SAMD21 ARM Cortex-M0+ Microcontroller & MCC-Harmony v3

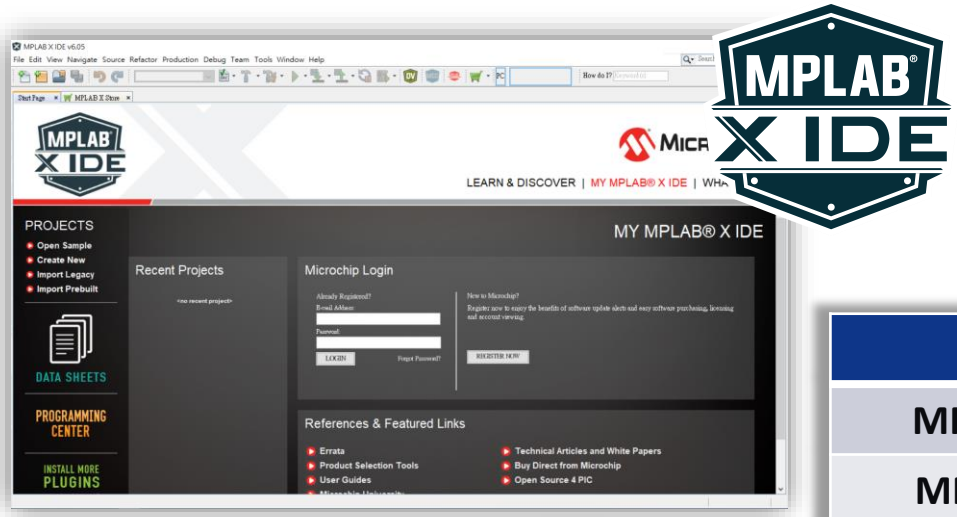
SAM2001ADV-MCC

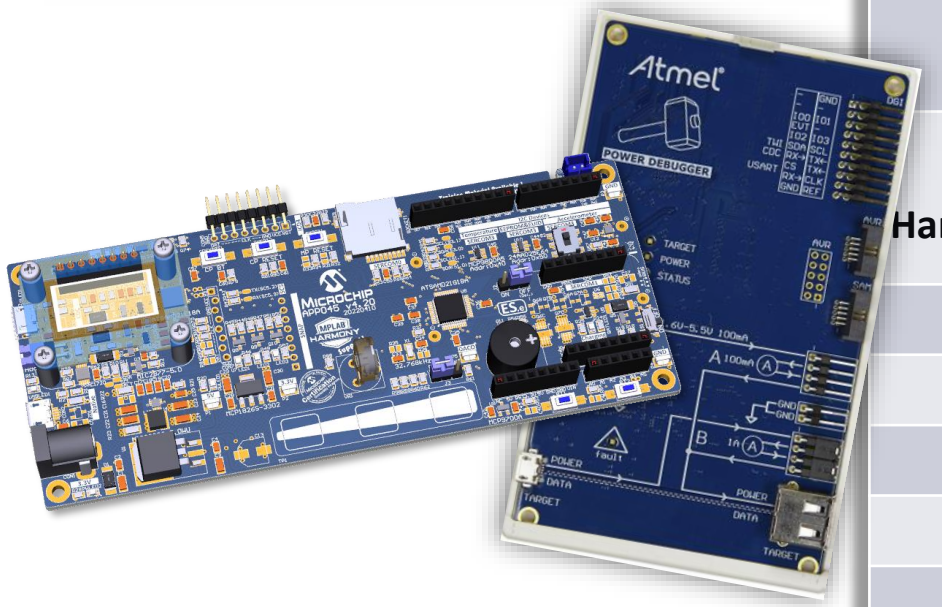
A Leading Provider of Smart, Connected and Secure Embedded Control Solutions

Adam Syu
CAE Taiwan
Microchip Technology Inc.

May 18 & June 1, 2023

Development Tools Needs





Tools	Version
MPLAB® X IDE	V6.05
MPLAB® XC32	V4.21 (-O1)
MPLAB® Code Configurator	v5.3.0
MPLAB® MCC Harmony Framework	csp v3.16.0 Harmony-services v1.2.0 dev_packs v3.16.0
SAMD21 DFP	v3.6.144
CMSIS	v5.8.0
EVM	APP045 v5.0/v4.2 ES.e/v4.2 ES.d
Tera Term	4.106
Tools	SNAP and Power Debugger

Index of SAM2001ADV

CAE Expert Classroom Index

- SAM2001ADV-01 (Day1)
  - SAM2001ADV-01 : Timer/Counter Operation Mode
  - SAM2001ADV-06 : DAC Architecture and MPLAB Data Visualizer
  - SAM2001ADV-07 : Analog Comparators (AC) Architecture
  - SAM2001ADV-08 : External Interrupt Controller (EIC) & Event System (EVSYS)

- SAM2001ADV-02 (Day2)
  - SAM2001ADV-10 : Direct Memory Access Controller (DMAC) - Single Transfer Block
  - SAM2001ADV-11 : Direct Memory Access Controller (DMAC) - Linked List for TB
  - SAM2001ADV-12 : Power Debugger and Sleep Mode
  - SAM2001ADV-13 : Application - SleepWalking

SAM2001ADV-01 (Day1)

- 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
- 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

SAM2001ADV-01 (Day1) cont.

- Event System (EVSYS)
  - Lab16-2 EXTINT Trigger Timer Counter
  - Lab16-3 EXTINT Callback and Trigger Timer Counter

SAM2001ADV-01 (Day2)

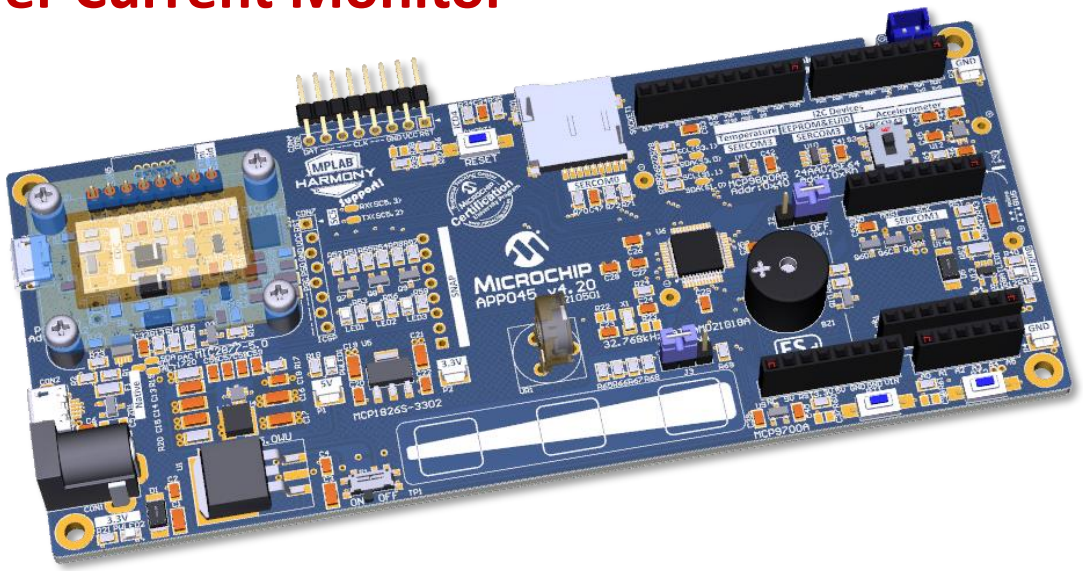
- Direct Memory Access Controller (DMAC) - Single Transfer Block
  - 💻Lab13-3 ADC with DMAC
  - 💻Lab13-4 ADC with DMAC and Interrupt
- Direct Memory Access Controller (DMAC) - Linked List Transfer
  - 💻Lab13-5 ADC with Descriptors Link of DMAC
- Power Debugger
- Power Manager - Sleep Mode
  - 💻Lab13-6 ADC with IDLE
  - 💻Lab13-7 ADC with STANDBY
- Application : Sleepwalking
  - 💻Lab13-8_ADC_with_DV
  - 💻Lab13-9_ADC_with_DV_DMAC_SleepWalking
  - 💻Lab13-10_ADC_with_DV_DMAC-Linked_SleepWalking

SAMD21 EVB APP045 v4
Introduction

MICROCHIP

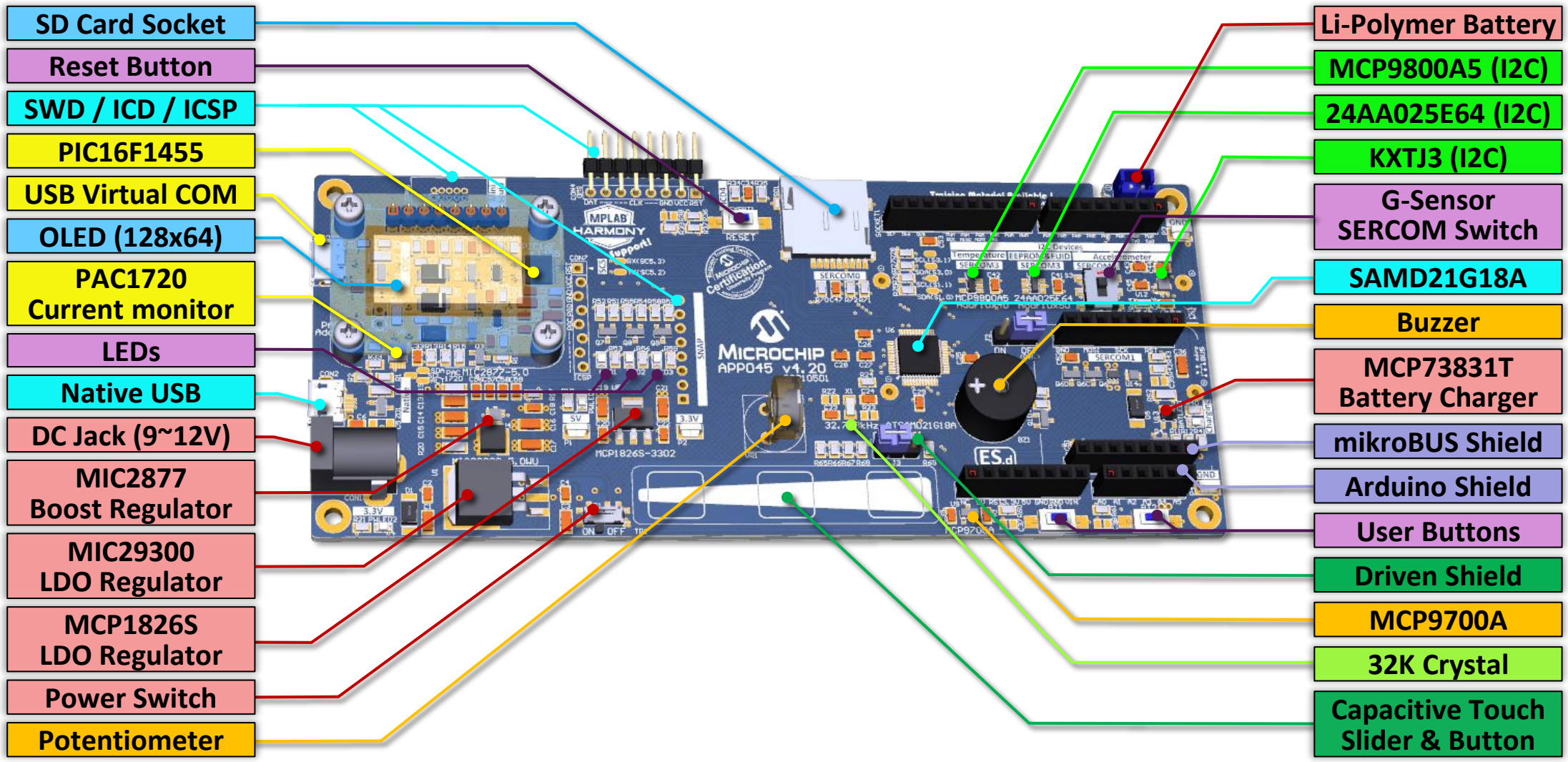
APP045 v4 Features

- 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)



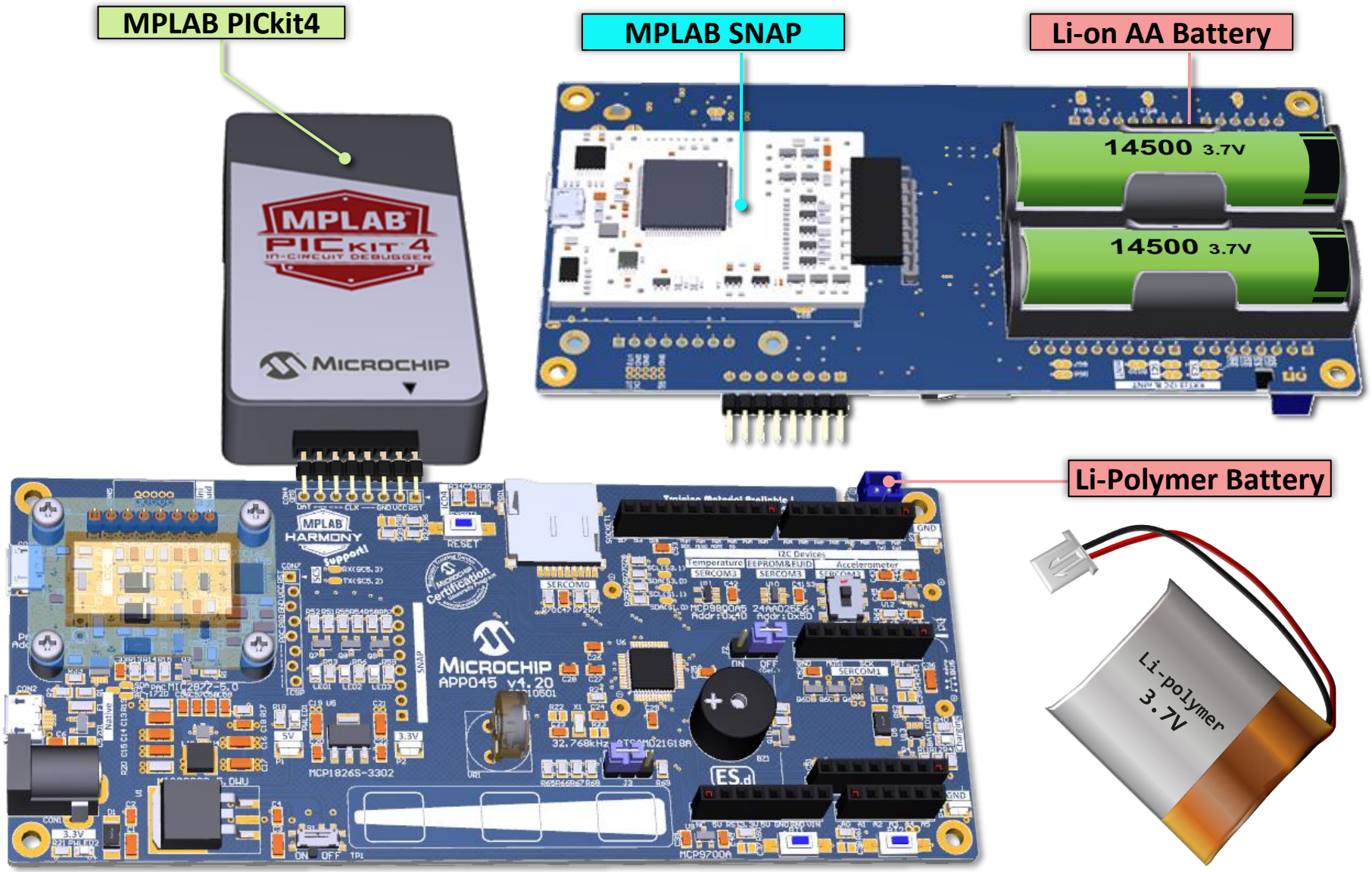
MICROCHIP

APP045 v4 Overview

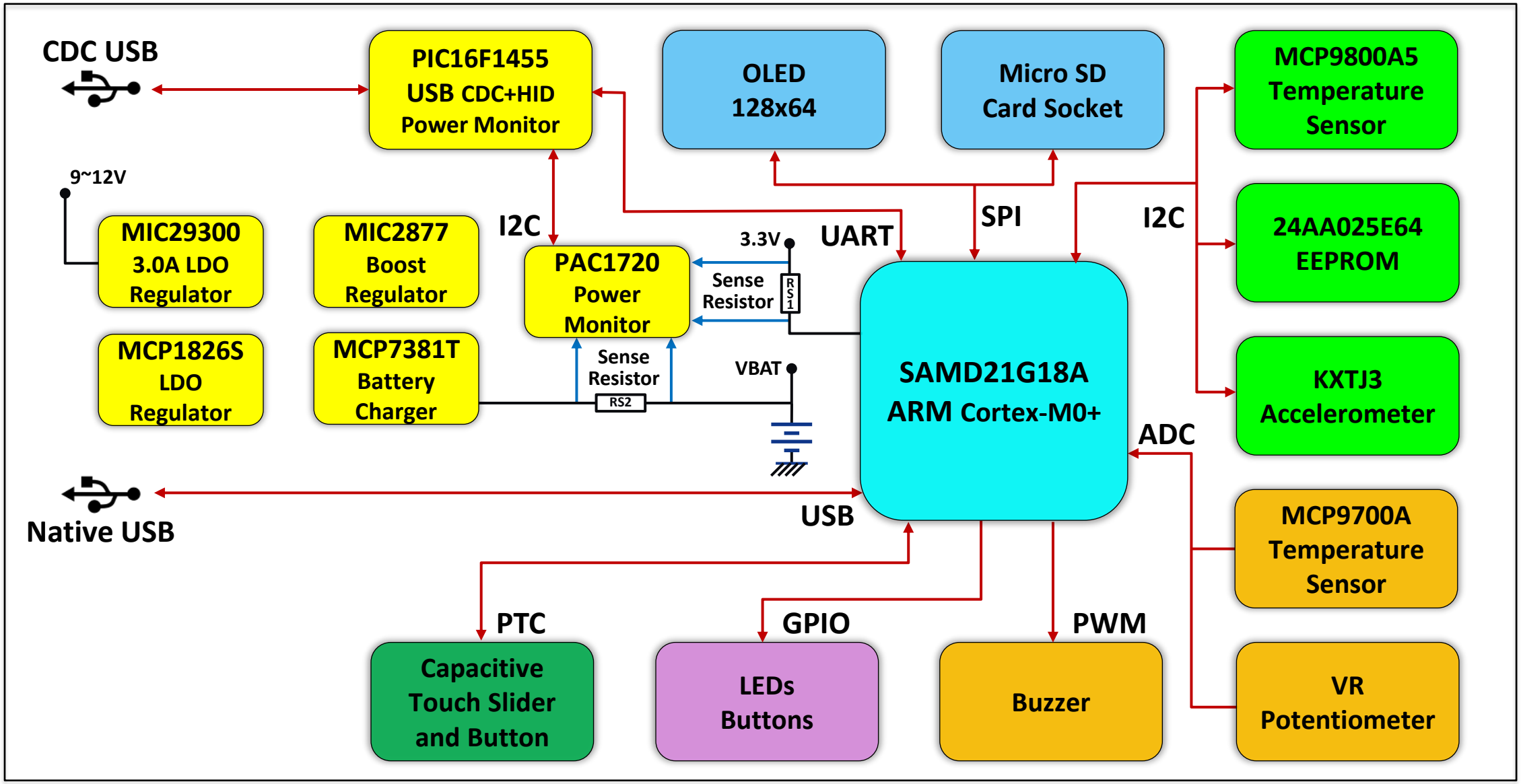


MICROCHIP

APP045 v4 Programmer and Debugger



APP045 v4 Block Diagram



Timer/Counter Operation Mode

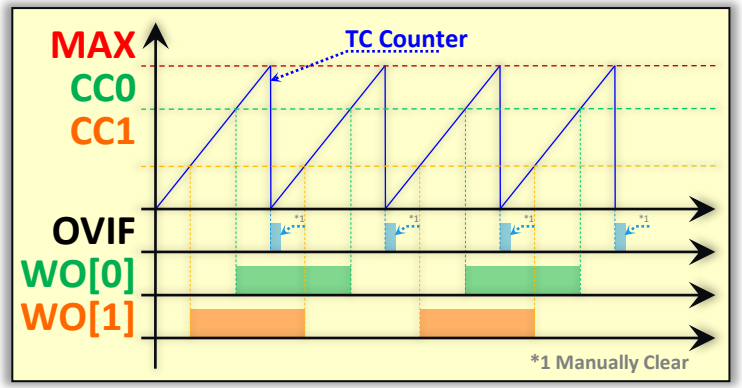
Review the Timer/Counter

- **The TC module consists 3 functions “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)

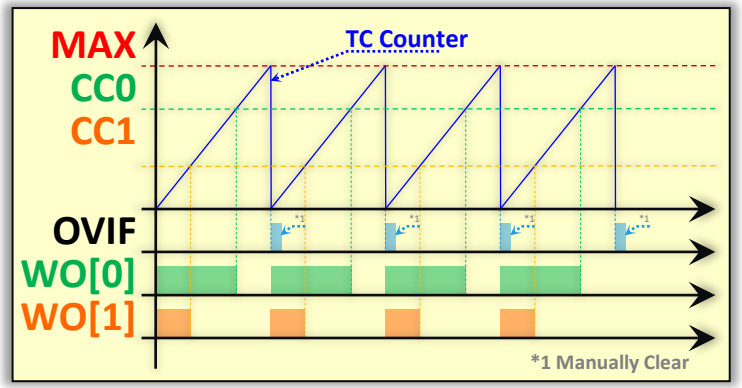
Compare 4 waveform generation mode

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

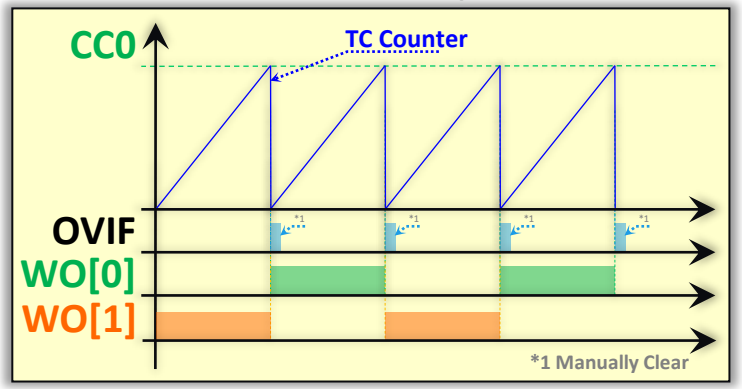
NFRQ, 16Bit, Up Count



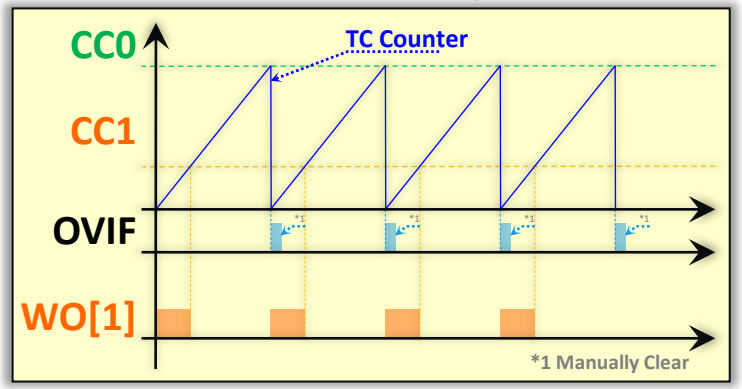
NPWM, 16Bit, Up Count



MFRQ, 16Bit, Up Count



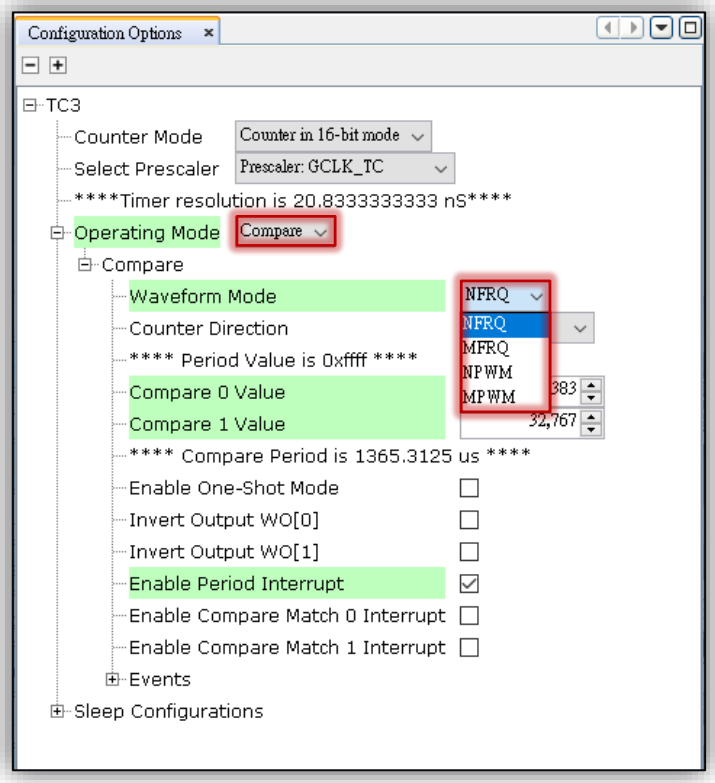
MPWM, 16Bit, Up Count



MICROCHIP

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 MCC-Harmony,**
  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**)



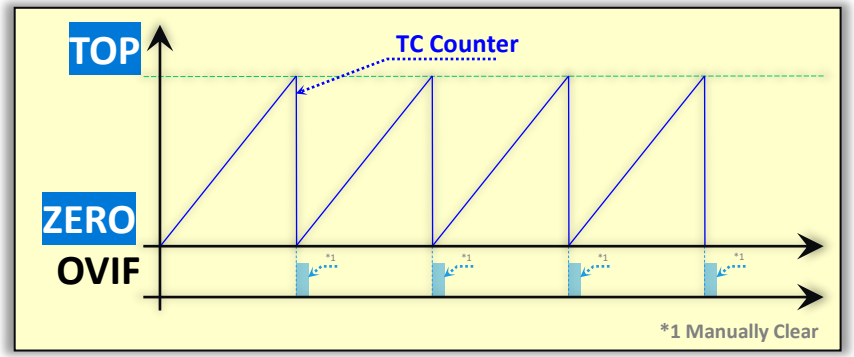
UP/DOWN Count and Counter Update

- 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

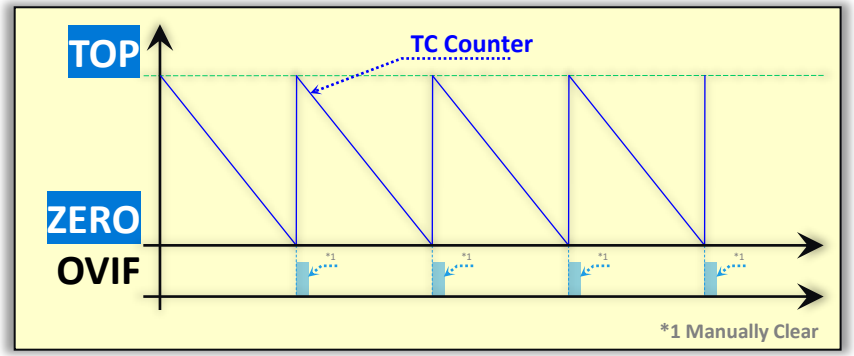
    ** If (Counter == TOP) Counter = 0;*
- For DOWN Count Mode
  - **Timing :** Counter = ZERO
  - **Result :** was updated to TOP

    ** If (Counter == ZREO) Counter = TOP;*

UP Count Mode



DOWN Count Mode



MICROCHIP

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])

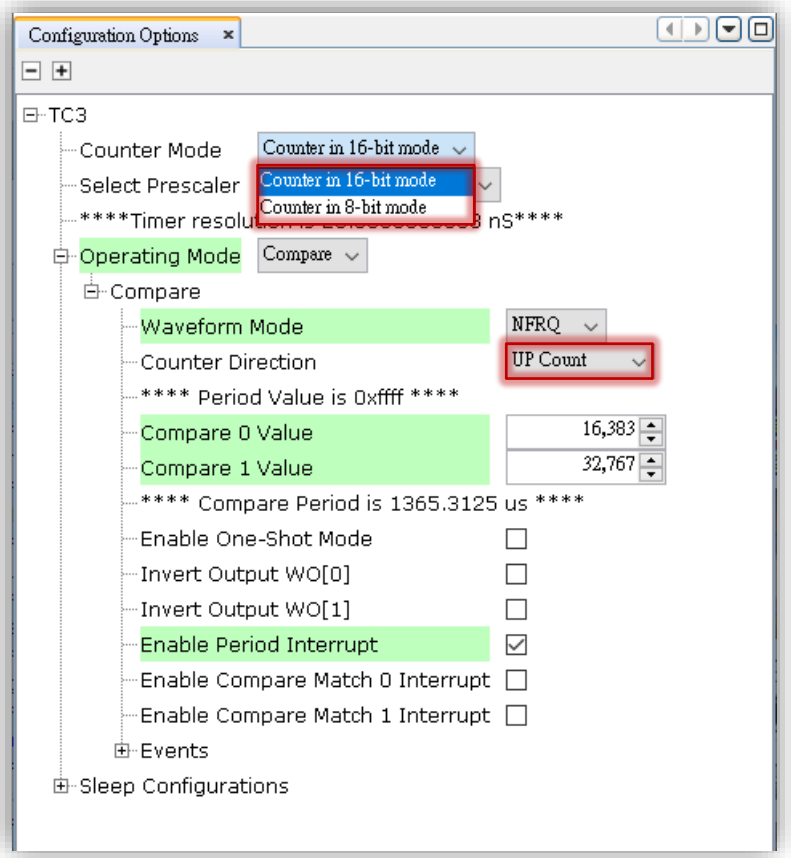
- **ZERO**
  - The **ZERO** always “0”.

	TOP			ZERO
	8-bits	16-bits	32-bits	
NFRQ	PER	MAX(2^{16}-1)	MAX(2^{32}-1)	0
MFRQ	$CC0_{(8\text{-bits})}$	$CC0_{(16\text{-bits})}$	$CC0_{(32\text{-bits})}$	0
NPWM	PER	MAX(2^{16}-1)	MAX(2^{32}-1)	0
MPWM	$CC0_{(8\text{-bits})}$	$CC0_{(16\text{-bits})}$	$CC0_{(32\text{-bits})}$	0

MICROCHIP

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

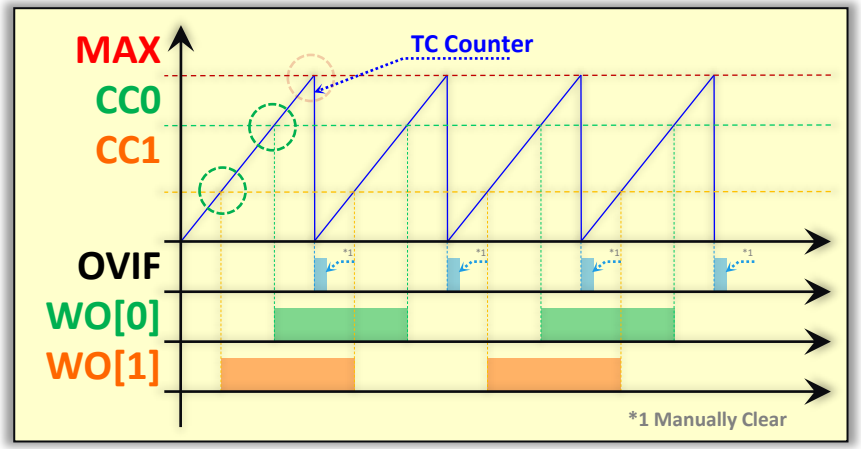
DOWN Count Mode

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

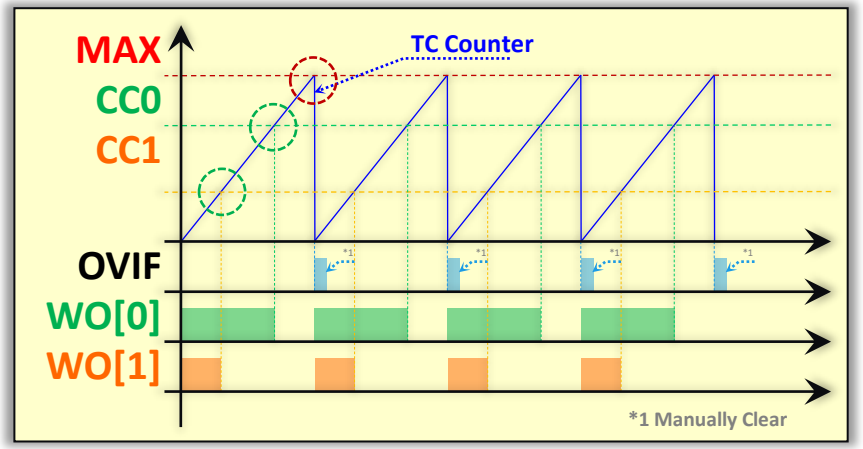
MICROCHIP

Waveform at difference mode - cont.

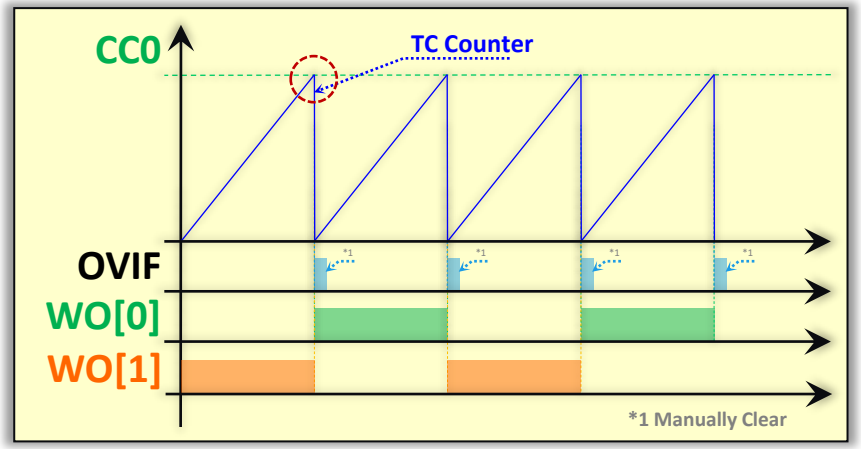
NFRQ, 16Bit, Up Count



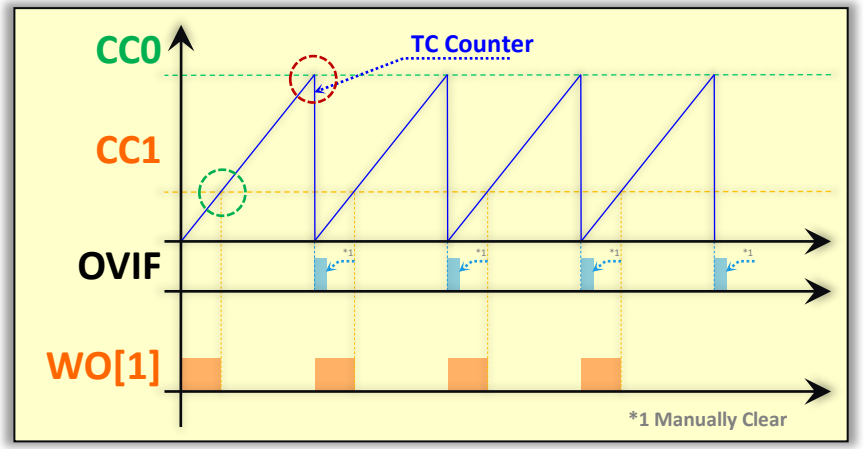
NPWM, 16Bit, Up Count



MFRQ, 16Bit, Up Count



MPWM, 16Bit, Up Count



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

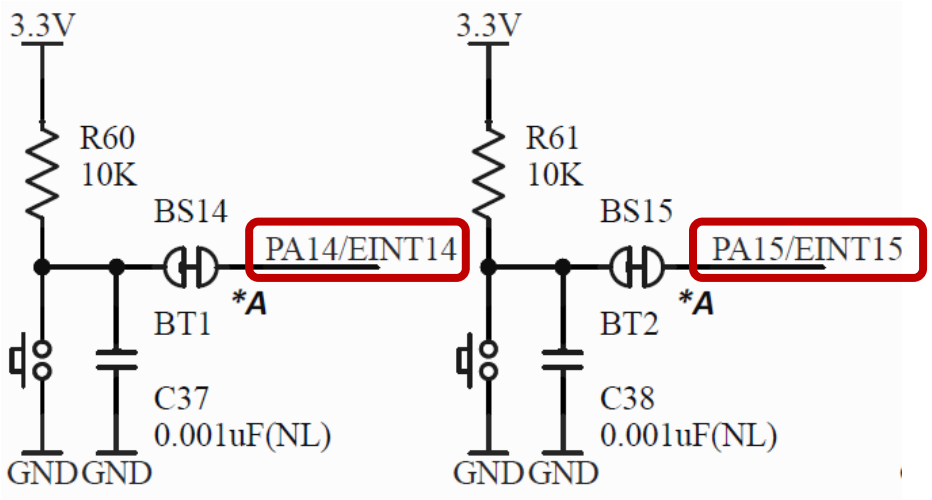
MICROCHIP

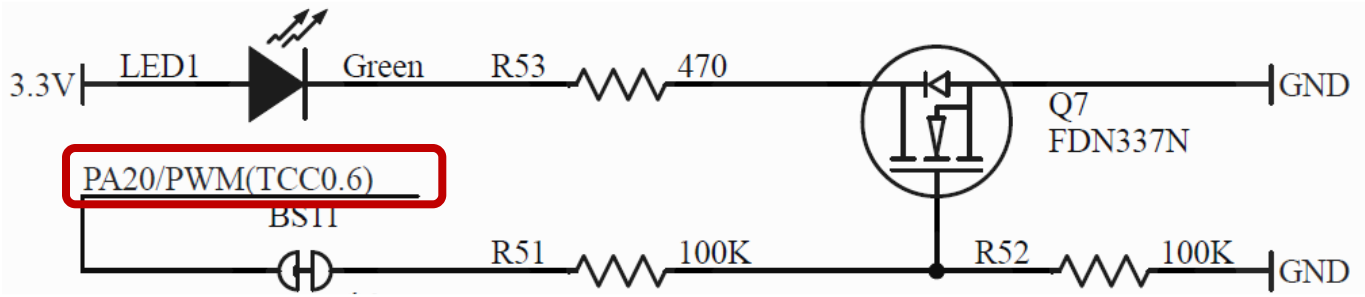
🛠 Lab6-1 TC Operation Mode - NFRQ

- 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

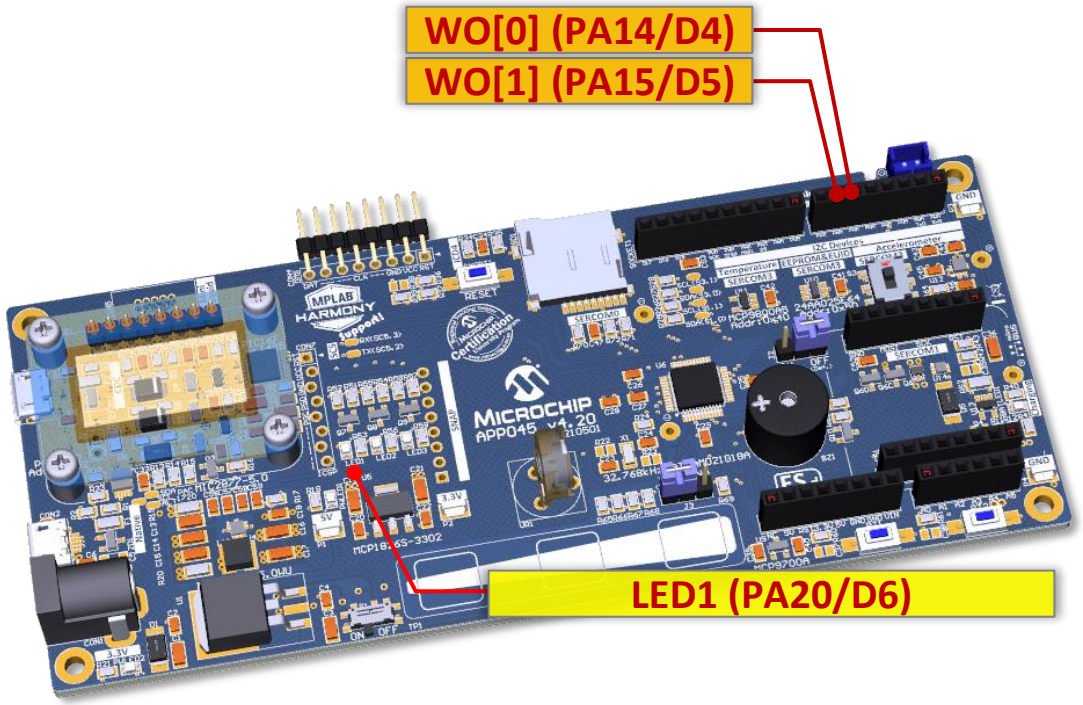
- **Let’s go !**

Lab6-1 TC Operation Mode - NFRQ





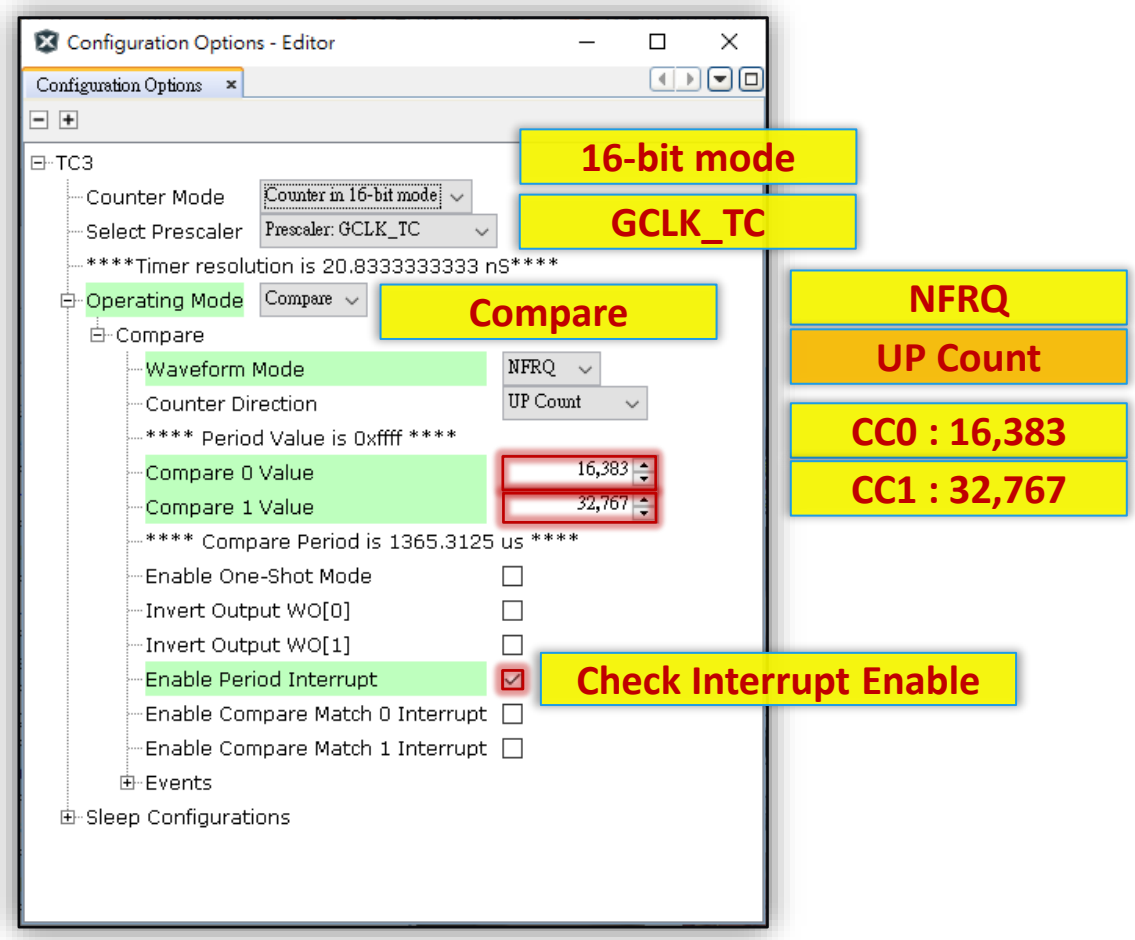
		SOCKET1B ARDUINO-M0-PRO
PA11/RX(S2.3)	7	D0/PWM/RX
PA10/TX(S2.2)	8	D1/PWM/TX
PA08/MISO(S0.0)	9	D2/PWM
PA09/SS(S0.1)	10	D3/PWM
PA14/EINT14	11	D4/PWM
PA15/EINT15	12	D5/PWM
PA20/PWM(TCC0.6)	13	D6/PWM
PA21/PWM(TCC0.7)	14	D7/PWM



🛠 Lab6-1 TC Operation Mode - NFRQ

Step 1

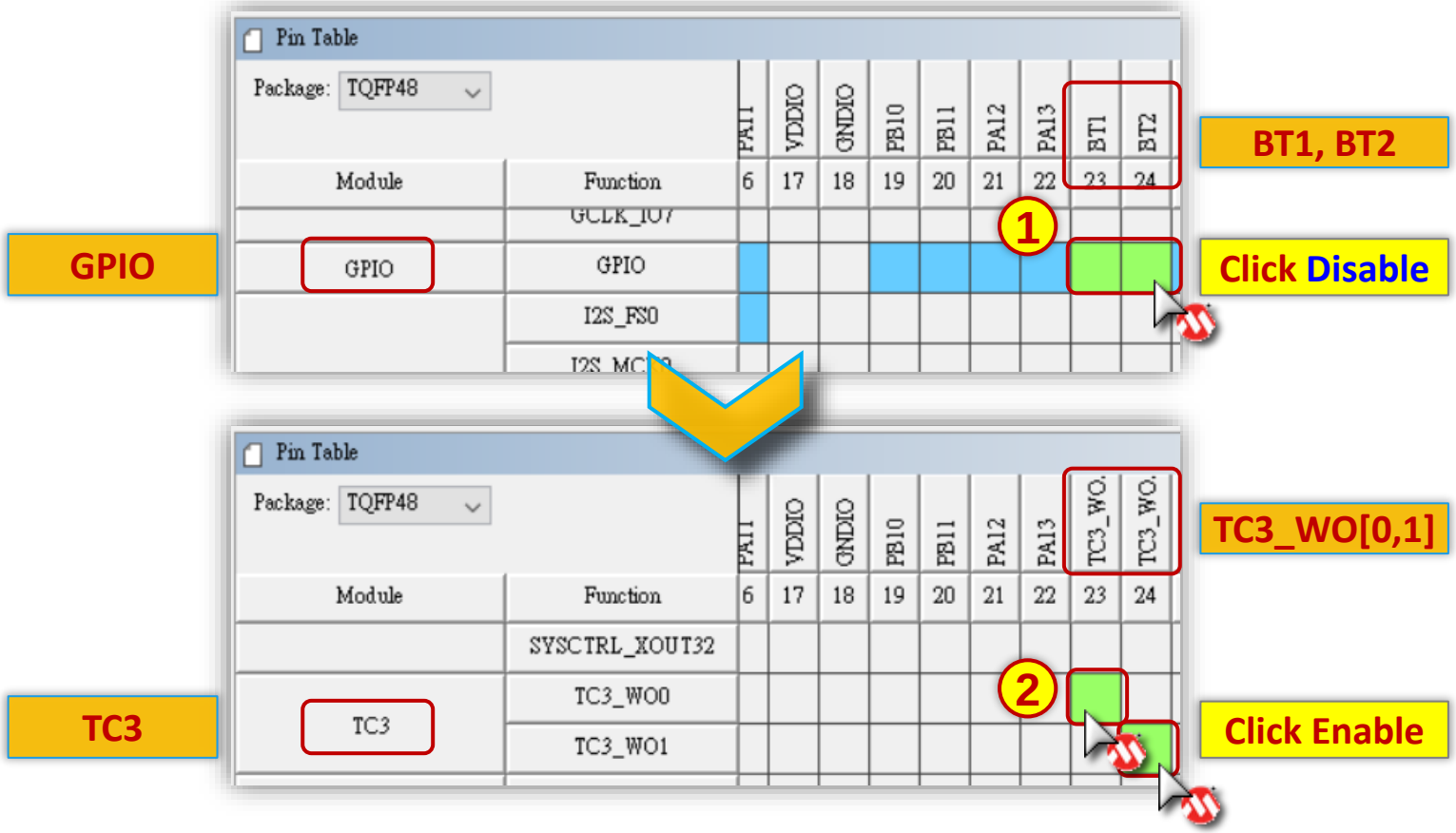
- **Configure TC3 Period to NFRQ mode and Enable Interrupts in configuration.**



Lab6-1 TC Operation Mode - NFRQ

Step 2

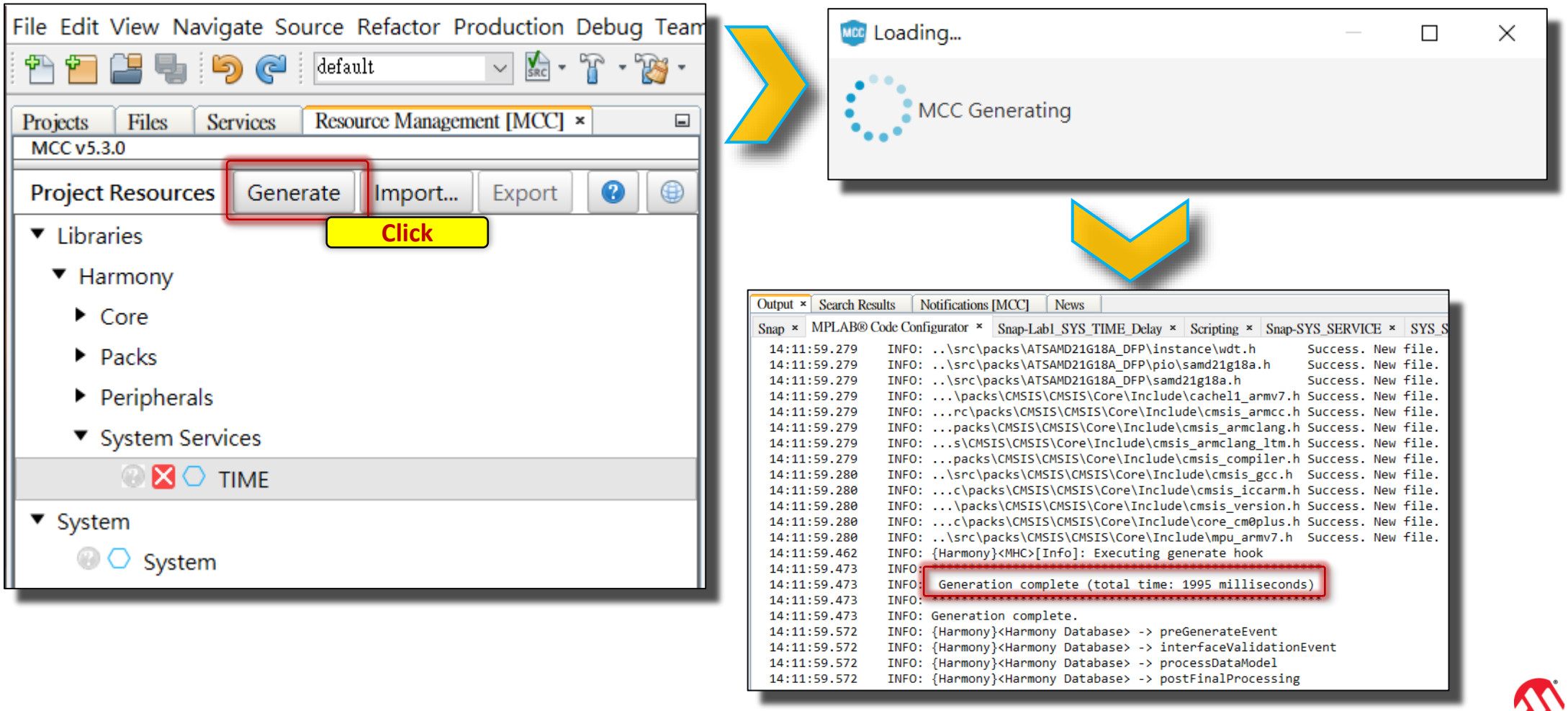
- Disable BT1(PA14) and BT2(PA15) GPIO function firstly.
- Enable TC3 Waveform Output 0 (W0[0], TC3_WO0) and TC3 Waveform Output 1 (W0[1], TC3_WO1) to PA14, PA15



🛠 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

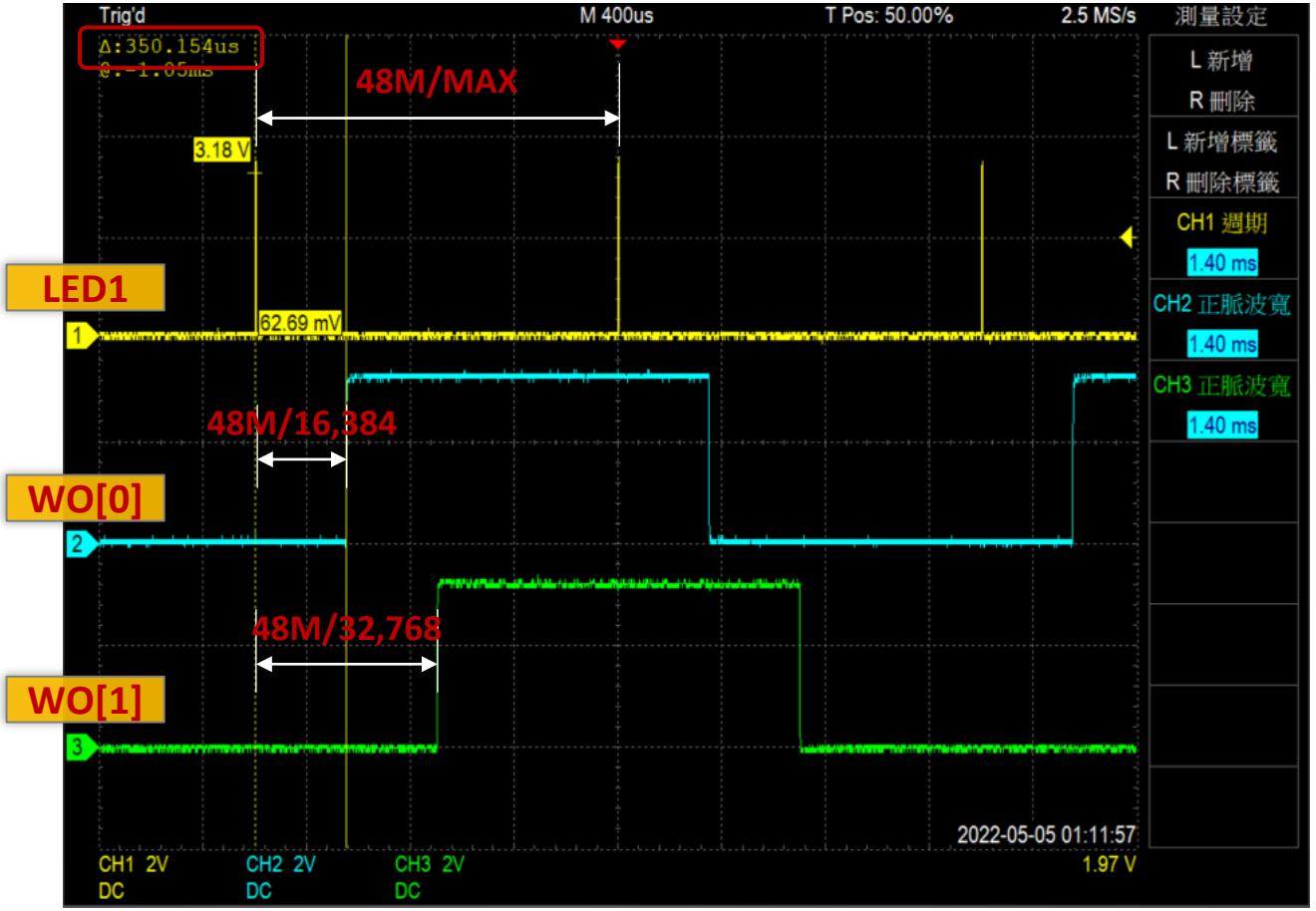
Step 5

```
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);
}
```

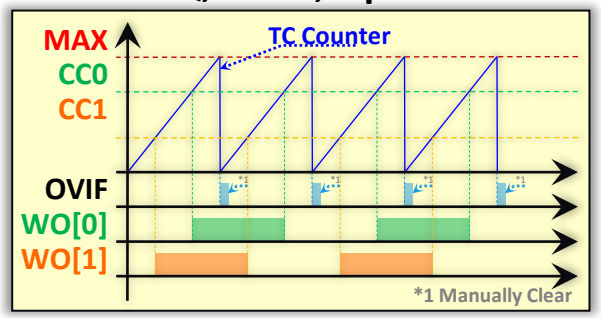
Lab6-1 TC Operation Mode - NFRQ

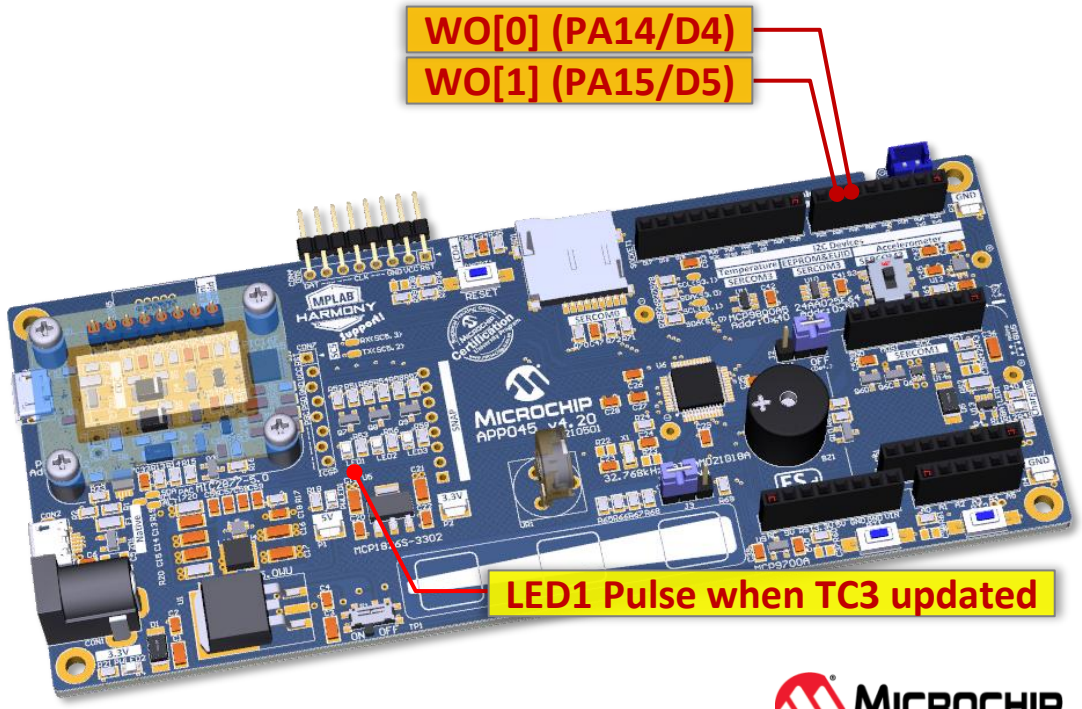
Result

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



NFRQ, 16Bit, Up Count





🛠 Lab6-2 TC Operation Mode - MFRQ

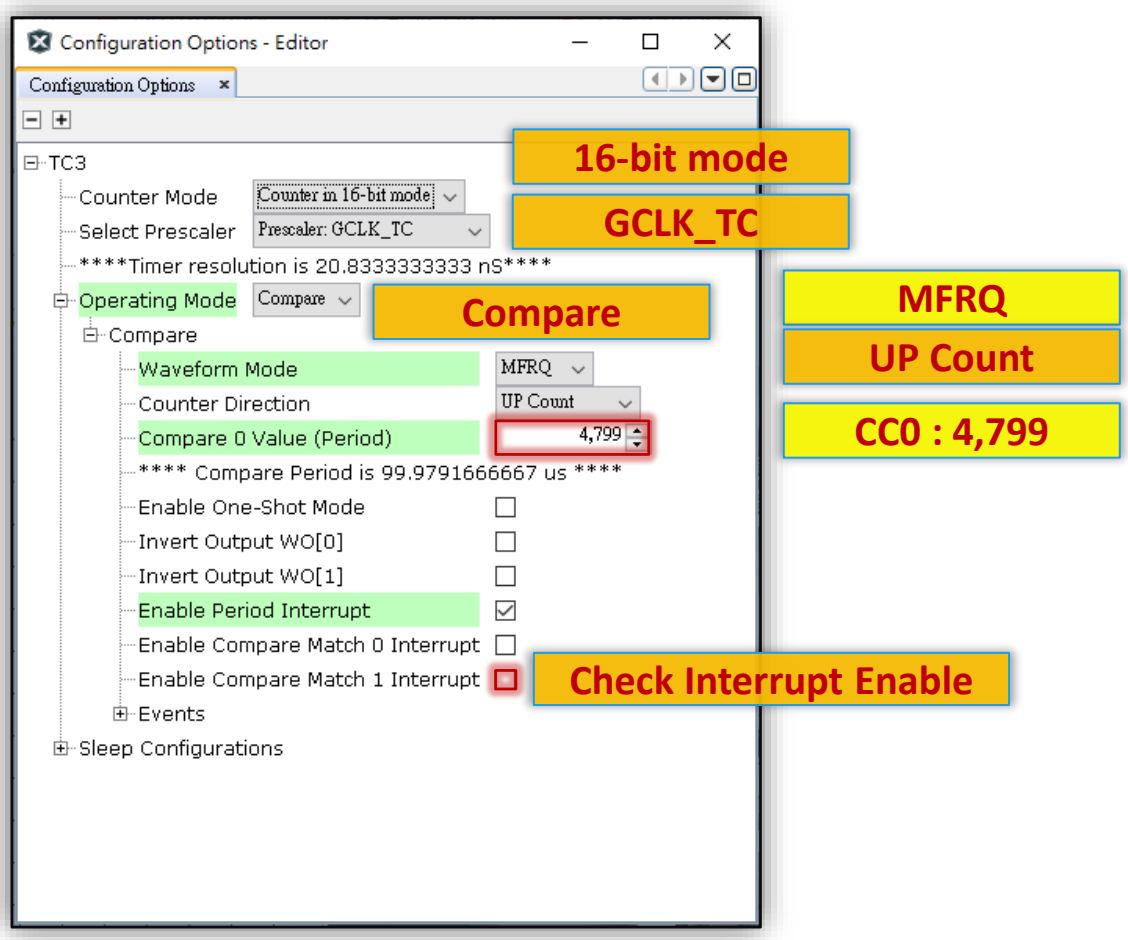
- Please continue from SAM2001 Lab6-1 (TC Compare Mode - NFRQ)
- Try to config TC3 to
  Compare - MFRQ (Match Frequency Generation) mode
  - CC0 -> 4,799

- **Let’s go !**

🛠 Lab6-2 TC Operation Mode - MFRQ

Step 1

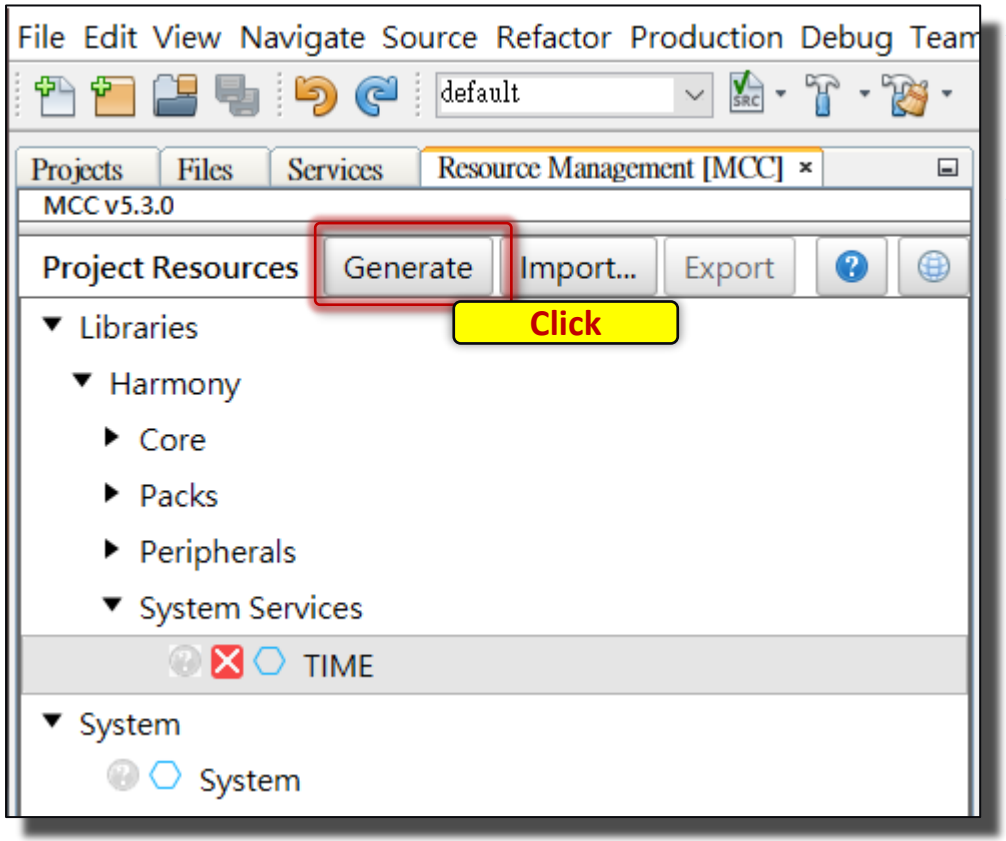
- **Configure TC3 Period to MFRQ mode in configuration.**

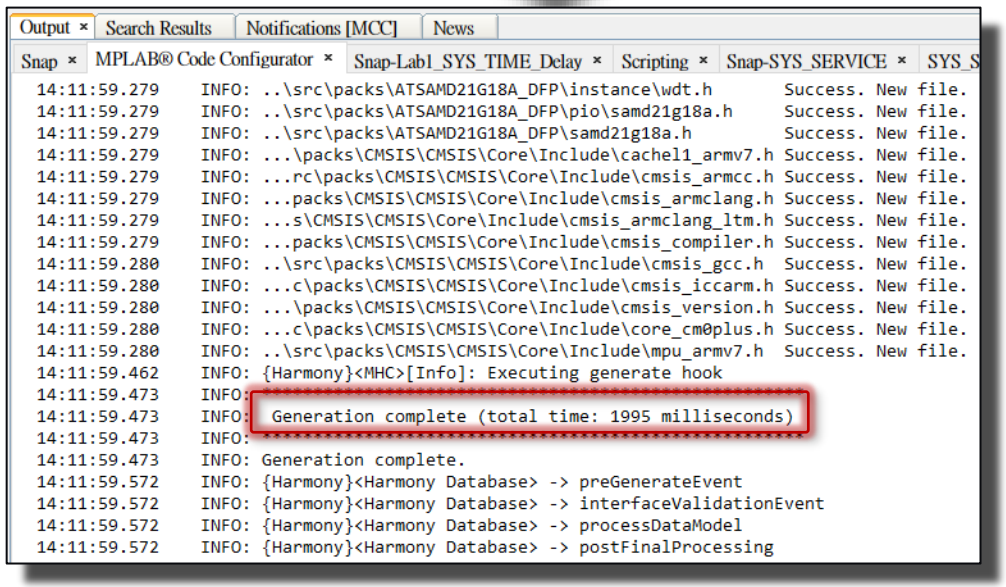


🛠 Lab6-2 TC Operation Mode - MFRQ

Step 1

- **Click to Generate Code.**

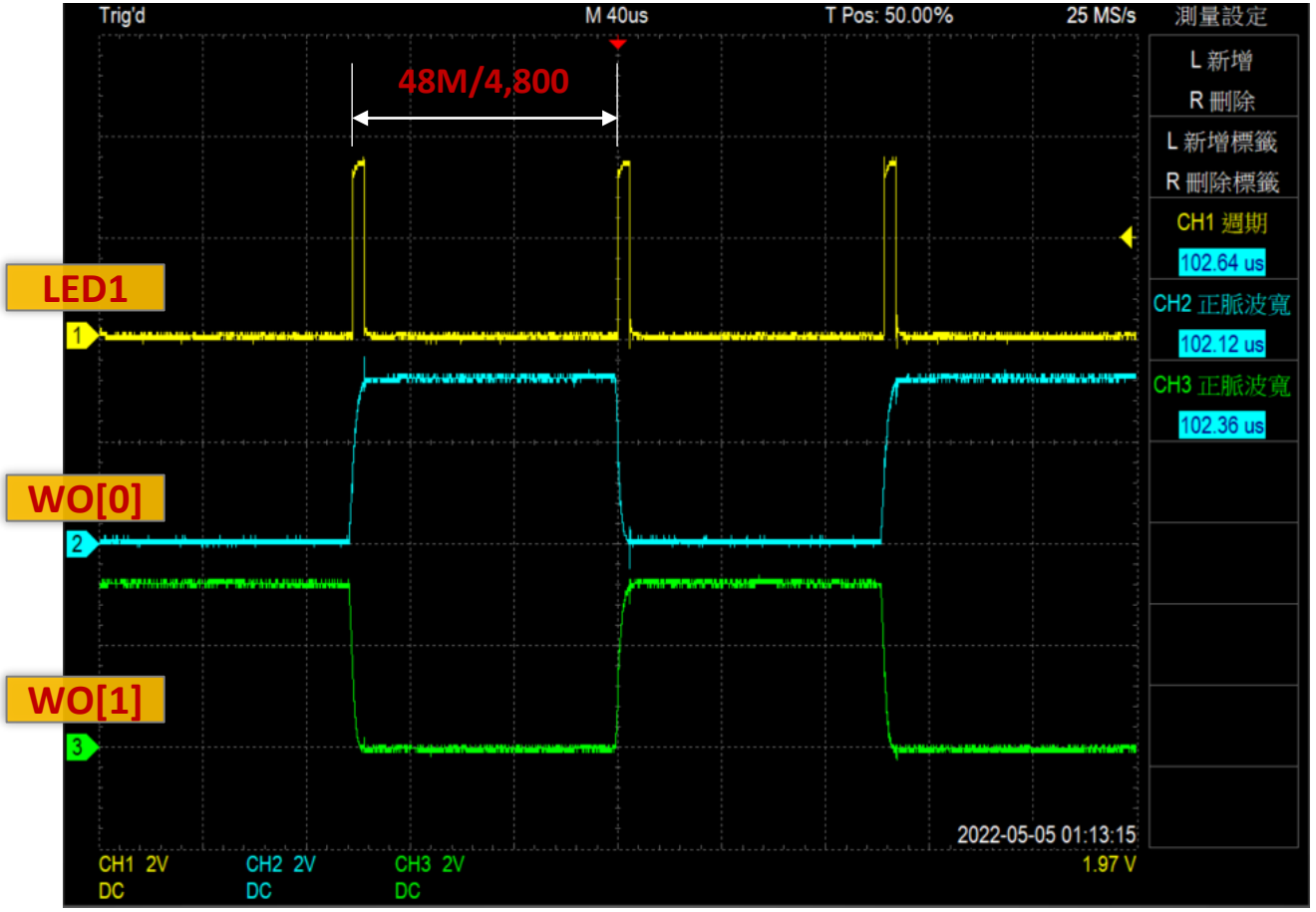




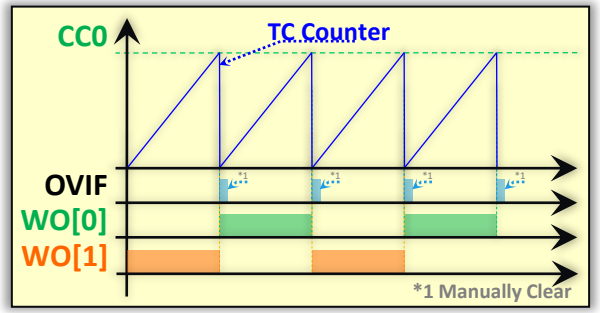
🛠 Lab6-2 TC Operation Mode - MFRQ

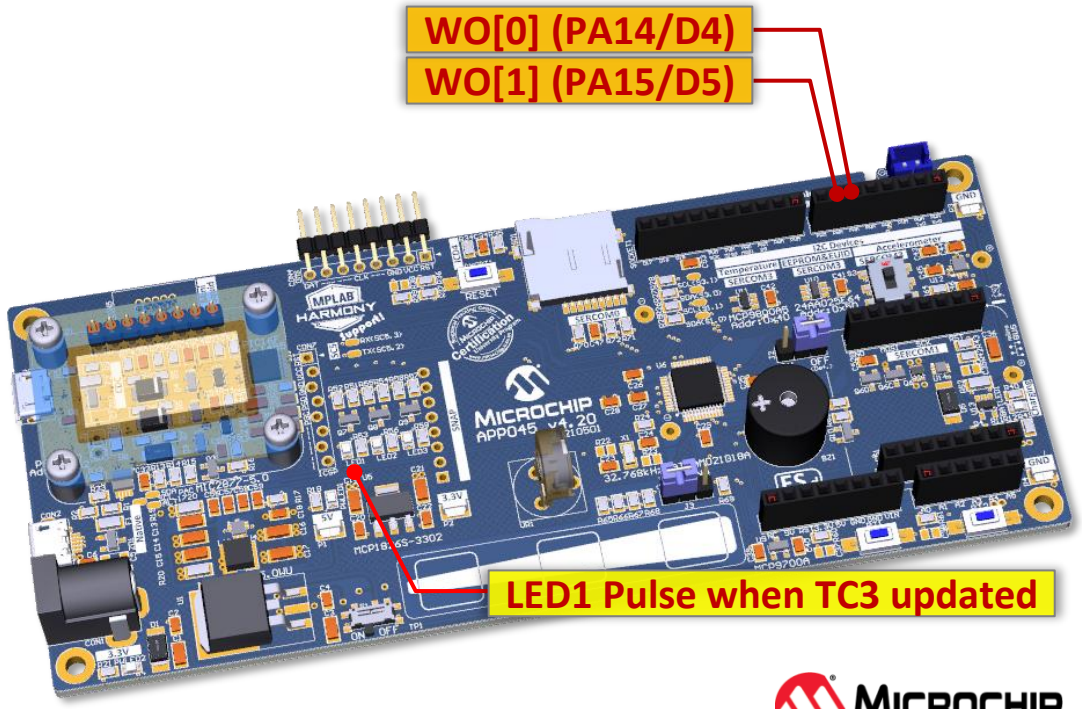
Result

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



MFRQ, 16Bit, Up Count





MICROCHIP

🛠 Lab6-3 TC Operation Mode - NPWM

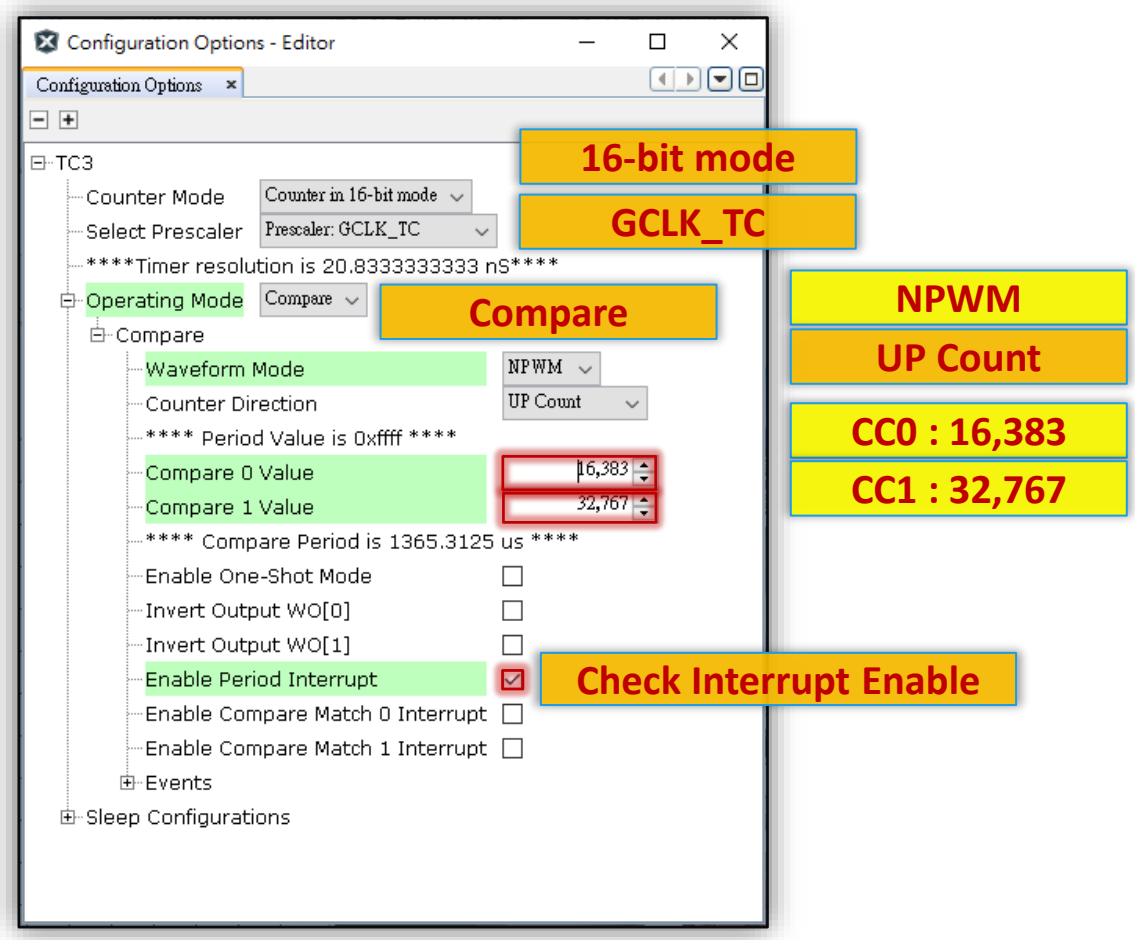
- Please continue from SAM2001 Lab6-1 (TC Compare Mode - NFRQ)
- Try to config TC3 to
  Compare - NPWM (Normal Pulse-Width Modulation Operation) mode
  - CC0 -> 16,383
  - CC1 -> 32,767

- **Let’s go !**

🛠 Lab6-3 TC Operation Mode - NPWM

Step 1

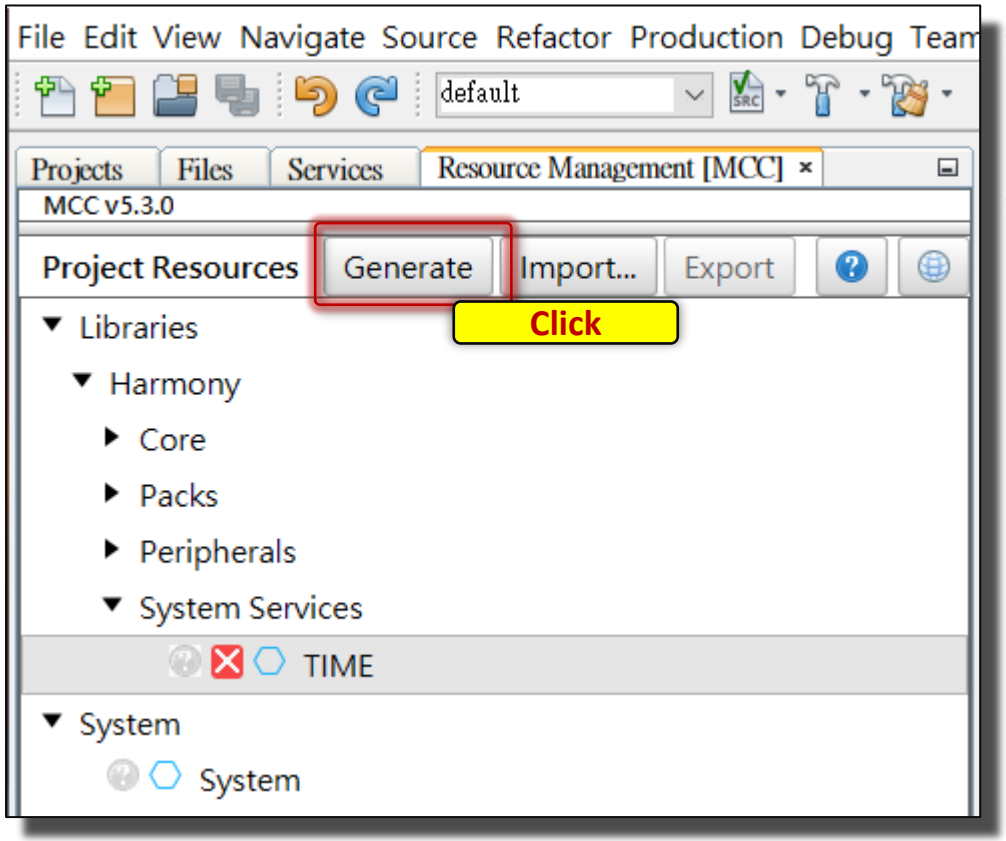
- **Configure TC3 Period to NPWM mode and Enable Interrupts in configuration.**

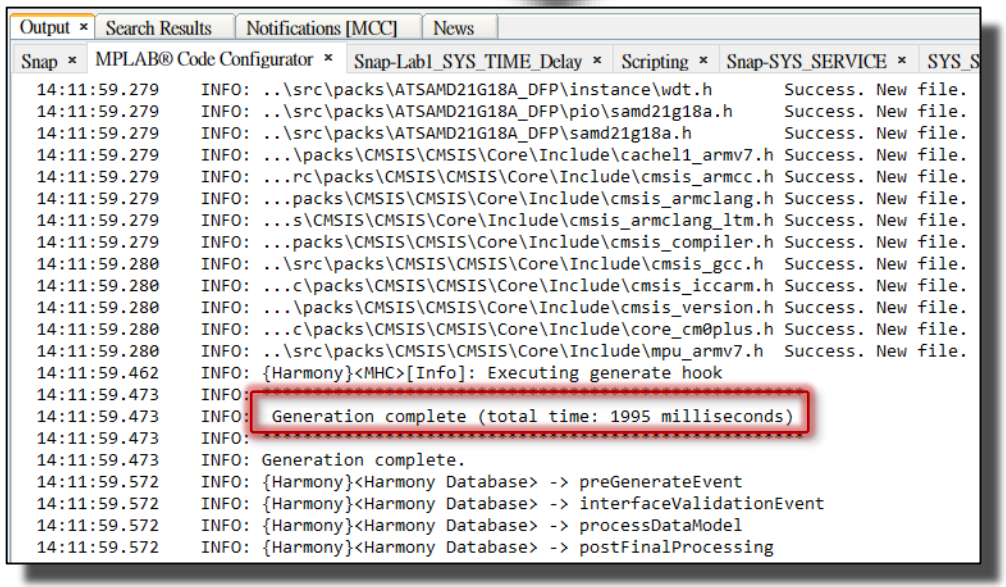


🛠 Lab6-3 TC Operation Mode - NPWM

Step 2

- **Click to Generate Code.**

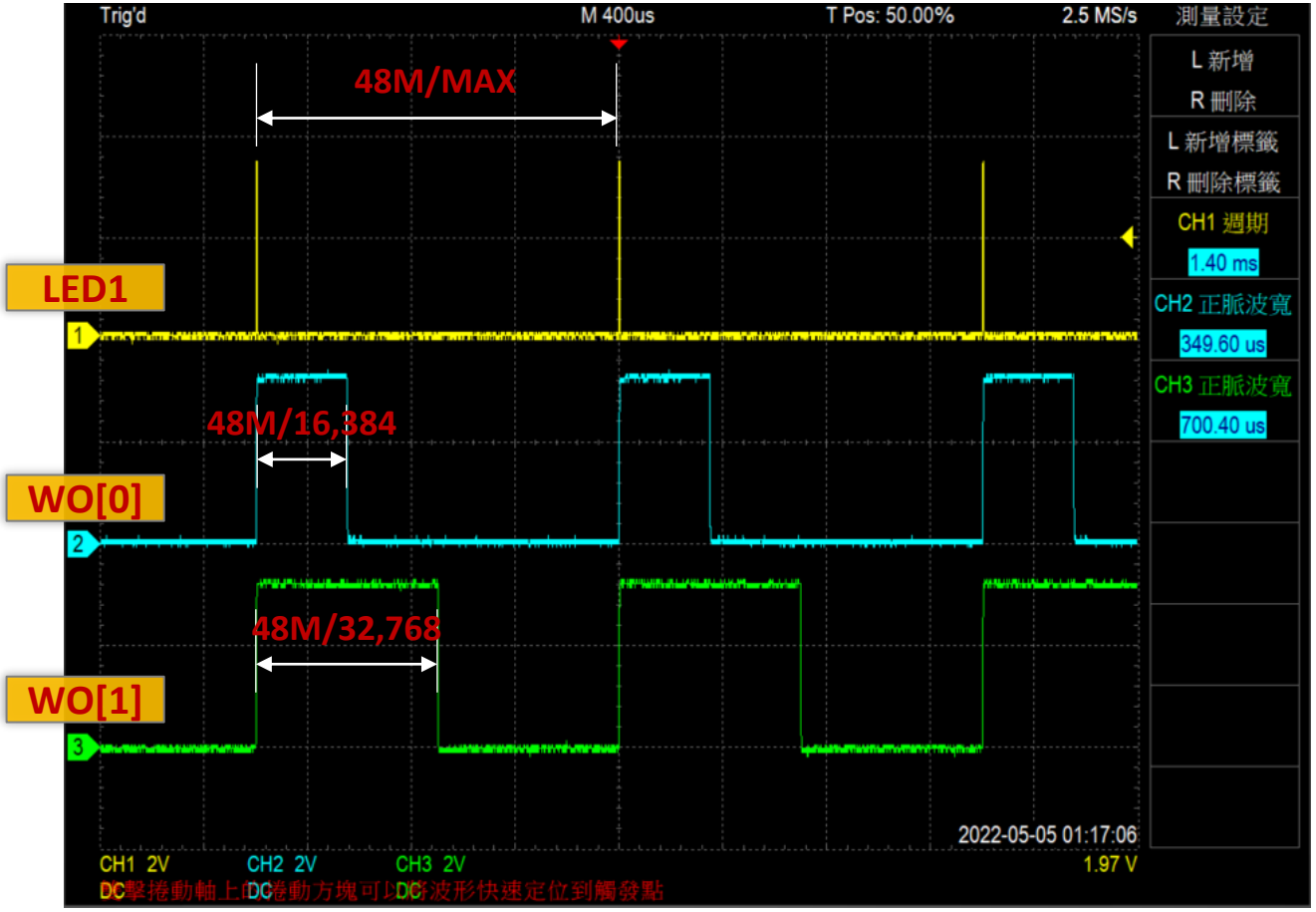




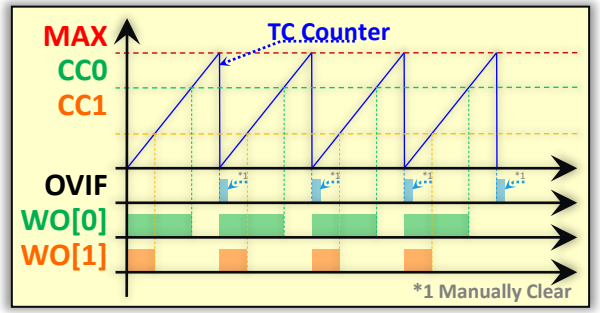
🛠 Lab6-3 TC Operation Mode - NPWM

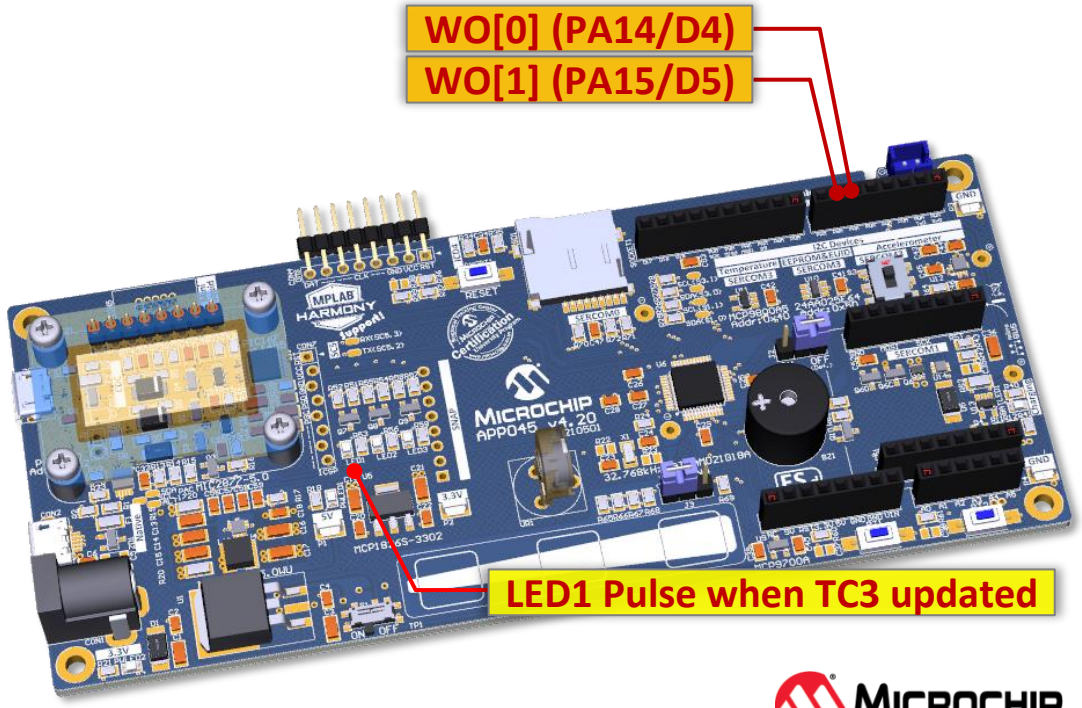
Result

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



NPWM, 16Bit, Up Count





MICROCHIP

🛠 Lab6-4 TC Operation Mode - MPWM

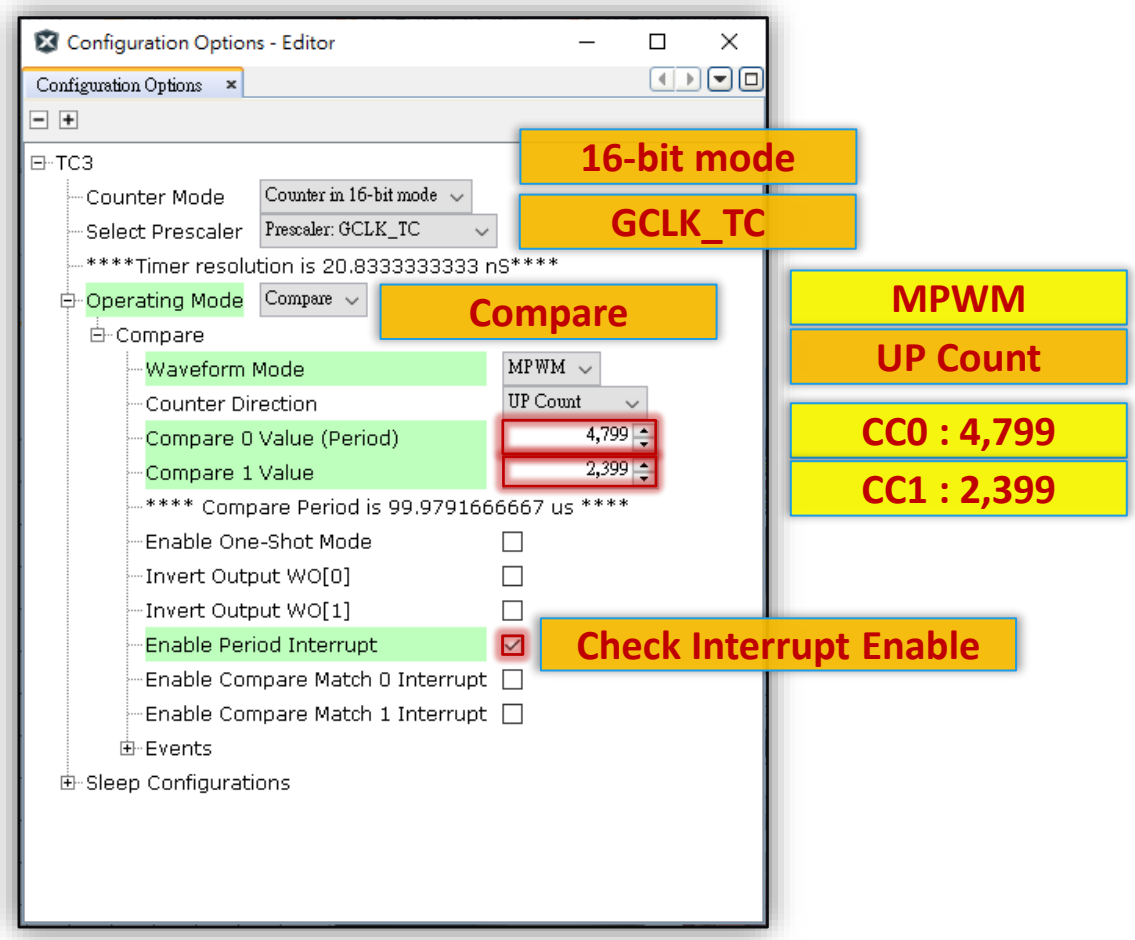
- Please continue from SAM2001 Lab6-1 (TC Compare Mode - NFRQ)
- Try to config TC3 to
  Compare - MPWM (Match Pulse-Width Modulation Operation) mode
  - CC0 -> 4,799
  - CC1 -> 2,399

- **Let’s go !**

🛠 Lab6-4 TC Operation Mode - MPWM

Step 1

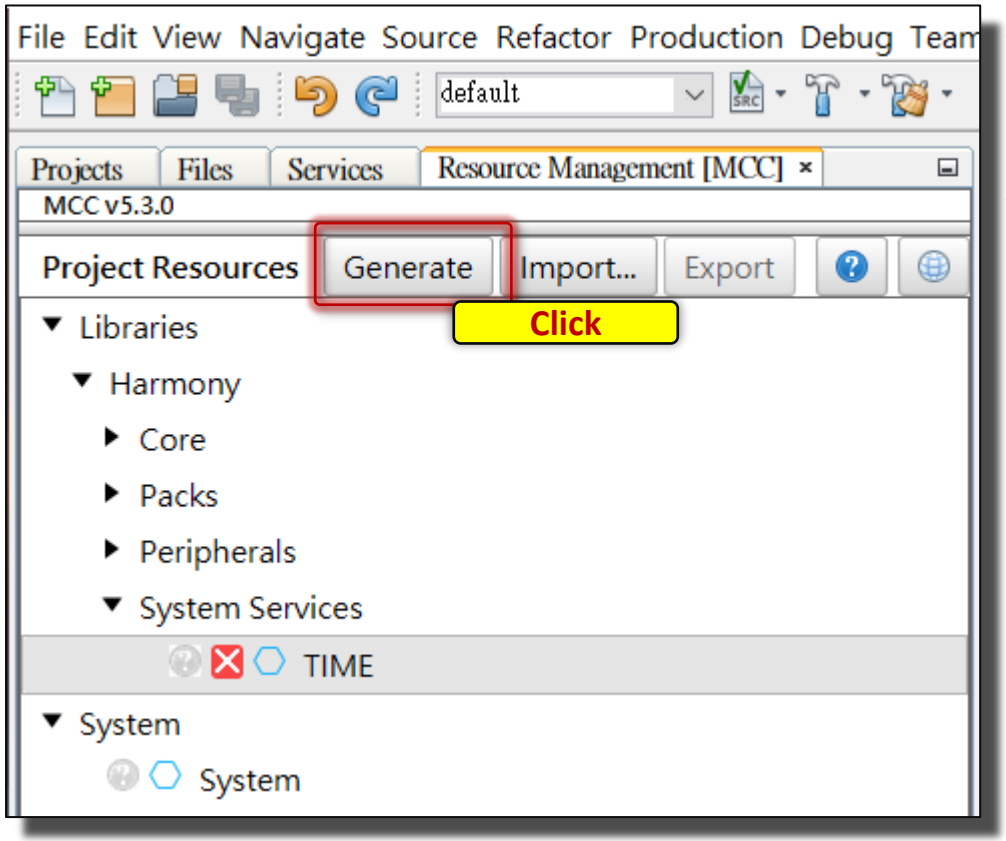
- **Configure TC3 Period to MPWM mode and Enable Interrupts in configuration.**

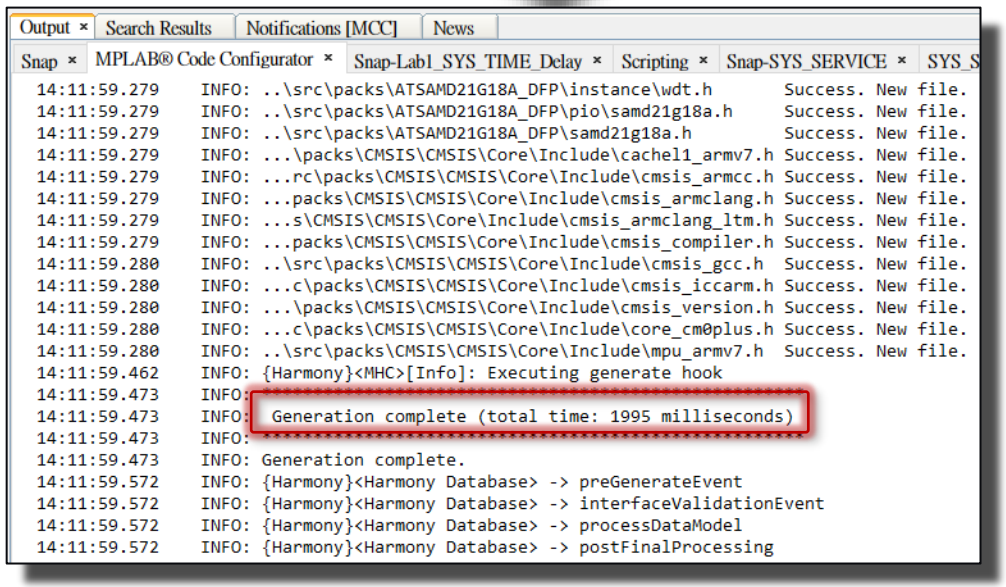


🛠 Lab6-4 TC Operation Mode - MPWM

Step 2

- **Click to Generate Code.**

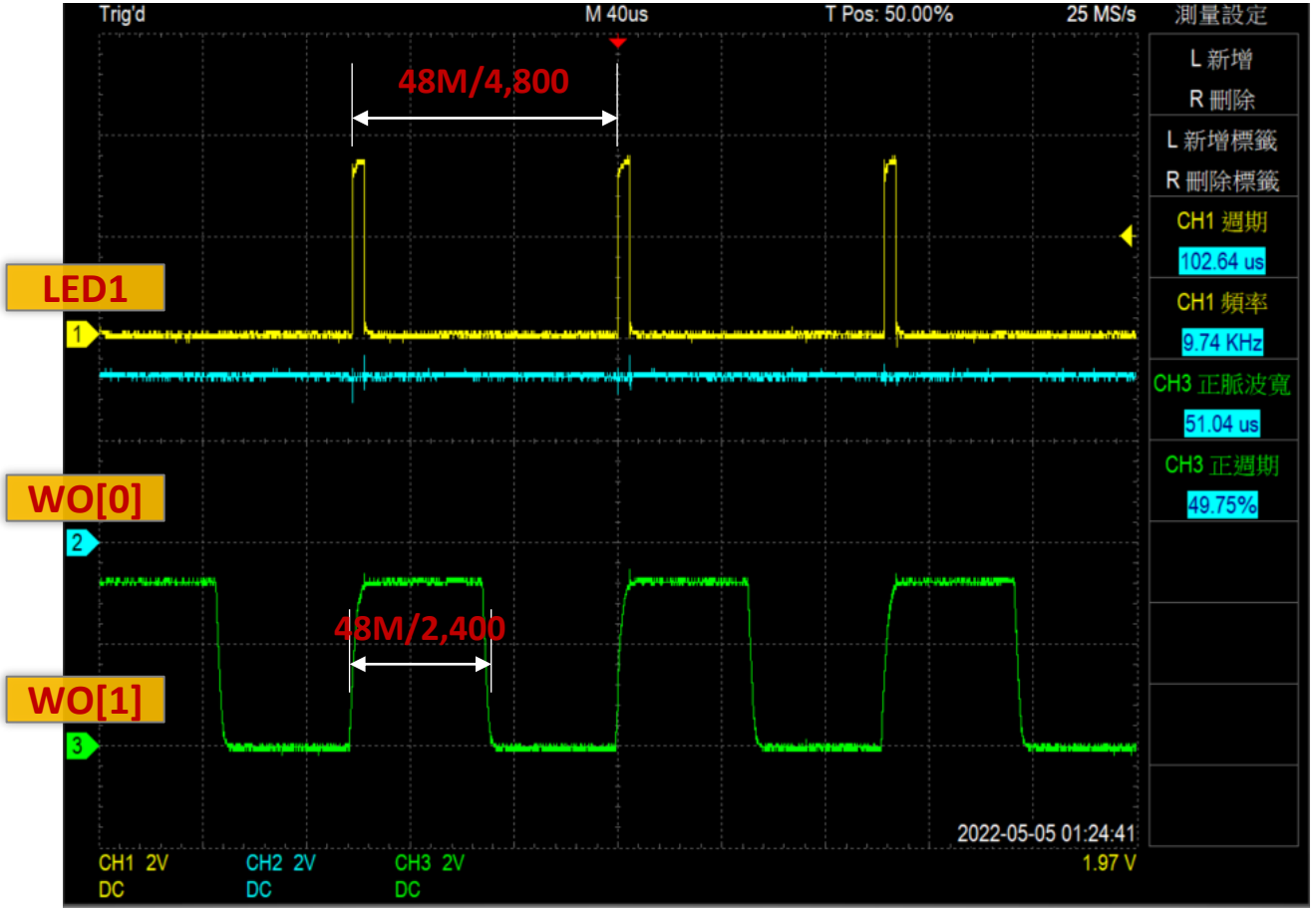




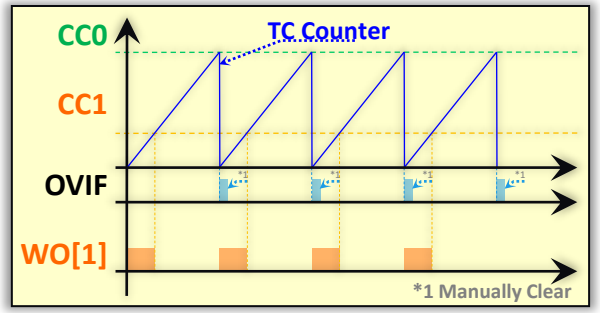
🛠 Lab6-4 TC Operation Mode - MPWM

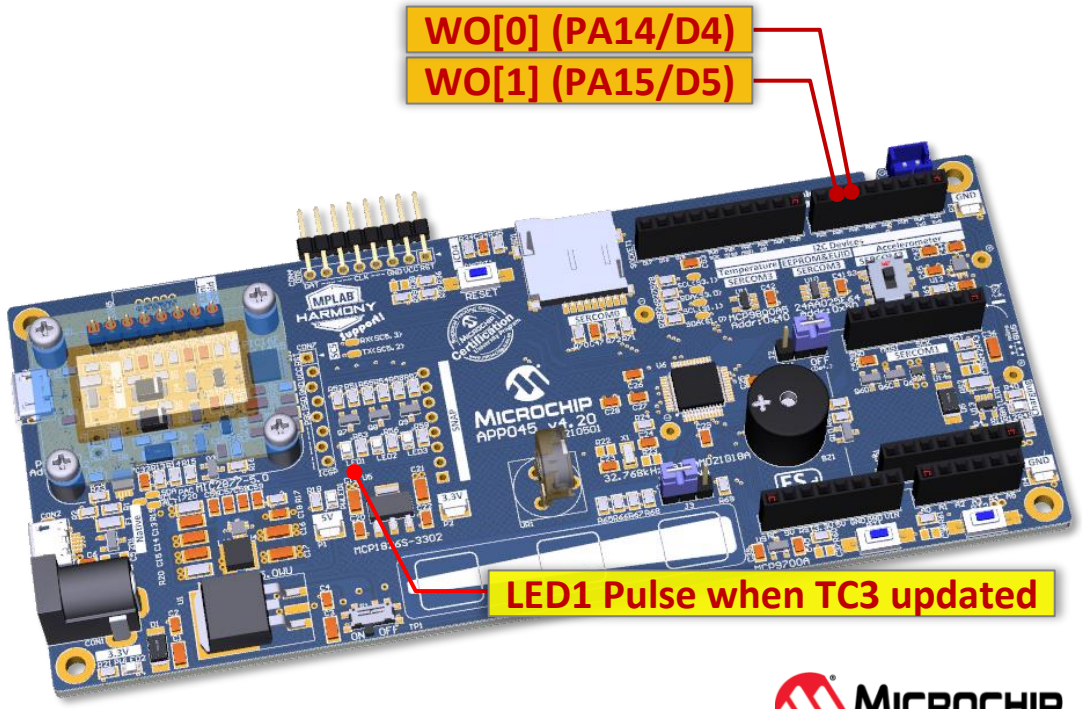
Result

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



MPWM, 16Bit, Up Count





MICROCHIP

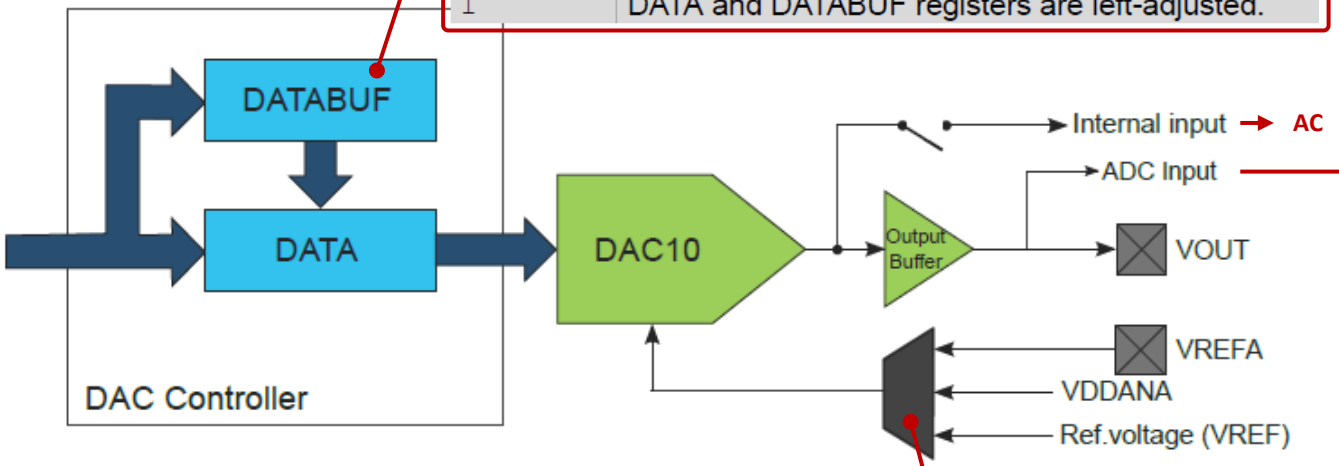
DAC Architecture

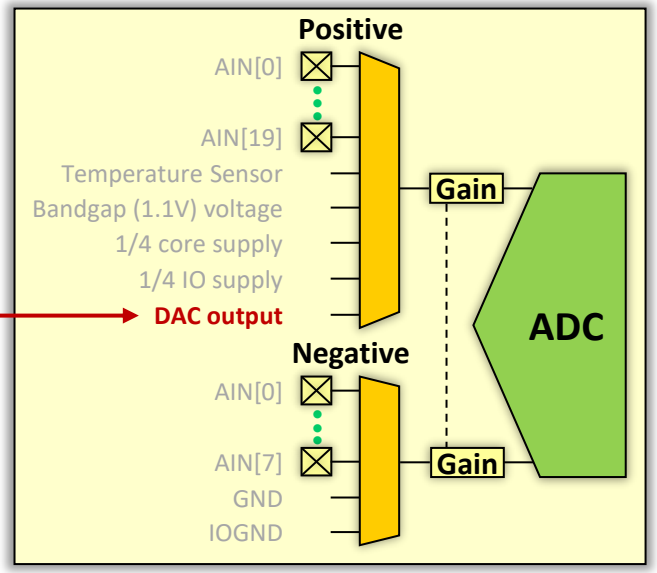
DAC, Digital-to-Analog Converter

- 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).

Fixed to Right-Adj. at Harmony

Value	Description
0	DATA and DATABUF registers are right-adjusted.
1	DATA and DATABUF registers are left-adjusted.





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

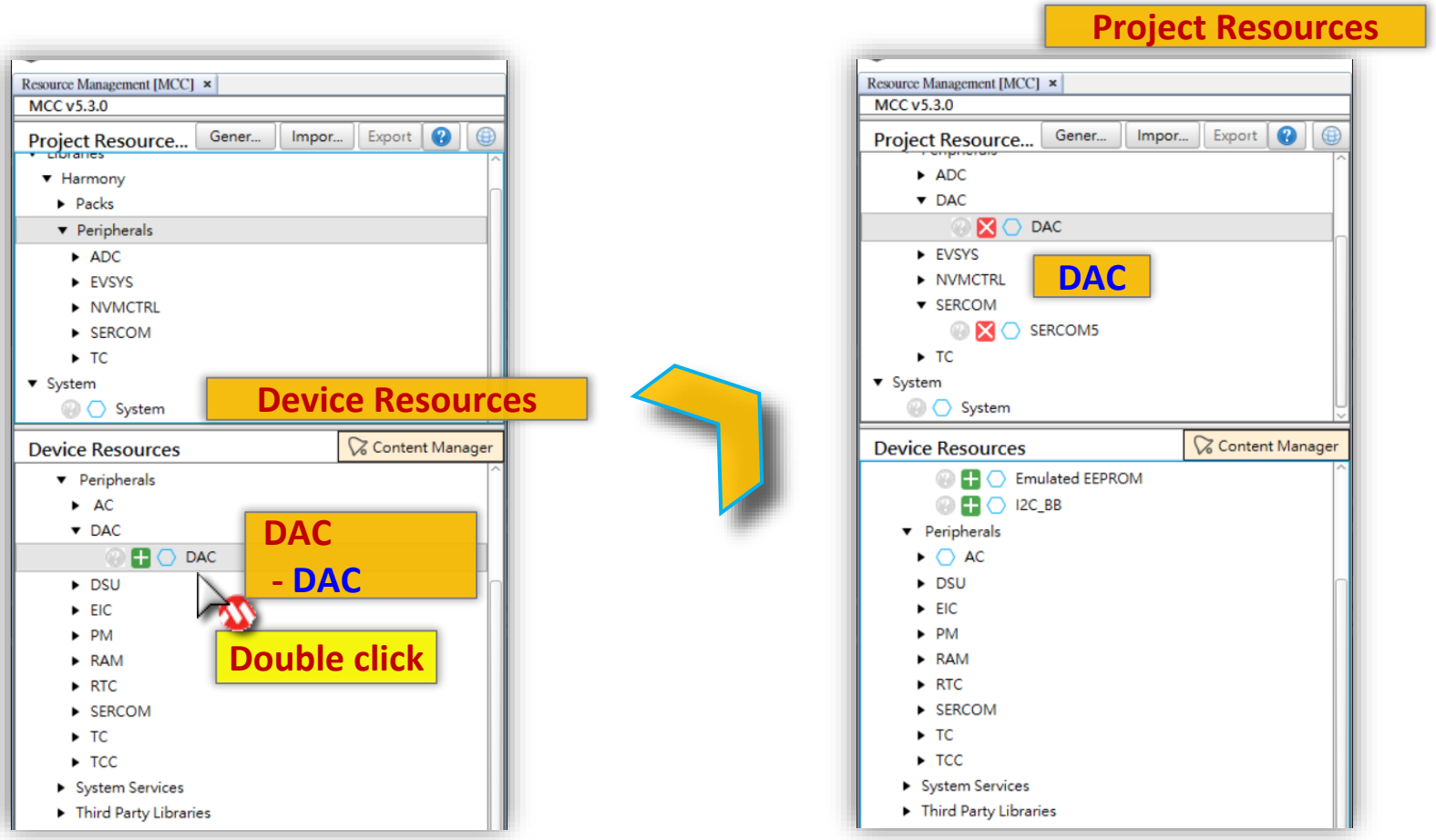
Value	Name	Description
0x0	INT1V	Internal voltage reference
0x1	VDDANA	Analog voltage supply
0x2	VREFA	External reference
0x3		Reserved

VREFA and VREFB range 1.0V ~ (V_{DDANA} - 0.6V)

MICROCHIP

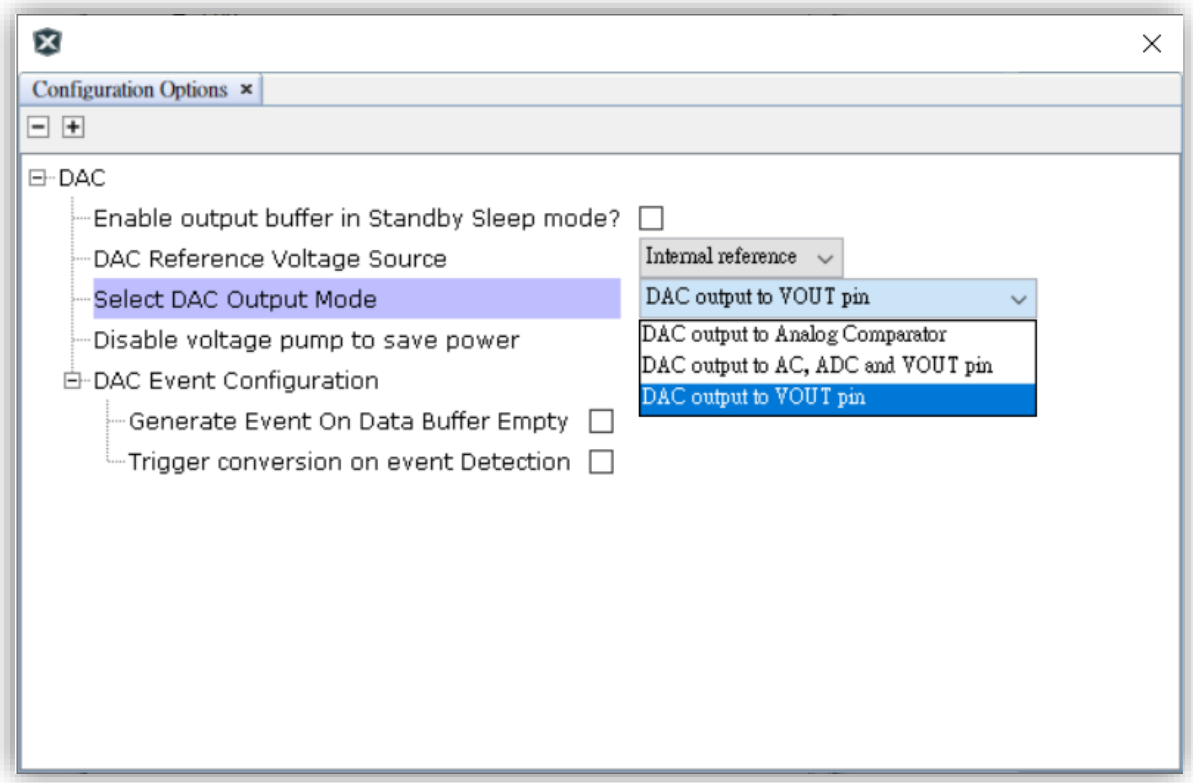
Add DAC Function using MCC-Harmony

- Find DAC component in “Device Resources” window.
- Double click DAC to add to “Project Resources” window.



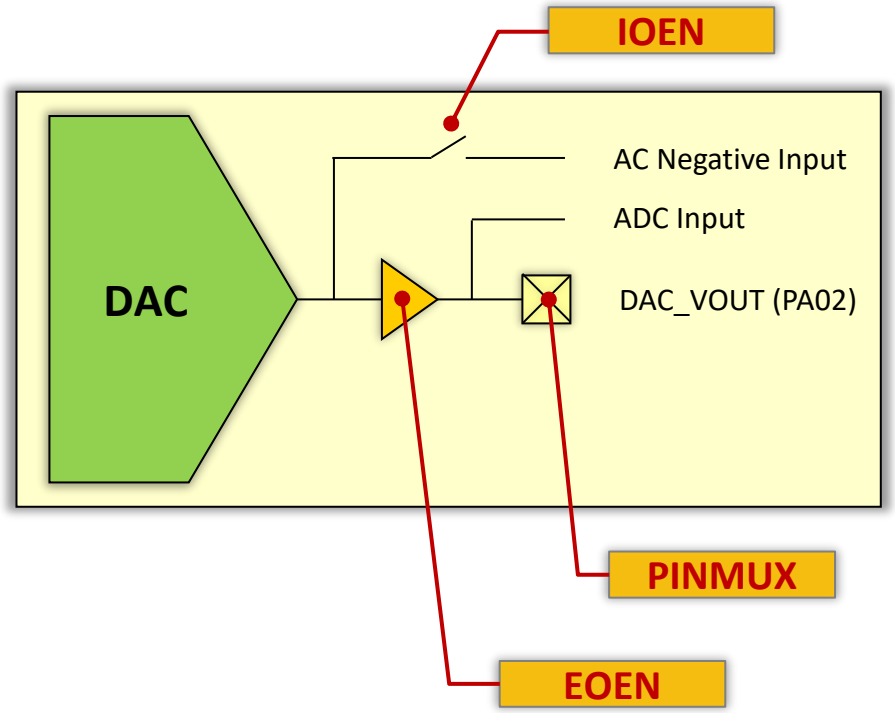
DAC Output

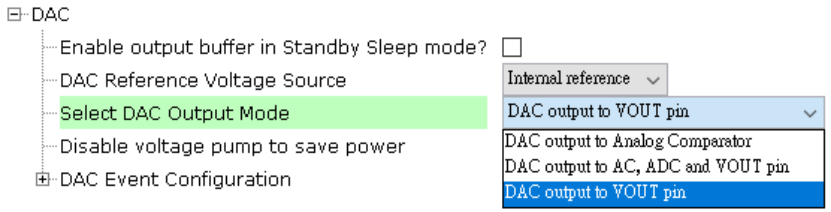
Configuration Options Example



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

MICROCHIP

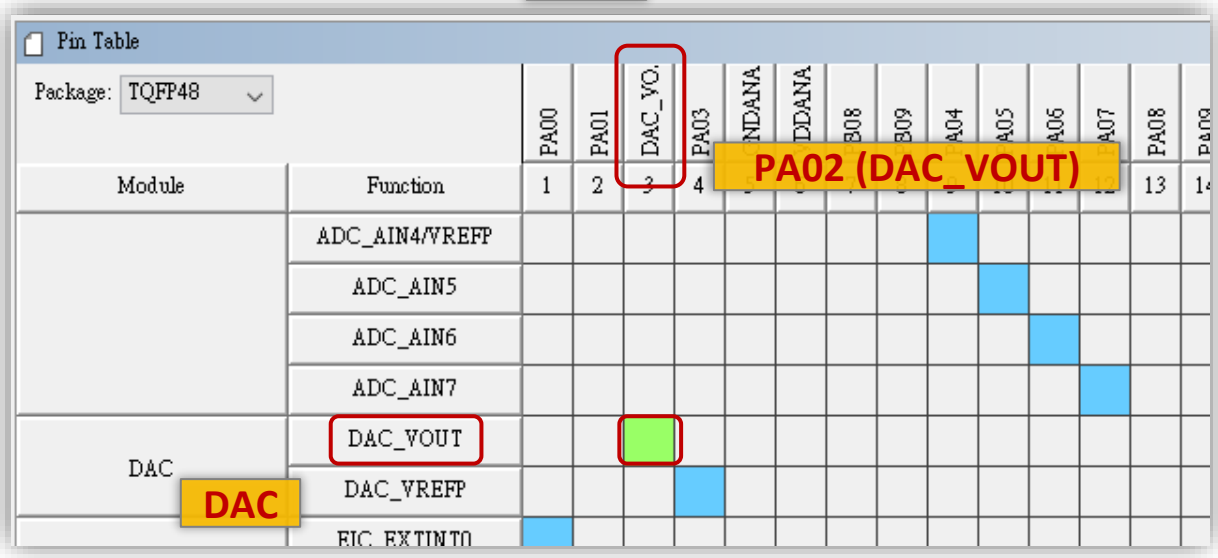
DAC Output

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/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/		

PA02 VOUT



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;
      }
   }
}
```

MICROCHIP

🛠 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

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

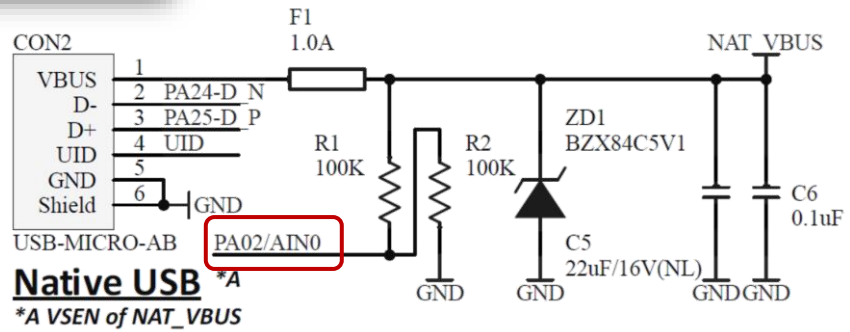
MICROCHIP

🛠 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.**

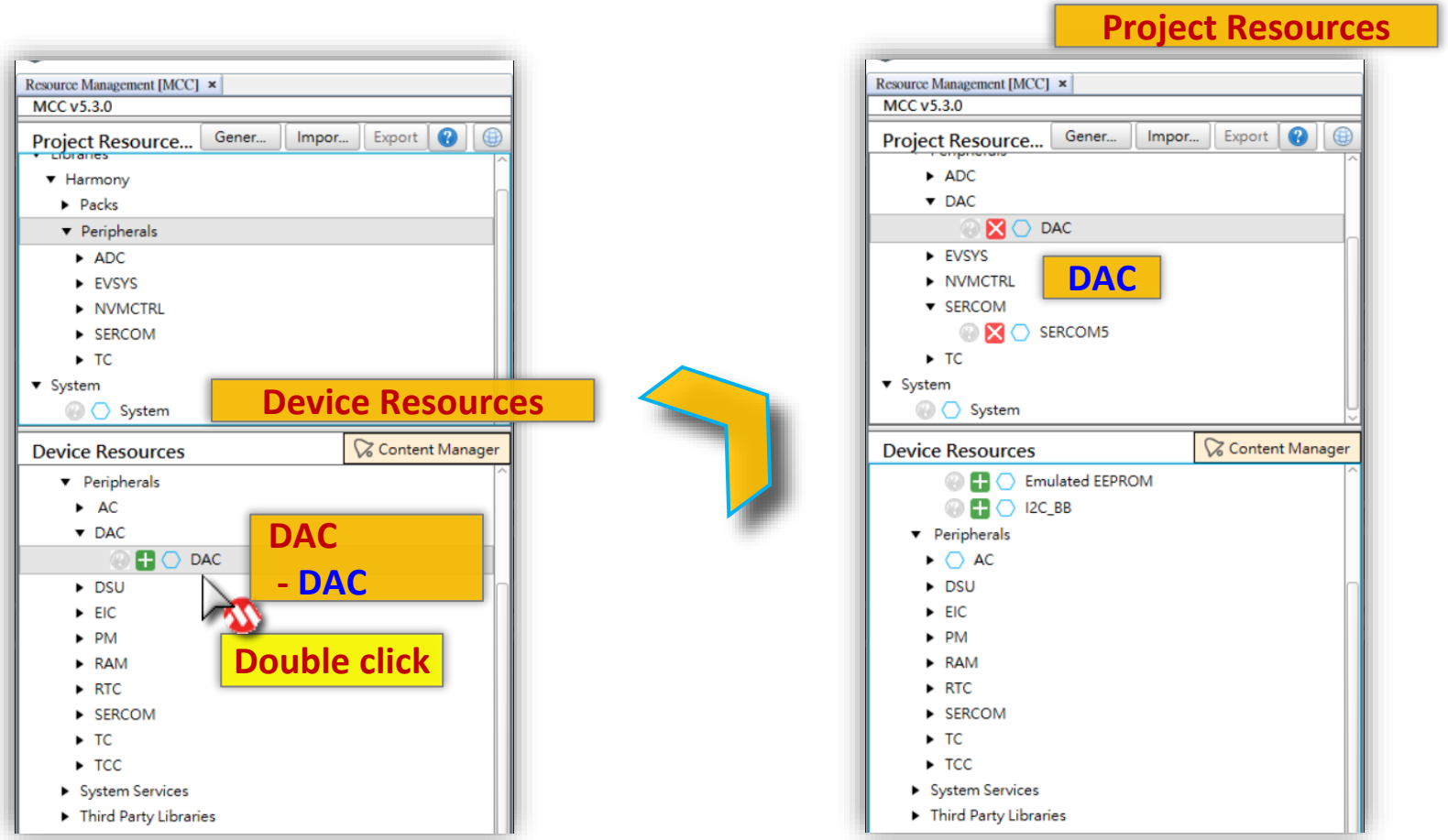
Pin(1)			I/O Pin	Supply	A	B(2)(3)					C	D	E	F	G	H
SAMD2xE	SAMD2xG	SAMD2xJ			EIC	REF	ADC	AC	PTC	DAC	SERCOM(2)(3)	SERCOM-ALT	TC(4) /TCC	TCC	COM	AC/ GCLK
1	1	1	PA00	VDDANA	EXTINT[0]							SERCOM1/ PAD[0]	TCC2/WO[0]			
2	2	2	PA01	VDDANA	EXTINT[1]							SERCOM1/ PAD[1]	TCC2/WO[1]			
3	3	3	PA02	VDDANA	EXTINT[2]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



🛠 Lab12-2 DAC Output to ADC Measurement

Step 1

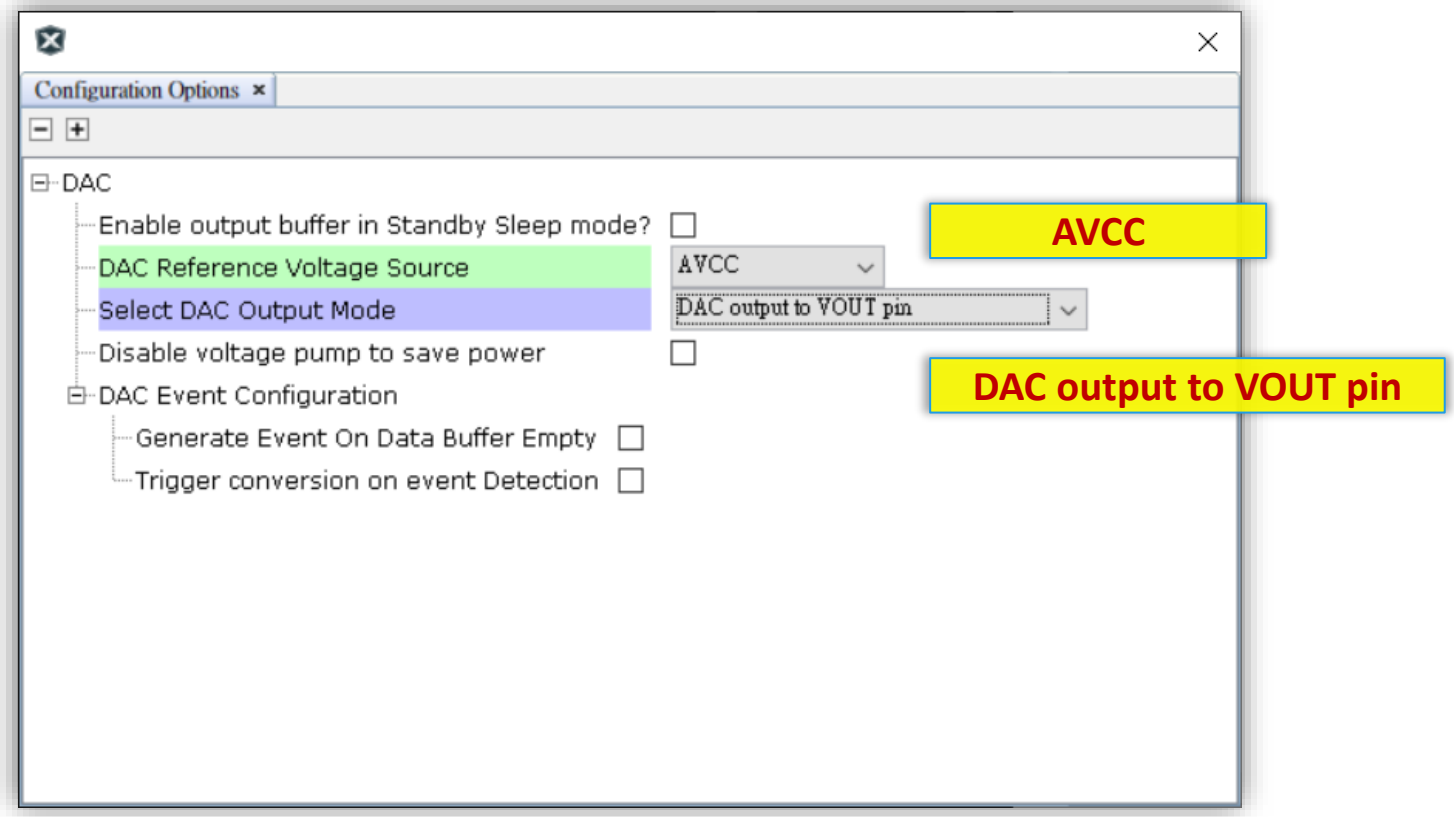
- Find DAC component in “Device Resources” window.
- Double click DAC to add to “Project Resources” window.



🛠 Lab12-2 DAC Output to ADC Measurement

Step 2

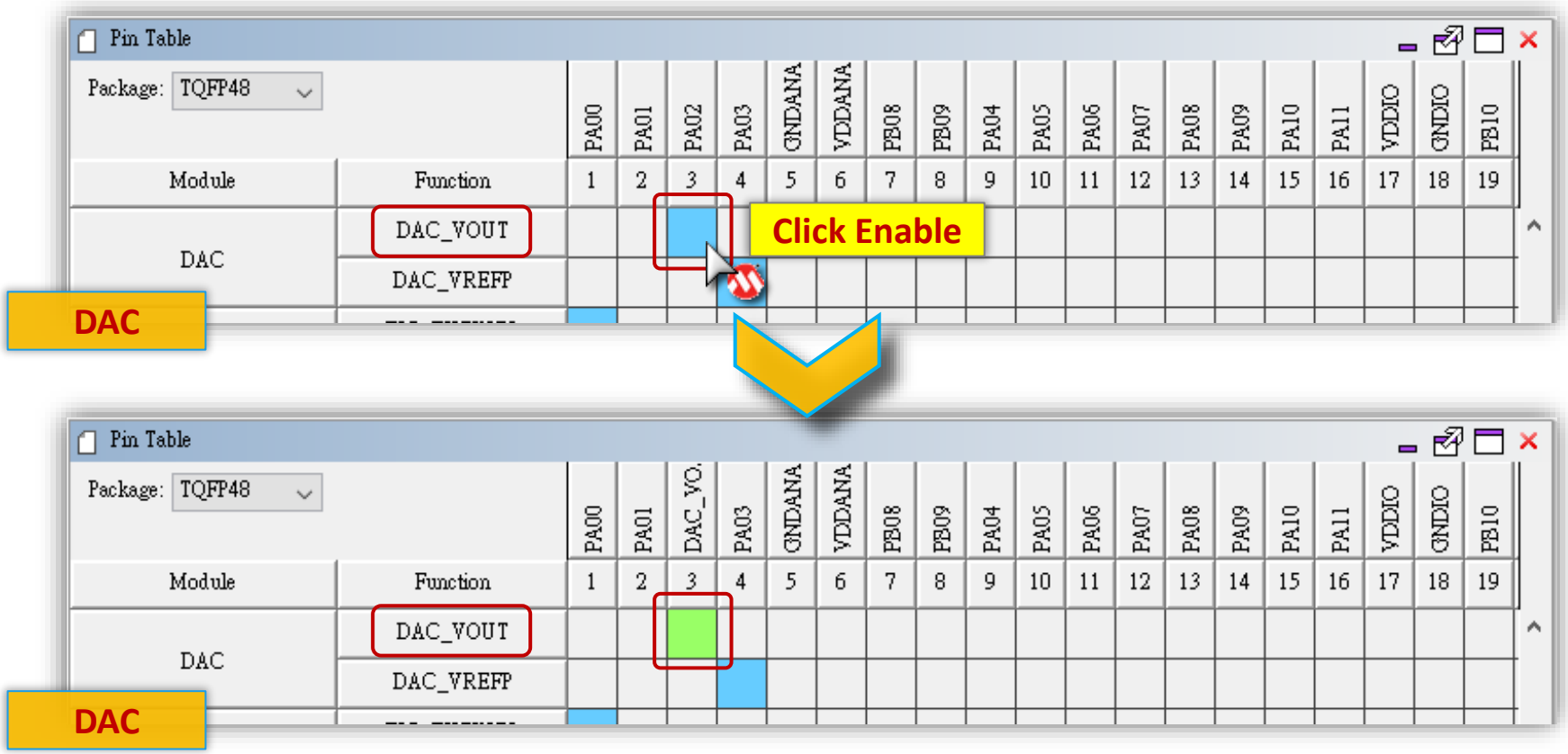
- **Configure DAC in configuration.**



🛠 Lab12-2 DAC Output to ADC Measurement

Step 3

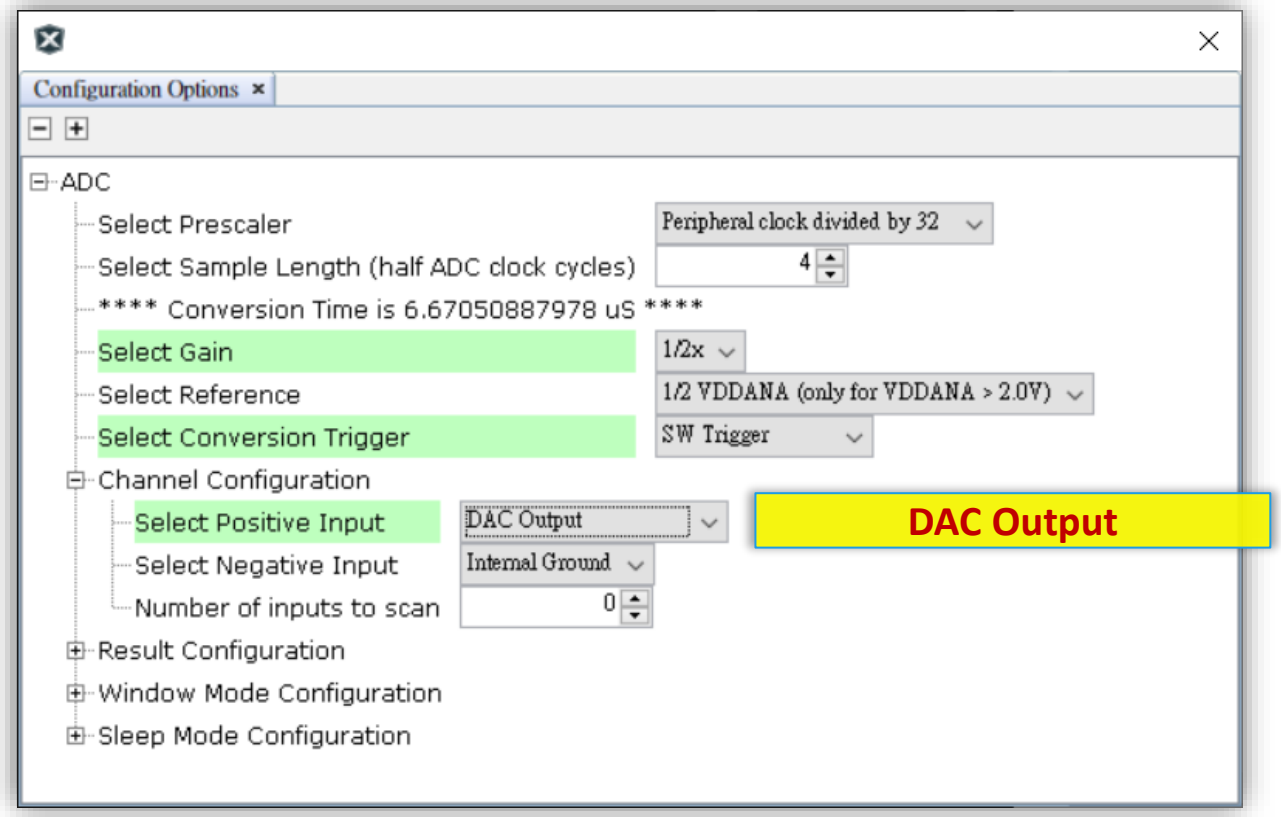
- **Enable DAC VOUT (DAC_VOUT, PA02)**



🛠 Lab12-2 DAC Output to ADC Measurement

Step 4

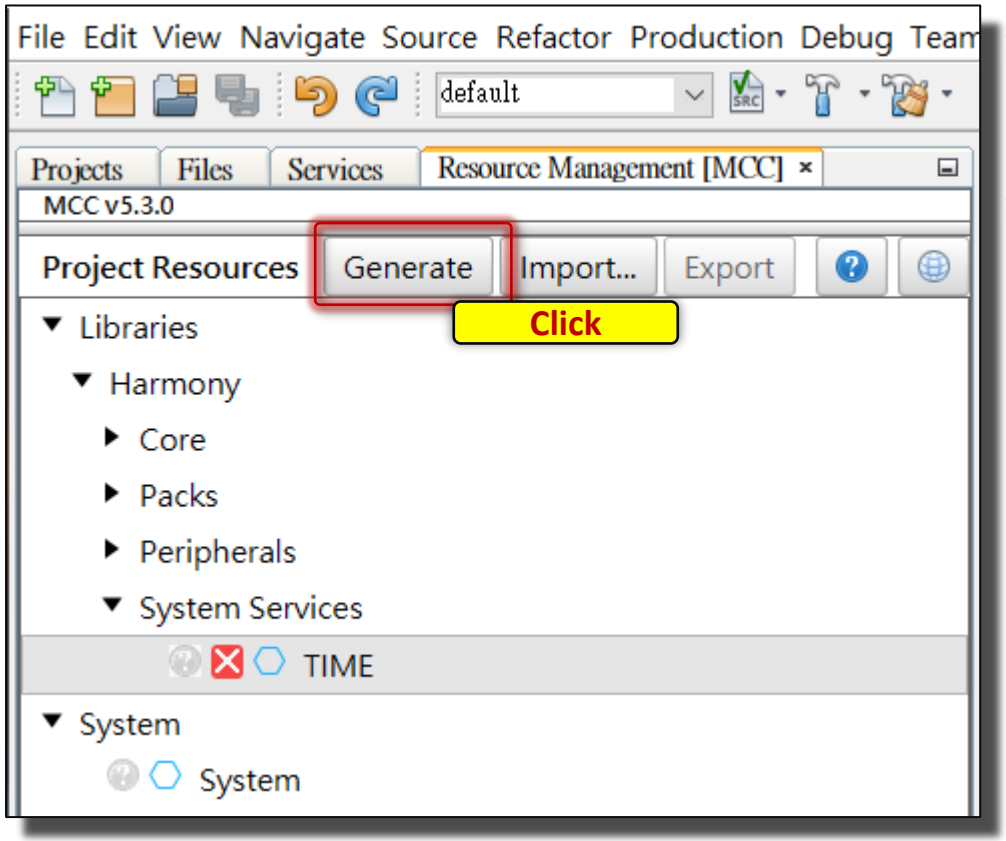
- Change ADC positive channel to “DAC Output” in ADC configuration.

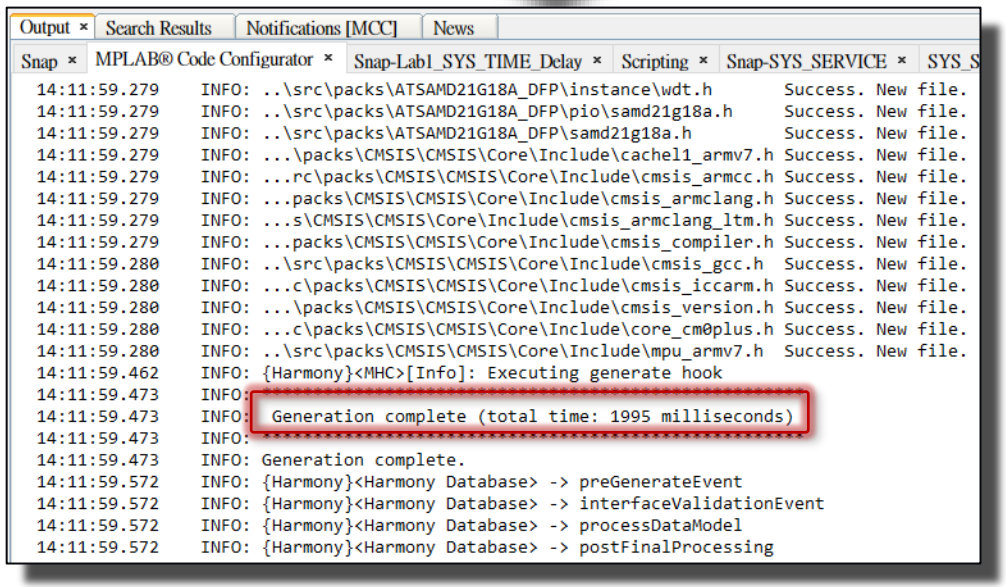


🛠 Lab12-2 DAC Output to ADC Measurement

Step 5

- **Click to Generate Code.**





🛠 Lab12-2 DAC Output to ADC Measurement

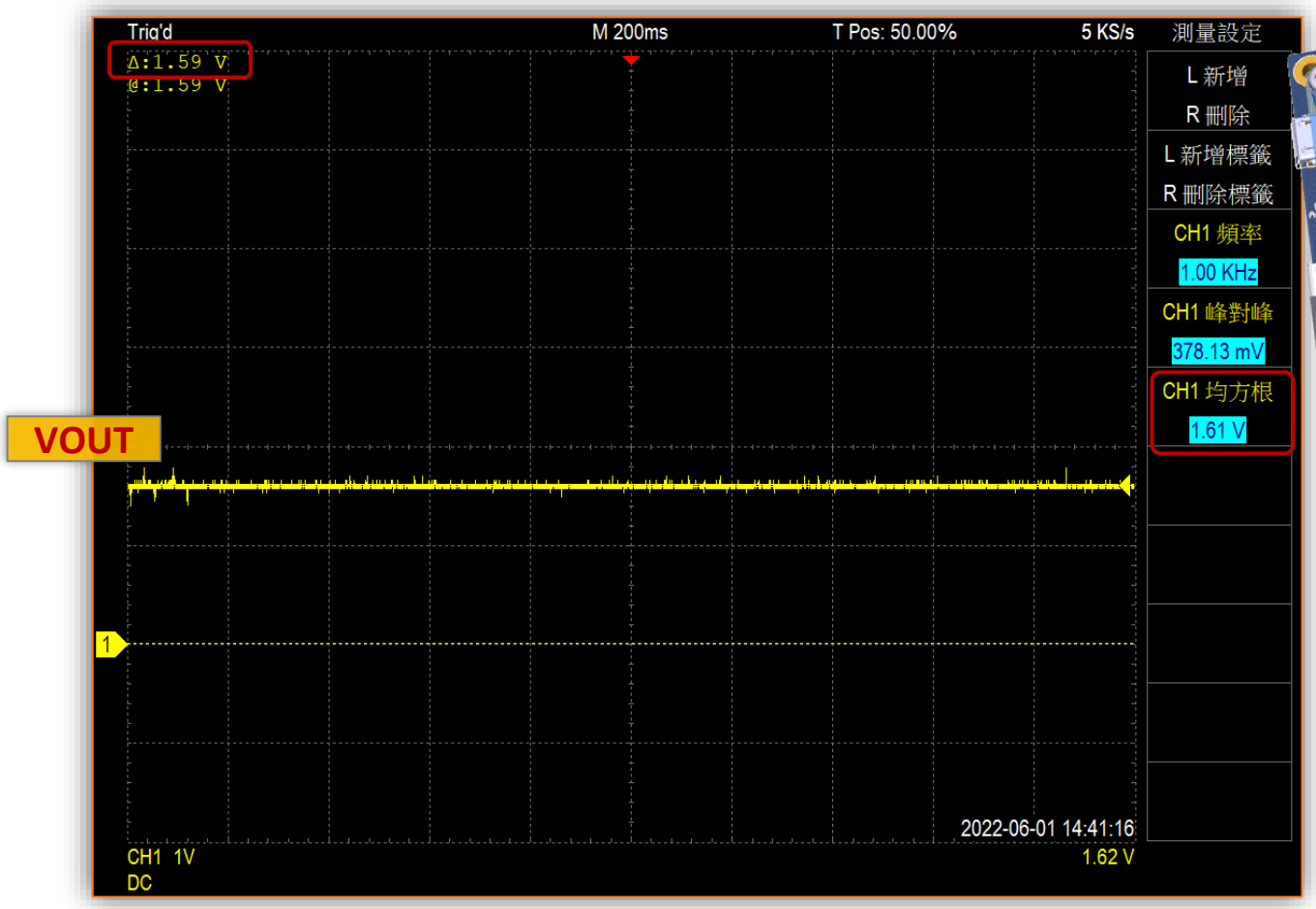
Step 6

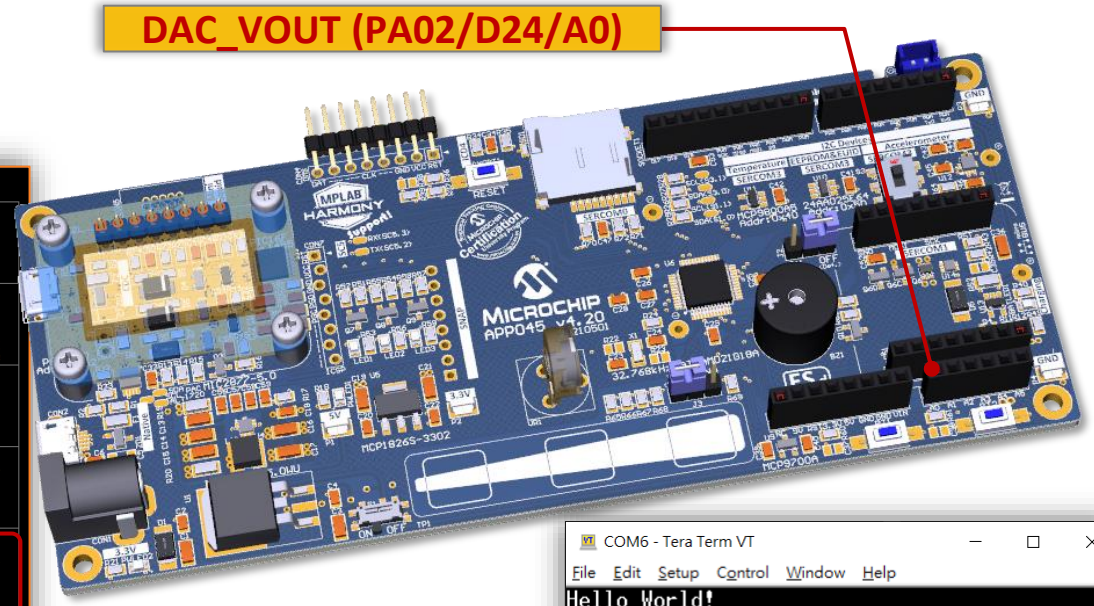
```
int main (void)
{

    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);
}
```

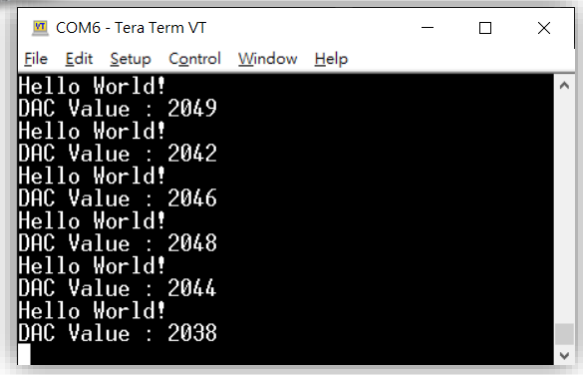
🛠 Lab12-2 DAC Output to ADC Measurement

Result

- **Measure the waveform at DAC_VOUT.**







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

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

MICROCHIP

🛠 Lab12-2 DAC Output to ADC Measurement

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

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

```
#include <math.h>

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

Use TCx to control output period.

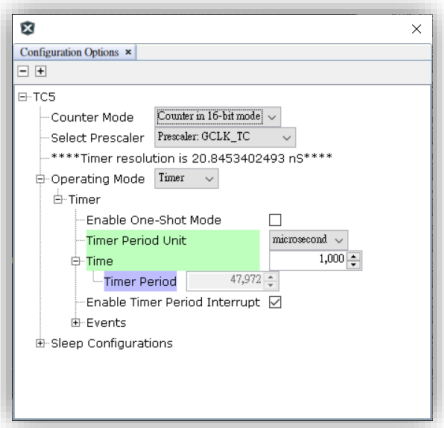
```
if (TCx_HasExpired) /* Period Control */
{
    TCx_HasExpired = 0;
    DAC_DataWrite(Wave_Array[Index]);
    if (++DACIndex >= DAC_Array_Size)
        DACIndex = 0;
}
```

  - 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
```

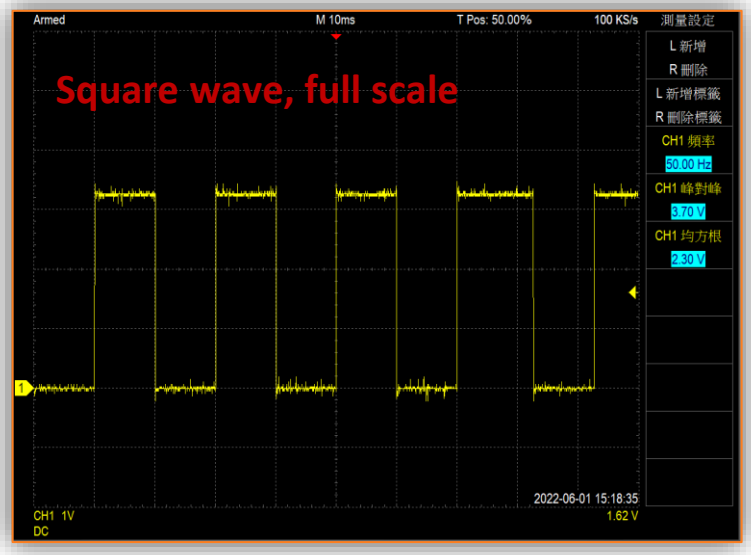


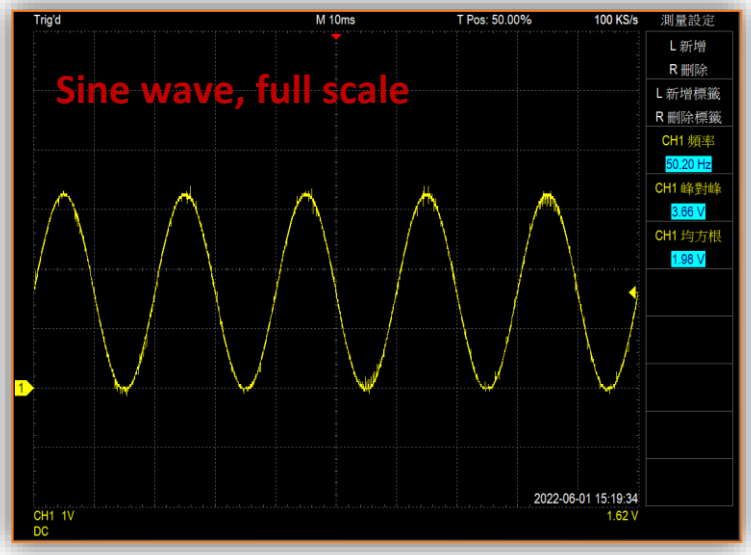
- **Let’s go !**

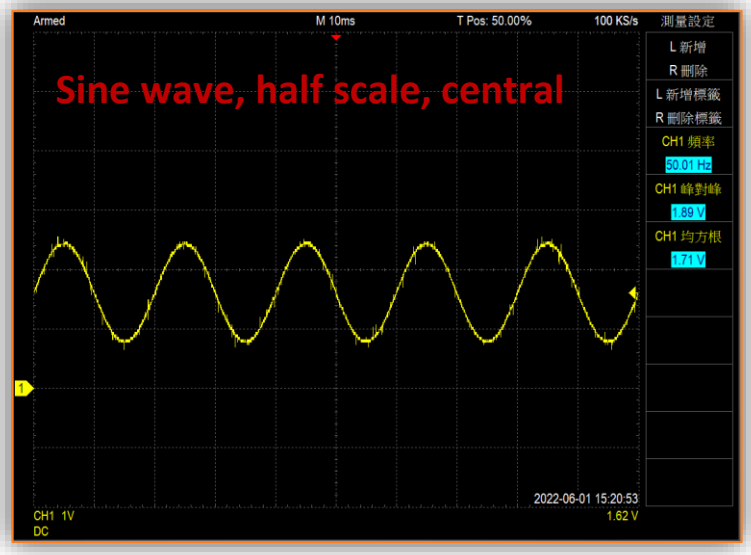
MICROCHIP

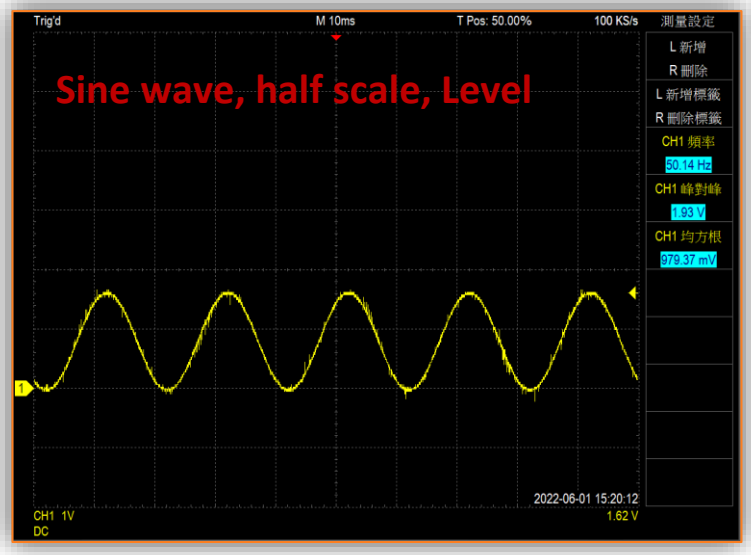
🛠 Lab12-3 DAC Wave Output

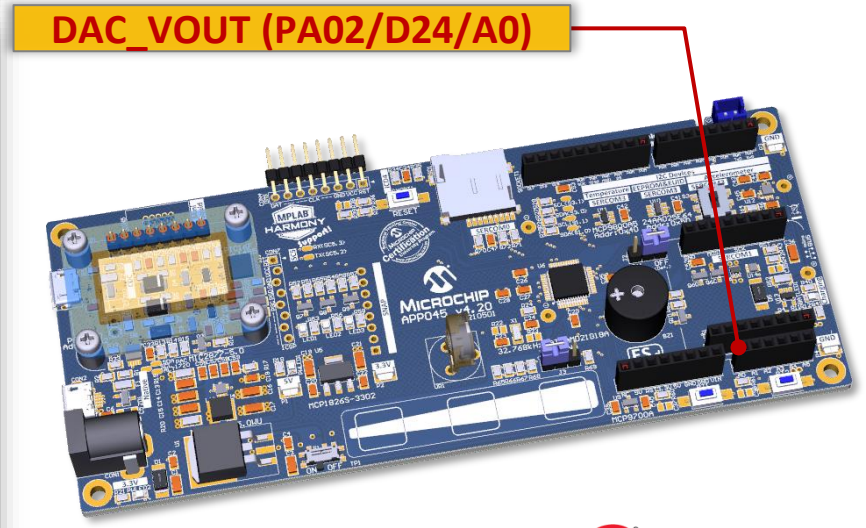
Result







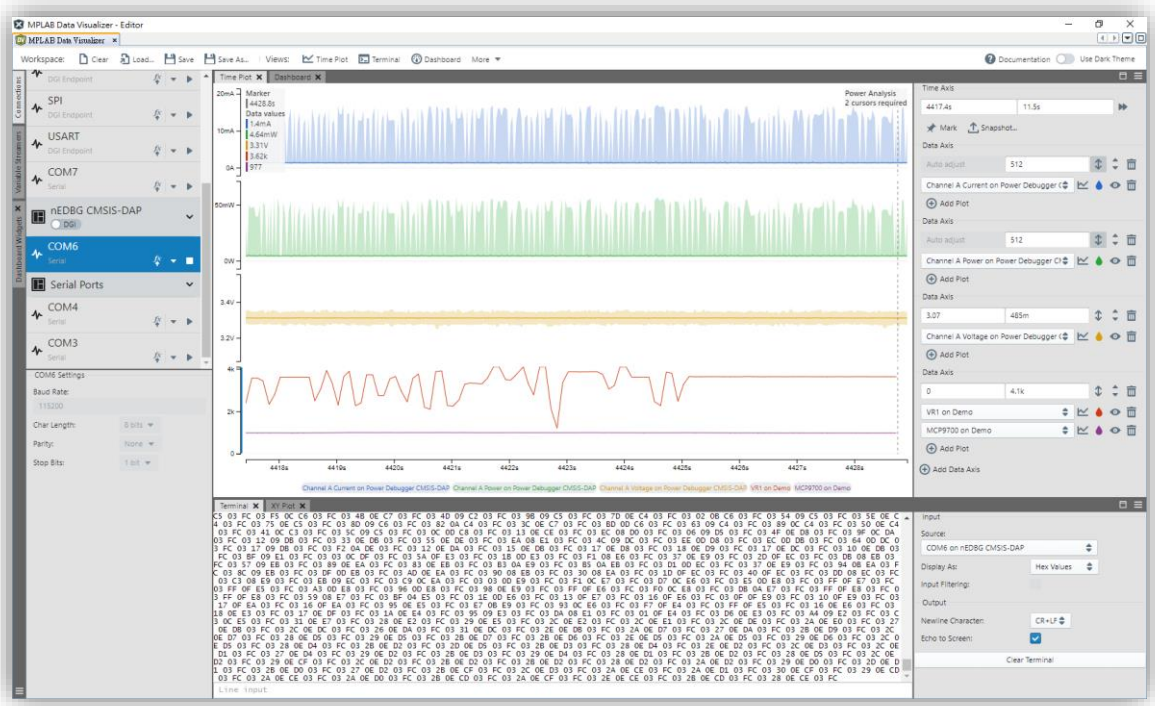




MPLAB® Data Visualizer

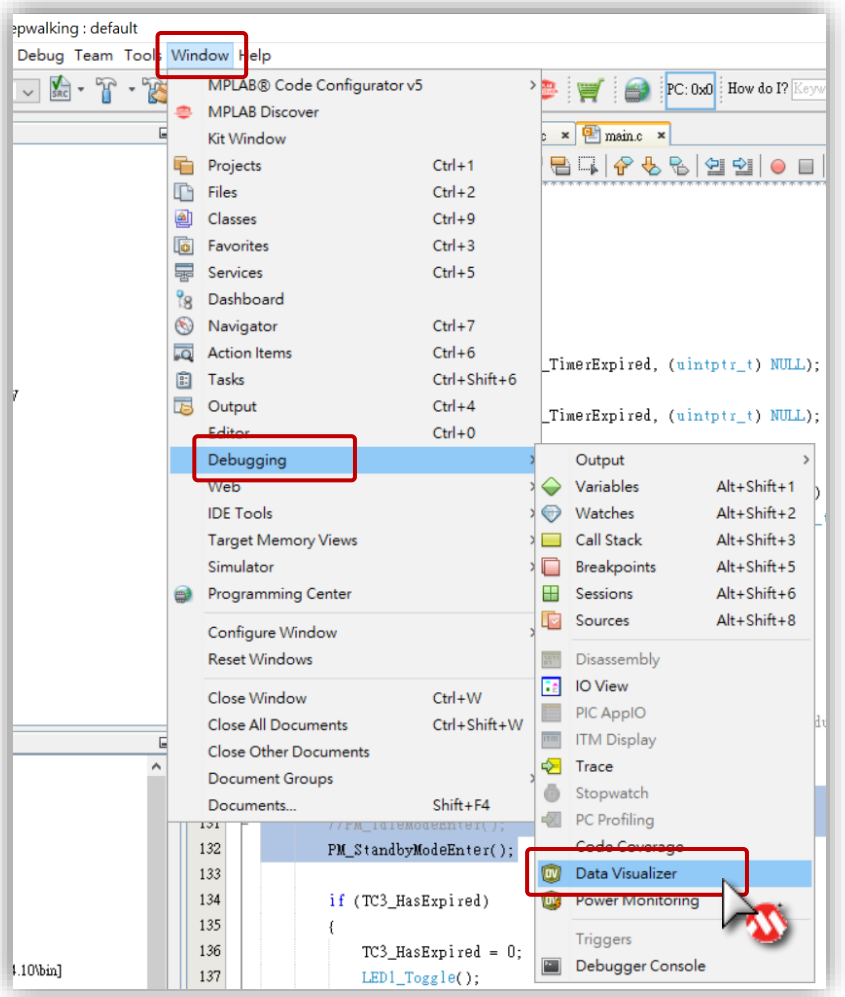
MPLAB Data Visualizer

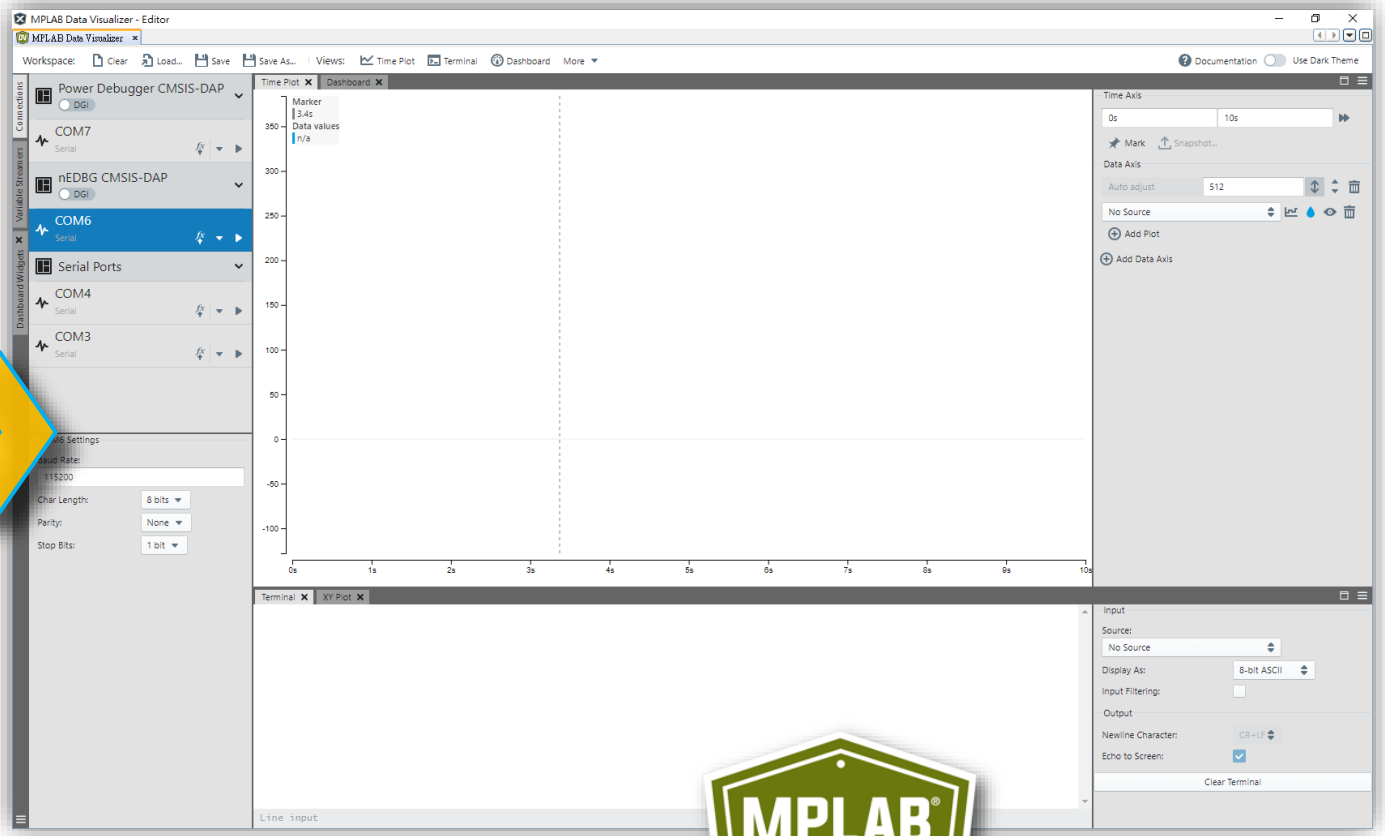
- MPLAB Data Visualizer is a program used for processing and visualizing data.
- It can receive data from various sources such as the DGI (Data gateway Interface) and COM ports.
- And has a graph plotter and an oscilloscope, a terminal and a configurable dashboard with buttons, sliders, and various indicators. 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



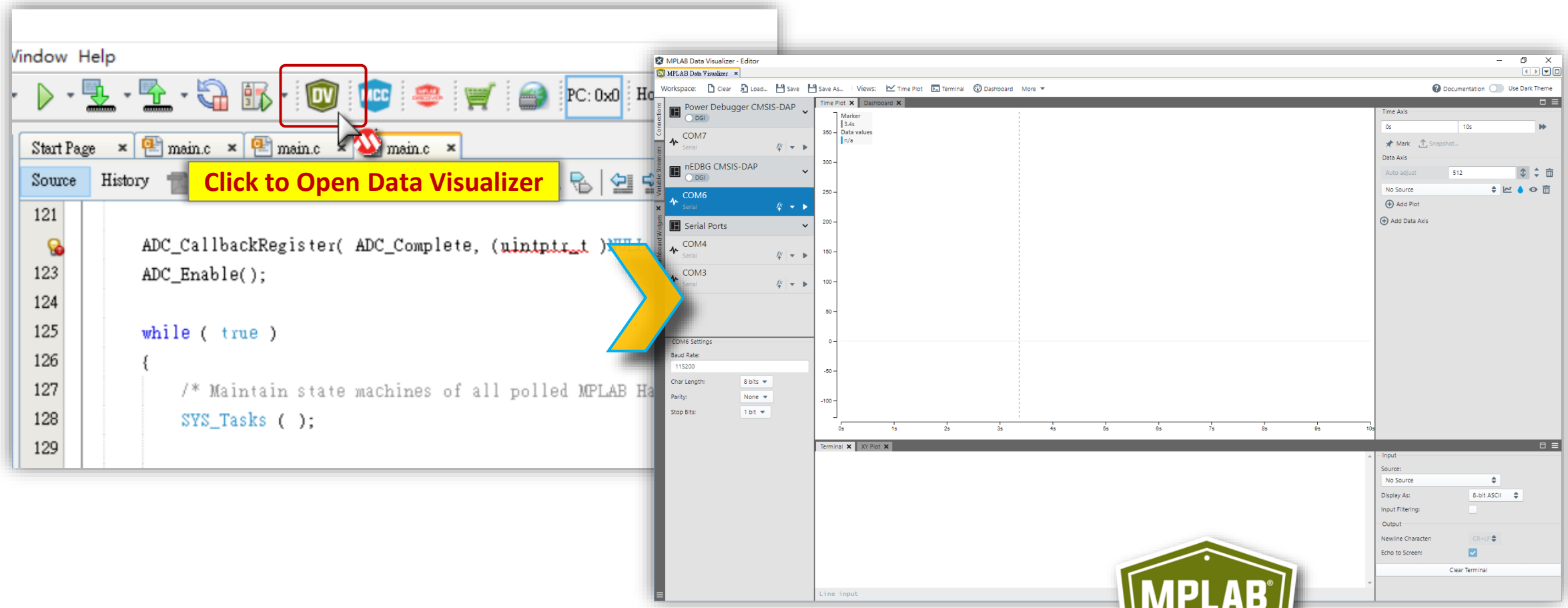


MPLAB® DATA VISUALIZER

MICROCHIP

MPLAB Data Visualizer

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

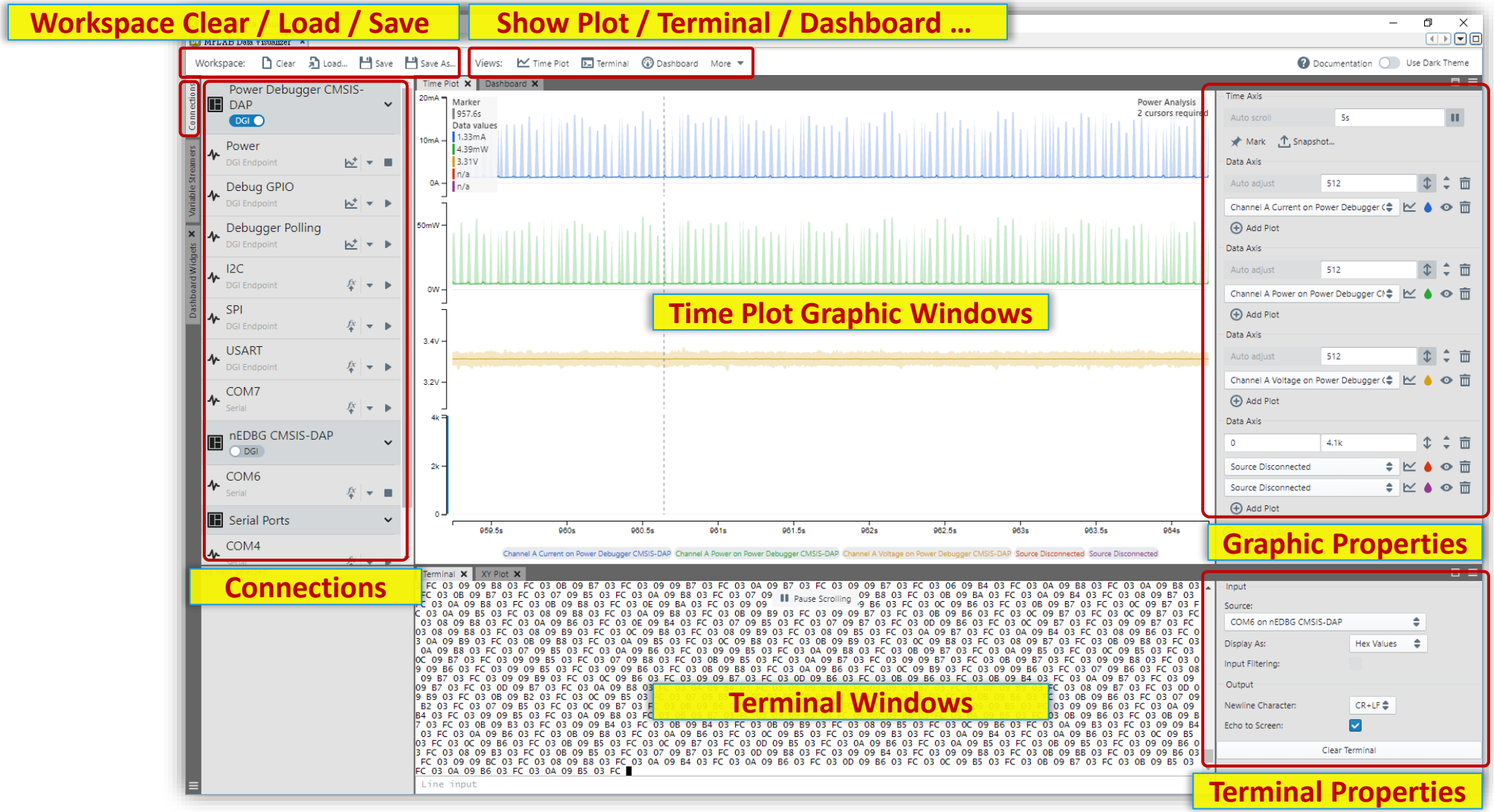


MPLAB® DATA VISUALIZER

MICROCHIP

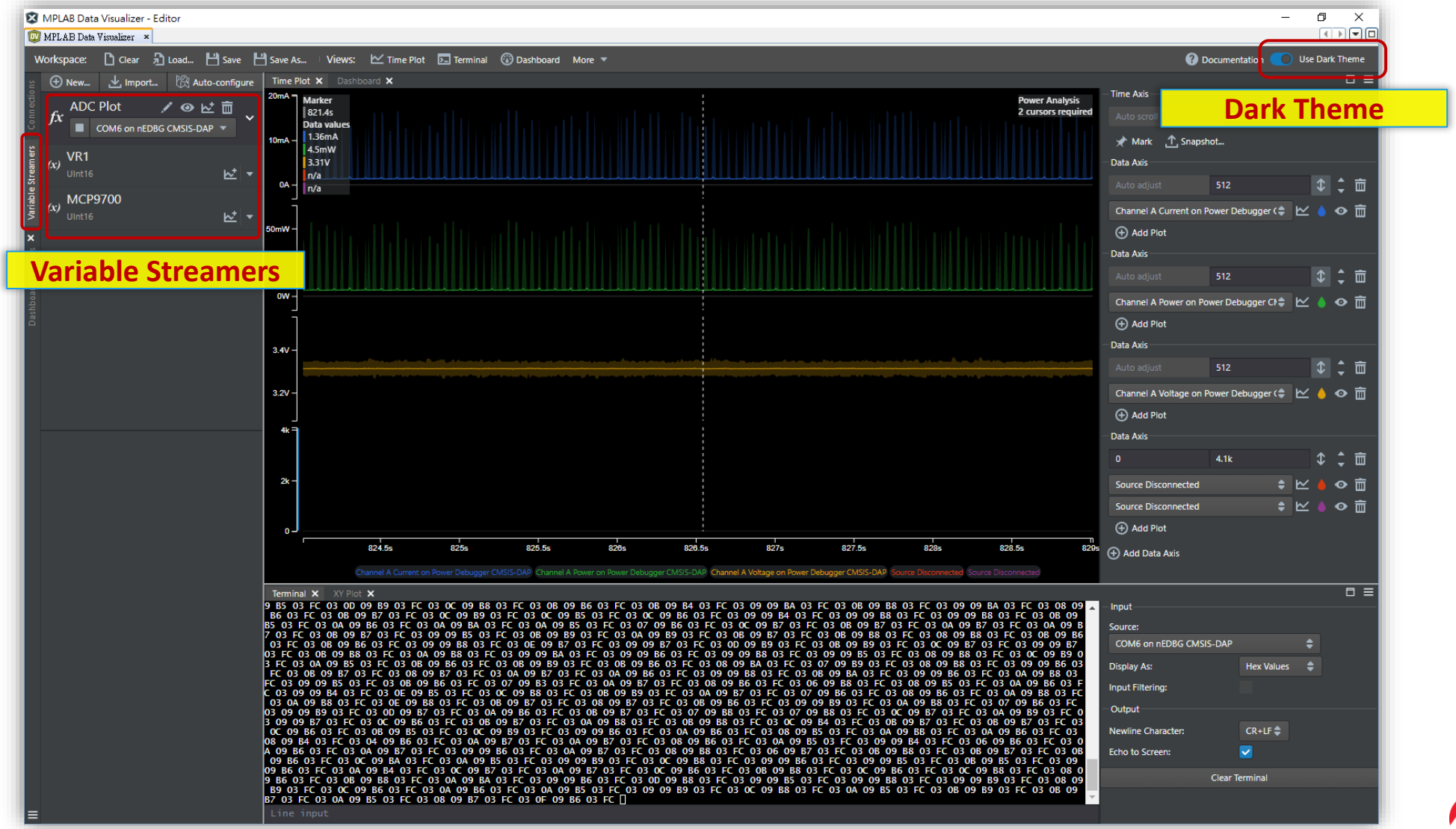
MPLAB Data Visualizer

Workspace of MPLAB Data Visualizer



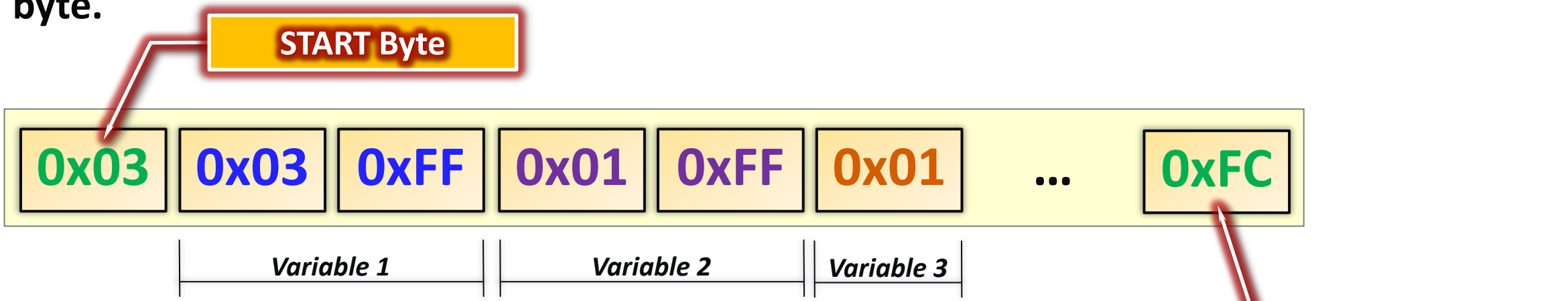
MPLAB Data Visualizer

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.

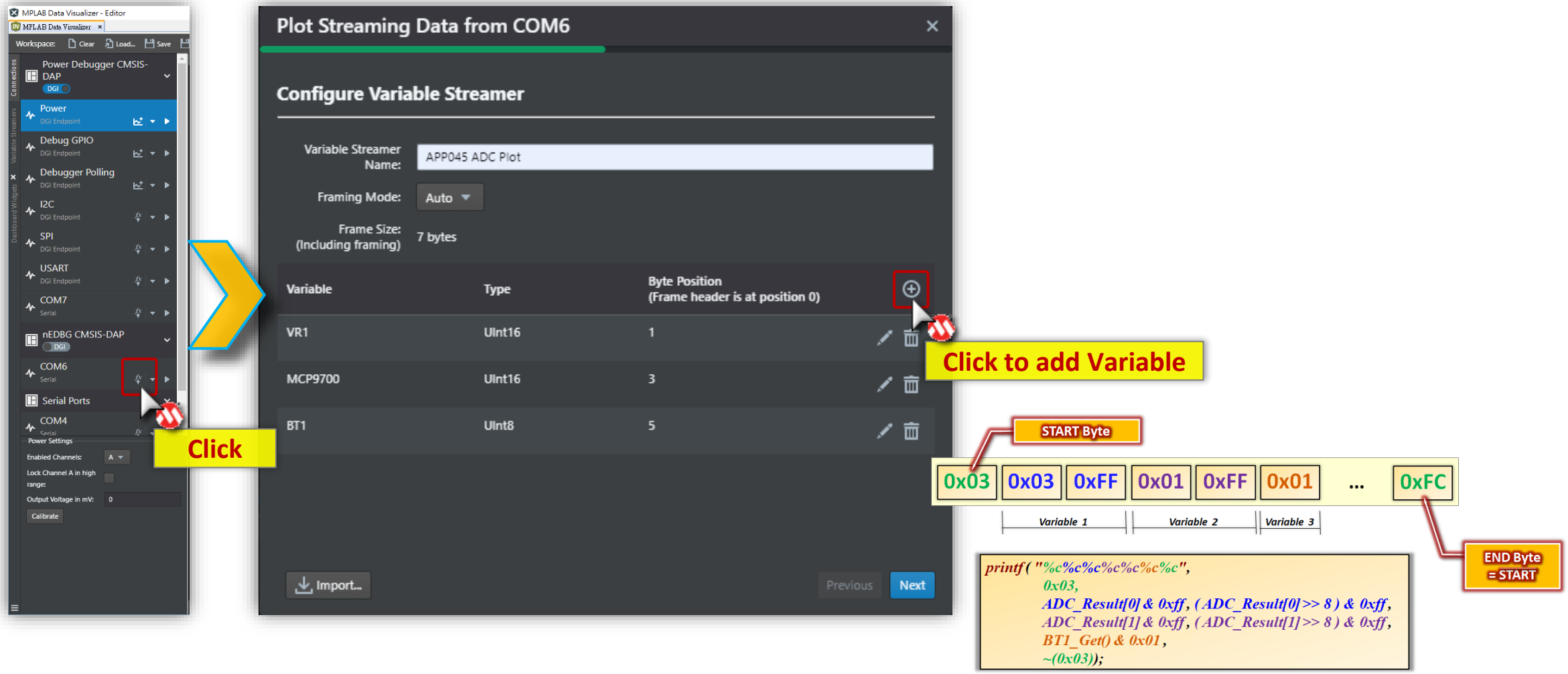



```
printf ( "%c%c%c%c%c%c%c",
         0x03,
         ADC_Result[0] & 0xff , ( ADC_Result[0] >> 8 ) & 0xff ,
         ADC_Result[1] & 0xff , ( ADC_Result[1] >> 8 ) & 0xff ,
         BT1_Get( ) & 0x01 ,
         ~(0x03));
```

MICROCHIP

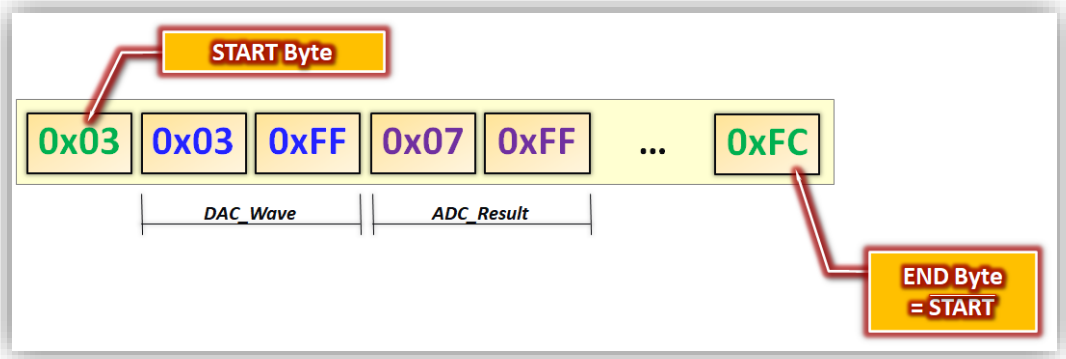
Data Streamer Setting

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



🛠 Lab12-4 DAC Wave Output DataVisualizer

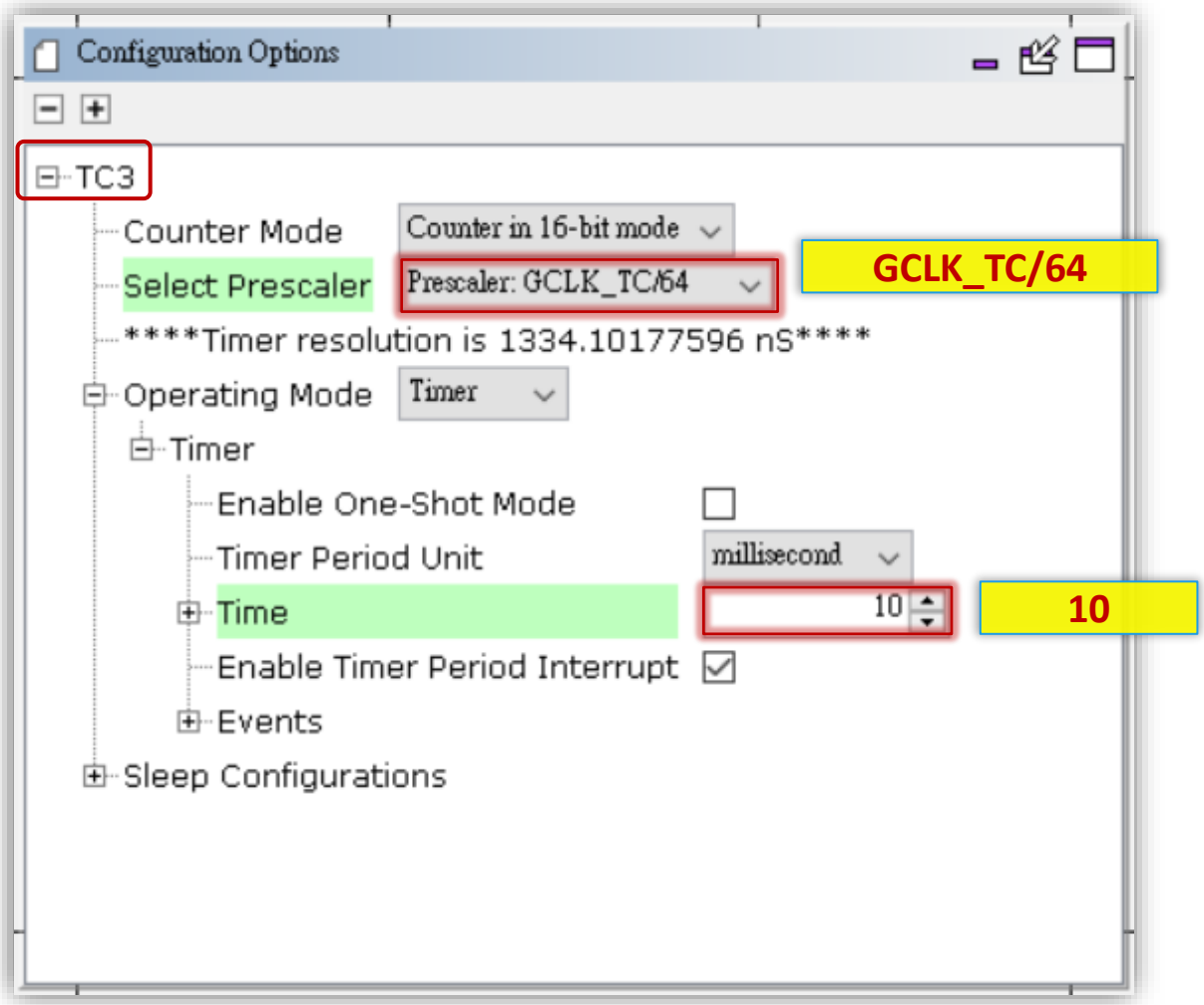
- Please continue from SAM2001 Lab12-3 (DAC Wave Output)
- Try modify UART output to observe
  DAC idea output and ADC real result for DAC on MPLAB Data Visualizer.
- Try plot the two waveform on MPLAB Data Visualizer.



- **Let’s go !**

🛠 Lab12-4 DAC Wave Output DataVisualizer

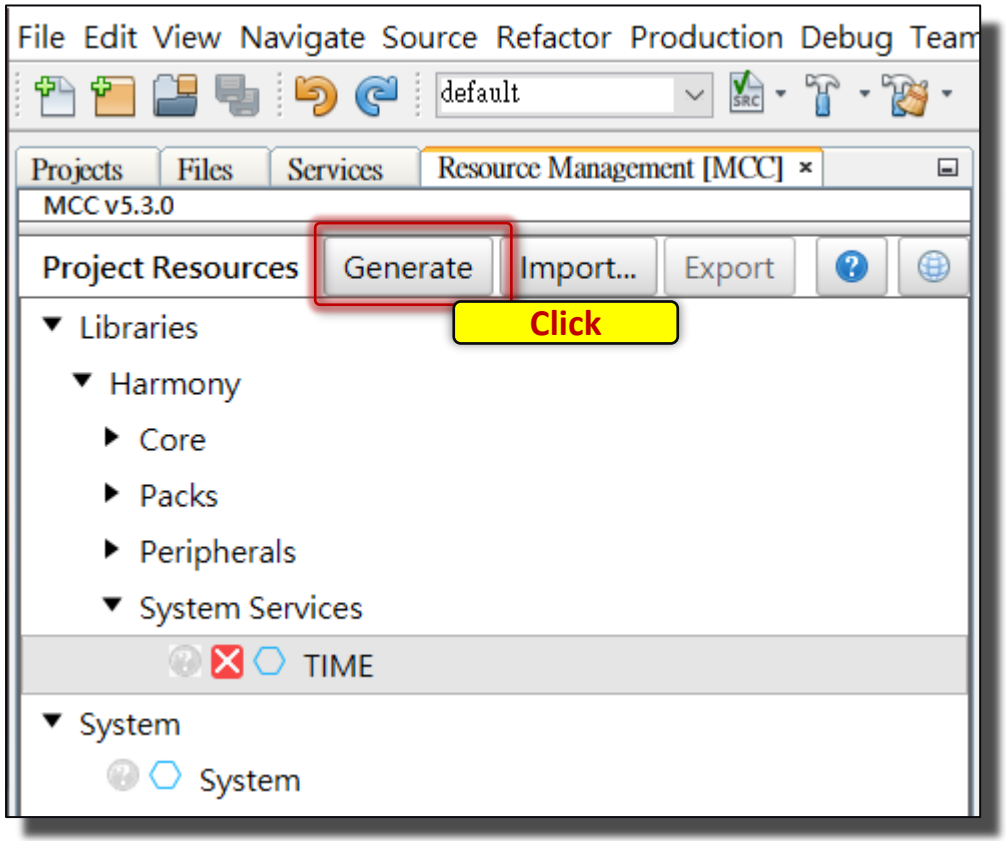
Step 1

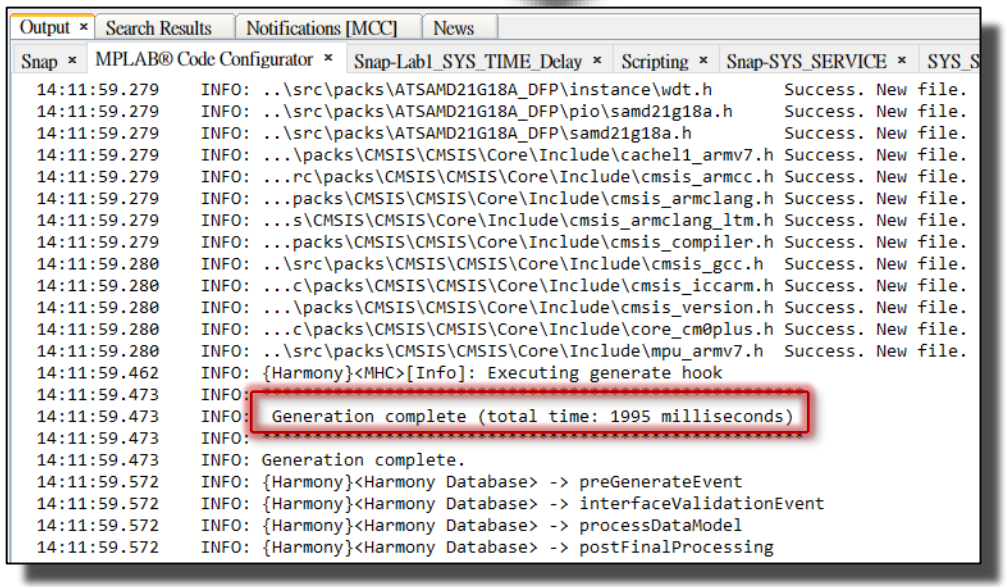


🛠 Lab12-4 DAC Wave Output DataVisualizer

Step 2

- **Click to Generate Code.**





🛠 Lab12-4 DAC Wave Output DataVisualizer

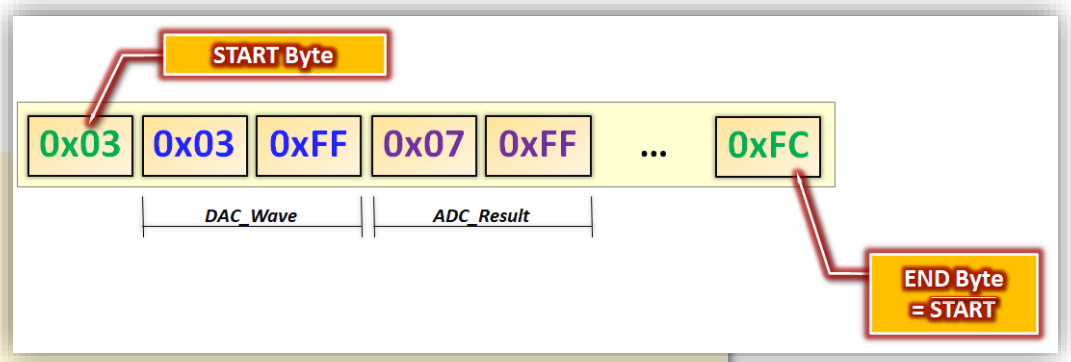
Step 3

```
...
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);
}
```

MICROCHIP

🛠 Lab12-4 DAC Wave Output DataVisualizer

Step 4



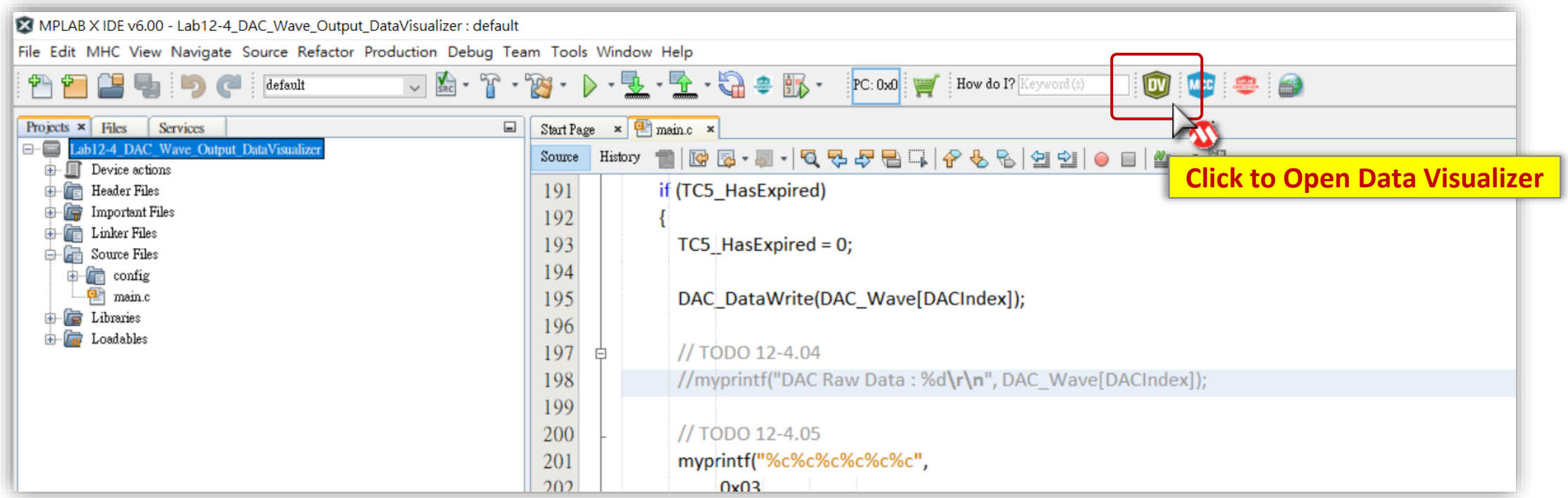

```
...
int main (void)
{
    SYS_Initialize(NULL);
    ...
    while ( true )
    {
        if (TC5_HasExpired)
        {
            ...
            // TODO 12-4.04
            //myprintf("DAC Raw Data : %d\r\n", DAC_Wave[DACIndex]);
            // TODO 12-4.05
            myprintf("%c%c%c%c%c%c",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

MICROCHIP

🛠 Lab12-4 DAC Wave Output DataVisualizer

Step 5

- **Open MPLAB Data Visualizer by icon at MPLAB X IDE**

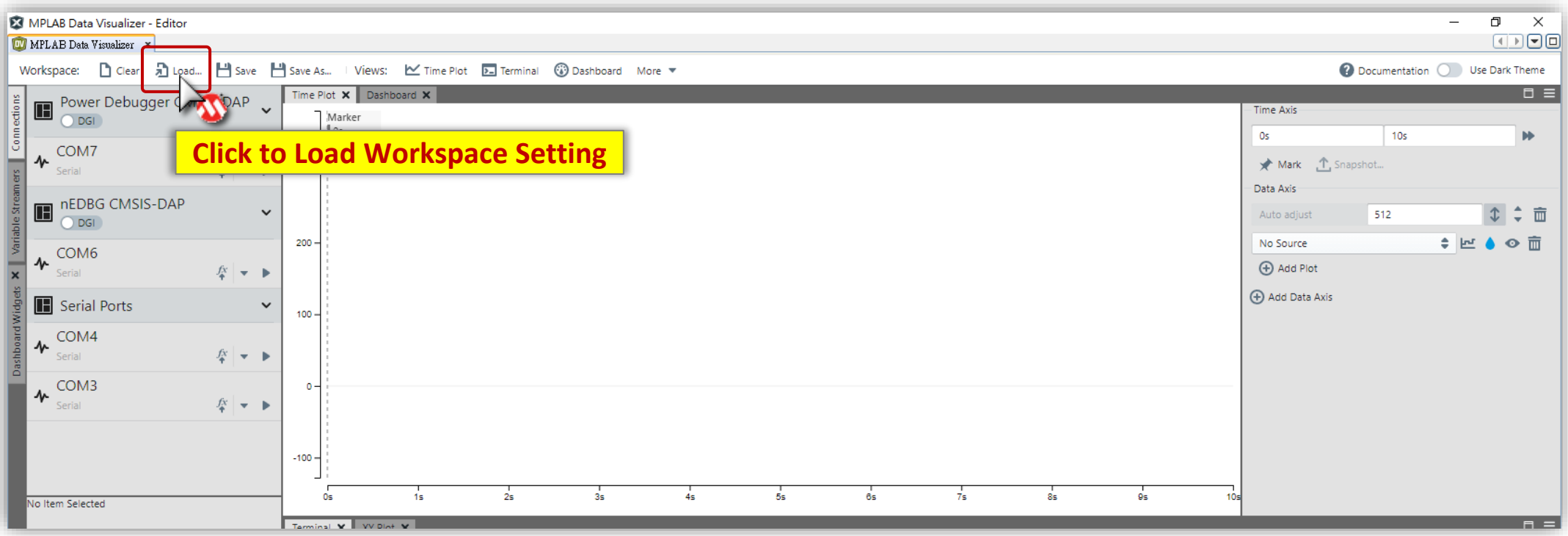


MICROCHIP

🛠 Lab12-4 DAC Wave Output DataVisualizer

Step 6

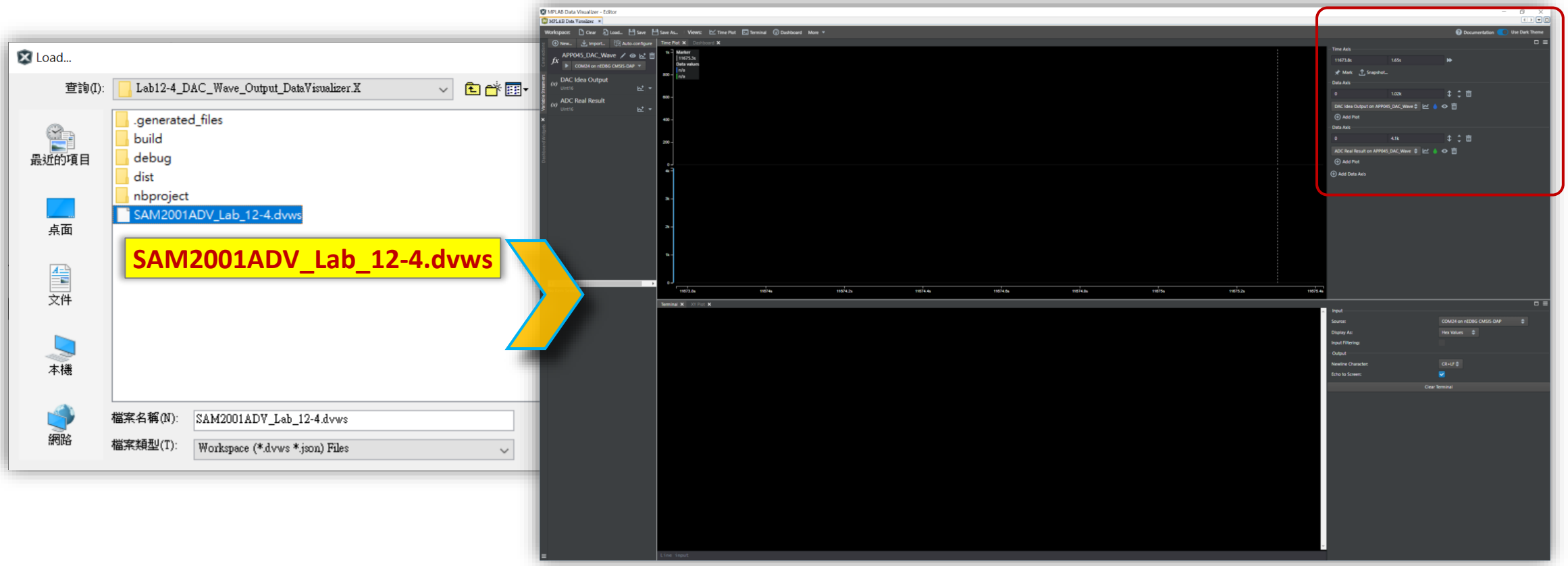
- **Just load the workspace file to reduce setting steps.**



🛠 Lab12-4 DAC Wave Output DataVisualizer

Step 7

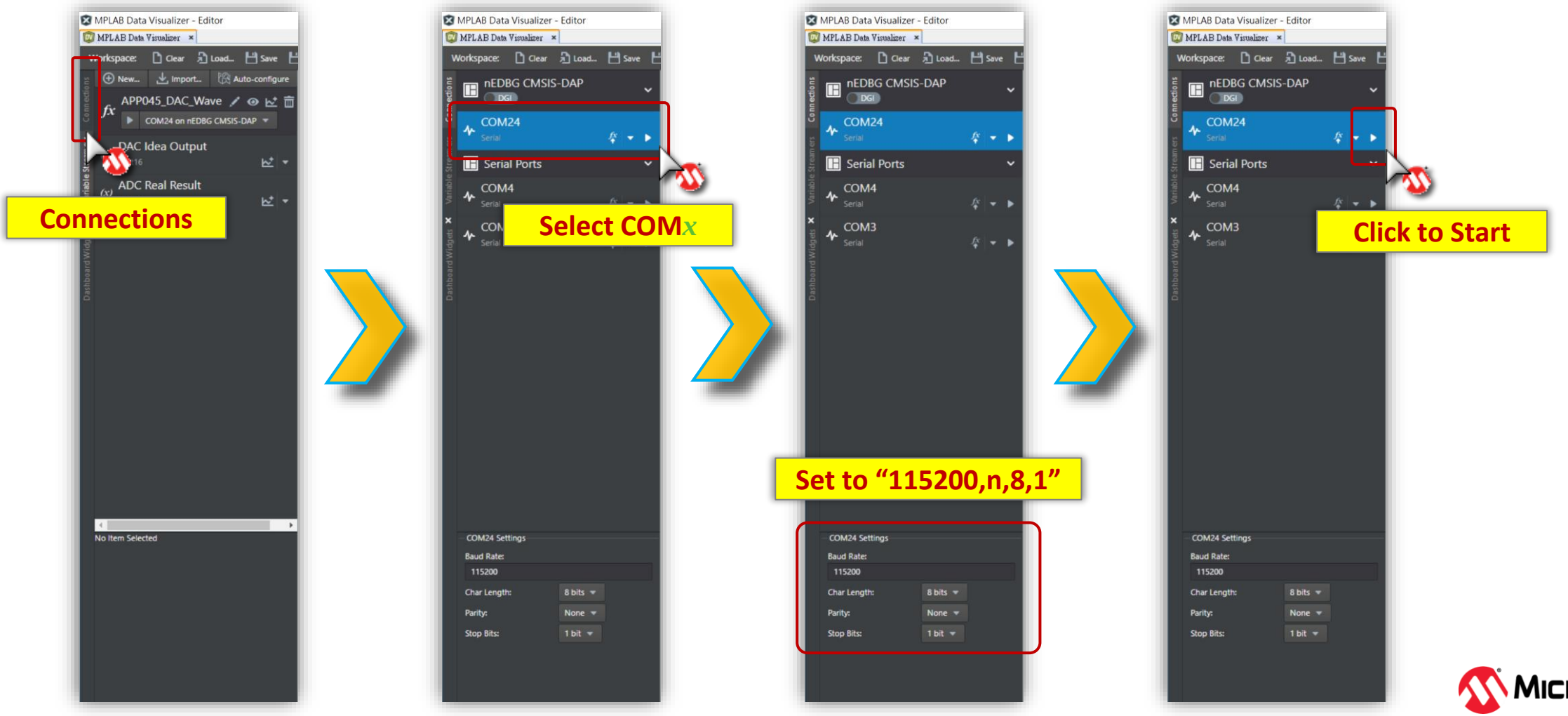
- **Select “SAM2001ADV_Lab_12-4.dvws” workspace file.**
  There are 2 data axis be set. That include “DAC Idea Output” and “ADC Real Result”.



🛠 Lab12-4 DAC Wave Output DataVisualizer

Step 8

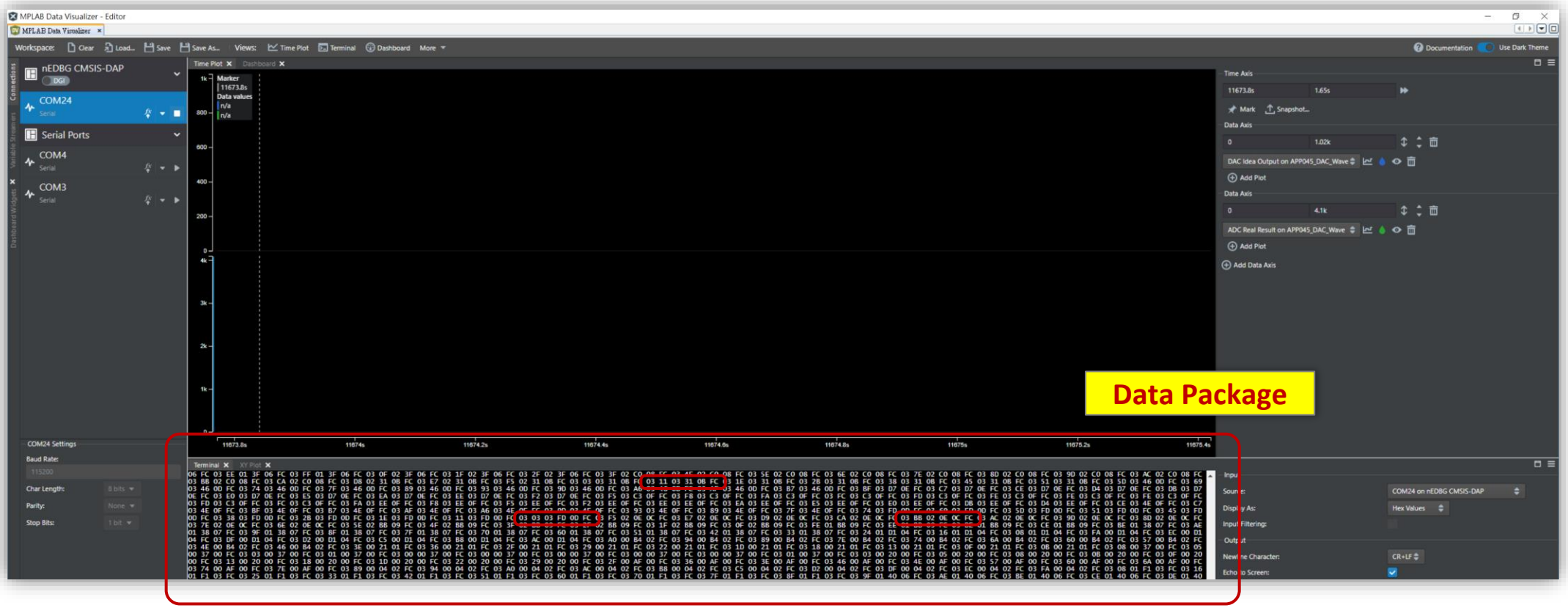
- **Set & Enable COM*x* on left side at MPLAB Data Visualizer.**



🛠 Lab12-4 DAC Wave Output DataVisualizer

Step 9

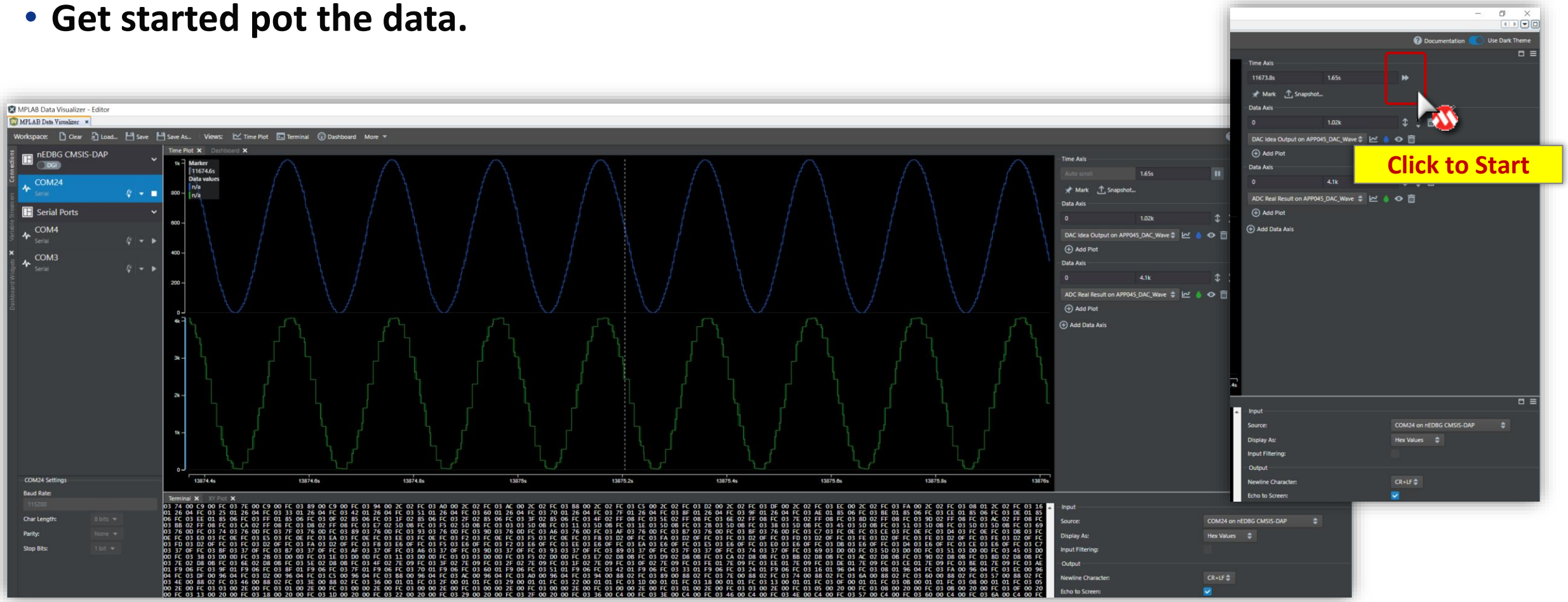
- The Data Package from COM*x* and shows on bottom side at MPLAB Data Visualizer.



🛠 Lab12-4 DAC Wave Output DataVisualizer

Step 10

- **Get started pot the data.**

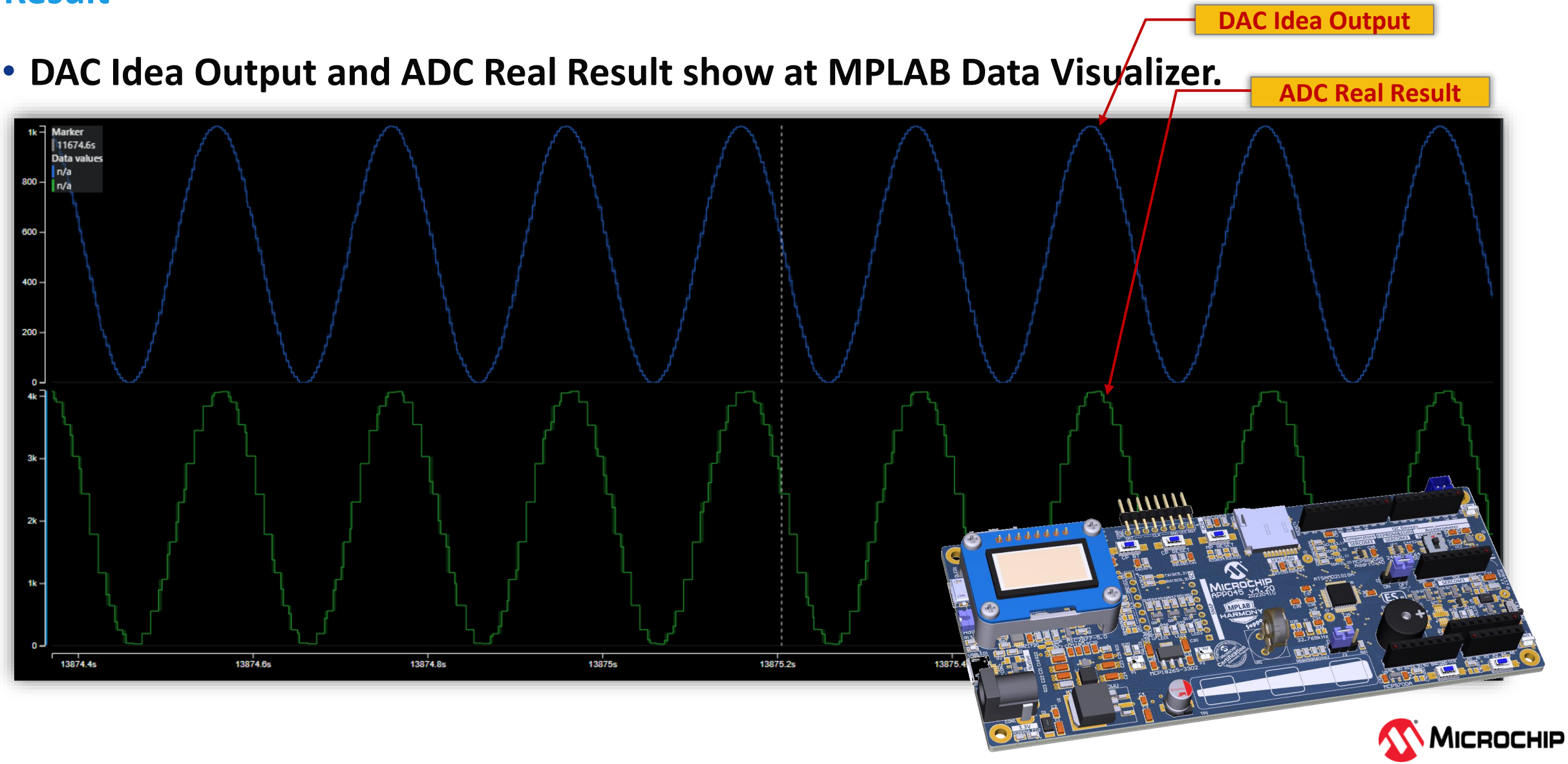


MICROCHIP

🛠 Lab12-4 DAC Wave Output DataVisualizer

Result

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

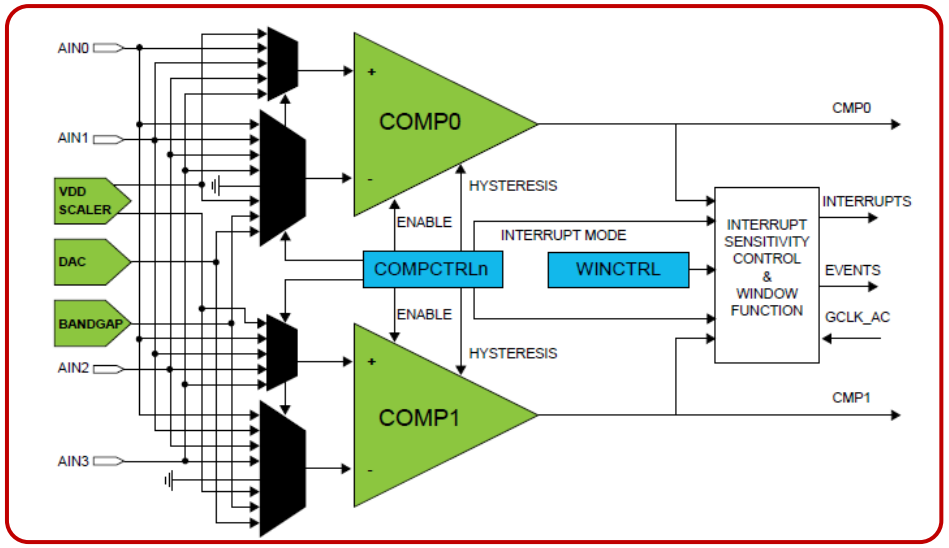


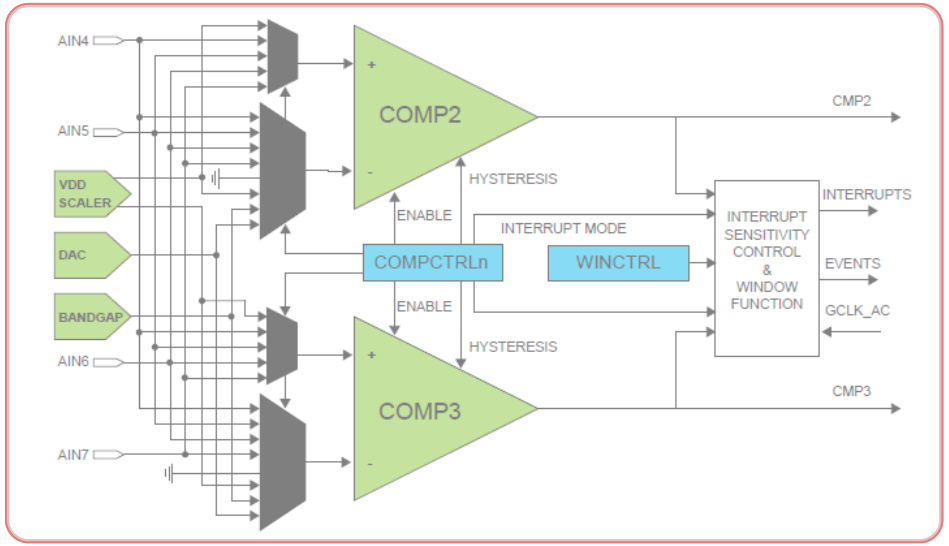
MICROCHIP

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**

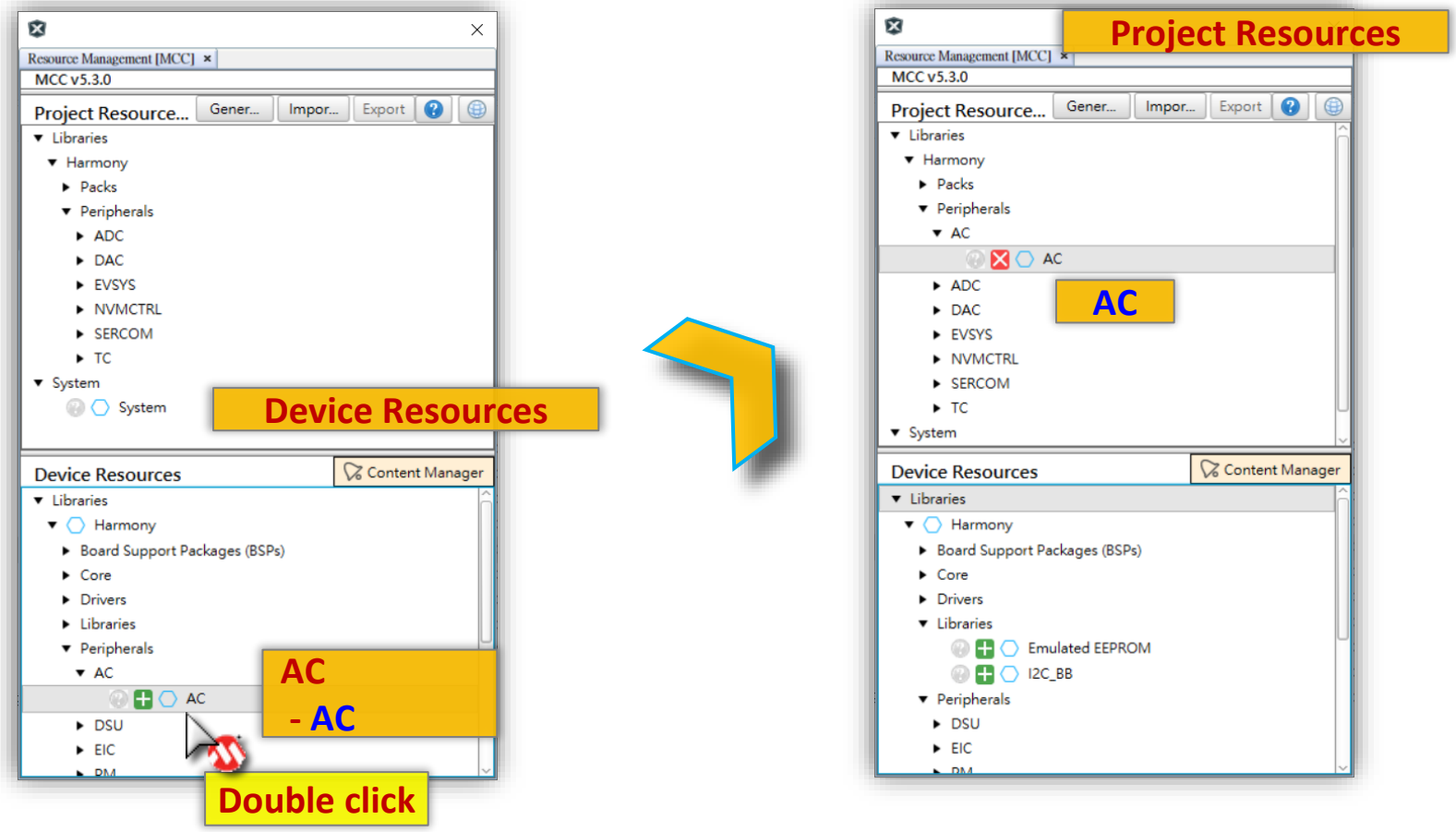




MICROCHIP

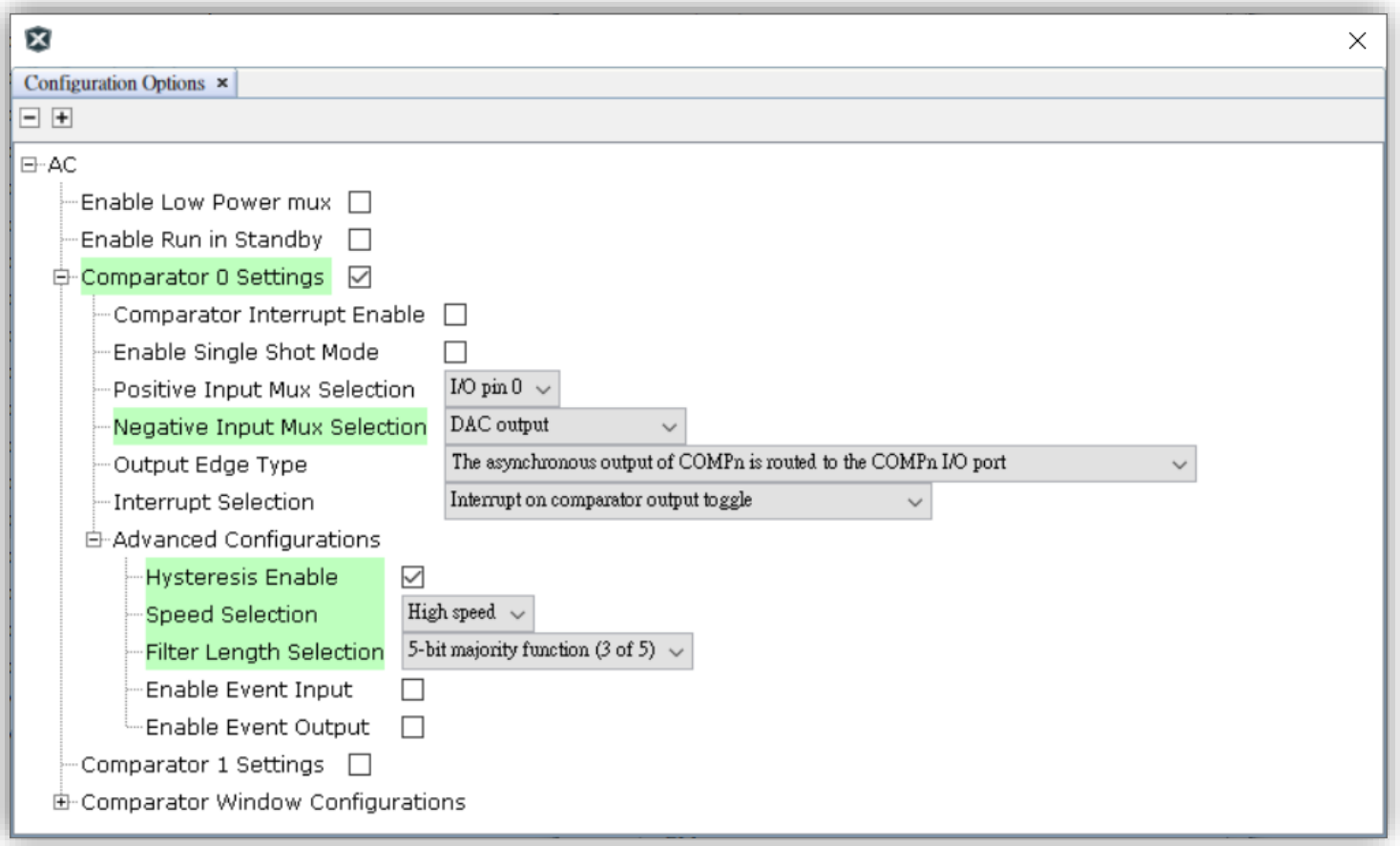
Add AC Function using MCC-Harmony

- Find AC component in “Device Resources” window.
- Double click AC to add to “Project Resources” window.



Single Comparator Example

Configuration Options Example



MICROCHIP

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.**

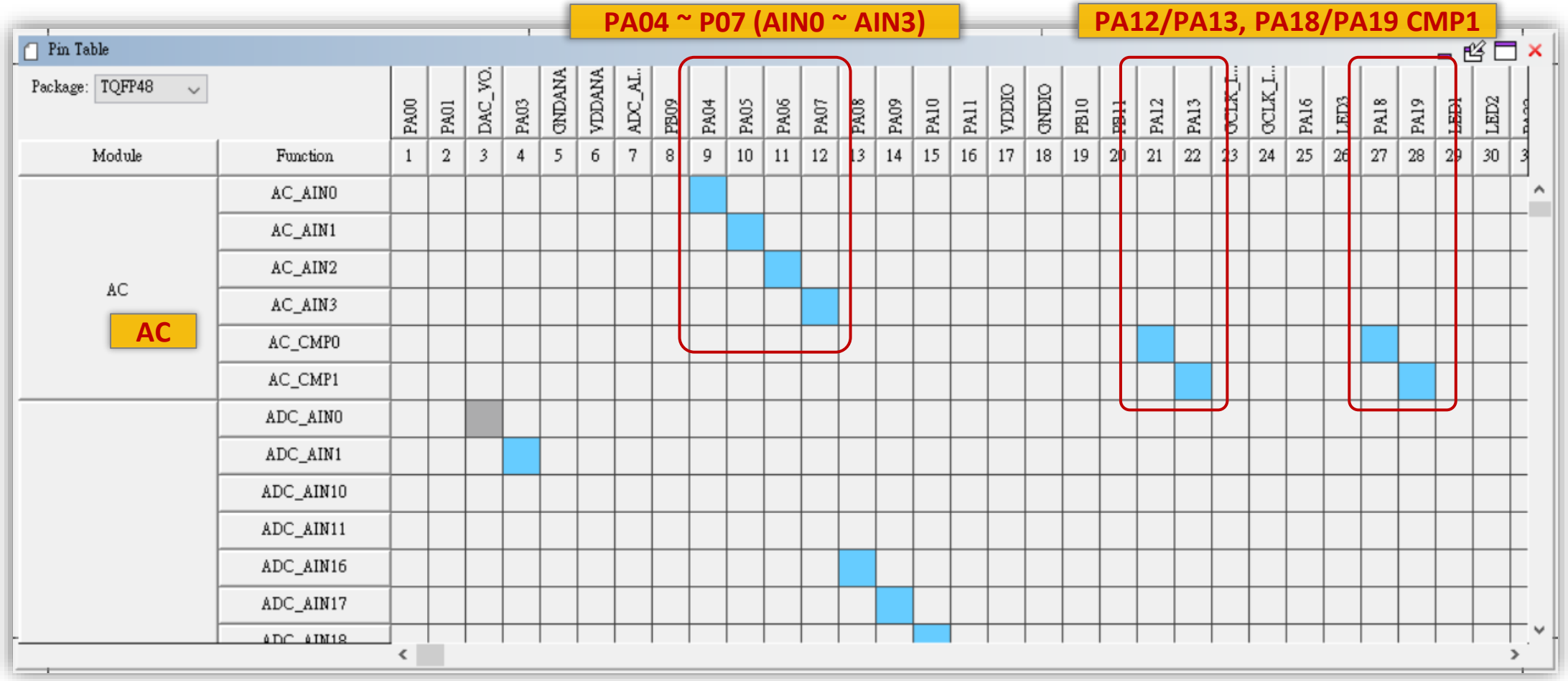
Pin(1)			I/O Pin	Supply	A	B(2)(3)					C	D	E	F	G	H
SAMD2xE	SAMD2xG	SAMD2xJ			EIC	REF	ADC	AC	PTC	DAC	SERCOM(2)(3)	SERCOM-ALT	TC(4) /TCC	TCC	COM	AC/ GCLK
5	9	13	PA04	VDDANA	EXTINT[4]	ADC/VREFB	AIN[4]	AIN[0]	Y[2]			SERCOM0/ PAD[0]	TCC0/WO[0]	TCC3/ WO[2]		
6	10	14	PA05	VDDANA	EXTINT[5]		AIN[5]	AIN[1]	Y[3]			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]

MICROCHIP

AC Pin Mapping cont.

PINMUX Configuration Example



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);
```

MICROCHIP

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	µs
		Enable to ready delay Low power mode	-	12	19	µs

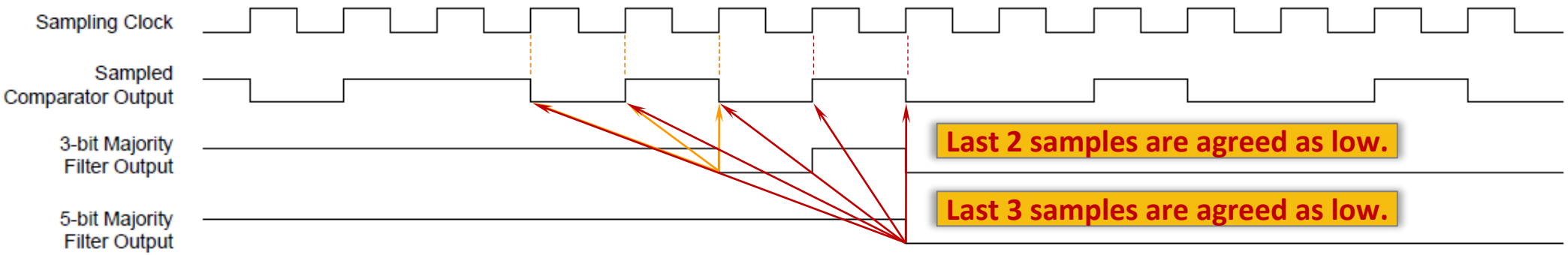
Input Hysteresis

- 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

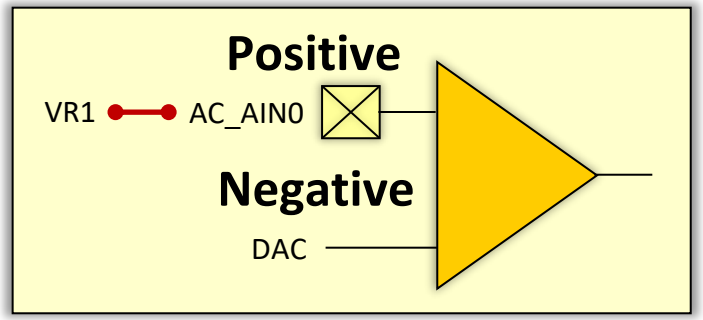
- **The output of the comparators can be filtered digitally to reduce noise.**
- **The filtering is determined by the Filter Length bits in the and is independent for each comparator.**
  - OFF, None (N=0)
  - MAJ3, 3-Bits majority (N=3, 2 of 3)
  - MAJ5, 5-Bits majority (N=5, 3 of 5)
- **Any change 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.**



MICROCHIP

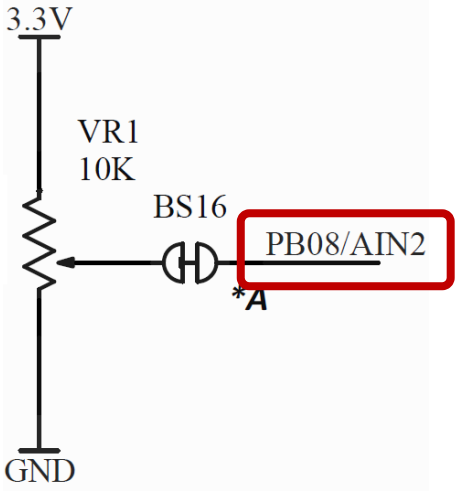
🛠 Lab12-5 AC Single Comparator

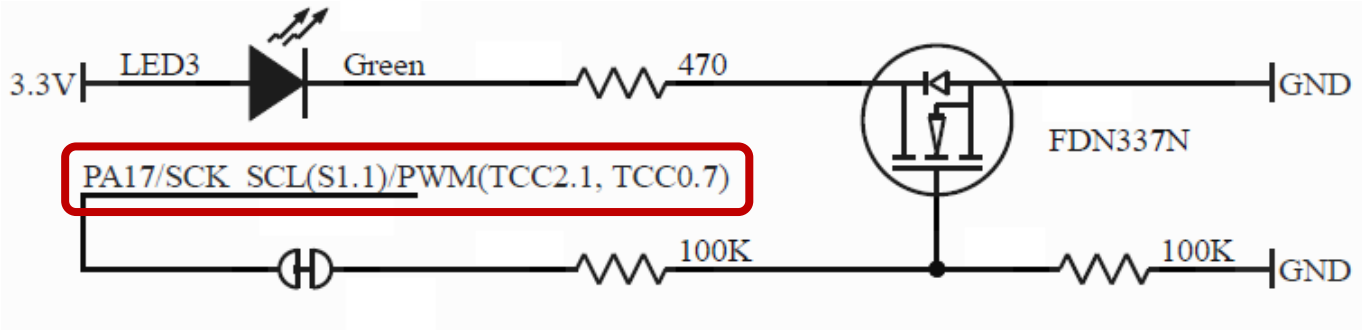
- **Please continue from SAM2001 Lab12-2 (DAC Output to ADC Measurement)**
- **Try to add AC module to project.**
- **Initial COMP0 and Set COMP0 to,**
  - Positive Input : $AIN0_{AC}$ (PA04)
  - Negative Input : DAC (1.65V recommended)
  - Continuous Measurement & Fast Mode
  - Enable Hysteresis & Digital 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 !**

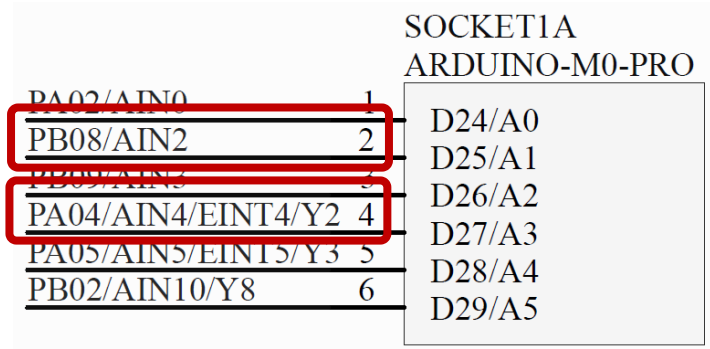


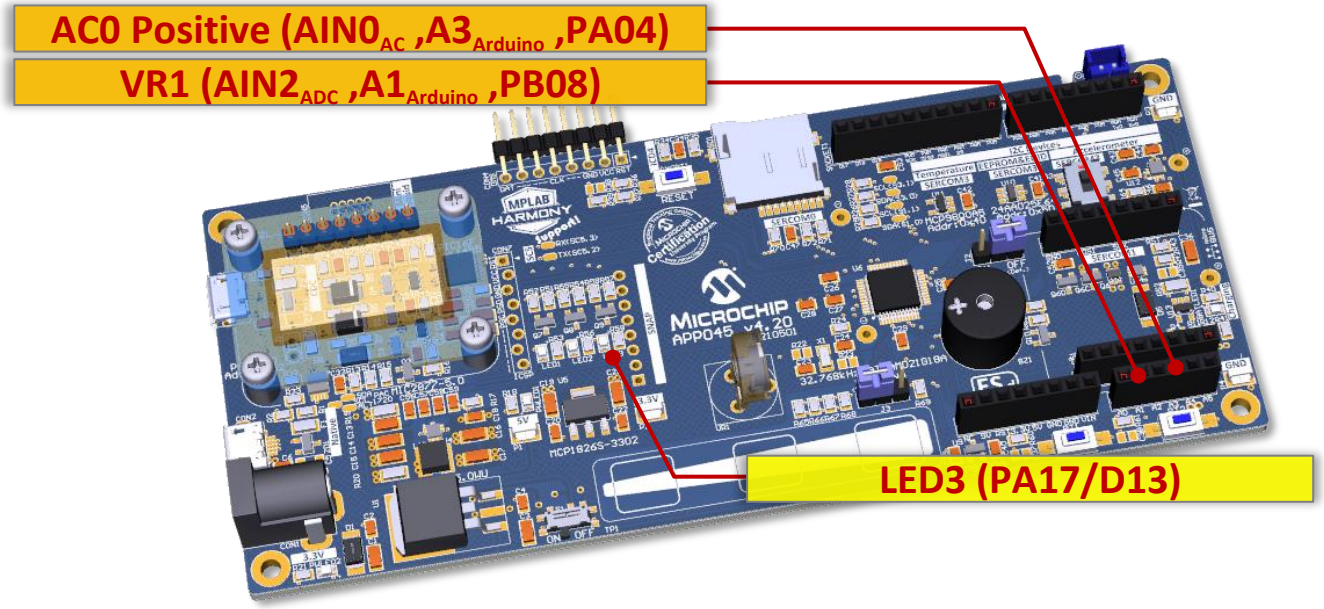
MICROCHIP

🛠 Lab12-5 AC Single Comparator







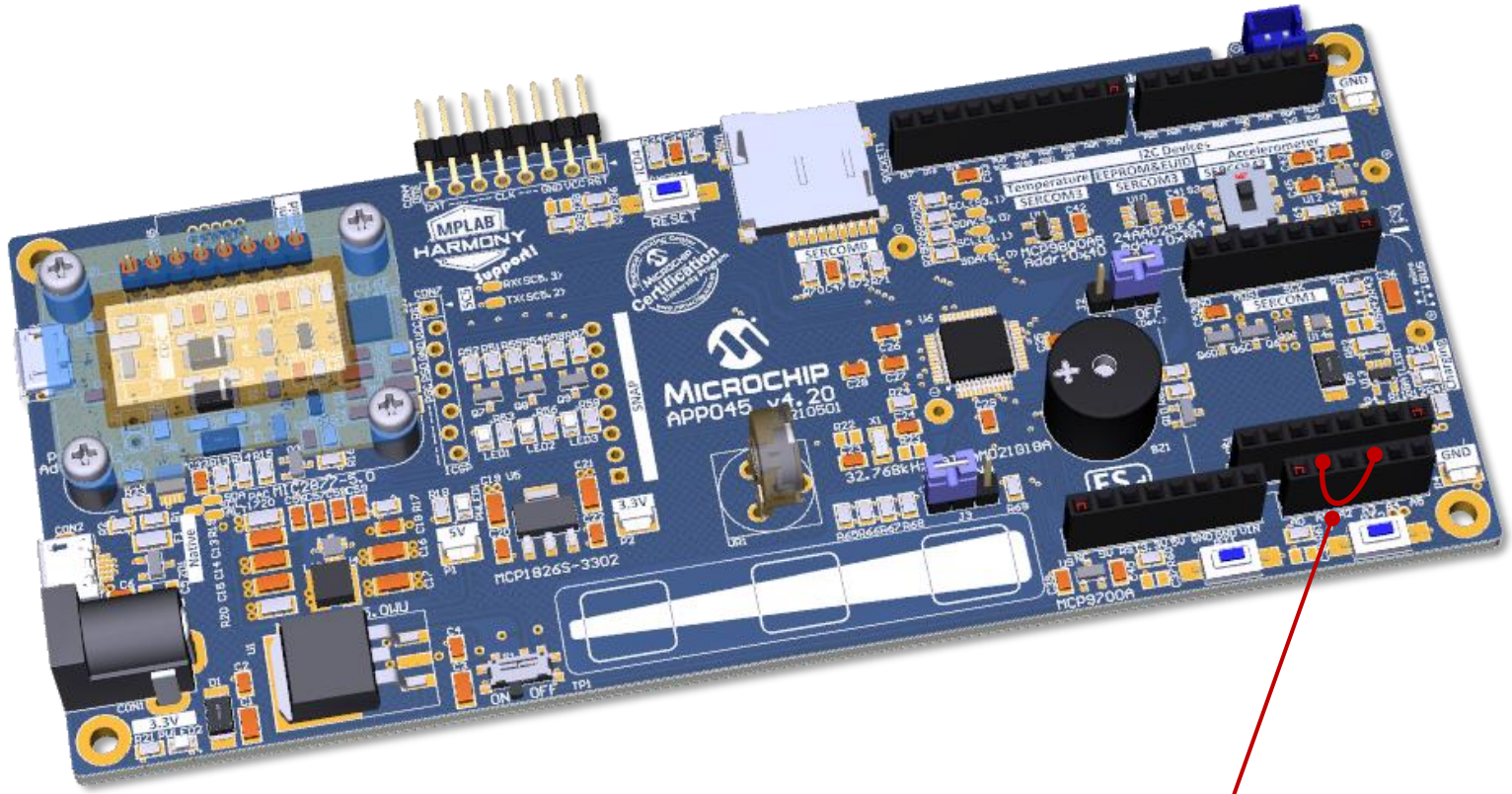


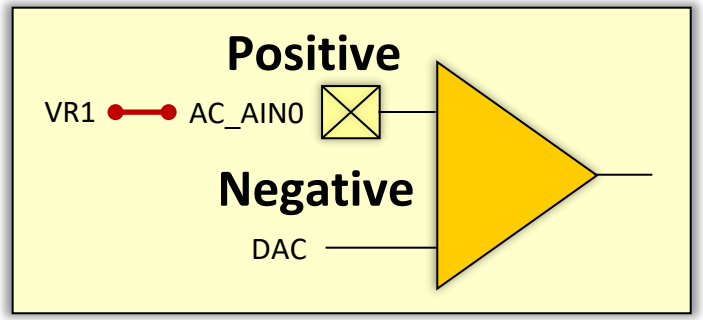
MICROCHIP

🛠 Lab12-5 AC Single Comparator

Step 1

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



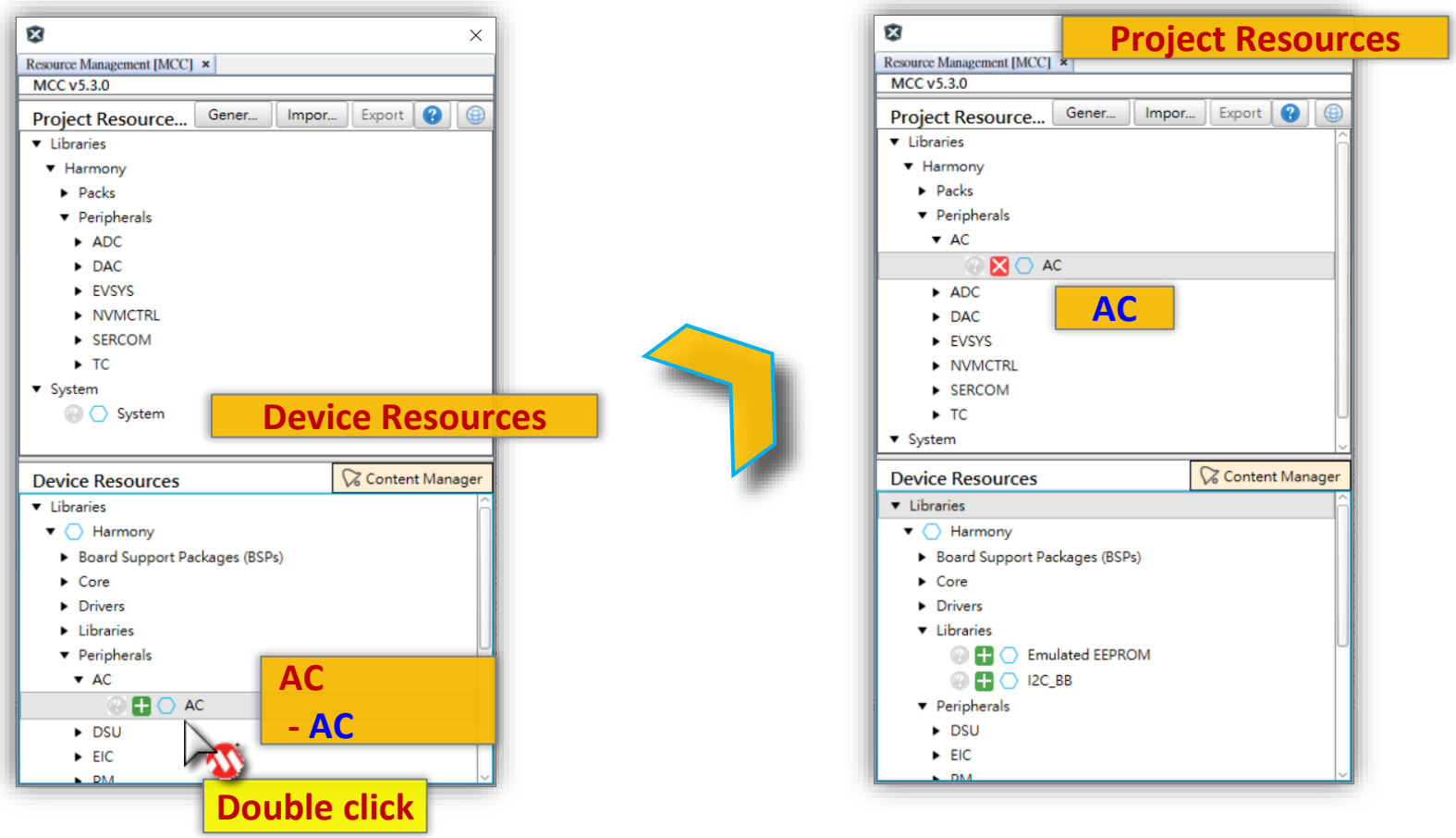


MICROCHIP

🛠 Lab12-5 AC Single Comparator

Step 2

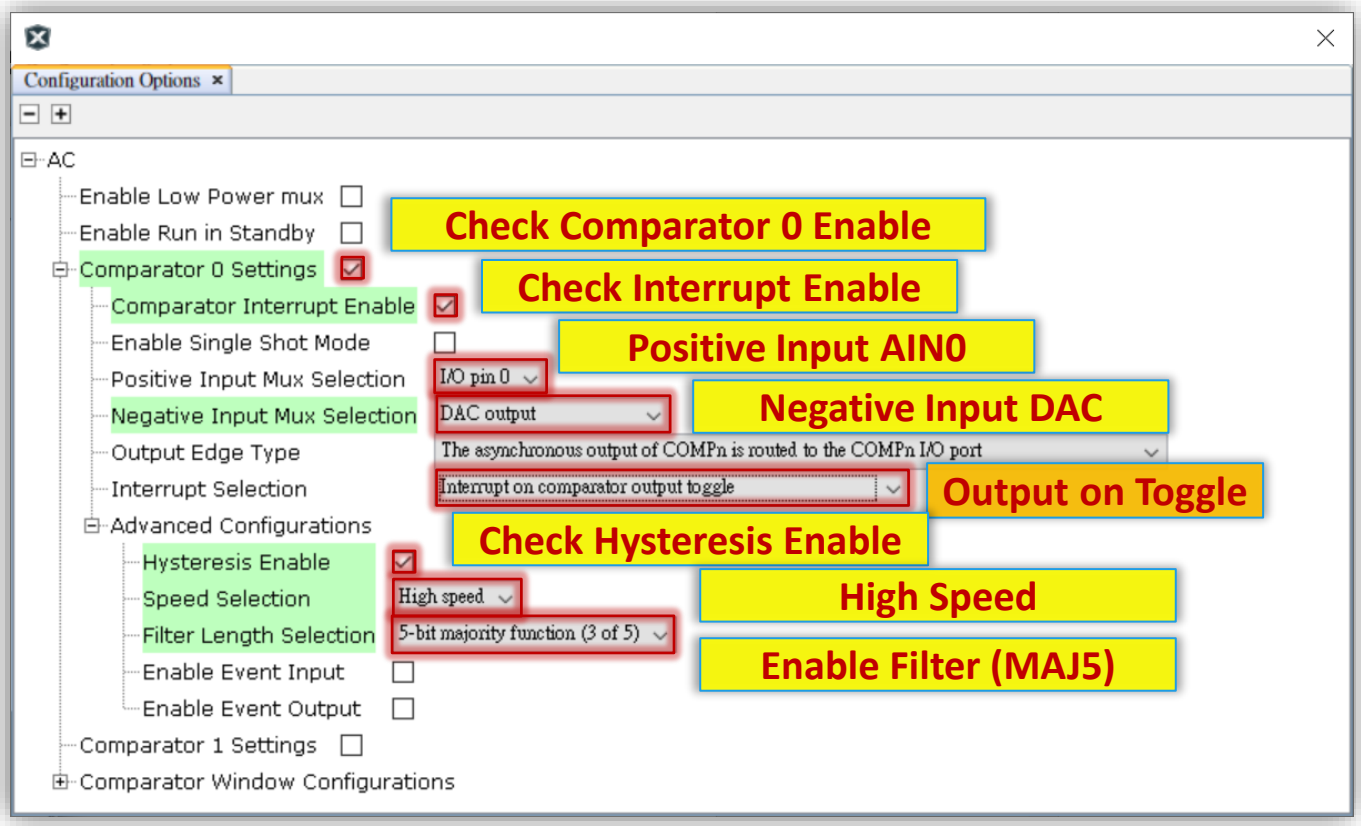
- Find AC component in “Device Resources” window.
- Double click AC to add to “Project Resources” window.



🛠 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**

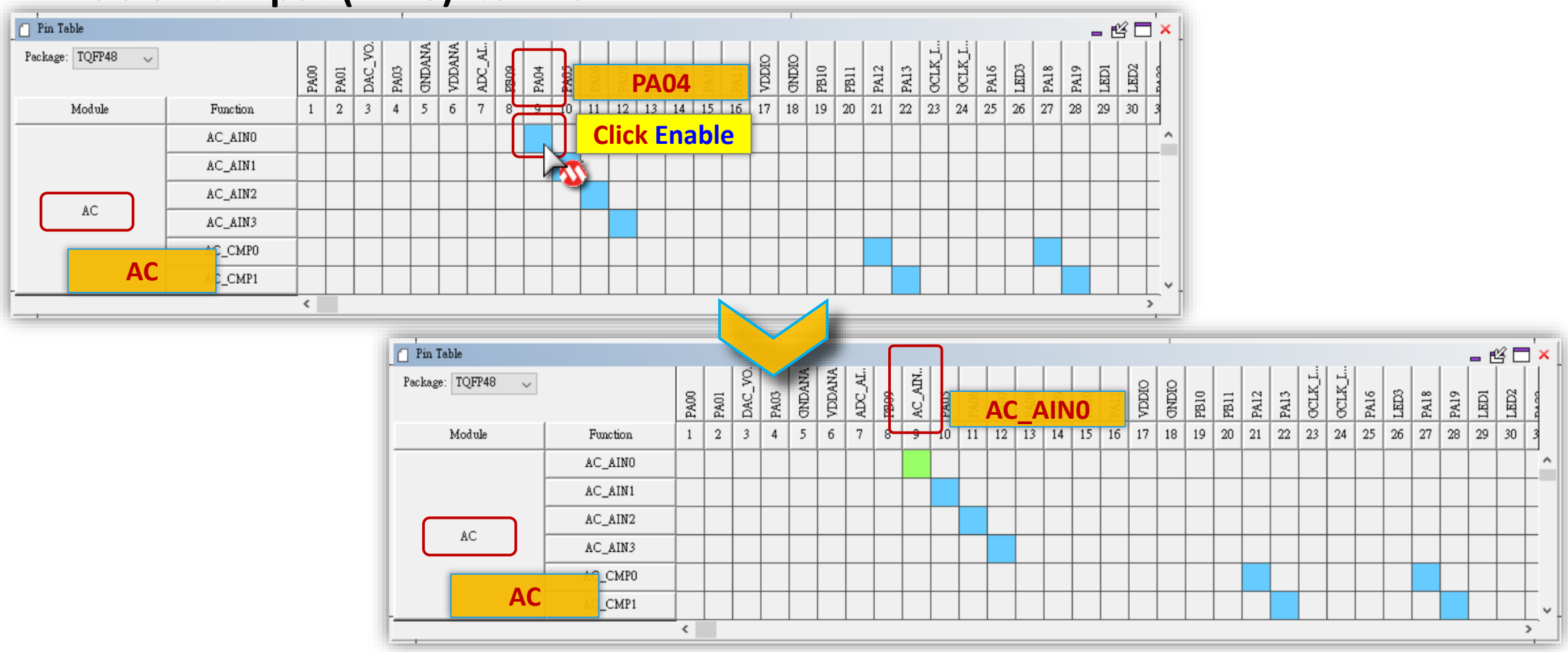


MICROCHIP

🛠 Lab12-5 AC Single Comparator

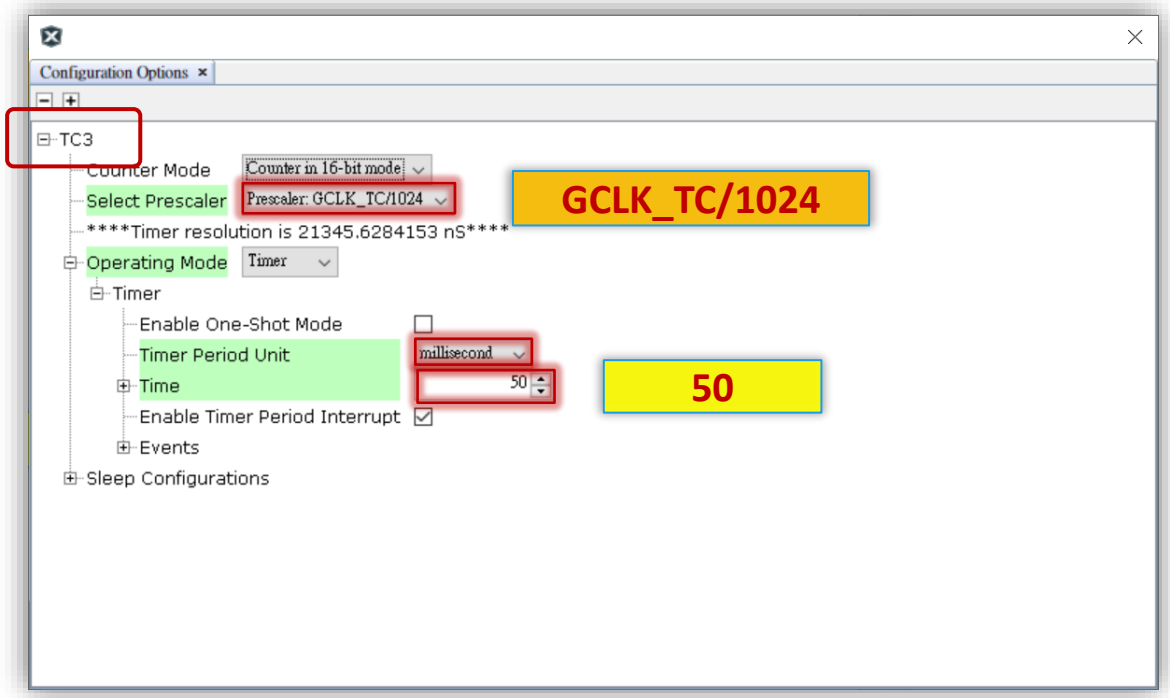
Step 4

- **Enable AC Input (AIN0) to PA04**



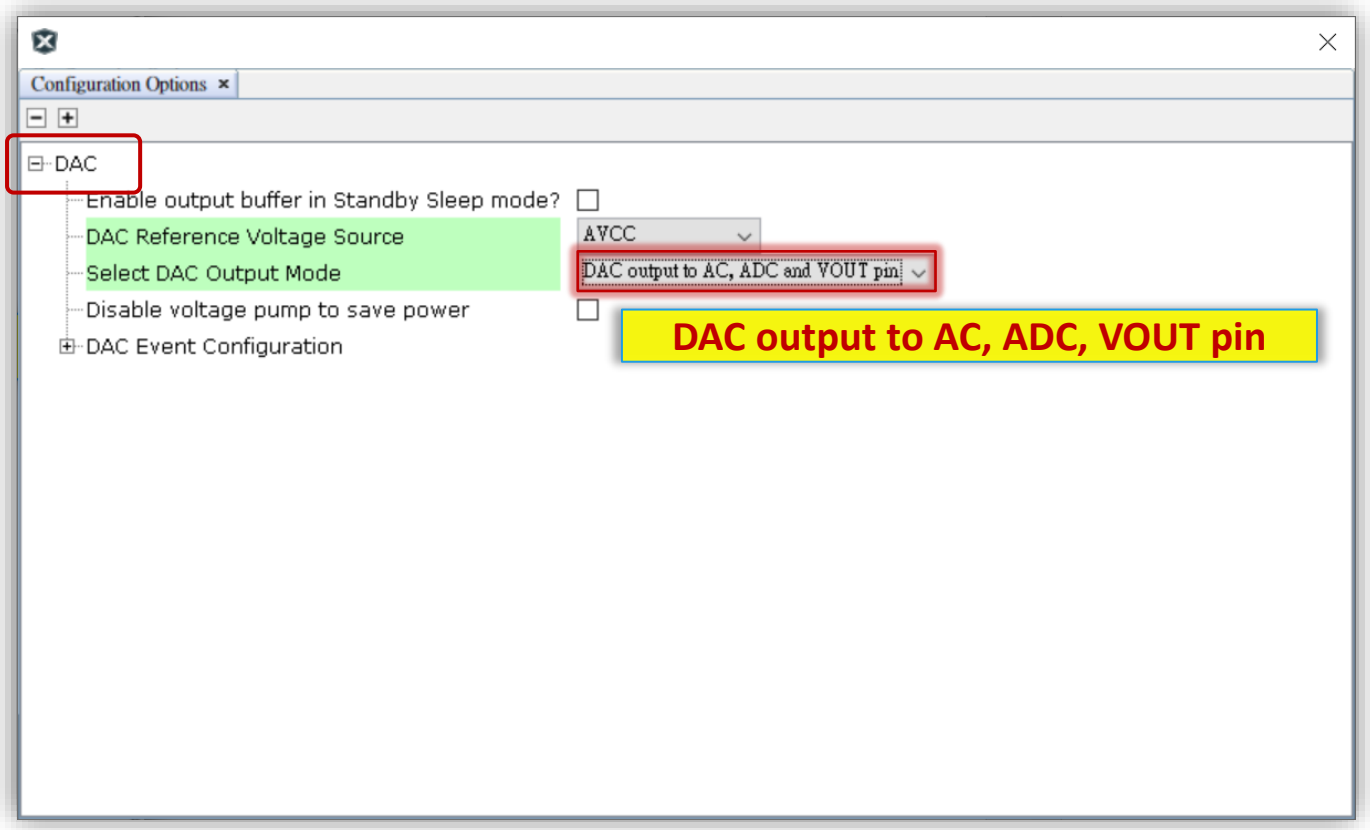
🛠 Lab12-5 AC Single Comparator

Step 5



🛠 Lab12-5 AC Single Comparator

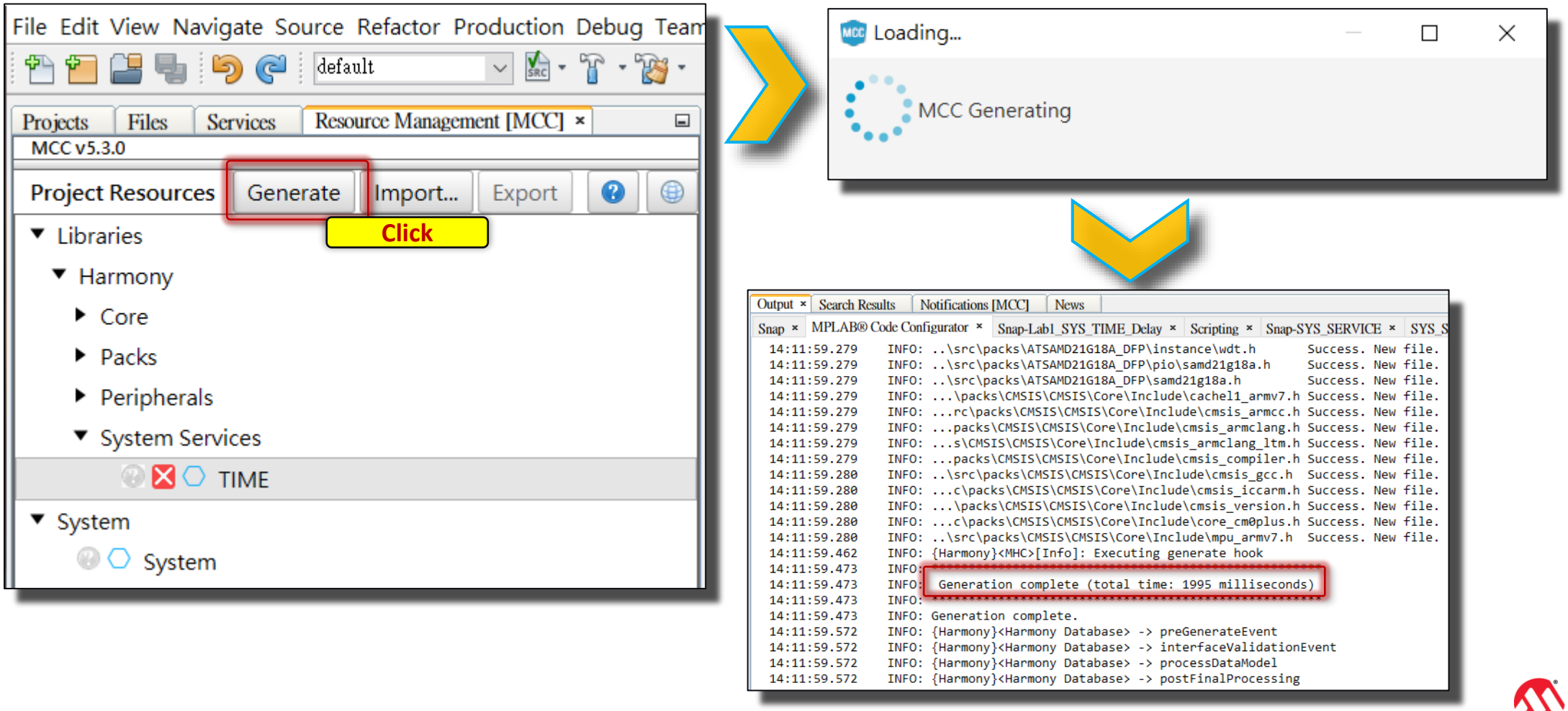
Step 6



🛠 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_HasActived = 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_HasActived = 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

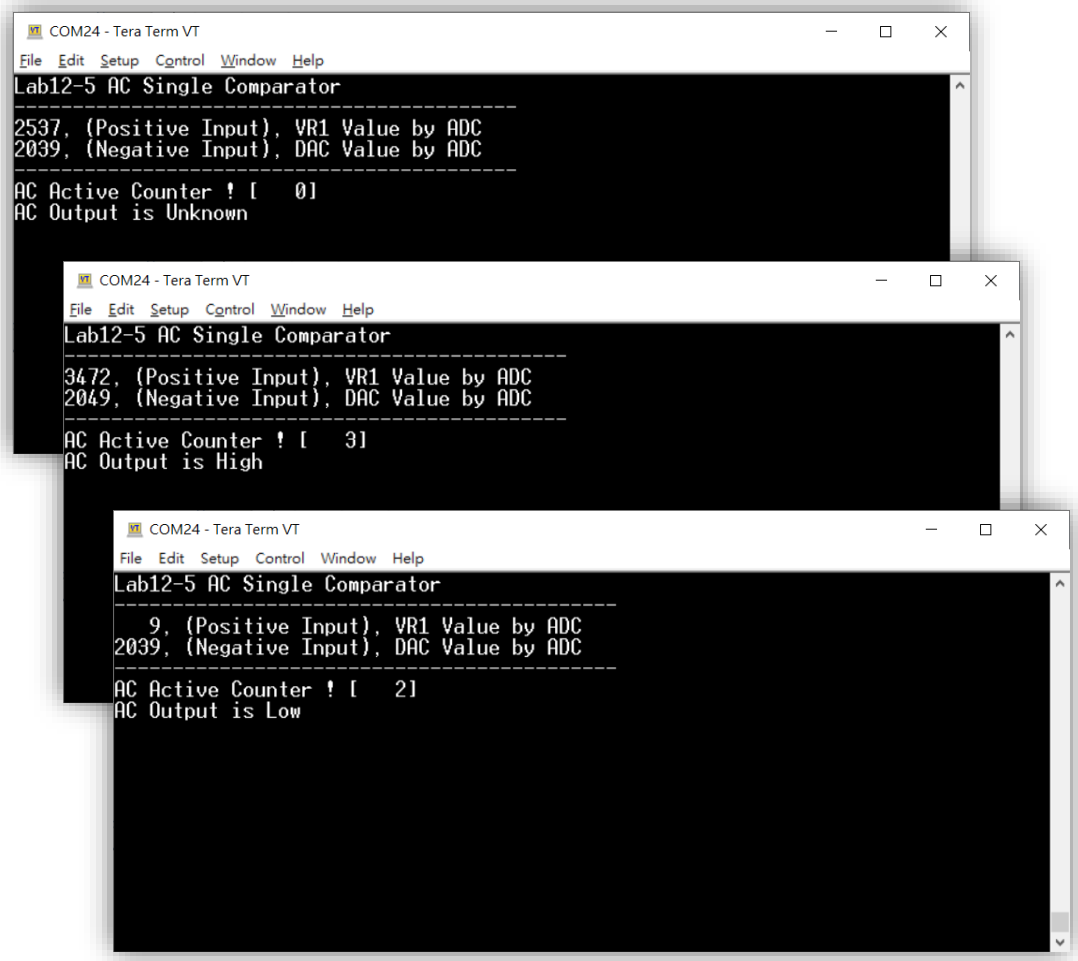
Step 14

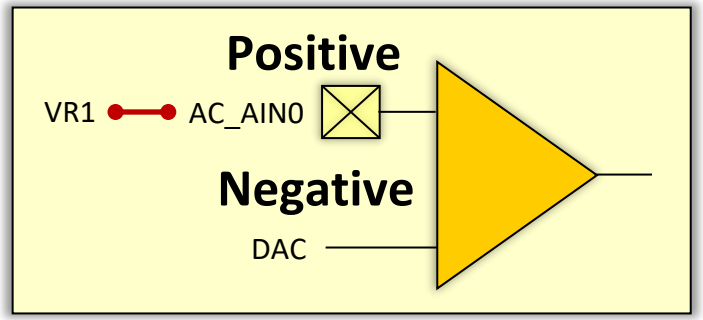
```
int main (void)
{
    SYS_Initialize(NULL);
    ...
    while ( true )
    {
        // TODO 12-5.14
        if (AC0_HasActived)
        {
            AC0_HasActived = 0;

            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);
}
```

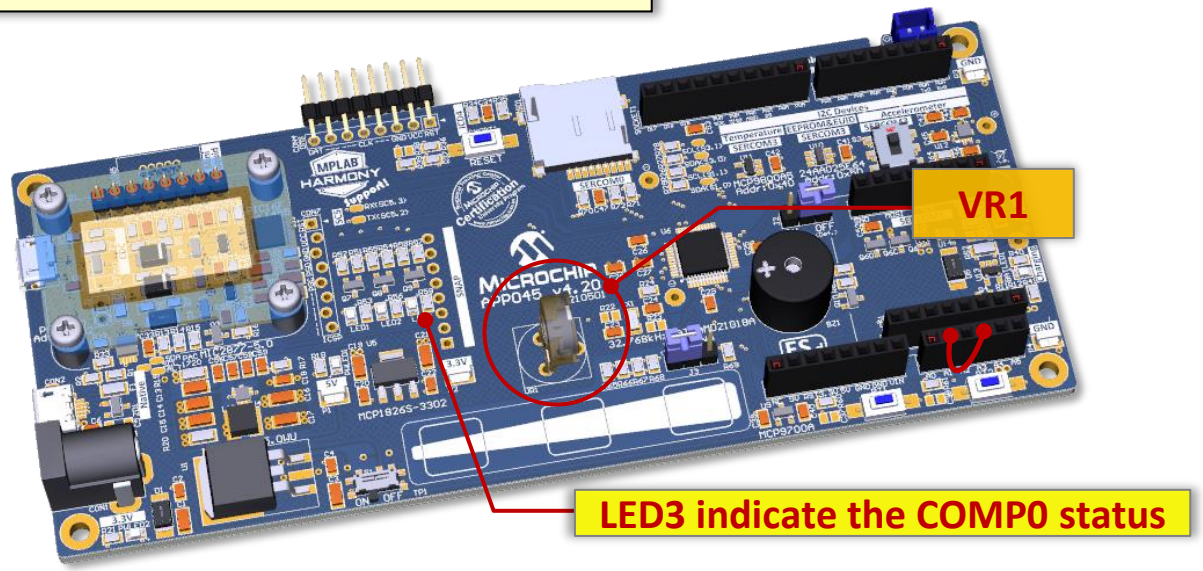
🛠 Lab12-5 AC Single Comparator

Result

- **LED3 indicate the status of COMP0 output.**



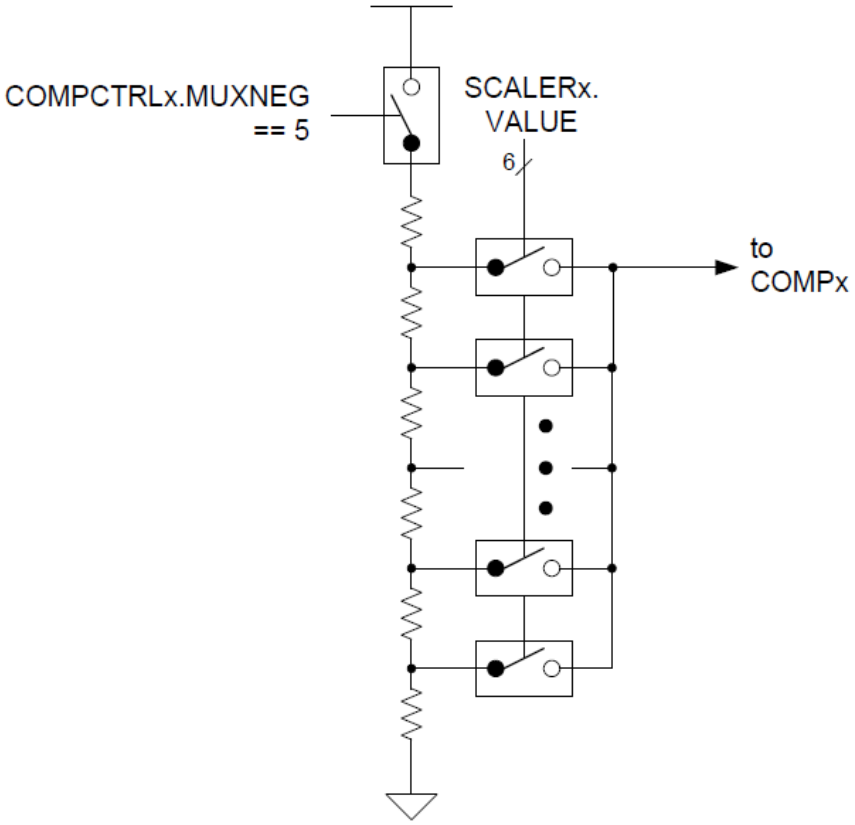




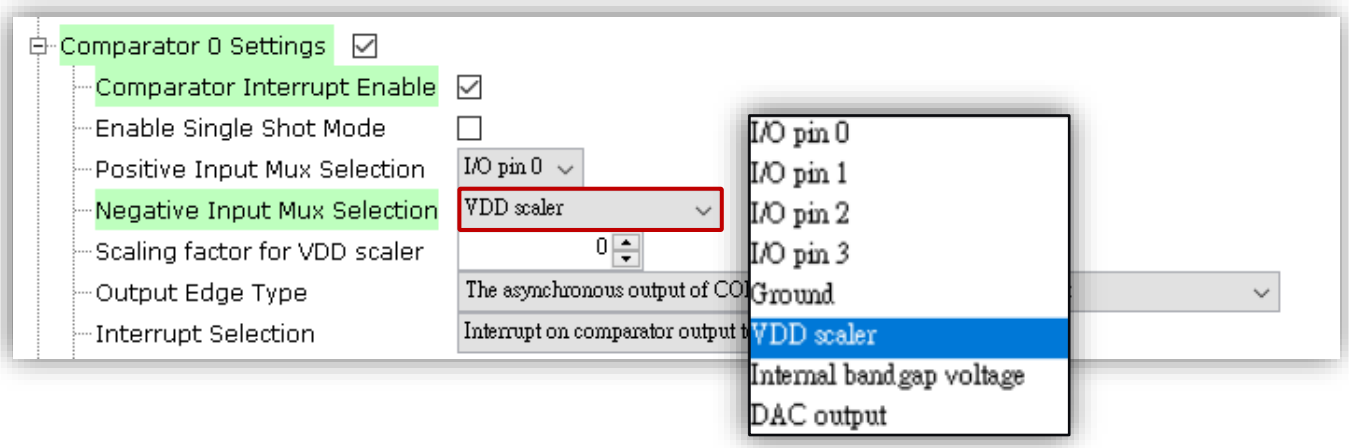
MICROCHIP

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.**

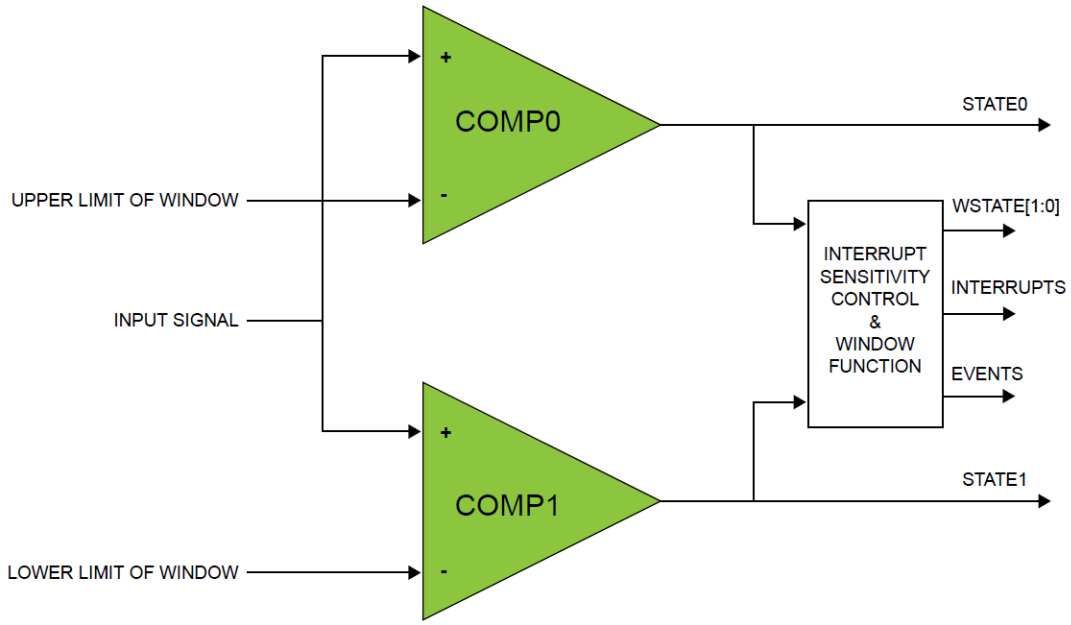


Value	Name	Description
0x0	PIN0	I/O pin 0
0x1	PIN1	I/O pin 1
0x2	PIN2	I/O pin 2
0x3	PIN3	I/O pin 3
0x4	GND	Ground
0x5	VSCALE	VDD scaler
0x6	BANDGAP	Internal bandgap voltage
0x7	DAC	DAC output



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.

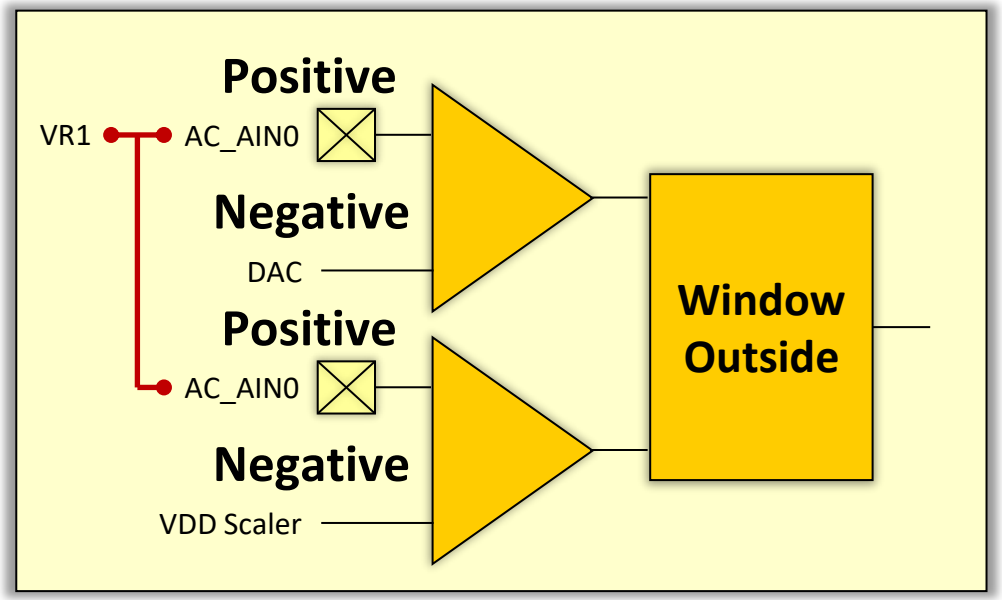


Value	Name	Description
0x0	ABOVE	Interrupt on signal above window
0x1	INSIDE	Interrupt on signal inside window
0x2	BELOW	Interrupt on signal below window
0x3	OUTSIDE	Interrupt on signal outside window

🛠 Lab12-6 AC Window

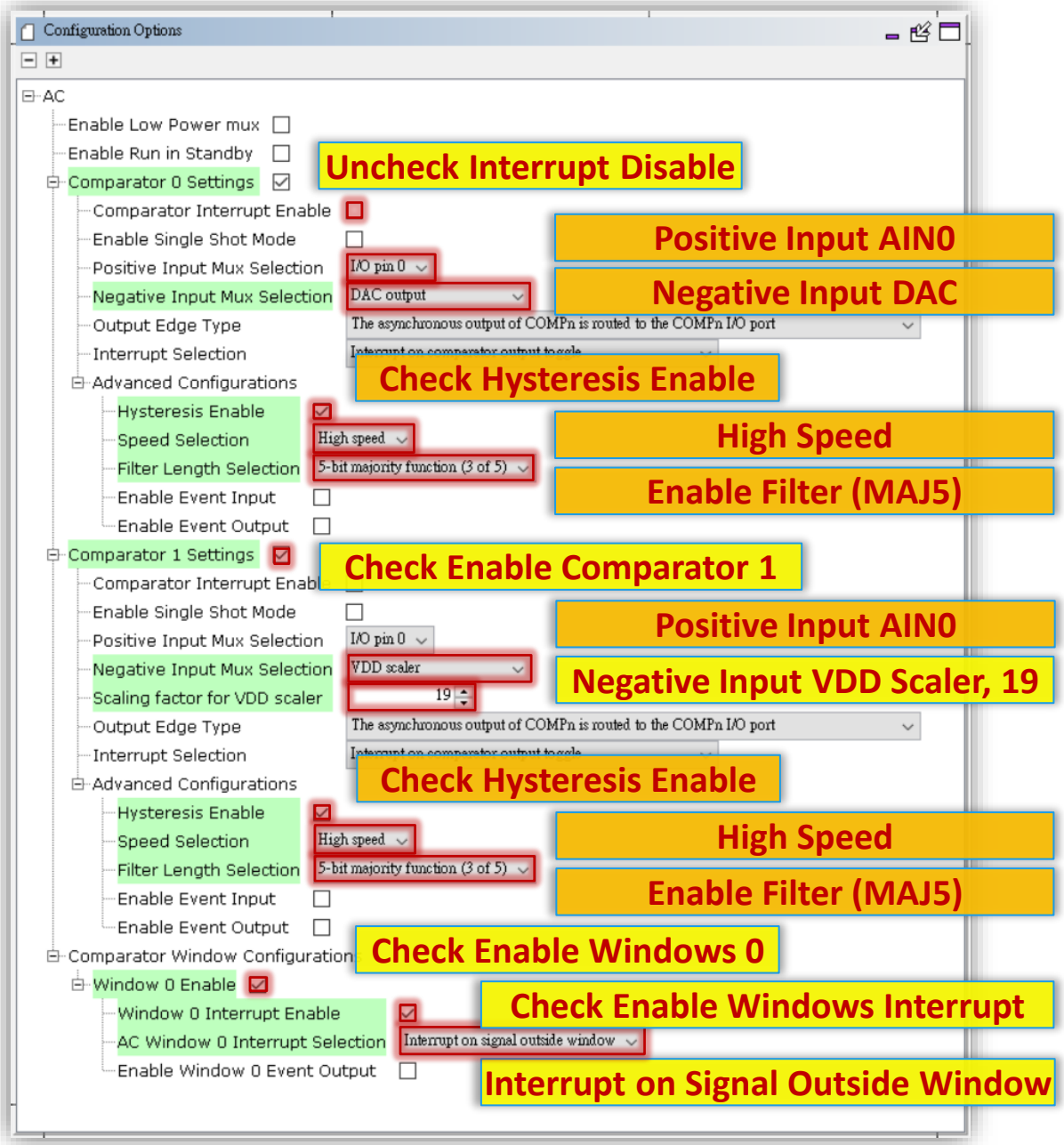
- **Please continue from SAM2001 Lab12-5 (AC Single Comparator)**
- **Try to enable AC Window mode.**
- **Set the Window mode,**
  - Upper Limit is DAC (3.0V recommended)
  - Lower Limit is VDD Scaler (1.0V recommended)
  - Interrupt on Signal “Outside” Window
- **Use AC to compare VR1 voltage, then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

- **Let’s go !**



Lab12-6 AC Window

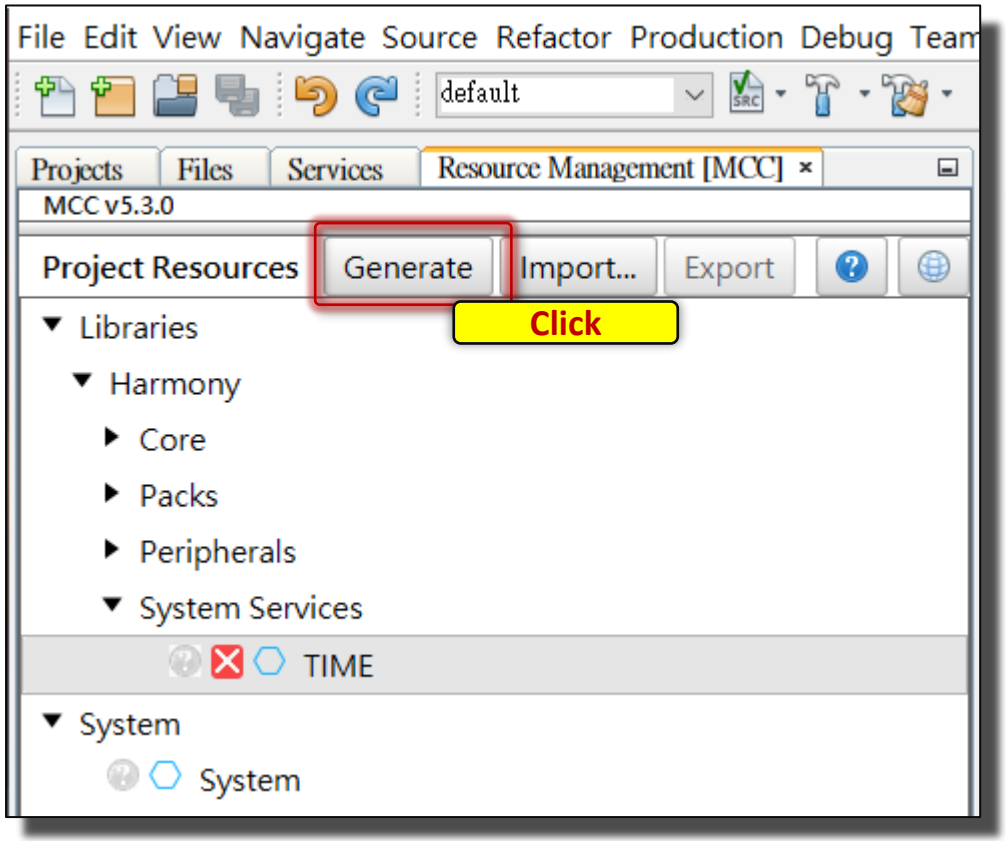
Step 1

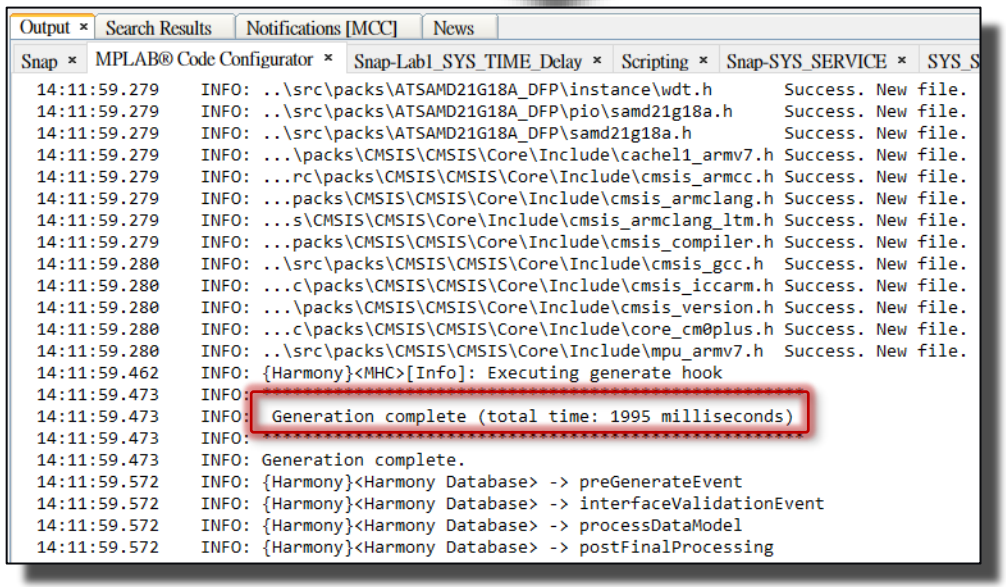


🛠 Lab12-6 AC Window

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_HasActived)
        {
            ...
            // 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

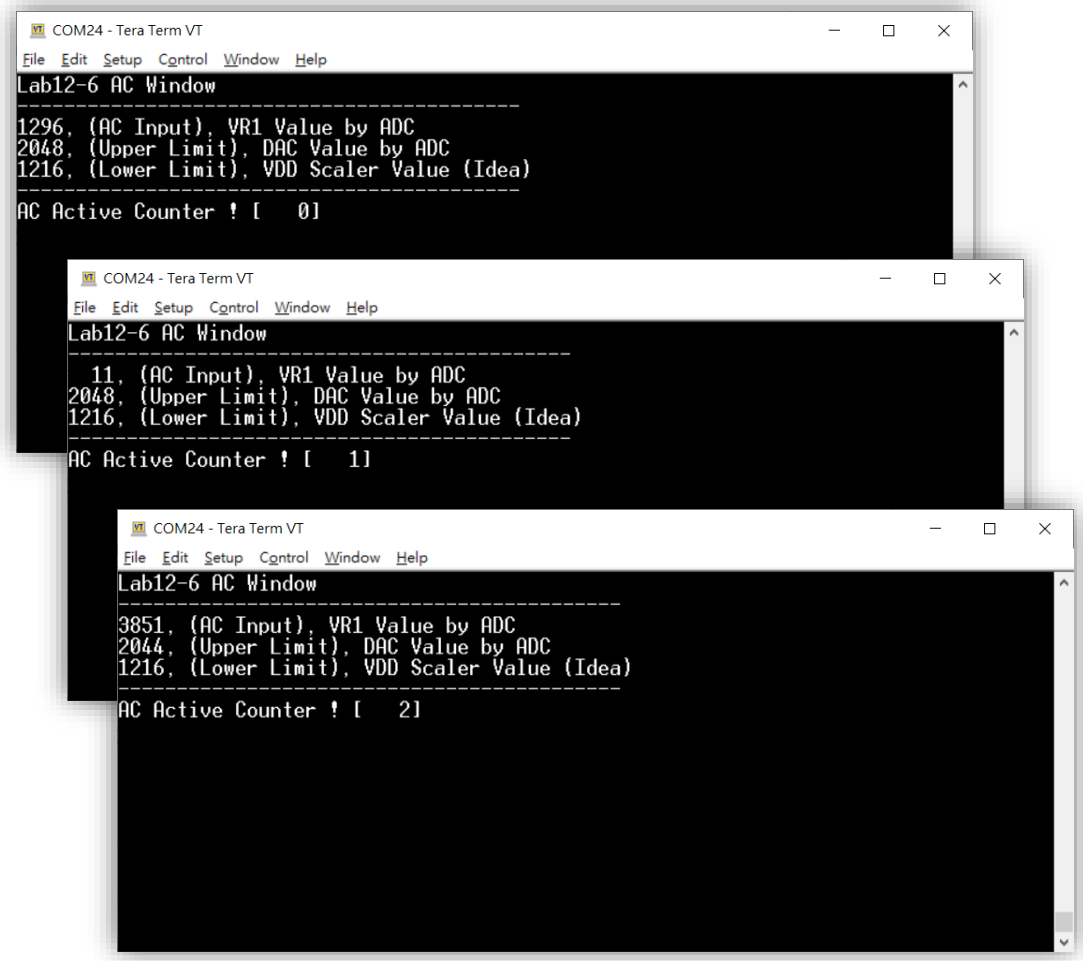
Step 5

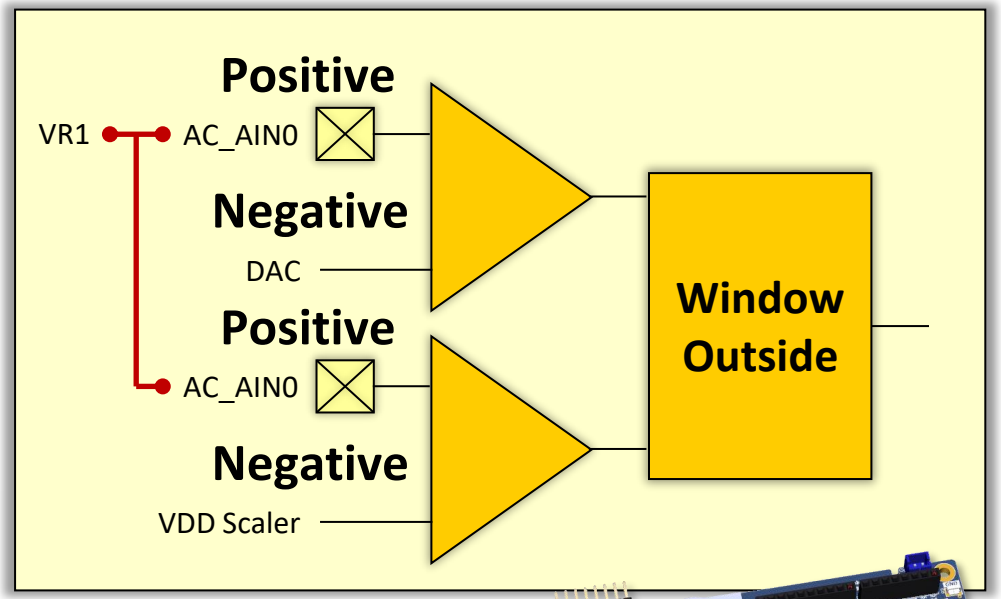
```
int main (void)
{
    // TODO 12-6.05
    DAC_DataWrite((uint16_t) 930);
    // TODO 12-6.06
    myprintf("Lab12-6 AC Window\r\n");
    // TODO 12-6.07
    myprintf("\033[7;1H");
    myprintf("AC Active Counter ! [%4d]\r\n", AC0_ActiveCounter);
    while ( true )
    {
        // TODO 12-6.08
        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_HasActived)
        {
            // TODO 12-6.09
            myprintf("\033[7;1H");
            myprintf("AC Active Counter ! [%4d]\r\n", AC0_ActiveCounter);
        }
    }
    return ( EXIT_FAILURE);
}
```

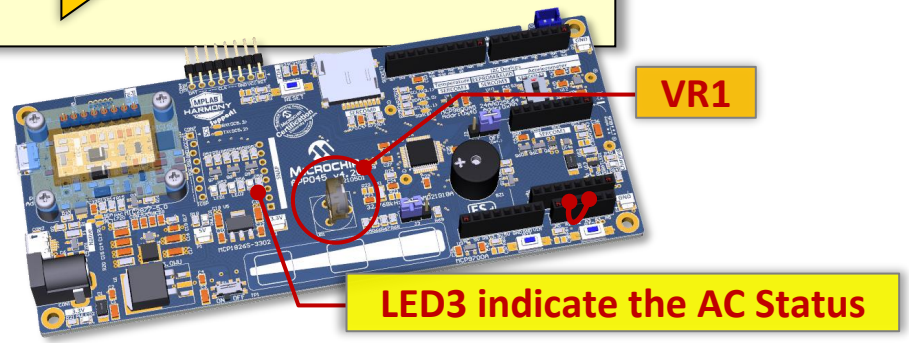
🛠 Lab12-6 AC Window

Result

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





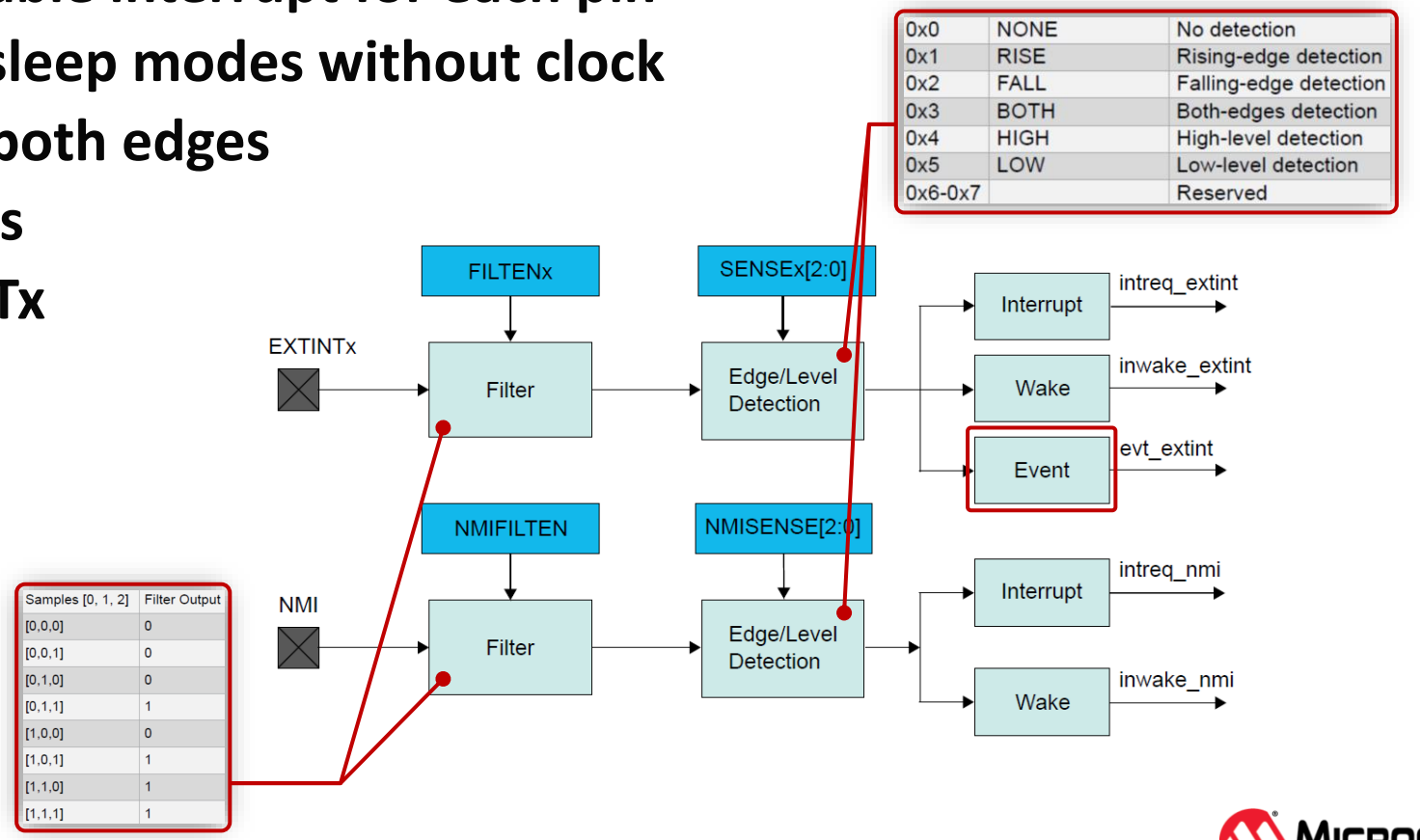


MICROCHIP

External Interrupt Controller (EIC)

External Interrupt Controller (EIC)

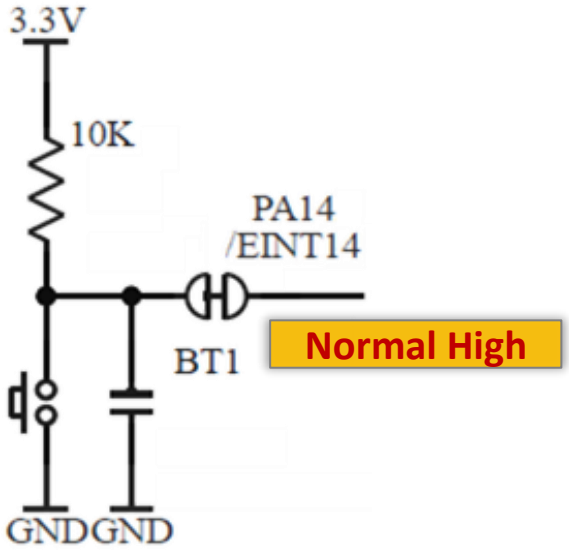
- The External Interrupt Controller (EIC) allows that the external pins to be configured as interrupt lines.
  - Up to 16 external pins (EXTINTx), plus one non-maskable pin (NMI)
  - Dedicated, individually maskable interrupt for each pin
  - Asynchronous interrupts for sleep modes without clock
  - 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.**

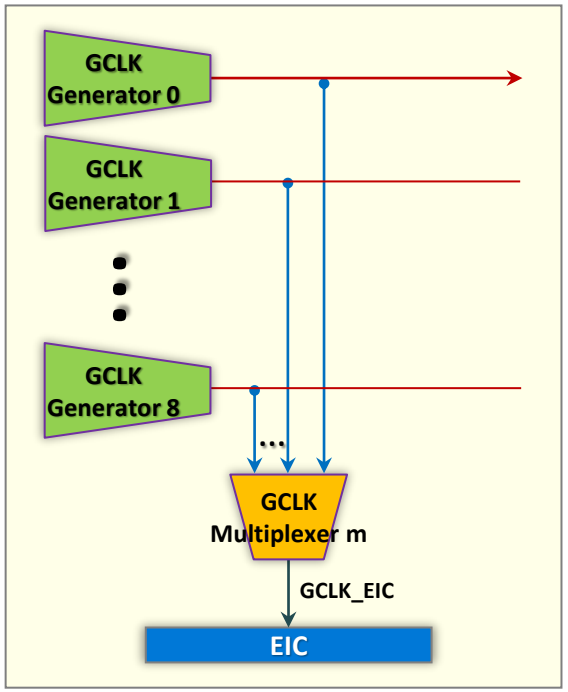
NONE	No detection
RISE	Rising-edge detection
FALL	Falling-edge detection
BOTH	Both-edges detection
HIGH	High-level detection
LOW	Low-level detection
	Reserved



Vote Filter

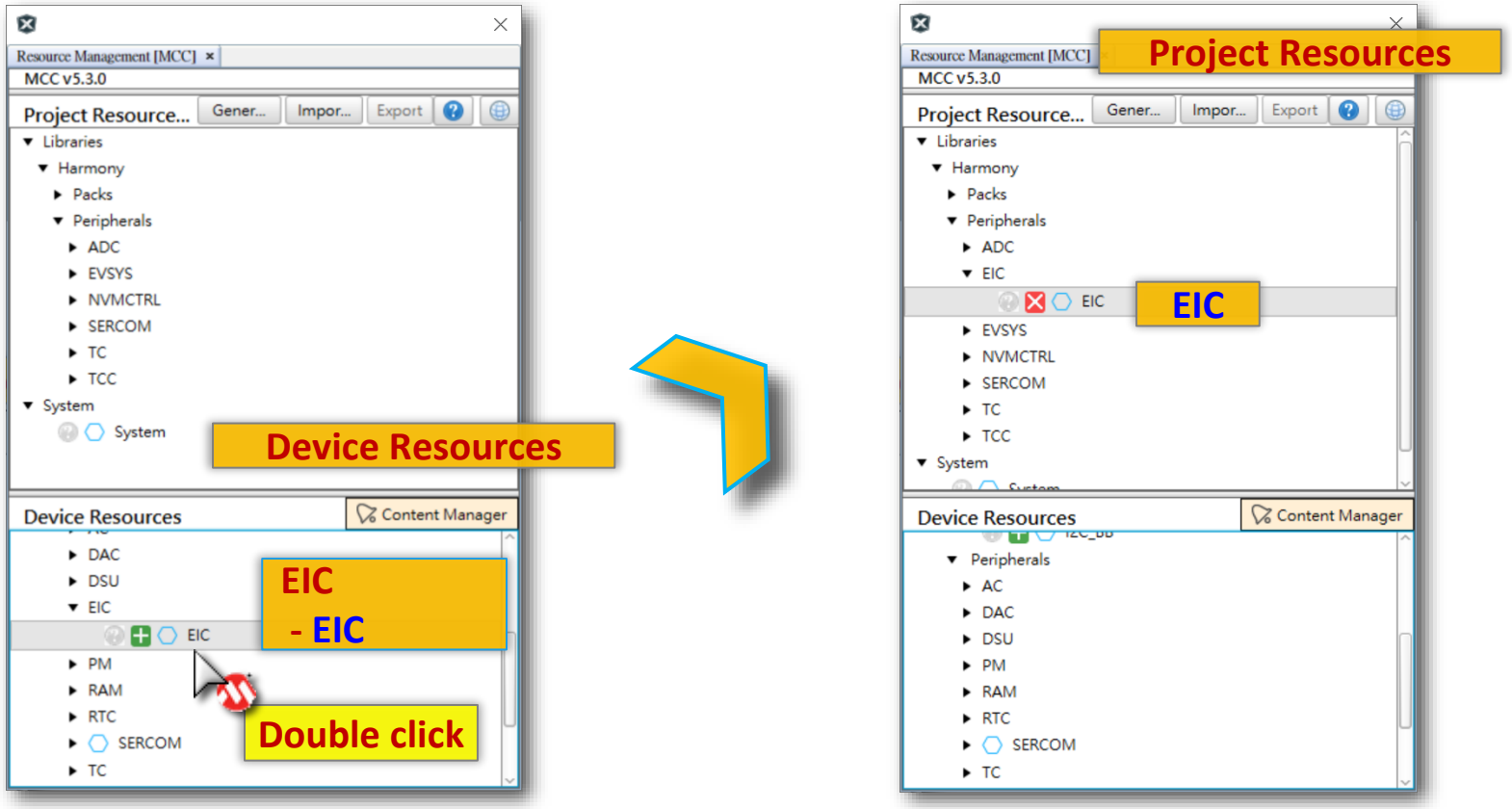
- External pin can be filtered by a majority vote filtering.
  Three times capture the pin value with GCLK_EIC and outputs the value when two or more samples are equal.
- The frequency of GCLK_EIC to decide the capture period. Lower frequency clock is more suit for user interface input, just like physical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



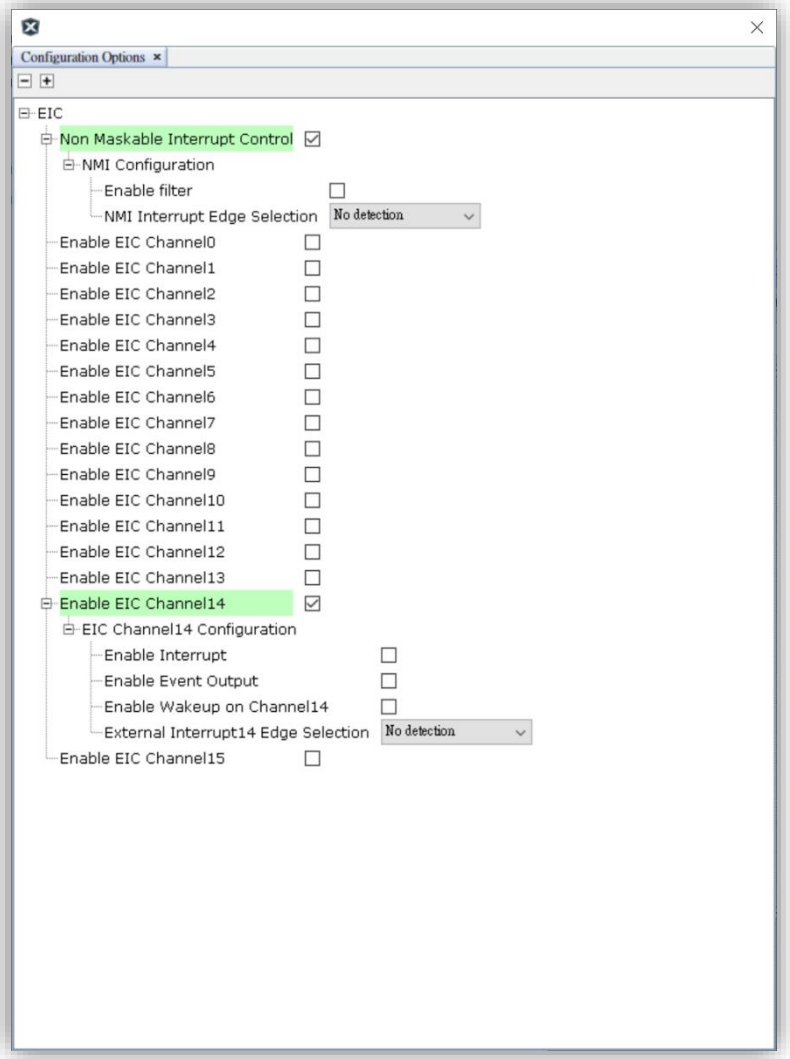
Add EIC Function using MCC-Harmony

- Find EIC component in “Device Resources” window.
- Double click EIC to add to “Project Resources” 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]

PA14

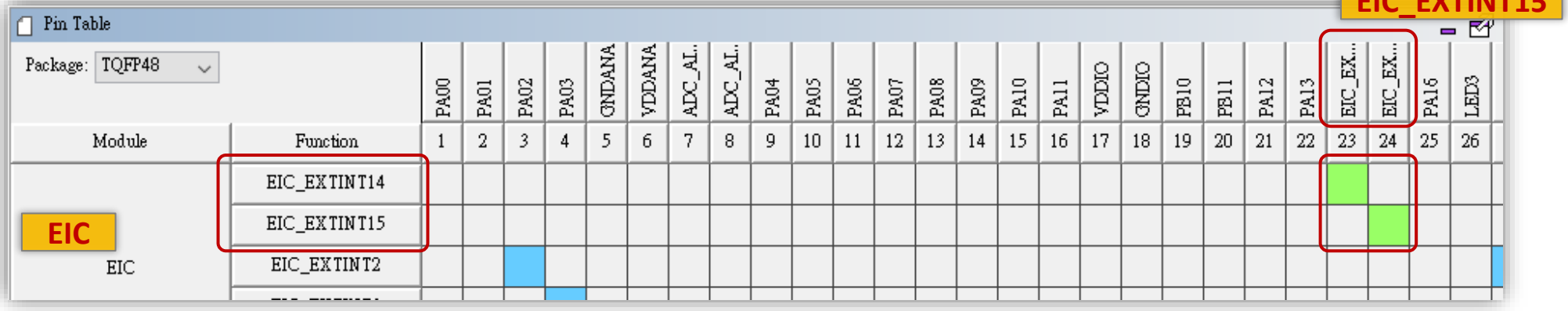
PA15

EXTINT[14]

EXTINT[15]

EIC_EXTINT14

EIC_EXTINT15



MICROCHIP

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 )
   {...}
}
```

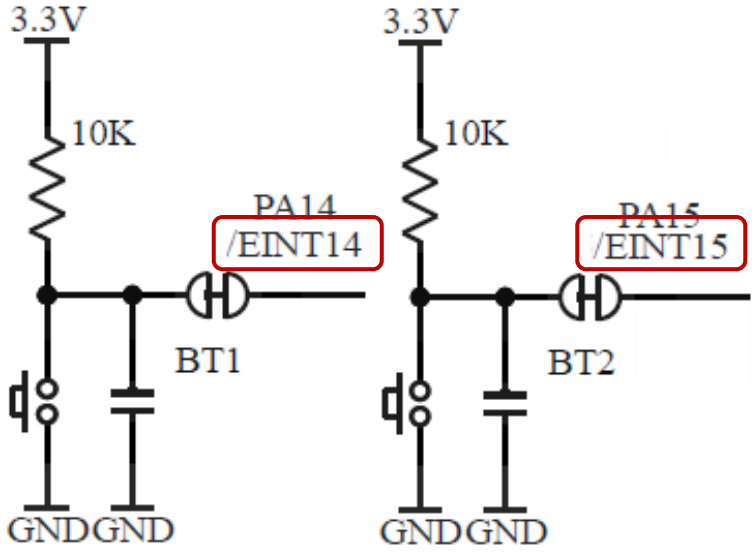
MICROCHIP

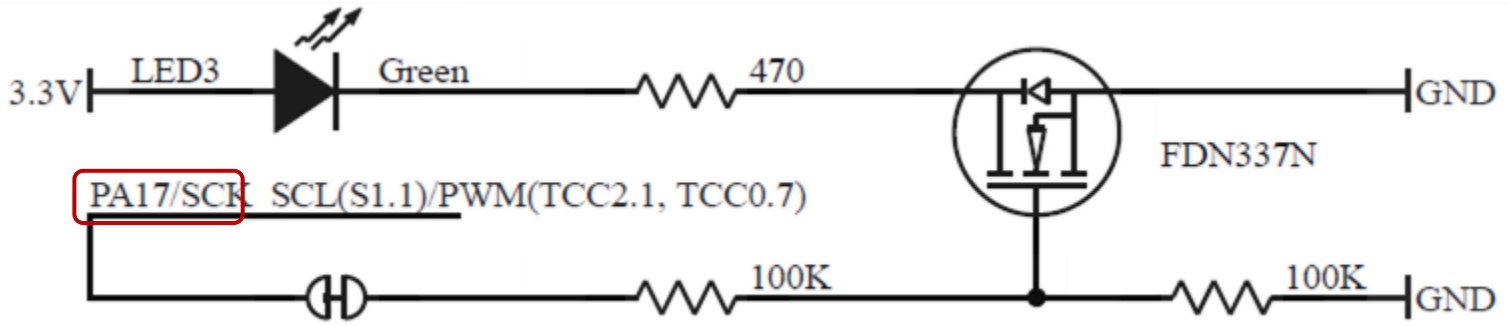
🛠 Lab16-1 Button as External Interrupt

- 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).
  - EXTNIT14 -> BT1(PA14), Low-Level detection, Enable filter.
  - EXTNIT15 -> 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]
16	24	32	PA15	VDDIO	EXTINT[15]						SERCOM2/ PAD[3]	SERCOM4/ PAD[3]	TC3/WO[1]	TCC0/ WO[5]		GCLK_IO[1]

PA14 EXTINT[14]

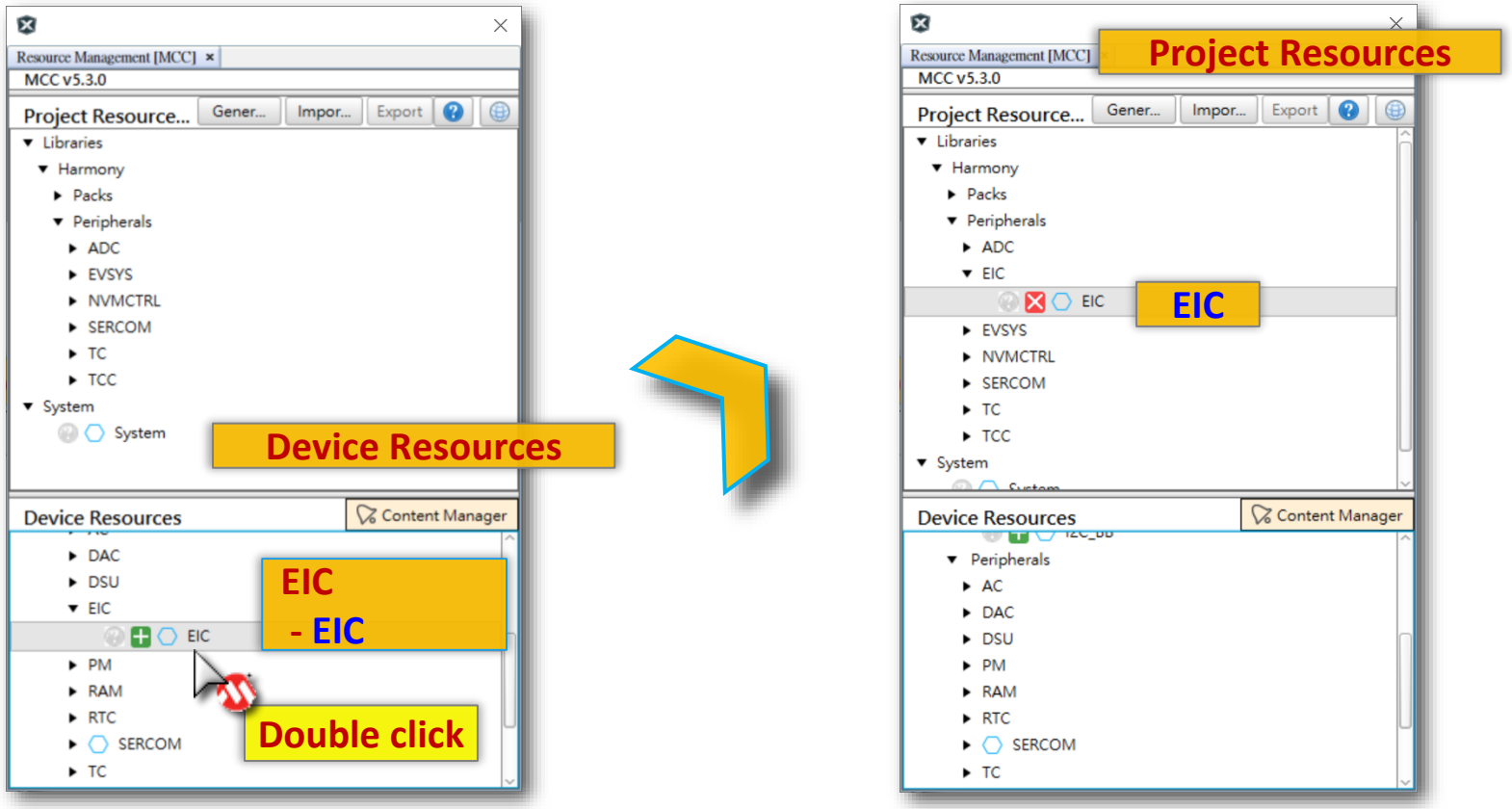
PA15 EXTINT[15]

MICROCHIP

🛠 Lab16-1 Button as External Interrupt

Step 1

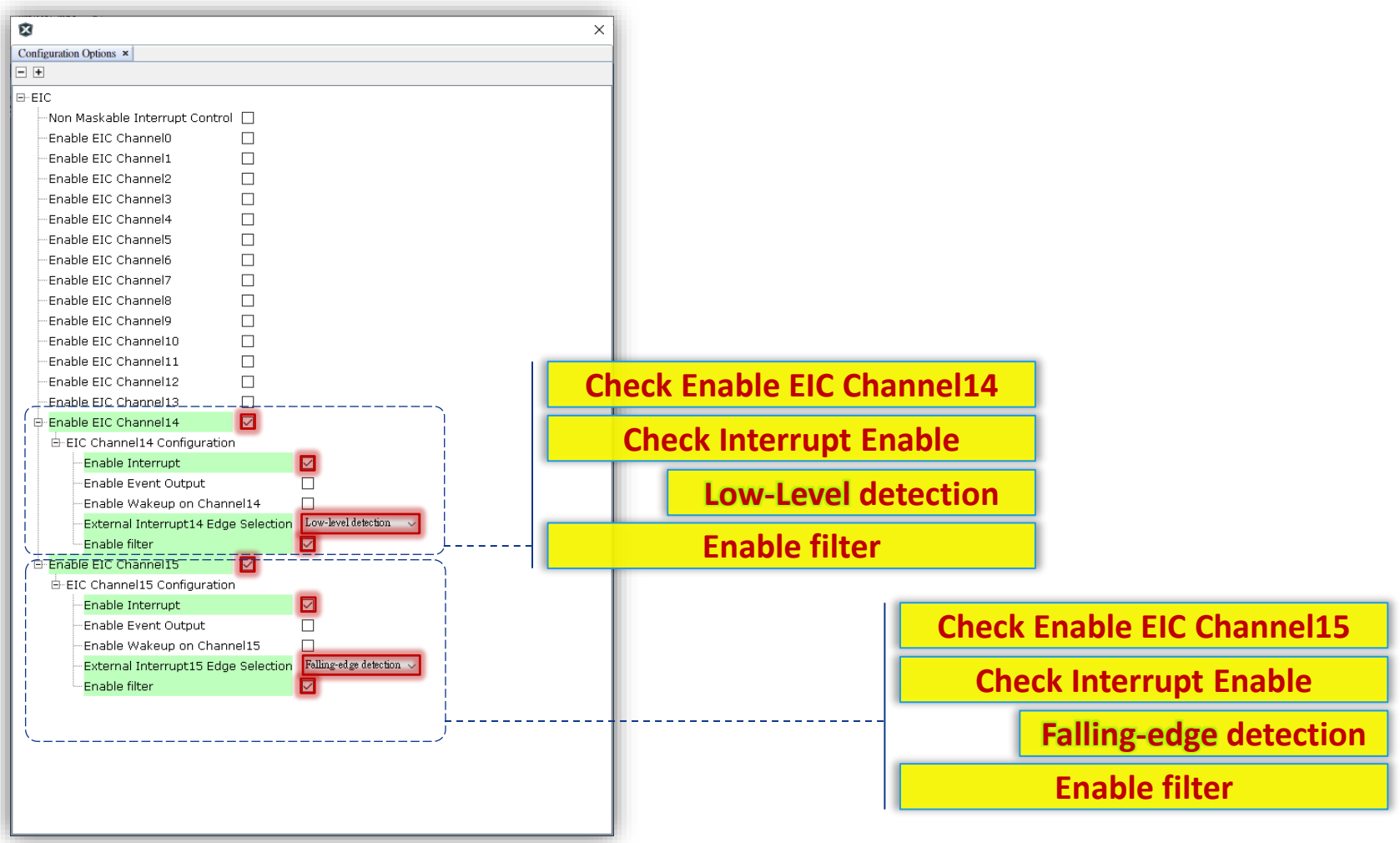
- Find EIC component in “Device Resources” window.
- Double click EIC to add to “Project Resources” window.



🛠 Lab16-1 Button as External Interrupt

Step 2

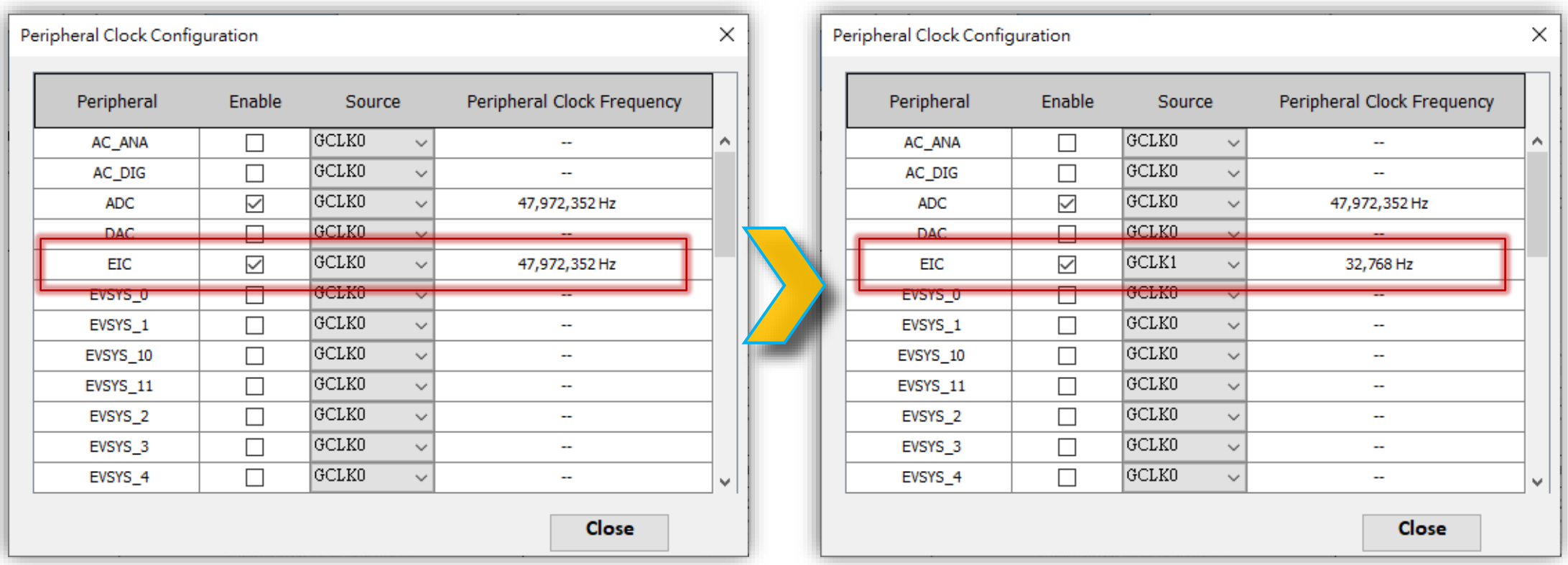
- **Configure EIC in configuration.**



🛠 Lab16-1 Button as External Interrupt

Step 3

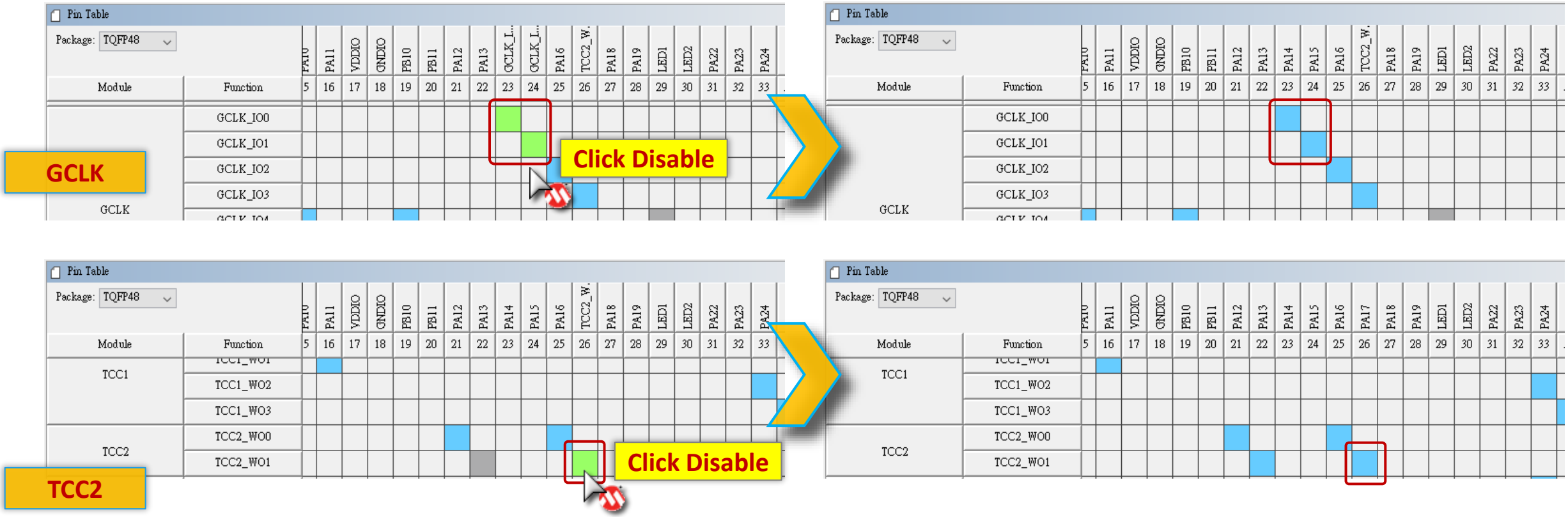
- **Configure EIC ASYNC clock to lower frequency in clock configuration.**



🛠 Lab16-1 Button as External Interrupt

Step 4

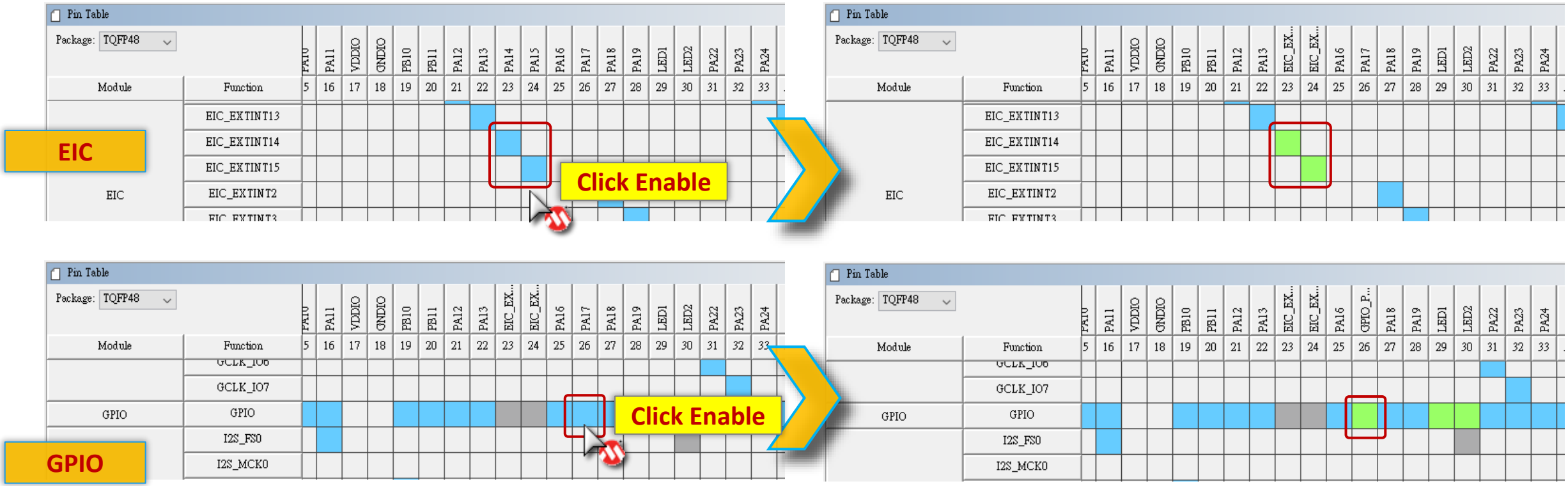
- **Disable clock and waveform output at PA14, PA15 and PA17.**



🛠 Lab16-1 Button as External Interrupt

Step 5

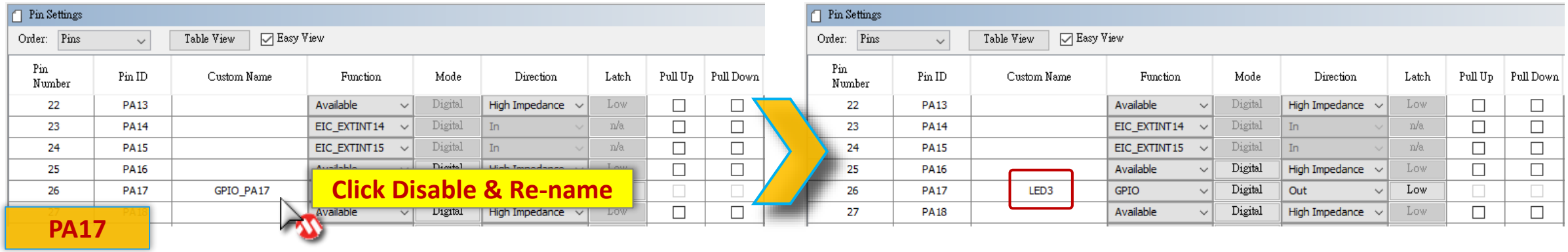
- **Enable EIC (EXTINT14, EXTINT15) and GPIO(LED3) at PA14, PA15 and PA17.**
  - EXTINT14 and EXTINT15 corresponding to BT1(PA14) and BT2(PA15).



🛠 Lab16-1 Button as External Interrupt

Step 6

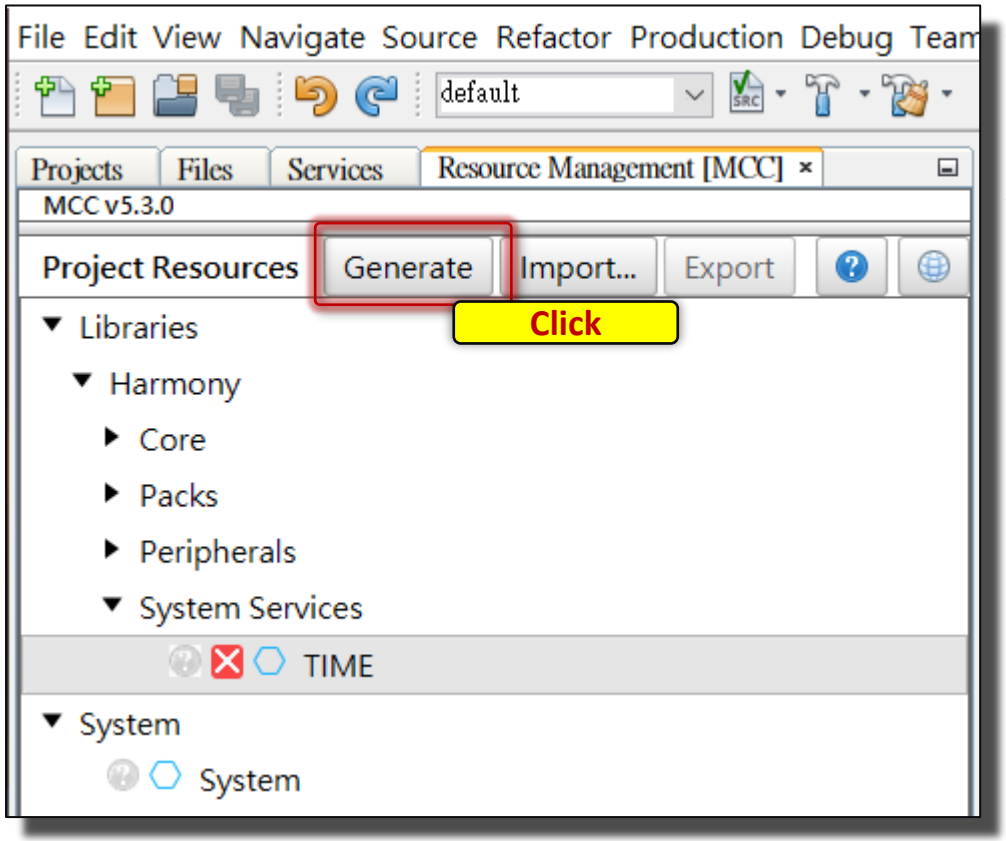
- **Assign customer name for PA17 -> LED3 at Pin Settings**

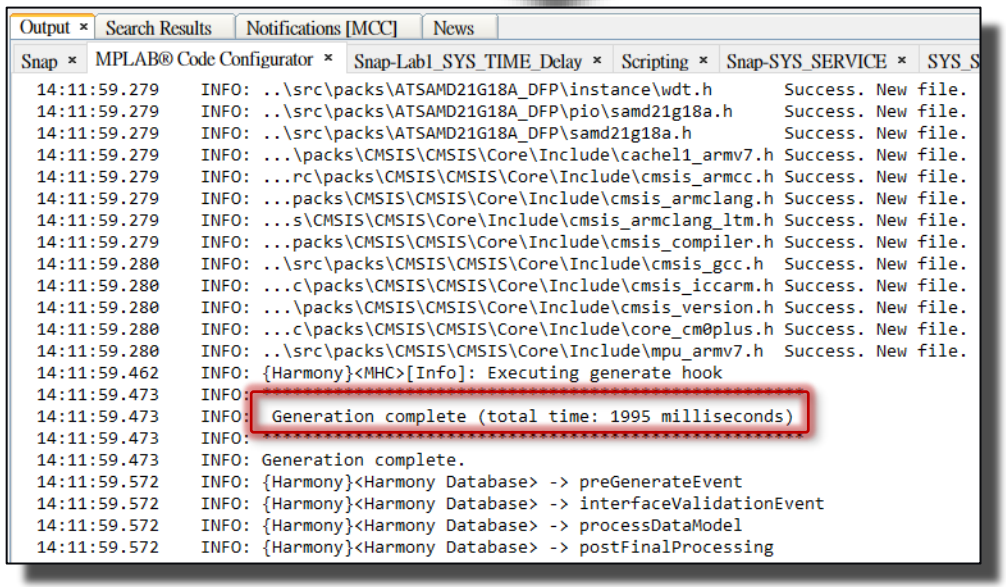


🛠 Lab16-1 Button as External Interrupt

Step 7

- **Click to Generate Code.**





🛠 Lab16-1 Button as External Interrupt

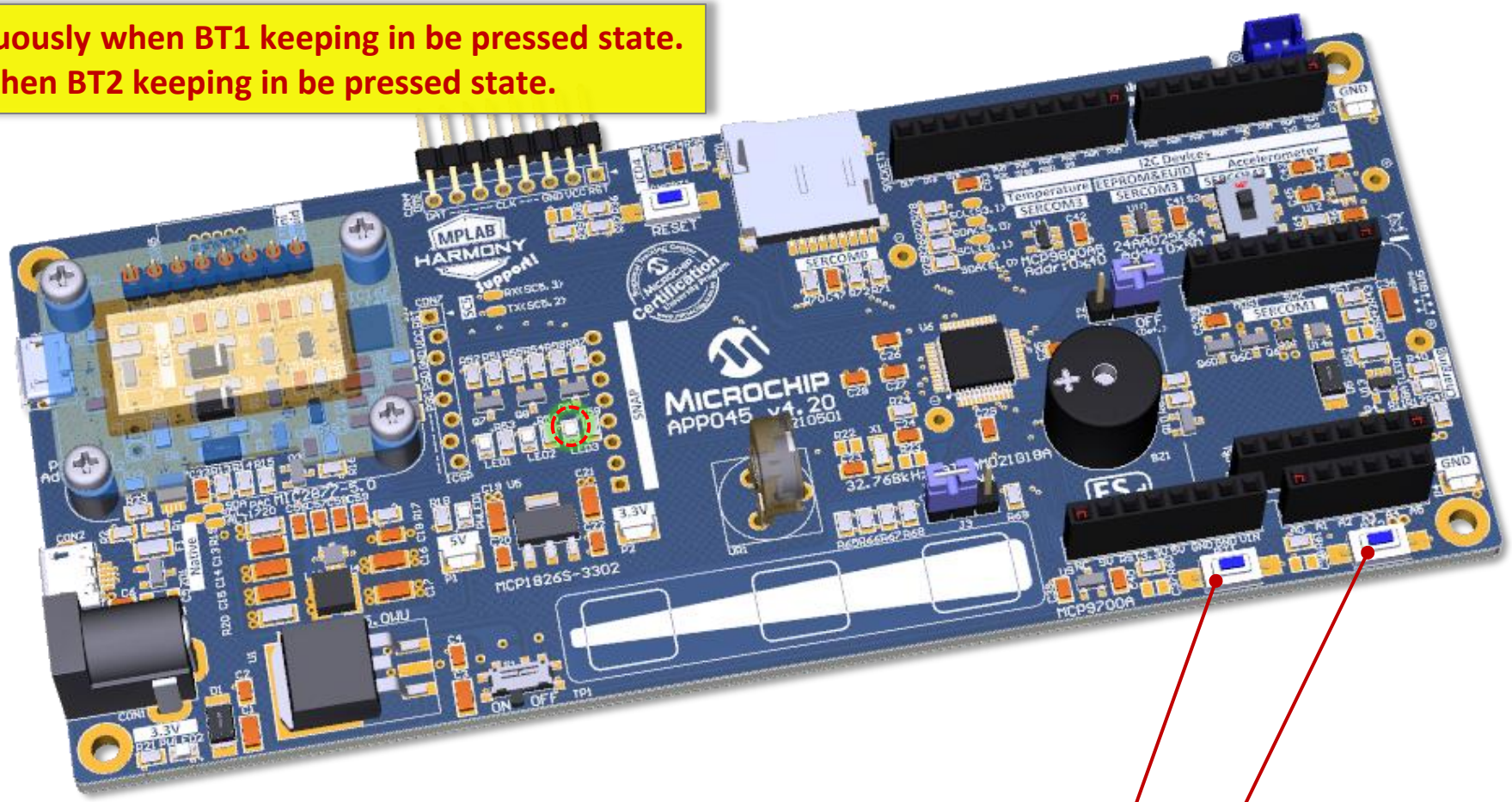
Step 8

```
...
// 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.



Event System (EVSYS)

Event System

- **The Event System (EVSYS) allows autonomous, low-latency and configurable communication between peripherals.**

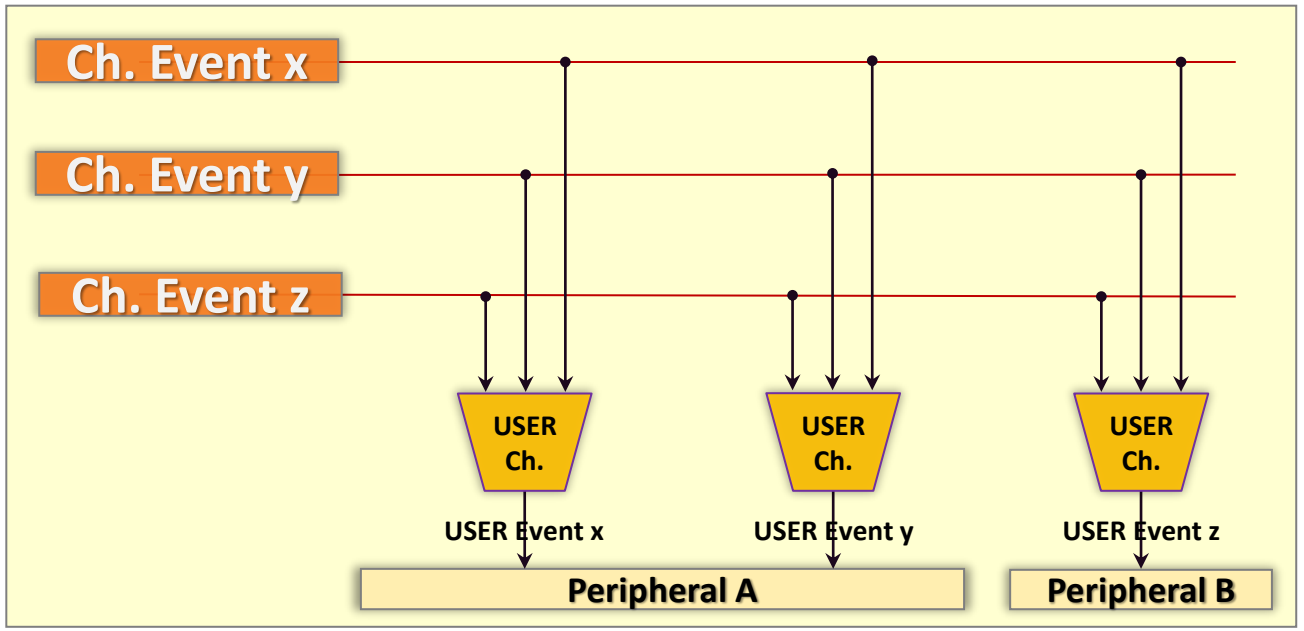
- **Role:**
  - Peripherals that **respond** to events are called event **users**.
  - Peripherals that **generate events** are called event **generators**.

- **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.**

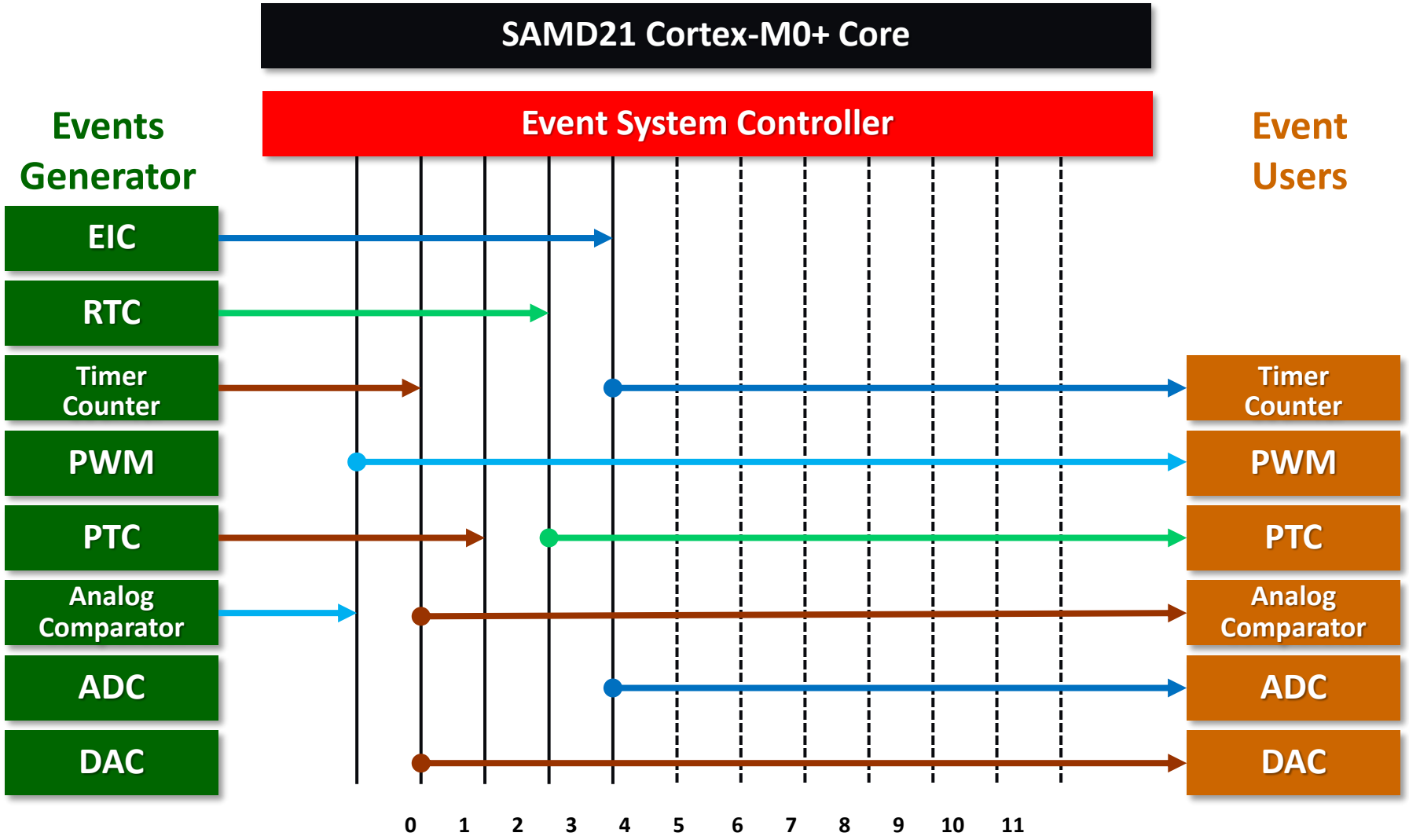
Event System

- **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



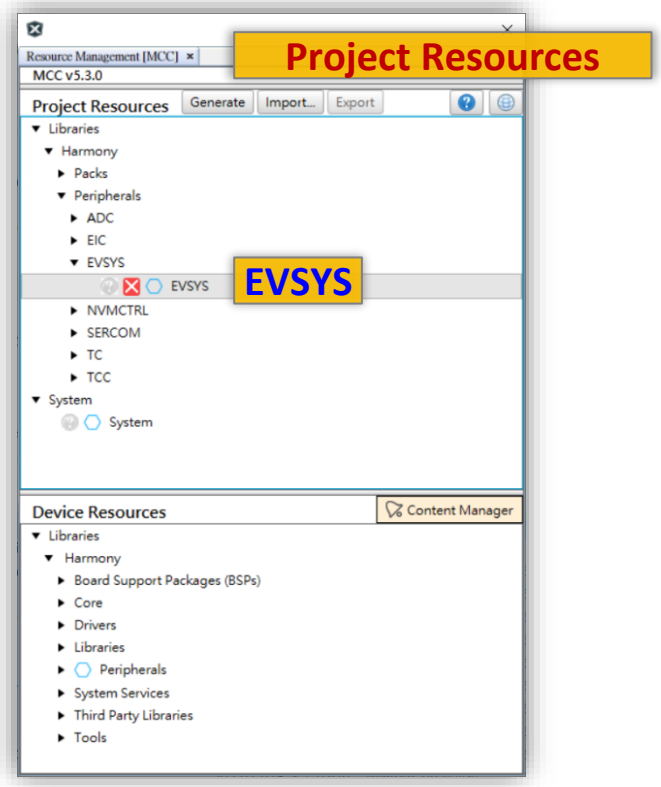
MICROCHIP

Event System Connection Demo



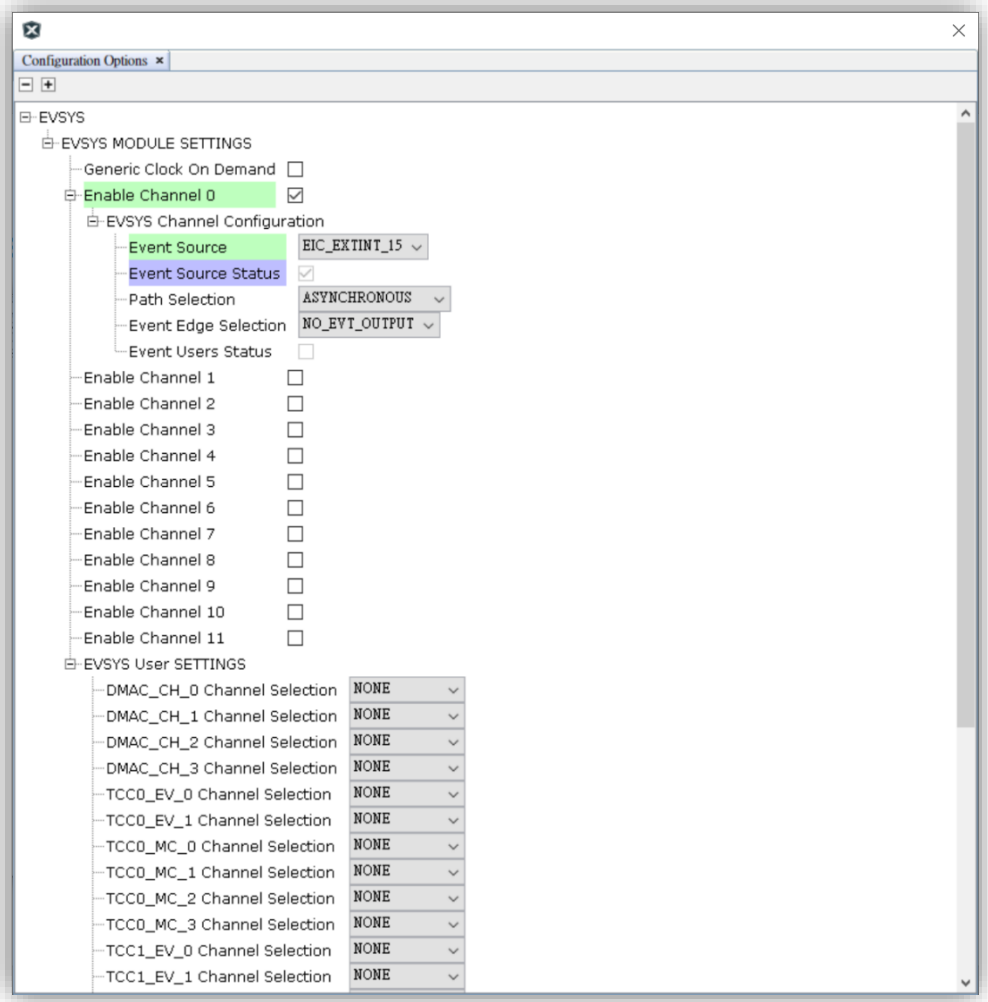
EVSYS Function Ready at MCC Harmony

- **EVSYS** already be active at new project create by “32-bit MCC Harmony Project”.



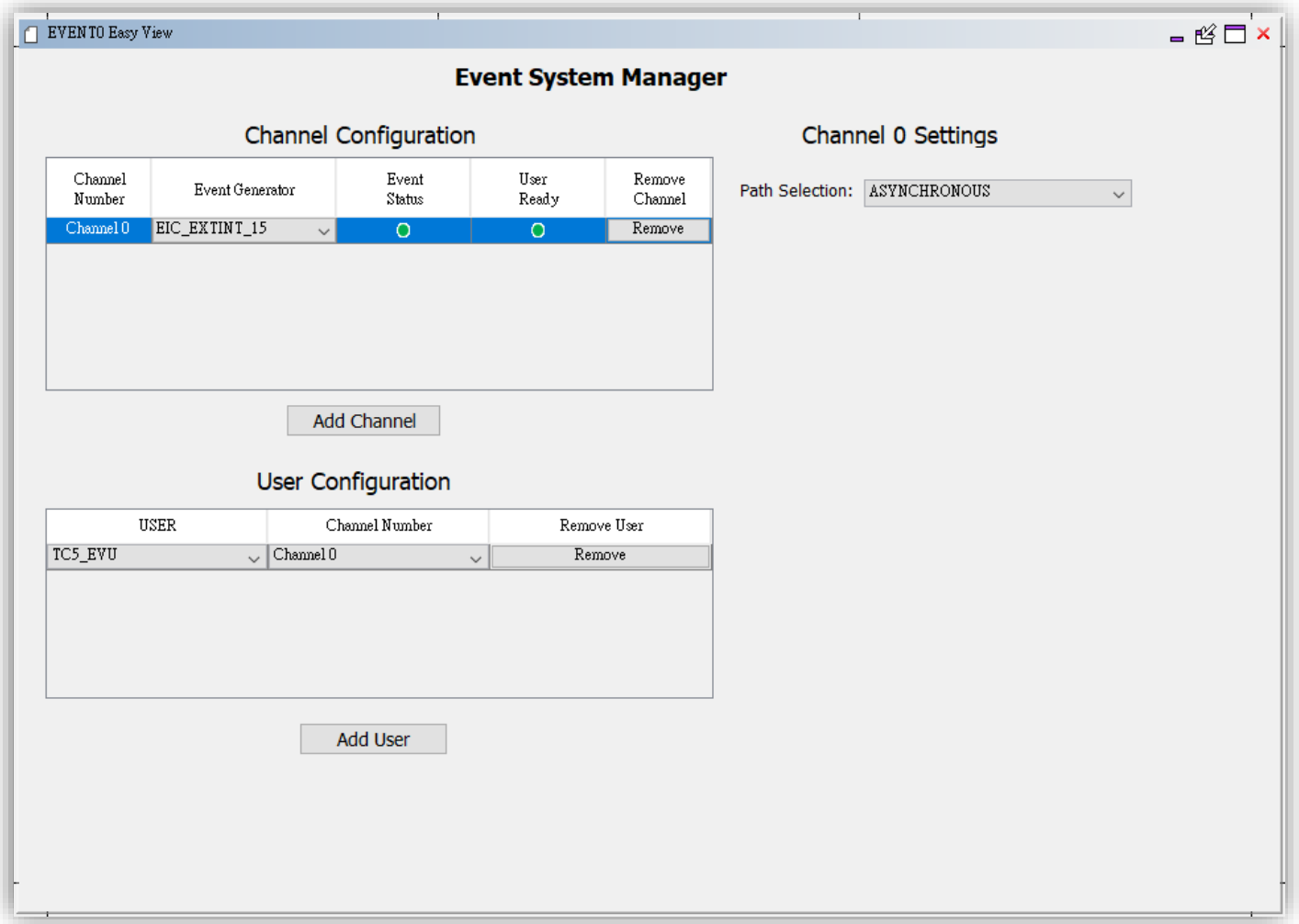
Event System

Configuration Options Example



Event System Manager

Configuration Options Example



MICROCHIP

EVSYS 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 );
```

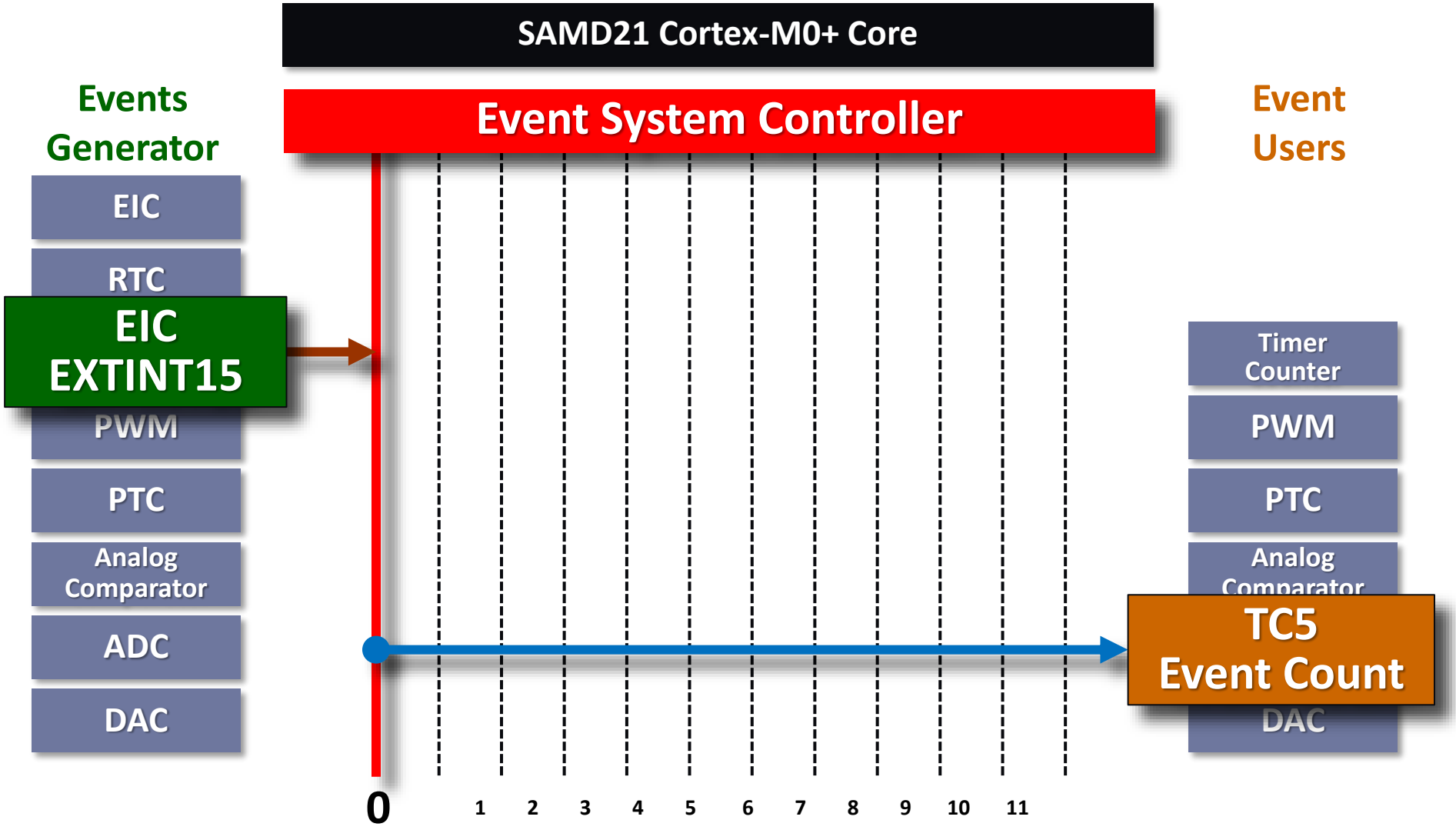
```
int main (void)
{
   SYS_Initialize ( NULL );
   ...
   /* EVSYS Fully automatic !! Not thing to do in user application */

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

MICROCHIP

🛠 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 !**

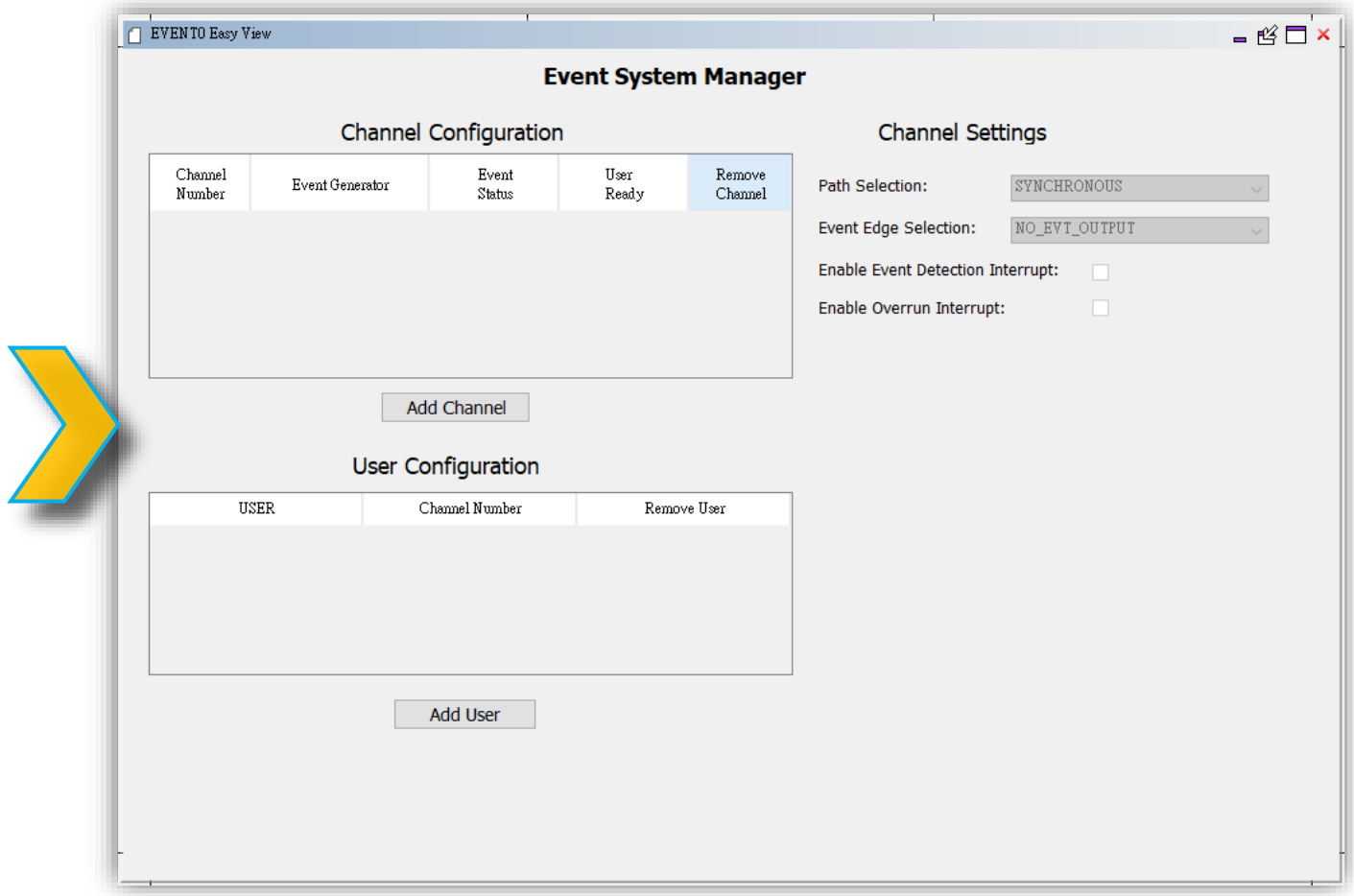
Lab16-2 EXTINT Trigger Timer Counter



🛠 Lab16-2 EXTINT Trigger Timer Counter

Step 1

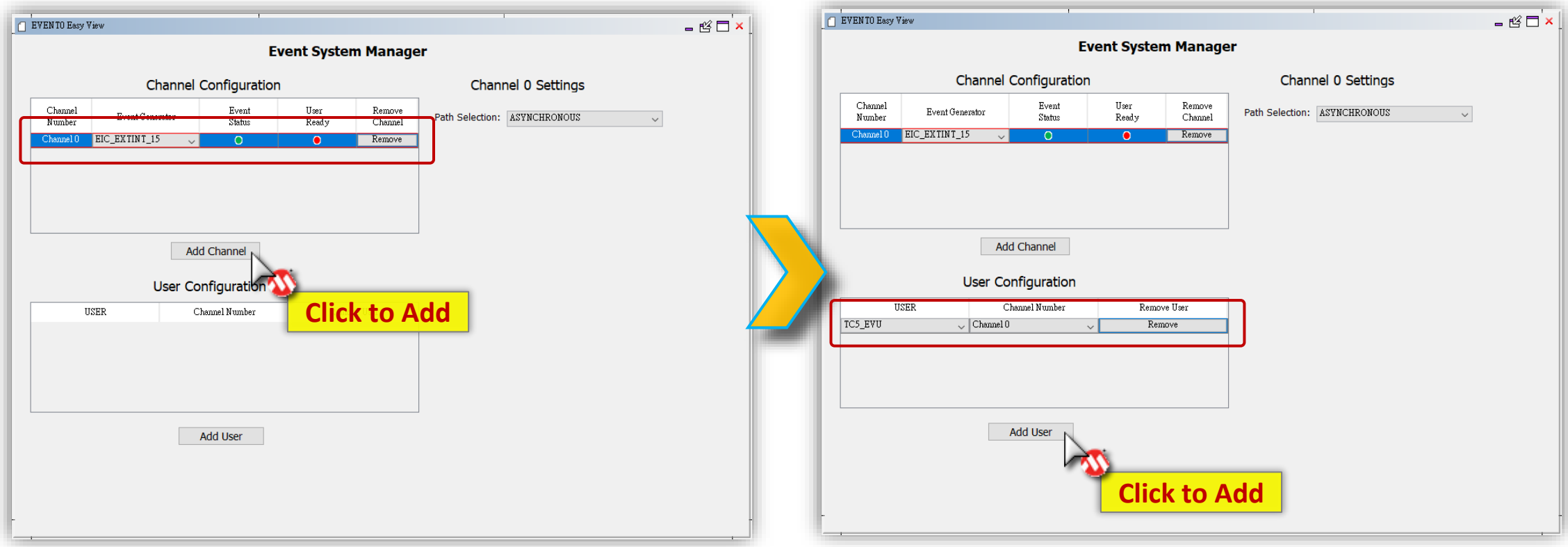
- **Configure EVSYS in Event System Manager.**



🛠 Lab16-2 EXTINT Trigger Timer Counter

Step 2

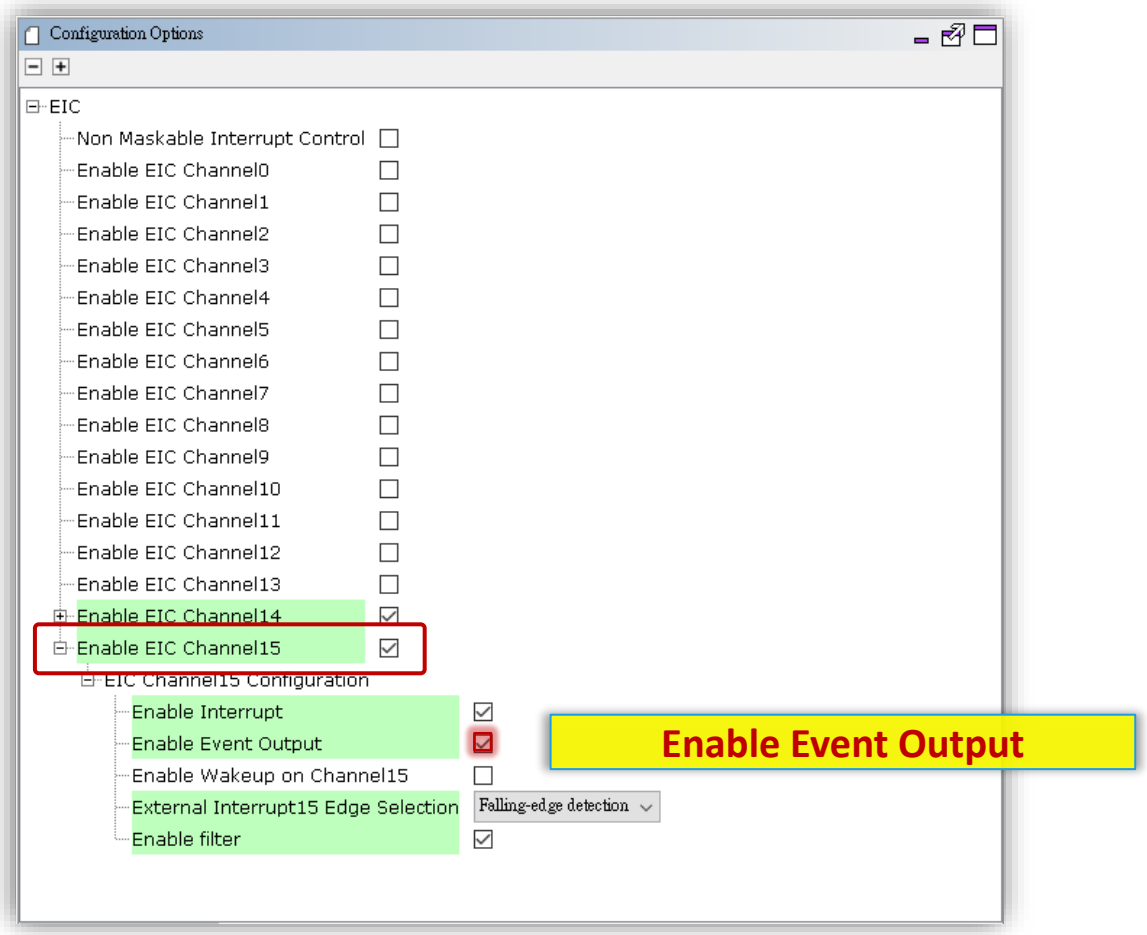
- add **Event Generator** : EIC_EXTINT_15 , **Event User** : TC5_EVU



🛠 Lab16-2 EXTINT Trigger Timer Counter

Step 3

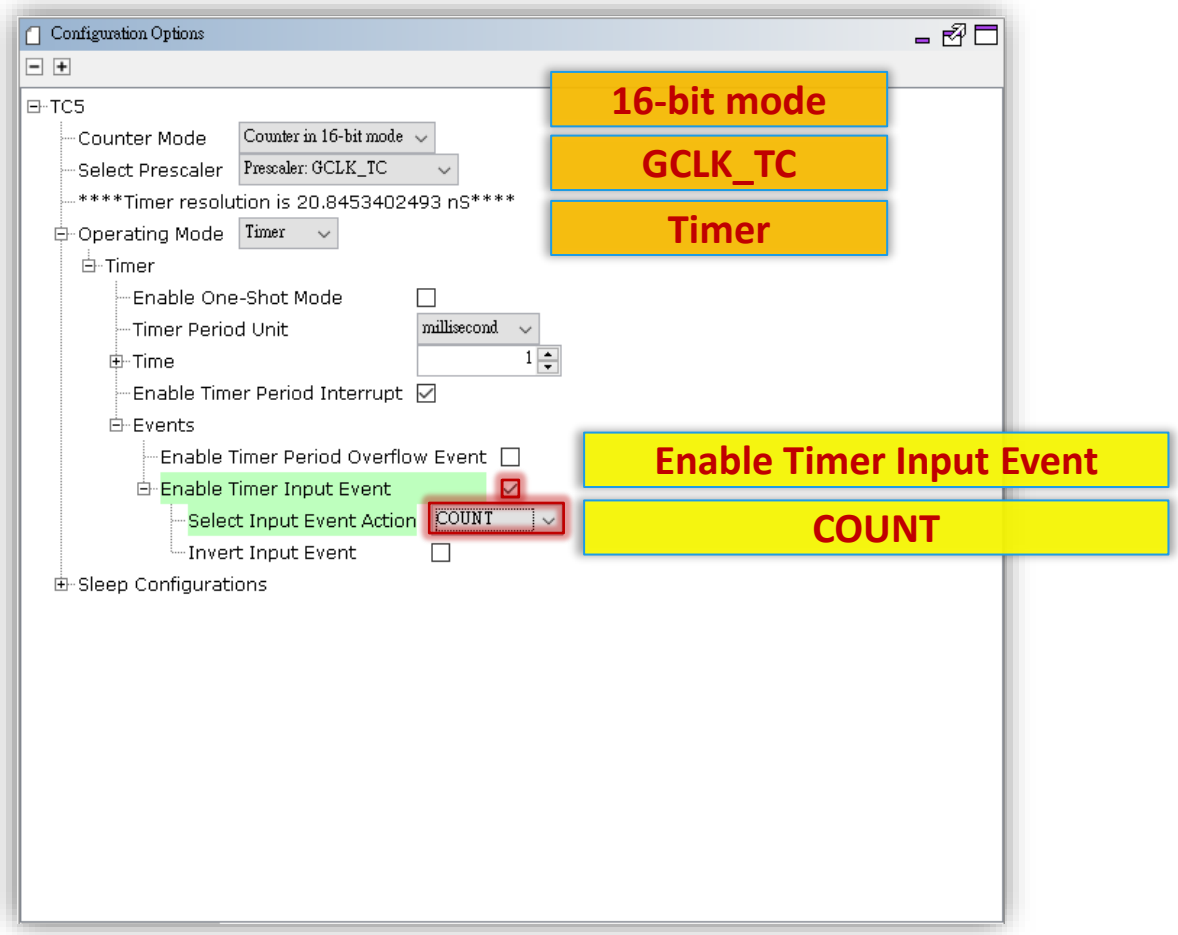
- **Enable Event be generated at EIC (EXTINT15).**



🛠 Lab16-2 EXTINT Trigger Timer Counter

Step 4

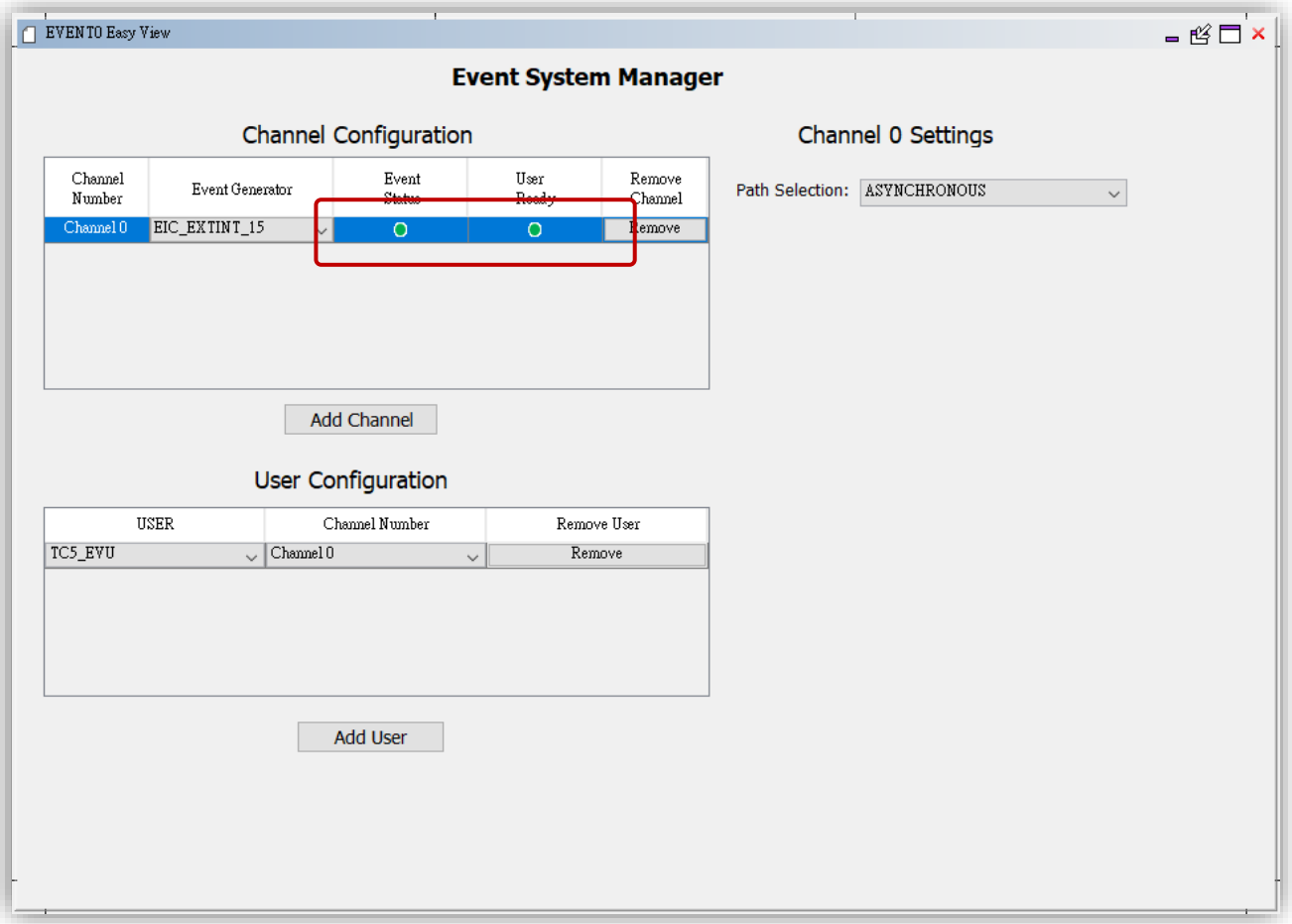
- **Add TC5 and set to Event Counter Mode**



🛠 Lab16-2 EXTINT Trigger Timer Counter

Step 5

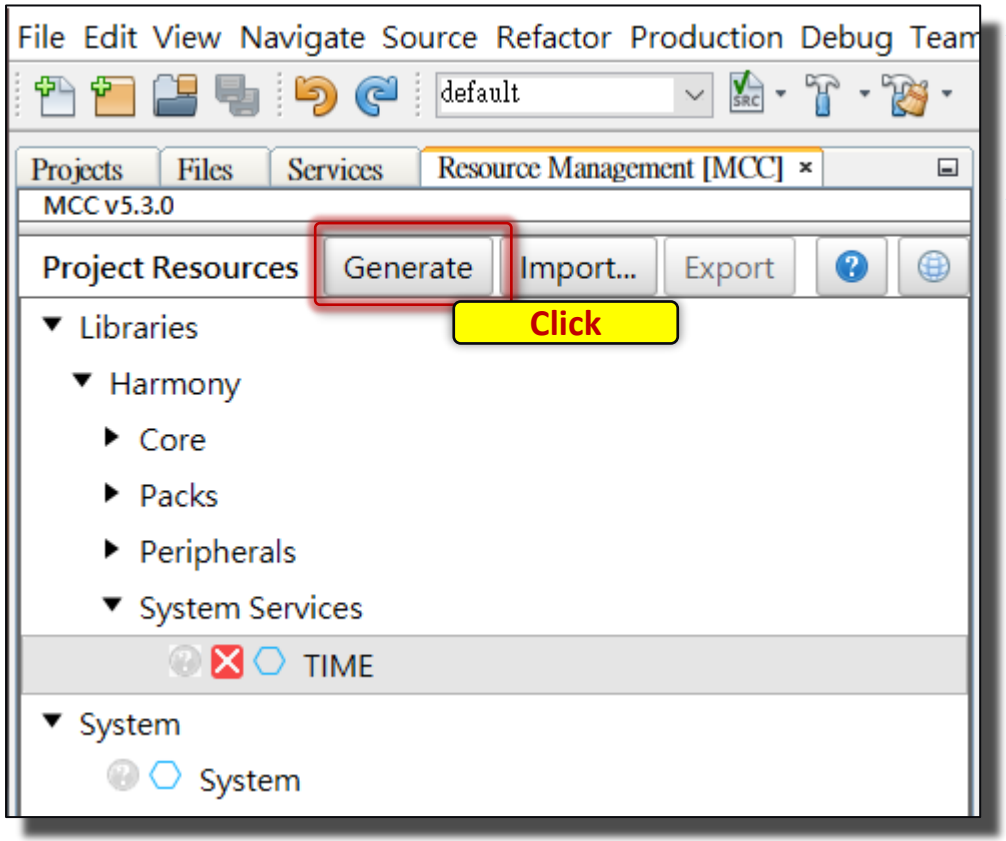
- **Confirm Event Status and User Ready is Green light.**

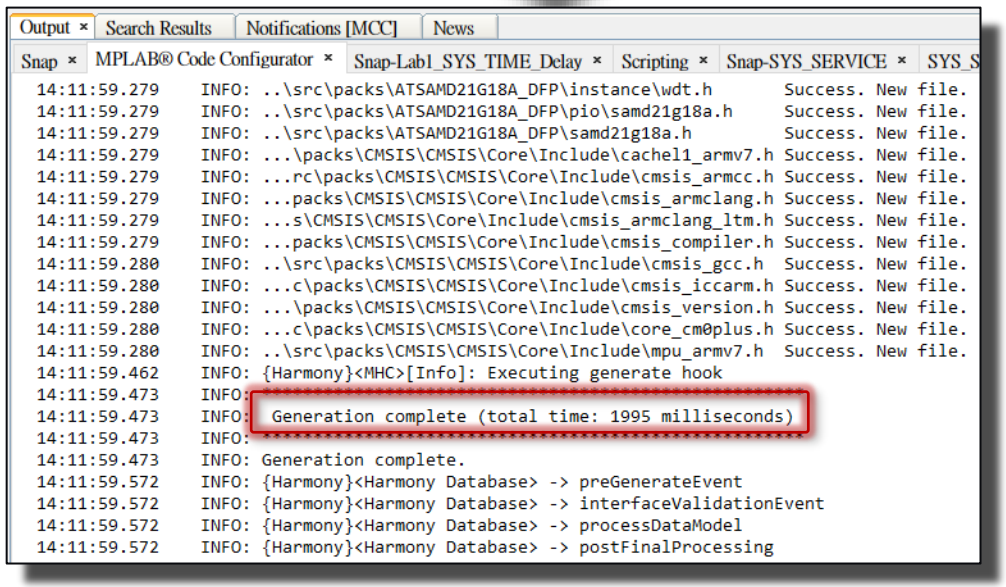


🛠 Lab16-2 EXTINT Trigger Timer Counter

Step 7

- **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

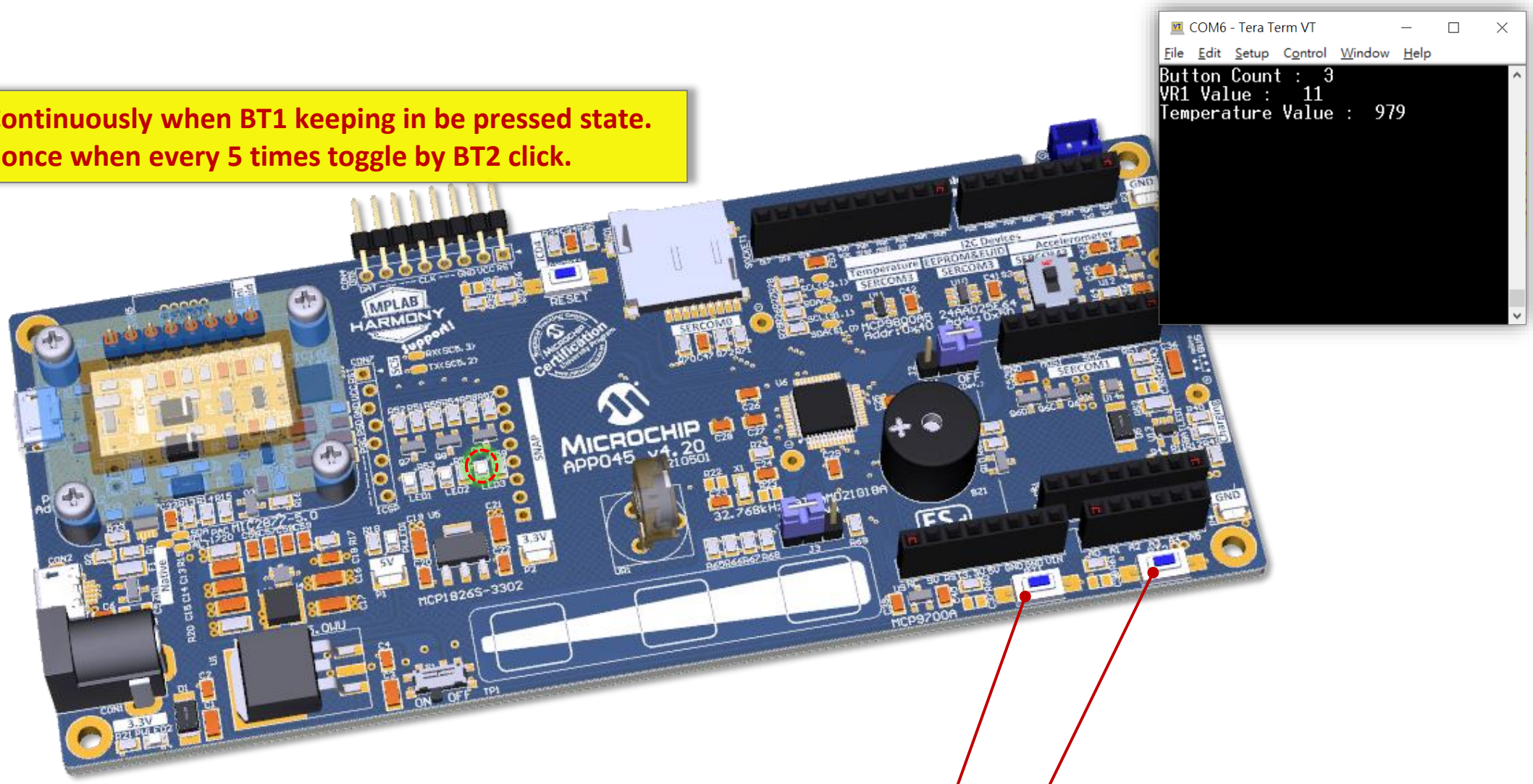
Step 8

```
...
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.

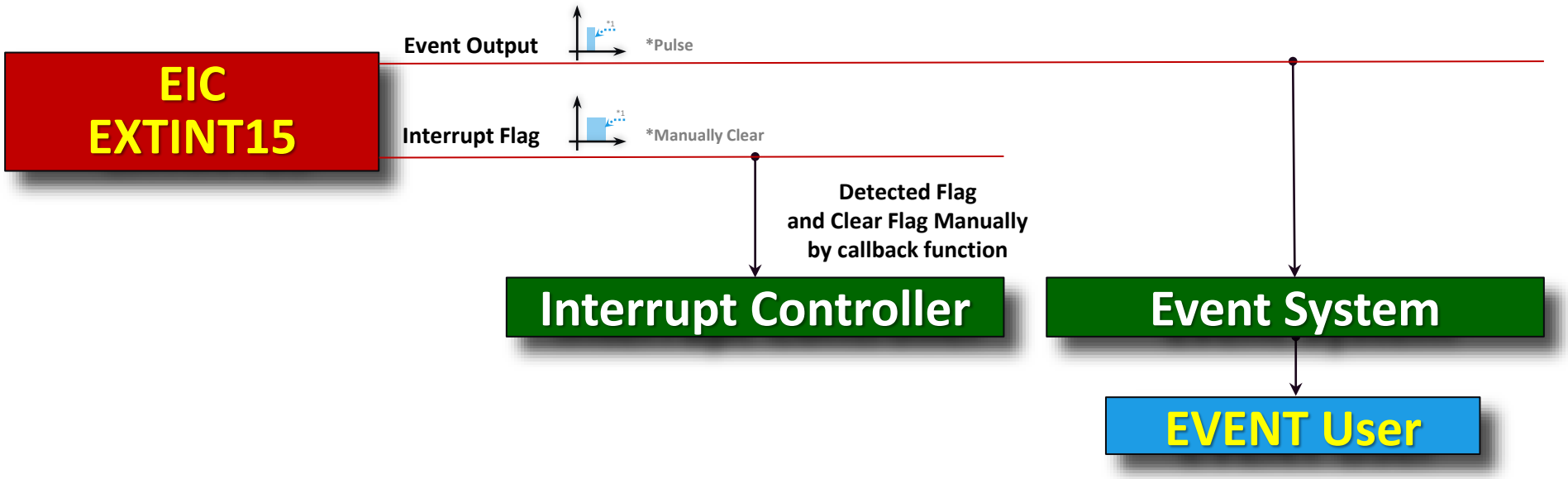


🛠 Lab16-2 EXTINT Trigger Timer Counter

- **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.



MICROCHIP

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

- **Let's go !**

**Reference*

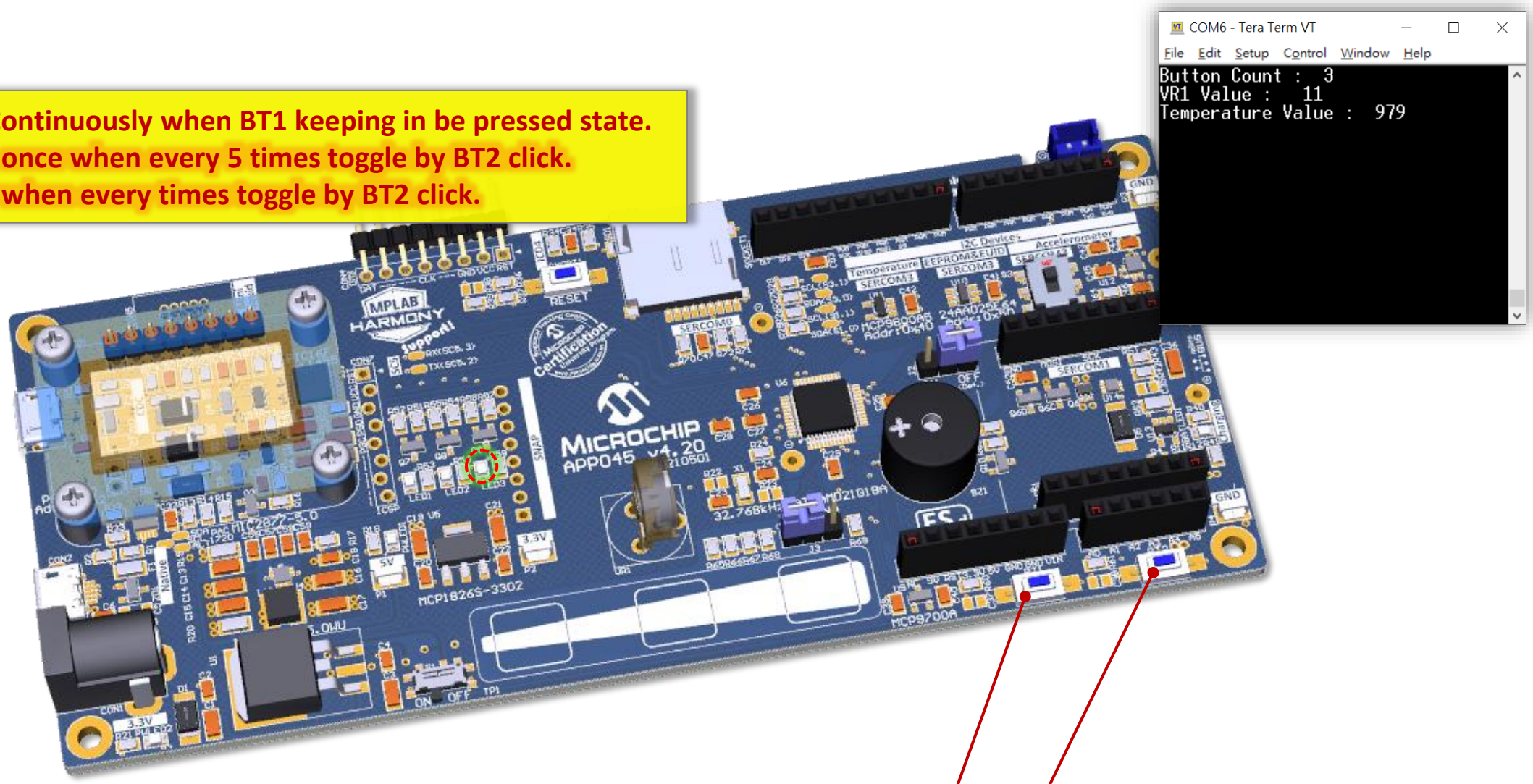
```
void LED_Control(uintptr_t context)
{
    static uint32_t i = 0;
    LED3_Toggle();
    for (i = 0; i < 1000000; i++);
}
```

```
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.



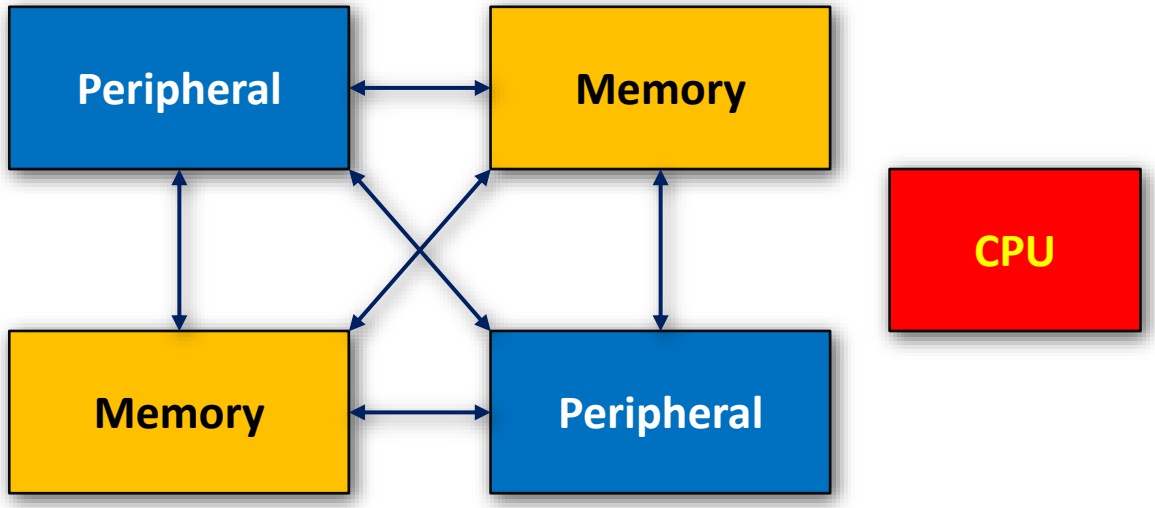
MICROCHIP

Direct Memory Access Controller (DMAC) - Single Transfer Block

MICROCHIP

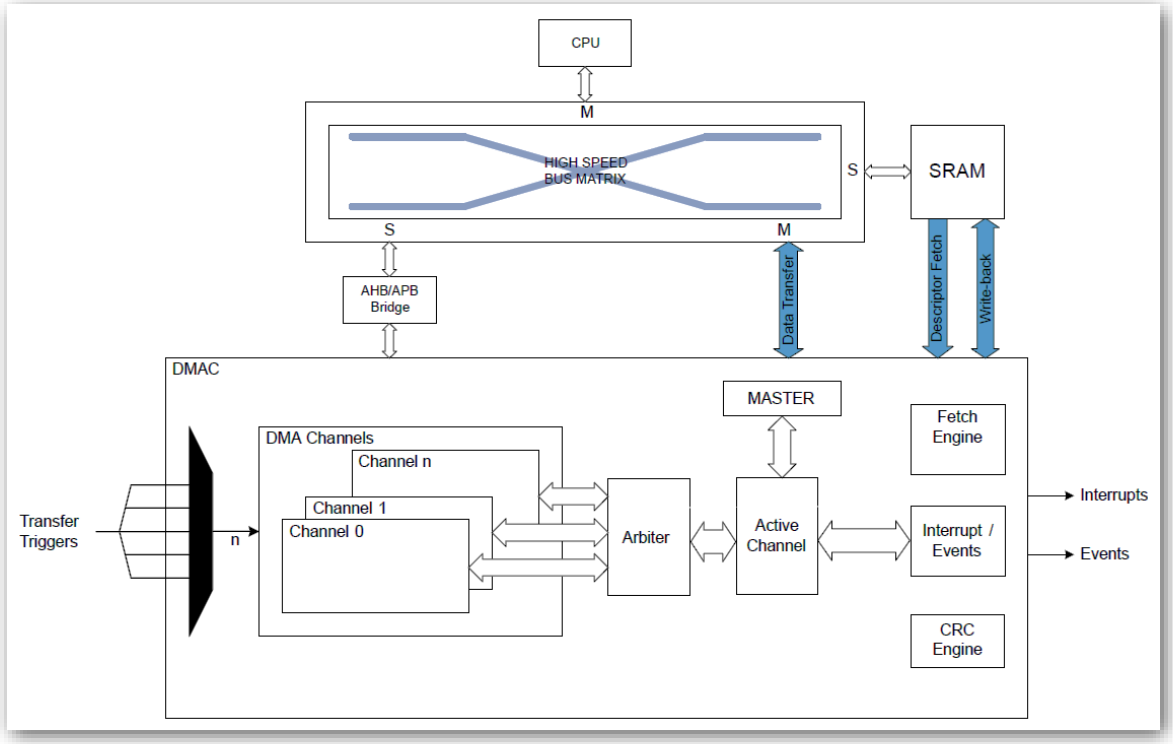
Cortex M0+ core's DMAC

- **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



Cortex M0+ core's 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)

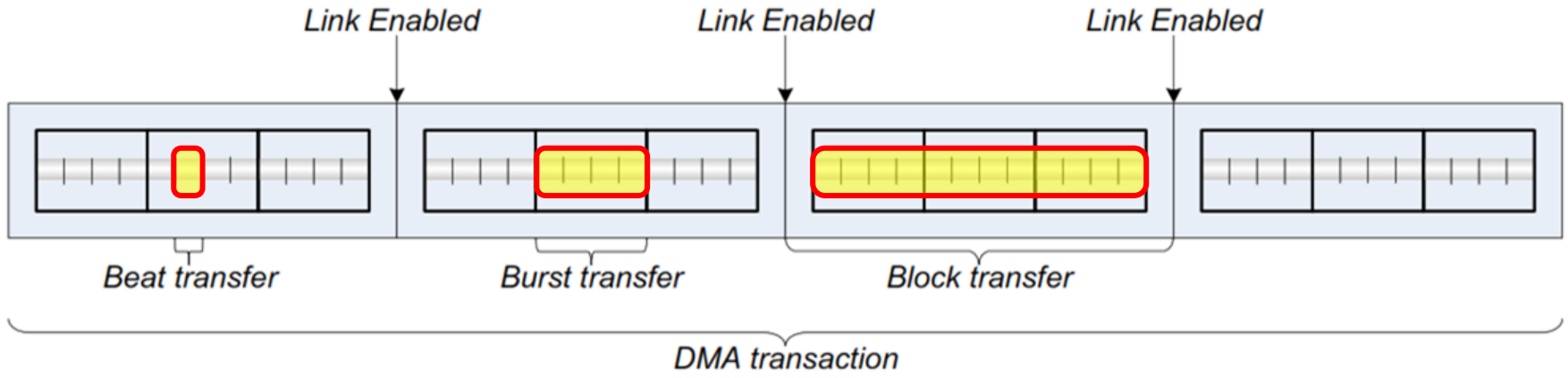


Transfer Types and Transfer Descriptor

MICROCHIP

Transfer Types

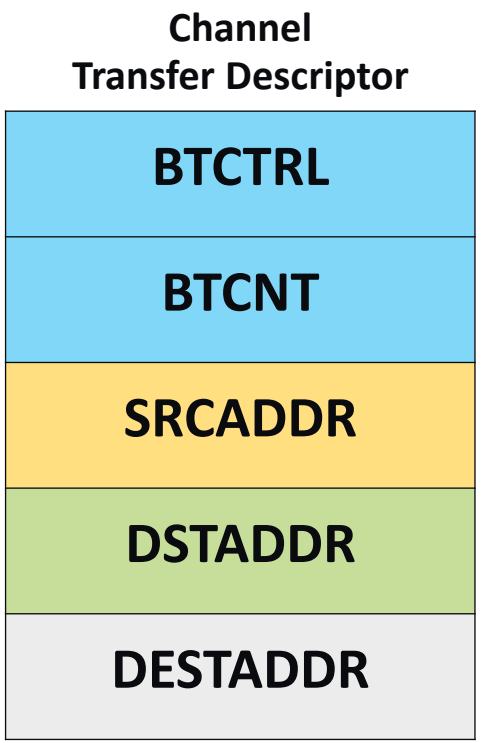
- **Beat transfer**
  - Size of data length for one of DMAC transfer.
  - 8-bits, 16-bits or 32-bits
- **Burst transfer**
  - n Beat transfer at same time, Burst is "**atomic**" transfer.
  - SAM M0+ families not support.
- **Block transfer**
  - Amount of beat can be transfer at one transfer descriptor.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



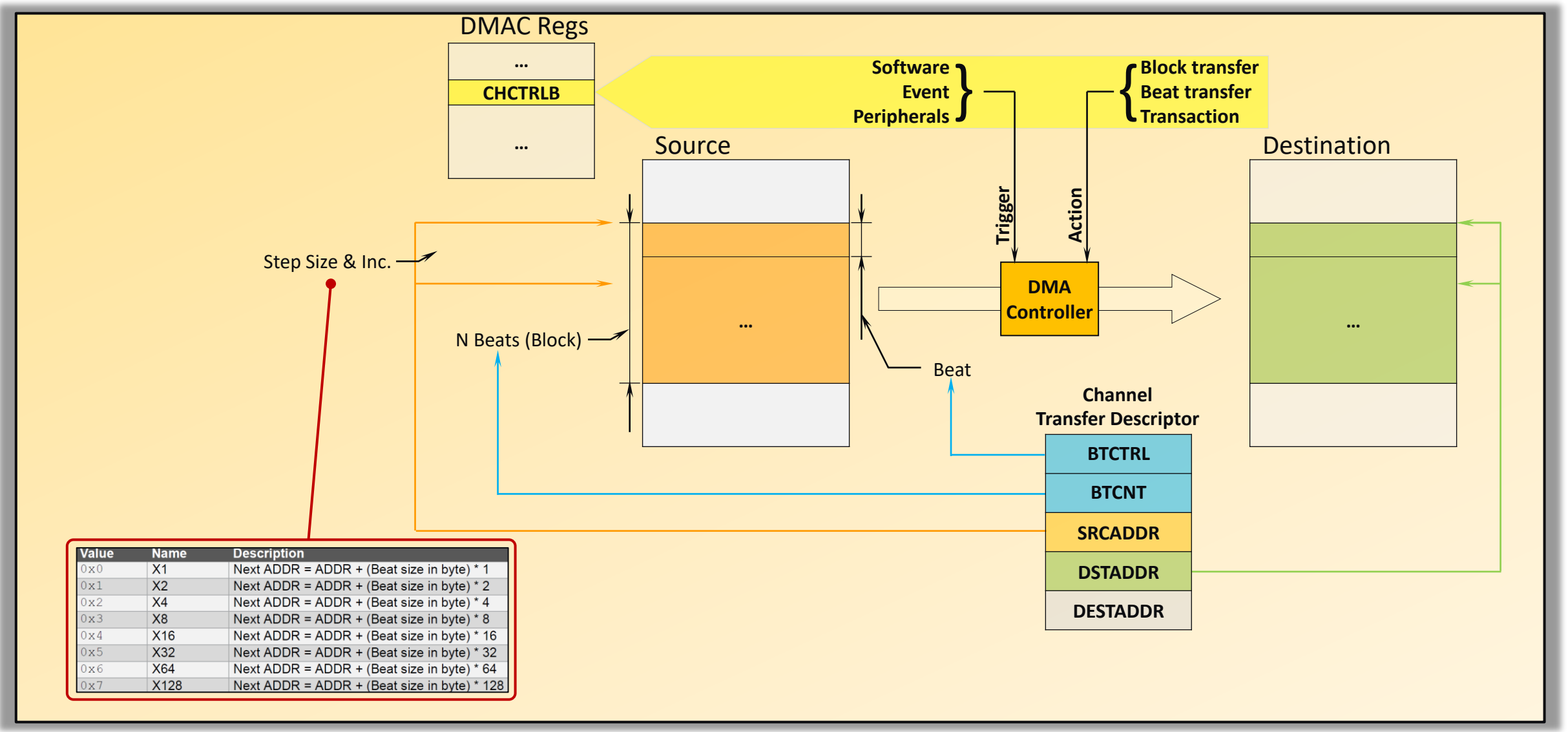
MICROCHIP

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



Block diagram of Transaction



Value	Name	Description
0x0	X1	Next ADDR = ADDR + (Beat size in byte) * 1
0x1	X2	Next ADDR = ADDR + (Beat size in byte) * 2
0x2	X4	Next ADDR = ADDR + (Beat size in byte) * 4
0x3	X8	Next ADDR = ADDR + (Beat size in byte) * 8
0x4	X16	Next ADDR = ADDR + (Beat size in byte) * 16
0x5	X32	Next ADDR = ADDR + (Beat size in byte) * 32
0x6	X64	Next ADDR = ADDR + (Beat size in byte) * 64
0x7	X128	Next ADDR = ADDR + (Beat size in byte) * 128

Trigger Sources

DMAC Trigger By Event or Dedicated Sources

Value	Name	Description
0x00	DISABLE	Only software/event triggers
0x01	SERCOM0 RX	SERCOM0 RX Trigger
0x02	SERCOM0 TX	SERCOM0 TX Trigger
0x03	SERCOM1 RX	SERCOM1 RX Trigger
0x04	SERCOM1 TX	SERCOM1 TX Trigger
0x05	SERCOM2 RX	SERCOM2 RX Trigger
0x06	SERCOM2 TX	SERCOM2 TX Trigger
0x07	SERCOM3 RX	SERCOM3 RX Trigger
0x08	SERCOM3 TX	SERCOM3 TX Trigger
0x09	SERCOM4 RX	SERCOM4 RX Trigger
0x0A	SERCOM4 TX	SERCOM4 TX Trigger
0x0B	SERCOM5 RX	SERCOM5 RX Trigger
0x0C	SERCOM5 TX	SERCOM5 TX Trigger
0x0D	TCC0 OVF	TCC0 Overflow Trigger
0x0E	TCC0 MC0	TCC0 Match/Compare 0 Trigger
0x0F	TCC0 MC1	TCC0 Match/Compare 1 Trigger
0x10	TCC0 MC2	TCC0 Match/Compare 2 Trigger
0x11	TCC0 MC3	TCC0 Match/Compare 3 Trigger
0x12	TCC1 OVF	TCC1 Overflow Trigger
0x13	TCC1 MC0	TCC1 Match/Compare 0 Trigger
0x14	TCC1 MC1	TCC1 Match/Compare 1 Trigger
		erflow Trigger
		ch/Compare 0 Trigger
		ch/Compare 1 Trigger
		low Trigger

Value	Name	Description
0x19	TC3 MC0	TC3 Match/Compare 0 Trigger
0x1A	TC3 MC1	TC3 Match/Compare 1 Trigger
0x1B	TC4 OVF	TC4 Overflow Trigger
0x1C	TC4 MC0	TC4 Match/Compare 0 Trigger
0x1D	TC4 MC1	TC4 Match/Compare 1 Trigger
0x1E	TC5 OVF	TC5 Overflow Trigger
0x1F	TC5 MC0	TC5 Match/Compare 0 Trigger
0x20	TC5 MC1	TC5 Match/Compare 1 Trigger
0x21	TC6 OVF	TC6 Overflow Trigger
0x22	TC6 MC0	TC6 Match/Compare 0 Trigger
0x23	TC6 MC1	TC6 Match/Compare 1 Trigger
0x24	TC7 OVF	TC7 Overflow Trigger
0x25	TC7 MC0	TC7 Match/Compare 0 Trigger
0x26	TC7 MC1	TC7 Match/Compare 1 Trigger
0x27	ADC RESRDY	ADC Result Ready Trigger
0x28	DAC EMPTY	DAC Empty Trigger
0x29	I2S RX 0	I2S RX 0 Trigger
0x2A	I2S RX 1	I2S RX 1 Trigger
0x2B	I2S TX 0	I2S TX 0 Trigger
0x2C	I2S TX 0	I2S TX 1 Trigger
0x2D	OVF	TCC3 Overflow Trigger
0x2E	TCC3 MC0	TCC3 Match/Compare 0 Trigger
0x2F	TCC3 MC1	TCC3 Match/Compare 1 Trigger
0x30	TCC3 MC2	Match/Compare 2 Trigger
0x31	TCC3 MC3	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

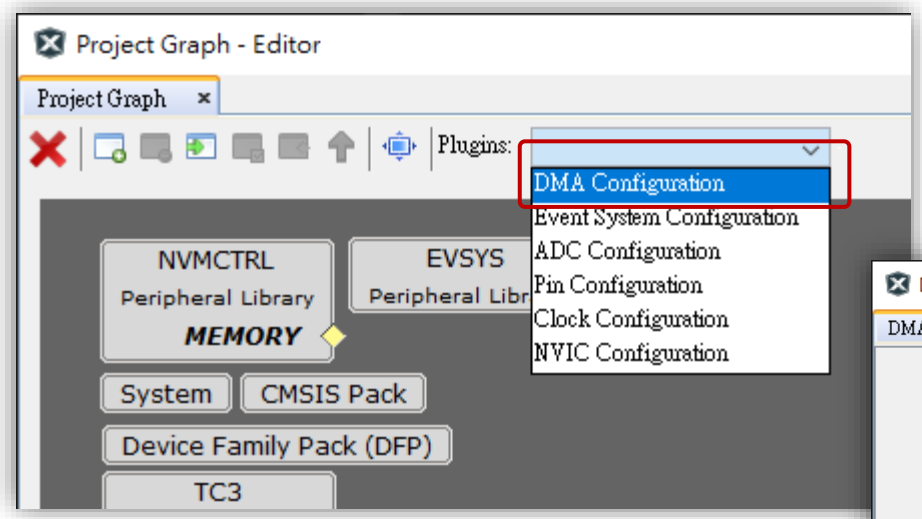
MICROCHIP

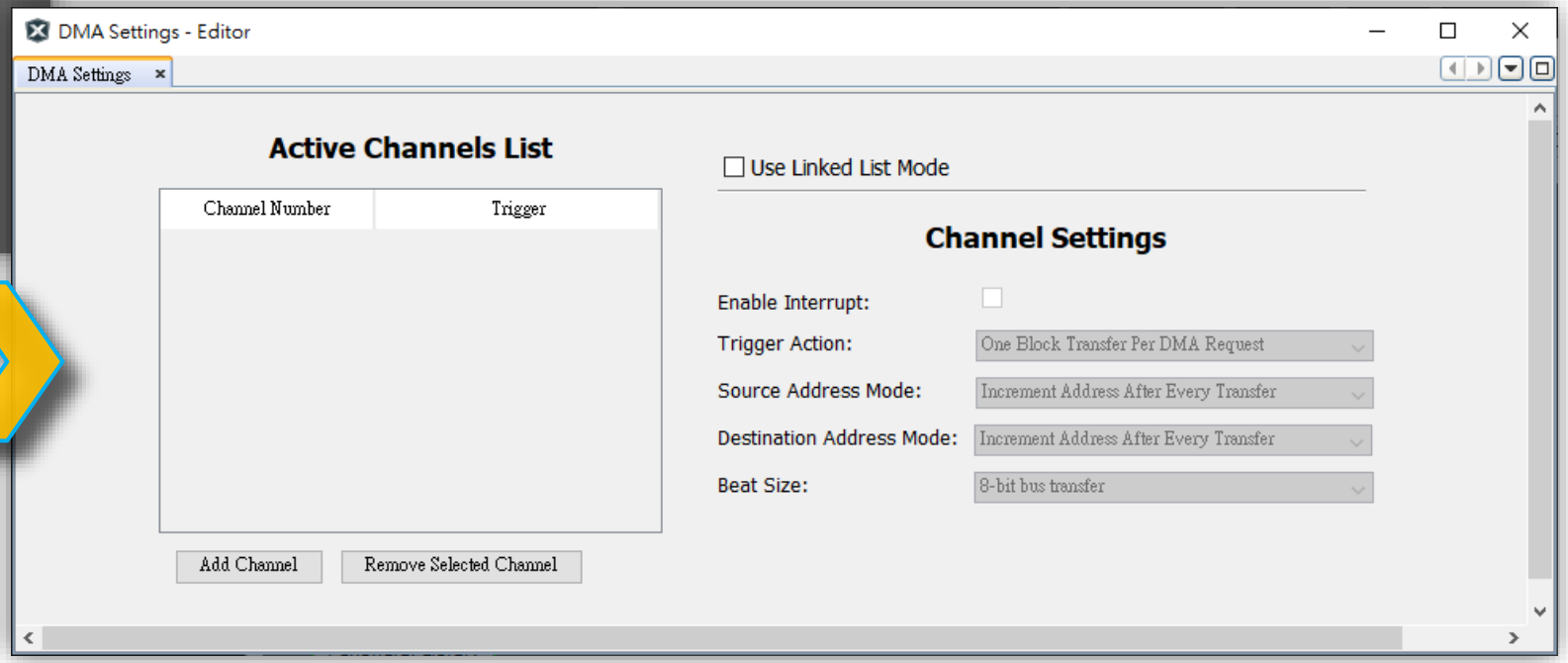
Function Setting at MCC-Harmony, APIs and Code Example

MICROCHIP

DMAC Function at MCC-Harmony

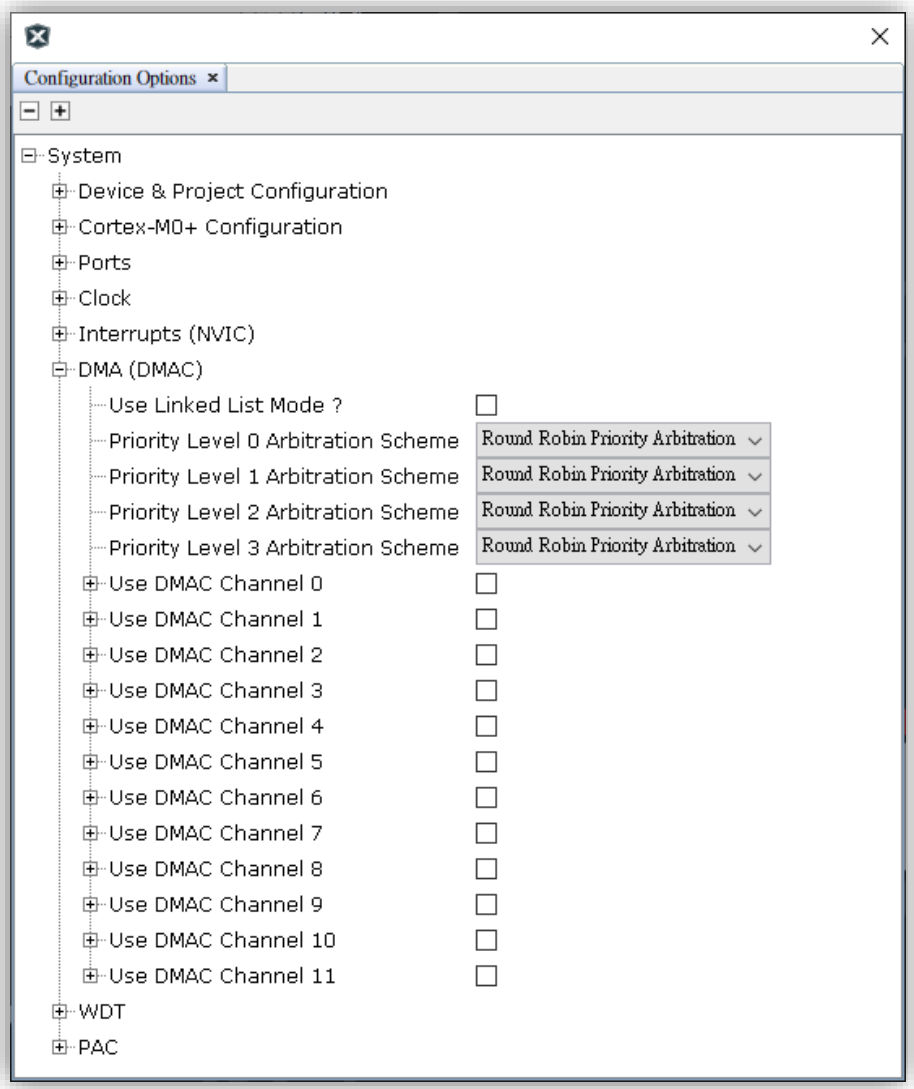
- **DMAC** already be active at new project created.
- Configure **DMAC** in DMA Settings.





DMAC Function at MCC-Harmony

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_ChannelIsBusy( 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 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_ChannelIsBusy( 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 )
```

MICROCHIP

DMAC Code Example

Code Example

```
void DMA_TransferCompleted (DMAC_TRANSFER_EVENT event, uintptr_t contextHandle)
{
   if (event == DMAC_TRANSFER_EVENT_COMPLETE)
   {
      DMA_TransferCompleted= 1;
      DMAC_ChannelTransfer(DMAC_CHANNEL_0,
                  (const void *) &(ADC_REGS->ADC_RESULT),
                  (const void *) ADC_Result[0], sizeof (ADC_Result));
   }
}

int main (void)
{
   SYS_Initialize ( NULL );
   ...
   DMAC_ChannelCallbackRegister(DMAC_CHANNEL_0, DMA_TransferCompleted, (uintptr_t) NULL);
   DMAC_ChannelTransfer(DMAC_CHANNEL_0,
               (const void *) &(ADC_REGS->ADC_RESULT),
               (const void *) ADC_Result[0], 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)

MICROCHIP

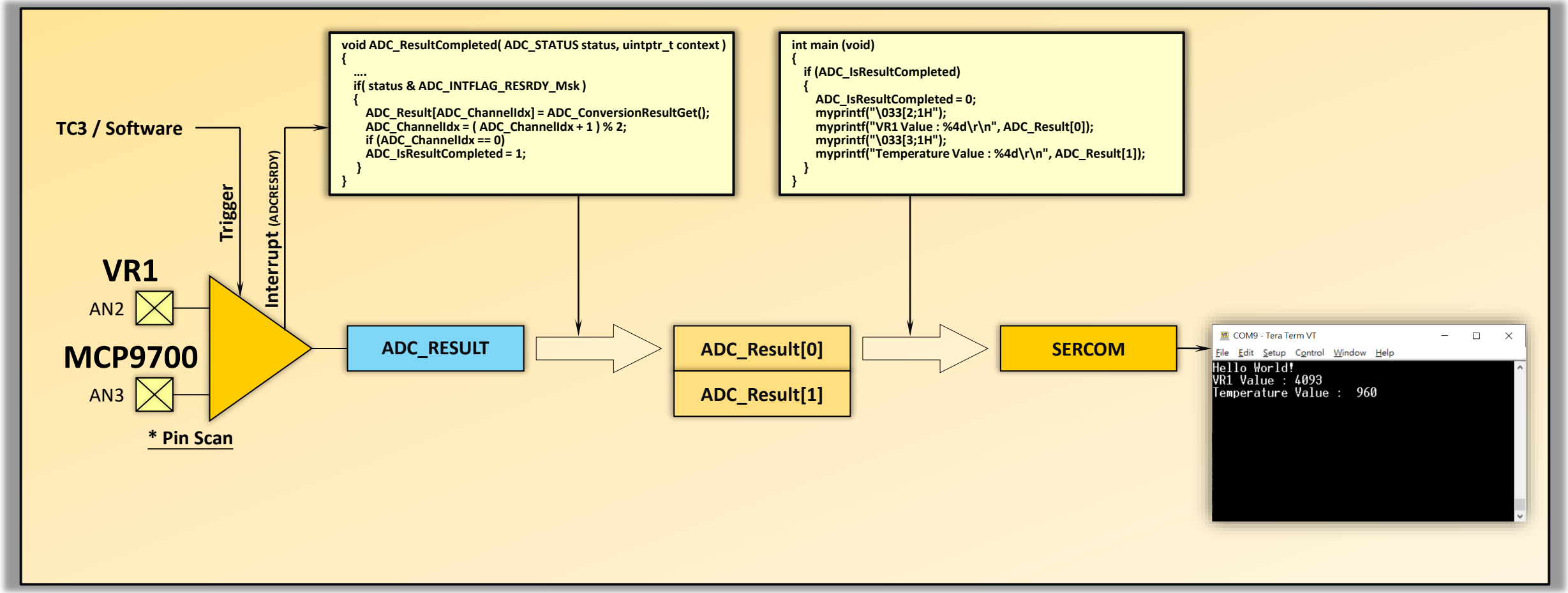
🛠 Lab13-3 ADC with DMAC

- **Please continue from SAM2001 Lab13 (ADC PinScan Interrupt Callback SWTrigger)**
  this lab run ADC pin scan mode to convent potentiometer and temperature sensor voltage and moving results to SRAM by CPU, and put it to UASRT, then show it at terminal.
- **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 without CPU resources.**
- **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 !**

MICROCHIP

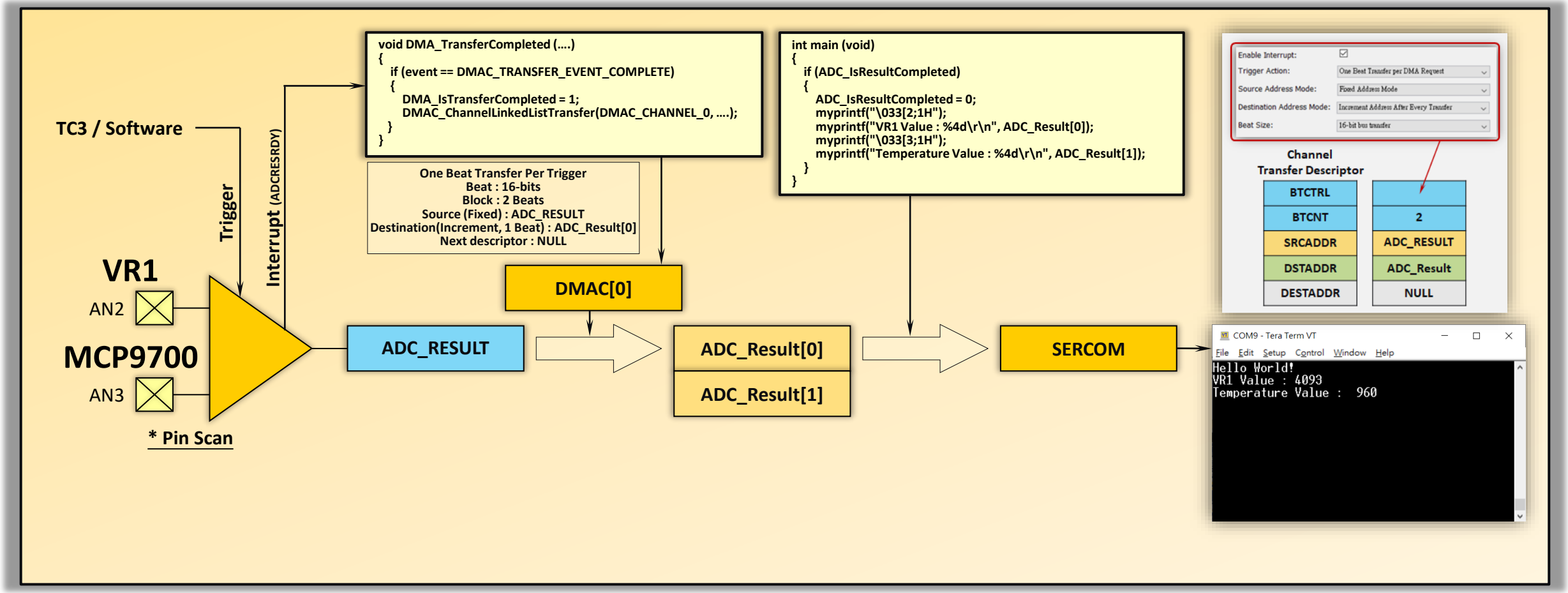
🛠 Lab13-3 ADC with DMAC

- Original block diagram for Lab13_ADC_with_DMAC. moving data and put to SERCOM by CPU.

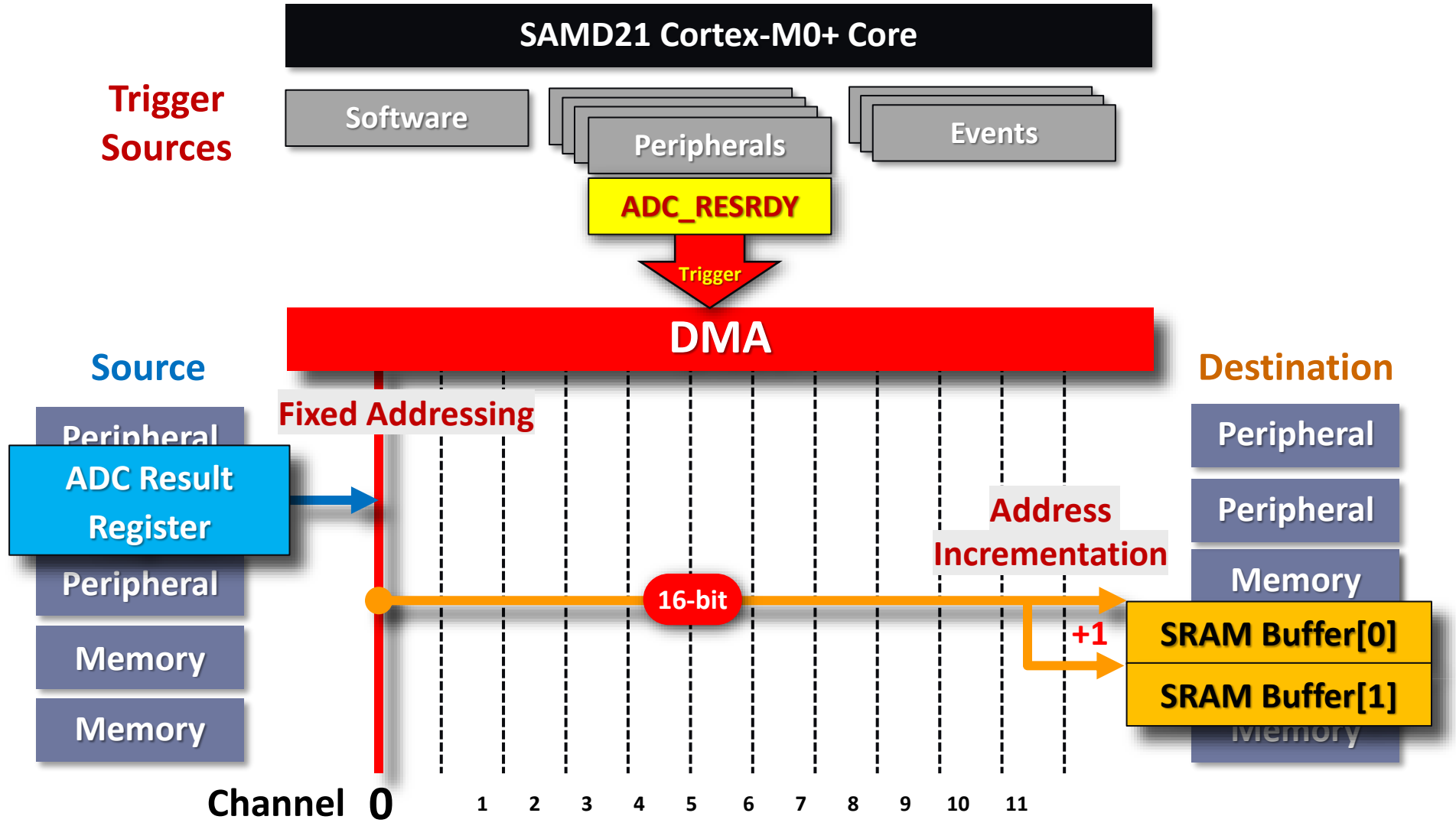


🛠 Lab13-3 ADC with DMAC

- New block diagram require for Lab13-3_ADC_with_DMAC. moving data by DMAC and put to SERCOM by CPU.



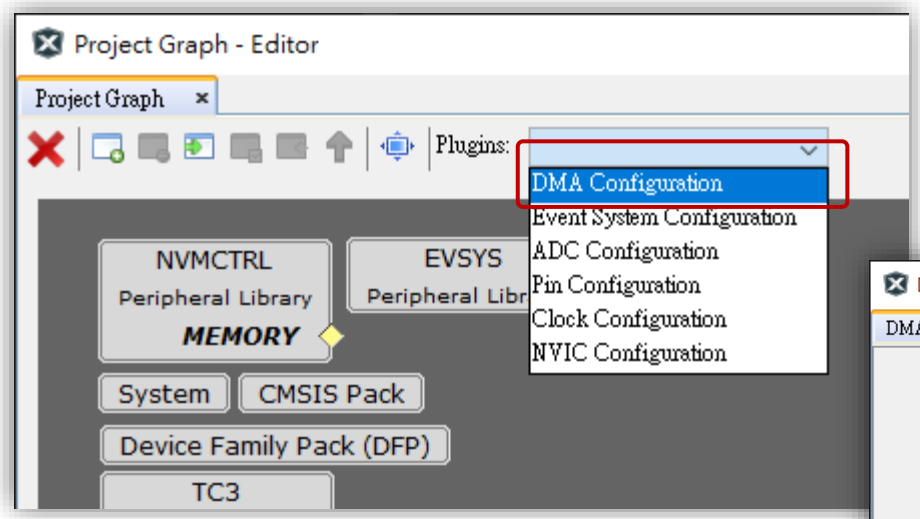
Lab13-3 ADC with DMAC

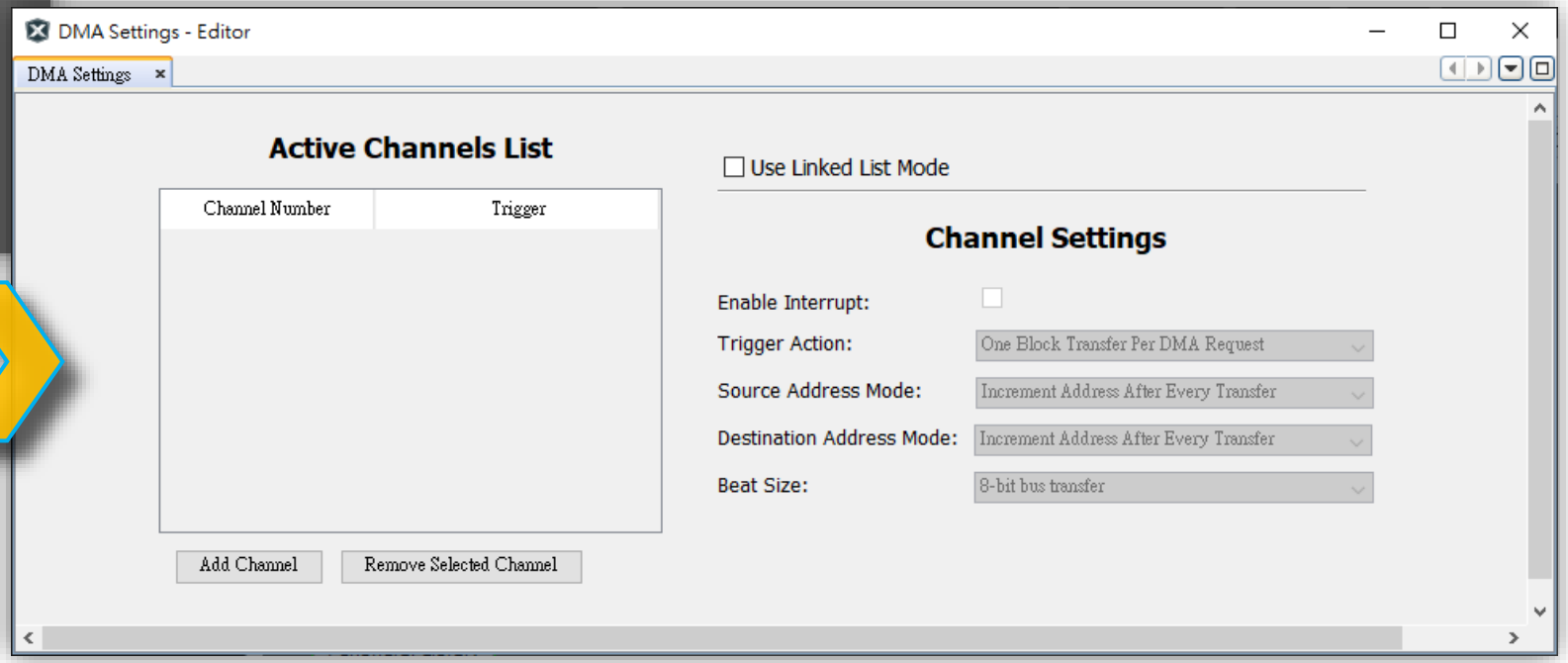


🛠 Lab13-3 ADC with DMAC

Step 1

- **Find “DMA Settings” at “DMA Configuration”**

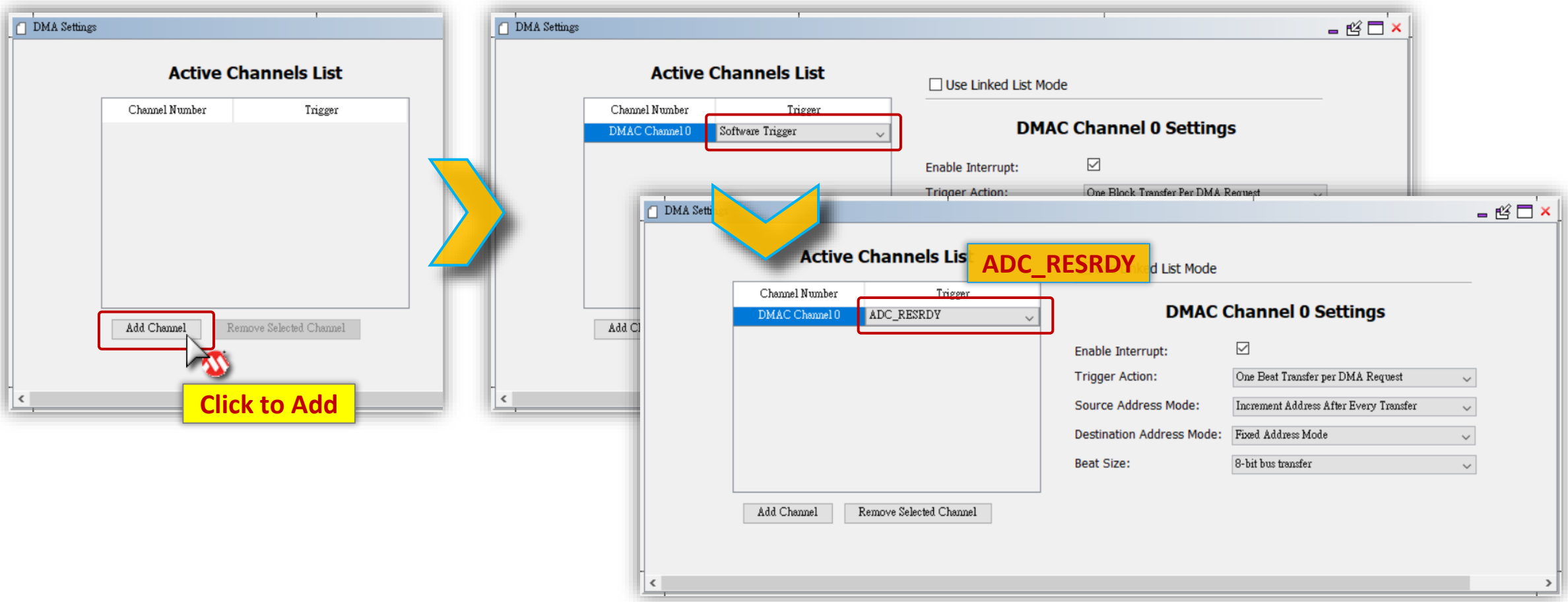




🛠 Lab13-3 ADC with DMAC

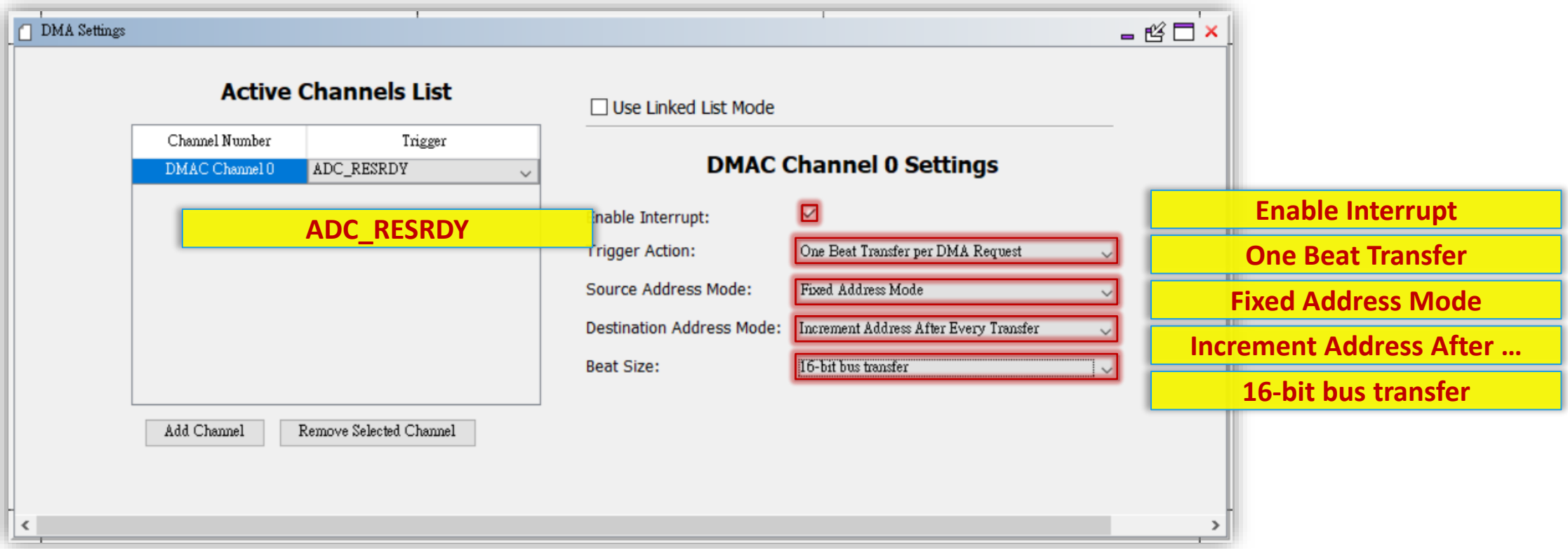
Step 2

- Add DMAC channel and set trigger source to ADC_RESRDY



🛠 Lab13-3 ADC with DMAC

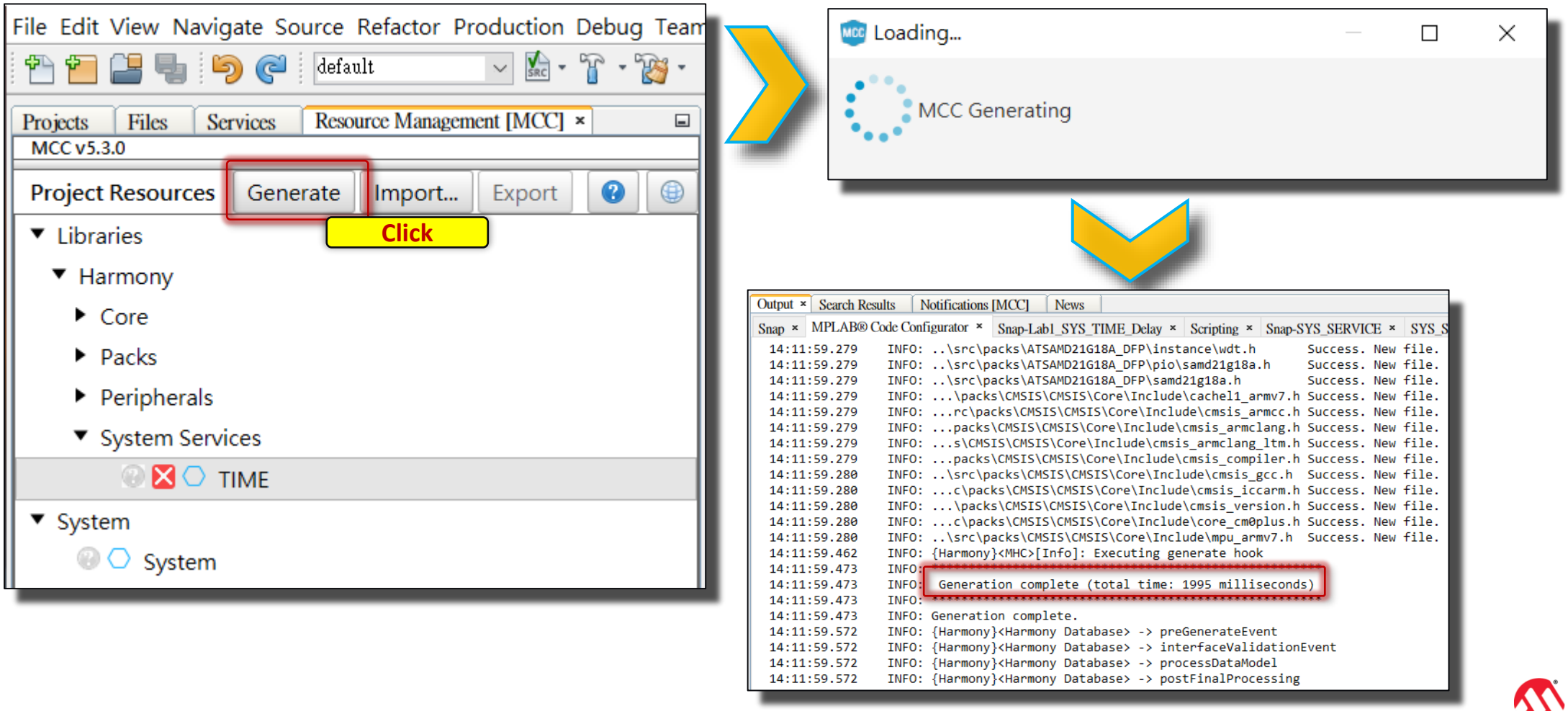
Step 3



🛠 Lab13-3 ADC with DMAC

Step 4

- **Click to Generate Code.**



MICROCHIP

🛠 Lab13-3 ADC with DMAC

Step 5

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

// TODO 13-3.02
//void ADC_ResultCompleted(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_IsResultCompleted = 1;
//        }
//    }
//}

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

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

MICROCHIP

🛠 Lab13-3 ADC with DMAC

Step 6

```
...

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

        // TODO 13-3.04
        // if (ADC_IsResultCompleted)
        // {
        //     ADC_IsResultCompleted = 0;
        //     myprintf("\033[2;1H");
        //     myprintf("VR1 Value : %4d\r\n", ADC_Result[0]);
        //     myprintf("\033[3;1H");
        //     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_IsTransferCompleted = 0;

// TODO 13-3.06
void DMA_TransferCompleted(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);
}
```

MICROCHIP

🛠 Lab13-3 ADC with DMAC

Step 8

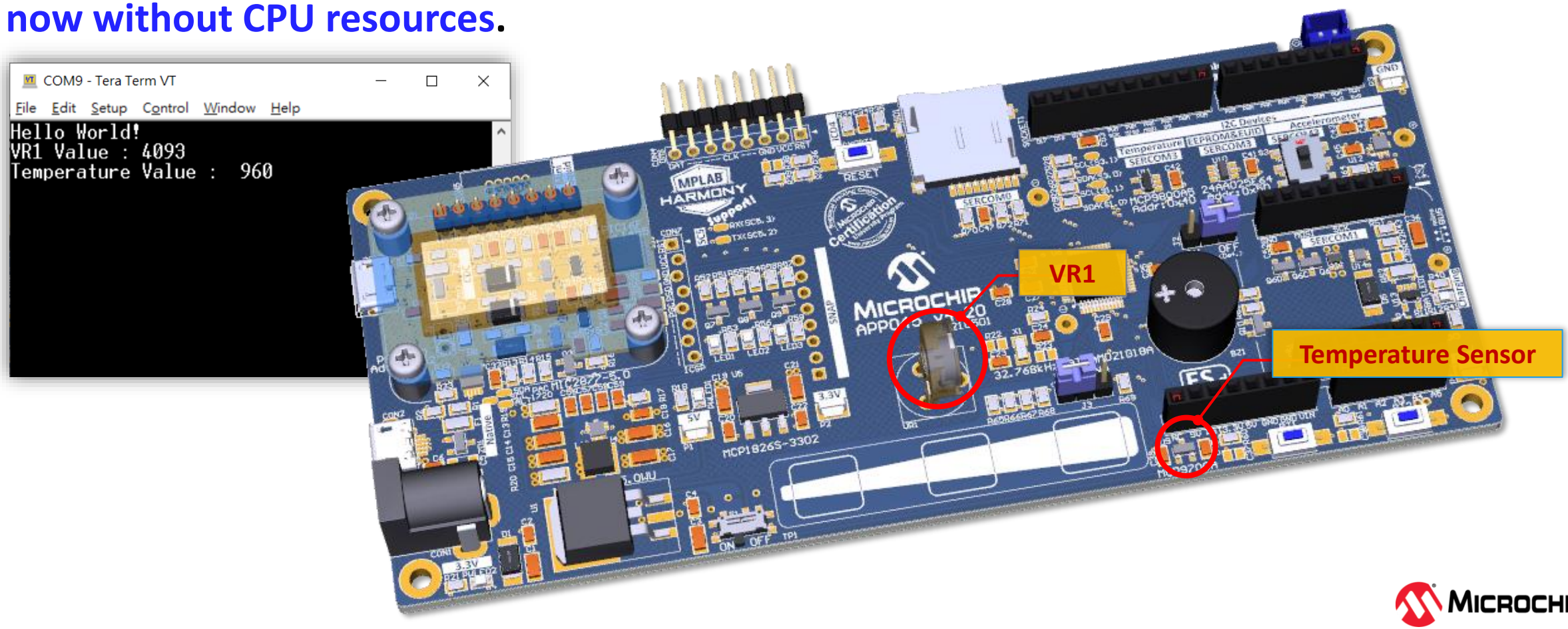
```
int main (void)
{
    SYS_Initialize(NULL);
    ...
    // TODO 13-3.07
    DMAC_ChannelCallbackRegister(DMAC_CHANNEL_0, DMA_TransferCompleted, (uintptr_t) NULL);
    DMAC_ChannelTransfer(DMAC_CHANNEL_0,
                         (const void *) &(ADC_REGS->ADC_RESULT),
                         (const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

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

        // TODO 13-3.08
        if (DMA_IsCompleted)
        {
            DMA_IsCompleted = 0;
            myprintf("\033[2;1H");
            myprintf("VR1 Value : %4d\r\n", ADC_Result[0]);
            myprintf("\033[3;1H");
            myprintf("Temperature Value : %4d\r\n", ADC_Result[1]);
        }
    }
    return ( EXIT_FAILURE);
}
```

MICROCHIP

🛠 Lab13-3 ADC with DMAC

Result

- **“Hello World!!” string and ADC result of Potentiometer / Temperature Sensor will output to terminal with 0.2 second period. Same as Lab13, but data moving by DMAC now without CPU resources.**

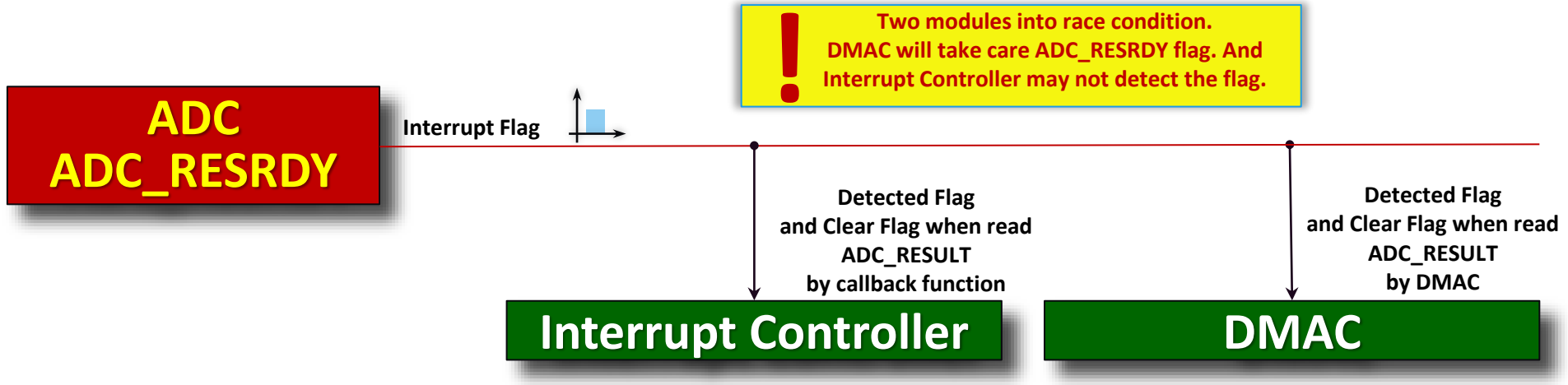


🛠 Lab13-3 ADC with DMAC

- **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.



MICROCHIP

🛠 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*

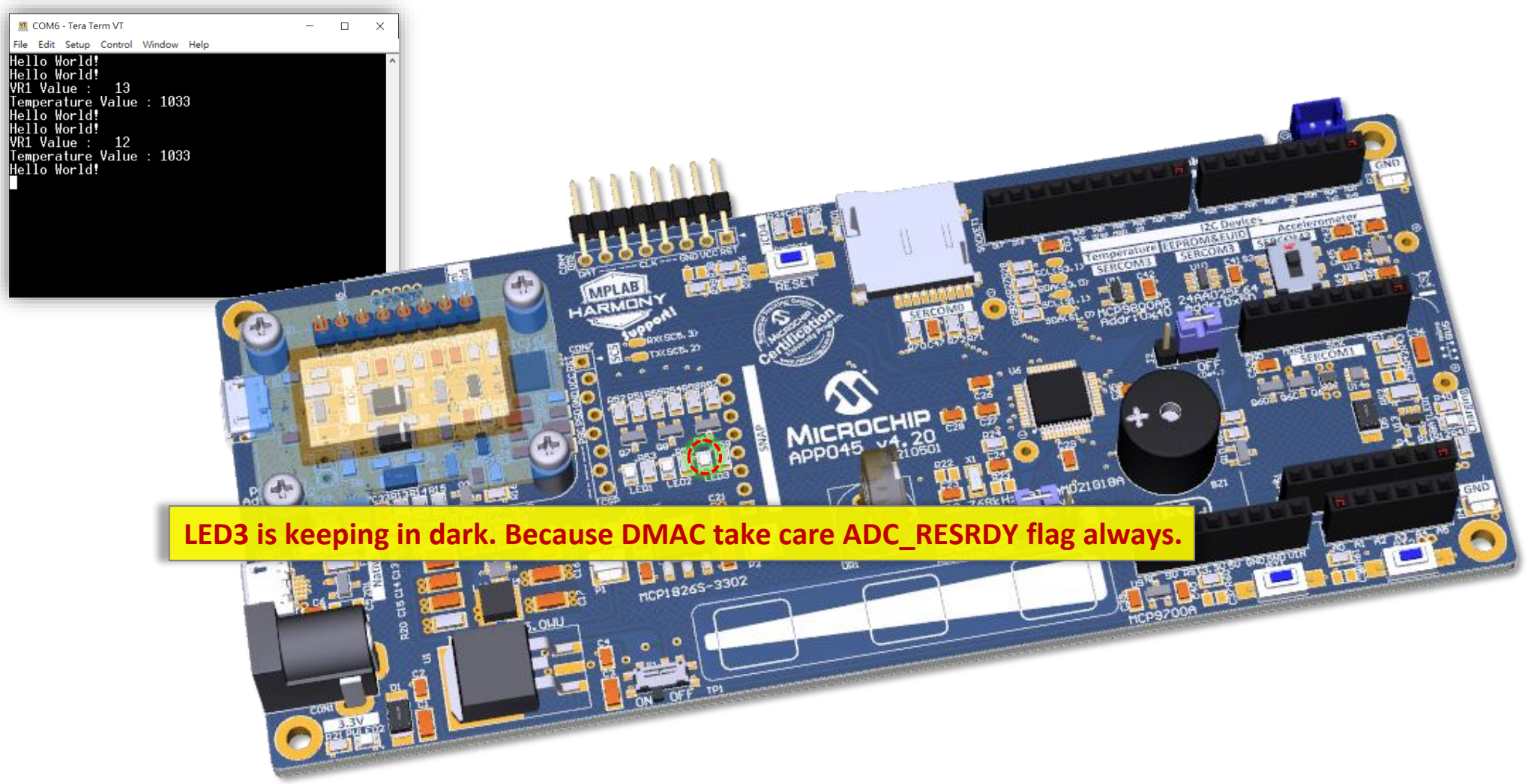
```
// TODO 13-3.01
volatile uint8_t ADC_IsCompleted = 0;
volatile int8_t ADC_ChannelIdx = 0;
```

```
// 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



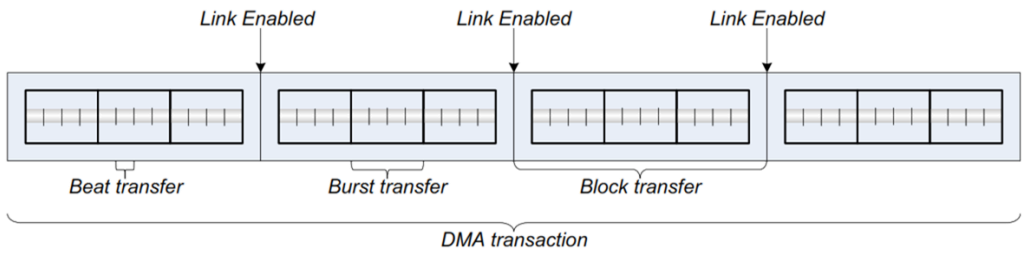
Direct Memory Access Controller (DMAC)
- Linked List for Transfer Block

MICROCHIP

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$_{(8, 16, 32\text{-bits})}$ 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
DESTADDR

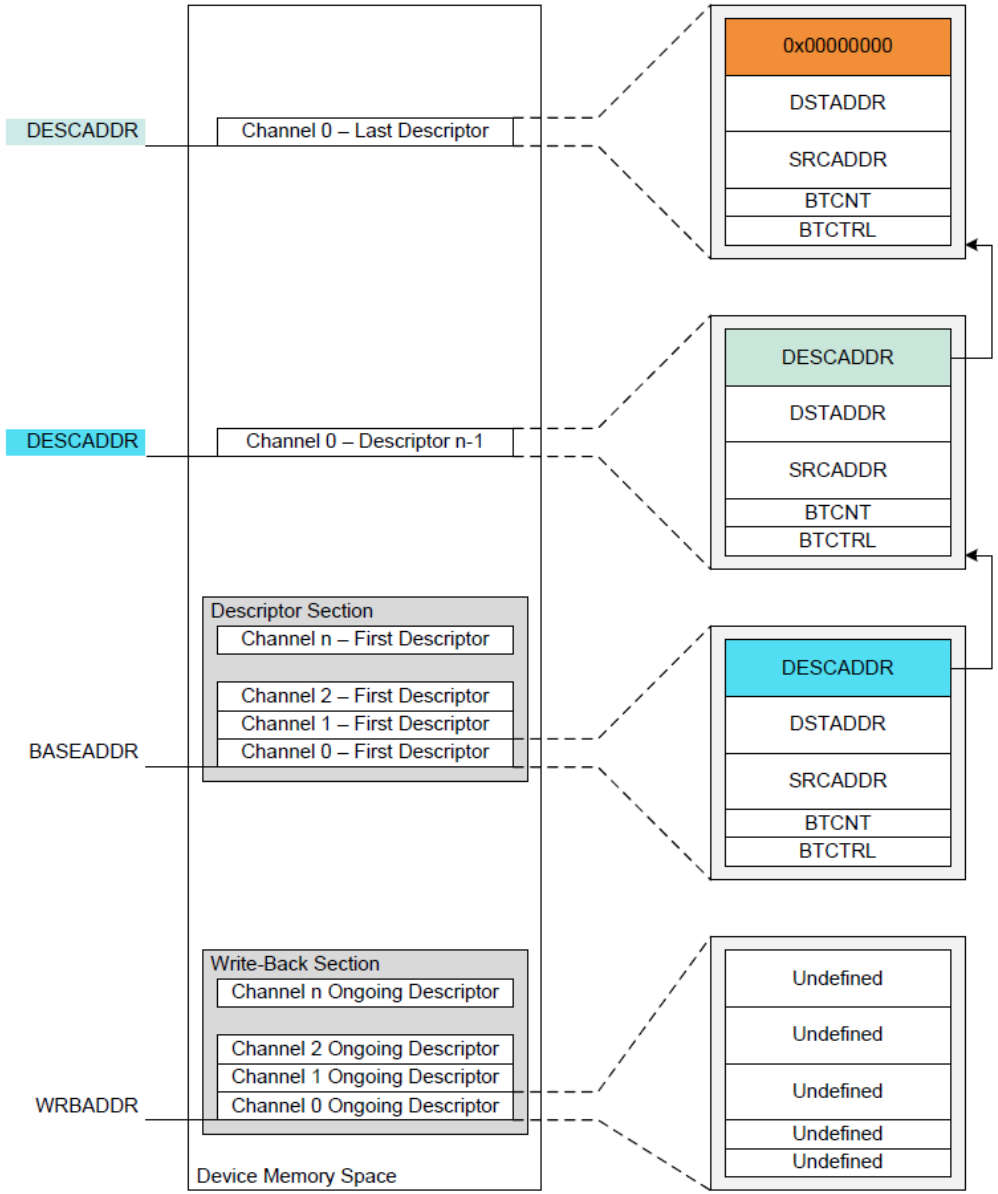
MICROCHIP

Linked List Mode of Transfer Descriptor

MICROCHIP

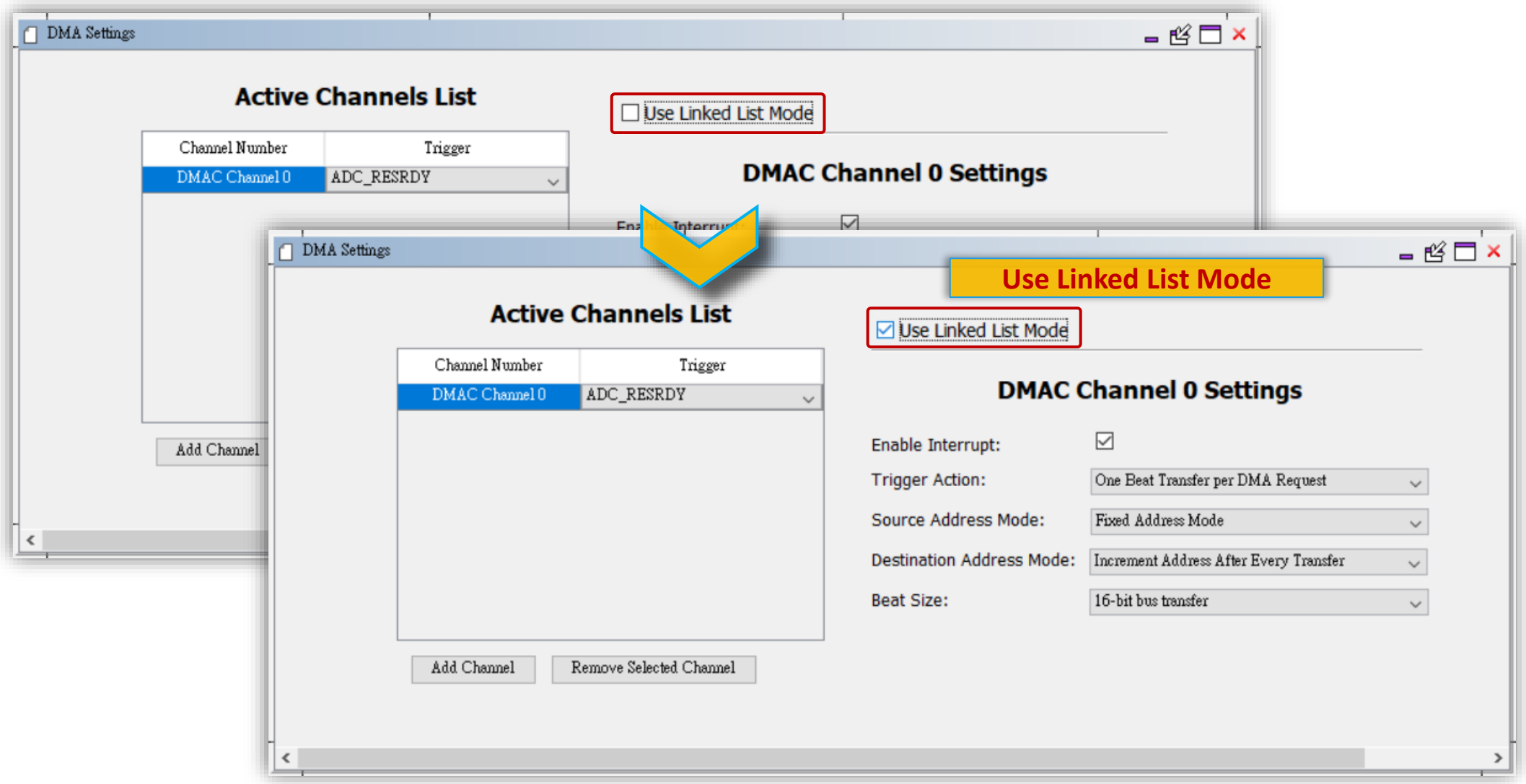
Linked Descriptors

- 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 DMAC Settings.



DMAC Functions for Link List Mode

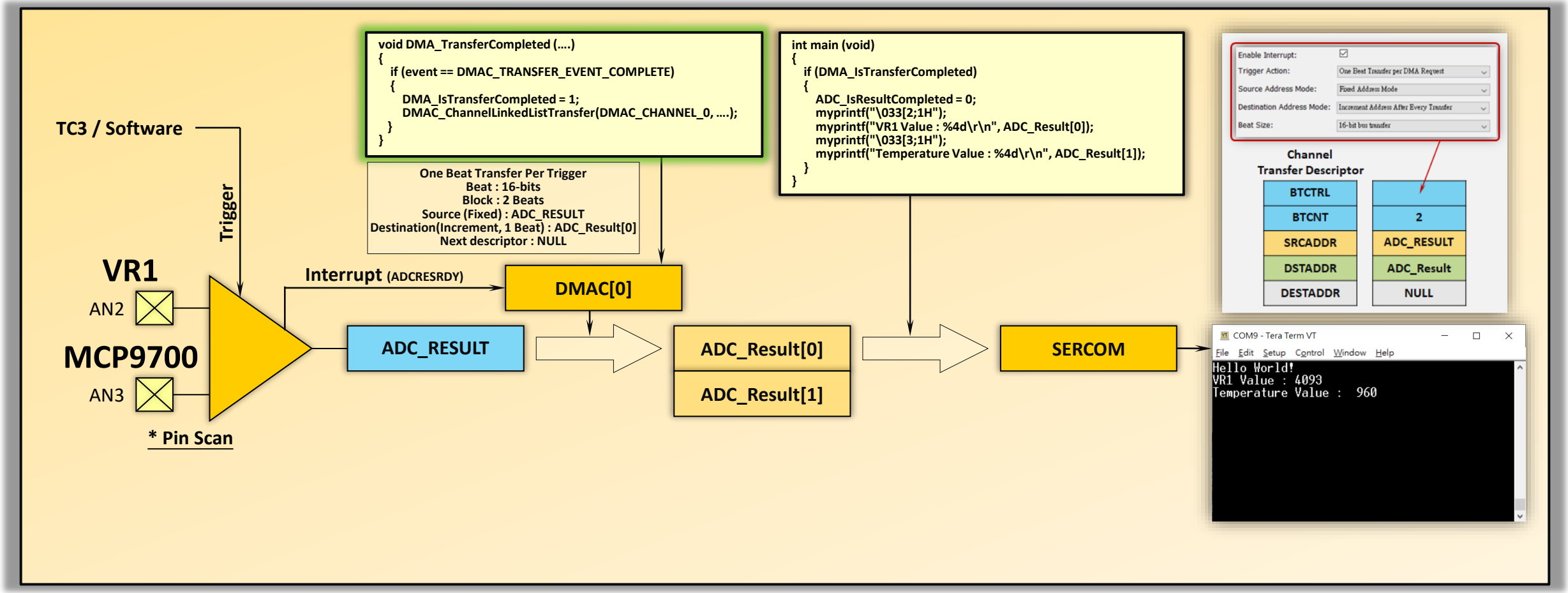
Interface Routines

```
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);
```

MICROCHIP

Review Block Diagram for Lab13-3_ADC_with_DMAC

- **Data transfer by DMAC and send it to SERCOM by CPU.**
  And assign new transfer job after moving finished every time at DMAC callback.



Think !

- **CPU still intervene the DMAC operation.**
  Because the code assign new transfer job after transfer finished every time at DMAC callback.

- **How to set DMAC transfer automatic without CPU intervene ?**
  - Single Transfer Block Mode is one short operation.
  - Linked List Transfer Block Mode can fit this require.

- **Note ! DAMC stop after Transaction finished.**
  - At Single Transfer Block Mode,
    Transaction finished means : One Block Transfer finished.
  - At Linked List Transfer Block Mode,
    Transaction finished means :
    All Linked Block Transfer finished.

- **So, to set an infinite(Circular) Linked List.**
  become a never stop “Transaction” can fit this require.

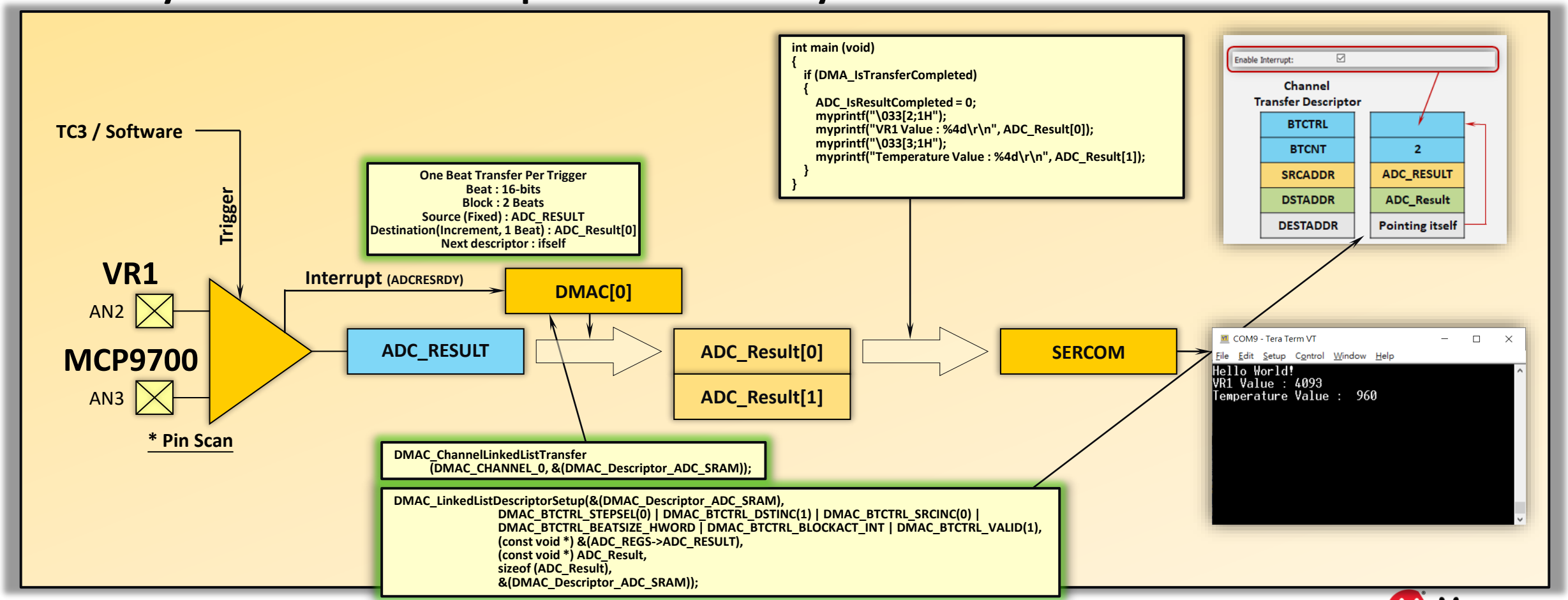
🛠 Lab13-5 ADC with Descriptors Link of DMAC

- Please continue from SAM2001ADV Lab13-3 (Lab13-3 ADC with DMAC)
- Try to set the DMAC to Linked List Operation Mode to transfer 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

- **New block diagram require for Lab13-5 ADC with Linked List DMAC.**
  Data transfer by DMAC and send it to SERCOM by CPU. Transfer require of DMAC be load by new Transfer Descriptor automatically.

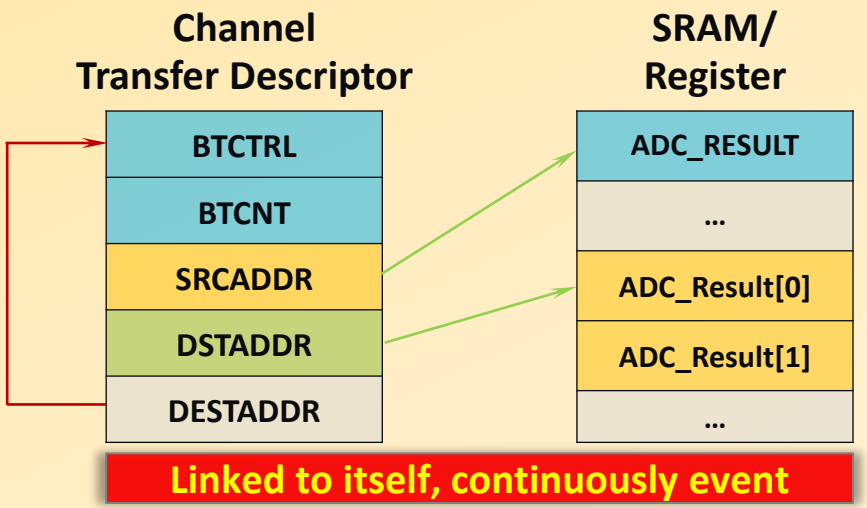


MICROCHIP

🛠 Lab13-5 ADC with Descriptors Link of DMAC

- **Linked List Mode C Code Example**

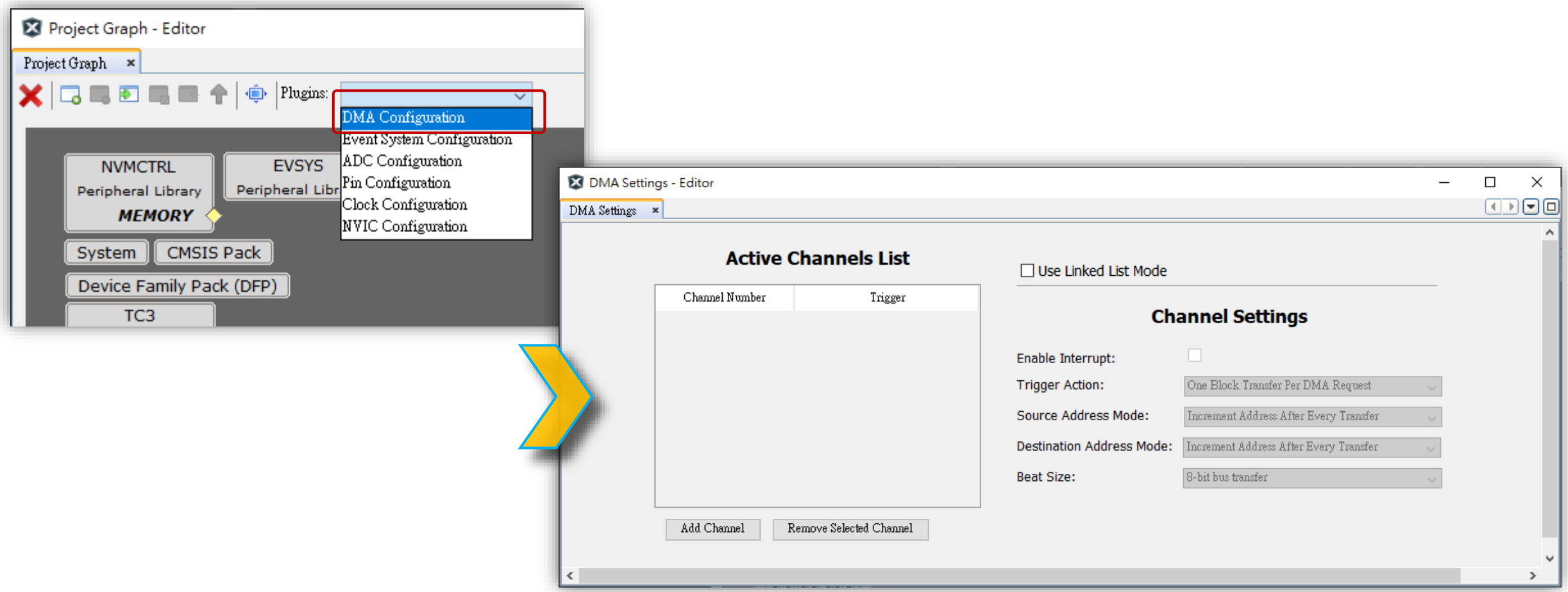
```
void DMA_TransferCompleted (DMAC_TRANSFER_EVENT event, uintptr_t contextHandle)
{
   if (event == DMAC_TRANSFER_EVENT_COMPLETE)
   {
      DMA_TransferCompleted= 1;
   }
}
dmac_descriptor_registers_t DMAC_Descriptor_ADC_SRAM;

int main (void)
{
   SYS_Initialize ( NULL );
   ...
   DMAC_ChannelCallbackRegister(DMAC_CHANNEL_0 DMA_TransferCompleted, (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));
   {...}
}
```



🛠 Lab13-5 ADC with Descriptors Link of DMAC

Step 1

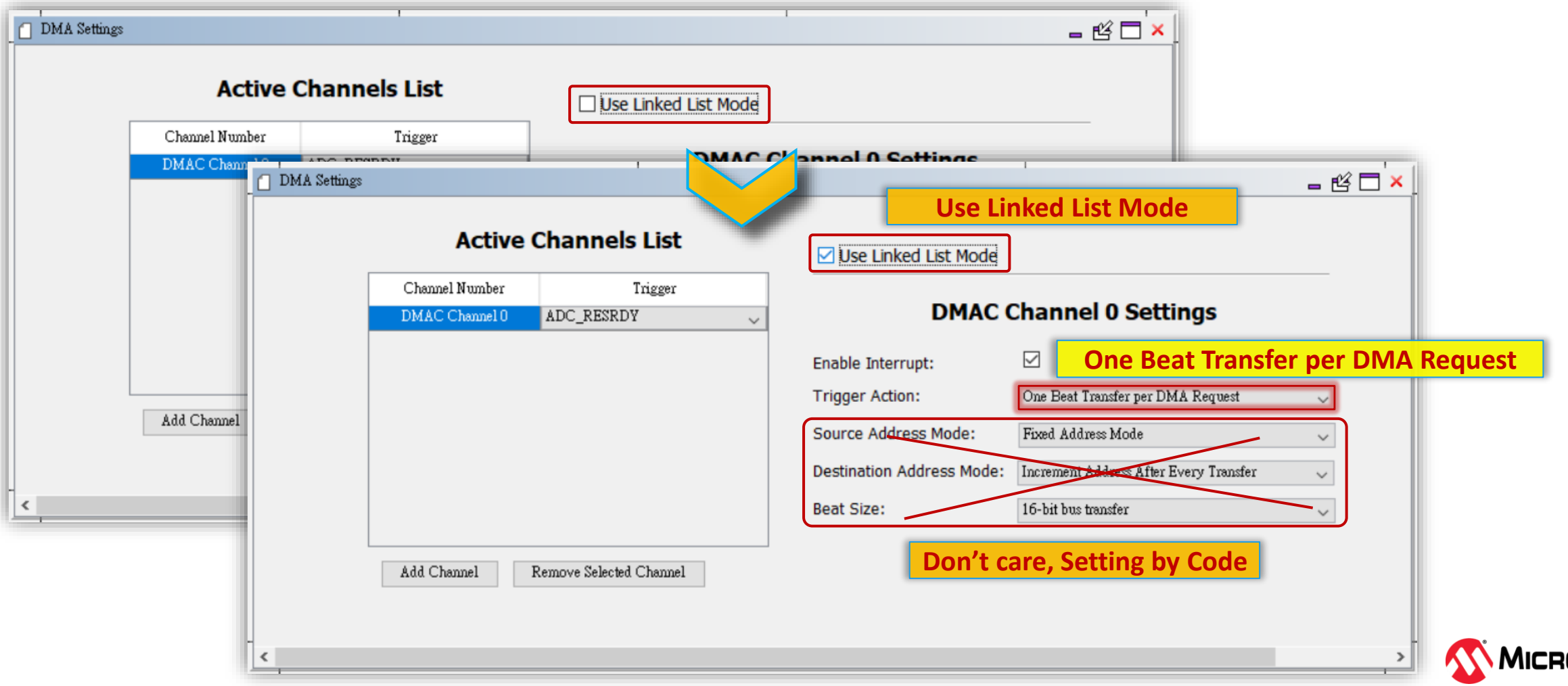
- **Find “DMA Settings” at “DMA Configuration”**



🛠 Lab13-5 ADC with Descriptors Link of DMAC

Step 2

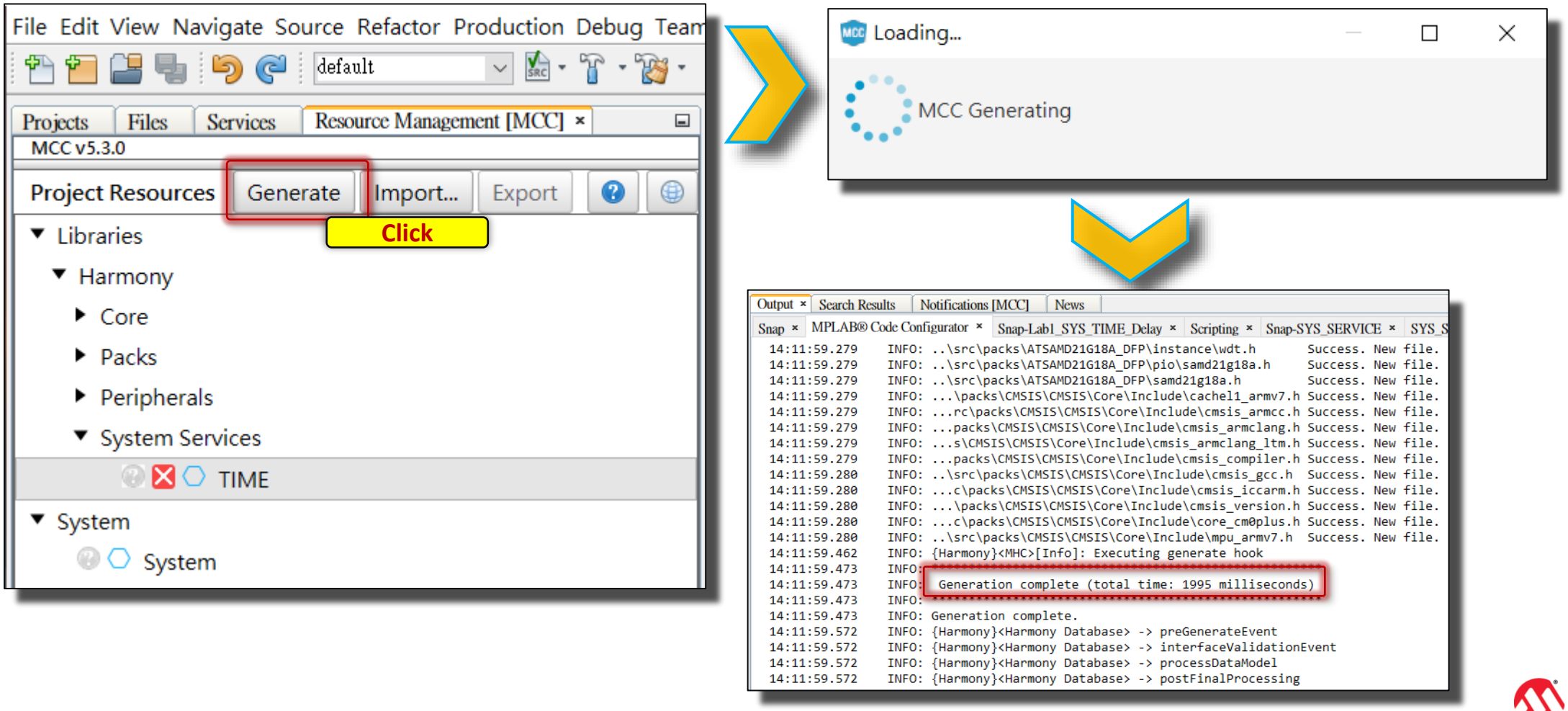
- **Enable Linked List Mode in DMA Settings.**



🛠 Lab13-5 ADC with Descriptors Link of DMAC

Step 3

- **Click to Generate Code.**



MICROCHIP

🛠 Lab13-5 ADC with Descriptors Link of DMAC

Step 4

```
void DMA_TransferCompleted(DMAC_TRANSFER_EVENT event, uintptr_t contextHandle)
{
    if (event == DMAC_TRANSFER_EVENT_COMPLETE)
    {
        DMA_IsTransferCompleted = 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)
{
    SYS_Initialize(NULL);
    ...
    // TODO 13-5.02
    //    DMAC_ChannelTransfer(DMAC_CHANNEL_0,
    //            (const void *) &(ADC_REGS->ADC_RESULT),
    //            (const void *) ADC_Result,
    //            sizeof (ADC_Result));

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

MICROCHIP

🛠 Lab13-5 ADC with Descriptors Link of DMAC

Step 5

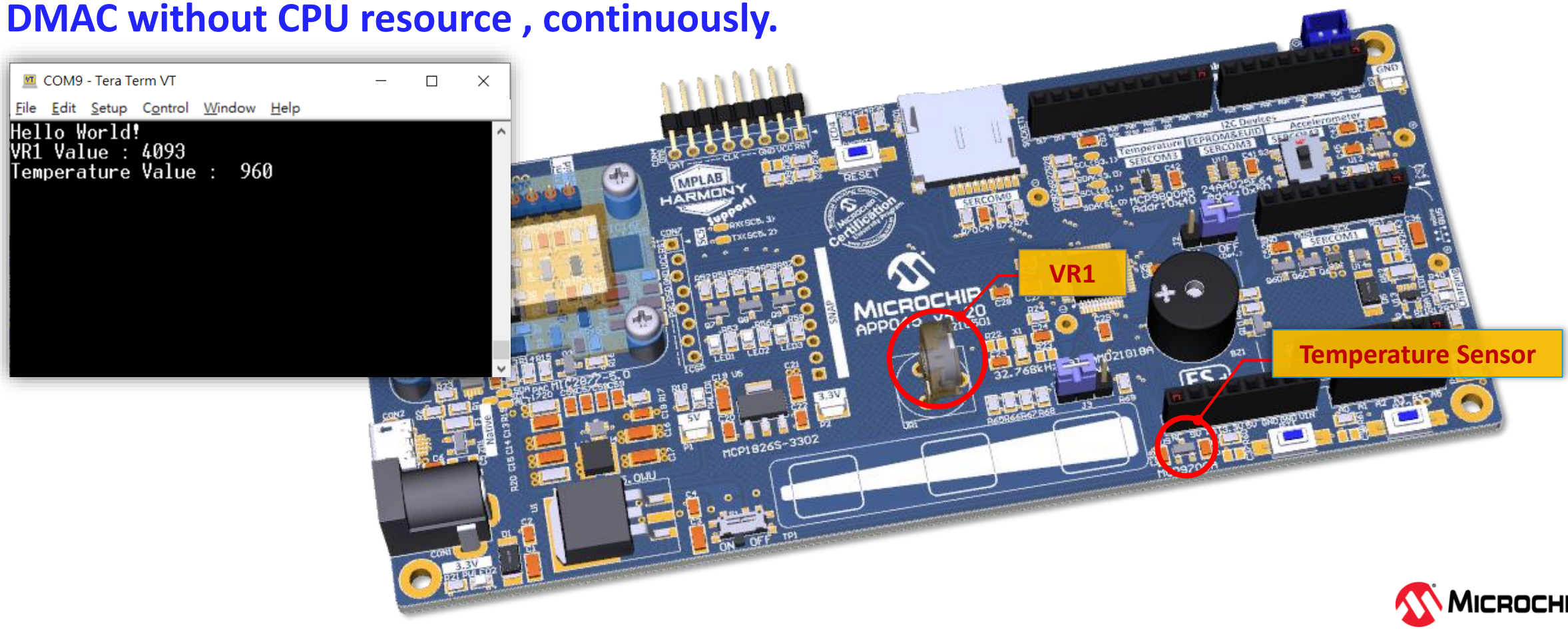
```
// TODO 3.03
dmac_descriptor_registers_t DMAC_Descriptor_ADC_SRAM;

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);
}
```

MICROCHIP

🛠 Lab13-5 ADC with Descriptors Link of DMAC

Result

- **“Hello World!!” string and ADC result of Potentiometer / Temperature Sensor will output to terminal with 0.2 second period. Same as lab13, lab13-3, but transfer by DMAC without CPU resource , continuously.**

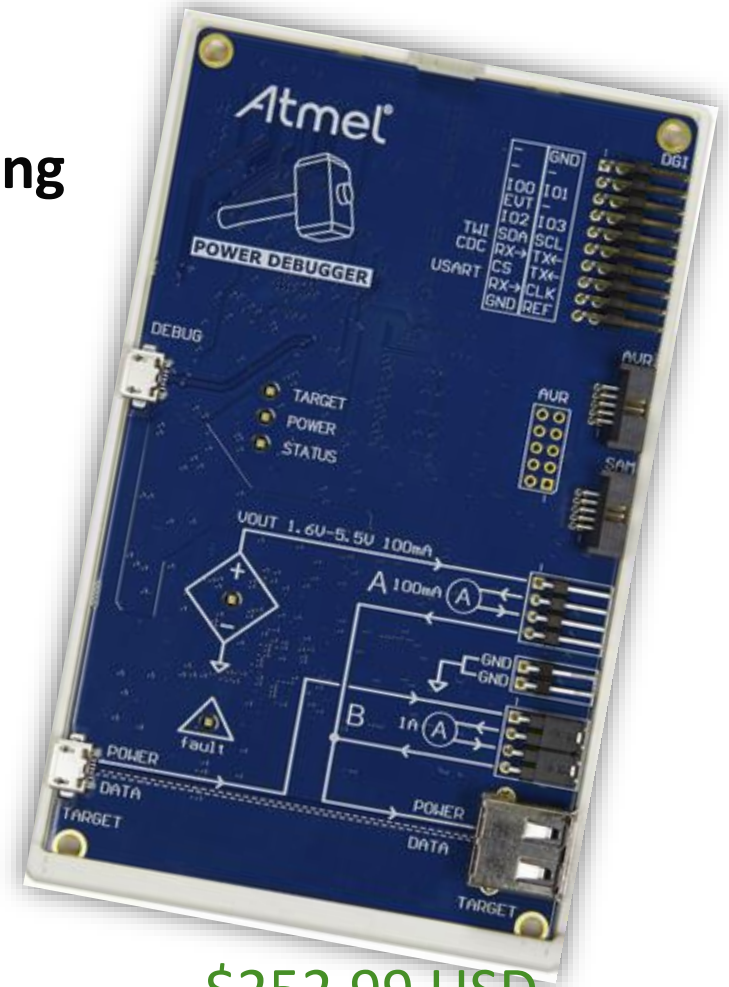


MICROCHIP

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



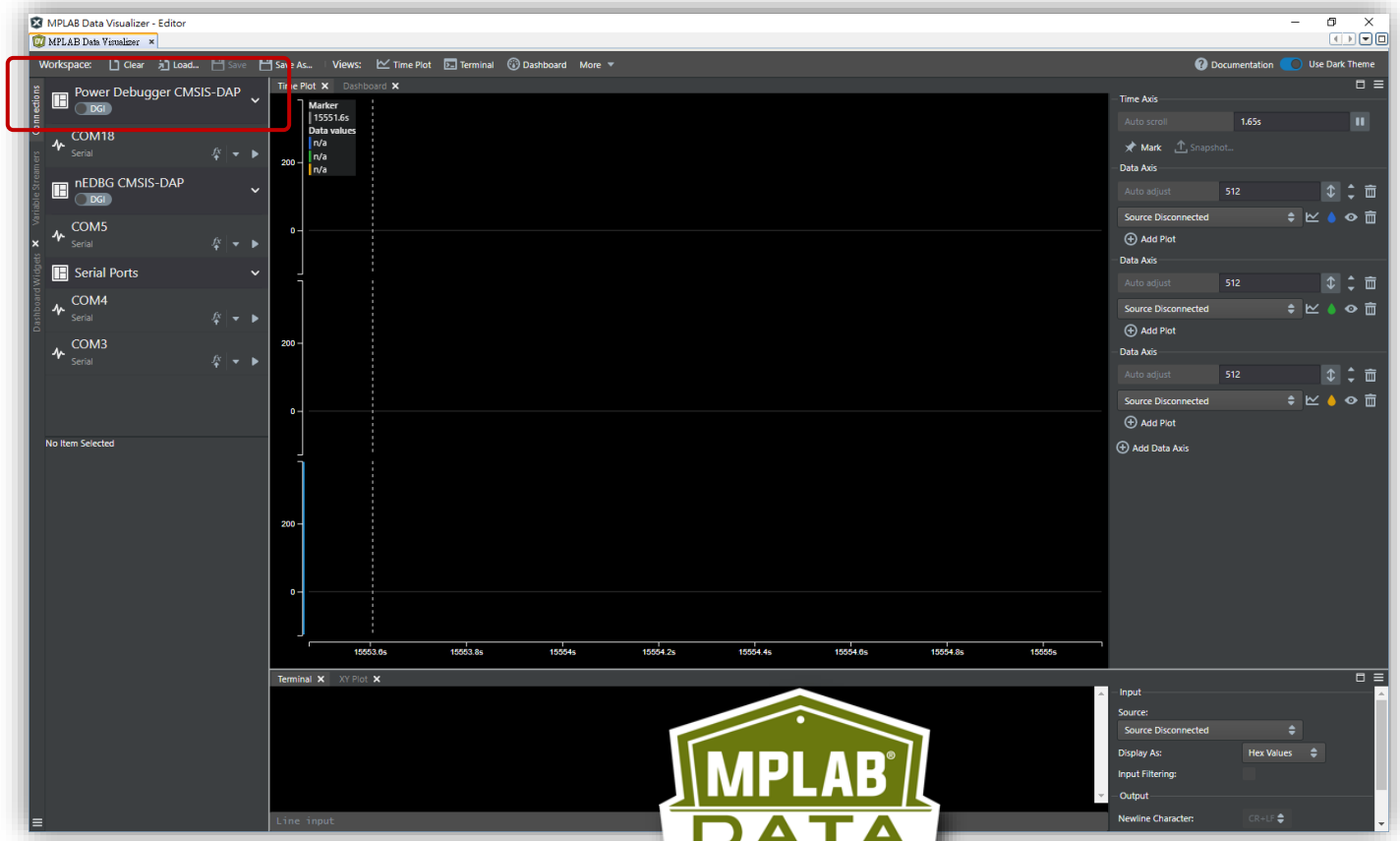
$252.99 USD

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

MICROCHIP

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.



MPLAB® DATA VISUALIZER

MICROCHIP

Power Manager - 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.

MICROCHIP

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

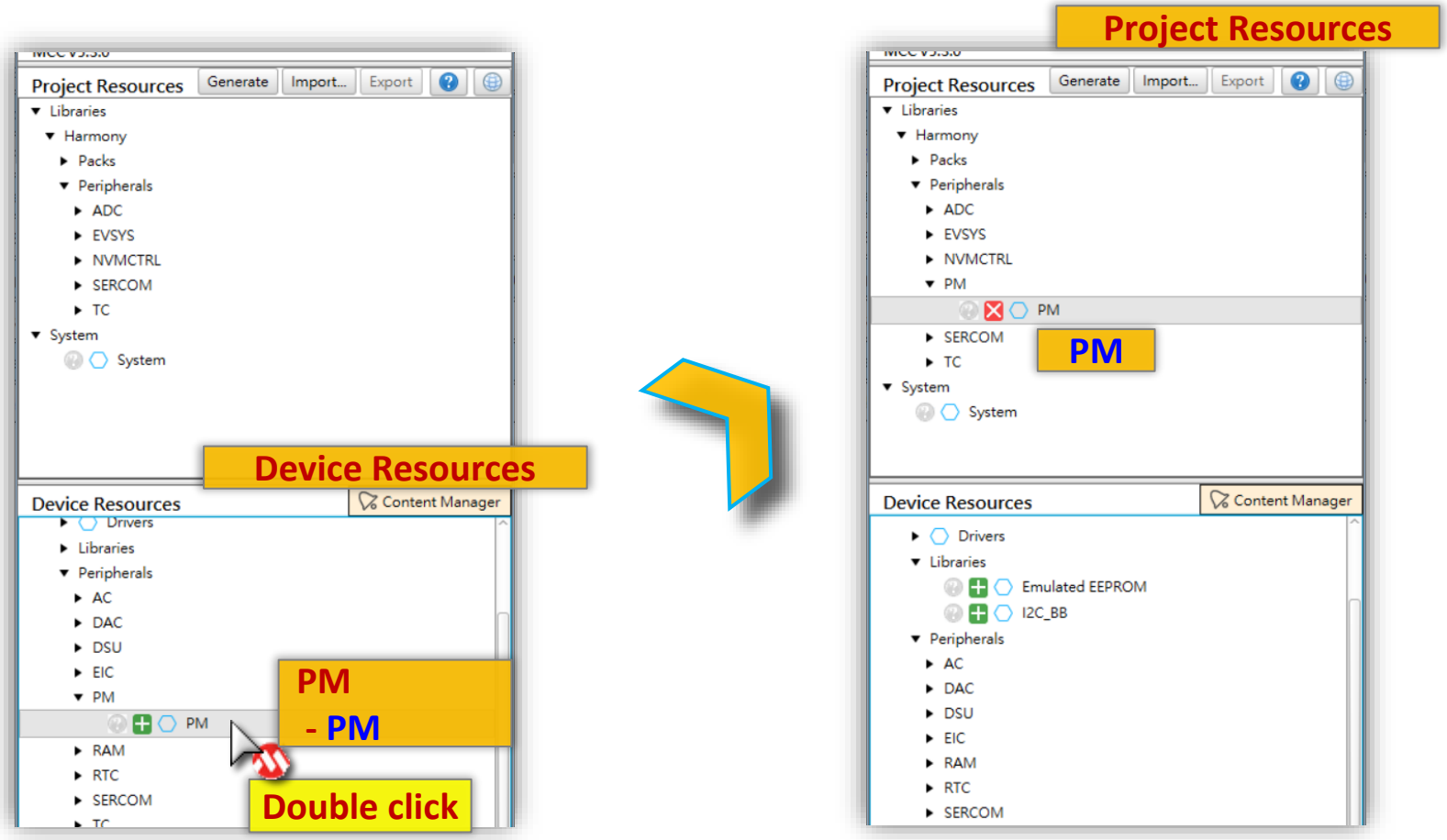
- **STANDBY Mode**
  - **All clock sources, clock generation and most peripheral are stopped, except those where the RUNSTDBY bit is set.**
  - **Regulator operates in low-power mode.**

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

MICROCHIP

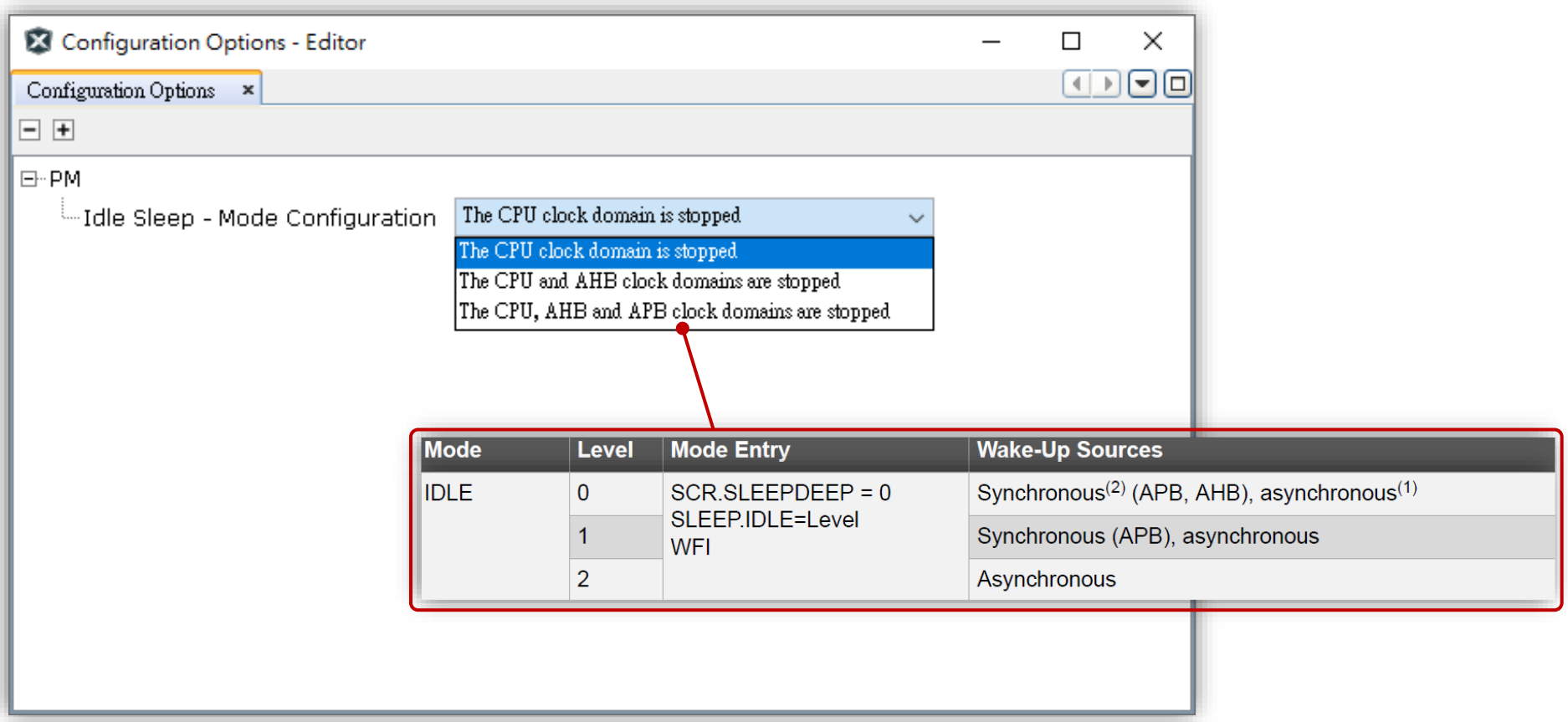
Add PM Function using MCC-Harmony

- Find PM component in “Device Resources” window.
- Double click PM to add to “Project Resources” window.



PM Function

Configuration Options Example



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

MICROCHIP

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

MICROCHIP

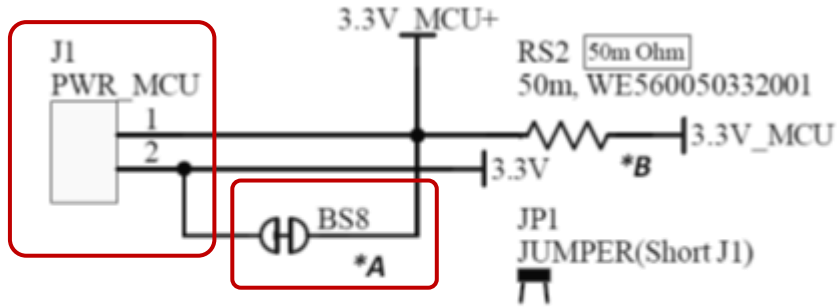
🛠 Lab13-6 ADC with IDLE

- 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.**

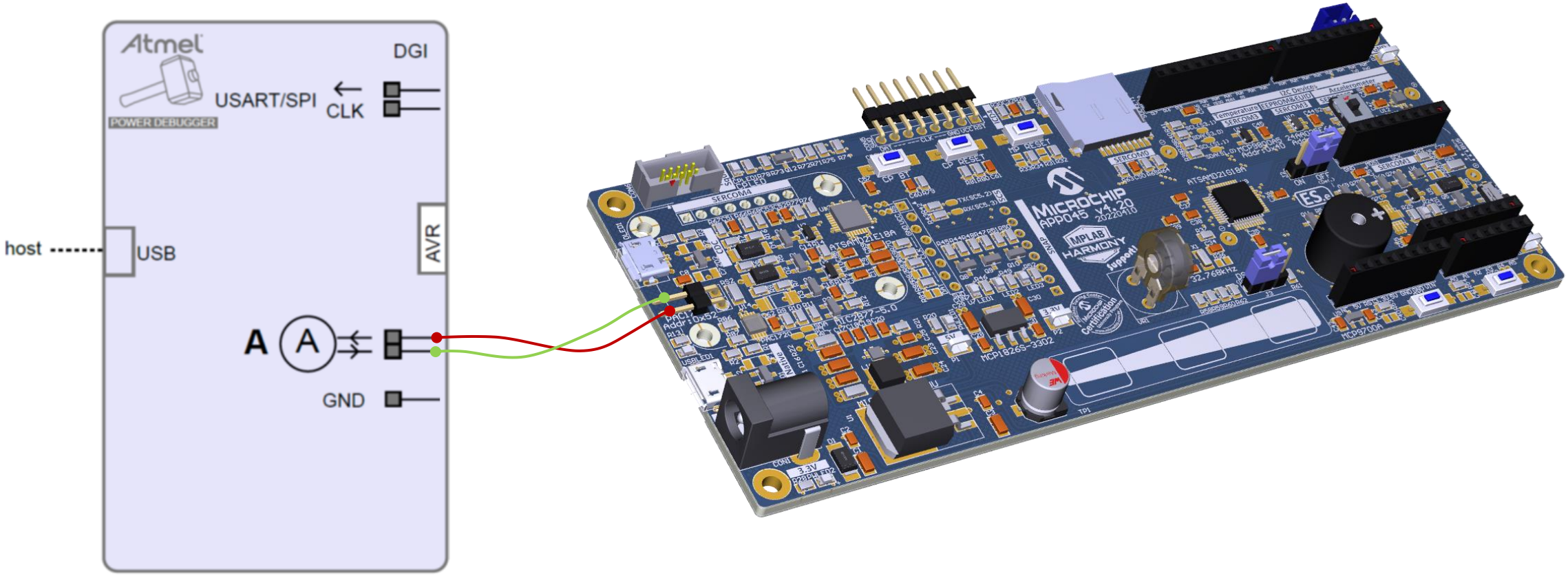


MCU Power with Monitor

**A PCB short by default*

Monitor By Power Debugger

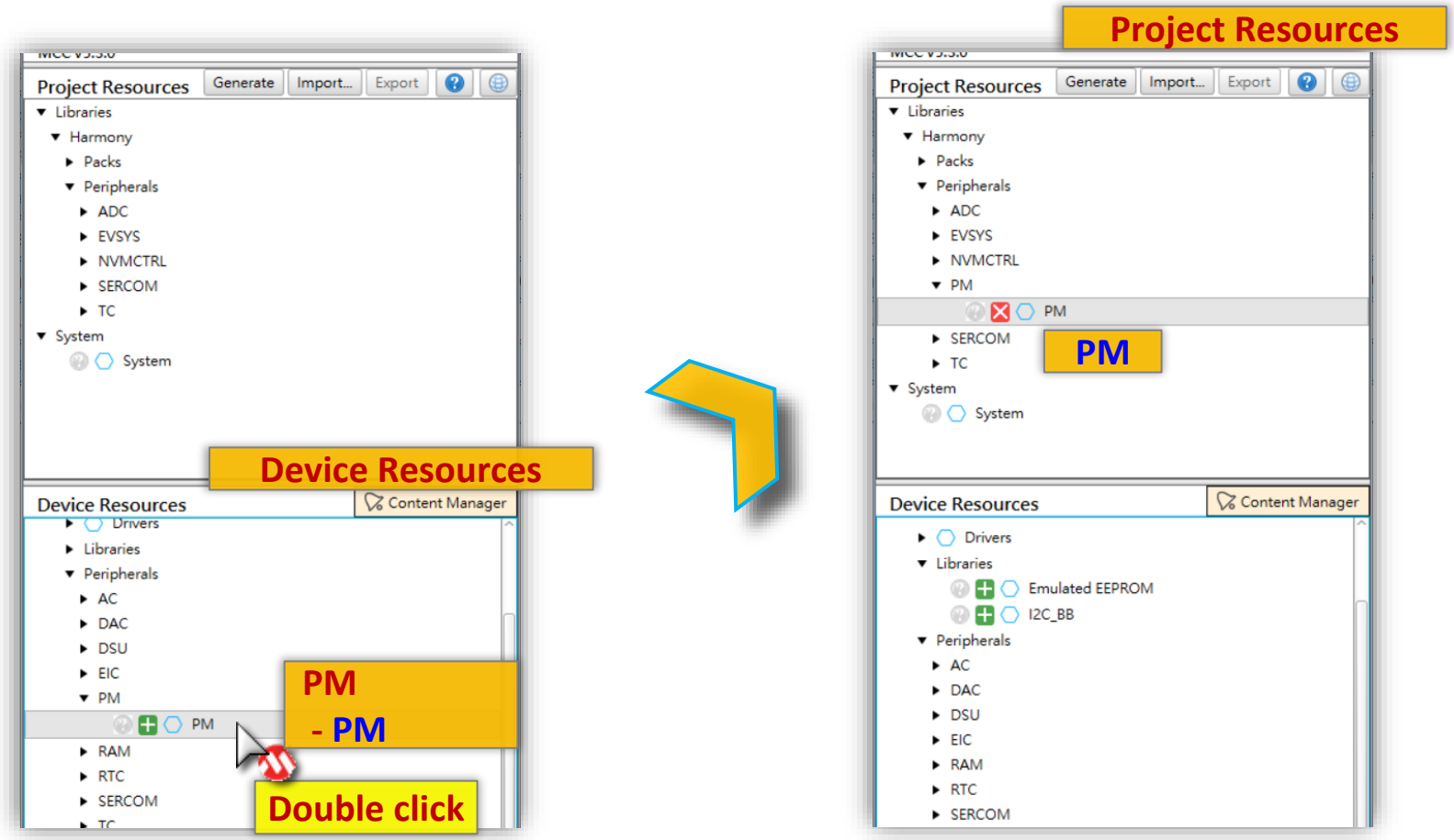
- **Connect MCU power monitor pins at APP045 to current monitor port (Channel A, 100mA) of Power Debugger.**



🛠 Lab13-6 ADC with IDLE

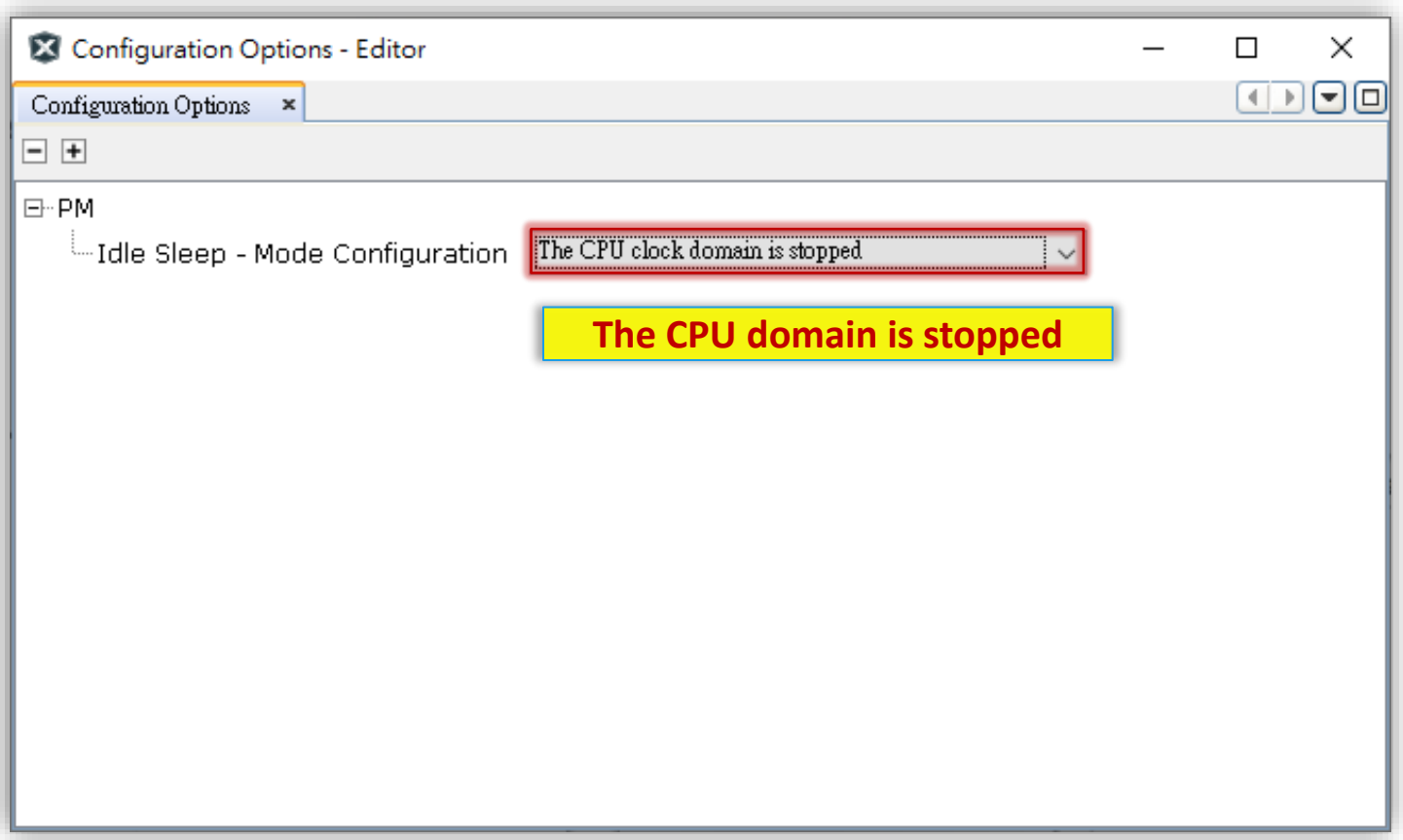
Step 1

- Find PM component in “Device Resources” window.
- Double click PM to add to “Project Resources” window.



🛠 Lab13-6 ADC with IDLE

Step 2

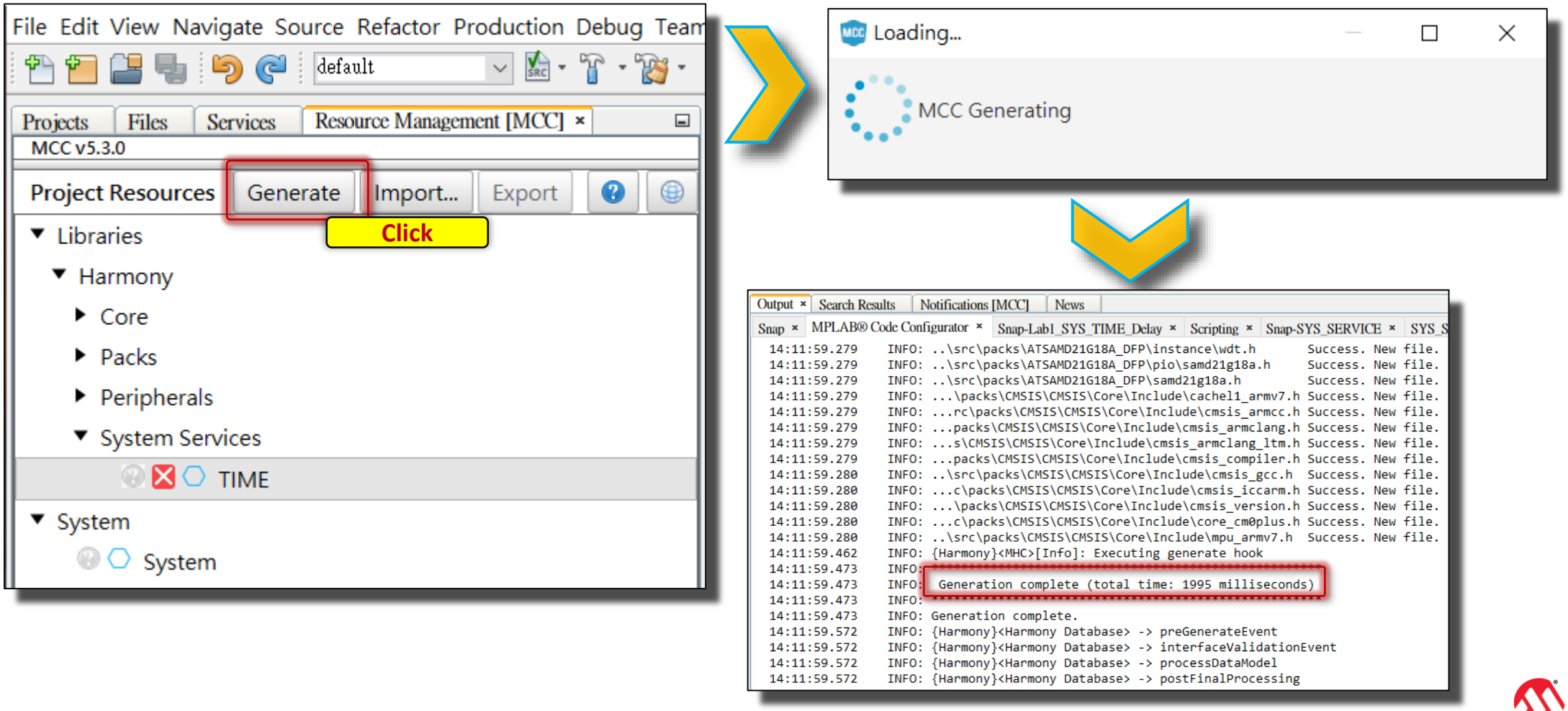


MICROCHIP

🛠 Lab13-6 ADC with IDLE

Step 3

- **Click to Generate Code.**



MICROCHIP

🛠 Lab13-6 ADC with IDLE

Step 4

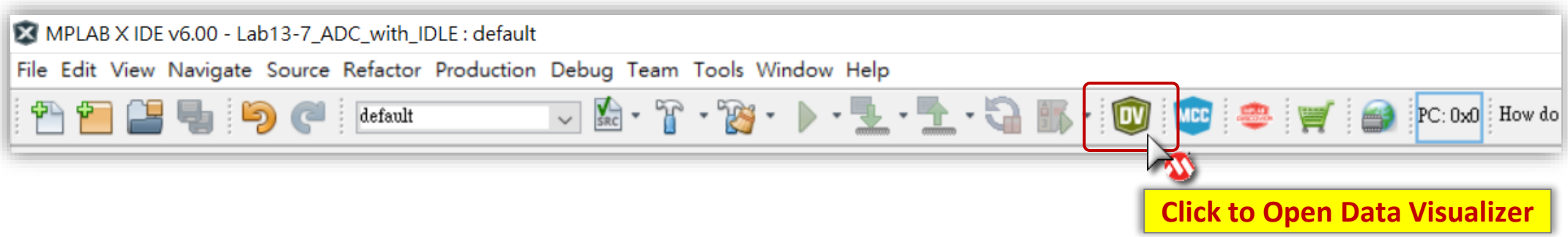
```
...
int main (void)
{
    SYS_Initialize(NULL);
    ...

    while ( true )
    {
        SYS_Tasks();

        // TODO 13-6.01
        PM_IdleModeEnter();
        ...
    }

    return ( EXIT_FAILURE);
}
```

MICROCHIP

🛠 Lab13-6 ADC with IDLE

Step 5

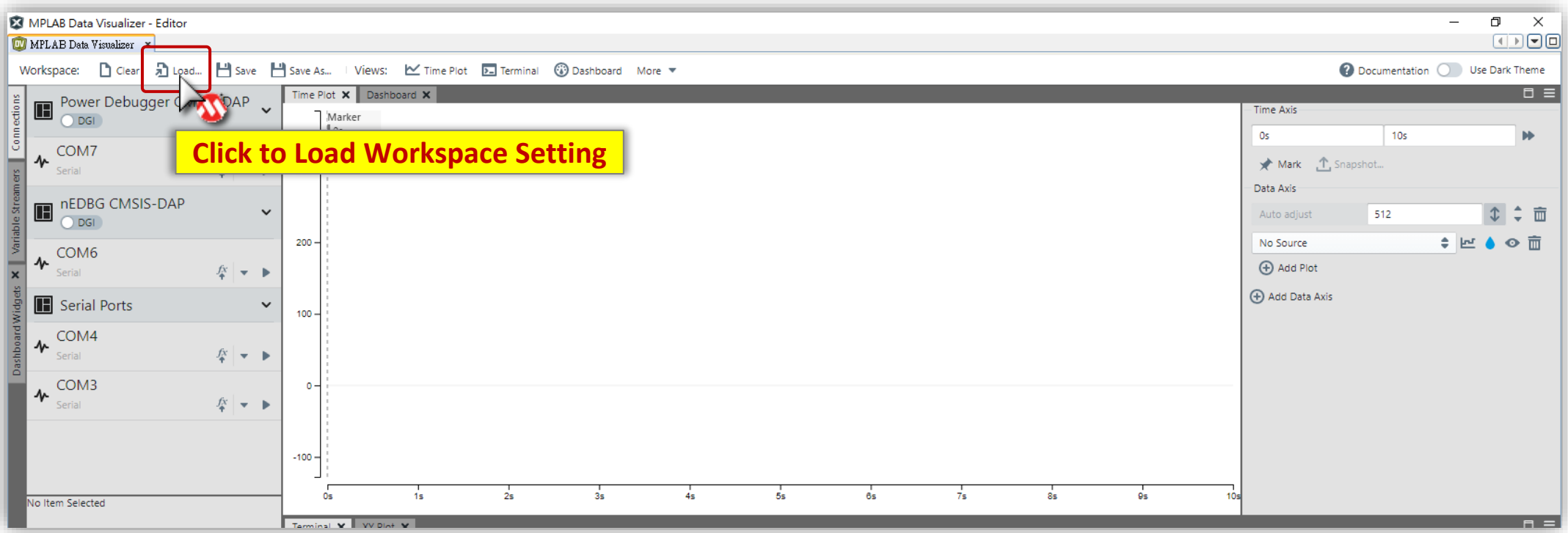
- **Open Data Visualizer by icon at MPLAB X IDE**



🛠 Lab13-6 ADC with IDLE

Step 6

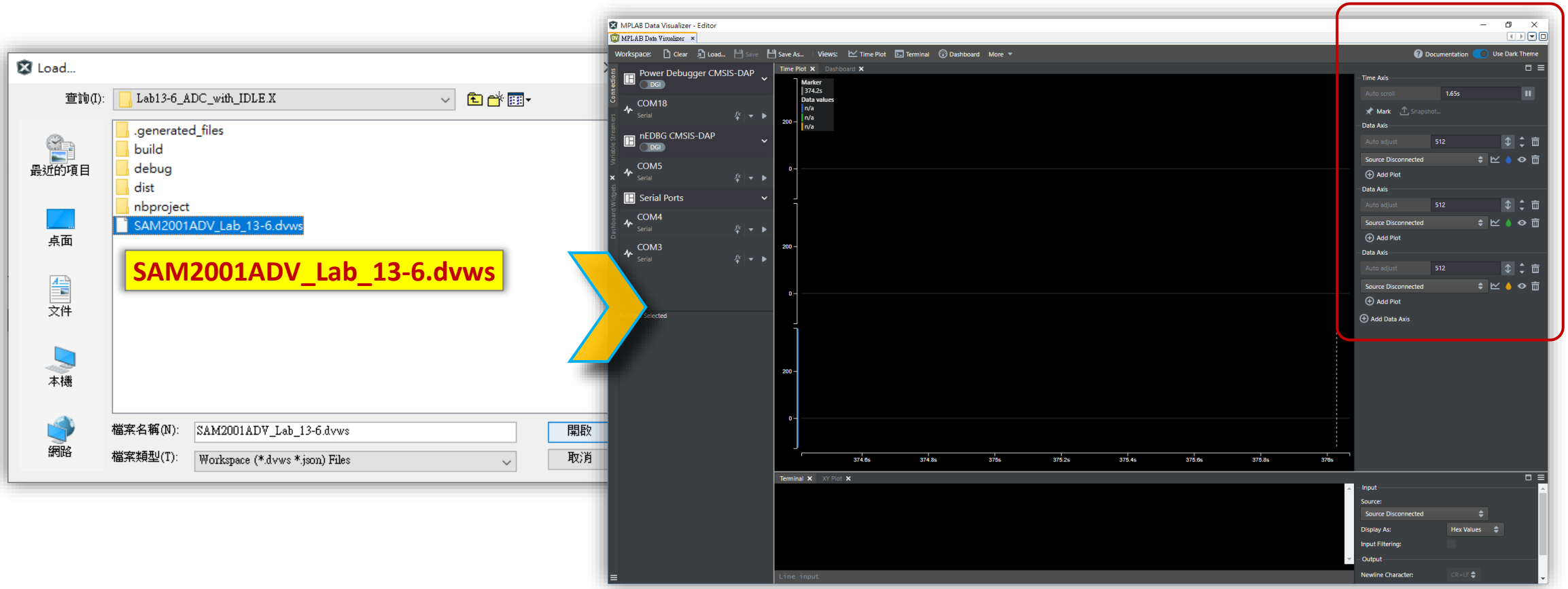
- **Load layout file to reduce setting steps.**



🛠 Lab13-6 ADC with IDLE

Step 7

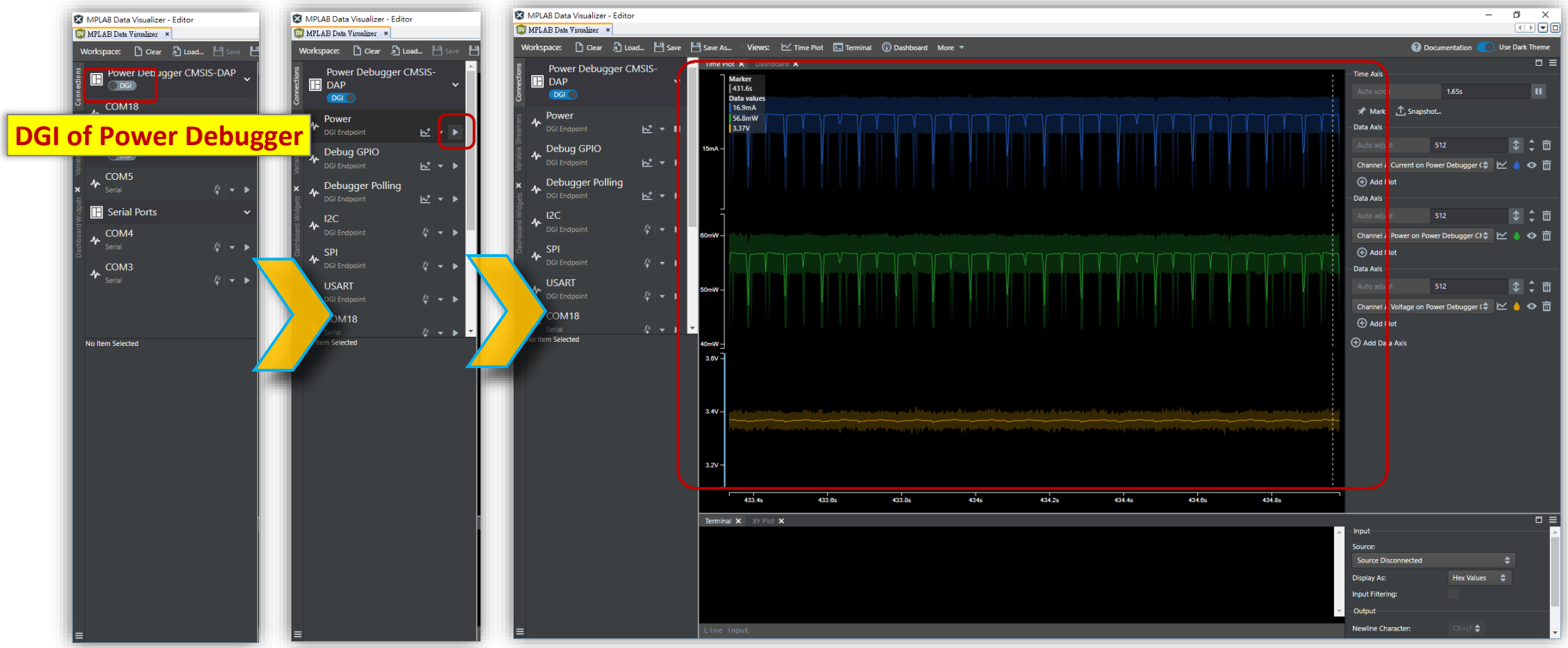
- **Select “SAM2001ADV_Lab_13-6.dvws” layout file. Then data axis be set. Current, Power and Voltage.**



🛠 Lab13-6 ADC with IDLE

Step 8

- Enable Power of DGI left side at Data Visualizer.

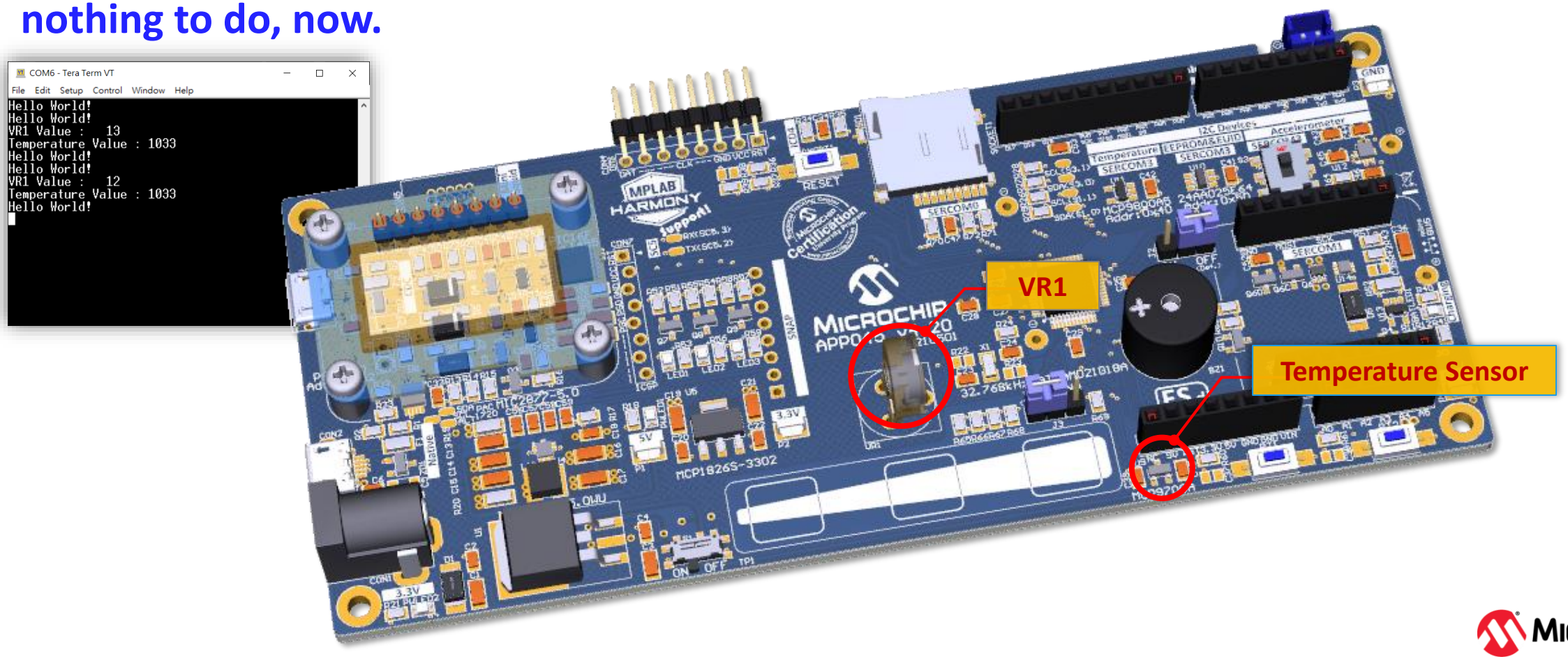


MICROCHIP

🛠 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.

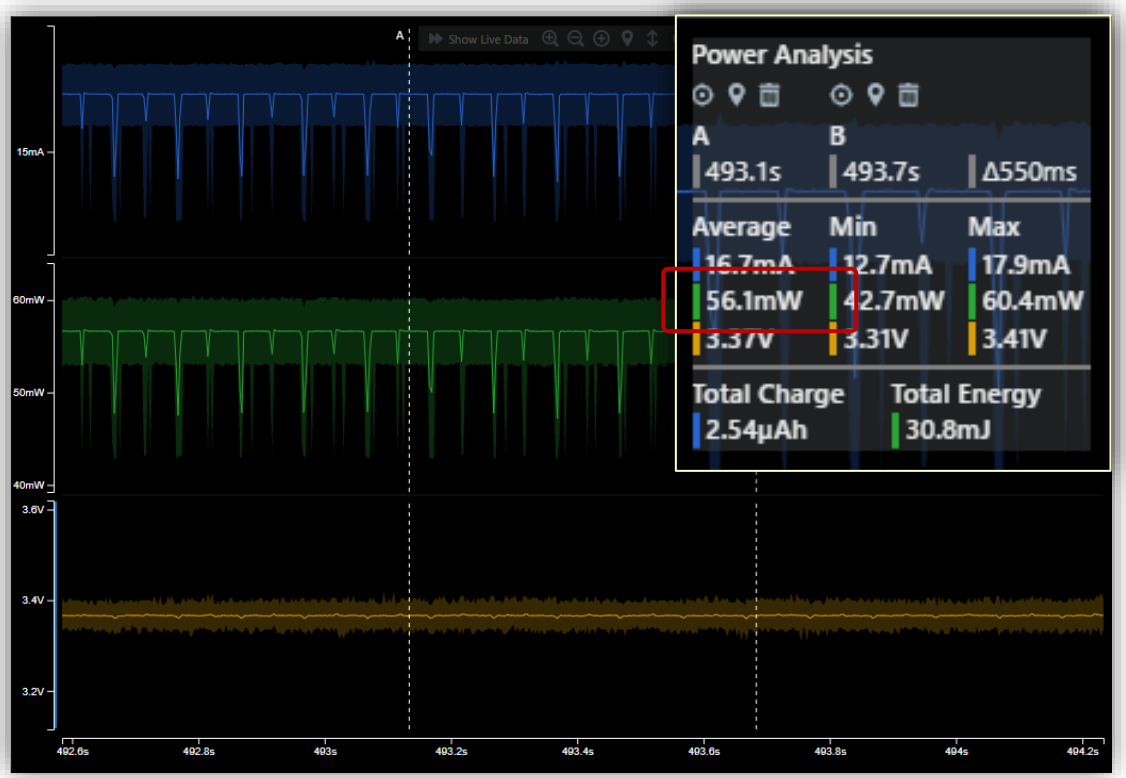


🛠 Lab13-6 ADC with IDLE

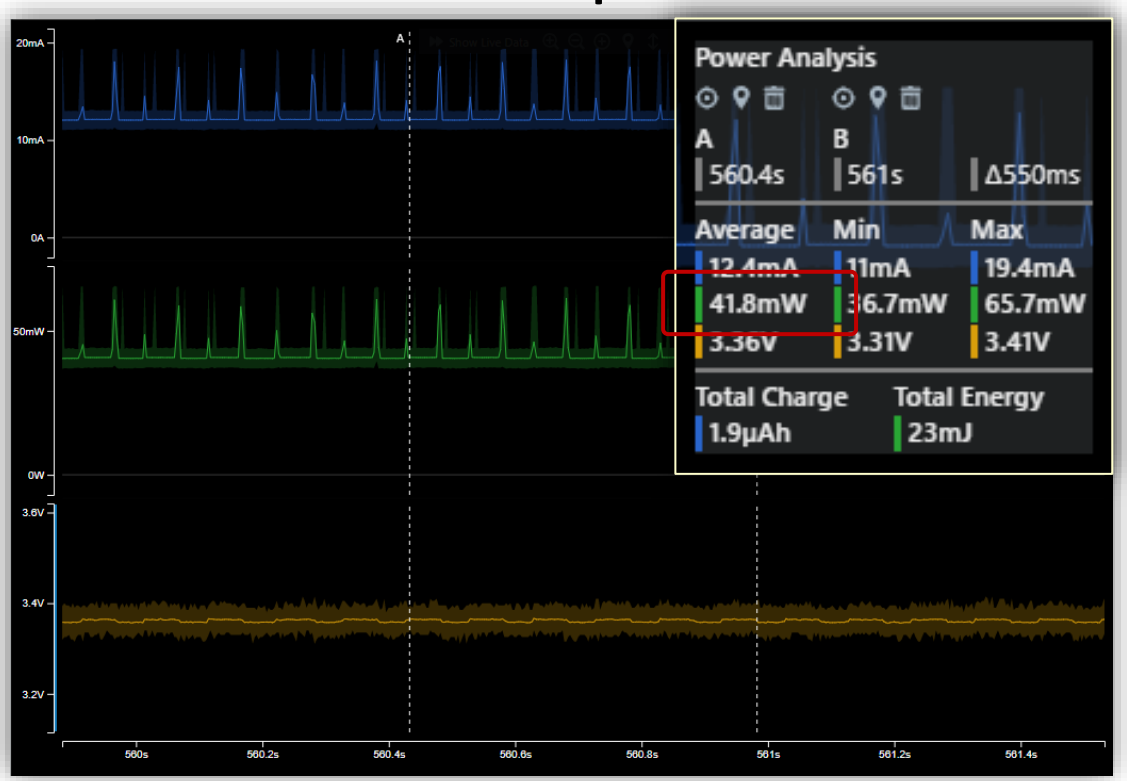
Result

- **Power consumption compared for Active and IDLE0 mode.**

Active Mode



IDLE0 Sleep Mode



MICROCHIP

🛠 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.

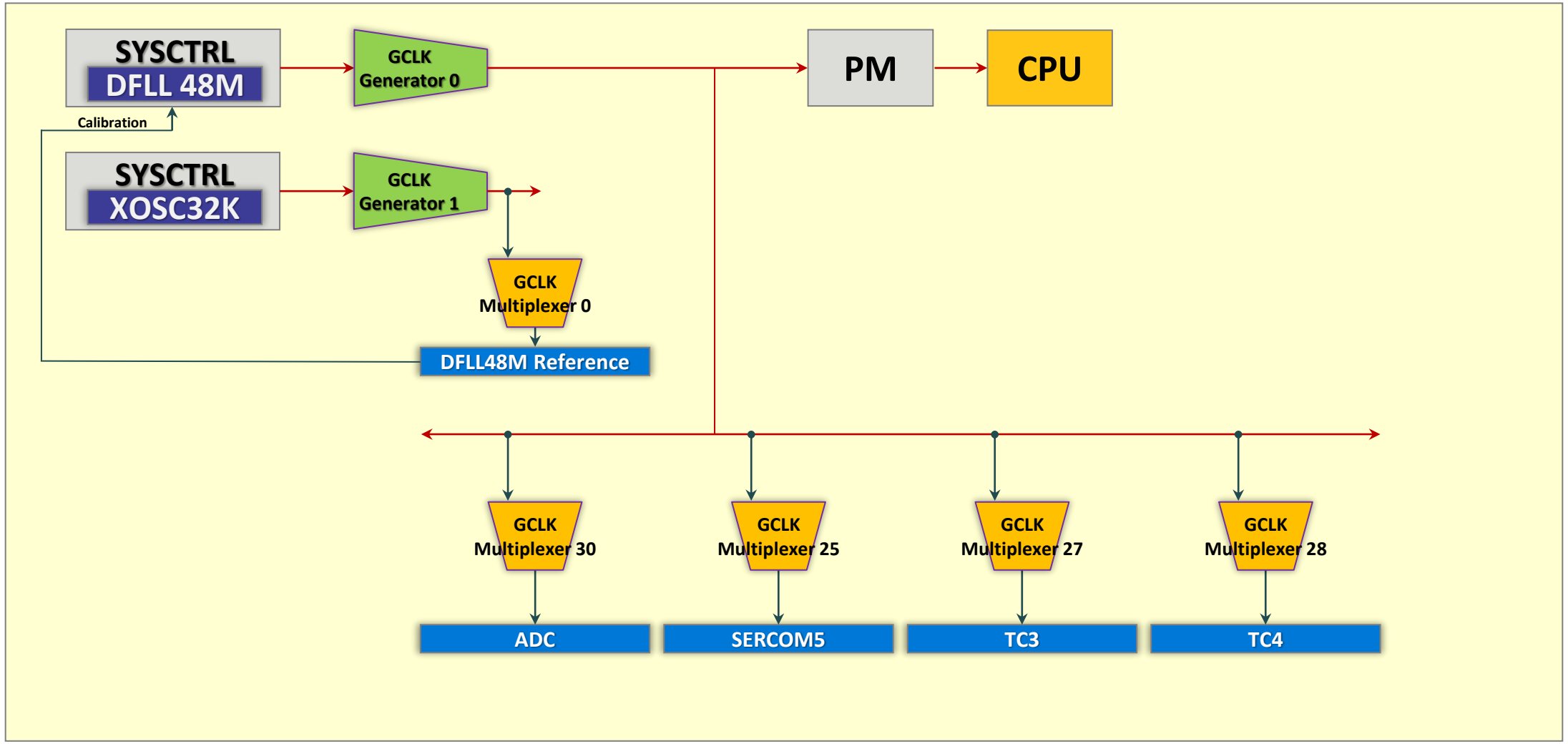
🛠 Lab13-7 ADC with STANDBY

- 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



MICROCHIP

🛠 Lab13-7 ADC with STANDBY

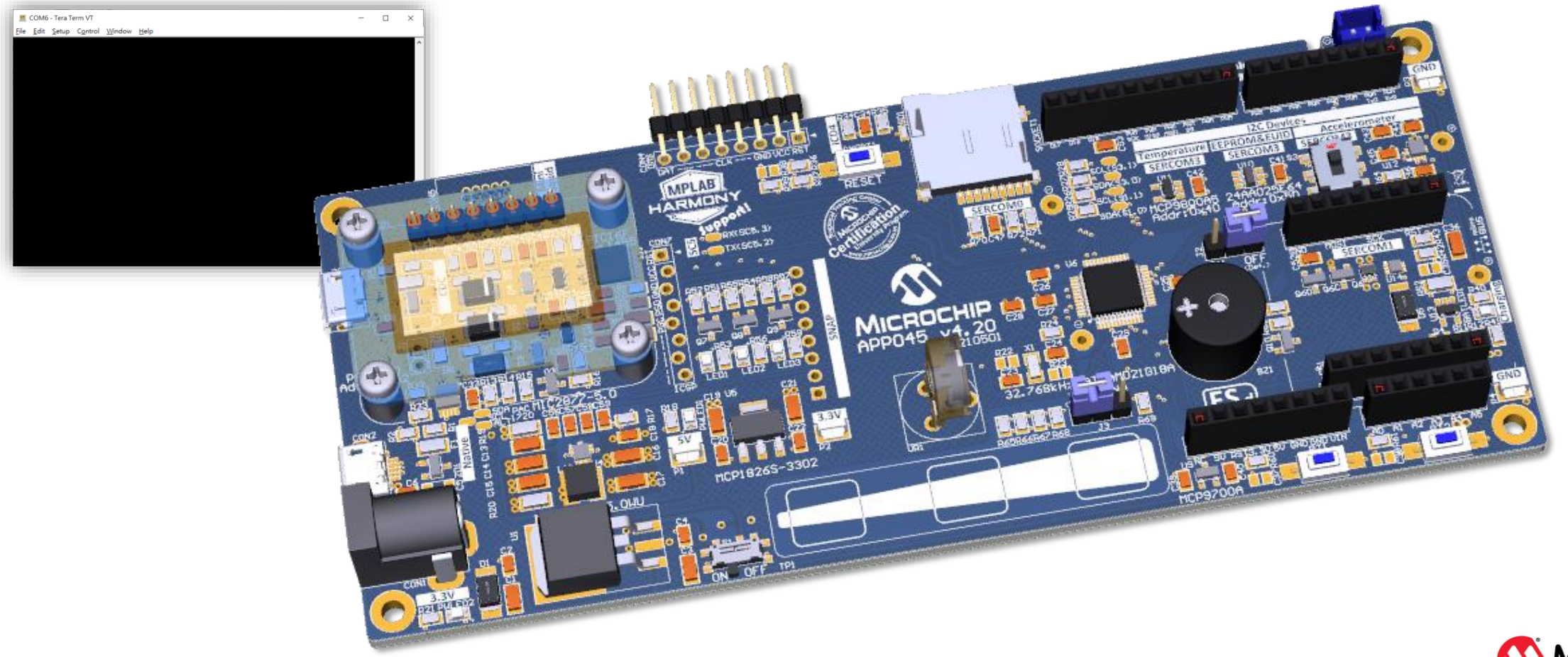
Step 1

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

    while ( true )
    {
        SYS_Tasks();

        // TODO 13-7.01
        //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

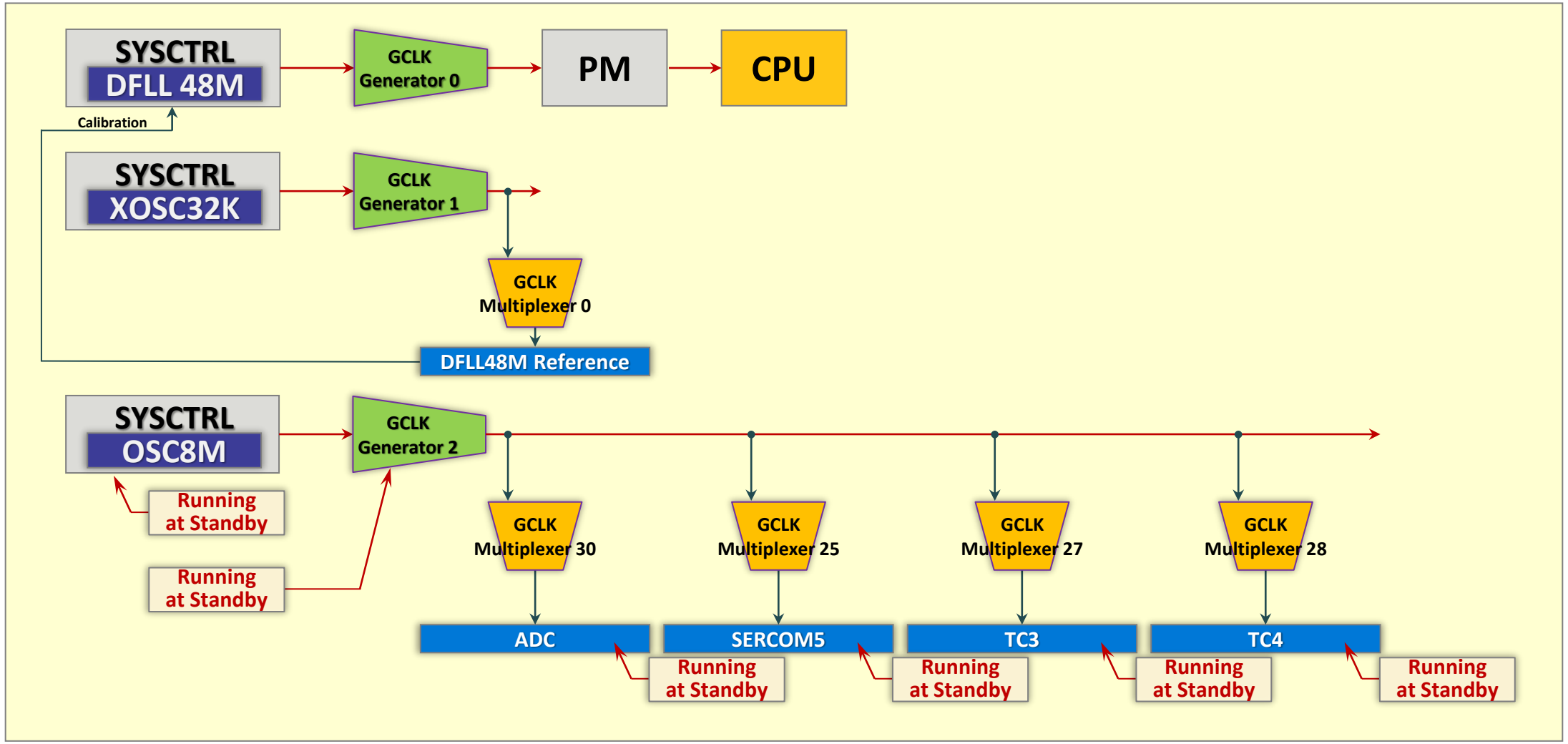
- **Remember !**
  “All clock sources, clock generation and most peripheral are stopped at standby mode, except those where the RUNSTDBY bit is set” And DFLL(DFLL48M) are disable at standby mode, also.

- **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.

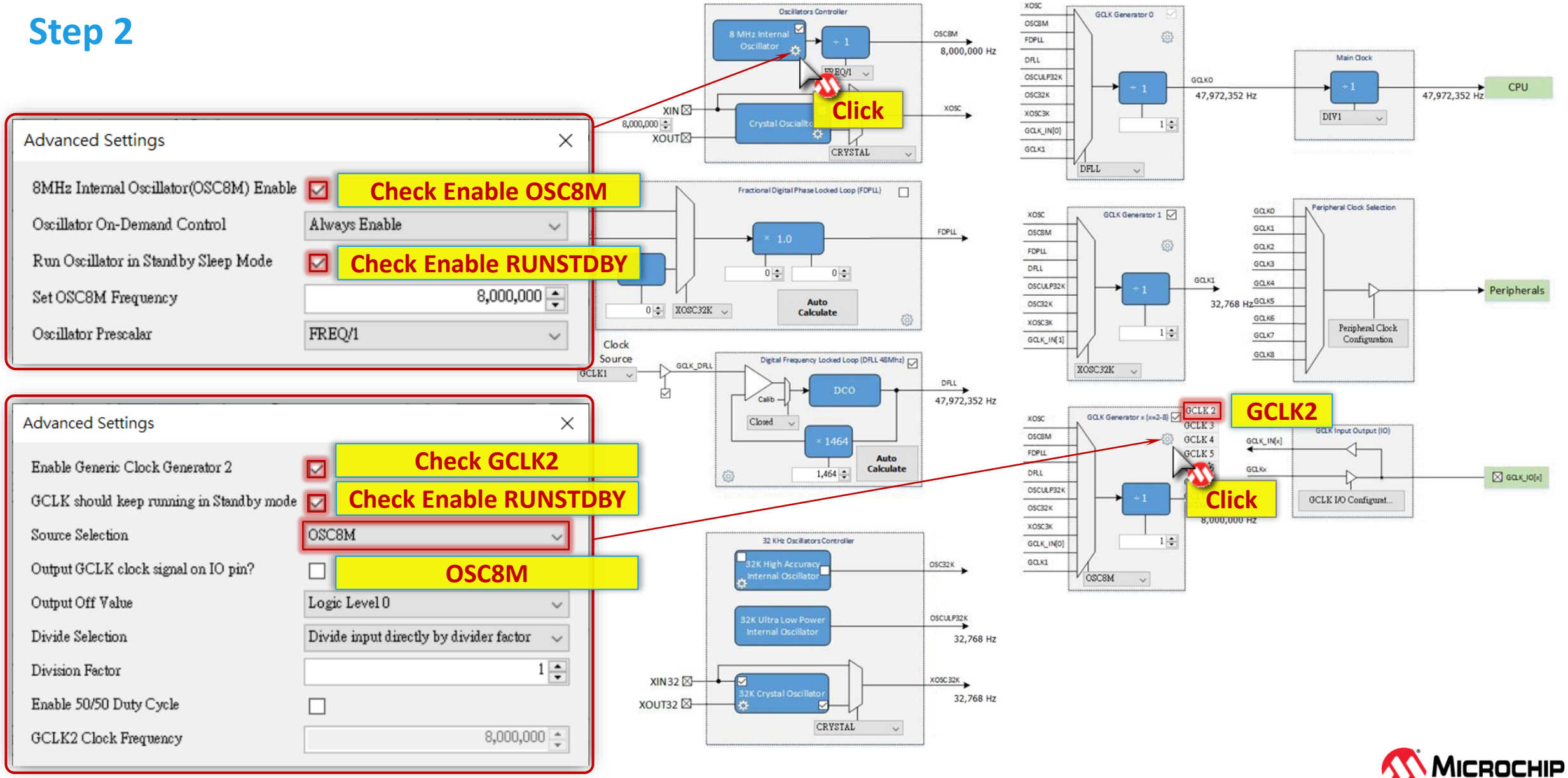
Lab13-7 ADC with STANDBY

- Clock block diagram for Lab 13-7



🛠 Lab13-7 ADC with STANDBY

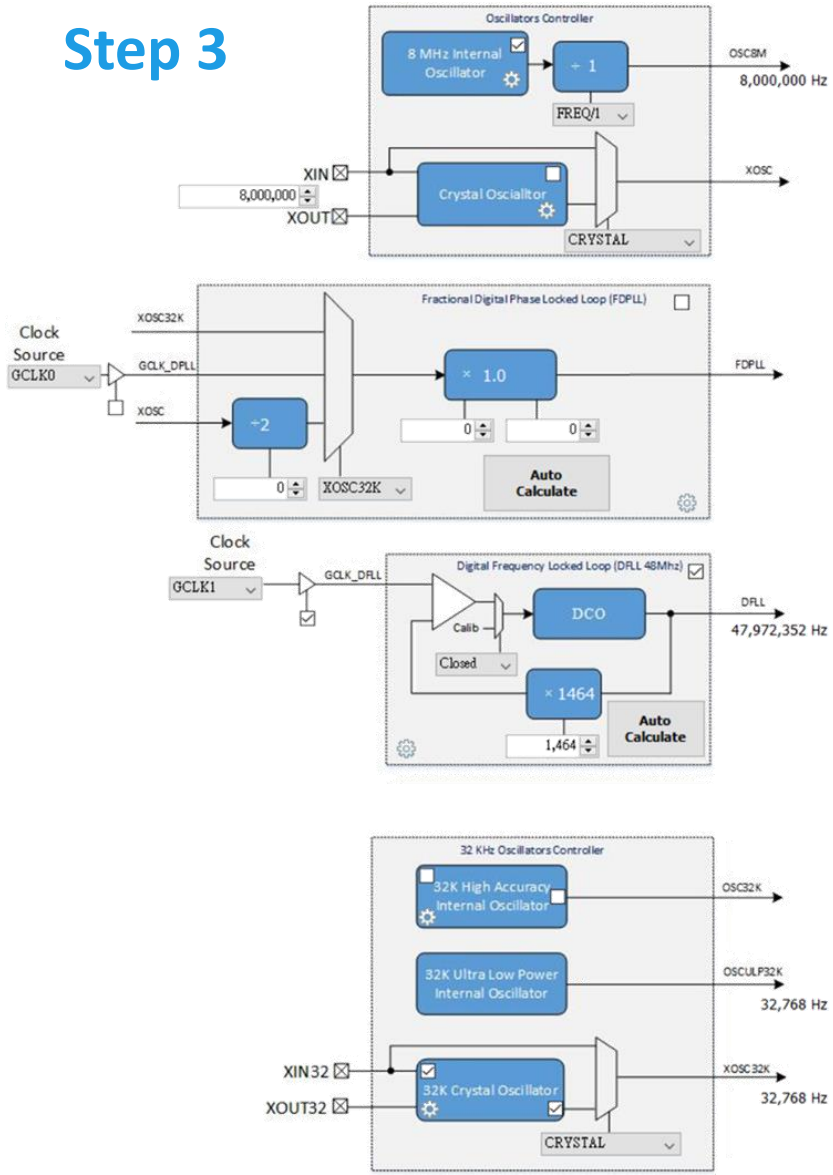
Step 2



MICROCHIP

Lab13-7 ADC with STANDBY

Step 3



Peripheral Clock Configuration

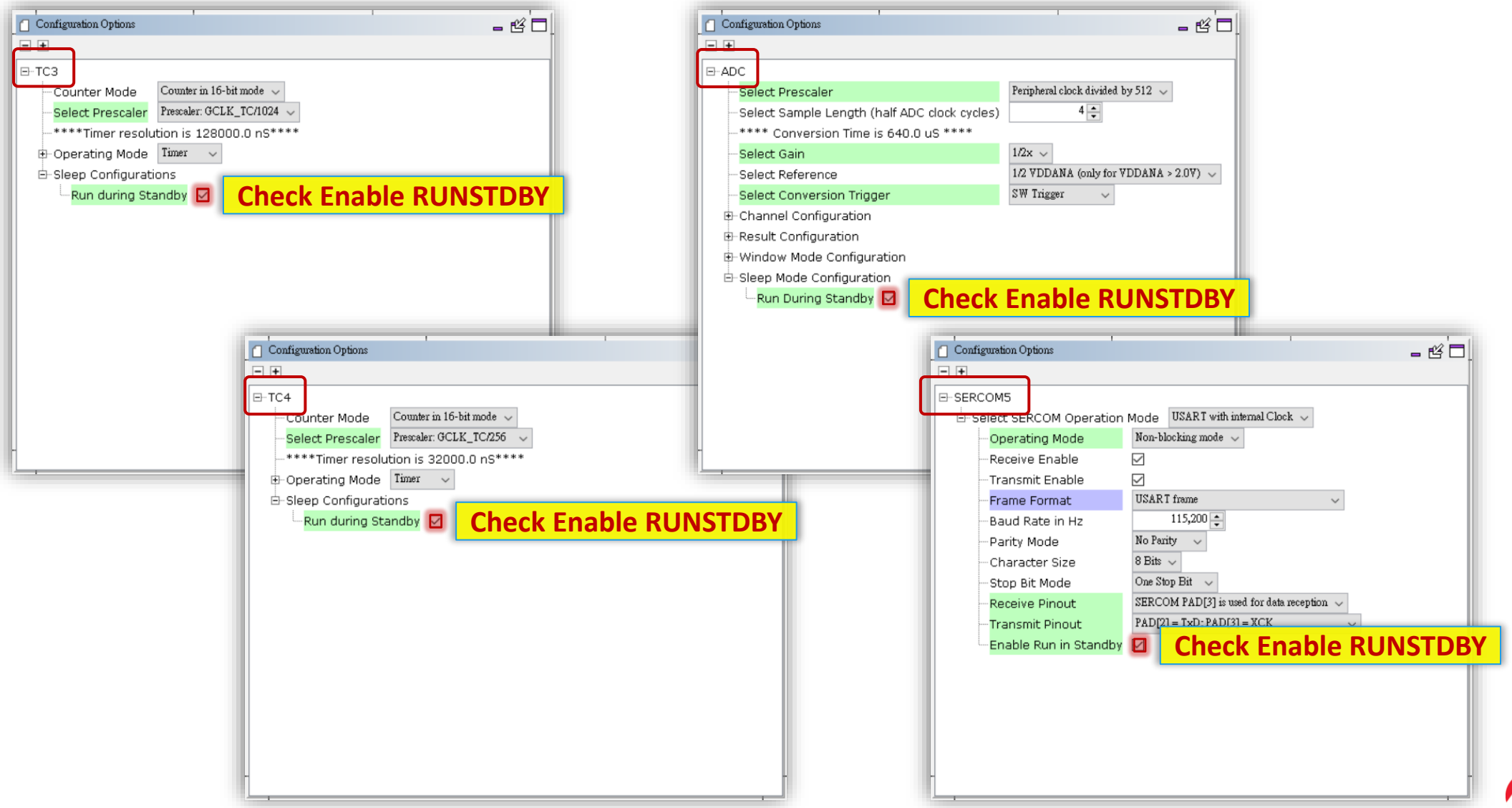
Peripheral	Enable	Source	Peripheral Clock Frequency
AC_ANA	☐	GCLK0	--
AC_DIG	☐	GCLK0	--
ADC	☑	GCLK2	8,000,000 Hz
DAC	☐	GCLK0	--
EIC	☐	GCLK0	--
SERCOM4_CORE	☐	GCLK0	--
SERCOM5_CORE	☑	GCLK2	8,000,000 Hz
SYSCTRL_DFLL48	☑	GCLK1	32,768 Hz
SYSCTRL_FDPLL	☐	GCLK0	--
SYSCTRL_FDPLL32K	☐	GCLK0	--
TC3, TCC2	☑	GCLK2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

Close

MICROCHIP

Lab13-7 ADC with STANDBY

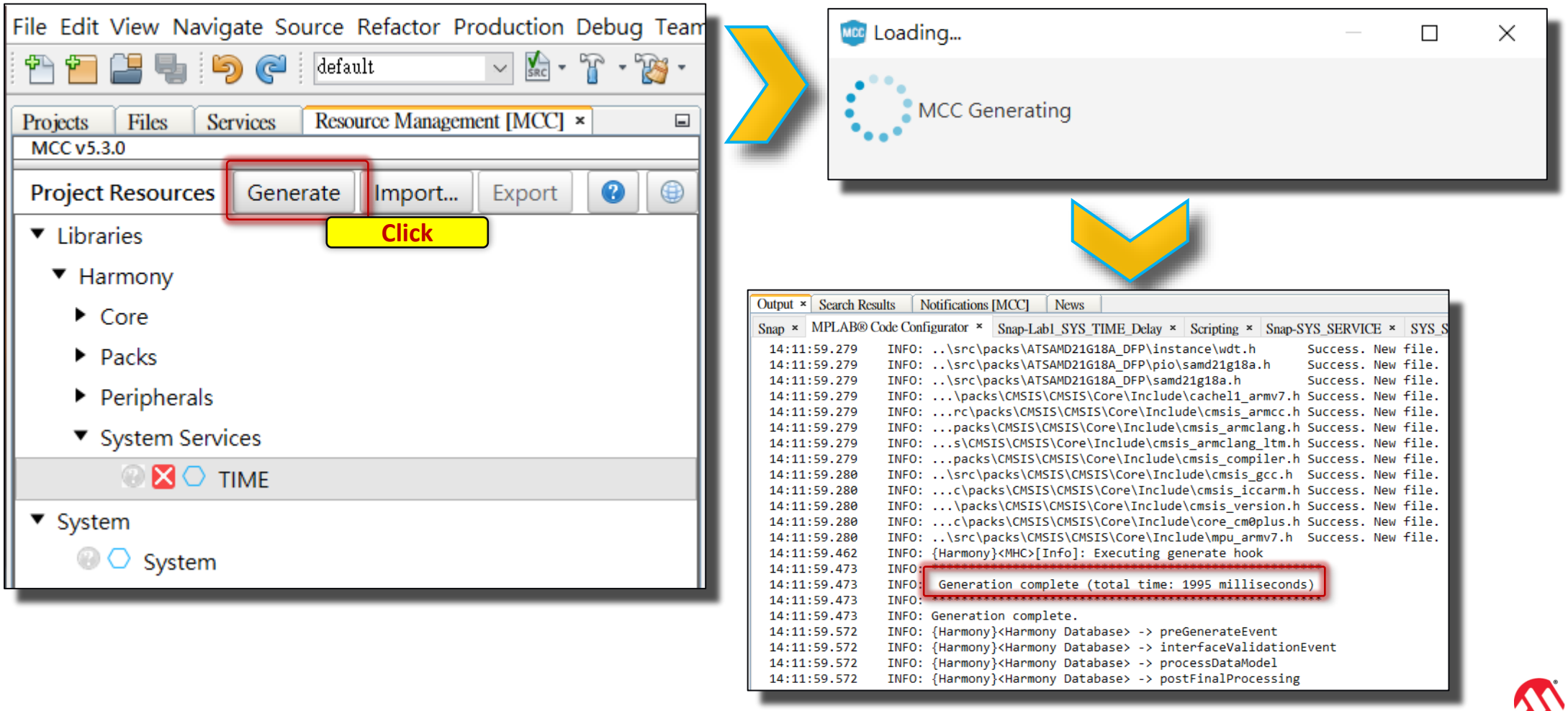
Step 4



🛠 Lab13-7 ADC with STANDBY

Step 5

- **Click to Generate Code.**

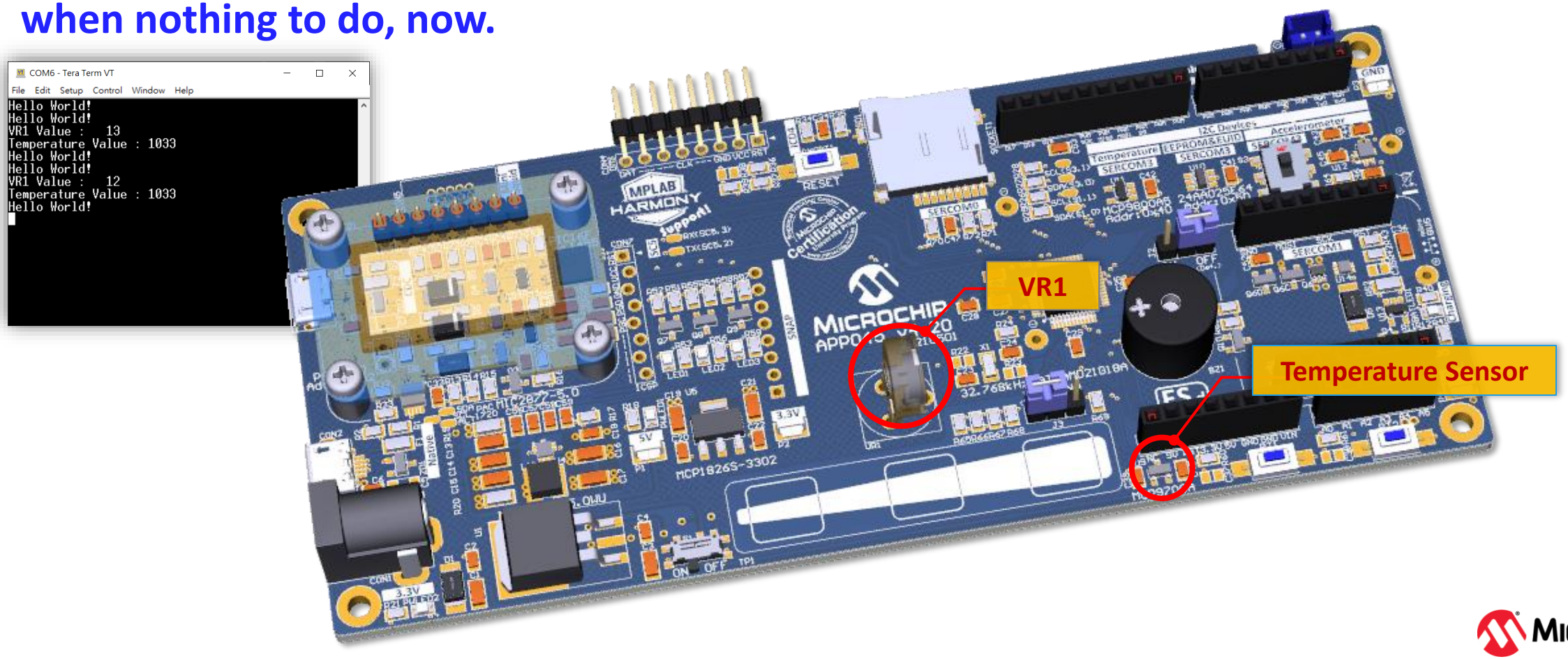


MICROCHIP

🛠 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.



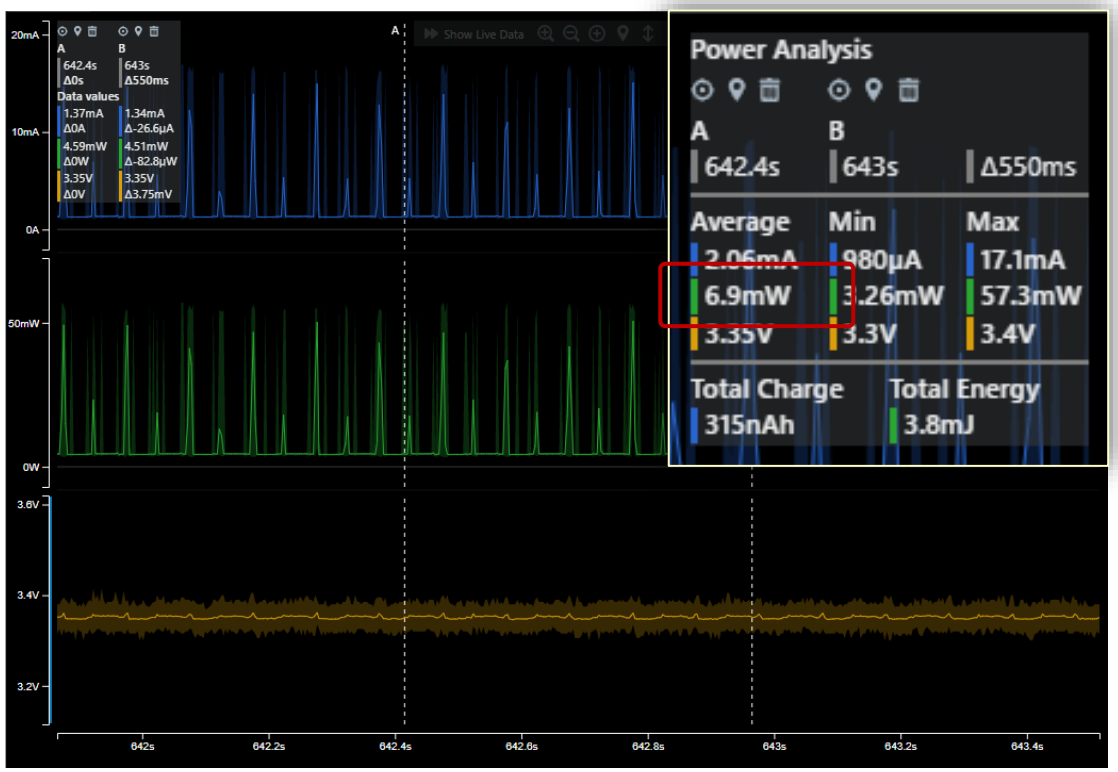
🛠 Lab13-7 ADC with STANDBY

Result

- **Power consumption compared for IDLE0 and STANDBY mode.**

IDLE0 Sleep Mode

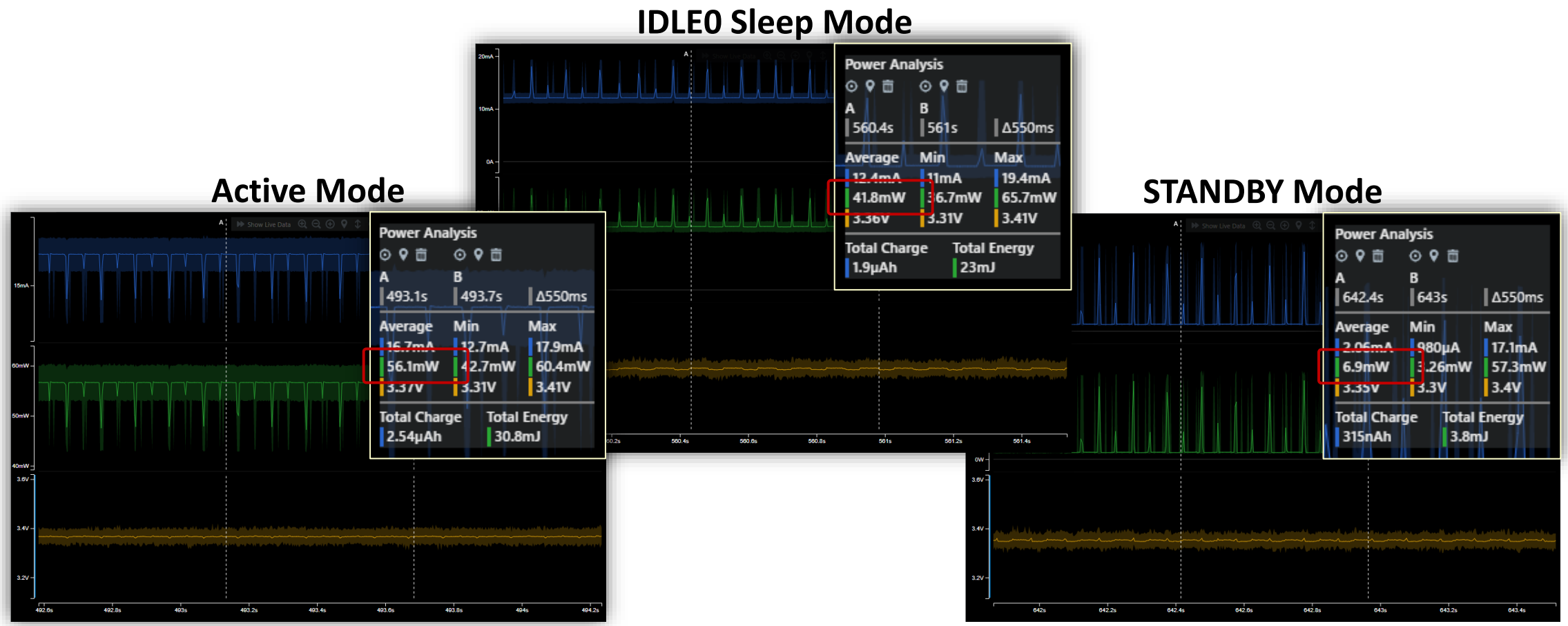
STANDBY Mode



MICROCHIP

🛠 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

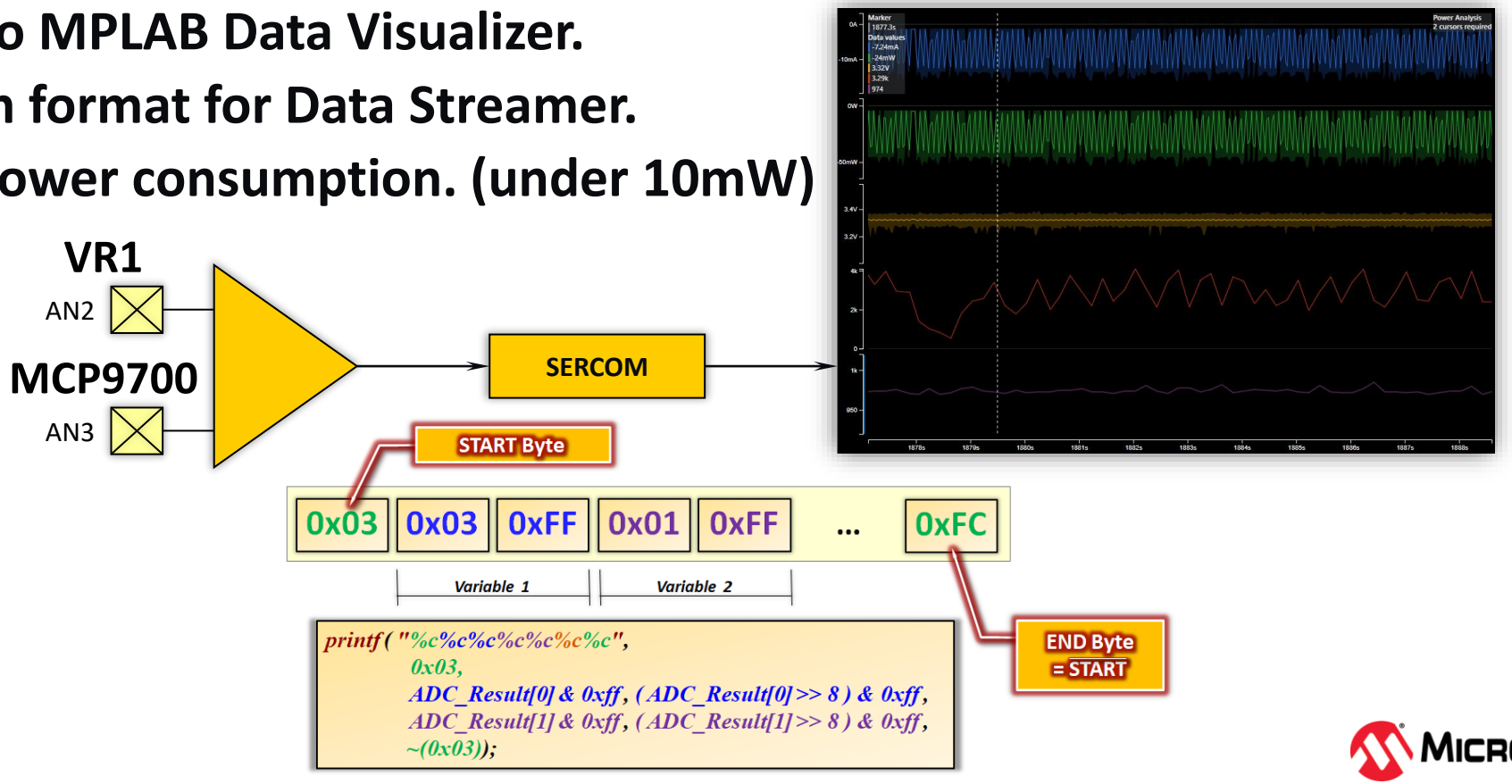
- There are two thing need to know when entry STANDBY sleep mode,
  - DMAC be disable at IDLE1, IDLE2 and STANDBY sleep mode except IDLE0.
    - DMAC clock by AHB clock.
    - So, must execute "wakes-up“ before DMAC action.
    - CPU be wakes-upped, also.
    - wakes-up CPU will resume all clocks, bus and peripherals.

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

  - For Generally, don’t enter sleep mode when running ISR.
    - When MCU enter sleep mode on ISR running, NIVC 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".

Application Require

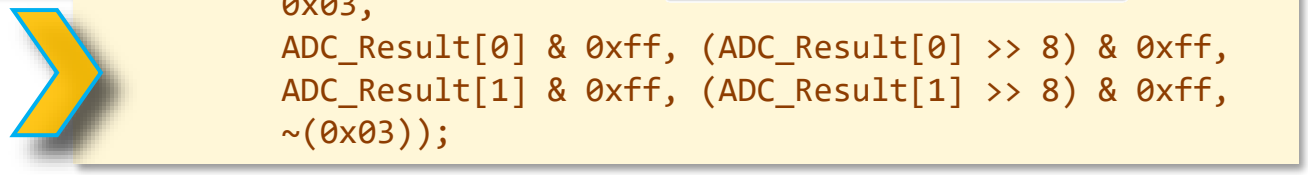
- At this section, we try to create an application to demo SleepWalking feature.
- The application require list at below,
  - To Convert VR1 and MCP9700 analog voltage
  - Based on ADC's Pin-Scan mode and sample time control by TC3.
  - Output the results to MPLAB Data Visualizer.
    - follow data stream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

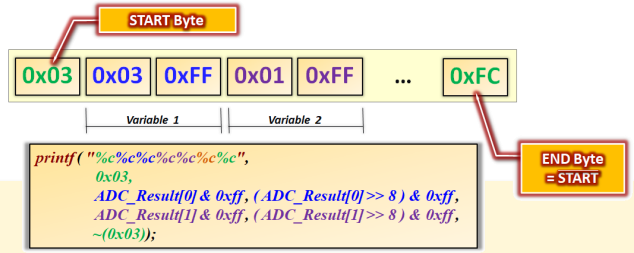


🛠 Lab 13-8 ADC with DV

- **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]);
```





- **Let's go !**

MICROCHIP

🛠 Lab 13-8 ADC with DV

Step 1

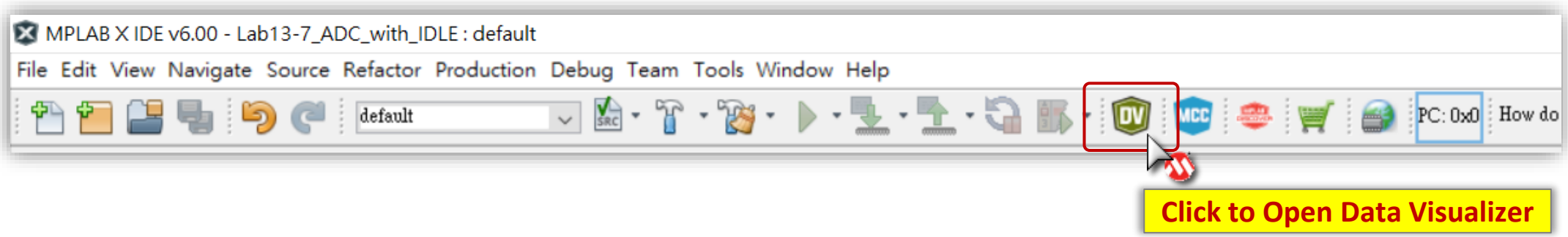
```
int main (void)
{
    if (TC3_HasExpired)
    {
        // TODO 13-8.01
        //myprintf("Hello World!\r\n");
    }
    if (USART5_IsReceived)
    {
        // TODO 13-8.02
        //myprintf("\r\nReceived Data : %1c\r\n", USART5_ReceiveData[0]);
    }
    if (ADC_IsCompleted)
    {
        //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

MICROCHIP

🛠 Lab 13-8 ADC with DV

Step 2

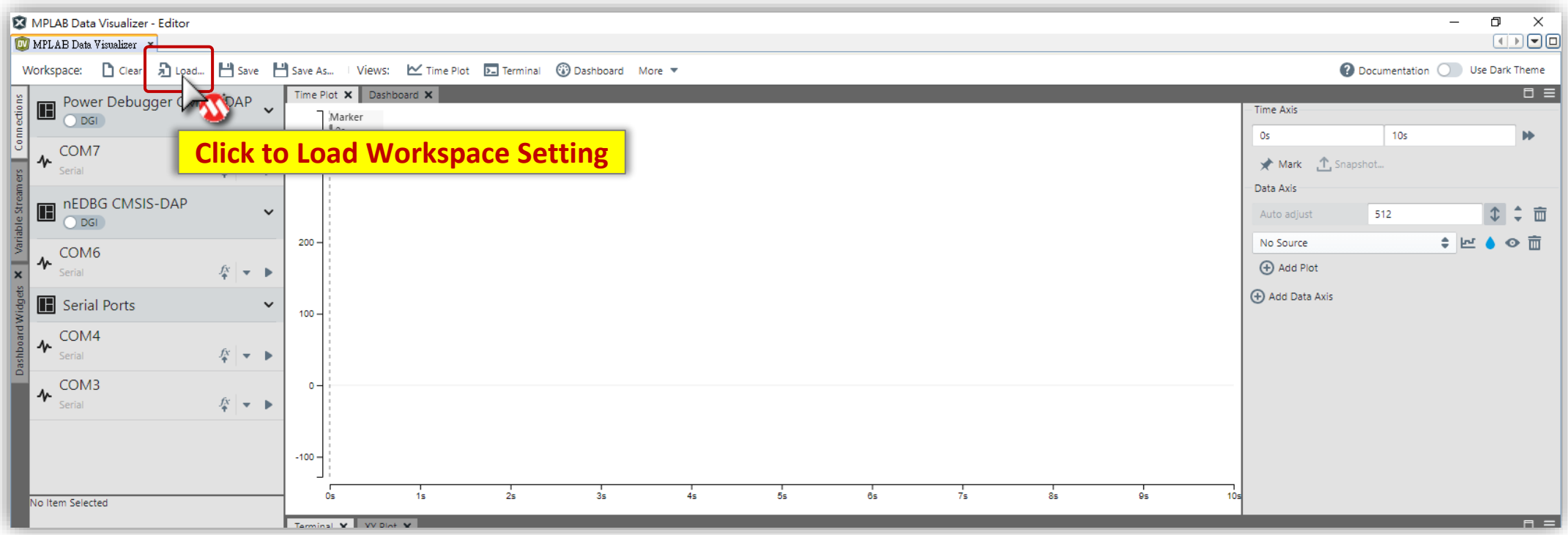
- **Open Data Visualizer by icon at MPLAB X IDE**



🛠 Lab 13-8 ADC with DV

Step 3

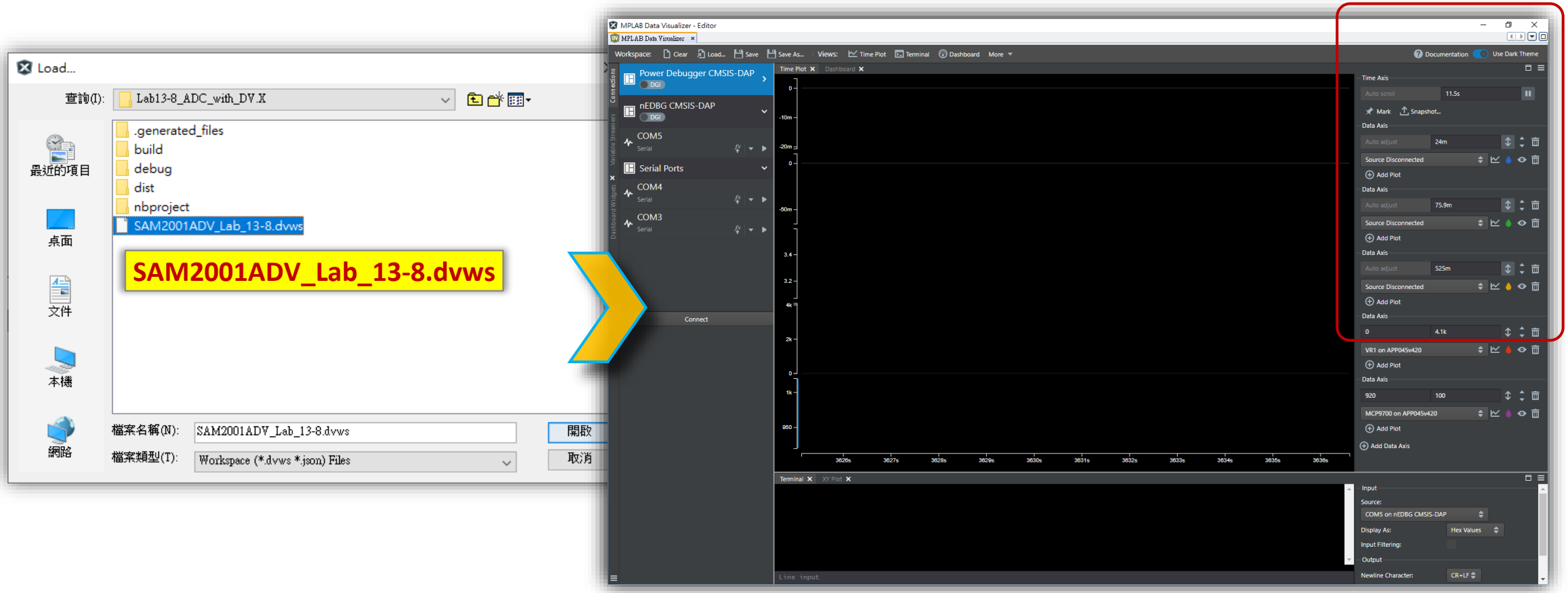
- **Load layout file to reduce setting steps.**



🛠 Lab 13-8 ADC with DV

Step 4

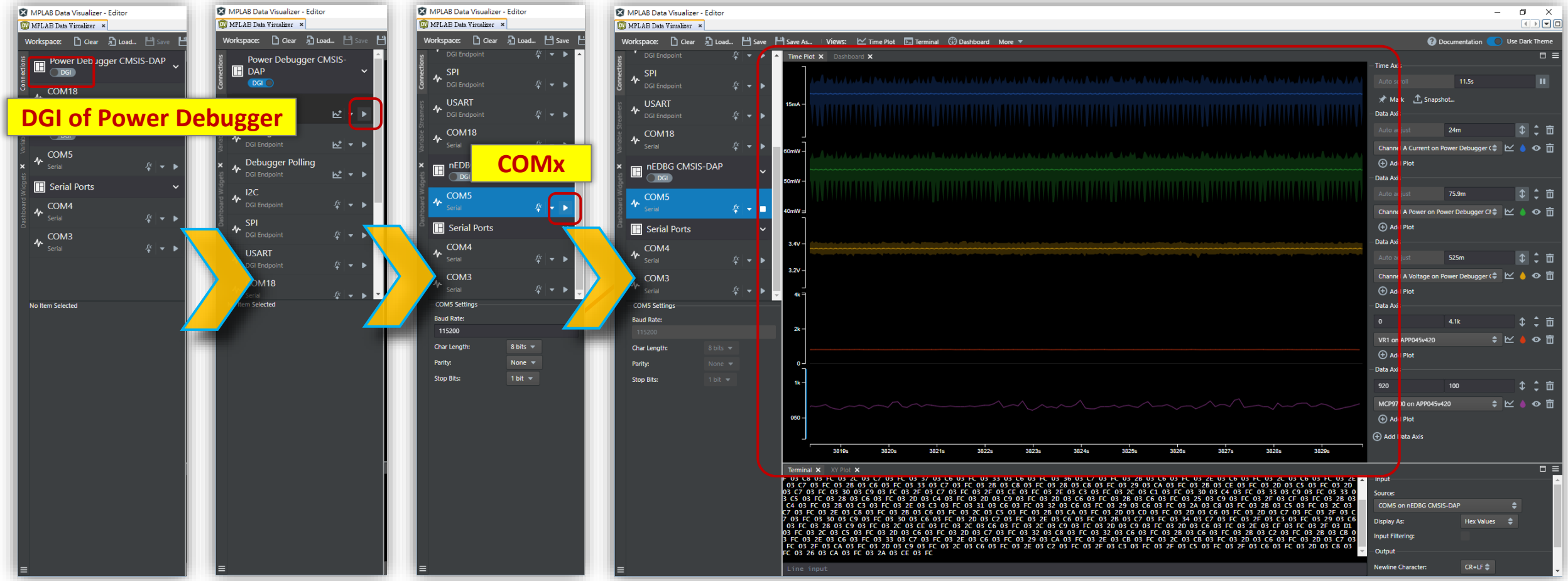
- **Select “SAM2001ADV_Lab_13-8.dvws” layout file. Then data axis be set. Current, Power, Voltage, VR1 and MCP9700.**



🛠 Lab 13-8 ADC with DV

Step 5

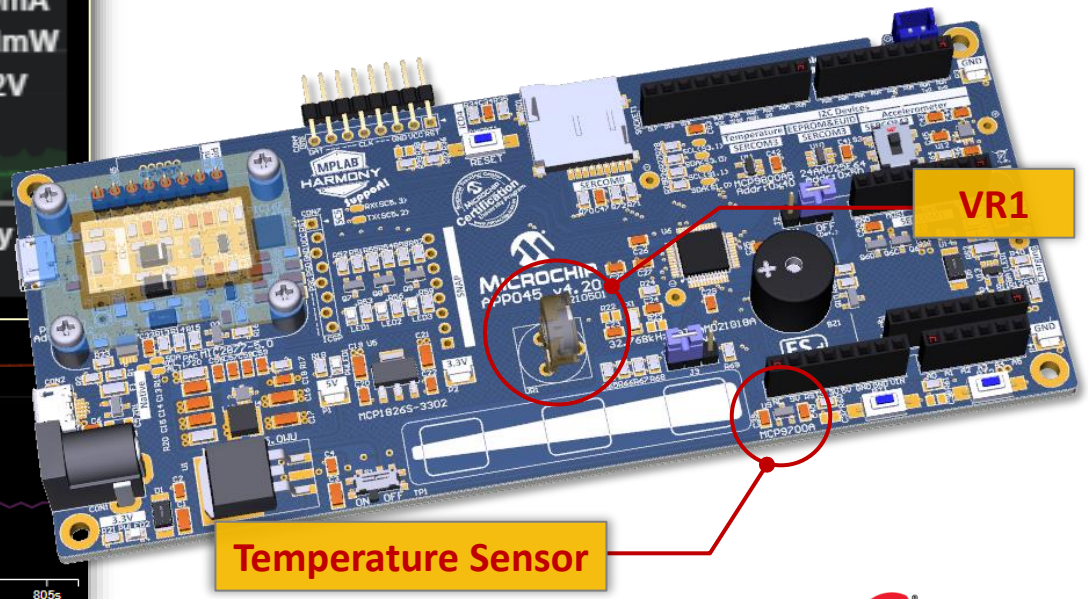
- Enable **Power** of **DGI** and **COMx** left side at Data Visualizer.



🛠 Lab 13-8 ADC with DV

Result

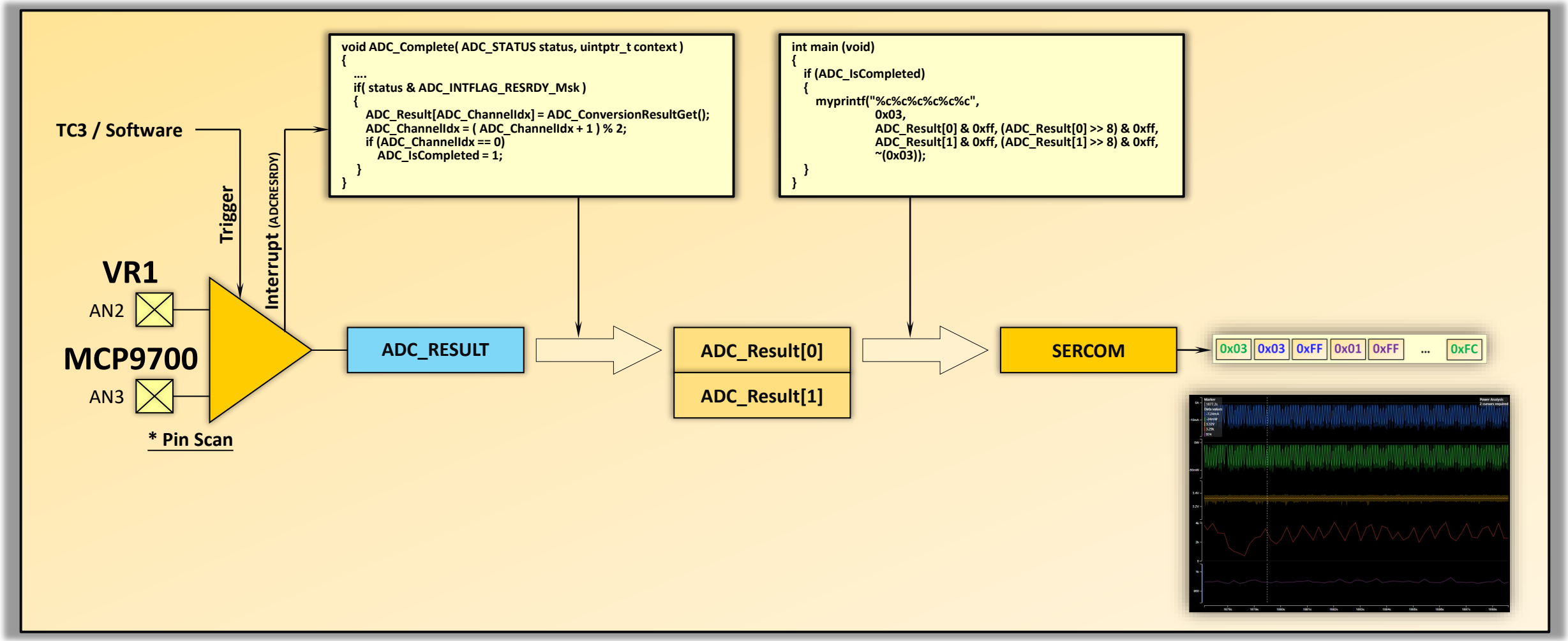
- **ADC result of VR1 / Temperature Sensor will output to DV with to second period. Same as SAM2001 Lab 13. But output follow data stream format.**



MICROCHIP

🛠 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 ?

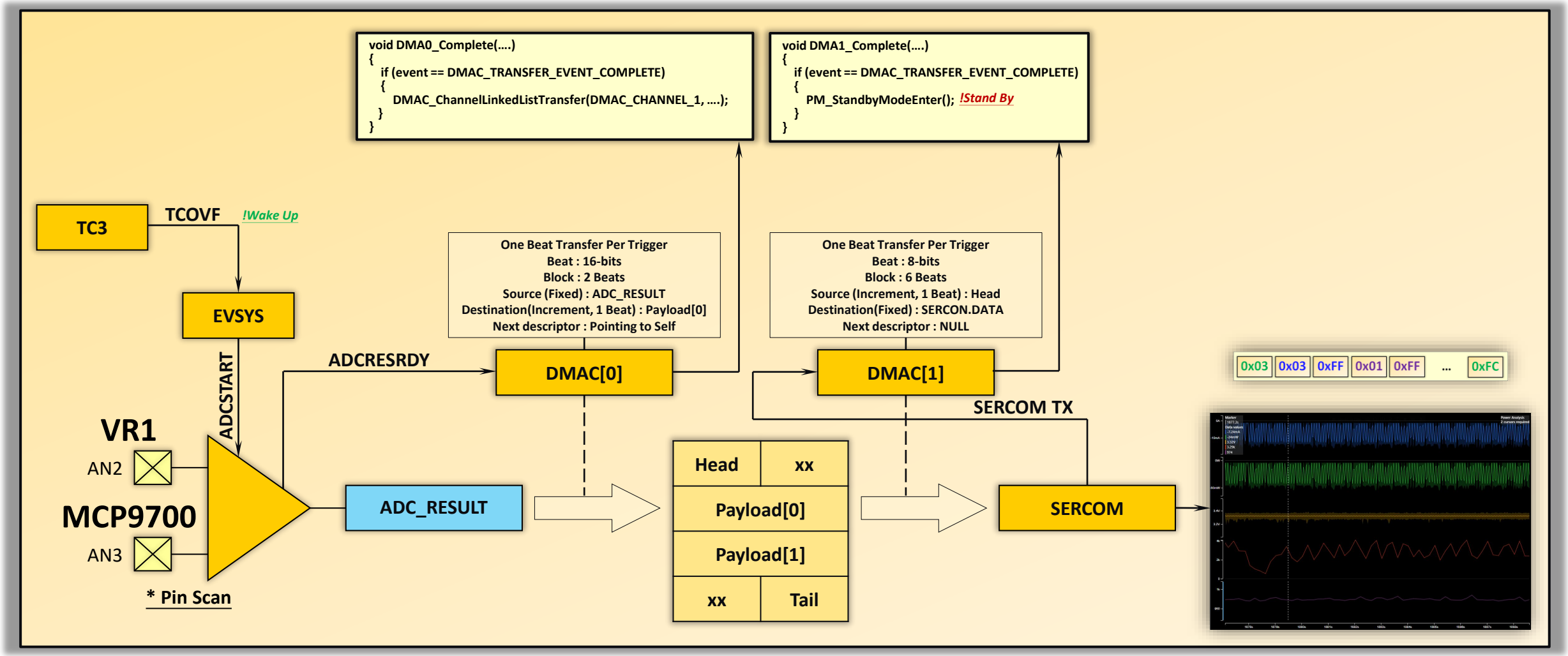
🛠 Lab13-9 ADC Scan DV DMAC SleepWalking

- 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 !**

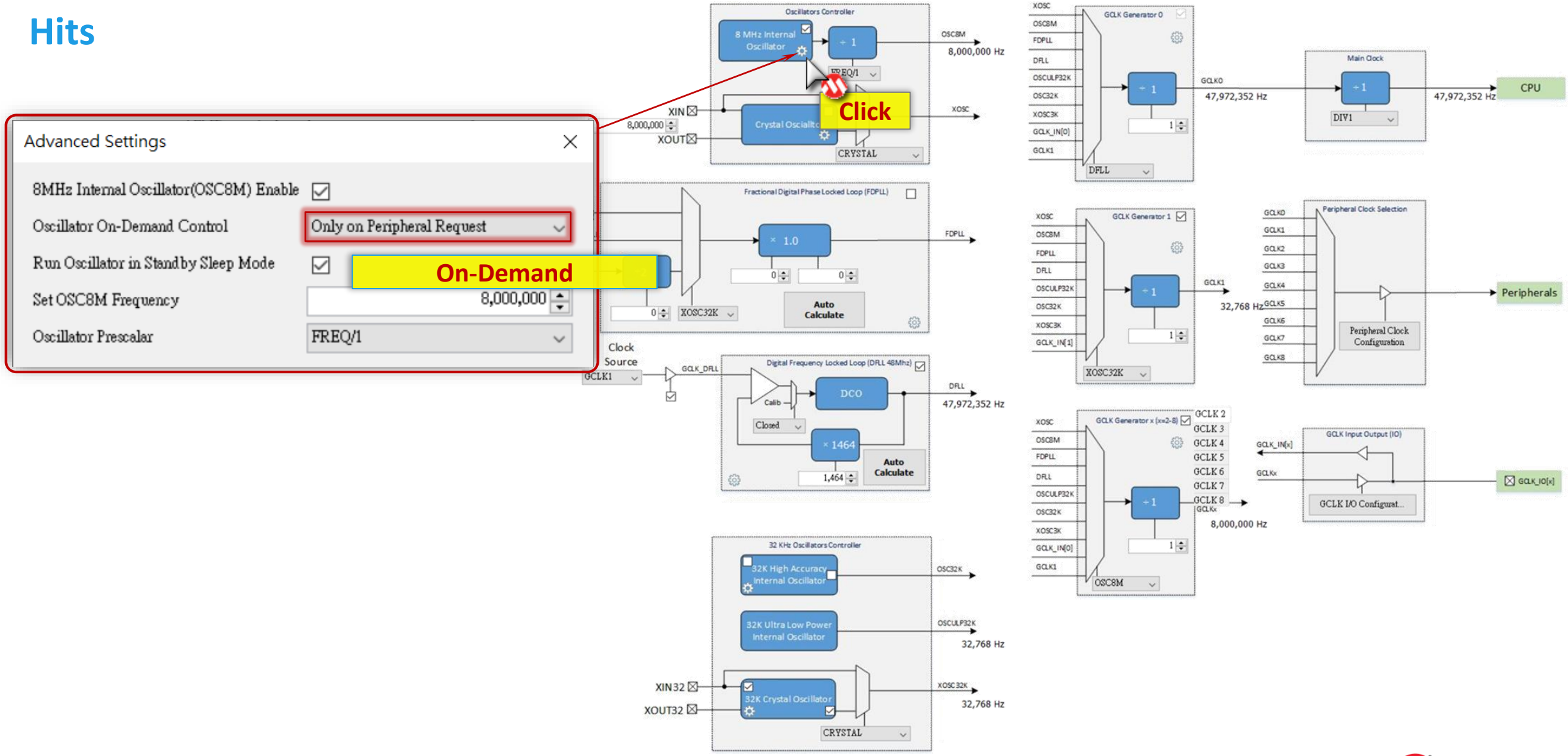
🛠 Lab13-9 ADC Scan DV DMAC SleepWalking

- **Apply new process to reduce CPU loading.**



🛠 Lab13-9 ADC Scan DV DMAC SleepWalking

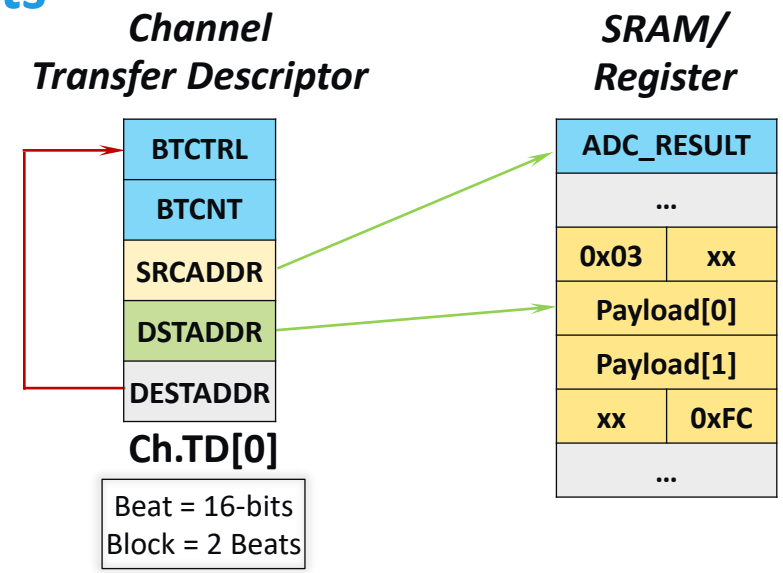
Hits



MICROCHIP

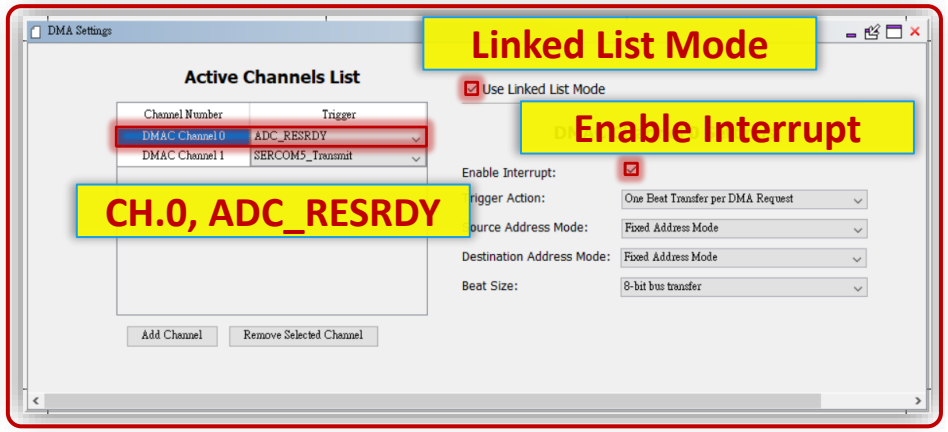
🛠 Lab13-9 ADC Scan DV DMAC SleepWalking

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]));
```

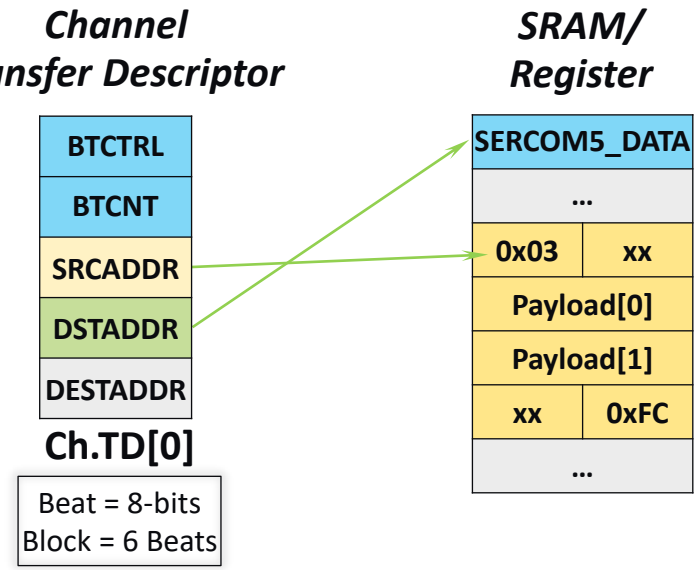
Linked List, Continuously Event



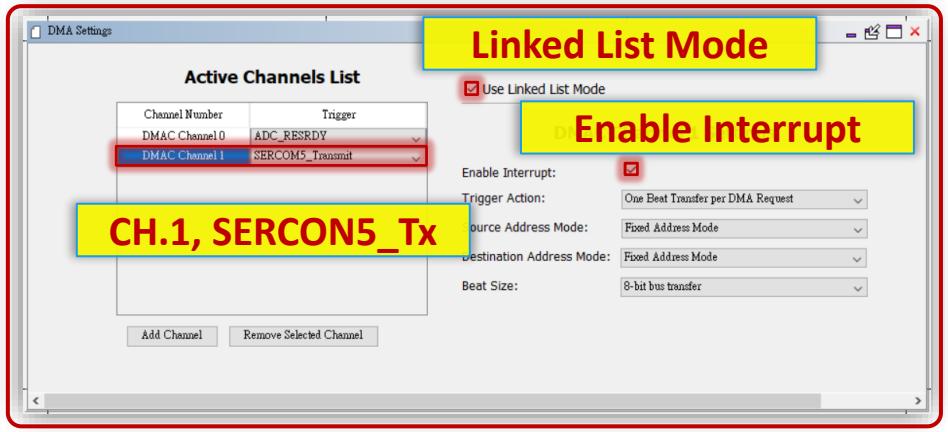
MICROCHIP

🛠 Lab13-9 ADC Scan DV DMAC SleepWalking

Hits



Link NULL, One-Shot Event

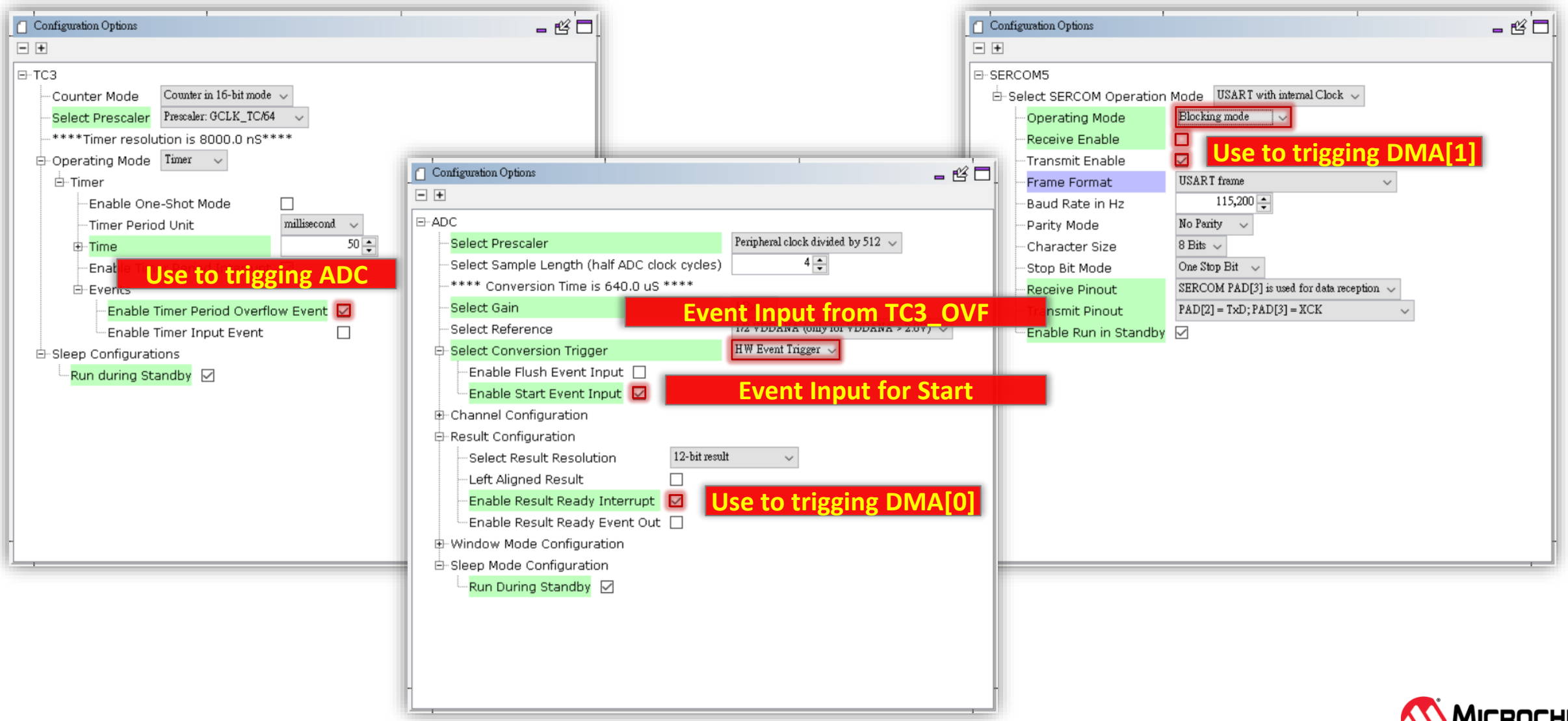



```
typedef struct
{
   uint8_t:8;
   uint8_t head;
   uint16_t Payload[2];
   uint8_t tail;
   uint8_t:8;

} 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);
```

MICROCHIP

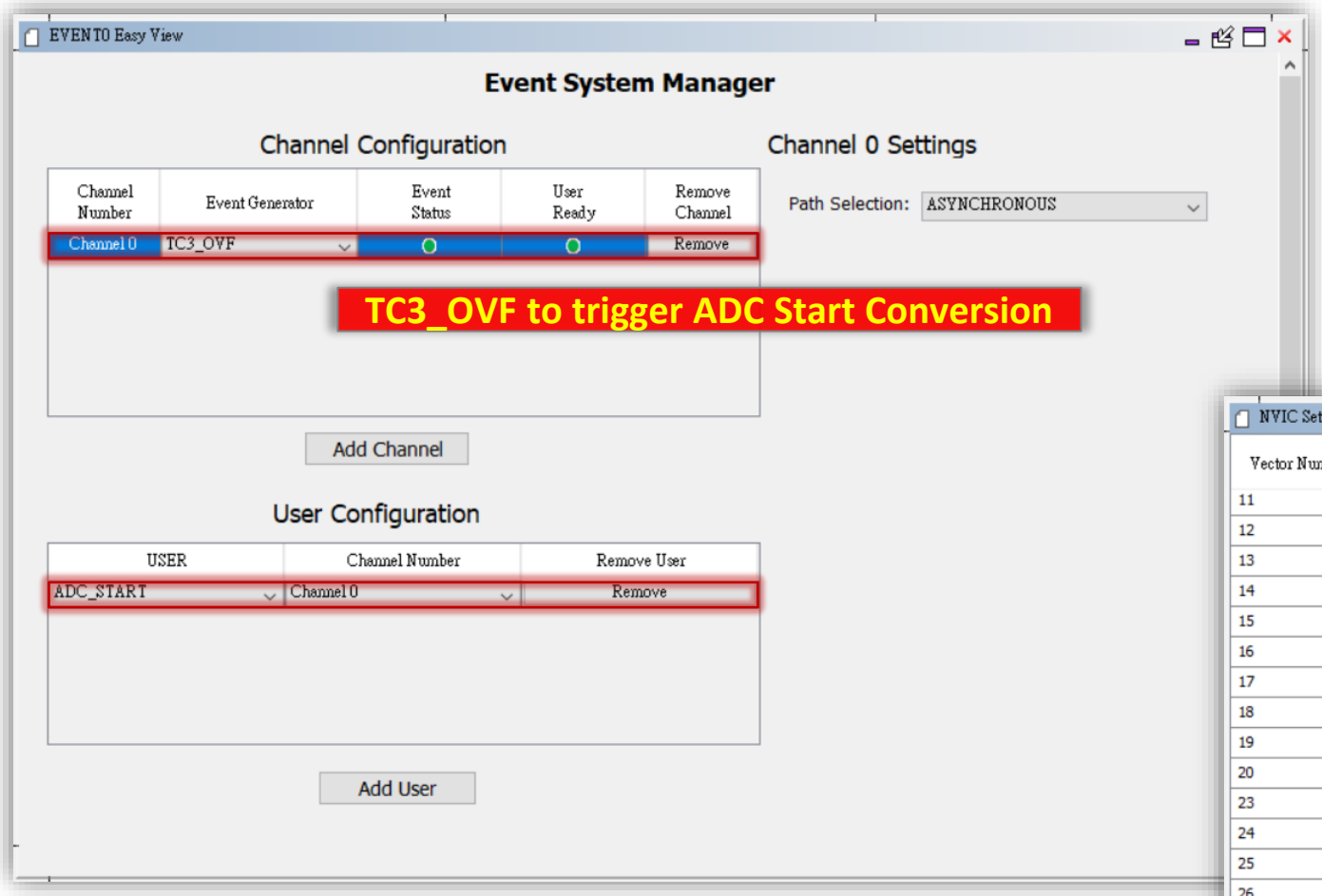
🛠 Lab13-9 ADC Scan DV DMAC SleepWalking

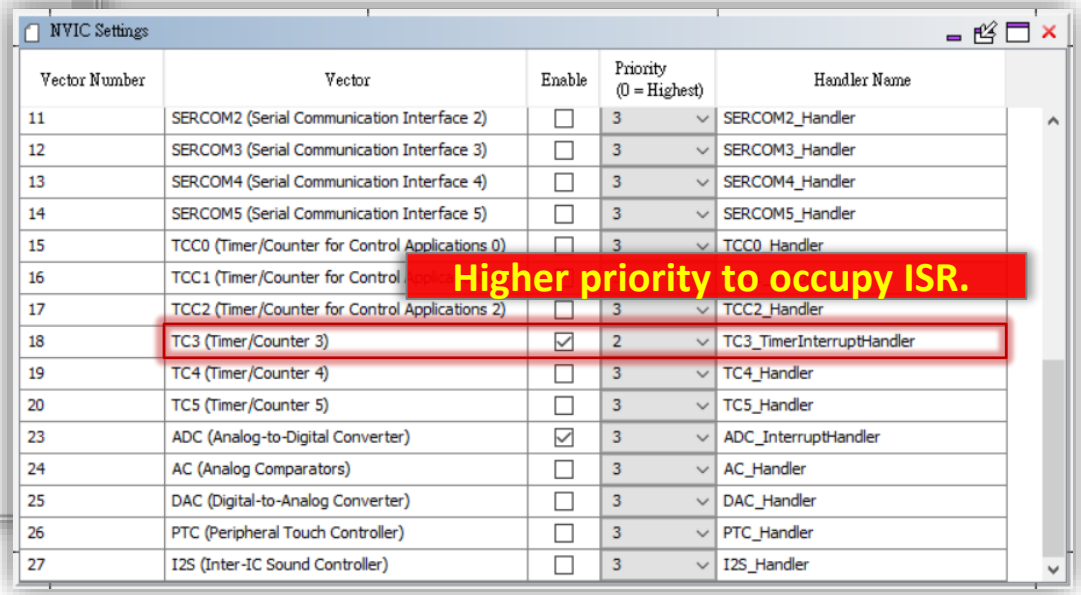
Hits



🛠 Lab13-9 ADC Scan DV DMAC SleepWalking

Hits



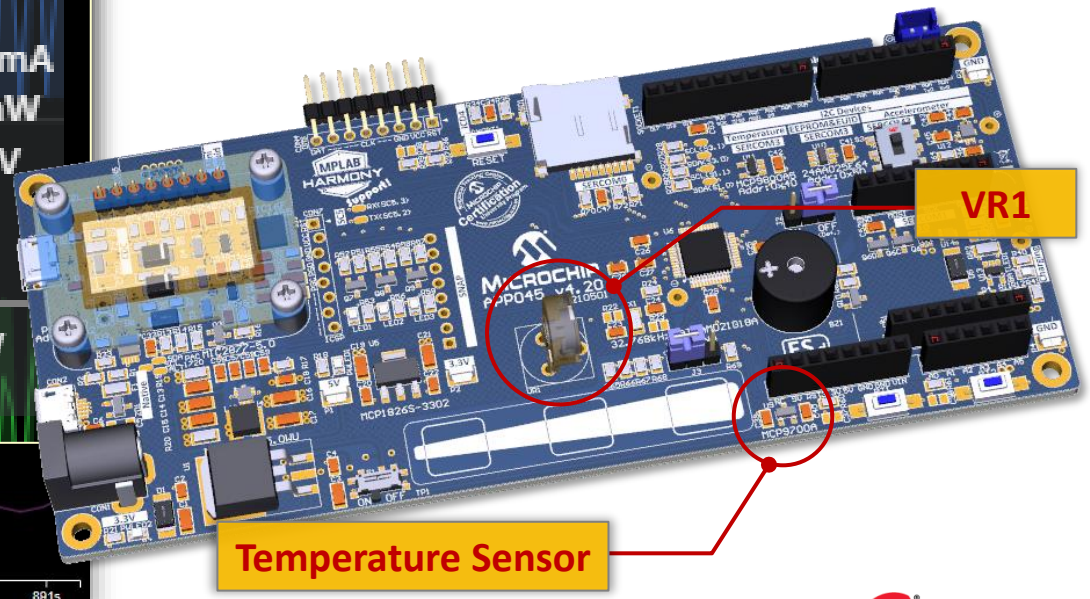


MICROCHIP

🛠 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.**



MICROCHIP

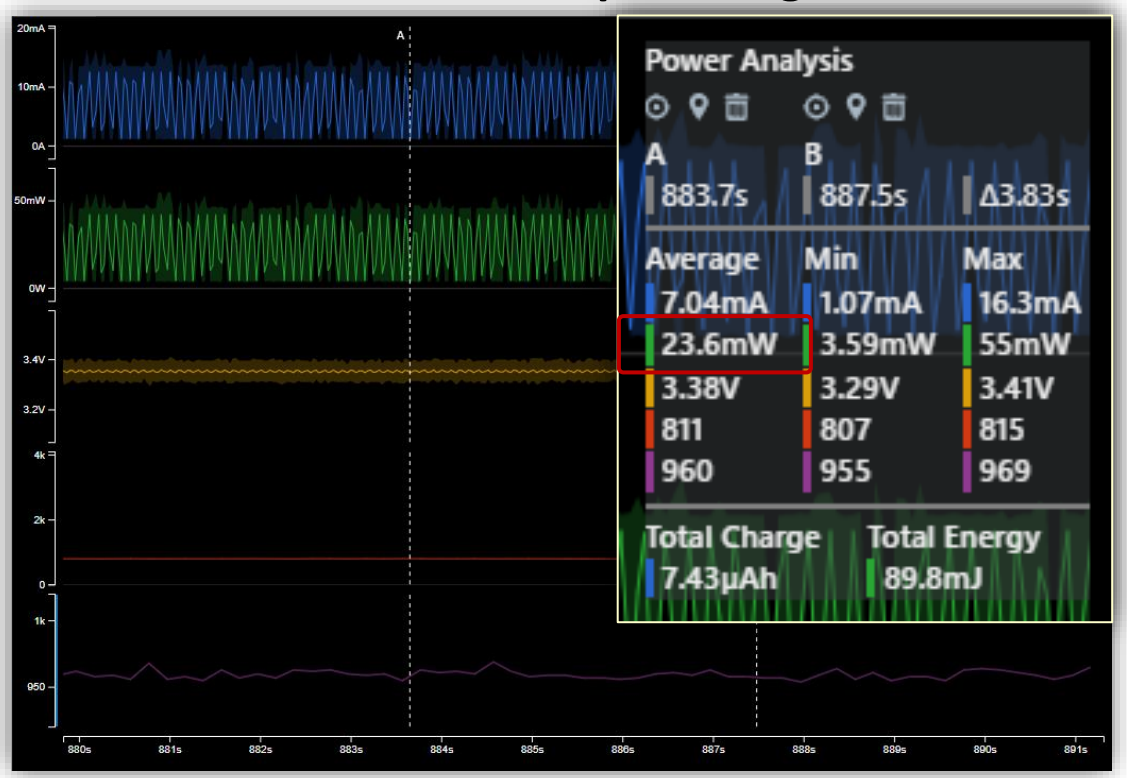
🛠 Lab13-9 ADC Scan DV DMAC SleepWalking

Result

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

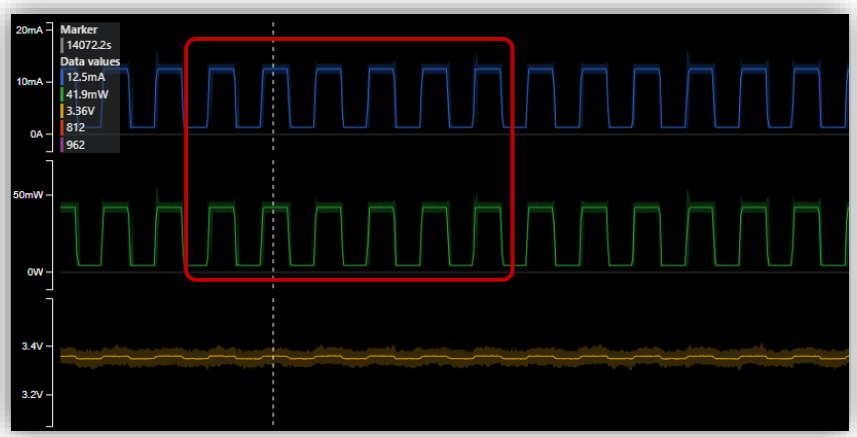
Lab13-8 Active

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.



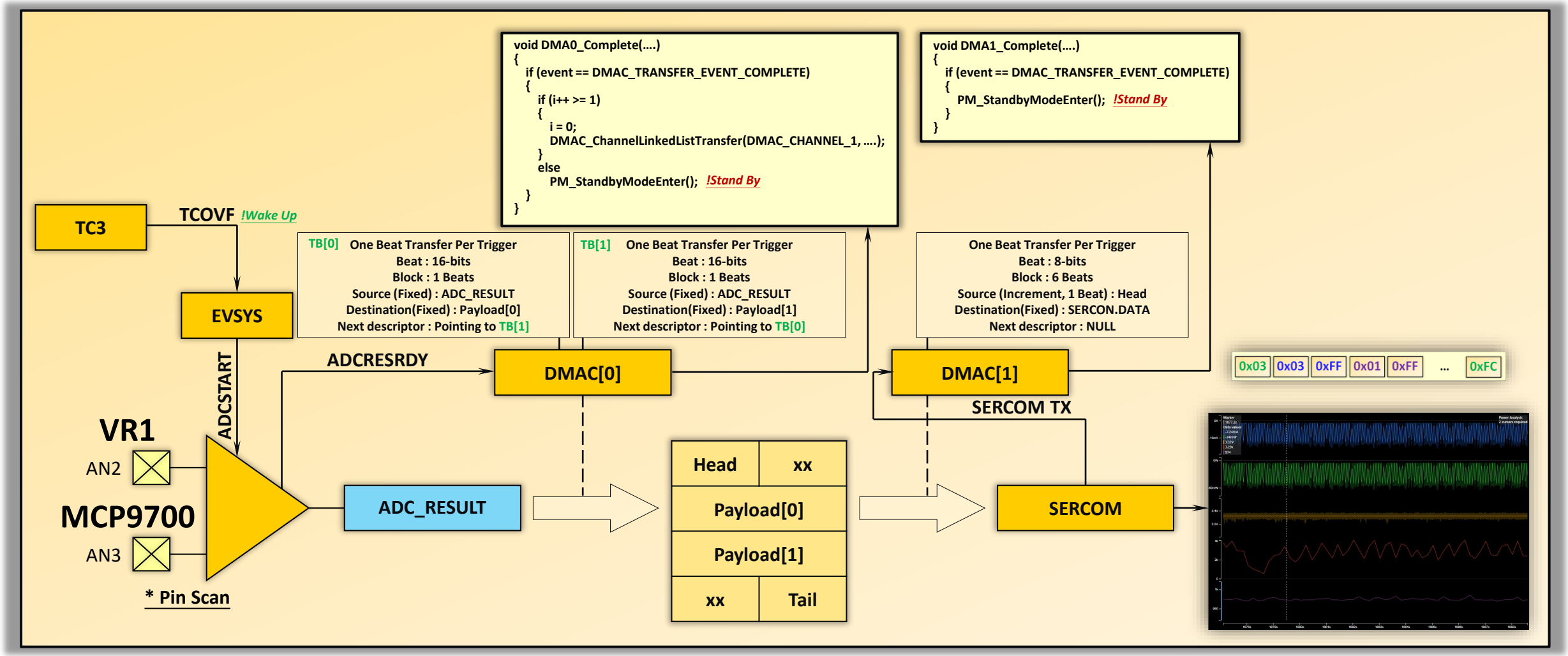
MICROCHIP

🛠 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 !**

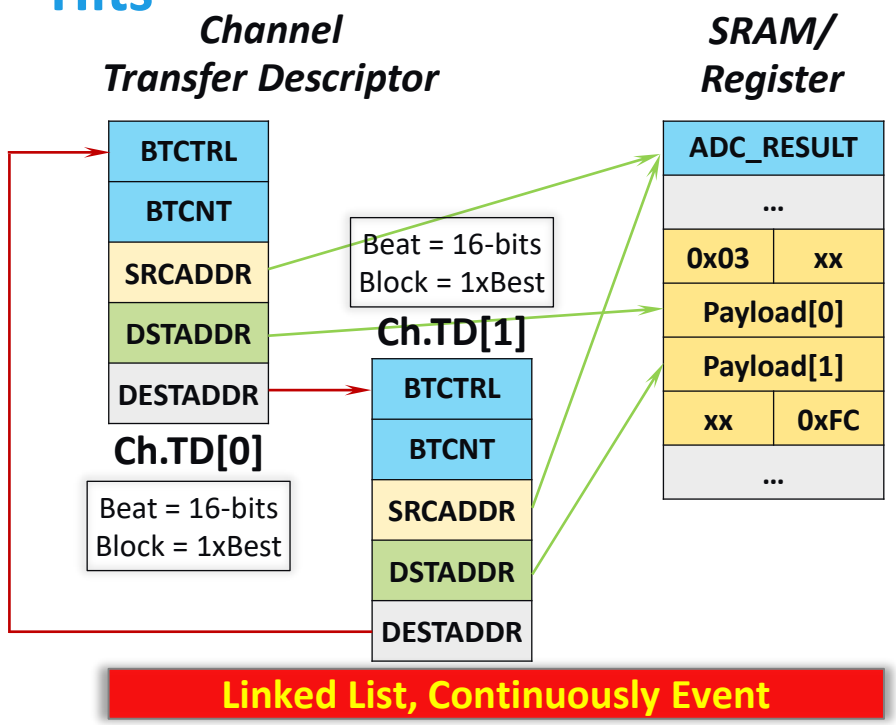
🛠 Lab13-10 ADC Scan DV DMAC-Linked SleepWalking

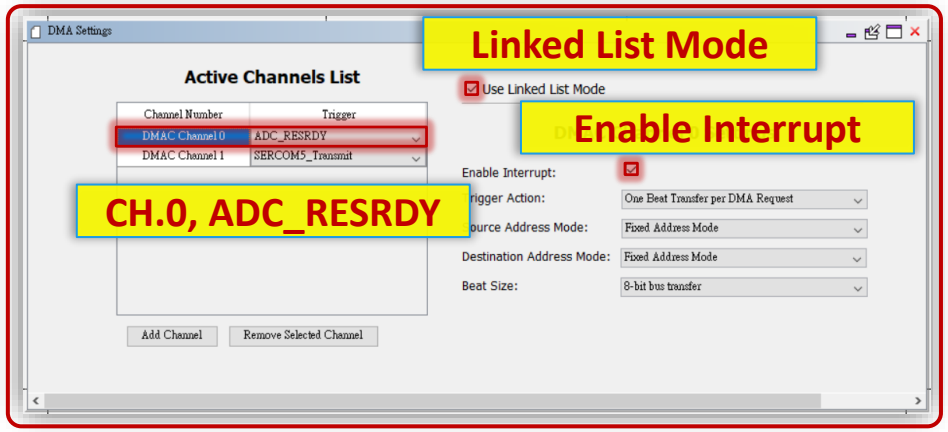
- **Apply new block diagram to save DMAC action time.**



🛠 Lab13-10 ADC Scan DV DMAC-Linked SleepWalking

Hits





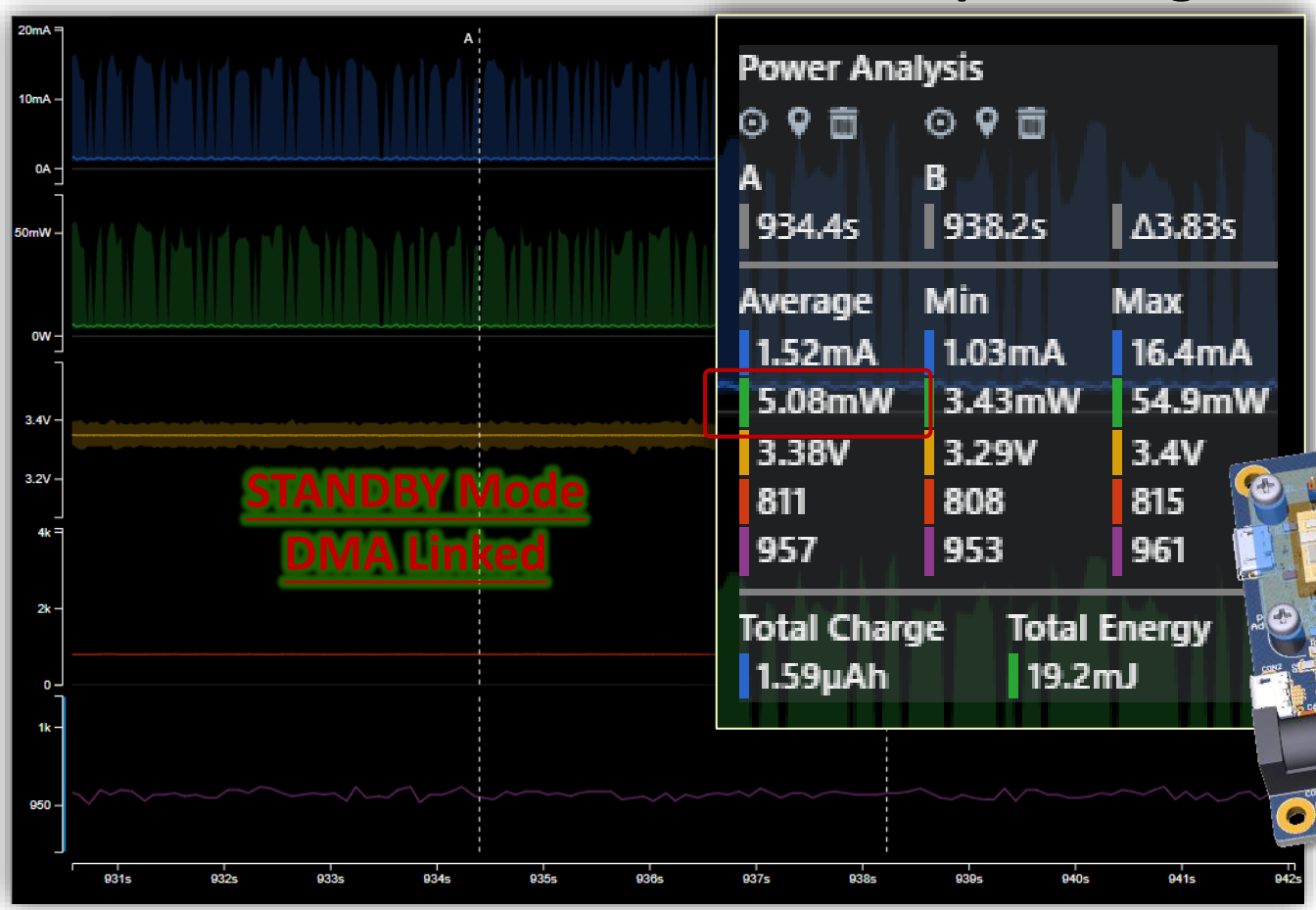

```
dmac_descriptor_registers_t DMAC_Descriptor_ADC_SRAM[2];

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]));
```

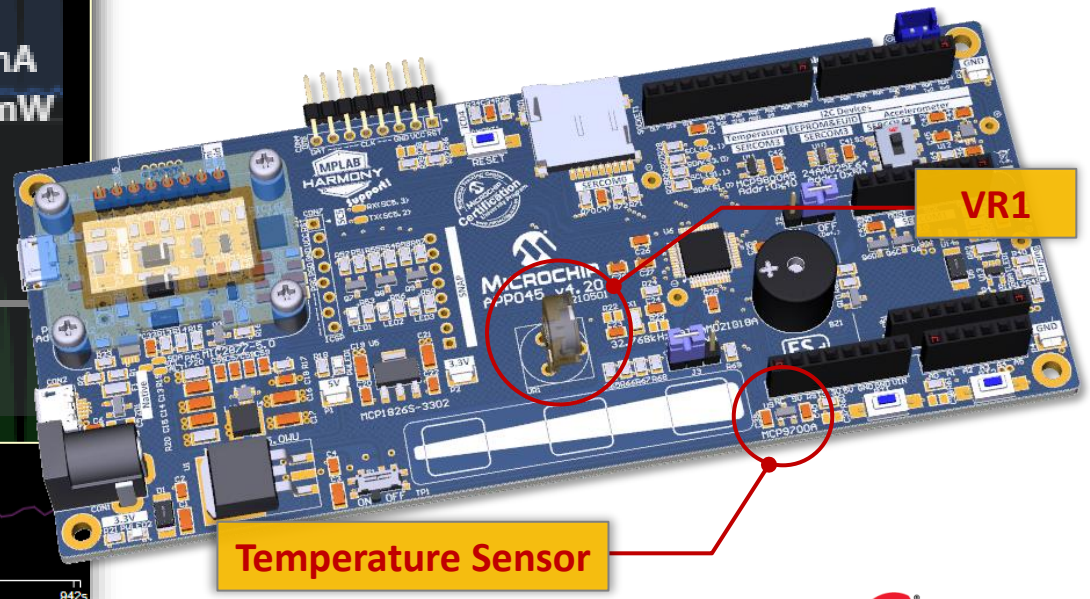
MICROCHIP

🛠 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.**



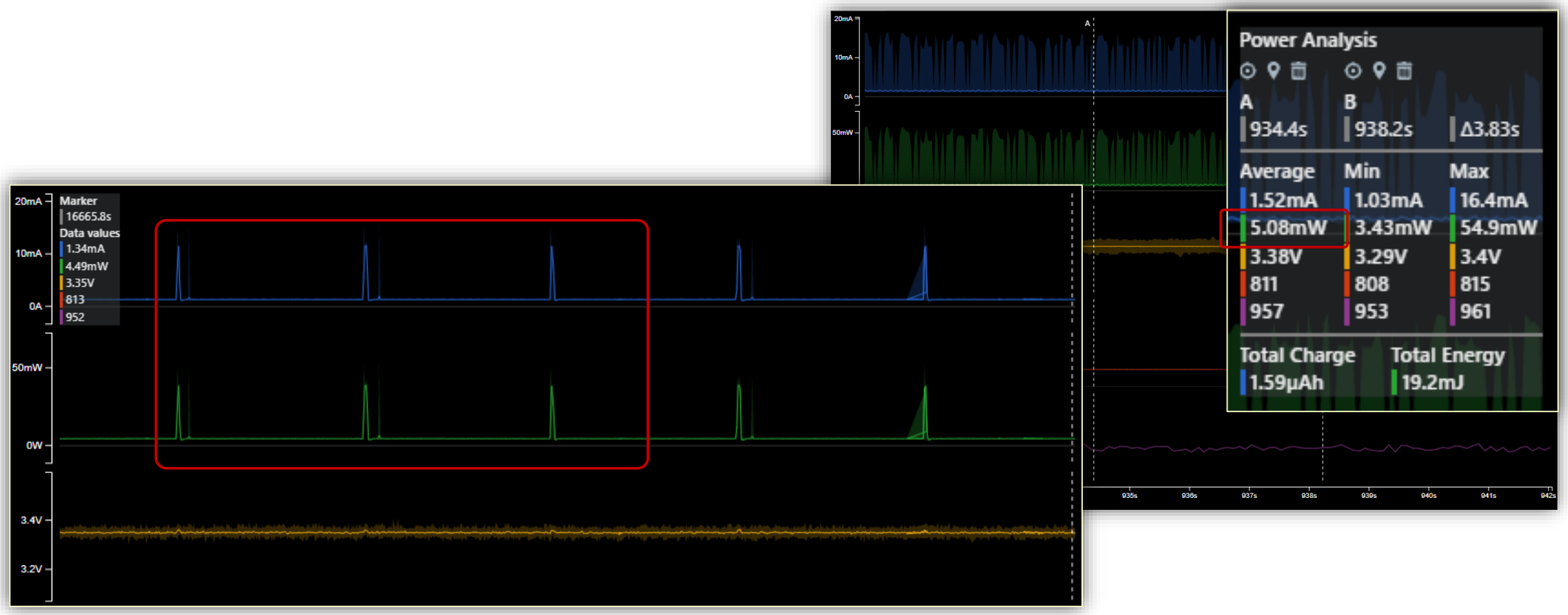


MICROCHIP

🛠 Lab13-10 ADC Scan DV DMAC-Linked SleepWalking

Result

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

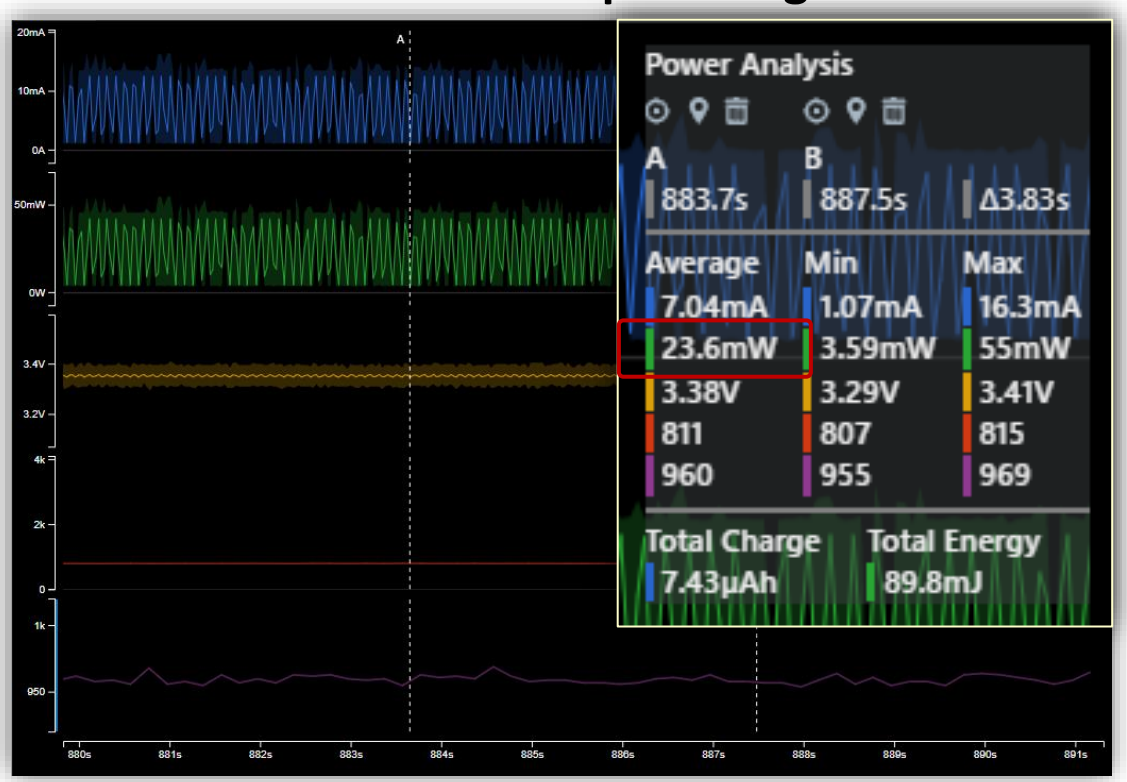


🛠 Lab13-10 ADC Scan DV DMAC-Linked SleepWalking

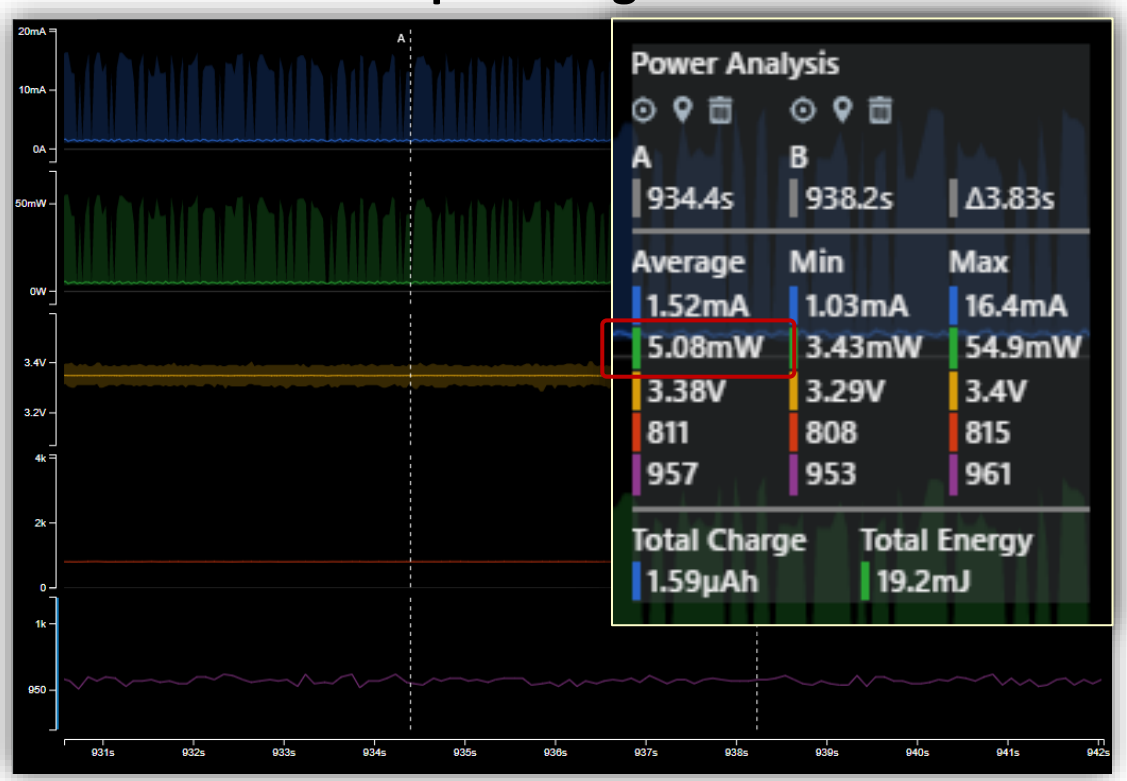
Result

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

Lab13-9 SleepWalking



Lab13-10 SleepWalking & DMAC-Linked



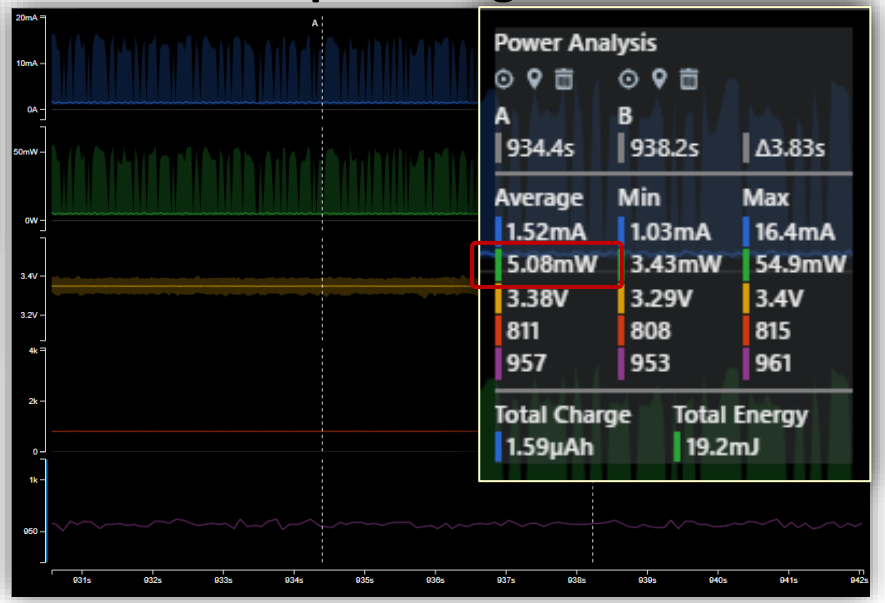
🛠 Lab13-10 ADC Scan DV DMAC-Linked SleepWalking

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

Lab13-9 SleepWalking

Lab13-8 Active

Lab13-10 SleepWalking & DMAC-Linked



MICROCHIP

🛠 Lab13-8, 9, 10 Finial 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.

MICROCHIP

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, Kleer, 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, SQI, 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

MICROCHIP