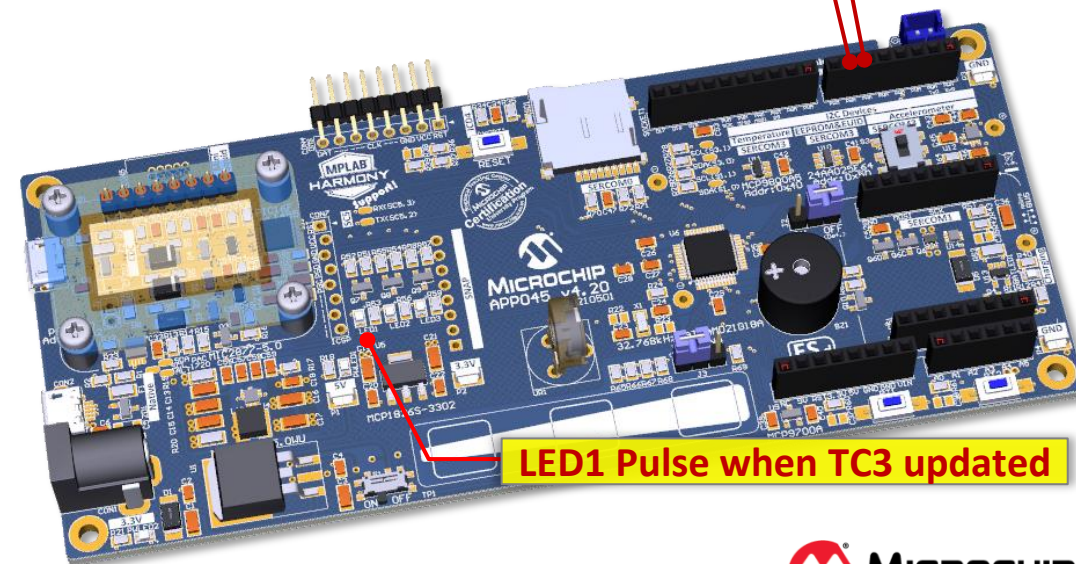


NPWM, 16Bit, Up Count



WO[0] (PA14/D4)

WO[1] (PA15/D5)



🔧 Lab6-4 TC Operation Mode - MPWM

- Please continue from [SAM2001 Lab6-1 \(TC Compare Mode - NFRQ\)](#)
- Try to config TC3 to
Compare - MPWM (Match Pulse-Width Modulation Operation) mode
 - CC0 -> 4,799
 - CC1 -> 2,399
- **Let's go !**

⚙️ Lab6-4 TC Operation Mode - MPWM

Step 1

- Configure **TC3 Period to MPWM mode and Enable Interrupts** in configuration.

The screenshot shows the configuration for TC3. The following settings are visible and annotated:

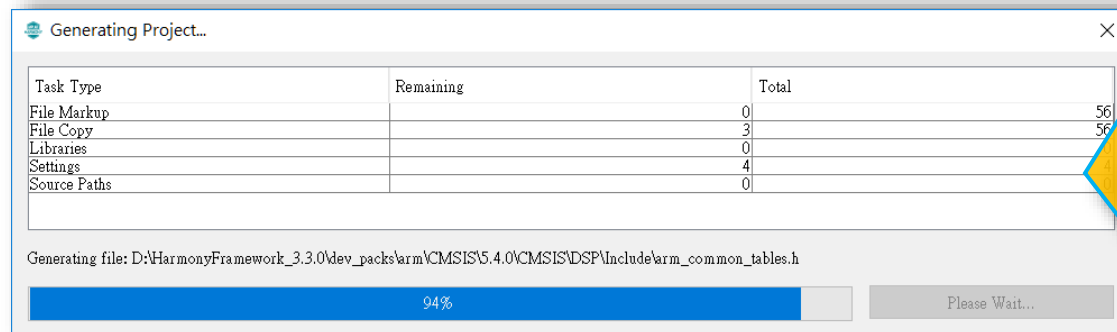
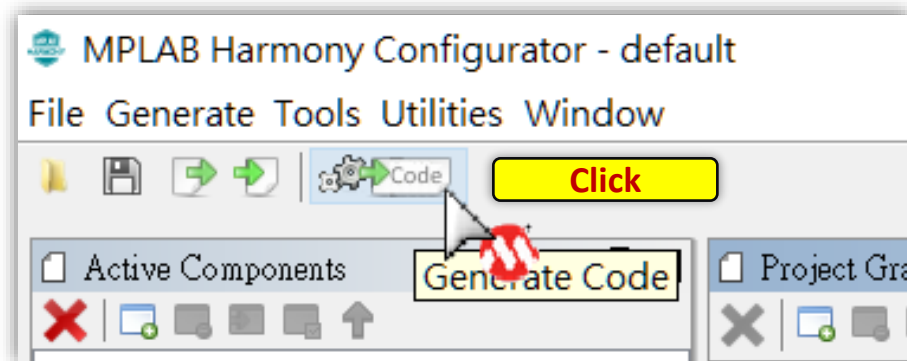
- Counter Mode:** Counter in 16-bit mode (Annotated: 16-bit mode)
- Select Prescaler:** Prescaler: GCLK_TC (Annotated: GCLK_TC)
- Operating Mode:** Compare (Annotated: Compare)
- Compare:**
 - Waveform Mode:** MPWM (Annotated: MPWM)
 - Counter Direction:** DOWN Count (Annotated: UP Count)
 - Compare 0 Value (Period):** 4,799 (Annotated: CC0 : 4,799)
 - Compare 1 Value:** 2,399 (Annotated: CC1 : 2,399)
- Interrupts:**
 - Enable Period Interrupt:** ☒ (Annotated: Check Interrupt Enable)
 - Enable Compare Match 0 Interrupt:** ☒
 - Enable Compare Match 1 Interrupt:** ☒

Additional text in the window: "****Timer resolution is 20.8333333333 ns" and "**** Compare Period is 99.9791666667 us ****".

⚙️ Lab6-4 TC Operation Mode - MPWM

Step 2

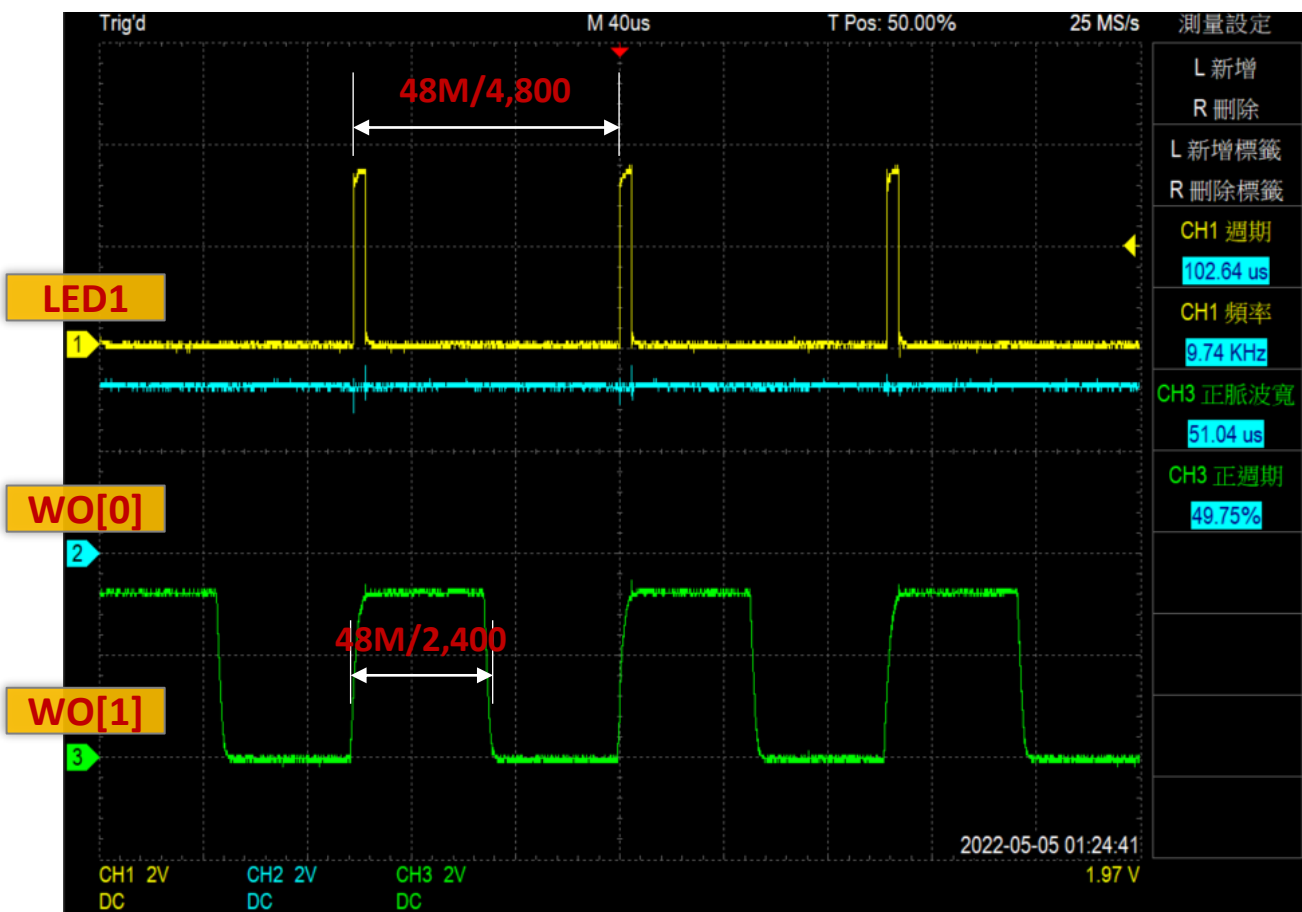
- Click  to Generate Code.



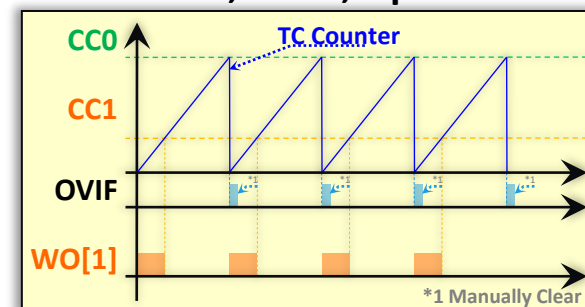
Lab6-4 TC Operation Mode - MPWM

Result

- Measure the waveform of LED1, WO[0] and WO[1].

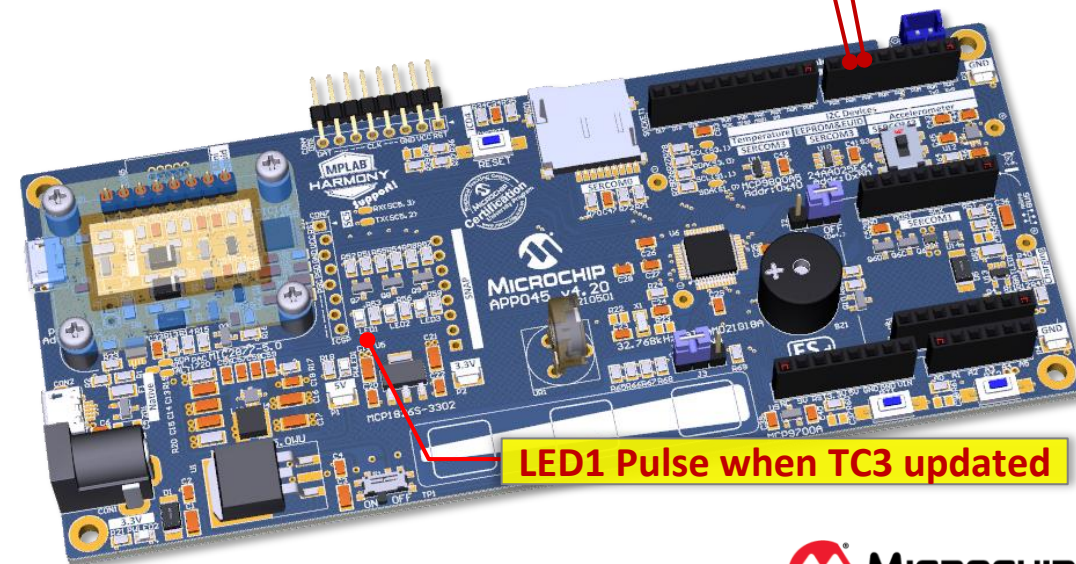


MPWM, 16Bit, Up Count



WO[0] (PA14/D4)

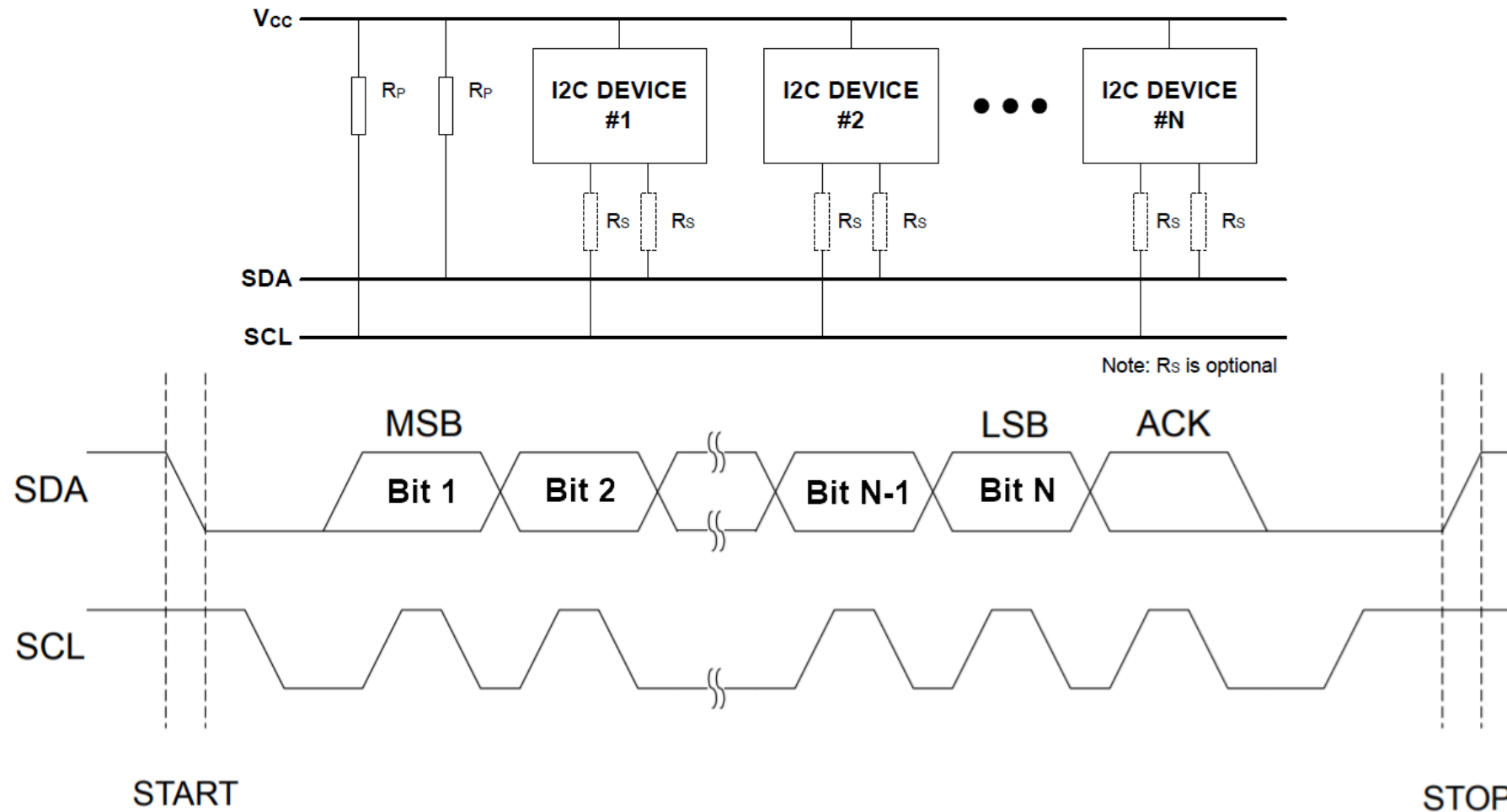
WO[1] (PA15/D5)



SERCOM – I2C Architecture

What's I2C ?

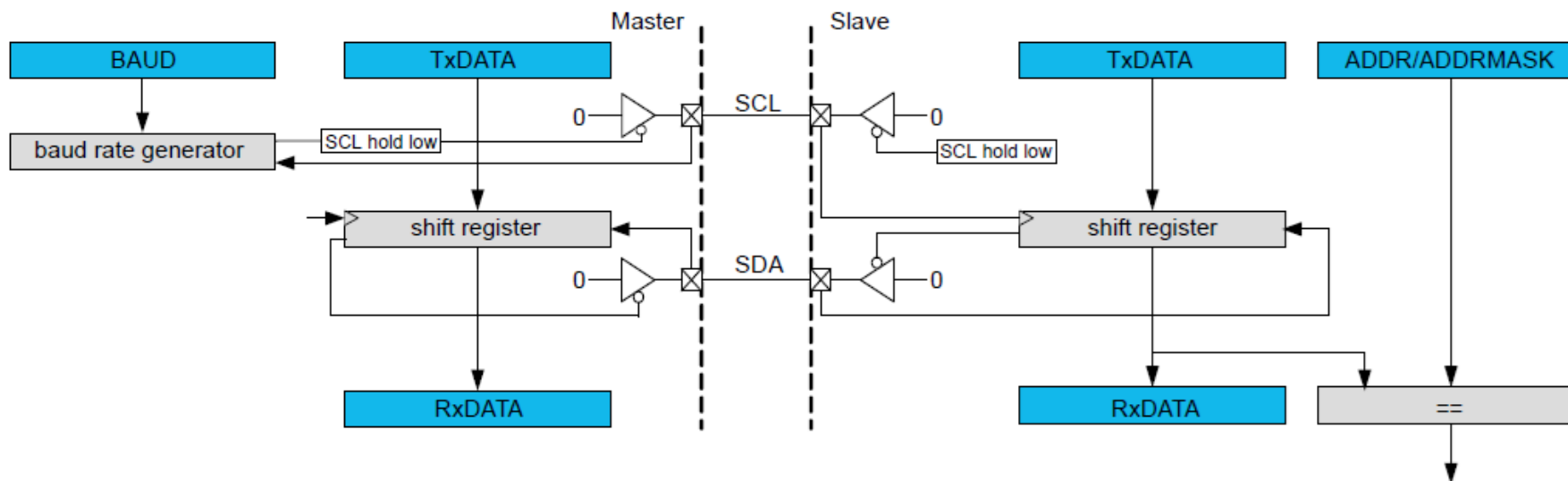
- The I2C (Inter-Integrated Circuit) is a synchronous, multi-master, multi-slave, packet switched, single-ended, serial computer bus invented in 1982 by Philips Semiconductor.



SERCOM – I2C

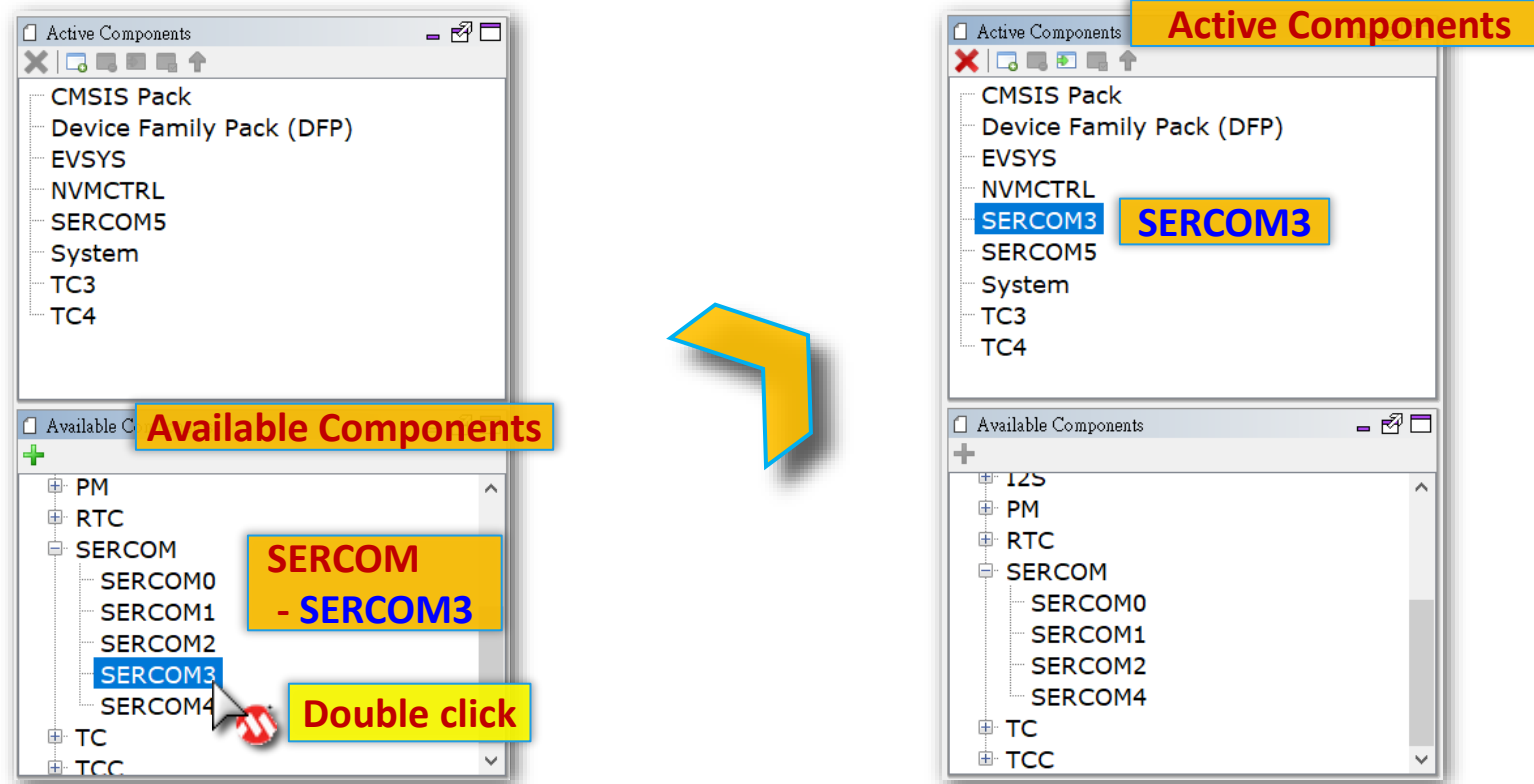
The I2C interface is one of the available modes in the serial communication interface (SERCOM).

- Master or slave operation can be used with DMA.
- Philips I2C / SMBus™/ PMBus compatible.
- Support of 100kHz and 400kHz, 1MHz and 3.4MHz mode.
- Up to 16-bytes internal FIFO.
- Operation in all sleep modes.
- Wake-up on address match (Slave mode)



Add SERCOM Function using MHC

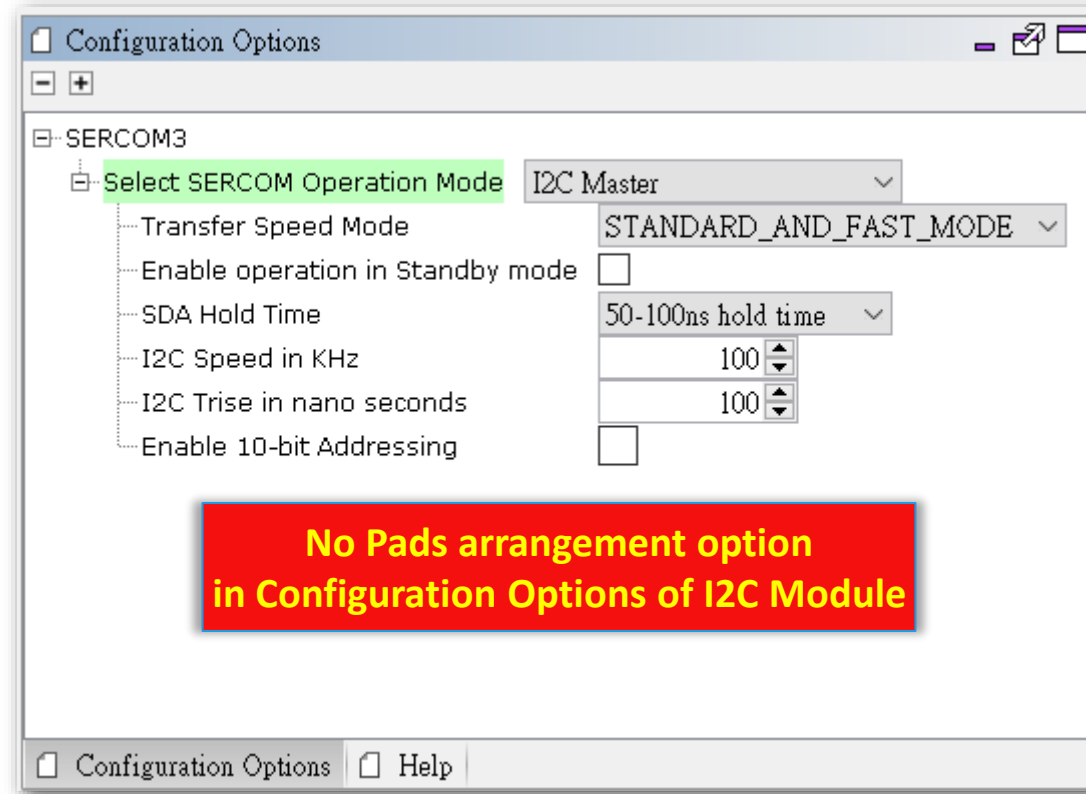
- Find **SERCOM** component in Available Components window.
- Double click **SERCOM3** to add to Active Components window.



SERCOM-I2C

PINMUX Configuration Example

- No need to arrange SERCOM pads in SERCOM-I2C module.
- SERCOM-I2C module use dedicate pads, **PAD0** for **SDA** and **PAD1** for **SCL**.



SERCOM-I2C

PLIB SERCOM-I2C pad configurations

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

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)

SERCOM-I2C

PINMUX Configuration Example

- SERCOM allow limited alternate pad index assignment.
- For example : PA22 as SERCOM3_PAD0 (SDA)
PA23 as SERCOM3_PAD1 (SCL)

Pin ⁽¹⁾			I/O Pin	Supply	A	B ⁽²⁾⁽³⁾					C	D	E	F	G	H
SAMD21E	SAMD21G	SAMD21J			EIC	REF	ADC	AC	PTC	DAC	SERCOM ⁽²⁾⁽³⁾	SERCOM-ALT	Tc ⁽⁴⁾ /TCC	TCC	COM	AC/ GCLK
21	31		PA22	VDDIO	EXTINT[6]				X[10]		SERCOM3/ PAD[0]					GCLK_IO[6]
22	32		PA23	VDDIO	EXTINT[7]				X[11]		SERCOM3/ PAD[1]				USB/SOF 1kHz	GCLK_IO[7]

Pin Table

Package: TQFP48

Module	Function	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44
	SERCOM2_PAD3																											
	SERCOM3_PAD0																											
	SERCOM3_PAD1																											
	SERCOM3_PAD2																											
	SERCOM3_PAD3																											
	SERCOM4_PAD0																											
	SERCOM4_PAD1																											

SERCOM-I2C

Interface Routines

```
void SERCOM3_I2C_Initialize( void );  
bool SERCOM3_I2C_Read(uint16_t address, uint8_t *pdata, uint32_t length);  
bool SERCOM3_I2C_Write(uint16_t address, uint8_t *pdata, uint32_t length);  
bool SERCOM3_I2C_WriteRead(uint16_t address, uint8_t *wdata,  
                             uint32_t wlength, uint8_t *rdata, uint32_t rlength);  
  
bool SERCOM3_I2C_IsBusy(void);  
SERCOM_I2C_ERROR SERCOM3_I2C_ErrorGet(void);  
void SERCOM3_I2C_CallbackRegister(SERCOM_I2C_CALLBACK callback, uintptr_t context);
```

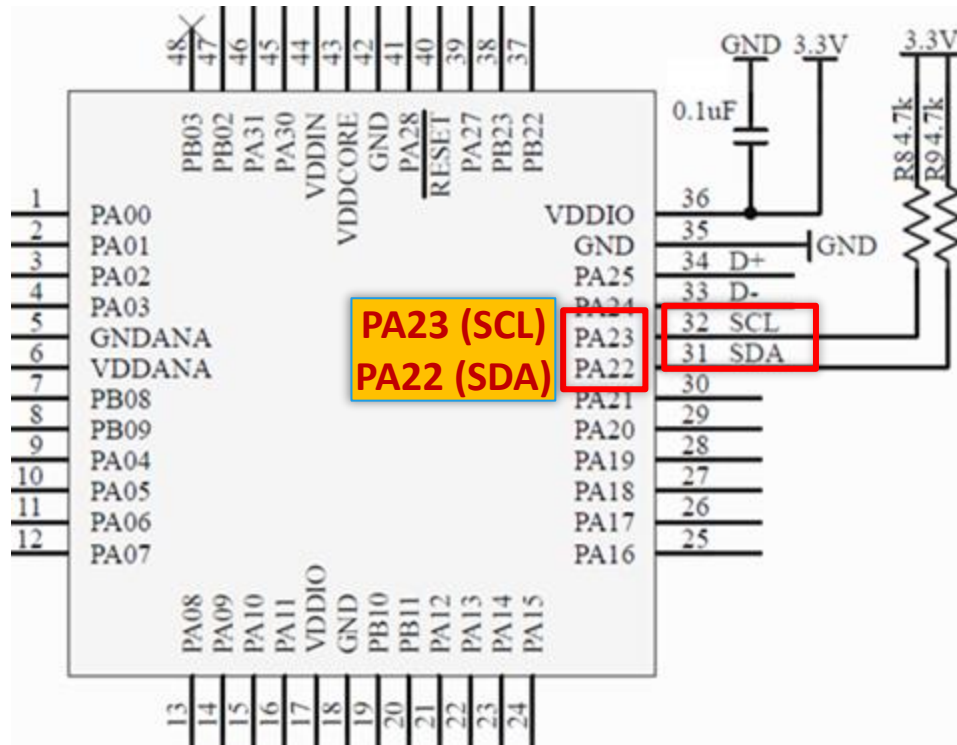
SERCOM-I2C Code Example

```
...  
  
int main (void)  
{  
    SYS_Initialize ( NULL );  
    ...  
    SERCOM3_I2C_WriteRead( Slave_Address, WriteBytes, 1, ReadBytes, 2 );  
    while( true )  
    {  
        ...  
        SERCOM3_I2C_Read( Slave_Address, ReadBytes, 2 );  
    }  
}
```

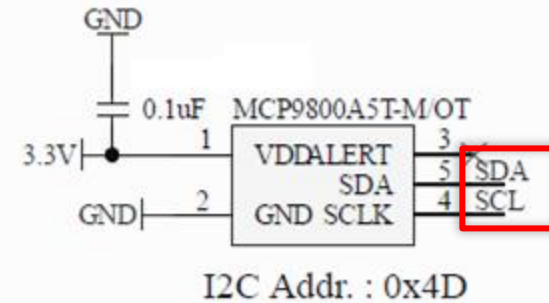
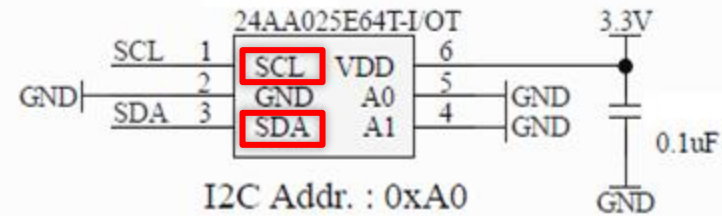
⚙️ Lab11-1 SERCOM – I2C Access Temperature Sensor

- Please continue from [SAM2001 Lab11 \(SERCOM - UART TxRx Interrupt Callback with myprintf\)](#)
- Add **SERCOM3-I2C** master function to project to communicate the on-board **Digital Temperature Sensor MCP9800A5T**.
- Enable **SERCOM3_PAD0(PA22)** and **SERCOM_3_PAD1(PA23)** for I2C **SDA** and **SCL**.
- Send Ambient Temperature Register (**TA**) command **0x00** to acquire temperature data from Digital Temperature Sensor **MCP9800A** and then continue to read data.
- Output temperature data to UART.
- **MCP9800 configuration:**
 - **SDA** connect to **PA22** (**SERCOM3 PAD0**)
 - **SCL** connect to **PA23** (**SERCOM3 PAD1**)
 - Slave address is **0x4D**.
 - **TA** register address is **0x00**.
- **Let's go !**

APP045 SERCOM-I2C Schematic



EEPROM
I2C Address : 0xA0



Digital Temperature Sensor
I2C Address : 0x4D

MCP9800A access command and Temperature calculation

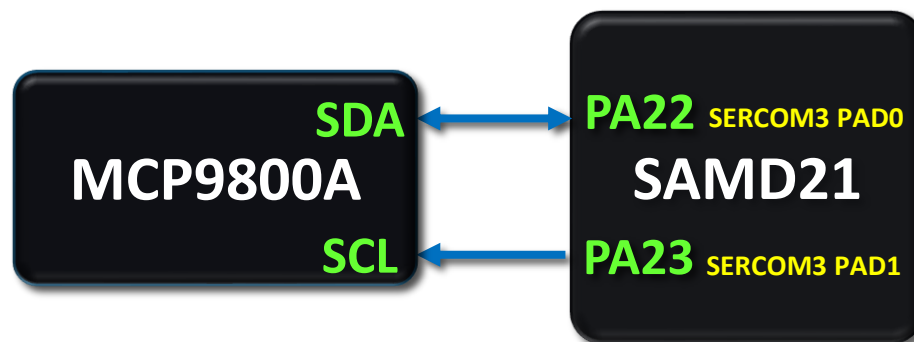
$$T_A = \text{Code} \times 2^{-4}$$

T_A = Ambient Temperature (°C)
Code = MCP9800 output in decimal

$$\text{Result} = (\text{TA}[0] \ll 4 \mid \text{TA}[1] \gg 4) / 16$$

Register Pointer P1 P0	MSB/ LSB	Bit Assignment							
		7	6	5	4	3	2	1	0
Ambient Temperature Register (T _A)									
0 0	MSB	Sign	2 ⁶ °C	2 ⁵ °C	2 ⁴ °C	2 ³ °C	2 ² °C	2 ¹ °C	2 ⁰ °C
	LSB	2 ⁻¹ °C	2 ⁻² °C	2 ⁻³ °C	2 ⁻⁴ °C	0	0	0	0

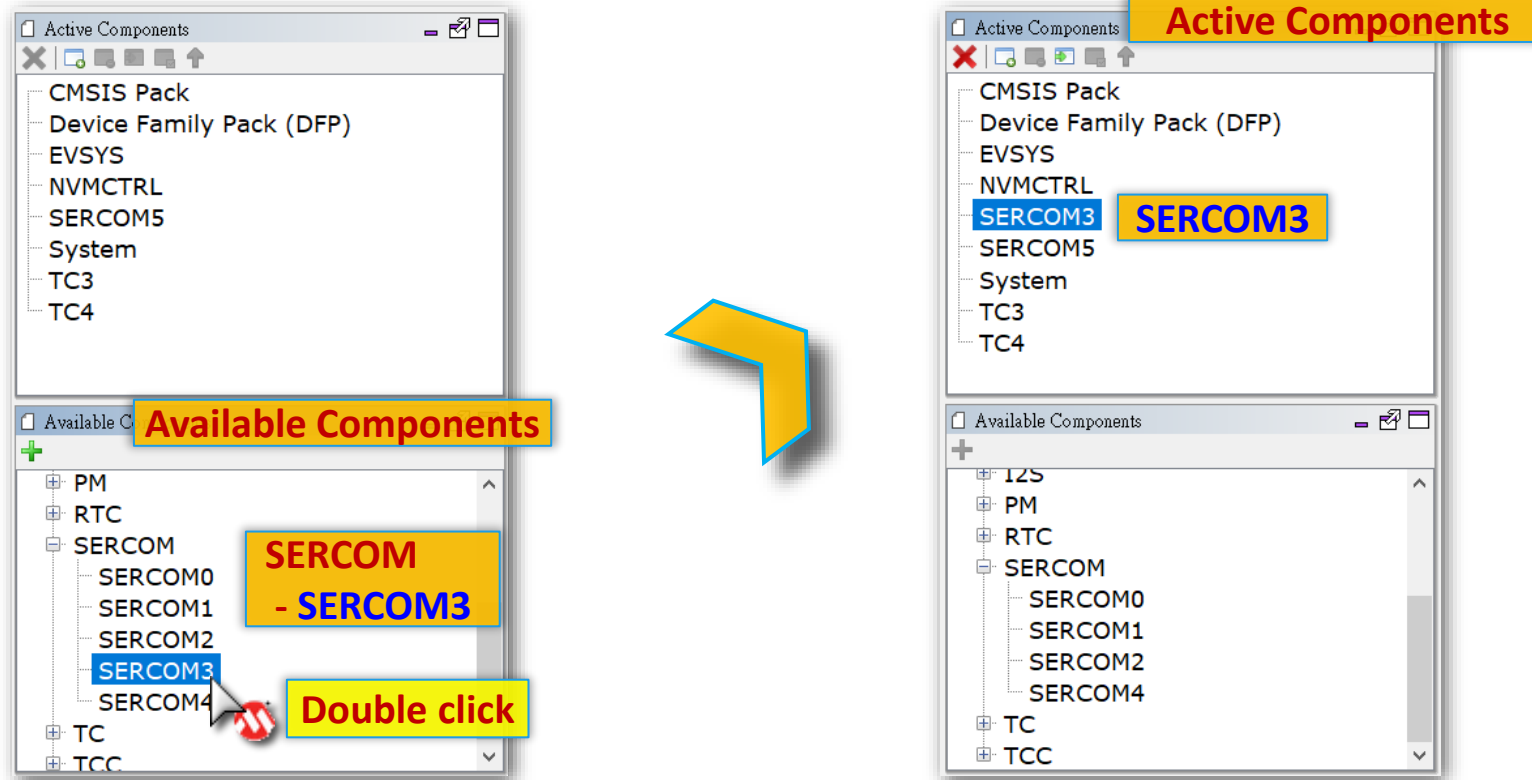
TA[0]
TA[1]



⚙️ Lab11-1 SERCOM – I2C Access Temperature Sensor

Step 1

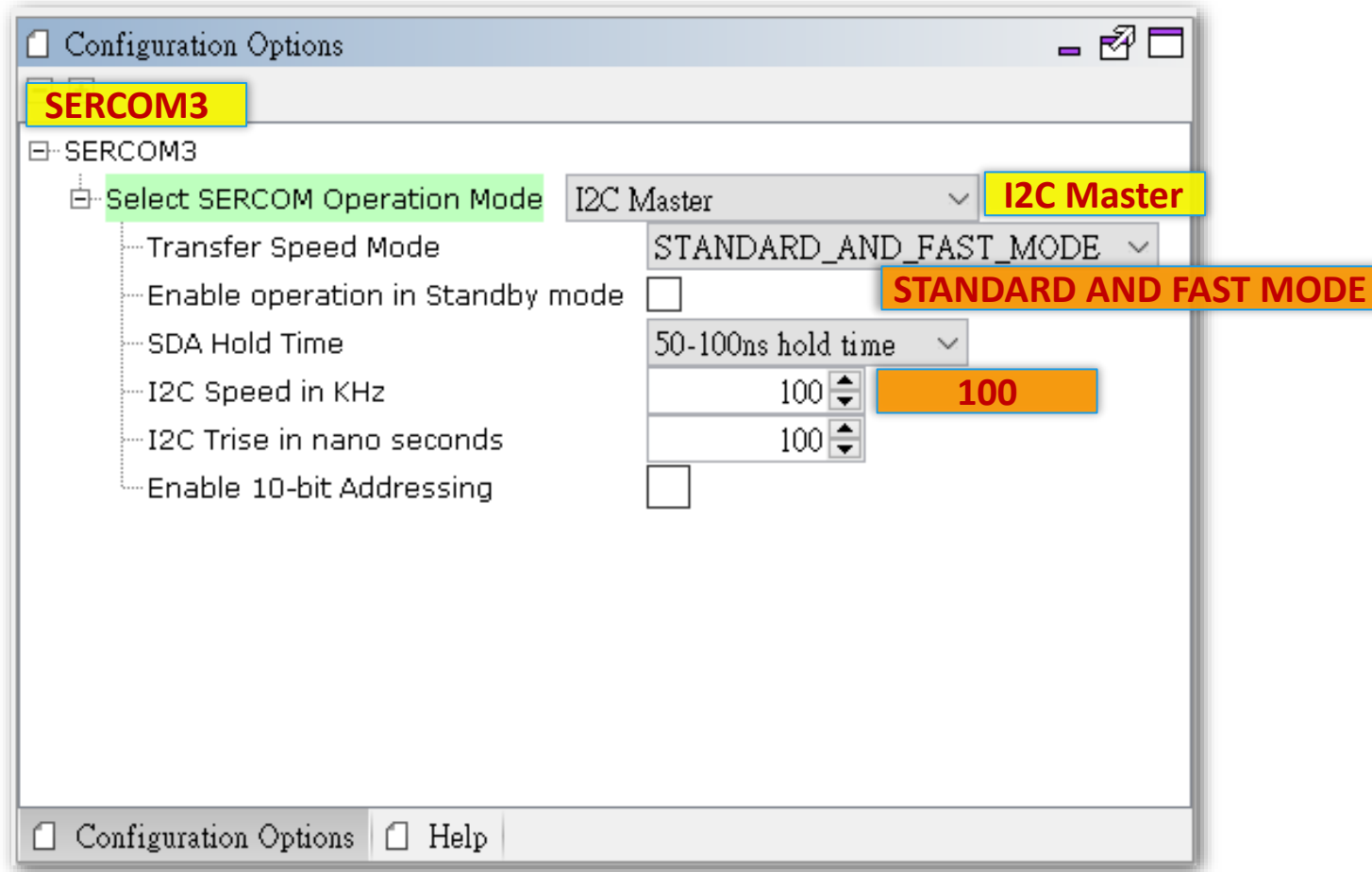
- Find **SERCOM** component in Available Components window.
- Double click **SERCOM3** to add to Active Components window.



✖ Lab11-1 SERCOM – I2C Access Temperature Sensor

Step 2

- Configure SERCOM3 to I2C Master mode.



✖ Lab11-1 SERCOM – I2C Access Temperature Sensor

Step 3

- Reassign **PA22** to **SERCOM3_PAD0** and **PA23** to **SERCOM3_PAD1**.

Pin Table

Package: TQFP48

Module	Function	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44
	SERCOM2_PAD3																											
SERCOM3	SERCOM3_PAD0																											
	SERCOM3_PAD1																											
	SERCOM3_PAD2																											
	SERCOM3_PAD3																											
	SERCOM4_PAD0																											
	SERCOM4_PAD1																											

Click to enable

SERCOM3_PAD0
SERCOM3_PAD1

SERCOM3

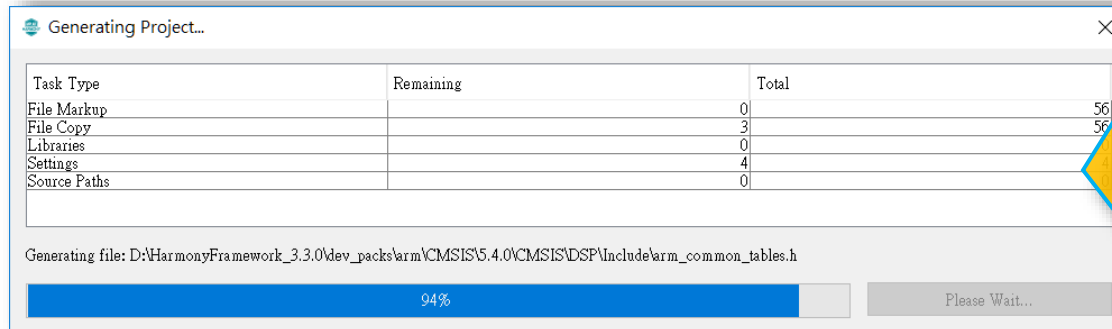
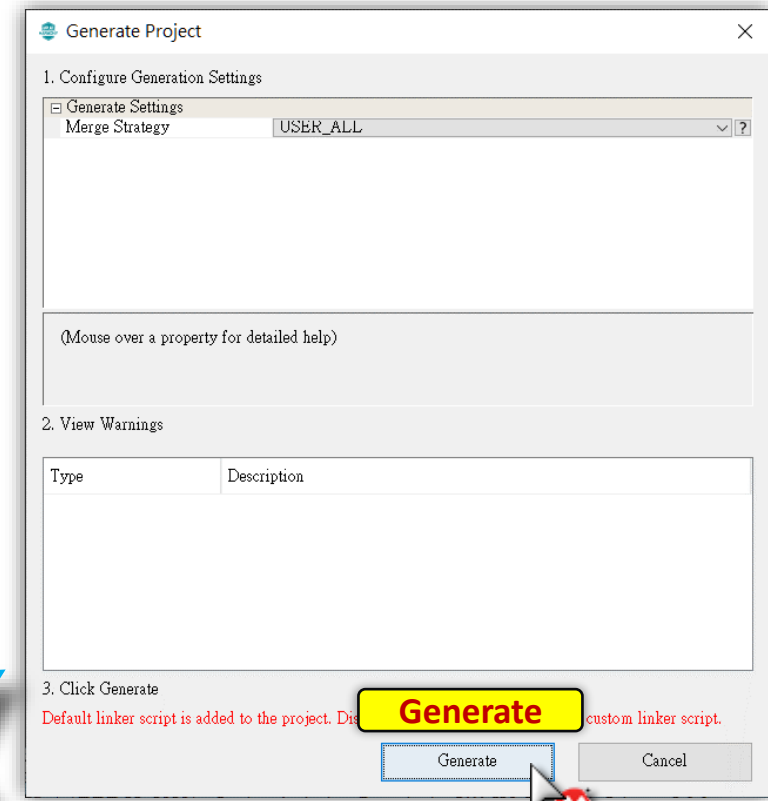
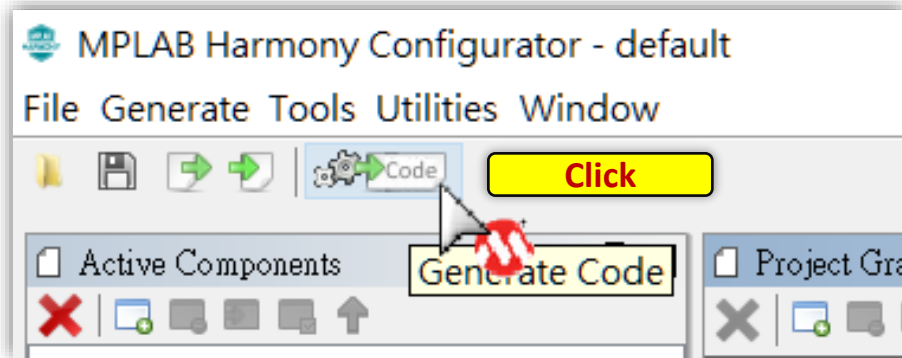
SERCOM3_PAD0
SERCOM3_PAD1

Green is enabled

⚙️ Lab11-1 SERCOM – I2C Access Temperature Sensor

Step 4

- Click  to Generate Code.



⚙️ Lab11-1 SERCOM – I2C Access Temperature Sensor

Step 5

```
// TODO 11-1.01
#define MCP9800_ADDRESS      0x4D
#define MCP9800_REG_TA      0x0
uint8_t MCP9800_Data[2];

void USART5_Transmit( uintptr_t context ) { ... }
...
int main (void)
{ ...
    // TODO 11-1.02
    MCP9800_Data[0] = MCP9800_REG_TA;
    SERCOM3_I2C_WriteRead( MCP9800_ADDRESS, MCP9800_Data, 1, MCP9800_Data, 2 );

    while ( true )
    { ...
        if (TC3_HasExpired)
        { ...
            // TODO 11-1.03
            SERCOM3_I2C_Read( MCP9800_ADDRESS, MCP9800_Data, 2 );
            myprintf( "Temperature : %2.1f'C\r\n",
                    (float)((MCP9800_Data[0]<<4)|(MCP9800_Data[1]>>4))/16.0f);
        }
    }
}
```

$$T_A = \text{Code} \times 2^{-4}$$

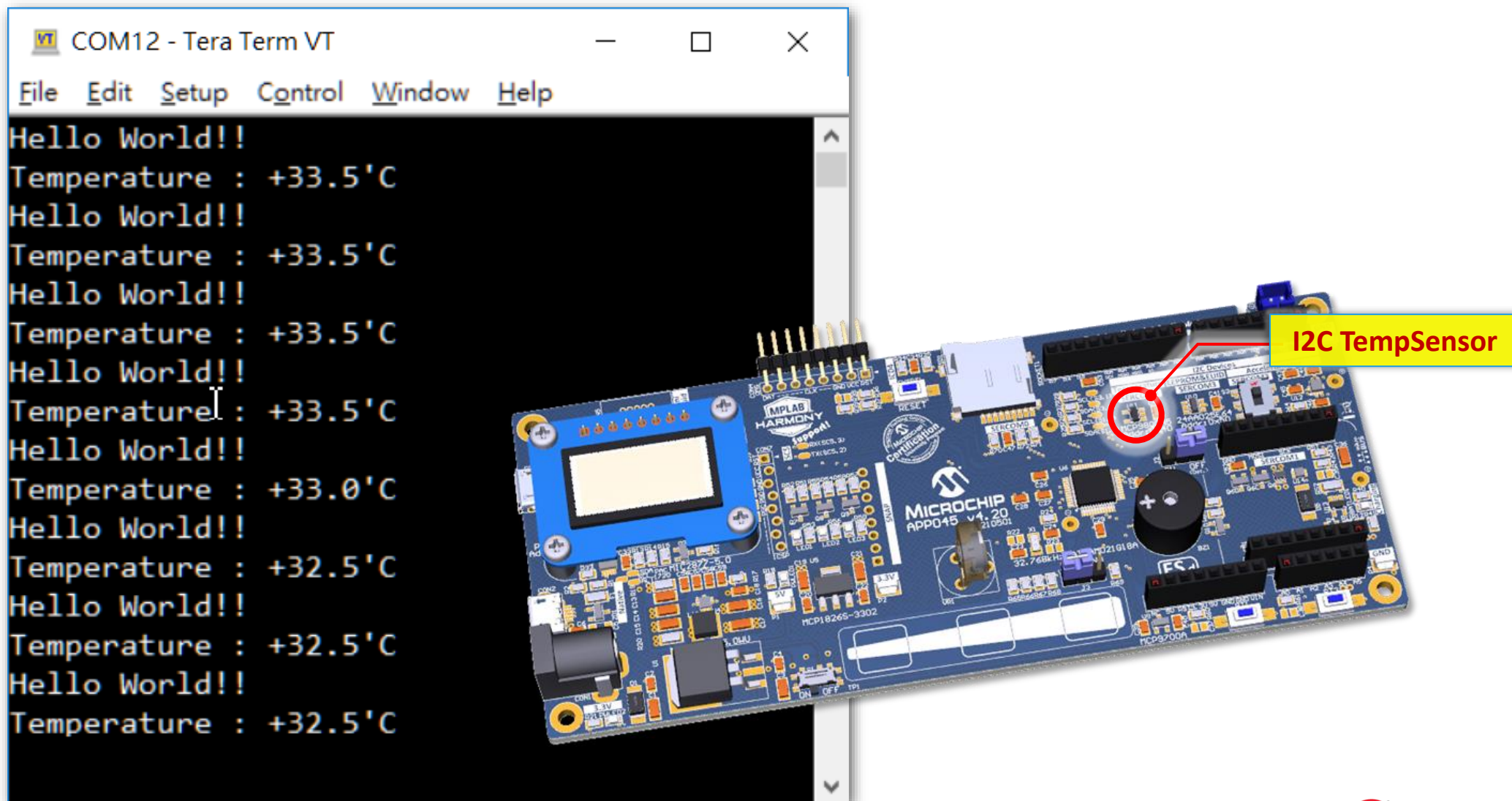
T_A = Ambient Temperature (°C)

Code = MCP9800 output in decimal

Result = (TA[0]<<4|TA[1]>>4)/16

⚙️ Lab11-1 SERCOM – I2C Access Temperature Sensor

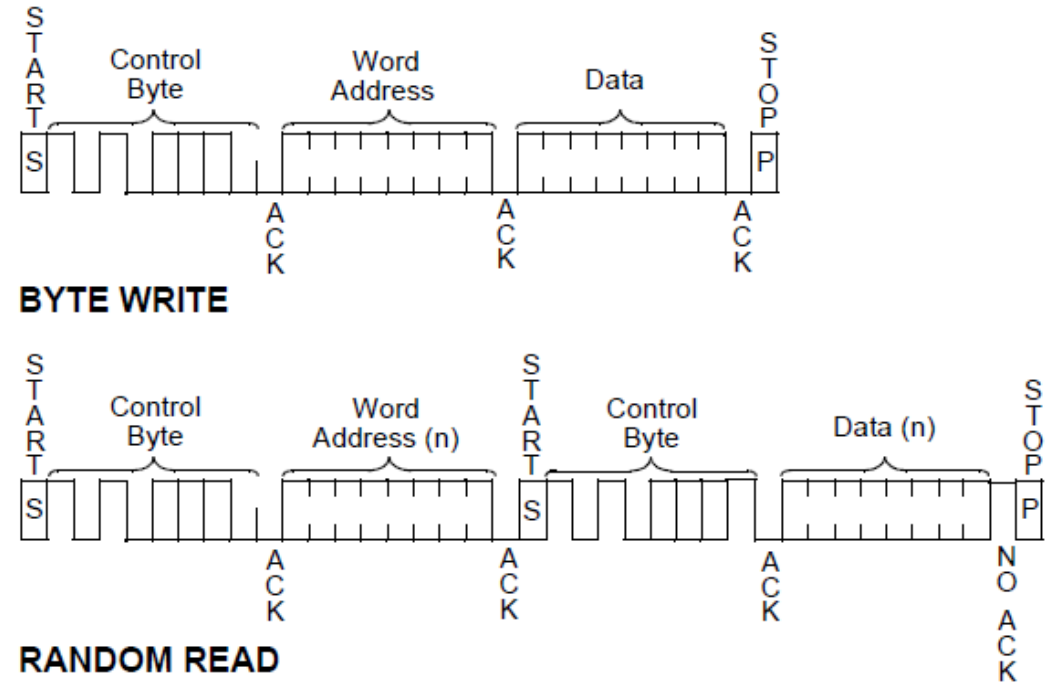
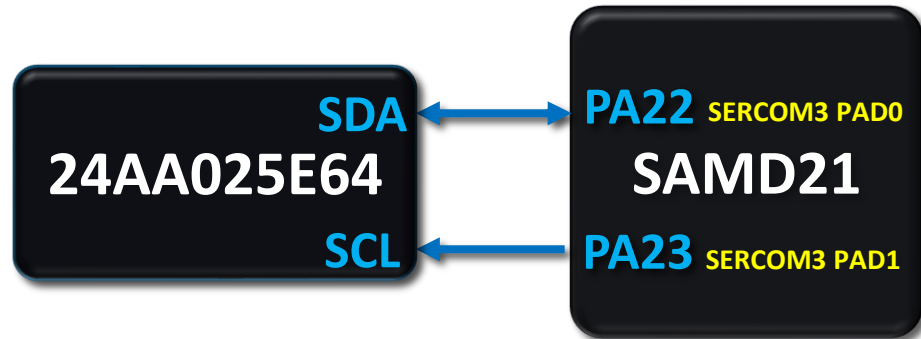
Result



Lab11-2 SERCOM – I2C Access Serial EEPROM

- Use **SERCOM3-I2C** master function to project to communication the on-board **Serial EEPROM 24AA025E64**.
- Write one byte “**0x24**” to target word address **0x00** after system initialize.
- Random Read it back and output to UART console.
- **24AA025E64** configuration:
 - **SDA** connect to **PA22** (**SERCOM3 PAD0**)
 - **SCL** connect to **PA23** (**SERCOM3 PAD1**)
 - Slave address is **0x50**.
- **Let's go !**

24AA025E64 access command and connection



Lab11-2 SERCOM – I2C Access Serial EEPROM

Step 1

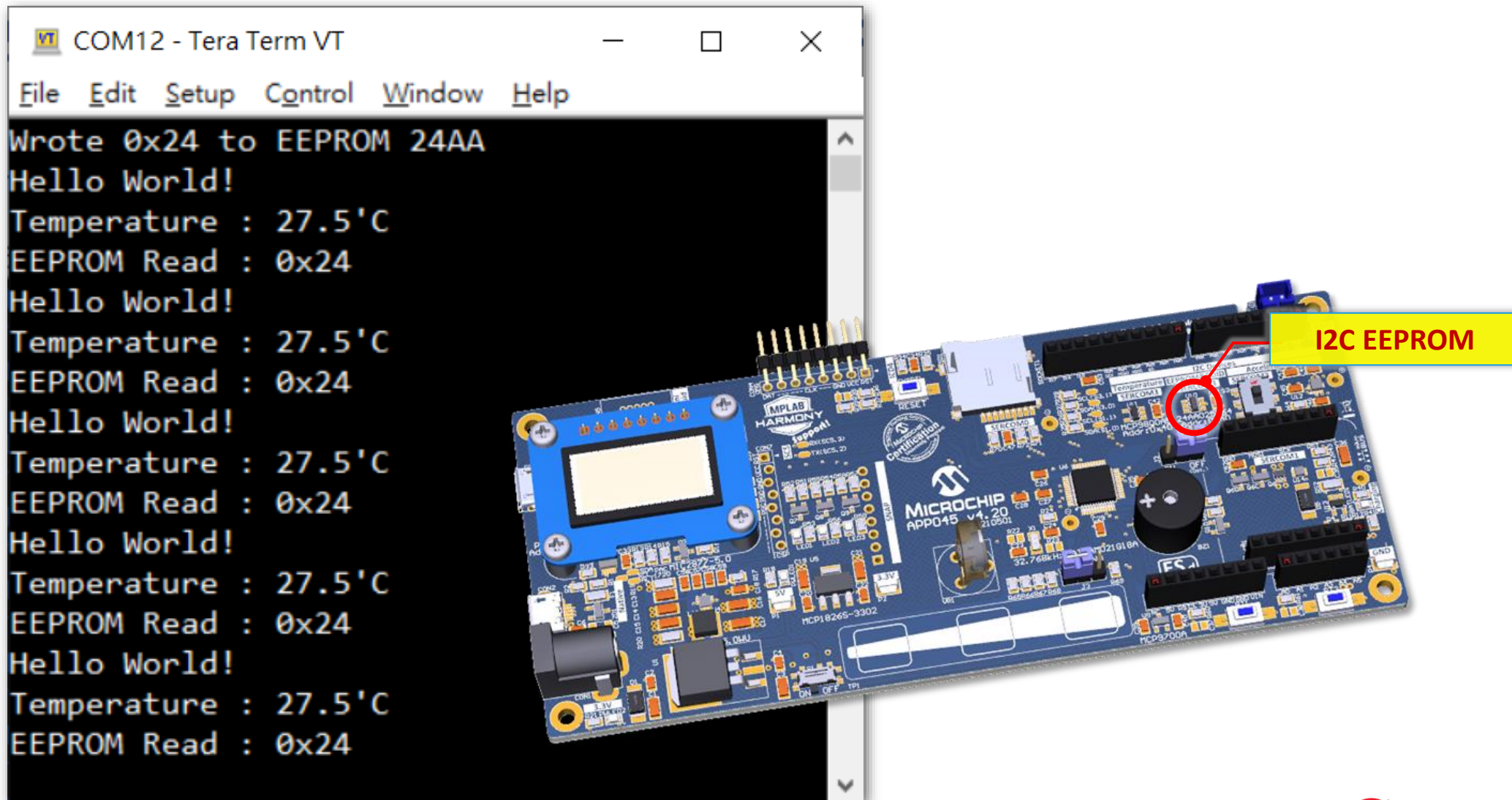
```
// TODO 11-2.01
#define MCP24AA_ADDRESS 0x50
uint8_t MCP24AA_WriteData[2];
uint8_t MCP24AA_ReadData[1];
...
int main (void)
{
    ...
    // TODO 11-2.02
    MCP24AA_WriteData[0] = 0x00;
    MCP24AA_WriteData[1] = 0x24;
    SERCOM3_I2C_Write( MCP24AA_ADDRESS, MCP24AA_WriteData, 2 );
    myprintf( "Wrote 0x24 to EEPROM 24AA\r\n");

    while ( true )
    {
        ...
        if (TC3_HasExpired)
        {
            ...
            // TODO 11-2.03
            MCP24AA_WriteData[0] = 0x00;
            SERCOM3_I2C_WriteRead( MCP24AA_ADDRESS, MCP24AA_WriteData, 1,
                                   MCP24AA_ReadData, 1 );

            while( SERCOM3_I2C_IsBusy() );
            myprintf( "EEPROM Read : 0x%X\r\n", MCP24AA_ReadData[0] );
        }
    }
}
```

⚙️ Lab11-2 SERCOM – I2C Access Serial EEPROM

Result

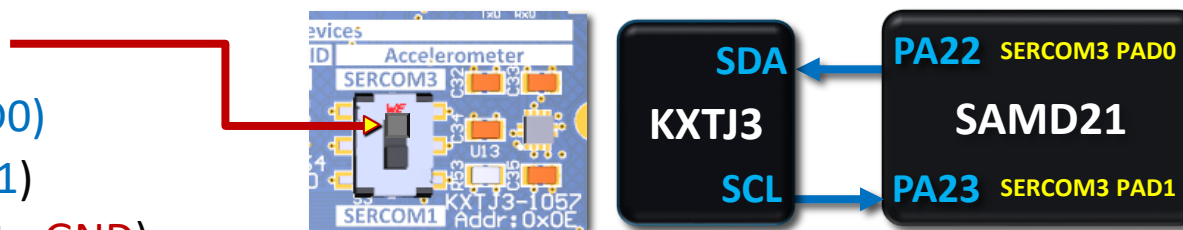


⚙️ Lab11-3 SERCOM – I2C Access G-Sensor

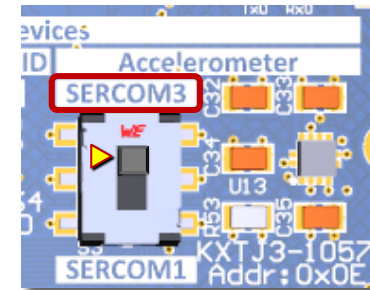
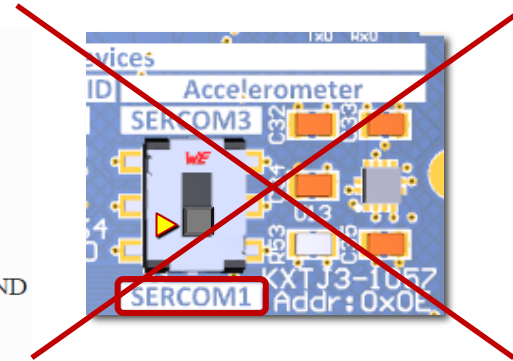
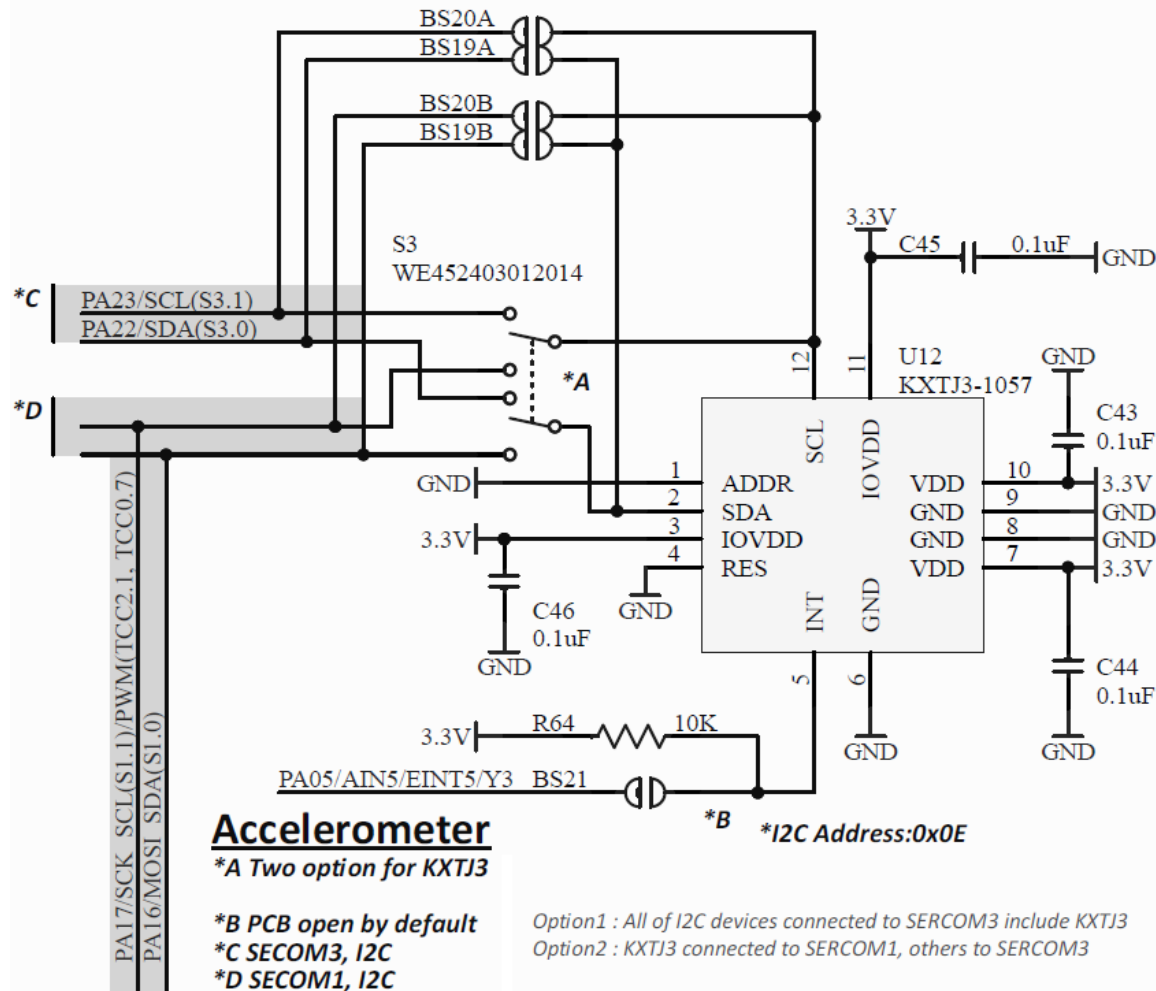
- Use **SERCOM3-I2C** master function to project to communication the on-board **Accelerometer KXTJ3**.
- **KXTJ3 initialize:**
 - Write Software Reset “0x80” to Control Register 2 “0x1D” and wait 10ms for stable.
 - Write follow Read the **WHO AM I Register** “0x0F” to check the ID is equal to “0x35” or not.
 - Write Initial Command “0xF0” to Control Register 1 “0x1B” for op-mode and resolution.
- **KXTJ3 Interrupt Source Register read by Callback:**
 - Write follow Read the **Interrupt Source Register** “0x16”.
- **KXTJ3 Data Ready Interrupt check in Callback function:**
 - Check “Bit 4 DRDY INT” of **Interrupt Source Register** to make sure data prepared for read.
- **KXTJ3 Axis Data Read:**
 - Write follow Read the **Axis Output Register** in below sequence to get X, Y, Z data.

XOUT_L	XOUT_H	YOUT_L	YOUT_H	ZOUT_L	ZOUT_H
0x06	0x07	0x08	0x09	0x0A	0x0B

- **KXTJ3 Axis Data Output:**
 - Output X, Y, Z Axis data as format “G-Sensor = (+XXXX, +YYYY, +ZZZZ)”
- **KXTJ3 configuration:**
 - **Switch the S3 to SERCOM3 on EVB**
 - **SDA** connect to **PA22** (SERCOM3 PAD0)
 - **SCL** connect to **PA23** (SERCOM3 PAD1)
 - Slave address is **0x0E**. (ADDR pin tie to GND)



KXTJ3 could connect to SERCOM3 or SERCOM1



Switch the **S3** to **SERCOM3** on EVB

Lab11-3 SERCOM – I2C Access G-Sensor

Step 1

```
// TODO 11-3.01
#define KXTJ3_ADDRESS    0x0E
uint8_t KXTJ3_WriteData[2];
uint8_t KXTJ3_ReadData[1];
volatile uint8_t I2C3_IsDataReady = 0;
uint8_t AxisOutReg[6] = { 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B };
uint8_t AxisOutByte[6];
```

XOUT_L	XOUT_H	YOUT_L	YOUT_H	ZOUT_L	ZOUT_H
0x06	0x07	0x08	0x09	0x0A	0x0B

```
void USART5_Transmit( uintptr_t context ) { ... }
...
// TODO 11-3.02
void I2C3_TransferCallback( uintptr_t contextHandle )
{
    if( KXTJ3_ReadData[0] & (1<<4) )
    {
        I2C3_IsDataReady = 1;
    }
}

void myprintf(const char *format, ...) { ... }
...
int main (void)
{
    ...
    while ( true )
    {
        ...
    }
}
```

Lab11-3 SERCOM – I2C Access G-Sensor

Step 2

```
int main (void)
{
    ...
    // TODO 11-3.03
    KXTJ3_WriteData[0] = 0x1D;
    KXTJ3_WriteData[1] = 0x80;
    SERCOM3_I2C_Write( KXTJ3_ADDRESS, KXTJ3_WriteData, 2 );
    myprintf( "G-Sensor Software Reset Completed\r\n" );
    KXTJ3_WriteData[0] = 0x0F;
    SERCOM3_I2C_WriteRead( KXTJ3_ADDRESS, KXTJ3_WriteData, 1, KXTJ3_ReadData, 1 );
    while( SERCOM3_I2C_IsBusy() );
    myprintf( "G-Sensor WHO AM I : 0x%X\r\n", KXTJ3_ReadData[0]);
    KXTJ3_WriteData[0] = 0x1B;
    KXTJ3_WriteData[1] = 0xF0;
    SERCOM3_I2C_Write( KXTJ3_ADDRESS, KXTJ3_WriteData, 2 );
    myprintf( "G-Sensor Initial Completed\r\n" );
    SERCOM3_I2C_CallbackRegister( I2C3_TransferCallback, 0 );

    while ( true )
    {
        ...
        if (TC3_HasExpired)
        {
            ...
            // TODO 11-3.04
            KXTJ3_WriteData[0] = 0x16;
            SERCOM3_I2C_WriteRead( KXTJ3_ADDRESS, KXTJ3_WriteData, 1, KXTJ3_ReadData, 1 );
        }
    }
}
```


⚙️ Lab11-3 SERCOM – I2C Access G-Sensor

Step 3

```
...
int main (void)
{
    ...
    while ( true )
    {
        ...
        if (TC3_HasExpired) { ... }

        if (TC4_HasExpired) { ... }

        // TODO 11-3.05
        if( I2C3_IsDataReady )
        {
            for( i=0 ; i<6 ; i++ )
            {
                SERCOM3_I2C_WriteRead( KXTJ3_ADDRESS, &AxisOutReg[i], 1,
                                         &AxisOutByte[i], 1 );

                while( SERCOM3_I2C_IsBusy() );
            }
            myprintf("G-Sensor = (%5d, %5d, %5d)\r\n",
                    (short)(AxisOutByte[1]<<8|AxisOutByte[0]),
                    (short)(AxisOutByte[3]<<8|AxisOutByte[2]),
                    (short)(AxisOutByte[5]<<8|AxisOutByte[4]));

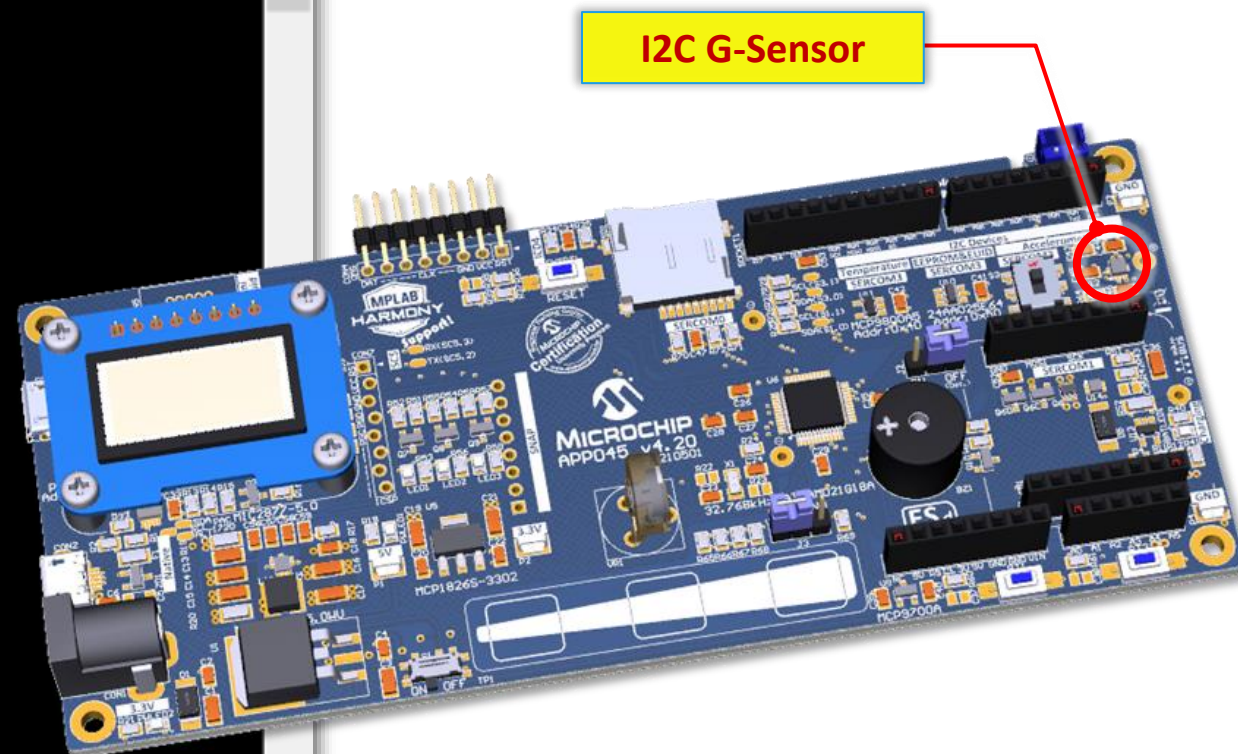
            I2C3_IsDataReady = 0;
        }
        ...
    }
}
```

XOUT_L	XOUT_H	YOUT_L	YOUT_H	ZOUT_L	ZOUT_H
0x06	0x07	0x08	0x09	0x0A	0x0B

⚙️ Lab11-3 SERCOM – I2C Access G-Sensor

Result

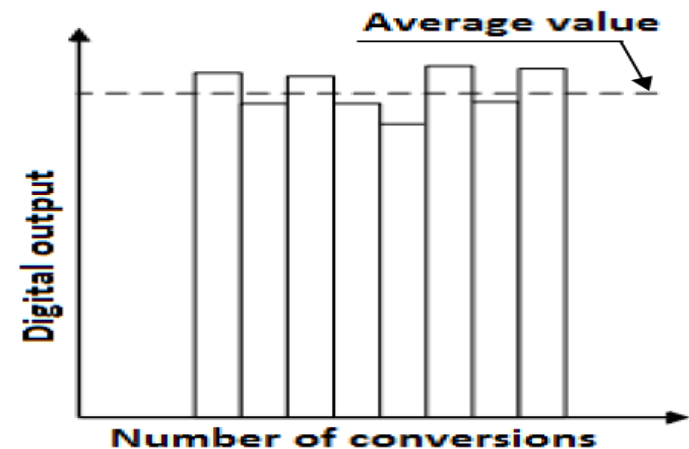
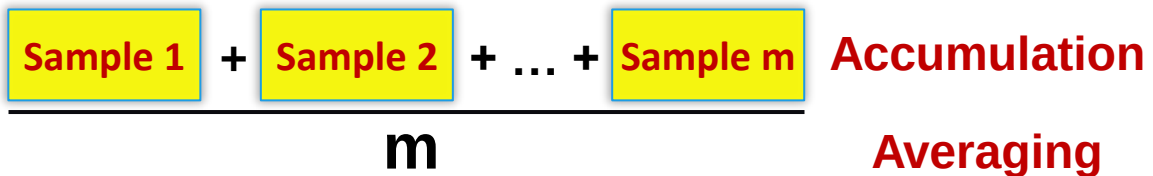
```
COM12 - Tera Term VT
File Edit Setup Control Window Help
Hello World!
Temperature : 27.5'C
EEPROM Read : 0x24
G-Sensor = (    0,    -32,  4224)
Hello World!
Temperature : 27.5'C
EEPROM Read : 0x24
G-Sensor = (    0,    -32,  4224)
Hello World!
Temperature : 27.5'C
EEPROM Read : 0x24
G-Sensor = (    0,   -48,  4224)
Hello World!
Temperature : 27.5'C
EEPROM Read : 0x24
G-Sensor = (    0,    -32,  4224)
```



ADC Accumulation and Averaging

ADC Accumulation and Averaging

- The result from multiple consecutive of ADC conversions can be accumulated. When accumulating **more than 16 samples**, the result will be too large to match the 16-bit RESULT register size. To avoid overflow, the result is **right shifted automatically** to fit within the available register size.
- Averaging is a feature that increases stability at the cost of a **reduced sampling rate**. This feature is suitable when operating in noisy conditions. Averaging is done by **accumulating m samples**, and **dividing the result by m**.



ADC Accumulation and Averaging

Example :

Number of Accumulated Samples : $\sum 128$ (2^7 samples)
 Intermediate Result precision : 19 bits (12bits + 7 = 19 bits overflow)
 Number of Automatic Right Shift : 3 bits (19bits >> 3 = 16bits)
 Adjust Division Factor : /16 (2^4)
 Final Result Precision : 12 bits (16bit >> 4 = 12 bits)

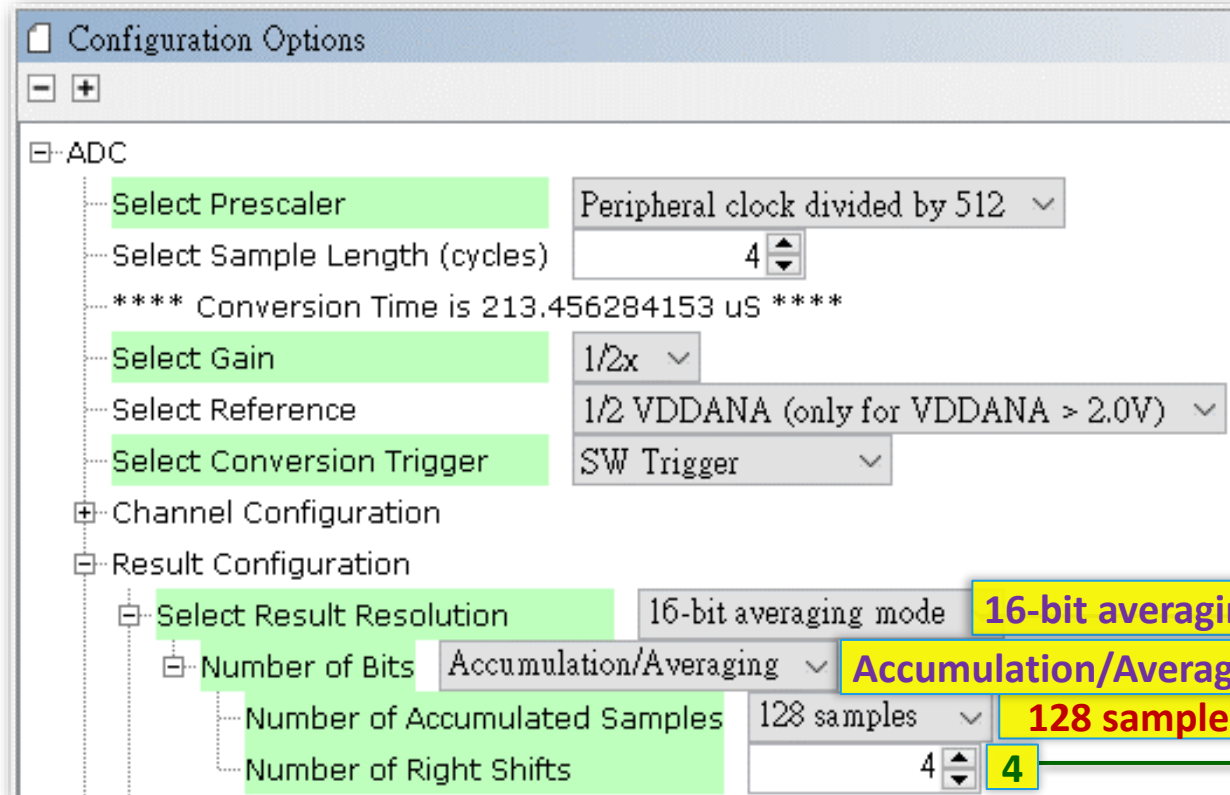
Number of Accumulated Samples	AVGCTRL. SAMPLENUM	Intermediate Result Precision	Number of Automatic Right Shifts	Automatic Division Factor	AVGCTRL. ADJRES	Adjust Division Factor	Total Number of Right Shifts	Final Result Precision
1	0x0	12 bits	0	0	0x0	1		12 bits
2	0x1	13 bits	0	0	0x1	2	1	12 bits
4	0x2	14 bits	0	0	0x2	4	2	12 bits
8	0x3	15 bits	0	0	0x3	8	3	12 bits
16	0x4	16 bits	(19>>3)	0	0x4	16	(3+4)	12 bits
32	0x5	17 bits	>>3	2	0x5	16	7	12 bits
64	2^7	18 bits		4	0x6	16		12 bits
128	0x7	19 bits	3	8	0x7	16	7	12 bits
$\sum 128$	0x8	19 bits	4	8	0x8	16	8	12 bits
(2^7)	0x9	(12+7)	5	(2 ³)	0x9	(2 ⁴)	9	(16>>4)
	0xA		6		0xA		10	
Reserved	0xB – 0xF							

RESSEL[1:0]	Name	Description
0x0	12BIT	12-bit result
0x1	16BIT	For averaging mode output
0x2	10BIT	10-bit result
0x3	8BIT	8-bit result

ADC Accumulation and Averaging

Example :

Number of Accumulated Samples : $\Sigma 128$ (2^7 samples) ←
Intermediate Result precision : 19 bits (12bits + 7 = 19 bits overflow)
Number of Automatic Right Shift : 3 bits (19bits >> 3 = 16bits) ←
Adjust Division Factor : /16 (2^4) ←
Final Result Precision : 12 bits (16bit >> 4 = 12 bits)



RESSEL[1:0]	Name	Description
0x0	12BIT	12-bit result
0x1	16BIT	For averaging mode output
0x2	10BIT	10-bit result
0x3	8BIT	8-bit result

Lab13-1 ADC Accumulation and Averaging

- Please continue from [SAM2001 Lab13 \(ADC Pin Scan\)](#)
- Accumulated 128 Samples and output 12 bits resolution.
- Modify ADC conversion interval to 10ms.
- Modify output message to keep update at the same line.
- **Let's go!**

✖ Lab13-1 ADC Accumulation and Averaging

Step 1

- Configure ADC configuration options as below.

The screenshot shows the 'Configuration Options*' dialog box for the ADC. The settings are as follows:

- ADC**
 - Select Prescaler: Peripheral clock divided by 512
 - Select Sample Length (half ADC clock cycles): 4
 - **** Conversion Time is 128.073770492 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 AIN2 Pin
 - Select Negative Input: Internal Ground
 - Number of inputs to scan: 1
 - Scan start offset: 0
- Result Configuration**
 - Select Result Resolution: 16-bit averaging mode
 - Number of Bits: Accumulation/Averaging
 - Number of Accumulated Samples: 128 samples
 - Number of Right Shifts: 4
 - Left Aligned Result: ☐
 - Enable Result Ready Interrupt: ☒
 - Enable Result Ready Event Out: ☐

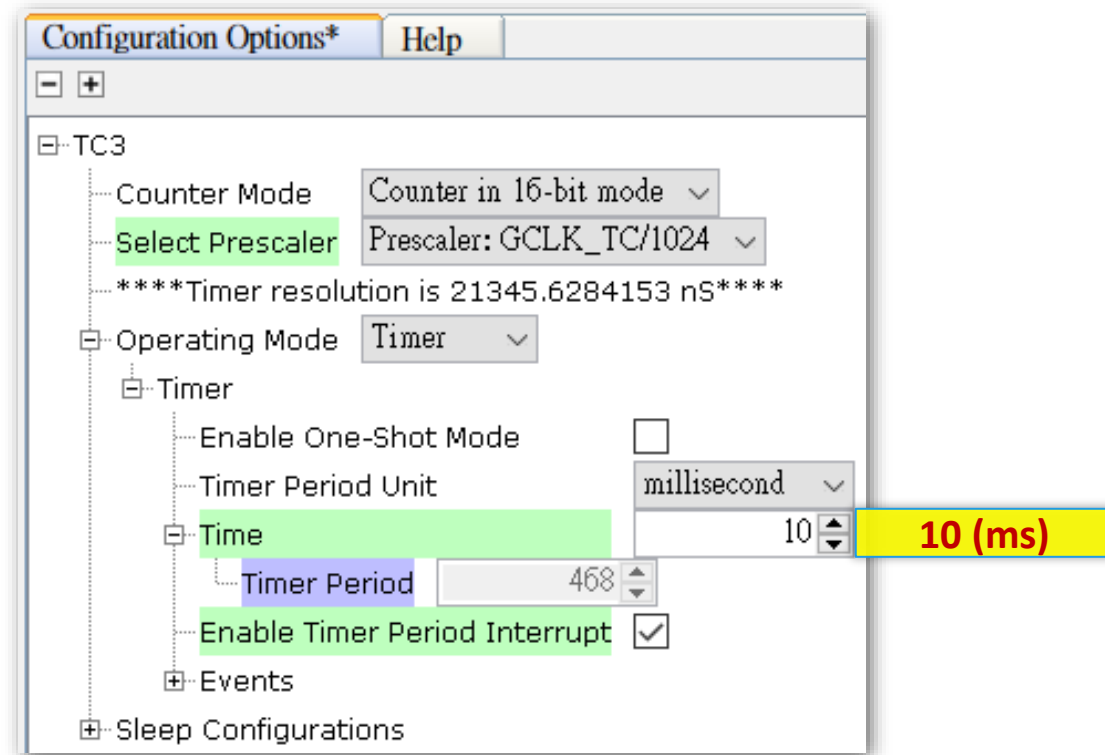
Yellow callouts highlight the following settings:

- 16-bit averaging mode
- Accumulation/Averaging
- 128 samples
- 4

✖ Lab13-1 ADC Accumulation and Averaging

Step 2

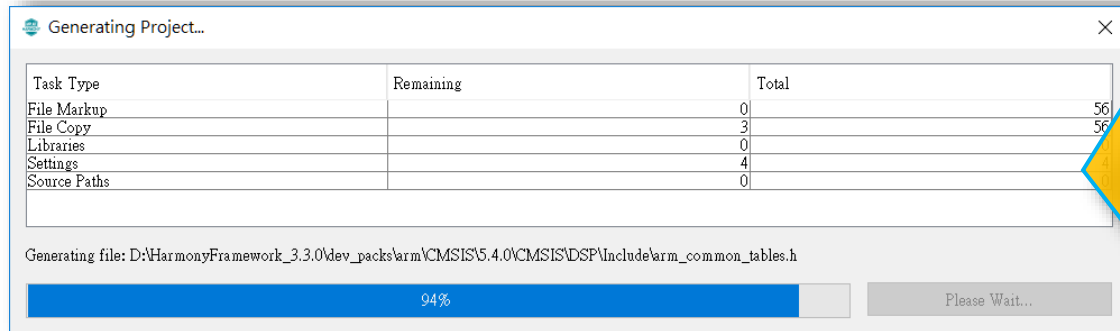
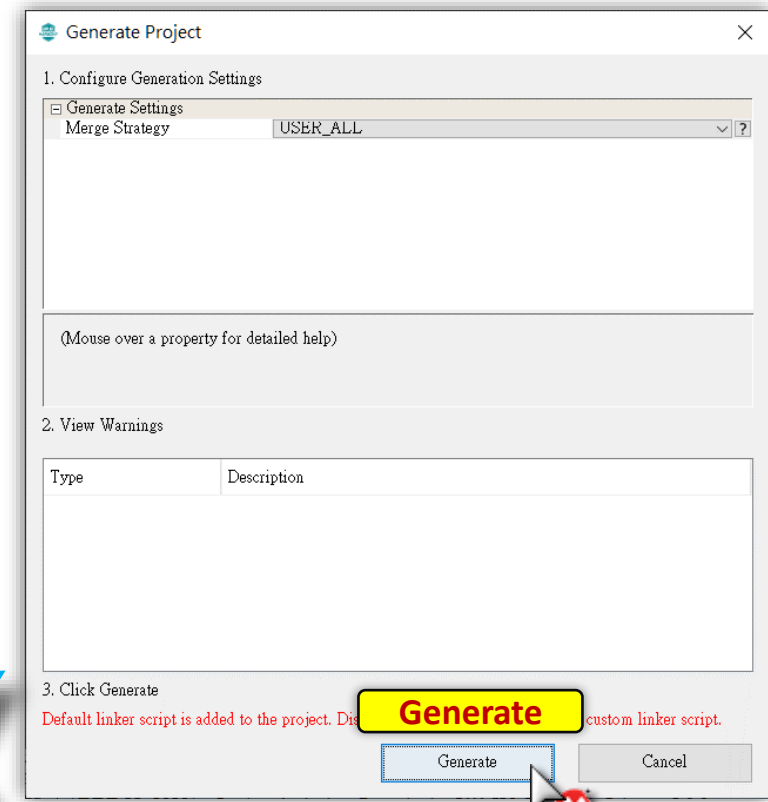
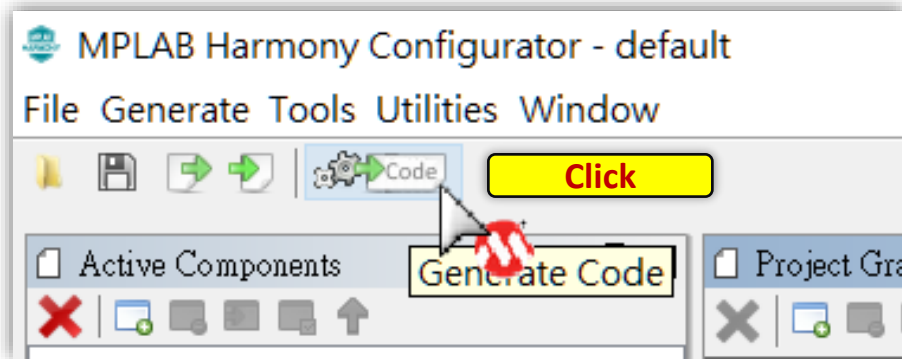
- Modify TC3 to set 10ms period interval for speed up ADC conversion rate.



⚙️ Lab13-1 ADC Accumulation and Averaging

Step 3

- Click  to Generate Code.



Lab13-1 ADC Accumulation and Averaging

Step 4

```
uint16_t ADC_Result[2];
volatile uint8_t ADC_IsCompleted = 0;
volatile int8_t ADC_ChannelIdx = 0;
...
void ADC_Complete( ADC_STATUS status, uintptr_t context ) { ... }
...
int main (void)
{
    ...
    while ( true )
    {
        ...
        if (TC3_HasExpired)
        {
            ...
            ADC_Convers
        }
        ...
        if( ADC_IsCompleted )
        {
            ADC_IsCompleted = 0;

            myprintf("\033[2;1H");
            myprintf( "VR1 Value : %4d", ADC_Result[0] );
            myprintf("\033[3;1H");
            myprintf( "Temperature Value : %4d", ADC_Result[1] );

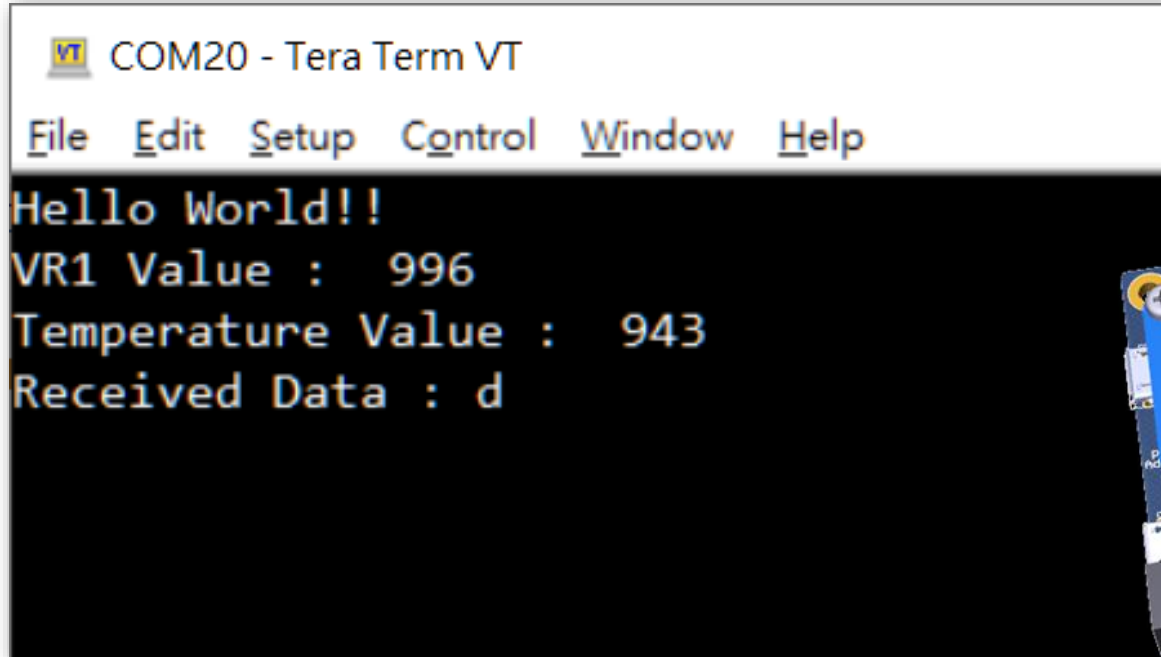
        }
    }
}
```

Nothing to modify in main.c

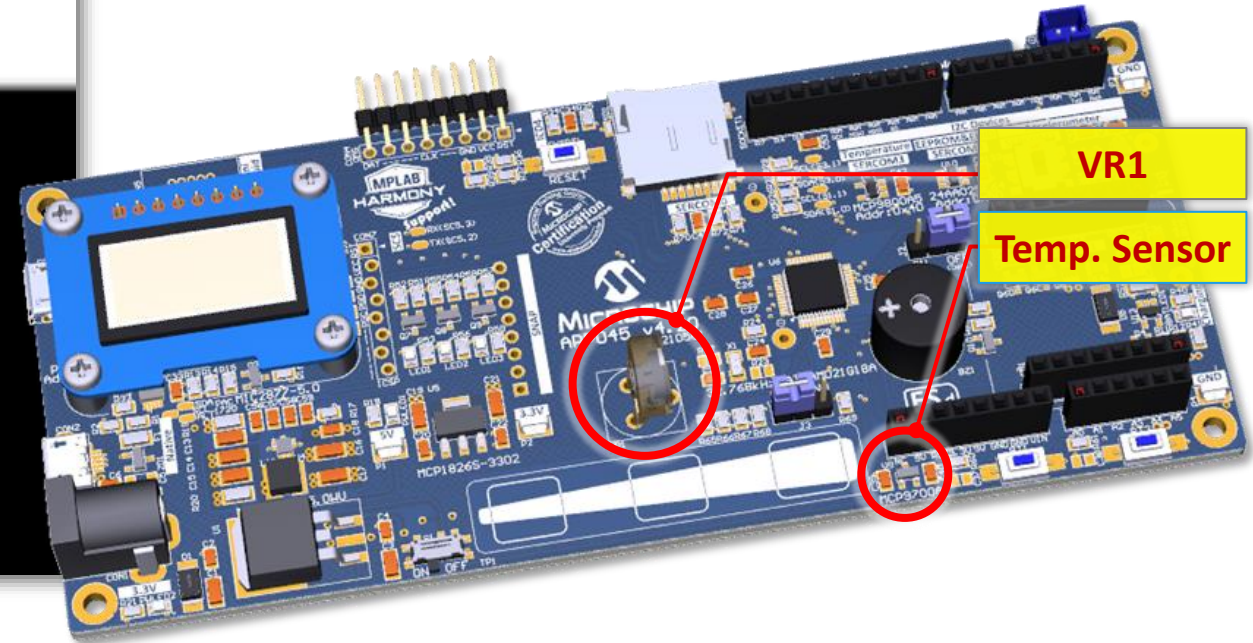
⚙️ Lab13-1 ADC Accumulation and Averaging

Result

- VR1 / Temperature Sensor ADC value will be averaged and more stable and smoothly.



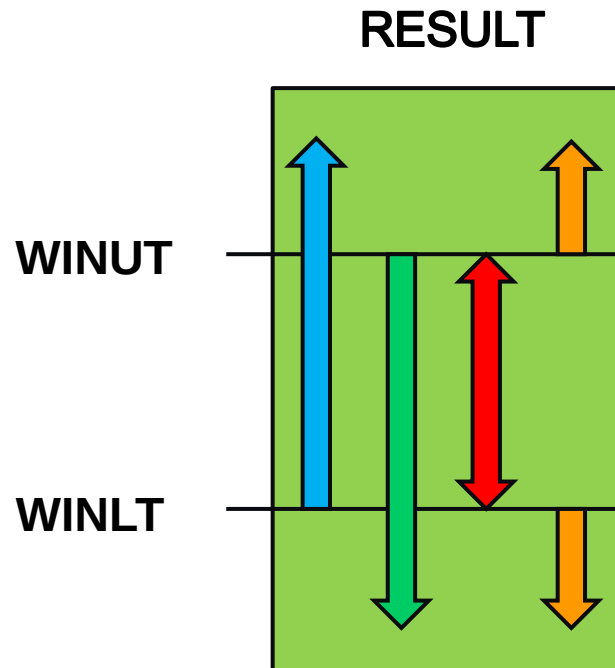
```
COM20 - Tera Term VT
File Edit Setup Control Window Help
Hello World!!
VR1 Value : 996
Temperature Value : 943
Received Data : d
```



ADC Window Monitor

ADC Window Monitor

- The window monitor feature allows the conversion result in the RESULT register to be compared to predefined threshold values.
- There have four window mode could be use for monitor.



WINMODE[2:0]	Name	Description
0x0	DISABLE	No window mode (default)
0x1	MODE1	Mode 1: $RESULT > WINLT$
0x2	MODE2	Mode 2: $RESULT < WINUT$
0x3	MODE3	Mode 3: $WINLT < RESULT < WINUT$
0x4	MODE4	Mode 4: $!(WINLT < RESULT < WINUT)$
0x5-0x7		Reserved

ADC Window Monitor

Example (Use Configuration Options)

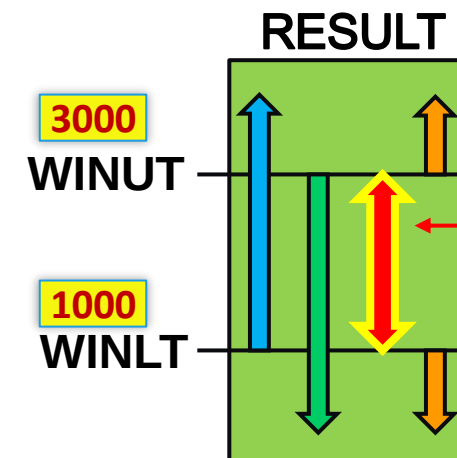
- Window Mode Mode3 with upper threshold 3000 and lower threshold 1000.

Configuration Options

ADC

- Select Prescaler: Peripheral clock divided by 512
- Select Sample Length (cycles): 4
- **** Conversion Time is 128.073770492 μ s ****
- Select Gain: 1/2x
- Select Reference: 1/2 VDDANA (only for VDDANA > 2.0V)
- Select Conversion Trigger: SW Trigger
- Channel Configuration
- Result Configuration
- Window Mode Configuration
 - Select Window Monitor Mode: Mode 3: WINLT < RESULT < WINUT
 - Window Upper Threshold: 3,000 **3000**
 - Window Lower Threshold: 1,000 **1000**
 - Enable Window Monitor Interrupt: ☒
 - Enable Window Monitor Event Out: ☐
- Sleep Mode Configuration

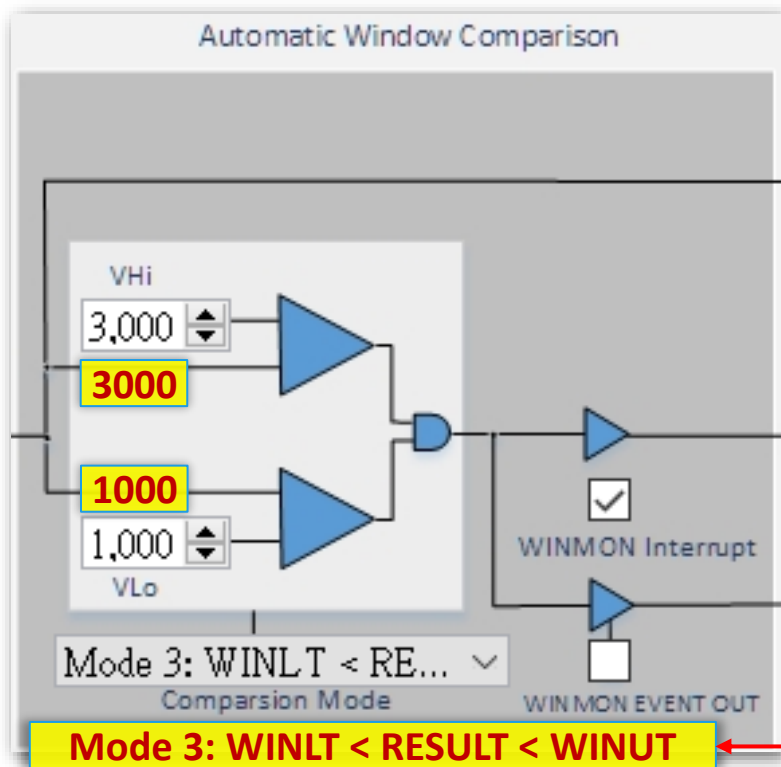
WINMODE[2:0]	Name	Description
0x0	DISABLE	No window mode (default)
0x1	MODE1	Mode 1: RESULT > WINLT
0x2	MODE2	Mode 2: RESULT < WINUT
0x3	MODE3	Mode 3: WINLT < RESULT < WINUT
0x4	MODE4	Mode 4: !(WINLT < RESULT < WINUT)
0x5-0x7		Reserved



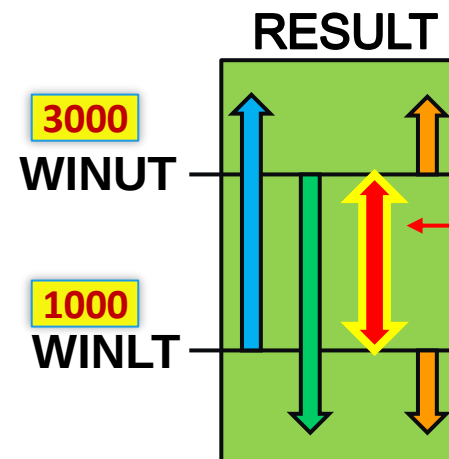
ADC Window Monitor

Example (Use Easy View)

- Window Mode Mode3 with upper threshold 3000 and lower threshold 1000.



WINMODE[2:0]	Name	Description
0x0	DISABLE	No window mode (default)
0x1	MODE1	Mode 1: $RESULT > WINLT$
0x2	MODE2	Mode 2: $RESULT < WINUT$
0x3	MODE3	Mode 3: $WINLT < RESULT < WINUT$
0x4	MODE4	Mode 4: $!(WINLT < RESULT < WINUT)$
0x5-0x7		Reserved



ADC Window Monitor

Interface Routines

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

Lab13-2 ADC Window Monitor

- Enable Window Monitor mode.
- Set upper threshold to 3000 and lower threshold to 1000.
- Output ADC window status while ADC value within upper and lower threshold.
(Mode 3).
- **Let's go!**

⚙️ Lab13-2 ADC Window Monitor

Step 1 (Use Configuration Options)

- Configure ADC configuration options as below.

Configuration Options

ADC

Select Prescaler: Peripheral clock divided by 512

Select Sample Length (cycles): 4

**** Conversion Time is 128.073770492 uS ****

Select Gain: 1/2x

Select Reference: 1/2 VDDANA (only for VDDANA > 2.0V)

Select Conversion Trigger: SW Trigger

Channel Configuration

Result Configuration

Window Mode Configuration

Select Window Monitor Mode: Mode 3: WINLT < RESULT < WINUT

Window Upper Threshold: 3,000

Window Lower Threshold: 1,000

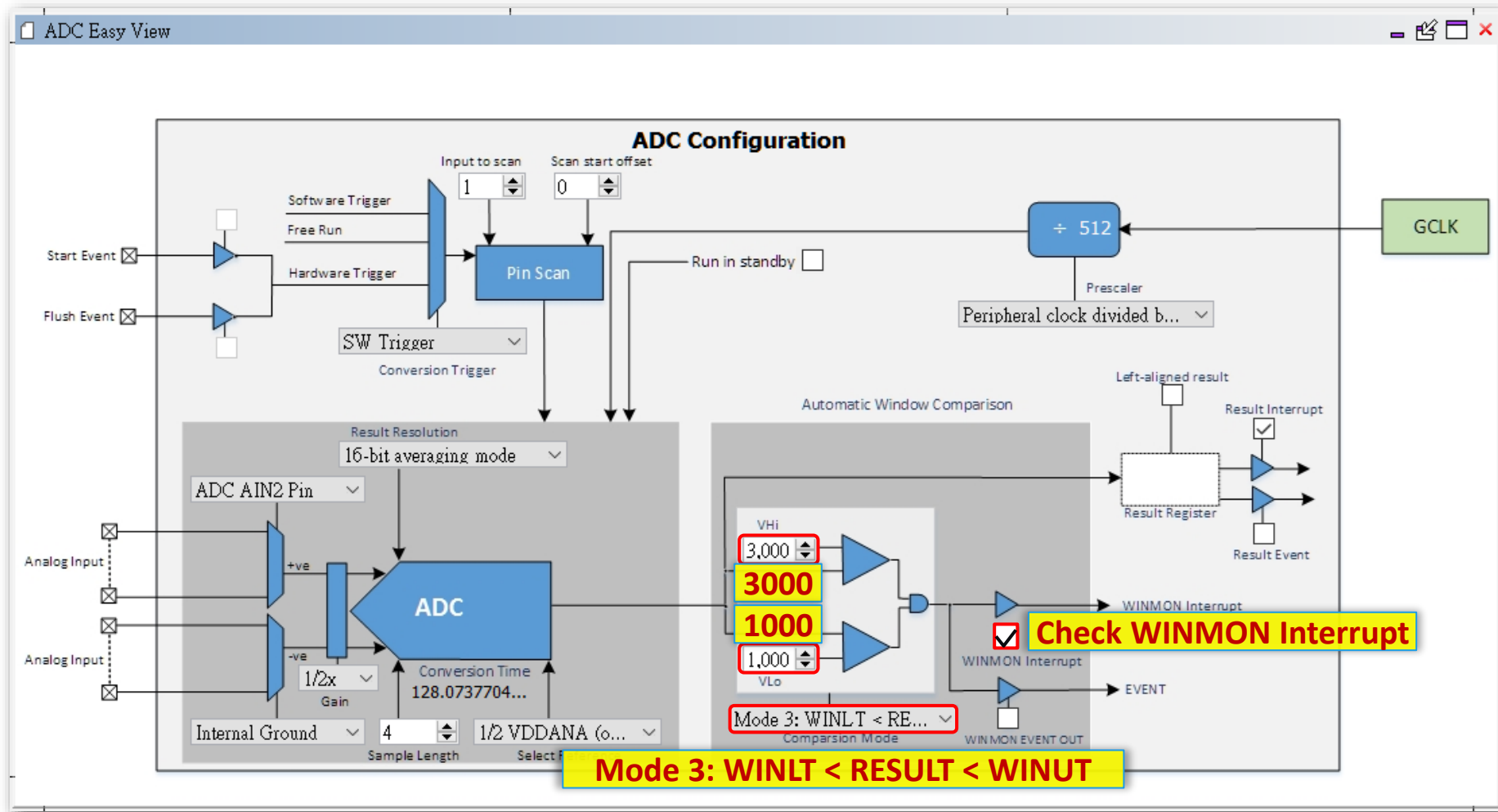
Enable Window Monitor Interrupt: ☒ Enable Window Monitor Interrupt

Enable Window Monitor Event Out: ☐

Sleep Mode Configuration

✖ Lab13-2 ADC Window Monitor

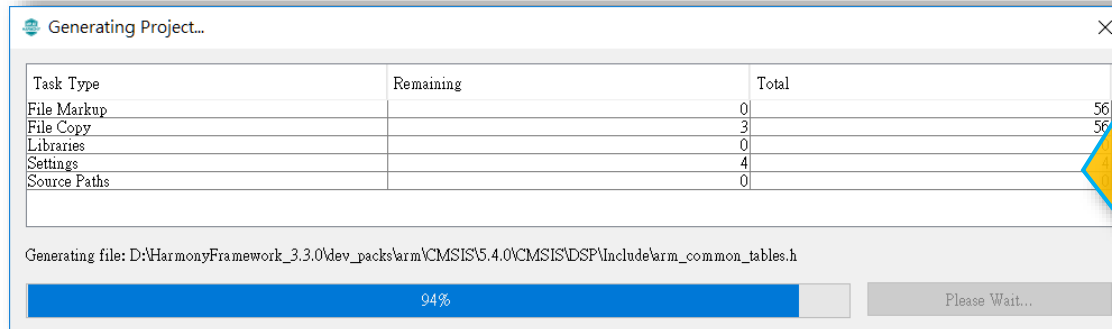
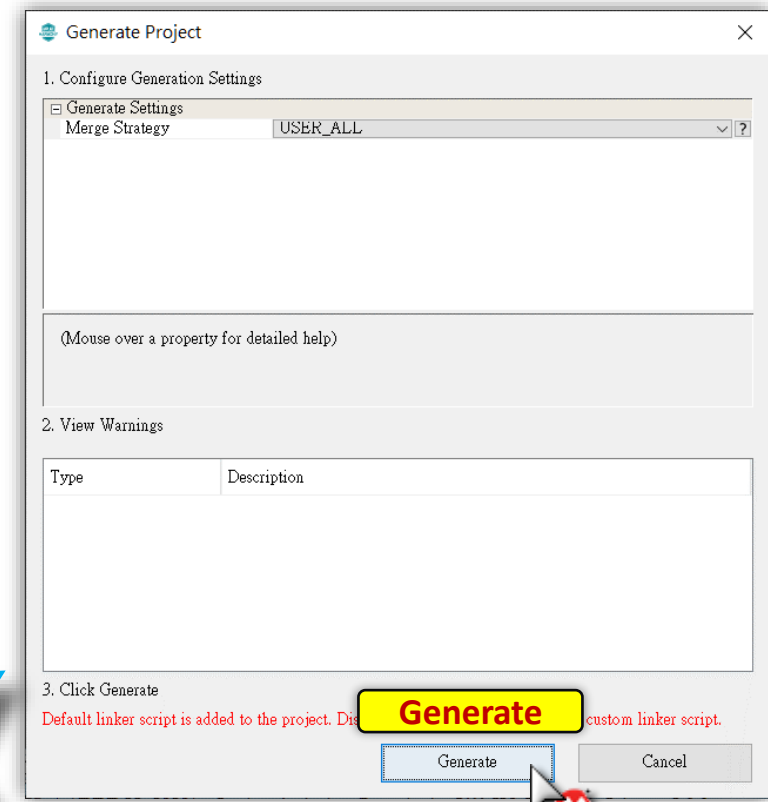
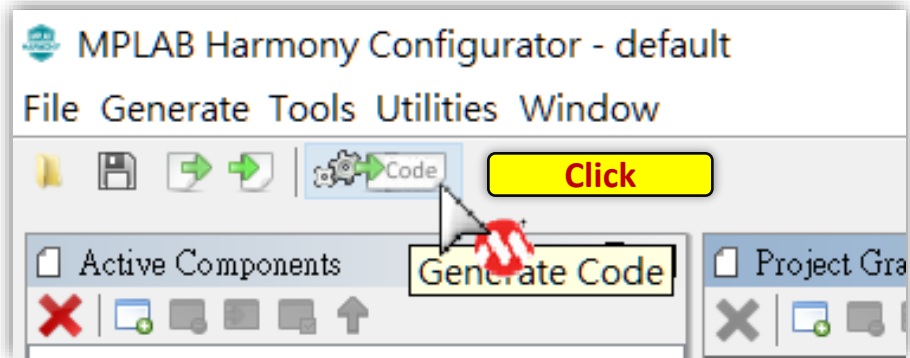
Step 1 (Use Easy View)



⚙️ Lab13-2 ADC Window Monitor

Step 2

- Click  to Generate Code.



Lab13-2 ADC Window Monitor

Step 3

```
// TODO 13-2.01
volatile uint16_t ADC_InWindow[2] = { 0, 0 };

void ADC_Complete( ADC_STATUS status, uintptr_t context )
{
    if( status & ADC_INTFLAG_RESRDY_Msk )
    {
        ADC_Result[ADC_ChannelIdx] = ADC_ConversionResultGet();

        // TODO 13-2.02
        ADC_InWindow[ADC_ChannelIdx] = 0;
        if( status & ADC_INTFLAG_WINMON_Msk )
        {
            ADC_InWindow[ADC_ChannelIdx] = 1;
        }

        ADC_ChannelIdx = ( ADC_ChannelIdx + 1 ) % 2;
        if( ADC_ChannelIdx == 0 )
        {
            ADC_IsCompleted = 1;
        }
    }
}

void USART5_Transmit( uintptr_t context ) { ... }
```

Lab13-2 ADC Window Monitor

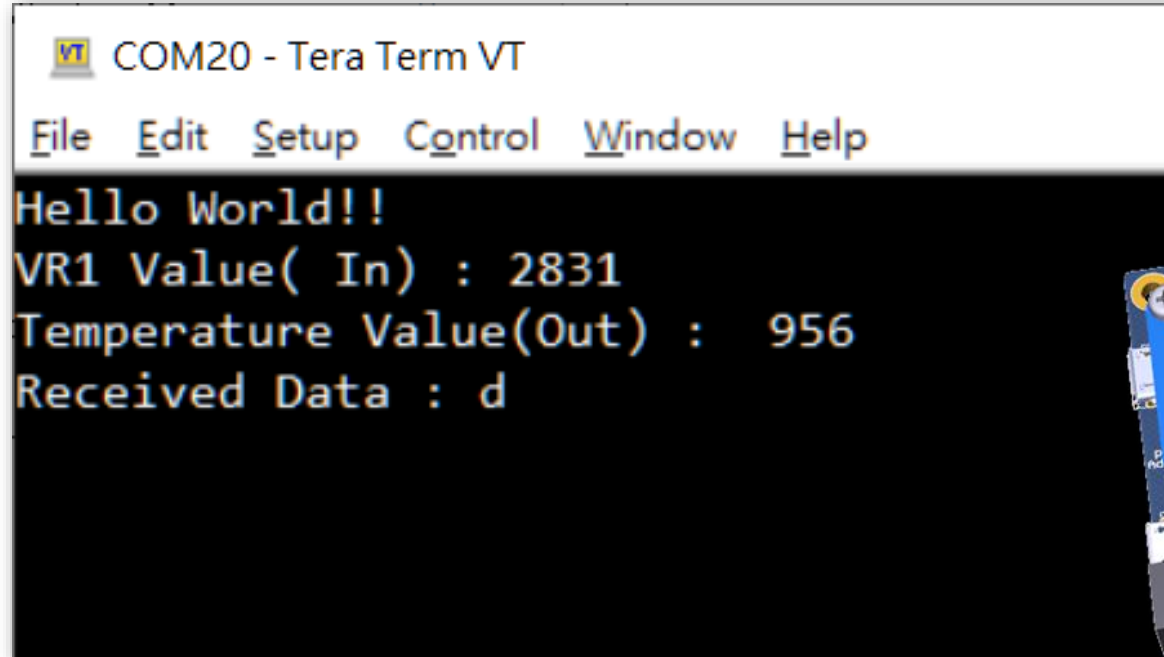
Step 4

```
...  
void ADC_Complete( ADC_STATUS status, uintptr_t context ) { ... }  
...  
int main (void)  
{ ...  
    while ( true )  
    {  
        ...  
        if( ADC_IsCompleted )  
        {  
            ADC_IsCompleted = 0;  
  
            myprintf("\033[2;1H");  
            // TODO 13-2.03  
            myprintf( "VR1 Value(%s) : %4d",  
                    (ADC_InWindow[0] ? " In" : "Out"), ADC_Result[0]);  
  
            myprintf("\033[3;1H");  
            // TODO 13-2.04  
            myprintf( "Temperature Value(%s) : %4d",  
                    (ADC_InWindow[1] ? " In" : "Out"), ADC_Result[1] );  
        }  
    }  
  
    return ( EXIT_FAILURE );  
}
```

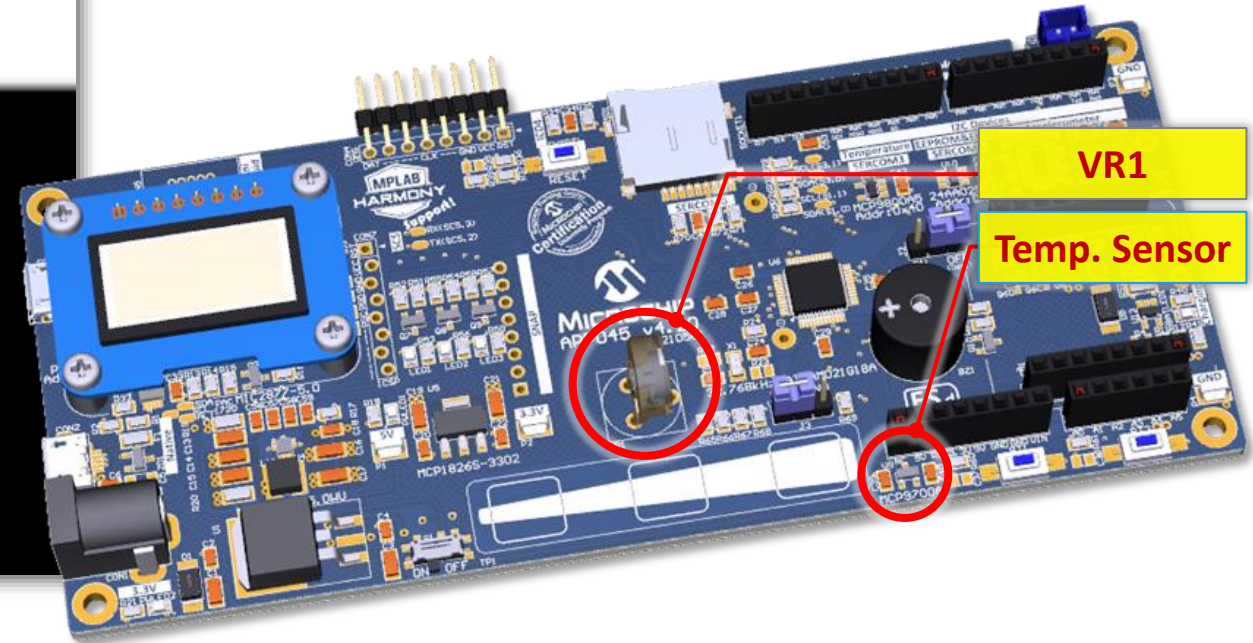
⚙️ Lab13-2 ADC Window Monitor

Result

- If VR1 / Temperature Sensor ADC value in window will shows (In) else shows (Out).



```
VT COM20 - Tera Term VT
File Edit Setup Control Window Help
Hello World!!
VR1 Value( In) : 2831
Temperature Value(Out) : 956
Received Data : d
```



Lab13-2 ADC Window Monitor

Discuss

- Use **INPUTOFFSET** of **INPUTCTRL** register to identify **next channel** of ADC PIN Scan.

```
void ADC_Complete( ADC_STATUS status, uintptr_t context )
{
    if( status & ADC_INTFLAG_RESRDY_Msk )
    {
        ADC_Result[ADC_ChannelIdx] = ADC_ConversionResultGet();

        ADC_InWindow[ADC_ChannelIdx] = 0;
        if( status & ADC_INTFLAG_WINMON_Msk )
        {
            ADC_InWindow[ADC_ChannelIdx] = 1;
        }

        // ADC_ChannelIdx = ( ADC_ChannelIdx + 1 ) % 2;
        ADC_ChannelIdx = (ADC_REGS->ADC_INPUTCTRL & ADC_INPUTCTRL_INPUTOFFSET_Msk)
                        >> ADC_INPUTCTRL_INPUTOFFSET_Pos;

        if( ADC_ChannelIdx == 0 )
        {
            ADC_IsCompleted = 1;
        }
    }
}
```

SAM2001 recall - ADC Pin Scan

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

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

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

OR

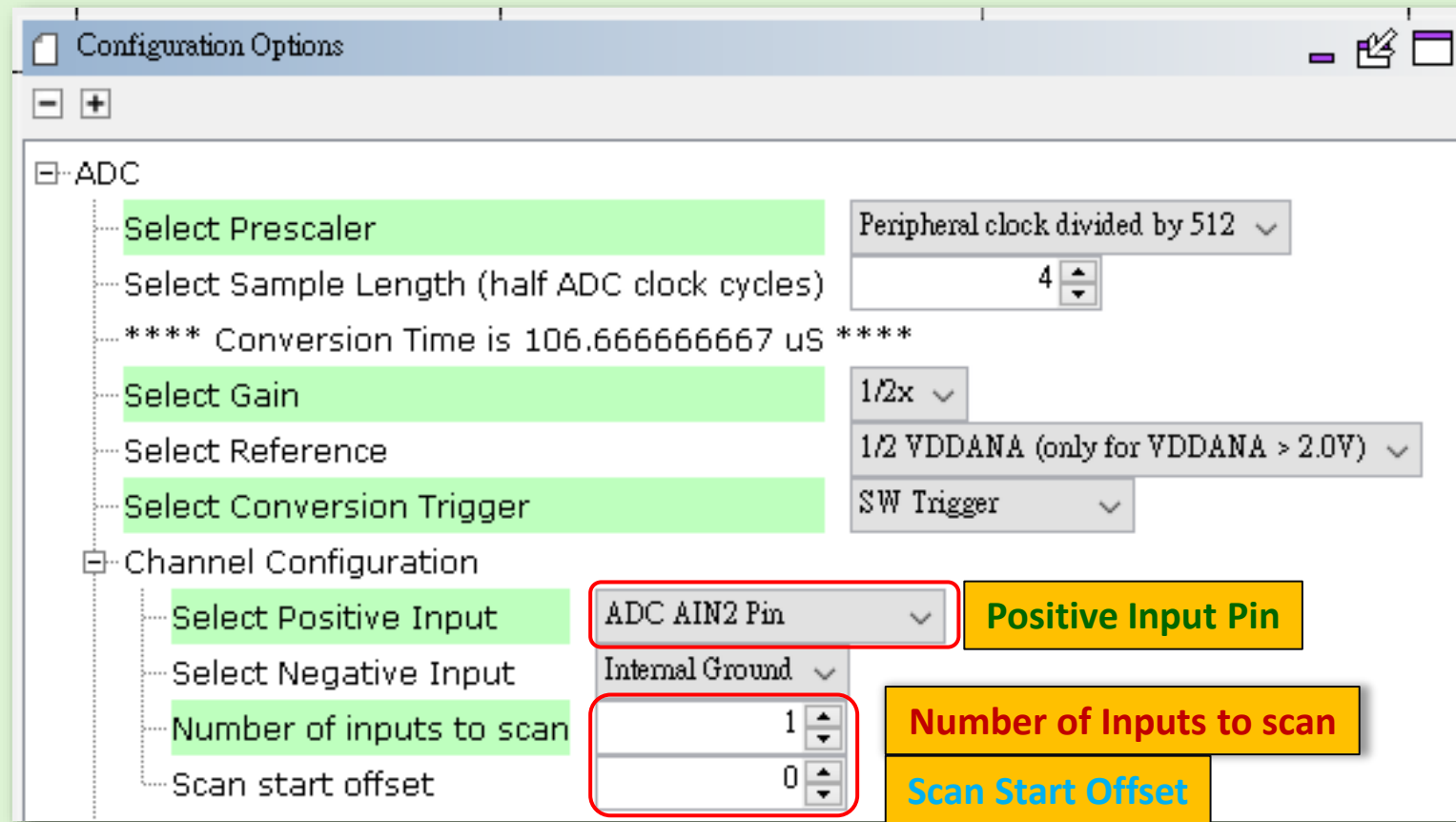
Positive Input Pin = AIN[2]
Scan Start Offset = 0
Start Pin = AIN[2+0] = AIN[2]

Number of Inputs to scan = 1

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

SAM2001 recall - ADC Pin Scan

Configuration Options Example



Configuration Options

ADC

- Select Prescaler: Peripheral clock divided by 512
- Select Sample Length (half ADC clock cycles): 4
- **** Conversion Time is 106.666666667 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 AIN2 Pin **Positive Input Pin**
- Select Negative Input: Internal Ground
- Number of inputs to scan: 1 **Number of Inputs to scan**
- Scan start offset: 0 **Scan Start Offset**

ADC Oversampling

Discuss

Example :

- Result Resolution : 15 bits
- Increase bits of resolution : 3 bits (15bits – 12bits = 3 bits)
- Number of Sample to Average : $\sum 64$ (4^3 samples = 2^6 samples)
- Intermediate Result precision : 18 bits (12bits + 6 = 18 bits overflow)
- Number of Automatic Right Shift : 2 bits (18bits >> 2 = 16bits)
- Adjust Division Factor : /2 (2^1)
- Finial Result Precision : 15 bits (16bit >> 1 = 15 bits)

4ⁿ samples must be accumulated

RESSEL[1:0]	Name	Description
0x0	12BIT	12-bit result
0x1	16BIT	For averaging mode output
0x2	10BIT	10-bit result
0x3	8BIT	8-bit result

Table 33-4. Configuration Required for Oversampling and Decimation

Result Resolution	Number of Samples to Average	AVGCTRL.SAMPLENUM[3:0]	Number of Automatic Right Shifts	AVGCTRL.ADJRES[2:0]
13 bits	$4^1 = 4$	0x2	0	0x1
14 bits	$4^2 = 16$	0x4	0	0x2
15 bits	$4^3 = 64$	0x6	2	0x1
16 bits	$\sum 64$	0x8	4	0x0

$(4^3 = 2^6)$

18 bits
(12+6)

16 bits
(18-2)

15 bits
(16>>1)

ADC Oversampling

Discuss

Example :

Result Resolution : 15 bits
Increase bits of resolution : 3 bits (15bits – 12bits = 3 bits)
Number of Sample to Average : $\Sigma 64$ (4^3 samples = 2^6 samples)
Intermediate Result precision : 18 bits (12bits + 6 = 18 bits overflow)
Number of Automatic Right Shift : 2 bits (18bits >> 2 = 16bits)
Adjust Division Factor : /2 (2^1)
Finial Result Precision : 15 bits (16bit >> 1 = 15 bits)

RESSEL[1:0]	Name	Description
0x0	12BIT	12-bit result
0x1	16BIT	For averaging mode output
0x2	10BIT	10-bit result
0x3	8BIT	8-bit result

Result Configuration

Select Result Resolution

16-bit averaging mode

16-bit averaging mode

Number of Bits

Accumulation/Averaging

Accumulation/Averaging

Number of Accumulated Samples

64 samples

64 samples

Number of Right Shifts

1

1

Left Aligned Result

☐

Enable Result Ready Interrupt

☒

Enable Result Ready Event Out

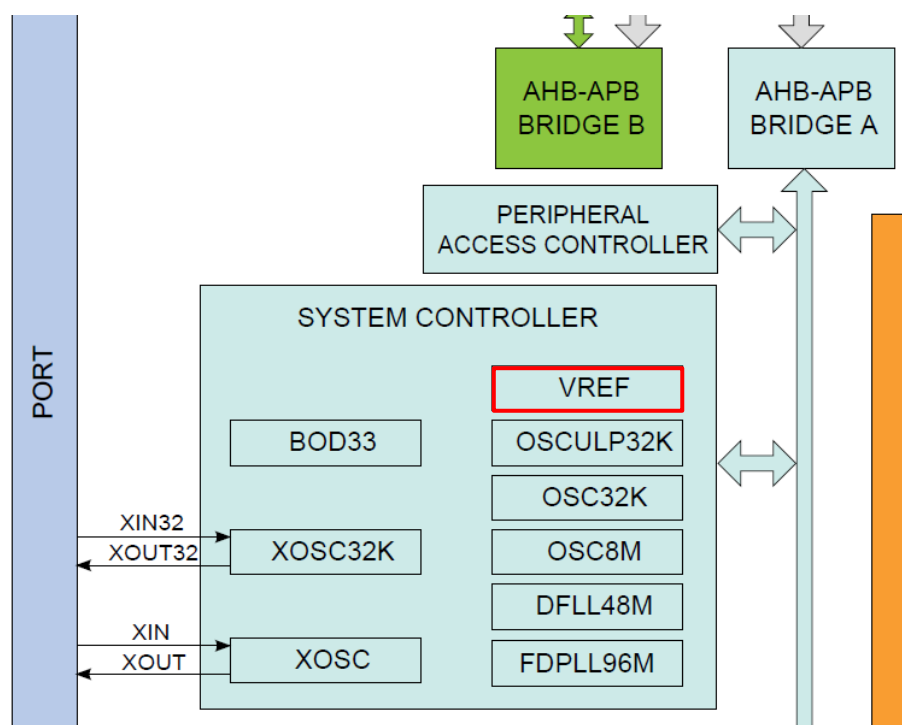
☐

Internal Temperature Sensor

Internal Temperature Sensor

Voltage Reference System

- The **Voltage Reference System (VREF)** consists of a **Bandgap Reference Voltage Generator** and a **Temperature Sensor**.
- Write one to the **Temperature Sensor Enable** bit (**VREF.TSEN**) in **SYSCTRL.VREF** register could enable the integrated temperature sensor.



SYSCTRL – System Controller									
Offset	Name	Bit Pos.							
0x40	VREF	7:0						BGOUTEN	TSEN
		15:8							
		23:16	CALIB[7:0]						
		31:24						CALIB[10:8]	

Note: The TSEN bit is not available on the SAMDA1 device

Internal Temperature Sensor

Characteristics

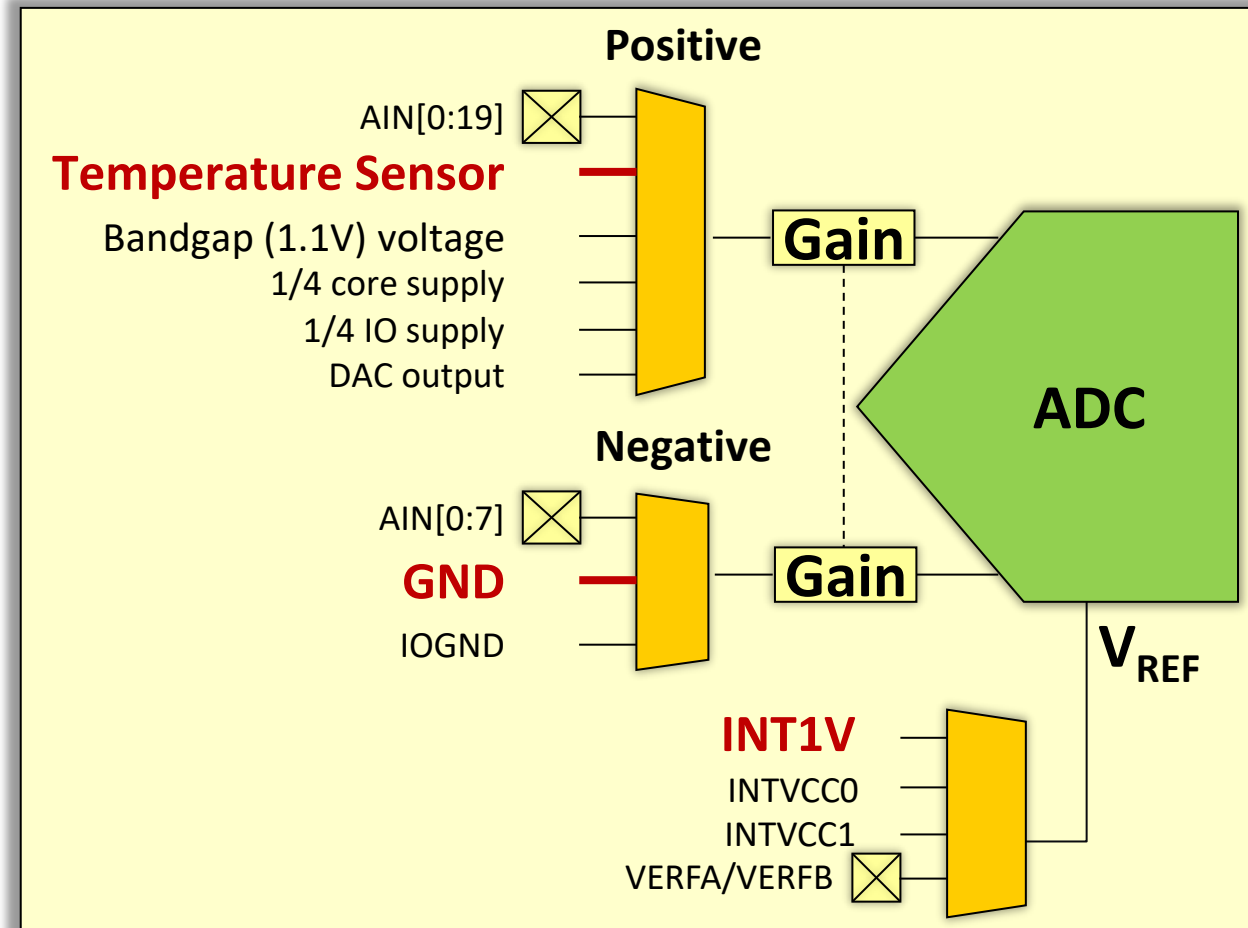
Table 37-40. Temperature Sensor Characteristics⁽¹⁾ (Device Variant B,C, D and L)

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
	Temperature sensor output voltage	T= 25°C, V _{DDANA} = 3.3V	-	0.688	-	V
	Temperature sensor slope		2.06	2.16	2.26	mV/°C
	Variation over V _{DDANA} voltage	V _{DDANA} =1.62V to 3.6V	-0.4	1.4	3	mV/V
	Temperature Sensor accuracy	Using the method described in the 37.11.8.2 Software-based Refinement of the Actual Temperature	-10	-	10	°C

Internal Temperature Sensor

Measurement by ADC with INT1V reference voltage

- The **Integrated Temperature Sensor** could be accessible by using **ADC** with **INT1V** reference voltage for measure the **Core Temperature** in linear voltage proportional.



Internal Temperature Sensor

Factory Calibration

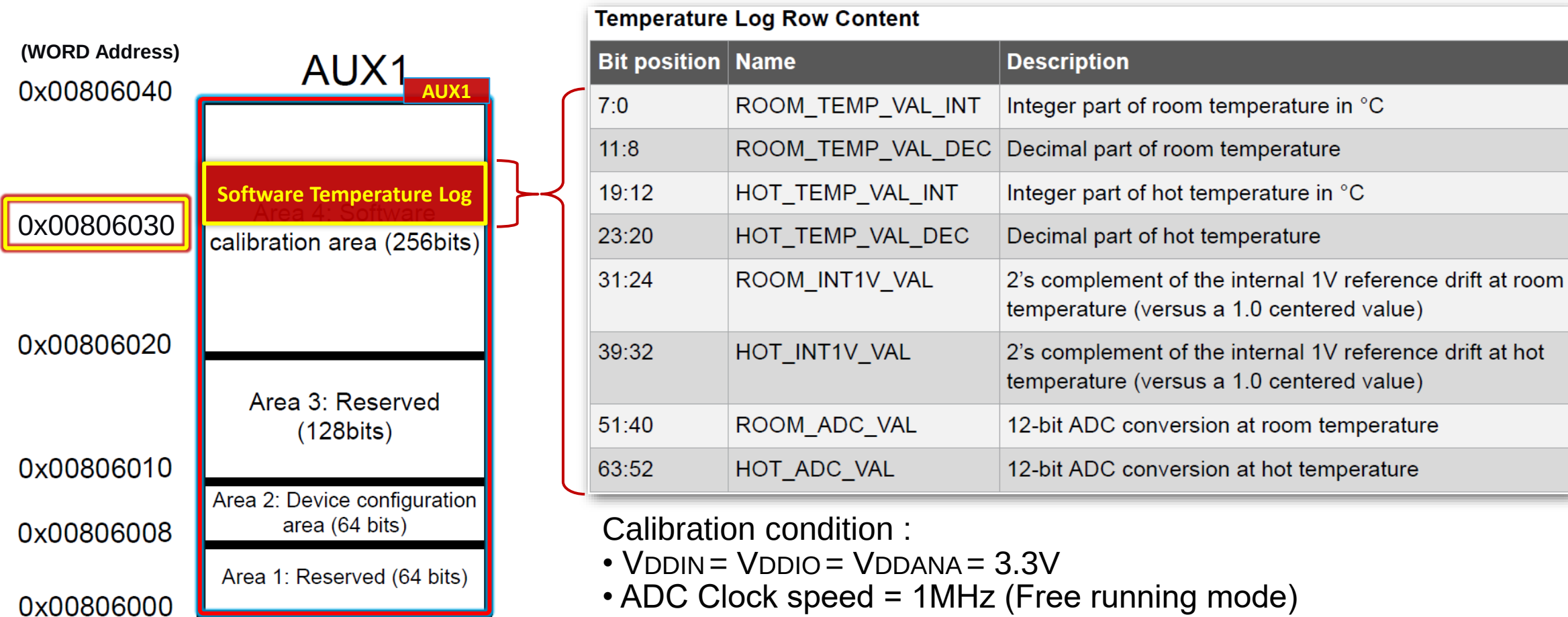
- **Software-based Refinement of the Actual Temperature**

The temperature sensor behavior is linear but depends on several parameters such as the **internal voltage reference**, which itself depends on the temperature.

- To take this into account, each device contains a **Temperature Log row** with data measured and **written during the production tests**. These calibration values should be read by software to infer the most accurate temperature readings possible.
- This Software **Temperature Log row** can be read at address **0x00806030**, This section specifies the Temperature Log row content and explains how to refine the temperature sensor output using the values in the Temperature Log row.

Temperature Log Row of NVM

AUX1 (Area 4 : Software Calibration area) – Read Only



Calibration condition :

- VDDIN = VDDIO = VDDANA = 3.3V
- ADC Clock speed = 1MHz (Free running mode)
- ADC averaging mode with 4 averaged samples
- ADC voltage reference = 1.0V (Internal reference INT1V)
- ADC input = Temperature sensor
- Room Temperature 25.2°C, • Hot temperature 83.3°C

Learn this detail in SAM2001ADV chapter
Auxiliary Space of NVM and Fuse Maintains

Temperature Log Row of NVM

Temperature Log could be read and check in Microchip Studio



MPLAB® Snap (BUR190973931) - Device Programming

Tool: MPLAB® Snap | Device: ATSAM21G18A | Interface: SWD | Device signature: 0x10010305 | Target Voltage: 3.3 V

Interface settings | Tool information | Device information | Memories | **Fuses** | Security

Fuse Name	Value
✓ TEMP_LOG_WORD_0.ROOM_TEMP_VAL_INT	0x1E
✓ TEMP_LOG_WORD_0.ROOM_TEMP_VAL_DEC	0x01
✓ TEMP_LOG_WORD_0.HOT_TEMP_VAL_INT	0x7D
✓ TEMP_LOG_WORD_0.HOT_TEMP_VAL_DEC	0x00
✓ TEMP_LOG_WORD_0.ROOM_INT1V_VAL	0xFD
✓ TEMP_LOG_WORD_1.HOT_INT1V_VAL	0xF8
✓ TEMP_LOG_WORD_1.ROOM_ADC_VAL	0x0B22
✓ TEMP_LOG_WORD_1.HOT_ADC_VAL	0x0E91

Software Temperature Log

Fuse Register	Value
TEMP_LOG_WORD_0	0xFD07D11E
TEMP_LOG_WORD_1	0xE91B22F8

Learn this detail in SAM2001ADV chapter
Auxiliary Space of NVM and Fuse Maintains

Temperature Log Row of NVM

Temperature Log Row cannot read for check in MPLAB X IDE

Temperature Log not in MPLAB XIDE Configuration Bits window



Configuration Bits ×							
Address	Name	Value	Field	Option	Category	Setting	
0080_4000	USER_WORD_0	D8E0C7FF	-	-	-	-	
	7		NVMCTRL_BOOTPROT	SIZE_0BYTES	Bootloader Size	0 Bytes	
	7		NVMCTRL_EEPROM_SIZE	SIZE_0BYTES	EEPROM Size	0 Bytes	
	7		BOD33USERLEVEL	User range: 0x0 - 0x3F	BOD33 User Level	Enter Hexadecimal value	
	1		BOD33_EN	ENABLED	BOD33 Enable	BOD33 is enabled	
	1		BOD33_ACTION	RESET	BOD33 Action	The BOD33 generates a reset	
	0		WDT_ENABLE	DISABLED	WDT Enable	WDT is disabled	
	0		WDT_ALWAYSON	DISABLED	WDT Always On	WDT is enabled and disabled through ENABLE bit	
	B		WDT_PER	CYCL6384	WDT Period	16384 clock cycles	
	1		WDT_WINDOW_0	SET	WDT Window bit 0	SET	
0080_4004	USER_WORD_1	FFFFFFC5C	-	-	-	-	
	4		WDT_WINDOW_1	User range: 0x0 - 0x7	WDT Window bits 3:1	Enter Hexadecimal value	
	B		WDT_EWOFFSET	CYCL6384	WDT Early Warning Offset	16384 clock cycles	
	0		WDT_WEN	DISABLED	WDT Window Mode Enable	WDT is disabled	
	0		BOD33_HYST	DISABLED	BOD33 Hysteresis	No Hysteresis	
	FFFF		NVMCTRL_REGION_LOCKS	User range: 0x0 - 0xFFFF	NVM Region Locks	Enter Hexadecimal value	

Use Temperature Log Row to Calculation Temperature

- (ROOM_TEMP_VAL_INT, ROOM_TEMP_VAL_DEC) to denote as **Temp_R**
- (HOT_TEMP_VAL_INT, HOT_TEMP_VAL_DEC) to denote as **Temp_H**
- ROOM_INT1V_VAL calculate by $1 - (\text{ROOM_INT1V_VAL} / 1000)$ to denote as **INT1V_R**
- HOT_INT1V_VAL calculate by $1 - (\text{HOT_INT1V_VAL} / 1000)$ to denote as **INT1V_H**
- ROOM_ADC_VAL to denote as **ADC_R** with V_{REF} **INT1V_R**, then the measured voltage is denoted **V_{ADCR}**
- HOT_ADC_VAL to denote as **ADC_H** with V_{REF} **INT1V_H**, then the measured voltage is denoted **V_{ADCH}**

	Source Field of Log Row	Symbol	Room	Hot
Temperature	XXXX_TEMP_VAL_INT XXXX_TEMP_VAL_DEC	Temp	30.1 °C	125 °C
V _{REF} voltage	XXXX_INT1V_VAL	INT1V	1.00 V	1.01 V
ADC value	XXXX_ADC_VAL	ADC	2850	3729
ADV voltage	XXXX_ADC_VAL	V _{ADC}	0.7 V	0.92 V

Note : Temperature Log Row are not equal from chip to chip, here is an example !!

Use Temperature Log Row to Calculation Temperature

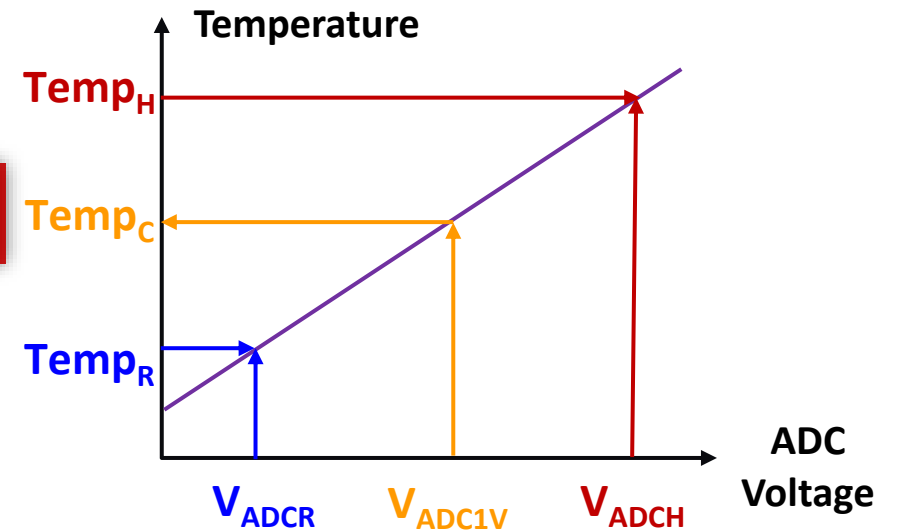
- (ROOM_TEMP_VAL_INT, ROOM_TEMP_VAL_DEC) to denote as **Temp_R**
- (HOT_TEMP_VAL_INT, HOT_TEMP_VAL_DEC) to denote as **Temp_H**
- ROOM_INT1V_VAL calculate by **1-(ROOM_INT1V_VAL/1000)** to denote as **INT1V_R**
- HOT_INT1V_VAL calculate by **1-(HOT_INT1V_VAL/1000)** to denote as **INT1V_H**
- ROOM_ADC_VAL to denote as **ADC_R** with V_{REF} **INT1V_R**, then the measured voltage is denoted **V_{ADCR}**
- HOT_ADC_VAL to denote as **ADC_H** with V_{REF} **INT1V_H**, then the measured voltage is denoted **V_{ADCH}**
- Temperature to measurement correspond to **Coarse** and **Finer** are denoted as **Temp_C** and **Temp_F**
- V_{REF} INT1V correspond to **Ideal 1.0V** and **reality Measurement** are denoted as **INT1V** and **INT1V_M**
- ADC result denoted **ADC_M**, the ADC voltage to use the **Ideal 1.0V** V_{REF} **INT1V** is denoted **V_{ADC1V}**
- ADC result denoted **ADC_M**, the ADC voltage to use the **reality Measurement** V_{REF} **INT1V_M** is denoted **V_{ADCM}**

Using the linear proportional, we got following equation:

$$\frac{(V_{ADC1V} - V_{ADCR})}{(Temp_C - Temp_R)} = \frac{(V_{ADCH} - V_{ADCR})}{(Temp_H - Temp_R)}$$

$$Temp_C = Temp_R + \frac{(V_{ADC1V} - V_{ADCR}) \times (Temp_H - Temp_R)}{(V_{ADCH} - V_{ADCR})}$$

**Coarse
Temperature**



with Ideal 1.0V_{REF}

Use Temperature Log Row to Calculation Temperature

- Now we convert the Voltage of previous expression to be expressed in ADC value to calculate the Coarse Temperature Value $Temp_C$

$$Temp_C = Temp_R + \frac{(V_{ADC1V} - V_{ADCR}) \times (Temp_H - Temp_R)}{(V_{ADCH} - V_{ADCR})}$$

$$Temp_C = Temp_R + \frac{\overset{\text{Ideal } 1.0V_{REF}}{\left((ADC_M \times \frac{INT1V}{4095}) - (ADC_R \times \frac{INT1V_R}{4095}) \right)} \times (Temp_H - Temp_R)}{\left((ADC_H \times \frac{INT1V_H}{4095}) - (ADC_R \times \frac{INT1V_R}{4095}) \right)}$$

$Temp_C$ we say the “Coarse Temperature value” because we assume the reference voltage $INT1V == 1.0V$ ideal voltage

The physical ADC resolution is 12 bits, then $2^{12}-1 = 4095$

Measured Voltage = ADC value $\times \frac{\text{Reference Voltage}}{4095}$

Use Temperature Log Row to Calculation Temperature

- Using the linear proportional and by previous calculated result $Temp_C$ to calculate the reality V_{REF} INT1V_M then derive the V_{ADCM}

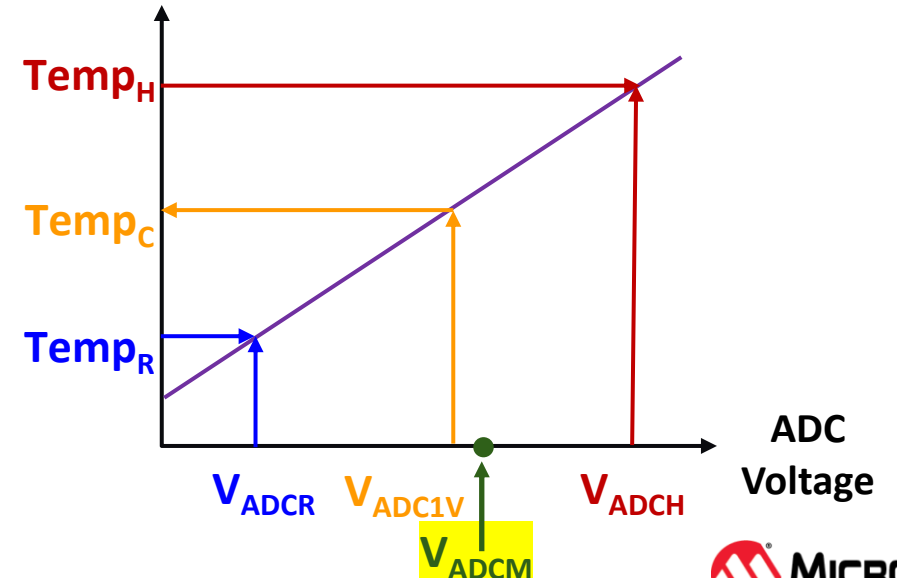
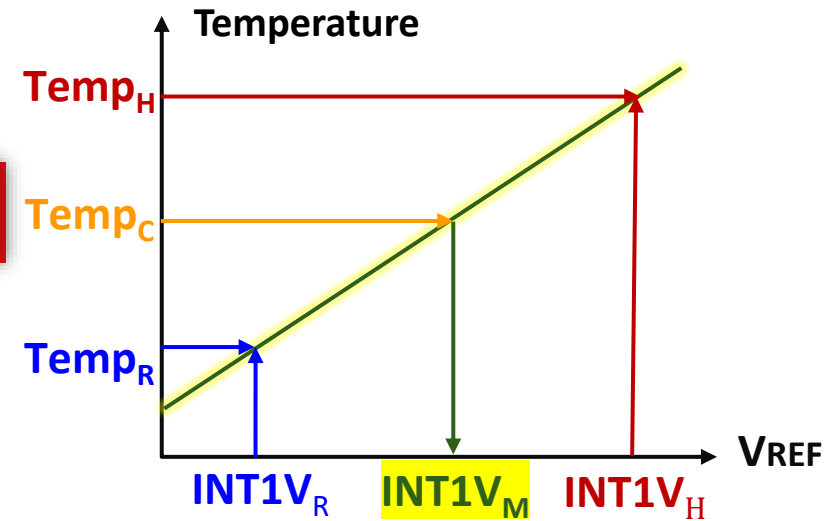
$$\frac{(INT1V_M - INT1V_R)}{(Temp_C - Temp_R)} = \frac{(INT1V_H - INT1V_R)}{(Temp_H - Temp_R)}$$

$$INT1V_M = INT1V_R + \frac{(INT1V_H - INT1V_R) \times (Temp_C - Temp_R)}{(Temp_H - Temp_R)}$$

$$V_{ADC1V} = ADC_M \times \frac{INT1V}{4095}, \quad V_{ADCM} = ADC_M \times \frac{INT1V_M}{4095}$$

Replace INT1V(1.0V ideal) by INT1V_M to calculate the V_{ADCM}

Coarse
Temperature



Use Temperature Log Row to Calculation Temperature

- Use V_{ADCM} to instead of the V_{ADC1V} to calculate the finer temperature value Temp_F

$$\frac{(V_{\text{ADC1V}} - V_{\text{ADCR}})}{(\text{Temp}_C - \text{Temp}_R)} = \frac{(V_{\text{ADCH}} - V_{\text{ADCR}})}{(\text{Temp}_H - \text{Temp}_R)}$$

Coarse
Temperature

Ideal 1.0V

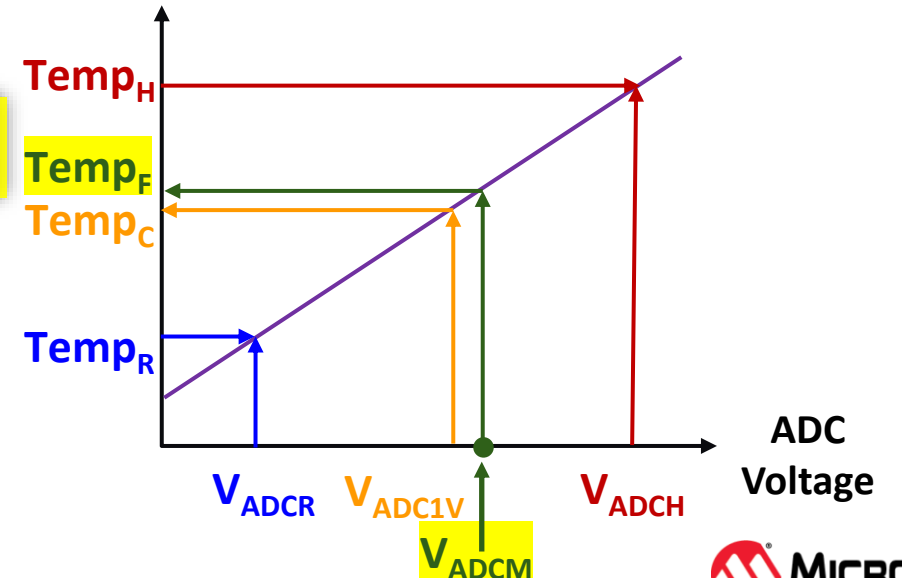
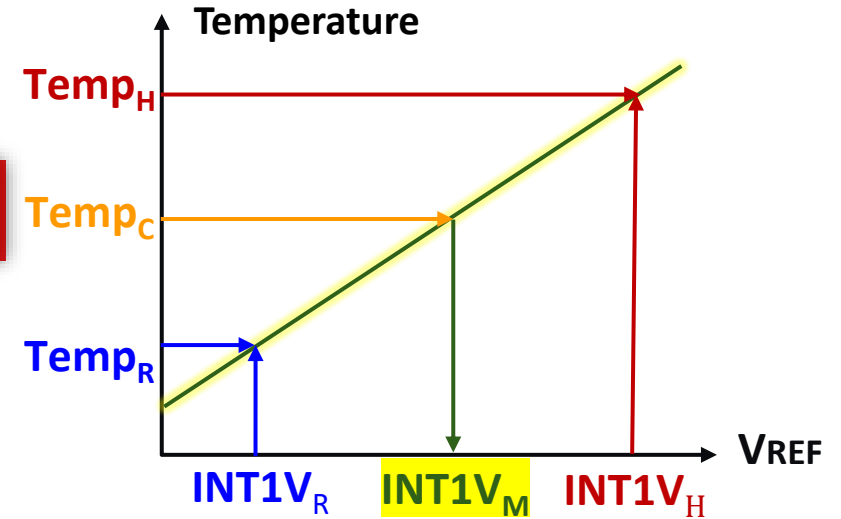
$$\text{Temp}_C = \text{Temp}_R + \frac{(V_{\text{ADC1V}} - V_{\text{ADCR}}) \times (\text{Temp}_H - \text{Temp}_R)}{(V_{\text{ADCH}} - V_{\text{ADCR}})}$$

Finer
Temperature

Reality 1.0V

$$\text{Temp}_F = \text{Temp}_R + \frac{(V_{\text{ADCM}} - V_{\text{ADCR}}) \times (\text{Temp}_H - \text{Temp}_R)}{(V_{\text{ADCH}} - V_{\text{ADCR}})}$$

Replace V_{ADC1V} by V_{ADCM} to calculate the finer result Temp_F



Lab12-1 Internal Temperature Sensor through ADC

- Please continue from [SAM2001 Lab12 \(ADC Polling Software Trigger\)](#)
- Try enable Internal Temperature Sensor and load NVM [Temperature Log Row](#) to calculate reality temperature through [ADC](#) conversion.
- Internal Temperature Sensor should configure to choose [internal 1.0V bandgap](#) voltage as reference voltage of ADC, no gain control ([Gain 1x](#)), [Temperature Reference](#) as positive input and [12-bits resolution](#).
- Try observe both ADC and Temperature Value use UART output.
- **Let's go!**

Lab12-1 Internal Temperature Sensor through ADC

Step 1 (Use Configuration Options)

Configuration Options

ADC

Select Prescaler: Peripheral clock divided by 512 **Prescaler : divided by 512**

Select Sample Length (cycles): 4 **Sample Length : 4**

**** Conversion Time is 101.391734973 uS ****

Select Gain: 1x **Gain : 1x**

Select Reference: 1.0V voltage reference **1.0V voltage reference**

Select Conversion Trigger: SW Trigger **SW Trigger**

Channel Configuration

Select Positive Input: Temperature Reference **Temperature Reference**

Select Negative Input: Internal Ground **Internal Ground**

Number of inputs to scan: 0

Result Configuration

Select Result Resolution: 12-bit result **12-bit result**

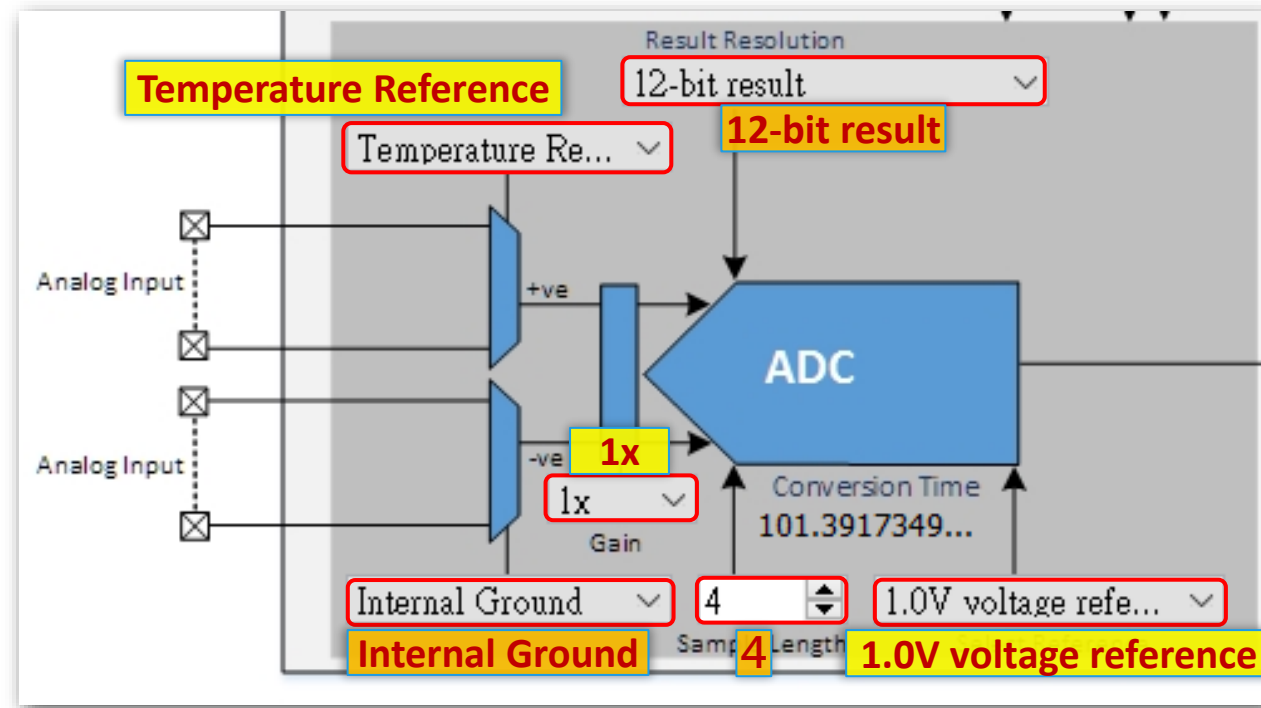
Left Aligned Result: ☐

Enable Result Ready Interrupt: ☐

Enable Result Ready Event Out: ☐

Lab12-1 Internal Temperature Sensor through ADC

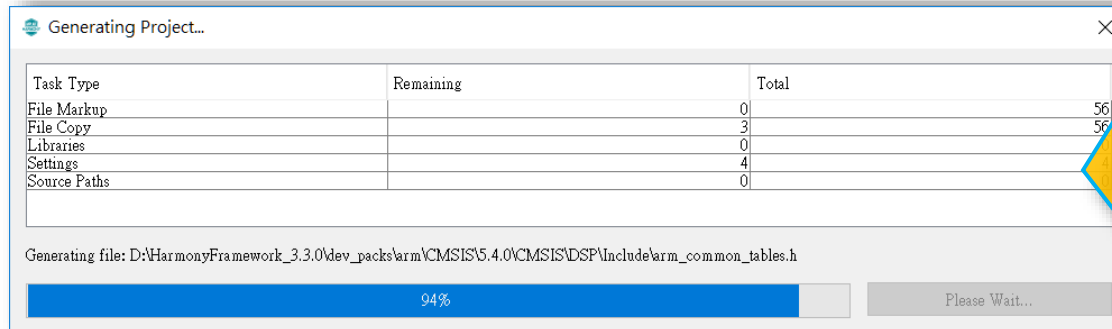
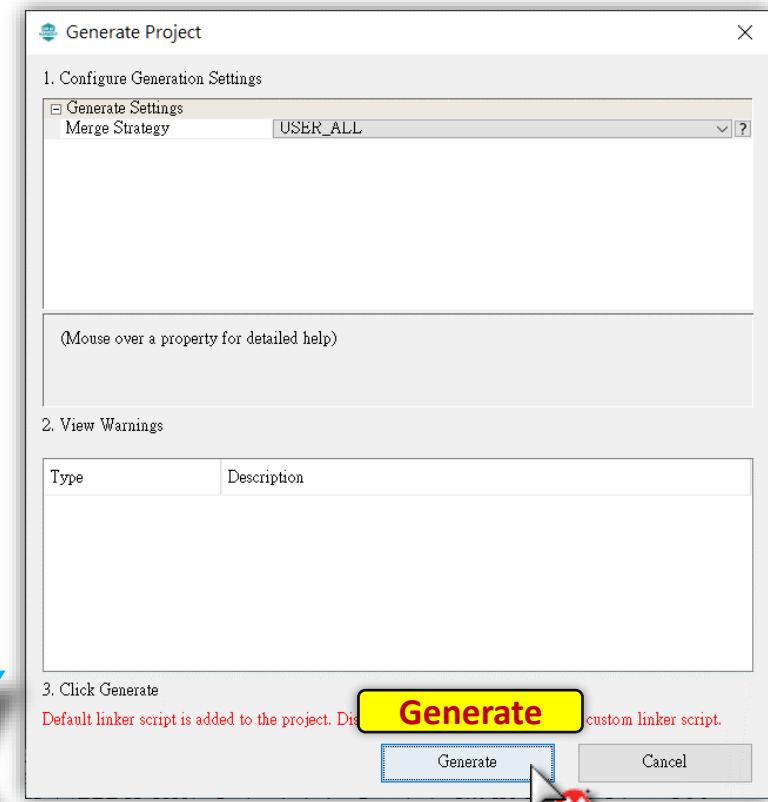
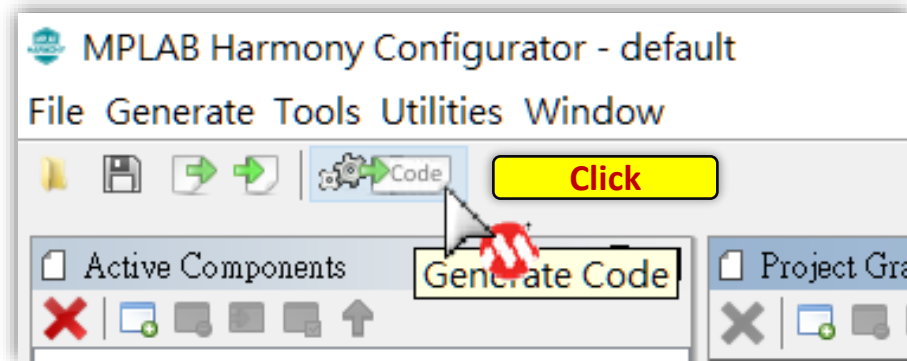
Step 1 (Use Easy View)



Lab12-1 Internal Temperature Sensor through ADC

Step 2

- Click  to Generate Code.



Lab12-1 Internal Temperature Sensor through ADC

Step 3

```
...
#include <string.h>
#include <stdarg.h>
// TODO 12-1.01
#include <math.h>

uint32_t i = 0;
...
uint16_t ADC_Result;

// TODO 12-1.02
float TempR;
float TempH;
float INT1VR;
float INT1VH;
float VADCR;
float VADCH;
float TempM;

float Dec2Frac( uint8_t DecVal )
{
    if      (DecVal < 10) return ((float)DecVal/10.0);
    else if (DecVal <100) return ((float)DecVal/100.0);
    else      return ((float)DecVal/1000.0);
}
```

Lab12-1 Internal Temperature Sensor through ADC

Step 4

```
// TODO 12-1.02
...
float TempH;
float INT1VR;
float INT1VH;
float VADCR;
float VADCH;
float TempM;
```

```
float Dec2Frac( uint8_t DecVal ) {...}
```

```
void Load_NVM_TempLogRow( void )
{
```

```
    uint8_t room_temp_val_int;
    uint8_t room_temp_val_dec;
    uint8_t hot_temp_val_int;
    uint8_t hot_temp_val_dec;
    int8_t room_int1v_val;
    int8_t hot_int1v_val;
    uint16_t ADCR;
    uint16_t ADCH;
```

```
// Enable System Voltage Reference of Internal Temperature Sensor
SYSCTRL_REGS->SYSCTRL_VREF|=SYSCTRL_VREF_TSEN(1);
```

SYSCTRL – System Controller

Offset	Name	Bit Pos.									
0x40	VREF	7:0								BGOUTEN	TSEN
		15:8									
		23:16	CALIB[7:0]								
		31:24								CALIB[10:8]	

Lab12-1 Internal Temperature Sensor through ADC

Step 5

```
// TODO 12-1.02
...

void Load_NVM_TempLogRow( void )
{
    ...
    SYSCTRL_REGS->SYSCTRL_VREF|=SYSCTRL_VREF_TSEN(1);

    // Load Temperature Sensor Calibration Data From NVM Fuse
    room_temp_val_int = (uint8_t)((*(uint32_t*)TEMP_LOG_ADDR)>>0) & 0xFF);
    room_temp_val_dec = (uint8_t)((*(uint32_t*)TEMP_LOG_ADDR)>>8) & 0xF);
    hot_temp_val_int = (uint8_t)((*(uint32_t*)TEMP_LOG_ADDR)>>12) & 0xFF);
    hot_temp_val_dec = (uint8_t)((*(uint32_t*)TEMP_LOG_ADDR)>>20) & 0xF);
    room_int1v_val = (int8_t)((*(uint32_t*)TEMP_LOG_ADDR)>>24) & 0xFF);
    hot_int1v_val = (int8_t)((*(uint32_t*)(TEMP_LOG_ADDR+4))>>0) & 0xFF);
    ADCR = (uint16_t)((*(uint32_t*)(TEMP_LOG_ADDR+4))>>8) & 0xFFF);
    ADCH = (uint16_t)((*(uint32_t*)(TEMP_LOG_ADDR+4))>>20) & 0xFFF);

    // Combination Integer and Decimal to a float type temperature value
    TempR = room_temp_val_int + Dec2Frac(room_temp_val_dec);
    TempH = hot_temp_val_int + Dec2Frac(hot_temp_val_dec);
    INT1VR = 1 - ((float)room_int1v_val / 1000.0);
    INT1VH = 1 - ((float)hot_int1v_val / 1000.0);
    VADCR = ((float)ADCR * INT1VR) / 4095.0;
    VADCH = ((float)ADCH * INT1VH) / 4095.0;
}
```

Lab12-1 Internal Temperature Sensor through ADC

Step 6

```
// TODO 12-1.02
...
void Float2IntDec( float FloatVal, int *IntVal, int *DecVal, int DecNum ) {...}
void Load_NVM_TempLogRow( void ) { ... }

float CalculateTemperature( uint16_t ADCM )
{
    float TempC;
    float TempF;
    float VADC1V;
    float VADCM;
    float INT1V;
    float INT1VM;

    INT1V = 1.0;
    VADC1V = (( float )ADCM * INT1V) / 4095.0;
    TempC = TempR + ((VADC1V - VADCR) * (TempH - TempR) / (VADCH - VADCR));
    INT1VM = INT1VR + ((INT1VH - INT1VR) * (TempC - TempR) / (TempH - TempR));
    VADCM = (( float )ADCM * INT1VM) / 4095.0;
    TempF = TempR + ((VADCM - VADCR) * (TempH - TempR) / (VADCH - VADCR));

    return TempF;
}

void USART5_Transmit( uintptr_t context ) { ... }
```

Lab12-1 Internal Temperature Sensor through ADC

Step 7

```
...
int main ( void )
{
    ...
    SERCOM5_USART_Read( USART5_ReceiveData, 1 );

    // TODO 12-1.03
    Load_NVM_TempLogRow();

    ADC_Enable();

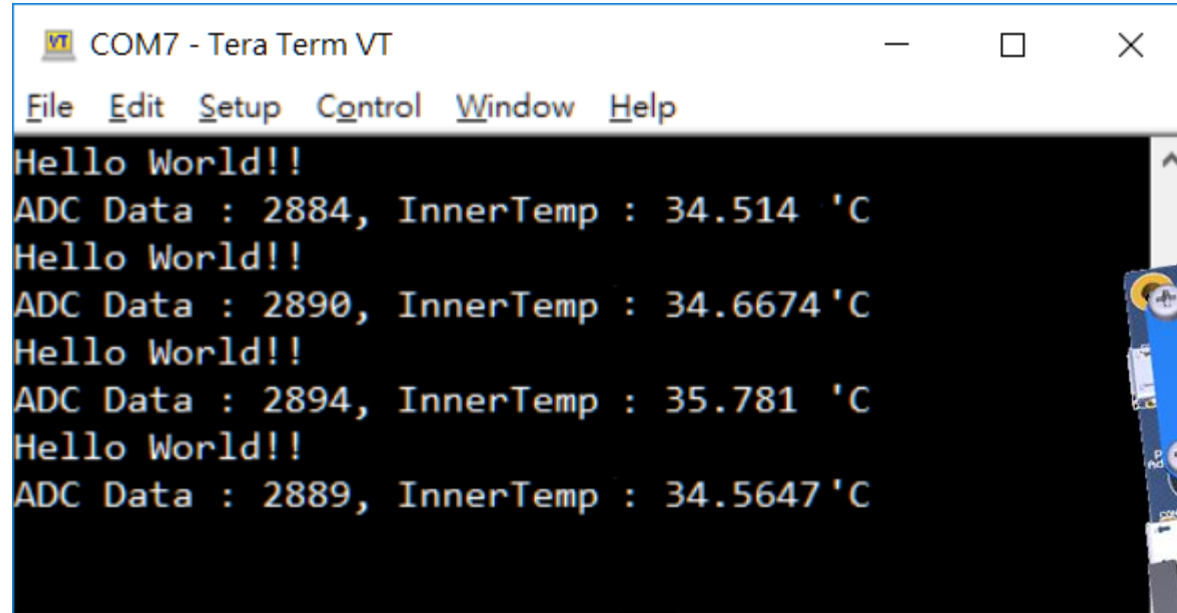
    while ( true )
    {
        ...
        if (TC3_HasExpired)
        {
            ...
            ADC_Result = ADC_ConversionResultGet();

            // TODO 12-1.04
            TempM = CalculateTemperature( ADC_Result );
            myprintf( "ADC Data : %4d, InnerTemp : %3.4f'C\r\n", ADC_Result, TempM );
        }
        ...
    }
}
```

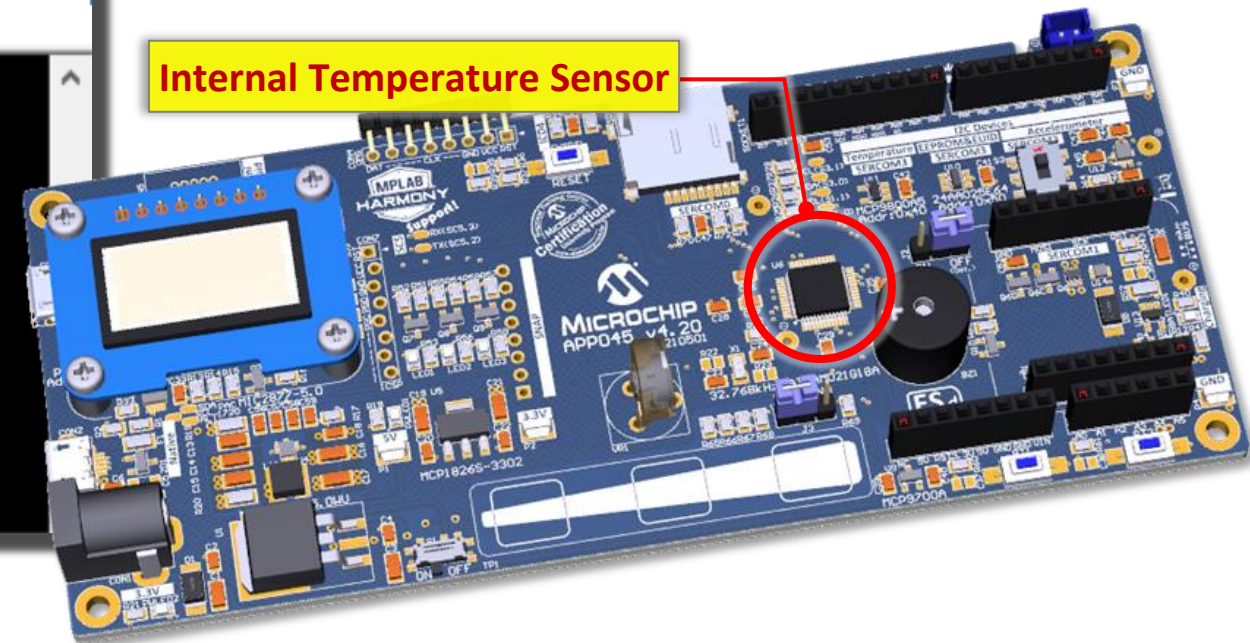
Lab12-1 Internal Temperature Sensor through ADC

Result

- MCU Internal Temperature Sensor result output.



```
COM7 - Tera Term VT
File Edit Setup Control Window Help
Hello World!!
ADC Data : 2884, InnerTemp : 34.514 'C
Hello World!!
ADC Data : 2890, InnerTemp : 34.6674 'C
Hello World!!
ADC Data : 2894, InnerTemp : 35.781 'C
Hello World!!
ADC Data : 2889, InnerTemp : 34.5647 'C
```



Lab12-1 Internal Temperature Sensor through ADC

Discuss

- Try output Temperature Log Row, Coarse and Finer result of data.

```
#include <math.h>
void myprintf(const char *format, ...);
...
void Load_NVM_TempLogRow( void )
{
    ...
    VADCH = ((float)ADCH * INT1VH) / 4095.0;

    myprintf("\033[2;1H\t\tSymbol\tRoom\tHot");
    myprintf("\033[3;1HTemperature\tTemp\t%3.2f'C\t%3.2f'C", TempR, TempH);
    myprintf("\033[4;1HVREF voltage\tINT1V\t%3.2fV\t%3.2fV", INT1VR, INT1VH);
    myprintf("\033[5;1HADC value\tADC\t%4d\t%4d", ADCR, ADCH);
    myprintf("\033[6;1HADC voltage\tVADC\t%3.2fV\t%3.2fV", VADCR, VADCH);
}

float CalculateTemperature( uint16_t ADCM )
{
    ...
    TempF = TempR + ((VADCM - VADCR) * (TempH - TempR) / (VADCH - VADCR));

    myprintf("\033[8;1HINT1V =%1.4fV, VADC1V = %1.4fV, TempC = %3.2f'C", INT1V, VADC1V, TempC);
    myprintf("\033[9;1HINT1VM=%1.4fV, VADCM = %1.4fV, TempF = %3.2f'C", INT1VM, VADCM, TempF);

    return TempF;
}
```

Lab12-1 Internal Temperature Sensor through ADC

Discuss

- Try output Temperature Log Row, Coarse and Finer result of data.

```
...
int main ( void )
{
    ...
    SERCOM5_USART_Read( USART5_ReceiveData, 1 );

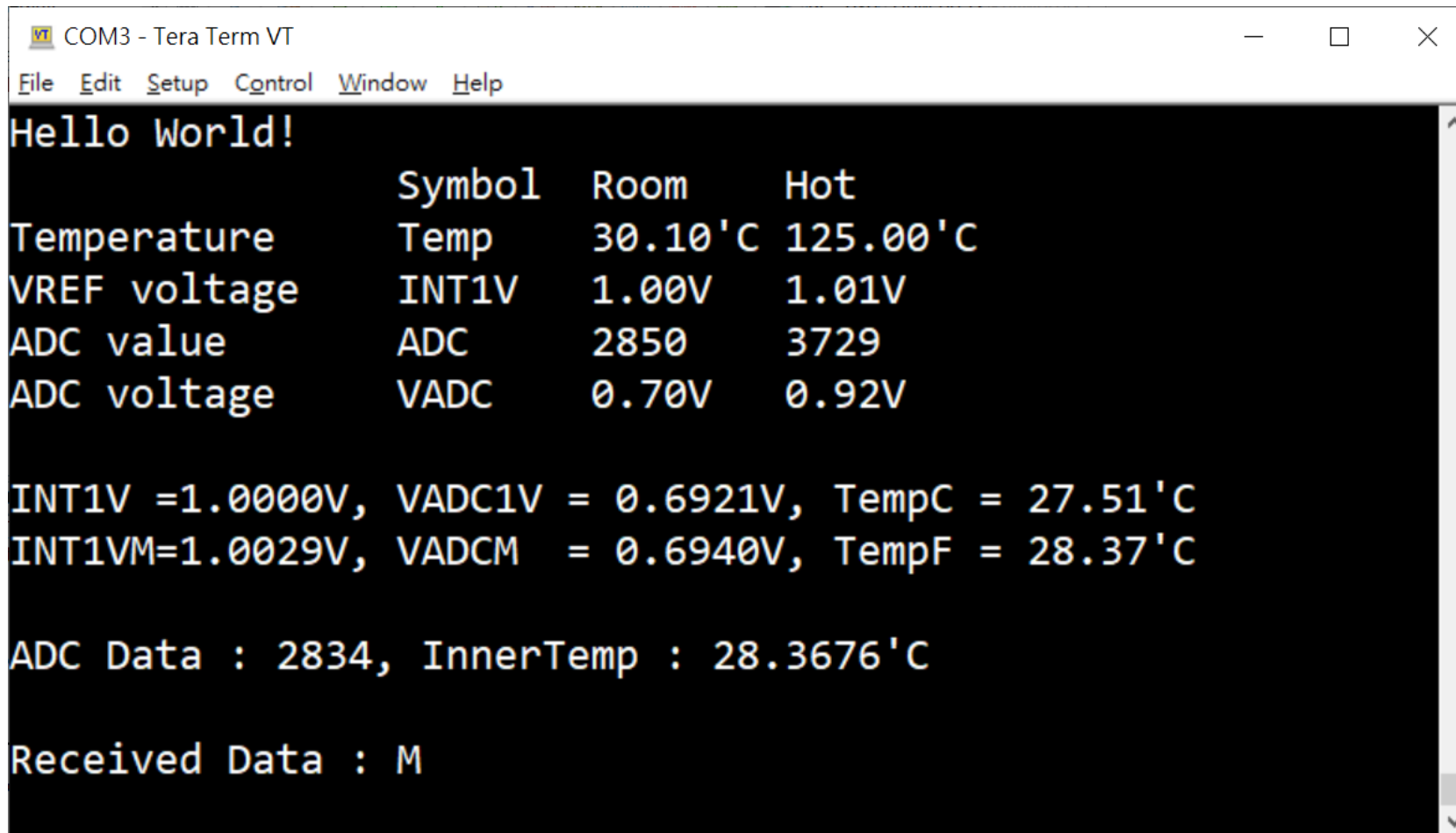
    myprintf("\033[2J");    // Clear screen
    myprintf("\033[?25l");  // Hide cursor
    ...
    while ( true )
    {
        ...
        if (TC3_HasExpired)
        {
            ...
            myprintf( "\033[1;1HHello World!" );

            ...
            myprintf( "\033[11;1HADC Data : %4d, InnerTemp : %3.4f'C", ADC_Result, TempM );
        }
        ...
    }
}
```

Lab12-1 Internal Temperature Sensor through ADC

Discuss

- Output result (Please use the console has supported VT100, for example the TeraTerm)



The screenshot shows a TeraTerm window titled 'COM3 - Tera Term VT'. The window contains the following text output:

```
Hello World!
```

	Symbol	Room	Hot
Temperature	Temp	30.10'C	125.00'C
VREF voltage	INT1V	1.00V	1.01V
ADC value	ADC	2850	3729
ADC voltage	VADC	0.70V	0.92V

INT1V =1.0000V, VADC1V = 0.6921V, TempC = 27.51'C
INT1VM=1.0029V, VADCM = 0.6940V, TempF = 28.37'C

ADC Data : 2834, InnerTemp : 28.3676'C

Received Data : M

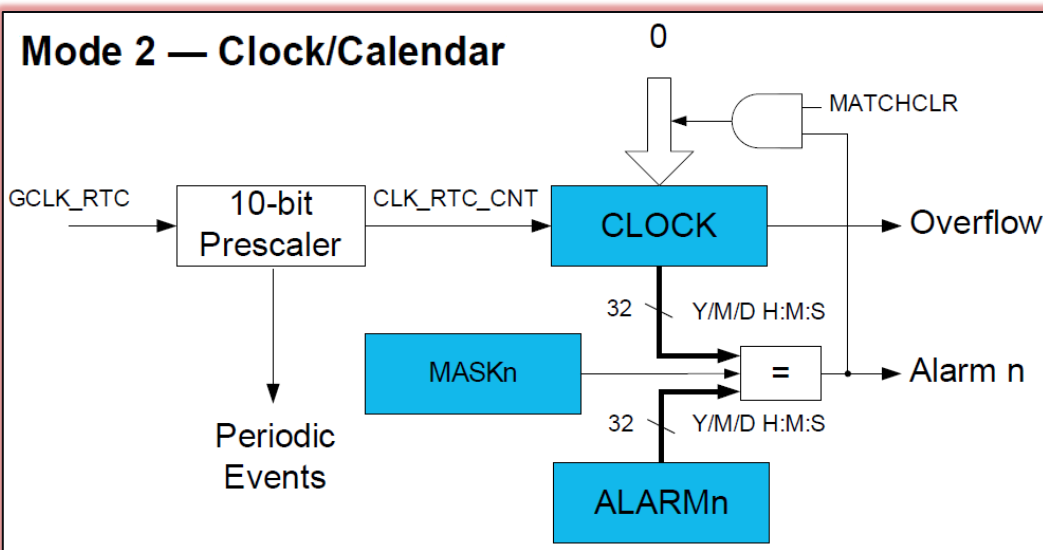
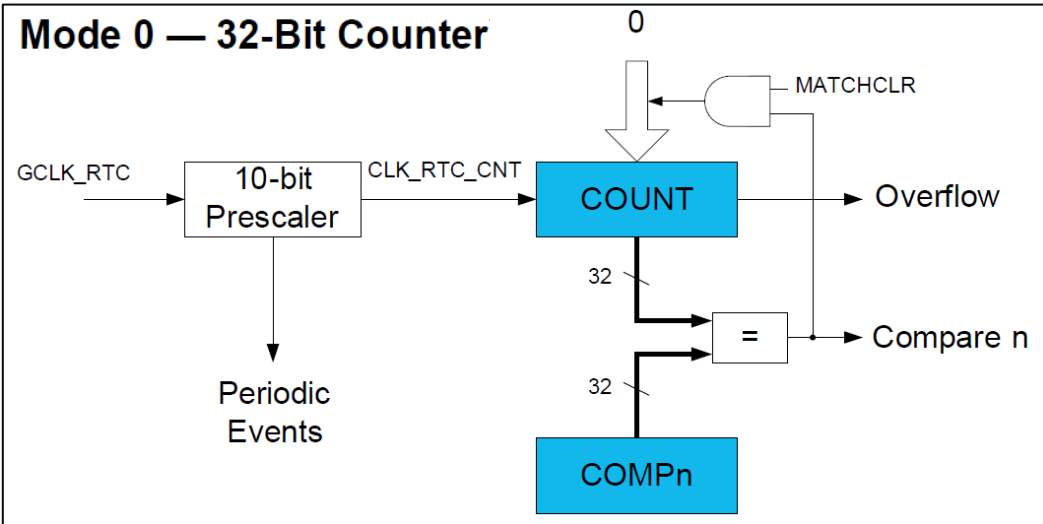
Real Time Counter (RTC)

Real Time Counter (RTC)

- The Real-Time Counter (RTC) is a 32-bit counter with a 10-bit programmable prescaler that typically runs continuously to keep track of time.
- The RTC can wake-up the device from Sleep modes using the alarm/compare wake-up, periodic wake-up, or overflow wake-up mechanisms.
- In **Clock/Calendar mode**, the selected clock source and RTC prescaler **must be configured to provide a 1Hz clock** to the counter for correct operation in this mode.
- Features
 - 32-bit counter with 10-bit prescaler
 - Multiple clock sources
 - 32-bit or 16-bit Counter mode
 - One 32-bit or two 16-bit compare values
 - Clock/Calendar mode
 - Time in seconds, minutes and hours (12/24)
 - Date in day of month, month and year
 - Leap year correction
 - Digital prescaler correction/tuning for increased accuracy
 - Overflow, alarm/compare match and prescaler interrupts and events
 - Optional clear on alarm/compare match

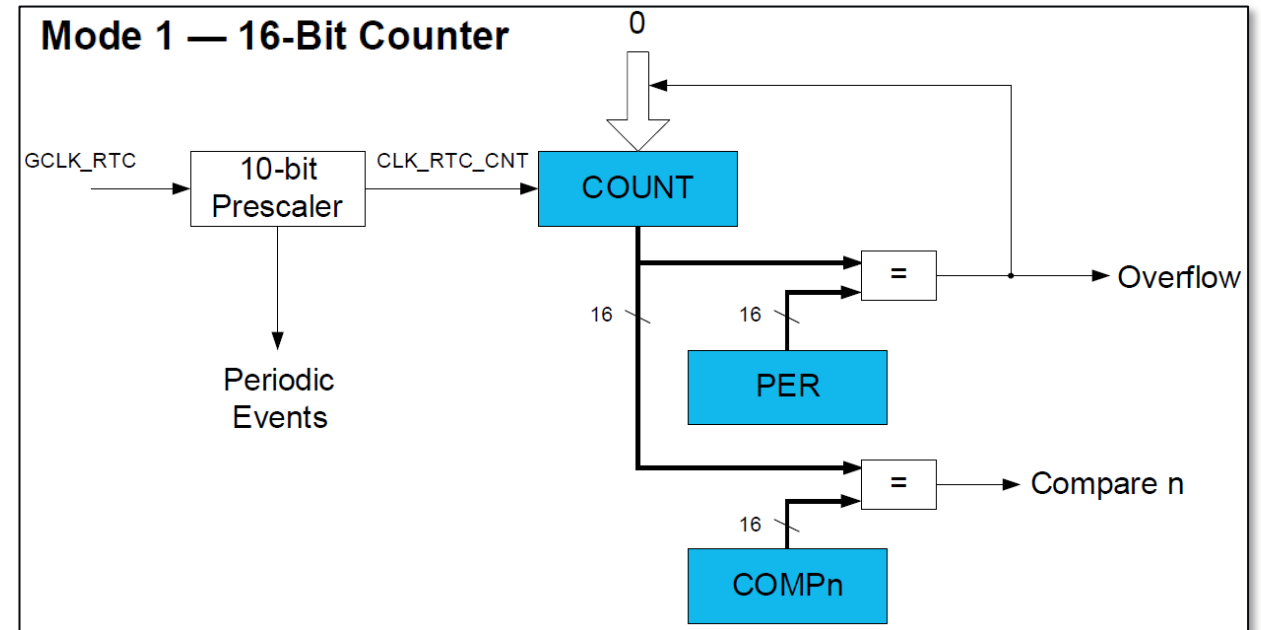
Real Time Counter (RTC)

Real Time Counter (RTC) Operation Mode



The RTC can function in one of these modes:

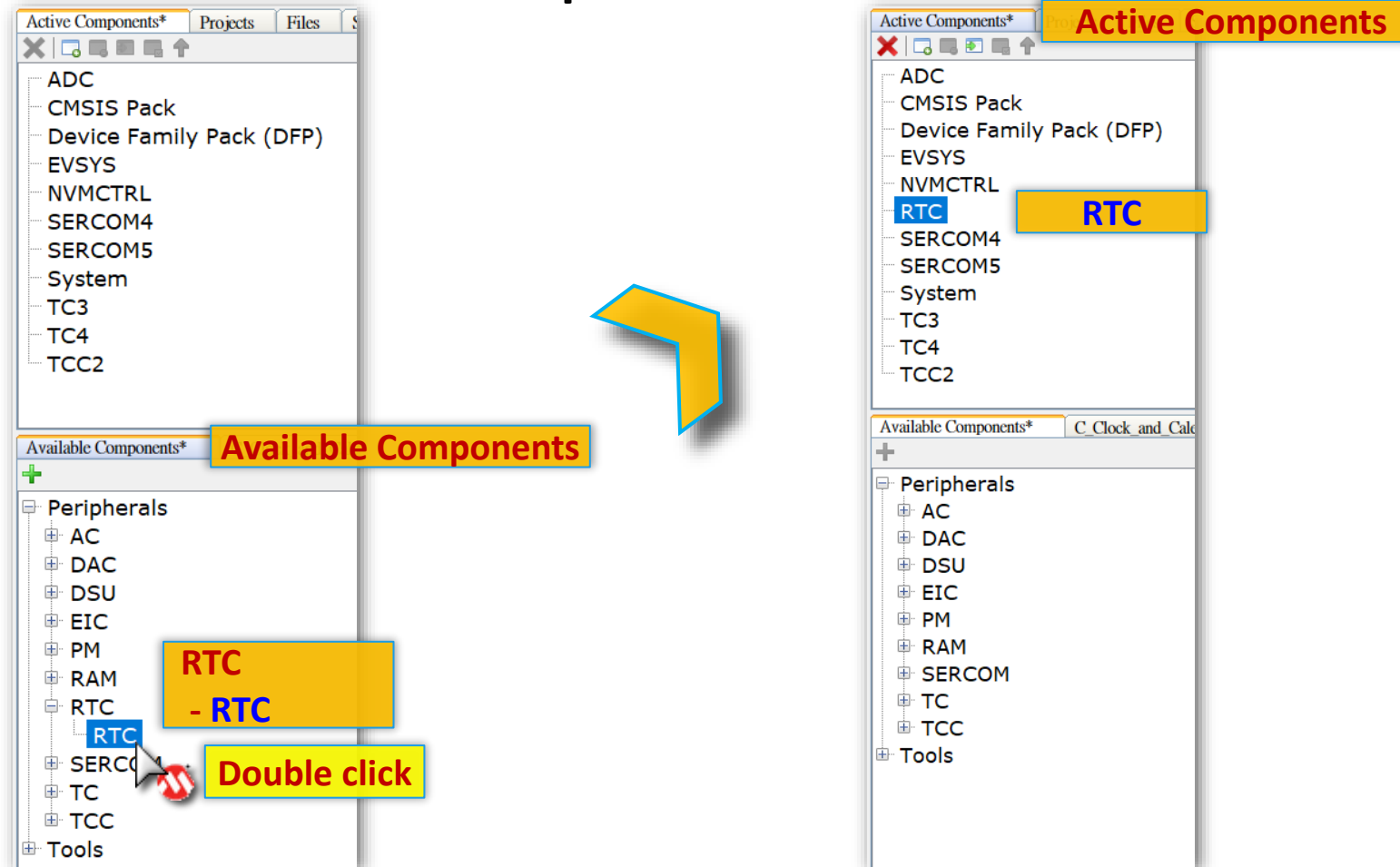
- Mode 0 - COUNT32
RTC serves as 32-bit counter
- Mode 1 - COUNT16
RTC serves as 16-bit counter
- Mode 2 – CLOCK
RTC serves as clock/calendar with alarm functionality



Real Time Counter (RTC)

Add RTC module using MHC

- Find **RTC** component in Available Components window.
- Double click **RTC** to add to Active Components window.



Real Time Counter (RTC)

Configuration Options Example

Mode 0 – 32-Bit Counter

RTC

Hardware Settings

- Generate Frequency Correction API ☐
- Continuous Synchronization for RTC Count/Clock Register ☐

RTC Operation Mode: 32-bit Counter with Single 32-bit Compare

RTC MODE 0 Configuration

- Enable Interrupts? ☒
 - Compare 0 Interrupt Enable ☐
 - Synchronization Ready Interrupt Enable ☐
 - Overflow Interrupt Enable ☐
- RTC Prescaler: DIV1
- Compare Value: 0x0
- Clear on compare Match ☐

RTC EVENTS configuration

- Periodic Interval 0 Event Output Enable ☐
- Periodic Interval 1 Event Output Enable ☐
- Periodic Interval 2 Event Output Enable ☐
- Periodic Interval 3 Event Output Enable ☐
- Periodic Interval 4 Event Output Enable ☐
- Periodic Interval 5 Event Output Enable ☐
- Periodic Interval 6 Event Output Enable ☐
- Periodic Interval 7 Event Output Enable ☐
- Compare 0 Event Output Enable ☐
- Overflow Event Output Enable ☐

Mode 1 – 16-Bit Counter

RTC

Hardware Settings

- Generate Frequency Correction API ☐
- Continuous Synchronization for RTC Count/Clock Register ☐

RTC Operation Mode: 16-bit Counter with Two 16-bit Compares

RTC MODE 1 Configuration

- Enable Interrupts? ☒
 - Compare 0 Interrupt Enable ☐
 - Compare 1 Interrupt Enable ☐
 - Synchronization Ready Interrupt Enable ☐
 - Overflow Interrupt Enable ☐
- RTC Prescaler: DIV1
- 16-bit Timer Period: 0x0
- Compare 0 Value: 0x0
- Compare 1 Value: 0x0

RTC EVENTS configuration

- Periodic Interval 0 Event Output Enable ☐
- Periodic Interval 1 Event Output Enable ☐
- Periodic Interval 2 Event Output Enable ☐
- Periodic Interval 3 Event Output Enable ☐
- Periodic Interval 4 Event Output Enable ☐
- Periodic Interval 5 Event Output Enable ☐
- Periodic Interval 6 Event Output Enable ☐
- Periodic Interval 7 Event Output Enable ☐
- Compare 0 Event Output Enable ☐
- Compare 1 Event Output Enable ☐
- Overflow Event Output Enable ☐

Mode 2 – Clock/Calendar

RTC

Hardware Settings

- Generate Frequency Correction API ☐
- Continuous Synchronization for RTC Count/Clock Register ☐

RTC Operation Mode: Clock/Calendar with Alarm

RTC MODE 2 Configuration

- Enable Interrupts? ☐
 - Alarm 0 Interrupt Enable ☐
 - Synchronization Ready Interrupt Enable ☐
 - Overflow Interrupt Enable ☐
- RTC Prescaler: DIV1
- Reference Year(Leap Year): 2,016

RTC EVENTS configuration

- Periodic Interval 0 Event Output Enable ☐
- Periodic Interval 1 Event Output Enable ☐
- Periodic Interval 2 Event Output Enable ☐
- Periodic Interval 3 Event Output Enable ☐
- Periodic Interval 4 Event Output Enable ☐
- Periodic Interval 5 Event Output Enable ☐
- Periodic Interval 6 Event Output Enable ☐
- Periodic Interval 7 Event Output Enable ☐
- Alarm 0 Event Output Enable ☐
- Overflow Event Output Enable ☐

Real Time Counter (RTC)

Interface Routines

void RTC_Initialize(void);

void RTC_InterruptHandler(void);

- **Mode 2 – Clock/Calendar** **PLIB_RTC_Clock.c**

bool RTC_RTCCTimeSet (struct tm * initialTime);

void RTC_RTCCTimeGet (struct tm * currentTime);

bool RTC_RTCCAlarmSet (struct tm * alarmTime, RTC_ALARM_MASK mask);

void RTC_RTCCallbackRegister (RTC_CALLBACK callback, uintptr_t context);

- **Mode 0 – 32-Bit Counter & Mode 1 – 16-Bit Counter** **PLIB_RTC_Timer.c**

void RTC_Timer32Start (void);

void RTC_Timer32Stop (void);

void RTC_Timer32CounterSet (uint32_t count);

uint32_t RTC_Timer32CounterGet (void);

uint32_t RTC_Timer32PeriodGet (void);

uint32_t RTC_Timer32FrequencyGet (void);

void RTC_Timer32CompareSet (uint32_t compareValue);

void RTC_Timer32InterruptEnable (RTC_TIMER32_INT_MASK interrupt);

void RTC_Timer32InterruptDisable (RTC_TIMER32_INT_MASK interrupt);

void RTC_Timer32CallbackRegister (RTC_TIMER32_CALLBACK callback, uintptr_t context);

void RTC_Timer16Start (void);

void RTC_Timer16Stop (void);

void RTC_Timer16CounterSet (uint16_t count);

uint16_t RTC_Timer16CounterGet (void);

void RTC_Timer16PeriodSet (uint16_t period);

uint16_t RTC_Timer16PeriodGet (void);

uint32_t RTC_Timer16FrequencyGet (void);

void RTC_Timer16Compare0Set (uint16_t comparisonValue);

void RTC_Timer16Compare1Set (uint16_t comparisonValue);

void RTC_Timer16InterruptEnable (RTC_TIMER16_INT_MASK interrupt);

void RTC_Timer16InterruptDisable (RTC_TIMER16_INT_MASK interrupt);

void RTC_Timer16CallbackRegister (RTC_TIMER16_CALLBACK callback, uintptr_t context);

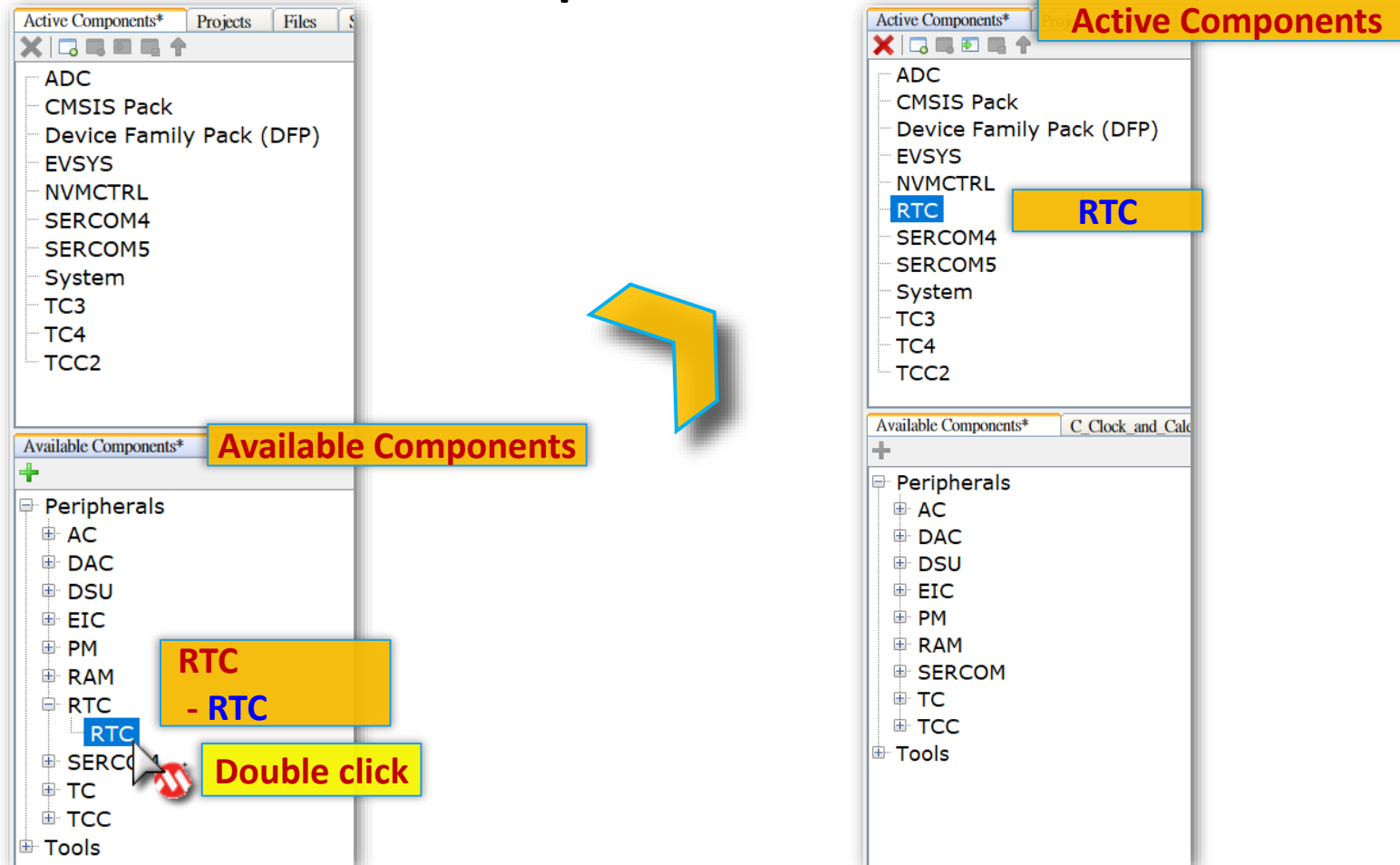
Lab17-1 RTC Clock and Calendar

- Please continue from **SAM2001 Lab17 (SERCOM - SPI SSD1306 OLED)**
 - Try add RTC module to current project.
 - Select **RTC Operation Mode** to **Clock/Calendar with Alarm (Mode2)**.
 - Select RTC Prescaler to **DIV1024**.
 - Generate a **1024Hz** clock through **GCLK1** by divide **32** of **XOSC32K** clock source.
 - Due to change of **GCLK1**, please recalculate **DFPLL 48Mhz clock** by click **Auto Calculate**.
 - Select RTC clock source in **Peripheral Clock Configuration** from **GCLK1** that is **1024Hz**.
 - RTC is **1024Hz** then after prescaler **DIV1024** to get $1024\text{Hz}/1024 = 1\text{Hz}$ Clock tick.
-
- **Let's go !**

⚙️ Lab17-1 RTC Clock and Calendar

Step 1

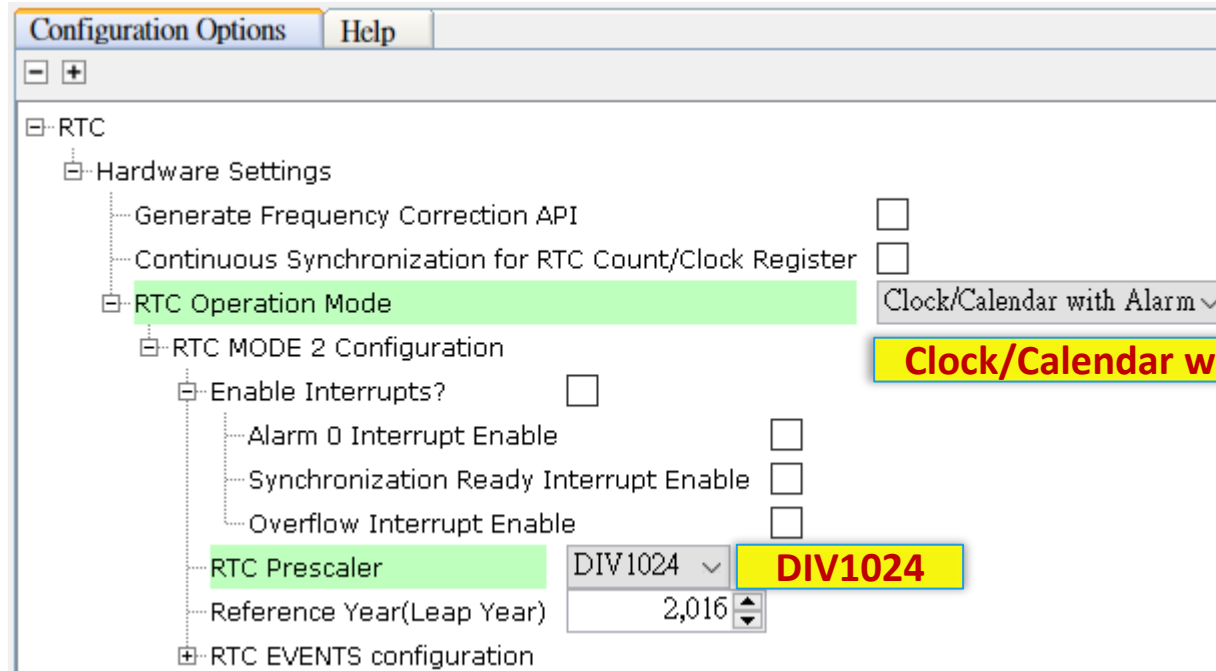
- Find **RTC** component in Available Components window.
- Double click **RTC** to add to Active Components window.



Lab17-1 RTC Clock and Calendar

Step 2

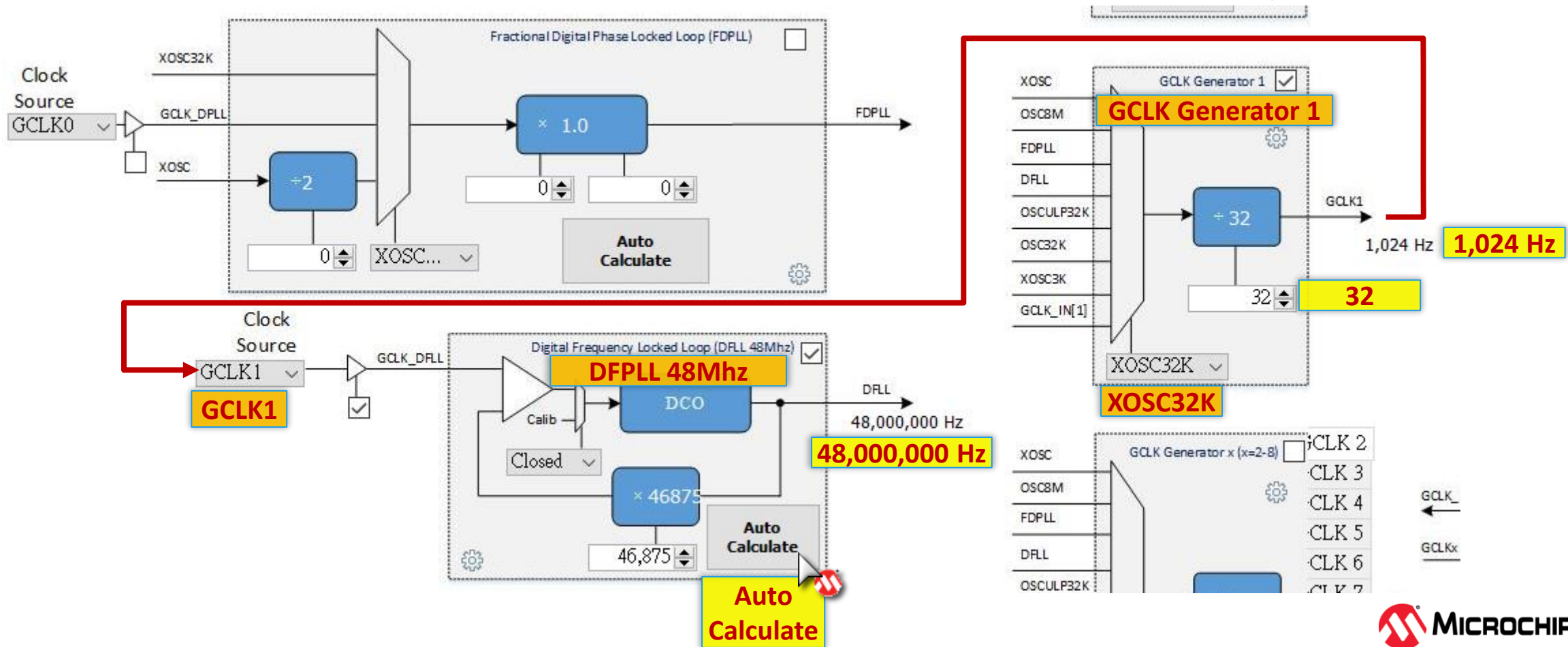
- Select RTC Operation Mode to **Clock/Calendar with Alarm** (Mode2).
- Set RTC Prescaler to **DIV1024**. (We will provide RTC 1024Hz to get **1Hz** clock tick)



Lab17-1 RTC Clock and Calendar

Step 3

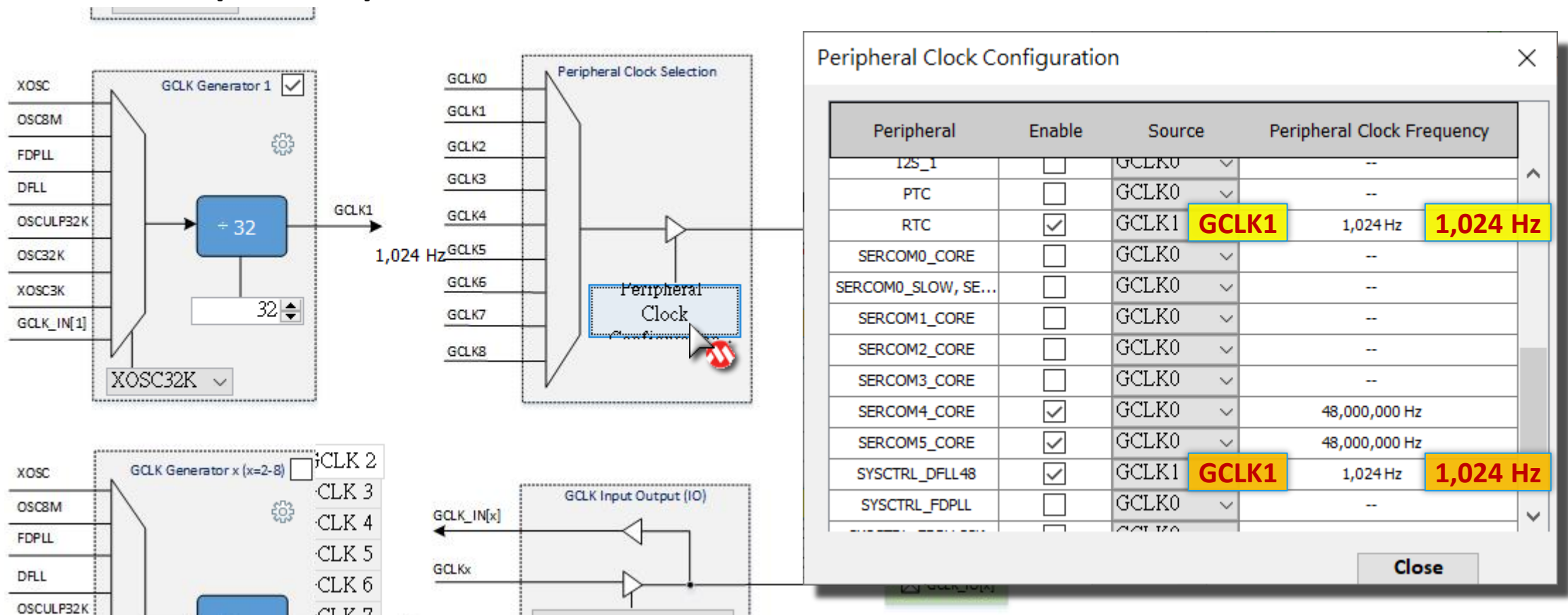
- In MHC → Tools → **Clock Configuration**, Select **GCLK1** to divided by **32**.
- Click on **Auto Calculate** of **DFPLL 48Mhz** to recalculate correct 48MHz output.



⚙️ Lab17-1 RTC Clock and Calendar

Step 4

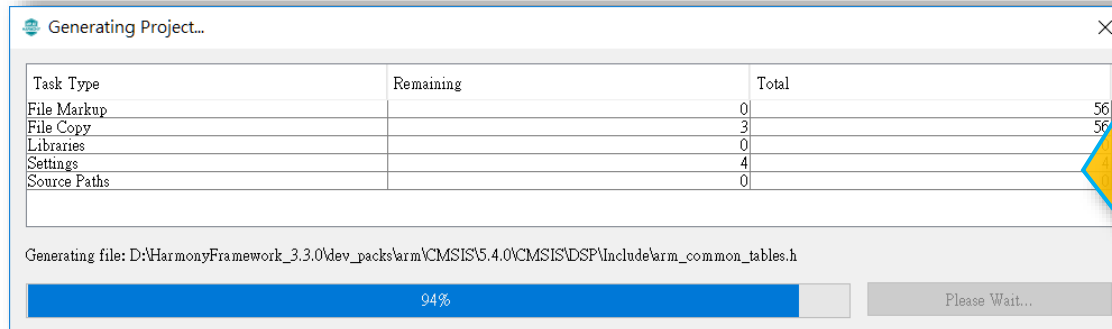
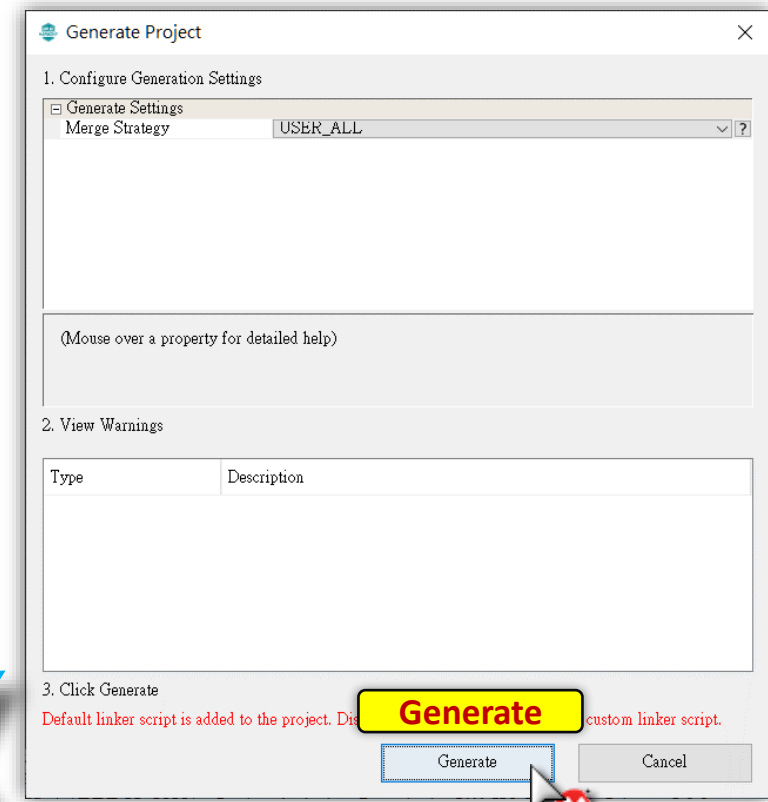
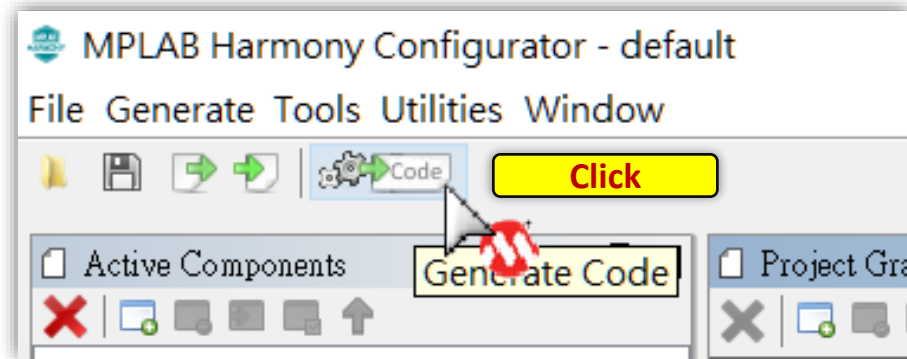
- Click on **Peripheral Clock Configuration** of **Peripheral Clock Selection**.
- Select **GCLK1 (1024Hz)** as source clock of **RTC**.



⚙️ Lab17-1 RTC Clock and Calendar

Step 5

- Click  to Generate Code.



Real Time Counter (RTC)

Time structure in time.h

- Structure containing a calendar date and time broken down into its components.

```
struct tm {  
    int tm_sec;           // Seconds , 0 ~ 59           (Seconds after the minute)  
    int tm_min;           // Minutes , 0 ~ 59           (Minutes after the hour)  
    int tm_hour;          // Hours , 0 ~ 23           (Hours since midnight)  
    int tm_mday;          // Day , 1 ~ 31           (Day of the month)  
    int tm_mon;           // Months , 0 ~ 11        (Months since January)  
    int tm_year;          // Year = tm_year + 1900  (Years since 1900)  
    int tm_wday;          // WeekDay , 0 ~ 6        (Days since Sunday)  
    int tm_yday;          // YearDay , 0 ~ 365      (Days since January 1)  
    int tm_isdst;         // Is daylight Saving?    (Daylight Saving Time flag)  
    long __tm_gmtoff;     // GMT Off  
    const char *__tm_zone; // Time zone  
};
```

Real Time Counter (RTC)

RTC Mode 2 Register

- Implement difference between **RTC Mode2 Register** and **Time structure**

RTC MODE2 CLOCK Register

```
RTC_REGS.MODE2.CLOCK[0:5]      // Seconds , 0 ~ 59
RTC_REGS.MODE2.CLOCK[11:6]     // Minutes , 0 ~ 59
RTC_REGS.MODE2.CLOCK[16:12]    // Hours , 0 ~ 23
RTC_REGS.MODE2.CLOCK[21:17]    // Day , 1 ~ 31
RTC_REGS.MODE2.CLOCK[25:22]    // Months , 1 ~ 12
RTC_REGS.MODE2.CLOCK[31:26]    // Year = [31:26] + [Reference Year]
----- Not implement ----- // WeekDay
----- Not implement ----- // YearDay
----- Not implement ----- // Is daylight Saving?
----- Not implement ----- // GMT Off
----- Not implement ----- // Time zone
```

Time structure

```
(tm_sec)
(tm_min)
(tm_hour)
(tm_mday)
(tm_mon)
(tm_year)
(tm_wday)
(tm_yday)
(tm_isdst)
(__tm_gmtoff)
(__tm_zone)
```

Real Time Counter (RTC)

RTC Mode 2 Register convert between Time structure

- Hamrony Interfaces routines in below will convert between **RTC register** and **struct tm**.

```
bool RTC_RTCCTimeSet ( struct tm * initialTime );
```

```
void RTC_RTCCTimeGet ( struct tm * currentTime );
```

```
bool RTC_RTCCAlarmSet (struct tm * alarmTime, RTC_ALARM_MASK mask);
```

```
/* Reference Year */
#define REFERENCE_YEAR          (2016U)

/* Reference Year in tm structure year (C standard) */
#define TM_STRUCT_REFERENCE_YEAR (1900U)

/* Adjust user year with respect to tm structure year (C Standard) */
#define ADJUST_TM_YEAR(year)      (year + TM_STRUCT_REFERENCE_YEAR)

/* Adjust user month */
#define ADJUST_MONTH(month)       (month + 1)

/* Adjust to tm structure month */
#define ADJUST_TM_STRUCT_MONTH(mon) (mon - 1)
```

```
RTC_REGS.MODE2.CLOCK[31:26] = TM_STRUCT_REFERENCE_YEAR + tm_year - REFERENCE_YEAR
tm_year = RTC_REGS.MODE2.CLOCK[31:26] + REFERENCE_YEAR - TM_STRUCT_REFERENCE_YEAR
```

🔧 Lab17-1 RTC Clock and Calendar

Calculate Weekday

- Zeller's congruence (Use Gregorian calendar)

$$h = \left(q + \left\lfloor \frac{13(m+1)}{5} \right\rfloor + K + \left\lfloor \frac{K}{4} \right\rfloor + \left\lfloor \frac{J}{4} \right\rfloor - 2J \right) \bmod 7$$

- h : The day of the week (0 = Saturday, 1 = Sunday, 2 = Monday, ..., 6 = Friday)
- q : The day of the month.
- m : The month (3 = March, 4 = April, 5 = May, ..., 12 = December, **13 = January**, **14 = February**)
- K : The year of the century ($year \bmod 100$), ex. 19**98** and 20**22** are **98** and **22** respectively.
- J : The zero-based century ($\lfloor year/100 \rfloor$), ex. **19**98 and **20**22 are **19** and **20** respectively.
- $L...J$: The floor function or integer part.
- mod : The modulo operation or remainder after division.

⚙️ Lab17-1 RTC Clock and Calendar

Step 6

```
...
char Alphabet[95] =
"0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz~!@#$$%^&()[]{}_\\"...

// TODO 17-1.01
struct tm RTC_TimeStruct;
const char *monthArray[] =
    {"Jan", "Feb", "Mar", "Apr", "May", "Jun",
     "Jul", "Aug", "Sep", "Oct", "Nov", "Dec" };
const char *weekArray[] =
    {"SUN", "MON", "TUE", "WED", "THR", "FRI", "SAT"};
int Year, Month, Date, Week;
int Hour, Minute, Second;

void RTC_CalculateWeekday( struct tm *pTimeStruct )
{
    // Zeller's congruence to use Gregorian calendar
    int J = (pTimeStruct->tm_year-(pTimeStruct->tm_mon<=2?1:0)+1900) / 100;
    int K = (pTimeStruct->tm_year-(pTimeStruct->tm_mon<=2?1:0)+1900) % 100;
    int m = (pTimeStruct->tm_mon<2?pTimeStruct->tm_mon+13:pTimeStruct->tm_mon+1);
    int q = pTimeStruct->tm_mday;
    int h = (q + (int)((13*((float)m+1))/5) + K + (int)(K/4) + (int)(J/4) - 2*J) % 7;

    pTimeStruct->tm_wday = (h+7-1)%7; // Convert to Time structure format
}
```


Lab17-1 RTC Clock and Calendar

Use C compiler time as Clock Initialize

- Standard Predefined Macros

__FILE__

This macro expands to the name of the current input file, in the form of a C string constant.

__LINE__

This macro expands to the current input line number, in the form of a **decimal integer** constant.

__FILE__ and __LINE__ are useful in generating an error message in line of program.

```
myprintf ( "Exception at line %d of file %s.", __LINE__, __FILE__ );
```

__DATE__

This macro expands to a string constant that describes the date on which the preprocessor is being run. The string constant contains eleven characters and looks like "**Feb 12 1996**".

__TIME__

This macro expands to a string constant that describes the time at which the preprocessor is being run. The string constant contains eight characters and looks like "**23:59:01**".

__DATE__ and __TIME__ could be use for output latest built date and time of current application program.

```
myprintf ( "Current program built at %s, %s.", __TIME__, __DATE__ );
```

Lab17-1 RTC Clock and Calendar

Step 7

```
// TODO 17-1.01
...
void RTC_CalculateWeekday( struct tm *pTimeStruct ) { ... };
void RTC_SetCompilerTime( struct tm *pTimeStruct )
{
    char monthStr[3];
    int year, month, day, hour, min, sec;

    sscanf( __DATE__, "%s %d %d", monthStr, &day, &year );
    sscanf( __TIME__, "%d:%d:%d", &hour, &min, &sec );

    for( month=0 ; month<12 && strcmp(monthStr, monthArray[month]) ; month++ );

    pTimeStruct->tm_year = year-1900;
    pTimeStruct->tm_mon  = month;
    pTimeStruct->tm_mday = day;
    pTimeStruct->tm_hour = hour;
    pTimeStruct->tm_min  = min;
    pTimeStruct->tm_sec  = sec;

    RTC_RTCCTimeSet( pTimeStruct );
    RTC_CalculateWeekday( pTimeStruct );
}

const uint8_t LCM_InitCMD[] = { ... };
```

Lab17-1 RTC Clock and Calendar

Step 8

```
int main ( void )
{
    ...
    // TODO 17-1.02
    RTC_SetCompilerTime( &RTC_TimeStruct );

    while ( true )
    {
        if (TC3_HasExpired)
        {
            ...
            // TODO 17-1.03
            RTC_RTCCTimeGet( &RTC_TimeStruct );
            RTC_CalculateWeekday( &RTC_TimeStruct );
            Year    = RTC_TimeStruct.tm_year+1900;    Month    = RTC_TimeStruct.tm_mon;
            Date    = RTC_TimeStruct.tm_mday;         Week     = RTC_TimeStruct.tm_wday;
            Hour    = RTC_TimeStruct.tm_hour;         Minute  = RTC_TimeStruct.tm_min;
            Second  = RTC_TimeStruct.tm_sec;
            sprintf( (char *)USARTRTxBuffer, "%04d-%s-%02d(%s)",
                    Year, monthArray[Month], Date, weekArray[Week] );
            LCM_DrawString( 0, 2, (char *)USARTRTxBuffer );
            myprintf( "\033[1;1H%s ", USARTRTxBuffer );
            sprintf( (char *)USARTRTxBuffer, "%02d:%02d:%02d", Hour, Minute, Second );
            LCM_DrawString( 0, 3, (char *)USARTRTxBuffer );
            myprintf( "%s", USARTRTxBuffer );

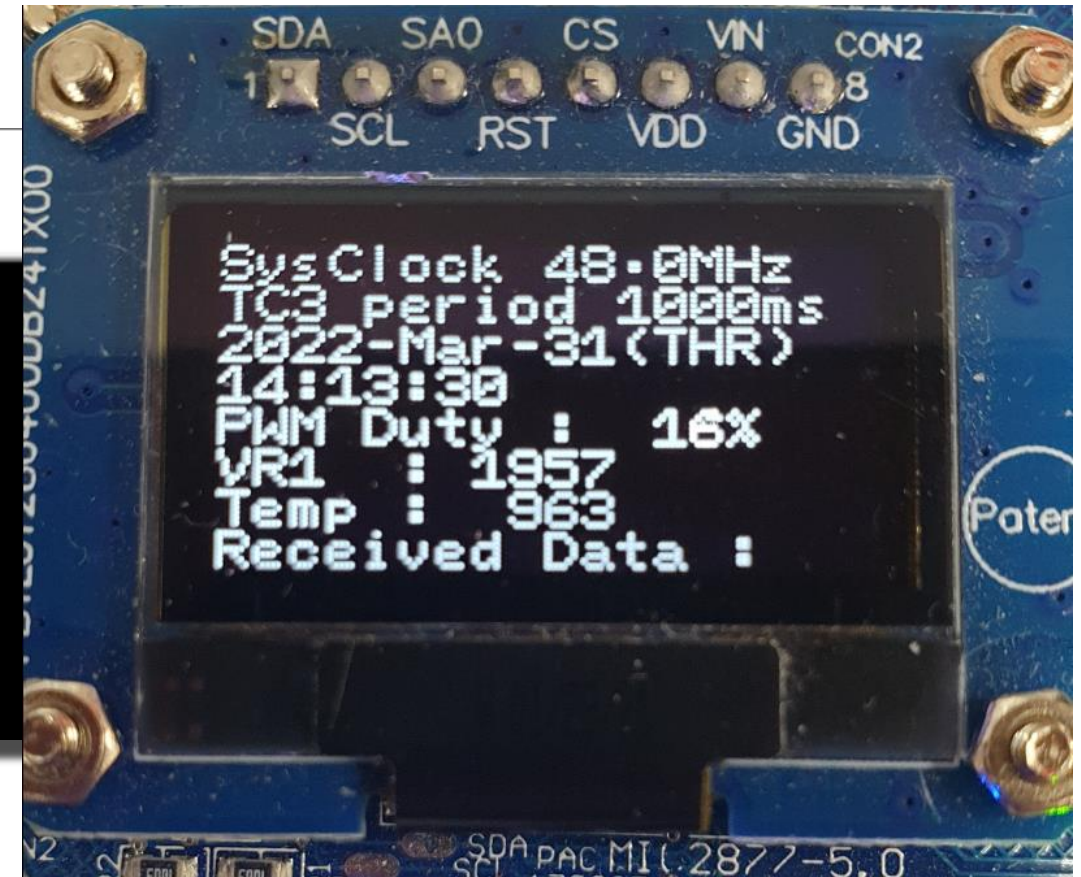
            ADC_ConversionStart();
        }
    }
}
```

🔧 Lab17-1 RTC Clock and Calendar

Result

- Calendar and Clock on OLED to use Compiler Time as initialize.

```
COM5 - Tera Term VT
File Edit Setup Control Window Help
2022-Mar-31(THR) 14:11:30
VR1 Value : 1953
Temperature Value : 960
```



⚙️ Lab17-1 RTC Clock and Calendar

Discuss

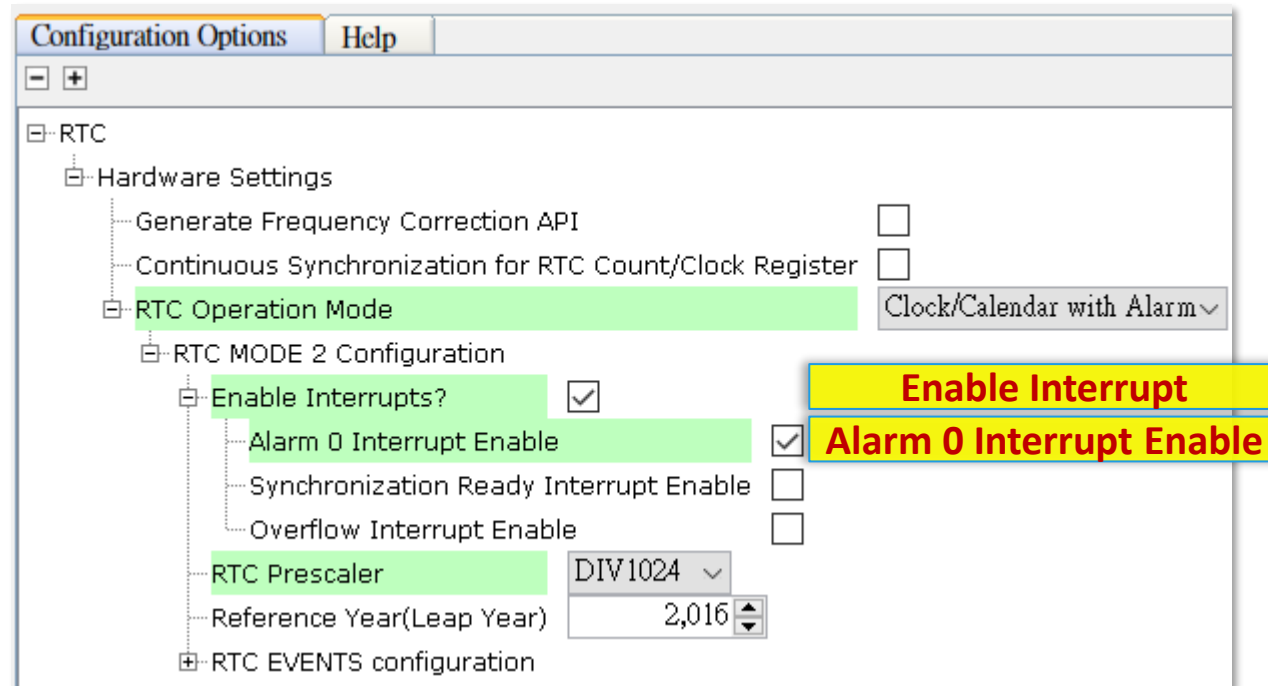
- How to setup an Alarm ?

Please **Enable Interrupt** and set **Alarm 0 Interrupt Enable** in Harmony.

Two extra interface routines will be generated.

```
bool RTCCAlarmSet (struct tm * alarmTime, RTC_ALARM_MASK mask);
```

```
void RTCCallbackRegister ( RTC_CALLBACK callback, uintptr_t context);
```



⚙️ Lab17-1 RTC Clock and Calendar

Discuss

- Implement an Alarm in 10 seconds after boot.

```
int Hour, Minute, Second;
struct tm RTC_AlarmStruct;
volatile int8_t RTC_IsAlarm = 0;
void RTC_AlarmCallback( RTC_CLOCK_INT_MASK intCause, uintptr_t context )
{
    if( intCause & RTC_CLOCK_INT_MASK_ALARM )
        RTC_IsAlarm = 1;
}
...
int main ( void )
{
    ...
    RTC_RTCCallbackRegister( RTC_AlarmCallback, 0 );
    RTC_AlarmStruct = RTC_TimeStruct;
    RTC_AlarmStruct.tm_sec = RTC_AlarmStruct.tm_sec + 10;
    RTC_RTCCAlarmSet( &RTC_AlarmStruct, RTC_ALARM_MASK_YMMDHMMSS );
    while ( true )
    {
        if (TC3_HasExpired)
        {
            ...
            sprintf( (char *)USARTRTxBuffer, "%02d:%02d:%02d %s",
                Hour, Minute, Second, (RTC_IsAlarm?"!!Alarm!!":"") );
            ...
        }
    }
}
```

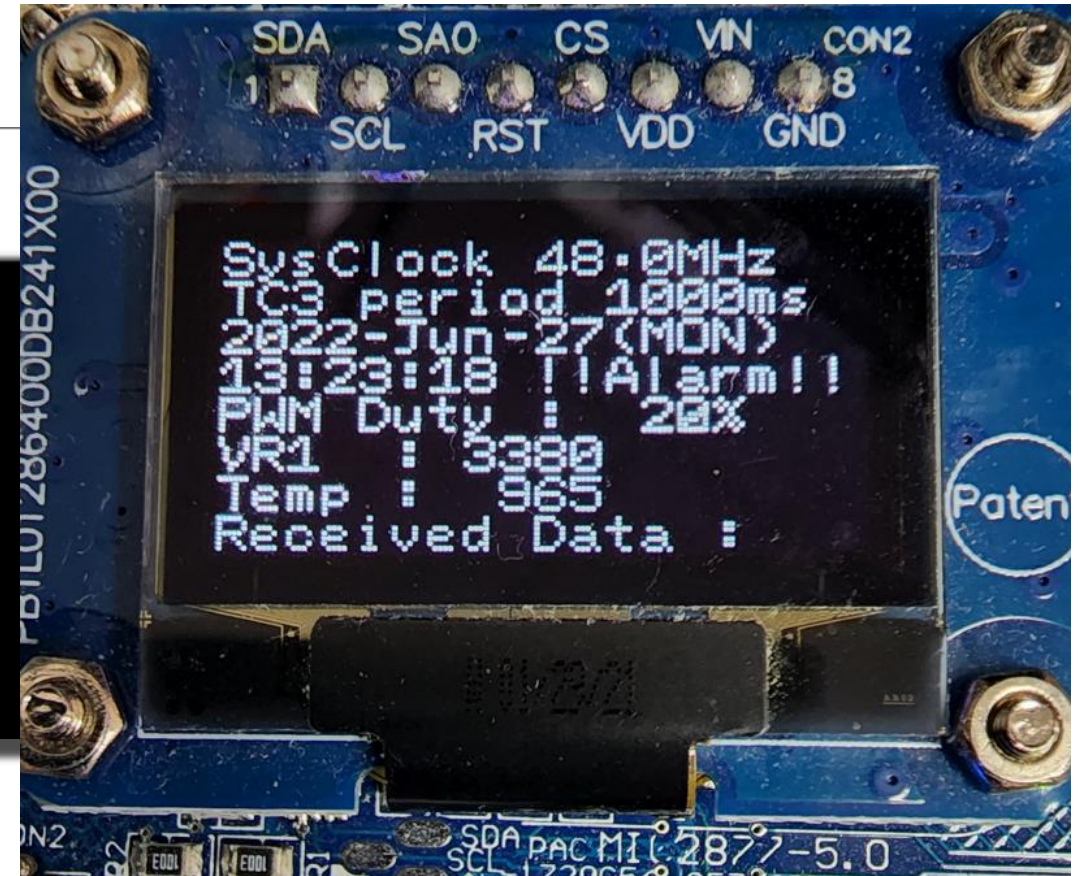
```
typedef enum
{
    RTC_ALARM_MASK_SS = 0x1,           //Alarm every minute
    RTC_ALARM_MASK_MMSS,               //Alarm every Hour
    RTC_ALARM_MASK_HHMMSS,             //Alarm Every Day
    RTC_ALARM_MASK_DDHHMMSS,           //Alarm Every Month
    RTC_ALARM_MASK_MMDDHHMMSS,         //Alarm Every year
    RTC_ALARM_MASK_YMMDHMMSS           //Alarm Once
} RTC_ALARM_MASK;
```


⚙️ Lab17-1 RTC Clock and Calendar

Discuss

- Show the “!!Alarm!!” message after 10 second.

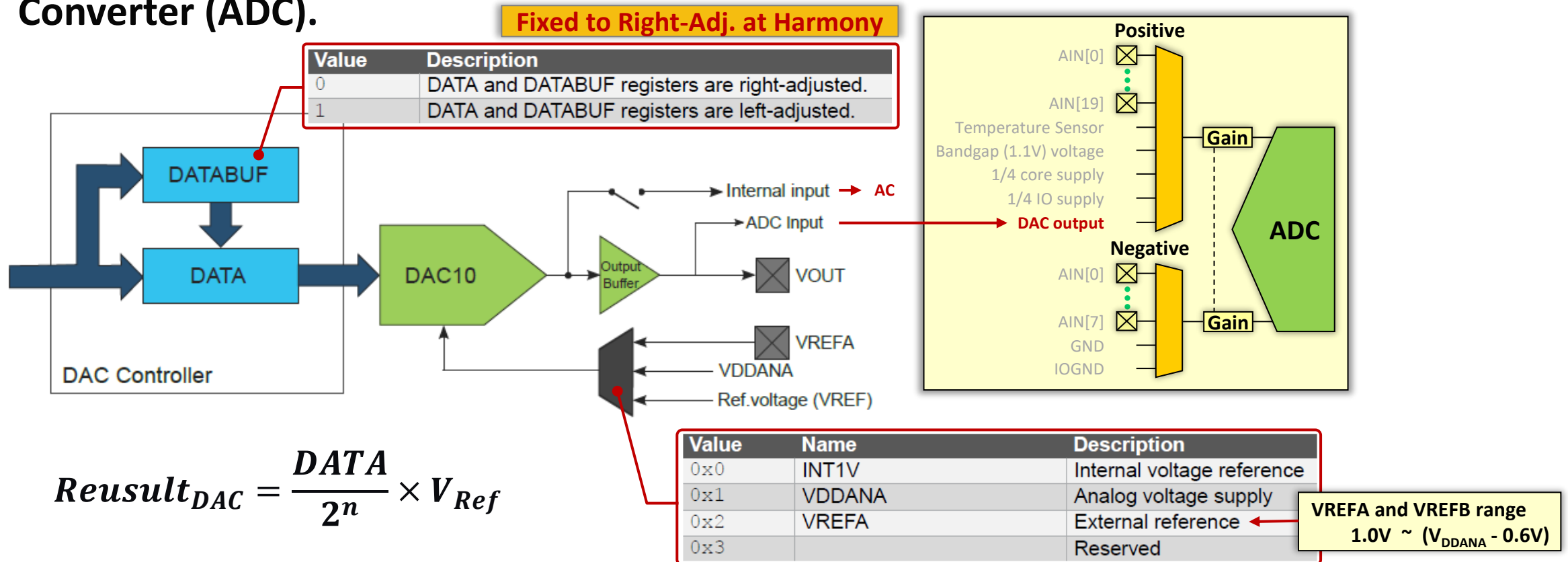
```
COM5 - Tera Term VT
File Edit Setup Control Window Help
2022-Jun-27(MON) 13:23:43 !!Alarm!!
VR1 Value : 3376
Temperature Value : 965
```



DAC Architecture

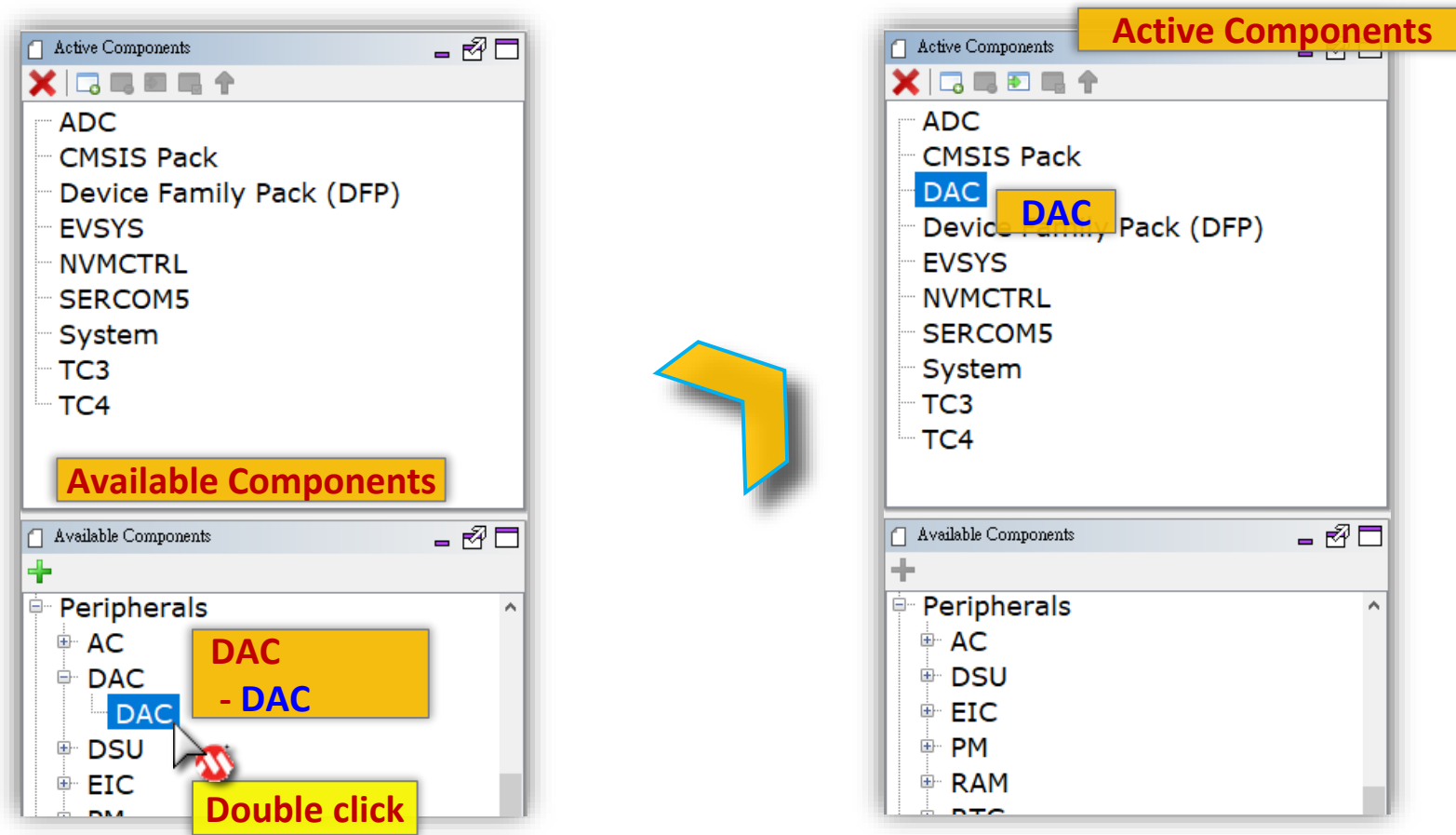
DAC, Digital-to-Analog Converter

- One channel analog output, 10-bits resolution. Up to 350ksps.
- Multiple trigger sources, High-drive capabilities and DMA support.
- Output can be used as input to the Analog Comparator (AC) or Analog-to-Digital Converter (ADC).



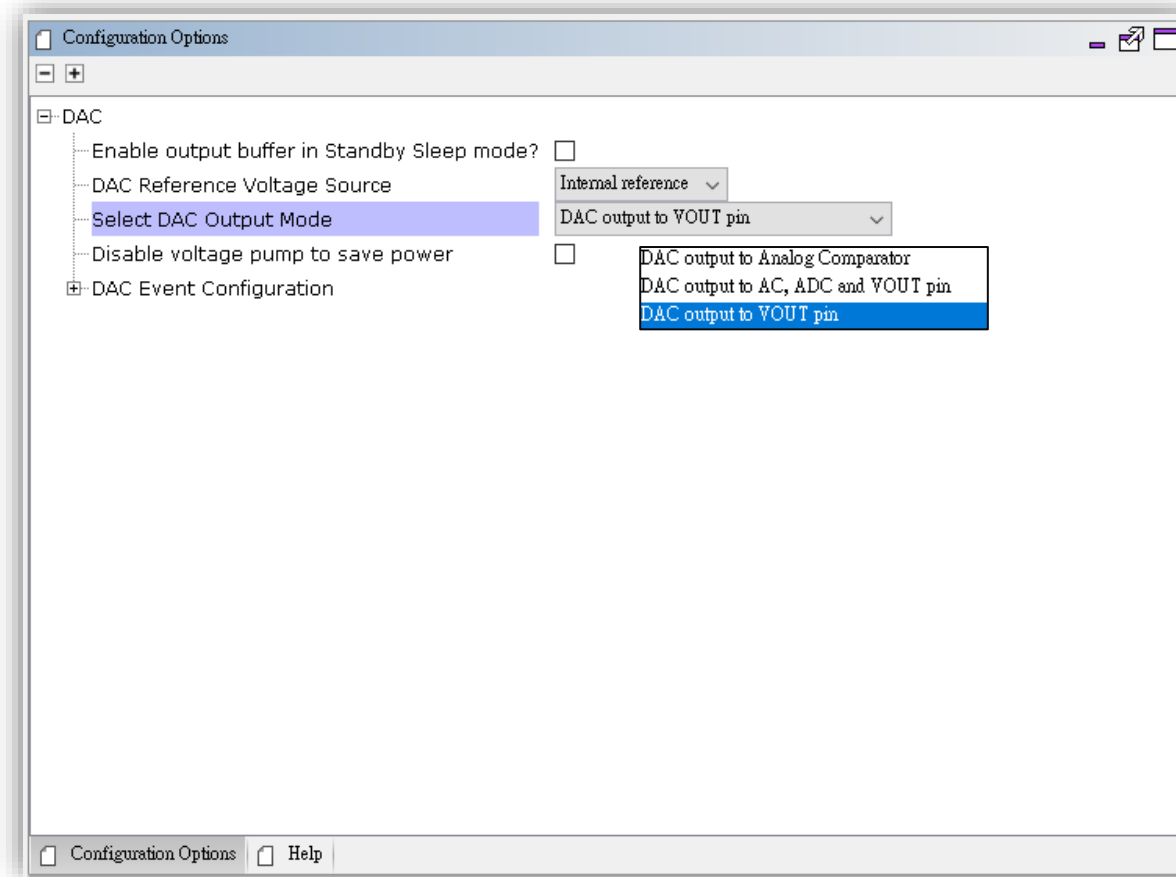
Add DAC Function using MHC

- Find **DAC** component in Available Components window.
- Double click **DAC** to add to Active Components window.



DAC Output

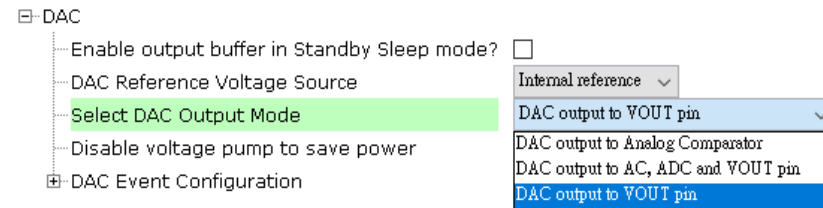
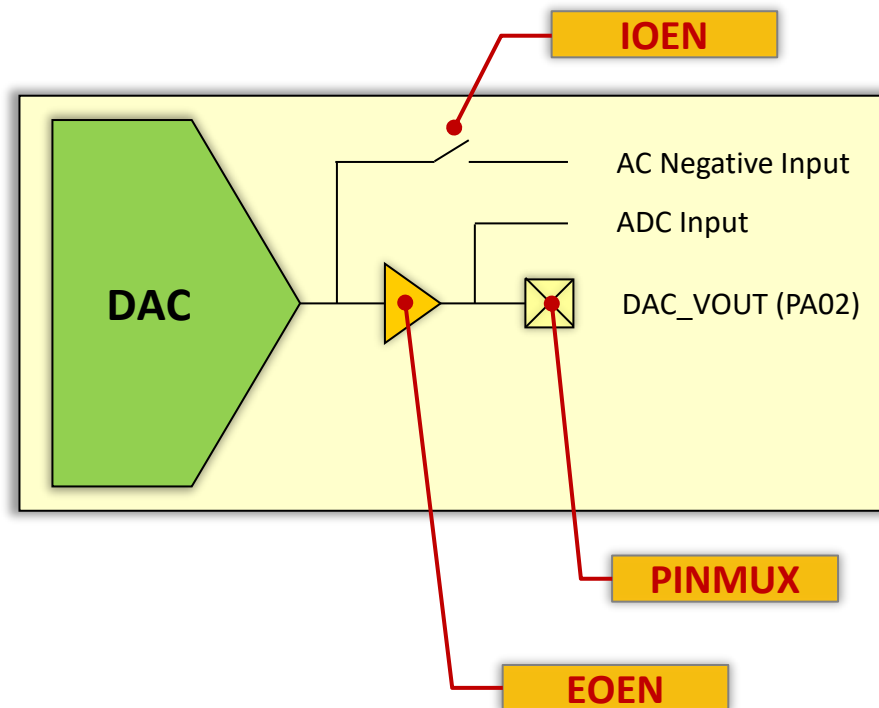
Configuration Options Example



DAC Output

- **The DAC output path is selectable.**

- IOEN : determine that the output is connected or not to internal Analog Compare (AC)
- EOEN : determine that the output buffer is enabled or not to driver to ADC input and VOUT pin.
- PINMUX : determine that the output connected or not to VOUT pin (PA02, fixed).



	IOEN	EOEN
To AC	1	0
To AC, ADC, VOUT	1	1
To VOUT, (ADC)	0	1

DAC Output

PINMUX Configuration Example

- Analog pins will all belong to **PINMUX** function group B.
- DAC output is fixed pin, that assign to PA02.

Pin ⁽¹⁾			I/O Pin	Supply	A	B ⁽²⁾⁽³⁾					C	D	E	F	G	H
SAMD2xE	SAMD2xG	SAMD2xJ			EIC	REF	ADC	AC	PTC	DAC	SERCOM ⁽²⁾⁽³⁾	SERCOM-ALT	Tc ⁽⁴⁾ /TCC	TCC	COM	AC/ GCLK
1	1	1	PA00	VDDANA	EXTINT[0]							SERCOM1/ PAD[0]	TCC2/VO[0]			
2	2	2	PA01	VDDANA	EXTINT[1]							SERCOM1/ PAD[1]	TCC2/VO[1]			
3	3	3	PA02	VDDANA	EXTINT[2]		AIN[0]		Y[0]	VOUT				TCC3/ TCC[0]		

PA02 **VOUT**

Pin Table		Package: TQFP48													
		PA00	PA01	DAC_VO.	PA03	NDANA	DDANA	B08	B09	A04	A05	A06	A07	PA08	PA09
Module	Function	1	2	3	4									13	14
	ADC_AIN4/VREFP														
	ADC_AIN5														
	ADC_AIN6														
	ADC_AIN7														
DAC	DAC_VOUT														
	DAC_VREFP														
	EIC_EXTINT0														

PA02 (DAC VOUT) **DAC**

DAC Functions & Code Example

Interface Routines & Waveform Output Example

void DAC_Initialize(void);

void DAC_DataWrite(uint16_t data);

```
...  
int main (void)  
{  
    SYS_Initialize ( NULL );  
    ...  
    while( true )  
    {  
        ...  
        if (TCx_HasExpired)  
        {  
            TCx_HasExpired = 0;  
            DAC_DataWrite( WaveArray[Index++] );  
            if ( Index >= ArraySize )  
                Index = 0;  
        }  
    }  
}
```

⚡ Lab12-2 DAC Output to ADC Measurement

- Please continue from [SAM2001 Lab12 \(ADC Polling Software Trigger\)](#)
- Try to add **DAC** module to project.
- Initial DAC to output 1.65V to **VOUT(PA02)** and ADC.
 - Voltage Reference : **AVCC** (3.3V at APP045)
 - DAC Output : **DAC output to VOUT pin**
- **Let's go !**

**Reference*

	IOEN	EOEN
To AC	1	0
To AC, ADC, VOUT	1	1
To (ADC), VOUT	0	1

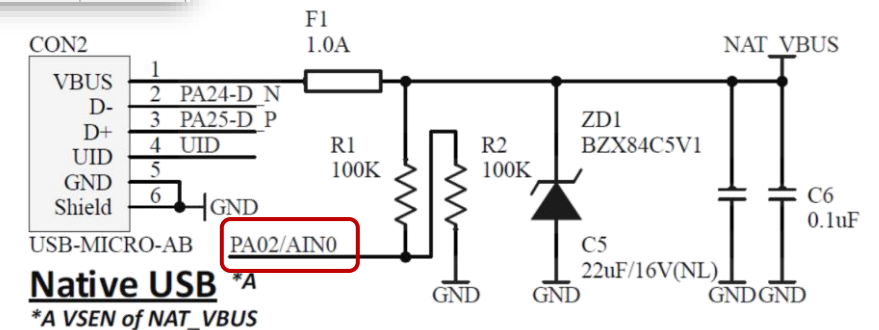
$$Result_{DAC} = \frac{DATA}{2^n} \times V_{Ref}$$

⚠ Lab12-2 DAC Output to ADC Measurement

Caution !!

- PA02 was connected to R1 and R2, that use for detected the Native USB was attached or not. So, Please don't Power by Native USB at this lab. Power by Native USB cause to MCU burn-out possible.

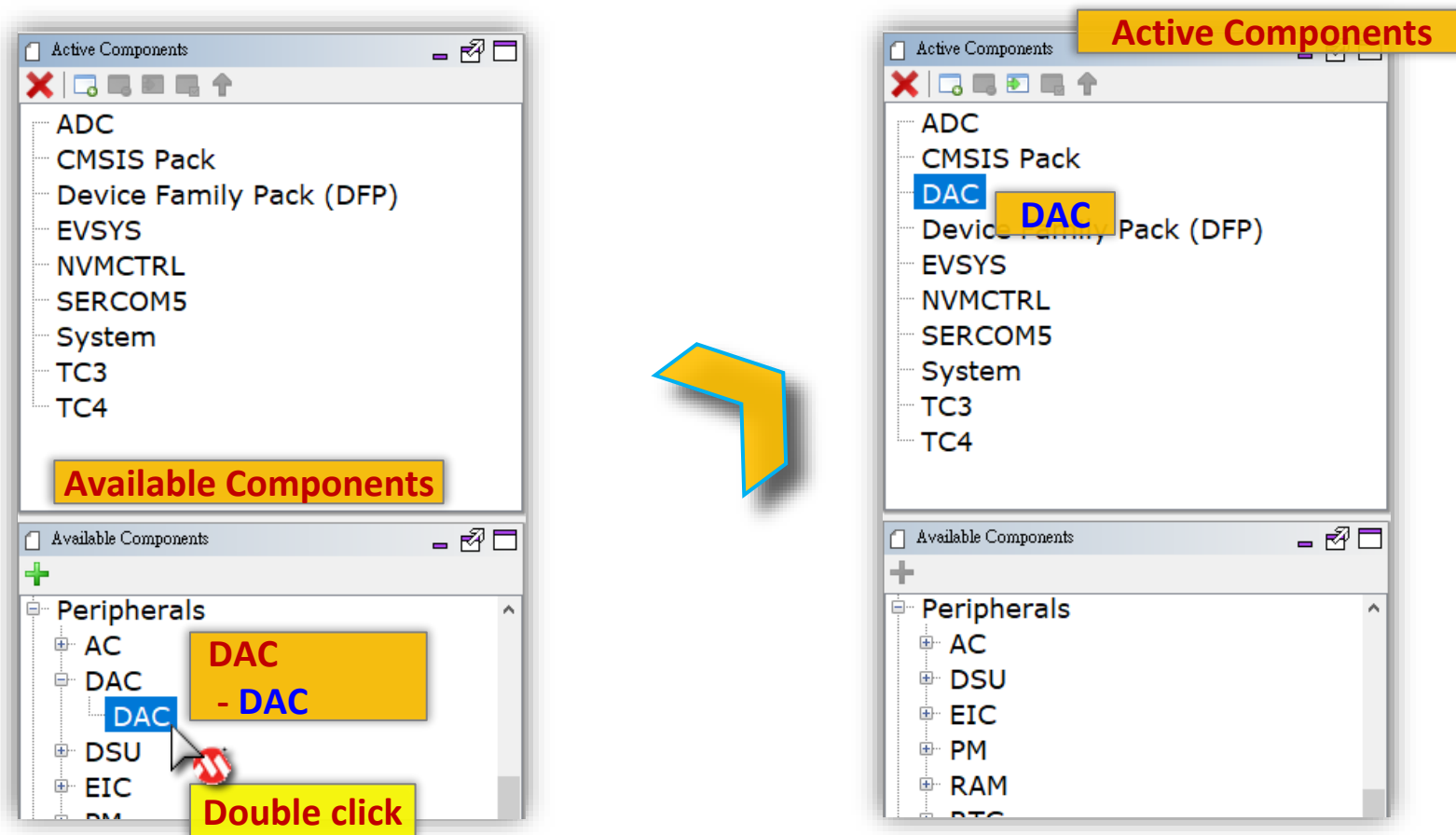
Pin ⁽¹⁾			I/O Pin	Supply	A	B ⁽²⁾⁽³⁾					C	D	E	F	G	H
SAMD2xE	SAMD2xG	SAMD2xJ			EIC	REF	ADC	AC	PTC	DAC	SERCOM ⁽²⁾⁽³⁾	SERCOM-ALT	TC ⁽⁴⁾ /TCC	TCC	COM	AC/ GCLK
1	1	1	PA00	VDDANA	EXTINT[0]							SERCOM1/ PAD[0]	TCC2/WO[0]			
2	2	2	PA01	VDDANA	EXTINT[1]							SERCOM1/ PAD[1]	TCC2/WO[1]			
3	3	3	PA02	VDDANA	EXTINT[2]		AIN[0]		Y[0]	VOUT				TCC3/ WO[0]		



⚡ Lab12-2 DAC Output to ADC Measurement

Step 1

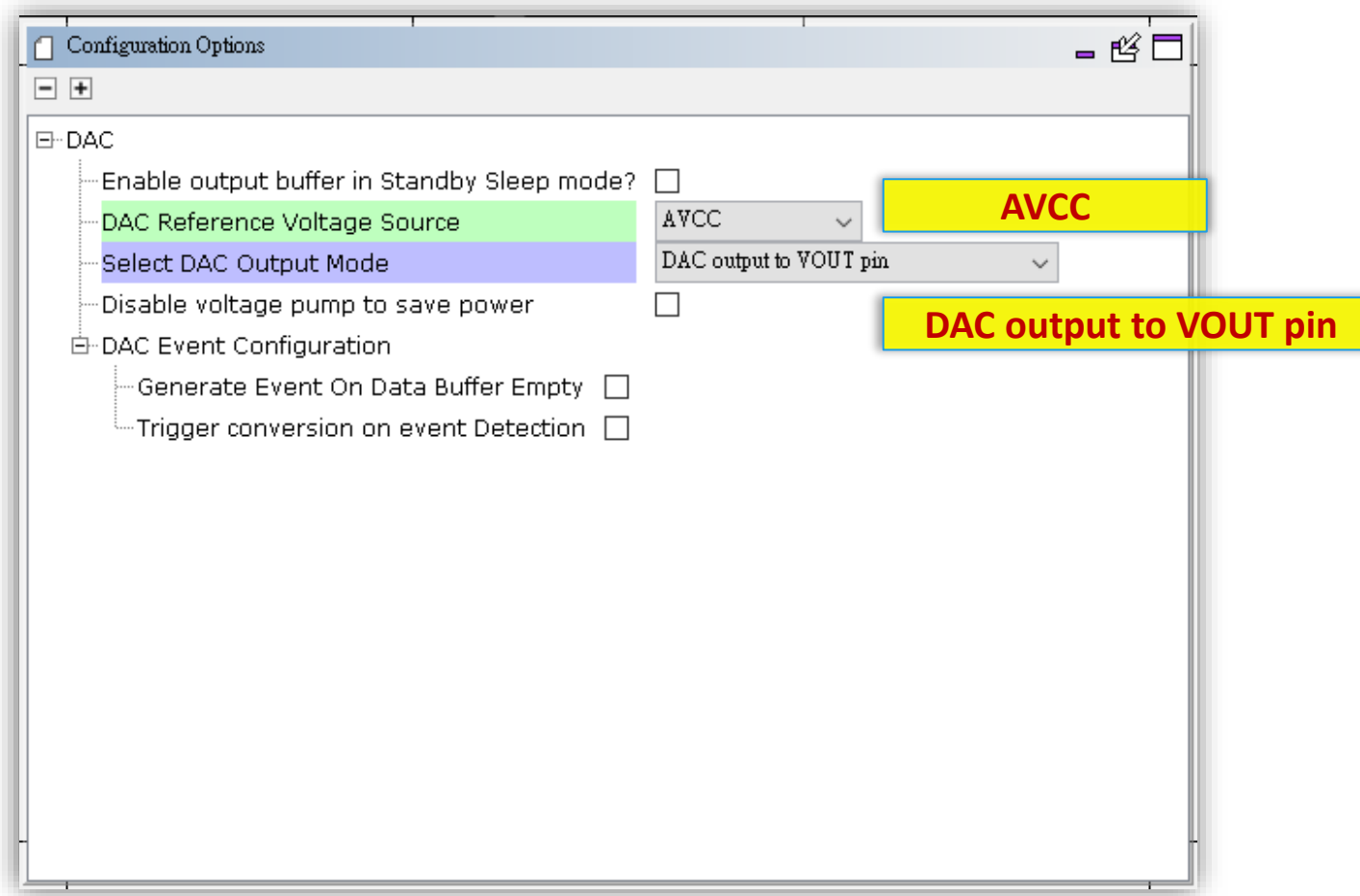
- Find **DAC** component in Available Components window.
- Double click **DAC** to add to Active Components window.



⚙️ Lab12-2 DAC Output to ADC Measurement

Step 2

- Configure **DAC** in configuration.



⚙️ Lab12-2 DAC Output to ADC Measurement

Step 3

- Enable DAC VOUT (DAC_VOUT, PA02)

The image shows two screenshots of the 'Pin Table' window for the TQFP48 package, illustrating the process of enabling DAC_VOUT on PA02.

Top Screenshot: The 'DAC_VOUT' function is selected in the 'Function' column. The 'PA02' pin is highlighted in blue. A yellow callout box with the text 'Click Enable' points to the 'PA02' cell. A red box highlights the 'DAC_VOUT' function. A yellow box labeled 'DAC' is positioned to the left of the table.

		PA00	PA01	PA02	PA03	GNDANA	VDDANA	PB08	PB09	PA04	PA05	PA06	PA07	PA08	PA09	PA10	PA11	VDDIO	GNDIO	PB10
Module	Function	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
DAC	DAC_VOUT																			
	DAC_VREFP																			

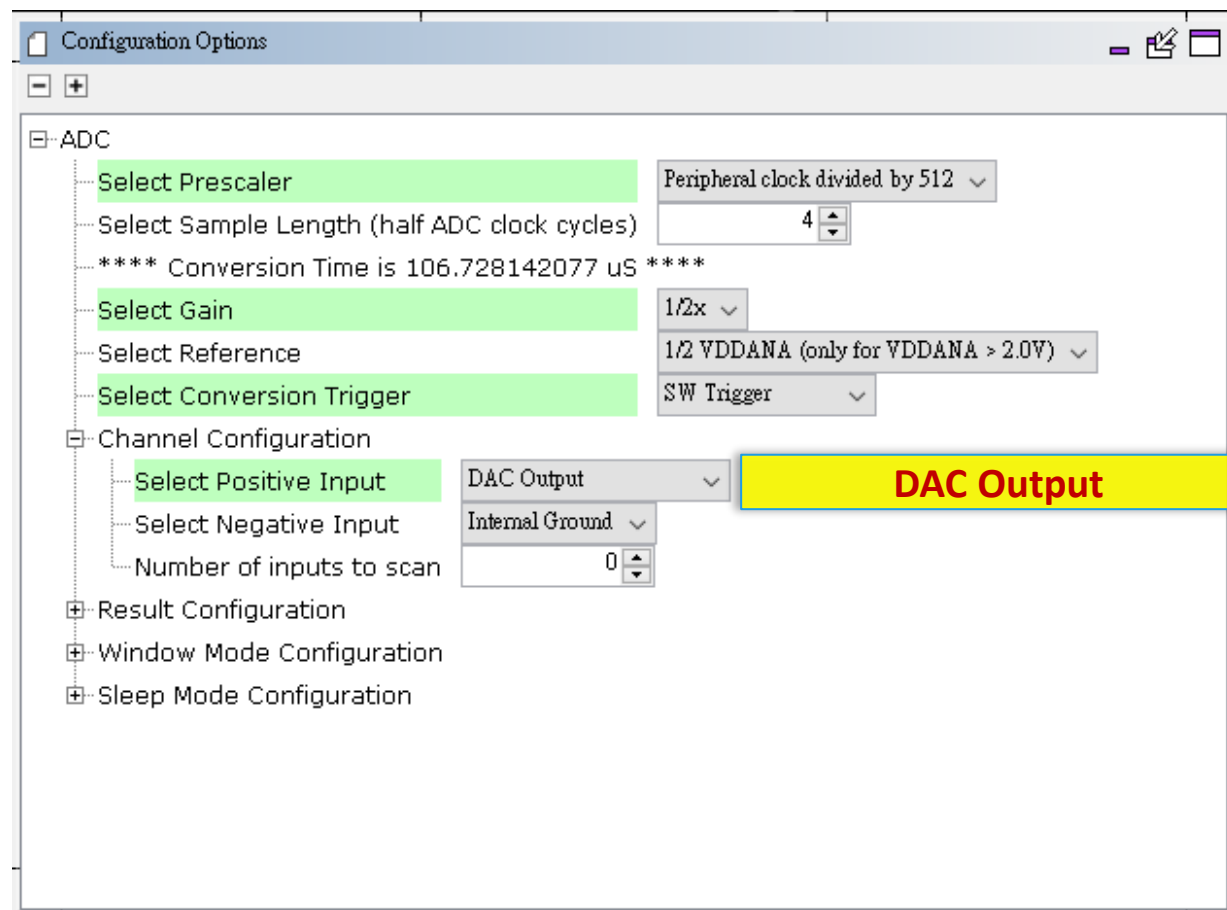
Bottom Screenshot: The 'DAC_VOUT' function is selected in the 'Function' column. The 'PA02' pin is highlighted in green, indicating it is enabled. A red box highlights the 'DAC_VOUT' function. A yellow box labeled 'DAC' is positioned to the left of the table.

		PA00	PA01	DAC_VOUT	PA03	GNDANA	VDDANA	PB08	PB09	PA04	PA05	PA06	PA07	PA08	PA09	PA10	PA11	VDDIO	GNDIO	PB10
Module	Function	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
DAC	DAC_VOUT																			
	DAC_VREFP																			

⚙️ Lab12-2 DAC Output to ADC Measurement

Step 4

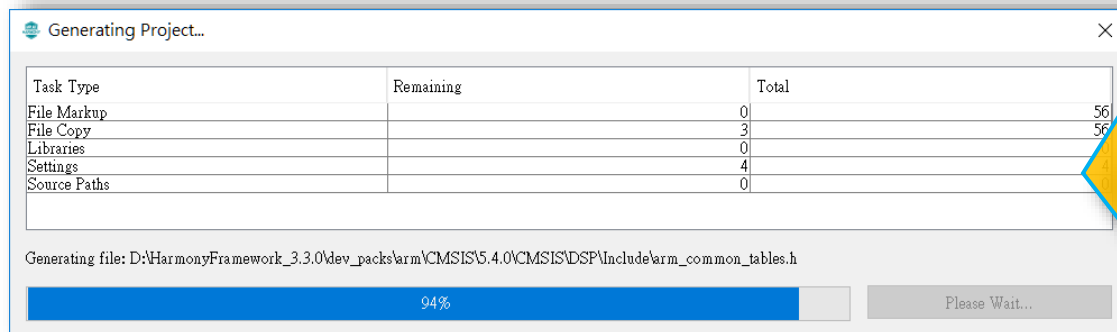
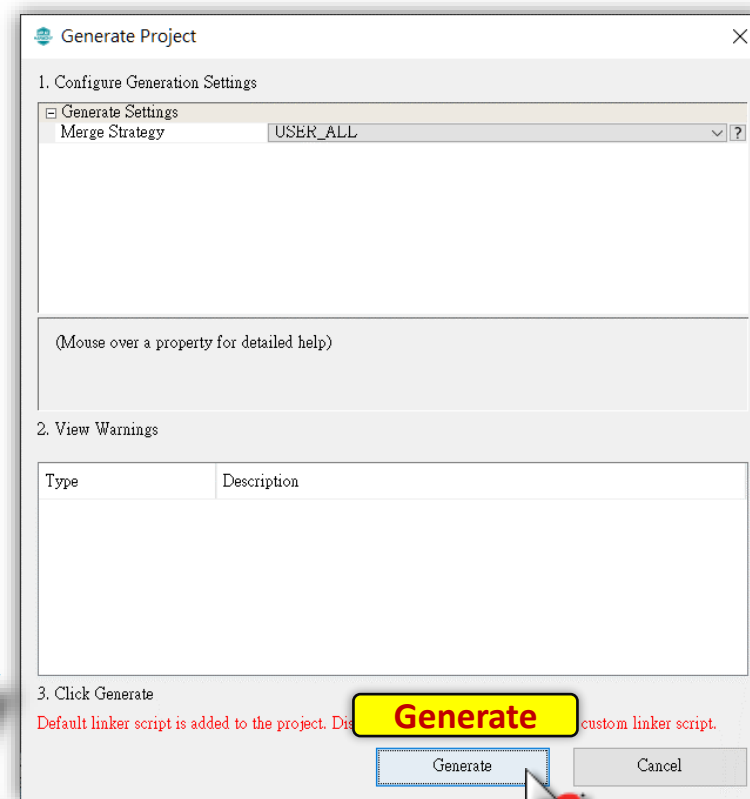
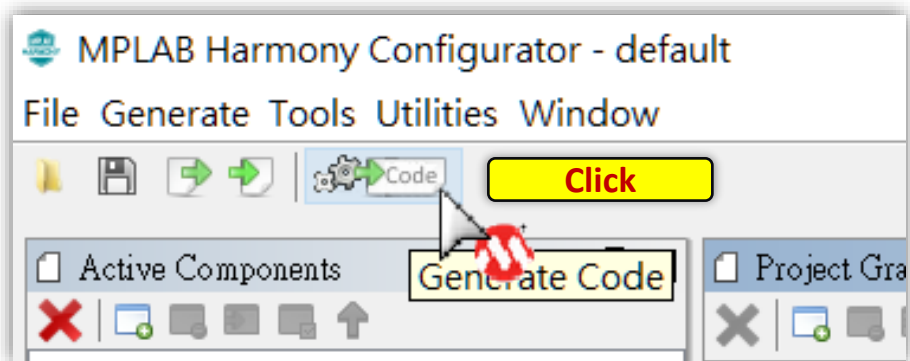
- Change **ADC** positive channel to “DAC Output” in ADC configuration.



⚙️ Lab12-2 DAC Output to ADC Measurement

Step 5

- Click  to Generate Code.



Lab12-2 DAC Output to ADC Measurement

Step 6

```
int main (void)
{

    SYS_Initialize(NULL);
    ...

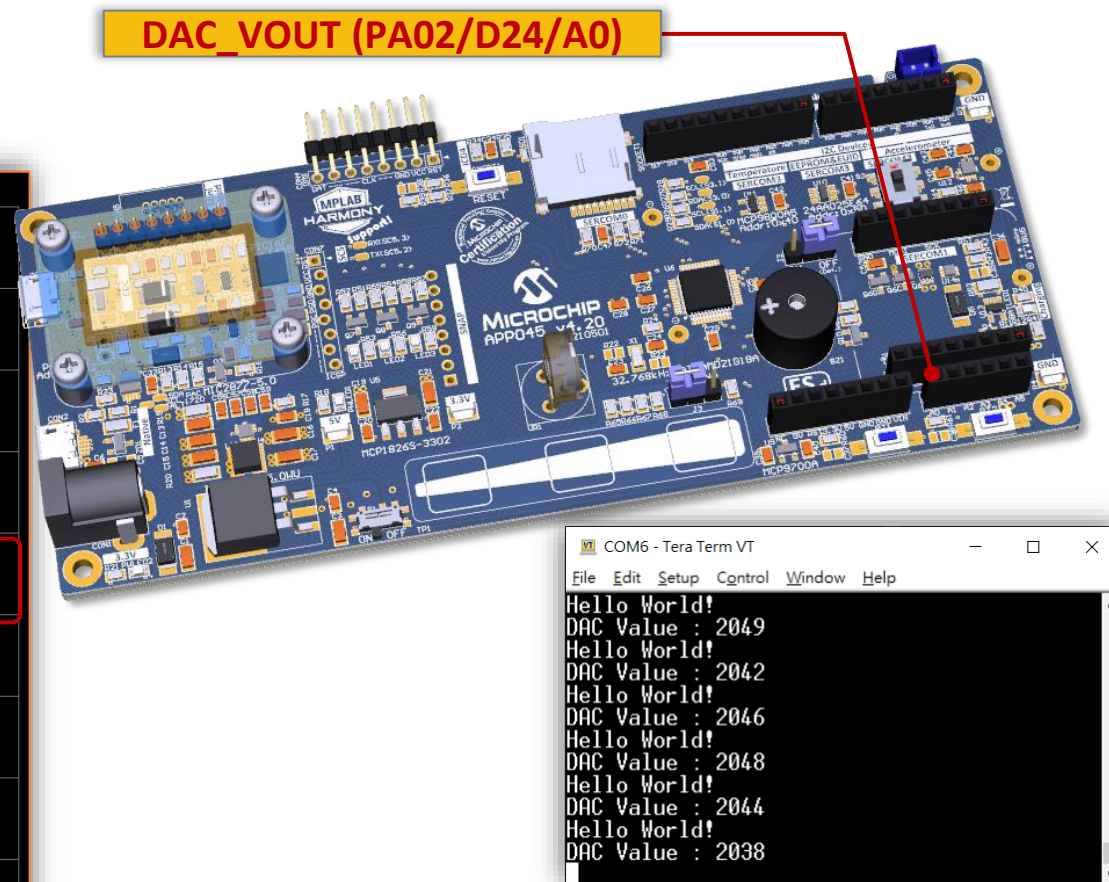
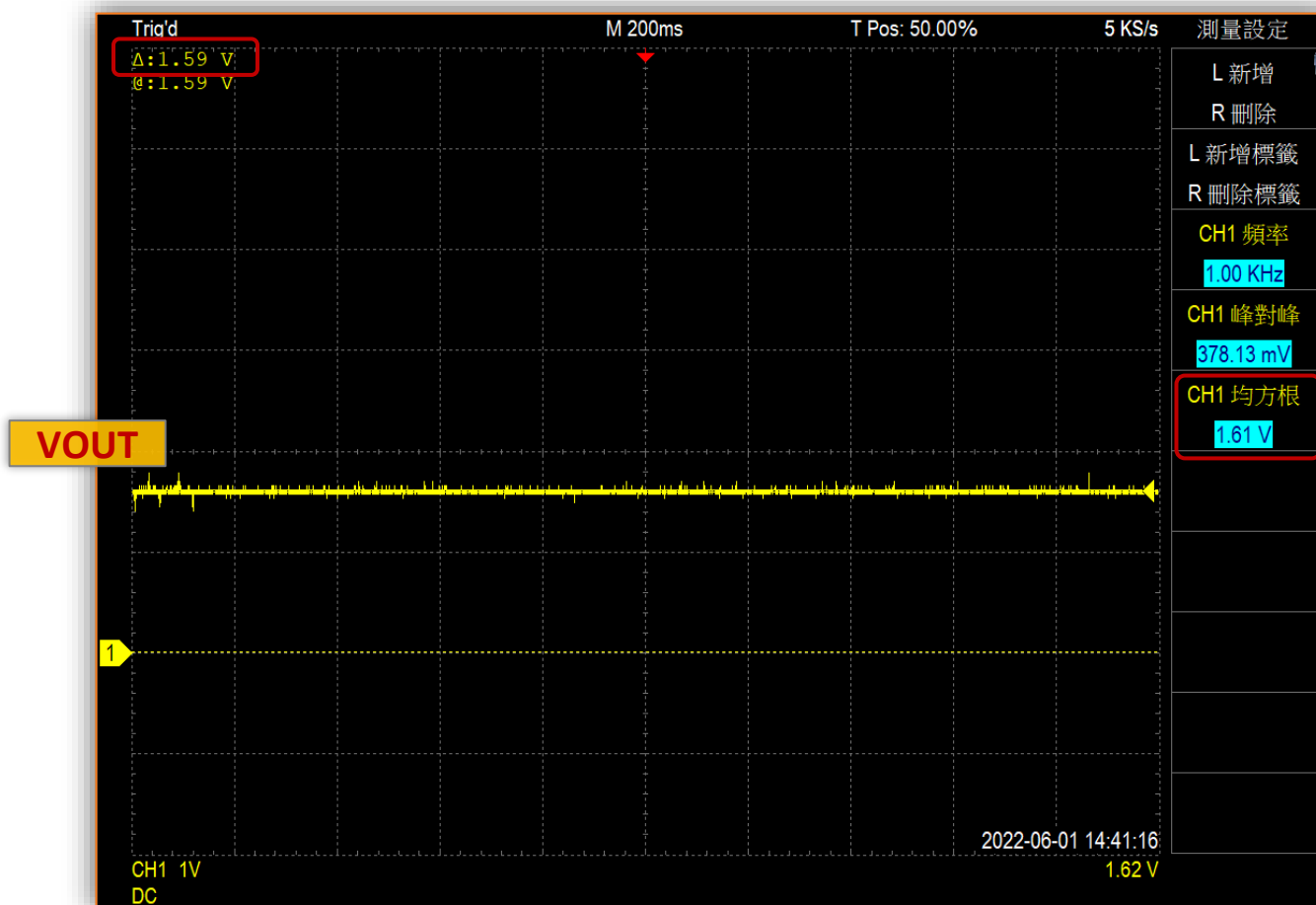
    // TODO 12-2.01
    DAC_DataWrite = ( (uint16_t) 511 );
    ADC_Enable();

    while ( true )
    {
        SYS_Tasks();
        ...
        if (TC3_HasExpired)
        {
            ...
            // TODO 12-2.02
            //myprintf("VR1 Value : %d\r\n", ADC_Result);
            myprintf("DAC Value : %d\r\n", ADC_Result);
        }
    }
    return ( EXIT_FAILURE);
}
```

Lab12-2 DAC Output to ADC Measurement

Result

- Measure the waveform at DAC_VOUT.



$$\text{Reusult}_{\text{Volaage}} = \left(\frac{\text{Reusult}_{\text{AD}} + 1}{2^n} \times V_{\text{Ref}+} - V_{\text{Ref}-} \right)$$

$$1.647\text{V} = \left(\frac{2044 + 1}{4096} \times 3.3\text{V} \right)$$

⚙️ Lab12-2 DAC Output to ADC Measurement

- **Discuss**

- How to generate a sine wave use DAC module ? How to control the frequency of waveform ?

```
#include <math.h>
```

```
for (i = 0; i < DAC_Array_Size; i++)  
{  
    Radas = (( (-180) + (AngleDistance * i)) * 3.14159) / 180.0;  
    Wave_Array[i] = (uint16_t) (sin(Radas) * 512.0 + 511.0);  
}
```

Use sin() function to generate the sine wave array.

```
if (TCx_HasExpired) /* Period Control */  
{  
    TCx_HasExpired = 0;  
    DAC_DataWrite(Wave_Array[Index]);  
    if (++DACIndex >= DAC_Array_Size)  
        DACIndex = 0;  
}
```

Use TCx to control output period.

- Please Refer Lab 12-3 DAC Wave Output for details.

Lab12-3 DAC Wave Output

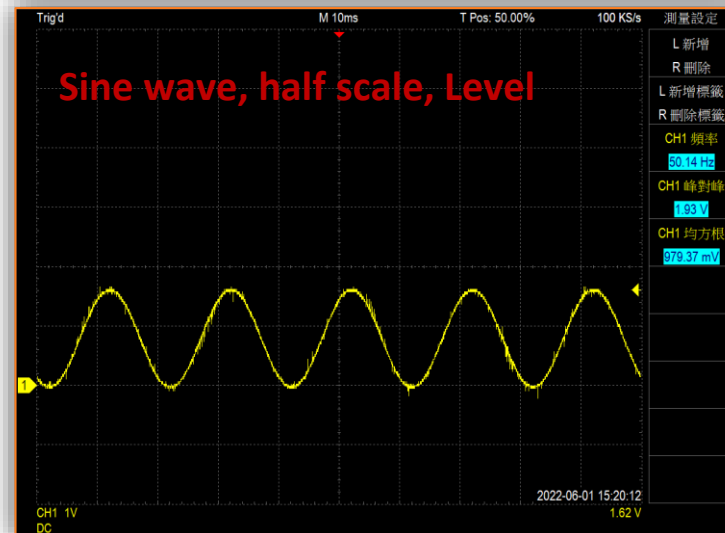
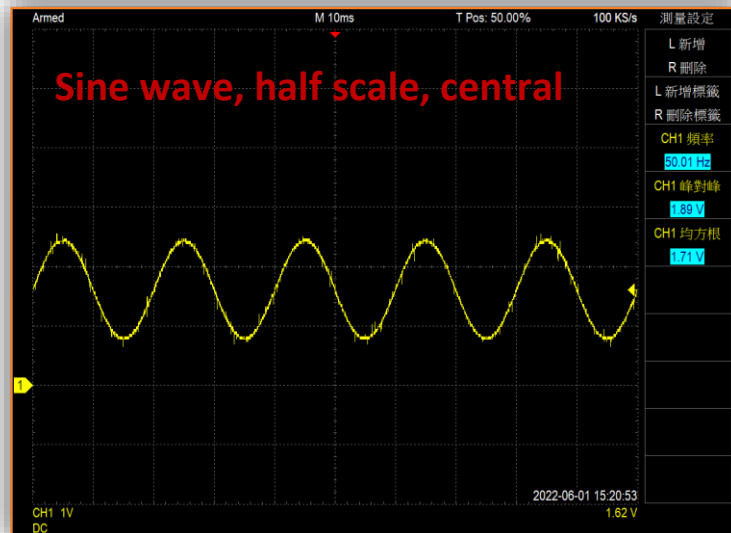
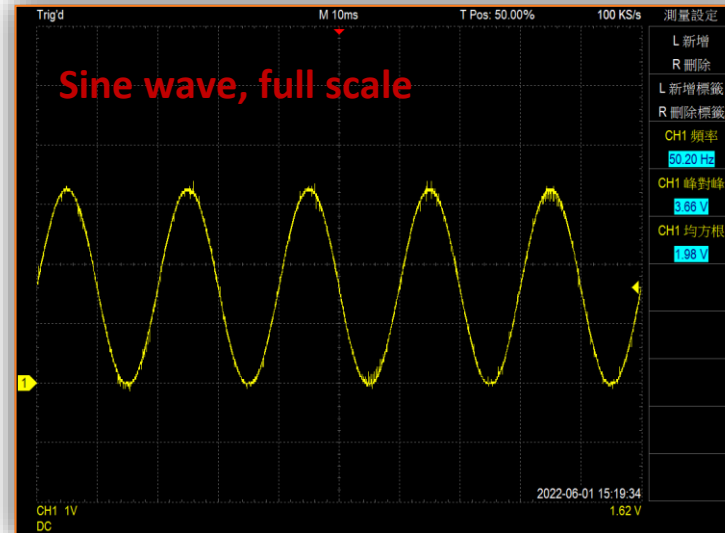
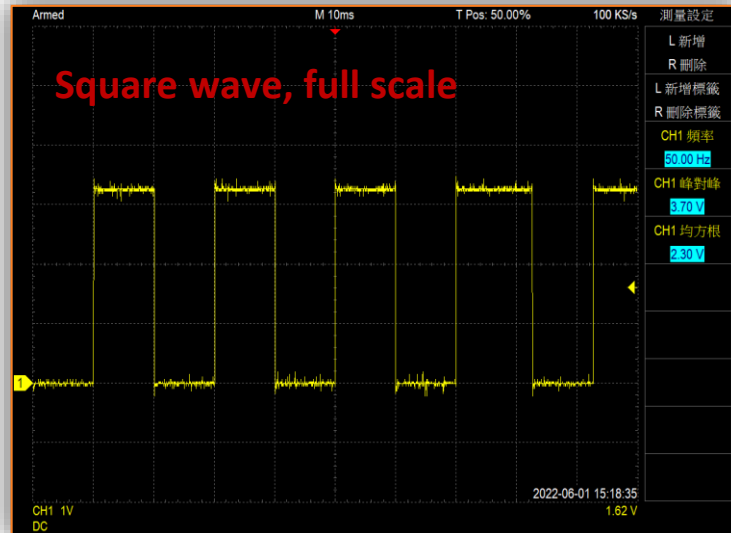
- This is a demo lab, that demo how to output a difference waveforms.
- Just open, build and program the firmware to MCU, then try it.
- The project provide a few definitions use to adjust waveform and alignment/scale of waveform.

```
/* Lab 12-3 */  
#define DAC_Array_Size 200  
  
//#define FIXED_VOLTAGE  
//#define SQUARE_WAVE  
#define SINE_WAVE_CENTRAL  
//#define SINE_WAVE_LEVEL  
  
#define DAC_Wave_Scale 0.5
```

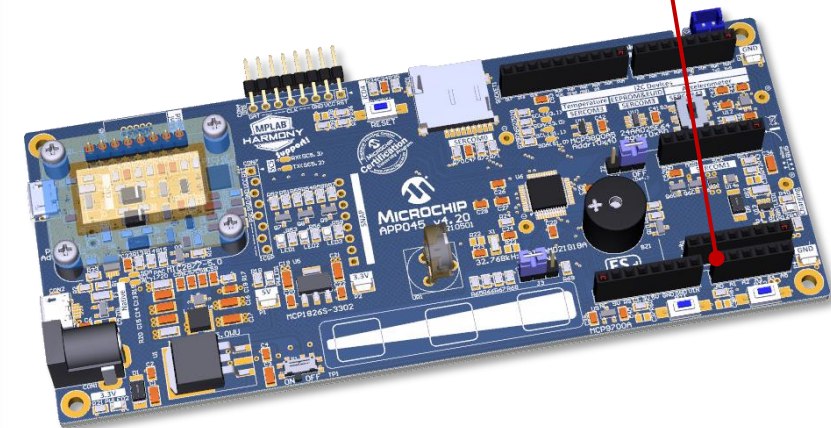
- **Let's go !**

Lab12-3 DAC Wave Output

Result



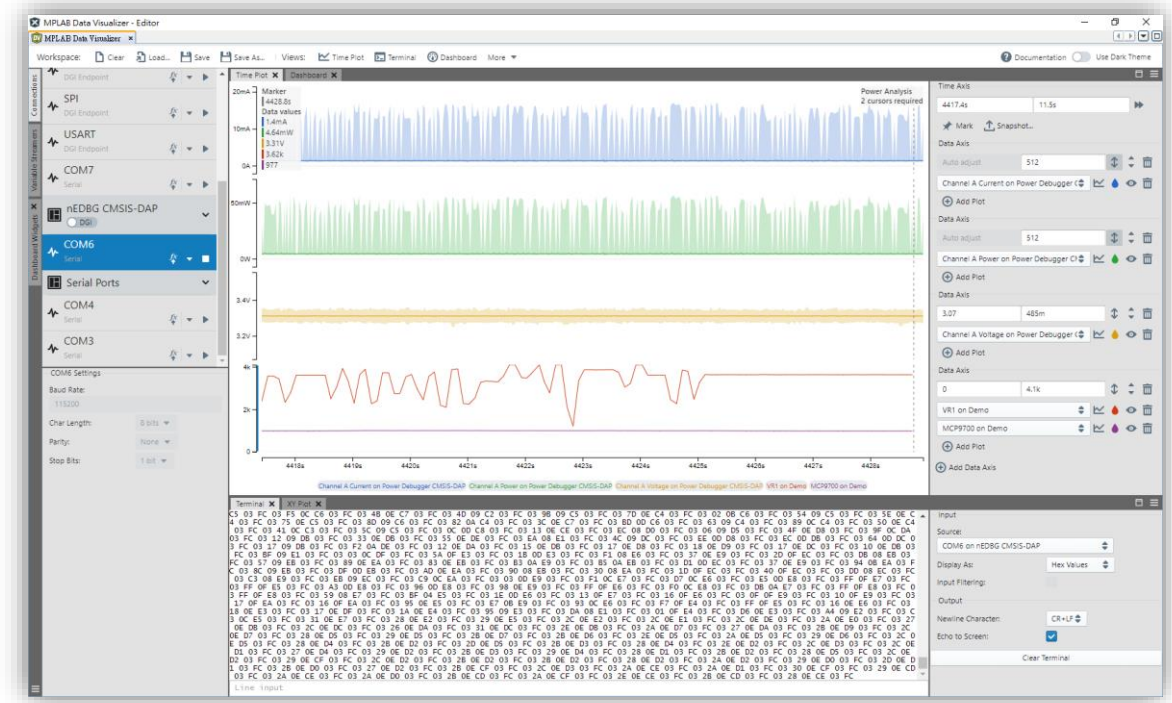
DAC_VOUT (PA02/D24/A0)



MPLAB[®] Data Visualizer

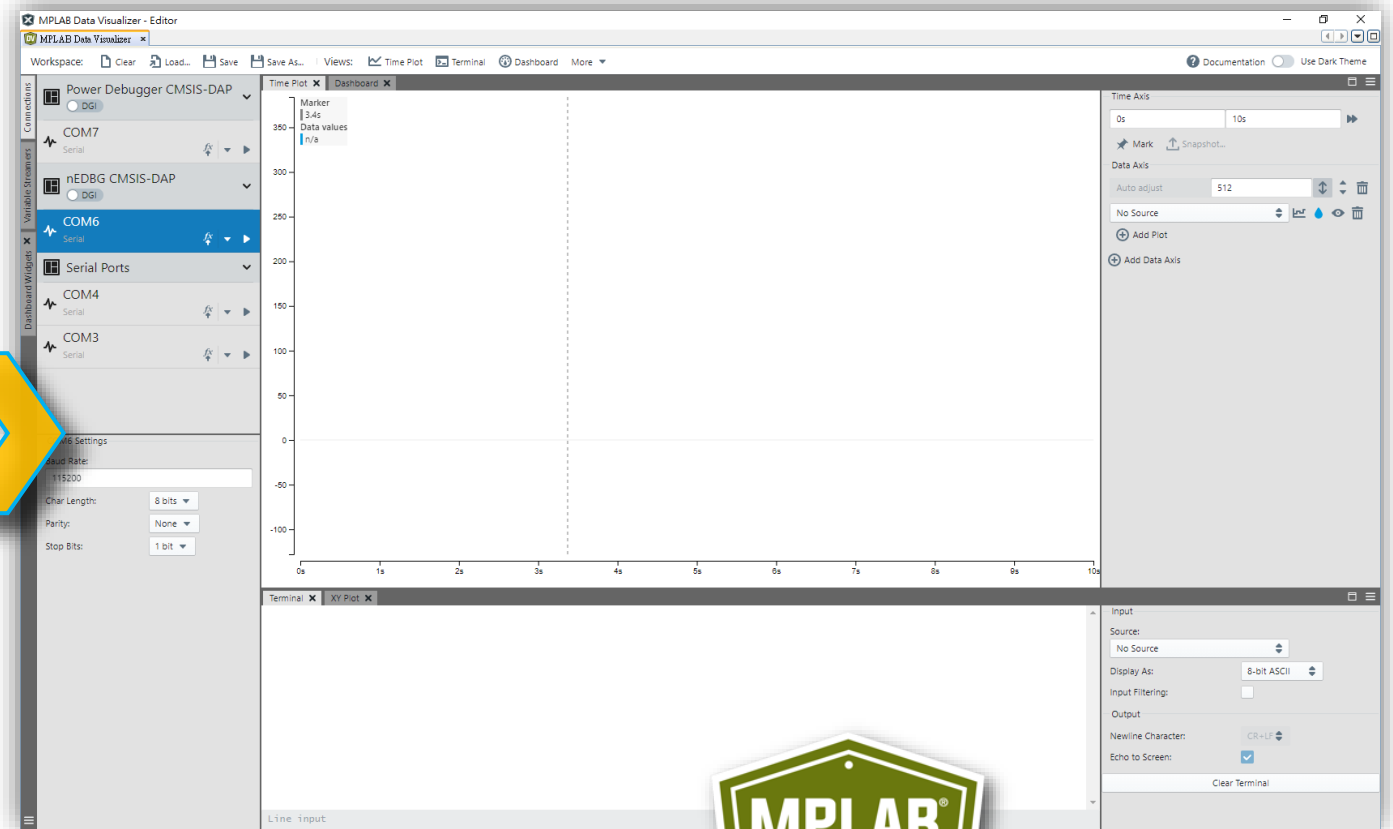
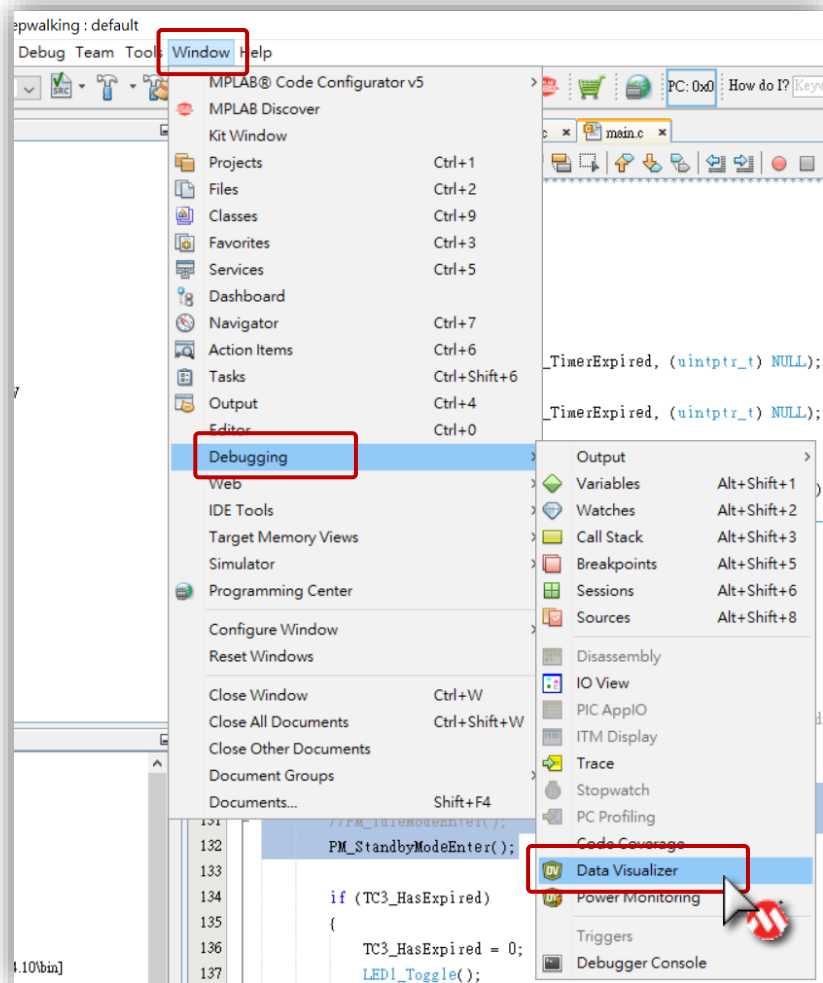
MPLAB Data Visualizer

- MPLAB Data Visualizer is a program used for processing and visualizing data.
- It can receive data from various sources such as the DGI (Data gateway Interface) and COM ports.
- And has a graph plotter and an oscilloscope, a terminal and a configurable dashboard with buttons, sliders, and various indicators. It can decode protocols and log data to file.
- It's an MPLAB X IDE plugin and cross platform standalone application. It's installed at X IDE right. don't need do anything before use.



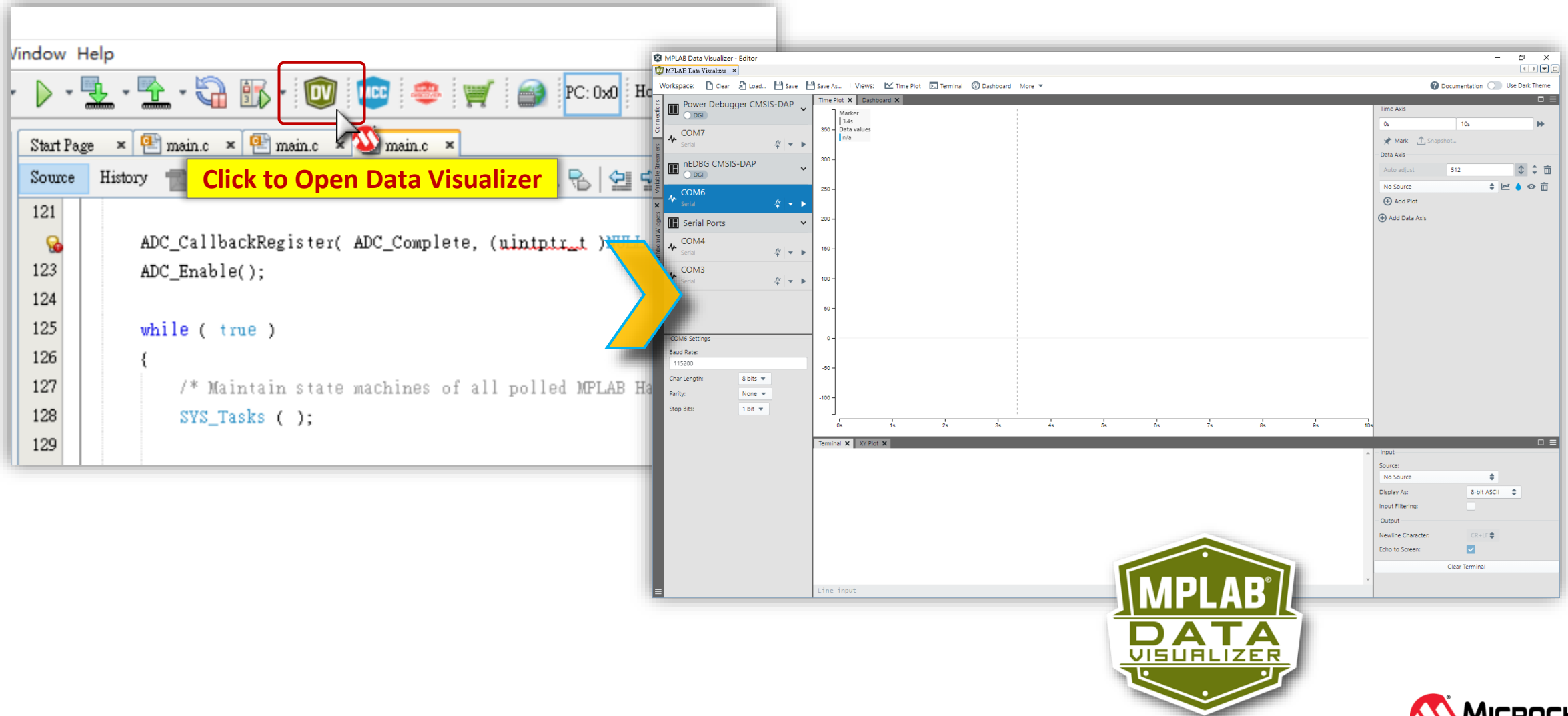
MPLAB Data Visualizer

- To open MPLAB Data Visualizer. You can select **Window > Debugging > Data Visualizer** at MPLAB X IDE

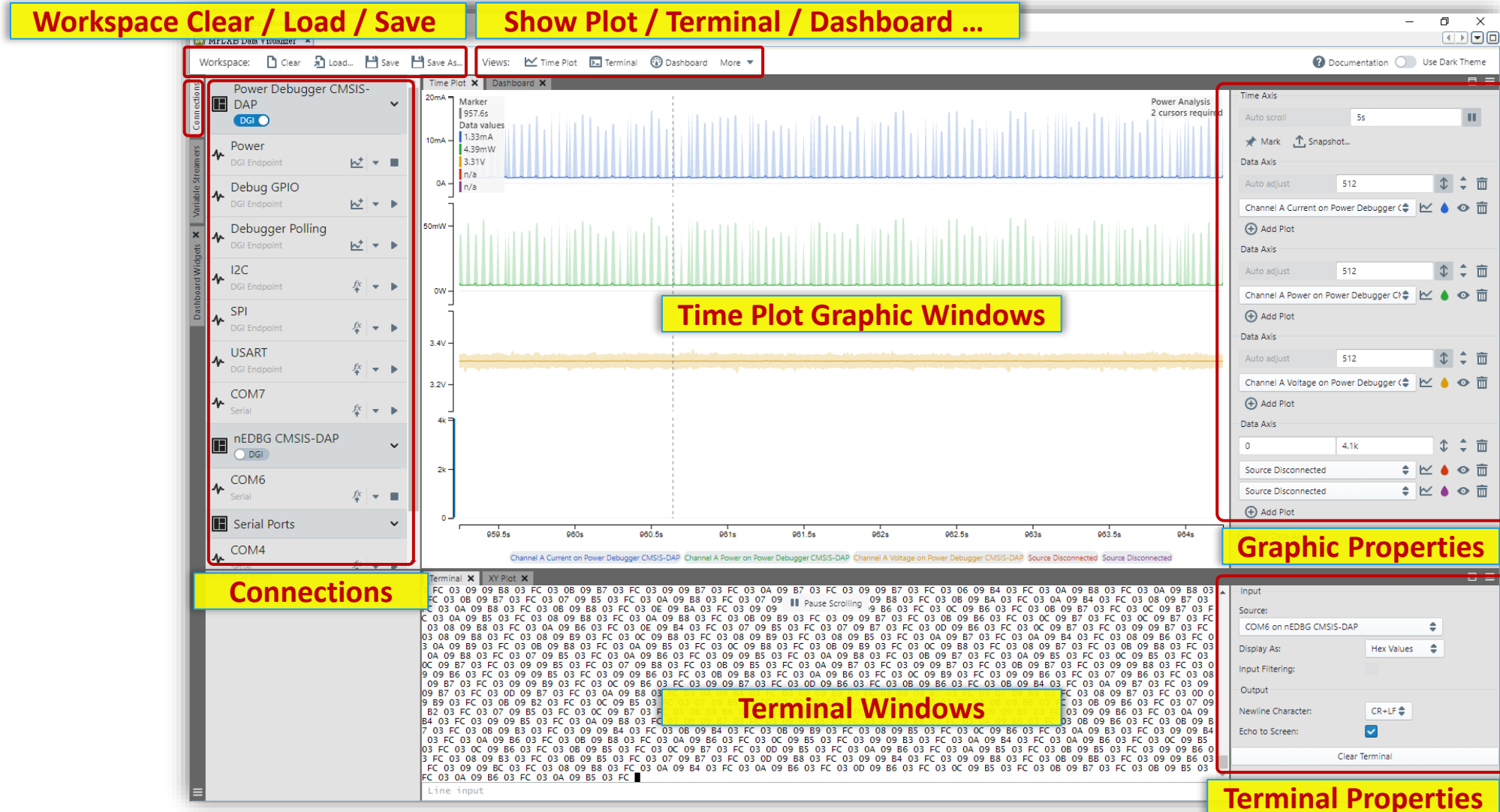


MPLAB Data Visualizer

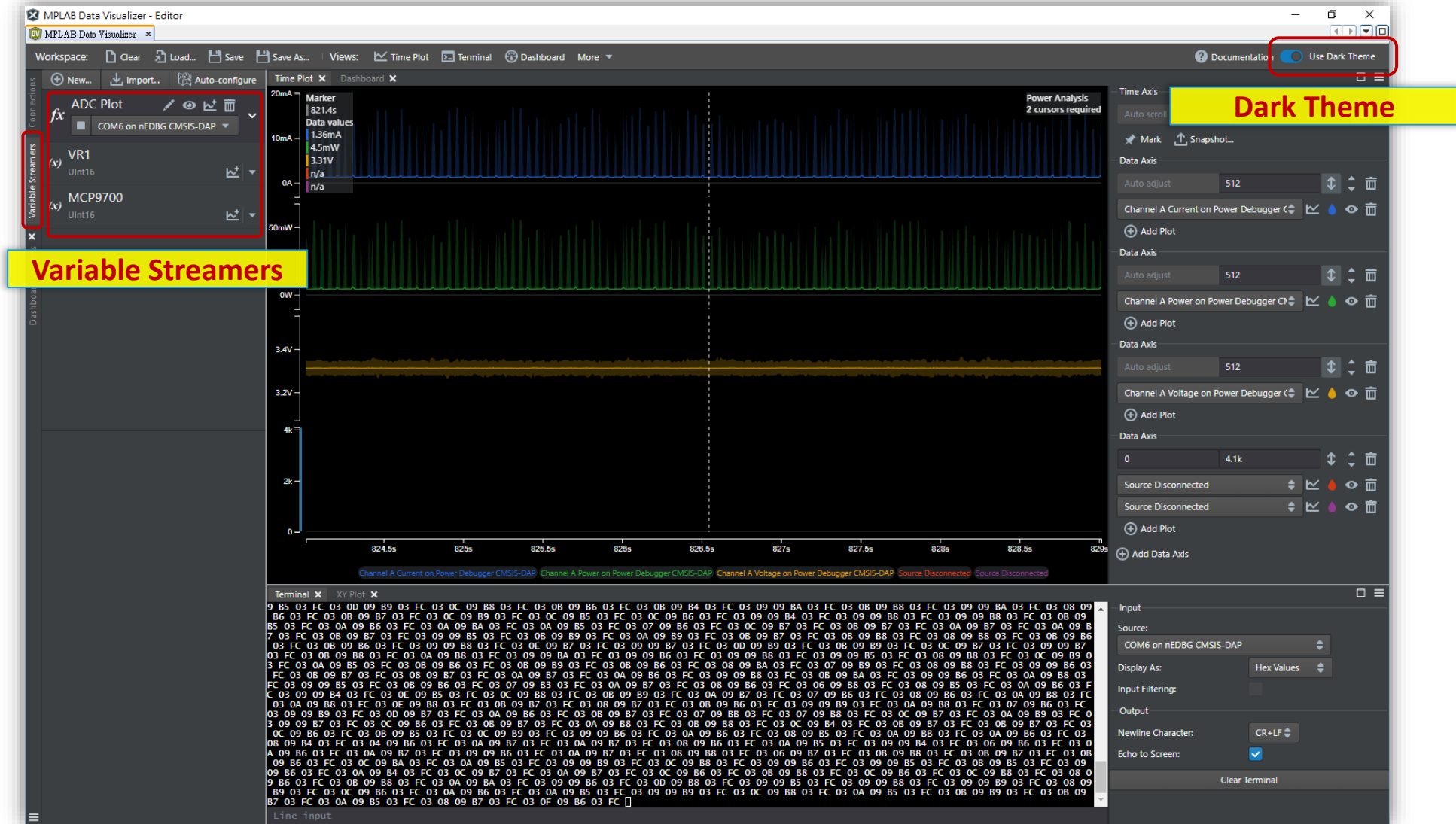
- To open MPLAB Data Visualizer. You can click icon at MPLAB X IDE, also.



Workspace of MPLAB Data Visualizer

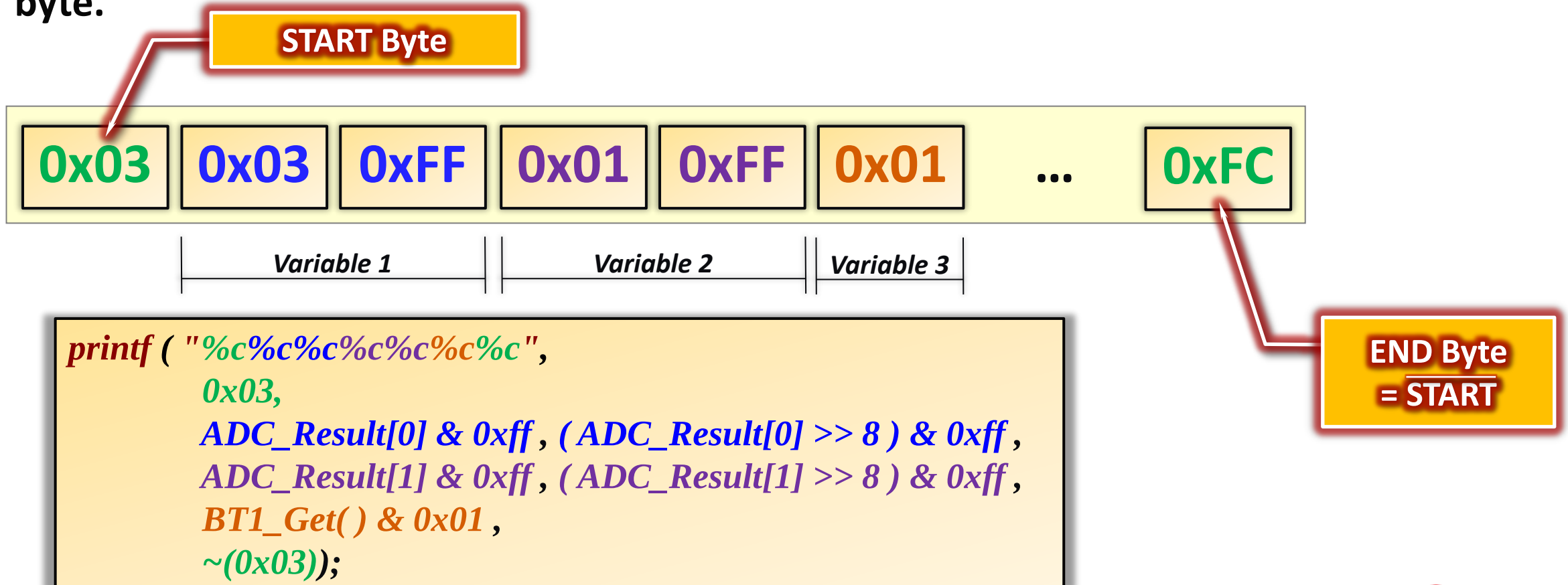


Dark Theme of MPLAB Data Visualizer



Data Stream of Variables

- There are the data stream format for Data Streamer.
- All data must be given as little-endian values.
- The start byte can be of an arbitrary, but the end byte must be the inverse of the Start byte.



Data Streamer Setting

- The correspond format of Data Streamer must be setting at MPLAB Data Visualizer.

Click

Click to add Variable

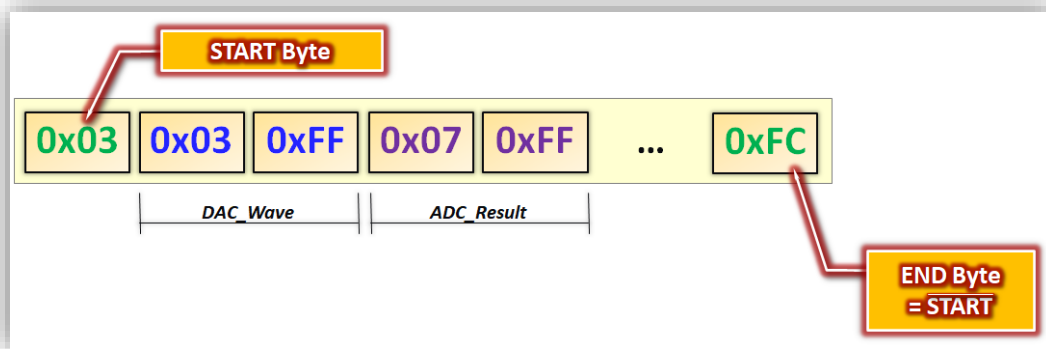
START Byte

END Byte = START

```
printf( "%c%c%c%c%c%c%c",
0x03,
ADC_Result[0] & 0xff, (ADC_Result[0] >> 8) & 0xff,
ADC_Result[1] & 0xff, (ADC_Result[1] >> 8) & 0xff,
BT1_Get() & 0x01,
~(0x03));
```

⚙️ Lab12-4 DAC Wave Output DataVisualizer

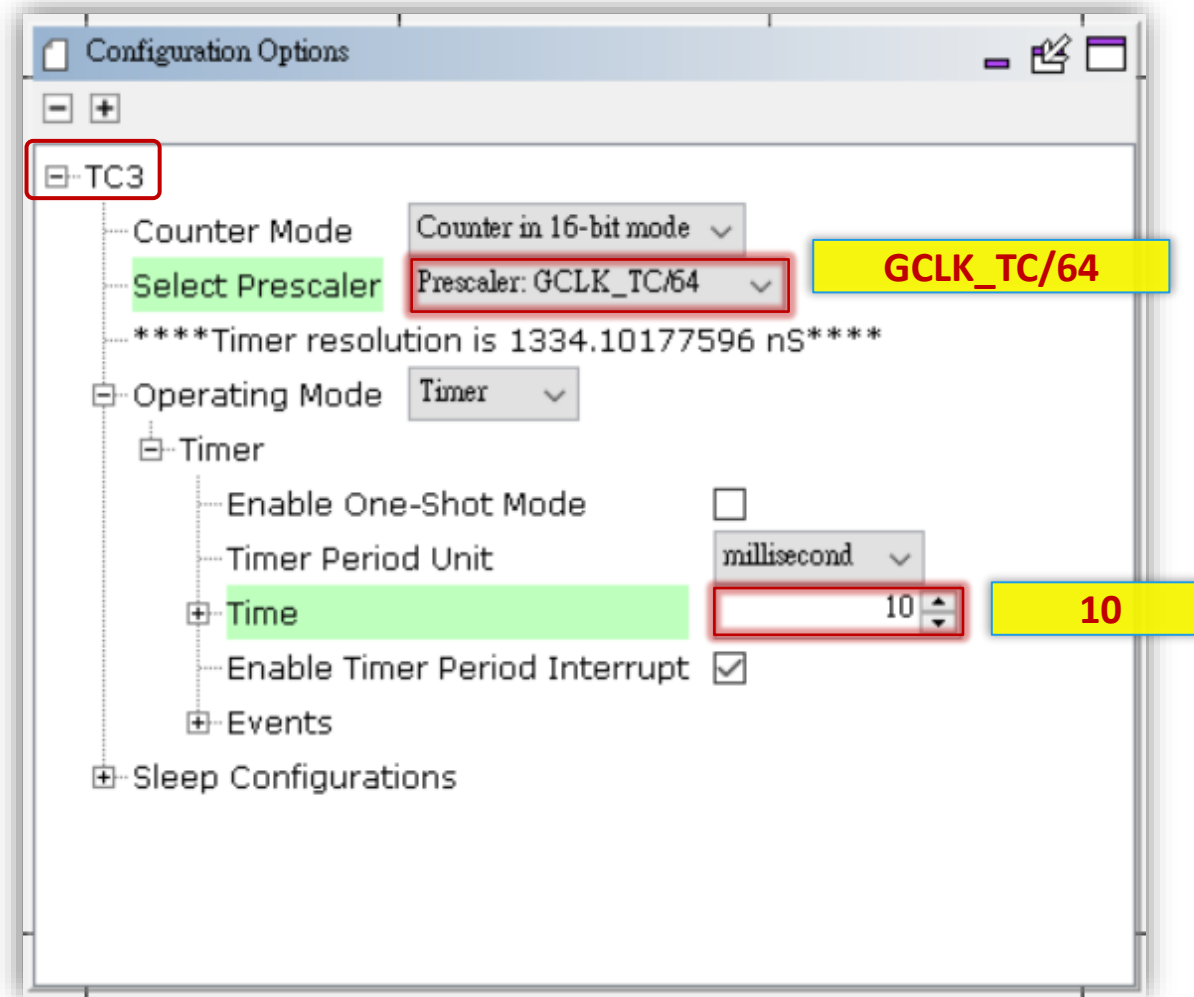
- Please continue from [SAM2001 Lab12-3 \(DAC Wave Output\)](#)
- Try modify UART output to observe DAC idea output and ADC real result for DAC on MPLAB Data Visualizer.
- Try plot the two waveform on MPLAB Data Visualizer.



- **Let's go !**

✖ Lab12-4 DAC Wave Output DataVisualizer

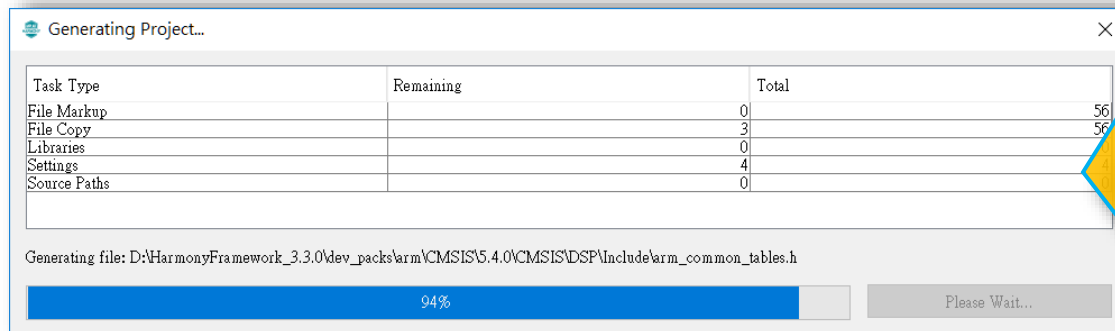
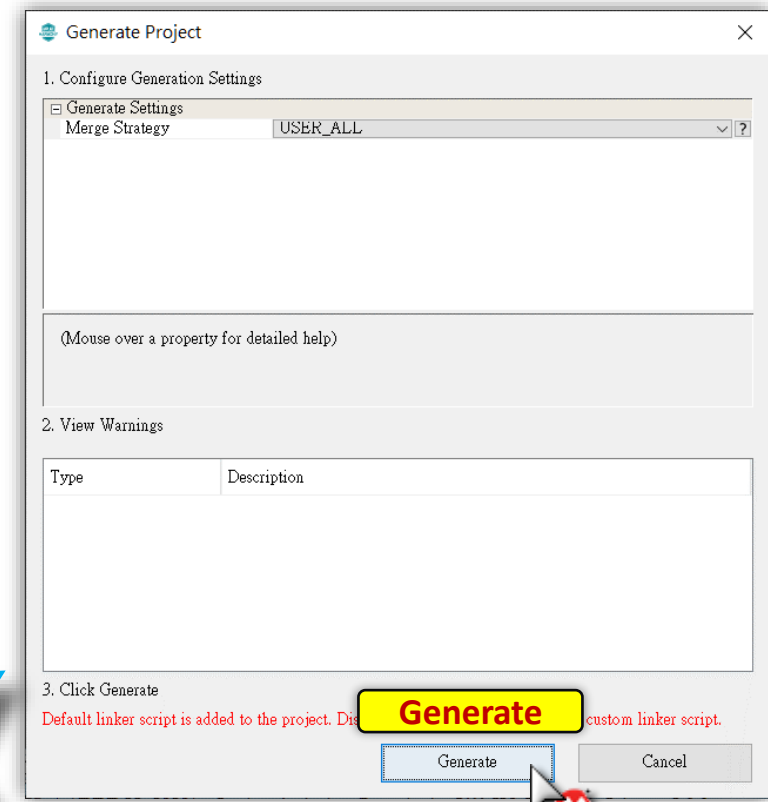
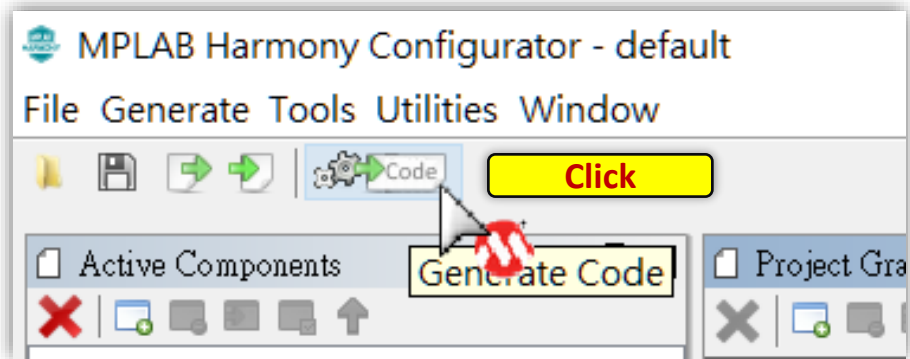
Step 1



⚙️ Lab12-4 DAC Wave Output DataVisualizer

Step 2

- Click  to Generate Code.



⚙️ Lab12-4 DAC Wave Output DataVisualizer

Step 3

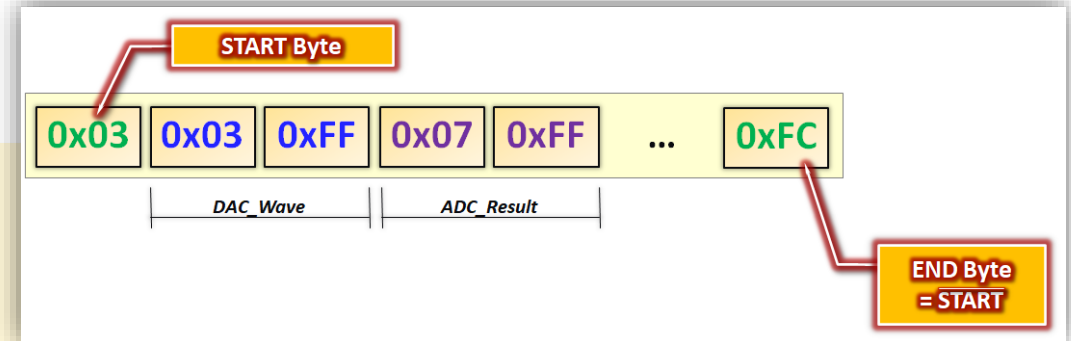
```
...
int main (void)
{
    SYS_Initialize(NULL);
    ...
    while ( true )
    {
        if (TC3_HasExpired)
        {
            ...
            // TODO 12-4.01
            //myprintf("Hello World!\r\n");
            // TODO 12-4.02
            //myprintf("DAC Value : %d\r\n", ADC_Result);
            ...
        }

        if (USART5_IsReceived)
        {
            ...
            // TODO 12-4.03
            //myprintf("\r\nReceived Data : %1c\r\n", USART5_ReceiveData[0]);
            ...
        }
    }
    return ( EXIT_FAILURE);
}
```

⚡ Lab12-4 DAC Wave Output DataVisualizer

Step 4

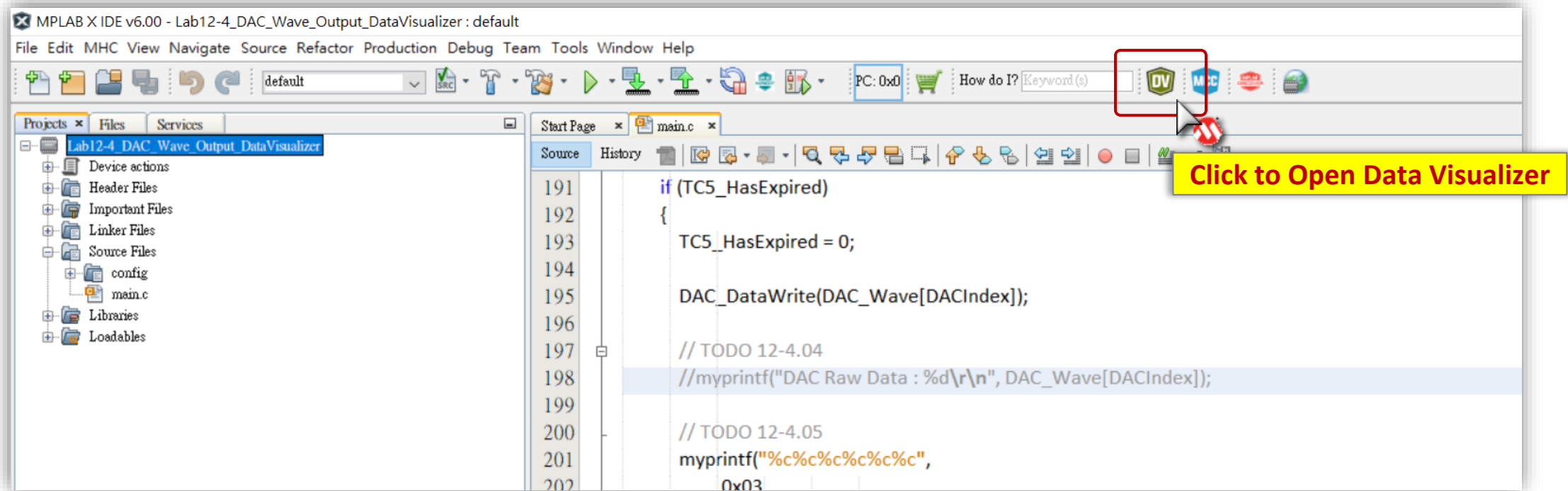
```
...
int main (void)
{
    SYS_Initialize(NULL);
    ...
    while ( true )
    {
        if (TC5_HasExpired)
        {
            ...
            // TODO 12-4.04
            //myprintf("DAC Raw Data : %d\r\n", DAC_Wave[DACIndex]);
            // TODO 12-4.05
            myprintf("%c%c%c%c%c%c",
                    0x03,
                    DAC_Wave[DACIndex] & 0xff, ( DAC_Wave[DACIndex] >> 8) & 0xff,
                    ADC_Result & 0xff, (ADC_Result >> 8) & 0xff,
                    ~(0x03));
        }
    }
    return ( EXIT_FAILURE);
}
```



⚙️ Lab12-4 DAC Wave Output DataVisualizer

Step 5

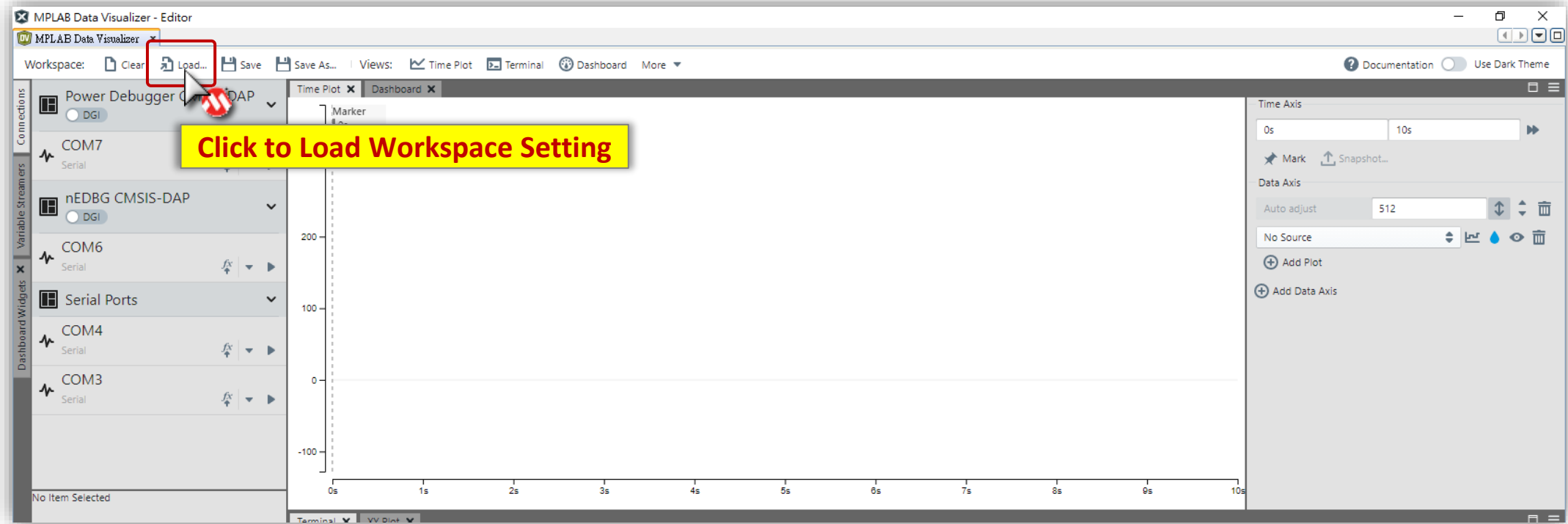
- Open MPLAB Data Visualizer by icon at MPLAB X IDE



⚙️ Lab12-4 DAC Wave Output DataVisualizer

Step 6

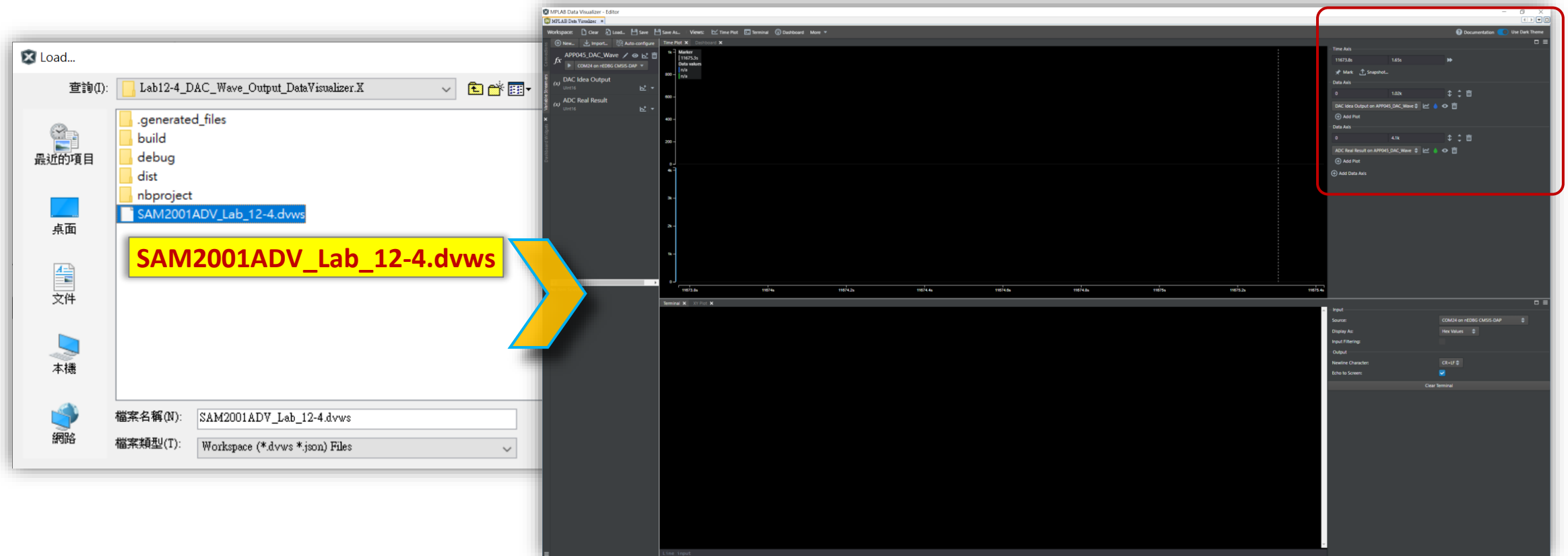
- Just load the workspace file to reduce setting steps.



✖ Lab12-4 DAC Wave Output DataVisualizer

Step 7

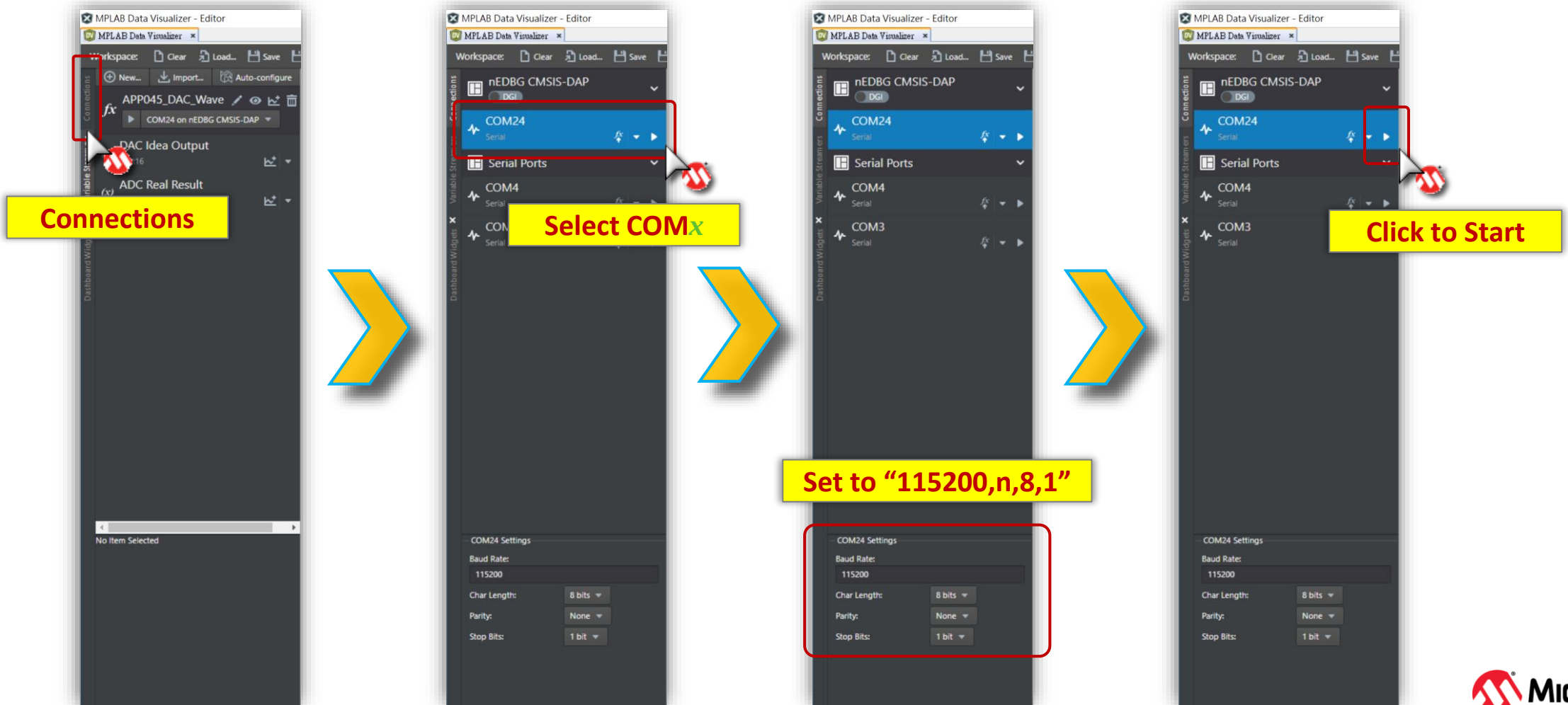
- Select “**SAM2001ADV_Lab_12-4.dvws**” workspace file.
There are 2 data axis be set. That include “DAC Idea Output” and “ADC Real Result”.



⚙️ Lab12-4 DAC Wave Output DataVisualizer

Step 8

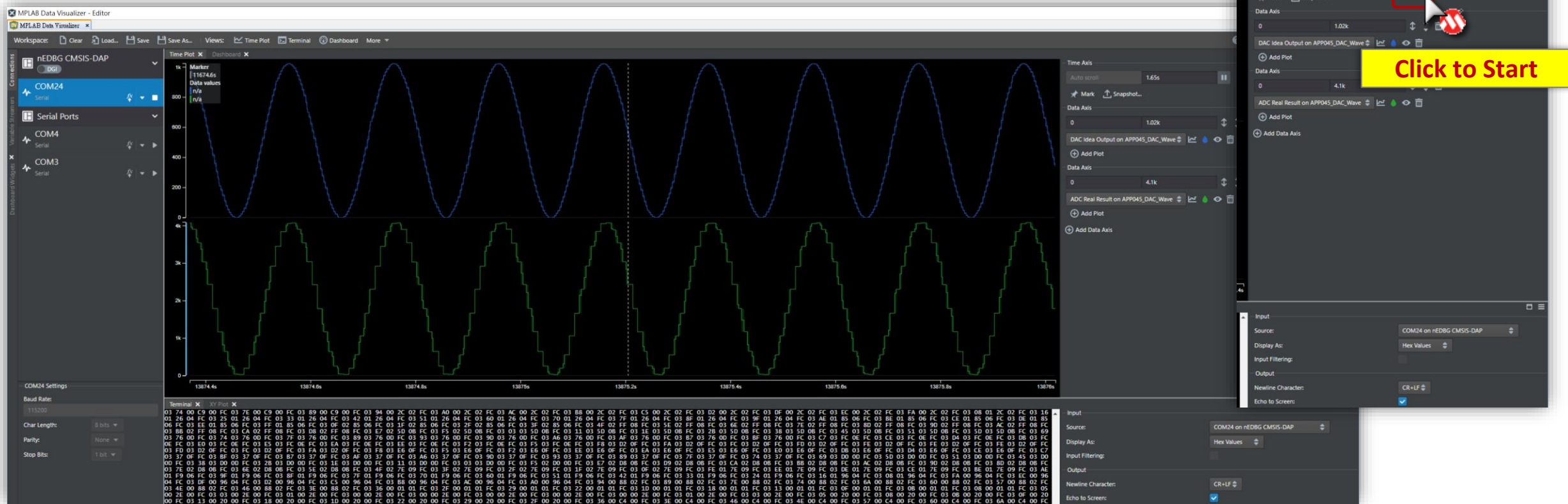
- Set & Enable COMx on left side at MPLAB Data Visualizer.



⚙️ Lab12-4 DAC Wave Output DataVisualizer

Step 10

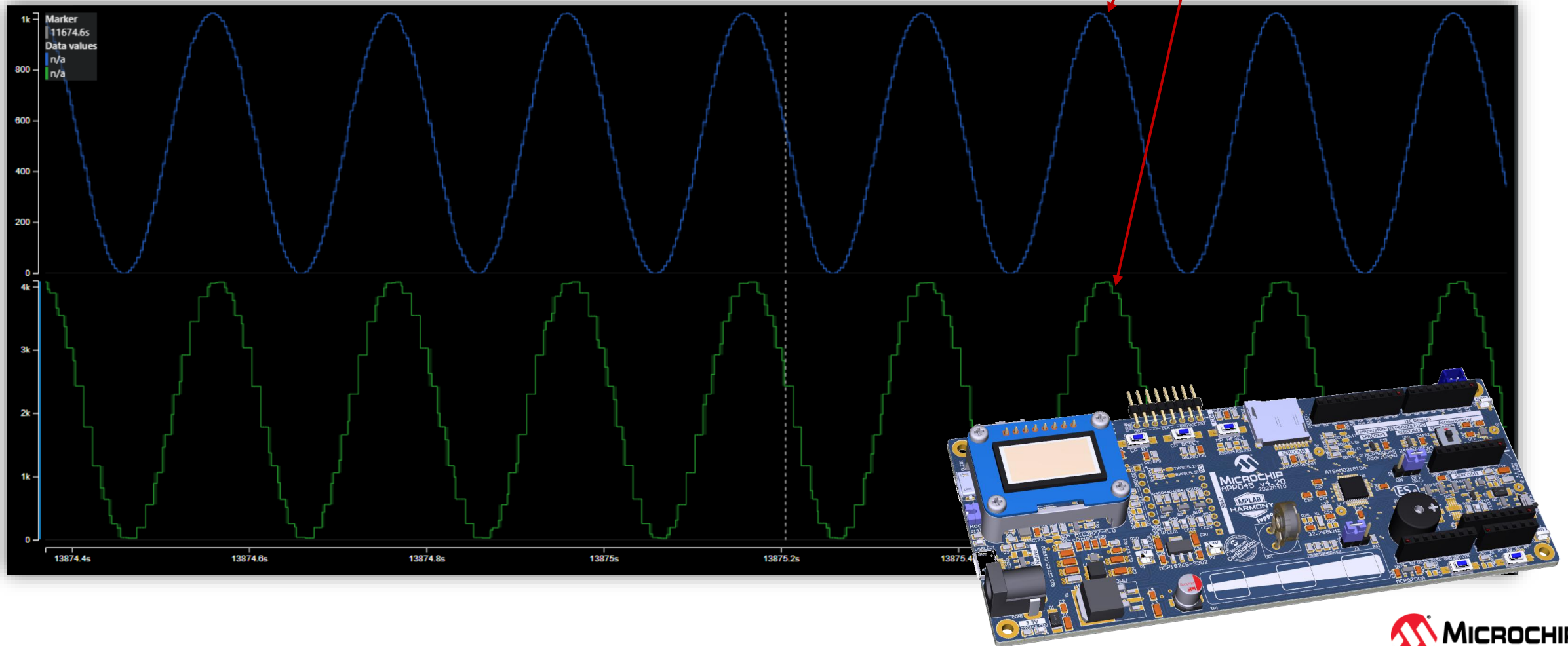
- Get started pot the data.



⚙️ Lab12-4 DAC Wave Output DataVisualizer

Result

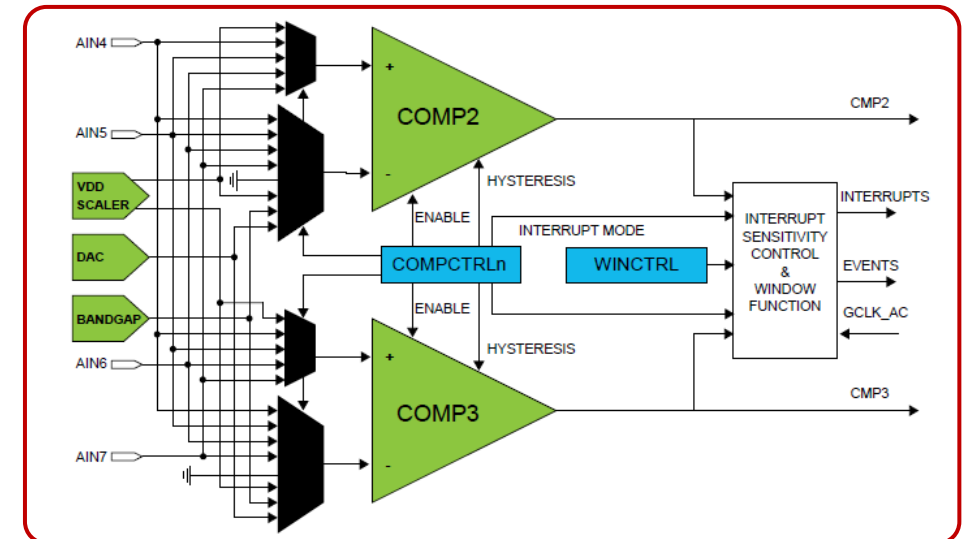
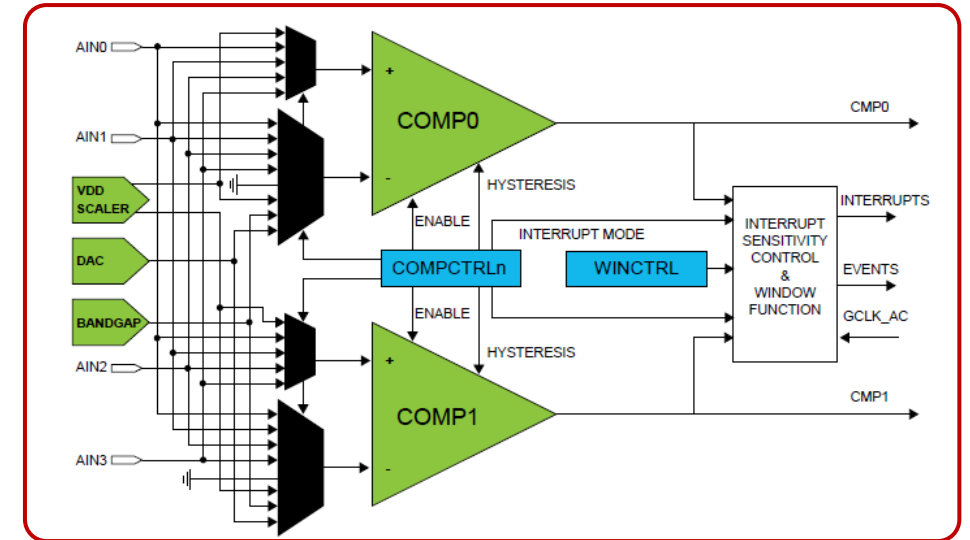
- DAC Idea Output and ADC Real Result show at MPLAB Data Visualizer.



Analog Comparators (AC) Architecture

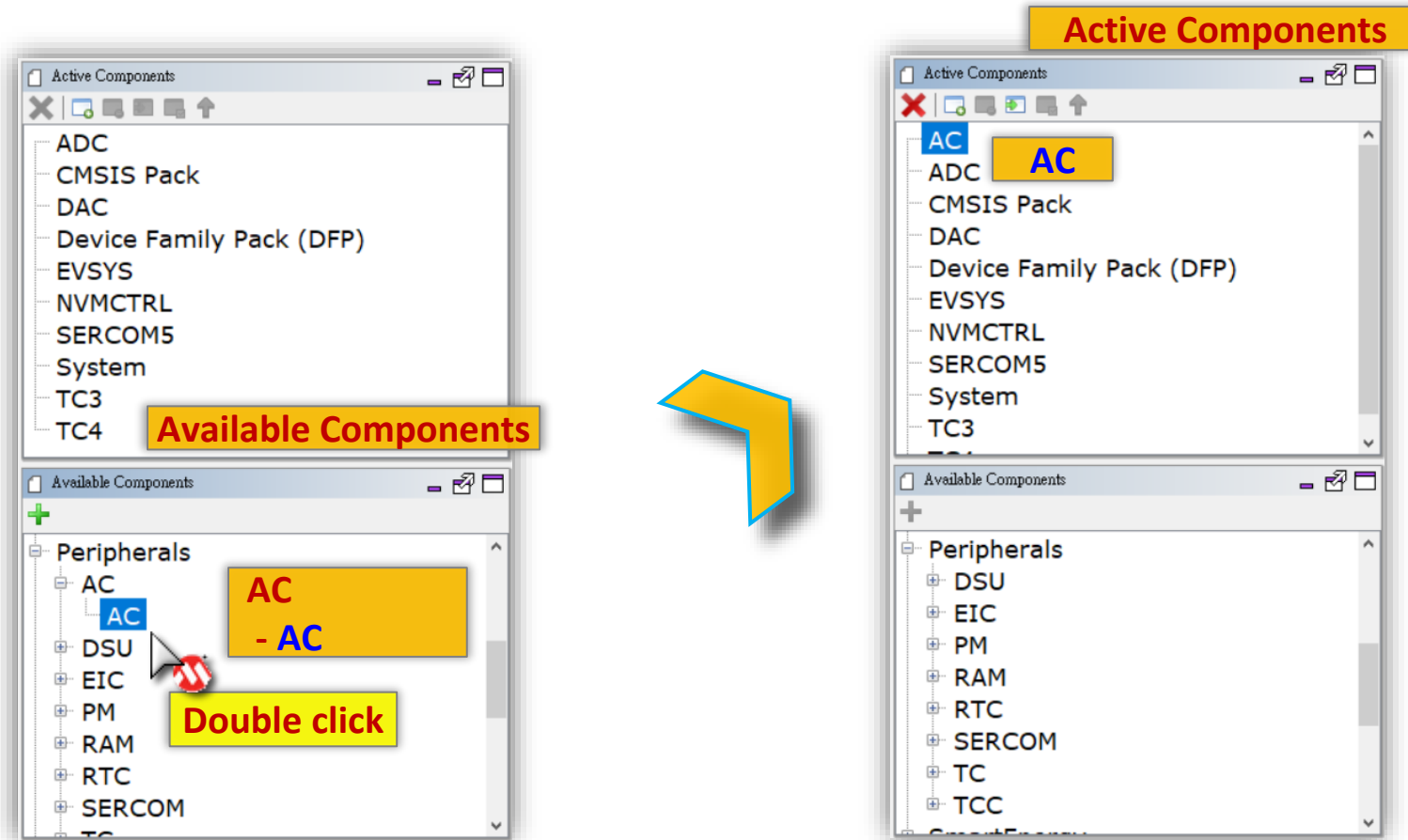
Analog Comparator (AC)

- The Analog Comparator (AC) supports multiple individual comparators.
- Up to 4 x Analog Comparator
 - (COMP0 - COMP3). **SAMD21G18A (COMP0, 1 Only)**
- Flexible input:
 - 4 x IO pins, Ground
 - Bandgap Voltage (1.1V), DAC, VDD Scaler
- Interrupt generation,
 - Rising, Falling, Toggle
 - End of comparison (single-shot mode only)
- Event generation,
- Comparator output, inside/outside window
- Support window Mode
 - Above, Inside, Below, Outside
- Optional Digital Filter



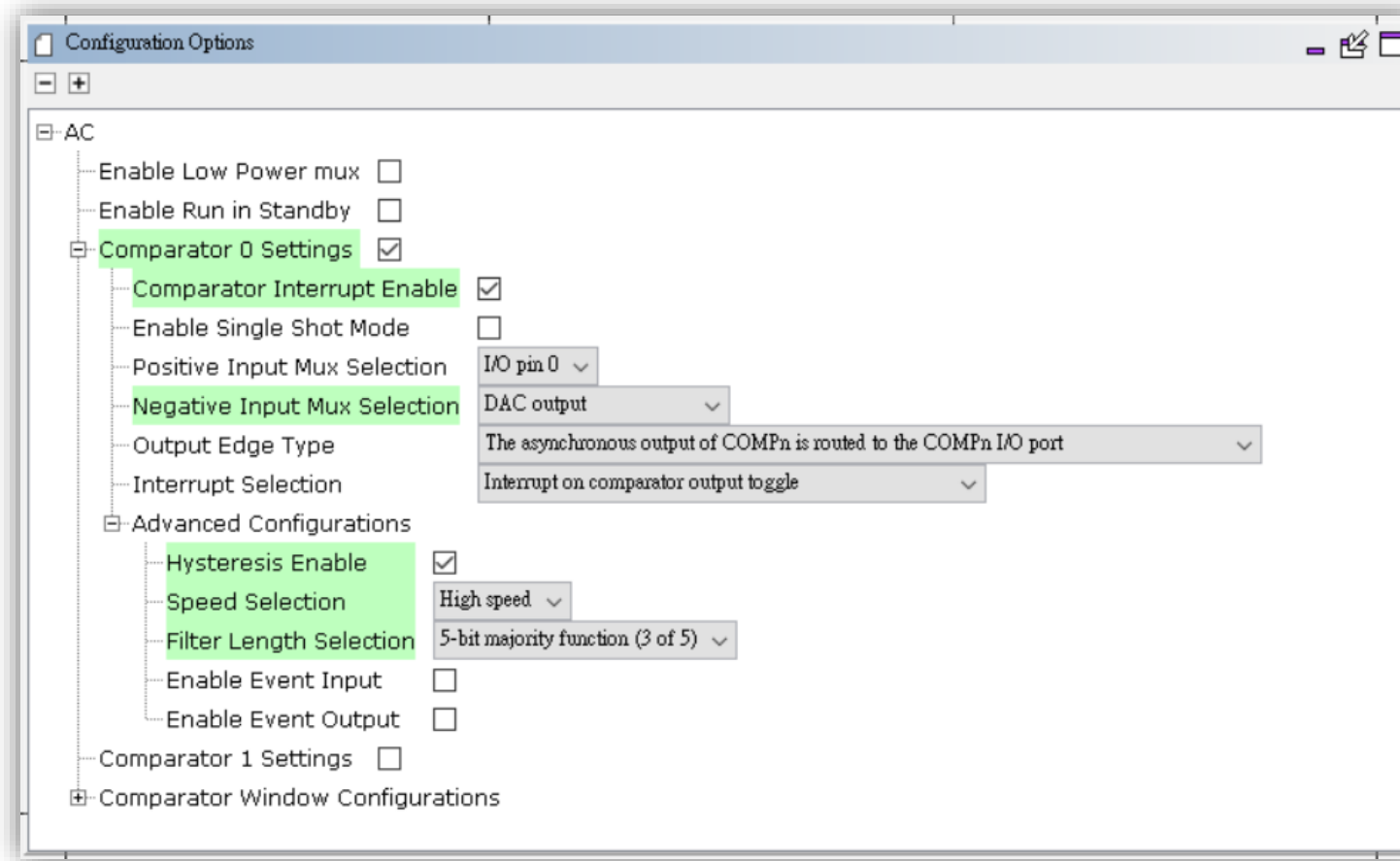
Add AC Function using MHC

- Find **AC** component in Available Components window.
- Double click **AC** to add to Active Components window.



Single Comparator Example

Configuration Options Example



AC Pin Mapping

PINMUX Configuration Example

- **Analog pins (AC)** will all belong to **PINMUX function group B and H**.
- **AC input (B)** is fixed pin, that assign to PA04 ~ PA07. AC output (H) pin is selectable.

Pin ⁽¹⁾			I/O Pin	Supply	A	B ⁽²⁾⁽³⁾					C	D	E	F	G	H
SAMD2xE	SAMD2xG	SAMD2xJ			EIC	REF	ADC	AC	PTC	DAC	SERCOM ⁽²⁾⁽³⁾	SERCOM-ALT	Tc ⁽⁴⁾ /TCC	TCC	COM	AC/ GCLK
5	9	13	PA04	VDDANA	EXTINT[4]	ADC/REFB	AIN[4]	AIN[0]	Y[2]			SERCOM0/ PAD[0]	TCC0/WO[0]	TCC3/ WO[2]		
6	10	14	PA05	VDDANA	EXTINT[5]		AIN[5]	AIN[1]	Y[3]			SERCOM0/ PAD[1]	TCC0/WO[1]	TCC3/ WO[3]		
7	11	15	PA06	VDDANA	EXTINT[6]		AIN[6]	AIN[2]	Y[4]			SERCOM0/ PAD[2]	TCC1/WO[0]	TCC3/ WO[4]		
8	12	16	PA07	VDDANA	EXTINT[7]		AIN[7]	AIN[3]	Y[5]			SERCOM0/ PAD[3]	TCC1/WO[1]	TCC3/ WO[5]	I2S/SD[0]	

Pin ⁽¹⁾			I/O Pin	Supply	A	B ⁽²⁾⁽³⁾					C	D	E	F	G	H
SAMD2xE	SAMD2xG	SAMD2xJ			EIC	REF	ADC	AC	PTC	DAC	SERCOM ⁽²⁾⁽³⁾	SERCOM-ALT	Tc ⁽⁴⁾ /TCC	TCC	COM	AC/ GCLK
	21	29	PA12	VDDIO	EXTINT[12]						SERCOM2/ PAD[0]	SERCOM4/ PAD[0]	TCC2/WO[0]	TCC0/ WO[6]		AC/CMP[0]
	22	30	PA13	VDDIO	EXTINT[13]						SERCOM2/ PAD[1]	SERCOM4/ PAD[1]	TCC2/WO[1]	TCC0/ WO[7]		AC/CMP[1]
15	23	31	PA14	VDDIO	EXTINT[14]						SERCOM2/ PAD[2]	SERCOM4/ PAD[2]	TC3/WO[0]	TCC0/ WO[4]		GCLK_IO[0]
16	24	32	PA15	VDDIO	EXTINT[15]						SERCOM2/ PAD[3]	SERCOM4/ PAD[3]	TC3/WO[1]	TCC0/ WO[5]		GCLK_IO[1]
17	25	35	PA16	VDDIO	EXTINT[0]				X[4]		SERCOM1/ PAD[0]	SERCOM3/ PAD[0]	TCC2/WO[0]	TCC0/WO[6]		GCLK_IO[2]
18	26	36	PA17	VDDIO	EXTINT[1]				X[5]		SERCOM1/ PAD[1]	SERCOM3/ PAD[1]	TCC2/WO[1]	TCC0/WO[7]		GCLK_IO[3]
19	27	37	PA18	VDDIO	EXTINT[2]				X[6]		SERCOM1/ PAD[2]	SERCOM3/ PAD[2]	TC3/WO[0]	TCC0/ WO[2]		AC/CMP[0]
20	28	38	PA19	VDDIO	EXTINT[3]				X[7]		SERCOM1/ PAD[3]	SERCOM3/ PAD[3]	TC3/WO[1]	TCC0/ WO[3]	I2S/SD[0]	AC/CMP[1]

AC Pin Mapping cont.

PINMUX Configuration Example

Pin Table

Package: TQFP48

PA04 ~ P07 (AIN0 ~ AIN3)

PA12/PA13, PA18/PA19 CMP1

Module	Function	PA00	PA01	DAC_VO	PA03	GNDANA	VDDANA	ADC_AL	PB09	PA04	PA05	PA06	PA07	PA08	PA09	PA10	PA11	VDDIO	GNDIO	PB10	PB11	PA12	PA13	OCLK_L	OCLK_L	PA16	LED3	PA18	PA19	LED1	LED2
AC	AC_AIN0																														
	AC_AIN1																														
	AC_AIN2																														
	AC_AIN3																														
	AC_CMP0																														
	AC_CMP1																														
	ADC_AIN0																														
	ADC_AIN1																														
	ADC_AIN10																														
	ADC_AIN11																														
	ADC_AIN16																														
	ADC_AIN17																														
	ADC_AIN18																														

AC Functions

Interface Routines

```
void AC_Start( AC_CHANNEL channel_id );  
void AC_SwapInputs( AC_CHANNEL channel_id );  
bool AC_StatusGet (AC_CHANNEL channel);  
void AC_CallbackRegister (AC_CALLBACK callback, uintptr_t context);  
void AC_SetVddScalar( AC_CHANNEL channel_id , uint8_t vdd_scalar);  
void AC_ChannelSelect  
    ( AC_CHANNEL channel_id , AC_POSINPUT positiveInput,  
      AC_NEGINPUT negativeInput);
```

AC Code Example

AC Example

```
...  
void AC0_Actived(uint8_t int_flags, uintptr_t context)  
{  
    LED3_Toggle();  
}  
  
int main (void)  
{  
    SYS_Initialize ( NULL );  
    AC_CallbackRegister(AC0_Actived, (uintptr_t) NULL);  
    ...  
    while( true )  
    {  
        ...  
    }  
}
```

Propagation Delay vs. Power Consumption

- It's possible to trade off comparison speed for power efficiency to get the shortest possible propagation delay or the lowest power consumption.
- The speed setting is configured for each comparator individually by the Speed bit.
- The Speed bits select the amount of bias current provided to the comparator, and as such will also affect the start-up time.

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
	Hysteresis	Hysteresis = 1, Fast mode	20	50	80	mV
		Hysteresis = 1, Low-power mode	15	40	75	mV
	Propagation delay	Changes for $V_{ACM} = V_{DDANA}/2$ 100 mV overdrive, Fast mode	-	60	116	ns
		Changes for $V_{ACM} = V_{DDANA}/2$ 100 mV overdrive, Low-power mode	-	225	370	ns
$t_{STARTUP}$	Startup time	Enable to ready delay Fast mode	-	1	2	μ s
		Enable to ready delay Low power mode	-	12	19	μ s

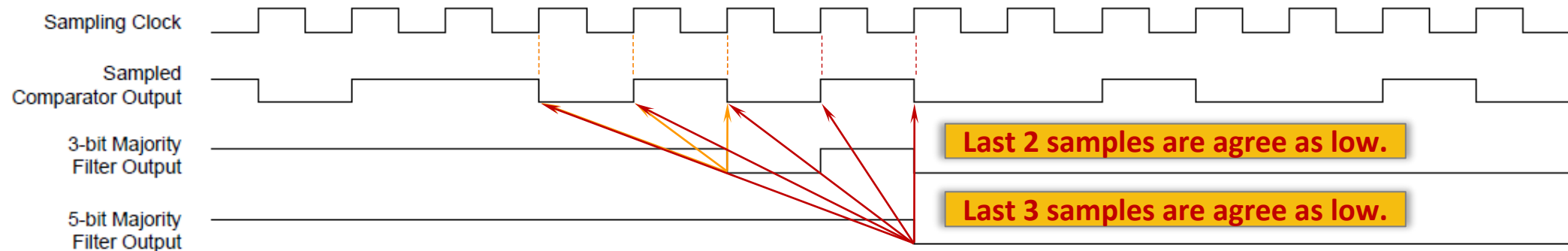
Input Hysteresis

- Application software can selectively enable/disable hysteresis for the comparison.
- Applying hysteresis will help prevent constant toggling of the output, which can be caused by noise when the input signals are close to each other.
- Hysteresis is enabled for each comparator individually by the Hysteresis Enable bit. It's available only in continuous mode.

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
	Hysteresis	Hysteresis = 1, Fast mode	20	50	80	mV
		Hysteresis = 1, Low-power mode	15	40	75	mV
	Propagation delay	Changes for $V_{ACM} = V_{DDANA}/2$ 100 mV overdrive, Fast mode	-	60	116	ns
		Changes for $V_{ACM} = V_{DDANA}/2$ 100 mV overdrive, Low-power mode	-	225	370	ns
$t_{STARTUP}$	Startup time	Enable to ready delay Fast mode	-	1	2	μs
		Enable to ready delay Low power mode	-	12	19	μs

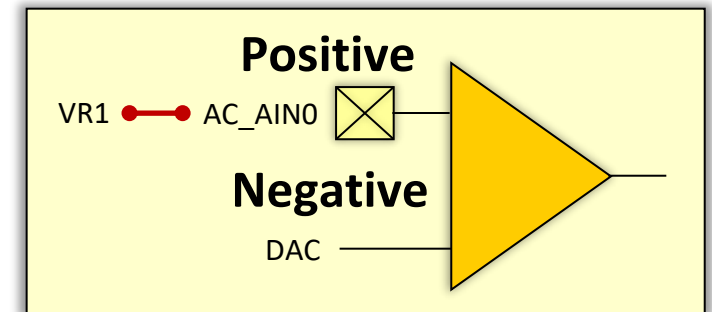
Digital Filter

- The output of the comparators can be filtered digitally to reduce noise.
- The filtering is determined by the Filter Length bits in the and is independent for each comparator.
 - OFF, None (N=0)
 - MAJ3, 3-Bits majority (N=3, 2 of 3)
 - MAJ5, 5-Bits majority (N=5, 3 of 5)
- Any change in the comparator output is considered valid only, **if (3/5) out of the last N samples agree.**
- The filter sampling rate is the GCLK_AC frequency.

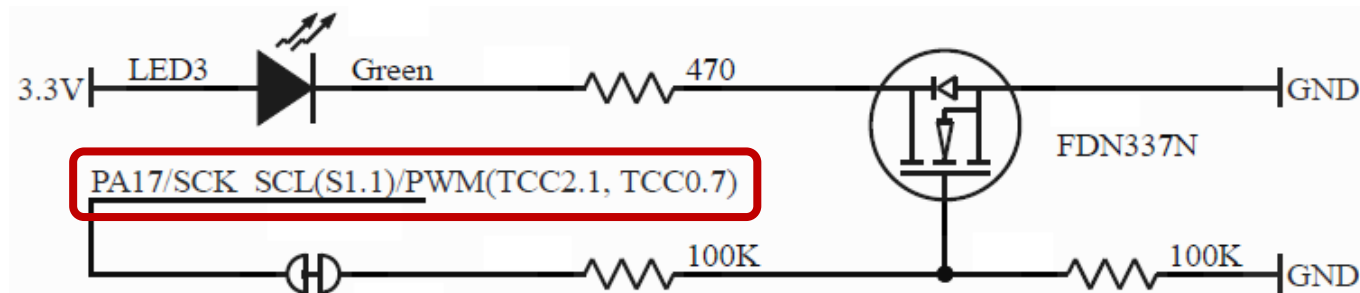
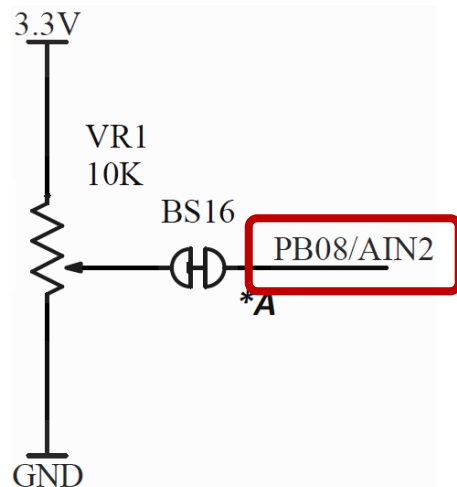


⚙️ Lab12-5 AC Single Comparator

- Please continue from [SAM2001 Lab12-2 \(DAC Output to ADC Measurement\)](#)
- Try to add **AC** module to project.
- Initial COMP0 and Set COMP0 to,
 - Positive Input : $AIN0_{AC}$ (PA04)
 - Negative Input : DAC (1.65V recommended)
 - Continuous Measurement & Fast Mode
 - Enable Hysteresis & Digital Filter (MAJ5)
 - Interrupt on comparator output toggle
- Wire VR1($AIN2_{ADC}$, A1_{Arduino}, PB08) to ($AIN0_{AC}$, A3_{Arduino}, PA04) and use COMP0 to compare VR1 and DAC voltage, then output the result to LED3 to indicate the status of COMP0 output.
- **Let's go !**



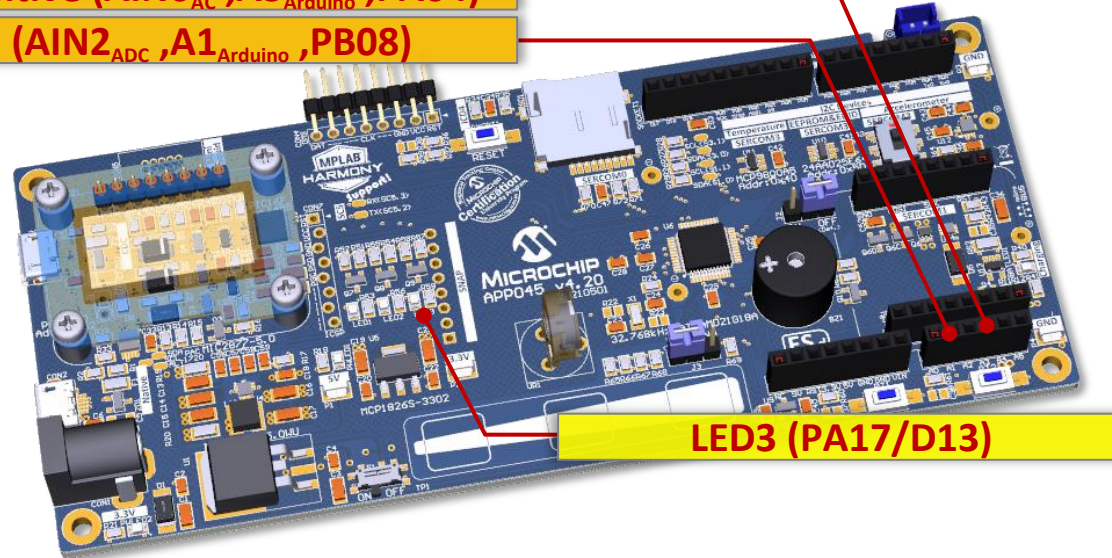
⚡ Lab12-5 AC Single Comparator



AC0 Positive (AIN0_{AC}, A3_{Arduino}, PA04)

VR1 (AIN2_{ADC}, A1_{Arduino}, PB08)

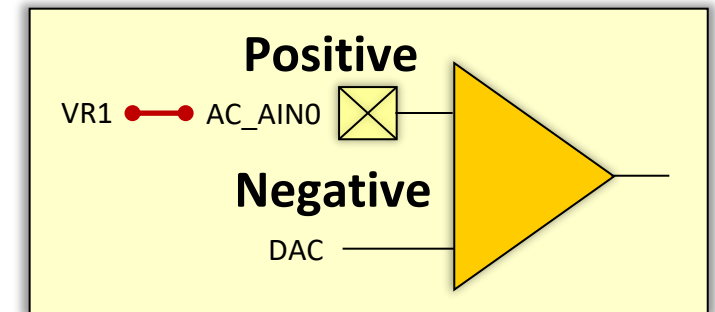
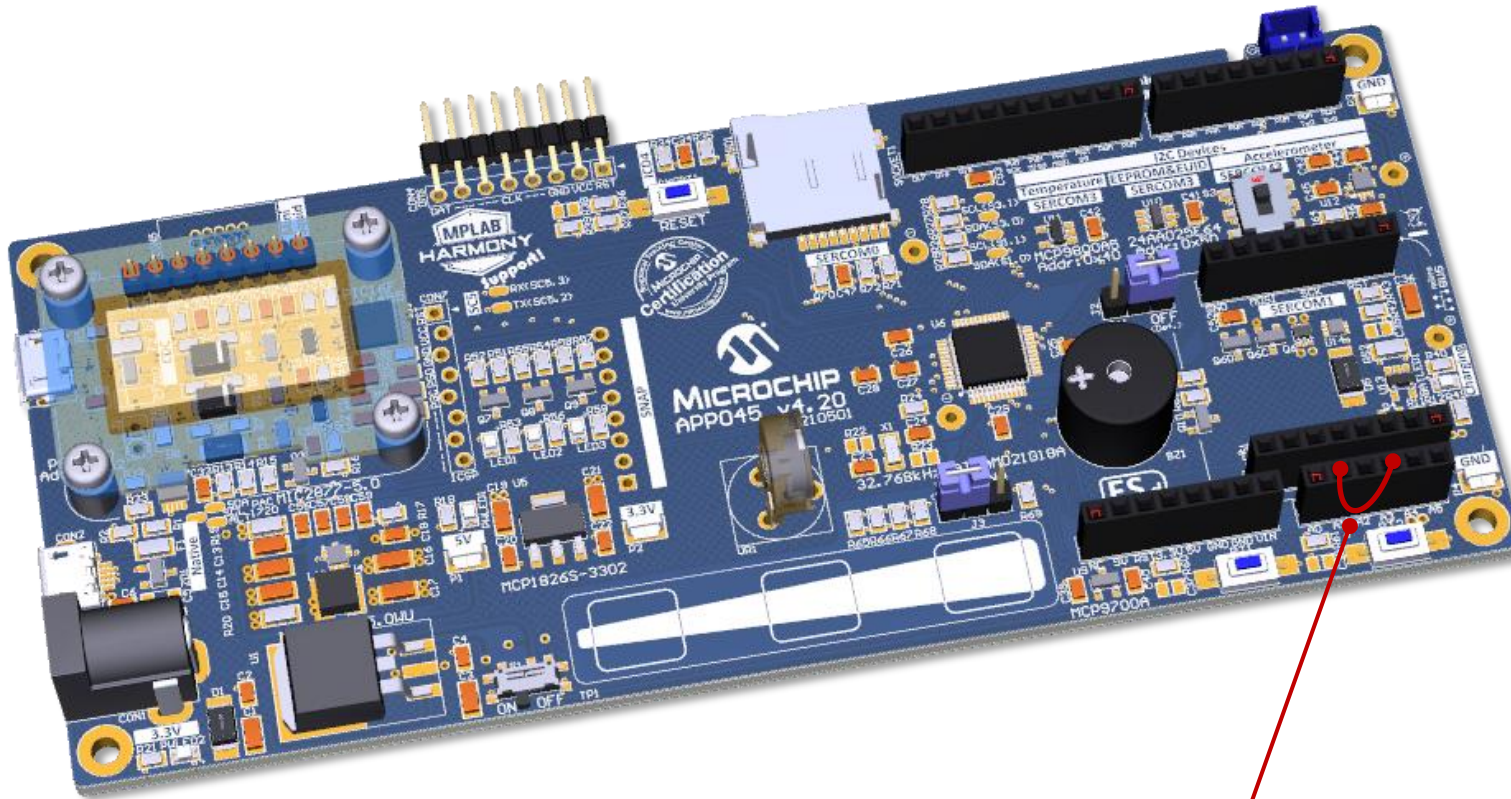
SOCKET1A		ARDUINO-M0-PRO	
PA02/AIN0	1	D24/A0	
PB08/AIN2	2	D25/A1	
PB09/AIN3	3	D26/A2	
PA04/AIN4/EINT4/Y2	4	D27/A3	
PA05/AIN5/EINT5/Y3	5	D28/A4	
PB02/AIN10/Y8	6	D29/A5	



⚡ Lab12-5 AC Single Comparator

Step 1

- Wire VR1(AIN2_{ADC} ,A1_{Arduino} ,PB08) to (AIN0_{AC} ,A3_{Arduino} ,PA04)

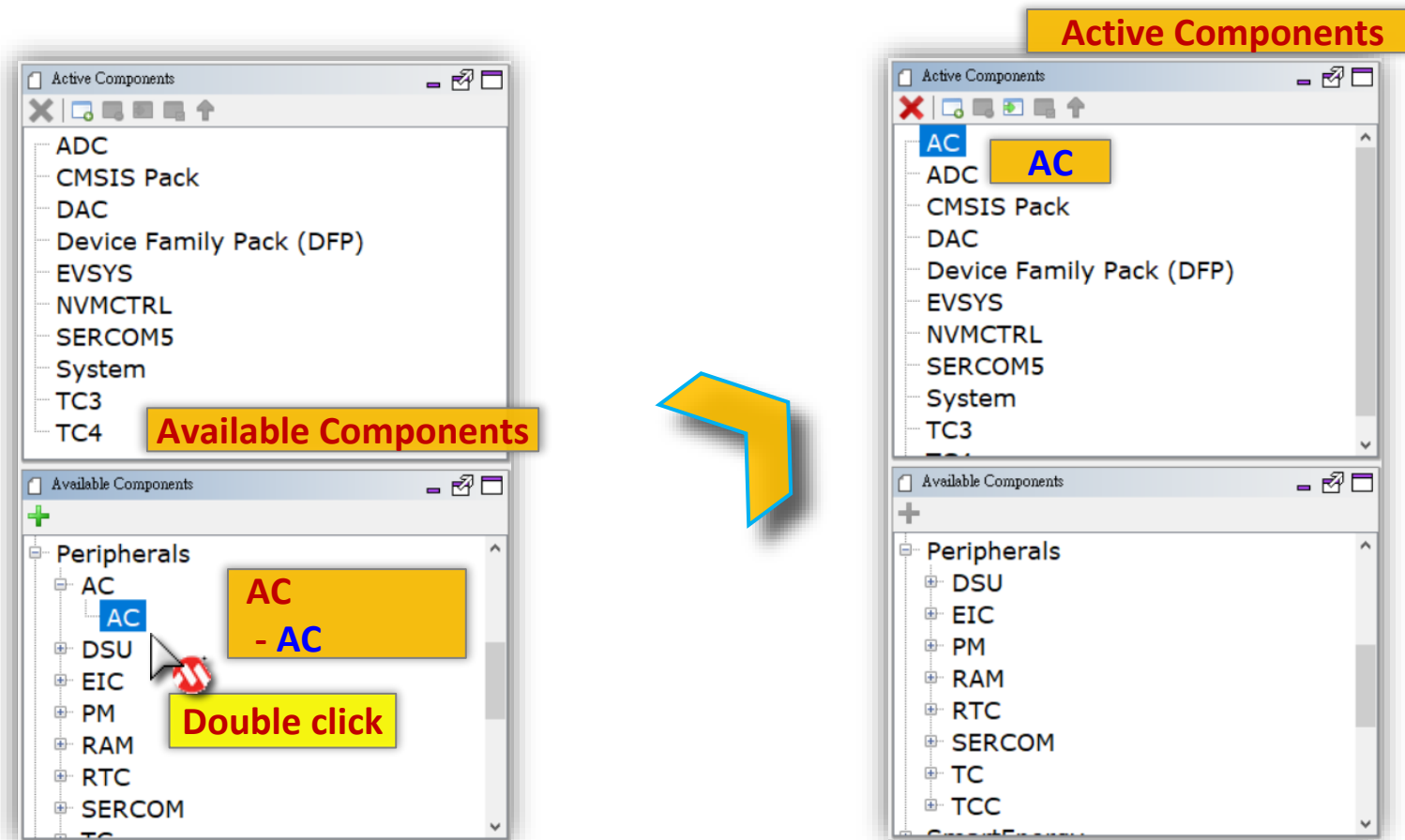


Wire VR1(AIN2_{ADC} ,A1_{Arduino} ,PB08) to (AIN0_{AC} ,A3_{Arduino} ,PA04)

✖ Lab12-5 AC Single Comparator

Step 2

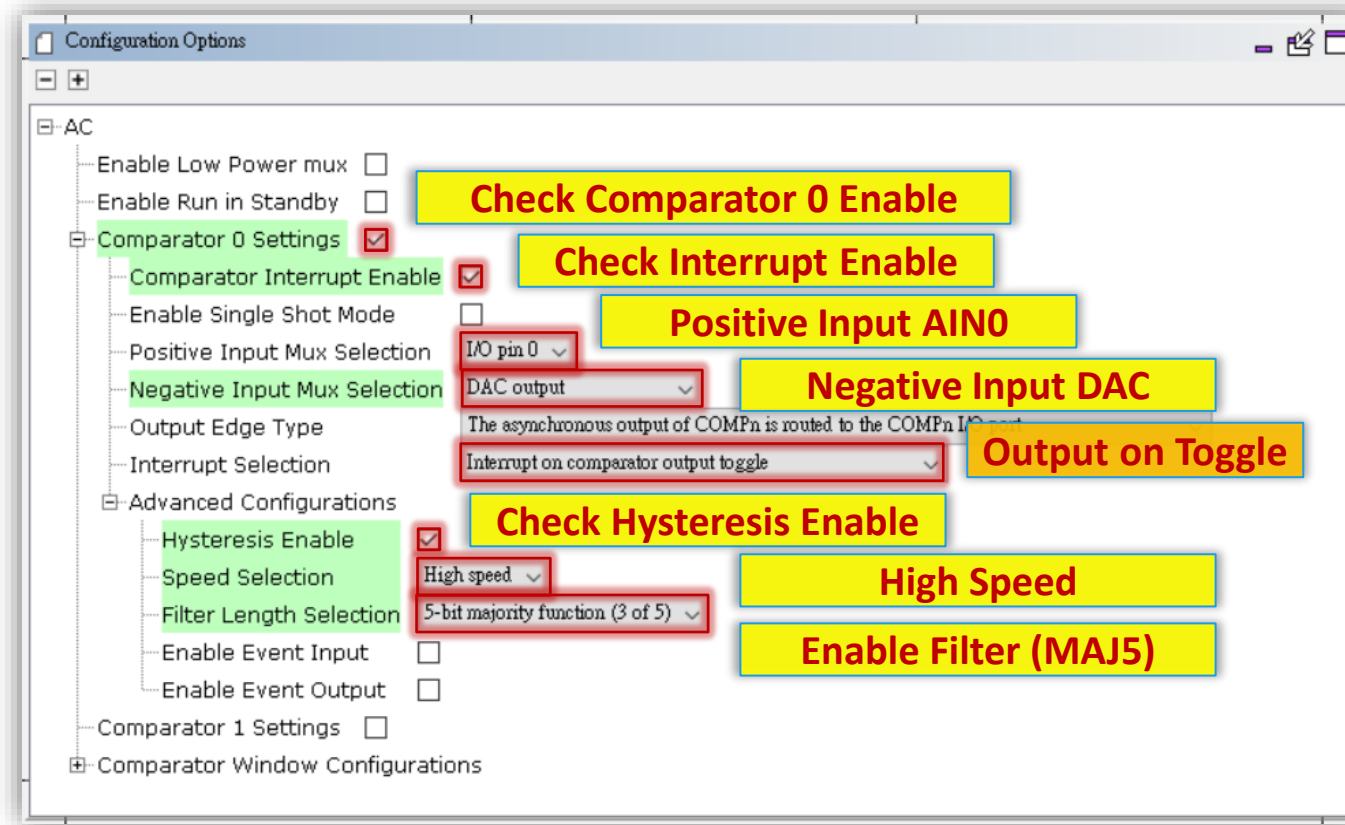
- Find **AC** component in Available Components window.
- Double click **AC** to add to Active Components window.



✂ Lab12-5 AC Single Comparator

Step 3

- Configure **AC** in configuration.
 - Positive : $AIN0_{AC}$ (PA04), Negative : DAC
 - Continuous & Fast Mode, Enable Hysteresis, Enable Digital Filter (MAJ5), Interrupt on toggle



✖ Lab12-5 AC Single Comparator

Step 4

- Enable AC Input (AIN0) to PA04

Pin Table

Package: TQFP48

Module	Function	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
AC	AC_AIN0																															
	AC_AIN1																															
	AC_AIN2																															
	AC_AIN3																															
	AC_CMP0																															
	AC_CMP1																															

PA04

Click Enable

Pin Table

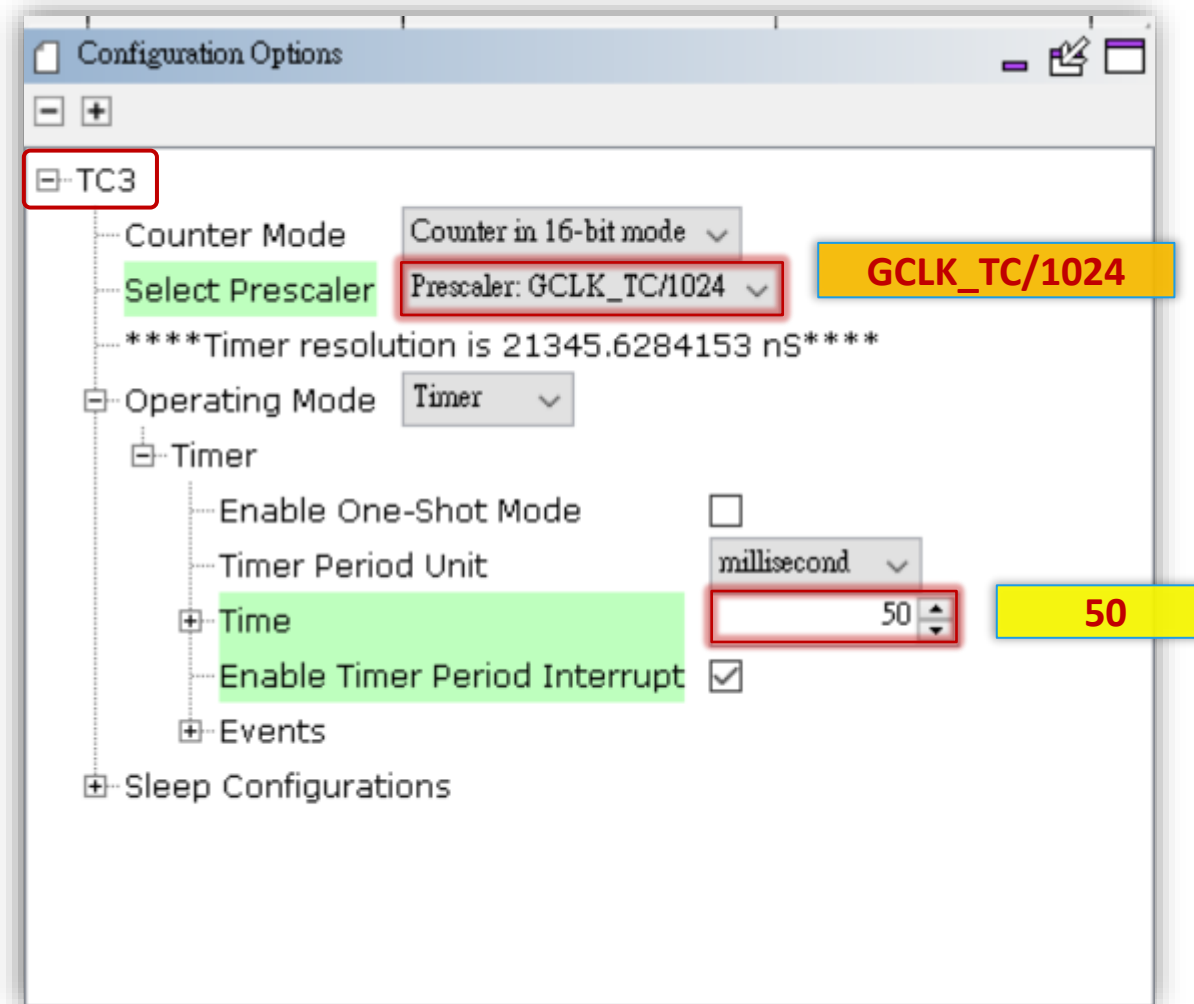
Package: TQFP48

Module	Function	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
AC	AC_AIN0																															
	AC_AIN1																															
	AC_AIN2																															
	AC_AIN3																															
	AC_CMP0																															
	AC_CMP1																															

AC_AIN0

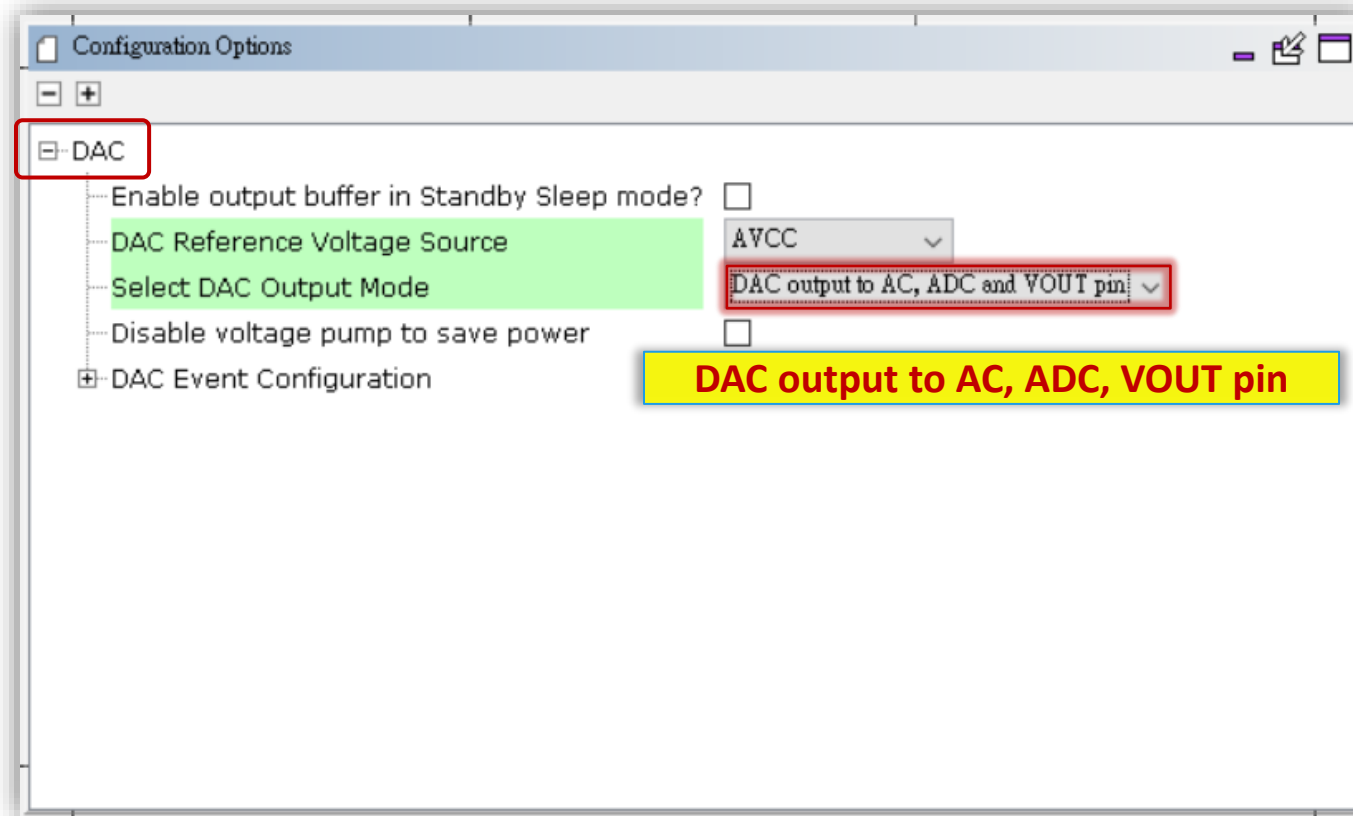
✖ Lab12-5 AC Single Comparator

Step 5



✖ Lab12-5 AC Single Comparator

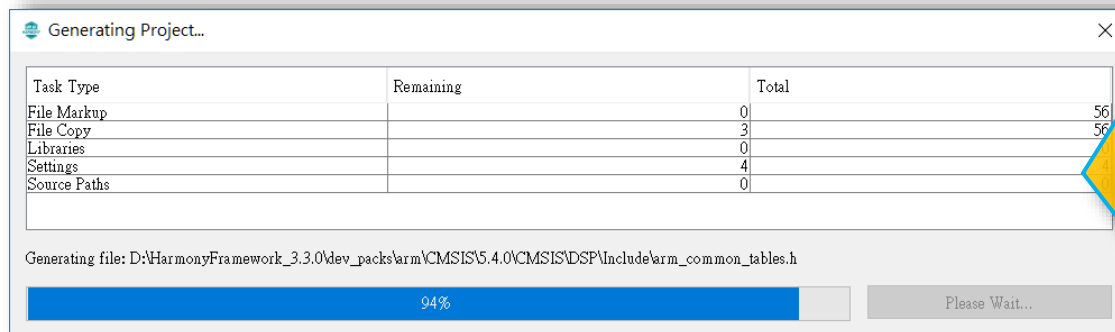
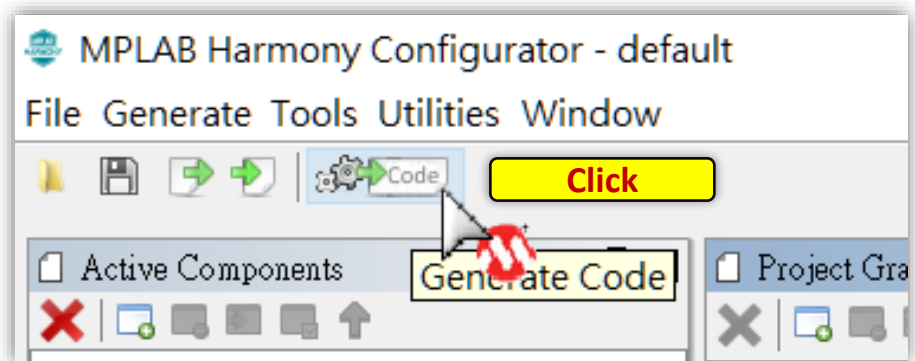
Step 6



⚙️ Lab12-5 AC Single Comparator

Step 7

- Click  to Generate Code.



🔧 Lab12-5 AC Single Comparator

Step 8

```
// TODO 12-5.01
//uint16_t ADC_Result;

int main (void)
{
    ...
    while ( true )
    {
        ...
        if (TC3_HasExpired)
        {
            ...
            // TODO 12-5.02
            //myprintf("Hello World!\r\n");
            ...
            // TODO 12-5.03
            //ADC_Result = ADC_ConversionResultGet();
            ...
            // TODO 12-5.04
            //myprintf("VR1 Value : %d\r\n", ADC_Result);
            //myprintf("DAC Value : %d\r\n", ADC_Result);
        }
    }
    return ( EXIT_FAILURE);
}
```

Lab12-5 AC Single Comparator

Step 9

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

    while ( true )
    {
        ...

        if (USART5_IsReceived)
        {
            ...

            // TODO 12-5.05
            //myprintf("\r\nReceived Data : %1c\r\n", USART5_ReceiveData[0]);
        }
    }
    return ( EXIT_FAILURE);
}
```

Lab12-5 AC Single Comparator

Step 10

```
// TODO 12-5.06
uint16_t ADC_Result[2];
volatile uint8_t ADC_BufferIndex = 0;

// TODO 12-5.07
volatile uint8_t AC0_HasActivated = 0;
volatile uint16_t AC0_ActiveCounter = 0;
volatile uint8_t AC0_OutputStatus = 0;

// TODO 12-5.08
void AC0_Actived(uint8_t int_flags, uintptr_t context)
{
    AC0_HasActivated = 1;
    LED3_Toggle();
    AC0_OutputStatus = (int_flags & AC_STATUSA_STATE0_Msk) >> AC_STATUSA_STATE0_Pos;
}

int main (void)
{
    ...
    while ( true )
    {
        ...
    }
    return ( EXIT_FAILURE);
}
```

Lab12-5 AC Single Comparator

Step 11

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

    // TODO 12-5.09
    AC_CallbackRegister(AC0_Actived, (uintptr_t) NULL);

    // TODO 12-5.10
    myprintf("\033[2J"); // Clear screen
    myprintf("\033[?25l"); // Hide cursor
    myprintf("\033[1;1H");
    myprintf("Lab12-5 AC Single Comparator\r\n");
    myprintf("-----");
    myprintf("\033[6;1H");
    myprintf("AC Active Counter ! [%4d]\r\n", AC0_ActiveCounter);
    myprintf("AC Output is Unknown    \r\n");

    while ( true )
    {
        ...
    }
    return ( EXIT_FAILURE);
}
```

Lab12-5 AC Single Comparator

Step 12

```
int main (void)
{
    ...
    while ( true )
    {
        ...
        if (TC3_HasExpired)
        {
            ...
            // TODO 12-5.11
            switch (ADC_BufferIndex)
            {
            default:
            case 0:
                ADC_ChannelSelect(ADC_POSINPUT_PIN2, ADC_NEGINPUT_GND);
                break;

            case 1:
                ADC_ChannelSelect(ADC_POSINPUT_DAC, ADC_NEGINPUT_GND);
                break;
            }
        }
    }
    return ( EXIT_FAILURE);
}
```

Lab12-5 AC Single Comparator

Step 13

```
int main (void)
{
    ...
    while ( true )
    {
        ...
        if (TC3_HasExpired)
        {
            ...
            // TODO 12-5.12
            ADC_Result[ADC_BufferIndex] = ADC_ConversionResultGet();
            if (ADC_BufferIndex++ > 1)
                ADC_BufferIndex = 0;

            ...
            // TODO 12-5.13
            myprintf("\033[3;1H");
            myprintf("%4d, (Positive Input), VR1 Value by ADC\r\n", ADC_Result[0]);
            myprintf("%4d, (Negative Input), DAC Value by ADC\r\n", ADC_Result[1]);
            myprintf("-----");
        }
    }
    return ( EXIT_FAILURE);
}
```


Lab12-5 AC Single Comparator

Step 14

```
int main (void)
{
    SYS_Initialize(NULL);
    ...
    while ( true )
    {
        // TODO 12-5.14
        if (AC0_HasActivated)
        {
            AC0_HasActivated = 0;

            if (++AC0_ActiveCounter > 9999)
                AC0_ActiveCounter = 0;

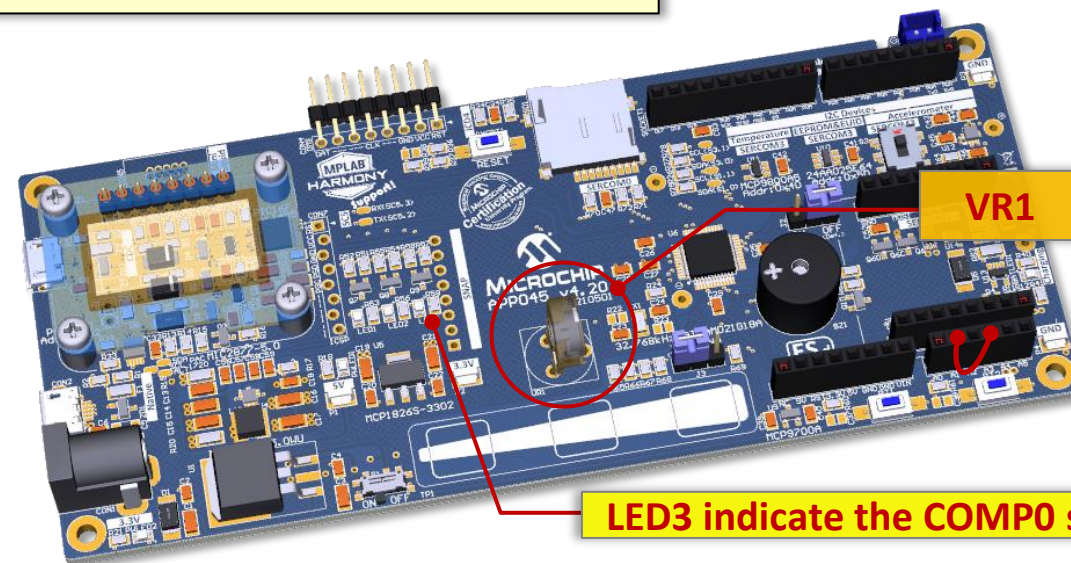
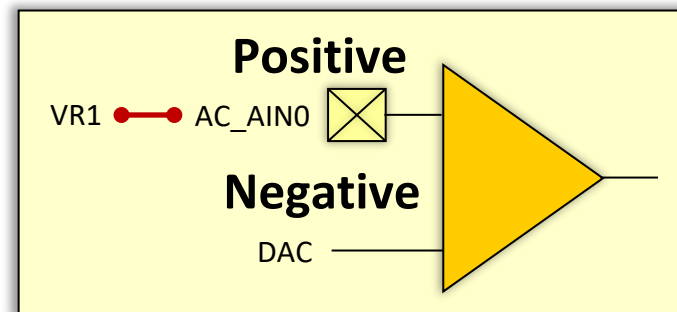
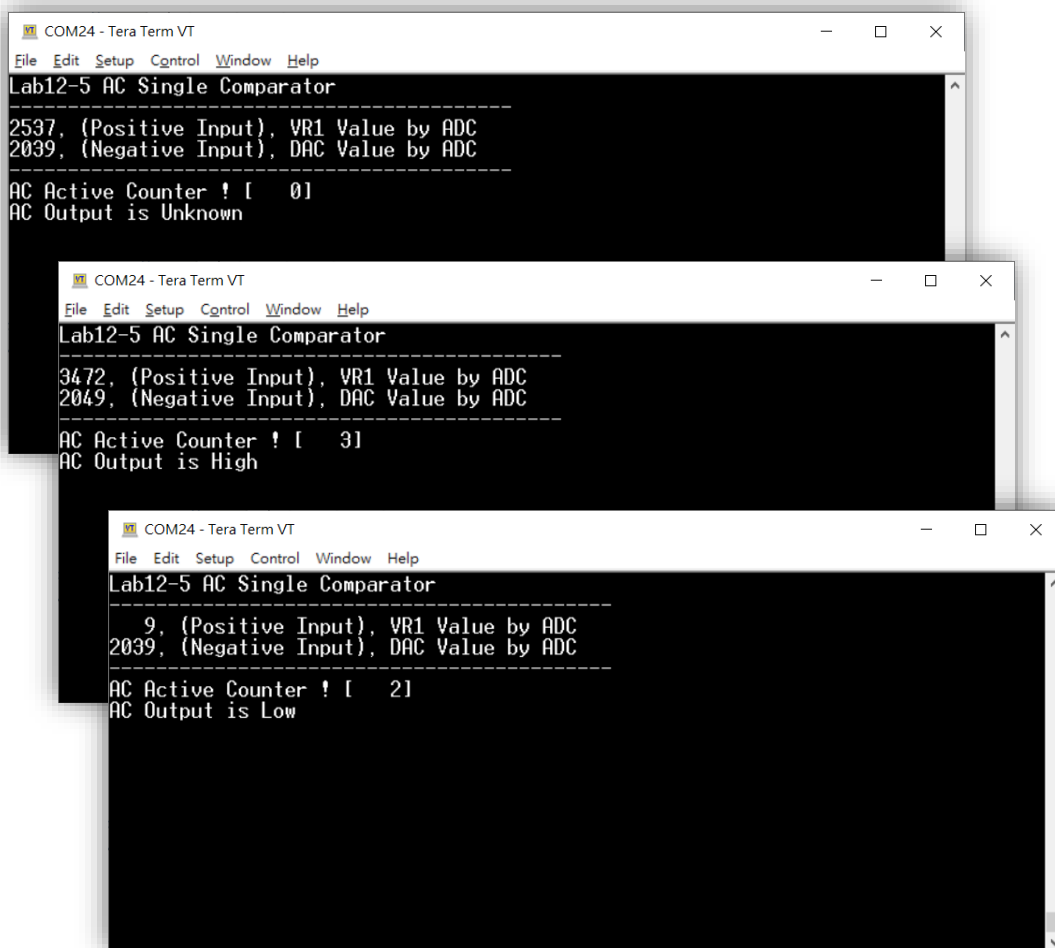
            myprintf("\033[6;1H");
            myprintf("AC Active Counter ! [%4d]\r\n", AC0_ActiveCounter);

            if (AC0_OutputStatus)
                myprintf("AC Output is High           \r\n");
            else
                myprintf("AC Output is Low           \r\n");
        }
    }
    return ( EXIT_FAILURE);
}
```

⚡ Lab12-5 AC Single Comparator

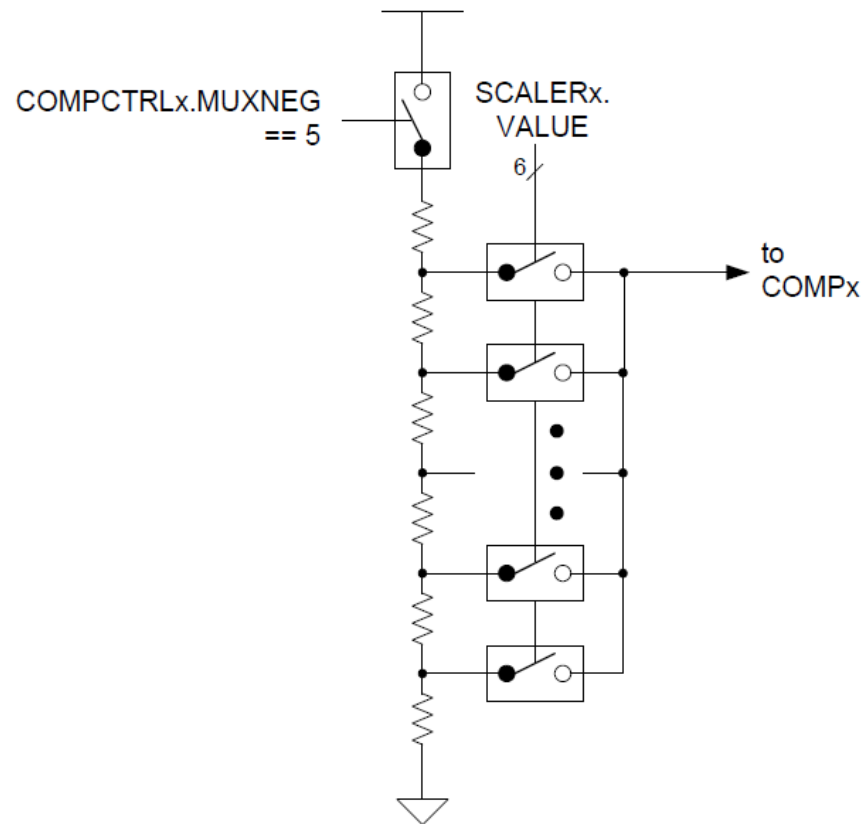
Result

- LED3 indicate the status of COMP0 output.

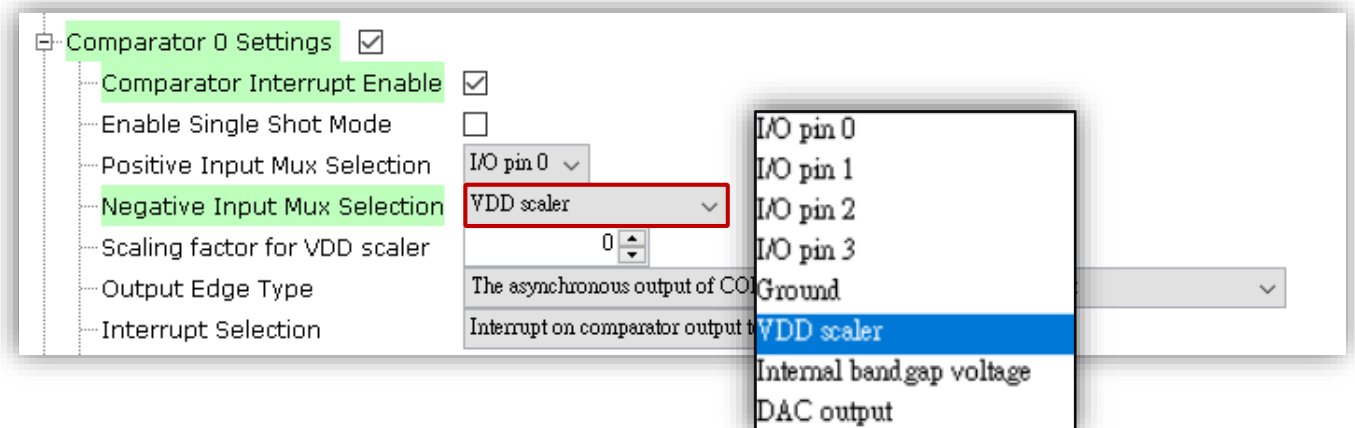


VDD Scaler

- The VDDANA scaler generates a reference voltage that is a fraction of the device's supply voltage. Dedicated for each comparator with 64 levels.

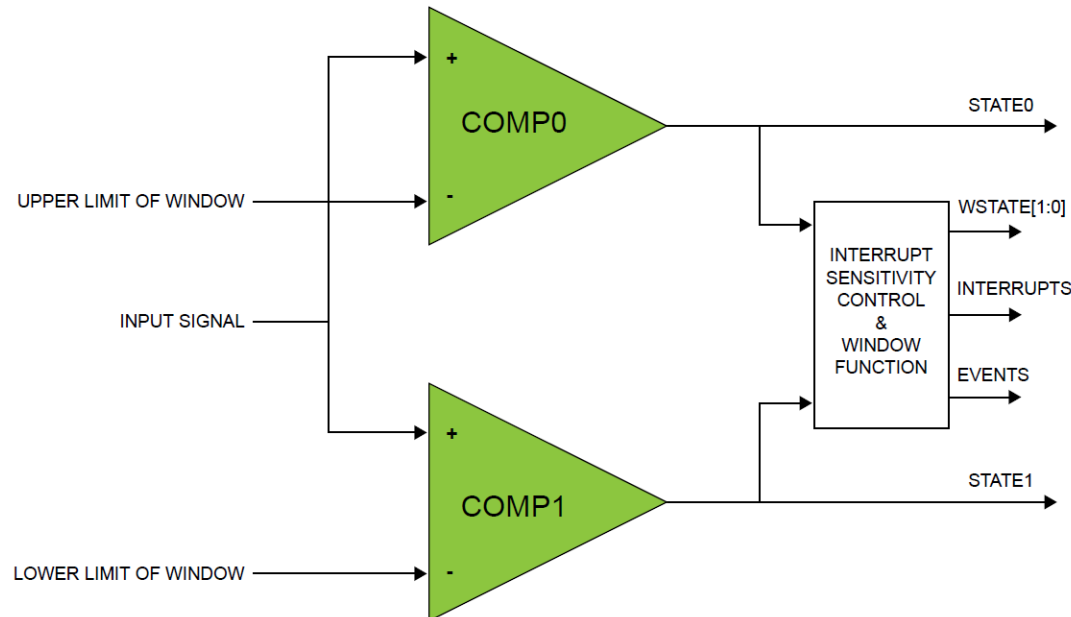


Value	Name	Description
0x0	PIN0	I/O pin 0
0x1	PIN1	I/O pin 1
0x2	PIN2	I/O pin 2
0x3	PIN3	I/O pin 3
0x4	GND	Ground
0x5	VSCALE	VDD scaler
0x6	BANDGAP	Internal bandgap voltage
0x7	DAC	DAC output



Window Mode

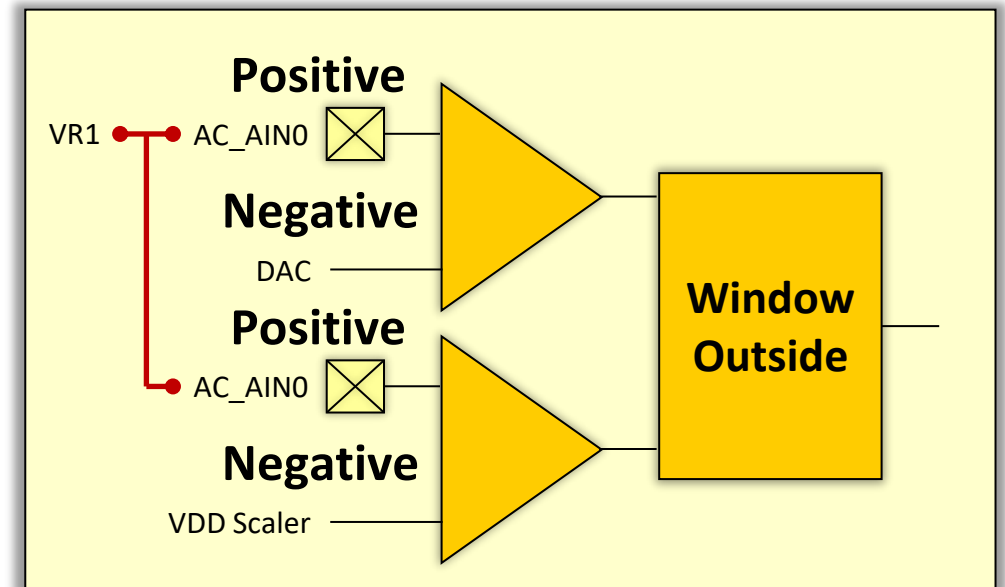
- Each comparator pair can be configured to work together in window mode.
- In this mode, a voltage range is defined, and the comparators give information about whether an input signal is within this range or not.



Value	Name	Description
0x0	ABOVE	Interrupt on signal above window
0x1	INSIDE	Interrupt on signal inside window
0x2	BELOW	Interrupt on signal below window
0x3	OUTSIDE	Interrupt on signal outside window

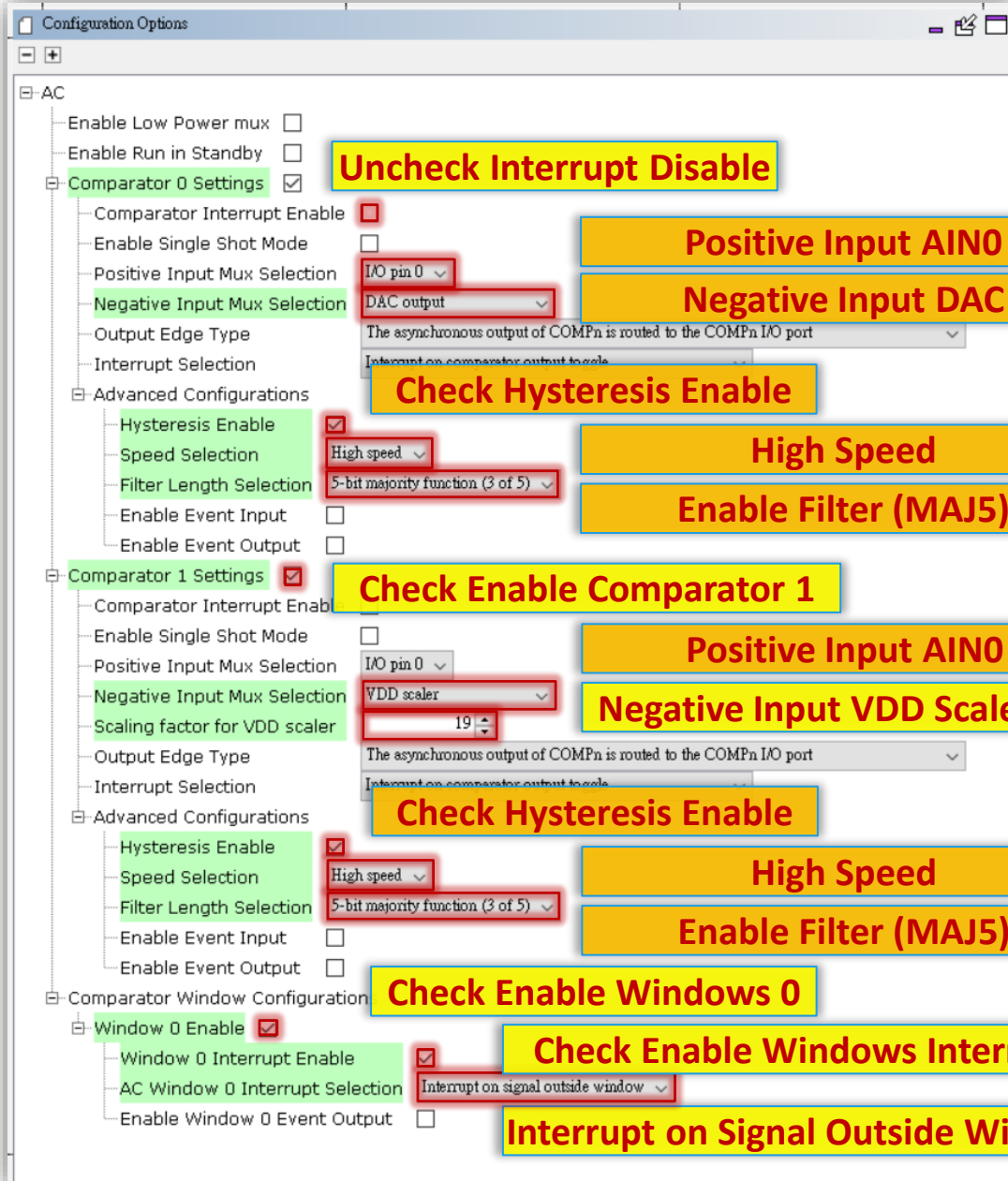
⚙️ Lab12-6 AC Window

- Please continue from [SAM2001 Lab12-5 \(AC Single Comparator\)](#)
- Try to enable AC Window mode.
- Set the Window mode,
 - Upper Limit is DAC (3.0V recommended)
 - Lower Limit is VDD Scaler (1.0V recommended)
 - Interrupt on Signal “Outside” Window
- Use AC to compare VR1 voltage, then LED3 will indicate the status of AC output, when voltage Outside window.
 - $VR1 \text{ Voltage} > 3.0V \ || \ VR1 \text{ Voltage} < 1.0V$
- **Let's go !**



Lab12-6 AC Window

Step 1



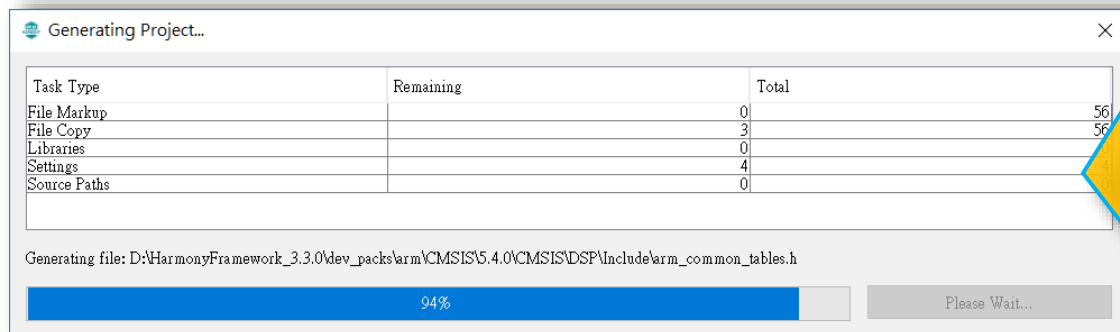
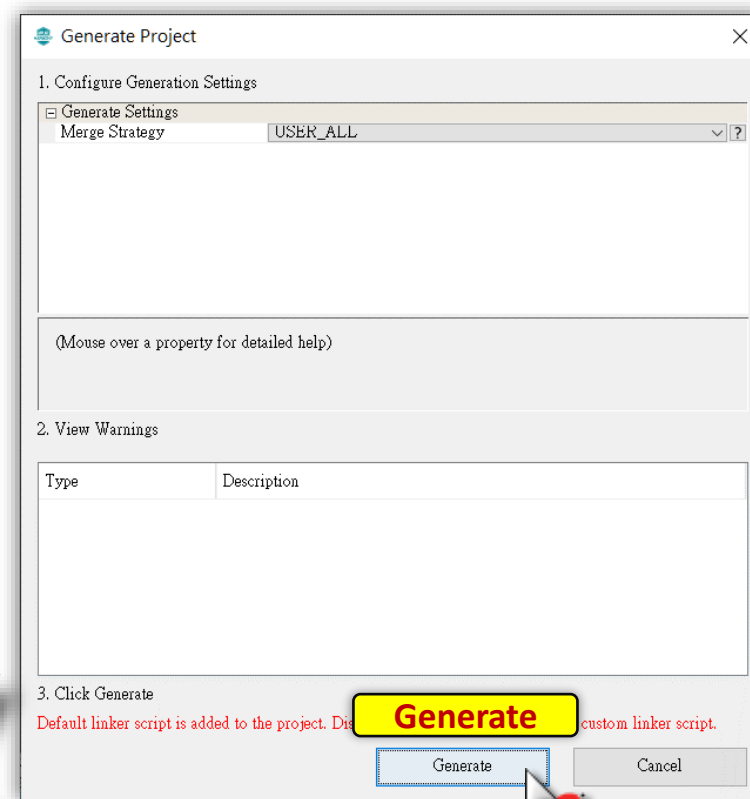
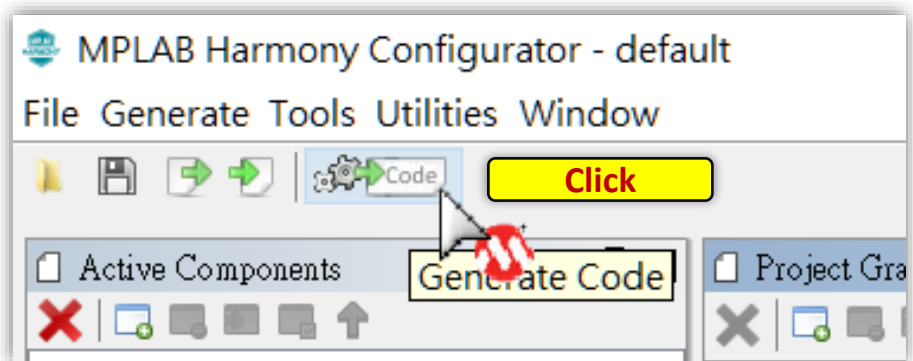
Configuration Options

- AC
 - Enable Low Power mux ☐
 - Enable Run in Standby ☐
 - Comparator 0 Settings ☒
 - Comparator Interrupt Enable ☐ **Uncheck Interrupt Disable**
 - Enable Single Shot Mode ☐
 - Positive Input Mux Selection I/O pin 0 **Positive Input AIN0**
 - Negative Input Mux Selection DAC output **Negative Input DAC**
 - Output Edge Type The asynchronous output of COMPn is routed to the COMPn I/O port
 - Interrupt Selection Interrupt on comparator output to scale
 - Advanced Configurations
 - Hysteresis Enable ☒ **Check Hysteresis Enable**
 - Speed Selection High speed **High Speed**
 - Filter Length Selection 5-bit majority function (3 of 5) **Enable Filter (MAJ5)**
 - Enable Event Input ☐
 - Enable Event Output ☐
 - Comparator 1 Settings ☒
 - Comparator Interrupt Enable ☐ **Check Enable Comparator 1**
 - Enable Single Shot Mode ☐
 - Positive Input Mux Selection I/O pin 0 **Positive Input AIN0**
 - Negative Input Mux Selection VDD scaler **Negative Input VDD Scaler, 19**
 - Scaling factor for VDD scaler 19
 - Output Edge Type The asynchronous output of COMPn is routed to the COMPn I/O port
 - Interrupt Selection Interrupt on comparator output to scale
 - Advanced Configurations
 - Hysteresis Enable ☒ **Check Hysteresis Enable**
 - Speed Selection High speed **High Speed**
 - Filter Length Selection 5-bit majority function (3 of 5) **Enable Filter (MAJ5)**
 - Enable Event Input ☐
 - Enable Event Output ☐
 - Comparator Window Configuration
 - Window 0 Enable ☒ **Check Enable Windows 0**
 - Window 0 Interrupt Enable ☒ **Check Enable Windows Interrupt**
 - AC Window 0 Interrupt Selection Interrupt on signal outside window **Interrupt on Signal Outside Window**
 - Enable Window 0 Event Output ☐

⚙️ Lab12-6 AC Window

Step 2

- Click  to Generate Code.



Lab12-6 AC Window

Step 3

```
int main (void)
{
    ...
    // TODO 12-6.01
    //myprintf("Lab12-5 AC Single Comparator\r\n");
    ...
    // TODO 12-6.02
    //myprintf("\033[6;1H");
    //myprintf("AC Active Counter ! [%4d]\r\n", AC0_ActiveCounter);
    //myprintf("AC Output is Unknown \r\n");
    while ( true )
    {
        ...
        // TODO 12-6.03
        //myprintf("\033[3;1H");
        //myprintf("%4d, (Positive Input), VR1 Value by ADC\r\n", ADC_Result[0]);
        //myprintf("%4d, (Negative Input), DAC Value by ADC\r\n", ADC_Result[1]);
        //myprintf("-----");
    }
    return ( EXIT_FAILURE);
}
```


Lab12-6 AC Window

Step 4

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

    while ( true )
    {
        ...

        if (AC0_HasActivated)
        {
            ...
            // TODO 12-6.04
            //myprintf("\033[6;1H");
            //myprintf("AC Active Counter ! [%4d]\r\n", AC0_ActiveCounter);

            //if (AC0_OutputStatus)
            //myprintf("AC Output is High\r\n");
            //else
            //myprintf("AC Output is Low\r\n");
        }
    }
    return ( EXIT_FAILURE);
}
```

Lab12-6 AC Window

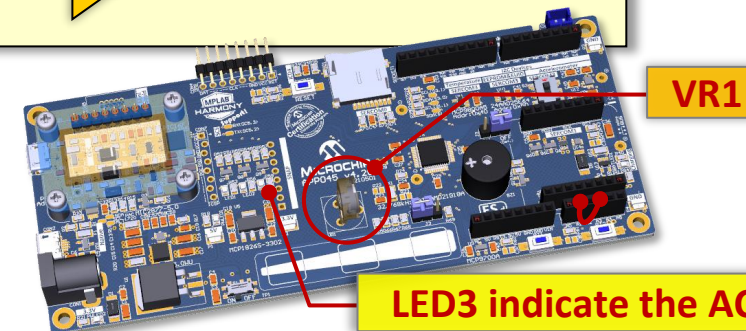
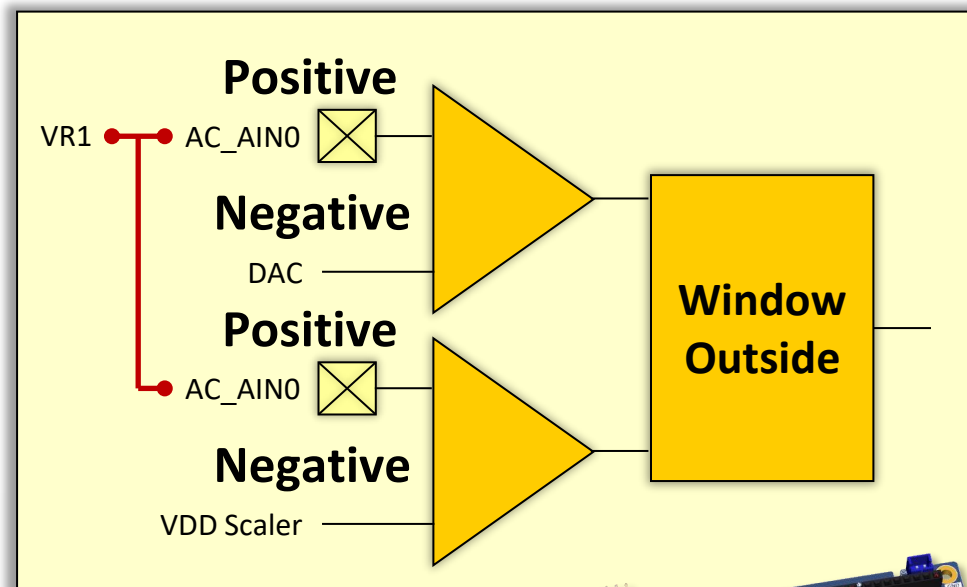
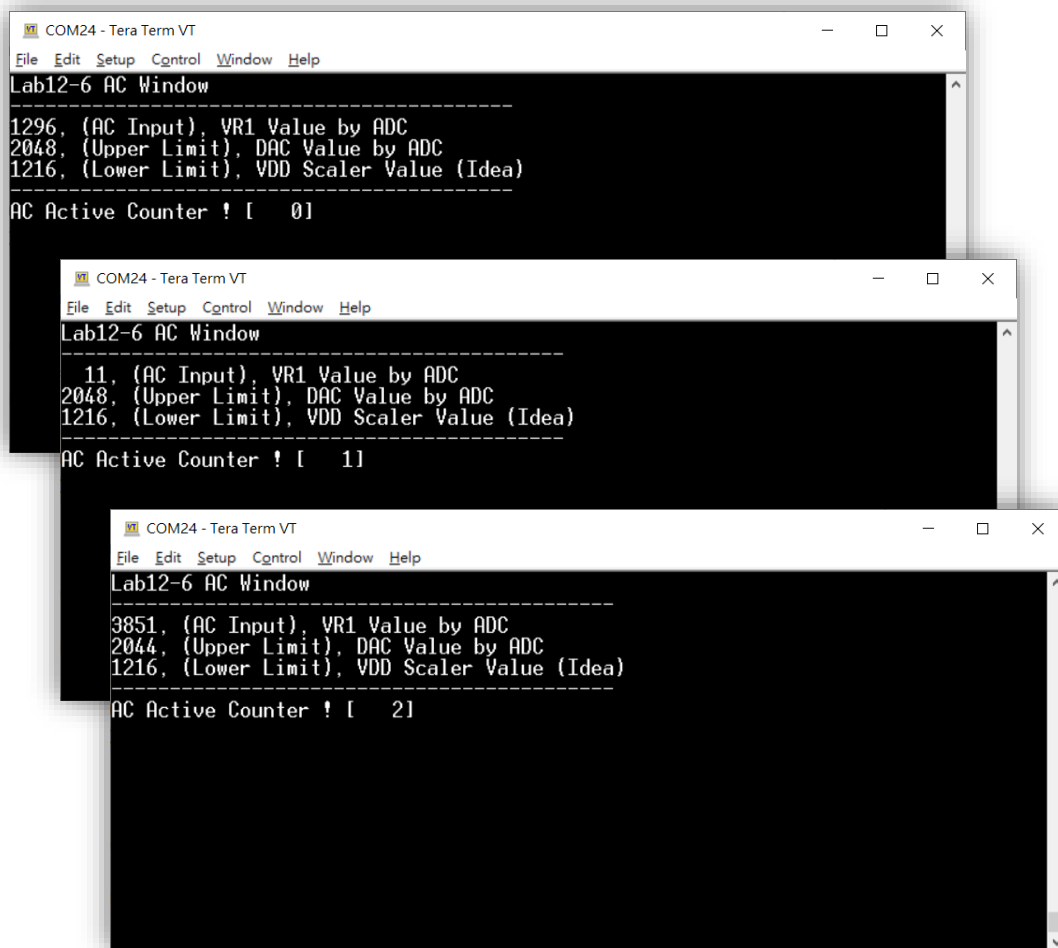
Step 5

```
int main (void)
{
    // TODO 12-6.05
    DAC_DataWrite((uint16_t) 930);
    // TODO 12-6.06
    myprintf("Lab12-6 AC Window\r\n");
    // TODO 12-6.07
    myprintf("\033[7;1H");
    myprintf("AC Active Counter ! [%4d]\r\n", AC0_ActiveCounter);
    while ( true )
    {
        // TODO 12-6.08
        myprintf("\033[3;1H");
        myprintf("%4d, (AC Input), VR1 Value by ADC\r\n", ADC_Result[0]);
        myprintf("%4d, (Upper Limit), DAC Value by ADC\r\n", ADC_Result[1]);
        myprintf("%4d, (Lower Limit), VDD Scaler Value (Idea)\r\n",
                AC_REGS->AC_SCALER[1] << 6);
        myprintf("-----");
        if (AC0_HasActivated)
        {
            // TODO 12-6.09
            myprintf("\033[7;1H");
            myprintf("AC Active Counter ! [%4d]\r\n", AC0_ActiveCounter);
        }
    }
    return ( EXIT_FAILURE);
}
```

⚙️ Lab12-6 AC Window

Result

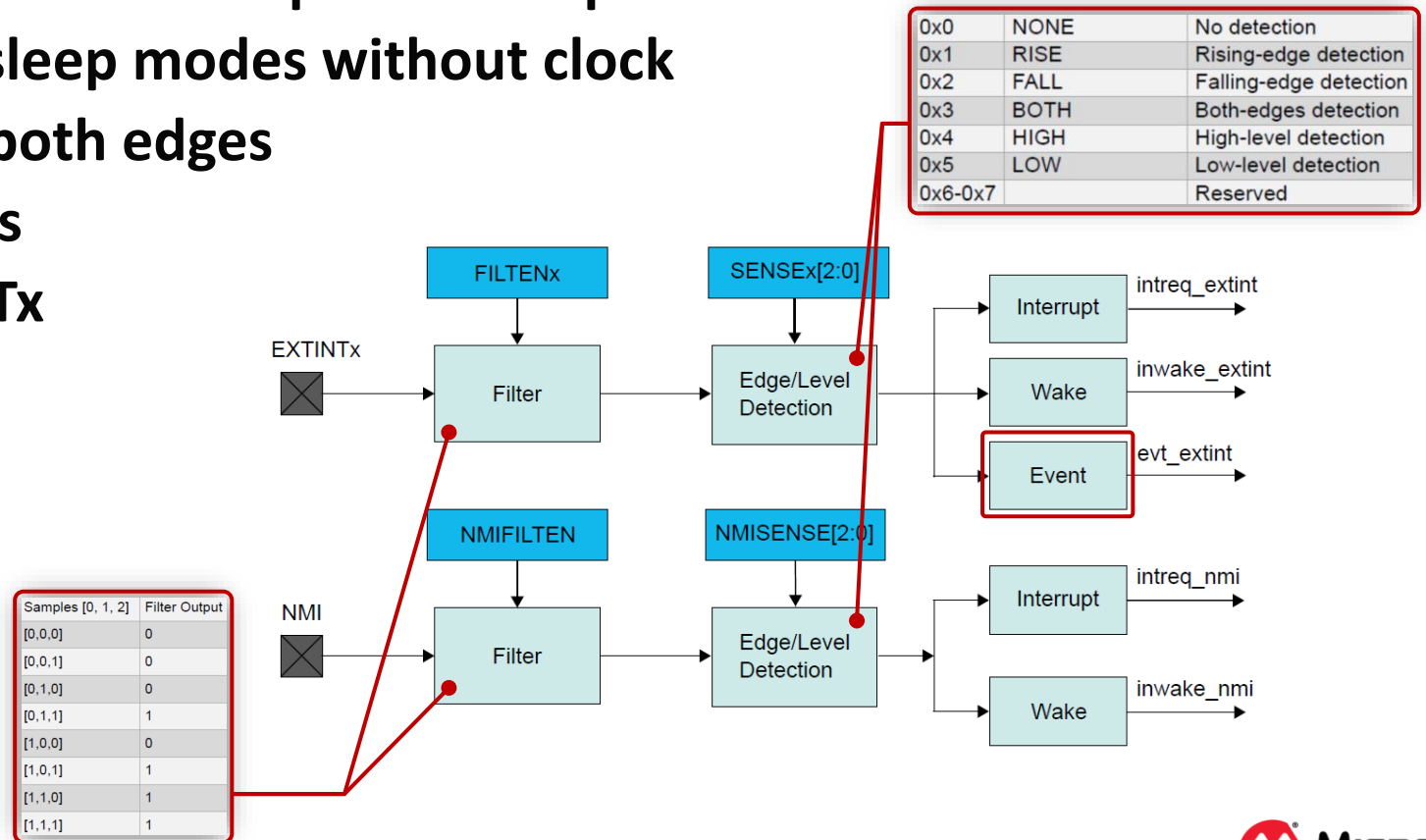
- LED3 will indicate the status of AC output, when voltage outside window.



External Interrupt Controller (EIC)

External Interrupt Controller (EIC)

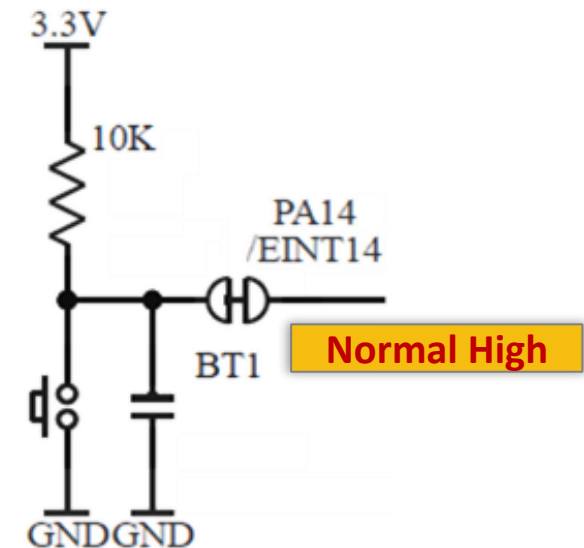
- The External Interrupt Controller (EIC) allows that the external pins to be configured as interrupt lines.
 - Up to 16 external pins (EXTINTx), plus one non-maskable pin (NMI)
 - Dedicated, individually maskable interrupt for each pin
 - Asynchronous interrupts for sleep modes without clock
 - Interrupt on rising, falling or both edges
 - Interrupt on high or low levels
 - Event generation from EXTINTx
 - Filtering of external pins



Edge & Level Detection

- **Edge Detection :**
EIC generate an interrupt event when Edge Detected.
- **Level Detection :**
EIC generate an interrupt event when Level Detected.
- In general,
edge detection is more suit physical button.
An interrupt event be generated by push and release once is more met common sense, even though the button keeping in push and release state.
- Interrupt event will be generated Continuously when the button keeping in push and release state at level detection mode.

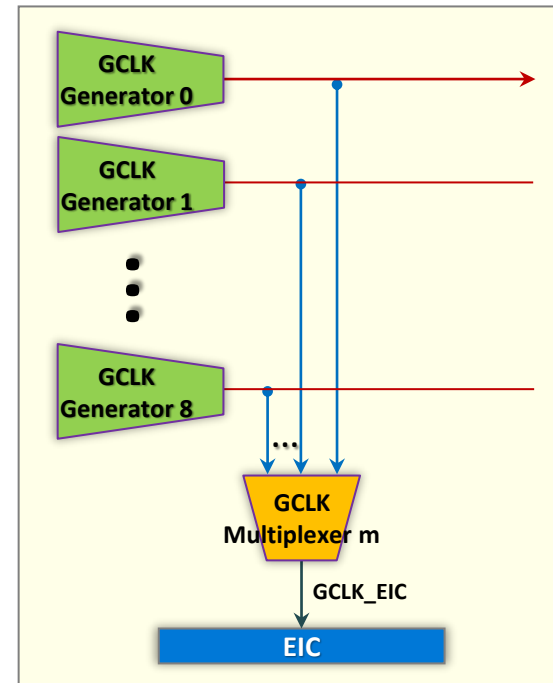
NONE	No detection
RISE	Rising-edge detection
FALL	Falling-edge detection
BOTH	Both-edges detection
HIGH	High-level detection
LOW	Low-level detection
	Reserved



Vote Filter

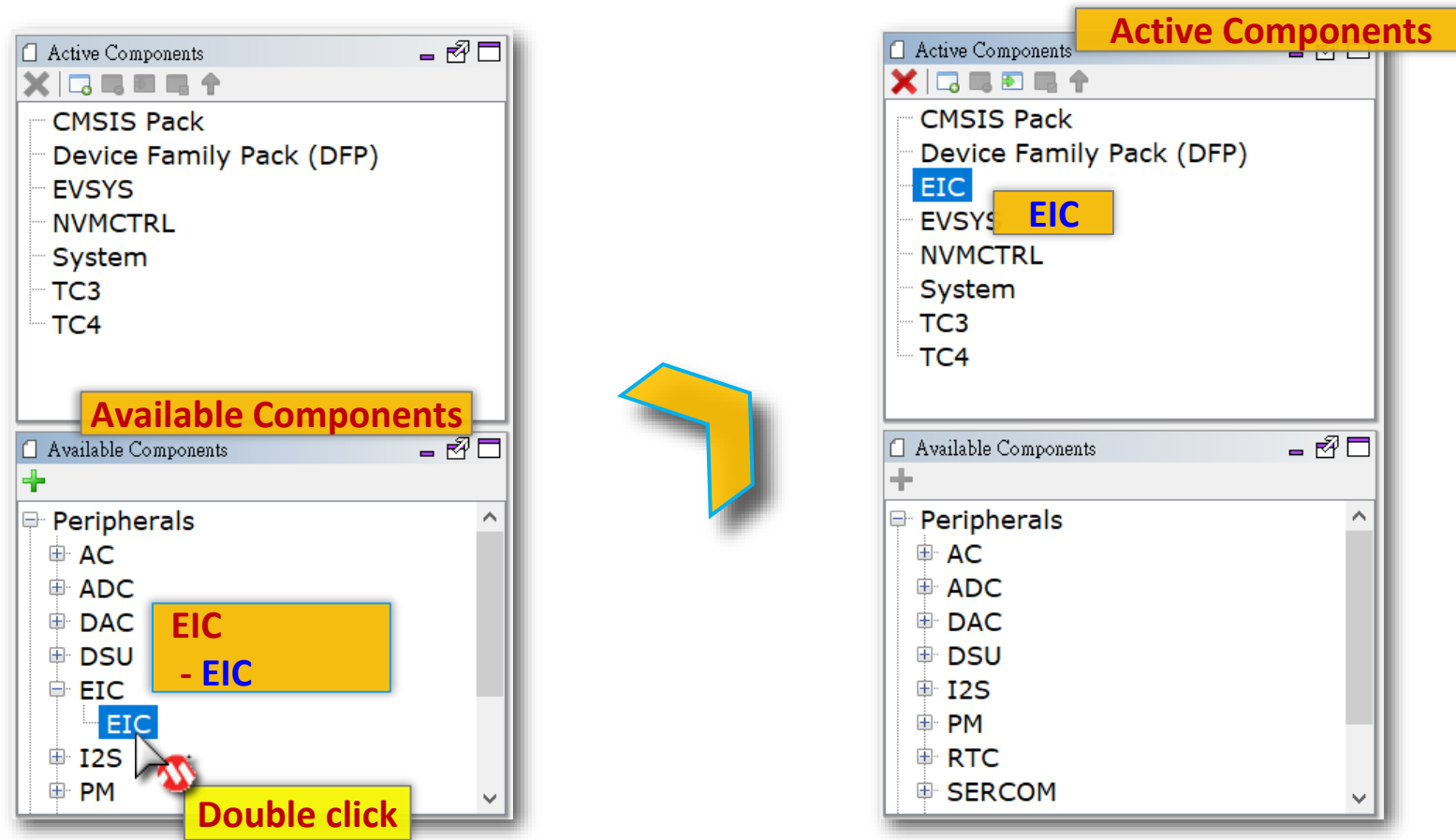
- External pin can be filtered by a majority vote filtering.
Three times capture the pin value with GCLK_EIC and outputs the value when two or more samples are equal.
- The frequency of GCLK_EIC to decide the capture period. Lower frequency clock is more suit for user interface input, just like physical button.

Samples [0, 1, 2]	Filter Output
[0,0,0]	0
[0,0,1]	0
[0,1,0]	0
[0,1,1]	1
[1,0,0]	0
[1,0,1]	1
[1,1,0]	1
[1,1,1]	1



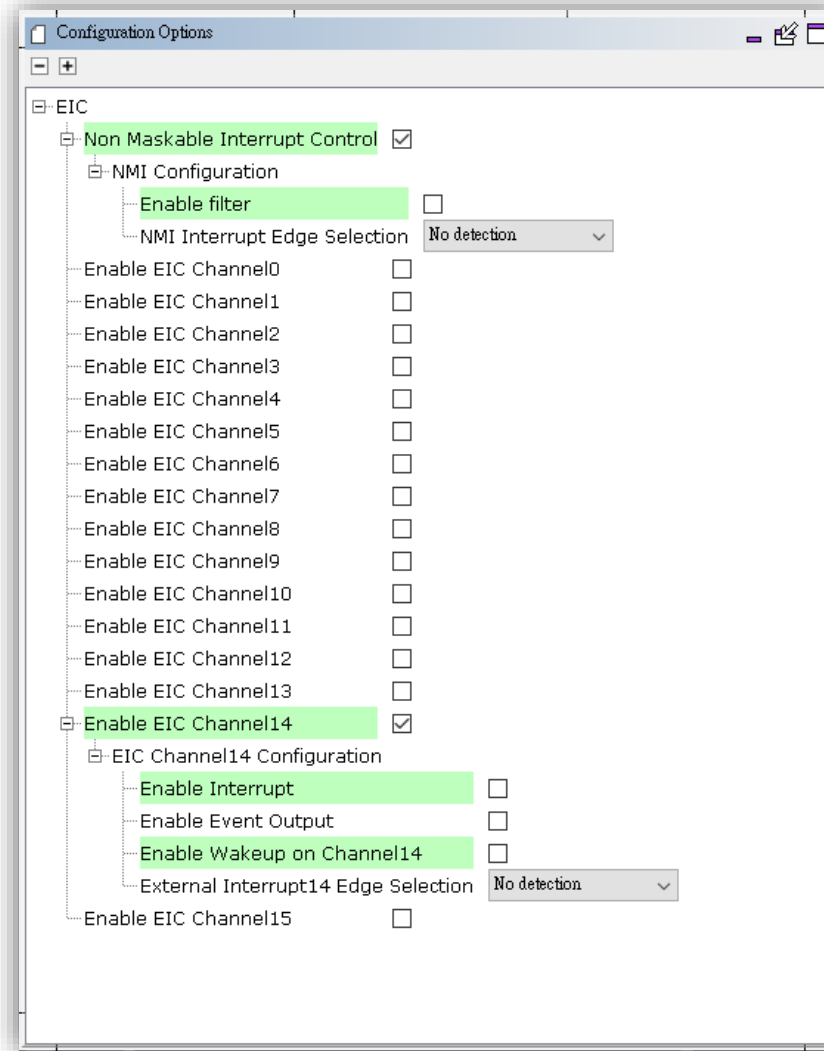
Add EIC Function using MHC

- Find **EIC** component in Available Components window.
- Double click **EIC** to add to Active Components window.



EIC Function

Configuration Options Example



EIC Function

- External Interrupts will all belong to **PINMUX** function group A.

Pin ⁽¹⁾			I/O Pin	Supply	A	B ⁽²⁾⁽³⁾					C	D	E	F	G	H
SAMD2xE	SAMD2xG	SAMD2xJ			EIC	REF	ADC	AC	PTC	DAC	SERCOM ⁽²⁾⁽³⁾	SERCOM-ALT	Tc ⁽⁴⁾ /TCC	TCC	COM	AC/ GCLK
15	23	31	PA14	VDDIO	EXTINT[14]						SERCOM2/ PAD[2]	SERCOM4/ PAD[2]	TC3/WO[0]	TCC0/ WO[4]		GCLK_IO[0]
16	24	32	PA15	VDDIO	EXTINT[15]						SERCOM2/ PAD[3]	SERCOM4/ PAD[3]	TC3/WO[1]	TCC0/ WO[5]		GCLK_IO[1]

Pin Table																											
Package: TQFP48		PA00	PA01	PA02	PA03	ONDANA	VDDANA	ADC_AL.	ADC_AL.	PA04	PA05	PA06	PA07	PA08	PA09	PA10	PA11	VDDIO	ONDIO	PB10	PB11	PA12	PA13	EIC_EX...	EIC_EX...	PA16	LED3
Module	Function	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
EIC	EIC_EXTINT14																										
	EIC_EXTINT15																										
	EIC_EXTINT2																										

EIC Functions & Code Example

Interface Routines & Example

```
void EIC_Initialize ( void );  
void EIC_InterruptEnable( EIC_PIN pin );  
void EIC_InterruptDisable( EIC_PIN pin );  
void EIC_CallbackRegister( EIC_PIN pin , EIC_CALLBACK callback , uintptr_t context );  
void EIC_NMICallbackRegister ( EIC_NMI_CALLBACK callback , uintptr_t context );
```

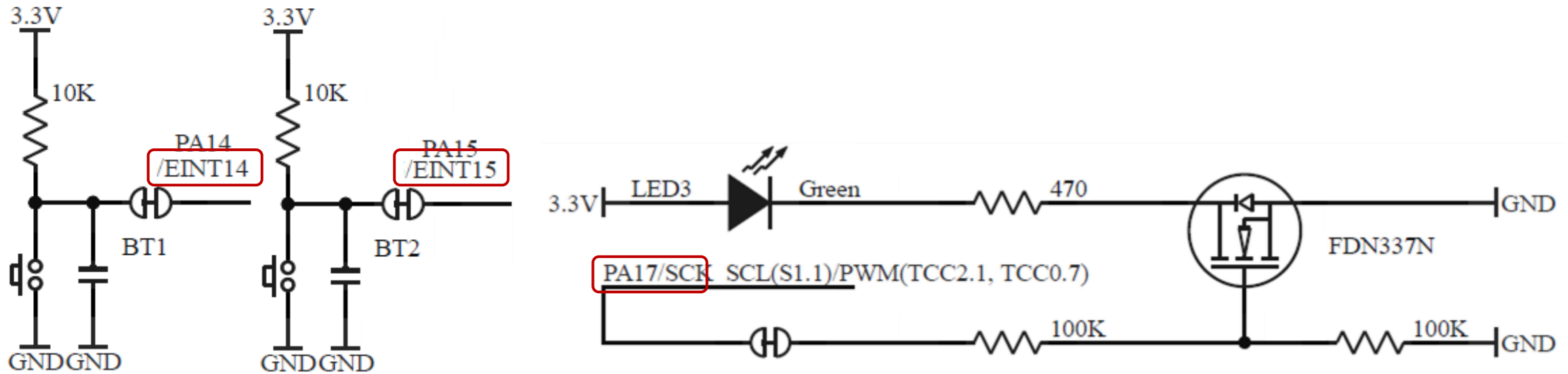
```
void LED_Control(uintptr_t context)  
{  
    LED3_Toggle();  
}  
  
int main (void)  
{  
    SYS_Initialize ( NULL );  
    ...  
    EIC_CallbackRegister(EIC_PIN_14, LED_Control, (uintptr_t) NULL);  
    while( true )  
    {...}  
}
```

🔧 Lab16-1 Button as External Interrupt

- Please continue from [SAM2001 Lab16 \(TCC NPWM Auto DutyCtrl\)](#)
- Try to add [EIC](#) module to project.
- Select [BT1\(PA14\)](#) and [BT2\(PA15\)](#) from GPIO become to External Interrupt input ([EIC_EXTINT](#)) and use the buttons to toggle [LED3\(PA17\)](#).
 - EXTINT14 -> [BT1\(PA14\)](#), Low-Level detection, Enable filter.
 - EXTINT15 -> [BT2\(PA15\)](#), Falling-edge detection, Enable filter.

• **Let's go !**

⚡ Lab16-1 Button as External Interrupt

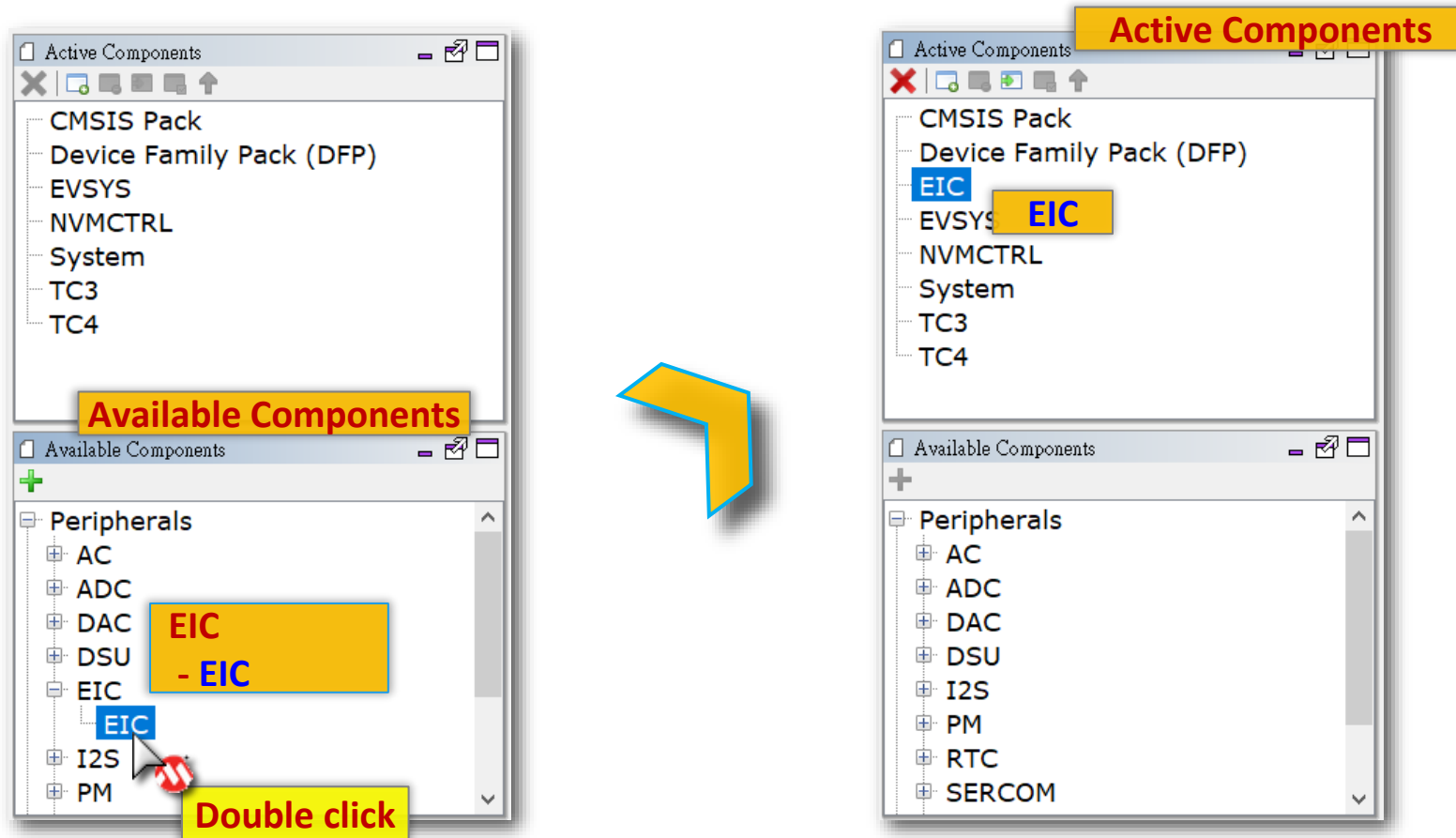


Pin ⁽¹⁾			I/O Pin	Supply	A	B ⁽²⁾⁽³⁾					C	D	E	F	G	H
SAMD2xE	SAMD2xG	SAMD2xJ			EIC	REF	ADC	AC	PTC	DAC	SERCOM ⁽²⁾⁽³⁾	SERCOM-ALT	TC ⁽⁴⁾ /TCC	TCC	COM	AC/ GCLK
15	23	31	PA14	VDDIO	EXTINT[14]						SERCOM2/ PAD[2]	SERCOM4/ PAD[2]	TC3/WO[0]	TCC0/ WO[4]		GCLK_IO[0]
		PA14				EXTINT[14]										
16	24	32	PA15	VDDIO	EXTINT[15]						SERCOM2/ PAD[3]	SERCOM4/ PAD[3]	TC3/WO[1]	TCC0/ WO[5]		GCLK_IO[1]
		PA15				EXTINT[15]										

⚡ Lab16-1 Button as External Interrupt

Step 1

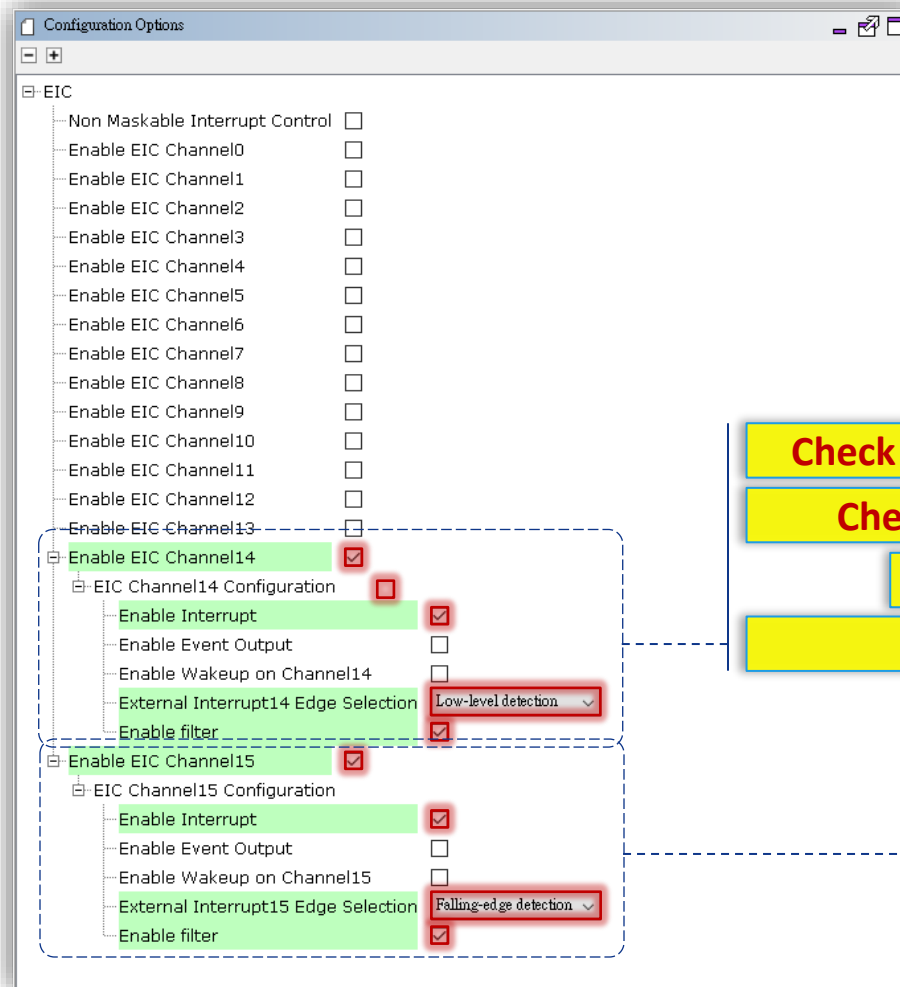
- Find **EIC** component in Available Components window.
- Double click **EIC** to add to Active Components window.



⚡ Lab16-1 Button as External Interrupt

Step 2

- Configure **EIC** in configuration.



Check Enable EIC Channel14

Check Interrupt Enable

Low-Level detection

Enable filter

Check Enable EIC Channel15

Check Interrupt Enable

Falling-edge detection

Enable filter

⚙️ Lab16-1 Button as External Interrupt

Step 3

- Configure **EIC** ASYNC clock to lower frequency in clock configuration.

Peripheral Clock Configuration

Peripheral	Enable	Source	Peripheral Clock Frequency
AC_ANA	☐	GCLK0	--
AC_DIG	☐	GCLK0	--
ADC	☒	GCLK0	47,972,352 Hz
DAC	☐	GCLK0	--
EIC	☒	GCLK0	47,972,352 Hz
EVSYS_0	☐	GCLK0	--
EVSYS_1	☐	GCLK0	--
EVSYS_10	☐	GCLK0	--
EVSYS_11	☐	GCLK0	--
EVSYS_2	☐	GCLK0	--
EVSYS_3	☐	GCLK0	--
EVSYS_4	☐	GCLK0	--

Close



Peripheral Clock Configuration

Peripheral	Enable	Source	Peripheral Clock Frequency
AC_ANA	☐	GCLK0	--
AC_DIG	☐	GCLK0	--
ADC	☒	GCLK0	47,972,352 Hz
DAC	☐	GCLK0	--
EIC	☒	GCLK1	32,768 Hz
EVSYS_0	☐	GCLK0	--
EVSYS_1	☐	GCLK0	--
EVSYS_10	☐	GCLK0	--
EVSYS_11	☐	GCLK0	--
EVSYS_2	☐	GCLK0	--
EVSYS_3	☐	GCLK0	--
EVSYS_4	☐	GCLK0	--

Close

⚡ Lab16-1 Button as External Interrupt

Step 4

- Disable clock and waveform output at PA14, PA15 and PA17.

The image shows four screenshots of the Pin Table tool, illustrating the steps to disable clock and waveform output for specific modules.

Top Left Screenshot (GCLK): The Pin Table for package TQFP48 shows the GCLK module. A red box highlights the GCLK_I03 function at pin PA14, with a yellow callout box saying "Click Disable".

Top Right Screenshot (GCLK): The Pin Table shows the GCLK module. A red box highlights the GCLK_I03 function at pin PA15, with a yellow callout box saying "Click Disable".

Bottom Left Screenshot (TCC2): The Pin Table for package TQFP48 shows the TCC2 module. A red box highlights the TCC2_WO1 function at pin PA17, with a yellow callout box saying "Click Disable".

Bottom Right Screenshot (TCC2): The Pin Table shows the TCC2 module. A red box highlights the TCC2_WO1 function at pin PA17, with a yellow callout box saying "Click Disable".

⚡ Lab16-1 Button as External Interrupt

Step 5

- Enable EIC (EXTINT14, EXTINT15) and GPIO(LED3) at PA14, PA15 and PA17.
 - EXTINT14 and EXTINT15 corresponding to BT1(PA14) and BT2(PA15).

EIC

Package: TQFP48

Module	Function	PA10	PA11	VDDIO	GNDDIO	PB10	PB11	PA12	PA13	PA14	PA15	PA16	PA17	PA18	PA19	LED1	LED2	PA22	PA23	PA24
EIC	EIC_EXTINT13																			
	EIC_EXTINT14																			
	EIC_EXTINT15																			
	EIC_EXTINT2																			
	EIC_EXTINT3																			

Click Enable

EIC

Package: TQFP48

Module	Function	PA10	PA11	VDDIO	GNDDIO	PB10	PB11	PA12	PA13	PA14	PA15	PA16	PA17	PA18	PA19	LED1	LED2	PA22	PA23	PA24
EIC	EIC_EXTINT13																			
	EIC_EXTINT14																			
	EIC_EXTINT15																			
	EIC_EXTINT2																			
	EIC_EXTINT3																			

GPIO

Package: TQFP48

Module	Function	PA10	PA11	VDDIO	GNDDIO	PB10	PB11	PA12	PA13	PA14	PA15	PA16	PA17	PA18	PA19	LED1	LED2	PA22	PA23	PA24
GPIO	GCLK_IO6																			
	GCLK_IO7																			
GPIO	GPIO																			
	I2S_FS0																			
	I2S_MCK0																			

Click Enable

GPIO

Package: TQFP48

Module	Function	PA10	PA11	VDDIO	GNDDIO	PB10	PB11	PA12	PA13	PA14	PA15	PA16	PA17	PA18	PA19	LED1	LED2	PA22	PA23	PA24
GPIO	GCLK_IO6																			
	GCLK_IO7																			
GPIO	GPIO																			
	I2S_FS0																			
	I2S_MCK0																			

⚡ Lab16-1 Button as External Interrupt

Step 6

- Assign customer name for PA17 -> LED3 at Pin Settings

Pin Settings

Order: Pins Table View Easy View

Pin Number	Pin ID	Custom Name	Function	Mode	Direction	Latch	Pull Up	Pull Down
22	PA13		Available	Digital	High Impedance	Low	☐	☐
23	PA14		EIC_EXTINT14	Digital	In	n/a	☐	☐
24	PA15		EIC_EXTINT15	Digital	In	n/a	☐	☐
25	PA16		Available	Digital	High Impedance	Low	☐	☐
26	PA17	GPIO_PA17	Available	Digital	High Impedance	Low	☐	☐

PA17

Click Disable & Re-name

Pin Settings

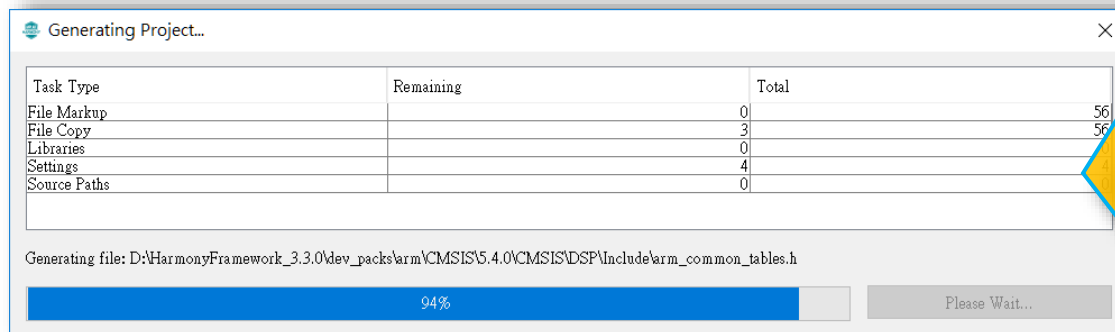
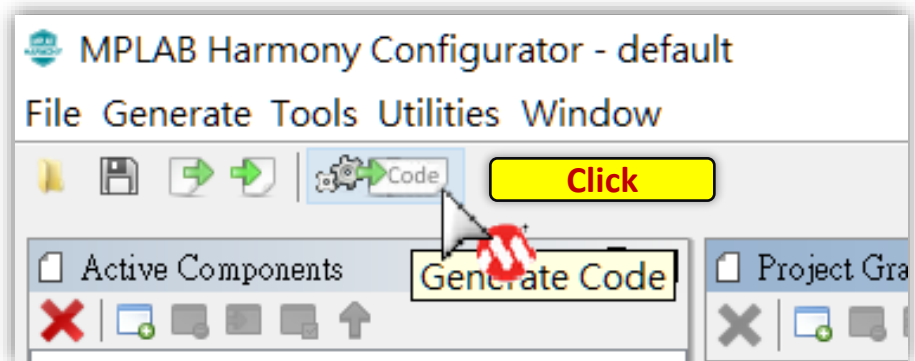
Order: Pins Table View Easy View

Pin Number	Pin ID	Custom Name	Function	Mode	Direction	Latch	Pull Up	Pull Down
22	PA13		Available	Digital	High Impedance	Low	☐	☐
23	PA14		EIC_EXTINT14	Digital	In	n/a	☐	☐
24	PA15		EIC_EXTINT15	Digital	In	n/a	☐	☐
25	PA16		Available	Digital	High Impedance	Low	☐	☐
26	PA17	LED3	GPIO	Digital	Out	Low	☐	☐
27	PA18		Available	Digital	High Impedance	Low	☐	☐

⚡ Lab16-1 Button as External Interrupt

Step 7

- Click  to Generate Code.



🔧 Lab16-1 Button as External Interrupt

Step 8

```
...
// TODO 16-1.01
void LED_Control(uintptr_t context)
{
    static uint32_t i = 0;
    LED3_Toggle();
    for (i = 0; i < 1000000; i++);
}

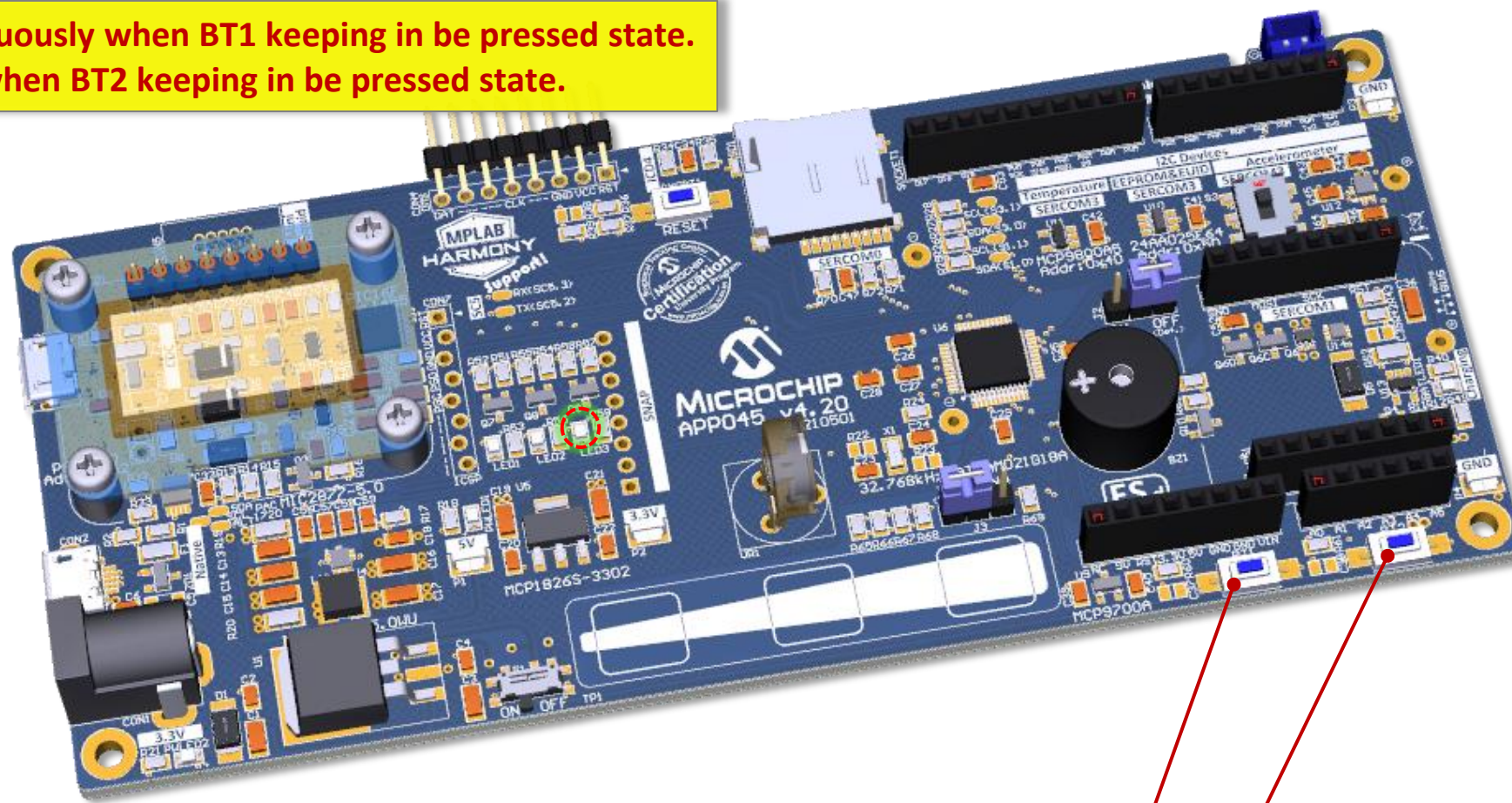
int main (void)
{
    SYS_Initialize(NULL);
    ...
    // TODO 16-1.02
    EIC_CallbackRegister(EIC_PIN_14, LED_Control, (uintptr_t) NULL);
    EIC_CallbackRegister(EIC_PIN_15, LED_Control, (uintptr_t) NULL);

    while ( true )
    {
        SYS_Tasks();
        ...
    }
    return ( EXIT_FAILURE);
}
```

⚙️ Lab16-1 Button as External Interrupt

Result

LED3 toggle Continuously when BT1 keeping in be pressed state.
LED3 toggle once when BT2 keeping in be pressed state.



Push BT1/BT2 to Toggle LED3

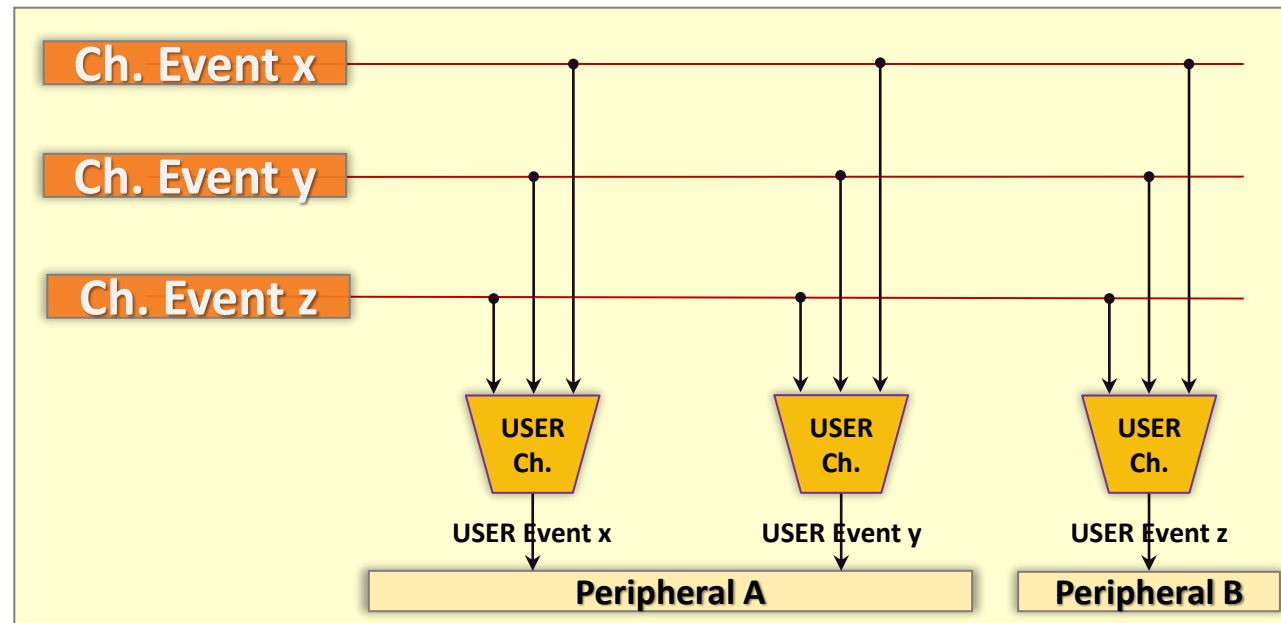
Event System (EVSYS)

Event System

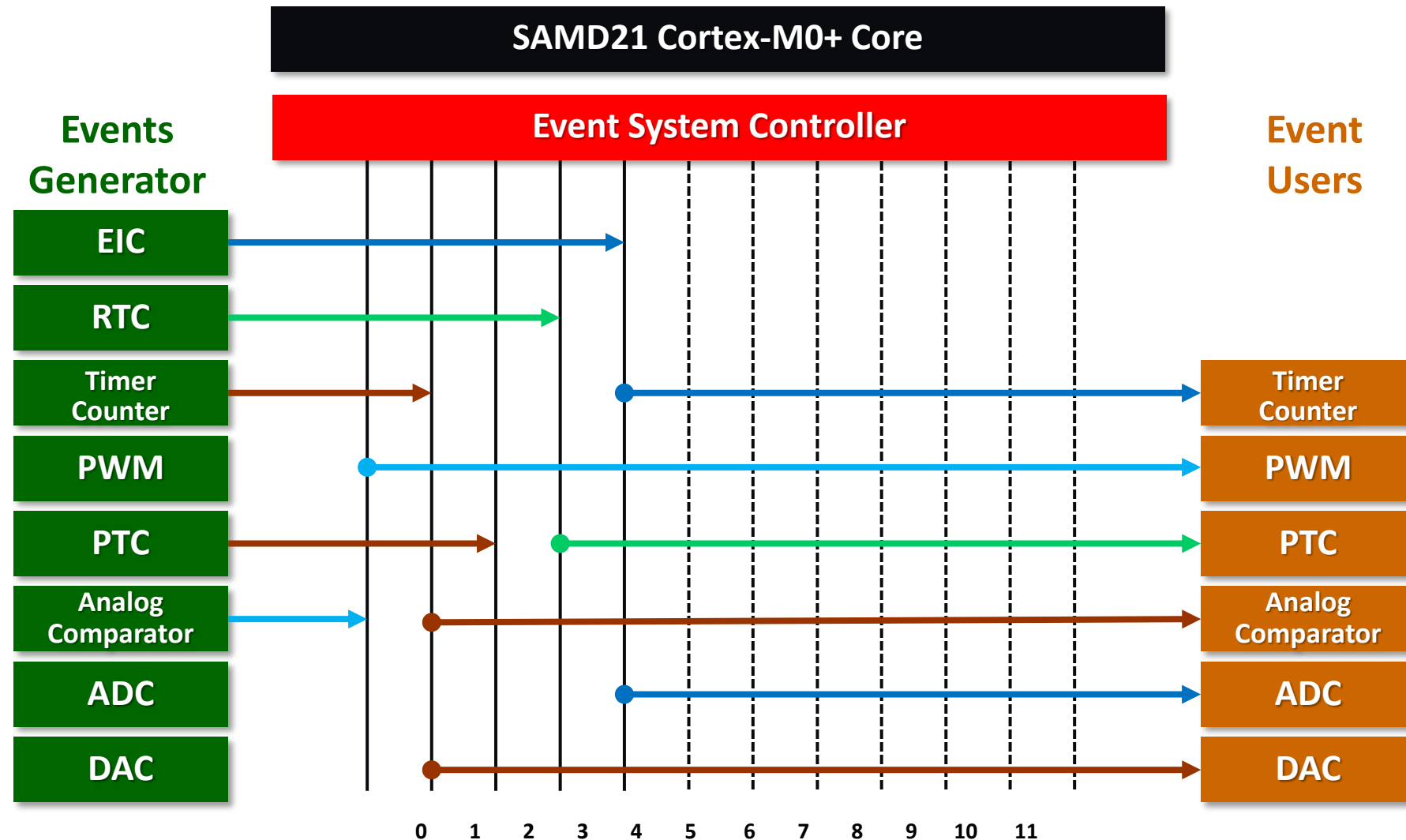
- The Event System (EVSYS) allows autonomous, low-latency and configurable communication between peripherals.
- Role:
 - Peripherals that **respond** to events are called event **users**.
 - Peripherals that **generate events** are called event **generators**.
- A peripheral can have one or more event generators and can have one or more event users, also.
- Communication is made without CPU intervention and without consuming system resources such as bus or RAM bandwidth. This reduces the load on the CPU and other system resources, compared to a traditional interrupt-based system.

Event System

- 12 configurable event channels
- 74 event generators & 29 event users
- Peripherals can be event generators, event users, or both.
- SleepWalking and interrupt for operation in sleep modes.
- Software event generation.
- 3 path
 - Asynchronous
 - Resynchronized
 - Synchronous

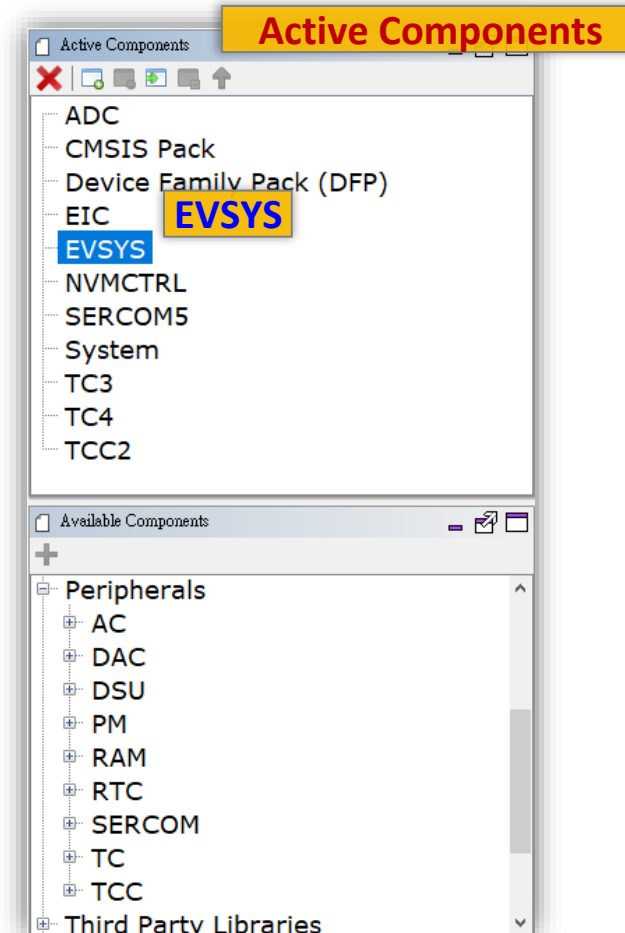
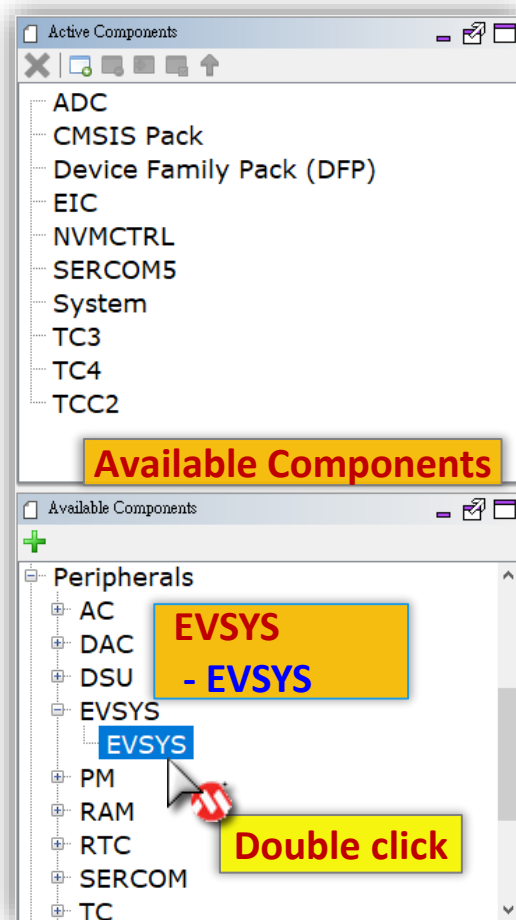


Event System Connection Demo



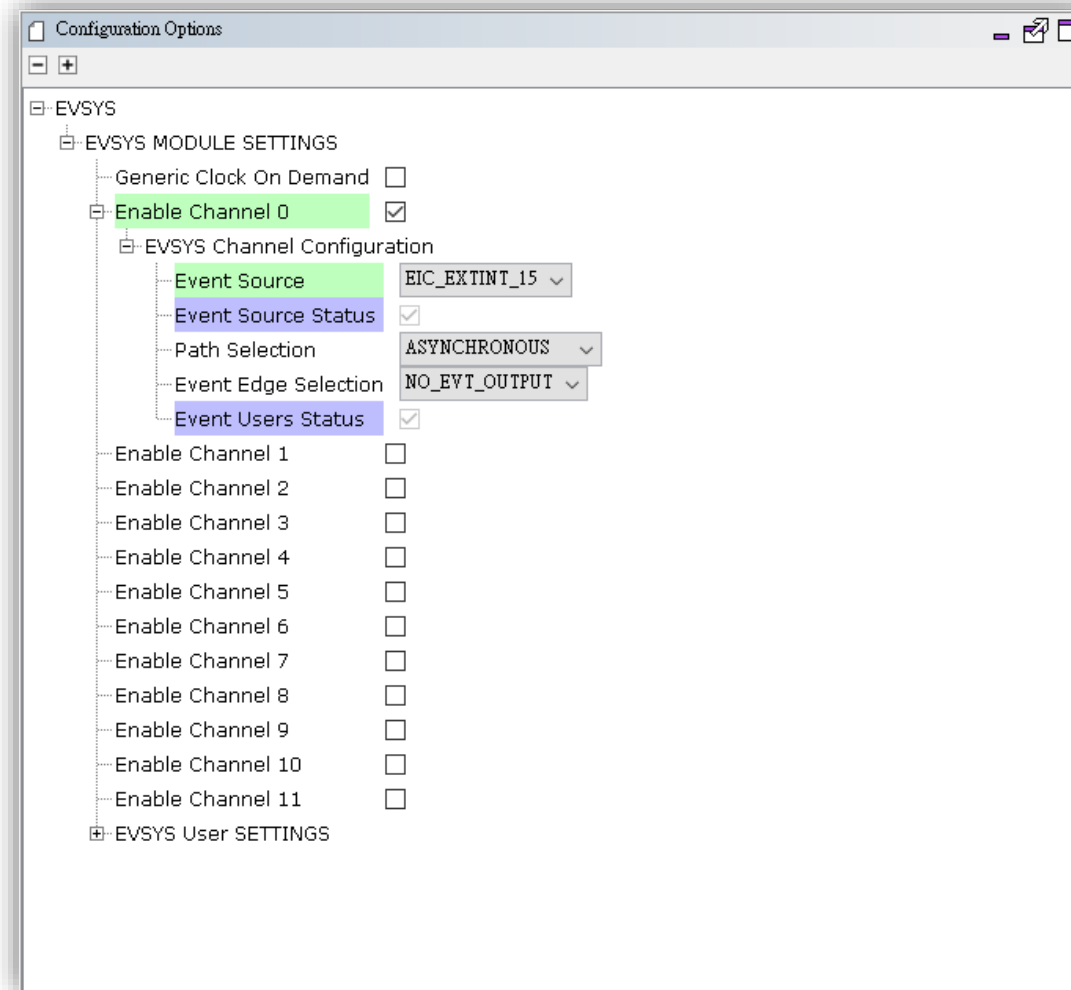
Add EVSYS Function using MHC

- **EVSYS** already be active at new project create by “32-bit MPLAB Harmony 3 Project”.
- Find **EVSYS** component in Available Components window.
- Double click **EVSYS** to add to Active Components window.



Event System

Configuration Options Example



Event System Manager

Configuration Options Example

EVENT0 Easy View

Event System Manager

Channel Configuration

Channel Number	Event Generator	Event Status	User Ready	Remove Channel
Channel 0	EIC_EXTINT_15			Remove

Add Channel

User Configuration

USER	Channel Number	Remove User
TC5_EVU	Channel 0	Remove

Add User

Channel 0 Settings

Path Selection: ASYNCHRONOUS

EVSY Functions & Code Example

Interface Routines & Example

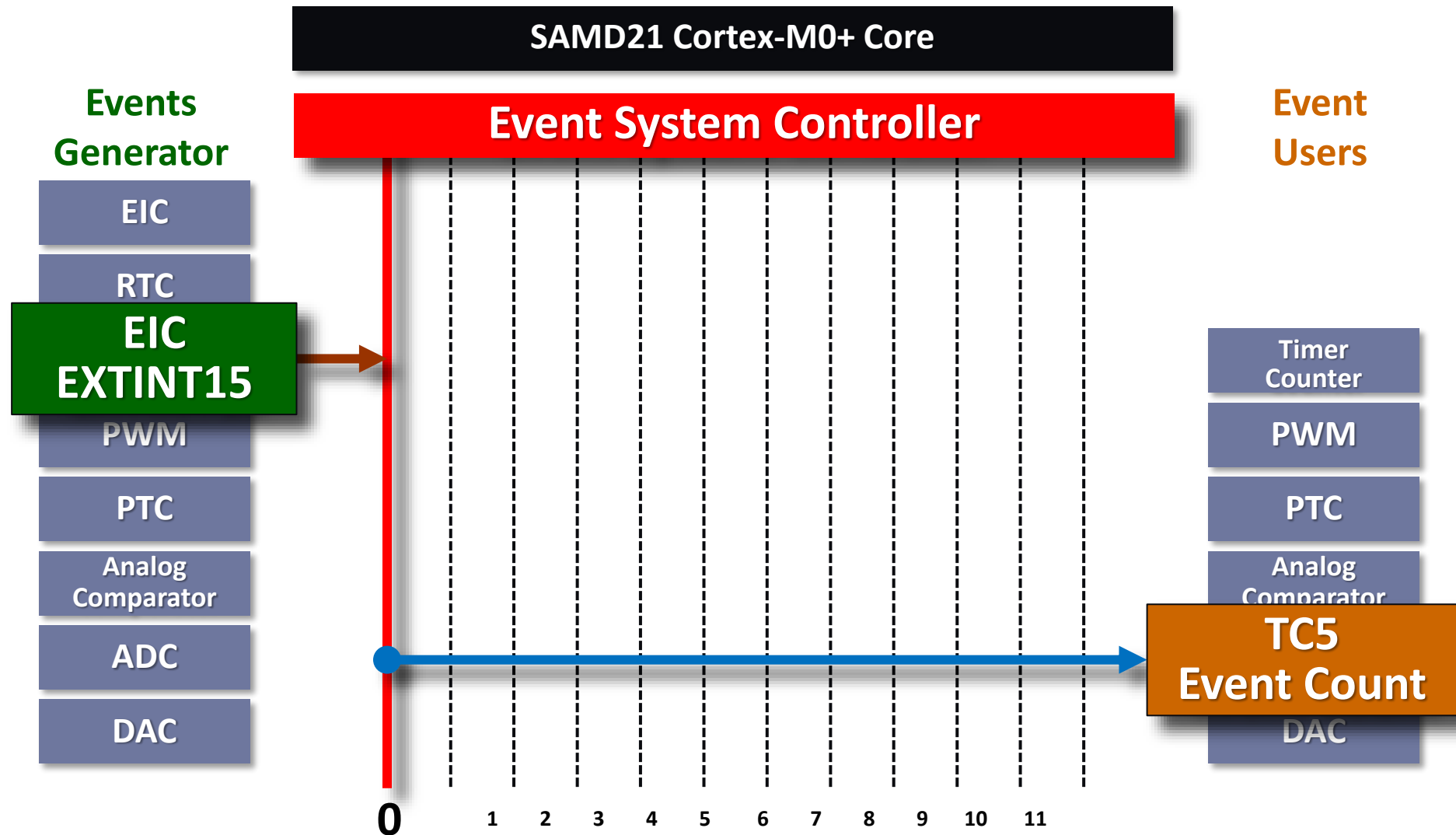
```
void EVSYS_Initialize ( void );  
void EVSYS_GeneratorEnable( EVSYS_CHANNEL channel , uint8_t generator );  
void EVSYS_GeneratorDisable(EVSYS_CHANNEL channel );  
void EVSYS_UserEnable( EVSYS_CHANNEL channel , uint8_t user );  
void EVSYS_UserEnable ( void );
```

```
int main (void)  
{  
    SYS_Initialize ( NULL );  
    ...  
    /* EVSYS Fully automatic !! Not thing to do in user application */  
    while( true )  
    {...}  
}
```

✂ Lab16-2 EXTINT Trigger Timer Counter

- Please continue from [SAM2001ADV Lab16-1 \(Button as External Interrupt\)](#)
- Try to add **EVSYS** module to project.
- Select **EXTINT15 (BT2, PA15)** become to event generator then select **TC5** become to the event user.
- Set the **TC5**'s mode to event counter, use for count the times of button click. Let **LED3(PA17)** toggle every 5 times of button click.
 - EVSYS Channel : 0
 - EVSYS Generator : EIC, EXTINT15
 - EVSYS User : TC5
 - TC5 Operation Mode : Count (Event)
 - TC5 Period : 4 (0 ~ 4, 5 count totally)
- **Let's go !**

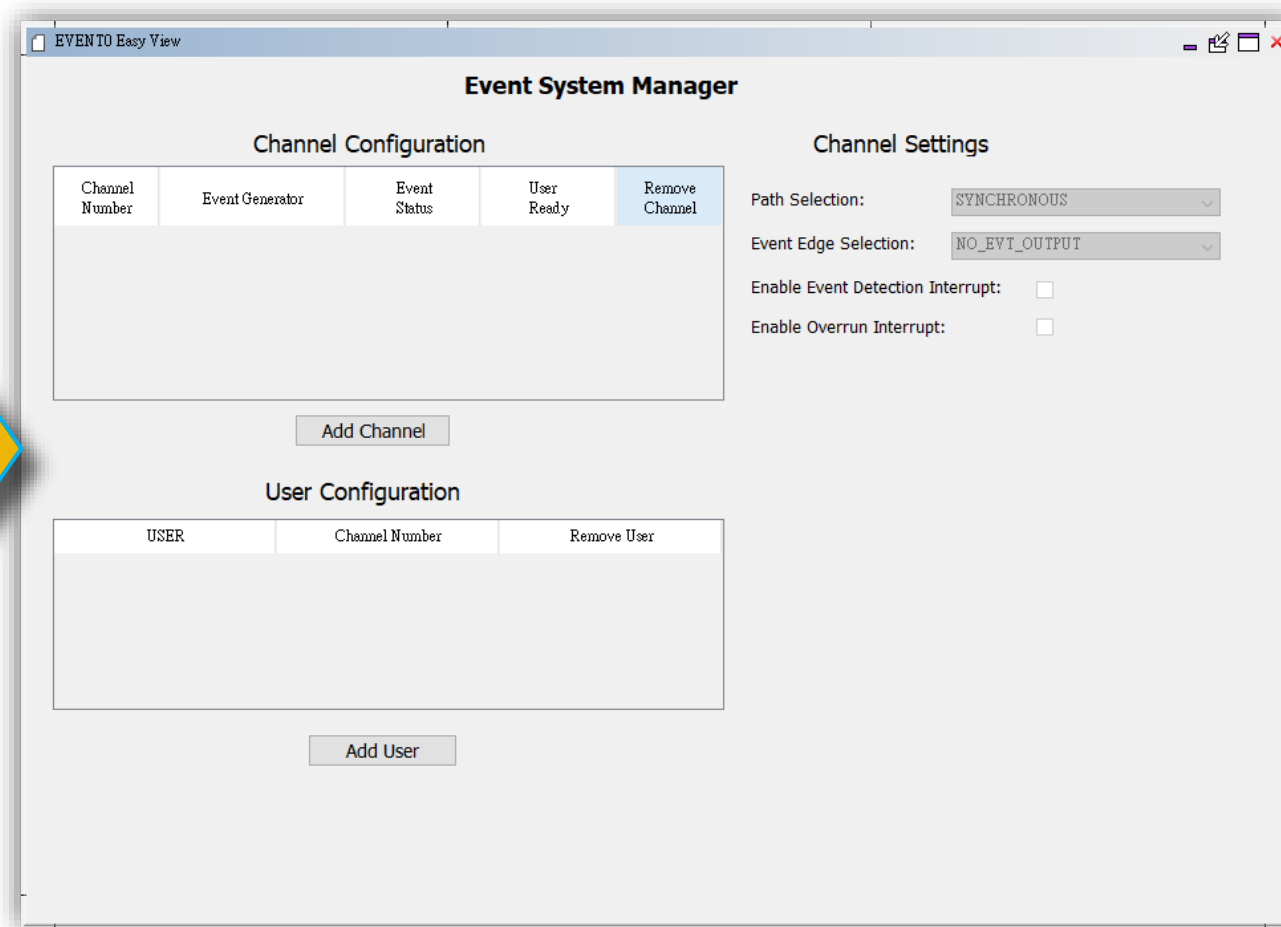
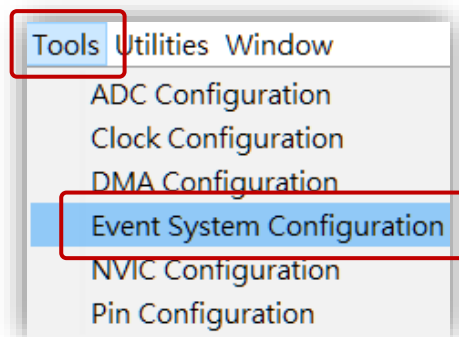
⚙️ Lab16-2 EXTINT Trigger Timer Counter



⚙️ Lab16-2 EXTINT Trigger Timer Counter

Step 1

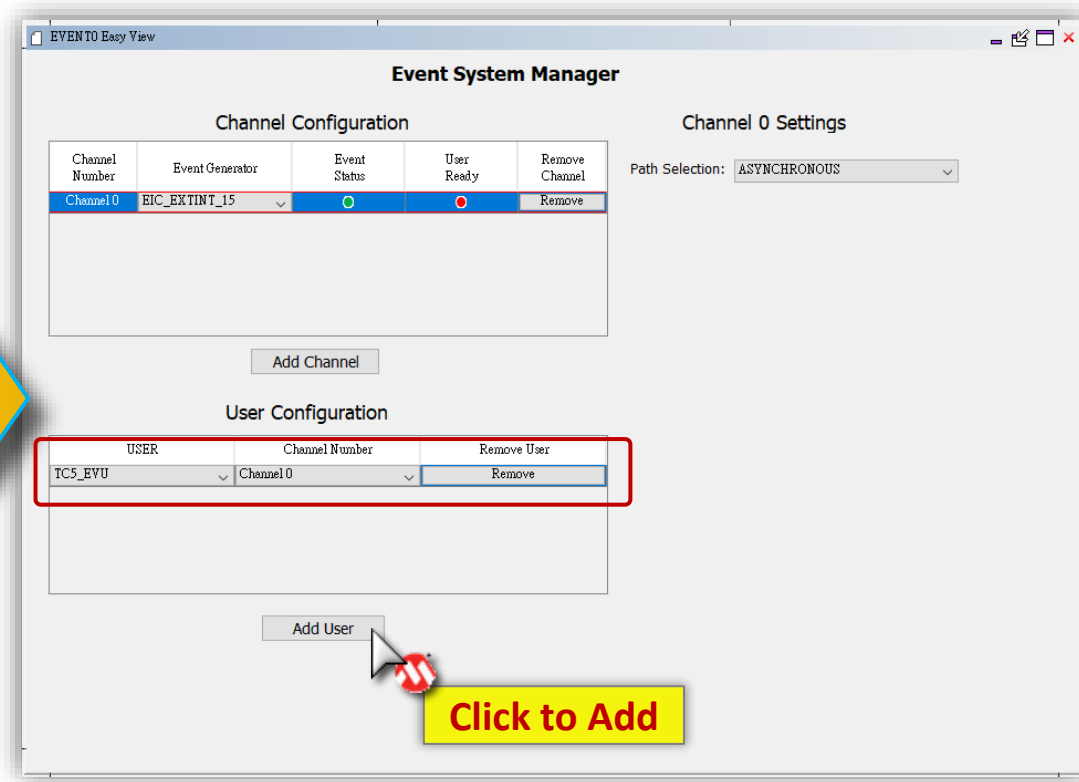
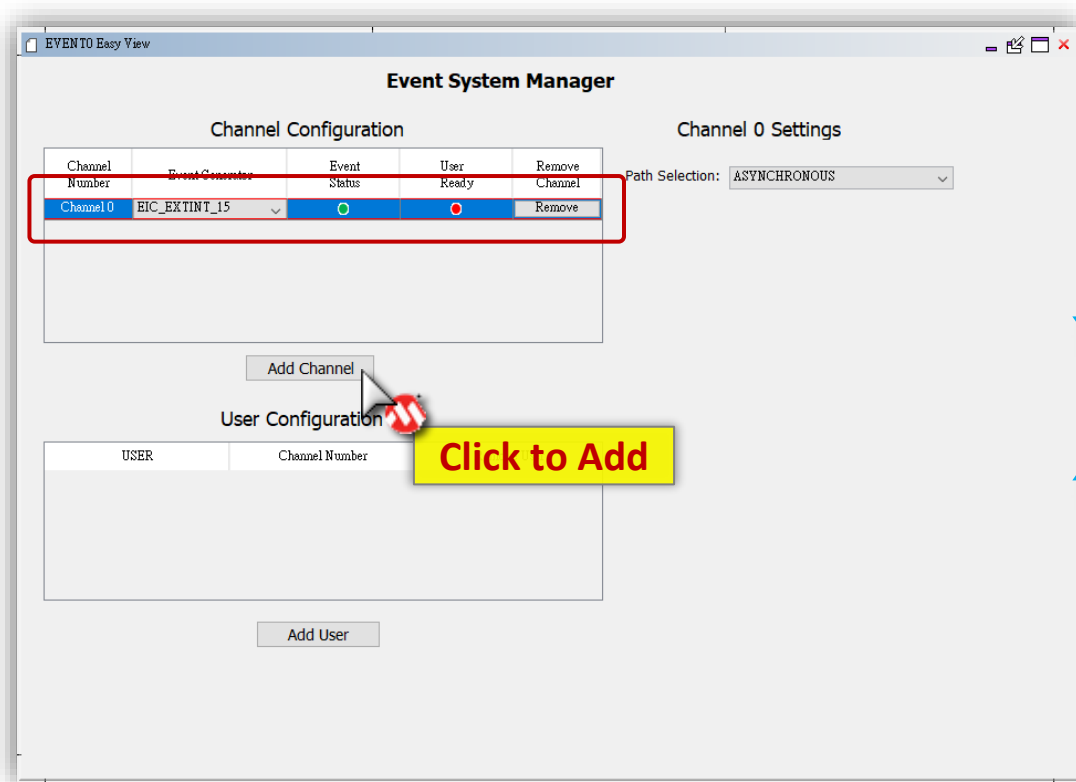
- Configure **EVSYS** in Event System Manager.



⚙️ Lab16-2 EXTINT Trigger Timer Counter

Step 2

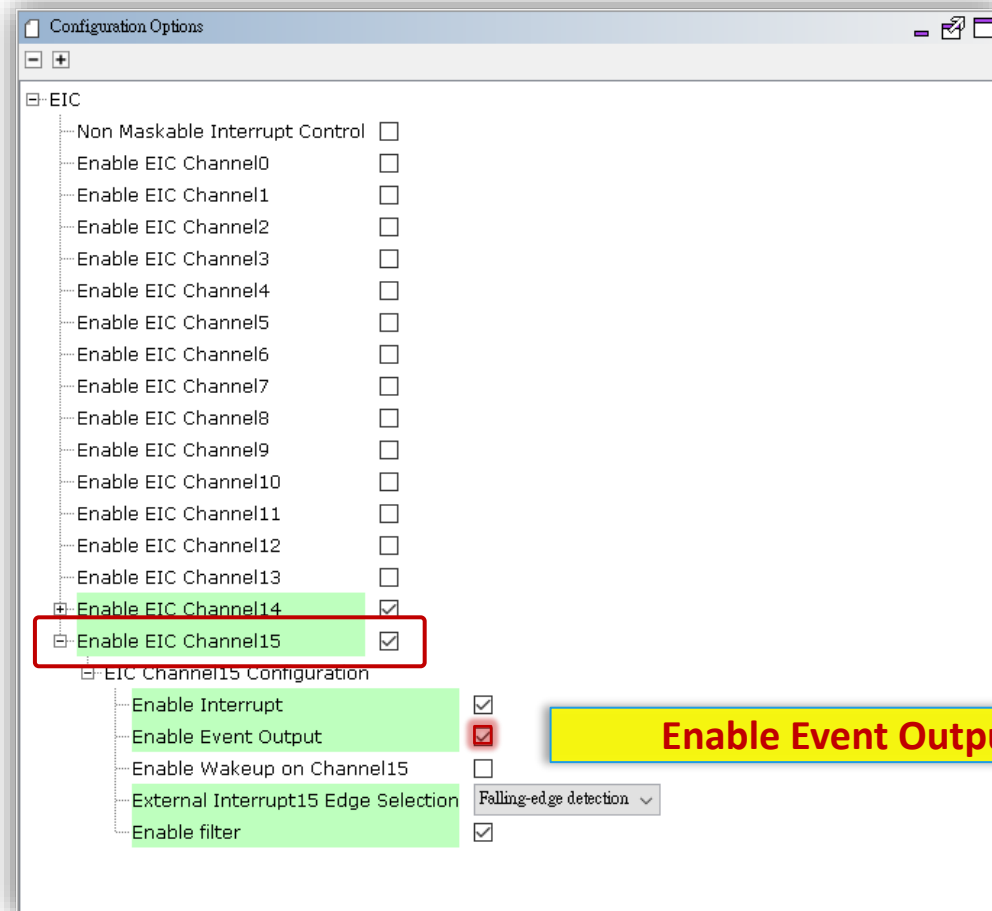
- add **Event Generator** : EIC_EXTINT_15 , **Event User** : TC5_EVU



⚙️ Lab16-2 EXTINT Trigger Timer Counter

Step 3

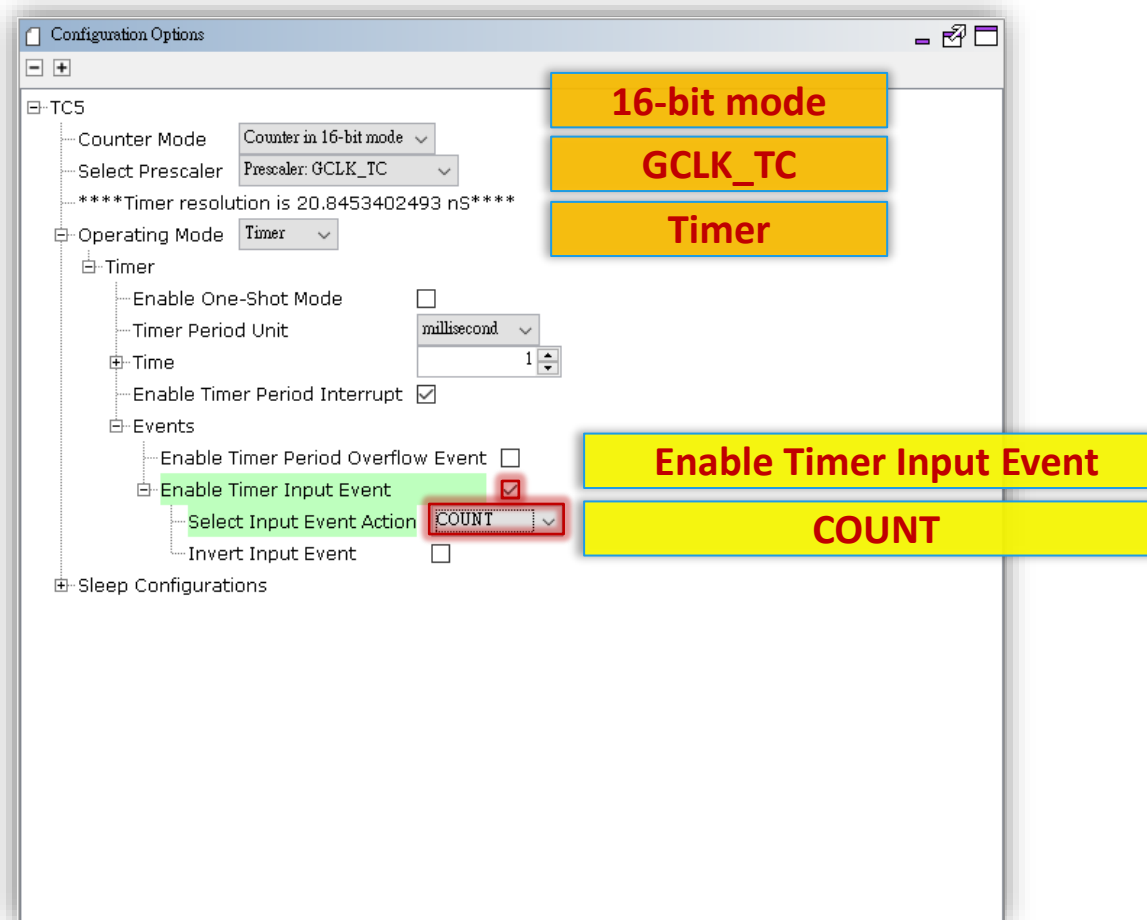
- Enable Event be generated at EIC (EXTINT15).



⚙️ Lab16-2 EXTINT Trigger Timer Counter

Step 4

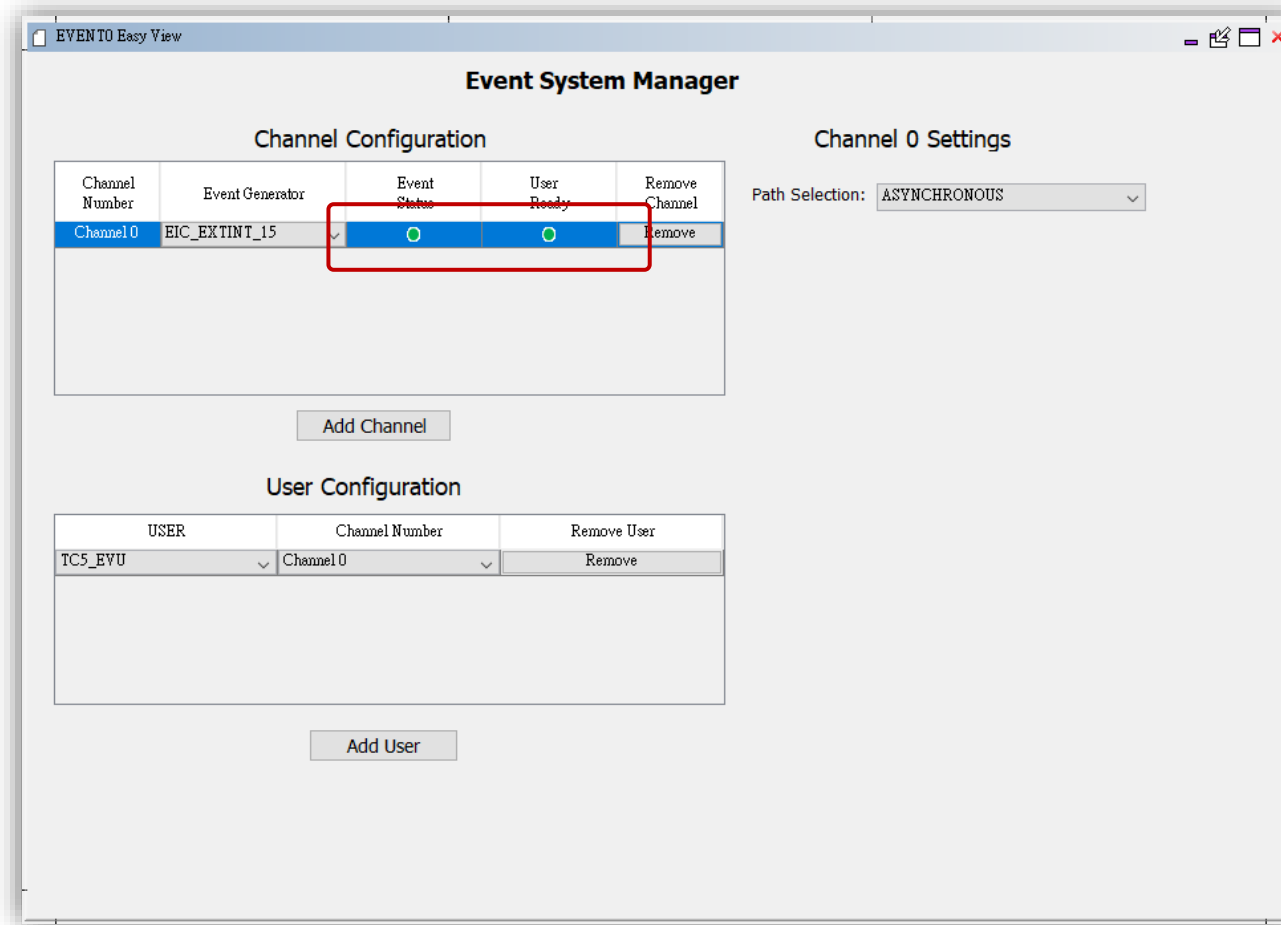
- Add TC5 and set to Event Counter Mode



⚙️ Lab16-2 EXTINT Trigger Timer Counter

Step 5

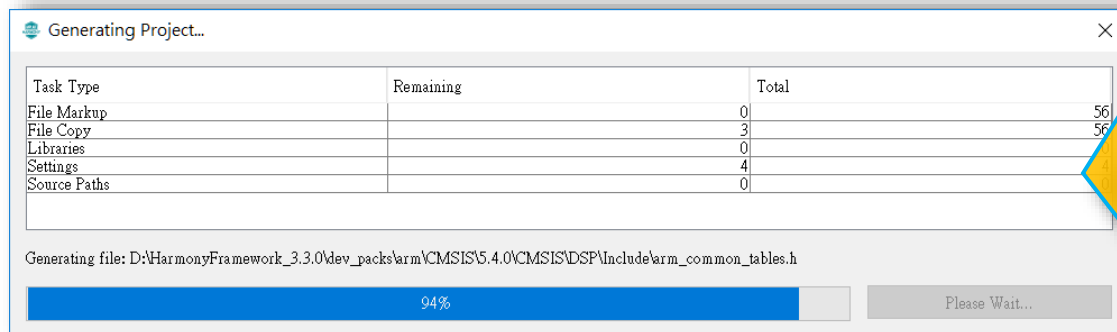
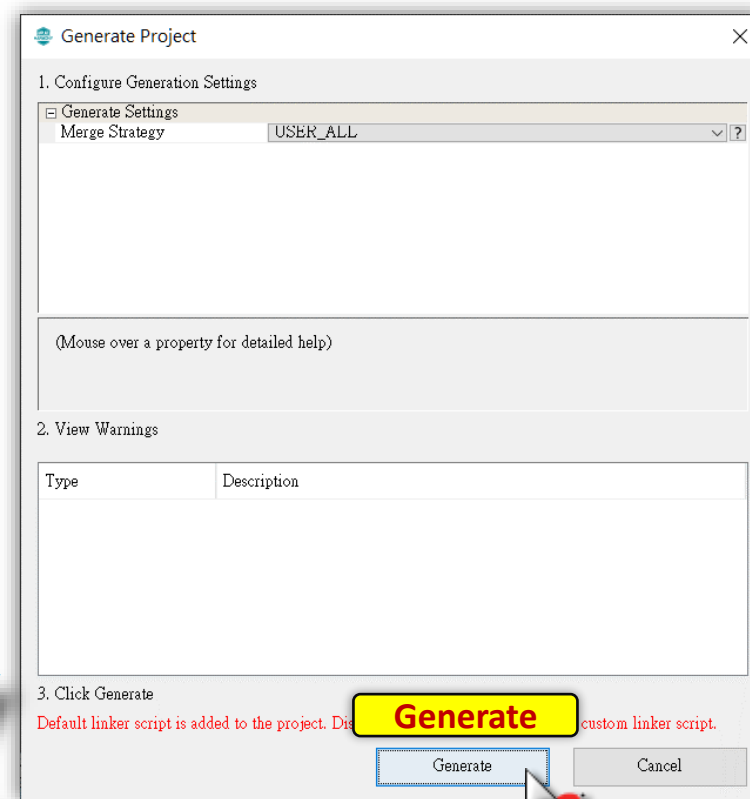
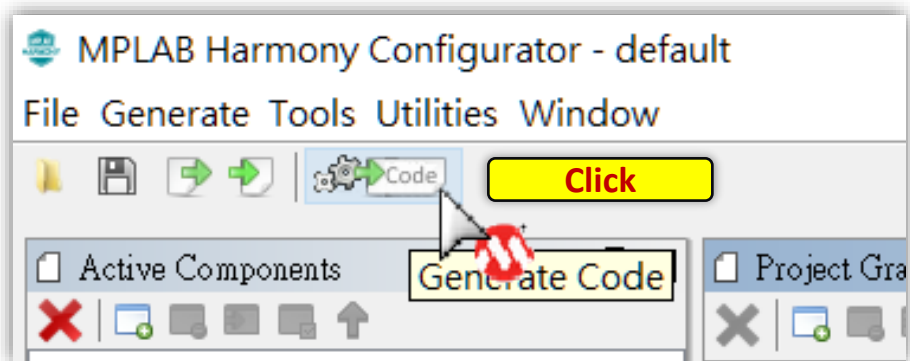
- Confirm Event Status and User Ready is **Green** light.



⚙️ Lab16-2 EXTINT Trigger Timer Counter

Step 6

- Click  to Generate Code.



⚙️ Lab16-2 EXTINT Trigger Timer Counter

Step 7

```
...
// TODO 16-2.01
void TC5_TimerExpired(TC_TIMER_STATUS status, uintptr_t context)
{
    if (status & TC_INTFLAG_OVF_Msk)
    {
        LED3_Toggle();
    }
}

int main (void)
{
    SYS_Initialize(NULL);
    ...
    // TODO 16-2.02
    TC5_TimerCallbackRegister(TC5_TimerExpired, (uintptr_t) NULL);
    TC5_Timer16bitPeriodSet(4);
    TC5_TimerStart();

    while ( true )
    {
        SYS_Tasks();
        ...
    }
    return ( EXIT_FAILURE);
}
```

Lab16-2 EXTINT Trigger Timer Counter

Step 8

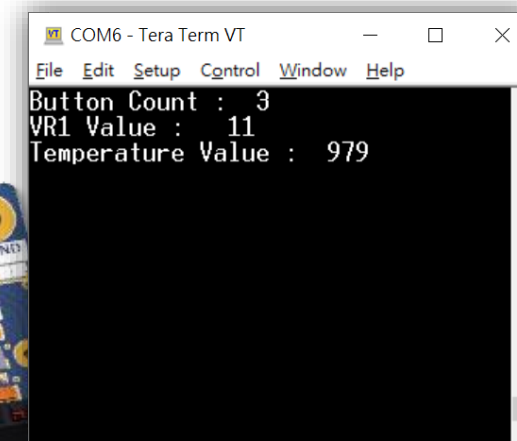
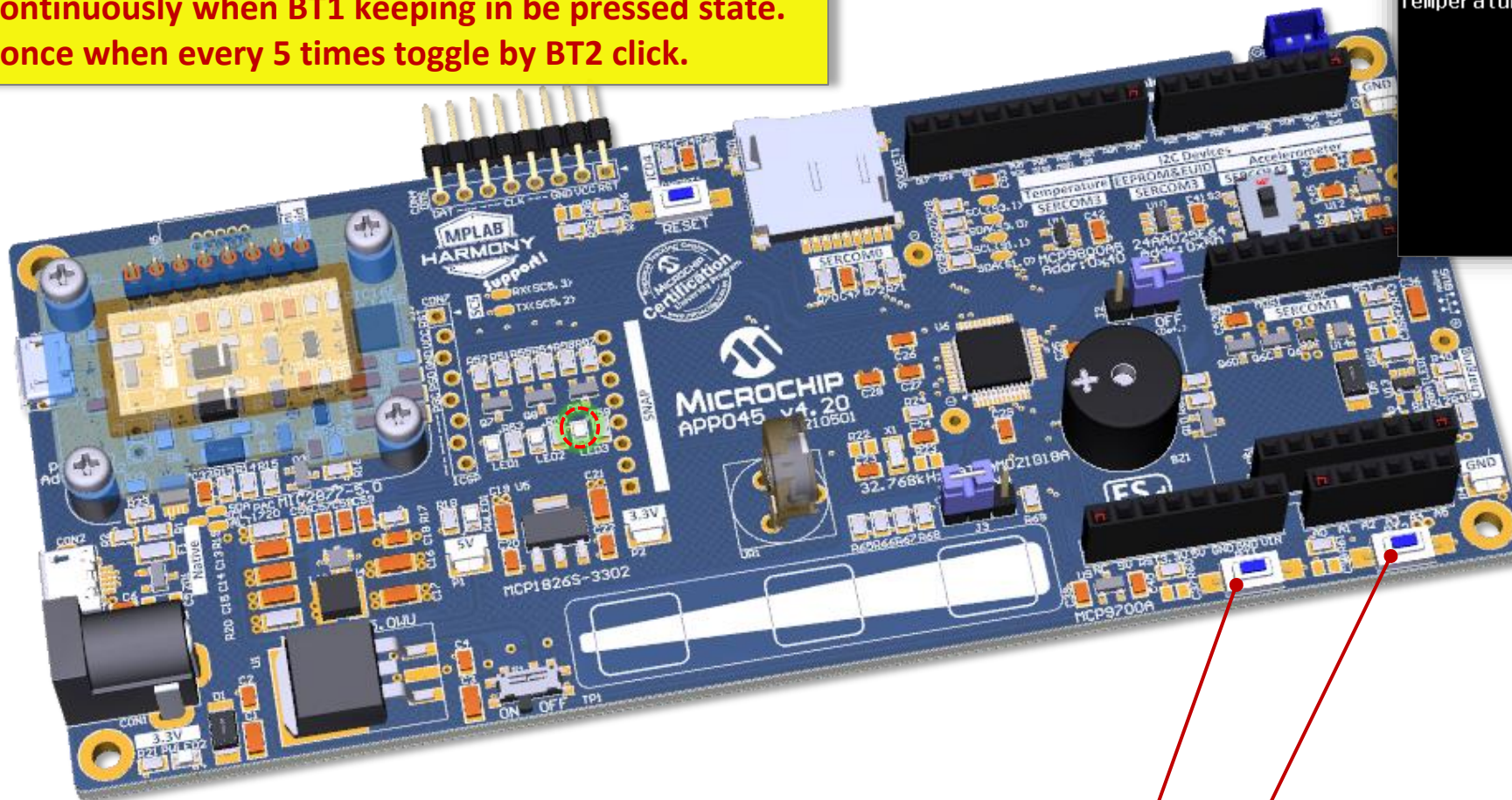
```
...
int main (void)
{
    SYS_Initialize(NULL);
    ...
    // TODO 16-2.03
    EIC_CallbackRegister(EIC_PIN_14, LED_Control, (uintptr_t) NULL);
    //EIC_CallbackRegister(EIC_PIN_15, LED_Control, (uintptr_t) NULL);

    while ( true )
    {
        SYS_Tasks();
        ...
        if (TC3_HasExpired)
        {
            ...
            // TODO 16-2.04
            //myprintf("Hello World!");
            myprintf("Button Count : %2d", TC5_Timer16bitCounterGet());
        }
    }
    return ( EXIT_FAILURE);
}
```


⚙️ Lab16-2 EXTINT Trigger Timer Counter

Result

LED3 blink Continuously when BT1 keeping in be pressed state.
LED3 toggle once when every 5 times toggle by BT2 click.



Push BT1/BT2 to Toggle LED3

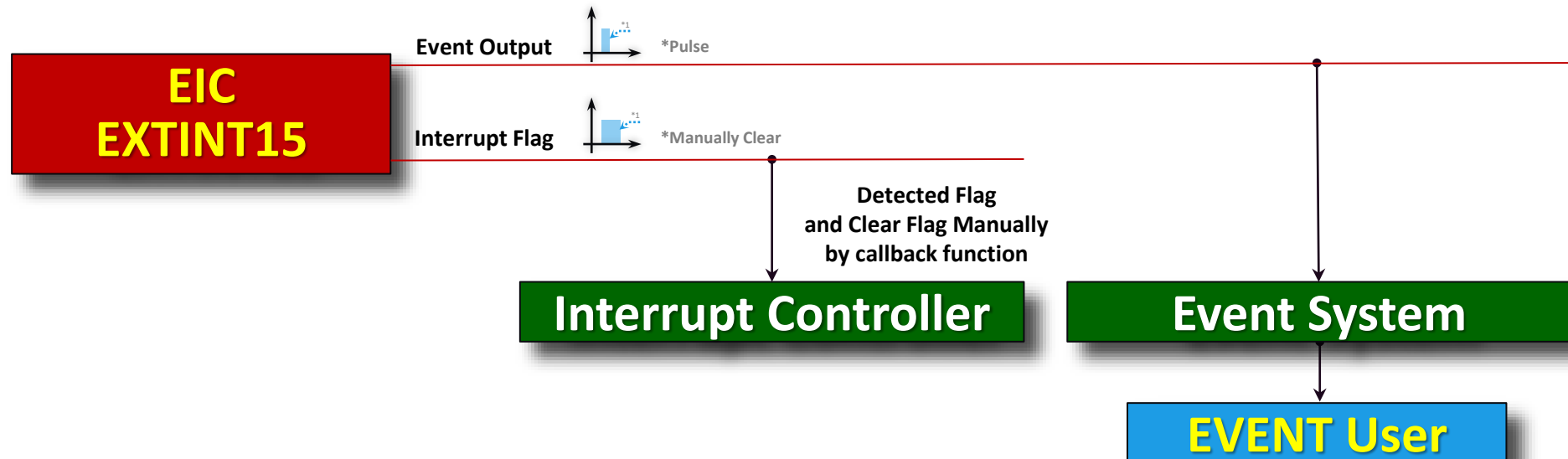
⚙️ Lab16-2 EXTINT Trigger Timer Counter

- **Discuss**

- Could set EIC callback of EIC_PIN15 and enable EIC_PIN15 event output at the same time ?

```
// TODO 16-2.03
EIC_CallbackRegister(EIC_PIN_14, LED_Control, (uintptr_t) NULL);
//EIC_CallbackRegister(EIC_PIN_15, LED_Control, (uintptr_t) NULL);
```

- The interrupt request of EIC_PIN15 and EIC_PIN15 event output could be enable at the same time.



✂ Lab16-3 EXTINT Callback and Trigger Timer Counter

- This is a demo lab, demo that the interrupt request of EIC_PIN15 and EIC_PIN15 event output be enable at the same time. Uncomment at Lab 16-3 for EIC_PIN_15 Callback.
- Just open, build and program the firmware to MCU, then try it.

```
// TODO 16-3.03
EIC_CallbackRegister(EIC_PIN_14, LED_Control, (uintptr_t) NULL);
EIC_CallbackRegister(EIC_PIN_15, LED_Control, (uintptr_t) NULL);
```

• **Let's go !**

**Reference*

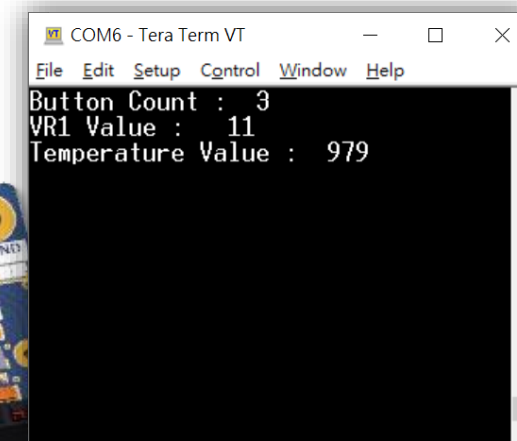
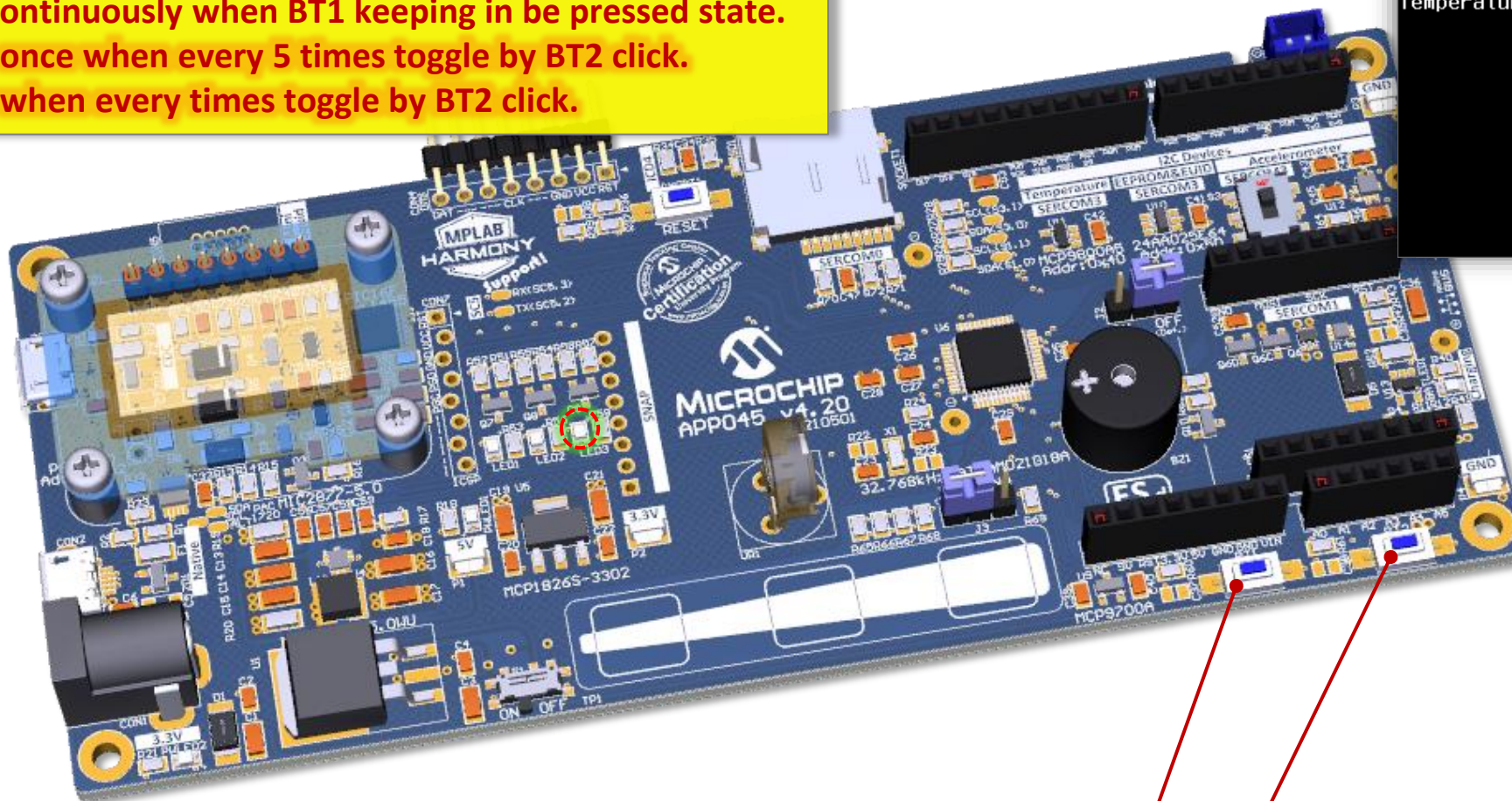
```
void LED_Control(uintptr_t context)
{
    static uint32_t i = 0;
    LED3_Toggle();
    for (i = 0; i < 1000000; i++);
}
```

```
void TC5_TimerExpired(TC_TIMER_STATUS status, uintptr_t context)
{
    if (status & TC_INTFLAG_OVF_Msk)
    {
        LED3_Toggle();
    }
}
```


⚙️ Lab16-3 EXTINT Callback and Trigger Timer Counter

Result

LED3 blink Continuously when BT1 keeping in be pressed state.
LED3 toggle once when every 5 times toggle by BT2 click.
LED3 toggle when every times toggle by BT2 click.



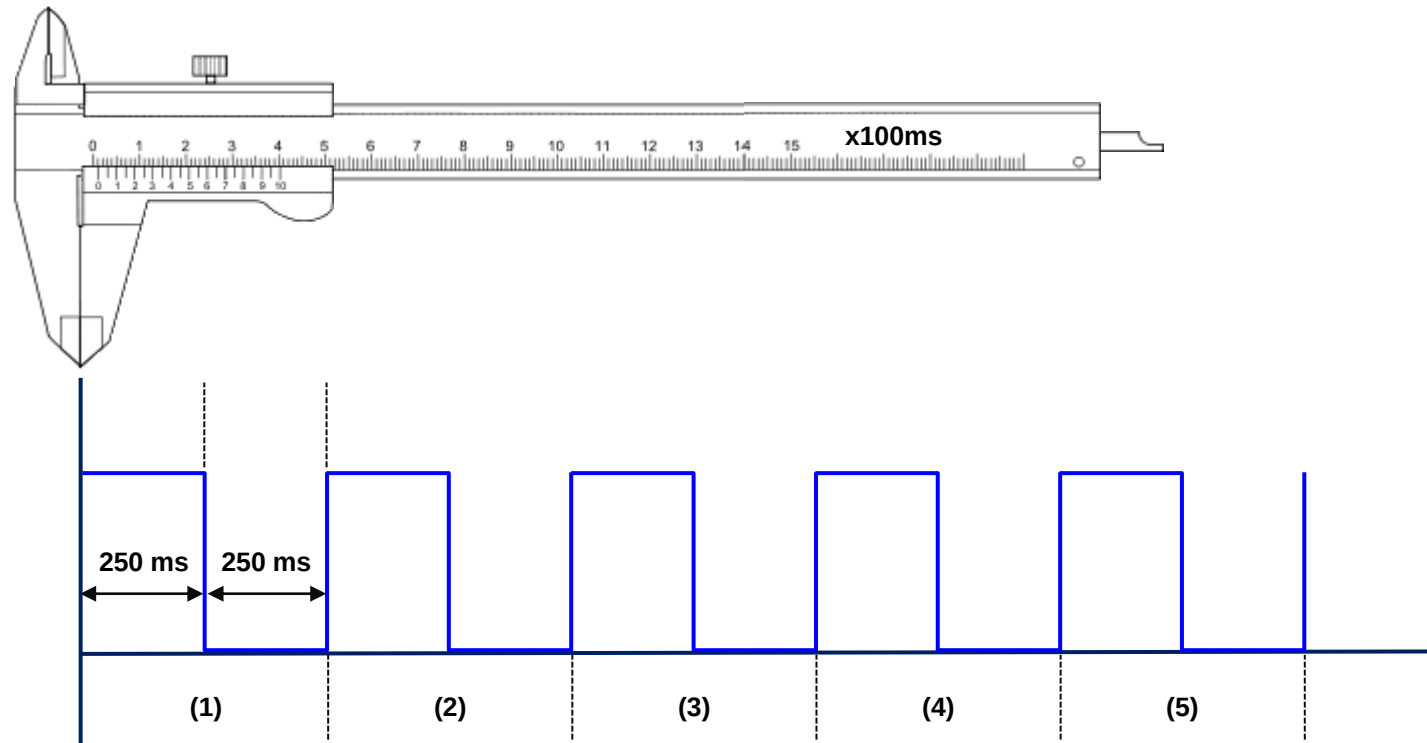
Push BT1/BT2 to Toggle LED3

TC Waveform Capture

Integrate TC, EIC and Event System

What's Waveform Capture?

- **Waveform Capture** is used to capture a timer value from an independent timer base upon an event on the input pin.
- The Capture features are useful in applications requiring frequency (time period) and pulse measurement.



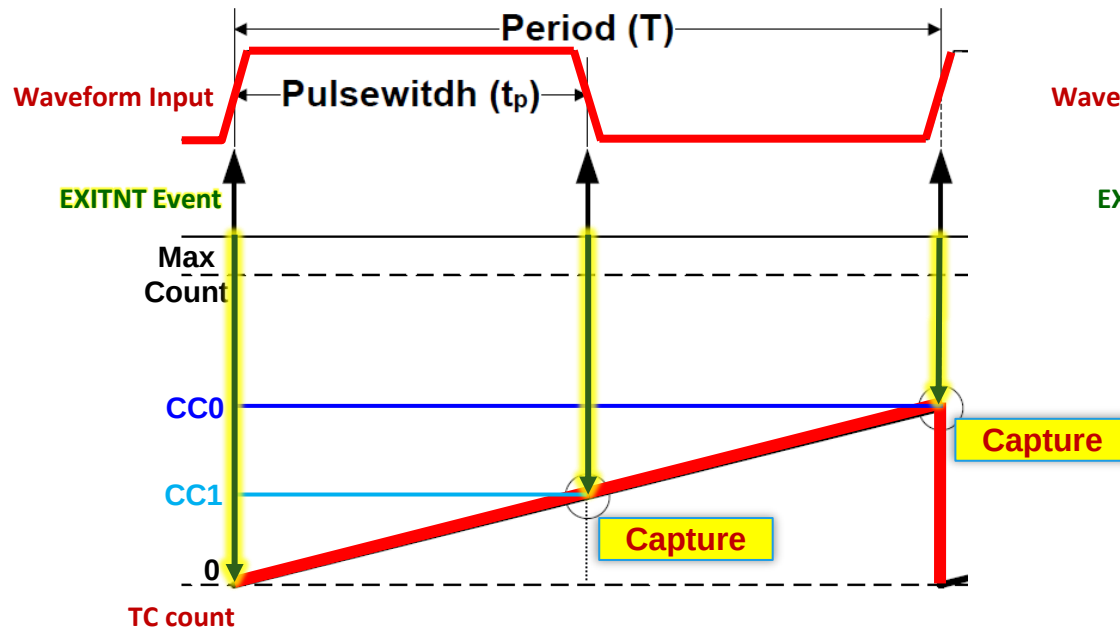
TC Capture with Event System

- The TC compare/capture channels can be used as input capture channels to capture events from the Event System and give them a timestamp.
- The TC can perform two input captures action on the rising and falling edge, counter will restart on one of the edges depend. This enables the TC to measure the pulse width and period and to characterize the frequency and duty cycle of an input signal.

PPW (Period and Pulse-width) :

Period (T) will be captured into **CC0**,

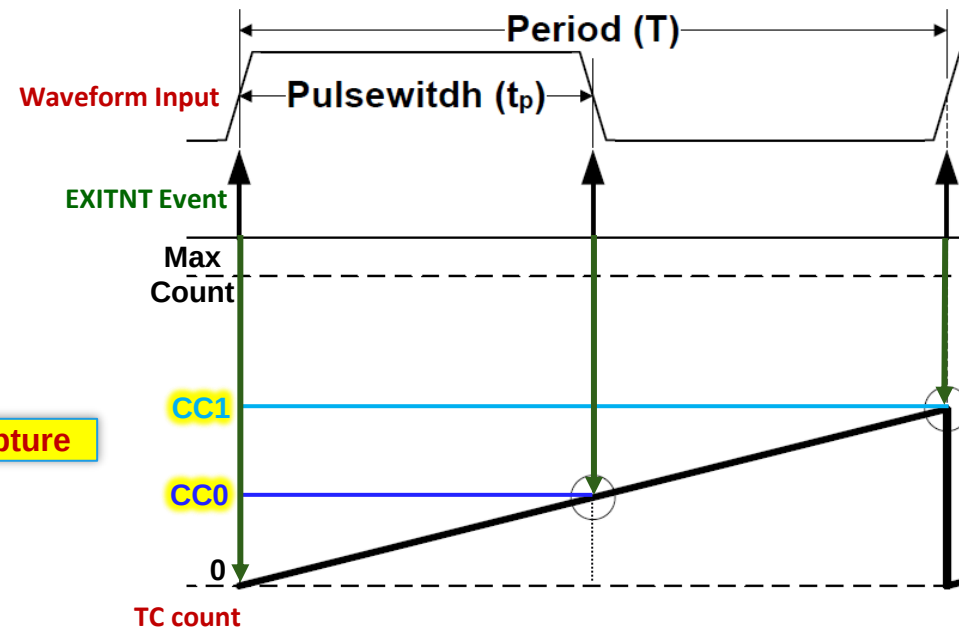
Pulsewidth (t_p) will be into **CC1**



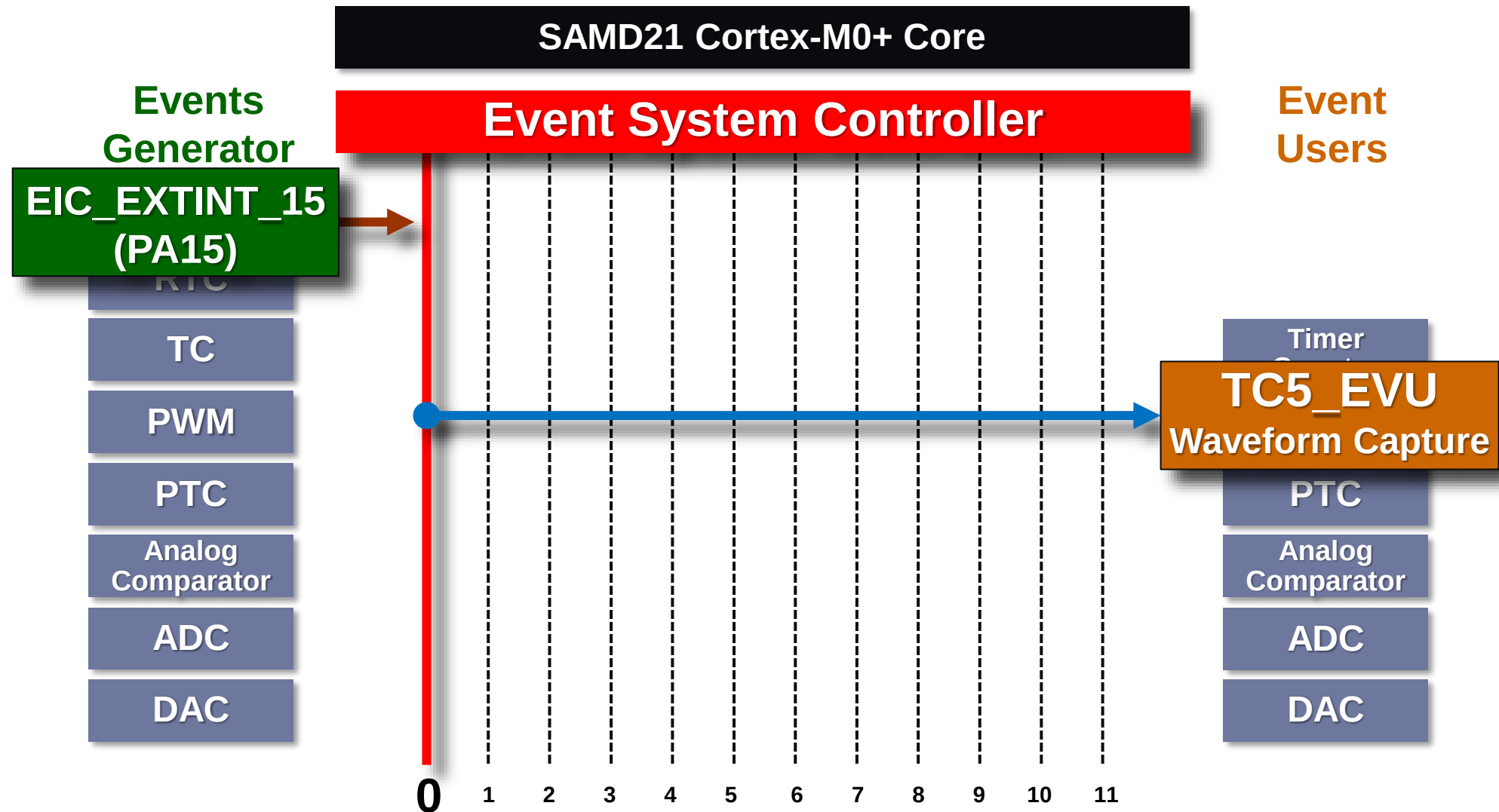
PWP (Pulse-width, Period) :

Period (T) will be captured into **CC1**,

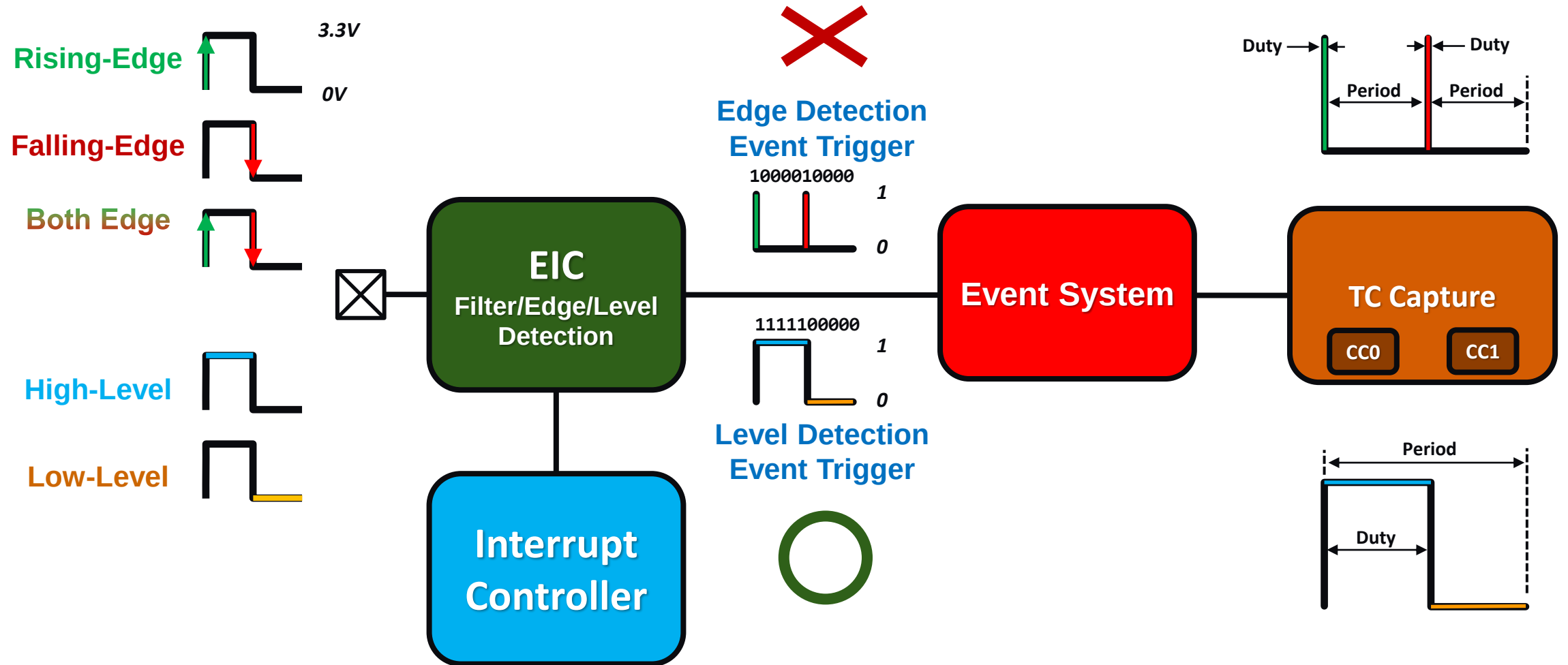
Pulsewidth (t_p) will be into **CC0**.



Routing EIC to TC Capture through Event System



Routing EIC to TC Capture through Event System



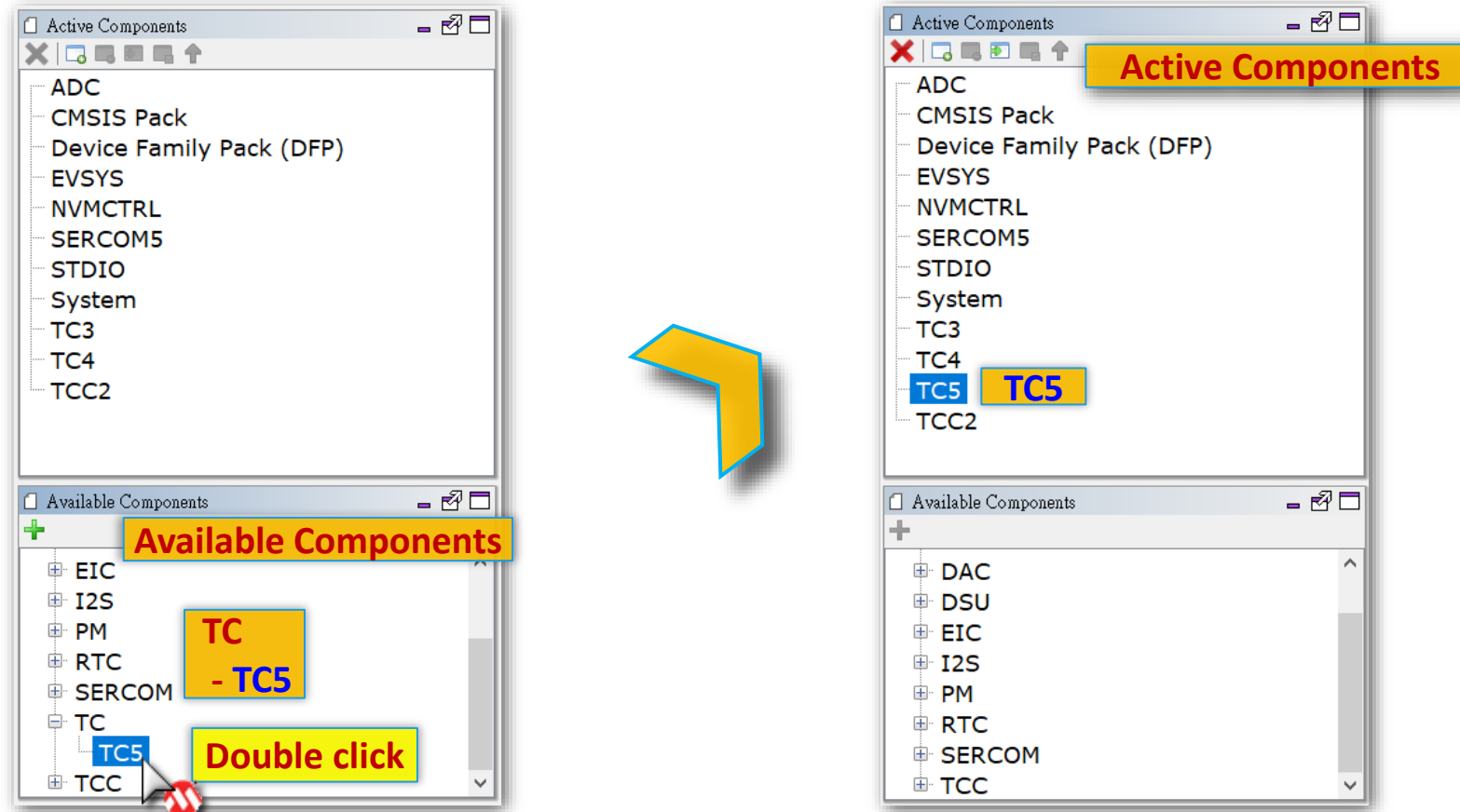
Lab16-4 TC Waveform Capture

- Please continue from **SAM2001 Lab16 (TCC NPWM Auto DutyCtrl)**
- Add new **TC5** module and configure to **Capture** mode.
- Add **EIC** module and enable **EIC_EXTINT15(PA15)** for Event output.
- Assign **EIC_EXTINT15(PA15)** as waveform capture input pin and configure it as the **Event System Generator**.
- Assign **TC5** module as **Event System User** to capture waveform from **EIC_EXTINT15(PA15)**.
- Output measured waveform **Period** and **Duty** to UART.
- **Let's go!**

Lab16-4 TC Waveform Capture

Step 1

- Find **TC5** component in Available Components window.
- Double click **TC5** to add to Active Components window.



Lab16-4 TC Waveform Capture

Step 2

- Config **TC5** as Capture Mode and enable Interrupt.

The screenshot shows the 'Configuration Options' dialog for TC5. The settings are as follows:

- Counter Mode:** Counter in 16-bit mode (16-bit mode)
- Select Prescaler:** Prescaler: GCLK_TC/1024 (GCLK_TC/1024)
- Operating Mode:** Capture (Capture)
- Capture:**
 - Enable Capture Channel 0:** ☒ (Enable Capture Channel 0)
 - Capture Trigger Source:** Input Event (Input Event)
 - Capture Trigger Edge:** Rising Edge (Rising Edge)
 - Capture Trigger Action:** Period captured in CC0, pulse width in CC1 (Period captured in CC0, pulse width in CC1 (PPW))
 - Enable Capture 0 Interrupt:** ☒ (Enable Capture 0 Interrupt)
 - Enable Capture 0 Event Out:** ☐
 - Enable Capture Channel 1:** ☒
 - Enable Capture Error Interrupt:** ☐
 - Enable Capture Overflow Interrupt:** ☐
- Sleep Configurations:** (collapsed)

****Timer resolution is 21345.6284153 nS****

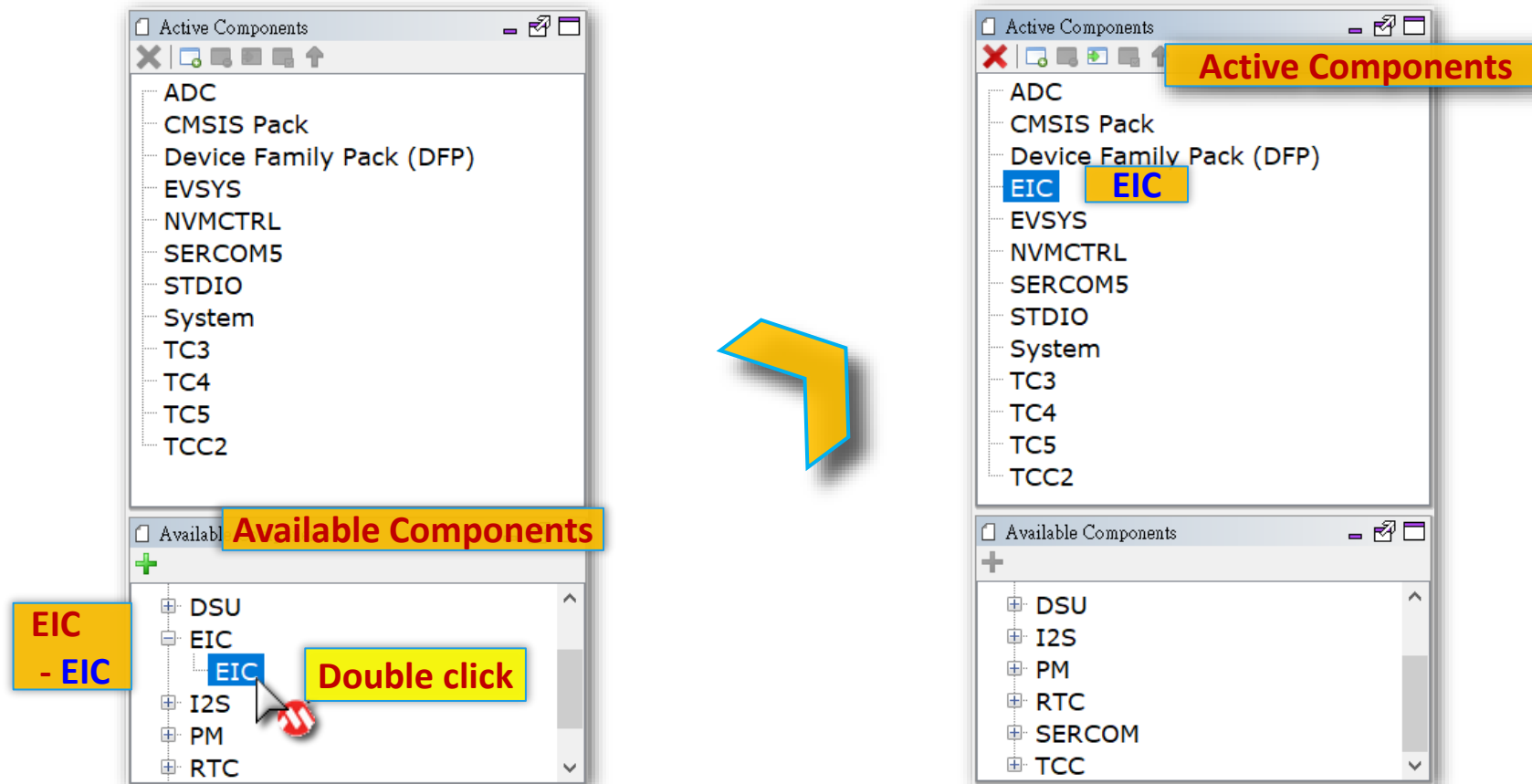
Calculation Box:

- Capture Resolution is :** $48\text{M}/1024 = 46.875\text{ KHz}$
- Max Capture Count is :** 0xFFFF (16bits)
- Max Capture Period is :** $0\text{xFFFF} / 46.875\text{KHz} = 1398\text{ ms}$

Lab16-4 TC Waveform Capture

Step 3

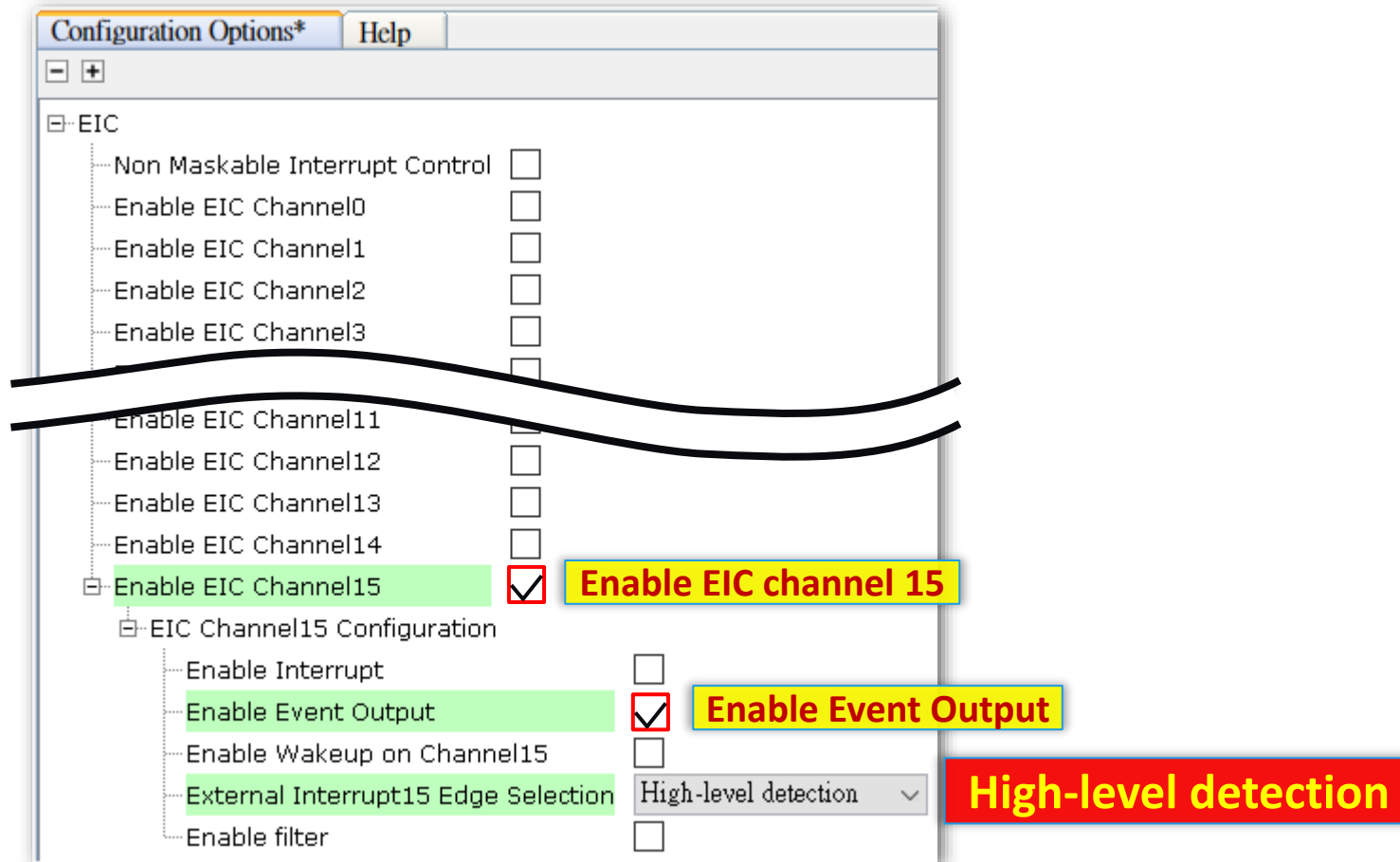
- Find **EIC** component in Available Components window.
- Double click **EIC** to add to Active Components window.



Lab16-4 TC Waveform Capture

Step 4

- Enable **EIC channel 15** in configuration options, it's corresponding to **PA15**.
- Enable **Event Output** and Edge selection to **High-level detection**.



Lab16-4 TC Waveform Capture

Step 5

- Disable **GCLK_IO1** on **PA15** and Enable it for **EIC_EXTINT15** pin function.

The diagram illustrates the configuration of pin PA15. It consists of two screenshots of the Pin Table tool, connected by a large yellow arrow pointing from left to right.

Left Screenshot: The Pin Table tool shows the package TQFP48. The pin PA15 is selected. The function GCLK_IO1 is currently assigned to PA15. A red box highlights GCLK_IO1 in the function list, and a red box highlights the PA15 pin header. A mouse cursor is clicking on the GCLK_IO1 entry, with a yellow callout box saying "Click Disable".

Right Screenshot: The Pin Table tool shows the same package TQFP48. The pin PA15 is selected. The function EIC_EXTINT15 is now assigned to PA15. A red box highlights EIC_EXTINT15 in the function list, and a red box highlights the PA15 pin header. A mouse cursor is clicking on the EIC_EXTINT15 entry, with a yellow callout box saying "Click Enable".

Module	Function	PB10	PB11	PA12	PA13	GCLK_I	GCLK_I	PA16
		19	20	21	22	23	24	25
EIC	EIC_EXTINT13							
	EIC_EXTINT14							
	EIC_EXTINT15							
	EIC_EXTINT2							
	EIC_EXTINT3							
	EIC_EXTINT4							
	EIC_EXTINT5							
	EIC_EXTINT6							
	EIC_EXTINT7							
	EIC_EXTINT8							
GCLK	GCLK_IO1							
	GCLK_IO2							

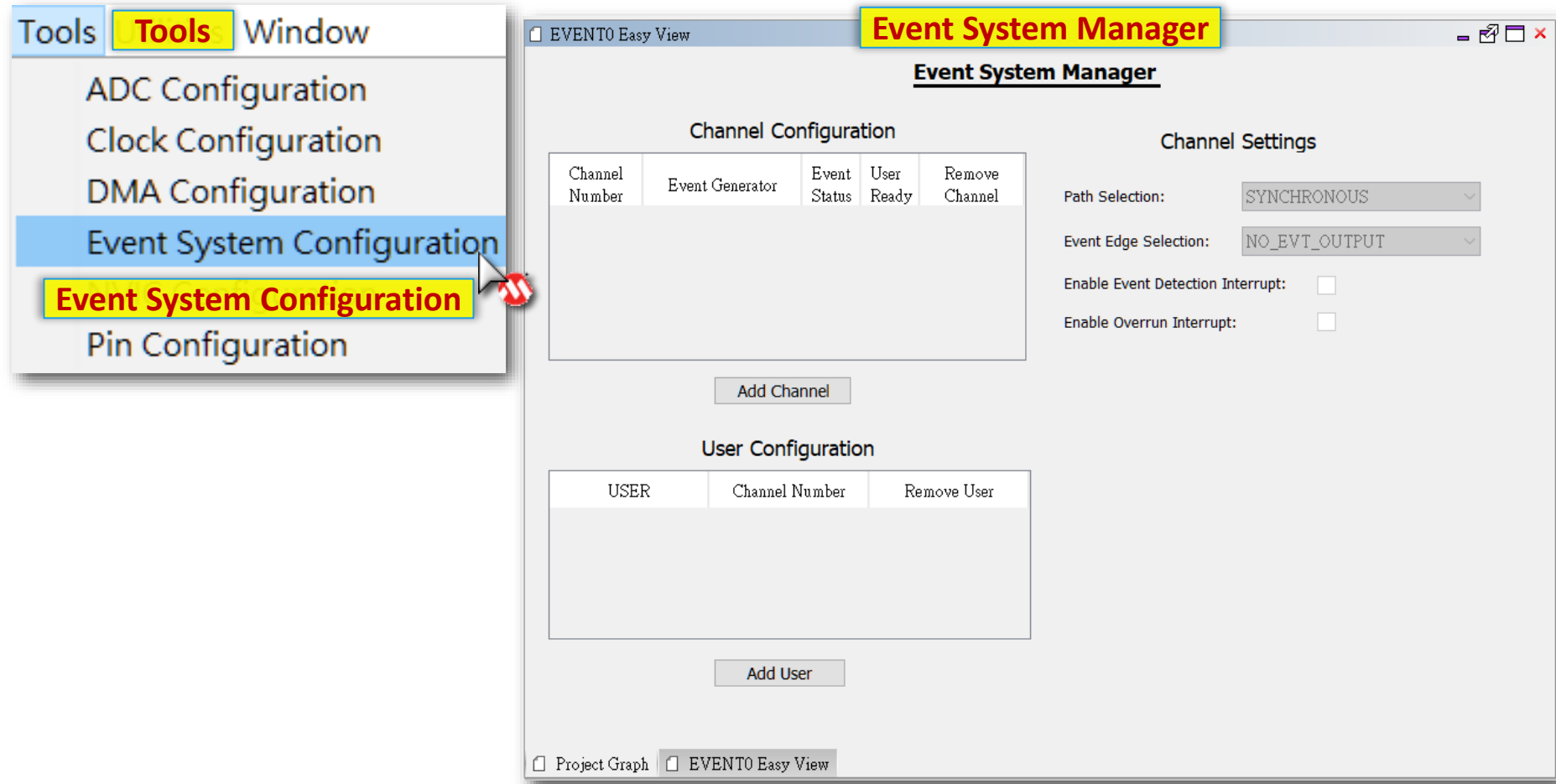
Right Screenshot Table:

Module	Function	PB10	PB11	PA12	PA13	GCLK_I	EIC_EX	PA16
		19	20	21	22	23	24	25
EIC	EIC_EXTINT13							
	EIC_EXTINT14							
	EIC_EXTINT15							
	EIC_EXTINT2							
	EIC_EXTINT3							
	EIC_EXTINT4							
	EIC_EXTINT5							
	EIC_EXTINT6							
	EIC_EXTINT7							
	EIC_EXTINT8							
GCLK	GCLK_IO0							
	GCLK_IO1							
	GCLK_IO2							

Lab16-4 TC Waveform Capture

Step 6

- Configure Event System through **Event System manager**.



Lab16-4 TC Waveform Capture

Step 7

- Add **Channel 0** to assign **EIC_EXTINT_15** as Event Generator.

Channel Configuration

Channel Number	Event Generator	Event Status	User Ready	Remove Channel
Channel 0	EIC_EXTINT_15			Remove

Channel 0

EIC_EXTINT_15

Add Channel

Click

Lab16-4 TC Waveform Capture

Step 8

- Add **User** and assign **TC5_EVU** as **Channel 0** Event **User**.

User Configuration

USER	Channel Number	Remove User
TC5 EVU	Channel 0	Remove

TC5_EVU

Channel 0

Add User

Click

Lab16-4 TC Waveform Capture

Step 9

- Event and User status shows in **green color** means ready.

Channel Configuration

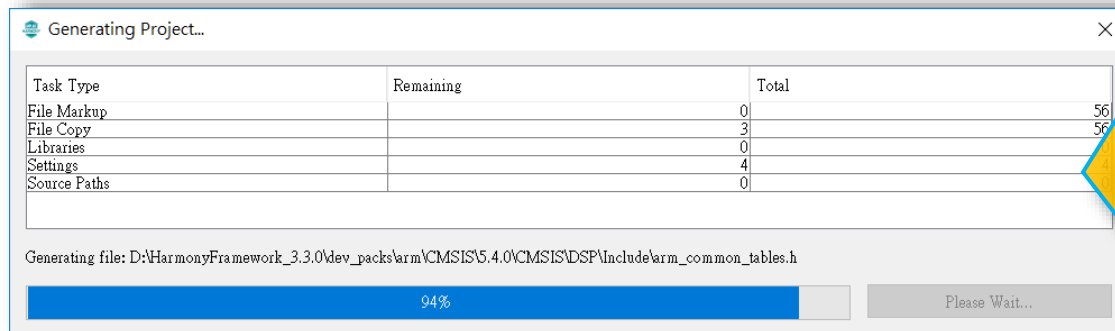
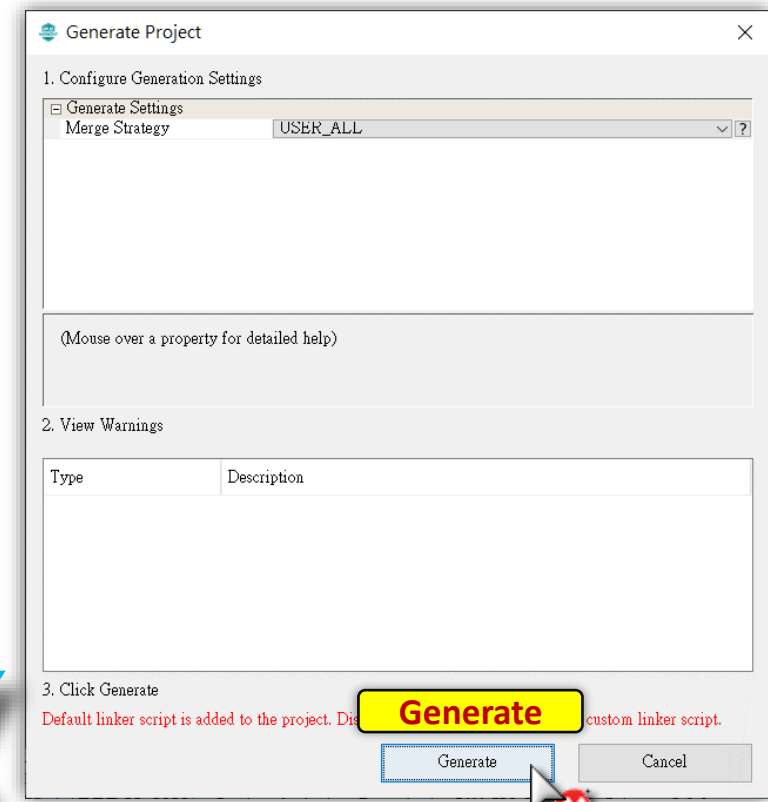
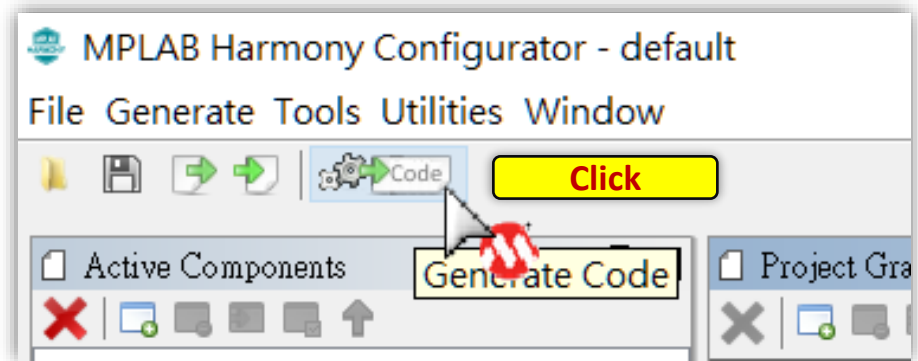
Channel Number	Event Generator	Event Status	User Ready	Remove Channel
Channel 0	EIC_EXTINT_15 ▾	●	●	Remove

Add Channel

Lab16-4 TC Waveform Capture

Step 10

- Click  to Generate Code.



TC Capture

Interface Routines

```
void TC5_CaptureInitialize( void );  
void TC5_CaptureStart( void );  
void TC5_CaptureStop( void );  
uint32_t TC5_CaptureFrequencyGet( void );  
uint16_t TC5_Capture16bitChannel0Get( void );  
uint16_t TC5_Capture16bitChannel1Get( void );  
void TC5_CaptureCallbackRegister( TC_CAPTURE_CALLBACK callback, uintptr_t context );  
void TC5_CaptureInterruptHandler( void );
```

Lab16-4 TC Waveform Capture

Step 11

```
...
int8_t DutyDistance = 2;
// TODO 16-4.01
volatile uint8_t TC5_IsCaptureReady = 0;
uint16_t TC5_Period = 0;
uint16_t TC5_Duty = 0;

void TC5_CaptureReady(TC_CAPTURE_STATUS status, uintptr_t context)
{
    if( (status & TC_CAPTURE_STAUTS_CAPTURE0_READY) == TC_CAPTURE_STAUTS_CAPTURE0_READY )
    {
        TC5_IsCaptureReady = 1;
    }
}

...
int main ( void )
{
    ...
    // TODO 16-4.02
    TC5_CaptureCallbackRegister( TC5_CaptureReady, (uintptr_t)NULL );
    TC5_CaptureStart();

    while ( true )
    { ... }
}
```

Lab16-4 TC Waveform Capture

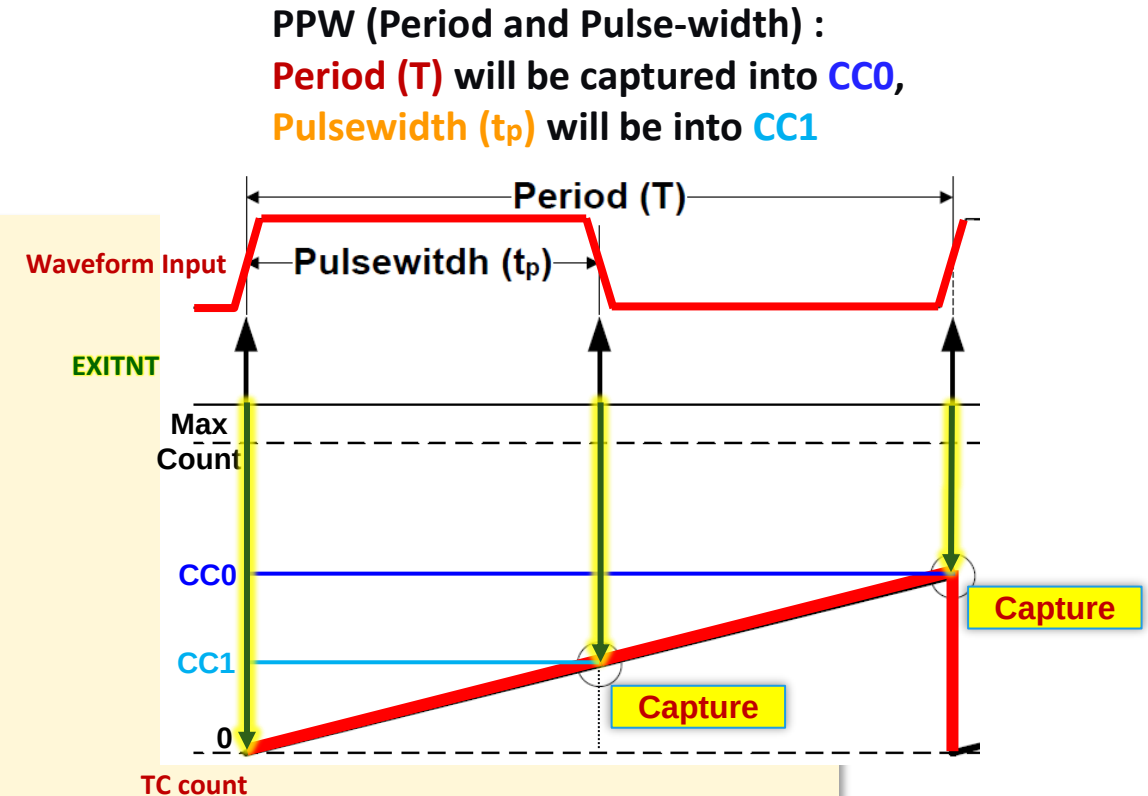
Step 12

```
...
int main ( void )
{
    ...
    while ( true )
    {
        ...
        if( ADC_IsCompleted )
        { ... }

        // TODO 16-4.03
        if( TC5_IsCaptureReady )
        {
            TC5_IsCaptureReady = 0;
            TC5_Period = TC5_Capture16bitChannel0Get();
            TC5_Duty = TC5_Capture16bitChannel1Get();

            myprintf("\033[5;1H");
            myprintf( "Capture Duty    = %3d%%", (int)(TC5_Duty*100/TC5_Period) );
            myprintf("\033[6;1H");
            myprintf( "Capture Period = %3d ms",
                    (int)(TC5_Period*1000/TC5_CaptureFrequencyGet()) );

        }
    }
}
```



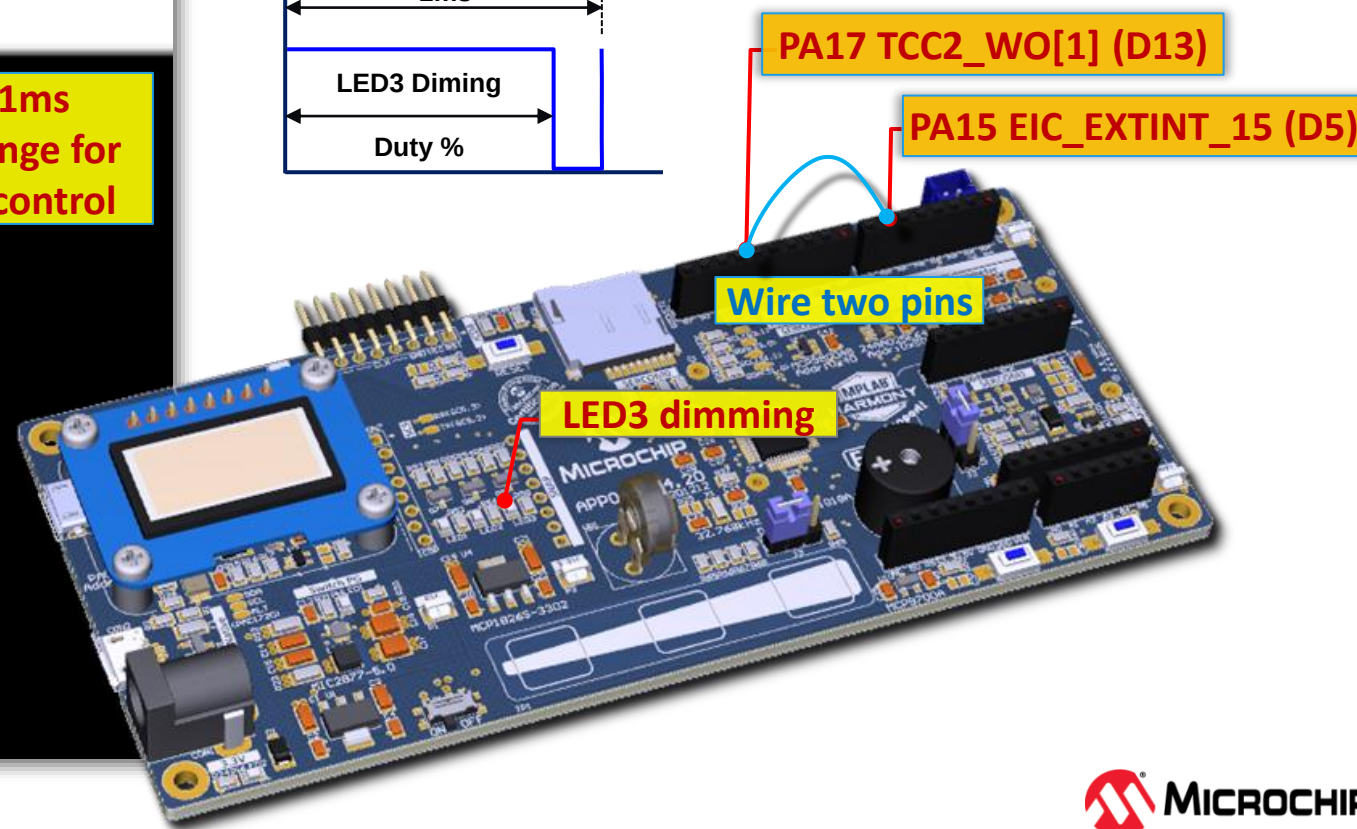
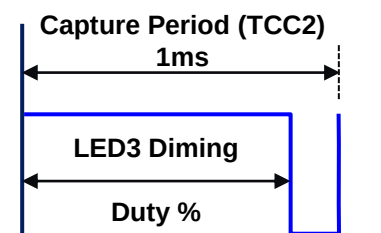
Lab16-4 TC Waveform Capture

Result 1 (Capture TCC2 PWM output)

- Wire from **PA17(TCC2/WO[1])** to **PA15(EIC_EXTINT_15)** could measure the period and duty output from **TCC2 waveform** that is LED3 dimming control.

```
VT COM20 - Tera Term VT
File Edit Setup Control Window Help
Hello World!!
VR1 Value : 2997
Temperature Value : 957
Received Data : p
Capture Duty    = 91%
Capture Period  = 1 ms
```

TCC2 period 1ms
Duty is auto change for
LED3 dimming control



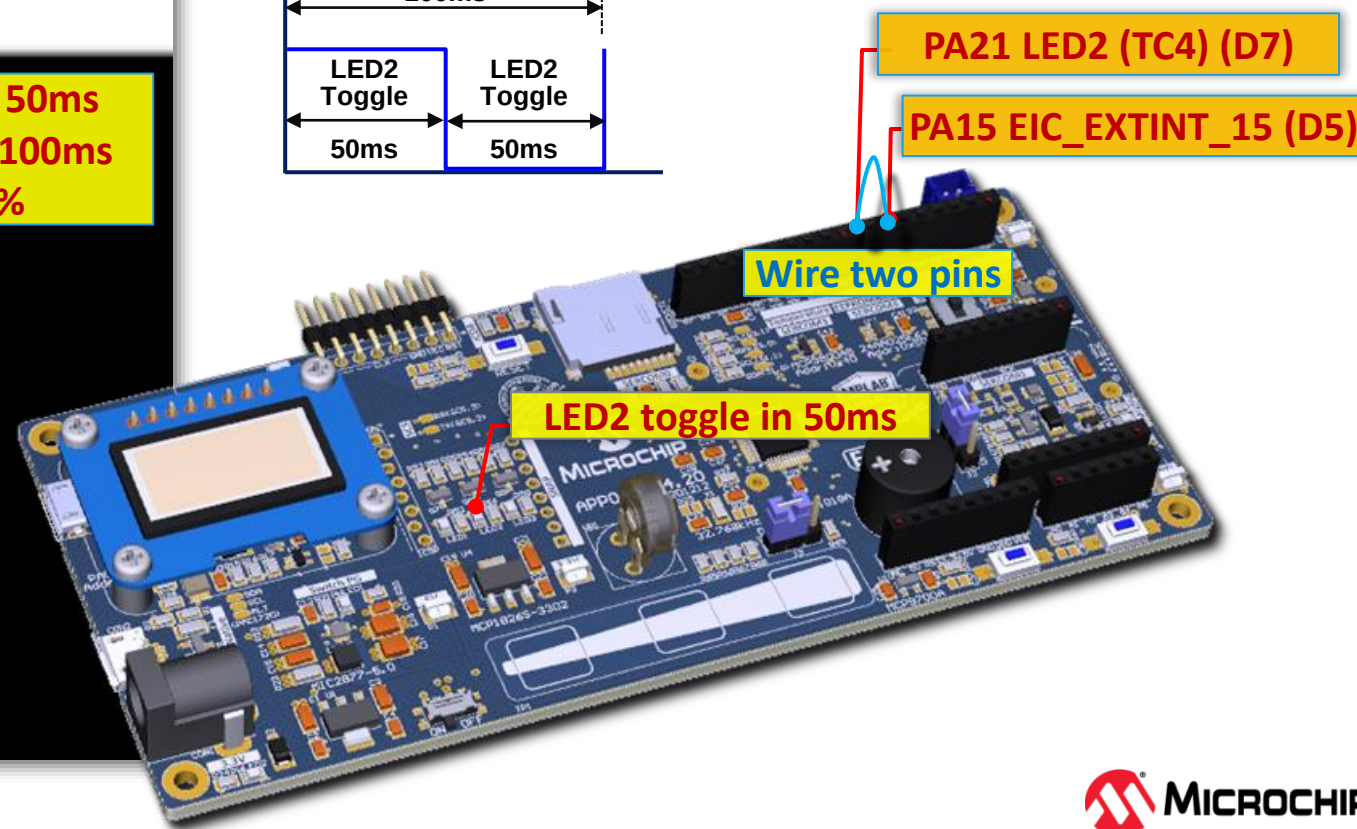
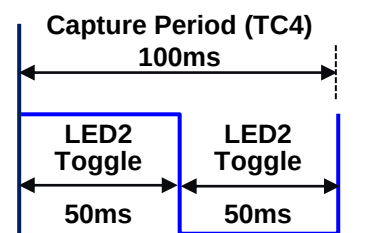
Lab16-4 TC Waveform Capture

Result 2 (Capture LED2 toggle frequency)

- Wire from **PA21(LED2)** to **PA15(EIC_EXTINT_15)** could measure the **LED2** toggle frequency that control by **TC4 Timer**.

```
VT COM20 - Tera Term VT
File Edit Setup Control Window Help
Hello World!!
VR1 Value : 2997
Temperature Value : 959
Received Data : c
Capture Duty   = 50%
Capture Period = 100 ms
```

LED2 toggle in 50ms
Clock Period is 100ms
Duty is 50%



Lab16-4 TC Waveform Capture

Discuss

- Wire from **PA20(LED1)** to **PA15(EIC_EXTINT_15)** and try to measure the **LED1** toggle frequency that control by **TC3 Timer**, what's happen and why?

VT COM20 - Tera Term VT

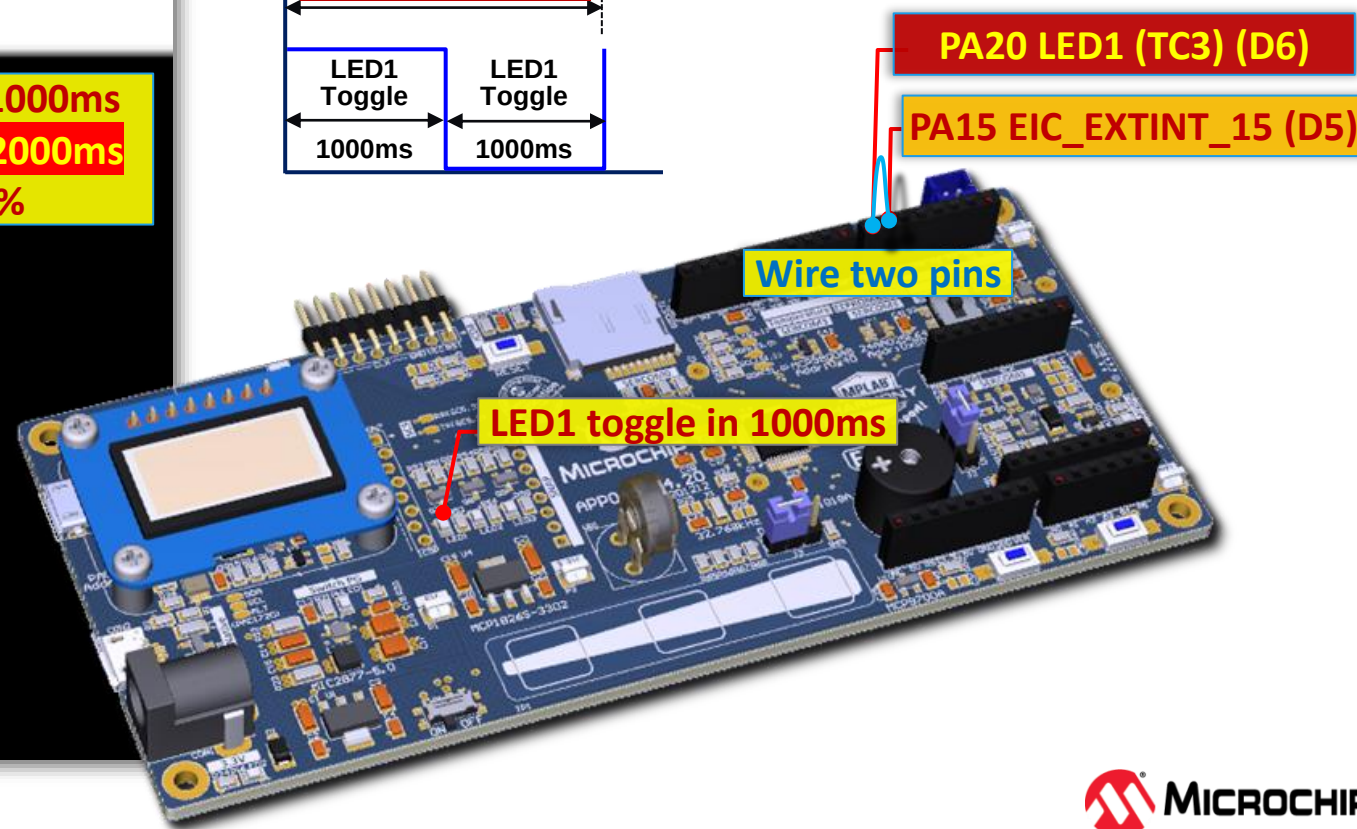
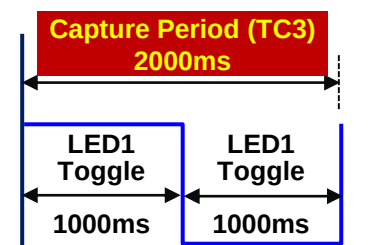
File Edit Setup Control Window Help

```

Hello World!
VR1 Value : 3381
Temperature Value : 958
Received Data : d
Capture Duty    = 166%
Capture Period  = 601 ms
  
```

LED1 toggle in 1000ms
Clock Period is 2000ms
Duty is 50%

TC5 Capture Resolution is :
 $48\text{M}/1024 = 46.875\text{ KHz}$
Max Capture Count is :
 0xFFFF (16bits)
Max Capture Period is :
 $0xFFFF / 46.875\text{KHz} = 1398\text{ ms}$



Lab16-4 TC Waveform Capture

Discuss

- To change the clock source of **TC5** from **GCLK0(48MHz)** to **GCLK1(32KHz)** then try measure the **LED1** frequency again, what's happen ? How about the **LED2** frequency?

COM20 - Tera Term VT

File Edit Setup Control Window Help

```

Hello World!
VR1 Value : 3381
Temperature Value : 955
Received Data : f
Capture Duty = 50%
Capture Period = 2000 ms
  
```

LED1 toggle in 1000ms
Clock Period is 2000ms
Duty is 50%

TC5 Capture Resolution is :
 $32K/1024 = 32 \text{ Hz}$
Max Capture Count is :
0xFFFF (16bits)
Max Capture Period is :
 $0xFFFF / 32\text{Hz} = 2048 \text{ s}$

Peripheral Clock Configuration

Peripheral	Enable	Source	Peripheral Clock Frequency
SERCOM2_CORE	☐	GCLK0	--
SERCOM3_CORE	☐	GCLK0	--
SERCOM4_CORE	☐	GCLK0	--
SERCOM5_CORE	☒	GCLK0	47,972,352 Hz
SYSCTRL_DFLL48	☒	GCLK1	32,768 Hz
SYSCTRL_FDPLL	☐	GCLK0	--
SYSCTRL_FDPLL32K	☐	GCLK0	--
TC3, TCC2	☒	GCLK0	47,972,352 Hz
TC4, TC5	☒	GCLK1	32,768 Hz
USB	☐	GCLK0	--
WDT	☐	GCLK0	--

TC4, TC5 **GCLK1** **32,768 Hz**

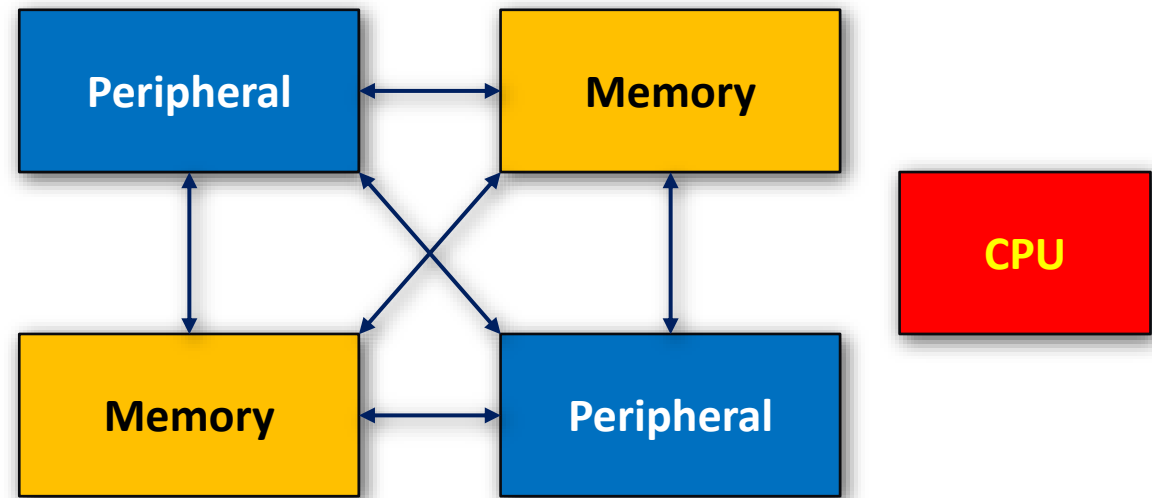
Close

Direct Memory Access Controller (DMAC)

- Single Transfer Block

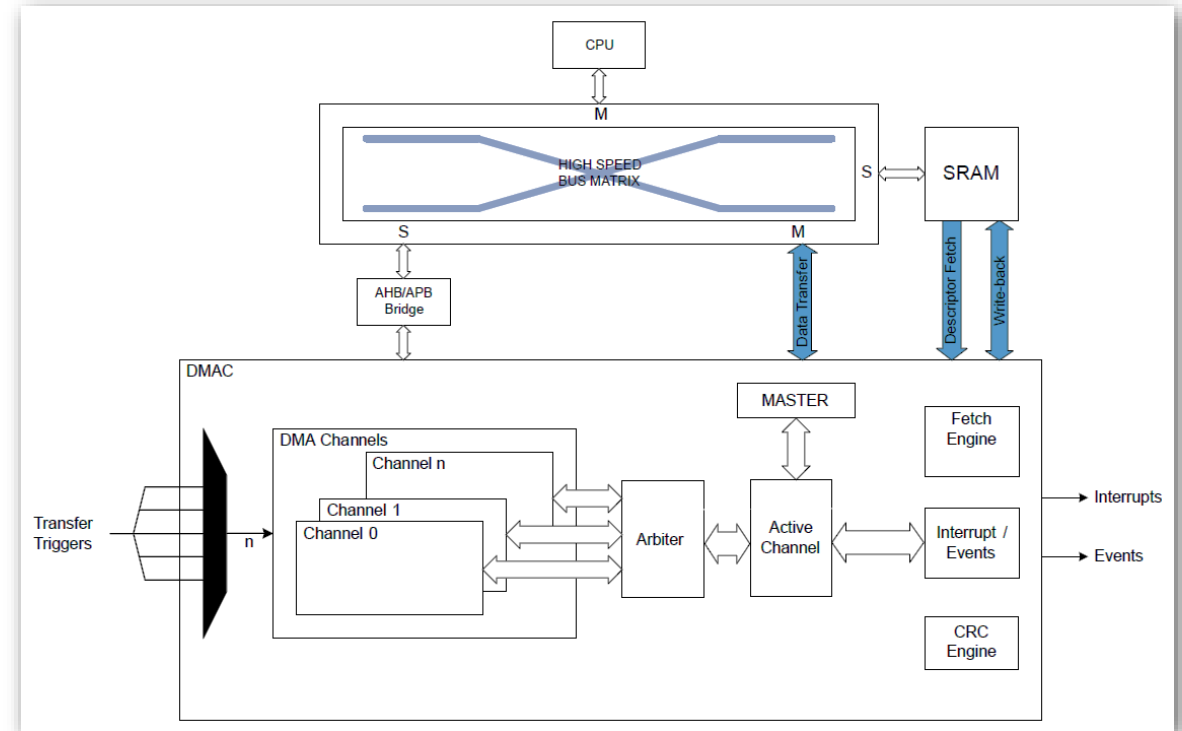
DMAC

- **DMAC allow transfer data between memories and peripheral without CPU interaction.**
- **Multiple data Transform,**
 - Peripheral <-> Peripheral
 - Memory <-> memory
 - Peripheral <-> memory
- **Up to 12 Channels, Flexible Channel arbitration**
 - 4 configurable priority level
 - Fixed or round-robin scheme
- **Four bus interfaces,**
 - Data Transfer Bus : DMA transfer
 - AHB/APB Bus : W/R IO Registers
 - Descriptor fetch bus : transfer descriptors Fetch
 - write-back bus : Write the transfer descriptor back to SRAM



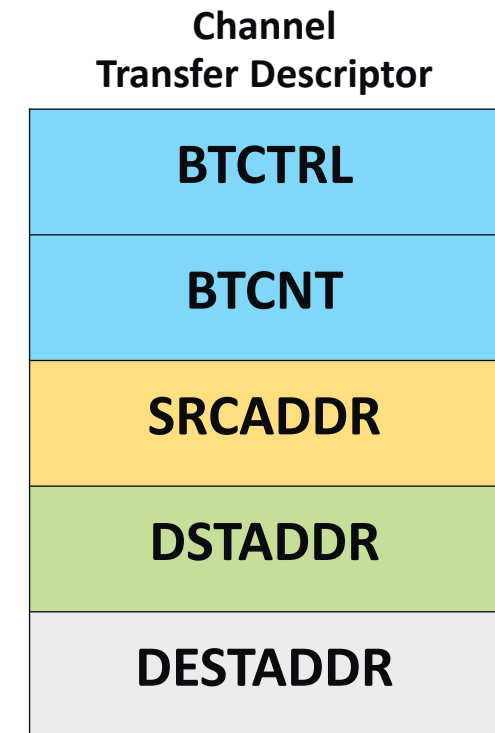
DMAC

- **3 Transfer Trigger Sources**
 - Software, Event and Dedicated Source
- **RAM-Based transfer descriptors**
 - Single transfer using one Descriptor
 - Multi-buffer or Circular Buffer
- **Address fixed or increment mode**
- **4 Event inputs & 4 Event outputs**
- **Optional interrupt generation**
 - Block transfer complete
 - On error detection
 - On channel suspend
- **CRC Polynomial Software**
Selectable to CRC-16 (CRC-CCITT) and CRC-32 (IEEE® 802.3)



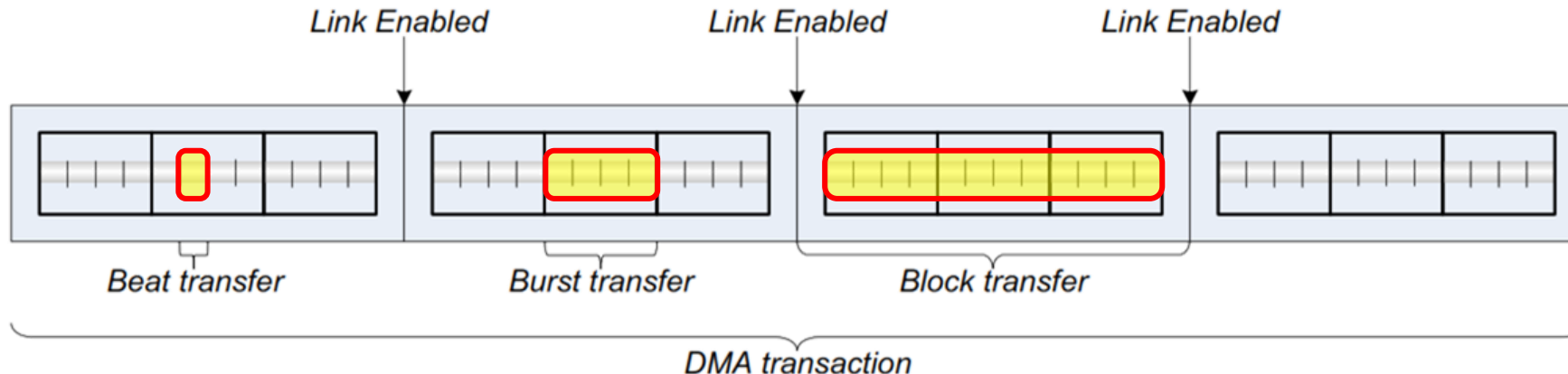
Transfer Descriptor

- The “Transfer Descriptor” use to describe that which data and how many data be transfer when DMAC be triggered.
- Channel Transfer Descriptor include below:
 - Operation mode
 - Beat size, Block Address increment
 - Interrupt action, Event out action
 - Block Size
 - Data from and to
 - Amount of beat to be transfer
 - Where is next Transfer Descriptor

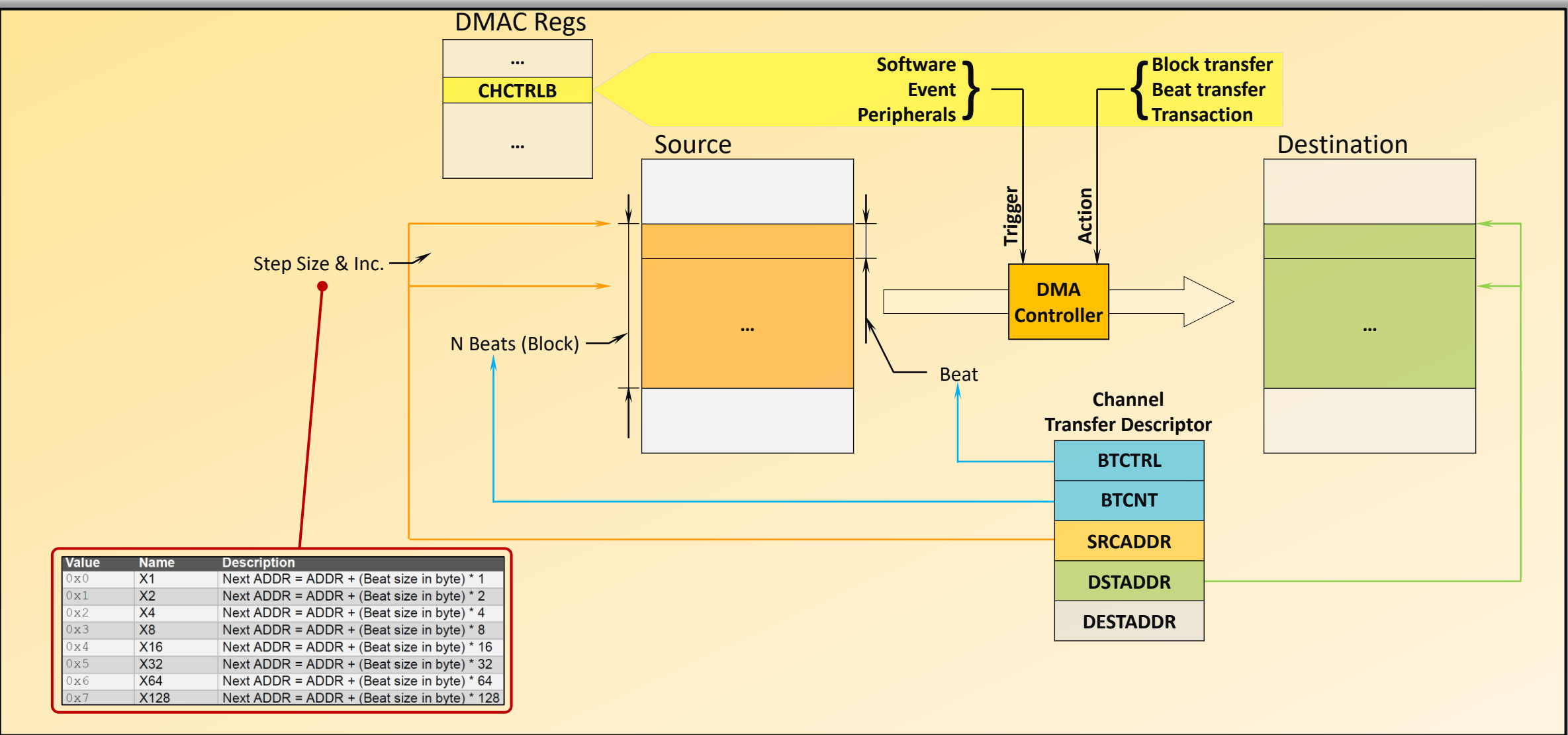


Transfer Size

- **Beat transfer**
 - Size of data length for one of DMAC transfer.
 - 8-bits, 16-bits or 32-bits
- **Burst transfer**
 - n Beat transfer at same time, Burst is “**atomic**” transfer.
 - SAM M0+ families not support.
- **Block transfer**
 - Amount of beat can be transfer at one transfer descriptor.
 - Max. number is $2^{16} * \text{beat} \Rightarrow 64K * 4 \text{ Bytes} = 256K \text{ Bytes}$
- **Transaction**
 - DMAC can link several transfer descriptors.
 - A transaction is complete transfer of all blocks.



Block diagram of Transaction



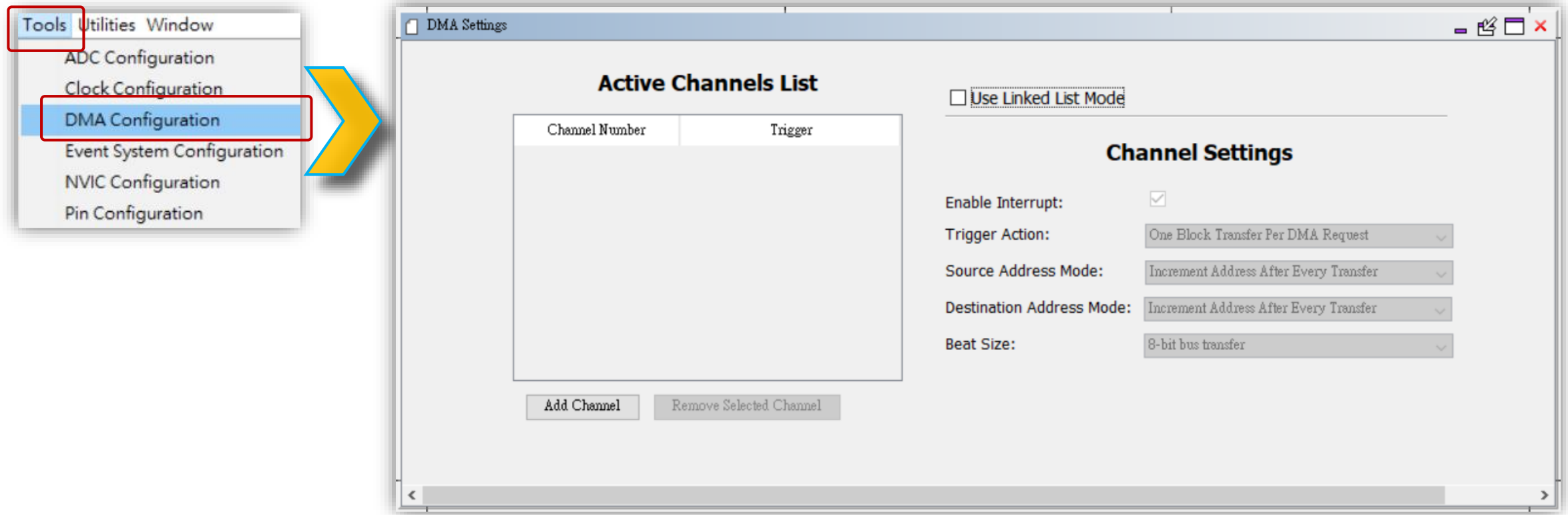
DMAC Trigger By Event or Dedicated Sources

Value	Name	Description	Value	Name	Description
0x00	DISABLE	Only software/event triggers	0x19	TC3 MC0	TC3 Match/Compare 0 Trigger
0x01	SERCOM0 RX	SERCOM0 RX Trigger	0x1A	TC3 MC1	TC3 Match/Compare 1 Trigger
0x02	SERCOM0 TX	SERCOM0 TX Trigger	0x1B	TC4 OVF	TC4 Overflow Trigger
0x03	SERCOM1 RX	SERCOM1 RX Trigger	0x1C	TC4 MC0	TC4 Match/Compare 0 Trigger
0x04	SERCOM1 TX	SERCOM1 TX Trigger	0x1D	TC4 MC1	TC4 Match/Compare 1 Trigger
0x05	SERCOM2 RX	SERCOM2 RX Trigger	0x1E	TC5 OVF	TC5 Overflow Trigger
0x06	SERCOM2 TX	SERCOM2 TX Trigger	0x1F	TC5 MC0	TC5 Match/Compare 0 Trigger
0x07	SERCOM3 RX	SERCOM3 RX Trigger	0x20	TC5 MC1	TC5 Match/Compare 1 Trigger
0x08	SERCOM3 TX	SERCOM3 TX Trigger	0x21	TC6 OVF	TC6 Overflow Trigger
0x09	SERCOM4 RX	SERCOM4 RX Trigger	0x22	TC6 MC0	TC6 Match/Compare 0 Trigger
0x0A	SERCOM4 TX	SERCOM4 TX Trigger	0x23	TC6 MC1	TC6 Match/Compare 1 Trigger
0x0B	SERCOM5 RX	SERCOM5 RX Trigger	0x24	TC7 OVF	TC7 Overflow Trigger
0x0C	SERCOM5 TX	SERCOM5 TX Trigger	0x25	TC7 MC0	TC7 Match/Compare 0 Trigger
0x0D	TCC0 OVF	TCC0 Overflow Trigger	0x26	TC7 MC1	TC7 Match/Compare 1 Trigger
0x0E	TCC0 MC0	TCC0 Match/Compare 0 Trigger	0x27	ADC RESRDY	ADC Result Ready Trigger
0x0F	TCC0 MC1	TCC0 Match/Compare 1 Trigger	0x28	DAC EMPTY	DAC Empty Trigger
0x10	TCC0 MC2	TCC0 Match/Compare 2 Trigger	0x29	I2S RX 0	I2S RX 0 Trigger
0x11	TCC0 MC3	TCC0 Match/Compare 3 Trigger	0x2A	I2S RX 1	I2S RX 1 Trigger
0x12	TCC1 OVF	TCC1 Overflow Trigger	0x2B	I2S TX 0	I2S TX 0 Trigger
0x13	TCC1 MC0	TCC1 Match/Compare 0 Trigger	0x2C	I2S TX 1	I2S TX 1 Trigger
0x14	TCC1 MC1	TCC1 Match/Compare 1 Trigger	0x2D	OVF	TCC3 Overflow Trigger
0x15	TCC1 MC2	TCC1 Match/Compare 2 Trigger	0x2E	TCC3 MC0	TCC3 Match/Compare 0 Trigger
0x16	TCC1 MC3	TCC1 Match/Compare 3 Trigger	0x2F	TCC3 MC1	TCC3 Match/Compare 1 Trigger
0x17	TCC2 OVF	TCC2 Overflow Trigger	0x30	TCC3 MC2	TCC3 Match/Compare 2 Trigger
0x18	TCC2 MC0	TCC2 Match/Compare 0 Trigger	0x31	TCC3 MC3	TCC3 Match/Compare 3 Trigger
0x19	TCC2 MC1	TCC2 Match/Compare 1 Trigger			
0x1A	TCC2 MC2	TCC2 Match/Compare 2 Trigger			
0x1B	TCC2 MC3	TCC2 Match/Compare 3 Trigger			
0x1C	TCC3 OVF	TCC3 Overflow Trigger			
0x1D	TCC3 MC0	TCC3 Match/Compare 0 Trigger			
0x1E	TCC3 MC1	TCC3 Match/Compare 1 Trigger			
0x1F	TCC3 MC2	TCC3 Match/Compare 2 Trigger			
0x20	TCC3 MC3	TCC3 Match/Compare 3 Trigger			

EVACT[2:0]	Name	Description
0x0	NOACT	No action
0x1	TRIG	Normal Transfer and Conditional Transfer on Strobe trigger
0x2	CTRIG	Conditional transfer trigger
0x3	CBLOCK	Conditional block transfer
0x4	SUSPEND	Channel suspend operation
0x5	RESUME	Channel resume operation
0x6	SSKIP	Skip next block suspend action
0x7	-	Reserved

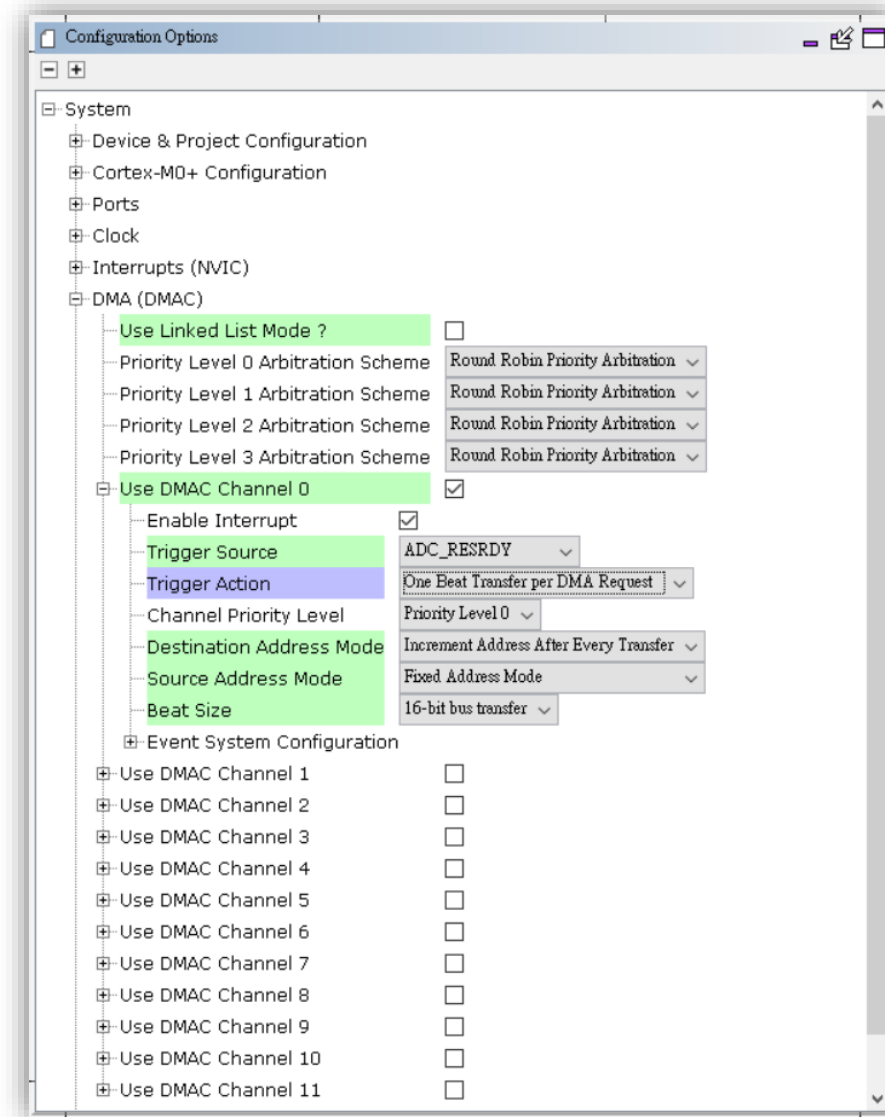
DMAC Function at MHC

- **DMAC** already be active at new project create by “32-bit MPLAB Harmony 3 Project”.
- Configure **DMAC** in DMA Settings.



DMAC Function

Configuration Options Example



DMAC Functions

Interface Routines

```
void DMAC_Initialize( void );
bool DMAC_ChannelTransfer
    ( DMAC_CHANNEL channel, const void *srcAddr, const void *destAddr, size_t blockSize )
void DMAC_ChannelCallbackRegister
    ( DMAC_CHANNEL channel, const DMAC_CHANNEL_CALLBACK eventHandler,
      const uintptr_t contextHandle )

uint16_t DMAC_ChannelGetTransferredCount( DMAC_CHANNEL channel )
DMAC_TRANSFER_EVENT DMAC_ChannelTransferStatusGet( DMAC_CHANNEL channel )
bool DMAC_ChannelsBusy( DMAC_CHANNEL channel )

DMAC_CHANNEL_CONFIG DMAC_ChannelSettingsGet( DMAC_CHANNEL channel )
bool DMAC_ChannelSettingsSet( DMAC_CHANNEL channel, DMAC_CHANNEL_CONFIG settings )
void DMAC_ChannelDisable( DMAC_CHANNEL channel )
void DMAC_ChannelSuspend( DMAC_CHANNEL channel )
void DMAC_ChannelResume( DMAC_CHANNEL channel )
```

DMAC Code Example

Code Example

```
void DMA_Complete(DMAC_TRANSFER_EVENT event, uintptr_t contextHandle)
{
    if (event == DMAC_TRANSFER_EVENT_COMPLETE)
    {
        DMA_IsCompleted = 1;
        DMAC_ChannelTransfer(DMAC_CHANNEL_0,
            (const void *) &(ADC_REGS->ADC_RESULT),
            (const void *) ADC_Result, sizeof (ADC_Result));
    }
}

int main (void)
{
    SYS_Initialize ( NULL );
    ...
    DMAC_ChannelCallbackRegister(DMAC_CHANNEL_0, DMA_Complete, (uintptr_t) NULL);
    DMAC_ChannelTransfer(DMAC_CHANNEL_0,
        (const void *) &(ADC_REGS->ADC_RESULT),
        (const void *) ADC_Result, sizeof (ADC_Result));

    while( true )
    {
        ...
    }
}
```

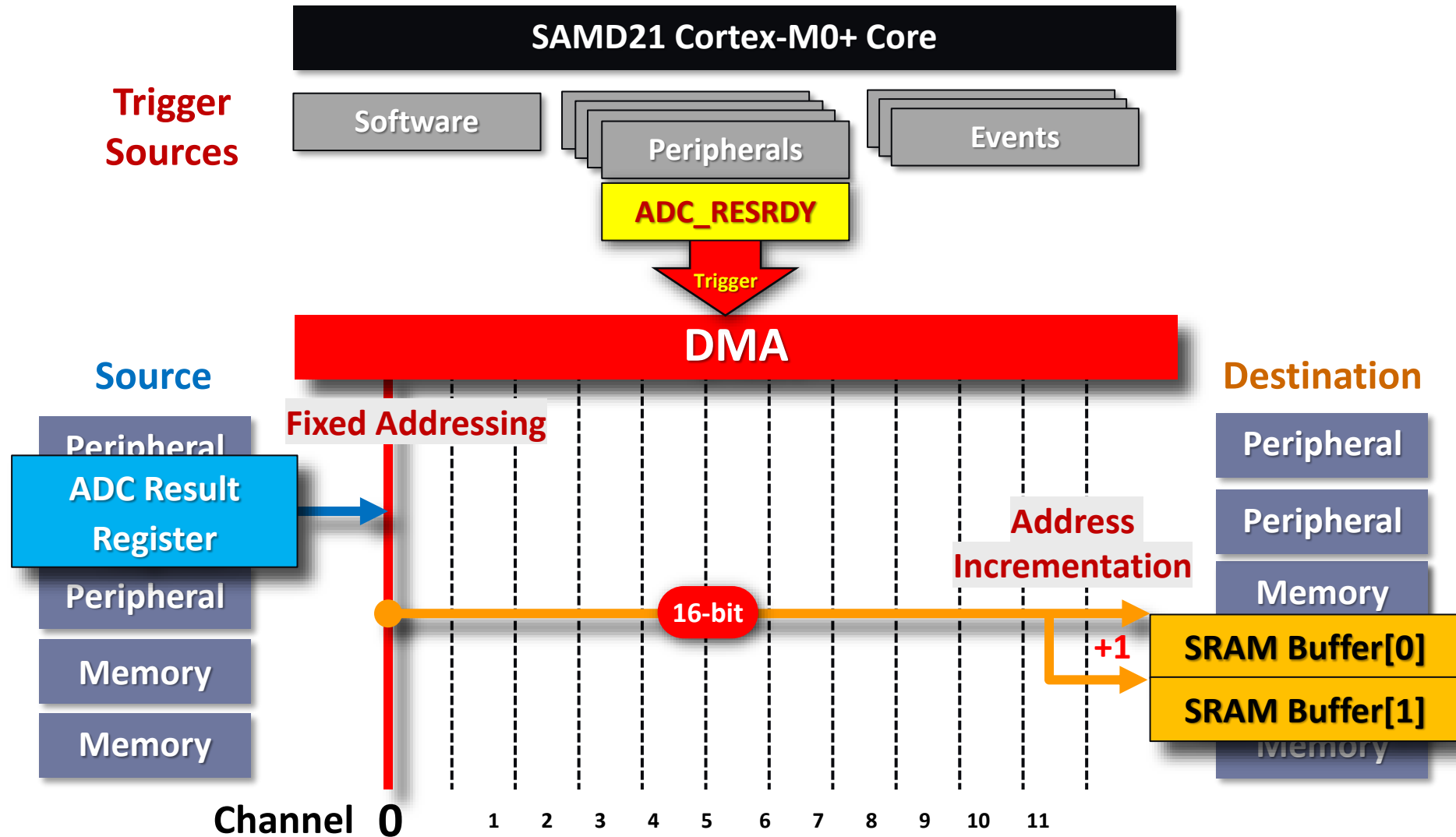
One-Shot Event

Channel	: 0
Source	: ADC_RESULT (SFR Register)
Destination	: ADC_Result[] (SRAM)
Size	: Size of ADC_Result[] (n Bytes)

🔧 Lab13-3 ADC with DMAC

- Please continue from [SAM2001 Lab13 \(ADC PinScan Interrupt Callback SWTrigger\)](#)
- Try to add new channel of **DMAC** at DMA Settings on the project.
- Use **DMAC** to move ADC conversion result to user defined memory (SRAM) when ADC converted.
- Move once by each ADC conversion completed (**ADC_RESRDY**) event.
 - DMAC Channel : **0**
 - Trigger Source : **ADC_RESRDY**
 - Trigger Action : **One Beat Transfer**
 - Source / Destination Address Mode : **Fixed / Increment**
 - Beat Size : **16-bits**
 - Source / Destination Register or memory : **ADC_RESULT** (Register) / **ADC_Result[]** (SRAM)
- **Let's go !**

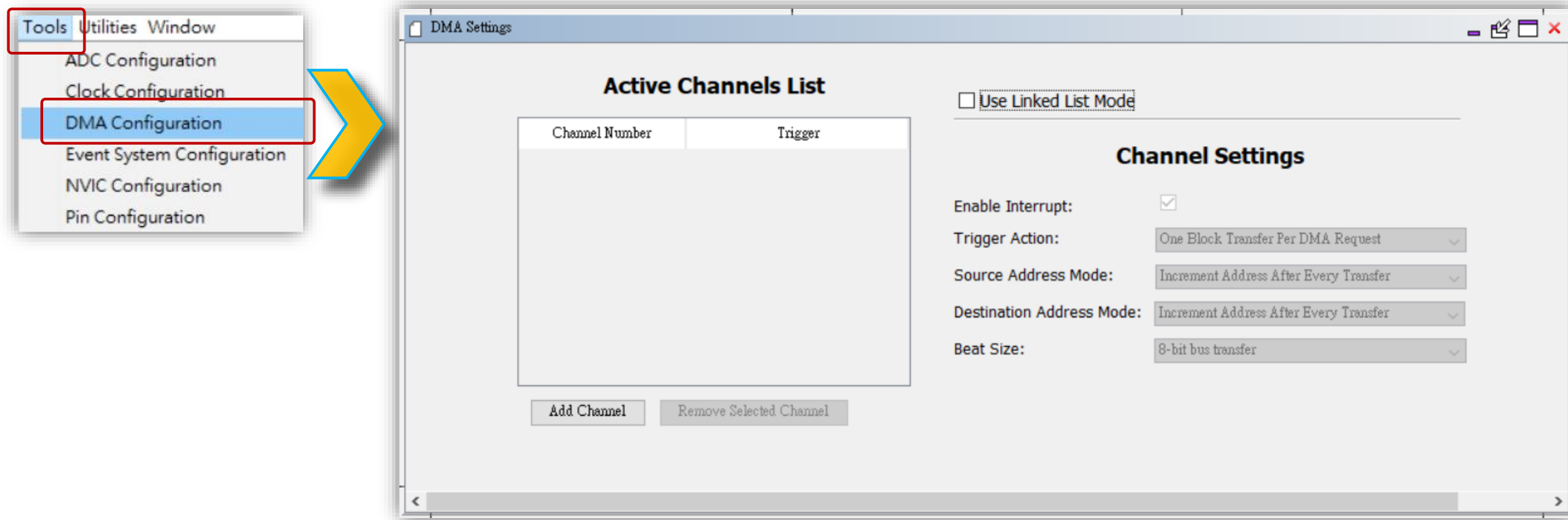
⚙️ Lab13-3 ADC with DMAC



⚙️ Lab13-3 ADC with DMAC

Step 1

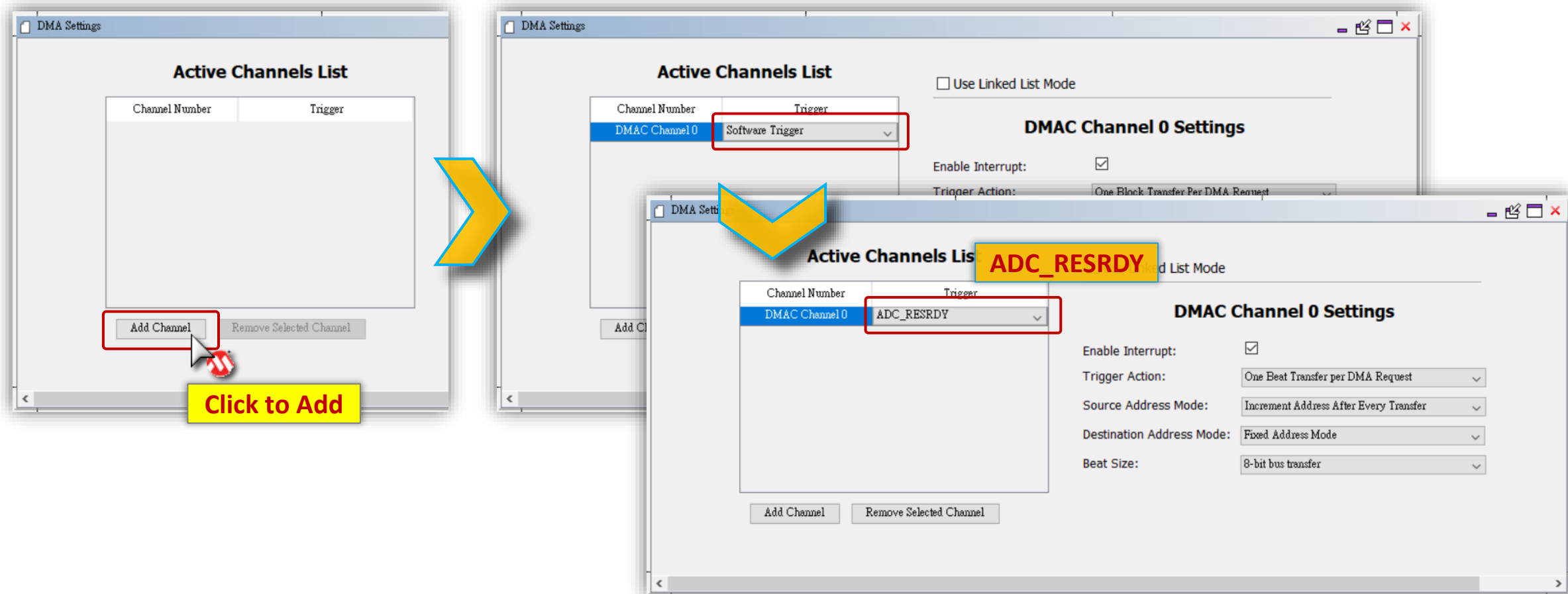
- Find DMA Settings at tools DMA Configuration



⚙️ Lab13-3 ADC with DMAC

Step 2

- Add DMAC channel and set trigger source to **ADC_RESRDY**



⚙️ Lab13-3 ADC with DMAC

Step 3

DMA Settings

Active Channels List

Channel Number	Trigger
DMAC Channel0	ADC_RESRDY

DMAC Channel 0 Settings

☐ Use Linked List Mode

Enable Interrupt: ☒

Trigger Action: One Beat Transfer per DMA Request

Source Address Mode: Fixed Address Mode

Destination Address Mode: Increment Address After Every Transfer

Beat Size: 16-bit bus transfer

Add Channel Remove Selected Channel

Enable Interrupt

One Beat Transfer

Fixed Address Mode

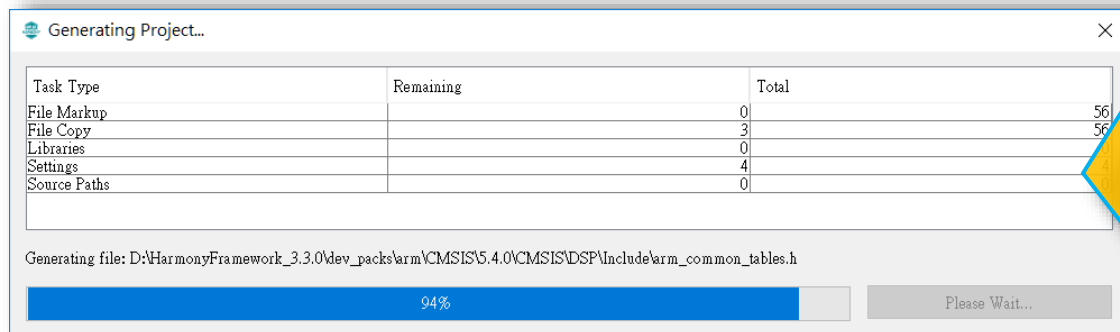
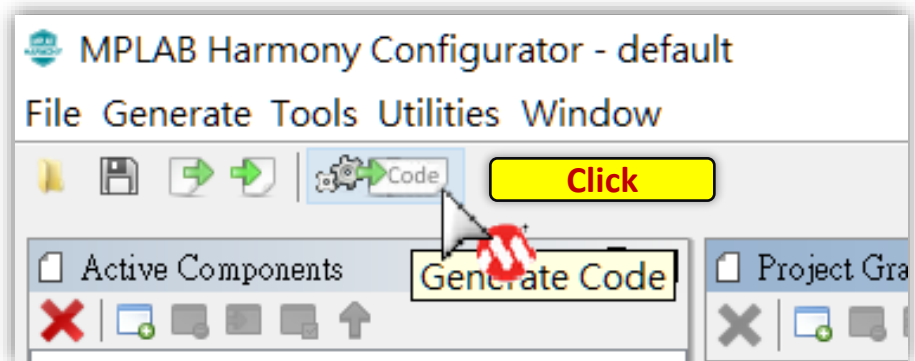
Increment Address After ...

16-bit bus transfer

⚙️ Lab13-3 ADC with DMAC

Step 4

- Click  to Generate Code.



Lab13-3 ADC with DMAC

Step 5

```
// TODO 13-3.01
//volatile uint8_t ADC_IsCompleted = 0;
//volatile int8_t ADC_ChannelIdx = 0;

// TODO 13-3.02
//void ADC_Complete(ADC_STATUS status, uintptr_t context) {
//    if (status & ADC_INTFLAG_RESRDY_Msk) {
//        ADC_Result[ADC_ChannelIdx] = ADC_ConversionResultGet();
//        ADC_ChannelIdx = (ADC_ChannelIdx + 1) % 2;
//        if (ADC_ChannelIdx == 0) {
//            ADC_IsCompleted = 1;
//        }
//    }
//}

int main (void)
{
    SYS_Initialize(NULL);
    ...
    // TODO 13-3.03
    //ADC_CallbackRegister( ADC_Complete, (uintptr_t )NULL );

    while ( true )
    {
    }
    return ( EXIT_FAILURE);
}
```

Lab13-3 ADC with DMAC

Step 6

```
...  
  
int main (void)  
{  
    SYS_Initialize(NULL);  
    ...  
    while ( true )  
    {  
  
        // TODO 13-3.04  
        // if (ADC_IsCompleted)  
        // {  
        //     ADC_IsCompleted = 0;  
        //     myprintf("VR1 Value : %4d\r\n", ADC_Result[0]);  
        //     myprintf("Temperature Value : %4d\r\n", ADC_Result[1]);  
        // }  
  
    }  
    return ( EXIT_FAILURE);  
}
```

Lab13-3 ADC with DMAC

Step 7

```
// TODO 13-3.05
volatile uint8_t DMA_IsCompleted = 0;

// TODO 13-3.06
void DMA_Complete(DMAC_TRANSFER_EVENT event, uintptr_t contextHandle)
{
    if (event == DMAC_TRANSFER_EVENT_COMPLETE)
    {
        DMA_IsCompleted = 1;
        DMAC_ChannelTransfer(DMAC_CHANNEL_0,
                             (const void *) &(ADC_REGS->ADC_RESULT),
                             (const void *) ADC_Result,
                             sizeof (ADC_Result));
    }
}

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

    while ( true )
    {
    }
    return ( EXIT_FAILURE);
}
```

Lab13-3 ADC with DMAC

Step 8

```
int main (void)
{
    SYS_Initialize(NULL);
    ...
    // TODO 13-3.07
    DMAC_ChannelCallbackRegister(DMAC_CHANNEL_0, DMA_Complete, (uintptr_t) NULL);
    DMAC_ChannelTransfer(DMAC_CHANNEL_0,
                        (const void *) &(ADC_REGS->ADC_RESULT),
                        (const void *) ADC_Result,
                        sizeof (ADC_Result));

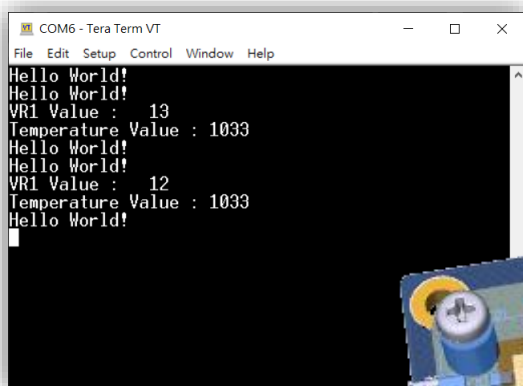
    while ( true )
    {
        SYS_Tasks();
        ...

        // TODO 13-3.08
        if (DMA_IsCompleted)
        {
            DMA_IsCompleted = 0;
            myprintf("VR1 Value : %4d\r\n", ADC_Result[0]);
            myprintf("Temperature Value : %4d\r\n", ADC_Result[1]);
        }
    }
    return ( EXIT_FAILURE);
}
```

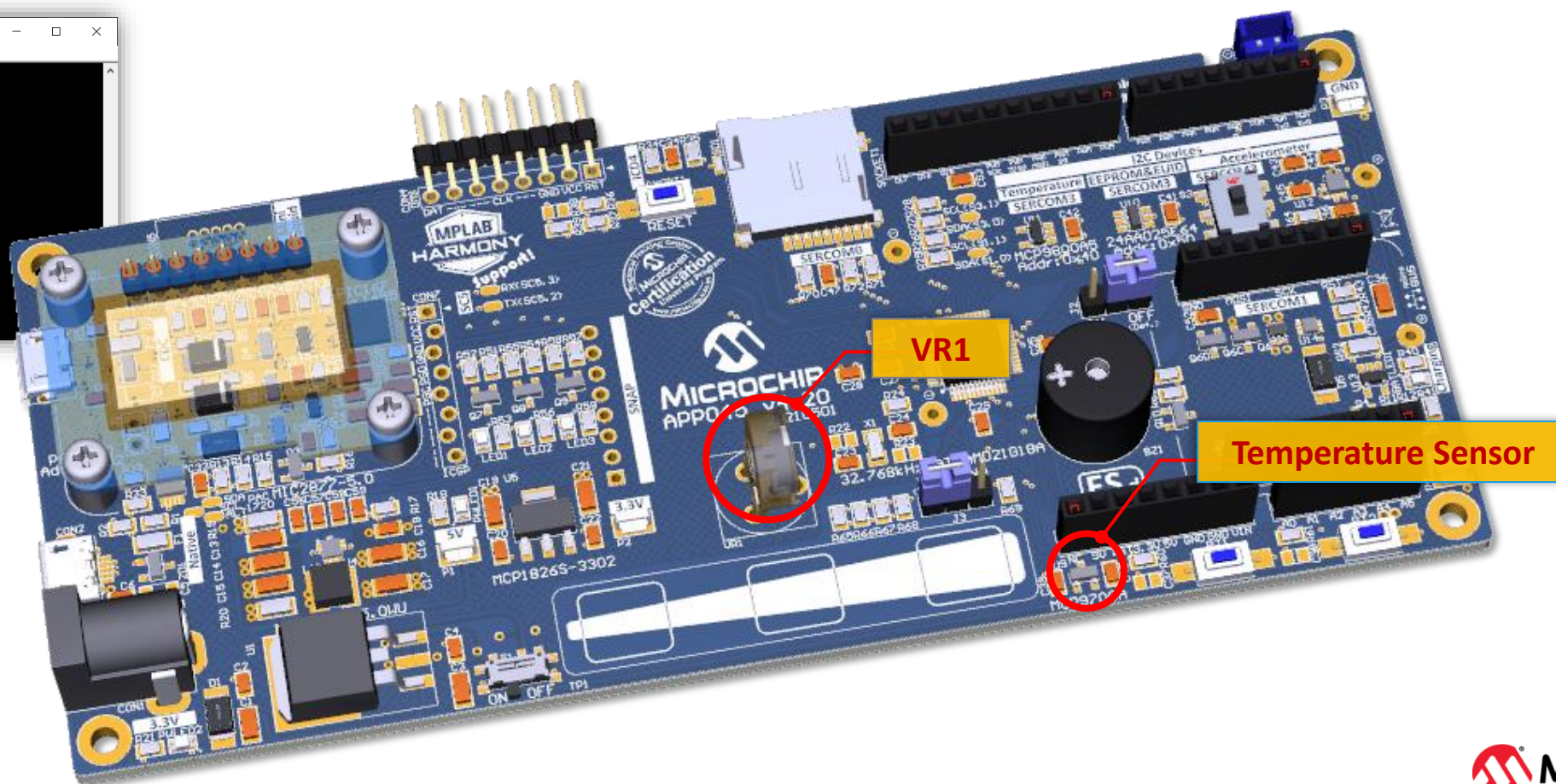

⚡ Lab13-3 ADC with DMAC

Result

- “Hello World!!” string and ADC result of VR1 / Temperature Sensor will output to terminal with two second period. Same as Lab 13, but data moving by DMAC now.



```
COM6 - Tera Term VT
File Edit Setup Control Window Help
Hello World!
Hello World!
VR1 Value : 13
Temperature Value : 1033
Hello World!
Hello World!
VR1 Value : 12
Temperature Value : 1033
Hello World!
```



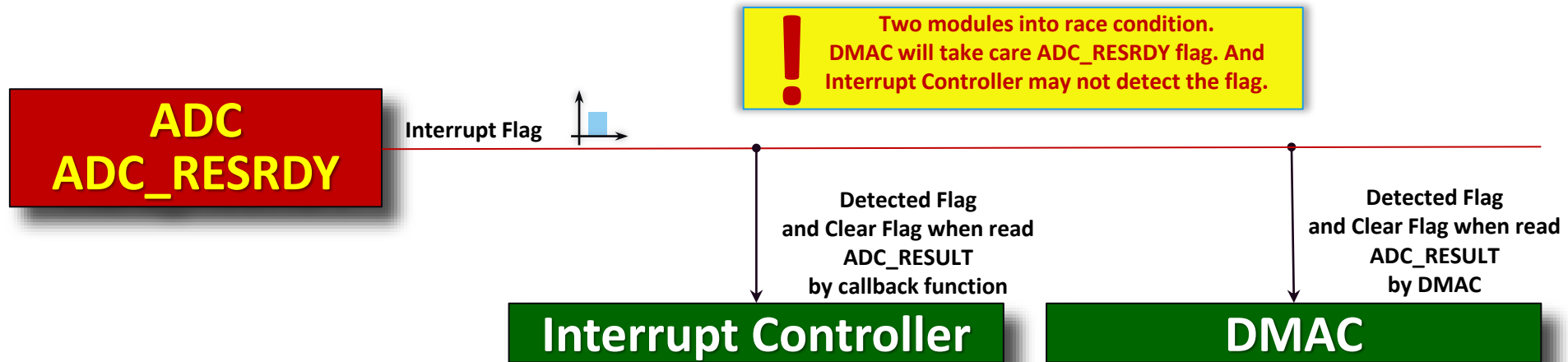
⚡ Lab13-3 ADC with DMAC

- **Discuss**

- Could set ADC callback and set the DMAC to moving ADC result when ADC result ready at the same time ?

```
// TODO 13-3.03  
//ADC_CallbackRegister( ADC_Complete, (uintptr_t )NULL );
```

- **The Answer is NO.** ADC_RESRDY flag will be cleared automatically when DMAC reading/moving **ADC_RESULT** (Register). So, interrupt request of ADC_RESRDY and DMAC action couldn't be enable at the same time.



⚡ Lab13-4 ADC with DMAC and Interrupt

- This is a demo lab. and the result is not correct. Just use to show race condition for **ADC_RESRDY** by interrupt controller and DMAC. Demo that the interrupt request of ADC result ready and DMAC action by the flay same.
- Uncomment at Lab 13-3.01 ~ 13-3.03 and modify 13-3.02 follow below for ADC Callback.
- Just open, build and program the firmware to MCU, then try it.
- **Let's go !**

**Reference*

```
// TODO 13-3.01
volatile uint8_t ADC_IsCompleted = 0;
volatile int8_t ADC_ChannelIdx = 0;
```

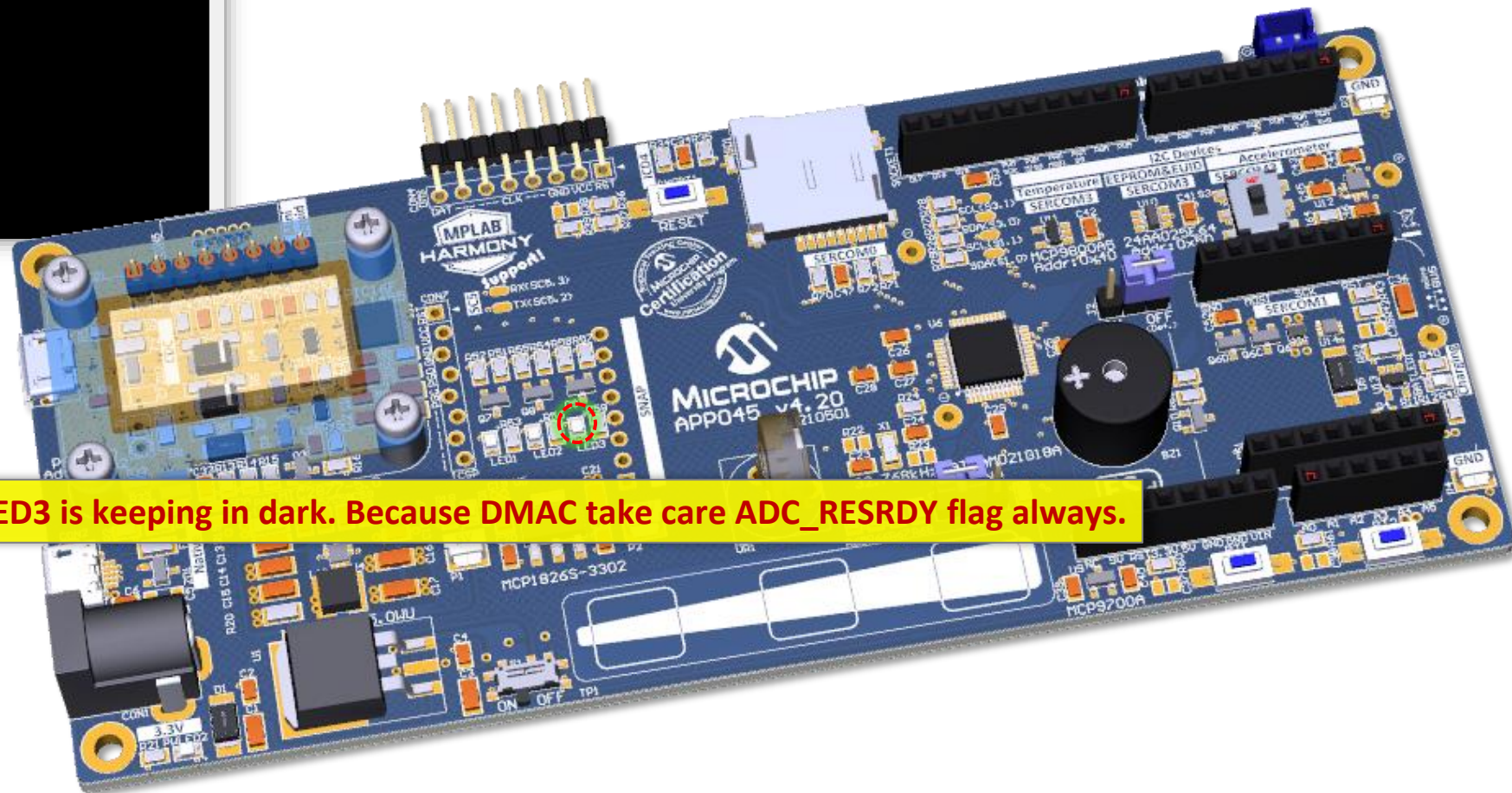
```
// TODO 13-3.03
ADC_CallbackRegister
    ( ADC_Complete, (uintptr_t )NULL );
```

```
// TODO 13-3.02
void ADC_Complete(ADC_STATUS status, uintptr_t context)
{
    if (status & ADC_INTFLAG_RESRDY_Msk)
    {
        /* Dummy Read */
        ADC_Result[ADC_ChannelIdx] =
            ADC_ConversionResultGet();
        LED3_Toggle();
    }
}
```


⚡ Lab13-4 ADC with DMAC and Interrupt

Result

```
COM6 - Tera Term VT
File Edit Setup Control Window Help
Hello World!
Hello World!
VR1 Value : 13
Temperature Value : 1033
Hello World!
Hello World!
VR1 Value : 12
Temperature Value : 1033
Hello World!
```



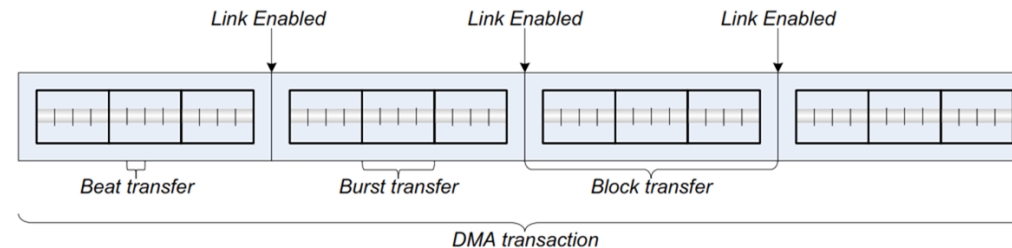
LED3 is keeping in dark. Because DMAC take care ADC_RESRDY flag always.

Direct Memory Access Controller (DMAC)

- Linked List for Transfer Block

Review the Transfer Descriptor

- The “Transfer Descriptor” used to describe that which data and how many data be transfer when DMAC be triggered.
 - Beat Count (BTCNT) : how many triggered to finish this Transfer Descriptor.
 - Source (SRCADDR) : It's indicated source address.
 - Destination (DSTADDR) : It's indicated Destination address.
 - Descriptor (DESCADDR) : It's indicated that where is next Transfer Descriptor.
-
- Beat_(8, 16, 32-bits) transfer : Data of amount be transferred every triggered.
 - Block transfer : To finished BTCNT times at single Transfer Descriptor.
 - Transaction : To finished all Transfer Descriptors.

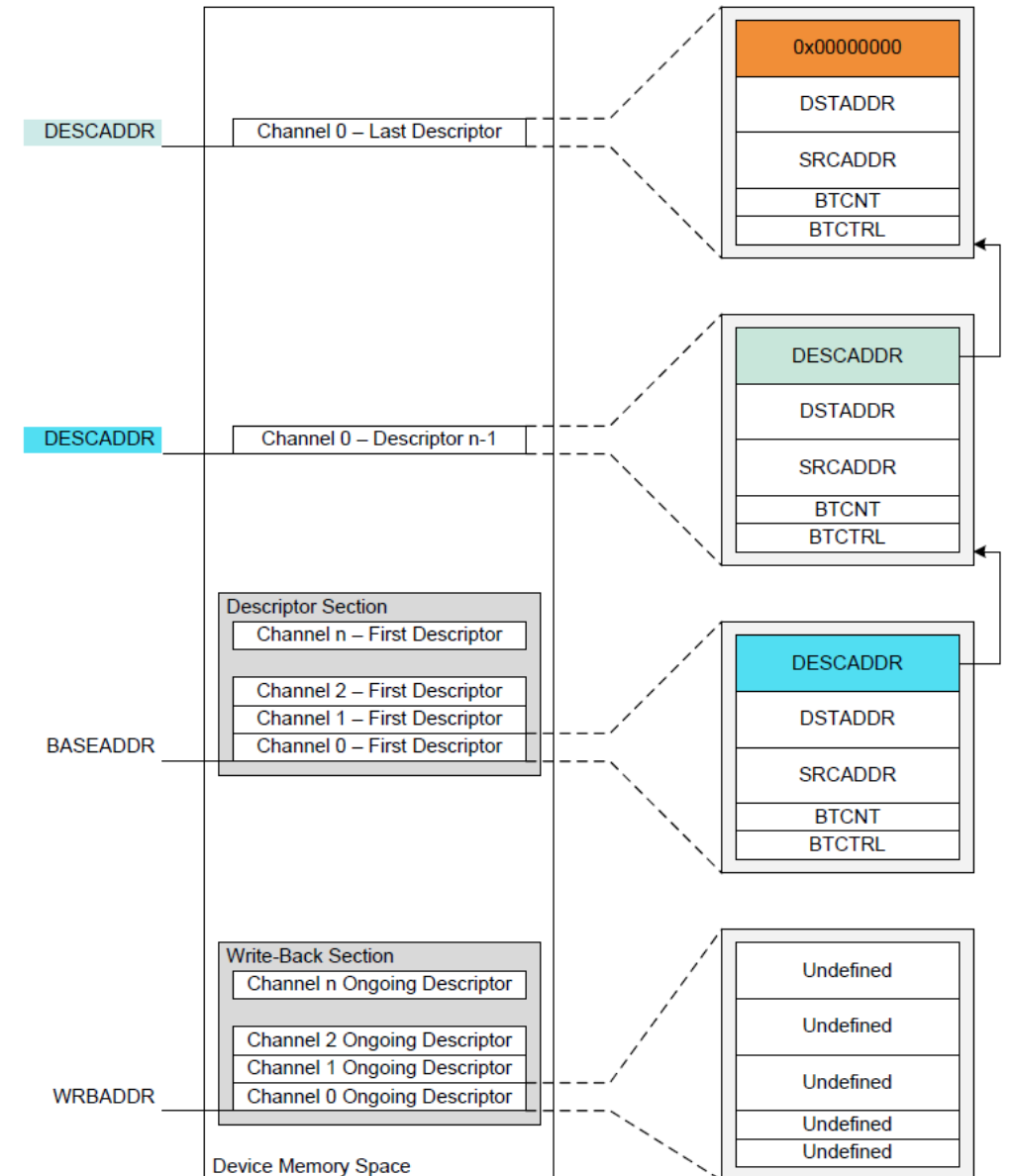


Channel
Transfer Descriptor

BTCTRL
BTCNT
SRCADDR
DSTADDR
DESCADDR

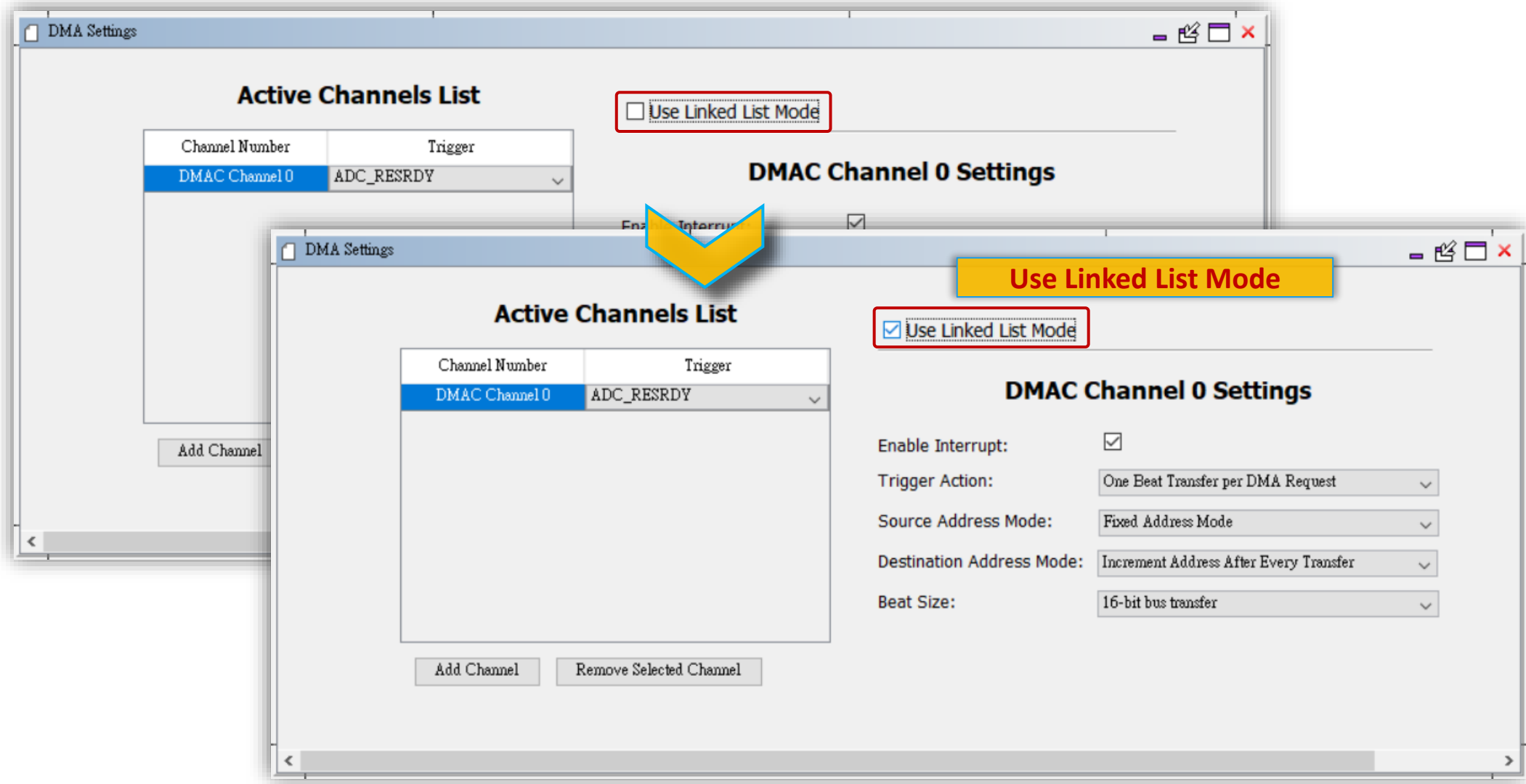
Linked Descriptors

- A transaction can consist of either a single block transfer or of several blocks transfer.
- When a transaction consists of several blocks transfer, it's done with the help of linked descriptors.
- Based on linked descriptors you could moving differ type source form / to
- differ destination by differ transfer descriptor. DESCADDR use to indicate where is next descriptor Address, 0x00000000 (NULL) means last one transfer descriptor.



Linked List Mode Enable at MHC

- Enable Linked List Mode in DMA Settings.



DMAC Functions for Link List Mode

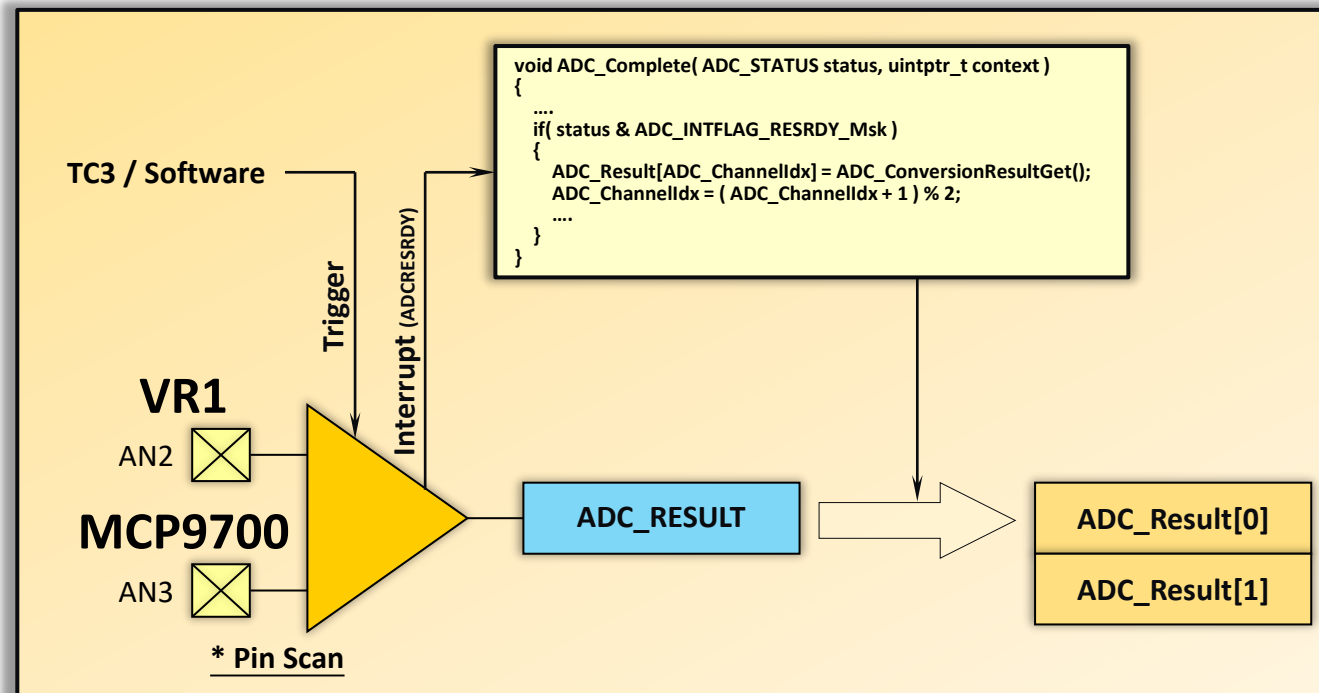
Interface Routines

```
bool DMAC_ChannelLinkedListTransfer  
(DMAC_CHANNEL channel, dmac_descriptor_registers_t* channelDesc);
```

```
void DMAC_LinkedListDescriptorSetup  
(dmac_descriptor_registers_t* currentDescriptor,  
DMAC_CHANNEL_CONFIG setting,  
const void *srcAddr,  
const void *destAddr,  
size_t blockSize,  
dmac_descriptor_registers_t* nextDescriptor);
```

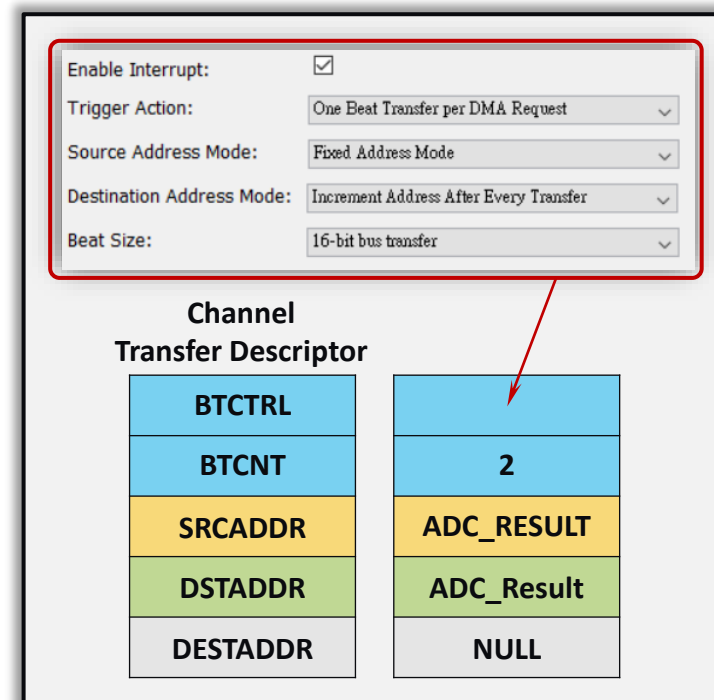
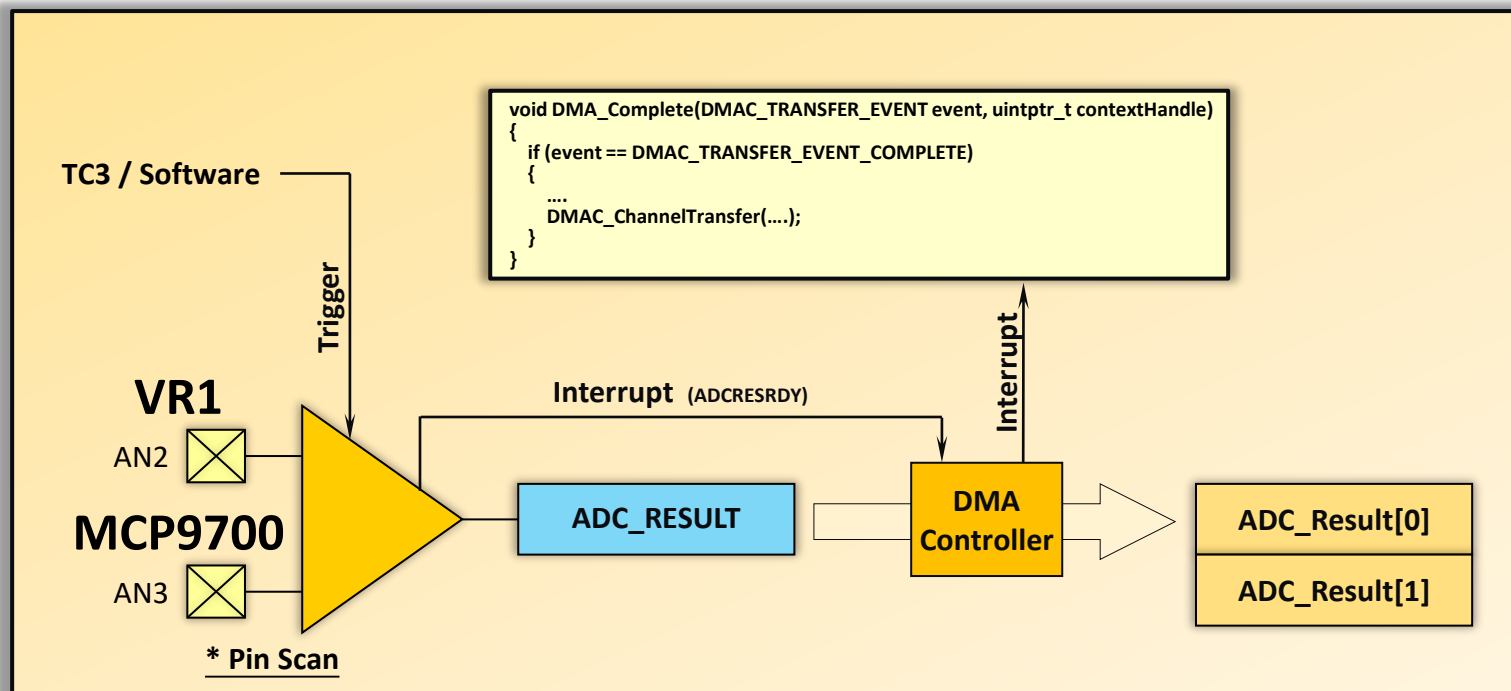
Review Block Diagram for SAM2001 Lab13

SAM2001 Lab13_ADC_PinScan_Interrupt_Callback_SWTrigger



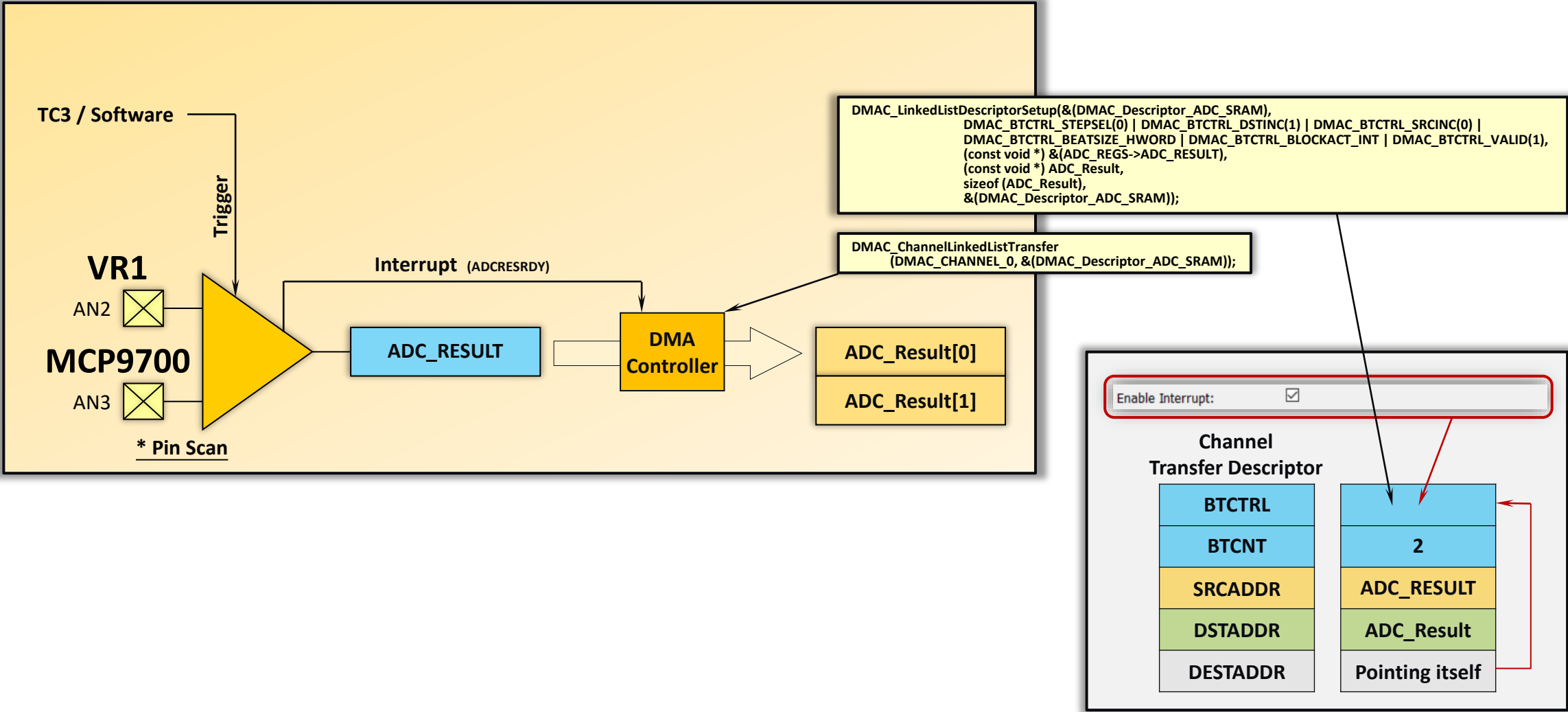
Review Block Diagram for SAM2001ADV Lab13-3

SAM2001ADV Lab13-3_ADC_with_DMAL



Block Diagram for SAM2001ADV Lab13-5

SAM2001ADV Lab13-5_ADC_with_Descriptors_Link_of_DMAL



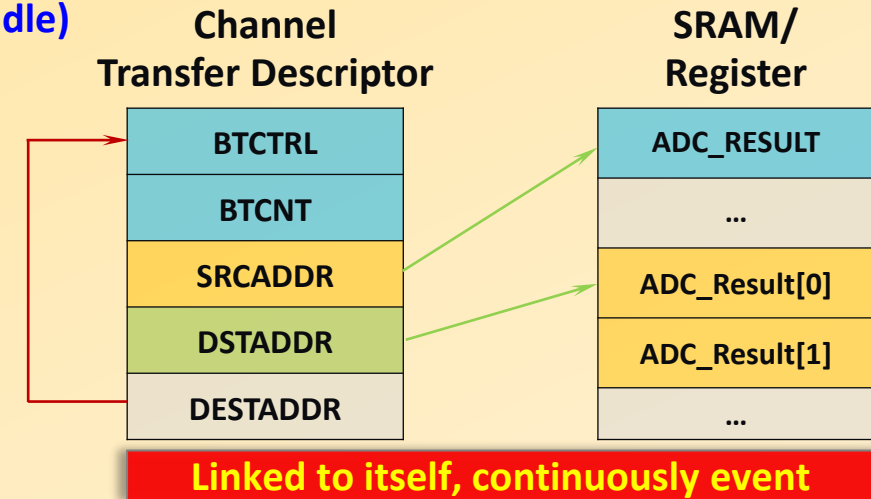
Linked List Mode C Code Example

Code Example

```
void DMA_Complete(DMAC_TRANSFER_EVENT event, uintptr_t contextHandle)
{
    if (event == DMAC_TRANSFER_EVENT_COMPLETE)
    {
        DMA_IsCompleted = 1;
    }
}

dmac_descriptor_registers_t DMAC_Descriptor_ADC_SRAM;

int main (void)
{
    SYS_Initialize ( NULL );
    ...
    DMAC_ChannelCallbackRegister(DMAC_CHANNEL_0, DMA_Complete, (uintptr_t) NULL);
    DMAC_LinkedListDescriptorSetup(&(DMAC_Descriptor_ADC_SRAM),
        DMAC_BTCTRL_STEPSEL(0) | DMAC_BTCTRL_DSTINC(1) | DMAC_BTCTRL_SRCINC(0) |
        DMAC_BTCTRL_BEATSIZE_HWORD | DMAC_BTCTRL_BLOCKACT_INT | DMAC_BTCTRL_VALID(1),
        (const void *) &(ADC_REGS->ADC_RESULT),
        (const void *) ADC_Result,
        sizeof (ADC_Result),
        &(DMAC_Descriptor_ADC_SRAM));
    DMAC_ChannelLinkedListTransfer(DMAC_CHANNEL_0, &(DMAC_Descriptor_ADC_SRAM));
    while( true )
    {...}
}
```



✂ Lab13-5 ADC with Descriptors Link of DMAC

- Please continue from [SAM2001 Lab13-3](#) (Lab13-3 ADC with DMAC)
- Try to use Linked List Mode to drive DMAC to moving `ADC_RESULT` _(Register) to `ADC_Result[]` _(SRAM) when `ADC_RESRDY`.
- Set and define customer transfer descriptor based on linked list mode.
 - DMAC Channel : 0
 - Trigger Source : `ADC_RESRDY`
 - Trigger Action : One Beat Transfer
 - Source / Destination Address Mode : Fixed / Increment
 - Beat Size : 16-bits
 - Source / Destination Register or memory : `ADC_RESULT` _(Register) / `ADC_Result` _(SRAM)
 - Descriptor Pointer : Pointer itself. (infinite linked list)
- **Let's go !**

⚡ Lab13-5 ADC with Descriptors Link of DMAC

Step 1

- Enable Linked List Mode in DMA Settings.

The screenshot shows the DMA Settings window with the following configuration for DMAC Channel 0:

- Active Channels List:** Channel Number: DMAC Channel 0, Trigger: ADC_RESRDY.
- DMAC Channel 0 Settings:**
 - Use Linked List Mode:** ☒ (Annotated with a yellow box and arrow: "Use Linked List Mode")
 - Enable Interrupt:** ☒ (Annotated with a yellow box: "One Beat Transfer per DMA Request")
 - Trigger Action:** One Beat Transfer per DMA Request (Annotated with a red box)
 - Source Address Mode:** Fixed Address Mode (Annotated with a red box and crossed out with a red line)
 - Destination Address Mode:** Increment Address After Every Transfer (Annotated with a red box and crossed out with a red line)
 - Beat Size:** 16-bit bus transfer (Annotated with a red box and crossed out with a red line)

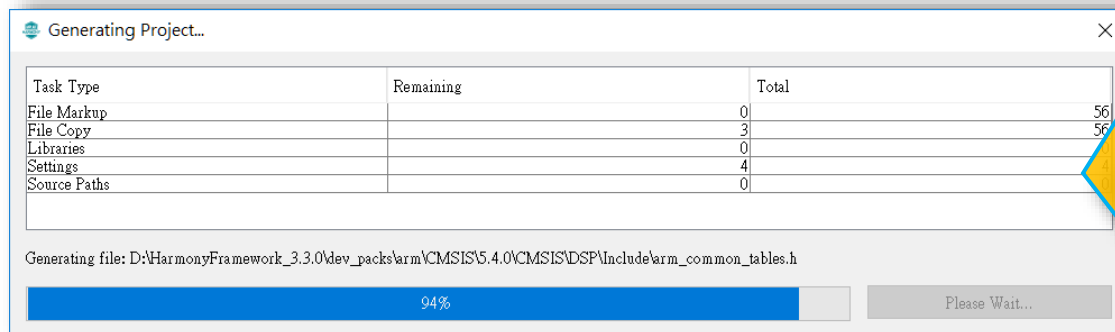
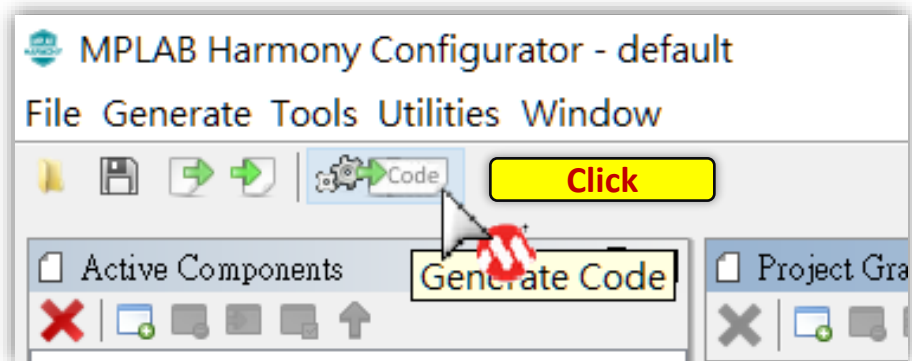
Buttons at the bottom: Add Channel, Remove Selected Channel.

Annotation at the bottom: "Don't care, Setting by Code" (Annotated with a yellow box)

⚙️ Lab13-5 ADC with Descriptors Link of DMAC

Step 2

- Click  to Generate Code.



⚙️ Lab13-5 ADC with Descriptors Link of DMAC

Step 3

```
void DMA_Complete(DMAC_TRANSFER_EVENT event, uintptr_t contextHandle)
{
    if (event == DMAC_TRANSFER_EVENT_COMPLETE)
    {
        DMA_IsCompleted = 1;

        // TODO 13-5.01
        //      DMAC_ChannelTransfer(DMAC_CHANNEL_0,
        //                          (const void *) &(ADC_REGS->ADC_RESULT),
        //                          (const void *) ADC_Result,
        //                          sizeof (ADC_Result));
    }
}

int main (void)
{
    DMAC_ChannelCallbackRegister(DMAC_CHANNEL_0, DMA_Complete, (uintptr_t) NULL);
    // TODO 13-5.02
    //      DMAC_ChannelTransfer(DMAC_CHANNEL_0,
    //                          (const void *) &(ADC_REGS->ADC_RESULT),
    //                          (const void *) ADC_Result,
    //                          sizeof (ADC_Result));

    return ( EXIT_FAILURE);
}
```

⚙️ Lab13-5 ADC with Descriptors Link of DMAC

Step 4

```
// TODO 13-5.03
dmac_descriptor_registers_t DMAC_Descriptor_ADC_SRAM;
int main (void)
{
    SYS_Initialize(NULL);
    ...
    // TODO 13-5.04
    DMAC_LinkedListDescriptorSetup(
        &(DMAC_Descriptor_ADC_SRAM),
        DMAC_BTCTRL_STEPSEL(0) | DMAC_BTCTRL_DSTINC(1) | DMAC_BTCTRL_SRCINC(0) |
        DMAC_BTCTRL_BEATSIZE_HWORD | DMAC_BTCTRL_BLOCKACT_INT | DMAC_BTCTRL_VALID(1),
        (const void *) &(ADC_REGS->ADC_RESULT),
        (const void *) ADC_Result,
        sizeof (ADC_Result),
        &(DMAC_Descriptor_ADC_SRAM));

    // TODO 13-5.05
    DMAC_ChannelLinkedListTransfer
        (DMAC_CHANNEL_0, &(DMAC_Descriptor_ADC_SRAM));

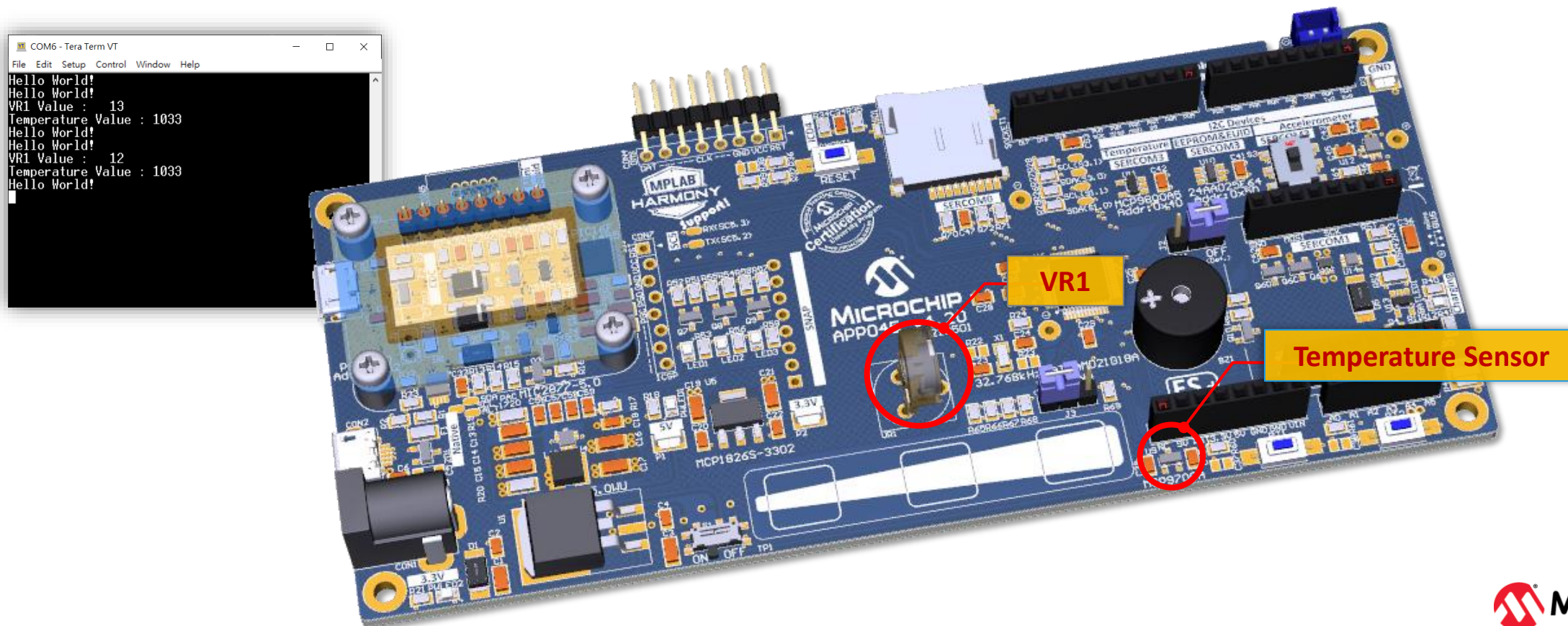
    while ( true )
    {
        ...
    }

    return ( EXIT_FAILURE);
}
```

⚙️ Lab13-5 ADC with Descriptors Link of DMAC

Result

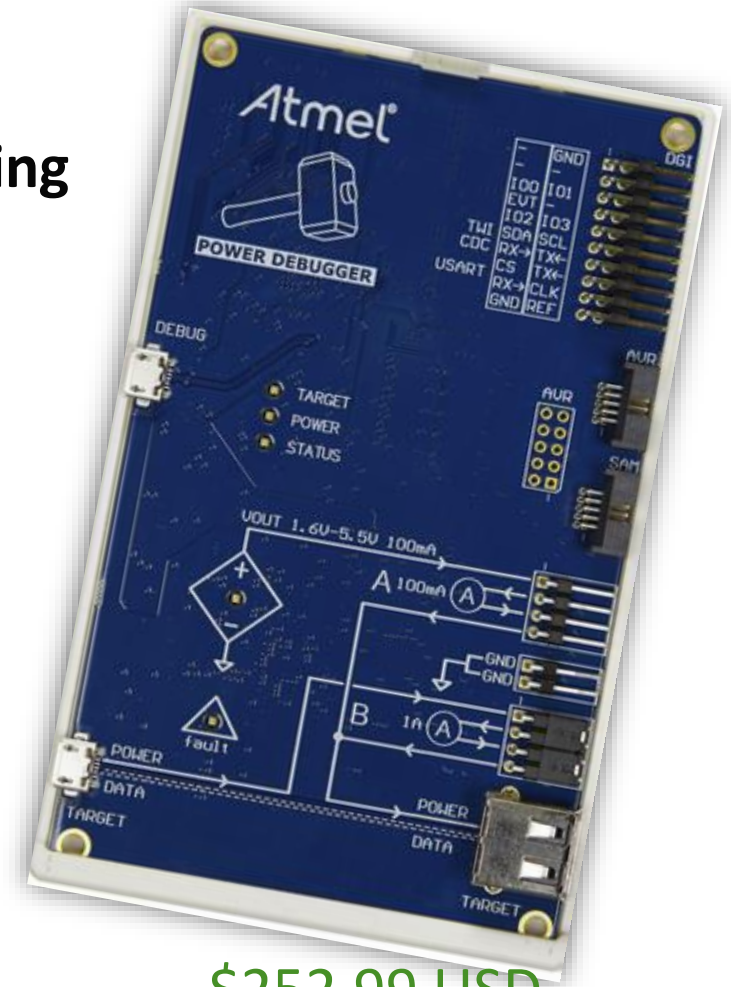
- “Hello World!!” string and ADC result of VR1 / Temperature Sensor will output to terminal with two second period. Same as Lab 13, but data moving by DMAC linked list mode now. And DMAC is continuously active don't need reinitialized.



Power Debugger

Power Debugger

- Power Debugger is a powerful development tool for ARM® Cortex®-M based SAM and AVR microcontrollers using JTAG, SWD, PDI, debugWIRE, aWire, TPI or SPI interfaces.
- In addition, Two independent current sensing channels for measuring and optimizing the power consumption of a design.
 - Channel A for high accuracy, lower currents
 - Channel B for higher currents with lower resolution
- A CDC virtual COM port interface
- A Data Gateway Interface channels for streaming application data to the host computer from a SPI, USART, TWI or GPIO source.
- Provides target power, 1.6 – 5.5V, up to 100mA Status LEDs for debugging & target supply

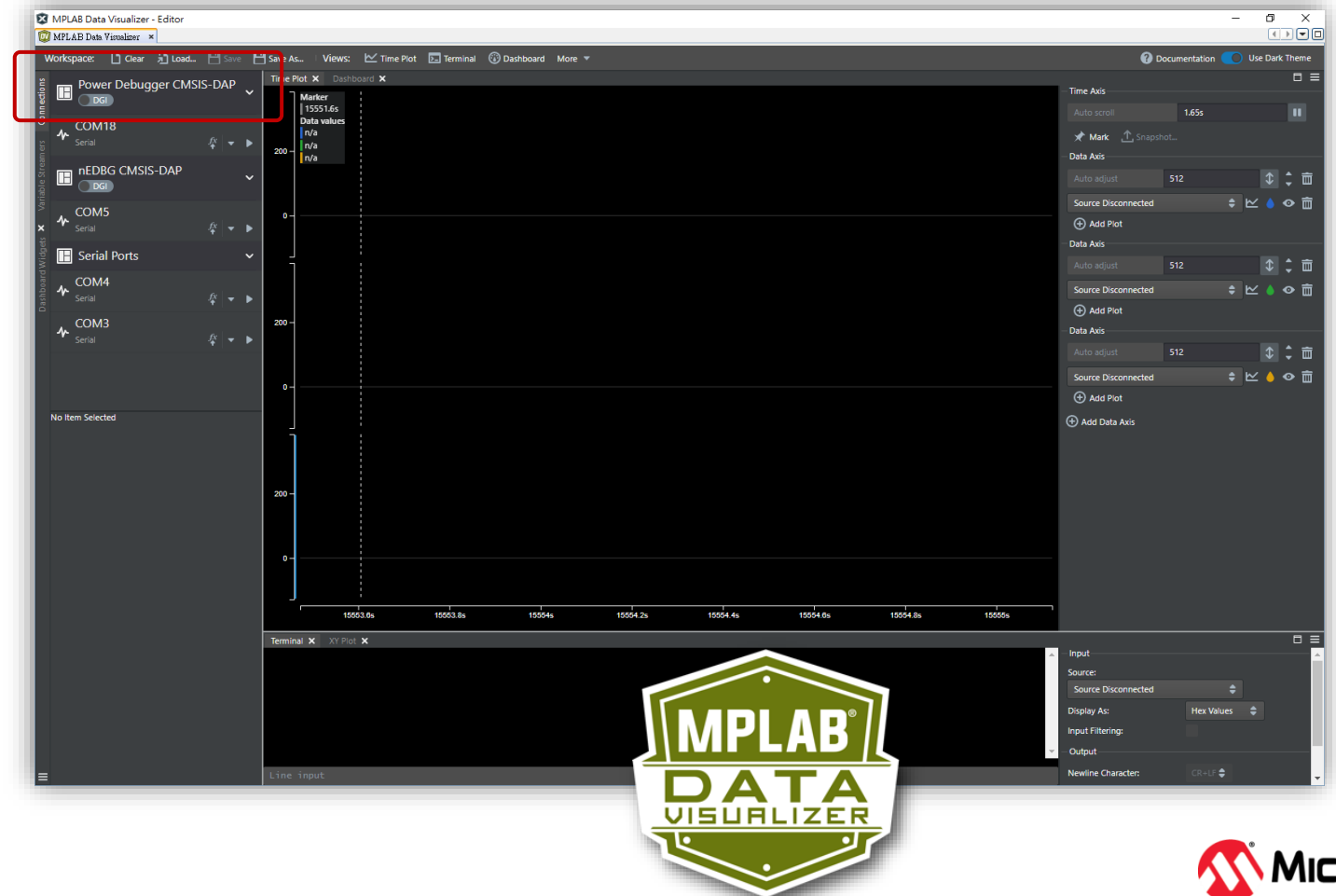


\$252.99 USD

<https://www.microchipdirect.com/dev-tools/ATPOWERDEBUGGER?allDevTools=true>

Power Consumption Measurement

- We talk about MPLAB Data Visualizer at above slices.
- MPLAB Data Visualizer support Power Debugger for power consumption Measurement based on DGI (Data gateway Interface), also.
- We will use Power Debugger help us to measurement the power consumption at differ power mode at below exercises.



Sleep Mode

Power Manager (PM)

- The synchronous system clock control by Power Manager (PM).
It's divided into several clock domains.
one for the **CPU** and **AHB** and one for each **APB**.
- PM controls the **reset**, **clock generation** and **sleep modes** of the device, also.
- The application code decides which sleep mode to enter and when.
- **Interrupts from enabled peripherals and all enabled reset sources can restore the device from a sleep mode to active mode.**
- The PM also contains a reset controller to collect all possible reset sources. It issues a device reset and sets the device to its initial state and allows the reset source to be identified by software.

Sleep Mode - IDLE

- **IDLE Mode**
 - The CPU is stopped.
 - Optionally, some synchronous clock domains are stopped, depending on the IDLE argument.
 - Regulator operates in normal mode.

Mode	Level	Mode Entry	Wake-Up Sources
IDLE	0	SCR.SLEEPDEEP = 0 SLEEP.IDLE=Level WFI	Synchronous ⁽²⁾ (APB, AHB), asynchronous ⁽¹⁾
	1		Synchronous (APB), asynchronous
	2		Asynchronous

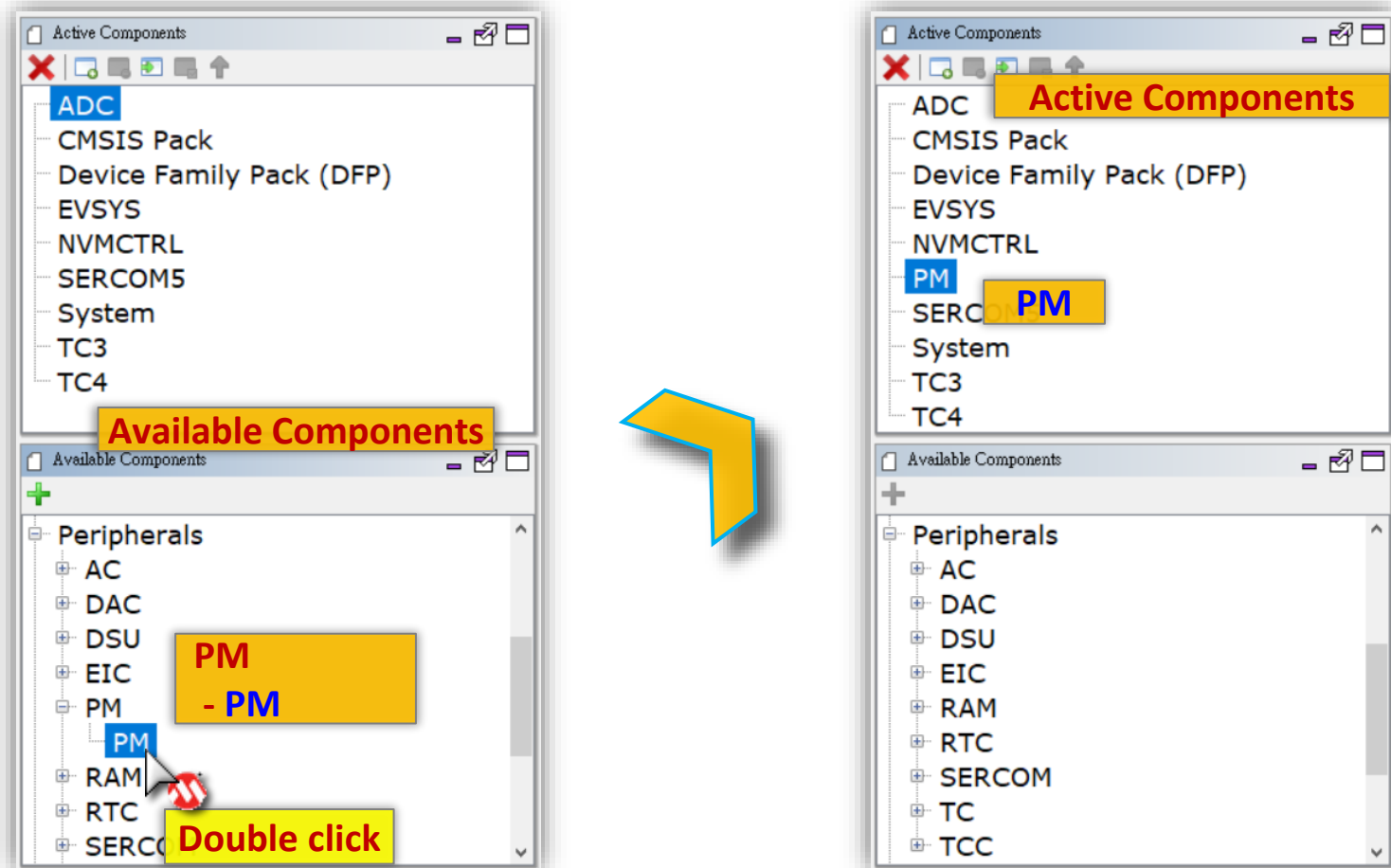
Sleep Mode - STANDBY

- **STANDBY Mode**
 - All clock sources, clock generation and most peripheral are stopped, except those where the **RUNSTDBY** bit is set.
 - Regulator operates in low-power mode.

Mode	Level	Mode Entry	Wake-Up Sources
STANDBY		SCR.SLEEPDEEP = 1 WFI	Asynchronous

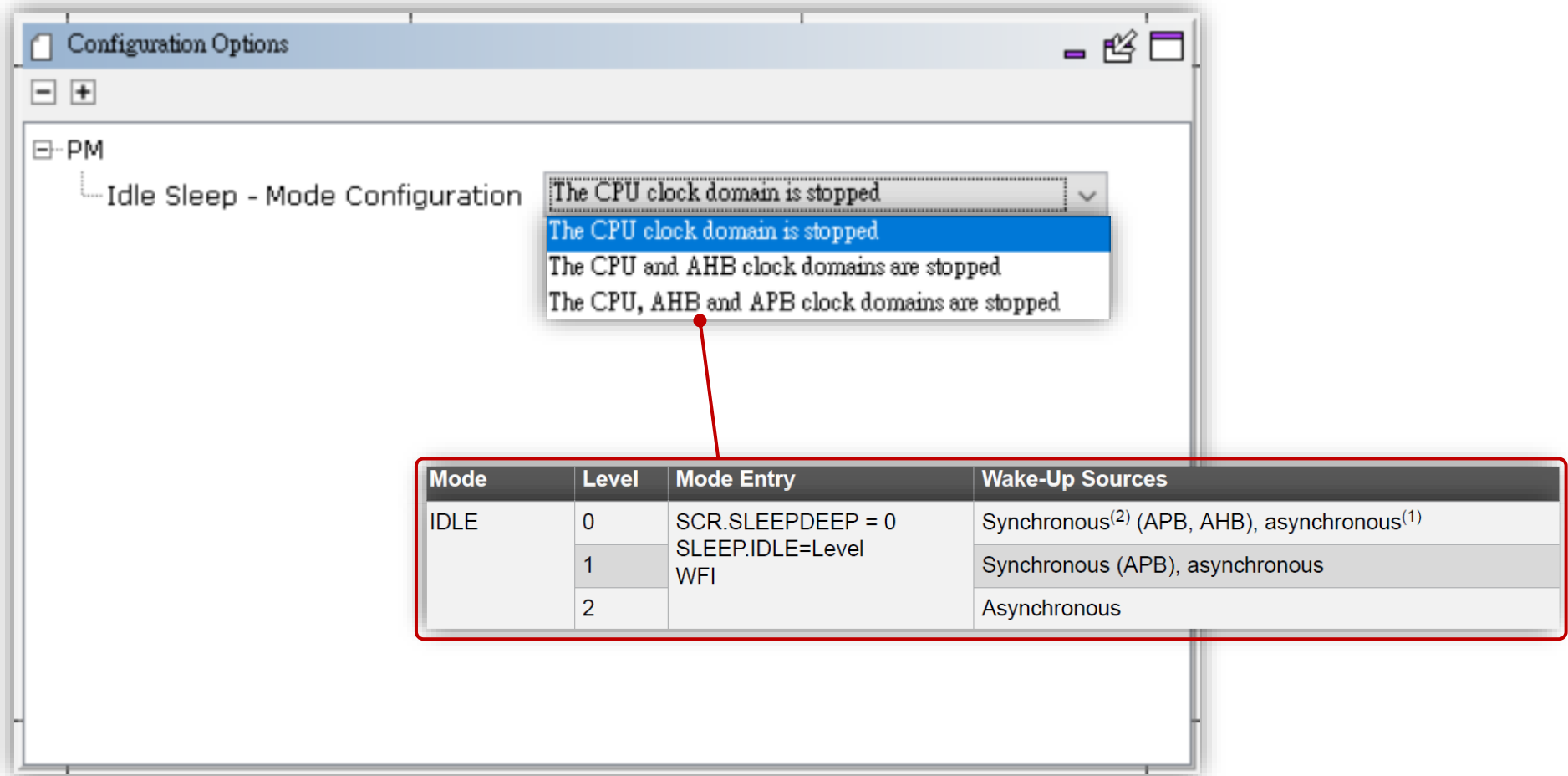
Add PM Function using MHC

- Find **PM** component in Available Components window.
- Double click **PM** to add to Active Components window.



PM Function

Configuration Options Example



The screenshot shows the 'Configuration Options' dialog box with the 'PM' section expanded. Under 'Idle Sleep - Mode Configuration', a dropdown menu is open, displaying four options. A red line points from the third option, 'The CPU, AHB and APB clock domains are stopped', to a table below.

Mode	Level	Mode Entry	Wake-Up Sources
IDLE	0	SCR.SLEEPDEEP = 0	Synchronous ⁽²⁾ (APB, AHB), asynchronous ⁽¹⁾
	1	SLEEP.IDLE=Level	Synchronous (APB), asynchronous
	2	WFI	Asynchronous

PM Functions & Code Example

Interface Routines & Example

void PM_IdleModeEnter(**void**);

void PM_StandbyModeEnter(**void**);

PM_RESET_CAUSE PM_ResetCauseGet(**void**);

```
int main (void)
{
    SYS_Initialize ( NULL );
    ...
    while( true )
    {
        PM_IdleModeEnter( );
        ...
    }
}
```

Enter Idle mode after function call

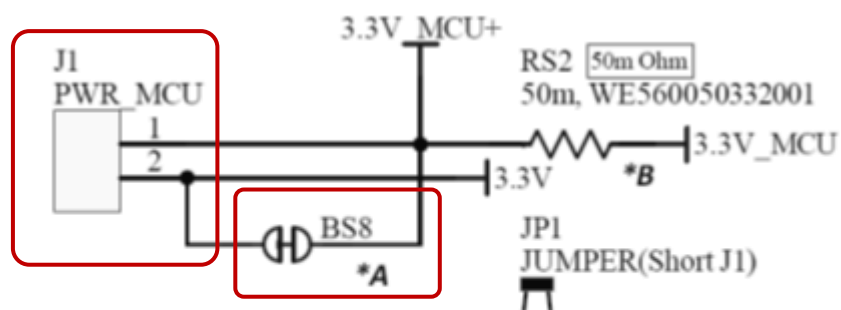
Lab13-6 ADC with IDLE

- Please continue from [SAM2001 Lab13 \(ADC PinScan Interrupt Callback SWTrigger\)](#)
- Try to let CPU enter IDLE0 mode, when nothing to do.
- Use power debugger or external current monitor to tracking power consumption at running time.
 - IDLE0 : The CPU domain is stopped

• **Let's go !**

Current Monitor Function at APP045 v4

- APP045 provide current monitor pin for MCU. Solder pin header at J1 and cut off BS8 to enable it.



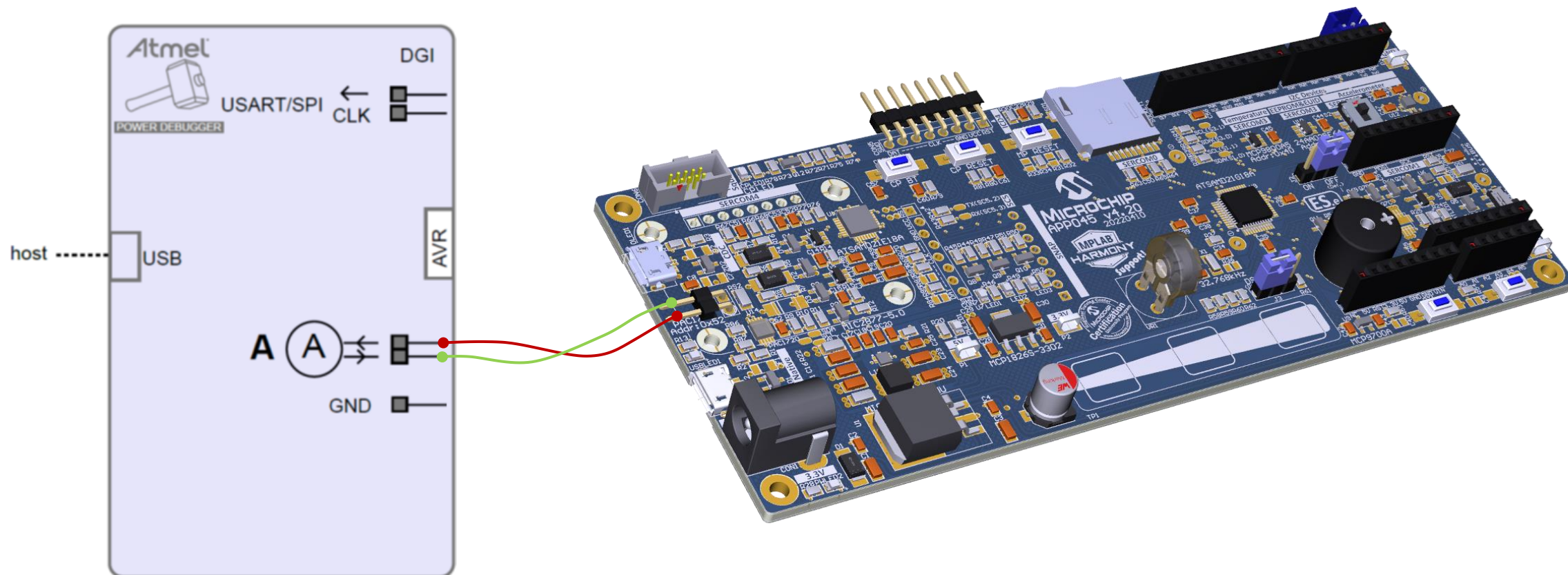
MCU Power with Monitor

*A PCB short by default



Monitor By Power Debugger

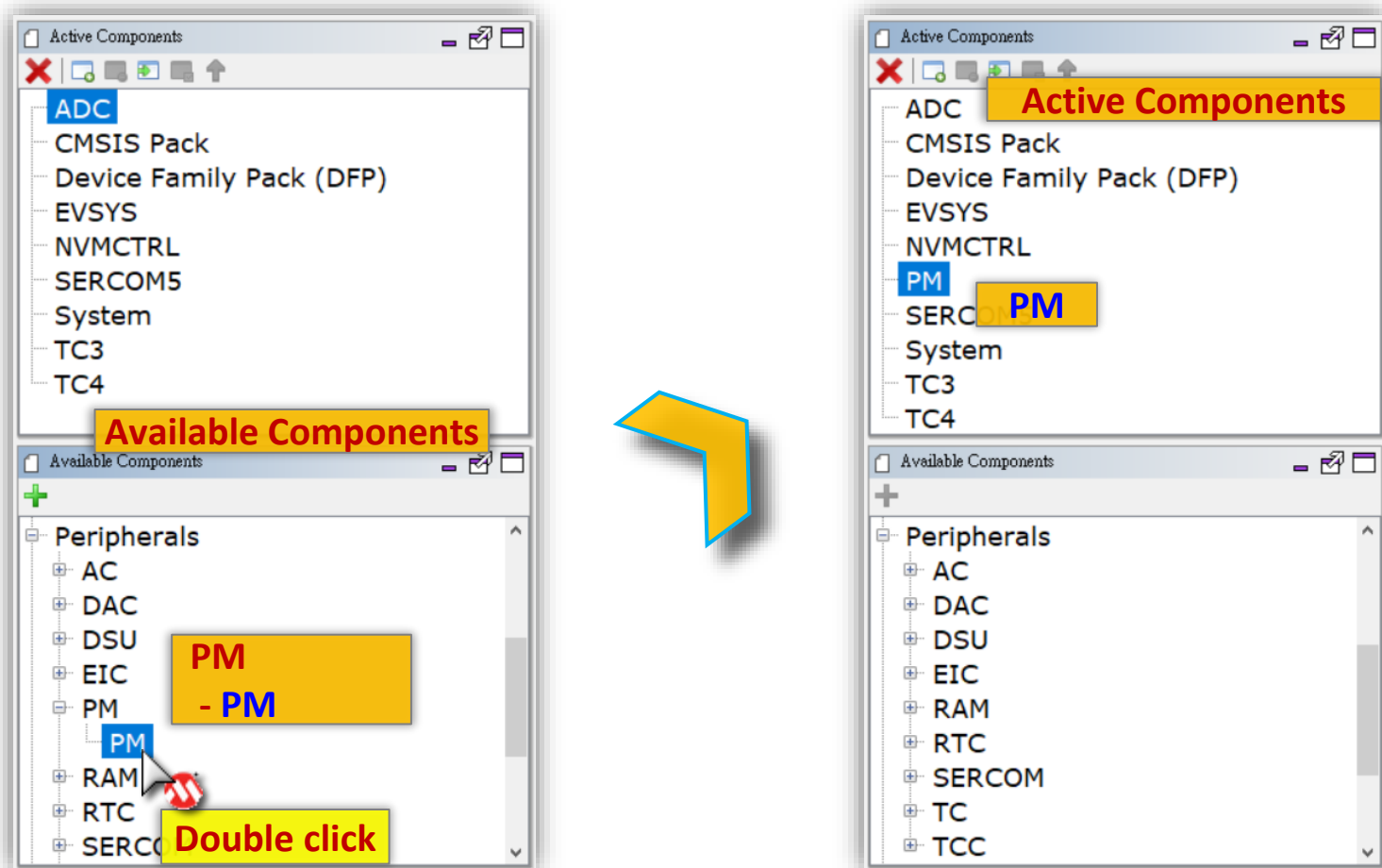
- Connect MCU power monitor pins at APP045 to current monitor port (Channel A, 100mA) of Power Debugger.



⚙️ Lab13-6 ADC with IDLE

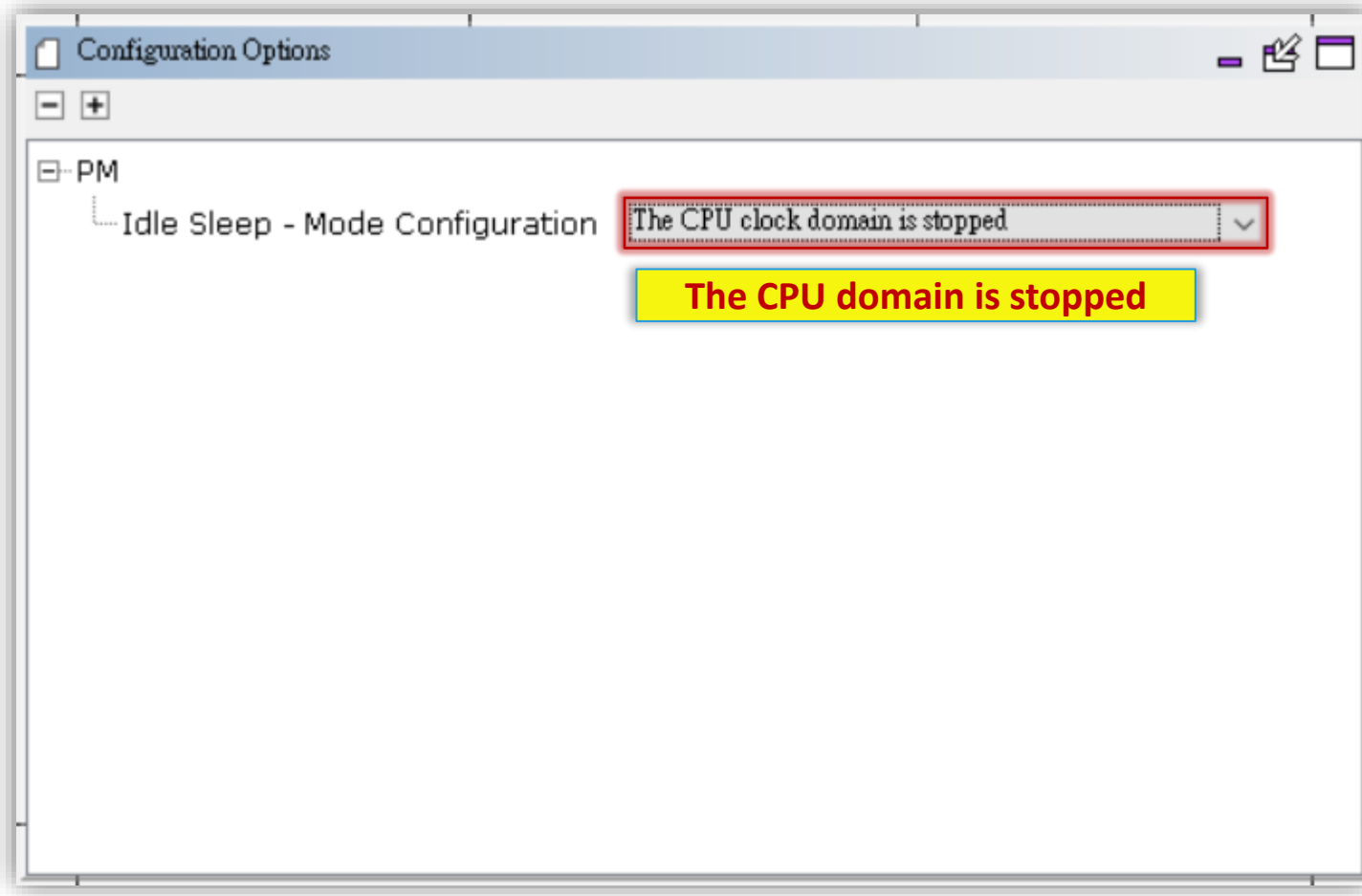
Step 1

- Find **PM** component in Available Components window.
- Double click **PM** to add to Active Components window.



⚙️ Lab13-6 ADC with IDLE

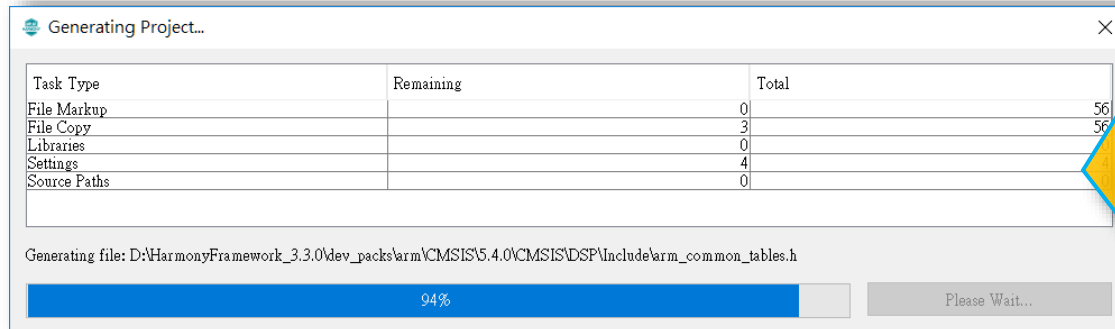
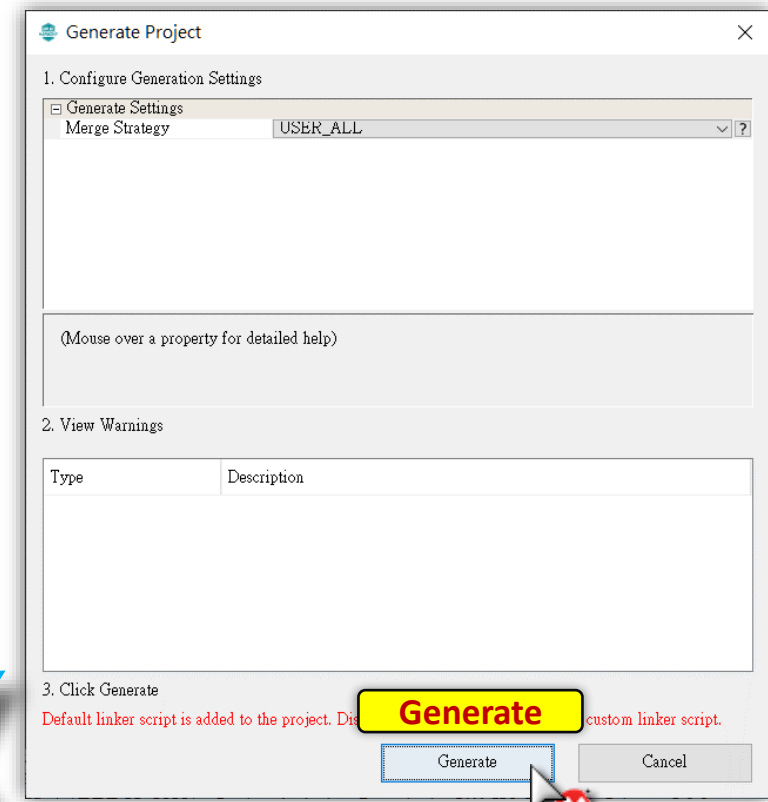
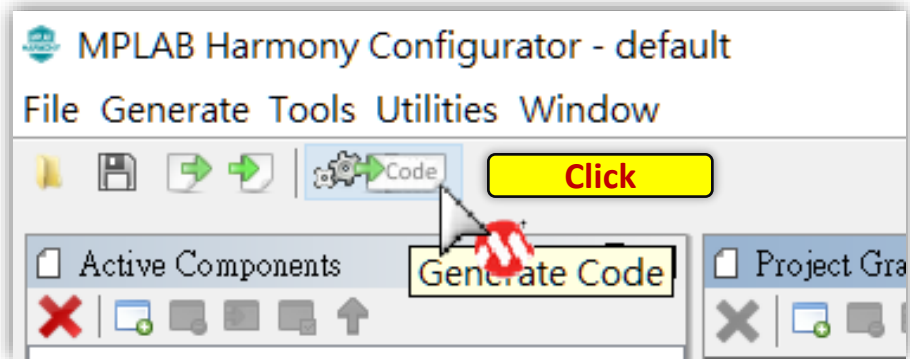
Step 2



⚙️ Lab13-6 ADC with IDLE

Step 3

- Click  to Generate Code.



Lab13-6 ADC with IDLE

Step 4

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

    while ( true )
    {
        SYS_Tasks();

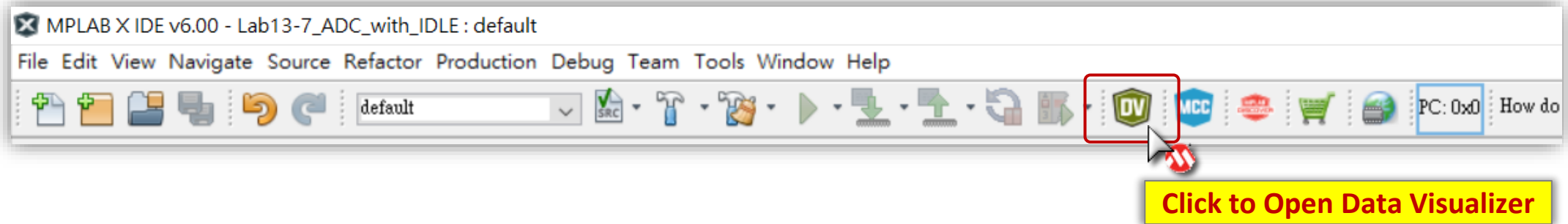
        // TODO 13-7.01
        PM_IdleModeEnter();
        ...
    }

    return ( EXIT_FAILURE);
}
```

⚙️ Lab13-6 ADC with IDLE

Step 5

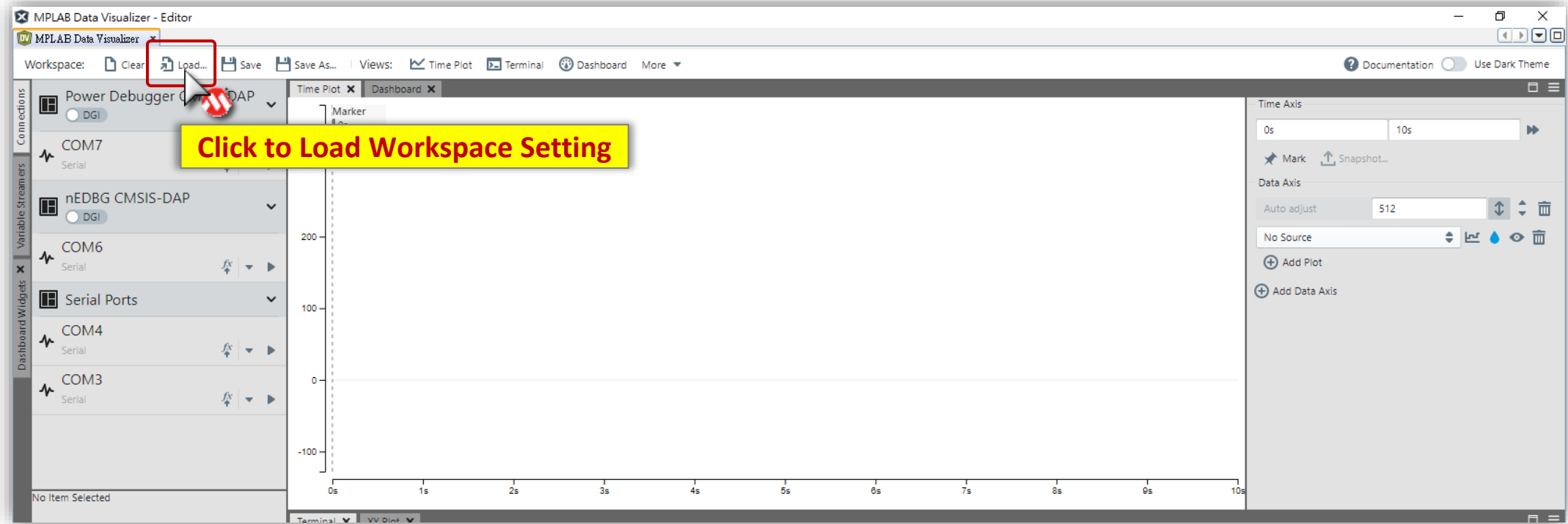
- Open Data Visualizer by icon at MPLAB X IDE



⚙️ Lab13-6 ADC with IDLE

Step 6

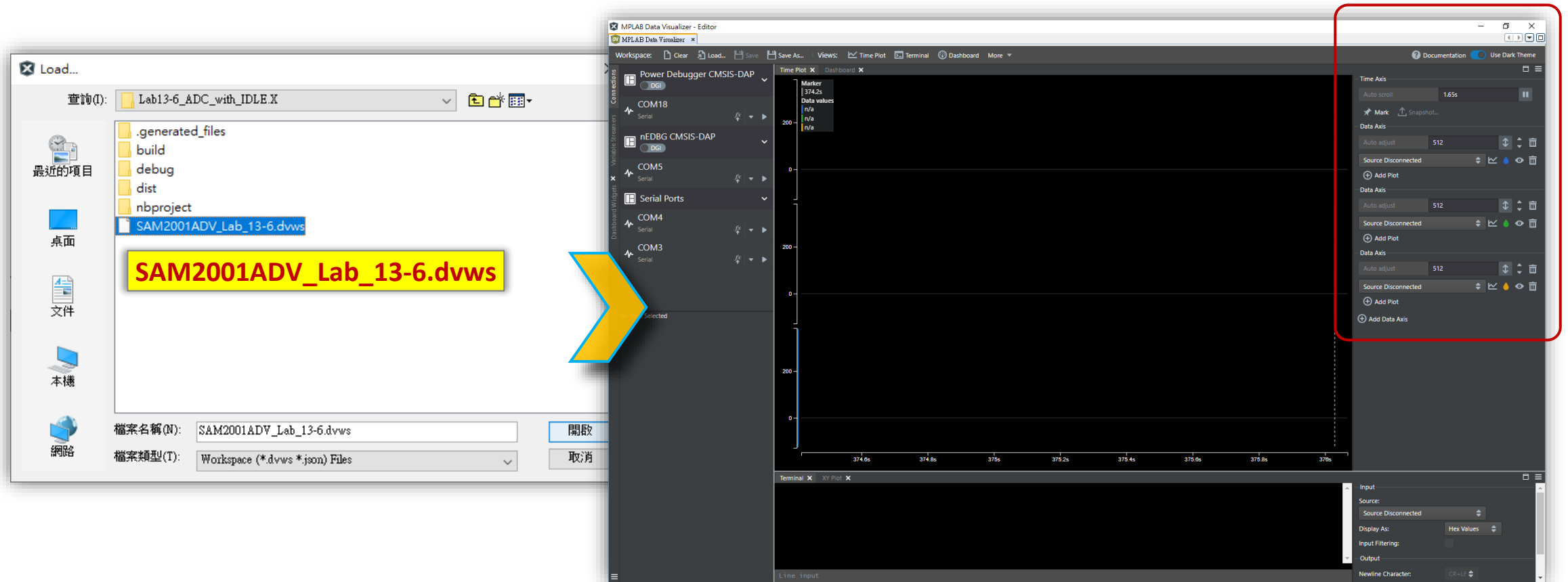
- Load layout file to reduce setting steps.



⚙️ Lab13-6 ADC with IDLE

Step 7

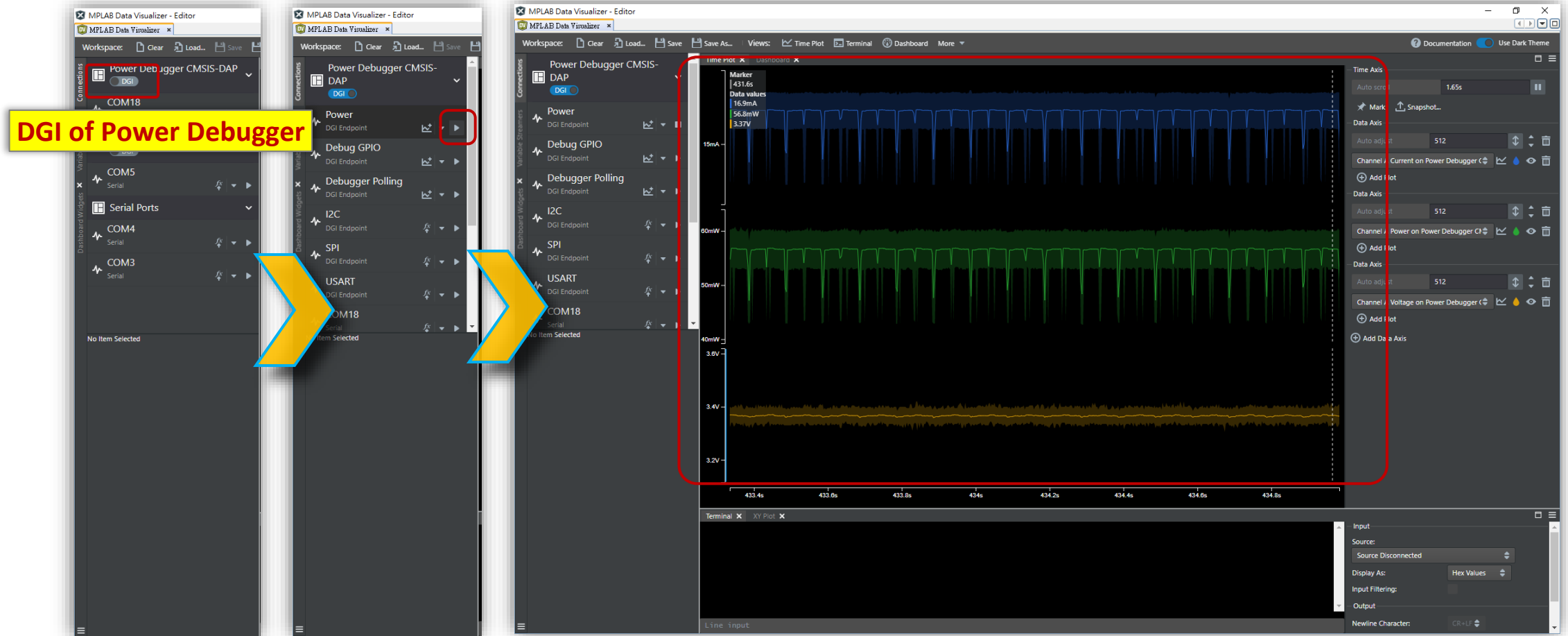
- Select “**SAM2001ADV_Lab_13-6.dvws**” layout file. Then data axis be set. Current, Power and Voltage.



⚙️ Lab13-6 ADC with IDLE

Step 8

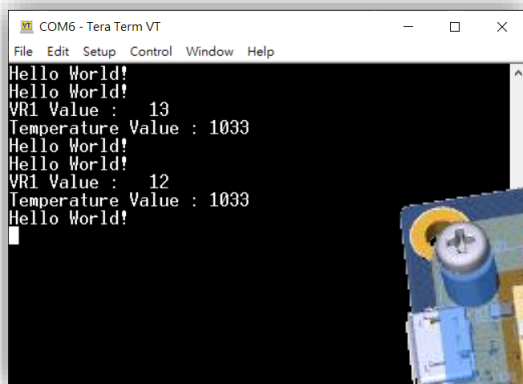
- Enable **Power** of **DGI** left side at Data Visualizer.



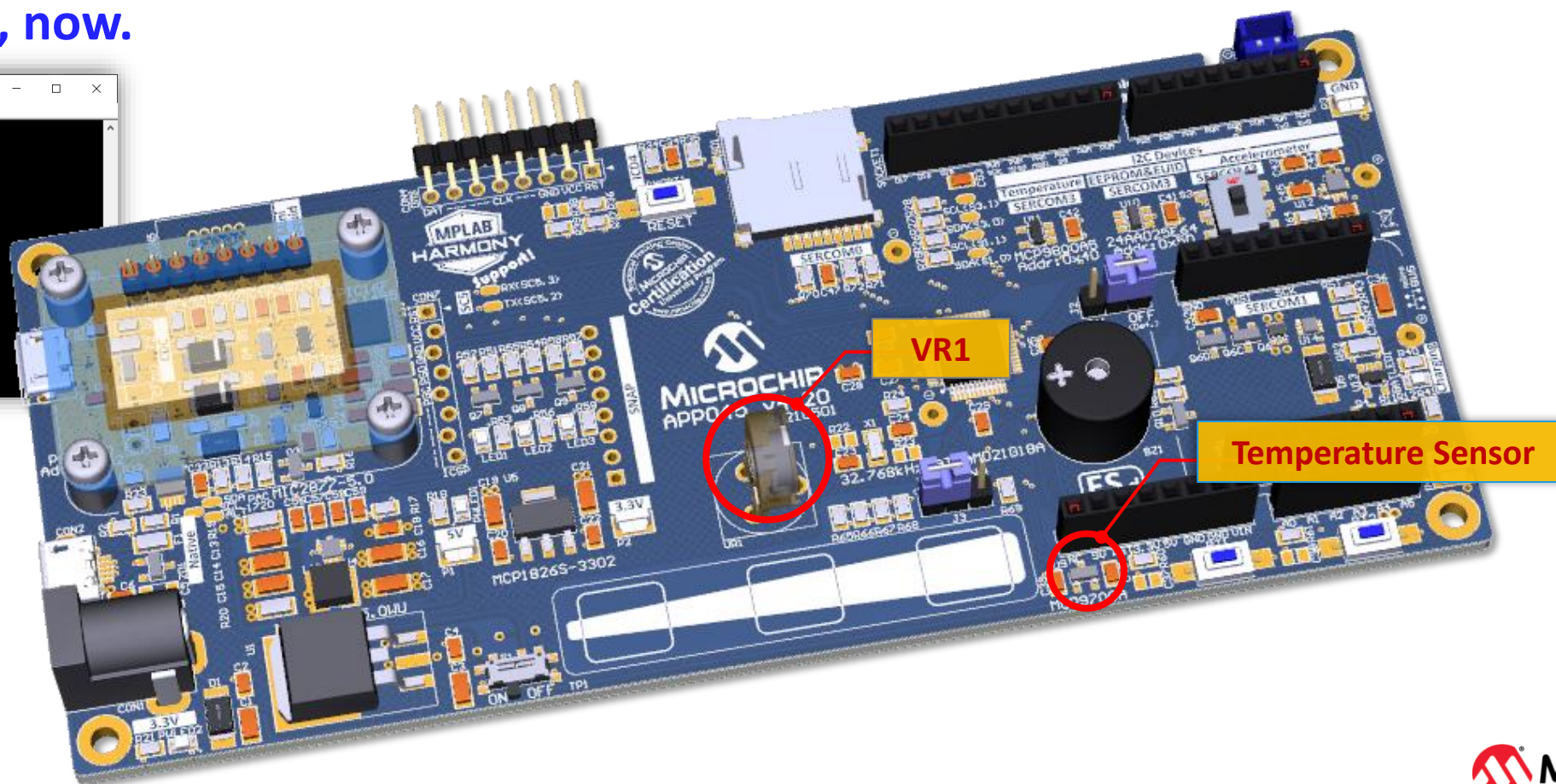
⚡ Lab13-6 ADC with IDLE

Result

- “Hello World!!” string and ADC result of VR1 / Temperature Sensor will output to terminal with two second period. Same as Lab 13, but MCU enter IDLE mode when nothing to do, now.



```
COM6 - Tera Term VT
File Edit Setup Control Window Help
Hello World!
Hello World!
VR1 Value : 13
Temperature Value : 1033
Hello World!
Hello World!
VR1 Value : 12
Temperature Value : 1033
Hello World!
```

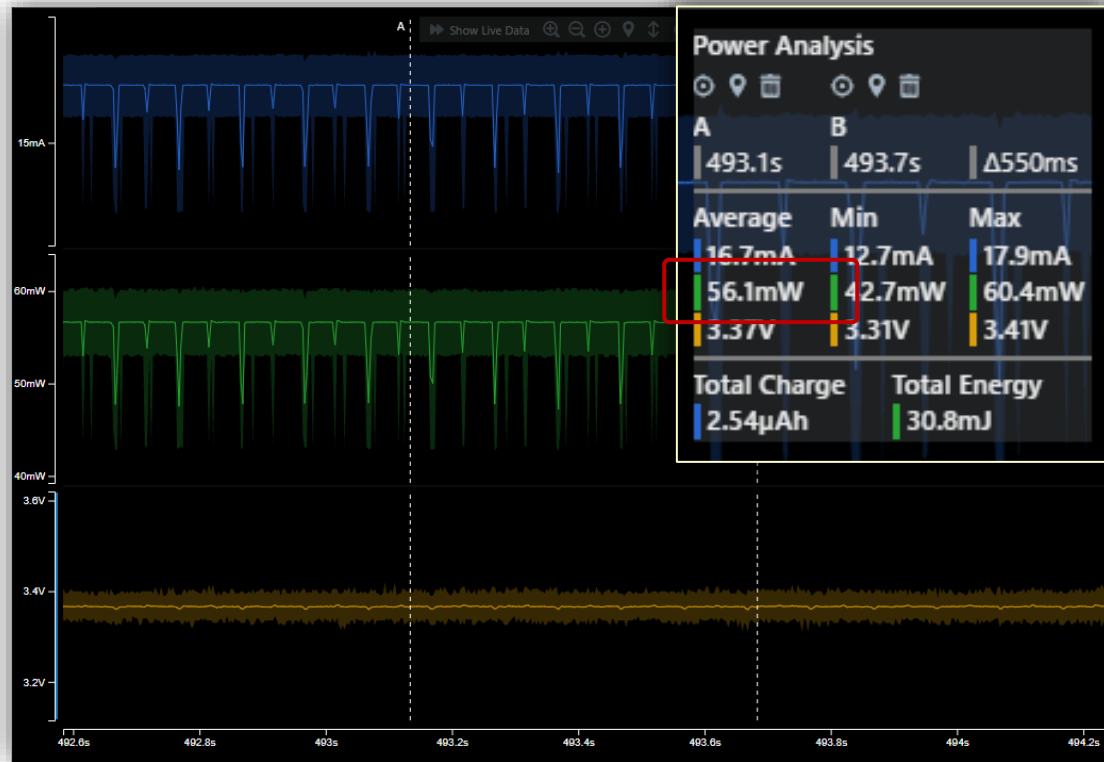


⚙️ Lab13-6 ADC with IDLE

Result

- Power consumption compared for Active and IDLE0 mode.

Active Mode



IDLE0 Sleep Mode



Lab13-6 ADC with IDLE

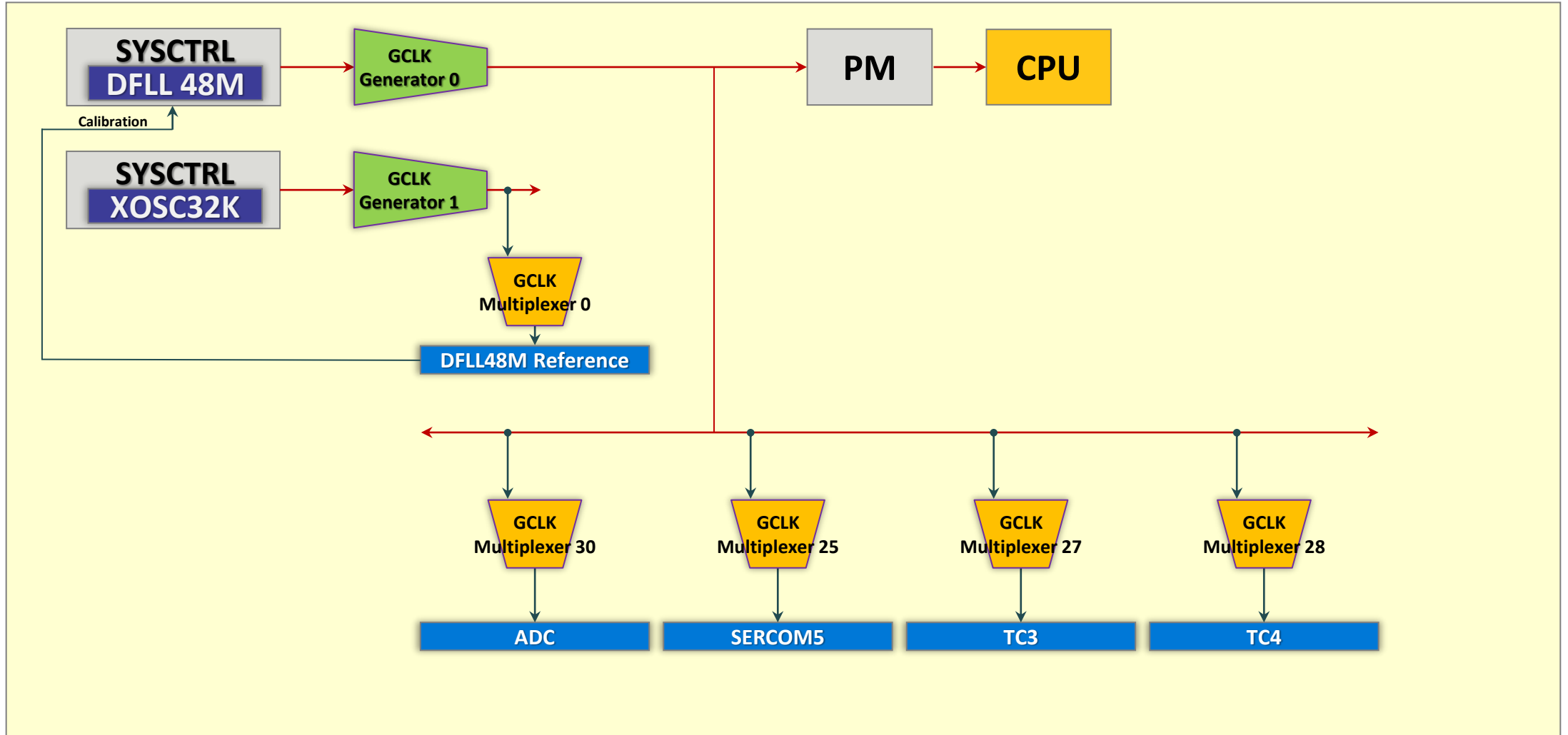
- Discuss
 - IDLE mode is easy to enter / leave, how about STANDBY ?
 - What challenge will be facing, when enter / leave STANDBY mode ?
- Remember that ?
 - “All clock sources, clock generation and most peripheral are stopped at standby mode, except those where the RUNSTDBY bit is set”
- And DFLL(DFLL48M) are disable at standby mode, also.
- So, an alternative clock source and GCLK must be set for peripheral you want to keep running at STANDBY mode.

✂ Lab13-7 ADC with STANDBY

- Please continue from [SAM2001 Lab13-6 \(ADC with IDLE\)](#)
 - Try to let CPU enter STANDBY mode, when nothing to do.
 - Use power debugger or external current monitor to tracking power consumption at running time.
 - STANDBY : All clock sources, clock generation and most peripheral are stopped.
 - OSC8M : Enable, Running STANDBY, Connect to GCLK2
 - GCLK2 : Enable, Running STANDBY
 - TC3/TC4/SEROM5/ADC : Running STANDBY, Connect to GCLK2
-
- **Let's go !**

⚙️ Lab13-7 ADC with STANDBY

- Before Lab13-7, check the block diagram of clock for Lab 13-6



Lab13-7 ADC with STANDBY

Step 1

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

    while ( true )
    {
        SYS_Tasks();

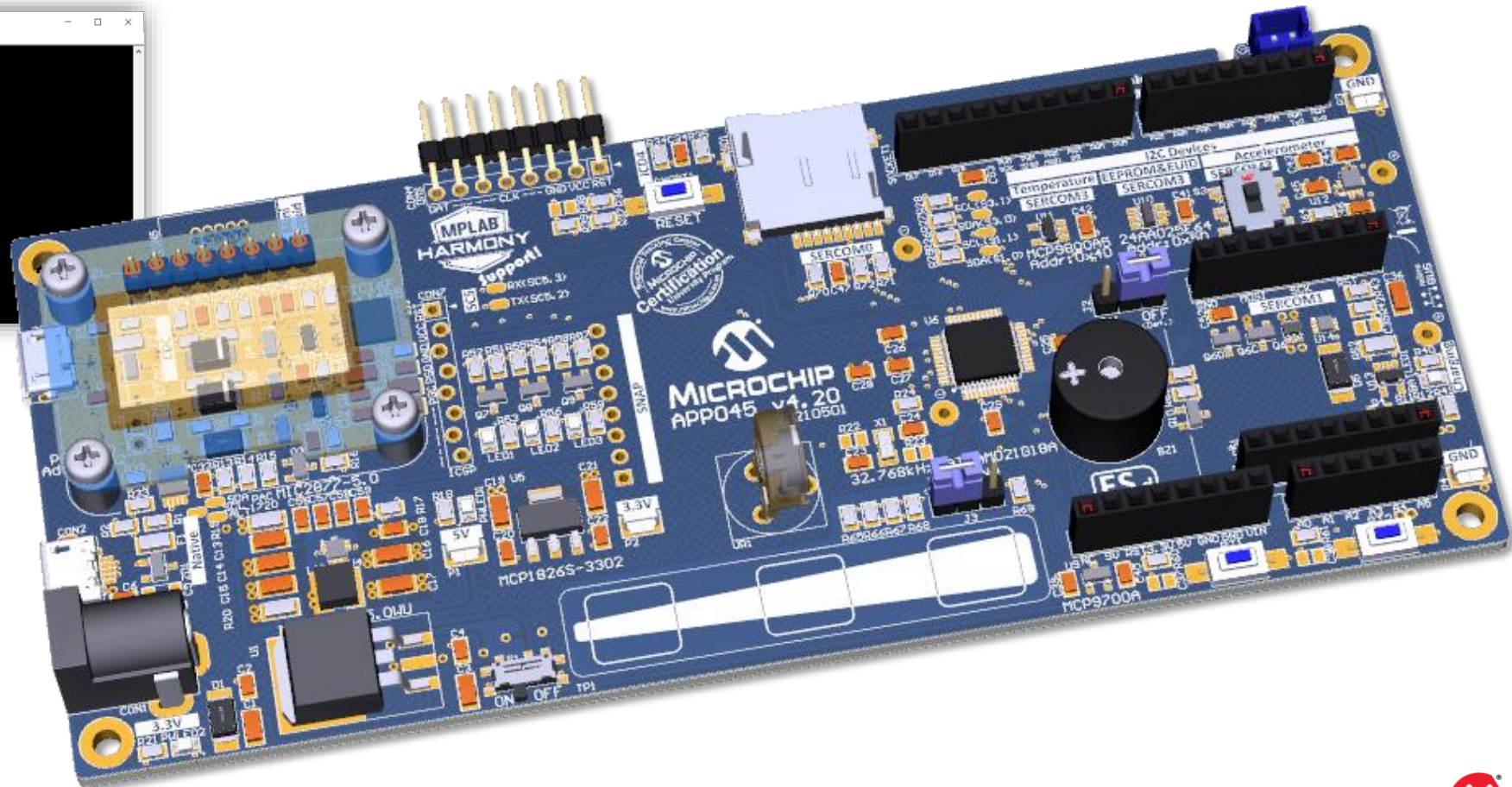
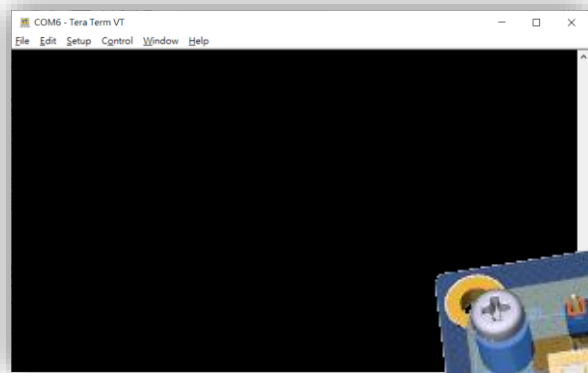
        // TODO 13-8.01
        //PM_IdleModeEnter();
        PM_StandbyModeEnter();
        ...
    }

    return ( EXIT_FAILURE);
}
```


⚡ Lab13-7 ADC with STANDBY

Result ?

- What Happened ? The board keeping in halt state, seem.

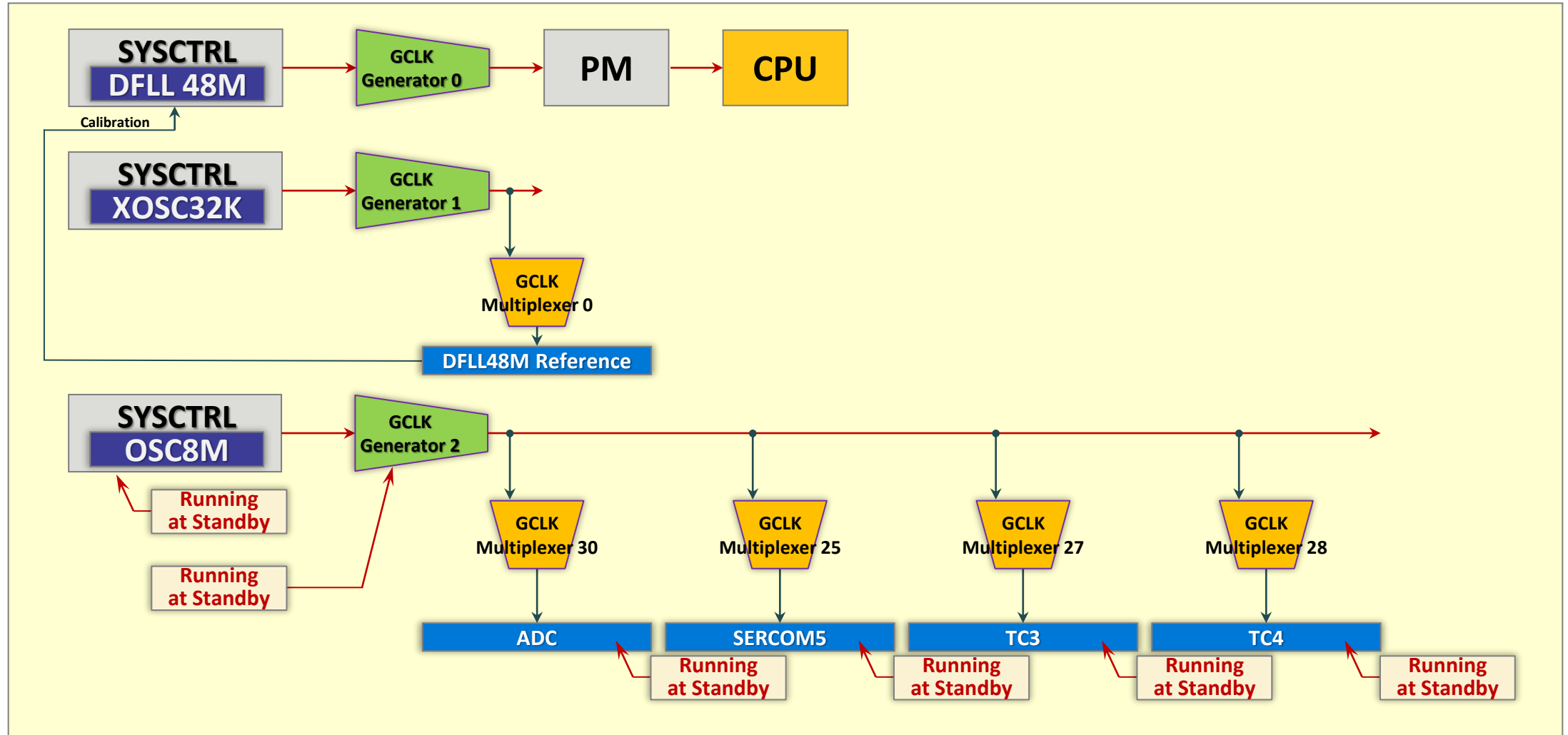


Lab13-7 ADC with STANDBY

- **Remember !**
“All clock sources, clock generation and most peripheral are stopped at standby mode, except those where the RUNSTDBY bit is set” And DFLL(DFLL48M) are disable at standby mode, also.
- So, an alternative clock source, GCLK and peripheral must be set for you want to keep running at STANDBY mode.
- AHB and APB clock are stopped at STANDBY mode.
DMAC clock source by AHB. So, the DMAC are disabled except IDLE0 mode.

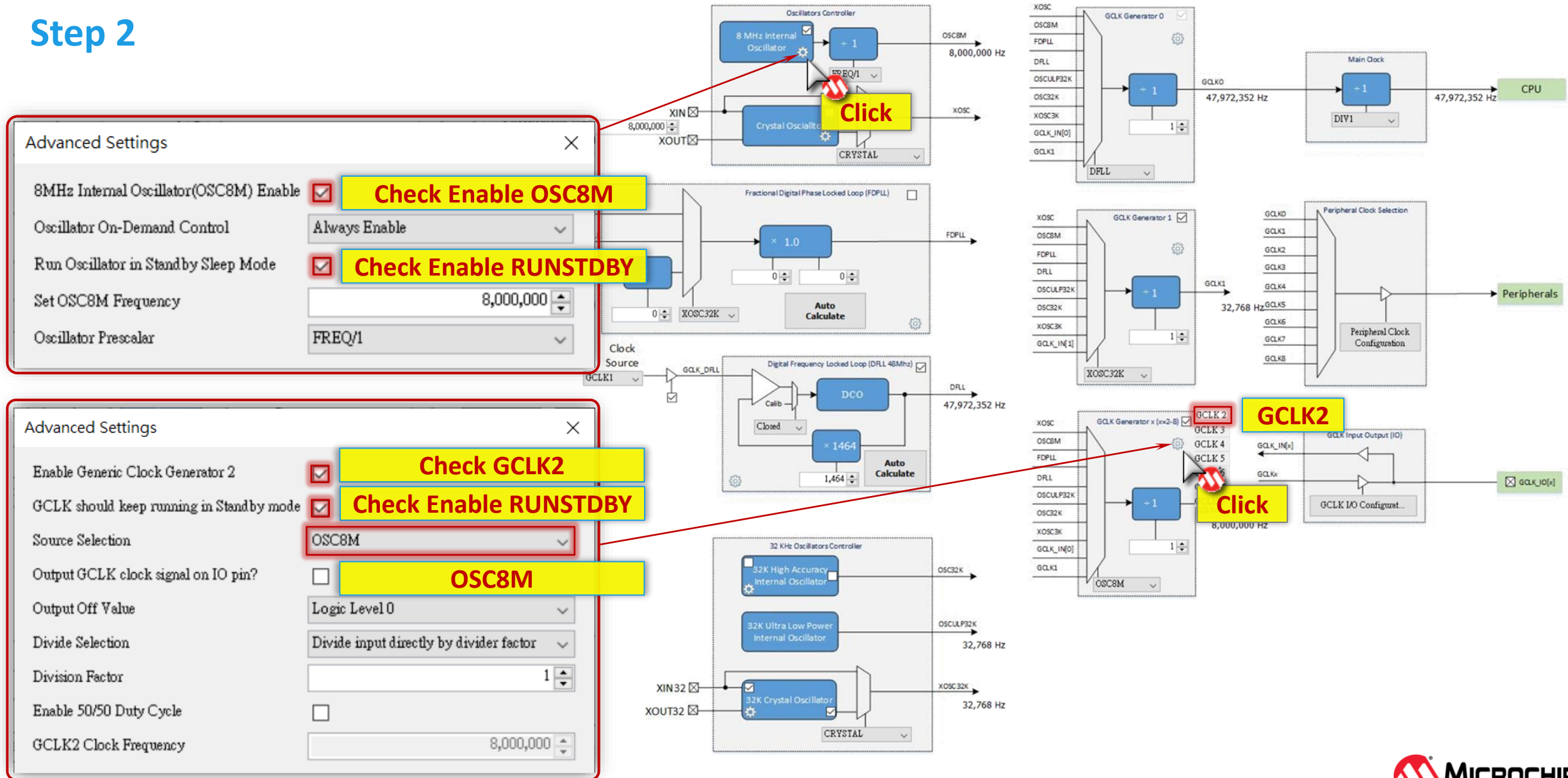
⚙️ Lab13-7 ADC with STANDBY

- Clock block diagram for Lab 13-7



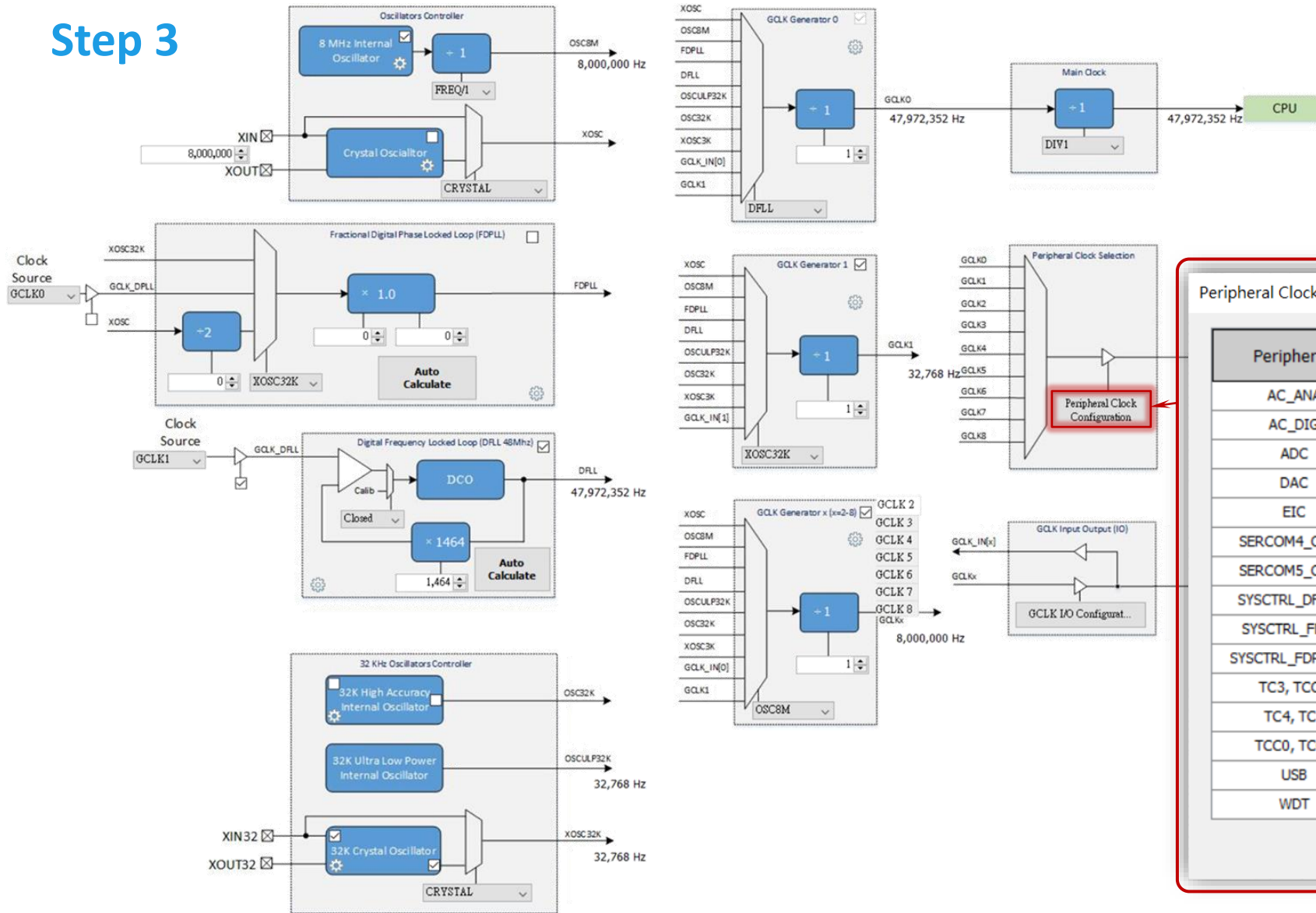
Lab13-7 ADC with STANDBY

Step 2



Lab13-7 ADC with STANDBY

Step 3



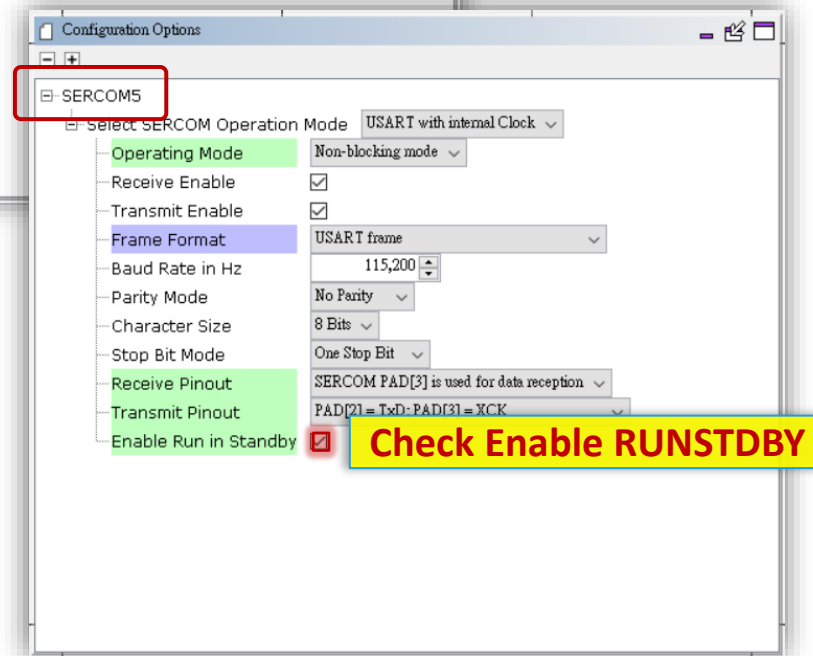
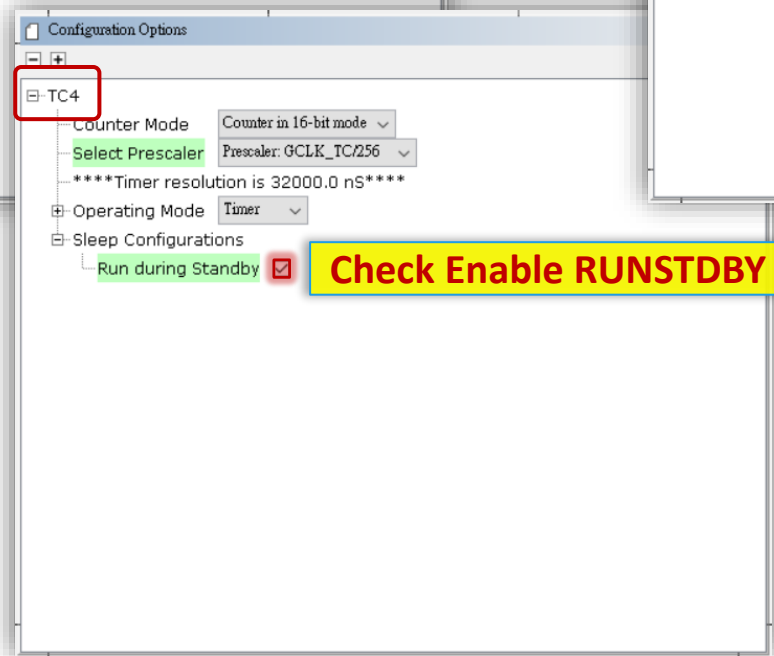
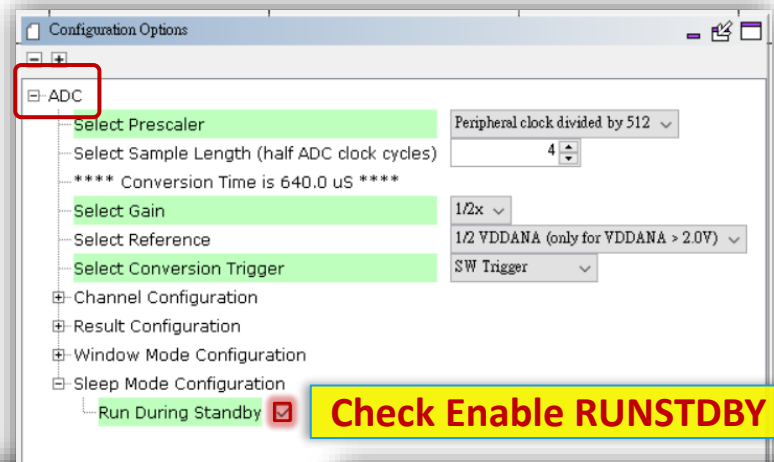
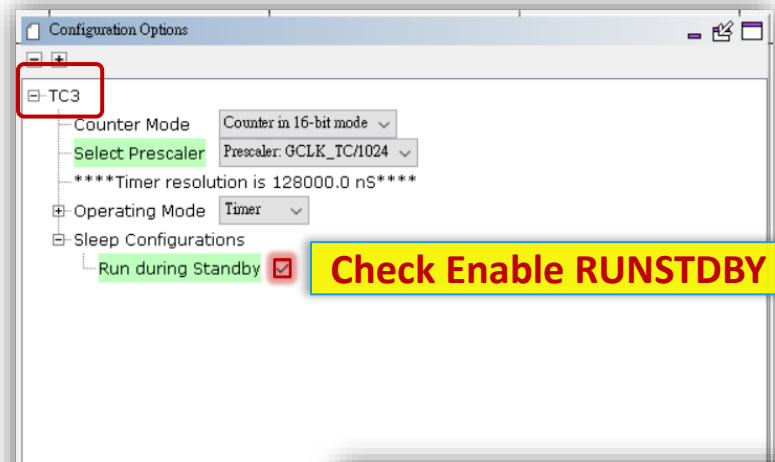
Peripheral Clock Configuration

Peripheral	Enable	Source	Peripheral Clock Frequency
AC_ANA	☐	GCLK0	--
AC_DIG	☐	GCLK0	--
ADC	☒	GCLK2	8,000,000 Hz
DAC	☐	GCLK0	--
EIC	☐	GCLK0	--
SERCOM4_CORE	☐	GCLK0	--
SERCOM5_CORE	☒	GCLK2	8,000,000 Hz
SYSCTRL_DFLL48	☒	GCLK1	32,768 Hz
SYSCTRL_FDPLL	☐	GCLK0	--
SYSCTRL_FDPLL32K	☐	GCLK0	--
TC3, TCC2	☒	GCLK2	8,000,000 Hz
TC4, TC5	☒	GCLK2	8,000,000 Hz
TCC0, TCC1	☐	GCLK0	--
USB	☐	GCLK0	--
WDT	☐	GCLK0	--

Close

⚙️ Lab13-7 ADC with STANDBY

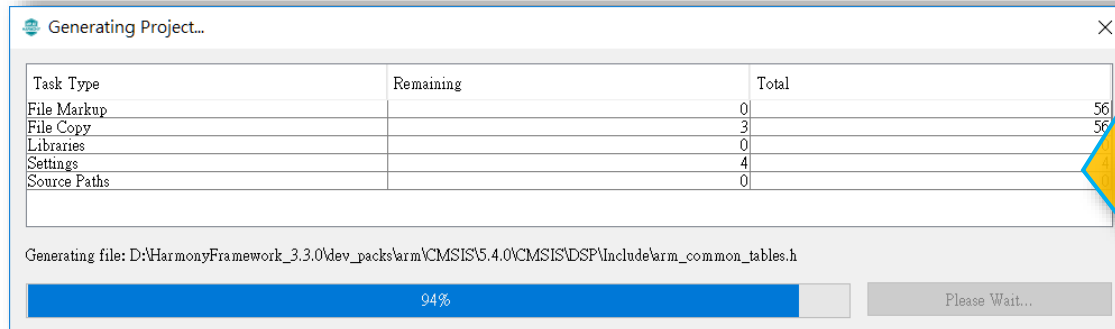
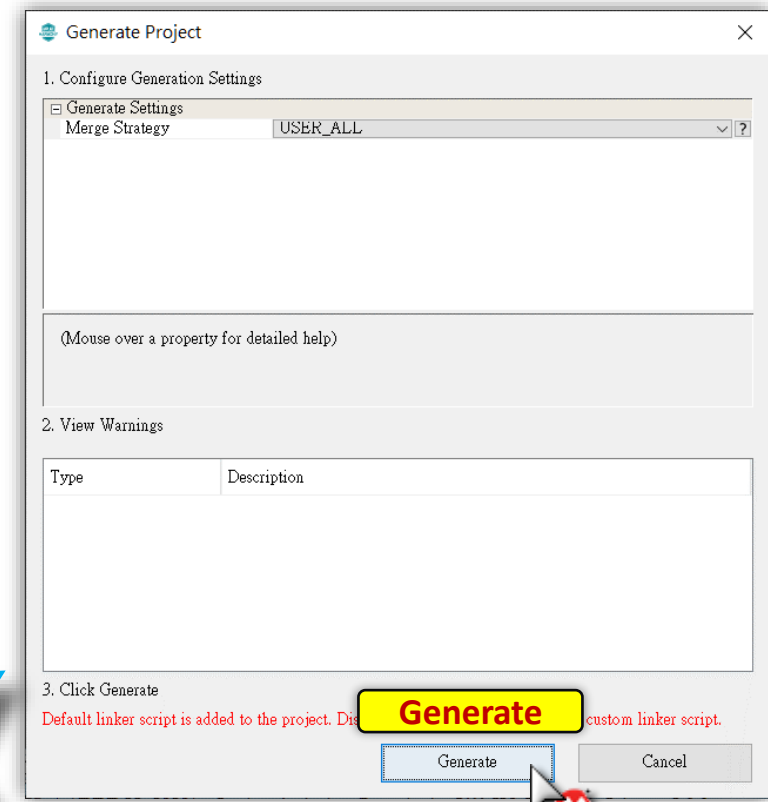
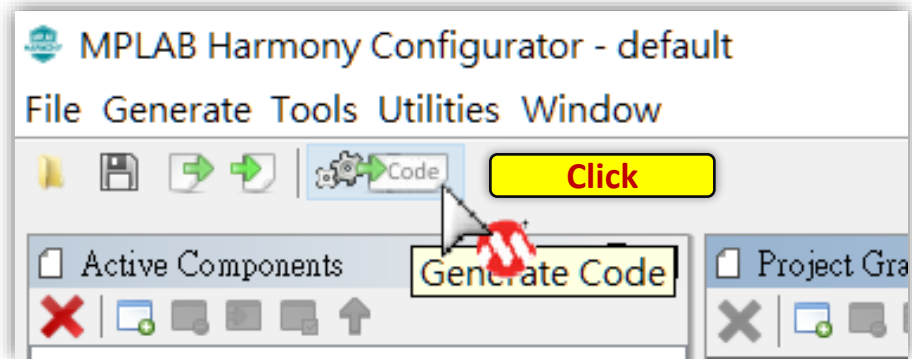
Step 4



⚙️ Lab13-7 ADC with STANDBY

Step 5

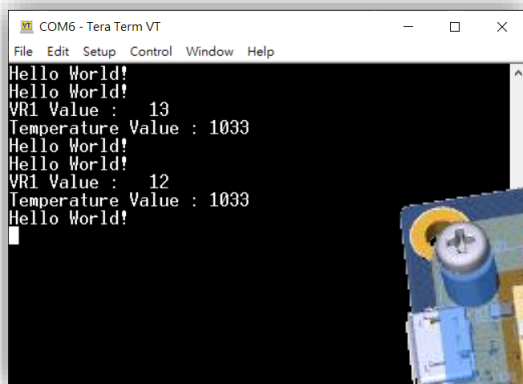
- Click  to Generate Code.



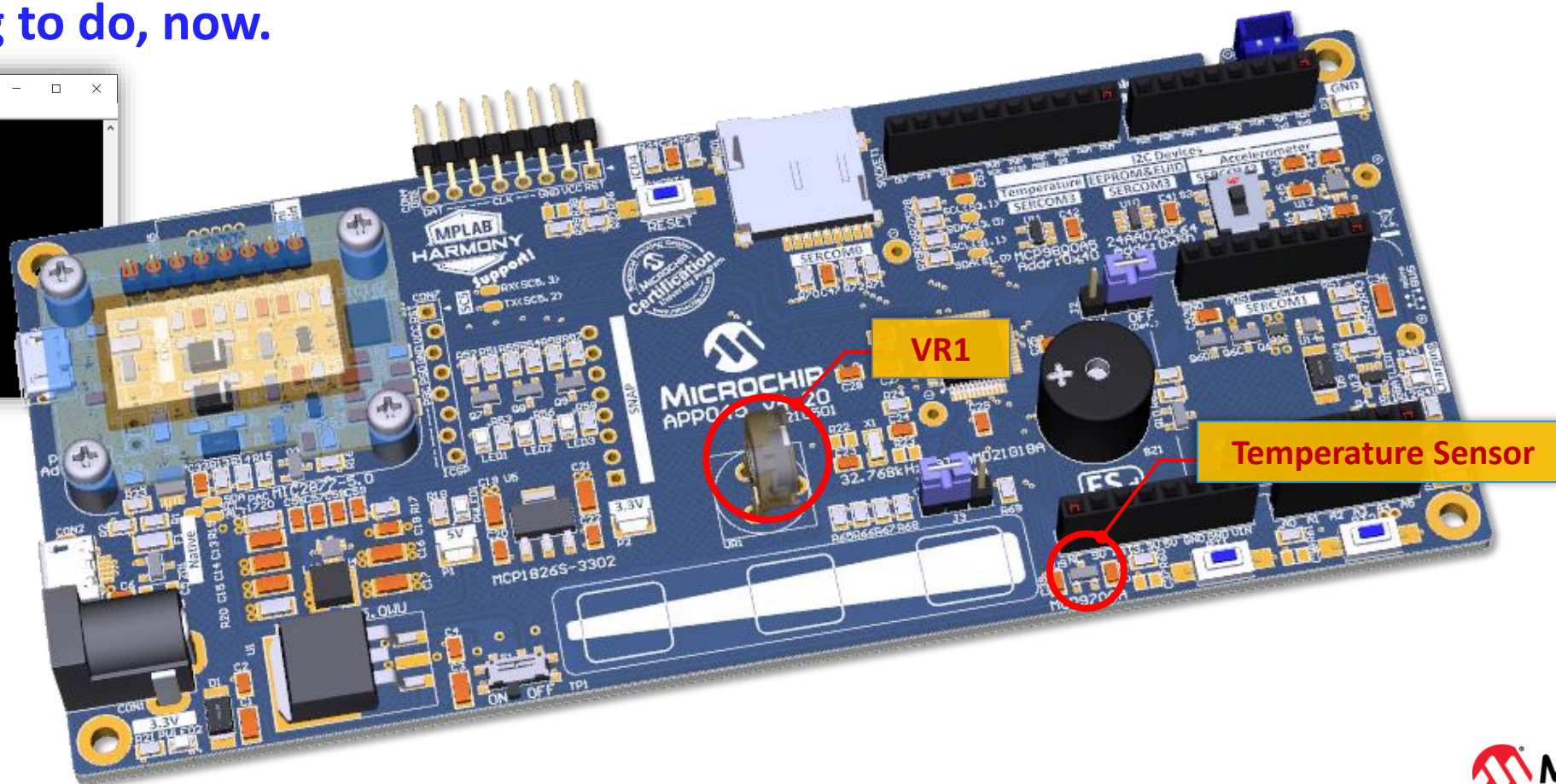
⚡ Lab13-7 ADC with STANDBY

Result

- “Hello World!!” string and ADC result of VR1 / Temperature Sensor will output to terminal with two second period. Same as Lab 13, but MCU enter STANDBY mode when nothing to do, now.



```
COM6 - Tera Term VT
File Edit Setup Control Window Help
Hello World!
Hello World!
VR1 Value : 13
Temperature Value : 1033
Hello World!
Hello World!
VR1 Value : 12
Temperature Value : 1033
Hello World!
```

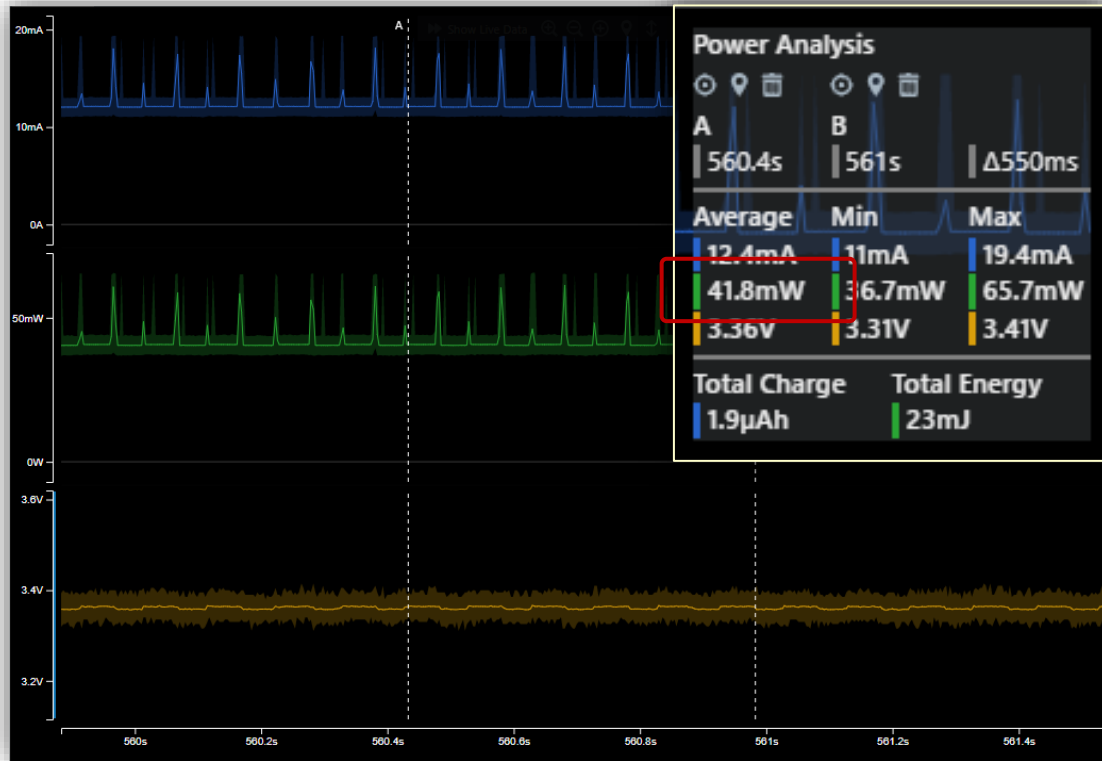


⚙️ Lab13-7 ADC with STANDBY

Result

- Power consumption compared for IDLE0 and STANDBY mode.

IDLE0 Sleep Mode

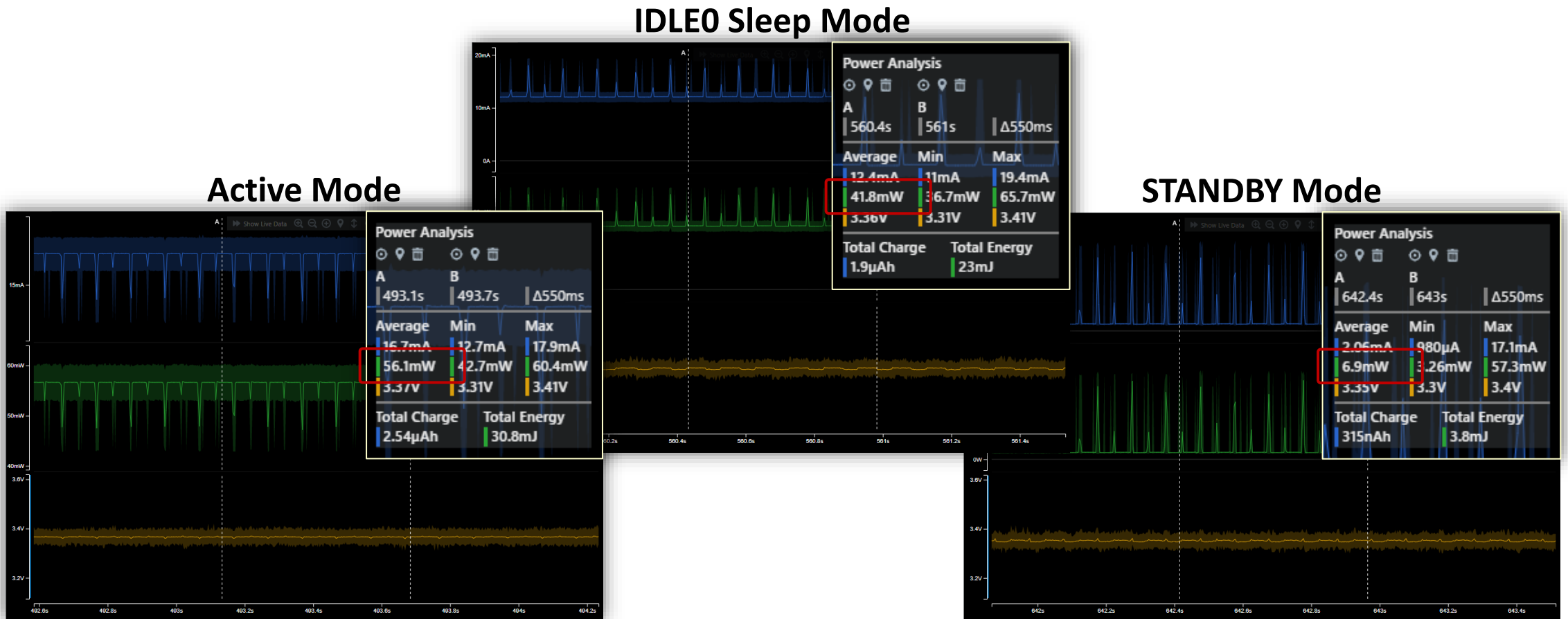


STANDBY Mode



⚙️ Lab13-7 ADC with STANDBY

- Discuss
 - Compare power consumption with Active, IDLE0 and STANDBY mode.



Application - SleepWalking

SleepWalking

- SleepWalking is the capability for a device to temporarily wake-up clocks for the peripheral to perform a task without waking-up the CPU in STANDBY sleep mode.
- Peripherals co-work at STANDBY sleep mode to reduce power consumption.
- Peripherals communication with by dedicated trigger source or event system (EVSYS).
- And wakes-up CPU when needs by interrupts or reset.

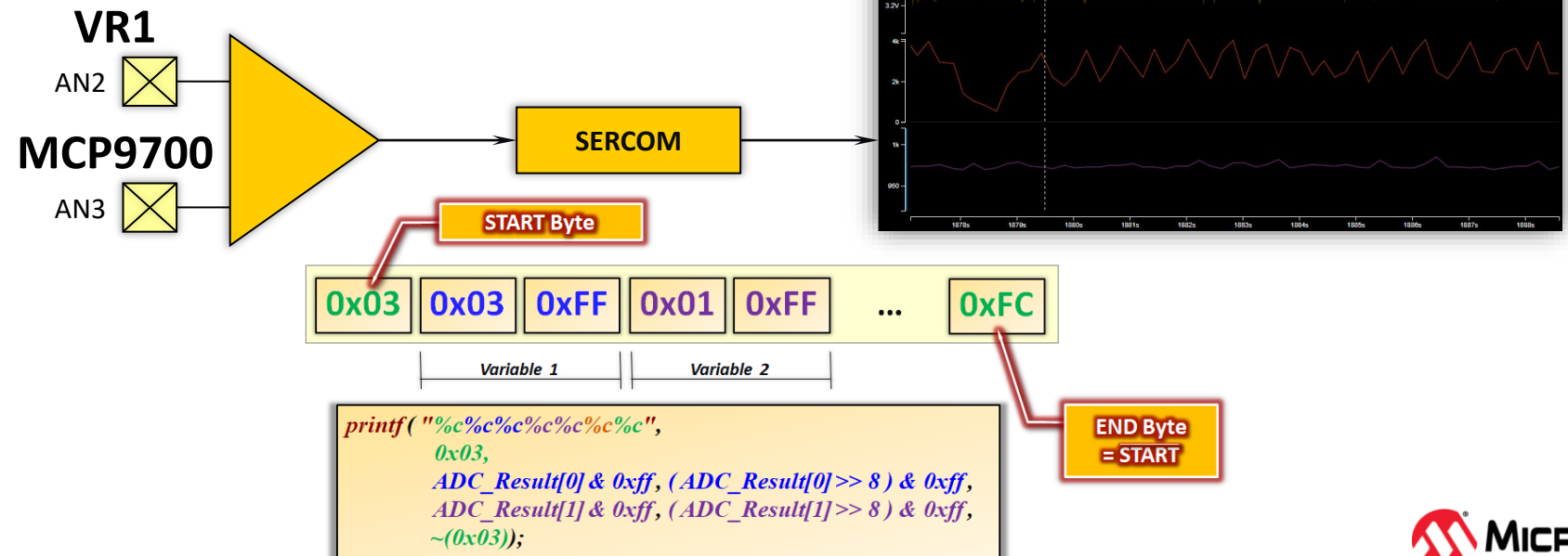
Hint & Tips for SleepWalking

- There are two things need to know when entry STANDBY sleep mode,
 - **DMAC be disable at IDLE1, IDLE2 and STANDBY sleep mode except IDLE0.**
 - DMAC clock by AHB clock.
 - So, must execute "wakes-up" before DMAC action.
 - CPU be wakes-upped, also.
 - wakes-up CPU will resume all clocks, bus and peripherals.
- **For Generally, don't enter sleep mode when running ISR.**
 - When MCU enter sleep mode on ISR running, NVIC will keep current interrupt priority and CPU be halted.
 - When same priority interrupt occurred, CPU be wakes-upped, but NVIC can't accept the interrupt require.
 - At this situation, CPU is alive, but can't do any thing. System will "Crash".

Sleep Mode	CPU Clock	AHB Clock	APB Clock	Oscillators				Main Clock	Regulator Mode	RAM Mode
				ONDEMAND = 0		ONDEMAND = 1				
				RUNSTDBY=0	RUNSTDBY=1	RUNSTDBY=0	RUNSTDBY=1			
Idle 0	Stop	Run	Run	Run	Run	Run if requested	Run if requested	Run	Normal	Normal
Idle 1	Stop	Stop	Run	Run	Run	Run if requested	Run if requested	Run	Normal	Normal
Idle 2	Stop	Stop	Stop	Run	Run	Run if requested	Run if requested	Run	Normal	Normal
Standby	Stop	Stop	Stop	Stop	Run	Stop	Run if requested	Stop	Low power	Low power

Application Require

- At this section, we try to create an application to demo SleepWalking feature.
- The application require list at below,
 - To Convert VR1 and MCP9700 analog voltage
 - Based on ADC's Pin-Scan mode and sample time control by TC3.
 - Output the results to MPLAB Data Visualizer.
 - follow data stream format for Data Streamer.
 - Require minimum power consumption. (under 10mW)



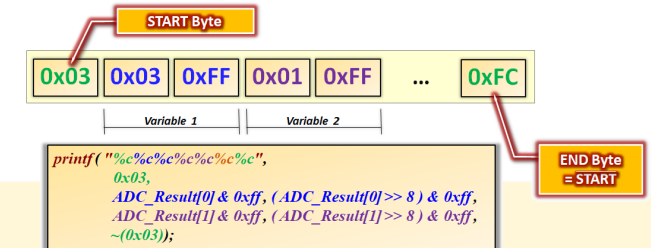
⚡ Lab 13-8 ADC with DV

- Please continue from
SAM2001 Lab13 (ADC PinScan Interrupt Callback SWTrigger)
The function of project very similar as SAM2001 Lab13.
- There are key differ is

```
myprintf("VR1 Value : %4d\r\n", ADC_Result[0]);  
myprintf("Temperature Value : %4d\r\n", ADC_Result[1]);
```



```
myprintf("%c%c%c%c%c%c",  
0x03,  
ADC_Result[0] & 0xff, (ADC_Result[0] >> 8) & 0xff,  
ADC_Result[1] & 0xff, (ADC_Result[1] >> 8) & 0xff,  
~(0x03));
```



- **Let's go !**

⚙️ Lab 13-8 ADC with DV

Step 1

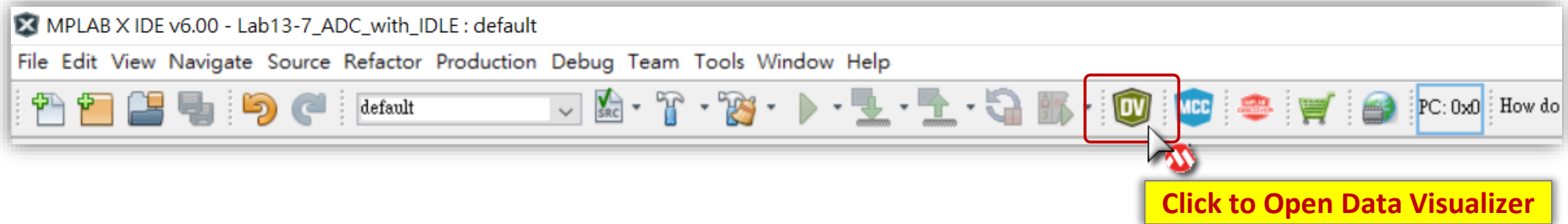
```
int main (void)
{
    if (TC3_HasExpired)
    {
        // TODO 13-8.01
        //myprintf("Hello World!\r\n");
    }
    if (USART5_IsReceived)
    {
        // TODO 13-8.02
        //myprintf("\r\nReceived Data : %1c\r\n", USART5_ReceiveData[0]);
    }
    if (ADC_IsCompleted)
    {
        // TODO 13-8.03
        //myprintf("VR1 Value : %4d\r\n", ADC_Result[0]);
        //myprintf("Temperature Value : %4d\r\n", ADC_Result[1]);

        // TODO 13-8.04
        myprintf("%c%c%c%c%c%c",
                0x03,
                ADC_Result[0] & 0xff, (ADC_Result[0] >> 8) & 0xff,
                ADC_Result[1] & 0xff, (ADC_Result[1] >> 8) & 0xff,
                ~(0x03));
    }
}
```

⚙️ Lab 13-8 ADC with DV

Step 2

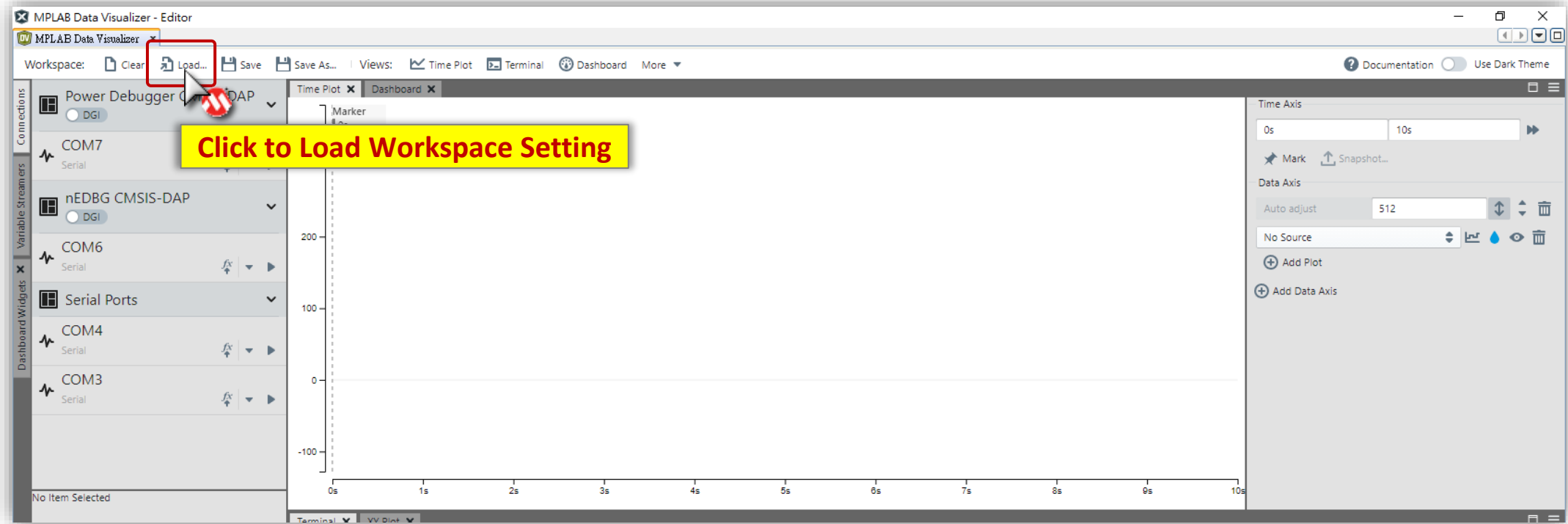
- Open Data Visualizer by icon at MPLAB X IDE



⚙️ Lab 13-8 ADC with DV

Step 3

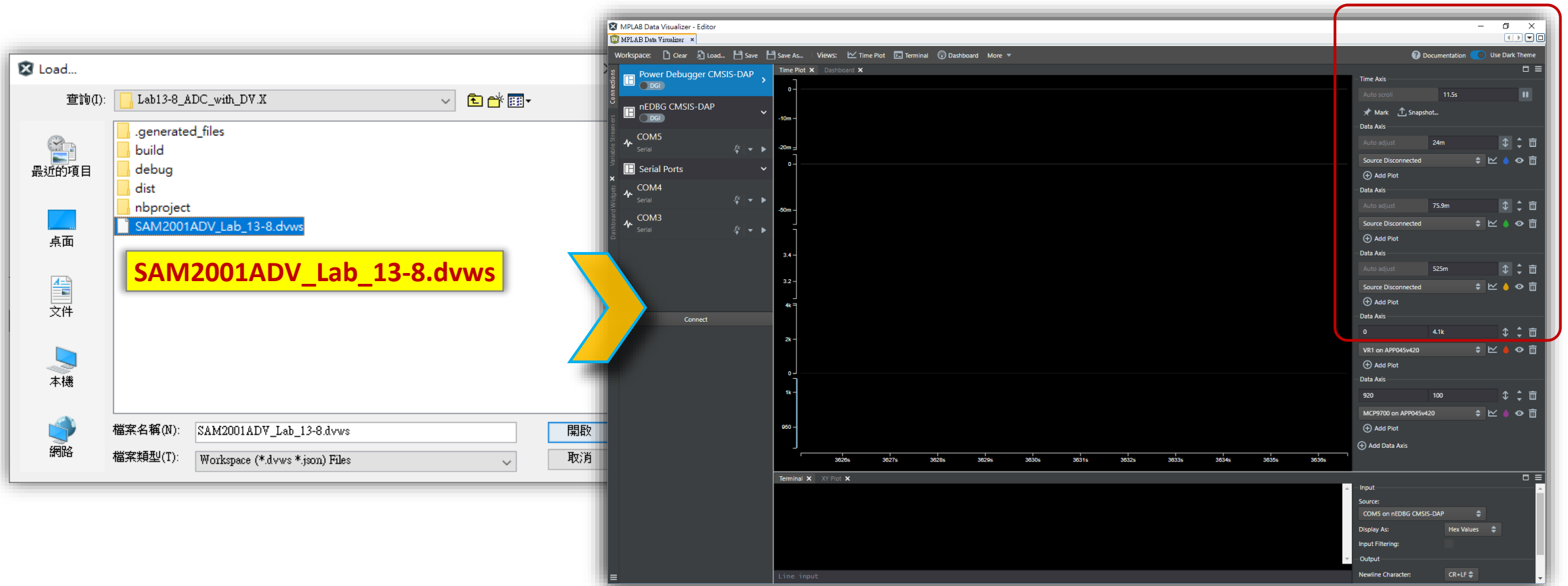
- Load layout file to reduce setting steps.



⚡ Lab 13-8 ADC with DV

Step 4

- Select “**SAM2001ADV_Lab_13-8.dvws**” layout file. Then data axis be set. Current, Power, Voltage, VR1 and MCP9700.



⚡ Lab 13-8 ADC with DV

Step 5

- Enable **Power** of **DGI** and **COMx** left side at Data Visualizer.

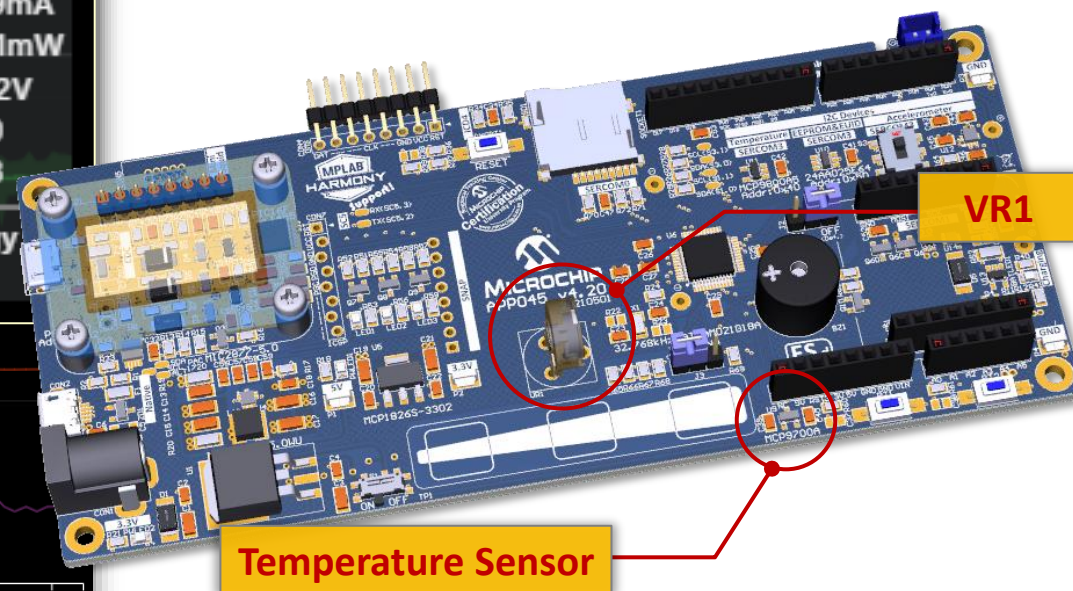
The sequence of screenshots illustrates the configuration steps for the MPLAB Data Visualizer:

- Screenshot 1:** The 'Connections' panel shows 'Power Debugger CMSIS-DAP' selected. A yellow box highlights the 'DGI' option. A yellow arrow points to the next screenshot.
- Screenshot 2:** The 'Dashboard Widgets' panel shows 'COM5' selected. A yellow box highlights the 'COMx' option. A yellow arrow points to the next screenshot.
- Screenshot 3:** The 'COM5 Settings' panel is shown with 'Baud Rate' set to 115200, 'Char Length' to 8 bits, 'Parity' to None, and 'Stop Bits' to 1 bit. A yellow arrow points to the final screenshot.
- Screenshot 4:** The 'Time Plot' view is shown with a red box highlighting the selected channels. The 'Terminal' view at the bottom shows the data stream.

⚡ Lab 13-8 ADC with DV

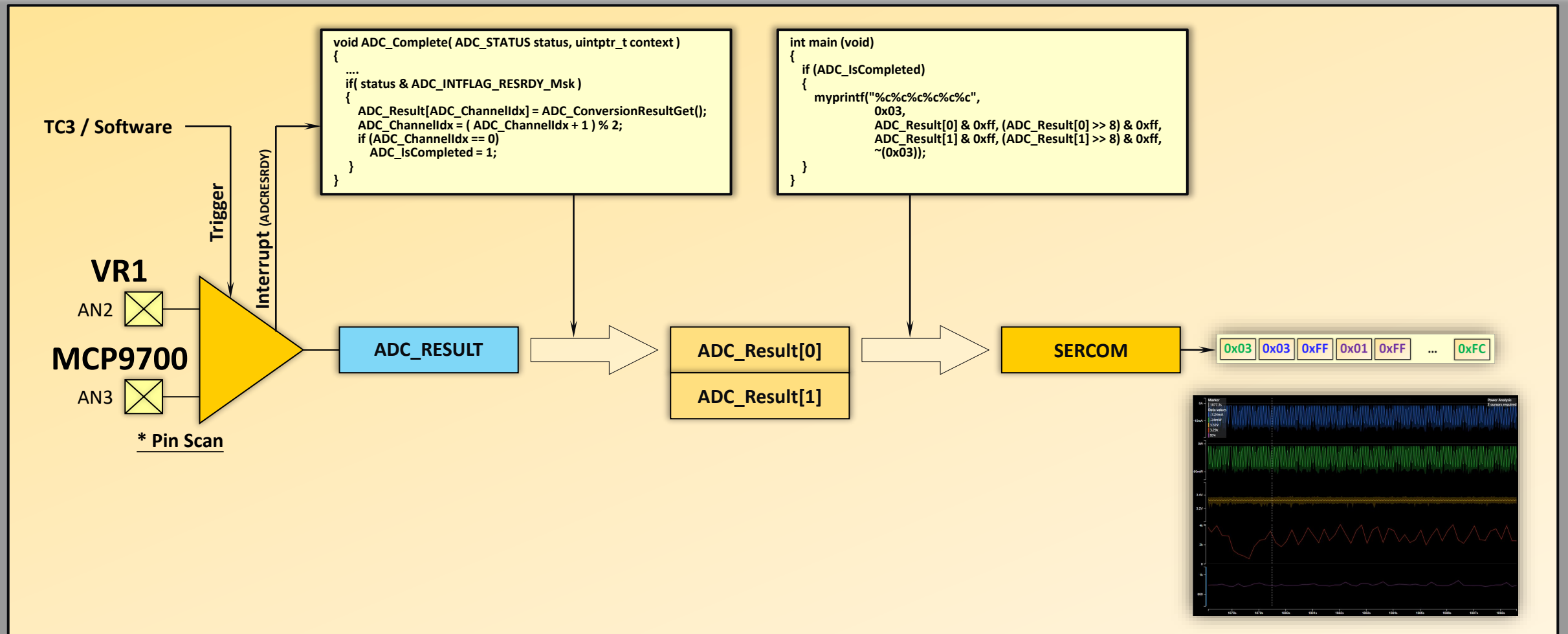
Result

- ADC result of VR1 / Temperature Sensor will output to DV with to second period. Same as SAM2001 Lab 13. But output follow data stream format.



✂ Lab 13-8 ADC with DV

- Discuss, this is block diagram for Lab 13-8



Lab 13-8 ADC with DV

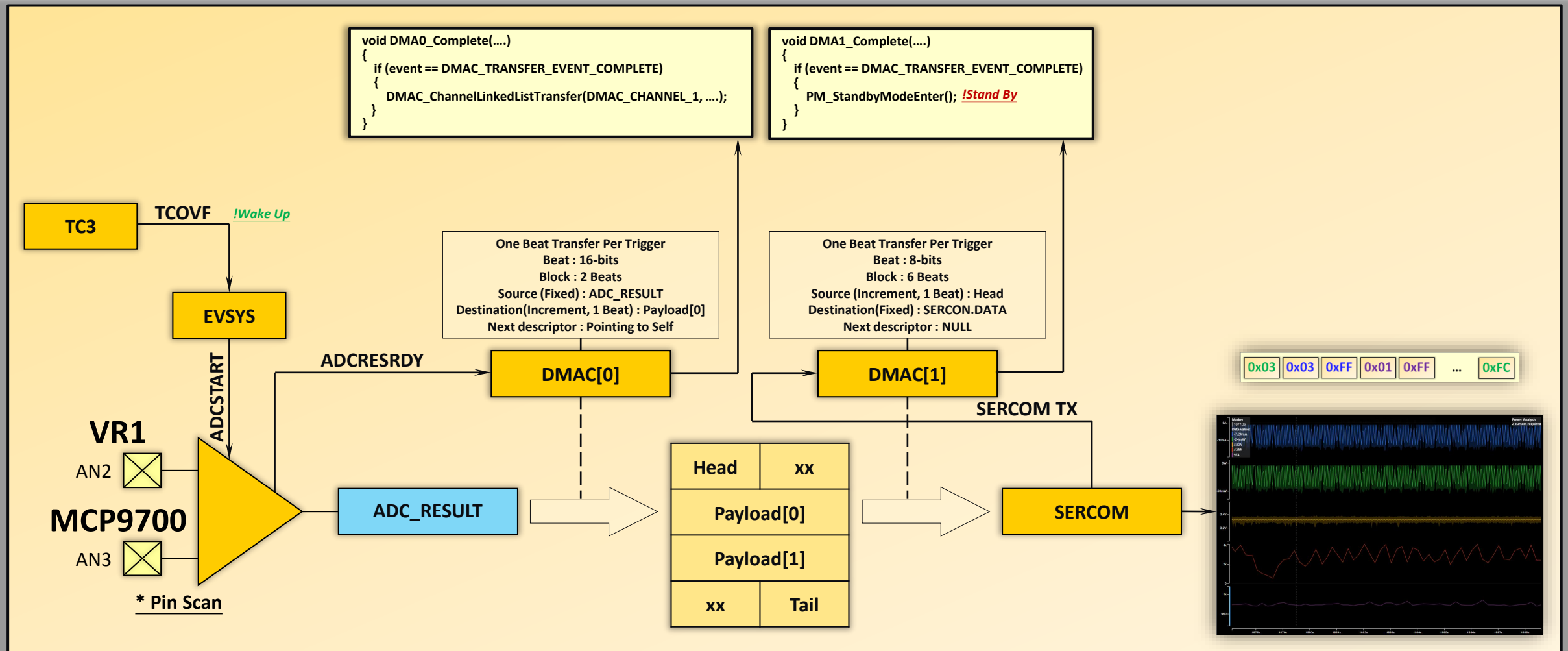
- Discuss
 - Lab13-8 Based on Active power mode.
CPU take logic control, data moving and more power needs.
 - How to reduce power consumption ?
- **SleepWalking is best chosen at there.**
- What key needs to know ?
- What kind different between
Lab13-9 and that a lab based-on SleepWalking ?

✂ Lab13-9 ADC Scan DV DMAC SleepWalking

- This is demo project.
- The function of project is same as [Lab13-8 \(ADC with DV\)](#).
- It's a new code architecture based-on SleepWalking and DMAC.
- Program running at Normal and STANDBY mode, alternately.
- Please just download the project and compare that two labs for power consumption.
 - SAM2001ADV Lab13-8 : Avg. 54.4mW -> ?
- **Let's go !**

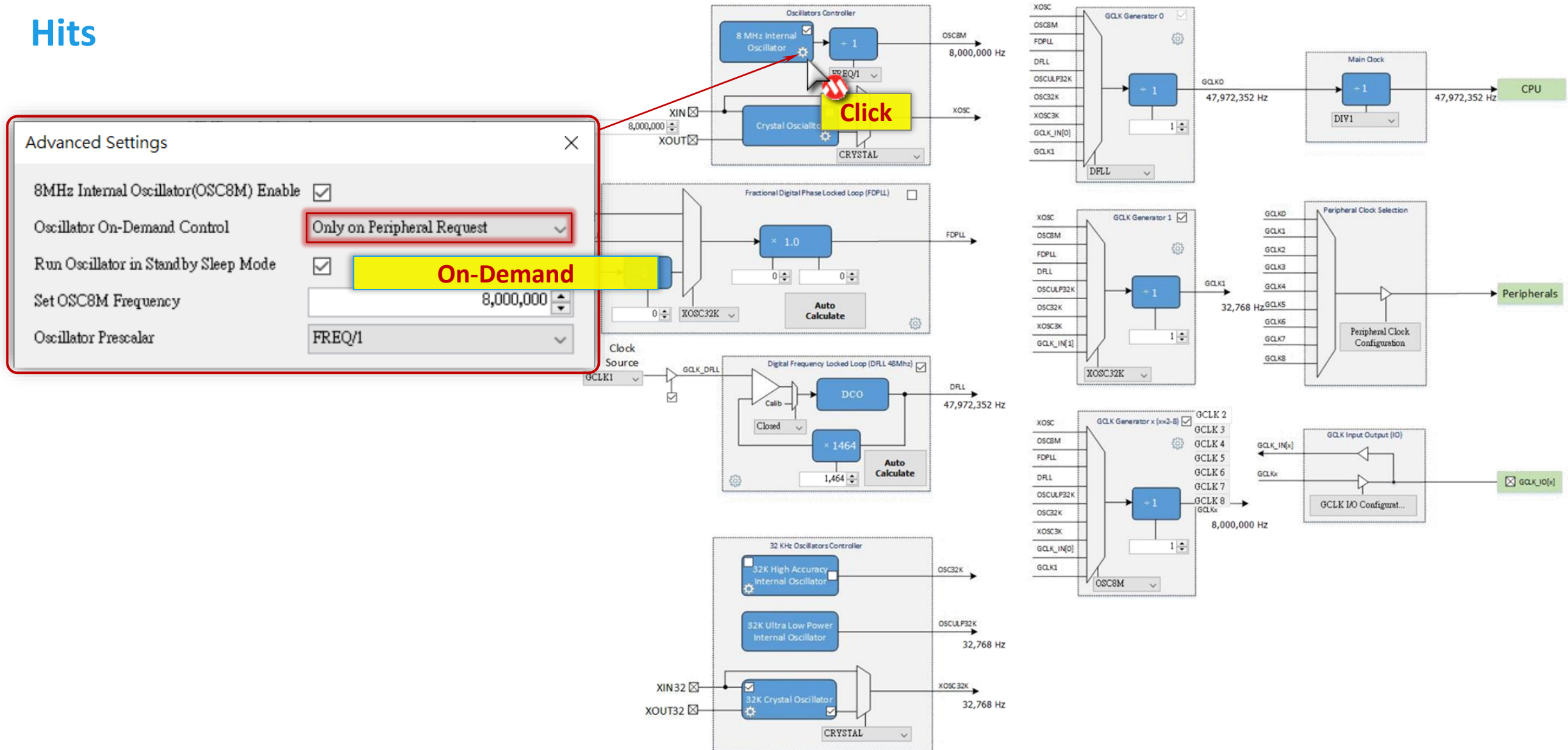
⚡ Lab13-9 ADC Scan DV DMAC SleepWalking

- Apply new process to reduce CPU loading.



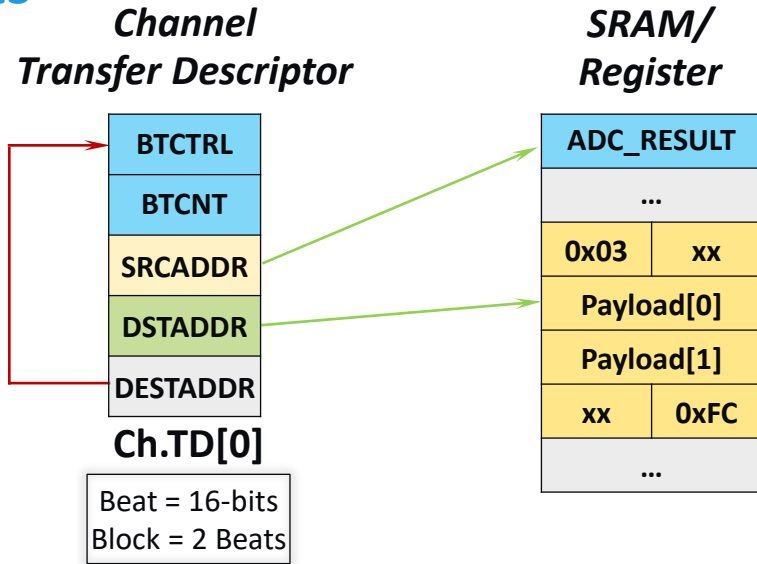
⚡ Lab13-9 ADC Scan DV DMAC SleepWalking

Hits



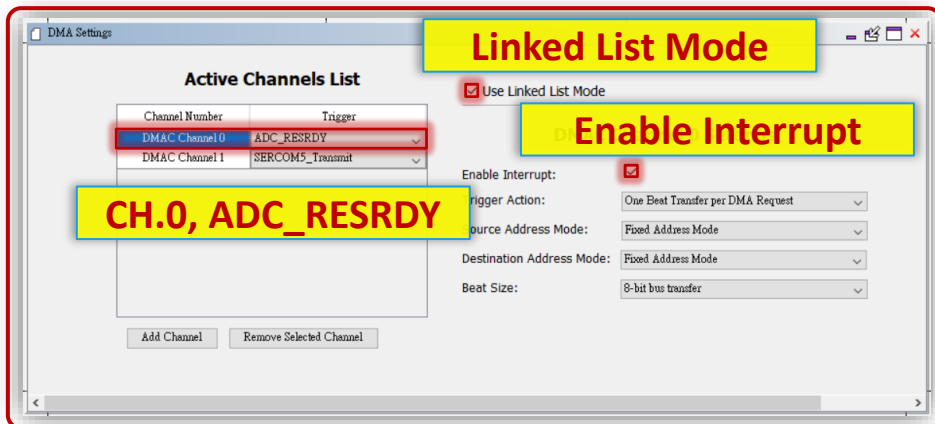
⚡ Lab13-9 ADC Scan DV DMAC SleepWalking

Hits



```
dmac_descriptor_registers_t DMAC_Descriptor_ADC_SRAM[1];  
  
DMAC_LinkedListDescriptorSetup(&(DMAC_Descriptor_ADC_SRAM[0]),  
    DMAC_BTCTRL_STEPSEL(0) | DMAC_BTCTRL_STEPSIZE(0) |  
    DMAC_BTCTRL_SRCINC(0) | DMAC_BTCTRL_DSTINC(1) |  
    DMAC_BTCTRL_BEATSIZE_HWORD | DMAC_BTCTRL_BLOCKACT_INT |  
    DMAC_BTCTRL_VALID(1),  
    (const void *) &(ADC_REGS->ADC_RESULT),  
    (const void *) &(DV_Package_ADC.Payload[0],  
    sizeof (DV_Package_ADC.Payload), /* 2 Beats */  
    &(DMAC_Descriptor_ADC_SRAM[0]));
```

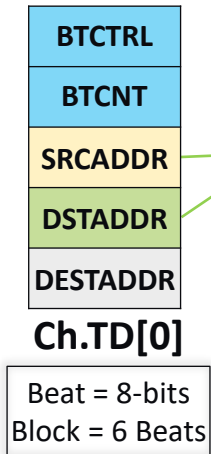
Linked List, Continuously Event



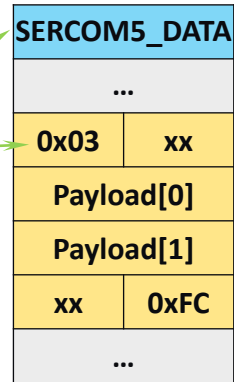
⚡ Lab13-9 ADC Scan DV DMAC SleepWalking

Hits

Channel Transfer Descriptor



SRAM/ Register



Link NULL, One-Shot Event

Linked List Mode

☒ Use Linked List Mode

☒ Enable Interrupt

CH.1, SERCON5_Tx

```
typedef struct  
{
```

```
    uint8_t:8;  
    uint8_t head;  
    uint16_t Payload[2];  
    uint8_t tail;  
    uint8_t:8;
```

```
} DV_PACKAGE_2x16BITS;
```

```
DV_PACKAGE_2x16BITS DV_Package_ADC;
```

```
DV_Package_ADC.head = 0x03;  
DV_Package_ADC.Payload[0] = 0x7FF; /* Dummy */  
DV_Package_ADC.Payload[1] = 0x7FF; /* Dummy */  
DV_Package_ADC.tail = ~DV_Package_ADC.head;
```

```
dmac_descriptor_registers_t DMAC_Descriptor_SRAM_SERCON5;
```

```
DMAC_LinkedListDescriptorSetup(&(DMAC_Descriptor_SRAM_SERCON5),  
    DMAC_BTCTRL_STEPSEL(0) | DMAC_BTCTRL_STEPSIZE(0) |  
    DMAC_BTCTRL_SRCINC(1) | DMAC_BTCTRL_DSTINC(0) |  
    DMAC_BTCTRL_BEATSIZE_BYTE | DMAC_BTCTRL_BLOCKACT_INT |  
    DMAC_BTCTRL_VALID(1),  
    (const void *) &(DV_Package_ADC) + 1,  
    (const void *) &(SERCOM5_REGS->USART_INT.SERCOM_DATA),  
    sizeof(DV_Package_ADC) - 2,  
    NULL);
```

⚙️ Lab13-9 ADC Scan DV DMAC SleepWalking

Hits

The image displays three screenshots of the Microchip Studio configuration interface, showing settings for TC3, ADC, and SERCOM5. Red annotations highlight specific configurations:

- TC3 Configuration:**
 - Counter Mode:** Counter in 16-bit mode
 - Select Prescaler:** Prescaler: GCLK_TC64
 - Operating Mode:** Timer
 - Timer:**
 - Time:** 50 (Use to triggering ADC)
 - Events:**
 - Enable Timer Period Overflow Event:** ☒ (Use to triggering ADC)
 - Sleep Configurations:**
 - Run during Standby:** ☒
- ADC Configuration:**
 - Select Prescaler:** Peripheral clock divided by 512
 - Select Sample Length (half ADC clock cycles):** 4
 - **** Conversion Time is 640.0 uS ******
 - Select Gain:** (Annotated: Event Input from TC3_OVF)
 - Select Reference:** 1/2 VDDANA (only for VDDANA > 2.0V)
 - Select Conversion Trigger:** HW Event Trigger (Annotated: Event Input for Start)
 - Channel Configuration:**
 - Enable Flush Event Input:** ☐
 - Enable Start Event Input:** ☒ (Annotated: Event Input for Start)
 - Result Configuration:**
 - Select Result Resolution:** 12-bit result
 - Left Aligned Result:** ☐
 - Enable Result Ready Interrupt:** ☒ (Annotated: Use to triggering DMA[0])
 - Enable Result Ready Event Out:** ☐
 - Window Mode Configuration:**
 - Run During Standby:** ☒
- SERCOM5 Configuration:**
 - Select SERCOM Operation Mode:** USART with internal Clock
 - Operating Mode:** Blocking mode (Annotated: Use to triggering DMA[1])
 - Receive Enable:** ☐
 - Transmit Enable:** ☒ (Annotated: Use to triggering DMA[1])
 - Frame Format:** USART frame
 - Baud Rate in Hz:** 115,200
 - Parity Mode:** No Parity
 - Character Size:** 8 Bits
 - Stop Bit Mode:** One Stop Bit
 - Receive Pinout:** SERCOM PAD[3] is used for data reception
 - Transmit Pinout:** PAD[2] = Tx; PAD[3] = XCK
 - Enable Run in Standby:** ☒

⚡ Lab13-9 ADC Scan DV DMAC SleepWalking

Hits

EVENT0 Easy View

Event System Manager

Channel Configuration

Channel Number	Event Generator	Event Status	User Ready	Remove Channel
Channel 0	TC3_OVF			Remove

TC3_OVF to trigger ADC Start Conversion

Add Channel

Channel 0 Settings

Path Selection: ASYNCHRONOUS

User Configuration

USER	Channel Number	Remove User
ADC_START	Channel 0	Remove

Add User

NVIC Settings

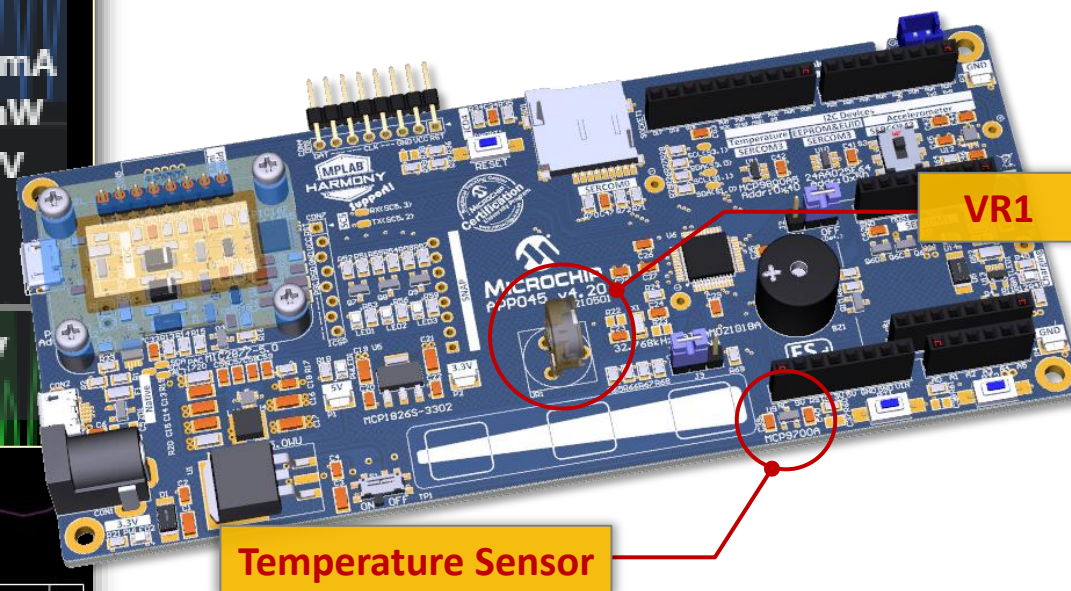
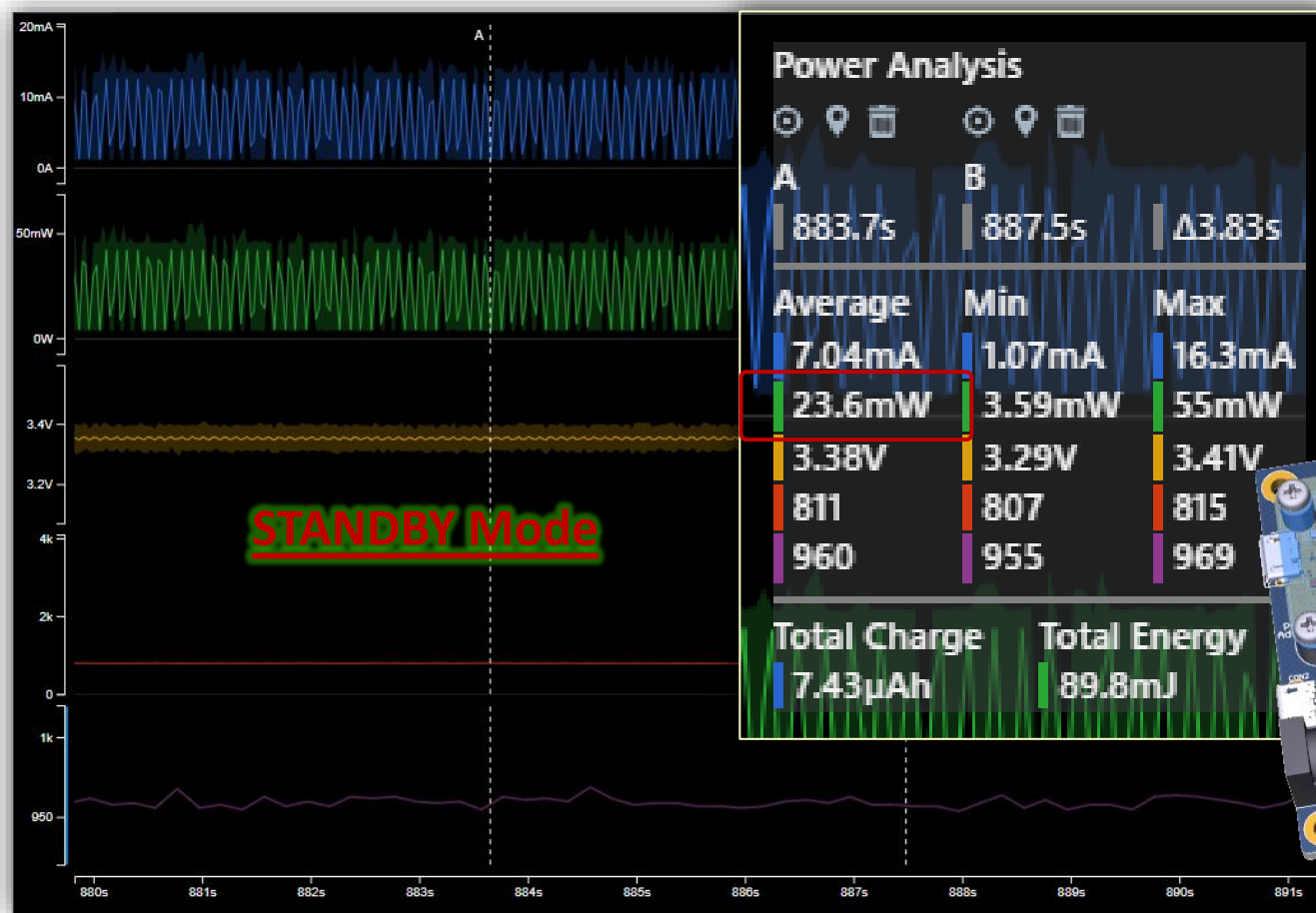
Vector Number	Vector	Enable	Priority (0 = Highest)	Handler Name
11	SERCOM2 (Serial Communication Interface 2)	☐	3	SERCOM2_Handler
12	SERCOM3 (Serial Communication Interface 3)	☐	3	SERCOM3_Handler
13	SERCOM4 (Serial Communication Interface 4)	☐	3	SERCOM4_Handler
14	SERCOM5 (Serial Communication Interface 5)	☐	3	SERCOM5_Handler
15	TCC0 (Timer/Counter for Control Applications 0)	☐	3	TCC0_Handler
16	TCC1 (Timer/Counter for Control Applications 1)	☐	3	TCC1_Handler
17	TCC2 (Timer/Counter for Control Applications 2)	☐	3	TCC2_Handler
18	TC3 (Timer/Counter 3)	☒	2	TC3_TimerInterruptHandler
19	TC4 (Timer/Counter 4)	☐	3	TC4_Handler
20	TC5 (Timer/Counter 5)	☐	3	TC5_Handler
23	ADC (Analog-to-Digital Converter)	☒	3	ADC_InterruptHandler
24	AC (Analog Comparators)	☐	3	AC_Handler
25	DAC (Digital-to-Analog Converter)	☐	3	DAC_Handler
26	PTC (Peripheral Touch Controller)	☐	3	PTC_Handler
27	I2S (Inter-IC Sound Controller)	☐	3	I2S_Handler

Higher priority to occupy ISR.

⚙️ Lab13-9 ADC Scan DV DMAC SleepWalking

Result

- ADC result of VR1 / Temperature Sensor will output to DV with to second period. Same as Lab13-8. But based on SleepWalking feature.

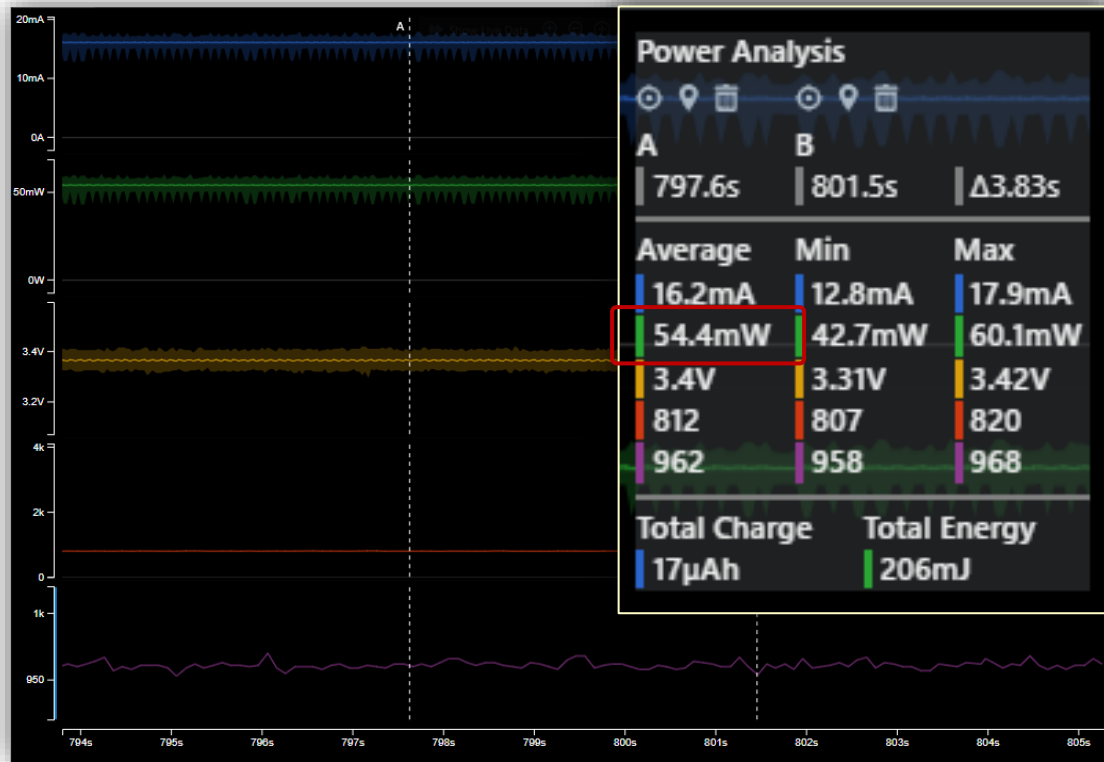


⚙️ Lab13-9 ADC Scan DV DMAC SleepWalking

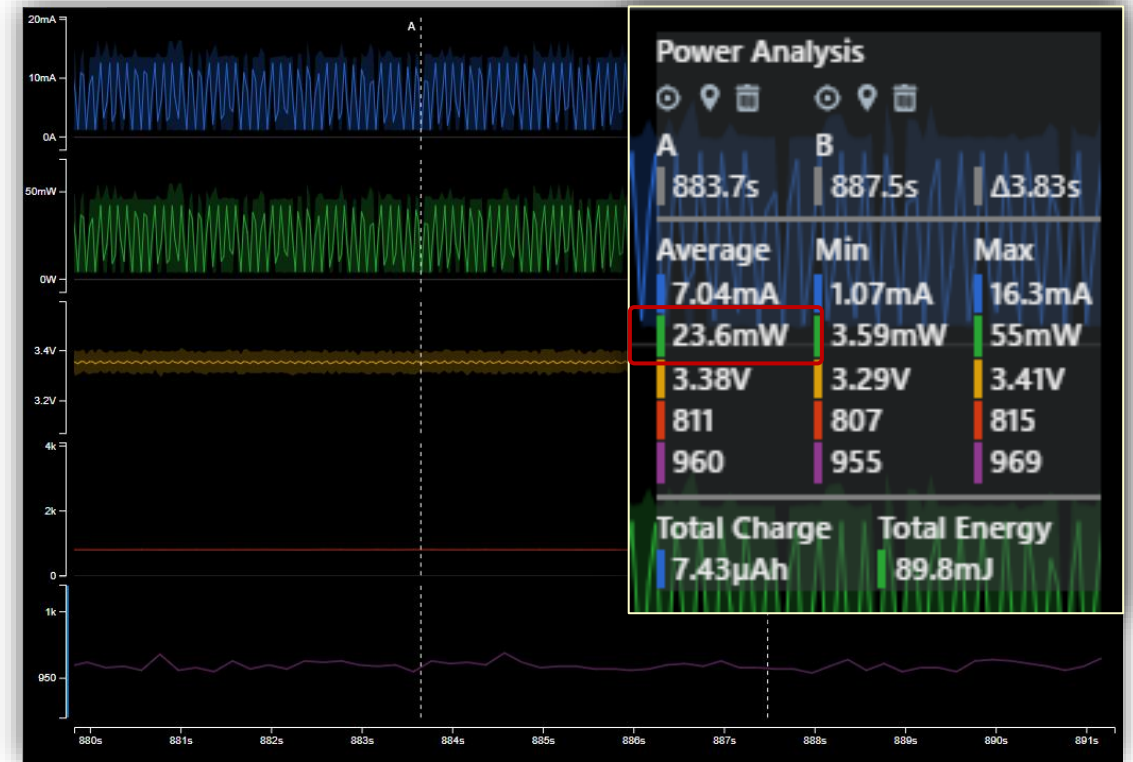
Result

- Power consumption compared for Lab13-8 and Lab13-9.

Lab13-8 Active

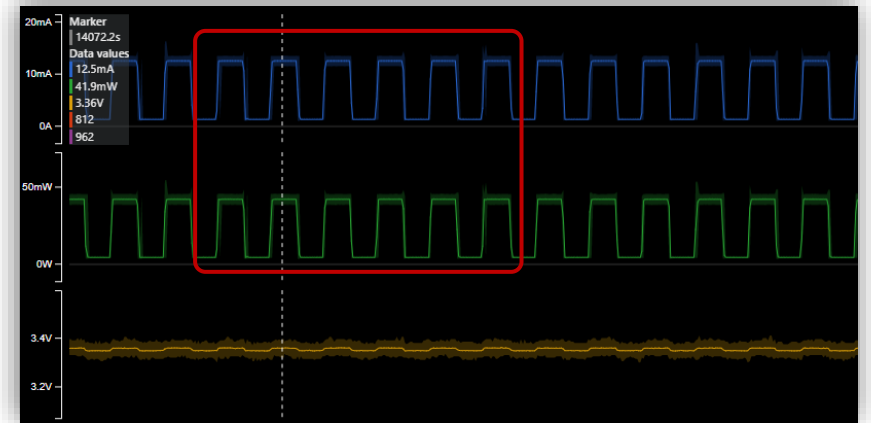


Lab13-9 SleepWalking



⚙️ Lab13-9 ADC Scan DV DMAC SleepWalking

- Discuss
 - The new code architecture based on SleepWalking and DMAC.
It's help reduce power consumption. You already saw it at Lab 13-9.
- But any thing make you confused When you watch the power waveform at MPLAB Data Visualizer ?
 - CPU seem be wake-up continually. it can't go to sleep mode most time.
it's not a good situation.
- Why ? Who wake-up CPU continually or some one occupy the clock / bus ?
 - CPU be wakes-upped When DMAC acting.
- So, Key is that try to save DMAC action time, Let CPU go to sleep mode most time.

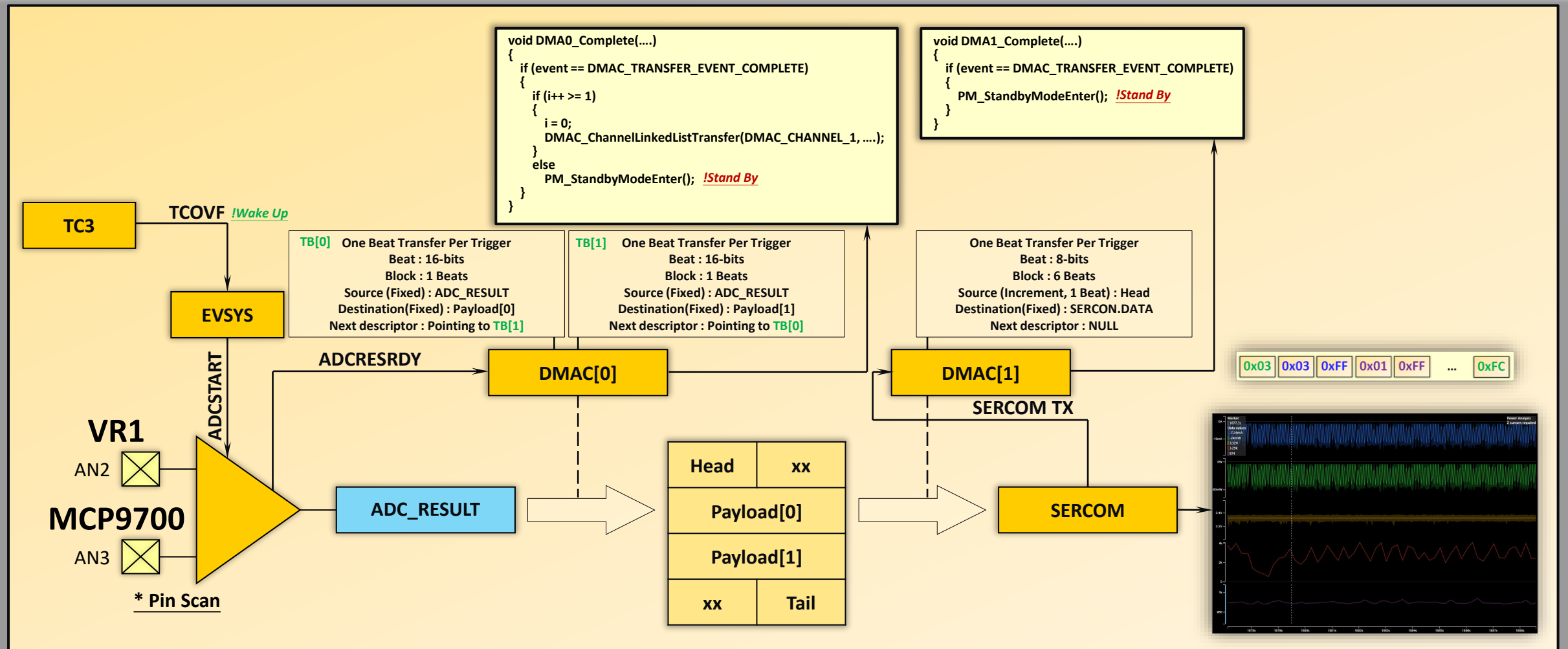


✂ Lab13-10 ADC Scan DV DMAC-Linked SleepWalking

- This is demo project.
- The function of project is same as [Lab13-8](#), [Lab13-9](#).
- The code architecture based-on Lab13-9 and set DMAC to Linked Transfer Block Mode to save action time of DMAC.
- Program running at Normal and STANDBY mode, alternately.
- Please just download the project and compare that two labs for power consumption.
 - SAM2001ADV Lab13-8 : Avg. **54.4mW**
 - SAM2001ADV Lab13-9 : Avg. **23.6mW** -> ?
- **Let's go !**

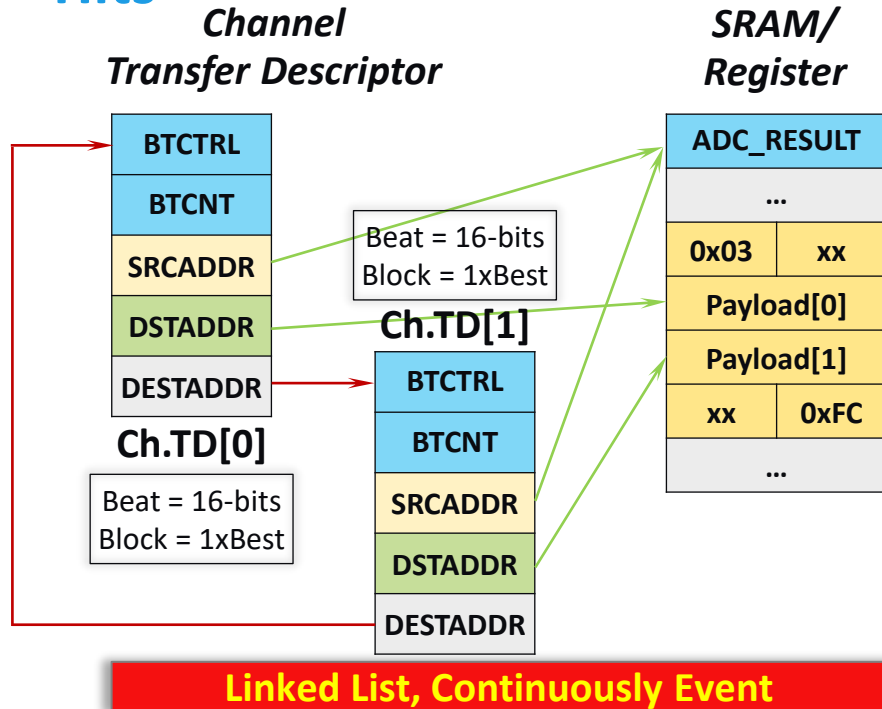
⚡ Lab13-10 ADC Scan DV DMAC-Linked SleepWalking

- Apply new block diagram to save DMAC action time.



⚡ Lab13-10 ADC Scan DV DMAC-Linked SleepWalking

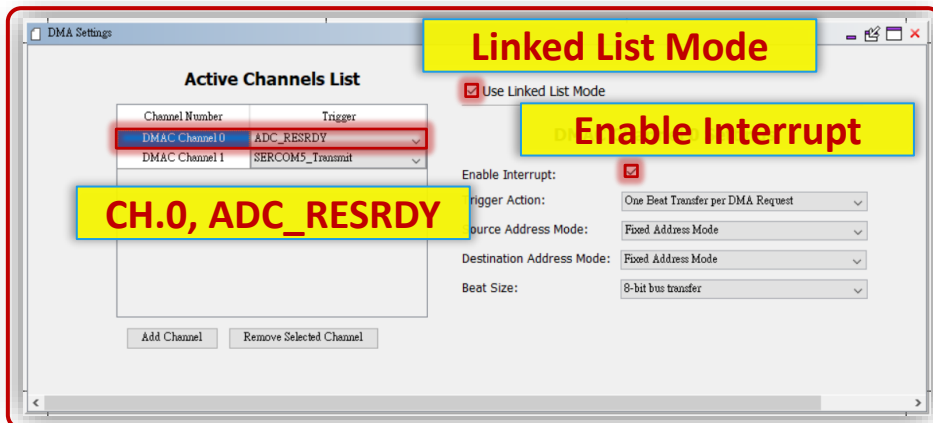
Hits



```
dmac_descriptor_registers_t DMAC_Descriptor_ADC_SRAM[2];
```

```
DMAC_LinkedListDescriptorSetup(&(DMAC_Descriptor_ADC_SRAM[0]),  
    DMAC_BTCTRL_STEPSEL(0) | DMAC_BTCTRL_STEPSIZE(0) |  
    DMAC_BTCTRL_SRCINC(0) | DMAC_BTCTRL_DSTINC(0) |  
    DMAC_BTCTRL_BEATSIZE_HWORD | DMAC_BTCTRL_BLOCKACT_INT |  
    DMAC_BTCTRL_VALID(1),  
    (const void *) &(ADC_REGS->ADC_RESULT),  
    (const void *) &(DV_Package_ADC.Payload[0],  
    sizeof (DV_Package_ADC.Payload[0]),  
    &(DMAC_Descriptor_ADC_SRAM[1]));
```

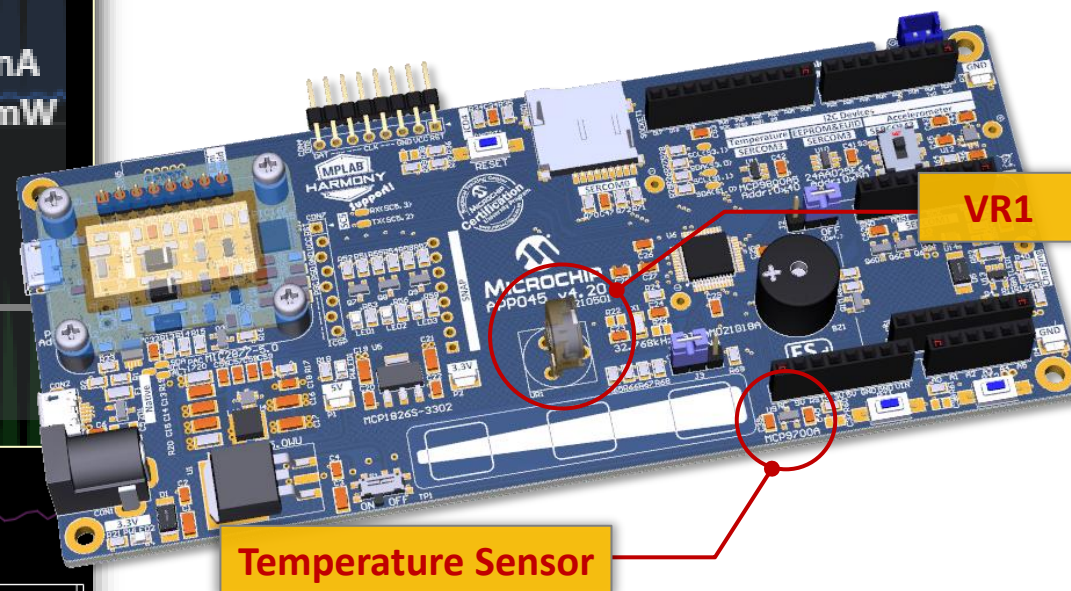
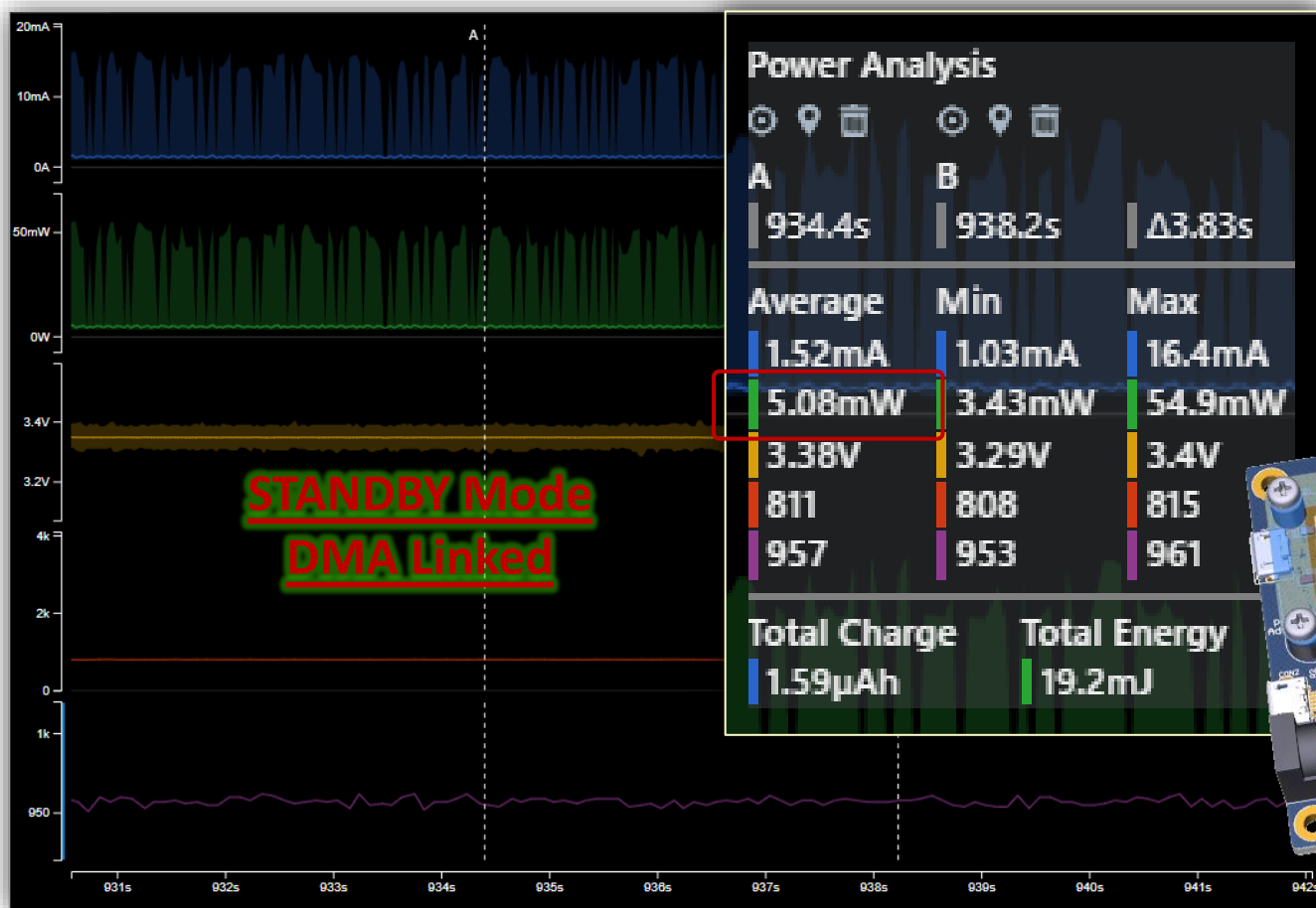
```
DMAC_LinkedListDescriptorSetup(&(DMAC_Descriptor_ADC_SRAM[0]),  
    DMAC_BTCTRL_STEPSEL(0) | DMAC_BTCTRL_STEPSIZE(0) |  
    DMAC_BTCTRL_SRCINC(0) | DMAC_BTCTRL_DSTINC(0) |  
    DMAC_BTCTRL_BEATSIZE_HWORD | DMAC_BTCTRL_BLOCKACT_INT |  
    DMAC_BTCTRL_VALID(1),  
    (const void *) &(ADC_REGS->ADC_RESULT),  
    (const void *) &(DV_Package_ADC.Payload[1],  
    sizeof (DV_Package_ADC.Payload[1]),  
    &(DMAC_Descriptor_ADC_SRAM[0]));
```



⚡ Lab13-10 ADC Scan DV DMAC-Linked SleepWalking

Result

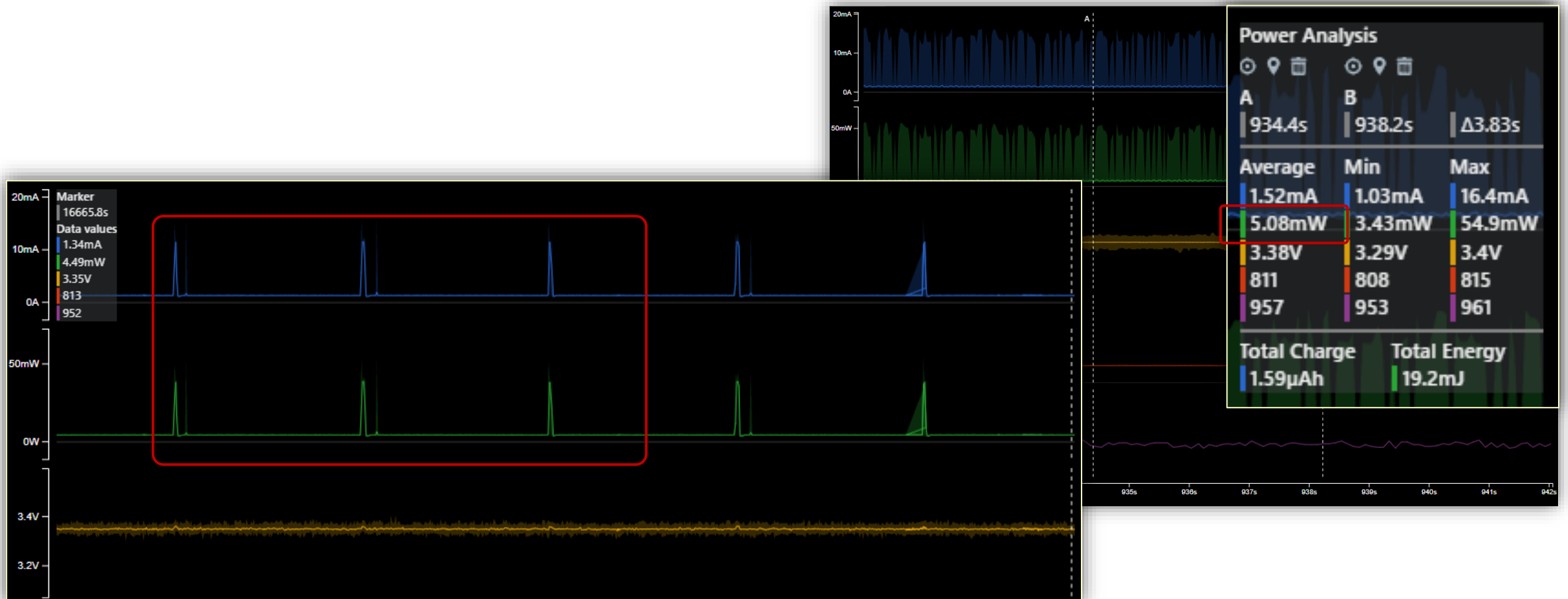
- ADC result of VR1 / Temperature Sensor will output to DV with to second period. Same as Lab 13-9. But based on SleepWalking feature and save DMAC action time.



⚙️ Lab13-10 ADC Scan DV DMAC-Linked SleepWalking

Result

- CPU go to sleep mode most time. And Avg. power is **5.08mW**.

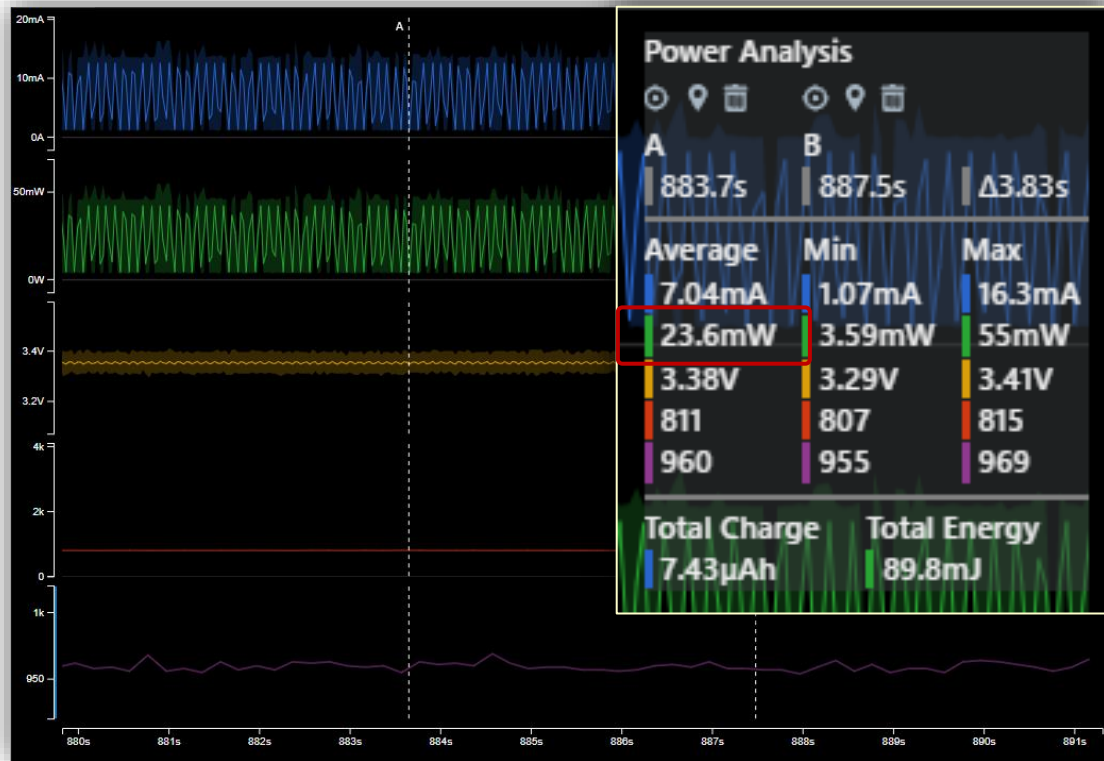


⚙️ Lab13-10 ADC Scan DV DMAC-Linked SleepWalking

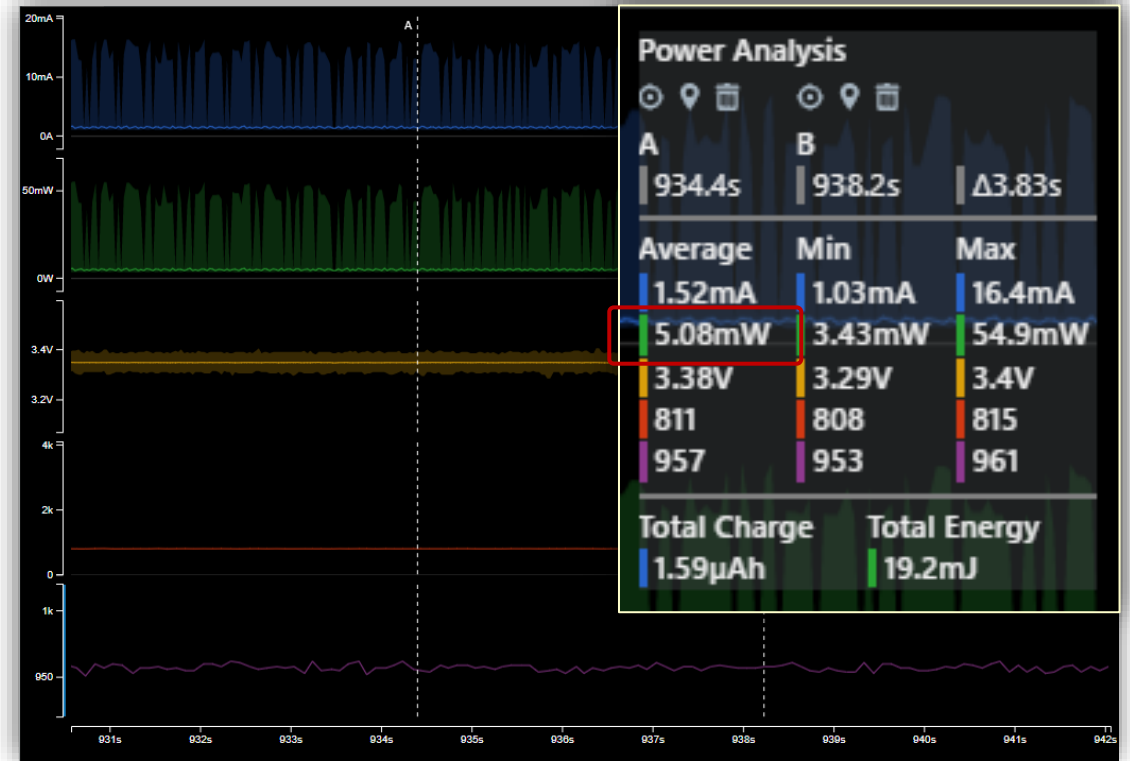
Result

- Power consumption compared for Lab13-9 and Lab13-10.

Lab13-9 SleepWalking



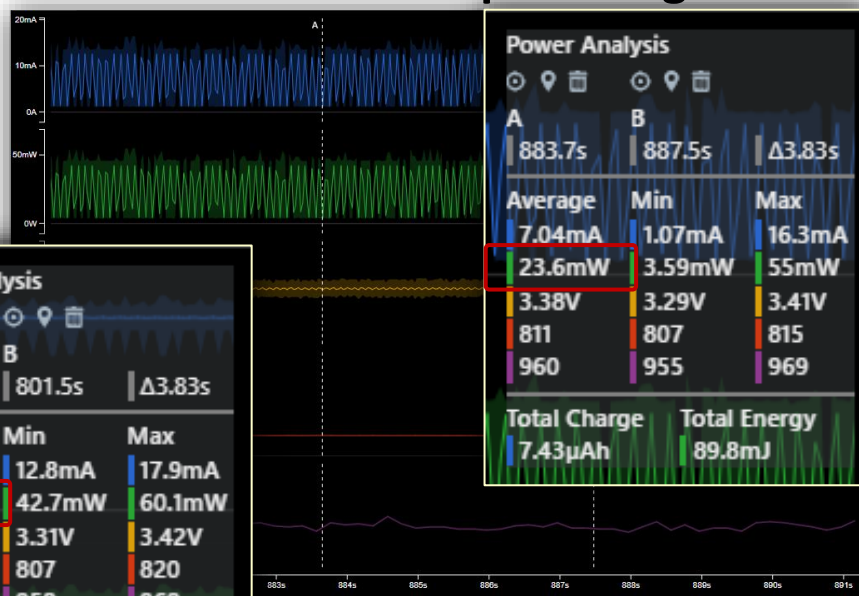
Lab13-10 SleepWalking & DMAC-Linked



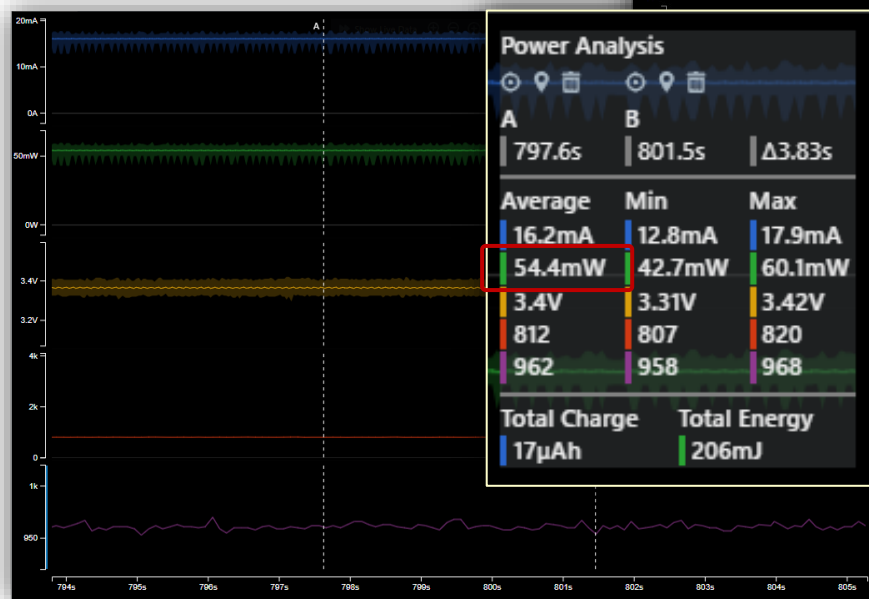
✂ Lab13-10 ADC Scan DV DMAC-Linked SleepWalking

- Discuss
 - Compare power consumption with Active, SleepWalking and SleepWalking with DMAC-Linked.

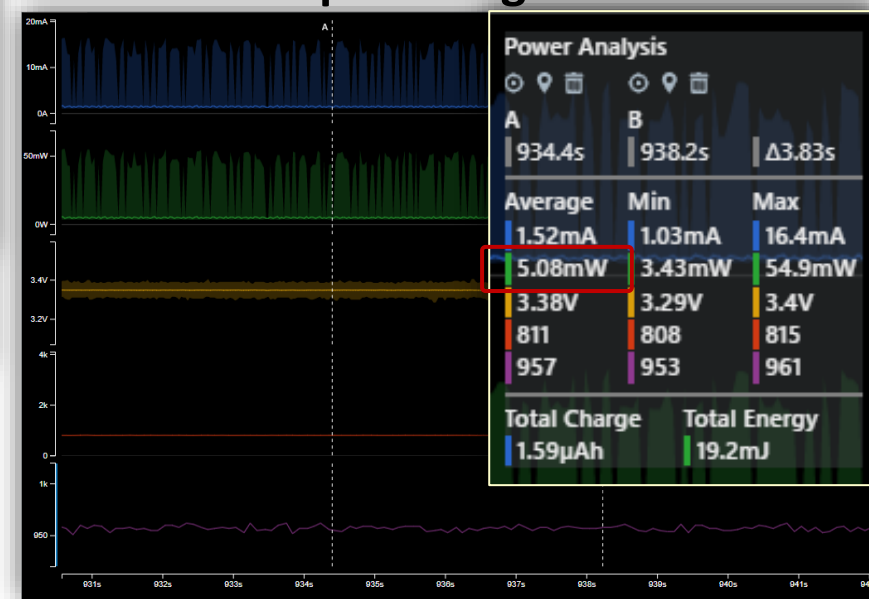
Lab13-9 SleepWalking



Lab13-8 Active



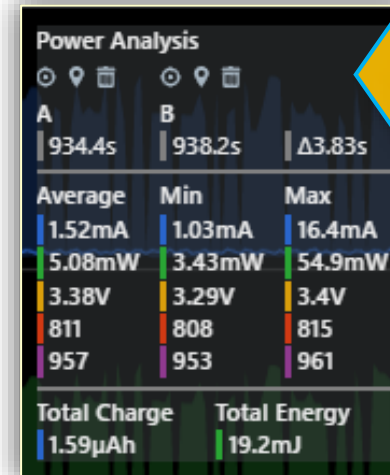
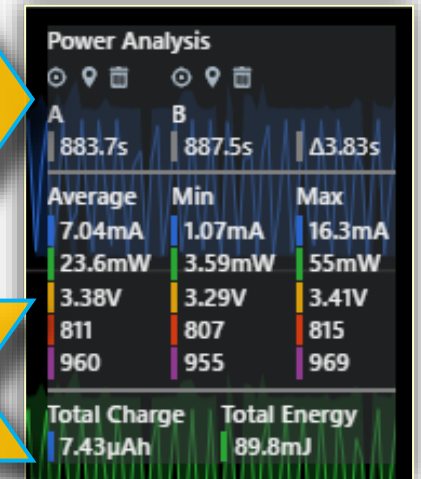
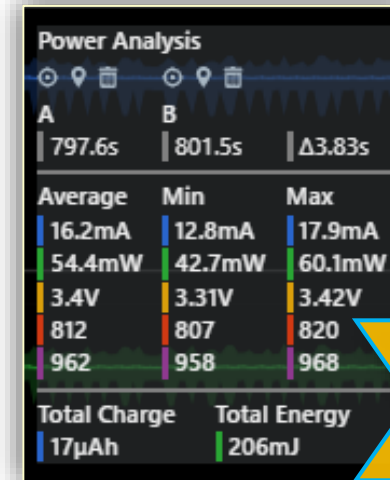
Lab13-10 SleepWalking & DMAC-Linked



✂ Lab13-8, 9, 10 Final Comparison

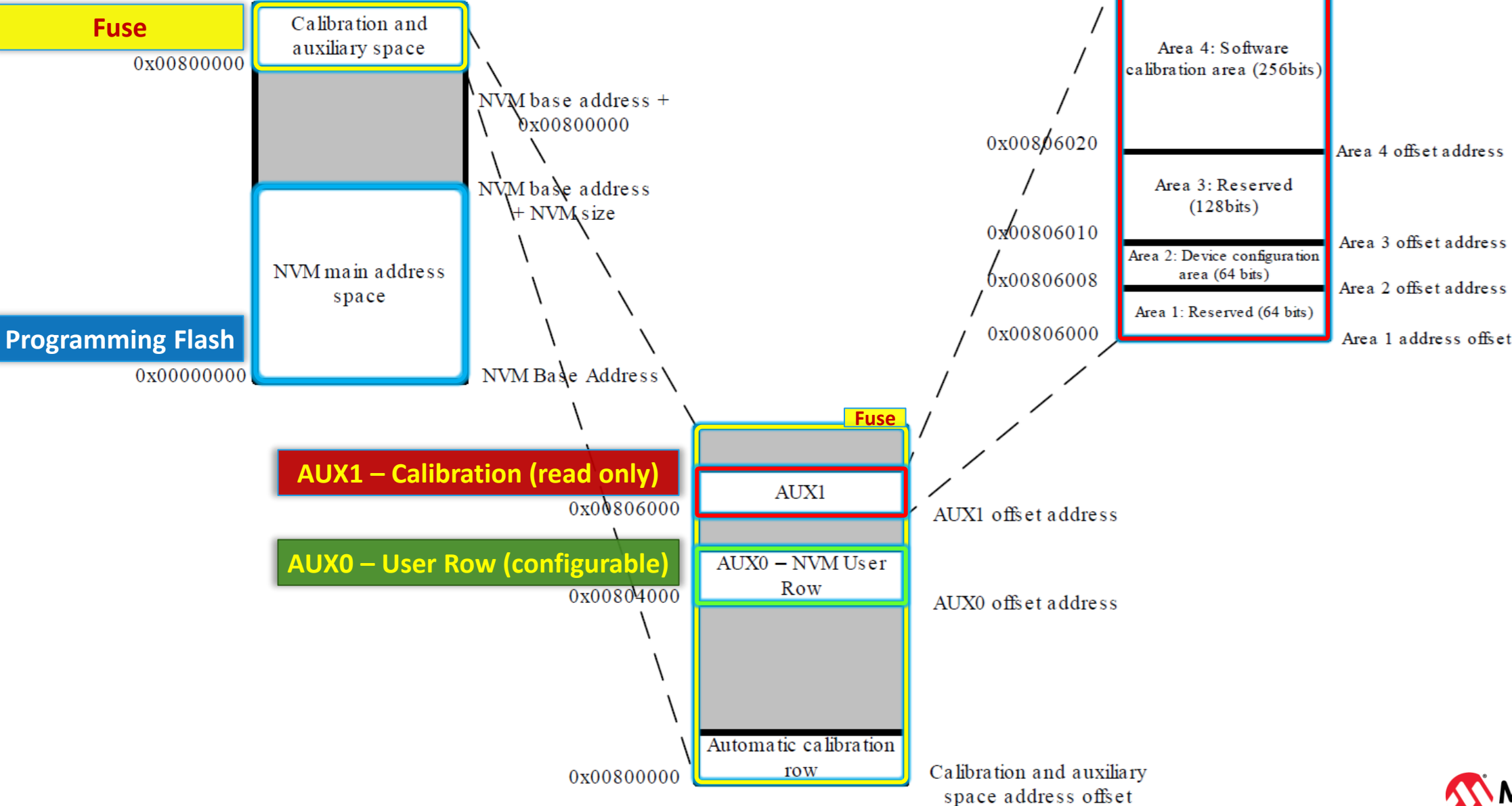
- Discuss

- SAM2001ADV Lab13-8 : Avg. **54.4mW**
 - **Active Mode.**
- SAM2001ADV Lab13-9 : Avg. **23.6mW**
 - **STANDBY mode.**
- SAM2001ADV Lab13-10 : Avg. **5.08mW**
 - **STANDBY mode, save DMAC action time.**



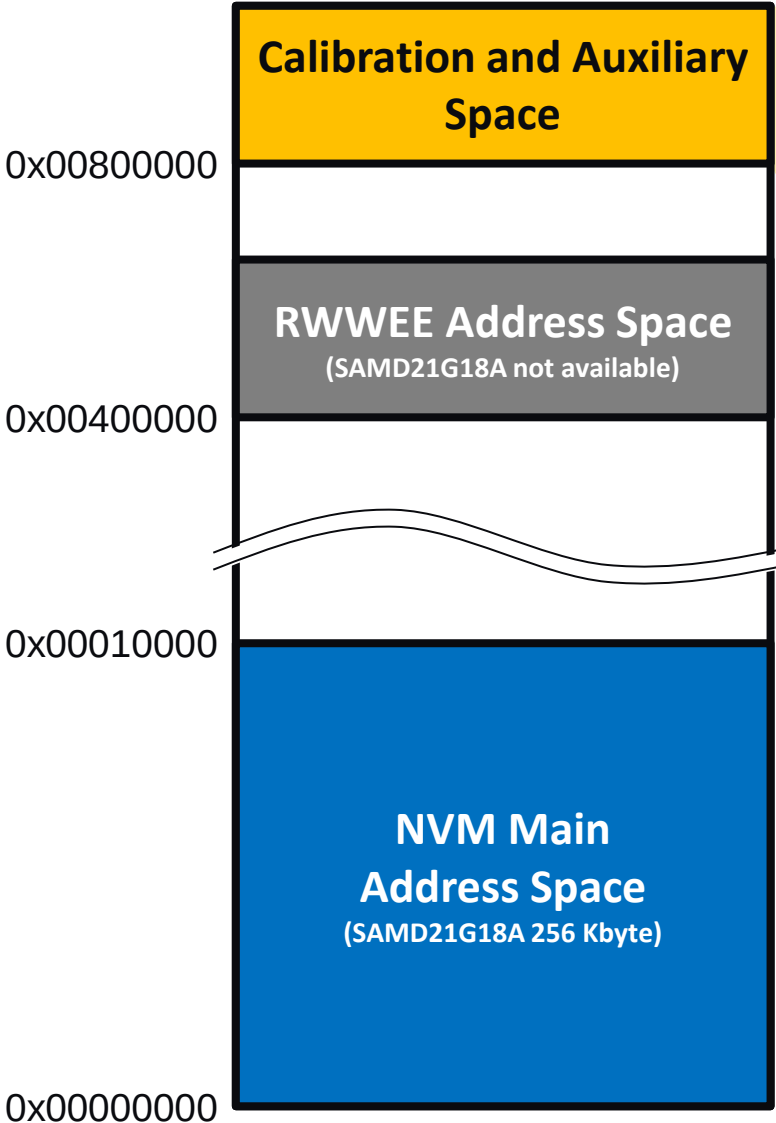
Auxiliary Space of NVM and Fuse Maintains

Auxiliary Space of NVM

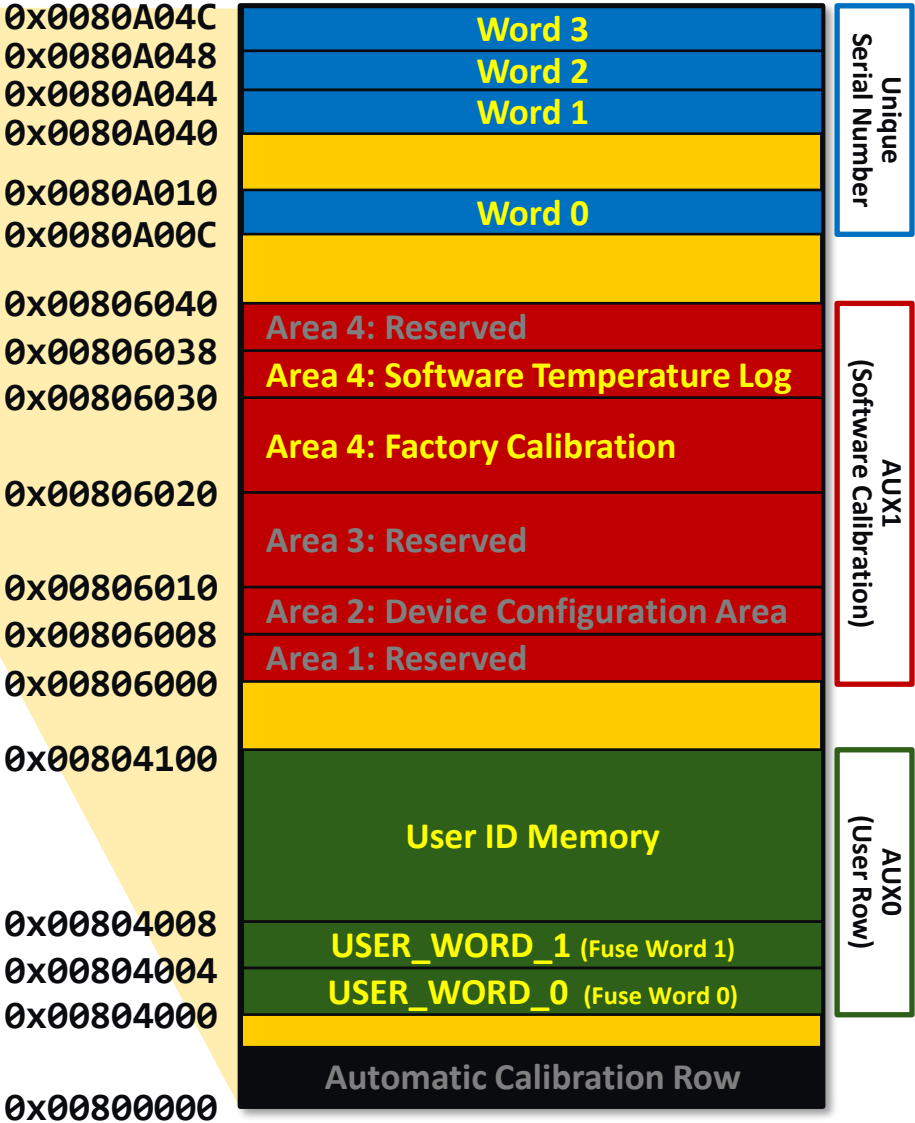


NVM Flash Memory Map

WORD Address

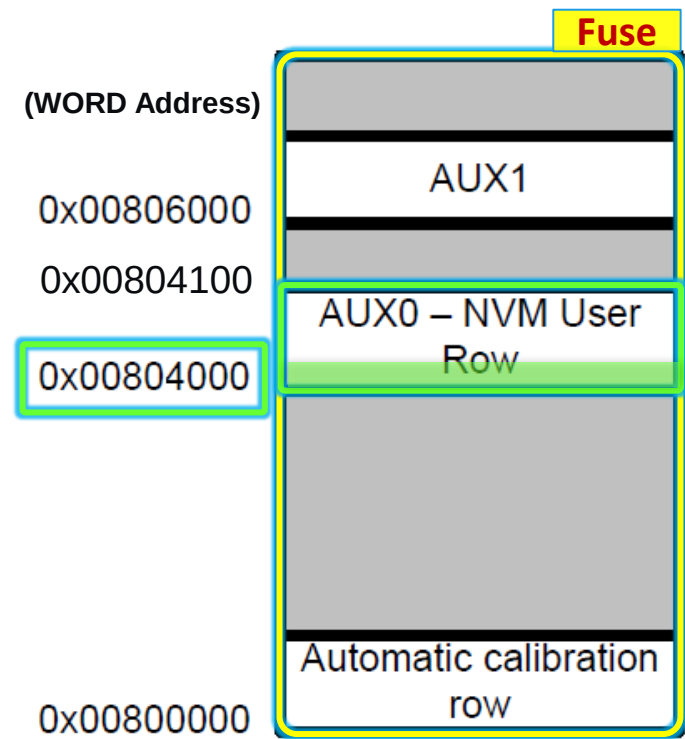


WORD Address



AUX0 (User Row)

User configurable Fuse settings



Bit Position	Name	Usage	
2:0	BOOTPROT	Used to select one of eight different bootloader sizes. Refer to “NVMCTRL – Non-Volatile Memory Controller”. Default value = 7 except for WLCSP (Default value = 3).	Boot Protect
3	Reserved		
6:4	EEPROM	Used to select one of eight different EEPROM Emulation sizes. Refer to “NVMCTRL – Non-Volatile Memory Controller”. Default value = 7.	EEPROM Emulation
7	Reserved		
13:8	BOD33 Level	BOD33 Threshold Level at power on. Refer to the SYSCtrl BOD33 register. Default value = 0x7 (non-AECQ100) Default value = 0x22 (AECQ100)	
14	BOD33 Enable	BOD33 enable at power on. Refer to the SYSCtrl BOD33 register. Default value = 1.	Brown Out Detect
16:15	BOD33 Action	BOD33 Action at power on. Refer to the SYSCtrl BOD33 register. Default value = 1.	
24:17	Reserved	Voltage Regulator Internal BOD (BOD12) configuration. These bits are written in production and must not be changed. Default value = 0x70.	
25	WDT Enable	WDT Enable at power on. Refer to the WDT CTRL register. Default value = 0.	
26	WDT Always-On	WDT Always-On at power on. Refer to the WDT CTRL register. Default value = 0.	
30:27	WDT Period	WDT Period at power on. Refer to the WDT CONFIG register. Default value = 0x0B.	
34:31	WDT Window	WDT Window mode time-out at power on. Refer to the WDT CONFIG register. Default value = 0x05.	Watch Dog Timer
38:35	WDT EWOFFSET	WDT Early Warning Interrupt Time Offset at power on. Refer to the WDT EWCTRL register. Default value = 0x0B.	
39	WDT WEN	WDT Timer Window Mode Enable at power on. Refer to the WDT CTRL register. Default value = 0.	
40	BOD33 Hysteresis	BOD33 Hysteresis configuration at power on. Refer to the SYSCtrl BOD33 register. Default value = 0.	
41	Reserved	Voltage Regulator Internal BOD(BOD12) configuration. This bit is written in production and must not be changed. Default value = 0.	
47:42	Reserved		
63:48	LOCK	NVM Region Lock Bits. Refer to “NVMCTRL – Non-Volatile Memory Controller”. Default value = 0xFFFF.	NVM Region Lock

AUX1 (Software Calibration - Factory Calibration)

Factory Calibration Value – read only

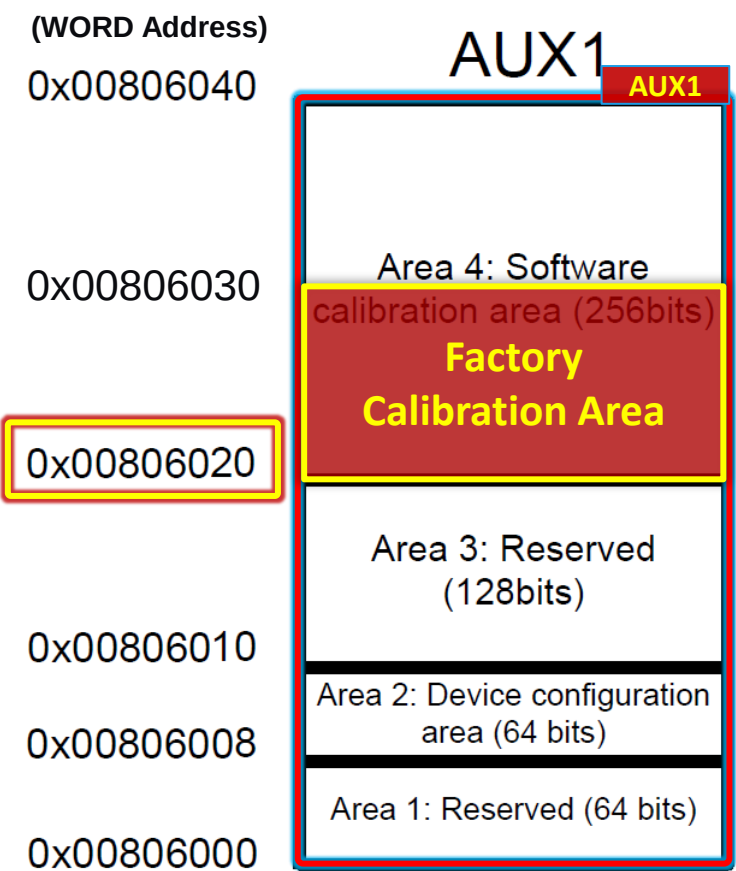


Table 10-5. NVM Software Calibration Area Mapping

Bit Position	Name	Description
2:0	Reserved	
14:3	Reserved	
26:15	Reserved	
34:27	ADC LINEARITY	ADC Linearity Calibration. Should be written to ADC CALIB register.
37:35	ADC BIASCAL	ADC Bias Calibration. Should be written to ADC CALIB register.
44:38	OSC32K CAL	OSC32KCalibration. Should be written to SYSCTRL OSC32K register.
49:45	USB TRANSN	USB TRANSN calibration value. Should be written to USB PADCAL register.
54:50	USB TRANSP	USB TRANSP calibration value. Should be written to USB PADCAL register.
57:55	USB TRIM	USB TRIM calibration value. Should be written to the USB PADCAL register.
63:58	DFLL48M CAL (COARSE and FINE)	DFLL48M Coarse calibration value. Should be written to SYSCTRL DFLLVAL register.
73:64		
127:74	Reserved	

AUX1 (Software Calibration - Factory Calibration)

Factory Calibration Value – read only

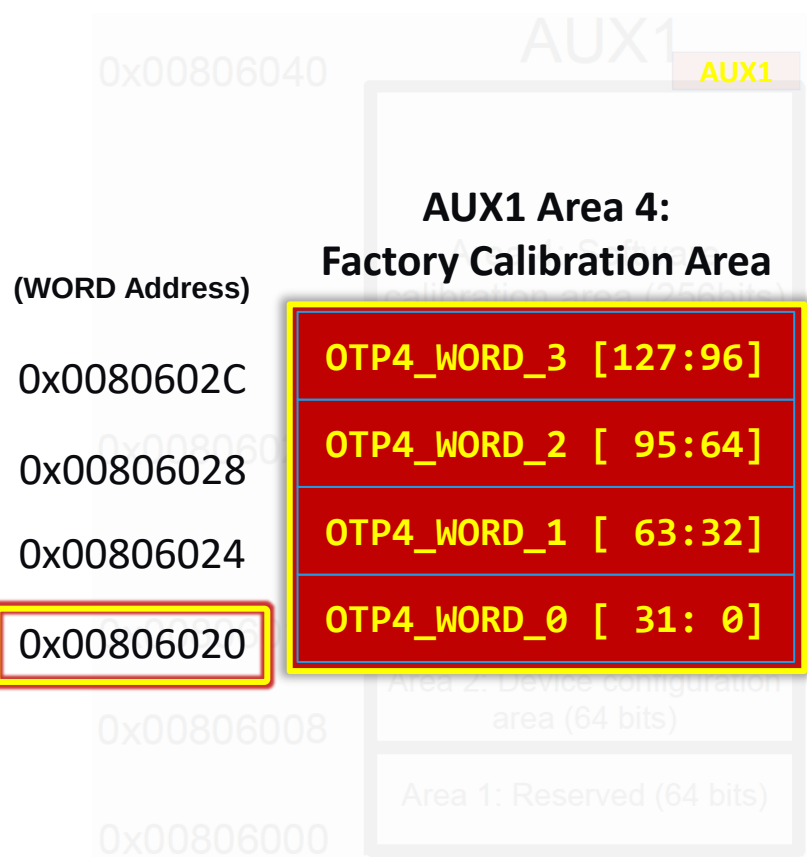


Table 10-5. NVM Software Calibration Area Mapping

Bit Position	Name	Description
2:0	Reserved	
14:3	Reserved	
26:15	Reserved	
34:27	ADC LINEARITY	ADC Linearity Calibration. Should be written to ADC CALIB register.
37:35	ADC BIASCAL	ADC Bias Calibration. Should be written to ADC CALIB register.
44:38	OSC32K CAL	OSC32KCalibration. Should be written to SYSCTRL OSC32K register.
49:45	USB TRANSN	USB TRANSN calibration value. Should be written to USB PADCAL register.
54:50	USB TRANSP	USB TRANSP calibration value. Should be written to USB PADCAL register.
57:55	USB TRIM	USB TRIM calibration value. Should be written to the USB PADCAL register.
63:58	DFLL48M CAL (COARSE and FINE)	DFLL48M Coarse calibration value. Should be written to SYSCTRL DFLLVAL register.
73:64		
127:74	Reserved	

AUX1 (Software Calibration - Factory Calibration)

Read and apply OSC32K Factory Calibration Value

- OSC Factory calibration value in **AUX1** will read and apply in **SYSCTRL_Initialize()** function of **plib_clock.c**

```
static void SYSCTRL_Initialize(void)
{
    /******* OSC32K Initialization *****/
    uint32_t calibValue = (uint32_t)((*(uint64_t*)0x806020U) >> 38U) & 0x7FU);

    /* Configure 32K RC oscillator */
    SYSCTRL_REGS->SYSCTRL_OSC32K =
        SYSCTRL_OSC32K_CALIB(calibValue) |
        SYSCTRL_OSC32K_STARTUP(0) |
        SYSCTRL_OSC32K_ENABLE_Msk |
        SYSCTRL_OSC32K_EN32K_Msk ;

    while( !( ( SYSCTRL_REGS->SYSCTRL_PCLKSR &
                SYSCTRL_PCLKSR_OSC32KRDY_Msk ) ==
                SYSCTRL_PCLKSR_OSC32KRDY_Msk ) )
    {
        /* Waiting for the OSC32K Ready state */
    }
    SYSCTRL_REGS->SYSCTRL_OSC32K = 0x0U;
}
```

Bit Position	Name
2:0	Reserved
14:3	Reserved
26:15	Reserved
34:27	ADC LINEARITY
37:35	ADC BIASCAL
44:38	OSC32K CAL
49:45	USB TRANSN
54:50	USB TRANSP
57:55	USB TRIM
63:58	DFLL48M COARSE CAL
73:64	Reserved
127:74	Reserved

AUX1 (Software Calibration - Factory Calibration)

Read and apply DFL48M Factory Calibration Value

- OSC Factory calibration value in **AUX1** will read and apply in **DFLL_Initialize()** function of **plib_clock.c**

```
static void DFLL_Initialize(void)
{
    /******* DFLL Initialization *****/
    ...

    /*Load DFLL48M CAL (COARSE) / OTP4_WORD_1 Address >> (58-32) [63:58] 6-bits*/
    uint32_t calibCoarse = (((*(uint32_t*)0x806024U) >> 26U) & 0x3FU);
    calibCoarse = (((calibCoarse) == 0x3FU) ? 0x1FU : (calibCoarse));

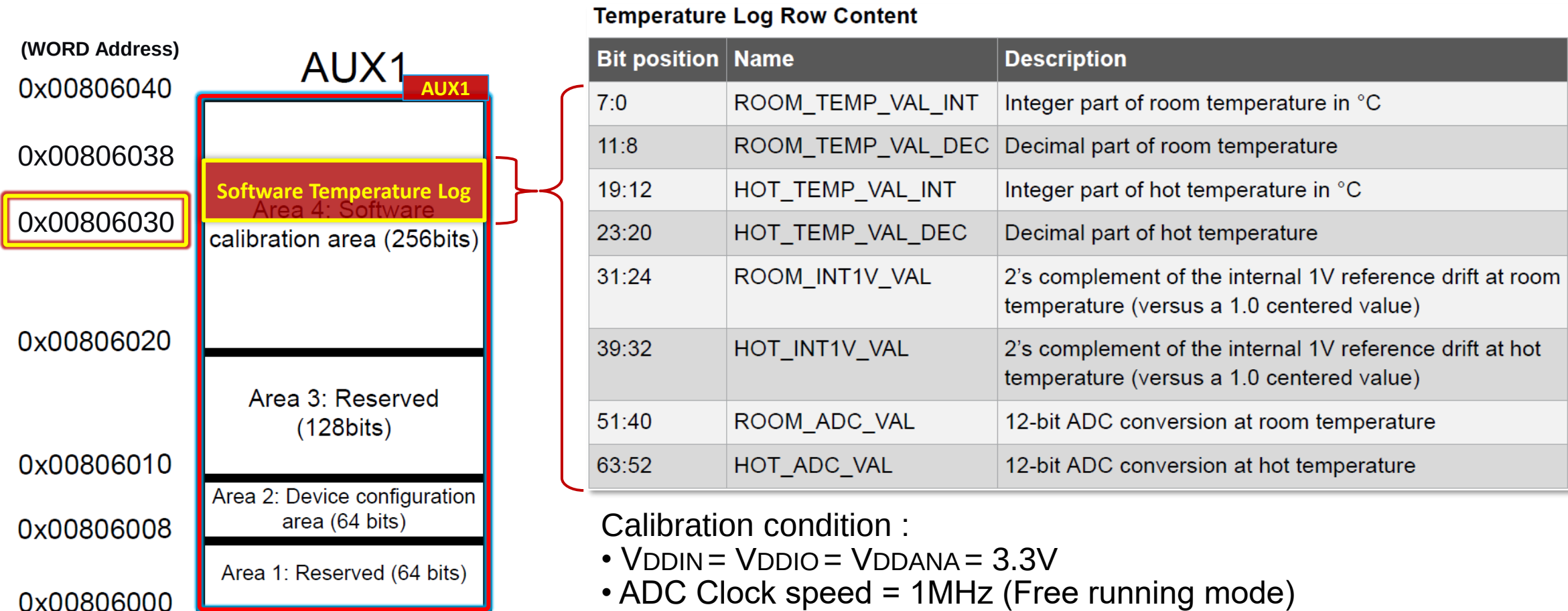
    SYSCTRL_REGS->SYSCTRL_DFLLVAL =
        SYSCTRL_DFLLVAL_COARSE(calibCoarse) |
        SYSCTRL_DFLLVAL_FINE(512U);

    GCLK_REGS->GCLK_CLKCTRL = GCLK_CLKCTRL_GEN(0x1U) |
        GCLK_CLKCTRL_CLKEN_Msk |
        GCLK_CLKCTRL_ID(0U);
    while((SYSCTRL_REGS->SYSCTRL_PCLKSR &
        SYSCTRL_PCLKSR_DFLLRDY_Msk) != SYSCTRL_PCLKSR_DFLLRDY_Msk)
    { ... }
}
```

Bit Position	Name
2:0	Reserved
14:3	Reserved
26:15	Reserved
34:27	ADC LINEARITY
37:35	ADC BIASCAL
44:38	OSC32K CAL
49:45	USB TRANSN
54:50	USB TRANSP
57:55	USB TRIM
63:58	DFLL48M CAL (COARSE and FINE)
73:64	
127:74	Reserved

AUX1 (Software Calibration - Software Temperature Log)

Software Temperature Log – Read Only



Learn this detail in SAM2001ADV chapter
[Internal Temperature Sensor](#)

Calibration condition :

- VDDIN = VDDIO = VDDANA = 3.3V
- ADC Clock speed = 1MHz (Free running mode)
- ADC averaging mode with 4 averaged samples
- ADC voltage reference = 1.0V (Internal reference INT1V)
- ADC input = Temperature sensor
- Room Temperature 25.2°C, • Hot temperature 83.3°C

AUX1 (Software Calibration - Software Temperature Log)

Temperature Log could be read and check in Microchip Studio



MPLAB® Snap (BUR190973931) - Device Programming

Tool: MPLAB® Snap | Device: ATSAM21G18A | Interface: SWD | Device signature: 0x10010305 | Target Voltage: 3.3 V

Interface settings | Tool information | Device information | Memories | **Fuses** | Security

Fuse Name	Value
✓ TEMP_LOG_WORD_0.ROOM_TEMP_VAL_INT	0x1E
✓ TEMP_LOG_WORD_0.ROOM_TEMP_VAL_DEC	0x01
✓ TEMP_LOG_WORD_0.HOT_TEMP_VAL_INT	0x7D
✓ TEMP_LOG_WORD_0.HOT_TEMP_VAL_DEC	0x00
✓ TEMP_LOG_WORD_0.ROOM_INT1V_VAL	0xFD
✓ TEMP_LOG_WORD_1.HOT_INT1V_VAL	0xF8
✓ TEMP_LOG_WORD_1.ROOM_ADC_VAL	0x0B22
✓ TEMP_LOG_WORD_1.HOT_ADC_VAL	0x0E91

Software Temperature Log

Fuse Register	Value
TEMP_LOG_WORD_0	0xFD07D11E
TEMP_LOG_WORD_1	0xE91B22F8

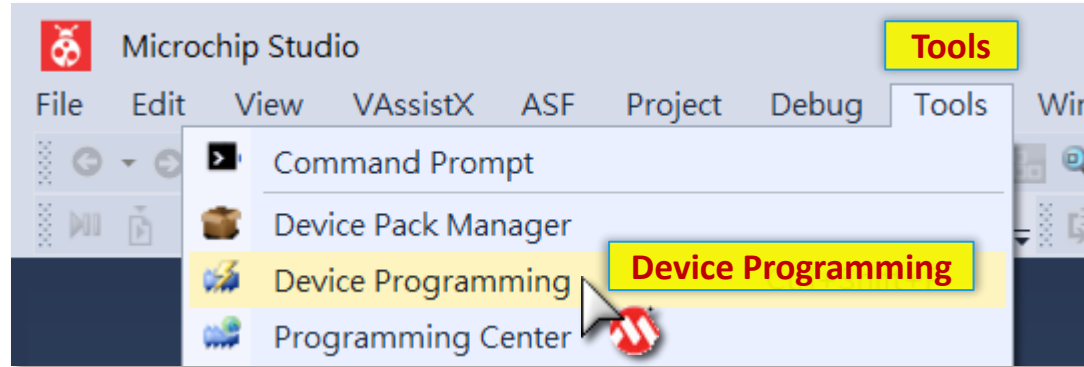
Learn this detail in SAM2001ADV chapter
[Internal Temperature Sensor](#)



Fuse (Configuration Bits) setting in Microchip Studio

Predecessor is Atmel Studio 7

Fuse (Configuration Bits) setting in Microchip Studio



Microchip Studio Fuses setting could see both **AUX0** and **AUX1**.

Open Tools

► Device Programming

Tool : MPLAB Snap

Device : ATSAM21G18A

Interface : SWD

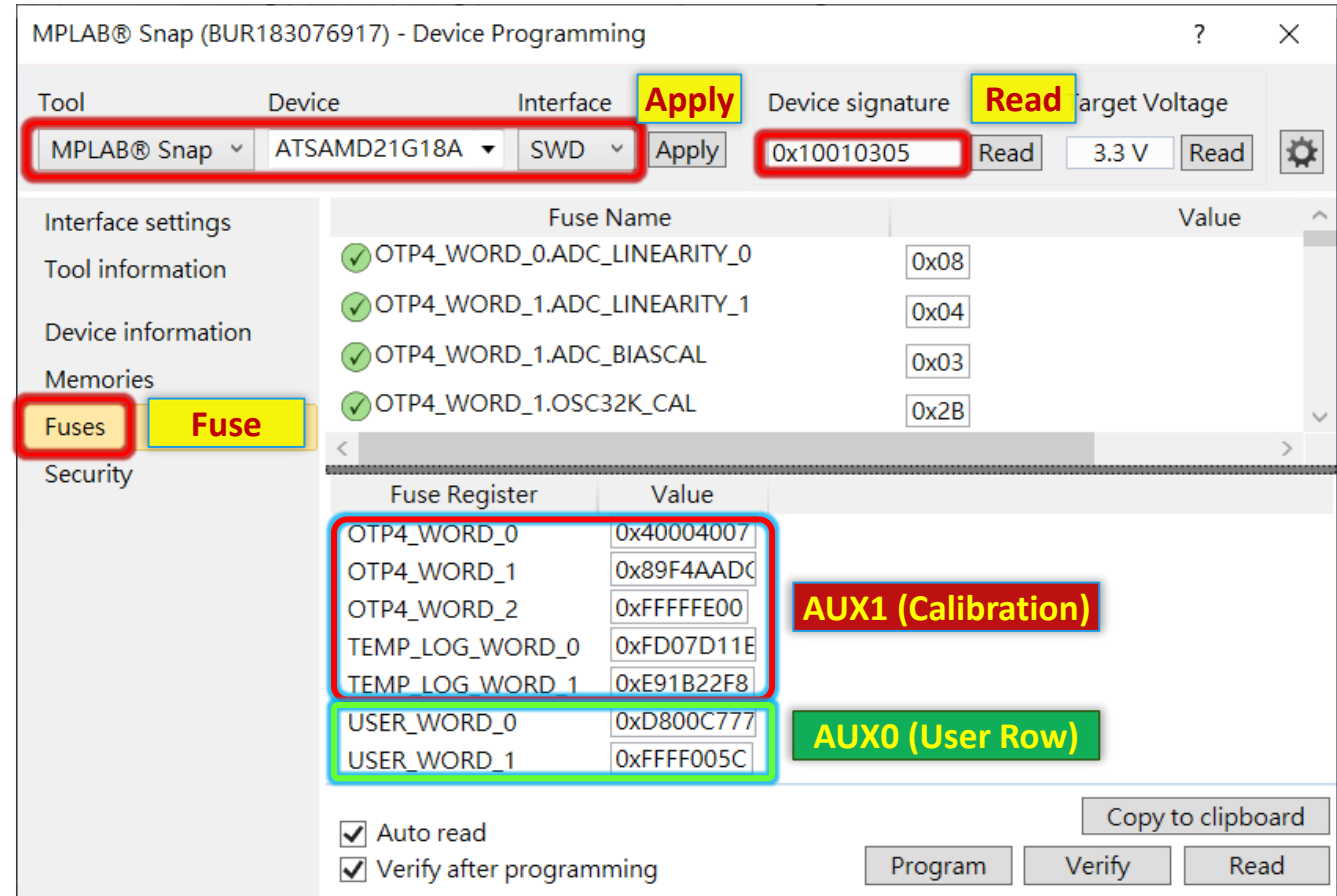
► [Apply]

SNAP will
Auto switch
to AVR mode

► [Read]

Device signature

► [Fuse]



Fuse (Configuration Bits) setting in Microchip Studio

Programming through Microchip Studio atprogram command line

- **Microchip Studio** has a standalone command line program **atprogram.exe** could programming HEX and Fuse settings. Please enter below directory (need Studio 7 pre-installed)

C:\Program Files (x86)\Atmel\Studio\7.0\atbackend

Ex1. Erase whole chip through EDBG's SWD interface.

atprogram -t snap -i swd -d atsamd21g18a erase

Ex2. Program HEX to MCU through EDBG's SWD interface.

atprogram -t snap -i swd -d atsamd21g18a program -f "Filename.hex"

Ex3. Program 0xD800C777 to USER_WORD_0 and 0xFFFF005C to USER_WORD_1 of AUX0(Fuse).

atprogram -t snap -i swd -d atsamd21g18a write -fs -o 0x804000 --values 77C700D85C00FFFF

Fuse Register	Value
OTP4_WORD_2	0xFFFFFE00
TEMP_LOG_WORD_0	0xFC25561F
TEMP_LOG_WORD_1	0xD35B2CFA
USER_WORD_0	0xD800C777
USER_WORD_1	0xFFFF005C

**atprogram now could support
Microchip Classic programmer
snap , pickit4**



Mind the data byte order !!

0xD800C777

0xFFFF005C

AUX0 (User Row)

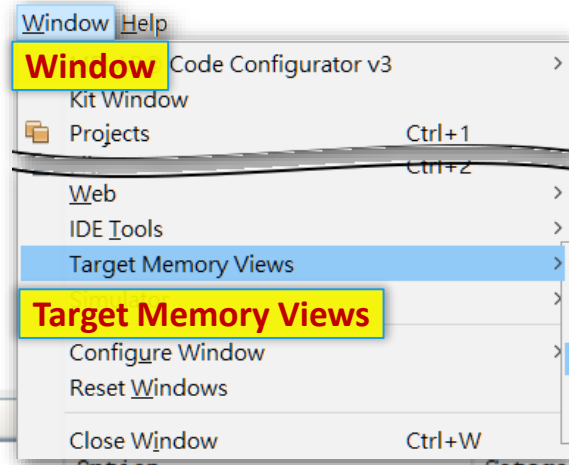
Configuration Bits (Fuse) setting in MPLAB X IDE

Configuration Bits (Fuse) setting in MPLAB X IDE



MPLAB Configuration Bits GUI

- Open Window
 - ▶ Target Memory Views
 - ▶ Configuration Bits



Configuration Bits will show Red if not to read it firstly

Output	Configuration Bits ×	Search Results	Notifications
	Read from chip		
		Value	Field
		0080_4000 USER_WORD_0 D8E0C7FF -	
		7	NVMCTRL_BOOTPROT SIZE_0BYTES
		7	NVMCTRL_EEPROM_SIZE SIZE_0BYTES
		7	BOD33USERLEVEL User range: 0x0 - 0x3F
		1	BOD33_EN ENABLED
		1	BOD33_ACTION RESET
		0	WDT_ENABLE DISABLED
		0	WDT_ALWAYS ON DISABLED
		B	WDT_PER CYC16384
		1	WDT_WINDOW_0 SET
		0080_4004 USER_WORD_1 FFFFFC5C -	
		4	WDT_WINDOW_1 User range: 0x0 - 0x7
		B	WDT_EWOFFSET CYC16384
		0	WDT_WEN DISABLED
		0	BOD33_HYST DISABLED
		FFFF	NVMCTRL_REGION_LOCKS User range: 0x0 - 0xFFFF

(Lab11-4_NVM_Fuse_ReadSN) - ARM Configuration Bits require a Read Device Memory step.

Must Read firstly before to configure

It is recommended that you read Configuration Bit settings on the ARM device before editing values. Note: Edit access is always enabled when the Simulator tool is selected.

Configuration Bits (Fuse) setting in MPLAB X IDE



MPLAB Configuration Bits GUI

- Open Window
 - ▶ Target Memory Views
 - ▶ Configuration Bits

MPLAB X IDE Configuration Bits
could see only **AUX0 (User Row)**

The screenshot shows the MPLAB X IDE interface. The 'Configuration Bits' window is open, displaying a table of configuration bits. The 'Target Memory Views' menu is open, showing the 'Configuration Bits' option. The 'Configuration Bits' window has a 'Program to chip' button and a 'Configure Fuse' button. The table shows the 'AUX0 (User Row)' and various configuration bits like 'NVMCTRL_BOOTPROT', 'BOD33_USERLEVEL', 'WDT_ENABLE', etc.

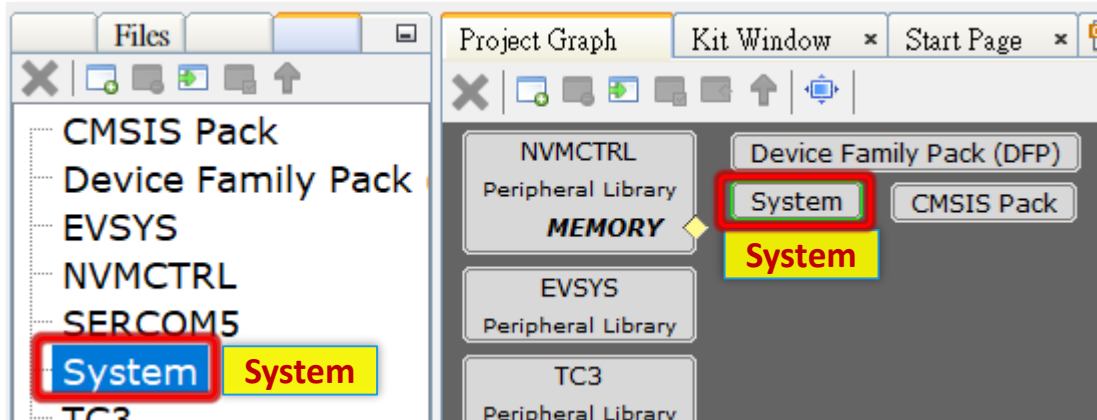
Address	Name	Value	Field	Option	Category	Setting
0080_4000	USER_WORD_0	D800C777	-	-	-	-
7	NVMCTRL_BOOTPROT	SIZE_0BYTES	Bootloader Size	0 Bytes		
7	NVMCTRL_EEPROM_SIZE	SIZE_0BYTES	EEPROM Size	0 Bytes		
7	BOD33_USERLEVEL	User range: 0x0 - 0x3F	BOD33 User Level	Enter Hexadecimal value		
1	BOD33_EN	ENABLED	BOD33 Enable	BOD33 is enabled		
1	BOD33_ACTION	RESET	BOD33 Action	The BOD33 generates a reset		
0	WDT_ENABLE	DISABLED	WDT Enable	WDT is disabled		
0	WDT_ALWAYS_ON	DISABLED	WDT Always On	WDT is enabled and disabled through ENABLE bit		
B	WDT_PER	ENABLED	WDT Period	16384 clock cycles		
1	WDT_WINDOW_0	SET	WDT Window bit 0	SET		
0080_4004	USER_WORD_1	FFFF005C	-	-	-	-
4	WDT_WINDOW_1	User range: 0x0 - 0x7	WDT Window bits 3:1	Enter Hexadecimal value		
B	WDT_EW_OFFSET	CYC16384	WDT Early Warning Offset	16384 clock cycles		
0	WDT_WEN	DISABLED	WDT Window Mode Enable	WDT is disabled		
0	BOD33_HYST	DISABLED	BOD33 Hysteresis	No Hysteresis		
FFFF	NVMCTRL_REGION_LOCKS	User range: 0x0 - 0xFFFF	NVM Region Locks	Enter Hexadecimal value		

Configuration Bits will show Black if success to read it before configure

Configuration Bits (Fuse) setting in MPLAB X IDE

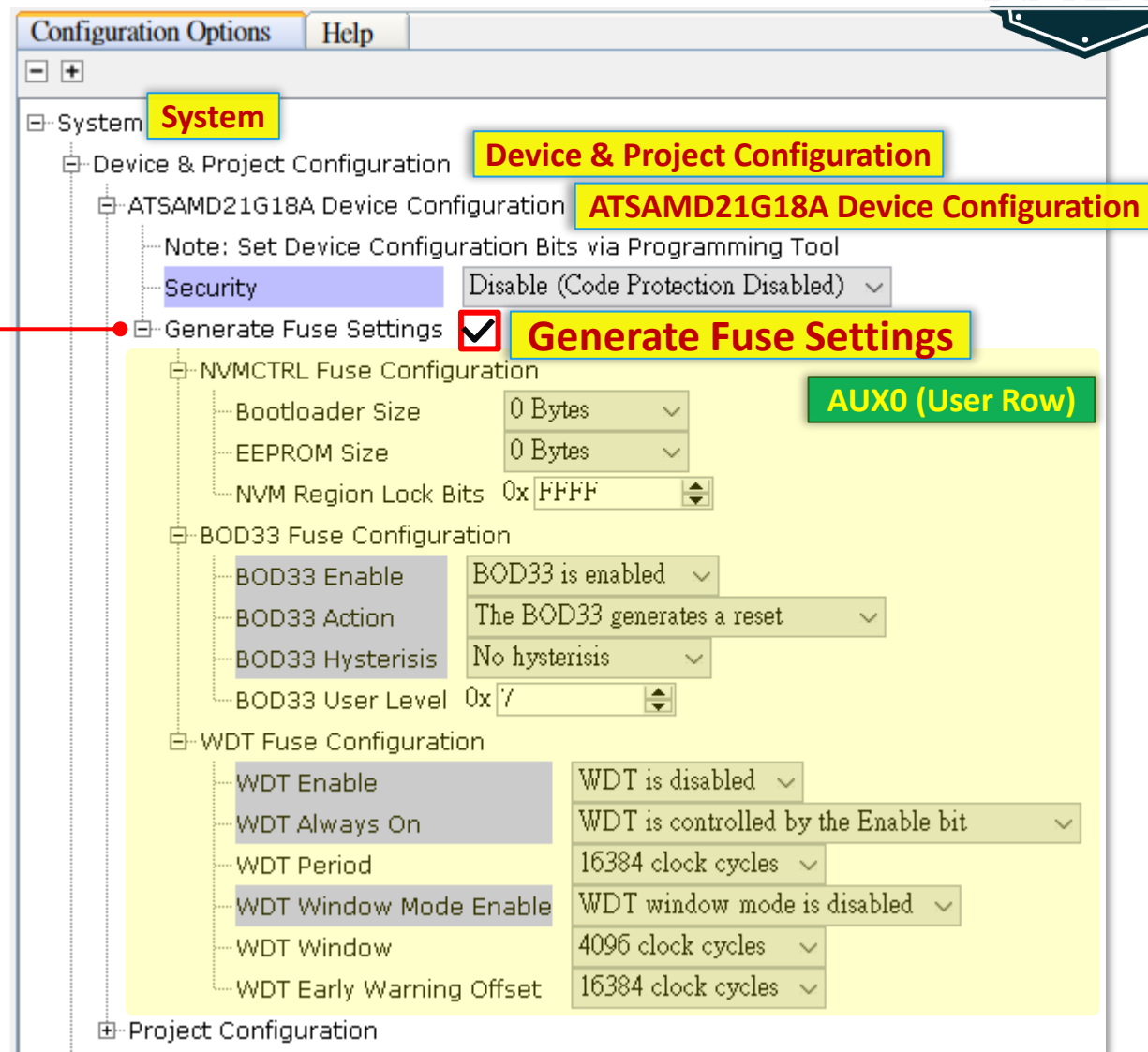


Generate Fuse Settings in Harmony



- Open System
 - ▶ Device & Project Configuration
 - ▶ ATSAM21G18A Device Configuration
 - ▶ Generate Fuse Settings ☒

Check this to include Fuse settings in HEX image
MPLAB X IDE and IPE supports support
program the Fuse setting with HEX image

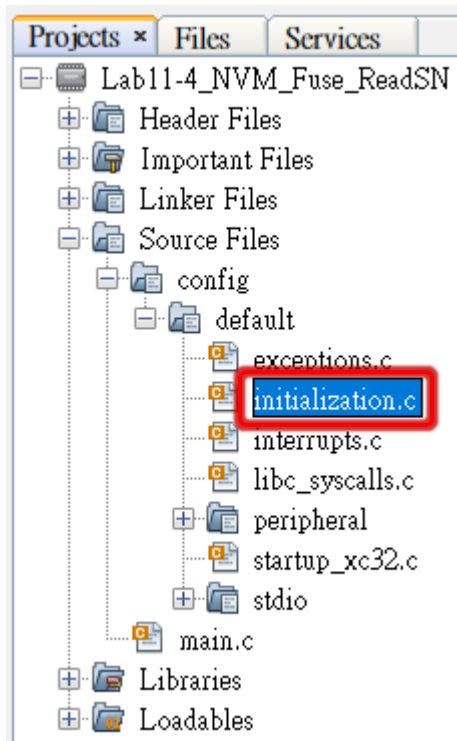


Configuration Bits (Fuse) setting in MPLAB X IDE



Generate Fuse Settings in Harmony

- Fuse setting will be generated as pre-define **#pragma** in **initialization.c**



```
// *****  
// Section: Configuration Bits  
// *****  
  
#pragma config NVMCTRL_BOOTPROT = SIZE_0BYTES  
#pragma config NVMCTRL_EEPROM_SIZE = SIZE_0BYTES  
#pragma config BOD33USERLEVEL = 0x7 // Enter Hexadecimal value  
#pragma config BOD33_EN = ENABLED  
#pragma config BOD33_ACTION = RESET  
  
#pragma config BOD33_HYST = DISABLED  
#pragma config NVMCTRL_REGION_LOCKS = 0xffff // Enter Hexadecimal value  
  
#pragma config WDT_ENABLE = DISABLED  
#pragma config WDT_ALWAYS_ON = DISABLED  
#pragma config WDT_PER = CYC16384  
  
#pragma config WDT_WINDOW_0 = SET  
#pragma config WDT_WINDOW_1 = 0x4 // Enter Hexadecimal value  
#pragma config WDT_EWOFFSET = CYC16384  
#pragma config WDT_WEN = DISABLED
```

AUX0 (User Row)

Configuration Bits (Fuse) setting in MPLAB X IDE



Generate Fuse Settings in Harmony

- Fuse settings will be appended in tail of **project's HEX image** if using **XC32 compiler**.

Generate Fuse Settings ☐

Project.hex

```
...  
:020000040000fa  
:020ad800704765  
:020000040000fa  
:020ada00fee735  
:020000040000fa  
:020adc00fee733  
:020000040000fa  
:020ade00fee731  
:00000001FF
```

Generate Fuse Settings ☒

Project.hex

```
...  
:020000040000fa  
:020ad800704765  
:020000040000fa  
:020ada00fee735  
:020000040000fa  
:020adc00fee733  
:020000040000fa  
:020ade00fee731  
:020000040000fa  
:0200000400807a  
:0440000077c700d8a6  
:020000040000fa  
:0200000400807a  
:044004005c00ffff5e  
:00000001FF
```

AUX0 (User Row) WORD 1

AUX0 (User Row) WORD 2

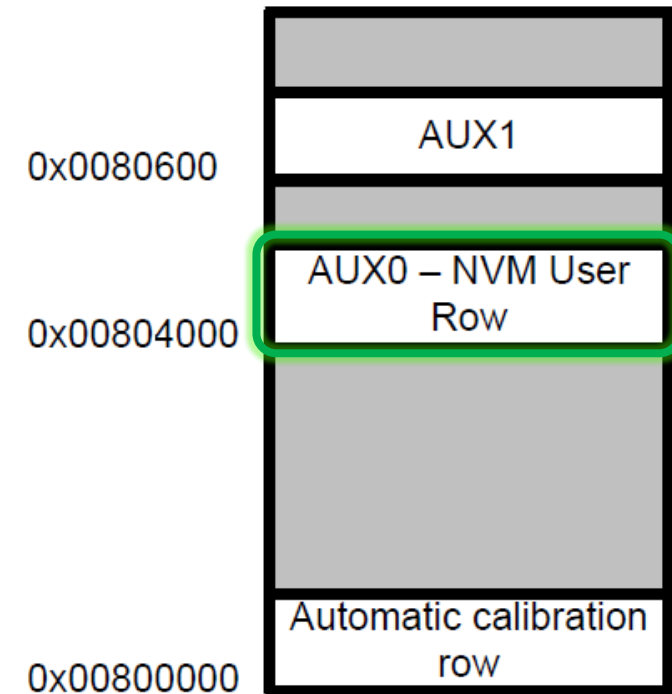
Configuration Bits (Fuse) setting in MPLAB X IDE



Bit format analysis of Fuse settings in HEX image

0440000077c700d8a6

04 : Four bytes
4000 : Starting address
00 : Record type
77c700d8 : Data payload
a6 : Checksum



Configuration Bits (Fuse) setting in MPLAB X IDE



Bit format analysis of Fuse settings in HEX image

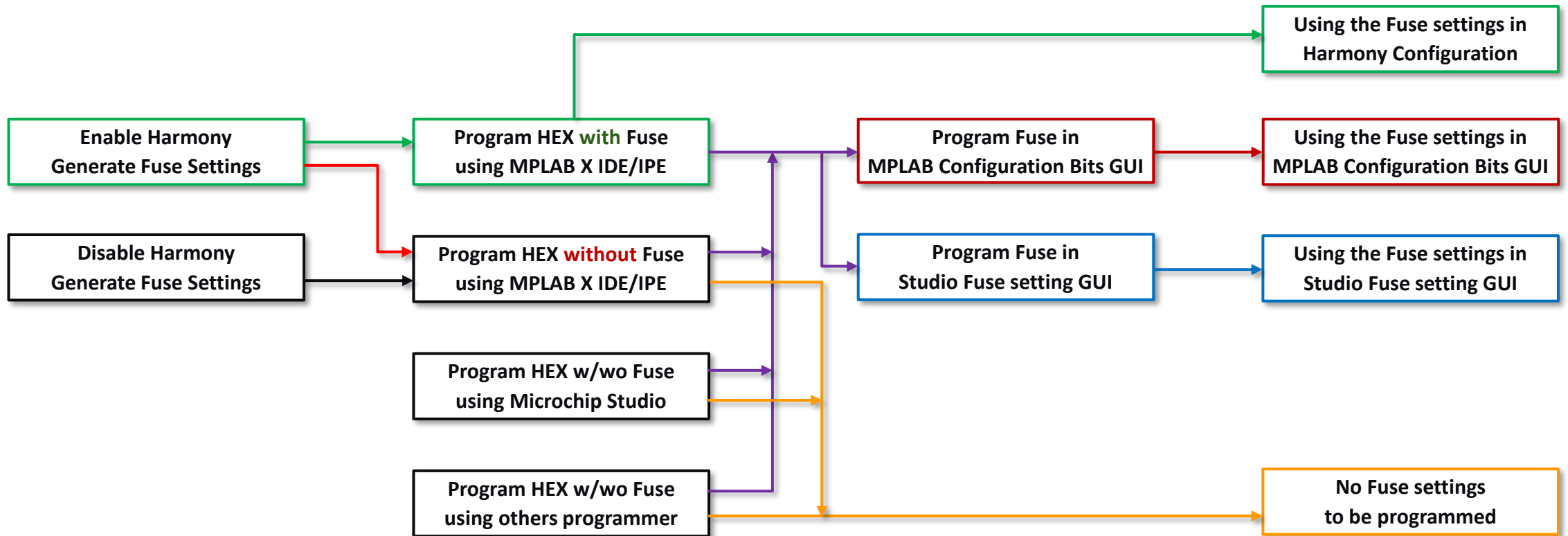
Bit Position	Name
2:0	BOOTPROT
3	Reserved
6:4	EEPROM
7	Reserved
13:8	BOD33 Level
14	BOD33 Enable
16:15	BOD33 Action
24:17	Reserved
25	WDT Enable
26	WDT Always-On
30:27	WDT Period
34:31	WDT Window
38:35	WDT EWOFFSET
39	WDT WEN
40	BOD33 Hysteresis
41	Reserved
47:42	Reserved
63:48	LOCK

BitPos	Name	Bit Mapping of HEX Data	User Row WORD 1
[2:0]	BOOTPORT	---- -XXX ----	----
[3]	reserved	----	----
[6:4]	EEPROM	-XXX ----	----
[7]	reserved	----	----
[13:8]	BOD33.UserLevel	---- --XX XXXX	----
[14]	BOD33.Enable	---- -X--	----
[16:15]	BOD33.Action	---- X---	----
[24:17]	reserved	----	----
[25]	WDT.Enable	----	---- --X-
[26]	WDT.AlwaysOn	----	---- -X--
[30:27]	WDT.Period	----	---- -XXX X---
[31]	WDT.Window	----	---- X----

BitPos	Name	Bit Mapping of HEX Data	User Row WORD 2
[34:32]	WDT.Window	---- -XXX ----	----
[38:35]	WDT.EWOFFSET	-XXX X---	----
[39]	WDT.WindowEnable	X---	----
[40]	BOD33.Hysteresis	---- ---X	----
[41]	reserved	----	----
[47:42]	reserved	----	----
[63:48]	LOCK	---- XXXX XXXX XXXX XXXX	XXXX XXXX XXXX XXXX

Configuration Bits (Fuse) setting in MPLAB X IDE

Configuration Bits GUI and Harmony Generated Fuse setting in HEX



Lab11-4 Read Unique Serial Number

- Please continue from [SAM2001 Lab11 \(SERCOM - UART TxRx Interrupt Callback with myprintf\)](#)
- In SAM D20 / SAM D21 / SAM R21 devices, each device has a unique 128-bit serial number which is a concatenation of four 32-bit words contained at the following addresses:
 - Word 0: 0x0080A00C
 - Word 1: 0x0080A040
 - Word 2: 0x0080A044
 - Word 3: 0x0080A048
- Try to read the unique serial number from your SAMD21 MCU and output as below
 - "Unique serial number : (4 Words)"
 - "0x0080A00C : 0x55574129"
 - "0x0080A010 : 0x514E5357"
 - "0x0080A014 : 0x4C202020"
 - "0x0080A018 : 0xFF130E33"
 - "- done"
- **Let's go !**

Lab11-4 Read Unique Serial Number

Step1

```
// TODO 11-4.01
uint32_t NVM_UQSN_Word[4];

void USART5_Transmit( uintptr_t context )
...

// TODO 11-4.02
void NVM_Read_UQSN( uint32_t *Word )
{
    volatile uint32_t *pWord, i;

    while( NVMCTRL_IsBusy() ) {}
    pWord = (volatile uint32_t *)0x0080A00C;
    myprintf( "Unique serial number : (4 Words)\r\n" );
    for( i=0 ; i<4 ; i++ )
    {
        Word[i] = *pWord;
        myprintf( "0x%08X : 0x%08X\r\n", 0x0080A00C+(i*4), Word[i] );
        if( i==0 ) pWord = (volatile uint32_t *)0x0080A040;
        else      pWord++;
    }
    myprintf( "- done\r\n\r\n" );
}

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

Lab11-4 Read Unique Serial Number

Step2

```
...
void myprintf(const char *format, ...) { ... };

// TODO 11-4.02
void NVM_Read_UQSN( uint32_t *Word )
{ ... }

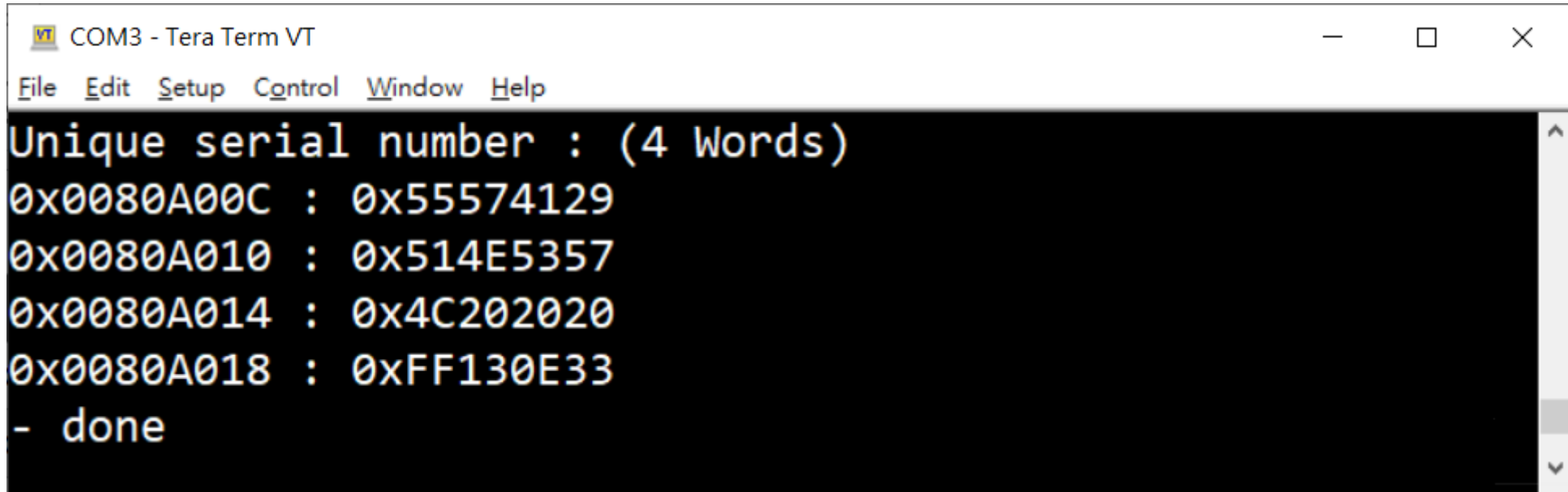
int main ( void )
{
    ...
    SERCOM5_USART_Read( USART5_ReceiveData, 1 );

    // TODO 11-4.03
    myprintf("\033[2J");
    myprintf("\033[?251\033[1;1H");
    NVM_Read_UQSN( NVM_UQSN_Word );

    while ( true )
    { ...
        if (TC3_HasExpired)
        { ...
            // TODO 11-4.04
            // myprintf( "Hello World!\r\n" );
        }
    }
}
```


✖ Lab11-4 Read Unique Serial Number

Result



The screenshot shows a terminal window titled "COM3 - Tera Term VT". The menu bar includes "File", "Edit", "Setup", "Control", "Window", and "Help". The terminal output displays the unique serial number of a device, consisting of four words. Each word is shown as a memory address followed by a colon and a hexadecimal value. The output ends with "- done".

```
Unique serial number : (4 Words)
0x0080A00C : 0x55574129
0x0080A010 : 0x514E5357
0x0080A014 : 0x4C202020
0x0080A018 : 0xFF130E33
- done
```

⚙️ Lab11-5 Read NVM AUX0 (User Row)

- Please continue from [SAM2001 Lab11-4 Read Unique Serial Number](#)
- Try to read the AUX0 (User Row) from NVM Fuse(Configuration Bit) which locate at address **0x00804000** and has total **64 WORDs** (256 Bytes).
- The **Fuse Settings** should be the first **two WORDs** of AUX0.
- Please output as below format :

“NVM-AUX0 User Row : (64 Words)”

“- Word 0 and Word 1 ”

“0x00804000 : 0xD800C777 ”

“0x00804004 : 0xFFFF005C ”

“- User ID Memory ”

“0x00804008 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF ”

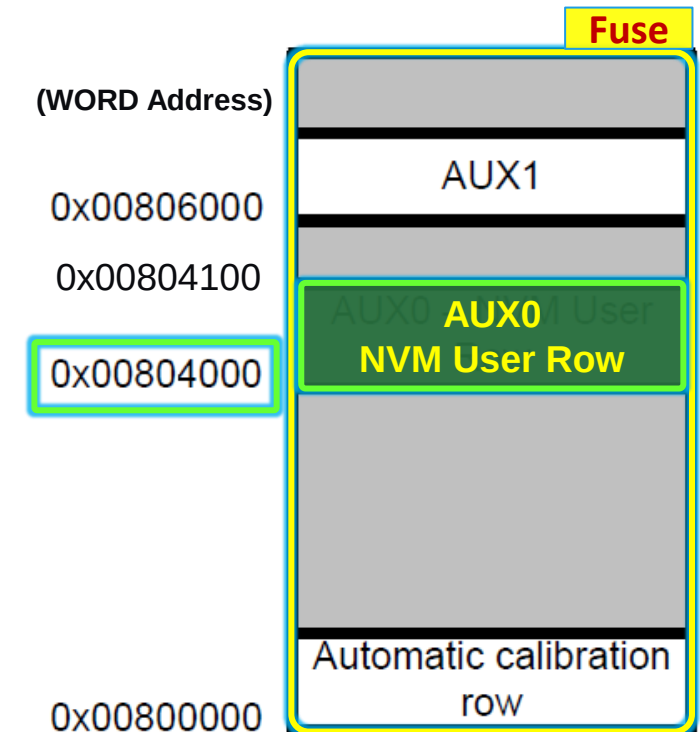
“0x00804018 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF ”

“0x00804028 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF ”

“ ”

“0x008040F8 : 0xFFFFFFFF 0xFFFFFFFF ”

“- done”



• **Let's go !**

⚙️ Lab11-5 Read NVM AUX0 (User Row)

Step1

```
uint32_t NVM_UQSN_Word[4];
// TODO 11-5.01
uint32_t NVM_AUX0_Word[64];
...
void NVM_Read_UQSN( uint32_t *Word )
// TODO 11-5.02
void NVM_Read_AUX0( uint32_t *Word )
{
    volatile uint32_t *pWord, i;

    while( NVMCTRL_IsBusy() ) {}
    pWord = (volatile uint32_t *)USER_PAGE_ADDR;
    myprintf( "NVM-AUX0 User Row : (64 Words)\r\n" );
    for( i=0 ; i<64 ; i++ )
    {
        Word[i] = *pWord;
        if( i==0 ) myprintf( "- Word 0 and Word 1\r\n" );
        if( i==2 ) myprintf( "- User ID Memory\r\n" );
        if( i<2 || (i-2+4)%4==0 ) myprintf( "0x%08X :", USER_PAGE_ADDR+(i*4) );
        myprintf( " 0x%08X", Word[i] );
        if( i<2 || (i-1)%4==0 ) myprintf( "\r\n" );
        pWord++;
    }
    myprintf( "\r\n- done\r\n\r\n" );
}
```

Lab11-5 Read NVM AUX0 (User Row)

Step2

```
uint32_t NVM_UQSN_Word[4];
// TODO 11-5.01
uint32_t NVM_AUX0_Word[64];
...
void NVM_Read_UQSN( uint32_t *Word )
// TODO 11-5.02
void NVM_Read_AUX0( uint32_t *Word )
{ ... }

int main ( void )
{
    ...
    SERCOM5_USART_Read( USART5_ReceiveData, 1 );

    myprintf("\033[2J");
    myprintf("\033[?25l\033[1;1H");
    NVM_Read_UQSN( NVM_UQSN_Word );
    // TODO 11-5.03
    NVM_Read_AUX0( NVM_AUX0_Word );

    while ( true )
    {
        ...
    }
}
```

✖ Lab11-5 Read NVM AUX0 (User Row)

Result

```
COM3 - Tera Term VT
File Edit Setup Control Window Help
NVM-AUX0 User Row : (64 Words)
- Word 0 and Word 1
0x00804000 : 0xD800C777
0x00804004 : 0xFFFF005C
- User ID Memory
0x00804008 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804010 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804018 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804020 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804028 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804030 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804038 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804040 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804048 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804050 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804058 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804060 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804068 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804070 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804078 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804080 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804088 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804090 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804098 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040A0 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040A8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040B0 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040B8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040C0 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040C8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040D0 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040D8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040E0 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040E8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040F0 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040F8 : 0xFFFFFFFF 0xFFFFFFFF
- done
```

⚙️ Lab11-6 Read NVM AUX1 (Software Calibration)

- Please continue from [SAM2001 Lab11-5 Read NVM AUX0 \(User Row\)](#)
- Try to read the AUX1 (Software Calibration) from NVM Fuse(Configuration Bit) which locate at address **0x00806000** and has total **16 WORDs** (64 Bytes).
- The **Factory Calibration** should in **OTP Area 4** and **Software Temperature Log** of AUX1.
- Please output as below format :

“NVM-AUX1 : (16 Words) ”

“- OTP Area 1 (Reserved) ”

“- OTP Area 2 (Device Configuration Area) ”

“- OTP Area 3 (Reserved) ”

“- OTP Area 4 (Software Calibration Area) ”

“0x00806020 : 0x40004007”

“ ”

“ > Software Temperature Log row”

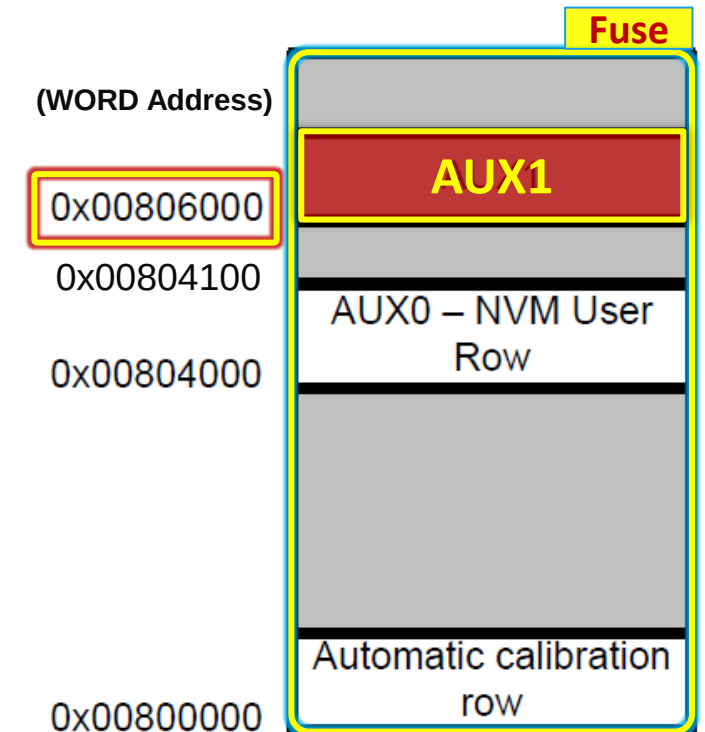
“0x00806030 : 0xFD07D11E”

“ ”

“ > Reserved”

“- done”

• **Let's go !**



Lab11-6 Read NVM AUX1 (Software Calibration)

Step1

```
...
uint32_t NVM_UQSN_Word[4];
uint32_t NVM_AUX0_Word[64];
// TODO 11-6.01
uint32_t NVM_AUX1_Word[16];
...
void NVM_Read_UQSN( uint32_t *Word )
{
    ...
}

void NVM_Read_AUX0( uint32_t *Word )
{
    ...
}

int main ( void )
{
    ...
    myprintf("\033[2J");
    myprintf("\033[?25l\033[1;1H");
    NVM_Read_UQSN( NVM_UQSN_Word );
    NVM_Read_AUX0( NVM_AUX0_Word );
    ...
}
```

⚙️ Lab11-6 Read NVM AUX1 (Software Calibration)

Step1

```
// TODO 11-6.01
uint32_t NVM_AUX1_Word[16];
...
// TODO 11-6.02
void NVM_Read_AUX1( uint32_t *Word )
{
    volatile uint32_t *pWord, i;

    while( NVMCTRL_IsBusy() ) {}
    pWord = (volatile uint32_t *)OTP1_ADDR;
    myprintf( "NVM-AUX1 : (16 Words)\r\n" );
    for( i=0 ; i<16 ; i++ )
    {
        Word[i] = *pWord;
        if( i== 0 ) myprintf( "- OTP Area 1 (Reserved)\r\n" );
        if( i== 2 ) myprintf( "- OTP Area 2 (Device Configuration Area)\r\n" );
        if( i== 4 ) myprintf( "- OTP Area 3 (Reserved)\r\n" );
        if( i== 8 ) myprintf( "- OTP Area 4 (Software Calibration Area)\r\n" );
        if( i==12 ) myprintf( " > Software Temperature Log row\r\n" );
        if( i==14 ) myprintf( " > Reserved\r\n" );
        myprintf( "0x%08X : 0x%08X\r\n", OTP1_ADDR+(i*4), Word[i] );
        pWord++;
    }
    myprintf( "- done\r\n\r\n" );
}
```


Lab11-6 Read NVM AUX1 (Software Calibration)

Step2

```
// TODO 11-6.01
uint32_t NVM_AUX1_Word[16];
...
// TODO 11-6.02
void NVM_Read_AUX1( uint32_t *Word )
{
    ...
}

int main ( void )
{
    ...

    myprintf("\033[2J");
    myprintf("\033[?25l\033[1;1H");
    NVM_Read_UQSN( NVM_UQSN_Word );
    NVM_Read_AUX0( NVM_AUX0_Word );
    // TODO 11-6.03
    NVM_Read_AUX1( NVM_AUX1_Word );

    while ( true )
    {
        ...
    }
}
```

⚙️ Lab11-6 Read NVM AUX1 (Software Calibration)

Result

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

NVM-AUX1 : (16 Words)
- OTP Area 1 (Reserved)
0x00806000 : 0x10000300
0x00806004 : 0x00000000
- OTP Area 2 (Device Configuration)
0x00806008 : 0x007DEB65
0x0080600C : 0x02200000
- OTP Area 3 (Reserved)
0x00806010 : 0xFFFFFFFF
0x00806014 : 0xFFFFFFFF
0x00806018 : 0xFFFFFFFF
0x0080601C : 0xFFFFFFFF
- OTP Area 4 (Software Calibration Area)
0x00806020 : 0x40004007
0x00806024 : 0x89F4AADC
0x00806028 : 0xFFFFFE00
0x0080602C : 0xFFFFFFFF
  > Software Temperature Log row
0x00806030 : 0xFD07D11E
0x00806034 : 0xE91B22F8
  > Reserved
0x00806038 : 0xFFFFFE795
0x0080603C : 0xFFFFFFFF
- done
```

User ID Memory in NVM AUX0 (User Row)

User ID Memory in NVM AUX0 (User Row)

User Fuse WORD 0, WORD1 and User ID Memory

NVM-AUX0 User Row : (64 Words)

- Word 0 and Word 1

0x00804000 : 0xD800C777

0x00804004 : 0xFFFF005C

- User ID Memory

0x00804008 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF

0x00804018 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF

0x00804028 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF

0x00804038 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF

0x00804048 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF

0x00804058 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF

0x00804068 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF

0x00804078 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF

0x00804088 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF

0x00804098 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF

0x008040A8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF

0x008040B8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF

0x008040C8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF

0x008040D8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF

0x008040E8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF

0x008040F8 : 0xFFFFFFFF 0xFFFFFFFF

- done

0x00804100

User ID Memory
(64 - 2) WORDs
248 Bytes

Fuse WORD 1

Fuse WORD 0

0x00804008

0x00804004

0x00804000

(WORD Address)

0x00806000

0x00804100

0x00804000

0x00800000

Fuse

AUX1

AUX0 AUX0 User
NVM User Row

Automatic calibration
row

User ID Memory in NVM AUX0 (User Row)

User Microchip Studio to access User ID Memory



MPLAB® Snap (BUR183076917) - Device Programming

Tool: MPLAB® Snap | Device: ATSAM21G18A | Interface: SWD | Apply

Interface settings
Tool information
Device information
Memories
Fuses
Security

Device: Erase Chip | Erase now

Flash (256 KB)

☐ Erase Flash before programming
☐ Verify Flash after programming
Advanced

User Page (256 bytes)

☒ Erase User Page before programming
☒ Verify User Page after programming
Advanced

Program | Verify | Read...

Save As

本機 > 桌面

組合管理 | 新增資料夾

名稱: Document.userpage

檔案名稱(N): Document.userpage | 存檔類型(T): User Page (.userpage) (*.userpage)

存檔(S) | 取消

Read

User ID Memory in NVM AUX0 (User Row)

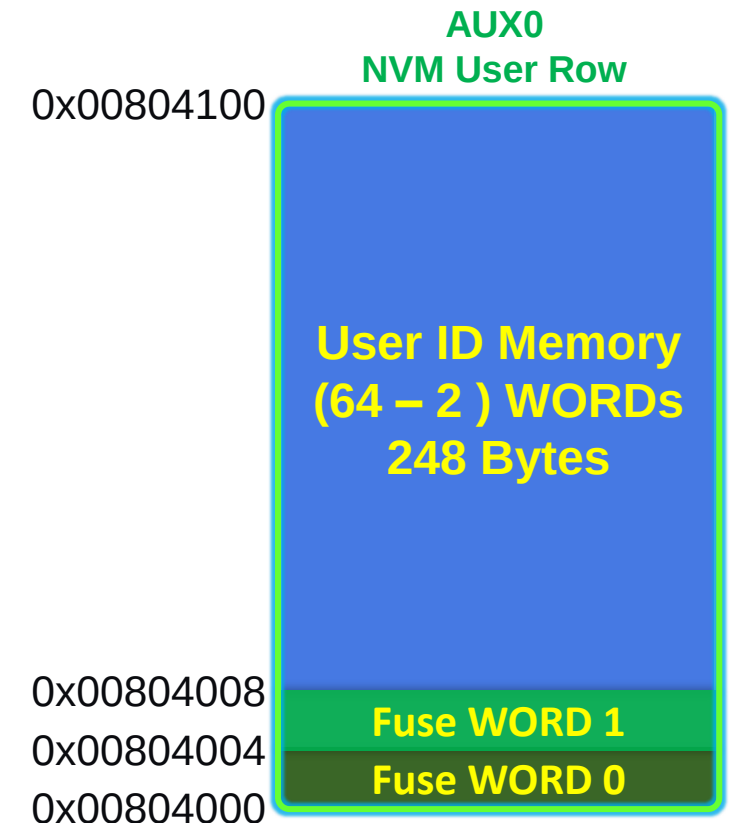
User Microchip Studio to access User ID Memory



Document.userpage

```
:0200000400807A
:1040000077C700D85C00FFFFFFFFFFF48
:10401000FFFFFFFFFFFFFFFFFFFFFFFFFBB0
:10402000FFFFFFFFFFFFFFFFFFFFFFFFFA0
:10403000FFFFFFFFFFFFFFFFFFFFFFFFF90
:10404000FFFFFFFFFFFFFFFFFFFFFFFFF80
:10405000FFFFFFFFFFFFFFFFFFFFFFFFF70
:10406000FFFFFFFFFFFFFFFFFFFFFFFFF60
:10407000FFFFFFFFFFFFFFFFFFFFFFFFF50
:10408000FFFFFFFFFFFFFFFFFFFFFFFFF40
:10409000FFFFFFFFFFFFFFFFFFFFFFFFF30
:1040A000FFFFFFFFFFFFFFFFFFFFFFFFF20
:1040B000FFFFFFFFFFFFFFFFFFFFFFFFF10
:1040C000FFFFFFFFFFFFFFFFFFFFFFFFF00
:1040D000FFFFFFFFFFFFFFFFFFFFFFFFFF0
:1040E000FFFFFFFFFFFFFFFFFFFFFFFFFE0
:1040F000FFFFFFFFFFFFFFFFFFFFFFFFFD0
:00000001FF
```

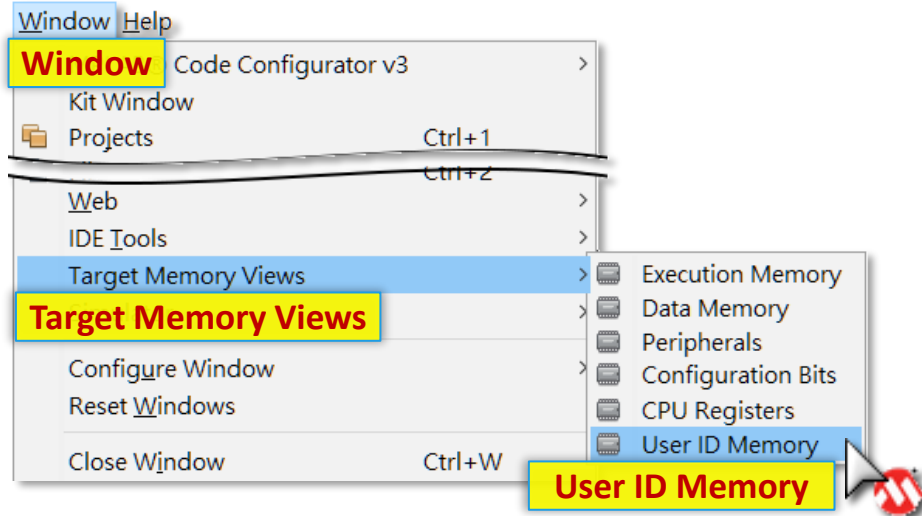
Microchip Studio IDE to store Fuse and User ID Memory together



User ID Memory in NVM AUX0 (User Row)



User MPLAB X IDE to access User ID Memory



User ID Memory × Output	
Address	User ID
0080_4008	0xFFFFFFFF
0080_400C	0xFFFFFFFF
0080_4010	0xFFFFFFFF
0080_4014	0xFFFFFFFF
0080_4018	0xFFFFFFFF
0080_401C	0xFFFFFFFF
0080_4020	0xFFFFFFFF
0080_4024	0xFFFFFFFF
0080_4028	0xFFFFFFFF
0080_402C	0xFFFFFFFF
0080_4030	0xFFFFFFFF
0080_4034	0xFFFFFFFF
0080_4038	0xFFFFFFFF
0080_403C	0xFFFFFFFF
0080_4040	0xFFFFFFFF
0080_4044	0xFFFFFFFF
0080_4048	0xFFFFFFFF
0080_404C	0xFFFFFFFF

Memory User ID Memory ▾
User ID Memory

MPLAB X IDE separate to shows
User ID Memory in GUI

AUX0
NVM User Row

0x00804100

User ID Memory
(64 – 2) WORDs
248 Bytes

0x00804008

0x00804004

0x00804000

Fuse WORD 1

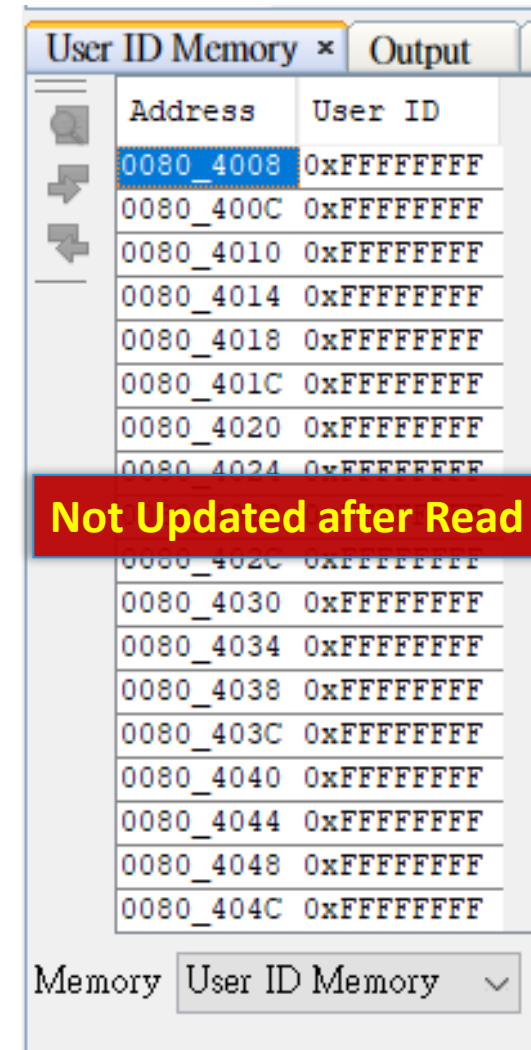
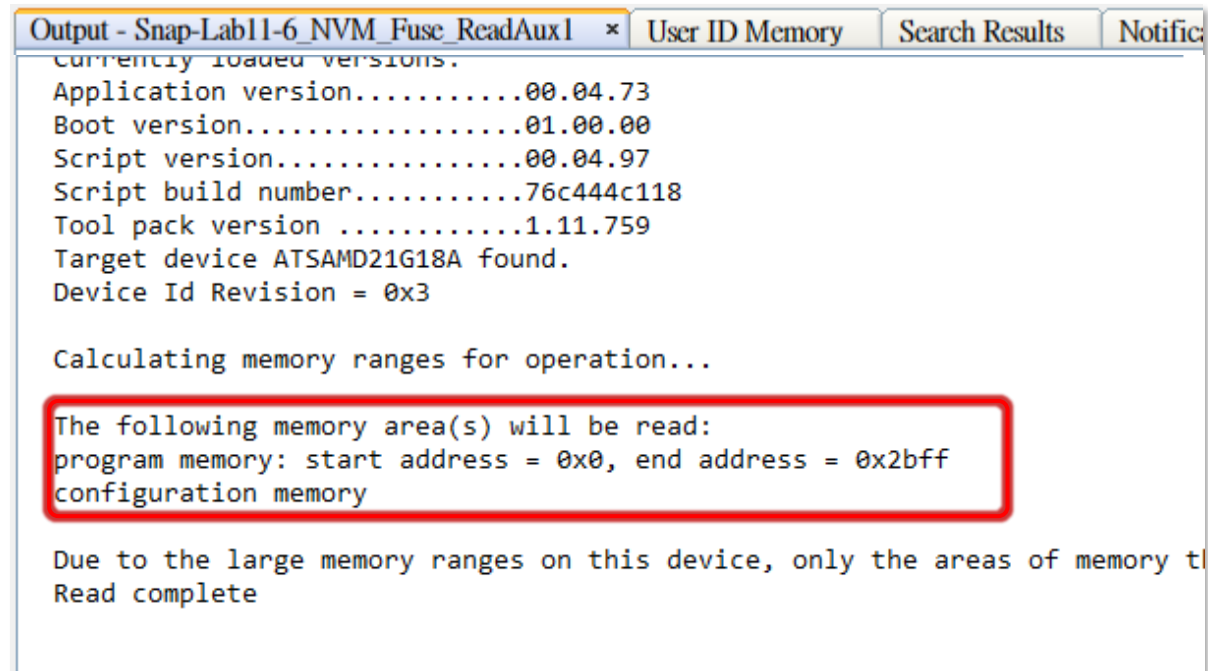
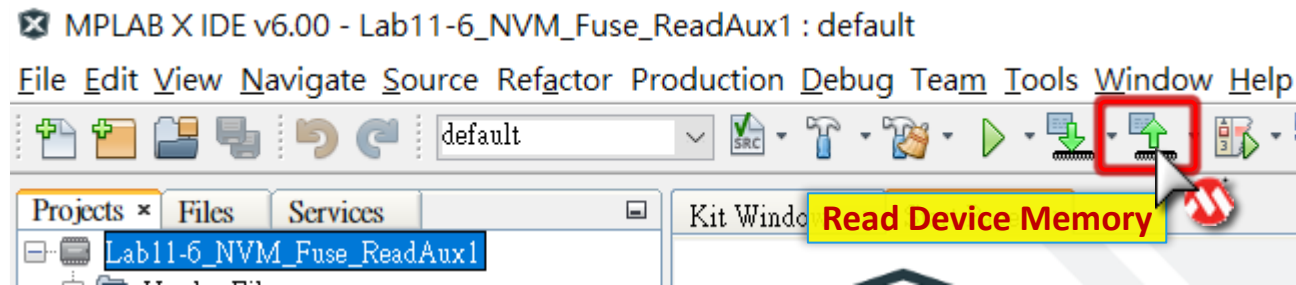
Fuse WORD 0

We cannot manually read and
write User ID Memory here

User ID Memory in NVM AUX0 (User Row)



Read User ID Memory in MPLAB X IDE



User ID Memory in NVM AUX0 (User Row)



Read User ID Memory in MPLAB X IDE

MPLAB X IDE v6.00 - Lab11-6_NVM_Fuse_ReadAux1 : default

File Edit View Navigate Source Refactor Production Debug Tools

default

Projects Files Services Kit Window

Lab11-6_NVM_Fuse_ReadAux1

Header Files
Important Files
Linker Files
Source Files
Libraries
Loadables

Navigator -6_NVM_Fuse_R

Lab11-6_NVM_Fuse_Re

Project Type: Applica

Device

ATSAMD21G18A

Checksum: Blank,

32 CRC32: Hex file t

Packs

SAMD21_DFP (3

CMSIS (5.8.0)

Compiler Toolchain

XC32 (v4.10) [C:

Production Image:

Properties

Properties

Project Properties - Lab11-6_NVM_Fuse_ReadAux1

Categories:

- General
- File Inclusion/Exclusion
- Conf: [default]
- Snap**
- Building
- XC32 (Global Options)
 - xc32-as
 - xc32-gcc
 - xc32-g++
 - xc32-ld
 - xc32-ar
 - Analysis

Debugger SNAP

Options for Snap

Memory to Program

Option categories: Memories to Program

Manual select memories and range

Auto select memories and ranges	Manually select memories and ranges
Configuration Memory	☒
ID	☒
Program Memory	☒
Program Memory Range(s)(hex)	0-3ffff
Preserve Program Memory	☐
Preserve Program Memory Range(s)(hex)	

Check ID

Option Description

Manage Configurations...

Manage Network Tools...

OK

Cancel

Apply

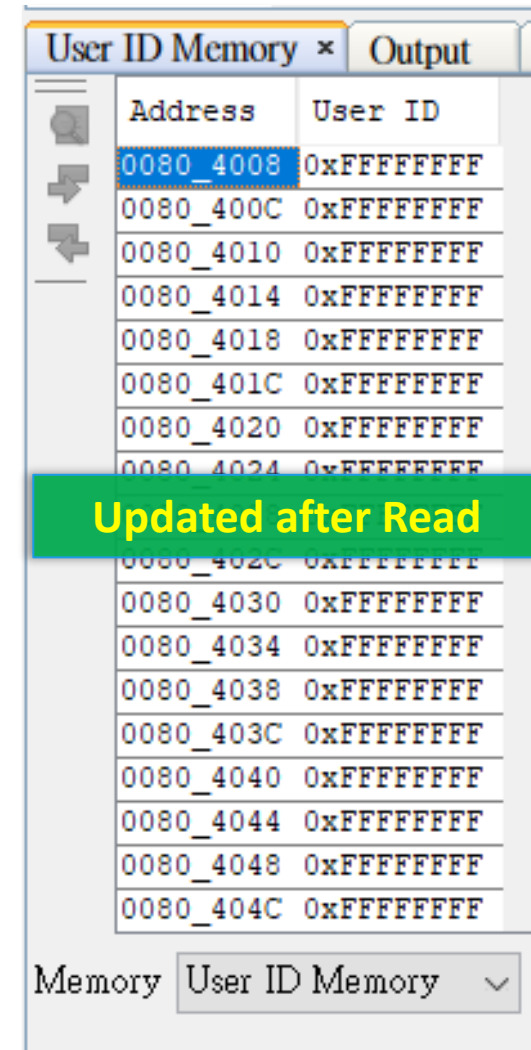
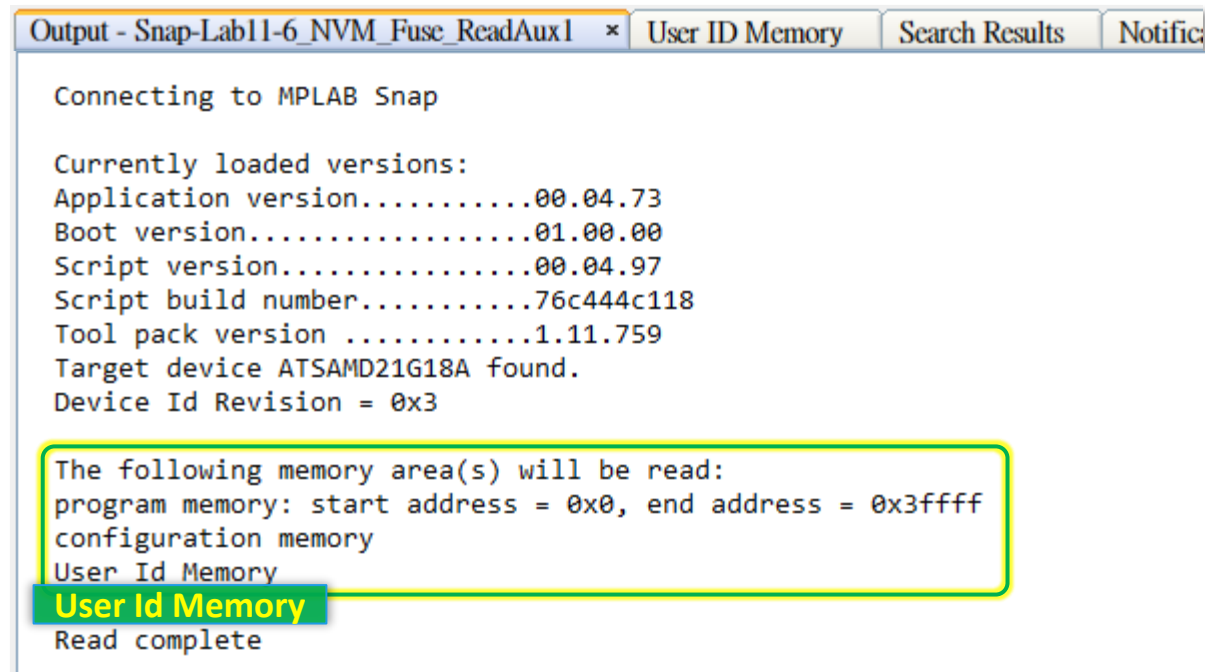
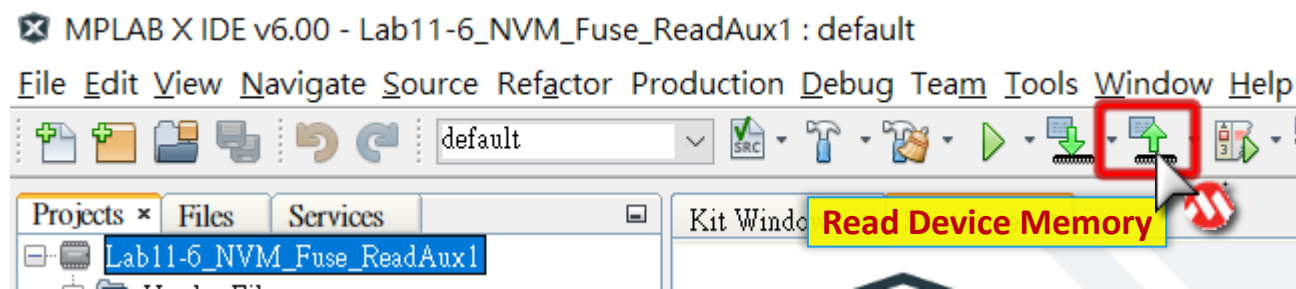
Unlock

Help

User ID Memory in NVM AUX0 (User Row)



Read User ID Memory in MPLAB X IDE



User ID Memory in NVM AUX0 (User Row)



Modify and write User ID Memory in MPLAB X IDE

The screenshot shows the MPLAB X IDE interface with the 'User ID Memory' window open. The window displays a table of memory addresses and their corresponding User IDs. The first row, at address 0080_4008, shows a User ID of 0xFFFFFFFF, which is highlighted with a red box. A yellow callout box with the text 'Try Click to Modify' points to this row. Below the table, there are dropdown menus for 'Memory' (set to 'User ID Memory') and 'Format' (set to 'Standard'). A red box highlights the 'Debug Build Required For Editing.' link in the error message dialog. The error message dialog also contains the text '(Lab11-6_NVM_Fuse_ReadAux1) - Memory access restricted' and a close button.

Address	User ID
0080_4008	0xFFFFFFFF
0080_400C	0xFFFFFFFF
0080_4010	0xFFFFFFFF
0080_4014	0xFFFFFFFF
0080_4018	0xFFFFFFFF
0080_401C	0xFFFFFFFF
0080_4020	0xFFFFFFFF
0080_4024	0xFFFFFFFF
0080_4028	0xFFFFFFFF
0080_402C	0xFFFFFFFF
0080_4030	0xFFFFFFFF
0080_4034	0xFFFFFFFF
0080_4038	0xFFFFFFFF
0080_403C	0xFFFFFFFF
0080_4040	0xFFFFFFFF

Memory: User ID Memory Format: Standard

(Lab11-6_NVM_Fuse_ReadAux1) - Memory access restricted
[Debug Build Required For Editing.](#)

Debug Build Required For Editing.

User ID Memory in NVM AUX0 (User Row)



Modify and write User ID Memory in MPLAB X IDE

Build for Debugging (Project Lab11-6_NVM_Fuse_ReadAux1)

Build for Debugging (Project Lab11-6_NVM_Fuse_ReadAux1)

Build with PRO Comparison (Project Lab11-6_NVM_Fuse_ReadAux1)

Build for Debugging (Project Lab11-6_NVM_Fuse_ReadAux1)

Click to Modify

Address	User ID
0080_4008	0x12345678
0080_400C	0xFFFFFFFF
0080_4010	0xFFFFFFFF
0080_4014	0xFFFFFFFF
0080_4018	0xFFFFFFFF
0080_401C	0xFFFFFFFF
0080_4020	0xFFFFFFFF
0080_4024	0xFFFFFFFF
0080_4028	0xFFFFFFFF
0080_402C	0xFFFFFFFF
0080_4030	0xFFFFFFFF
0080_4034	0xFFFFFFFF
0080_4038	0xFFFFFFFF
0080_403C	0xFFFFFFFF
0080_4040	0xFFFFFFFF

Memory User ID Memory Format Standard

User ID Memory in NVM AUX0 (User Row)



Modify and write User ID Memory in MPLAB X IDE

The screenshot shows the MPLAB X IDE interface. The 'User ID Memory' window is open, displaying a table of memory addresses and their corresponding User IDs. The address 0080_4008 is highlighted, and its User ID is 0x12345678. A green box with the text 'Success to Program' is overlaid on the table. The 'Programmer' menu is open, showing various programming options. The 'Program Device for Debugging' option is highlighted in yellow.

Address	User ID
0080_4008	0x12345678
0080_4010	0xFFFFFFFF
0080_4014	0xFFFFFFFF
0080_4018	0xFFFFFFFF
0080_401C	0xFFFFFFFF
0080_4020	0xFFFFFFFF
0080_4024	0xFFFFFFFF
0080_4028	0xFFFFFFFF
0080_402C	0xFFFFFFFF
0080_4030	0xFFFFFFFF
0080_4034	0xFFFFFFFF
0080_4038	0xFFFFFFFF
0080_403C	0xFFFFFFFF
0080_4040	0xFFFFFFFF

Memory: User ID Memory Format: Standard

Programmer menu options:

- Make and Program Device (Project Lab11-6_NVM_Fuse_ReadAux1)
- Program Device for Debugging (Project Lab11-6_NVM_Fuse_ReadAux1)
- Program Device for Production (Project Lab11-6_NVM_Fuse_ReadAux1)
- Erase Device Memory (Project Lab11-6_NVM_Fuse_ReadAux1)
- Programmer To Go PICkit3/PICkit4 (Project Lab11-6_NVM_Fuse_ReadAux1)

User ID Memory in NVM AUX0 (User Row)

NVMCTRL User Row Interface function

- Latest Harmony Framework(PLIB) to support below User Row NVM operation.

bool **NVMCTRL_USER_ROW_RowErase**(**uint32_t** address)

NVM AUX0(User Row) Erase Row of given address.

bool **NVMCTRL_USER_ROW_PageWrite**(**uint32_t** *data, **const uint32_t** address)

NVM AUX0(User Row) Page Write to given address.

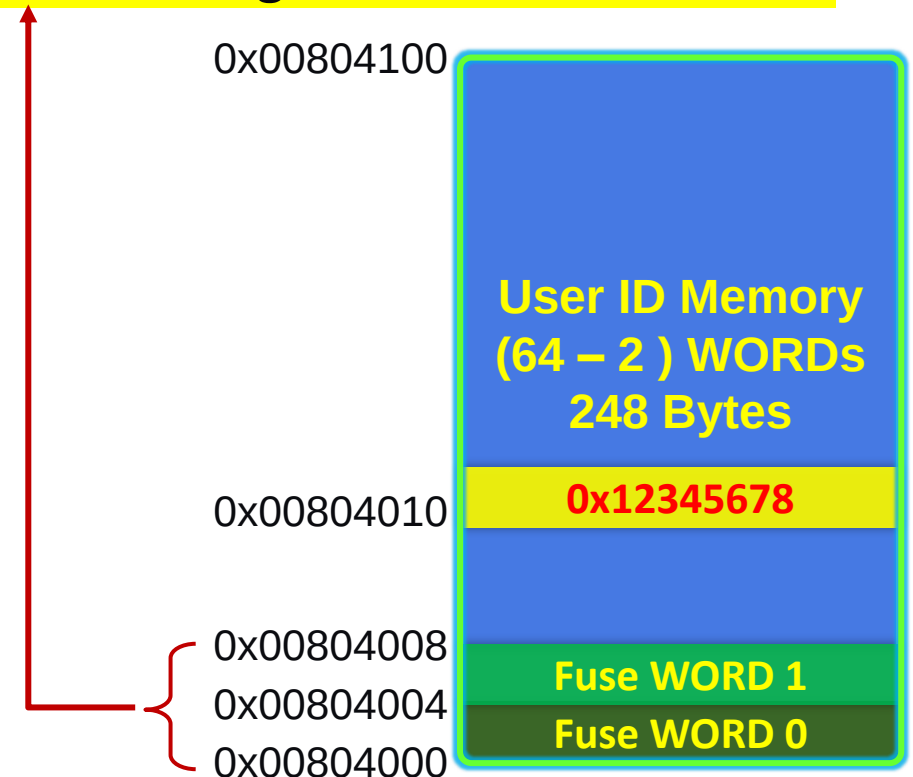
void **NVMCTRL_SecurityBitSet**(**void**)

Set the Security Bit by software.

⚙️ Lab11-7 Write NVM AUX0 (User Row)

- Please continue from [SAM2001 Lab11-6 Read NVM AUX1 \(Software Calibration\)](#)
- Try to write one WORD user data **0x12345678** to **AUX0 User ID Memory** address **0x00804010**, and then read it back to make sure write has succussed.
- Remember always execute **NVM_Read_AUX0(NVM_AUX0_Word)** once, before you to execute the **NVM_Write_AUX0(NVM_AUX0_Word)** function for make sure to get the first two WORD of **Fuse Setting** and then execute **erase** and write it back.
- Please output AUX0 write function as below format :
 - “Erase NVM-AUX0 User Row :”
 - “ - success”
 - “Write NVM-AUX0 User Row :”
 - “ - Page 0 : success”
 - “ - Page 1 : success”
 - “ - Page 2 : success”
 - “ - Page 3 : success”
 - “done”

Let's go !



⚙️ Lab11-7 Write NVM AUX0 (User Row)

Step1

```
...  
void NVM_Read_AUX1( uint32_t *Word ) {...}  
  
// TODO 11-7.01  
bool NVMCTRL_GetSecurityBitStatus( void )  
{  
    if( NVMCTRL_REGS->NVMCTRL_STATUS & NVMCTRL_STATUS_SB(1UL) )  
    {  
        return true;  
    }  
  
    return false;  
}  
  
void NVM_Write_AUX0( uint32_t *Word )  
{  
    uint32_t i;  
  
    while( NVMCTRL_IsBusy() ) {}  
  
    if( NVMCTRL_GetSecurityBitStatus() )  
    {  
        myprintf( "Security Bit has been set!\r\n" );  
        return;  
    }  
}
```

Check **Security Bits** before Write AUX0
Let's talk it later in next LAB

Lab11-7 Write NVM AUX0 (User Row)

Step2

```
...
void NVM_Read_AUX1( uint32_t *Word ) {...}

// TODO 11-7.01
bool NVMCTRL_GetSecurityBitStatus( void ) { ... }

void NVM_Write_AUX0( uint32_t *Word )
{
    ...
    myprintf( "Erase NVM-AUX0 User Row : \r\n" );
    NVMCTRL_USER_ROW_RowErase((uint32_t)USER_PAGE_ADDR);
    while( NVMCTRL_IsBusy() ) {}
    myprintf( " - success\r\n" );

    myprintf( "Write NVM-AUX0 User Row : \r\n" );
    for( i=0 ; i<4 ; i++ )
    {
        myprintf( " - Page %d : ", i );
        NVMCTRL_USER_ROW_PageWrite((uint32_t *)Word, (uint32_t)USER_PAGE_ADDR+(i*64));
        while( NVMCTRL_IsBusy() ) {}
        myprintf( "success\r\n" );
        Word+=16;
    }
    myprintf( "done\r\n\r\n" );
}
```

⚙️ Lab11-7 Write NVM AUX0 (User Row)

Step3

```
...
void NVM_Read_AUX1( uint32_t *Word ) {...}

// TODO 11-7.01
bool NVMCTRL_GetSecurityBitStatus( void ) { ... }

void NVM_Write_AUX0( uint32_t *Word ) { ... }

int main ( void )
{
    ...
    myprintf("\033[2J");
    myprintf("\033[?25l\033[1;1H");
    NVM_Read_UQSN( NVM_UQSN_Word );
    NVM_Read_AUX0( NVM_AUX0_Word );
    NVM_Read_AUX1( NVM_AUX1_Word );

    // TODO 11-7.02
    NVM_AUX0_Word[4] = 0x12345678;
    NVM_Write_AUX0( NVM_AUX0_Word );
    NVM_Read_AUX0( NVM_AUX0_Word );

    while ( true )
    {
        ...
    }
}
```

Remember always execute `NVM_Read_AUX0(NVM_AUX0_Word)` once, before you to execute the `NVM_Write_AUX0(NVM_AUX0_Word)` function for make sure to get the first two WORD of Fuse Setting

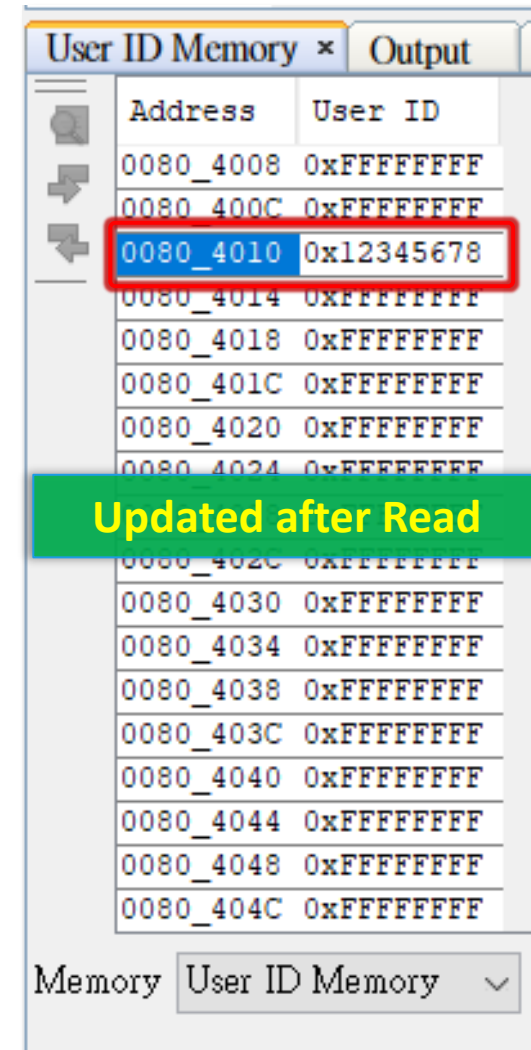
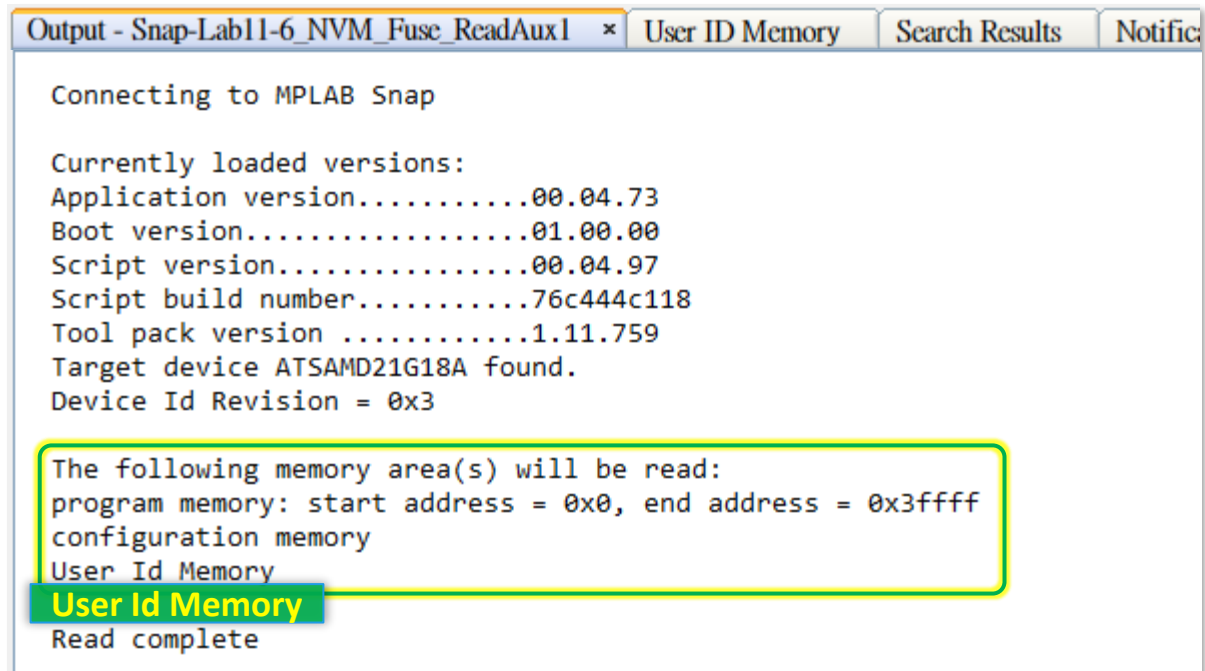
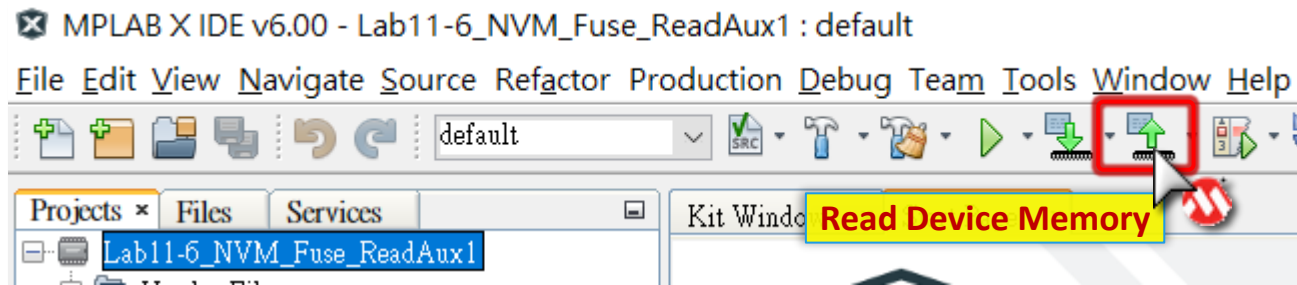
✖ Lab11-7 Write NVM AUX0 (User Row)

Result

```
COM3 - Tera Term VT
File Edit Setup Control Window Help
Erase NVM-AUX0 User Row :
- success
Write NVM-AUX0 User Row :
- Page 0 : success
- Page 1 : success
- Page 2 : success
- Page 3 : success
done
NVM-AUX0 User Row : (64 Words)
- Word 0 and Word 1
0x00804000 : 0xD800C777
0x00804004 : 0xFFFF005C
- User ID Memory
0x00804008 : 0xFFFFFFFF 0xFFFFFFFF 0x12345678 0xFFFFFFFF
0x00804018 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804028 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
```

✖ Lab11-7 Write NVM AUX0 (User Row)

Read User ID Memory in MPLAB X IDE



⚙️ Lab11-7 Write NVM AUX0 (User Row)

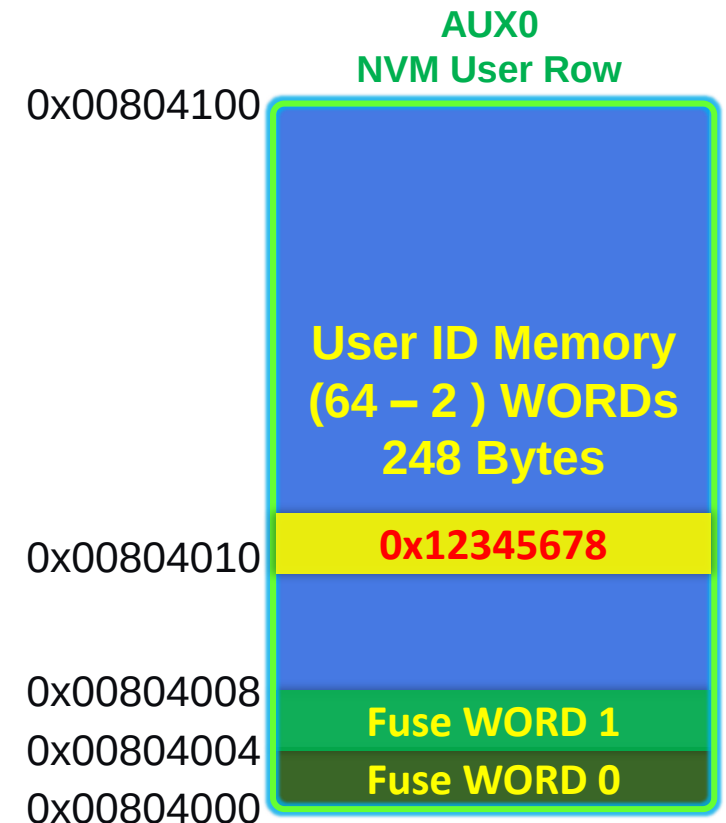
Read User ID Memory in Microchip Studio



Document.userpage

```
:0200000400807A
:1040000077C700D85C00FFFFFFFFFFF48
:1040100078563412FFFFFFF98
:10402000FFFFFFFFFA0
:10403000FFFFFFFFF90
:10404000FFFFFFFFF80
:10405000FFFFFFFFF70
:10406000FFFFFFFFF60
:10407000FFFFFFFFF50
:10408000FFFFFFFFF40
:10409000FFFFFFFFF30
:1040A000FFFFFFFFF20
:1040B000FFFFFFFFF10
:1040C000FFFFFFFFF00
:1040D000FFFFFFFFF0
:1040E000FFFFFFFFFE0
:1040F000FFFFFFFFFD0
:00000001FF
```

Microchip Studio IDE to store Fuse and User ID Memory together



NVM Security Bit

- **Security Bit**

- The security bit allows the entire chip to be locked from external access for code security.
- The security bit can be written by a dedicated command, Set **Security Bit (SSB)**.
- Once set Security Bit, the only way to **clear** the security bit is **through a debugger Chip Erase command**.
- After issuing the SSB command, the PROGE error bit can be checked.
- In order to increase the security level, it is recommended to enable the internal BOD33 when the security bit is set.

- **Chip Erase**

- Chip-Erase consists of removing all sensitive information stored in the chip and **clearing the NVMCTRL security bit**.
- **All volatile memories**, the **Flash memory** (including the EEPROM Emulation area) and the **RWWEE Emulation** section will be erased.
- **The Flash auxiliary rows, including the user row, will not be erased**.

- **Programming**

- Programming the Flash or RAM memories **through external programmer** is only possible when the device is **not protected by the NVMCTRL security bit**.

- **NVM Auxiliary space**

- Auxiliary space **cannot be accessed from outside** if the security bit is set.

- **Debug Operation**

- When an external debugger forces the CPU into debug mode, the peripheral continues normal operation.
- **Access to the NVM block can be protected by the security bit**. In this case, the NVM block will not be accessible.

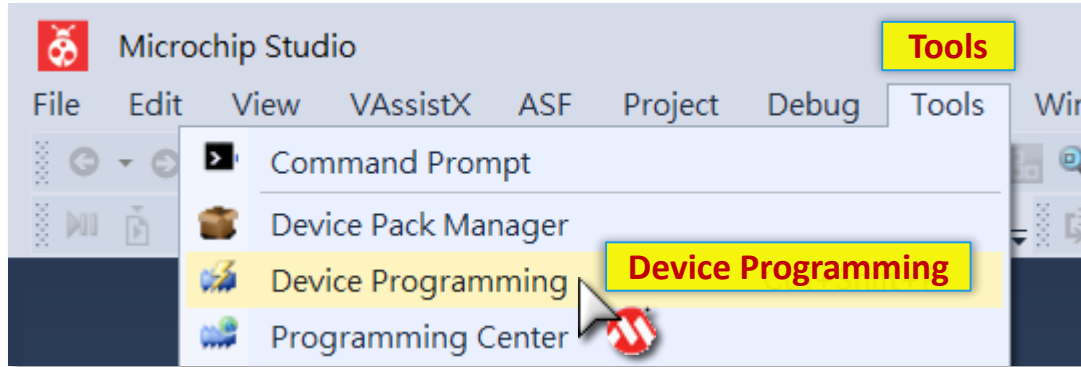
Lab11-8 Code Protect by NVM Security Bit

- Please continue from **SAM2001 Lab11-7 Write NVM AUX0 (User Row)**
- You will practice below exercises in this LAB.
 - Try use **Microchip Studio** to enable the Security Bit.
 - Try under **MPLAB X IDE** and use **Harmony** to enable the Security Bit.
 - Try read Flash memory while Security Bit has been settled.
 - Try **un-seal** the Security Bit by **Erase Chip**. In both **Microchip Studio** and **MPLAB X IDE**.
 - Make sure the “Self Flash Access” is executable while enable the Security Bit.

Let's go !

⚙️ Lab11-8 Code Protect by NVM Security Bit

Use Microchip Studio to enable Security Bit



Open Tools

► Device Programming

Tool : MPLAB Snap

Device : ATSAM21G18A

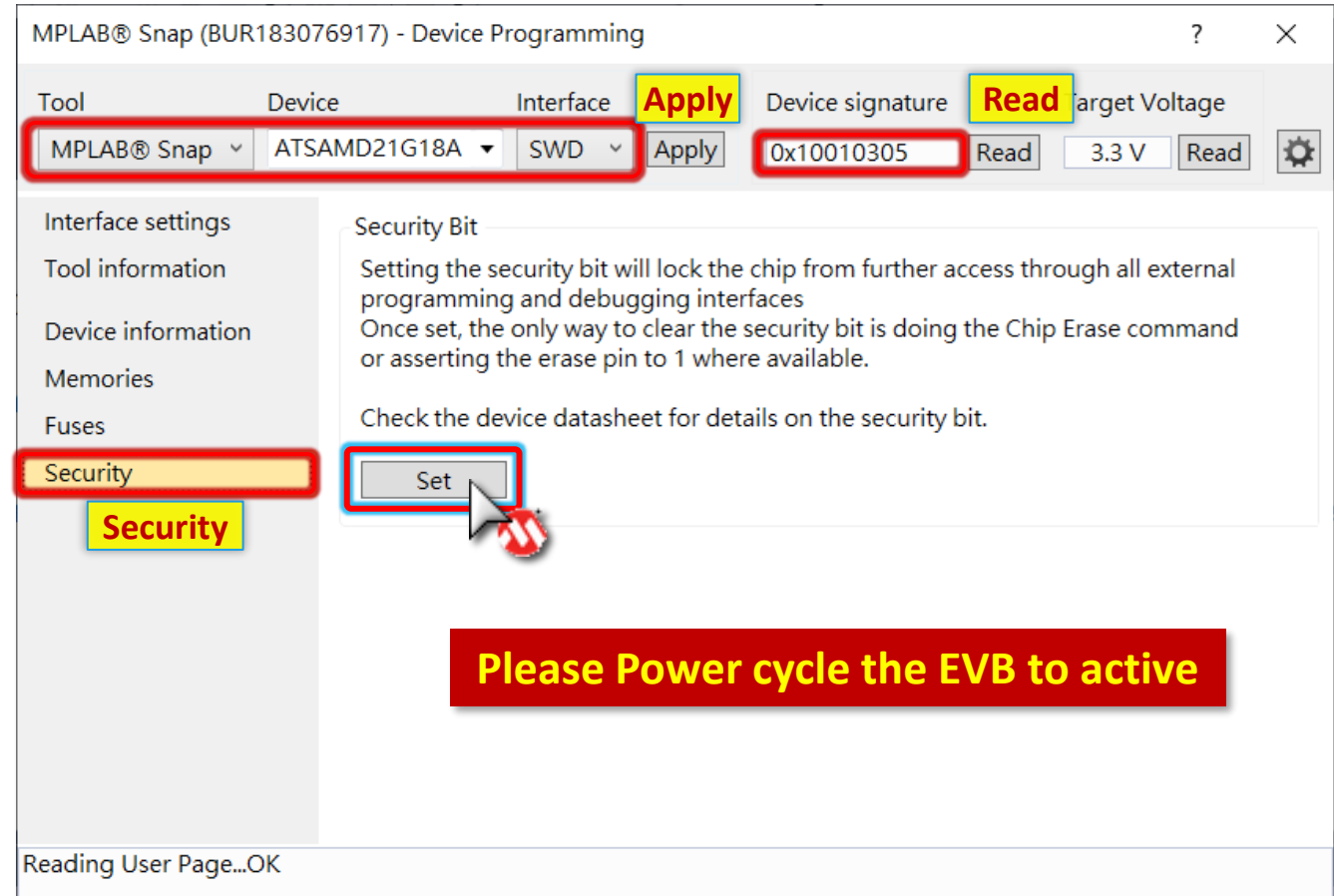
Interface : SWD

► [Apply]

► [Read]

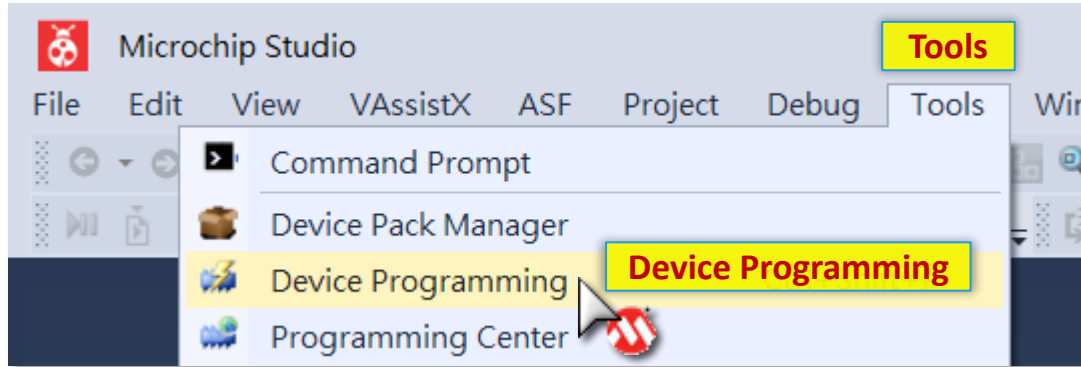
Device signature

► [Security]



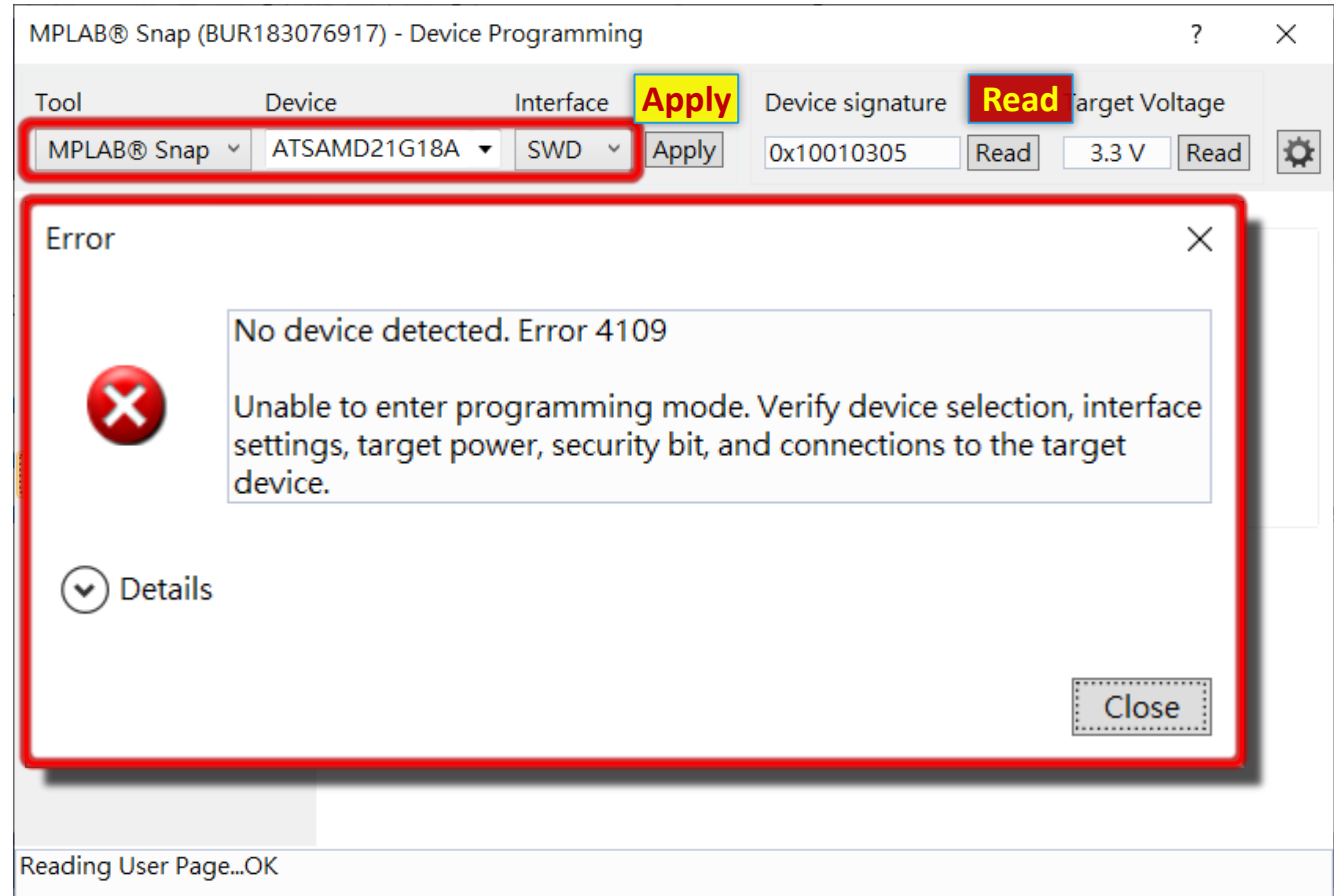
⚙️ Lab11-8 Code Protect by NVM Security Bit

Use Microchip Studio to Read Device signature after enable Security Bit



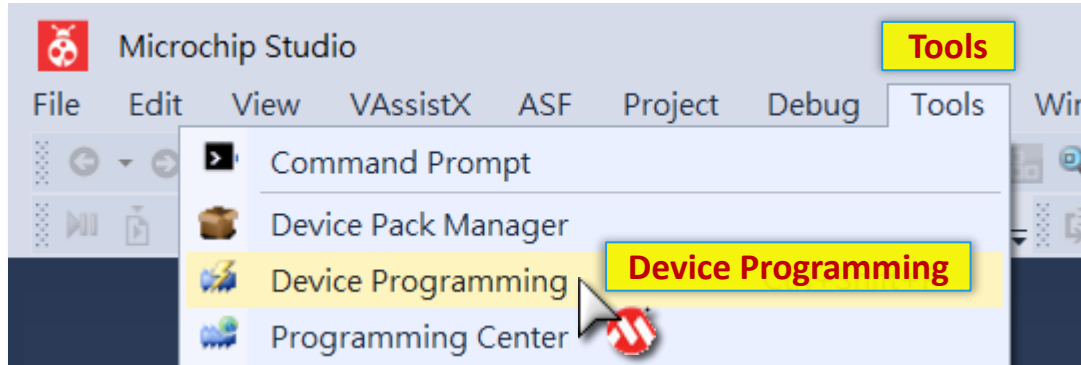
Power cycle the EVB

- ▶ **Tools**
- ▶ **Device Programming**
 - Tool : MPLAB Snap
 - Device : ATSAM21G18A
 - Interface : SWD
- ▶ **[Apply]**
- ▶ **[Read]**



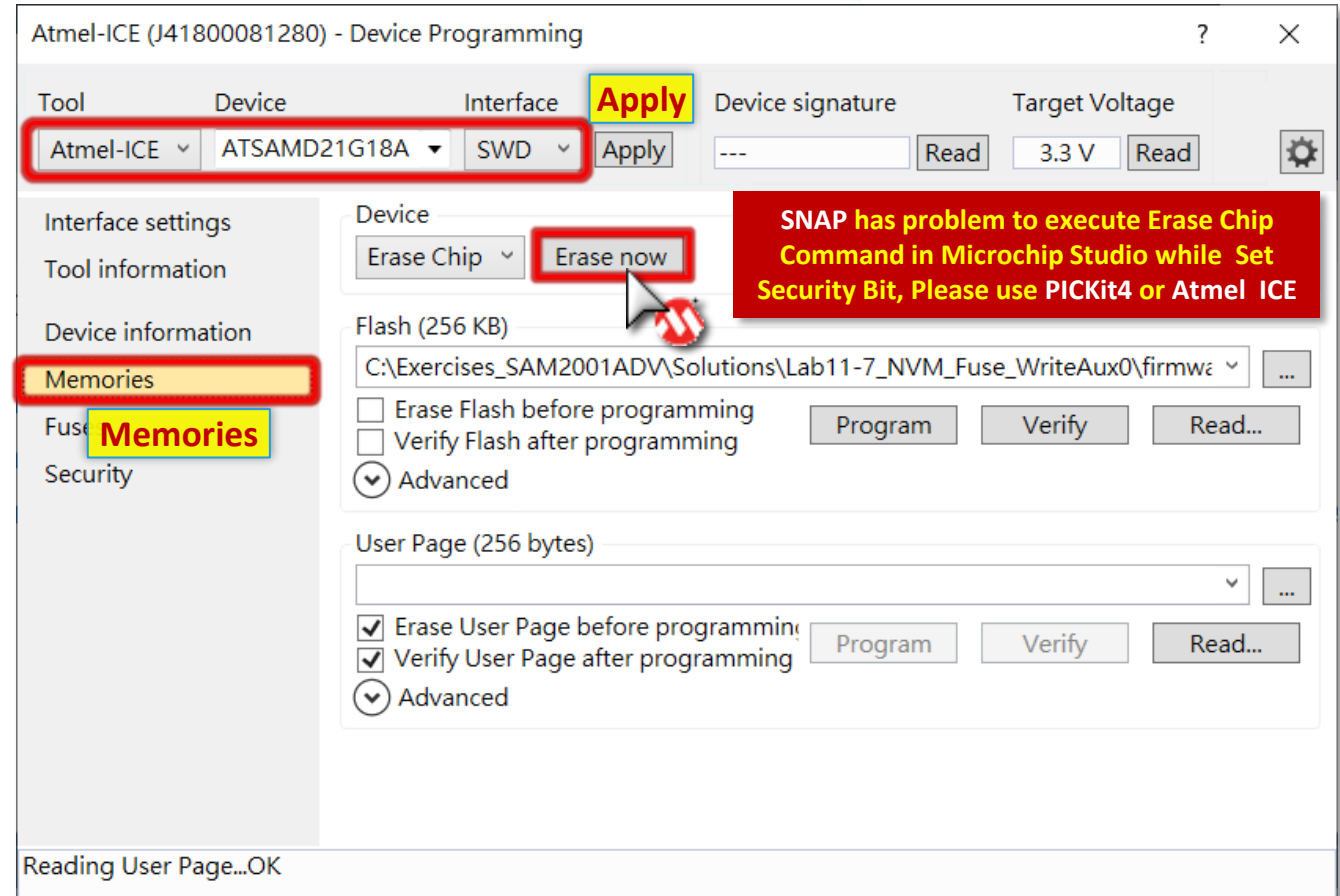
⚙️ Lab11-8 Code Protect by NVM Security Bit

Use Microchip Studio to Erase Chip for un-seal the Security Bit



Power cycle the EVB

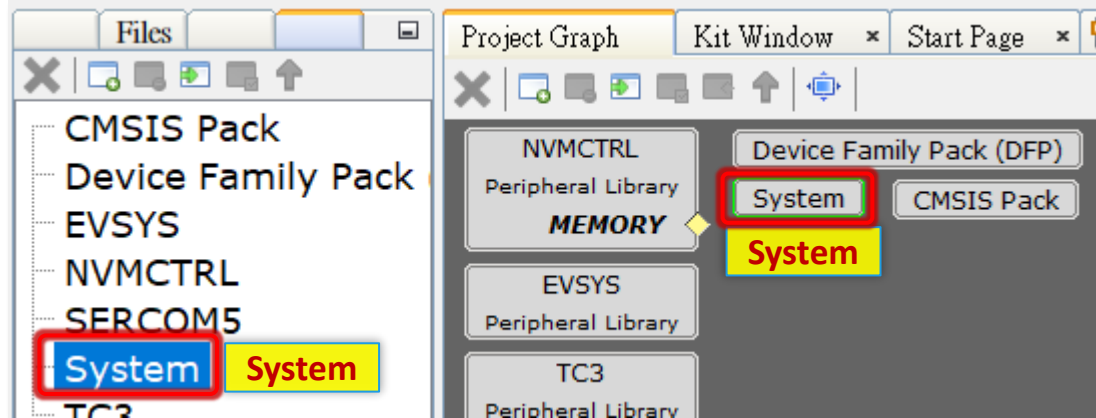
- ▶ **Tools**
- ▶ **Device Programming**
Tool : **PICKit4 or Atmel ICE**
Device : **ATSAMD21G18A**
Interface : **SWD**
- ▶ **[Apply]**
- ▶ **[Read]**
- ▶ **[Memory]**



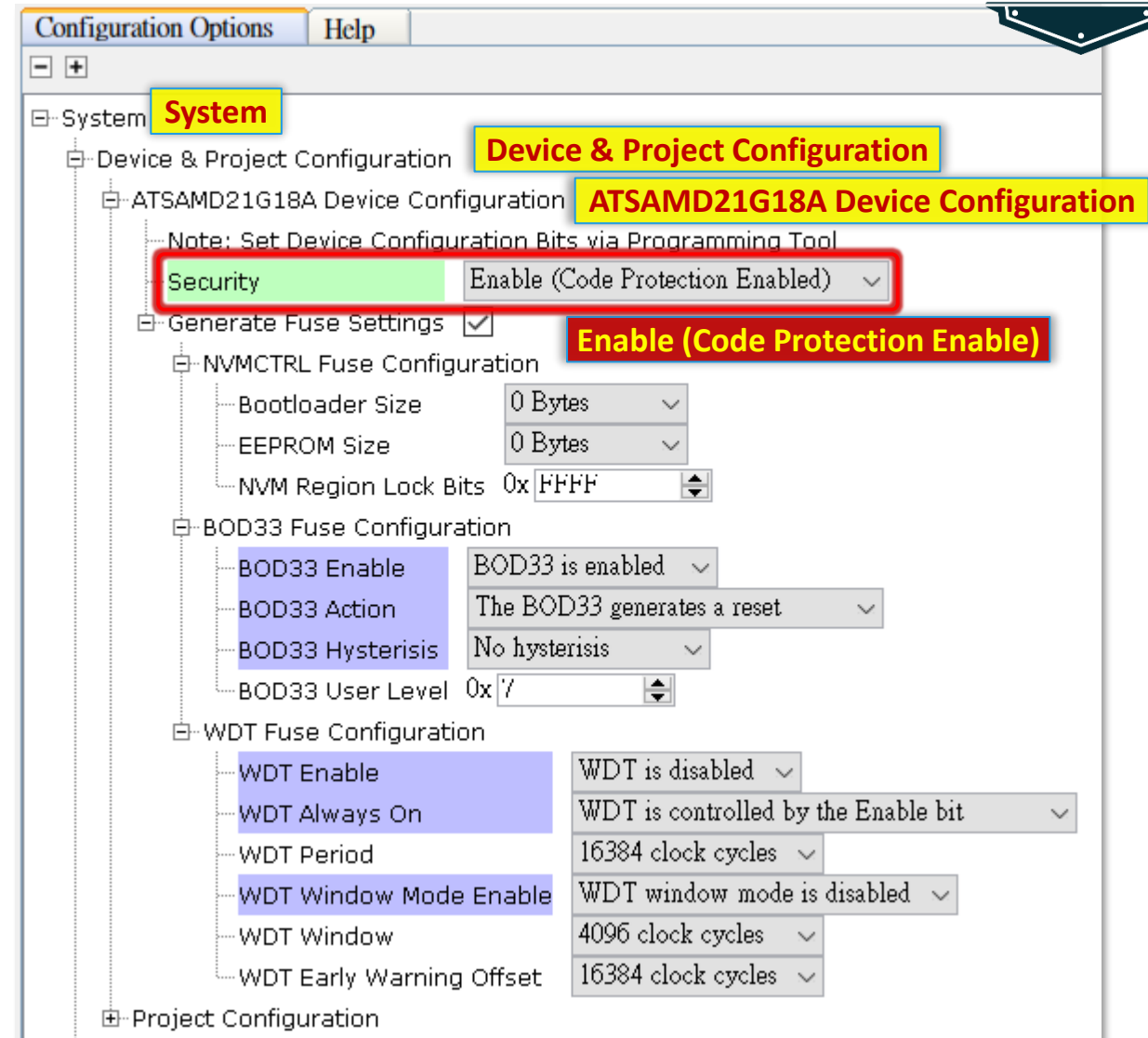
⚙️ Lab11-8 Code Protect by NVM Security Bit



Enable Code Protect in MPLAB Harmony



- Open **System**
 - ▶ **Device & Project Configuration**
 - ▶ **ATSAMD21G18A Device Configuration**
 - ▶ **Security**
 - ▶ **Enable (Code Protection Enable)**

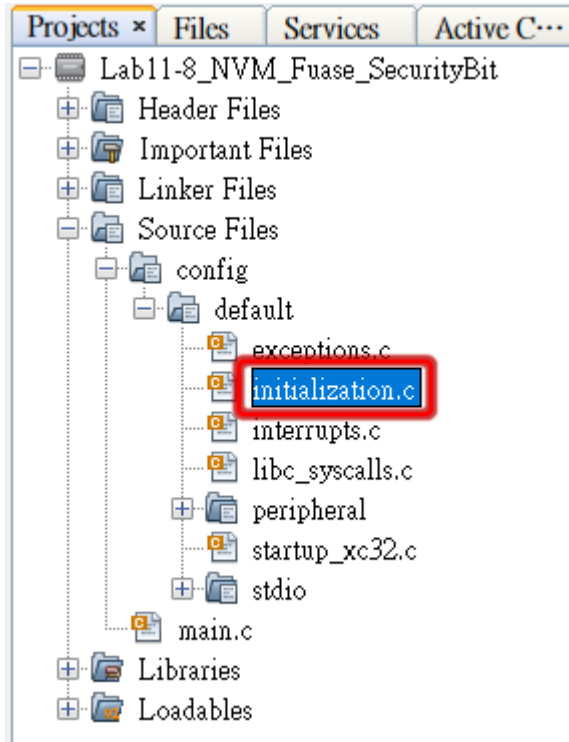


⚙️ Lab11-8 Code Protect by NVM Security Bit



Enable Code Protect in MPLAB Harmony

- Security Bit will be configured in initialization.c



```
// *****  
// Section: Configuration Bits  
// *****  
  
#pragma config NVMCTRL_BOOTPROT = SIZE_0BYTES  
#pragma config NVMCTRL_EEPROM_SIZE = SIZE_0BYTES  
#pragma config BOD33USERLEVEL = 0x7 // Enter Hexadecimal value  
#pragma config BOD33_EN = ENABLED  
#pragma config BOD33_ACTION = RESET  
  
#pragma config BOD33_HYST = DISABLED  
...  
  
#pragma config WDT_WINDOW_0 = SET  
#pragma config WDT_WINDOW_1 = 0x4 // Enter Hexadecimal value  
#pragma config WDT_EWOFFSET = CYC16384  
#pragma config WDT_WEN = DISABLED
```

**HEX image has enabled Code Protect in Harmony
will only active while using MPLAB X IDE/IPE
to program such kind of HEX image**

```
#ifndef __DEBUG  
const unsigned int __attribute__((space(prog), keep, address(0x41004000))) SET_SECURITY_BIT = 1;  
#endif
```

Security Bit

✖ Lab11-8 Code Protect by NVM Security Bit

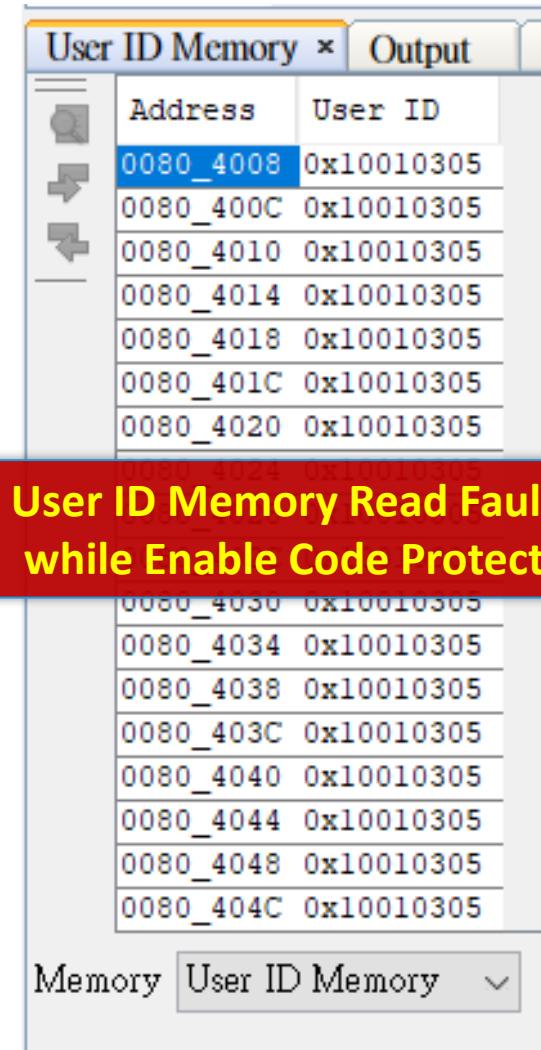
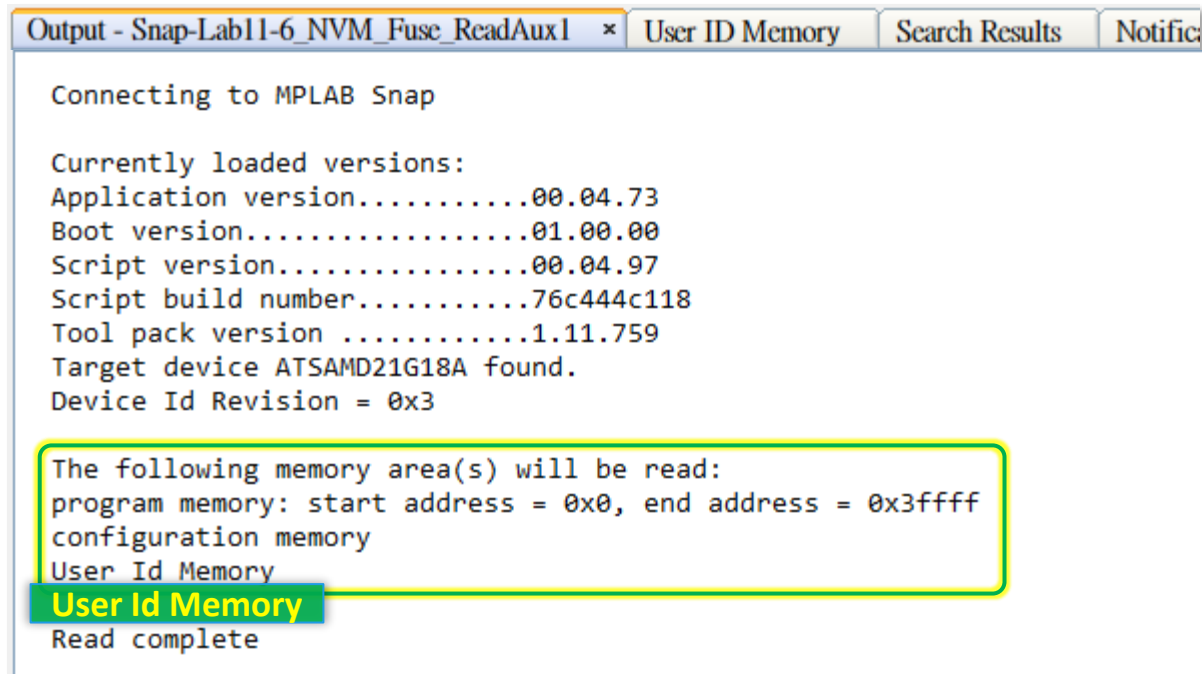
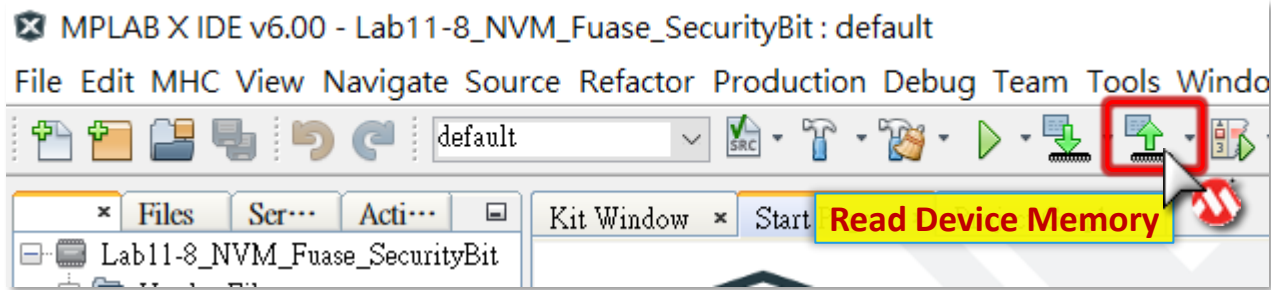
Result

```
COM3 - Tera Term VT
File Edit Setup Control Window Help
Security Bit has been set!
NVM-AUX0 User Row : (64 Words)
- Word 0 and Word 1
0x00804000 : 0xD800C777
0x00804004 : 0xFFFF005C
- User ID Memory
0x00804008 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804018 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804028 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804038 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804048 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804058 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804068 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
```

⚙️ Lab11-8 Code Protect by NVM Security Bit



Read User ID Memory in MPLAB X IDE while Code Protect enabled



⚙️ Lab11-8 Code Protect by NVM Security Bit



Read Execution Memory in MPLAB X IDE after execute Erase Chip

MPLAB X IDE v6.00 - Lab11-8_NVM_Fuase_SecurityBit : default

File Edit MHC View Navigate Source Refactor Production Debug Team Tools Window

default

Files Ser... Acti... Kit Window Start Page Project

Lab11-8_NVM_Fuase_SecurityBit

Maximize and Program Device (Project Lab11-8_NVM_Fuase_SecurityBit)
Program Device for Debugging (Project Lab11-8_NVM_Fuase_SecurityBit)
Program Device for Production (Project Lab11-8_NVM_Fuase_SecurityBit)
Erase Device Memory (Project Lab11-8_NVM_Fuase_SecurityBit)
Programmer To Go PIC18-4

Erase Device Memory

Output - Snap-Lab11-6_NVM_Fuse_ReadAux1 x User ID Memory

Connecting to MPLAB Snap

Currently loaded versions:
Application version.....00.04.73
Boot version.....01.00.00
Script version.....00.04.97
Script build number.....76c444c118
Tool pack version1.11.759
Target device ATSAMD21G18A found.
Device Id Revision = 0x3

Erasing...
Erase successful

Erase to un-seal Security Bit

Execution Memory x Output Search Results Notifications Console New

Address	00	02	04	06	08	0A	0C	0E	ASCII
									IFLASH Memory
0000_0000	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	
0000_0010	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	
0000_0020	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	
0000_0030	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	
0000_0040	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	

Execution Memory Read Successful after Erase Chip

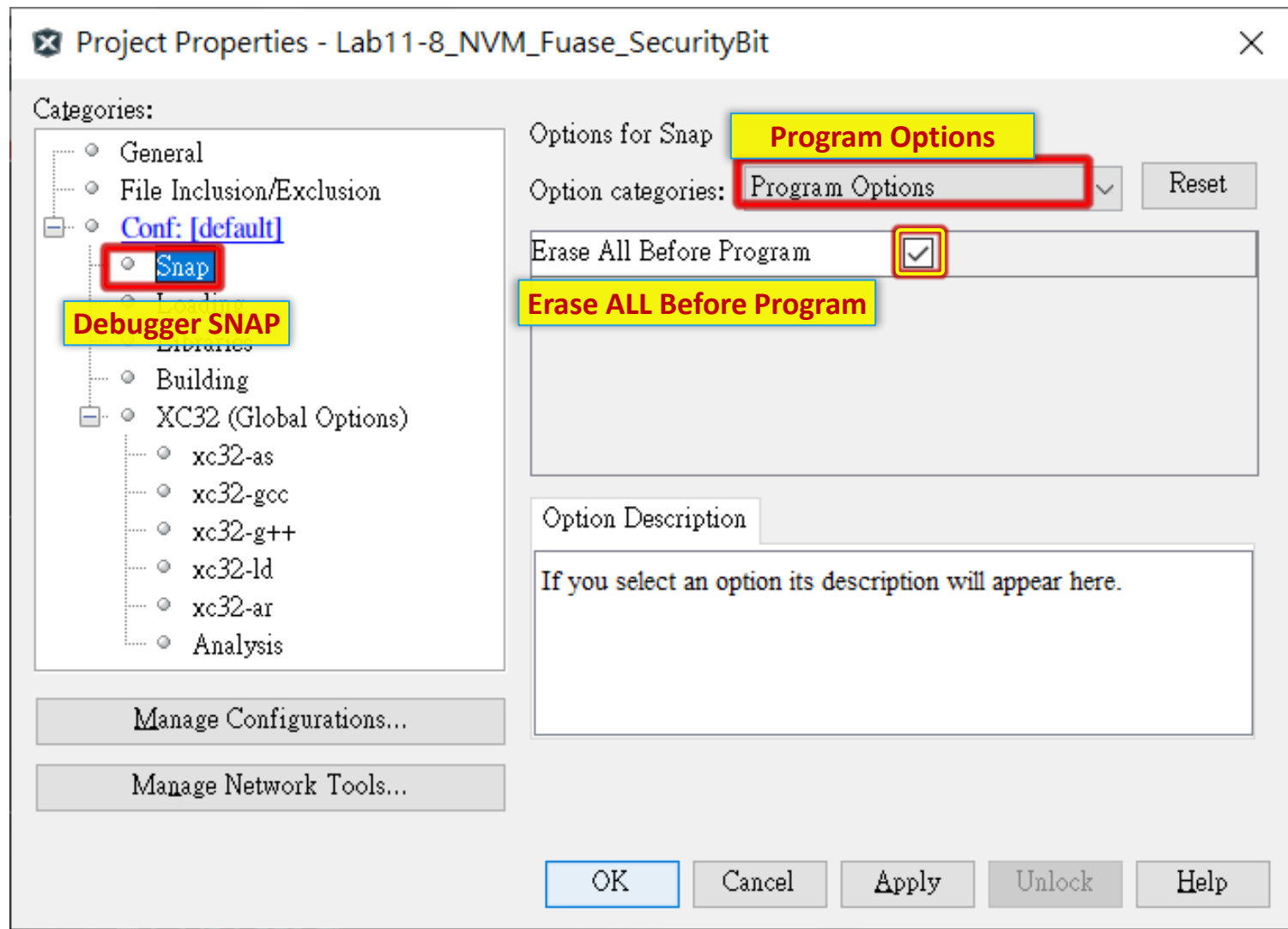
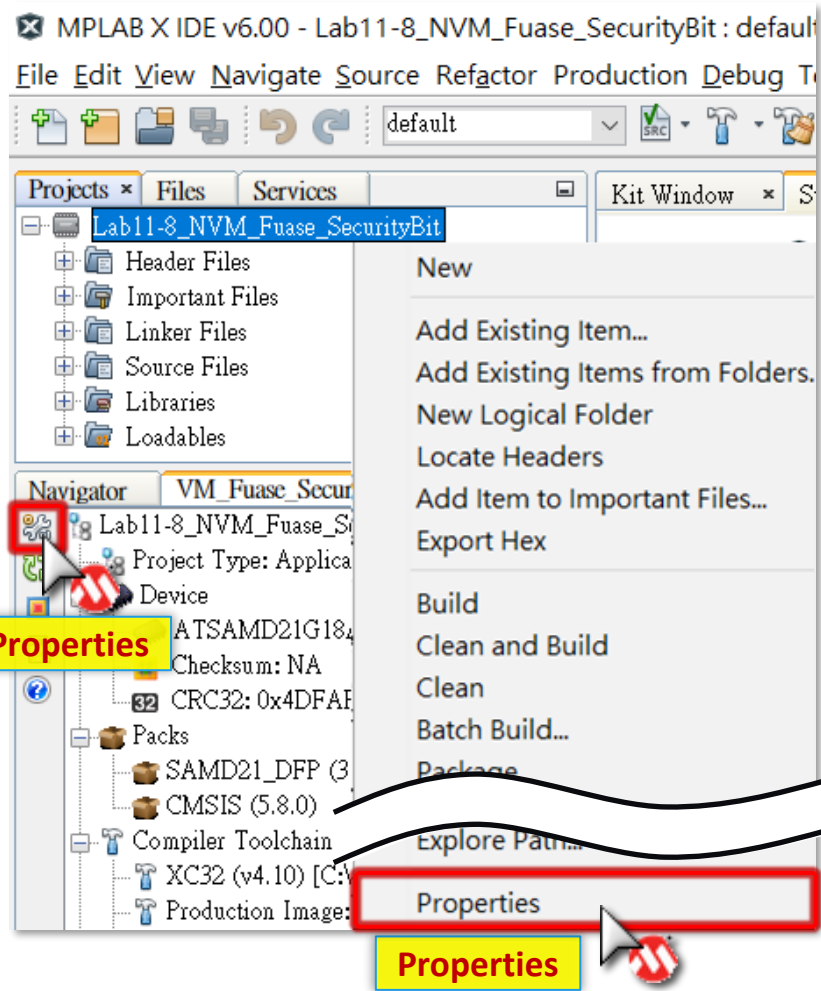
Memory Execution Mem... Format Data

Execution Memory

⚙️ Lab11-8 Code Protect by NVM Security Bit



Harmony New project will enable Erase All Before Program option



⚙️ Lab11-8 Code Protect by NVM Security Bit

Step1 : Implement a software Security Bit configure function

```
...
bool NVMCTRL_GetSecurityBitStatus( void ) {...}
void NVM_Write_AUX0( uint32_t *Word ) {...}

// TODO 11-8.01
void NVM_Code_Protect( void )
{
    while( NVMCTRL_IsBusy() ) {}

    if( NVMCTRL_GetSecurityBitStatus() )
        return;

    NVMCTRL_SecurityBitSet();

    myprintf( "Security Bit set successfully, please cycle power to active!\r\n\r\n" );
}

int main ( void )
{
    ...
    myprintf("\033[2J");
    myprintf("\033[?251\033[1;1H");
    NVM_Read_UQSN( NVM_UQSN_Word );
    NVM_Read_AUX0( NVM_AUX0_Word );
    ...
}
```

Security Bit will be enabled after execute this function whatever to using MPLAB X IDE/IPE , Microchip Studio or others programmer to program this HEX image

⚙️ Lab11-8 Code Protect by NVM Security Bit

Step2 : Implement Self NVM Flash Read and Write function

```
...
bool NVMCTRL_GetSecurityBitStatus( void ) {...}
void NVM_Write_AUX0( uint32_t *Word ) {...}

// TODO 11-8.01
void NVM_Code_Protect( void ) {...}

#define SELF_RW_ADDRESS 0x3000
void NVM_Self_ReadWrite()
{
    uint32_t ReadWord[4];
    uint32_t WritePage[16];

    while( NVMCTRL_IsBusy() ) {}

    NVMCTRL_Read( ReadWord, 16, SELF_RW_ADDRESS );

    myprintf( "Self Read (4 Words)\r\n" );
    myprintf( "0x%08X : 0x%08X\r\n", SELF_RW_ADDRESS+0x0, ReadWord[0] );
    myprintf( "0x%08X : 0x%08X\r\n", SELF_RW_ADDRESS+0x4, ReadWord[1] );
    myprintf( "0x%08X : 0x%08X\r\n", SELF_RW_ADDRESS+0x8, ReadWord[2] );
    myprintf( "0x%08X : 0x%08X\r\n", SELF_RW_ADDRESS+0xC, ReadWord[3] );
    NVMCTRL_RowErase( SELF_RW_ADDRESS );
    while( NVMCTRL_IsBusy() ) {}
    myprintf( "Erase Row at address 0x%08X\r\n", SELF_RW_ADDRESS );
}
```

⚙️ Lab11-8 Code Protect by NVM Security Bit

Step3 : Implement Self NVM Flash Read and Write function (cont.)

```
...
bool NVMCTRL_GetSecurityBitStatus( void ) {...}
void NVM_Write_AUX0( uint32_t *Word ) {...}

// TODO 11-8.01
void NVM_Code_Protect( void ) {...}

#define SELF_RW_ADDRESS 0x3000
void NVM_Self_ReadWrite()
{
    ...
    myprintf( "Erase Row  at address 0x%08X\r\n", SELF_RW_ADDRESS );

    WritePage[0] = ReadWord[0]-0x11111111;
    WritePage[1] = ReadWord[1]-0x22222222;
    WritePage[2] = ReadWord[2]-0x33333333;
    WritePage[3] = ReadWord[3]-0x44444444;
    NVMCTRL_PageWrite( WritePage, SELF_RW_ADDRESS );
    while( NVMCTRL_IsBusy() ) {}
    myprintf( "Write Page at address 0x%08X\r\n", SELF_RW_ADDRESS );
    myprintf( "done\r\n\r\n" );
}

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

Lab11-8 Code Protect by NVM Security Bit

Step4 : Execute Self NVM Flash Read and Write function

```
...  
bool NVMCTRL_GetSecurityBitStatus( void ) {...}  
void NVM_Write_AUX0( uint32_t *Word ) {...}
```

```
// TODO 11-8.01  
void NVM_Code_Protect( void ) {...}
```

```
#define SELF_RW_ADDRESS 0x3000  
void NVM_Self_ReadWrite() {...}
```

```
int main ( void )  
{  
    ...  
    NVM_Read_UQSN( NVM_UQSN_Word );  
    NVM_Read_AUX0( NVM_AUX0_Word );  
    NVM_Read_AUX1( NVM_AUX1_Word );  
    NVM_AUX0_Word[4] = 0x12345678;  
    NVM_Write_AUX0( NVM_AUX0_Word );  
    NVM_Read_AUX0( NVM_AUX0_Word );
```

```
// TODO 11-8.02  
NVM_Code_Protect();  
NVM_Self_ReadWrite();
```

```
while ( true ) { ... }
```

Lab11-8 Code Protect by NVM Security Bit

Result

```
COM3 - Tera Term VT
File Edit Setup Control Window Help
1st boot after program

Erase NVM-AUX0 User Row :
- success
Write NVM-AUX0 User Row :
- Page 0 : success
- Page 1 : success
- Page 2 : success
- Page 3 : success
done

NVM-AUX0 User Row : (64 Words)
- Word 0 and Word 1
0x00804000 : 0xD800C777
0x00804004 : 0xFFFF005C
- User ID Memory
0x00804008 : 0xFFFFFFFF 0xFFFFFFFF 0x12345678 0xFFFFFFFF
0x00804018 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804028 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804038 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804048 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804058 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804068 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804078 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804088 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804098 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040A8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040B8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040C8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040D8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040E8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040F8 : 0xFFFFFFFF 0xFFFFFFFF
- done

Security Bit set successfully, please cycle power to active!

Self Read (4 Words)
0x00003000 : 0xFFFFFFFF
0x00003004 : 0xFFFFFFFF
0x00003008 : 0xFFFFFFFF
0x0000300C : 0xFFFFFFFF
Erase Row at address 0x00003000
Write Page at address 0x00003000
done
```

```
COM3 - Tera Term VT
File Edit Setup Control Window Help
2nd boot after power cycle

Security Bit has been set!
NVM-AUX0 User Row : (64 Words)
- Word 0 and Word 1
0x00804000 : 0xD800C777
0x00804004 : 0xFFFF005C
- User ID Memory
0x00804008 : 0xFFFFFFFF 0xFFFFFFFF 0x12345678 0xFFFFFFFF
0x00804018 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804028 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804038 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804048 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804058 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804068 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804078 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804088 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804098 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040A8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040B8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040C8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040D8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040E8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040F8 : 0xFFFFFFFF 0xFFFFFFFF
- done

Self Read (4 Words)
0x00003000 : 0xEEEEEEEE
0x00003004 : 0xDDDDDDDD
0x00003008 : 0CCCCCCCC
0x0000300C : 0BBBBBBBB
Self Flash Write successfully
While Security Bit enabled

Erase Row at address 0x00003000
Write Page at address 0x00003000
done
```

Microchip Trademarks

Overview

Microchip Technology Incorporated's products are marketed and sold under one or more of the following trademarks belonging to Microchip or one of Microchip's subsidiaries. Questions regarding use of these trademarks should be addressed to the Microchip's Legal Department via email: legal.department@microchip.com.

The following are registered trademarks of Microchip in the U.S.A. and other countries:

The Microchip name and logo, the Microchip logo, Adaptec, AnyRate, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Klear, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PackeTime, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA

The following are registered trademarks of Microchip in the U.S.A:

AgileSwitch, APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, Flashtec, Hyper Speed Control, HyperLight Load, IntelliMOS, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, TrueTime, WinPath, and ZL

The following are registered trademarks of Microchip in other countries: BeaconThings, GestIC, Silicon Storage Technology

The following are trademarks of Microchip in the U.S.A. and other countries:

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, Augmented Switching, BlueSky, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, Espresso T1S, EtherGREEN, IdealBridge, In-Circuit Serial Programming, ICSP, INICnet, Intelligent Paralleling, Inter-Chip Connectivity, JitterBlocker, maxCrypto, maxView, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICKit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, RTAX, RTG4, SAM-ICE, Serial Quad I/O, simpleMAP, SimpliPHY, SmartBuffer, SMART-I.S., storClad, SQL, SuperSwitcher, SuperSwitcher II, Switchtec, SynchroPHY, Total Endurance, TSHARC, USBCheck, VariSense, VectorBlox, VeriPHY, ViewSpan, WiperLock, XpressConnect, and ZENA

The following is a service mark of Microchip in the U.S.A.: SQTP

The following are logos of Microchip in the U.S.A. and other countries: The Adaptec logo, Frequency on Demand, Silicon Storage Technology, Symmcom, and Trusted Time

The following is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries: GestIC

