

SAMD21 ARM Cortex-M0+ Microcontroller & MCC Harmony SAM2001_{ADV} v2.02



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



SMART | CONNECTED | SECURE

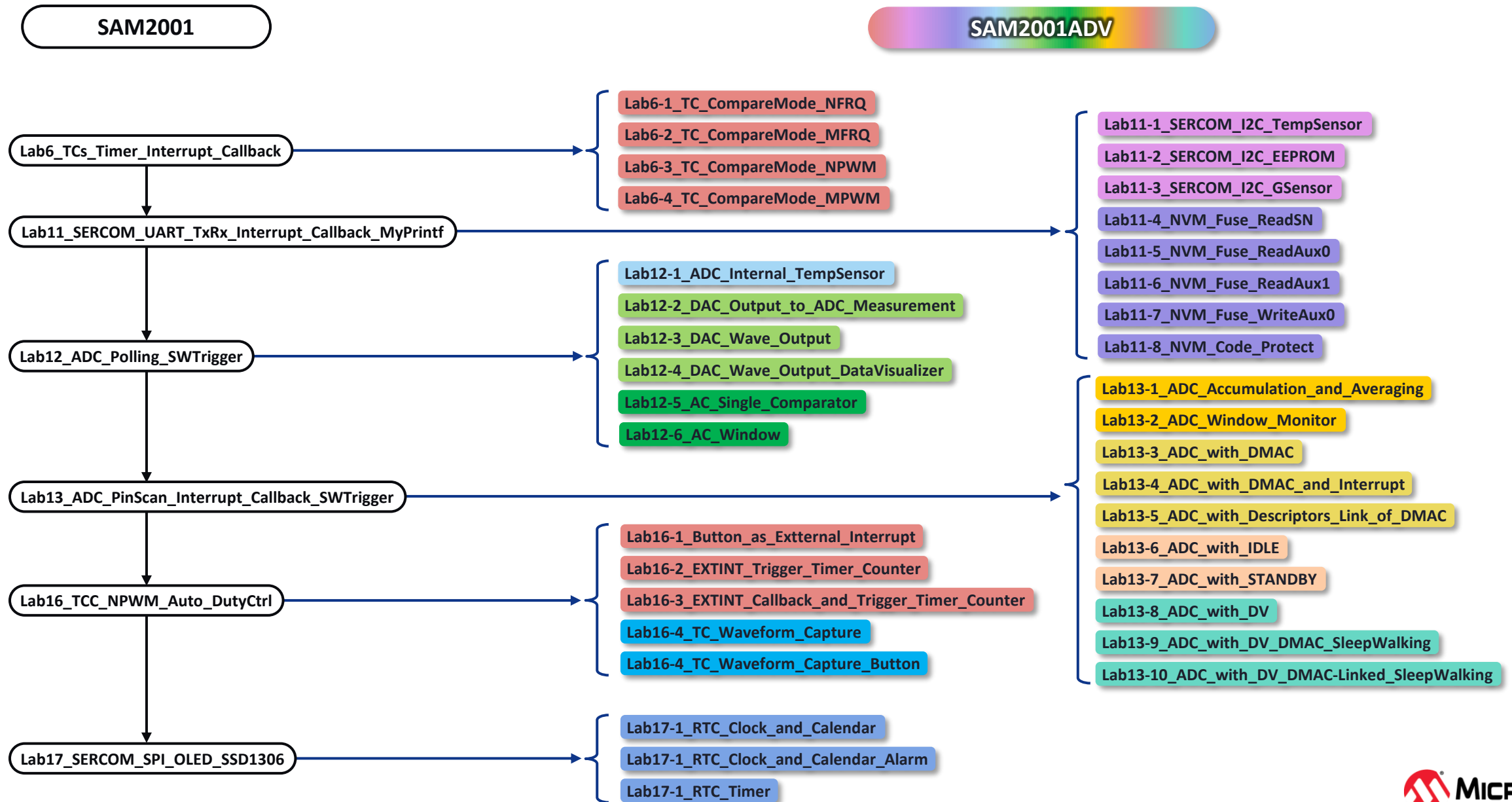
**CAE Taiwan Team
Microchip Technology Inc.**

2024 March

Class Objectives

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

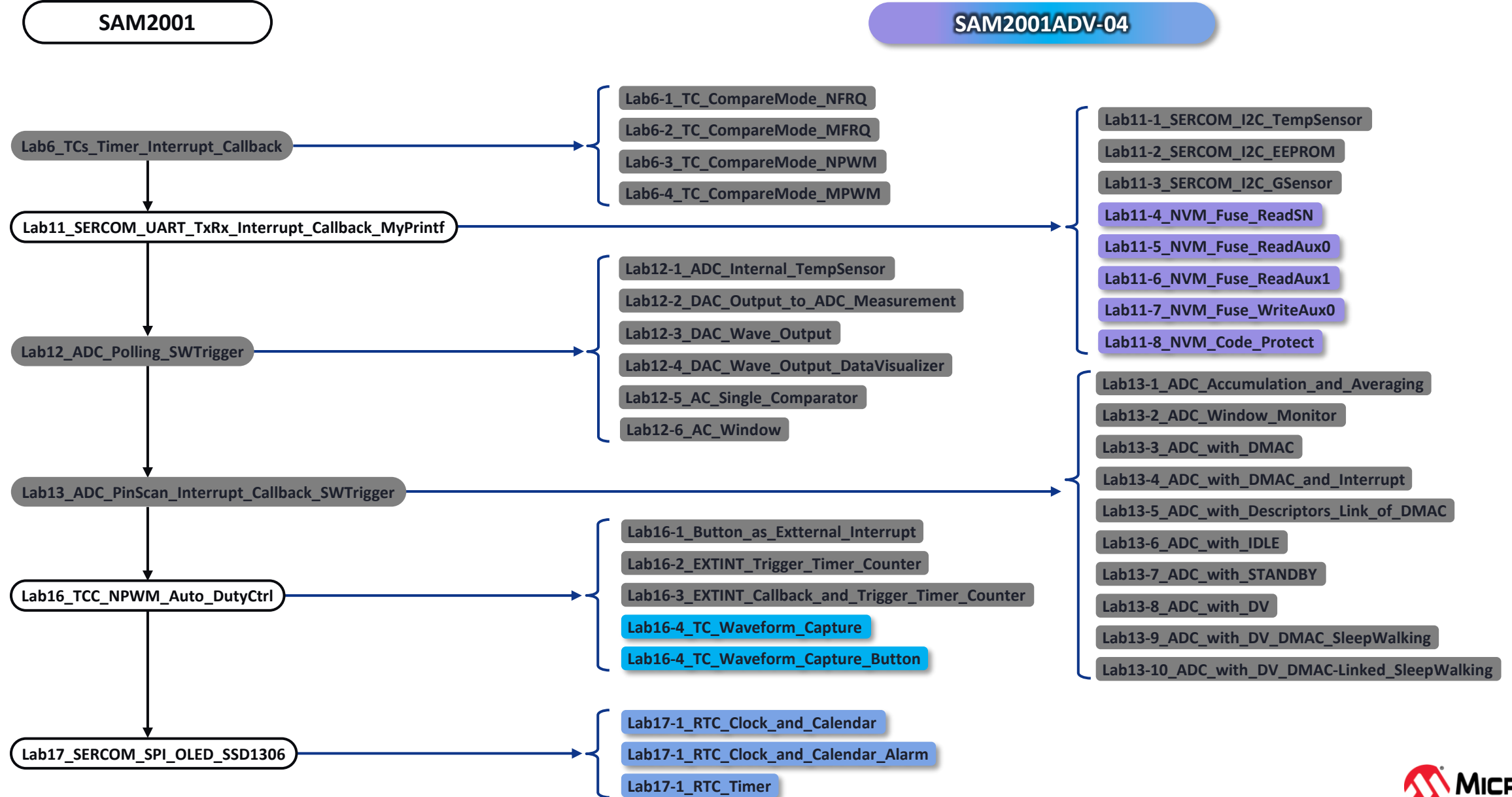
Lab Tree Branch from SAM2001 to SAM2001ADV



Separate Index for SAM2001ADV-04

- SAM2001ADV-04
 - Real Time Counter (RTC)
 - TC Waveform Capture (TC, EIC and Event System)
 - Auxiliary Space of NVM and Fuse Maintains

Separate Labs for SAM2001ADV-04

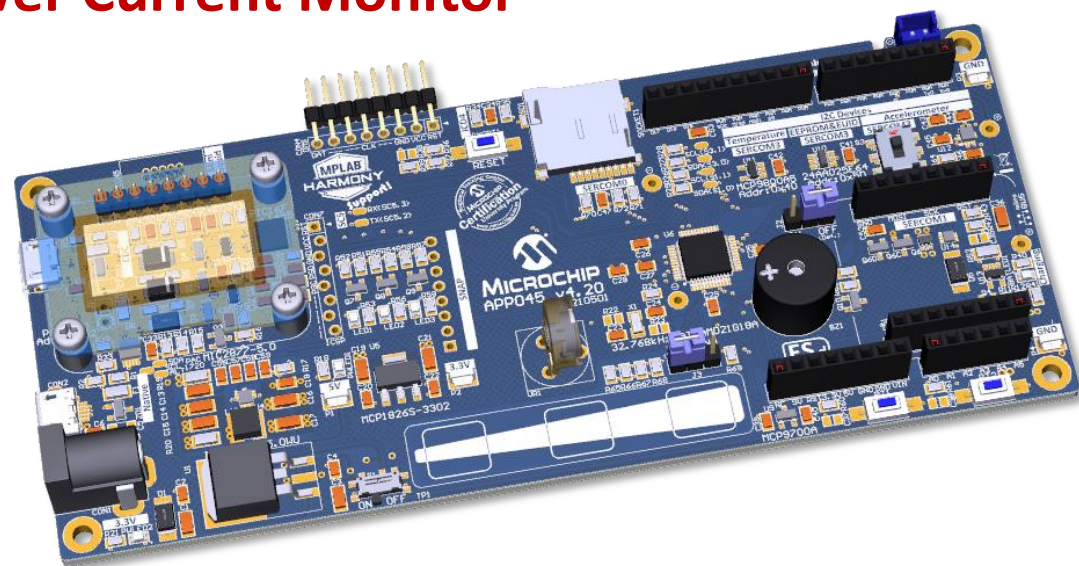


SAMD21 EVB APP045 v4

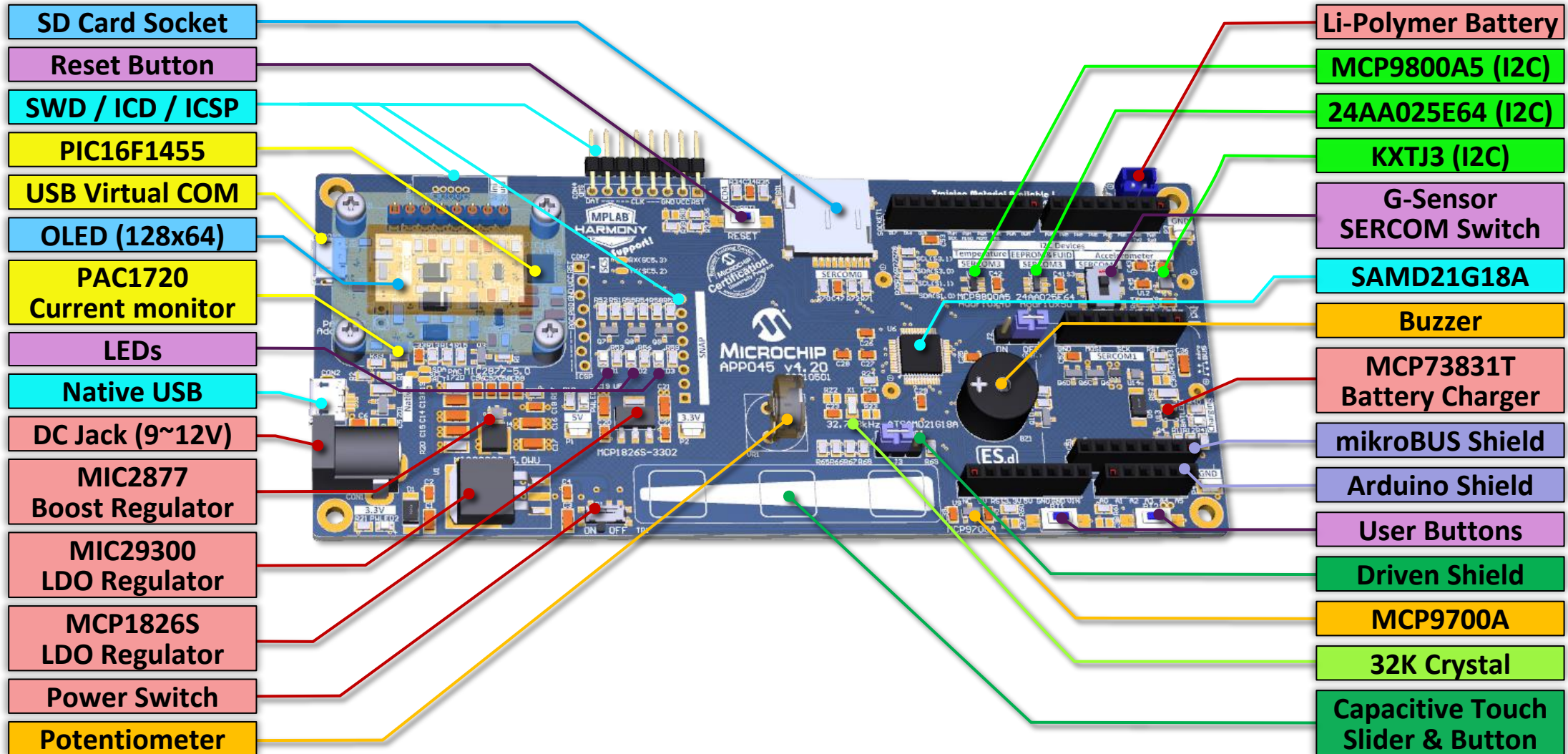
Introduction

APP045 v4 Features

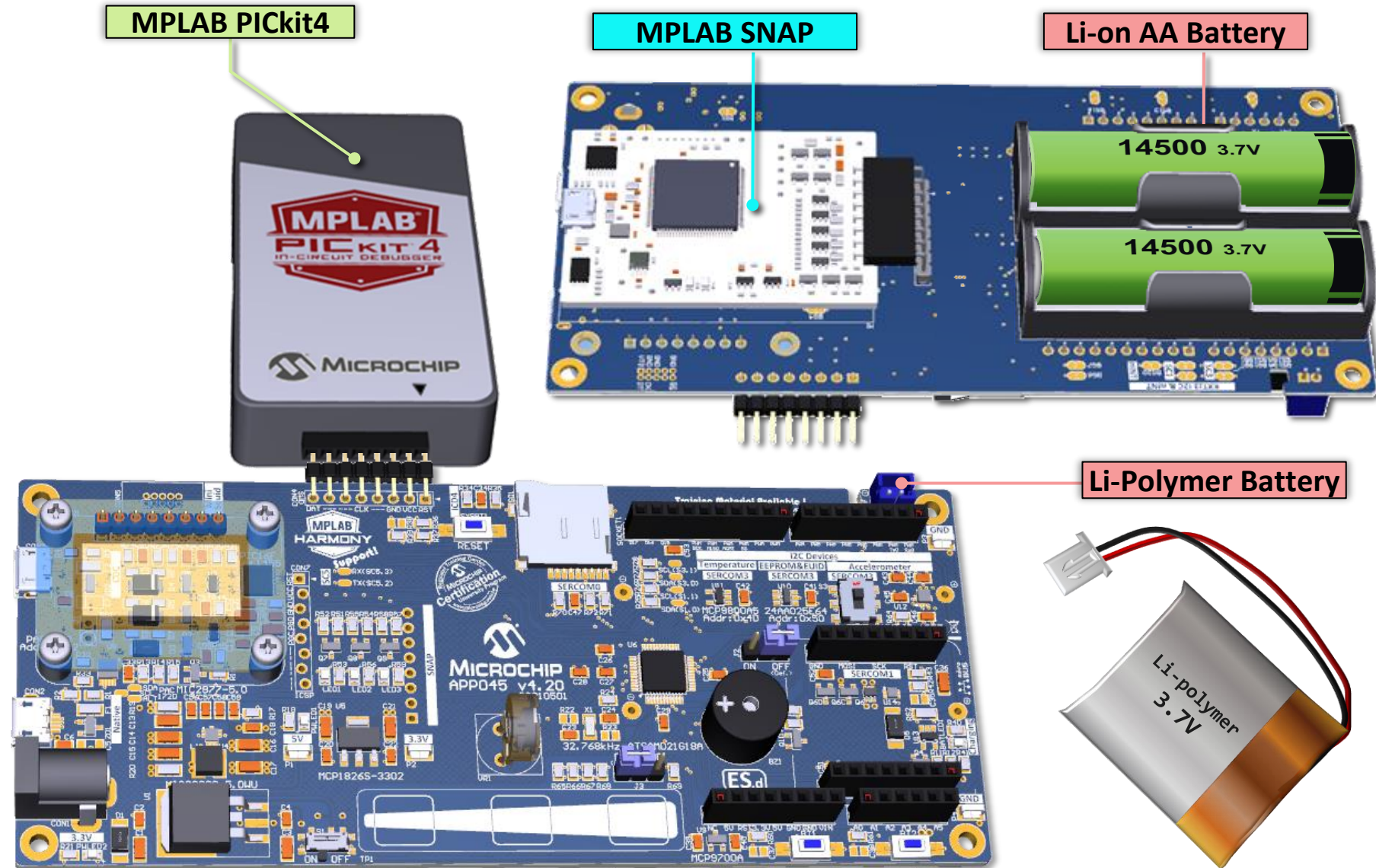
- On board **SAMD21G18A** 32Bit ARM Cortex M0+ Microcontroller
- Support plugged **MPLAB SNAP** external programmer/debugger
- Onboard **USB CDC Virtual COM Port** and **Power Current Monitor**
- Multiple Power Supply Source
- **Support Li-on Battery, boost and charger**
- 2 x Buttons, 3 x LEDs
- 1 x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



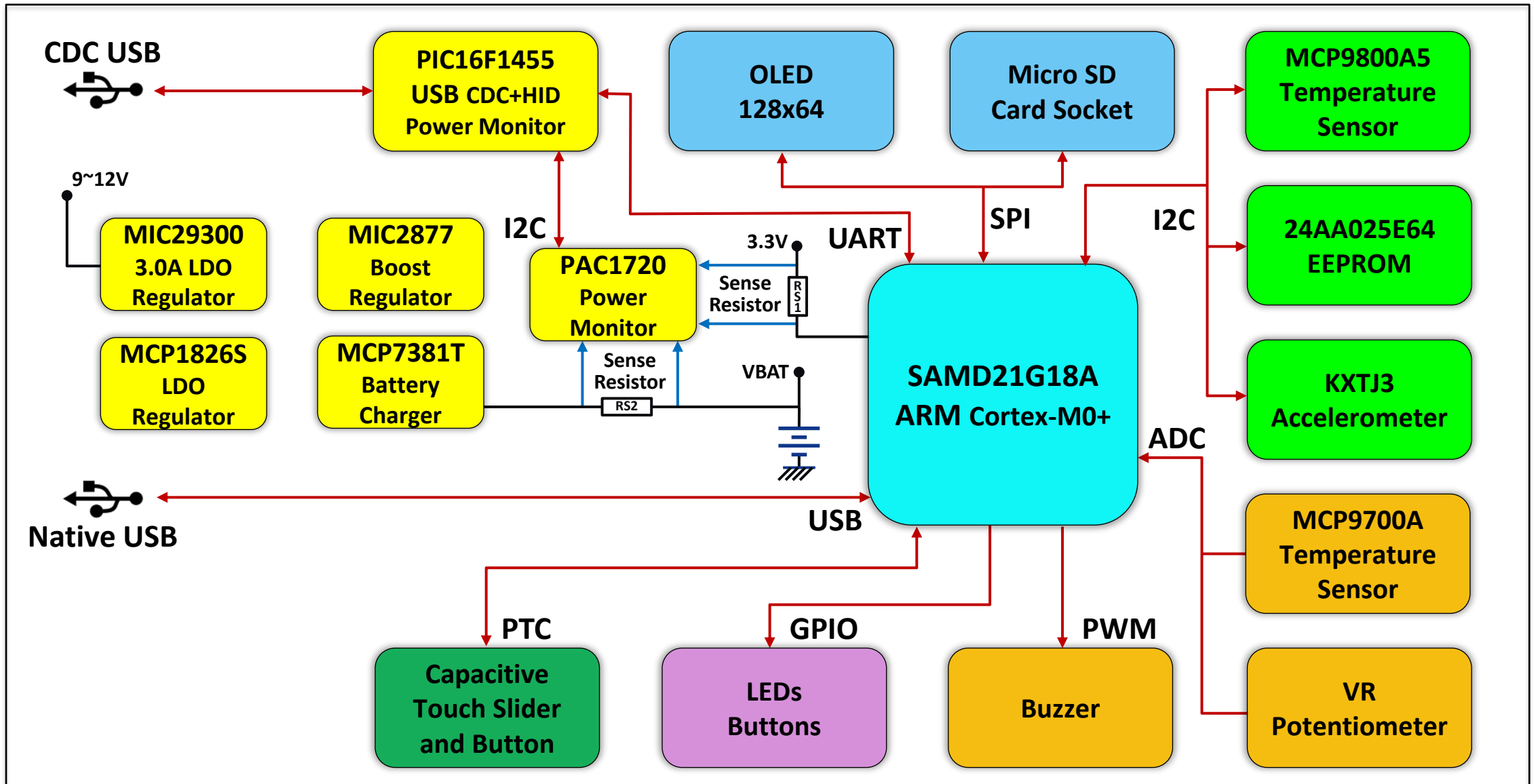
APP045 v4 Overview



APP045 v4 Programmer and Debugger



APP045 v4 Block Diagram



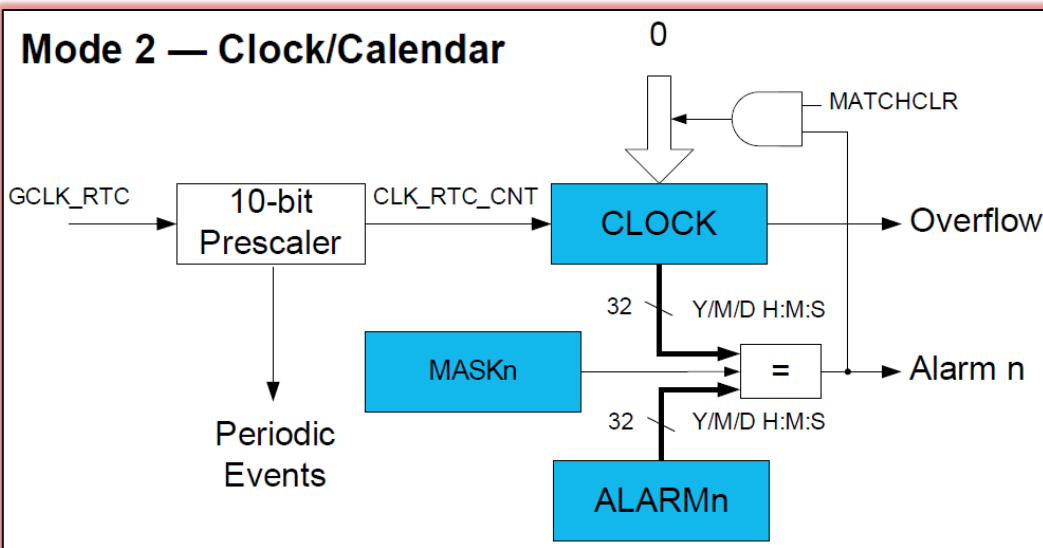
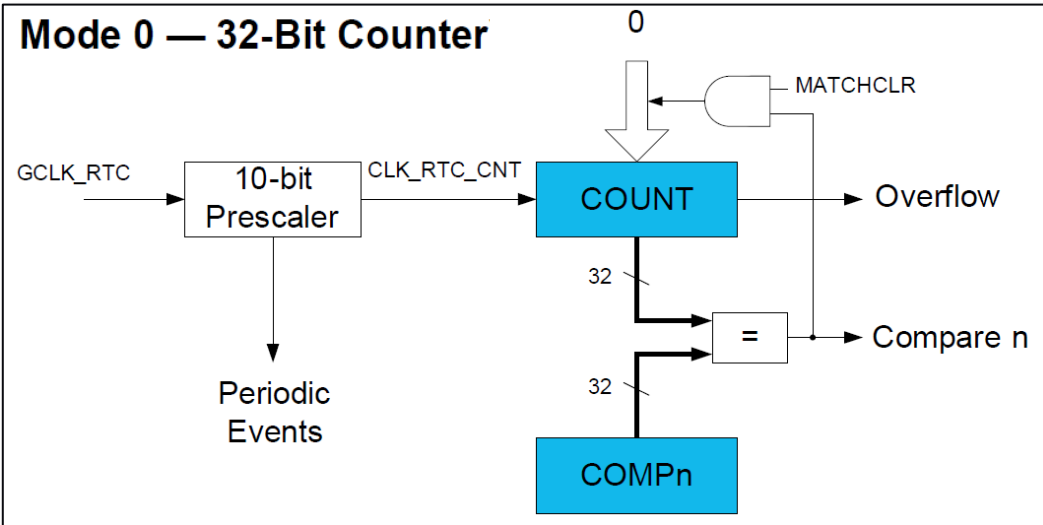
Real Time Counter (RTC)

Real Time Counter (RTC)

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

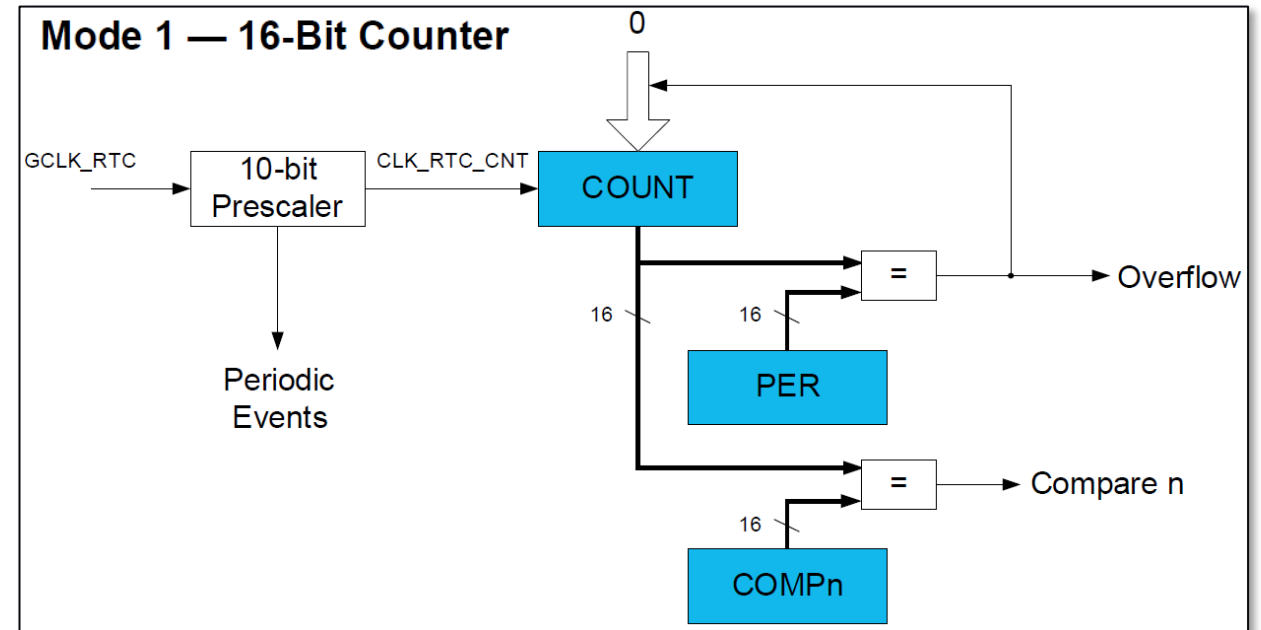
Real Time Counter (RTC)

Real Time Counter (RTC) Operation Mode



The RTC can function in one of these modes:

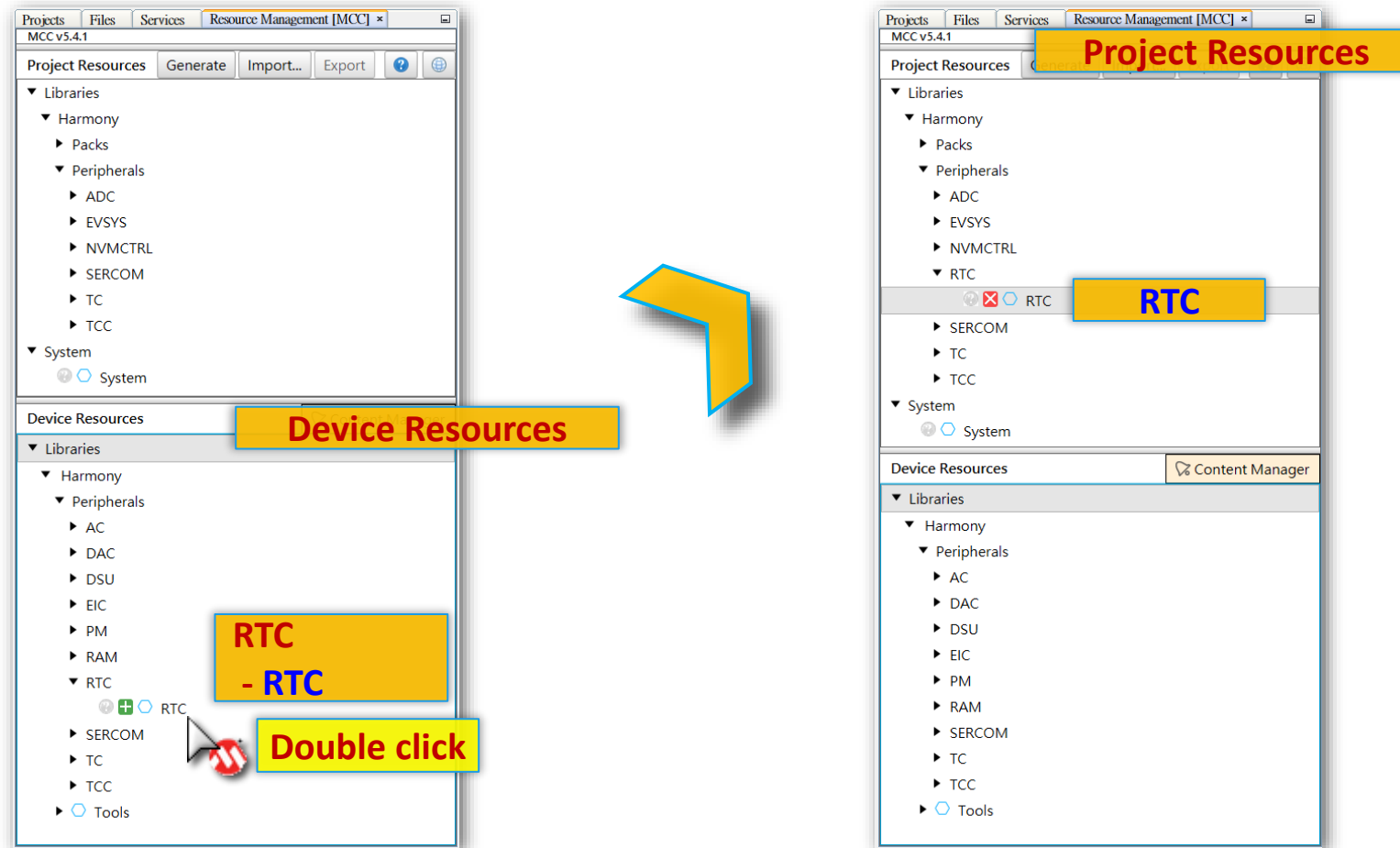
- Mode 0 - COUNT32
RTC serves as 32-bit counter
- Mode 1 - COUNT16
RTC serves as 16-bit counter
- Mode 2 – CLOCK
RTC serves as clock/calendar with alarm functionality



Real Time Counter (RTC)

Add RTC module using MHC

- Find **RTC** component in Device Resources window.
- Double click **RTC** to add to Project Resources window.



Real Time Counter (RTC)

Configuration Options Example

Mode 0 – 32-Bit Counter

RTC

Hardware Settings

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

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

RTC MODE 0 Configuration

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

RTC EVENTS configuration

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

Mode 1 – 16-Bit Counter

RTC

Hardware Settings

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

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

RTC MODE 1 Configuration

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

RTC EVENTS configuration

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

Mode 2 – Clock/Calendar

RTC

Hardware Settings

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

RTC Operation Mode: Clock/Calendar with Alarm

RTC MODE 2 Configuration

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

RTC EVENTS configuration

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

Real Time Counter (RTC)

Interface Routines

void RTC_Initialize(void);

void RTC_InterruptHandler(void);

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

bool RTC_RTCCTimeSet (struct tm * initialTime);

void RTC_RTCCTimeGet (struct tm * currentTime);

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

void RTC_RTCCallbackRegister (RTC_CALLBACK callback, uintptr_t context);

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

void RTC_Timer32Start (void);

void RTC_Timer32Stop (void);

void RTC_Timer32CounterSet (uint32_t count);

uint32_t RTC_Timer32CounterGet (void);

uint32_t RTC_Timer32PeriodGet (void);

uint32_t RTC_Timer32FrequencyGet (void);

void RTC_Timer32CompareSet (uint32_t compareValue);

void RTC_Timer32InterruptEnable (RTC_TIMER32_INT_MASK interrupt);

void RTC_Timer32InterruptDisable (RTC_TIMER32_INT_MASK interrupt);

void RTC_Timer32CallbackRegister (RTC_TIMER32_CALLBACK callback, uintptr_t context);

void RTC_Timer16Start (void);

void RTC_Timer16Stop (void);

void RTC_Timer16CounterSet (uint16_t count);

uint16_t RTC_Timer16CounterGet (void);

void RTC_Timer16PeriodSet (uint16_t period);

uint16_t RTC_Timer16PeriodGet (void);

uint32_t RTC_Timer16FrequencyGet (void);

void RTC_Timer16Compare0Set (uint16_t comparisonValue);

void RTC_Timer16Compare1Set (uint16_t comparisonValue);

void RTC_Timer16InterruptEnable (RTC_TIMER16_INT_MASK interrupt);

void RTC_Timer16InterruptDisable (RTC_TIMER16_INT_MASK interrupt);

void RTC_Timer16CallbackRegister (RTC_TIMER16_CALLBACK callback, uintptr_t context);

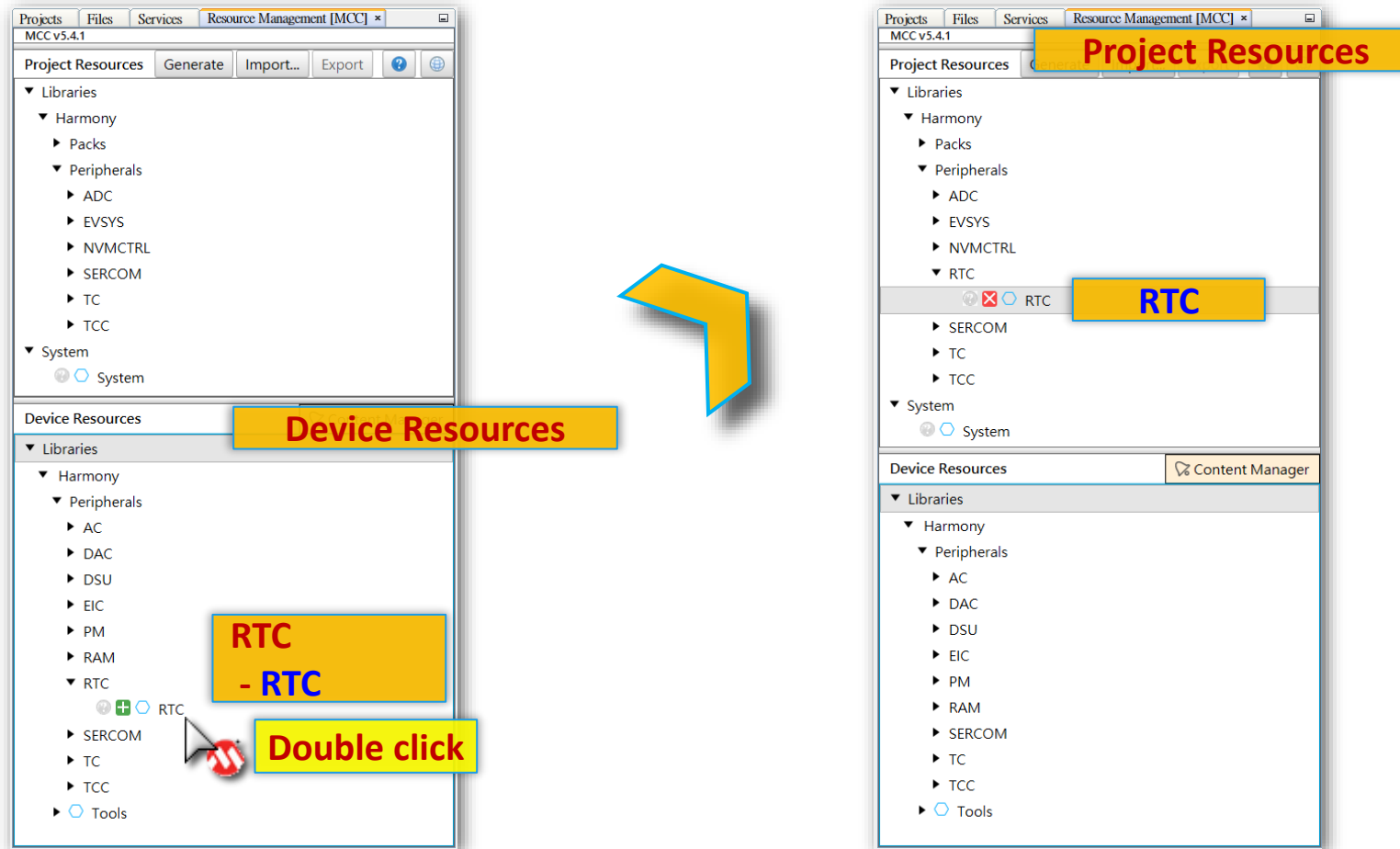
Lab17-1 RTC Clock and Calendar

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

⚙️ Lab17-1 RTC Clock and Calendar

Step 1

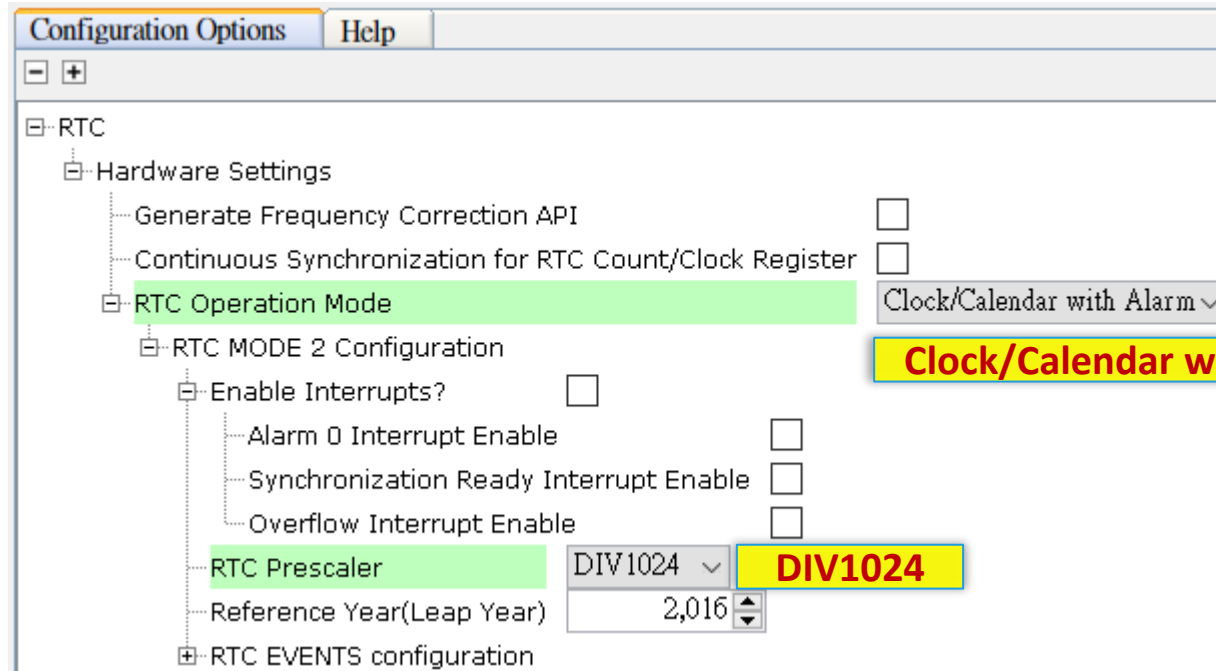
- Find **RTC** component in Device Resources window.
- Double click **RTC** to add to Project Resources window.



Lab17-1 RTC Clock and Calendar

Step 2

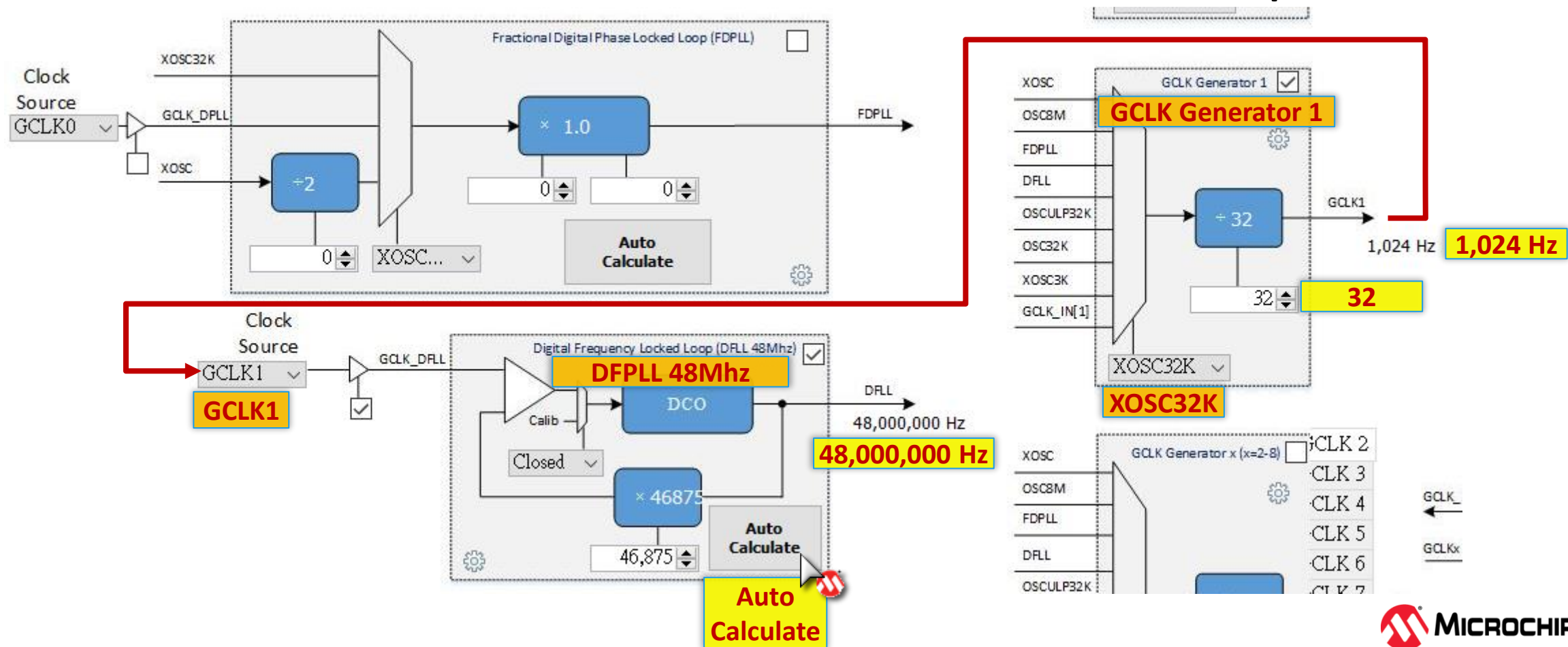
- Select RTC Operation Mode to **Clock/Calendar with Alarm** (Mode2).
- Set RTC Prescaler to **DIV1024**. (We will provide RTC 1024Hz to get **1Hz** clock tick)



⚙️ Lab17-1 RTC Clock and Calendar

Step 3 (This step already done in current project)

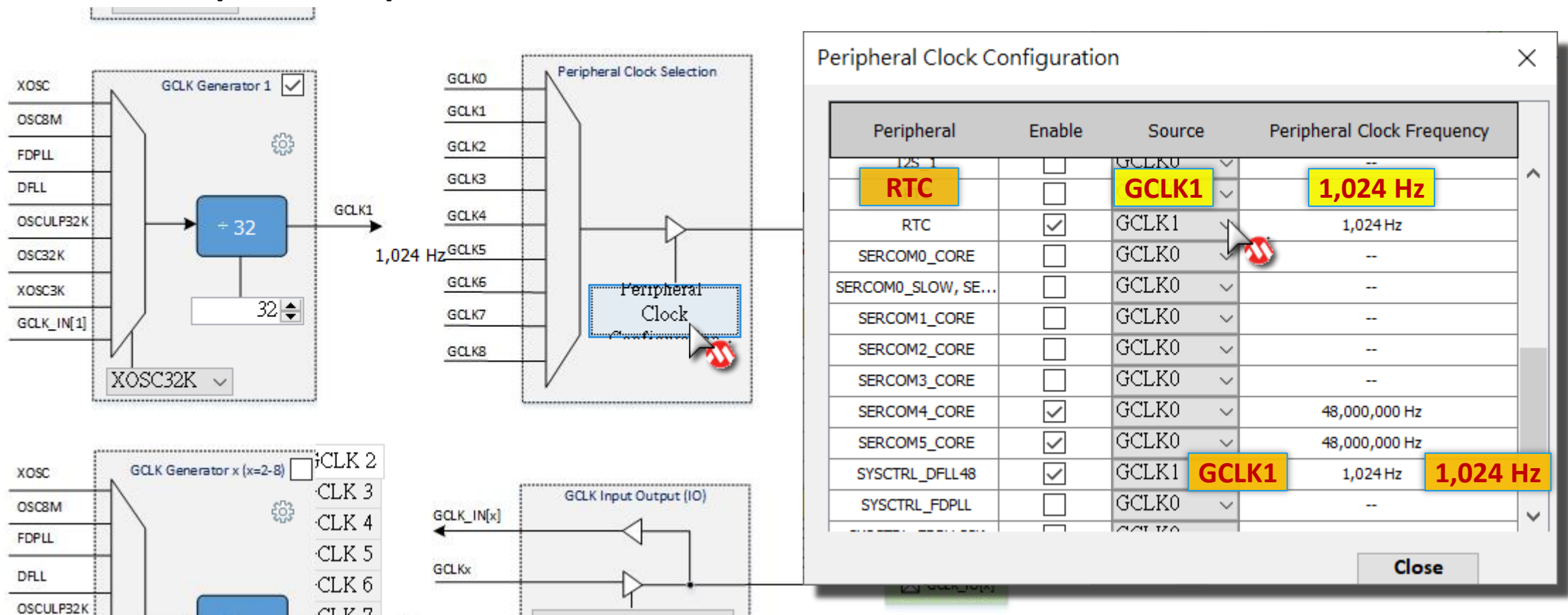
- In MHC → Tools → **Clock Configuration**, Select **GCLK1** to divided by **32**.
- Click on **Auto Calculate** of **DFPLL 48Mhz** to recalculate correct 48MHz output.



⚙️ Lab17-1 RTC Clock and Calendar

Step 4

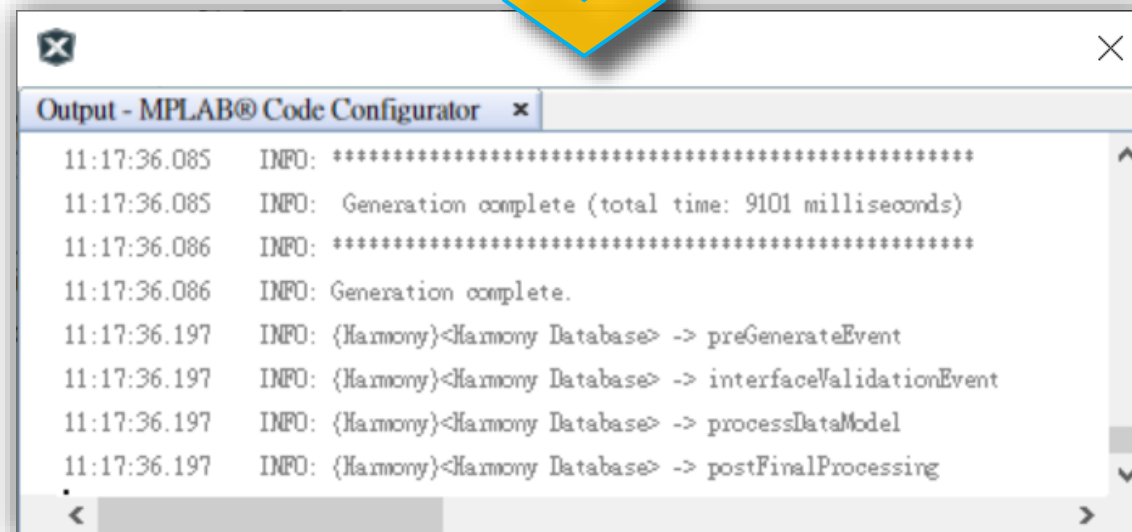
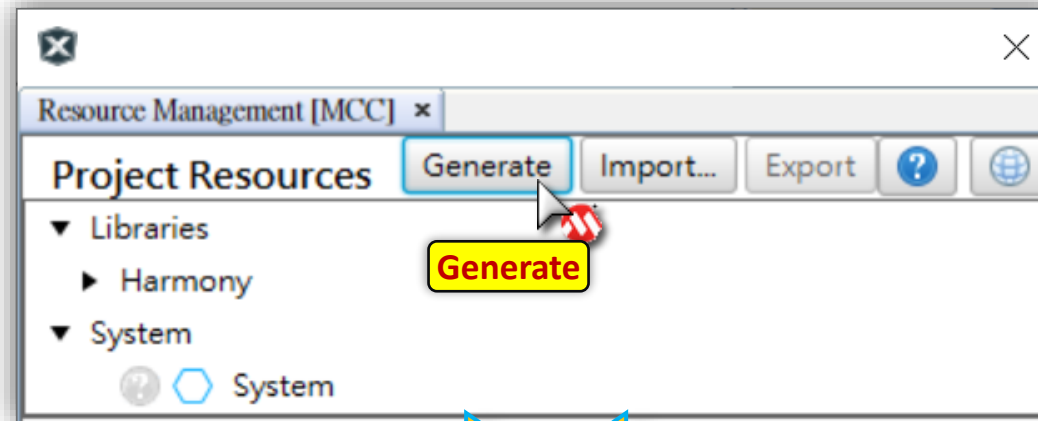
- Click on **Peripheral Clock Configuration** of **Peripheral Clock Selection**.
- Select **GCLK1 (1,024 Hz)** as source clock of **RTC**.



⚙️ Lab17-1 RTC Clock and Calendar

Step 5

- Click "**Generate**" Button to Generate your Code.



Real Time Counter (RTC)

Time structure in time.h

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

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

Real Time Counter (RTC)

RTC Mode 2 Register

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

RTC MODE2 CLOCK Register

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

Time structure

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

Real Time Counter (RTC)

RTC Mode 2 Register convert between Time structure

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

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

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

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

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

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

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

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

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

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

🔧 Lab17-1 RTC Clock and Calendar

Calculate Weekday

- Zeller's congruence (Use Gregorian calendar)

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

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

⚙️ Lab17-1 RTC Clock and Calendar

Step 6

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

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

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

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

Lab17-1 RTC Clock and Calendar

Use C compiler time as Clock Initialize

- Standard Predefined Macros

__FILE__

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

__LINE__

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

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

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

__DATE__

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

__TIME__

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

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

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

Lab17-1 RTC Clock and Calendar

Step 7

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

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

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

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

    RTC_RTCCTimeSet( pTimeStruct );
    RTC_CalculateWeekday( pTimeStruct );
}

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

Lab17-1 RTC Clock and Calendar

Step 8

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

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

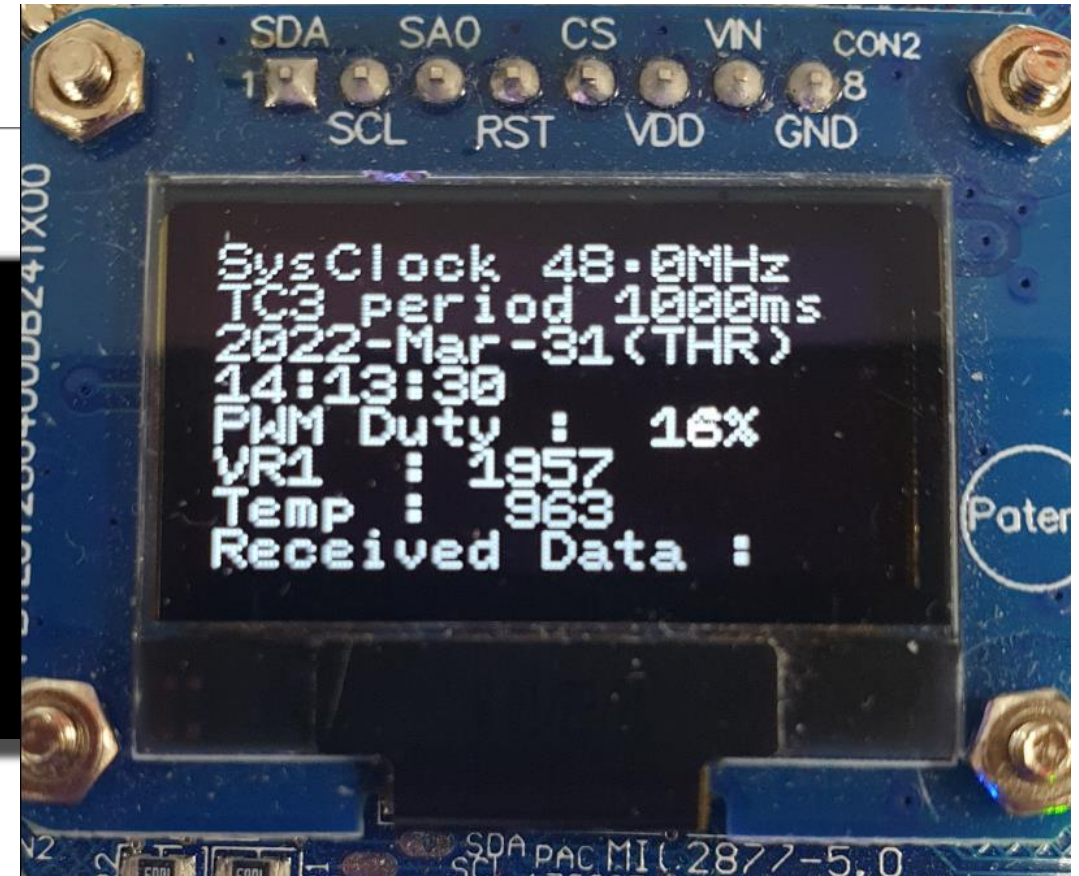
            ADC_ConversionStart();
        }
    }
}
```

🔧 Lab17-1 RTC Clock and Calendar

Result

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

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



⚙️ Lab17-1 RTC Clock and Calendar

Discuss

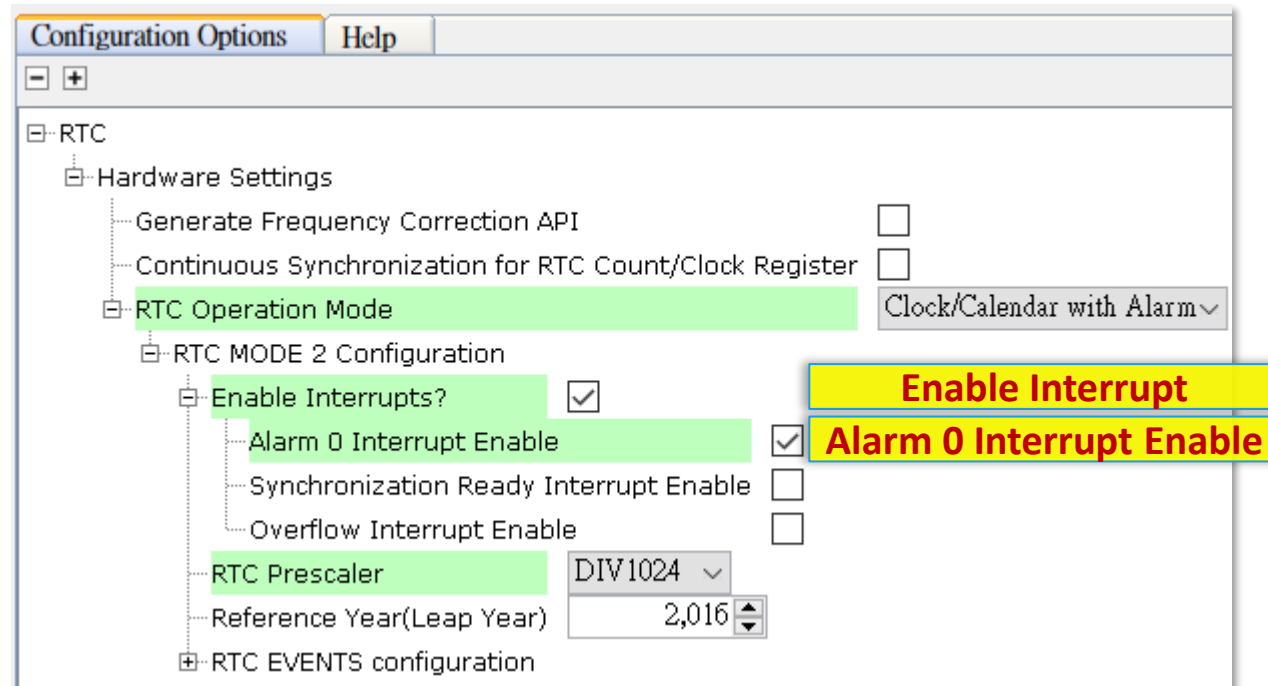
- How to setup an Alarm ?

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

Two extra interface routines will be generated.

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

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



⚙️ Lab17-1 RTC Clock and Calendar

Discuss

- Implement an Alarm in 10 seconds after boot.

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

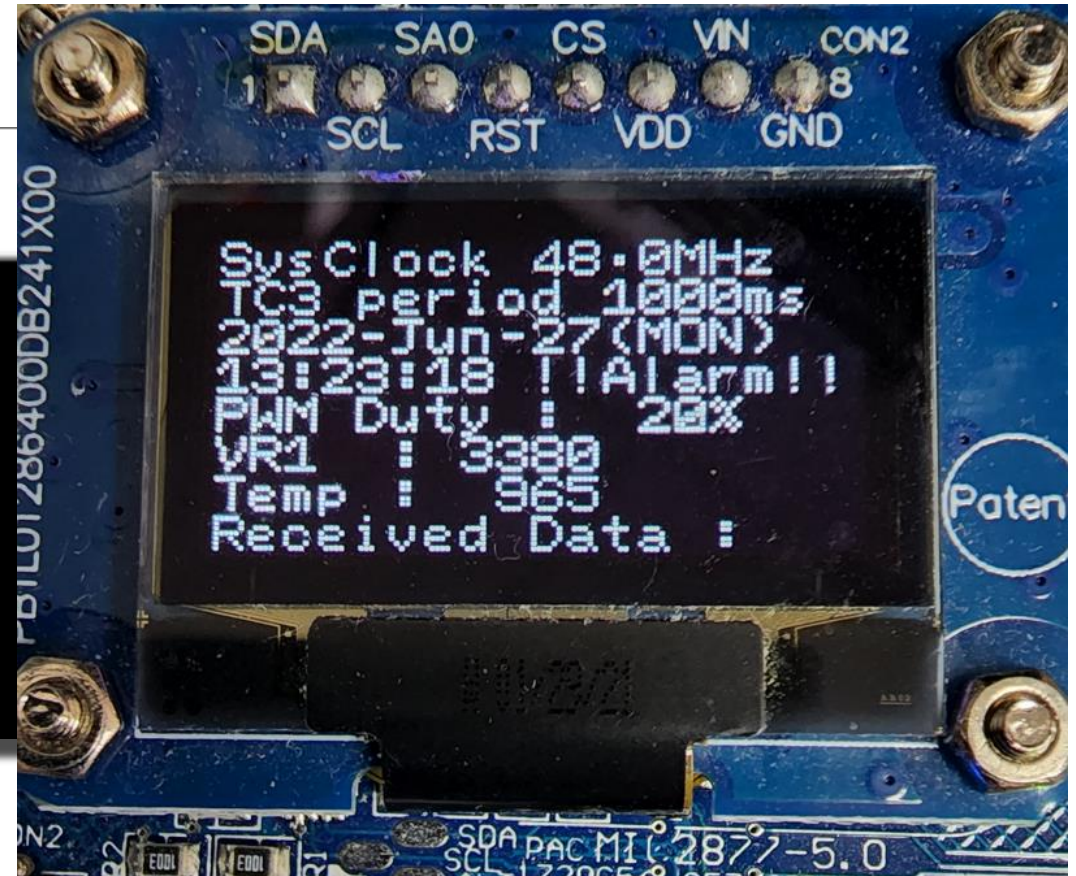
```
typedef enum
{
    RTC_ALARM_MASK_SS = 0x1,           //Alarm every minute
    RTC_ALARM_MASK_MMSS,               //Alarm every Hour
    RTC_ALARM_MASK_HHMMSS,             //Alarm Every Day
    RTC_ALARM_MASK_DDHHMMSS,           //Alarm Every Month
    RTC_ALARM_MASK_MMDDHHMMSS,         //Alarm Every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

⚙️ Lab17-1 RTC Clock and Calendar

Discuss

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

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

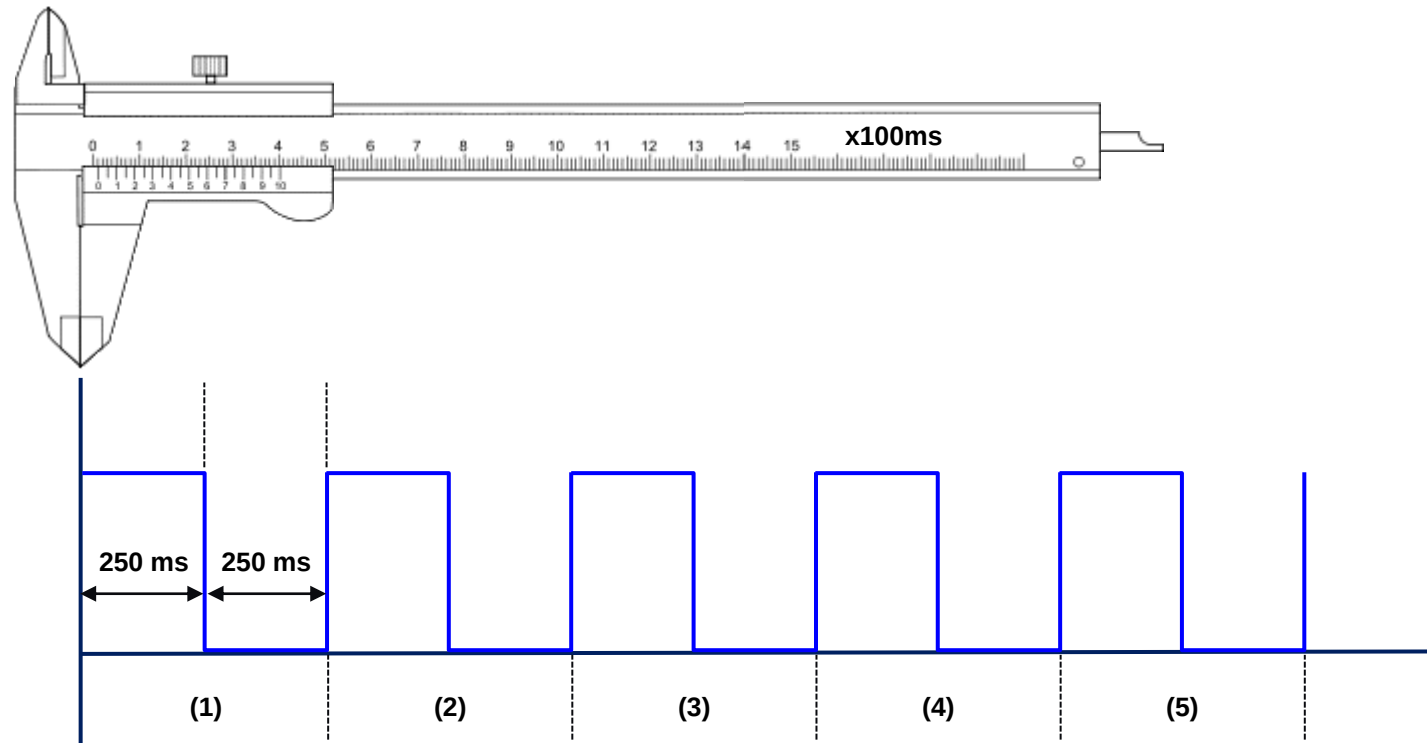


TC Waveform Capture

Integrate TC, EIC and Event System

What's Waveform Capture?

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



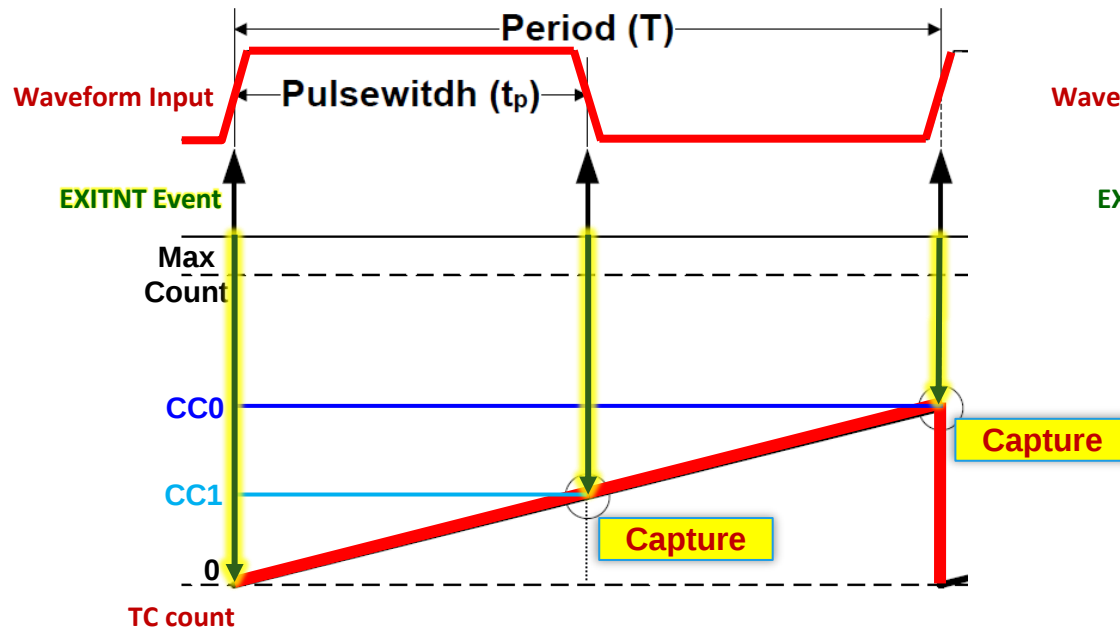
TC Capture with Event System

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

PPW (Period and Pulse-width) :

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

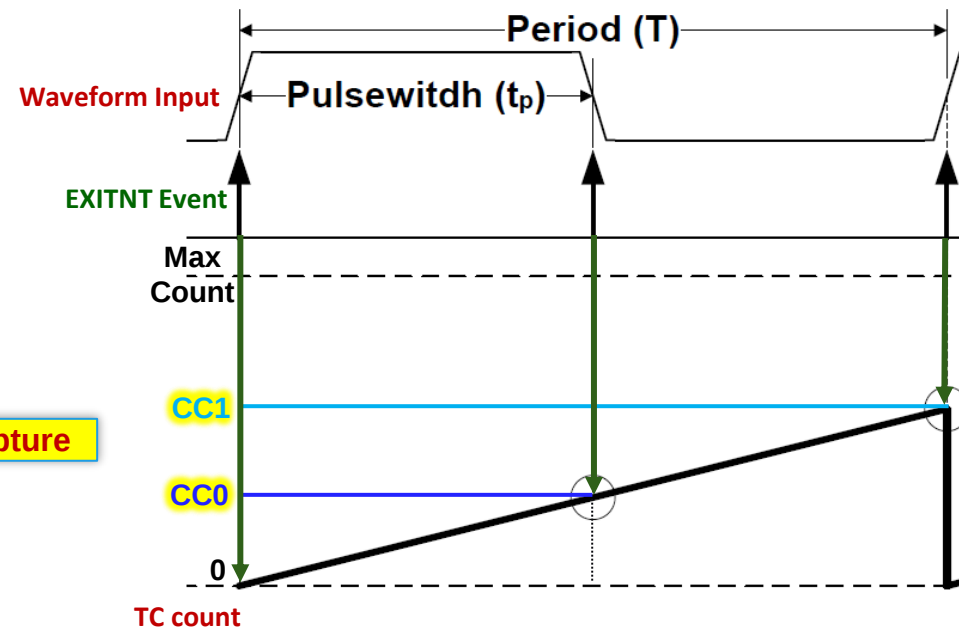
Pulsewidth (t_p) will be into **CC1**



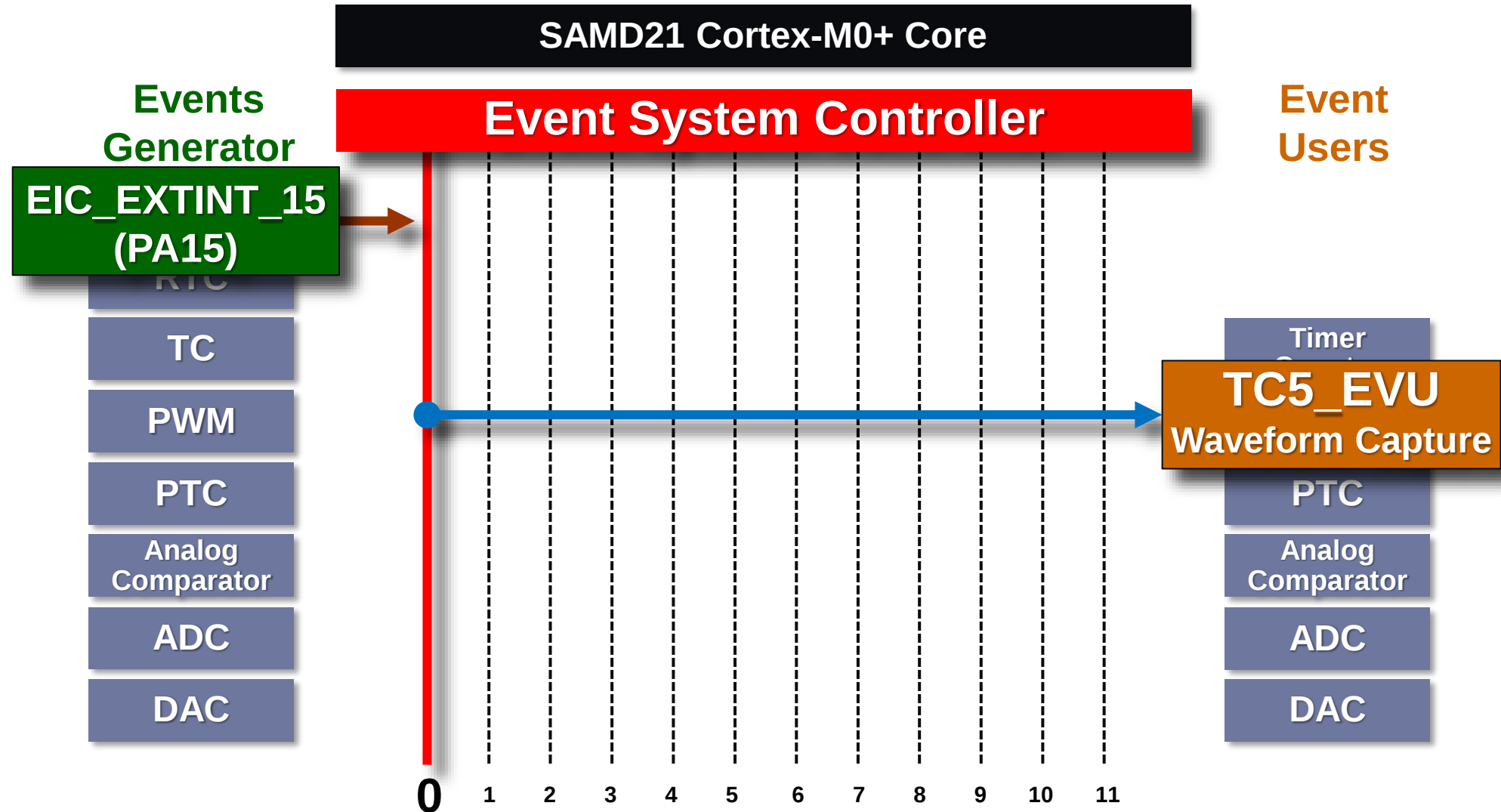
PWP (Pulse-width, Period) :

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

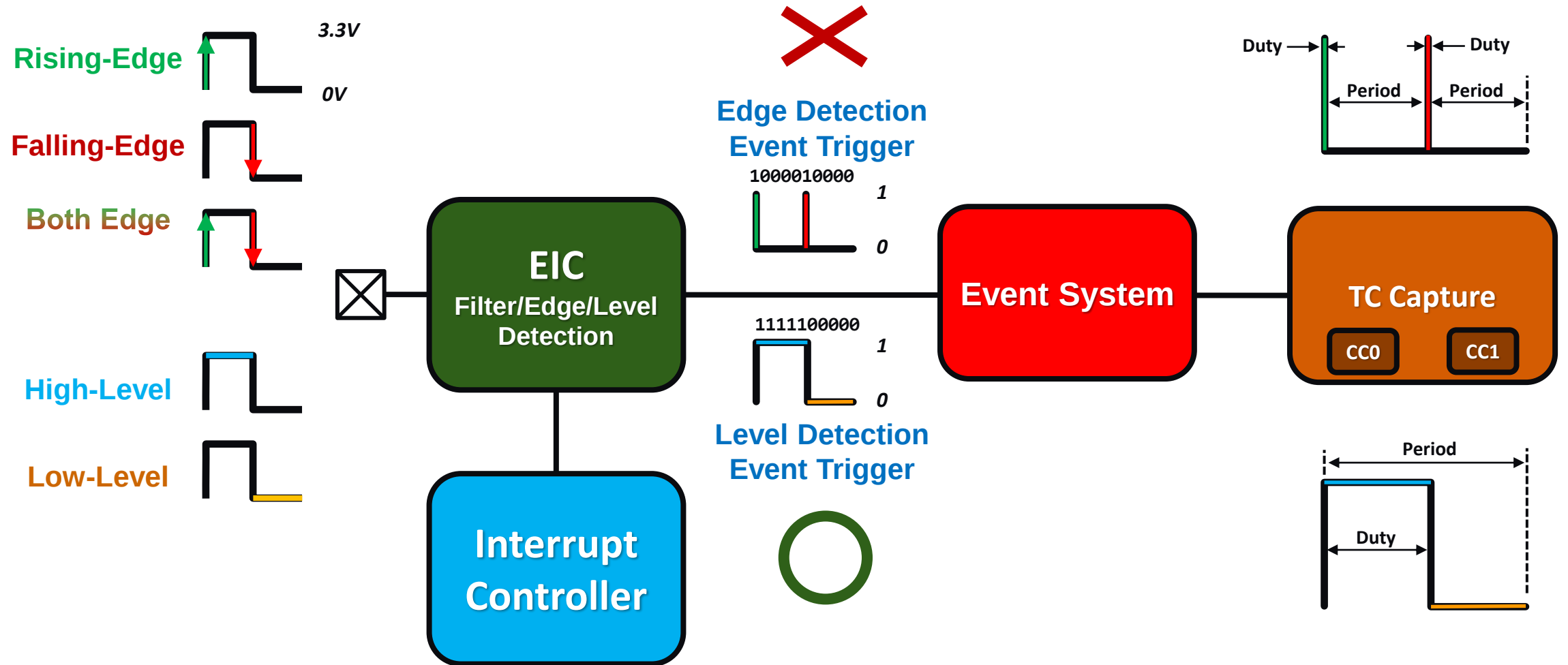
Pulsewidth (t_p) will be into **CC0**.



Routing EIC to TC Capture through Event System



Routing EIC to TC Capture through Event System



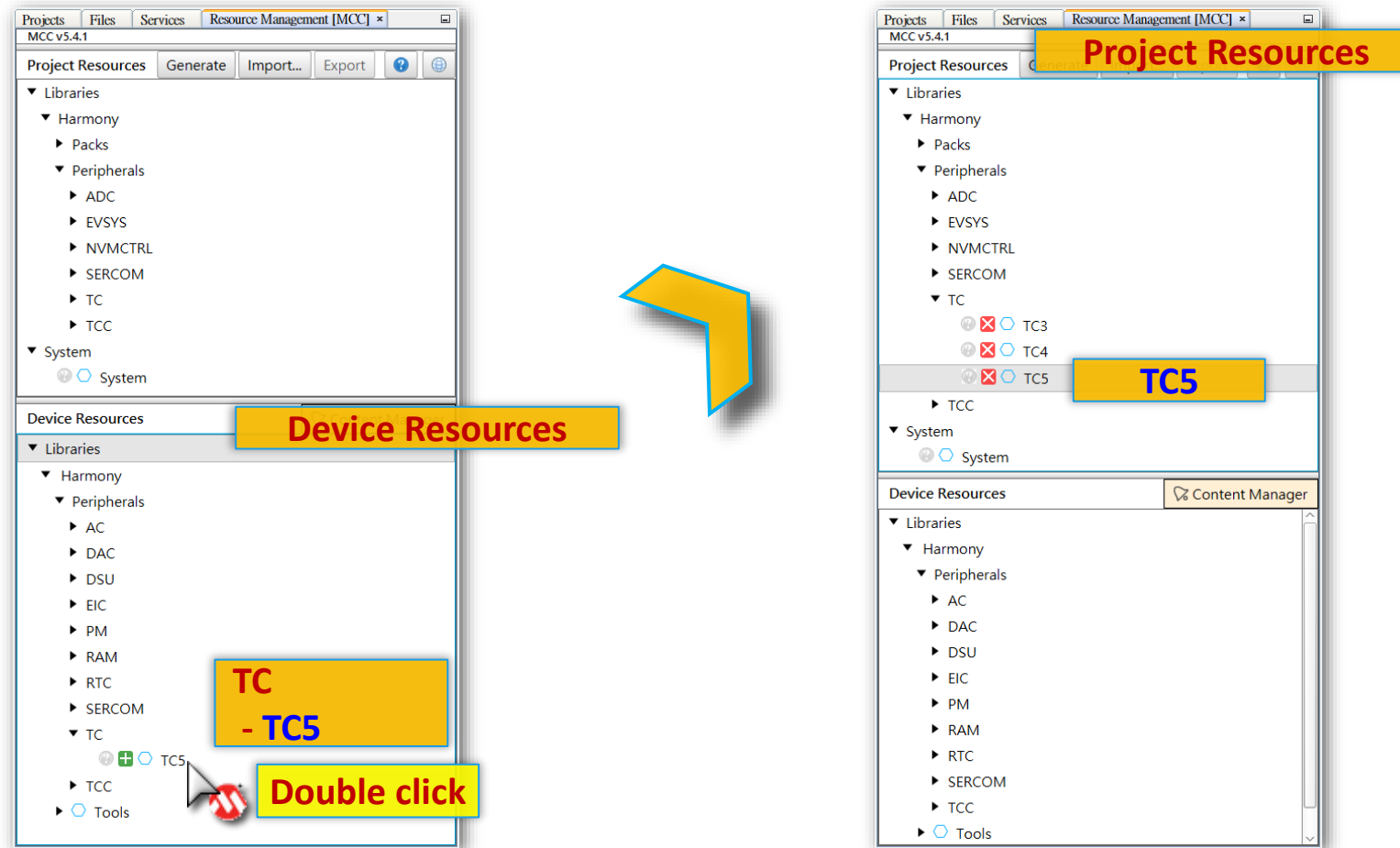
Lab16-4 TC Waveform Capture

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

Lab16-4 TC Waveform Capture

Step 1

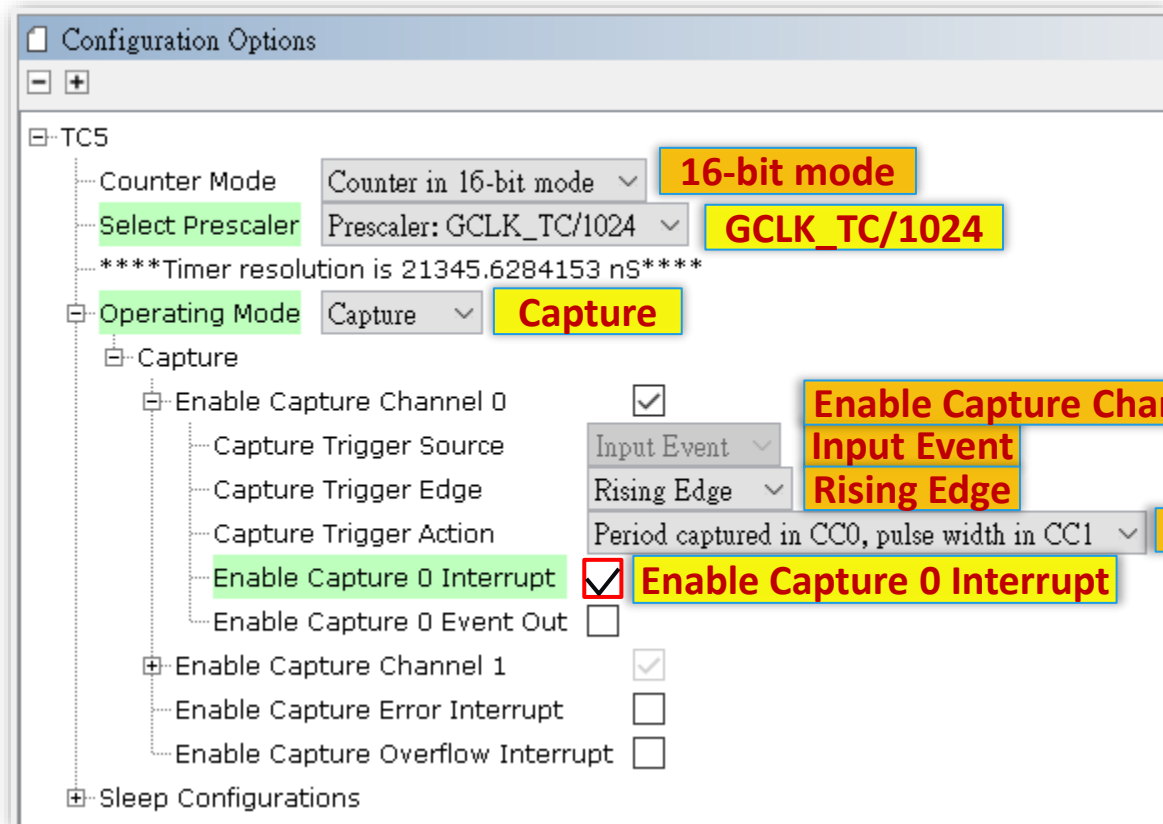
- Find **TC5** component in Device Resources window.
- Double click **TC5** to add to Project Resources window.



Lab16-4 TC Waveform Capture

Step 2

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

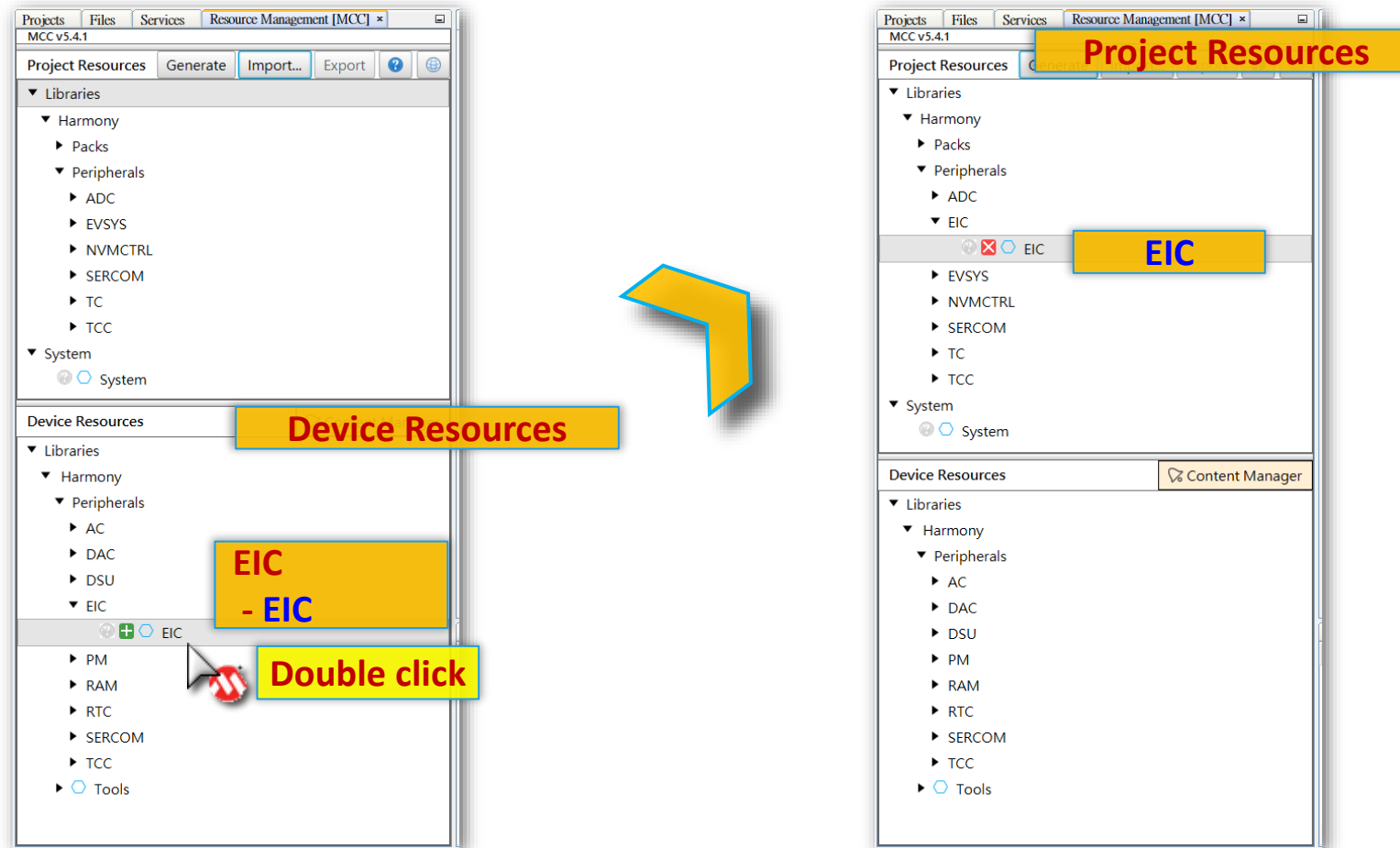


Capture Resolution is :
 $48\text{M}/1024 = 46.875\text{ KHz}$
Max Capture Count is :
0xFFFF (16bits)
Max Capture Period is :
 $0\text{xFFFF} / 46.875\text{KHz} = 1398\text{ ms}$

Lab16-4 TC Waveform Capture

Step 3

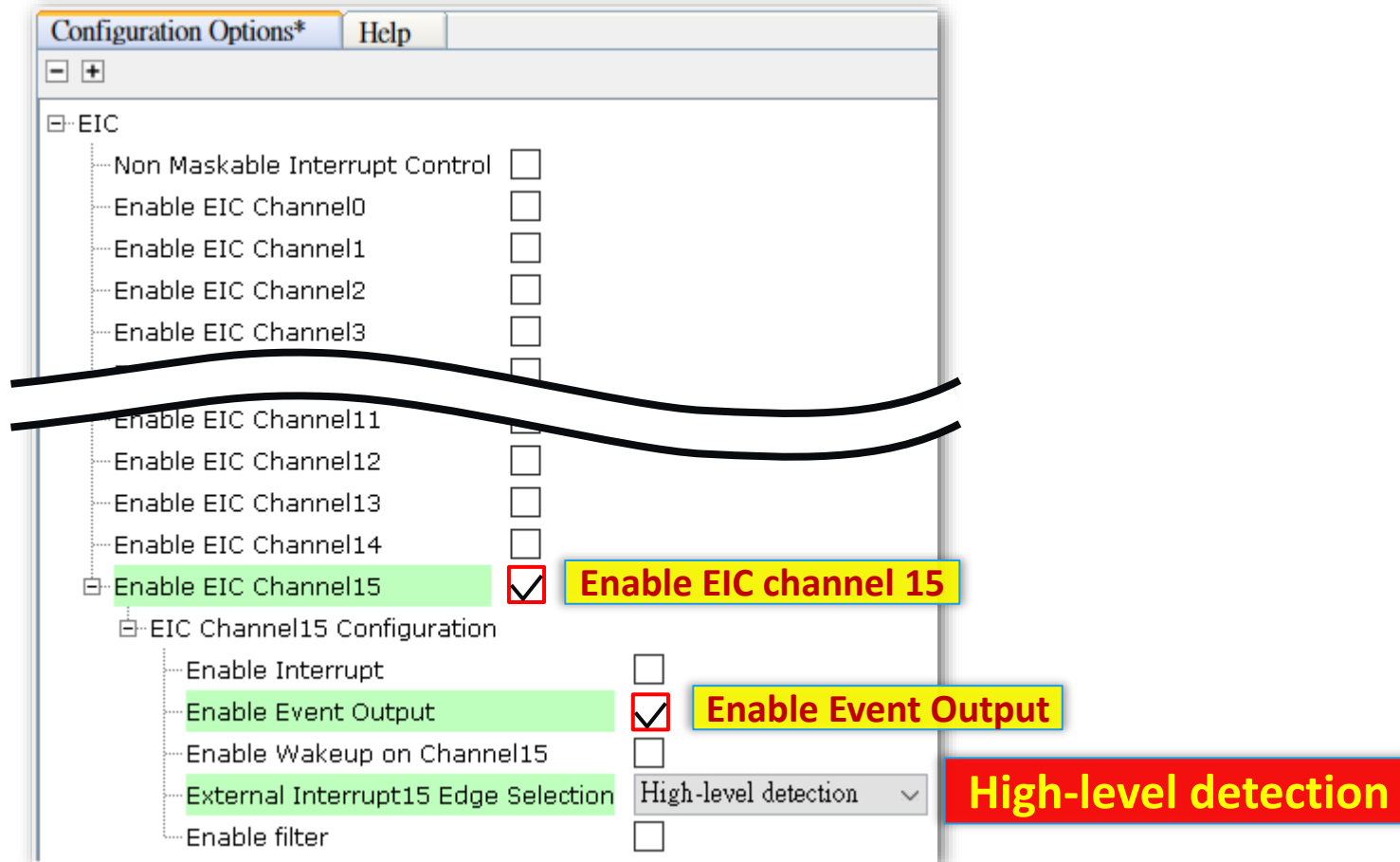
- Find **EIC** component in Device Resources window.
- Double click **EIC** to add to Project Resources window.



Lab16-4 TC Waveform Capture

Step 4

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



Lab16-4 TC Waveform Capture

Step 5

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

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

Left Screenshot: The Pin Table tool shows the package TQFP48. The pin PA15 is selected. The function GCLK_IO1 is currently assigned to PA15. A red box highlights GCLK_IO1, and a mouse cursor is clicking on it with a red 'X' icon, indicating it will be disabled. A yellow box labeled "Click Disable" points to the mouse cursor.

Right Screenshot: The Pin Table tool shows the same package TQFP48. The pin PA15 is selected. The function EIC_EXTINT15 is now assigned to PA15. A red box highlights EIC_EXTINT15, and a mouse cursor is clicking on it with a green checkmark icon, indicating it will be enabled. A yellow box labeled "Click Enable" points to the mouse cursor.

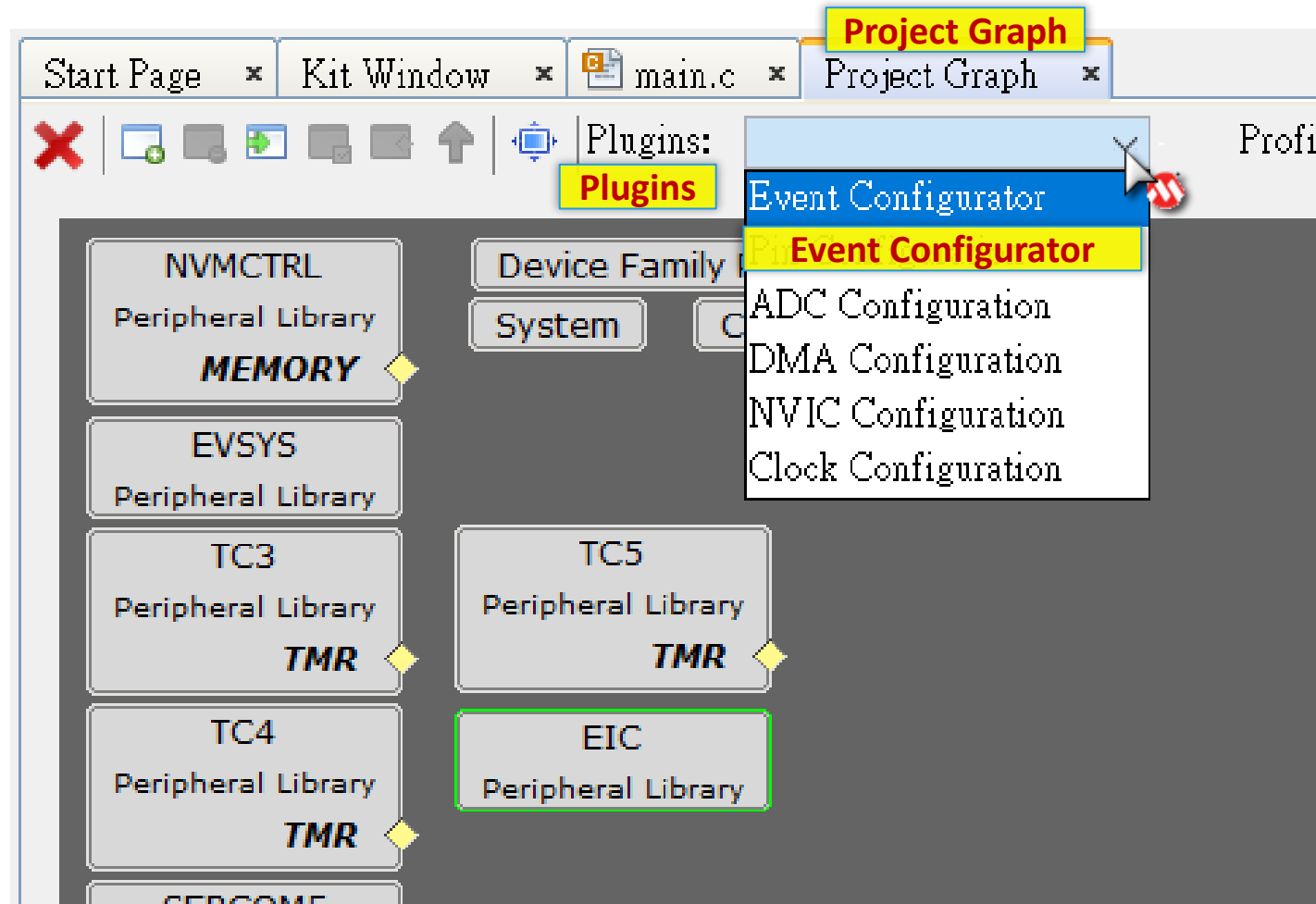
Pin Table Data:

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

Lab16-4 TC Waveform Capture

Step 6

- Find **Event Configurator** in Plugins of **Project Graph** window.



Lab16-4 TC Waveform Capture

Step 7

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

The screenshot shows the 'Event Configurator' window. The title bar reads 'Event Configurator'. The main heading is 'EVENT CONFIGURATOR'. Below this, there are two sections: 'Channel Configuration' and 'Channel 0 Settings'.

Channel Configuration

Channel Number	Event Generator	Event Status	User Ready	Remove Channel
Channel 0 Channel 0	EIC_EXTINT_15 EIC_EXTINT_15 ▼	●	●	🗑️

Below the table, there is a blue button labeled 'Add Channel'. A yellow callout box with the text 'Click on Add Channel' points to this button. There are also red 'no' symbols (a circle with a diagonal line) over the 'Event Status' and 'User Ready' columns of the table.

Channel 0 Settings

Path Selection: ASYNCHRONOUS ▼

Event Edge Selection: NO_EVT_OUTPUT ▼

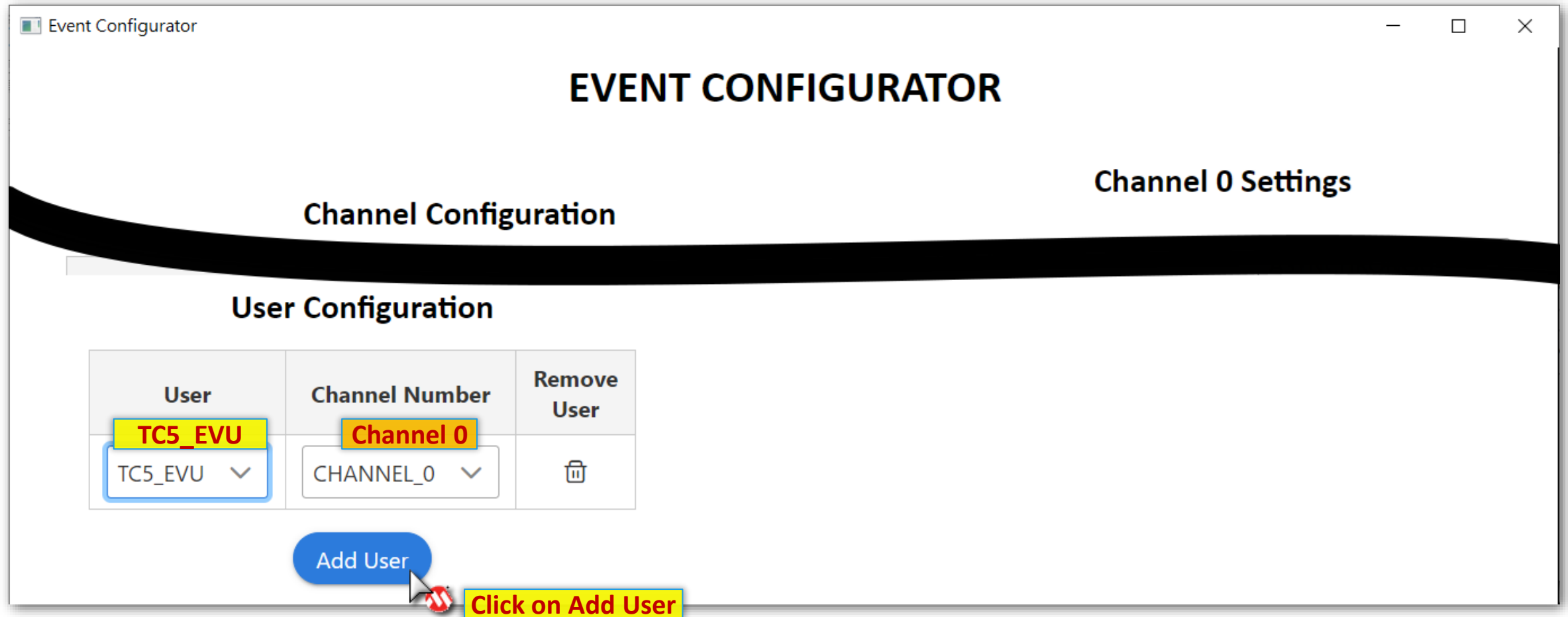
Enable Event Detection Interrupt

Enable Overrun Interrupt

Lab16-4 TC Waveform Capture

Step 8

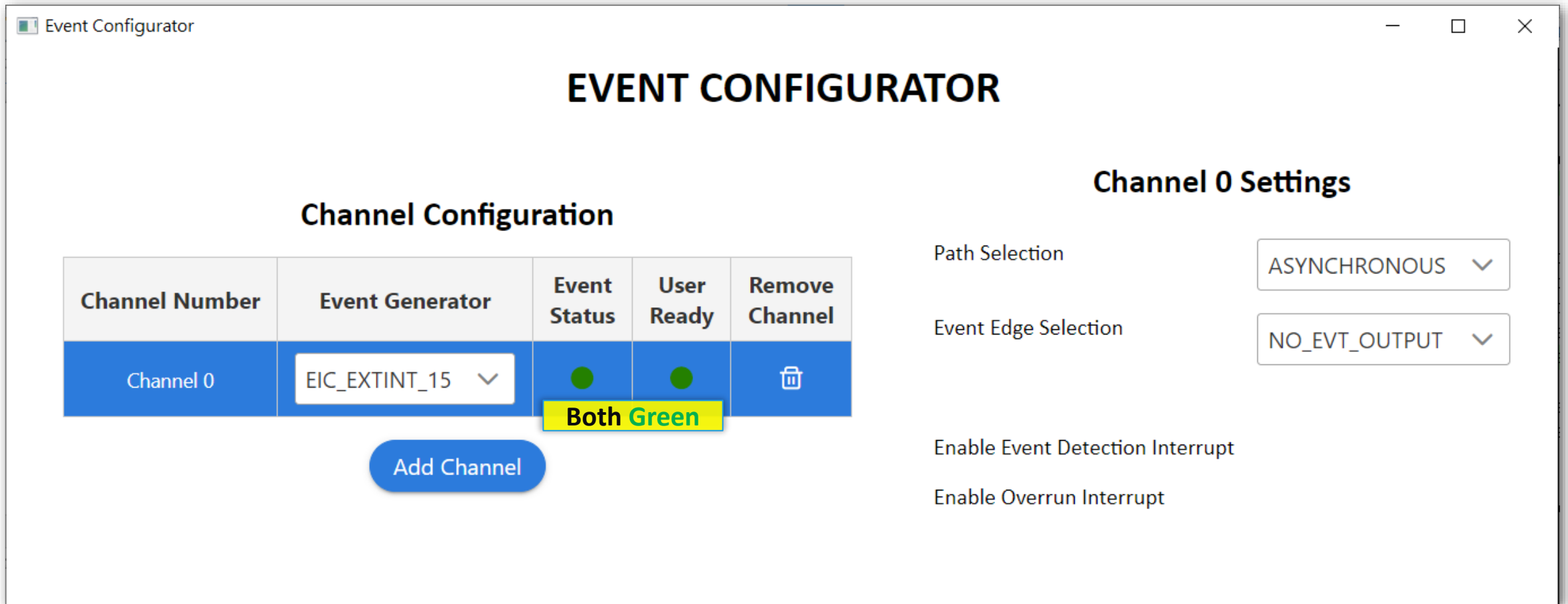
- Add **User** and assign **TC5_EVU** as **Event User** of **Channel 0**.



Lab16-4 TC Waveform Capture

Step 9

- **Event Status** and **User Ready** turned to **green light** means a good Event configuration.



The screenshot shows the 'Event Configurator' window. The main title is 'EVENT CONFIGURATOR'. On the left, under 'Channel Configuration', there is a table with one channel, 'Channel 0'. The table has columns for 'Channel Number', 'Event Generator', 'Event Status', 'User Ready', and 'Remove Channel'. The 'Event Generator' is set to 'EIC_EXTINT_15'. Both 'Event Status' and 'User Ready' are indicated by green circles, with a yellow box labeled 'Both Green' highlighting these two columns. Below the table is a blue 'Add Channel' button. On the right, under 'Channel 0 Settings', there are two dropdown menus: 'Path Selection' set to 'ASYNCHRONOUS' and 'Event Edge Selection' set to 'NO_EVT_OUTPUT'. Below these are two checkboxes: 'Enable Event Detection Interrupt' and 'Enable Overrun Interrupt', both of which are currently unchecked.

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

Both Green

Add Channel

Channel 0 Settings

Path Selection: ASYNCHRONOUS

Event Edge Selection: NO_EVT_OUTPUT

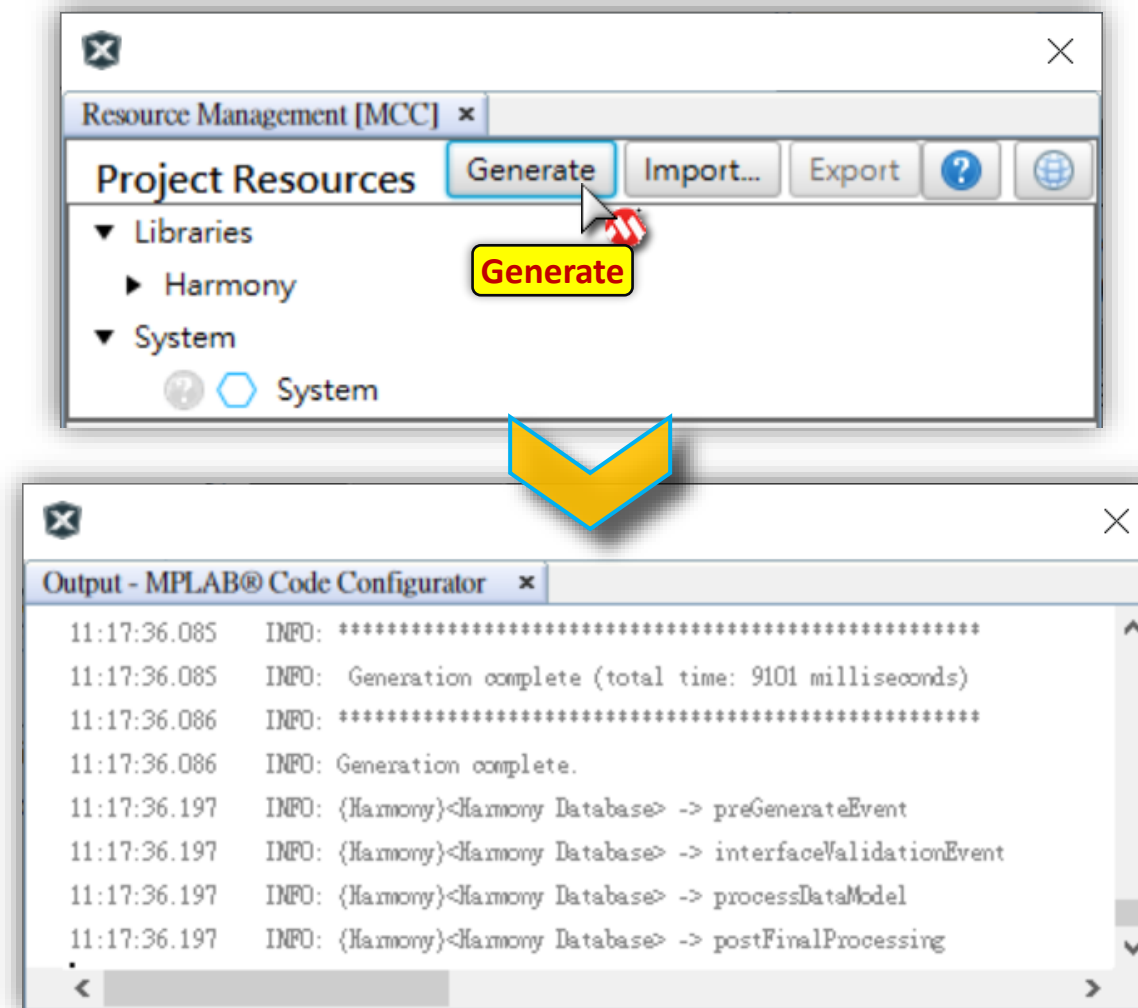
Enable Event Detection Interrupt

Enable Overrun Interrupt

Lab16-4 TC Waveform Capture

Step 10

- Click "**Generate**" Button to Generate your Code.



TC Capture

Interface Routines

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

Lab16-4 TC Waveform Capture

Step 11

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

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

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

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

Lab16-4 TC Waveform Capture

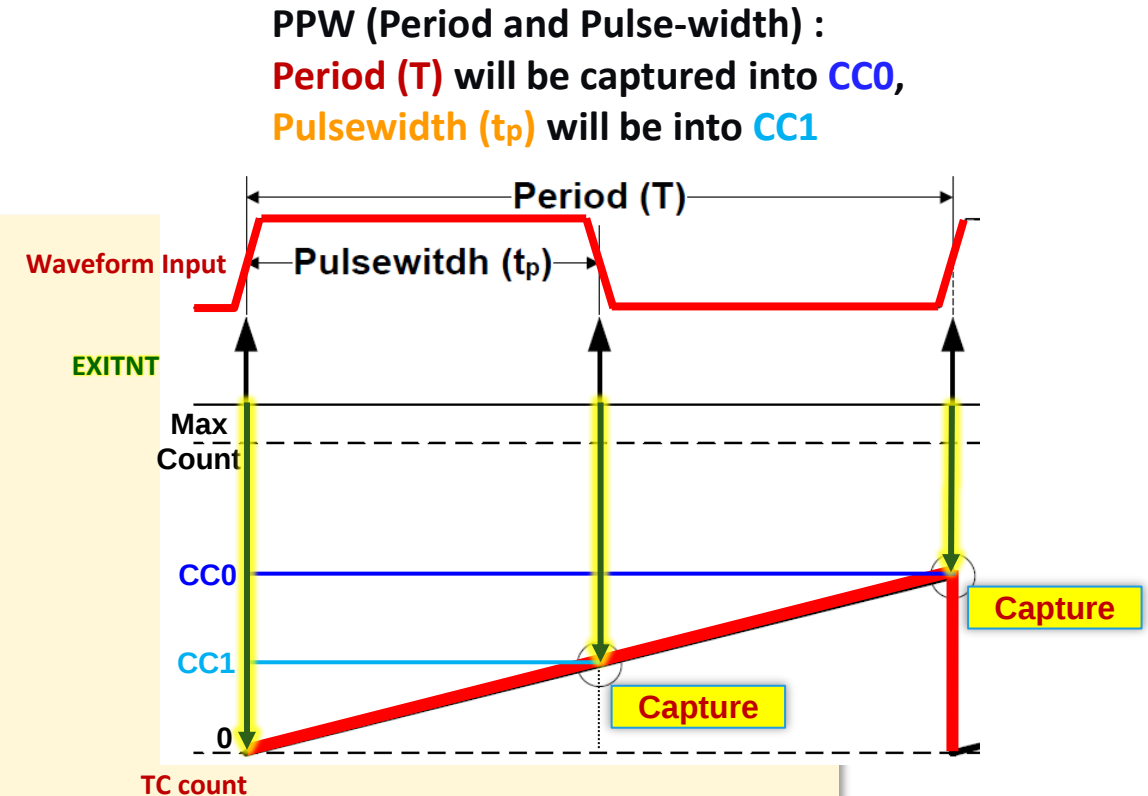
Step 12

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

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

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

        }
    }
}
```



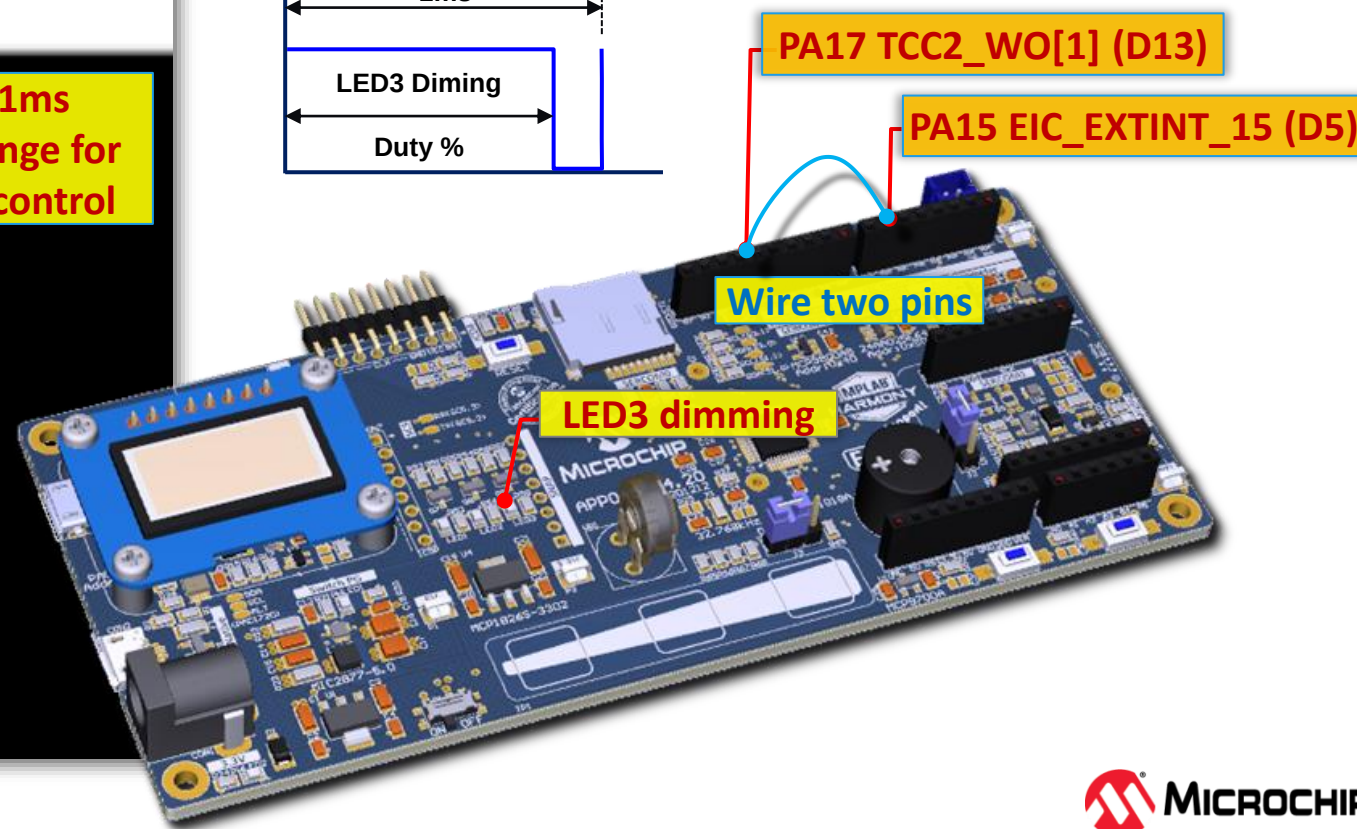
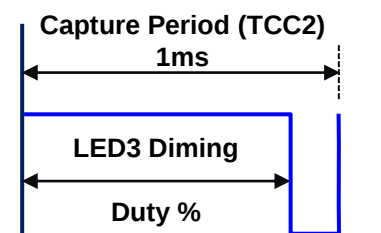
Lab16-4 TC Waveform Capture

Result 1 (Capture TCC2 PWM output)

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

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

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



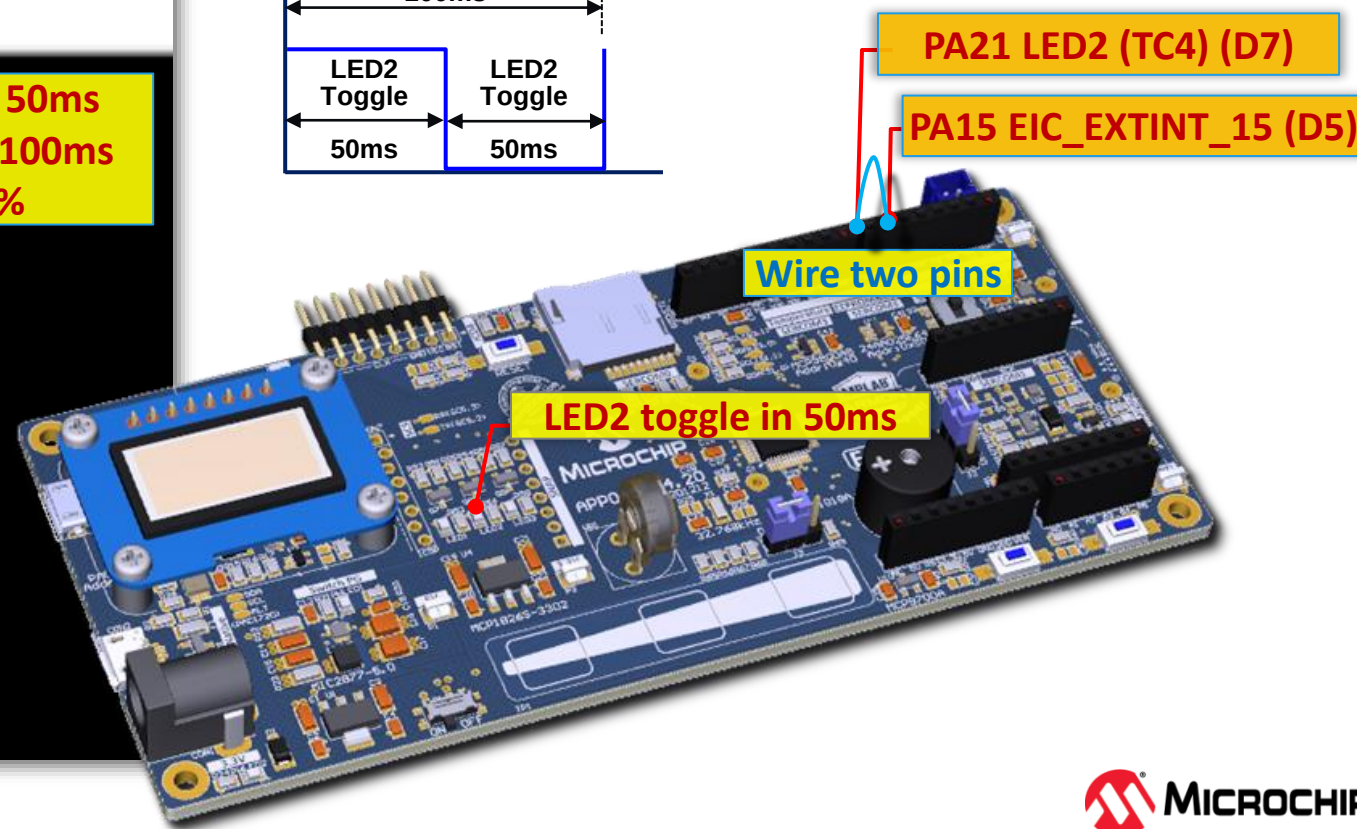
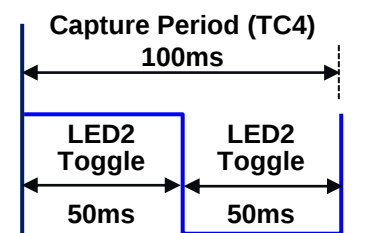
Lab16-4 TC Waveform Capture

Result 2 (Capture LED2 toggle frequency)

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

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

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



Lab16-4 TC Waveform Capture

Discuss

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

VT COM20 - Tera Term VT

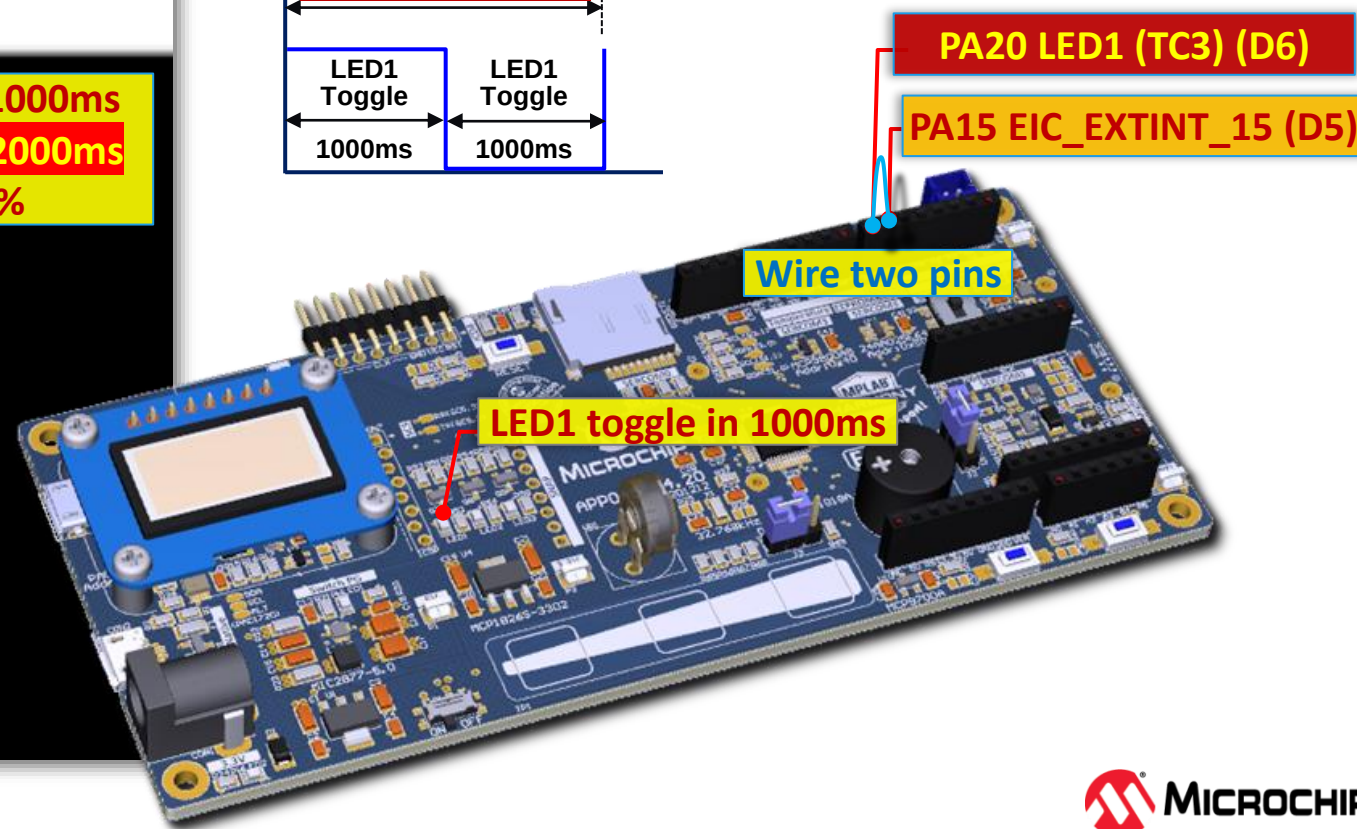
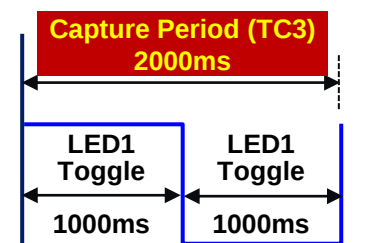
File Edit Setup Control Window Help

```

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

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

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



Lab16-4 TC Waveform Capture

Discuss

- To change the clock source of **TC5** from **GCLK0(48MHz)** to **GCLK1(1KHz)** then try measure the **LED1** frequency again, what's happen ? How about the **LED2** frequency?

COM20 - Tera Term VT

File Edit Setup Control Window Help

```

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

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

TC5 Capture Resolution is :
 $1\text{KHz}/1024 = 1\text{Hz}$
Max Capture Count is :
0xFFFF (16bits)
Max Capture Period is :
 $0xFFFF / 1\text{Hz} = 65535 \text{ sec}$

Peripheral Clock Configuration

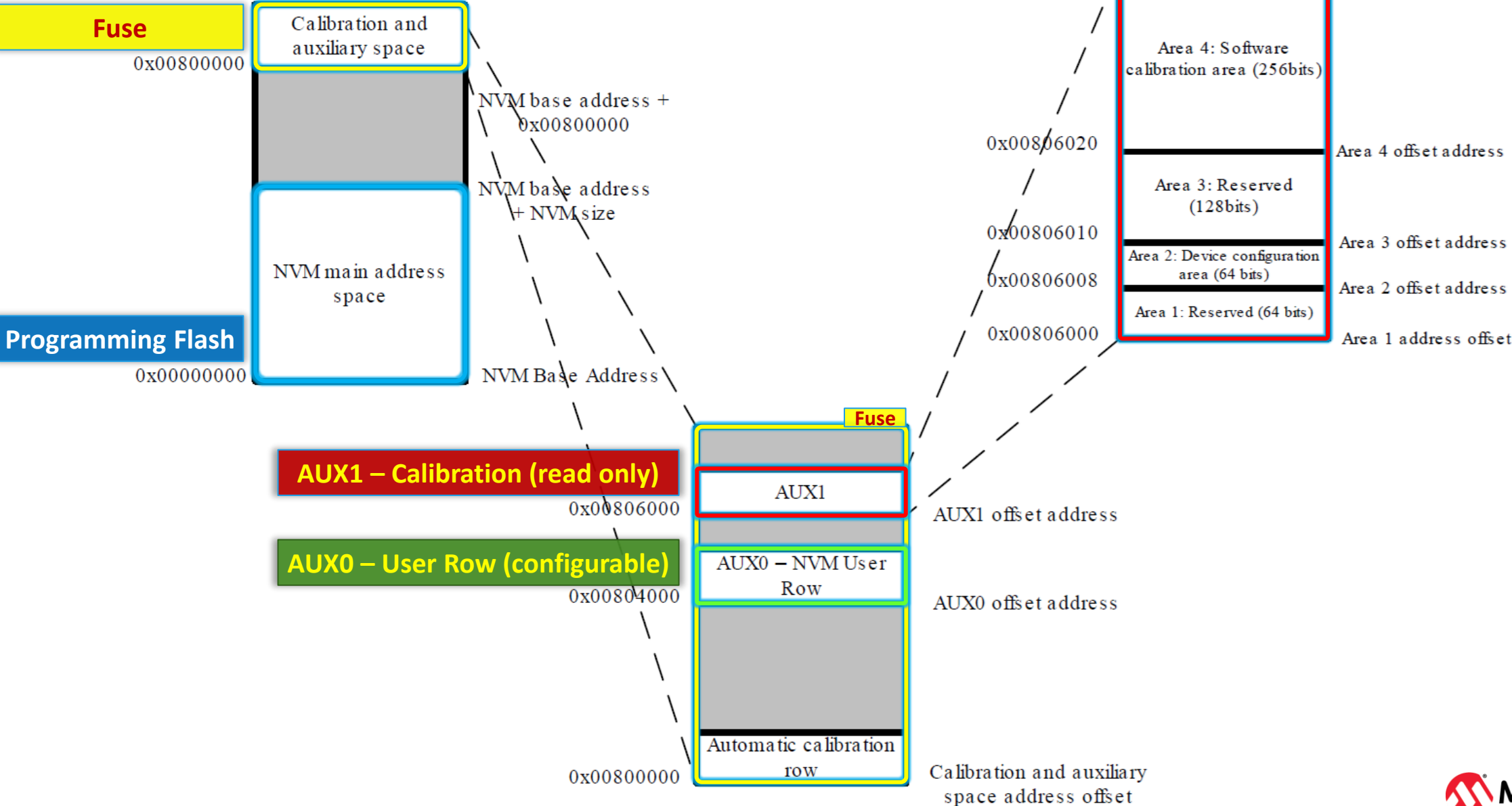
Peripheral	Enable	Source	Peripheral Clock Frequency
SERCOM2_CORE	☐	GCLK0	--
SERCOM3_CORE	☐	GCLK0	--
SERCOM4_CORE	☐	GCLK0	--
SERCOM5_CORE	☒	GCLK0	48,000,000 Hz
SYSCTRL_DFLL48	☒	GCLK1	1,024 Hz
SYSCTRL_FDPLL	☐	GCLK0	--
SYSCTRL_FDPLL32K	☐	GCLK0	--
TC3, TCC2	☒	GCLK0	48,000,000 Hz
TC4, TC5	☒	GCLK1	1,024 Hz
USB	☐	GCLK0	--
WDT	☐	GCLK0	--

TC4, TC5 **GCLK1** **1KHz**

Close

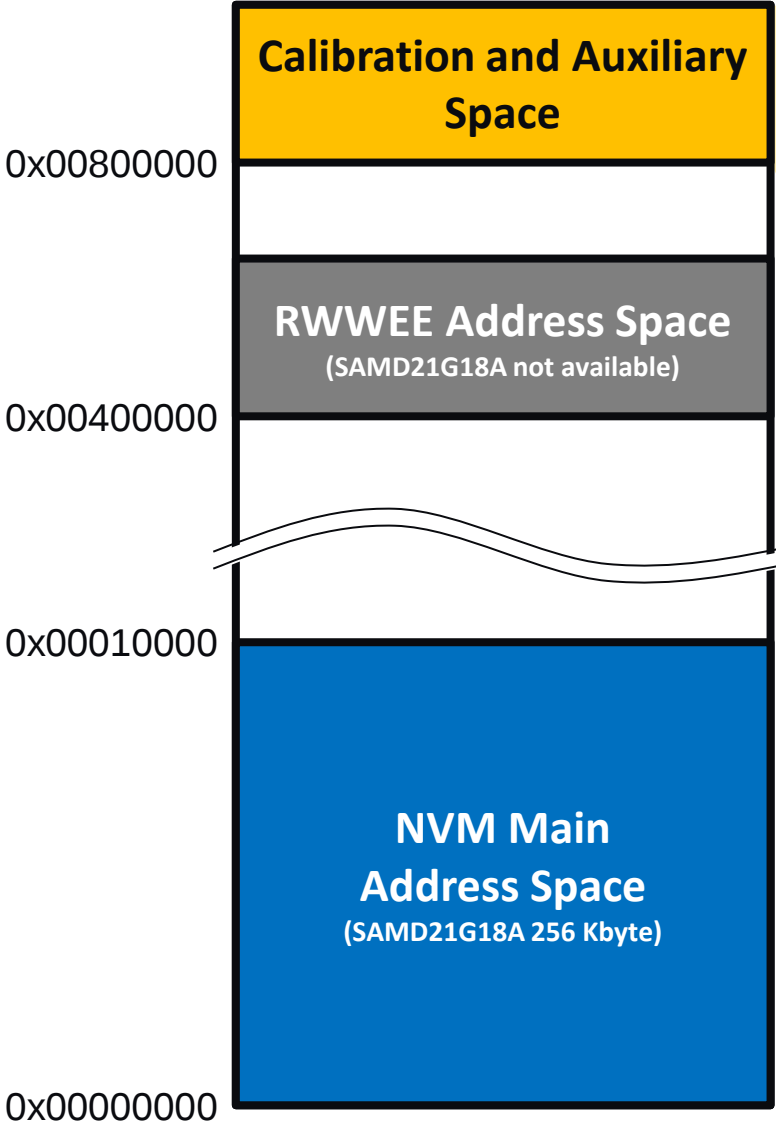
Auxiliary Space of NVM and Fuse Maintains

Auxiliary Space of NVM

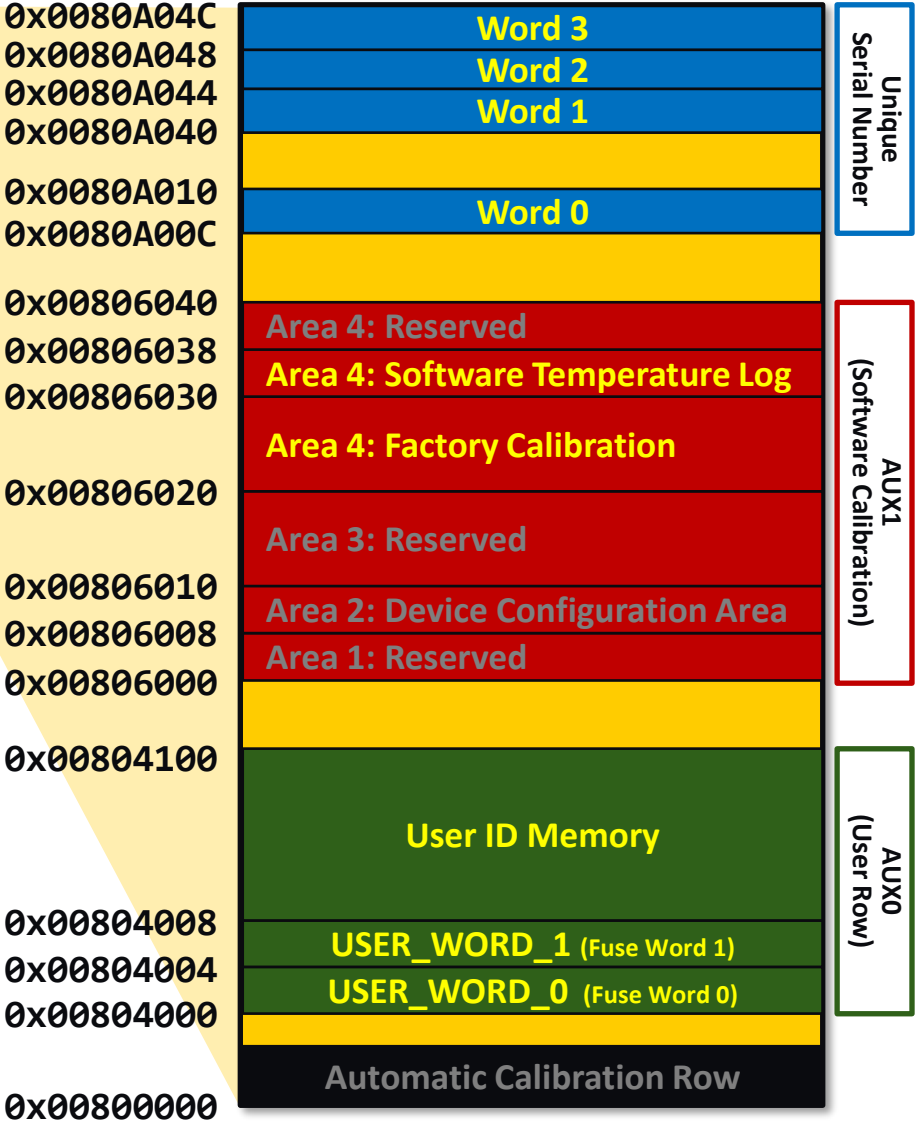


NVM Flash Memory Map

WORD Address

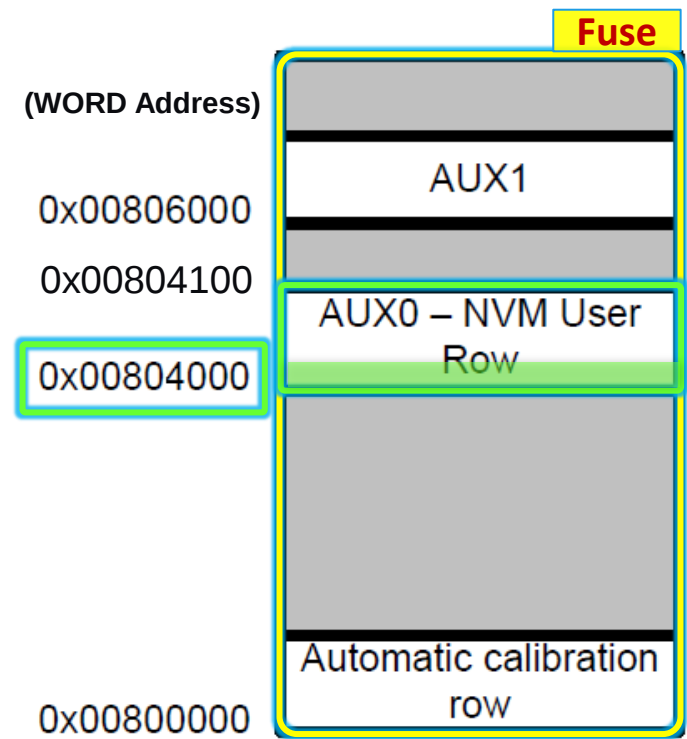


WORD Address



AUX0 (User Row)

User configurable Fuse settings



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

AUX1 (Software Calibration - Factory Calibration)

Factory Calibration Value – read only

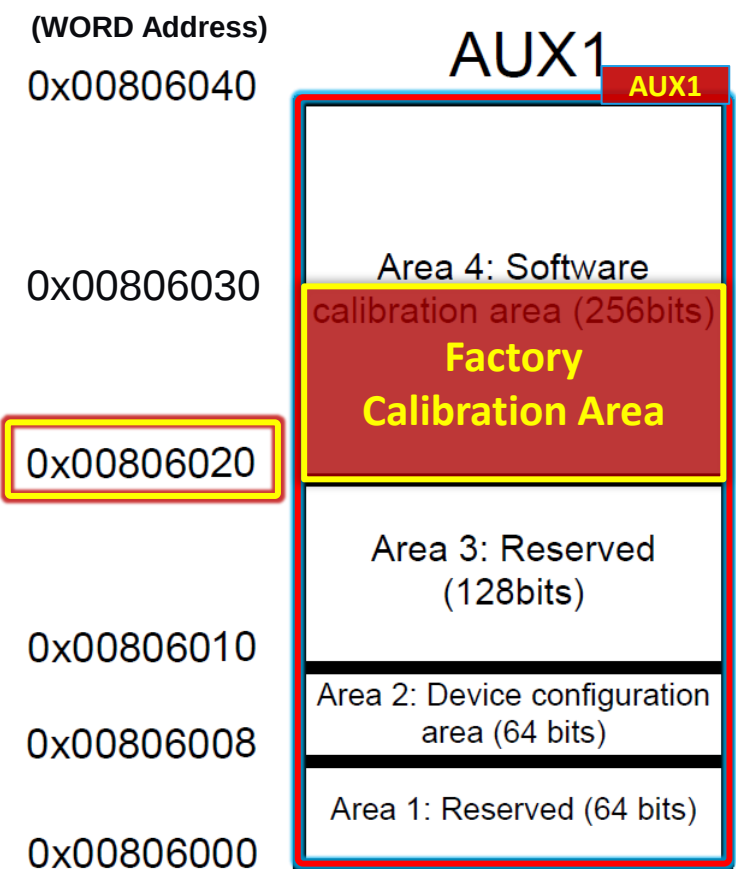


Table 10-5. NVM Software Calibration Area Mapping

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

AUX1 (Software Calibration - Factory Calibration)

Factory Calibration Value – read only

AUX1 Area 4: Factory Calibration Area

OTP4_WORD_3 [127:96]
OTP4_WORD_2 [95:64]
OTP4_WORD_1 [63:32]
OTP4_WORD_0 [31: 0]

Table 10-5. NVM Software Calibration Area Mapping

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

AUX1 (Software Calibration - Factory Calibration)

Read and apply DFL48M Factory Calibration Value

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

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

    /*Load DFLL48M CAL (COARSE) / OTP4_WORD_1 Address >> (58-32) [63:58] 6-bits*/
    uint32_t calibCoarse = (((*(uint32_t*)0x806020U + 1U) >> 26U) & 0x3FU);
    calibCoarse = (((calibCoarse) == 0x3FU) ? 0x1FU : (calibCoarse));

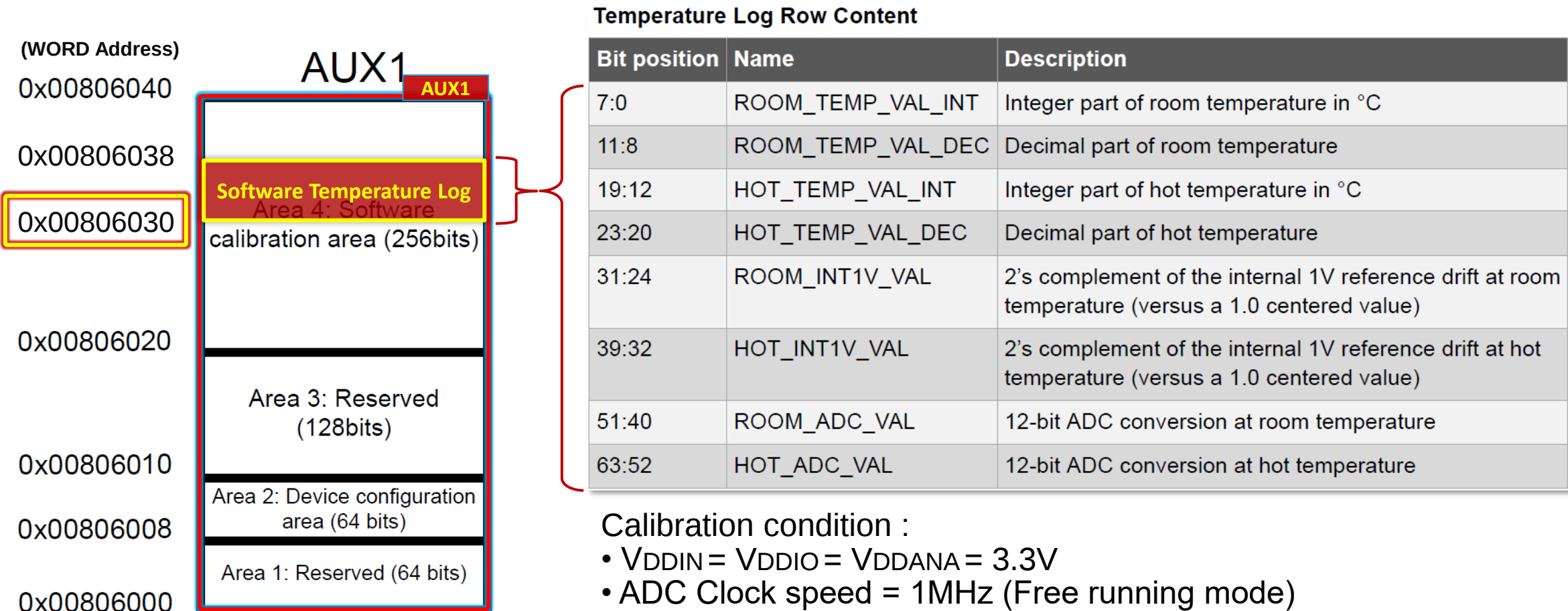
    SYSCTRL_REGS->SYSCTRL_DFLLVAL =
        SYSCTRL_DFLLVAL_COARSE(calibCoarse) |
        SYSCTRL_DFLLVAL_FINE(512U);

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

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

AUX1 (Software Calibration - Software Temperature Log)

Software Temperature Log – Read Only



Calibration condition :

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

Learn this detail in SAM2001ADV chapter
Internal Temperature Sensor

AUX1 (Software Calibration - Software Temperature Log)

Temperature Log could be read and check in Microchip Studio



MPLAB® Snap (BUR190973931) - Device Programming

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

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

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

Software Temperature Log

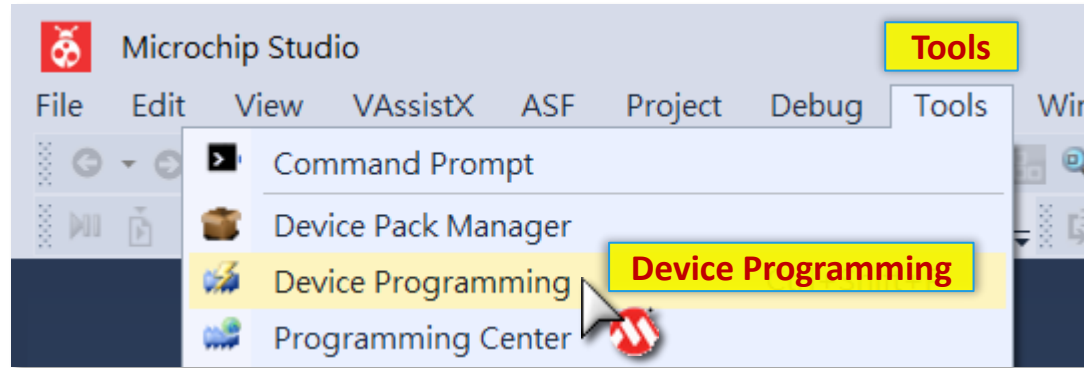
Fuse Register	Value
TEMP_LOG_WORD_0	0xFD07D11E
TEMP_LOG_WORD_1	0xE91B22F8

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

Fuse (Configuration Bits) setting in Microchip Studio

Predecessor is Atmel Studio 7

Fuse (Configuration Bits) setting in Microchip Studio



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

Open Tools

► Device Programming

Tool : MPLAB Snap

Device : ATSAM21G18A

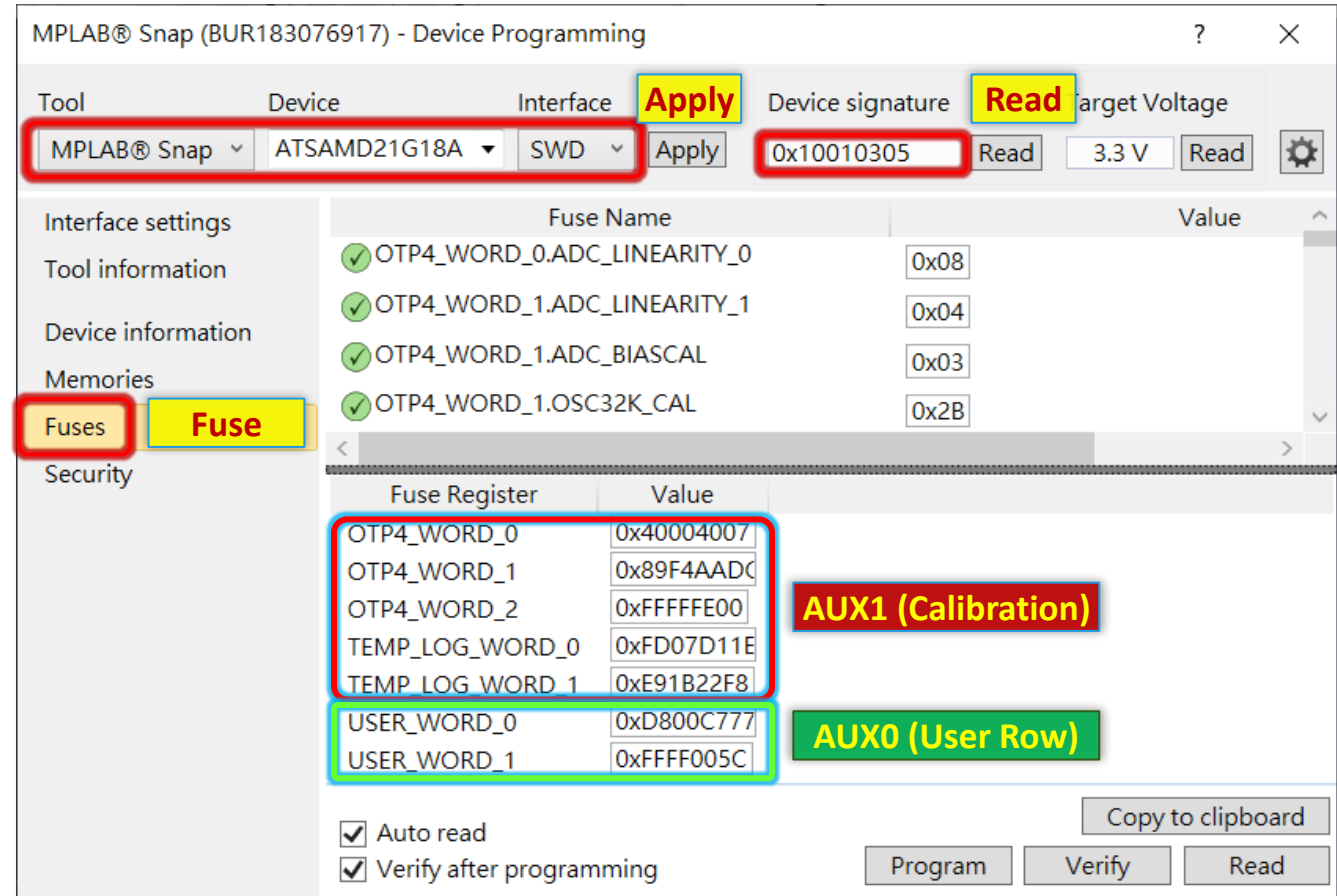
Interface : SWD

► [Apply] **SNAP will Auto switch to AVR mode**

► [Read]

Device signature

► [Fuse]



Fuse (Configuration Bits) setting in Microchip Studio

Programming through Microchip Studio atprogram command line

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

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

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

atprogram -t snap -i swd -d atsamd21g18a erase

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

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

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

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

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

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

Mind the data byte order !!

0xD800C777

0xFFFF005C

AUX0 (User Row)



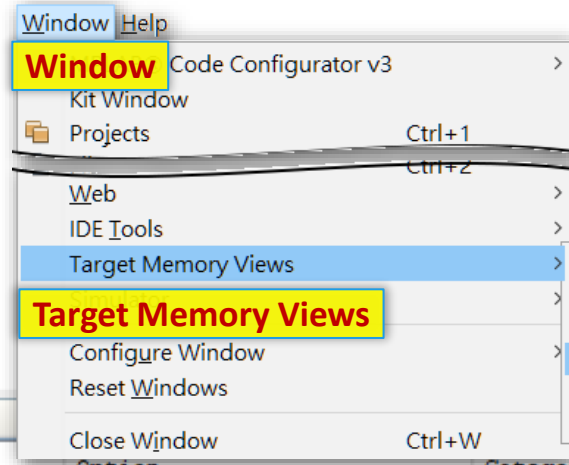
Configuration Bits (Fuse) setting in MPLAB X IDE

Configuration Bits (Fuse) setting in MPLAB X IDE



MPLAB Configuration Bits GUI

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



Configuration Bits will show Red if not to read it firstly

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

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

Must Read firstly before to configure

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

Configuration Bits (Fuse) setting in MPLAB X IDE



MPLAB Configuration Bits GUI

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

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

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

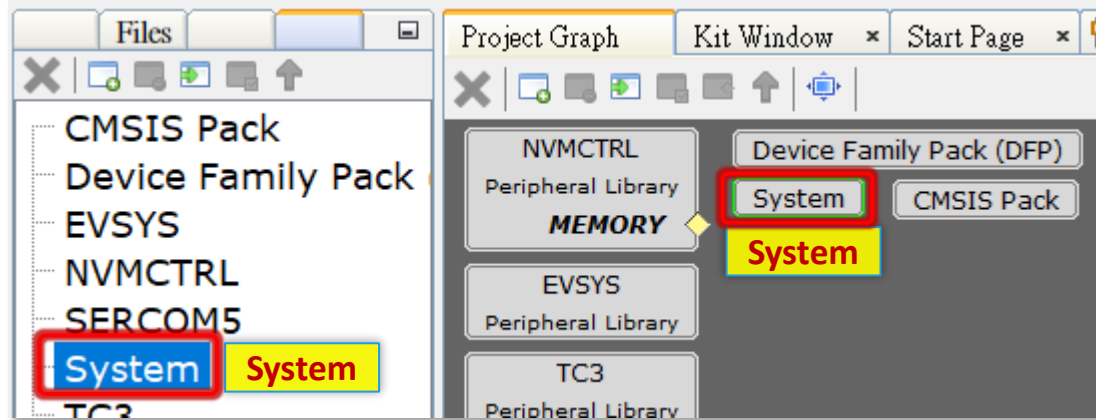
Address	Name	Value	Field	Option	Category	Setting
0080_4000	USER_WORD_0	D800C777	-	-	-	-
7	NVMCTRL_BOOTPROT	SIZE_0BYTES	Bootloader Size	0 Bytes		
7	NVMCTRL_EEPROM_SIZE	SIZE_0BYTES	EEPROM Size	0 Bytes		
7	BOD33_USERLEVEL	User range: 0x0 - 0x3F	BOD33 User Level	Enter Hexadecimal value		
1	BOD33_EN	ENABLED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

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

Configuration Bits (Fuse) setting in MPLAB X IDE

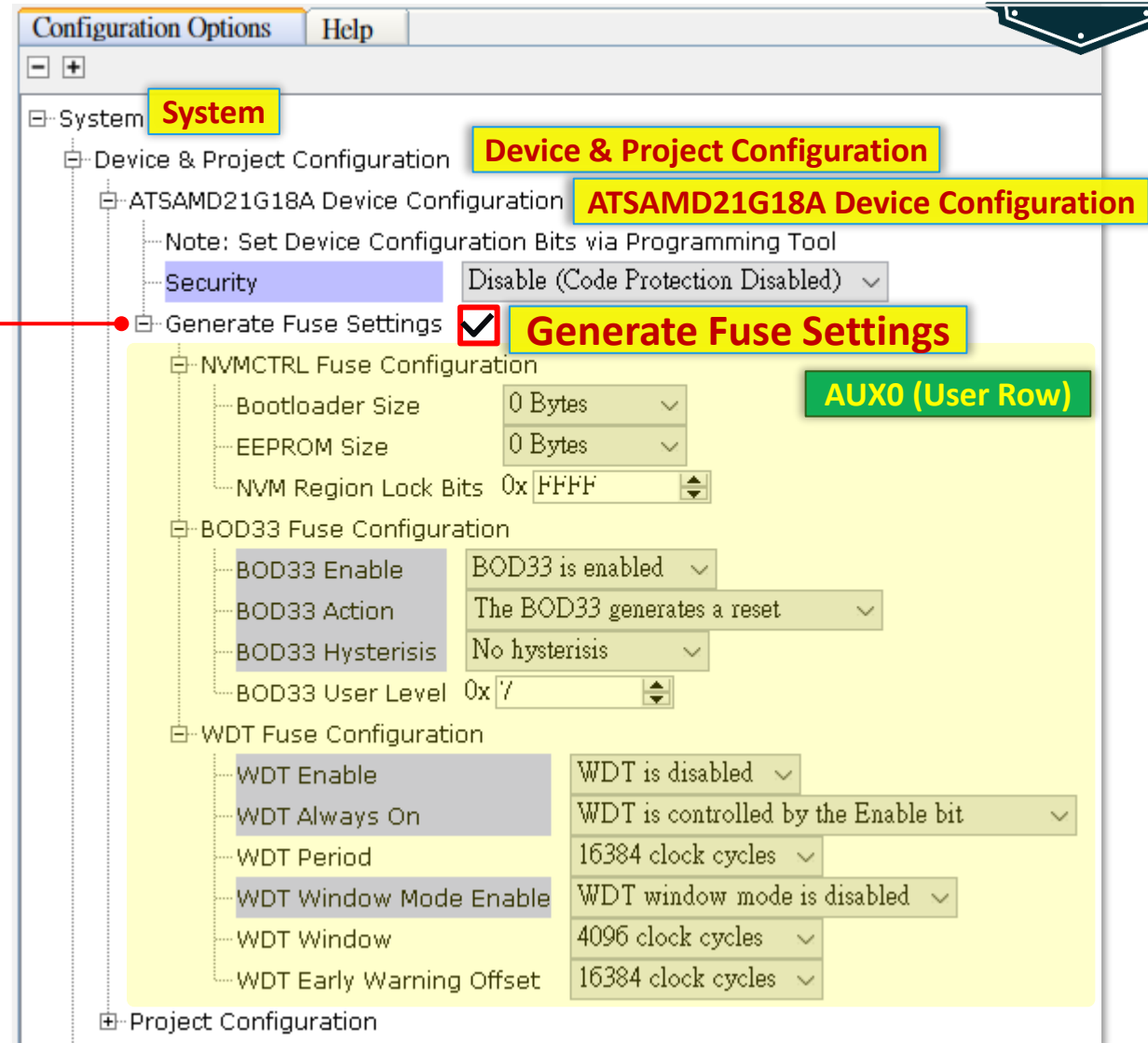


Generate Fuse Settings in Harmony



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

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

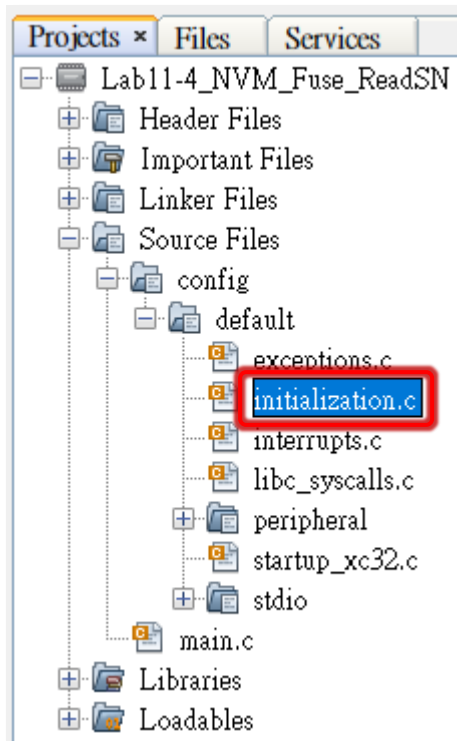


Configuration Bits (Fuse) setting in MPLAB X IDE



Generate Fuse Settings in Harmony

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



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

AUX0 (User Row)

Configuration Bits (Fuse) setting in MPLAB X IDE



Generate Fuse Settings in Harmony

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

Generate Fuse Settings ☐

Project.hex

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

Generate Fuse Settings ☒

Project.hex

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

AUX0 (User Row) WORD 1

AUX0 (User Row) WORD 2

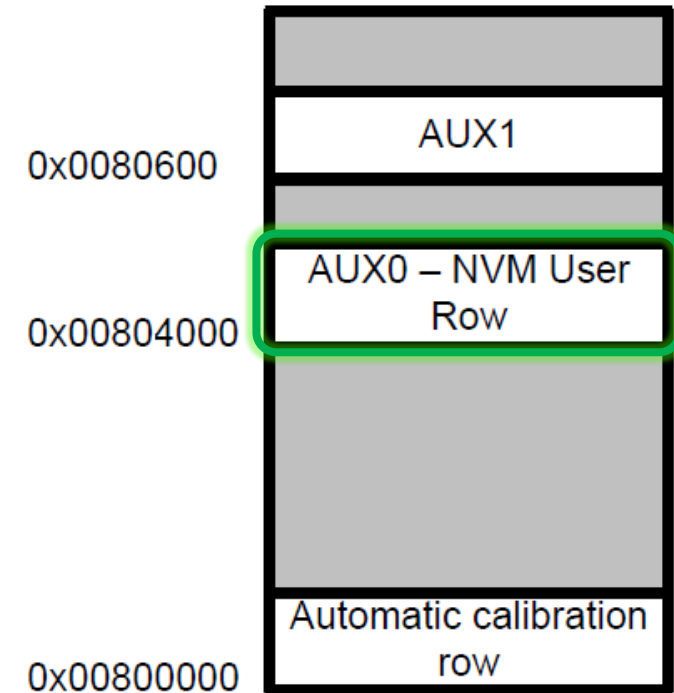
Configuration Bits (Fuse) setting in MPLAB X IDE



Bit format analysis of Fuse settings in HEX image

0440000077c700d8a6

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



Configuration Bits (Fuse) setting in MPLAB X IDE



Bit format analysis of Fuse settings in HEX image

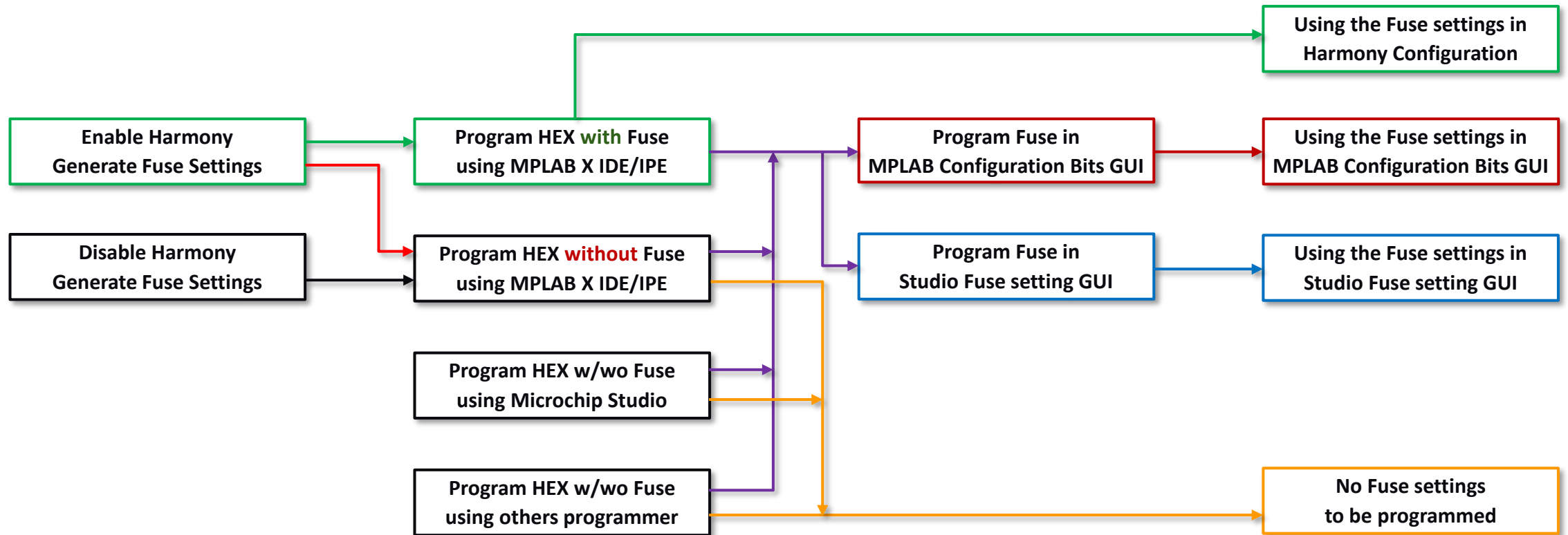
Bit Position	Name
2:0	BOOTPROT
3	Reserved
6:4	EEPROM
7	Reserved
13:8	BOD33 Level
14	BOD33 Enable
16:15	BOD33 Action
24:17	Reserved
25	WDT Enable
26	WDT Always-On
30:27	WDT Period
34:31	WDT Window
38:35	WDT EWOFFSET
39	WDT WEN
40	BOD33 Hysteresis
41	Reserved
47:42	Reserved
63:48	LOCK

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

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

Configuration Bits (Fuse) setting in MPLAB X IDE

Configuration Bits GUI and Harmony Generated Fuse setting in HEX



Lab11-4 Read Unique Serial Number

- Please continue from [SAM2001 Lab11 \(SERCOM - UART TxRx Interrupt Callback with myprintf\)](#)
- In SAM D20 / SAM D21 / SAM R21 devices, each device has a unique 128-bit serial number which is a concatenation of four 32-bit words contained at the following addresses:

Word 0: 0x0080A00C

Word 1: 0x0080A040

Word 2: 0x0080A044

Word 3: 0x0080A048

- Try to read the unique serial number from your SAMD21 MCU and output as below

"Unique serial number : (4 Words)"

"0x0080A00C : 0x55574129"

"0x0080A010 : 0x514E5357"

"0x0080A014 : 0x4C202020"

"0x0080A018 : 0xFF130E33"

"- done"

The unique serial number will vary depending on the device

- **Let's go !**

Lab11-4 Read Unique Serial Number

Step1

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

void USART5_Transmit( uintptr_t context )
...

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

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

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

Lab11-4 Read Unique Serial Number

Step2

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

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

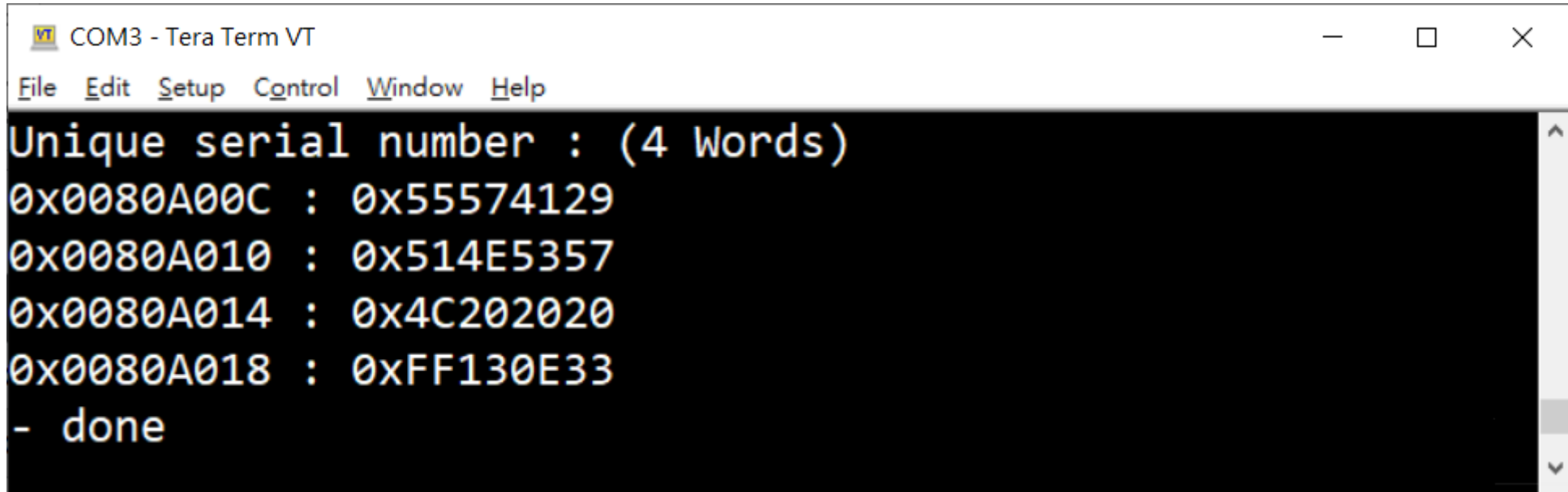
int main ( void )
{
    ...
    SERCOM5_USART_Read( USART5_ReceiveData, 1 );

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

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

Lab11-4 Read Unique Serial Number

Result



A screenshot of a Tera Term VT window titled "COM3 - Tera Term VT". The window has a menu bar with "File", "Edit", "Setup", "Control", "Window", and "Help". The main display area shows the following text:

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

The text is displayed in a monospaced font on a black background. A vertical scrollbar is visible on the right side of the text area.

⚙️ Lab11-5 Read NVM AUX0 (User Row)

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

“NVM-AUX0 User Row : (64 Words)”

“- Word 0 and Word 1 ”

“0x00804000 : 0xD800C777 ”

“0x00804004 : 0xFFFF005C ”

“- User ID Memory ”

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

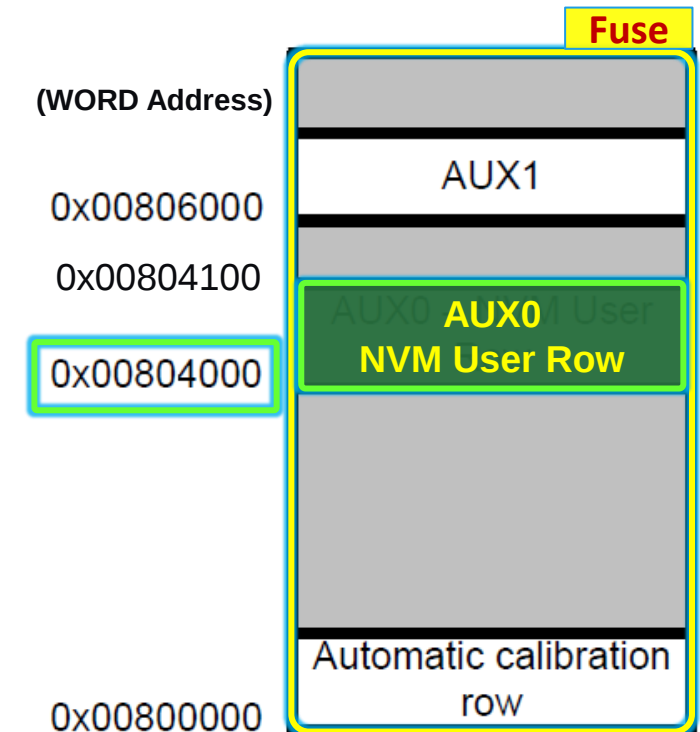
“0x00804018 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF ”

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

“ ”

“0x008040F8 : 0xFFFFFFFF 0xFFFFFFFF ”

“- done”



• **Let's go !**

⚙️ Lab11-5 Read NVM AUX0 (User Row)

Step1

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

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

Lab11-5 Read NVM AUX0 (User Row)

Step2

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

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

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

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

✖ Lab11-5 Read NVM AUX0 (User Row)

Result

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

⚙️ Lab11-6 Read NVM AUX1 (Software Calibration)

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

“NVM-AUX1 : (16 Words) ”

“- OTP Area 1 (Reserved) ”

“- OTP Area 2 (Device Configuration Area) ”

“- OTP Area 3 (Reserved) ”

“- OTP Area 4 (Software Calibration Area) ”

“0x00806020 : 0x40004007”

“ ”

“ > Software Temperature Log row”

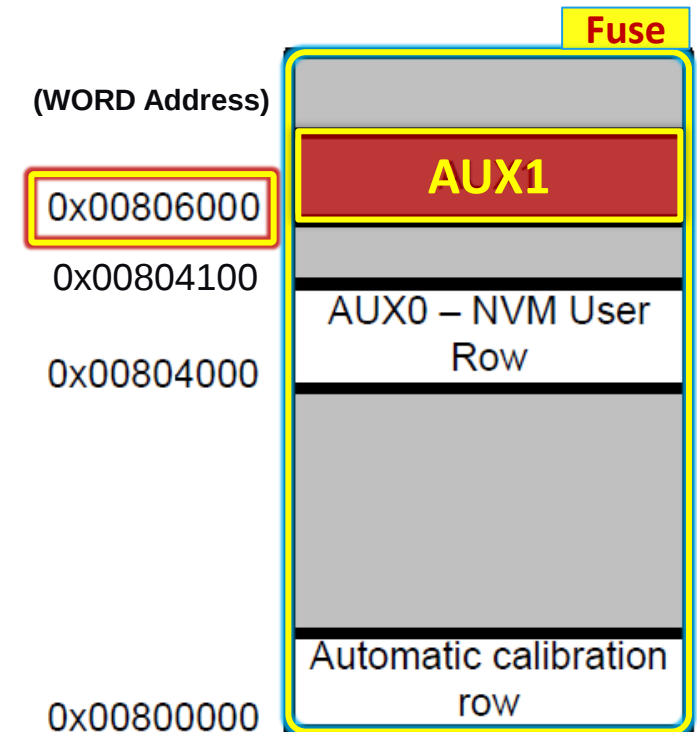
“0x00806030 : 0xFD07D11E”

“ ”

“ > Reserved”

“- done”

• **Let's go !**



⚙️ Lab11-6 Read NVM AUX1 (Software Calibration)

Step1

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

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

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

⚙️ Lab11-6 Read NVM AUX1 (Software Calibration)

Step1

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

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

Lab11-6 Read NVM AUX1 (Software Calibration)

Step2

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

int main ( void )
{
    ...

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

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

⚙️ Lab11-6 Read NVM AUX1 (Software Calibration)

Result

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

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

User ID Memory in NVM AUX0 (User Row)

User ID Memory in NVM AUX0 (User Row)

User Fuse WORD 0, WORD1 and User ID Memory

NVM-AUX0 User Row : (64 Words)

- Word 0 and Word 1

0x00804000 : 0xD800C777

0x00804004 : 0xFFFF005C

- User ID Memory

0x00804008 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF

0x00804018 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF

0x00804028 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF

0x00804038 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF

0x00804048 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF

0x00804058 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF

0x00804068 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF

0x00804078 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF

0x00804088 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF

0x00804098 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF

0x008040A8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF

0x008040B8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF

0x008040C8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF

0x008040D8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF

0x008040E8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF

0x008040F8 : 0xFFFFFFFF 0xFFFFFFFF

- done

0x00804100

User ID Memory
(64 - 2) WORDs
248 Bytes

Fuse WORD 1

Fuse WORD 0

0x00804008

0x00804004

0x00804000

(WORD Address)

0x00806000

0x00804100

0x00804000

0x00800000

Fuse

AUX1

AUX0 AUX0 User
NVM User Row

Automatic calibration
row

User ID Memory in NVM AUX0 (User Row)

User Microchip Studio to access User ID Memory



MPLAB® Snap (BUR183076917) - Device Programming

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

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

Device: Erase Chip | Erase now

Flash (256 KB)

☐ Erase Flash before programming
☐ Verify Flash after programming
Advanced

User Page (256 bytes)

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

Program | Verify | Read...

Save As

本機 > 本機 > 桌面

組合管理 | 新增資料夾

本機磁碟 (C:) | 本機磁碟 (D:) | 網路

名稱: Document.userpage

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

隱藏資料夾

存檔(S) | 取消

Save

Read

User ID Memory in NVM AUX0 (User Row)

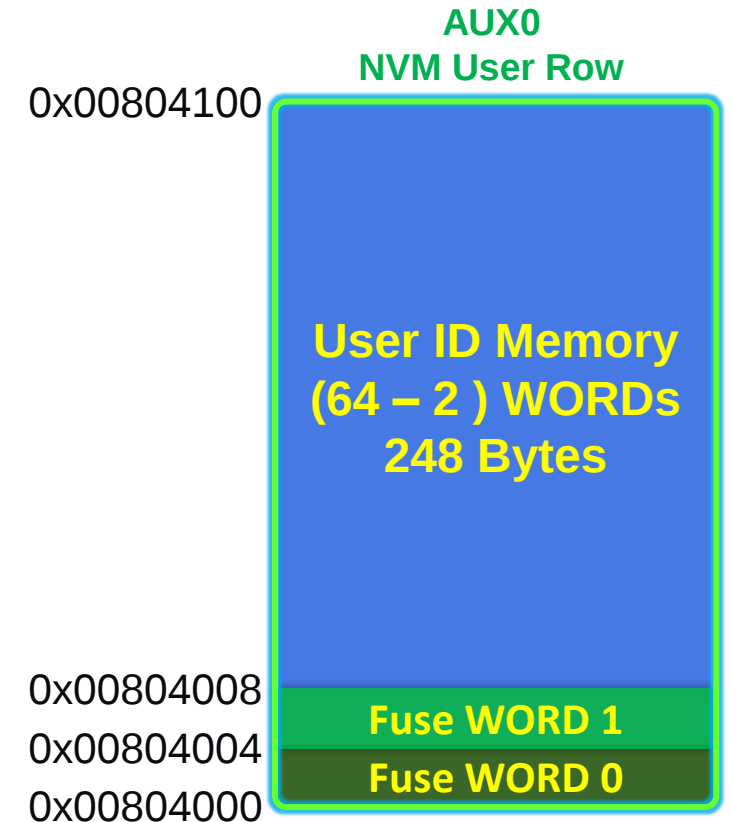
User Microchip Studio to access User ID Memory



Document.userpage

```
:02000000400807A
:1040000077C700D85C00FFFFFFFFFFF48
:10401000FFFFFFFFFFFFFFFFFFFFFFFFFBB0
:10402000FFFFFFFFFFFFFFFFFFFFFFFFFA0
:10403000FFFFFFFFFFFFFFFFFFFFFFFFF90
:10404000FFFFFFFFFFFFFFFFFFFFFFFFF80
:10405000FFFFFFFFFFFFFFFFFFFFFFFFF70
:10406000FFFFFFFFFFFFFFFFFFFFFFFFF60
:10407000FFFFFFFFFFFFFFFFFFFFFFFFF50
:10408000FFFFFFFFFFFFFFFFFFFFFFFFF40
:10409000FFFFFFFFFFFFFFFFFFFFFFFFF30
:1040A000FFFFFFFFFFFFFFFFFFFFFFFFF20
:1040B000FFFFFFFFFFFFFFFFFFFFFFFFF10
:1040C000FFFFFFFFFFFFFFFFFFFFFFFFF00
:1040D000FFFFFFFFFFFFFFFFFFFFFFFFFF0
:1040E000FFFFFFFFFFFFFFFFFFFFFFFFFE0
:1040F000FFFFFFFFFFFFFFFFFFFFFFFFFD0
:000000001FF
```

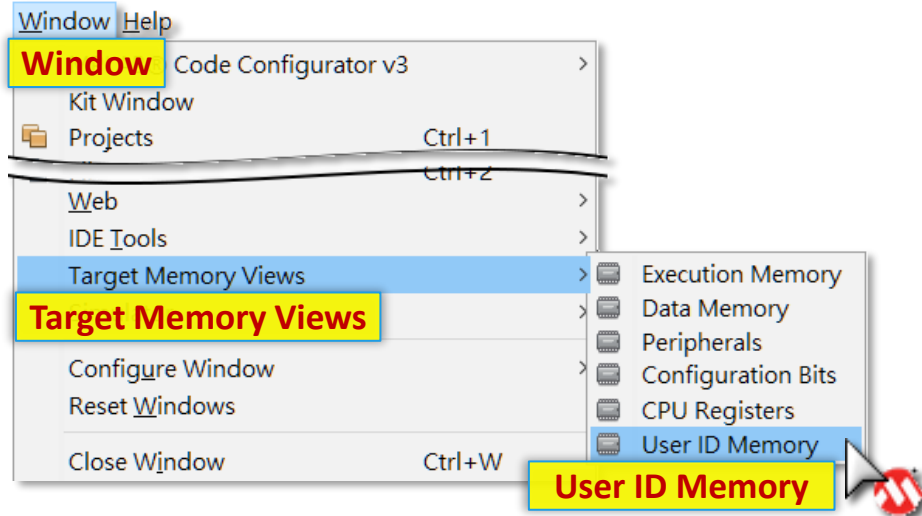
Microchip Studio IDE to store Fuse and User ID Memory together



User ID Memory in NVM AUX0 (User Row)



User MPLAB X IDE to access User ID Memory



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

Memory User ID Memory ▾
User ID Memory

MPLAB X IDE separate to shows
User ID Memory in GUI

AUX0
NVM User Row

0x00804100

User ID Memory
(64 – 2) WORDs
248 Bytes

0x00804008

0x00804004

0x00804000

Fuse WORD 1

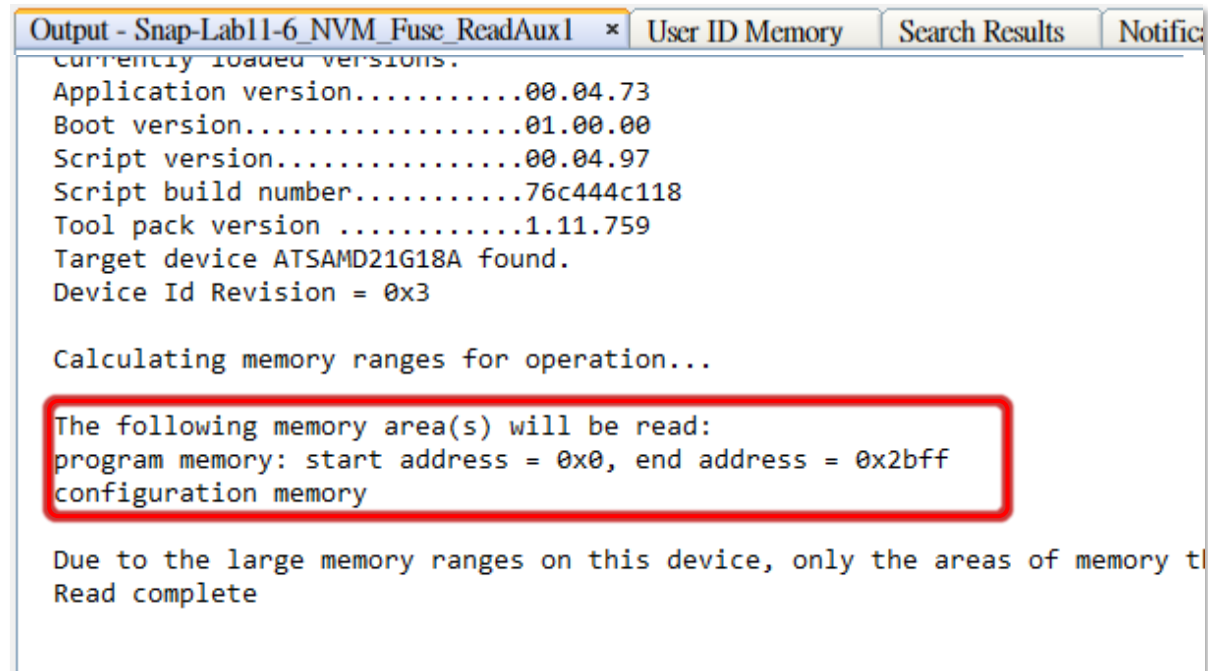
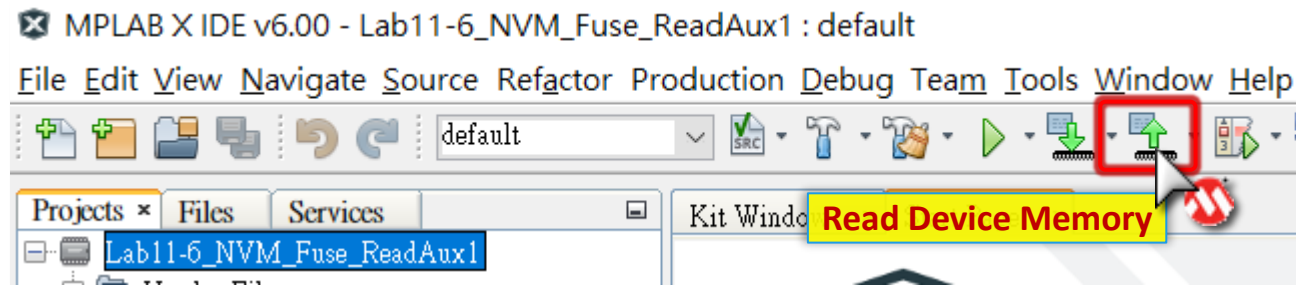
Fuse WORD 0

We cannot manually read and
write User ID Memory here

User ID Memory in NVM AUX0 (User Row)



Read User ID Memory in MPLAB X IDE



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

Memory User ID Memory v

Not Updated after Read

User ID Memory in NVM AUX0 (User Row)



Read User ID Memory in MPLAB X IDE

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

File Edit View Navigate Source Refactor Production Debug T

default

Projects x Files Services Kit Window x St

Lab11-6_NVM_Fuse_ReadAux1

Header Files

Important Files

Linker Files

Source Files

Libraries

Loadables

Navigator -6_NVM_Fuse_R

Lab11-6_NVM_Fuse_Re

Project Type: Applica

Device

ATSAMD21G18A

Checksum: Blank,

32 CRC32: Hex file t

Packs

SAMD21_DFP (3

CMSIS (5.8.0)

Compiler Toolchain

XC32 (v4.10) [C:

Production Image:

Properties

Properties

Project Properties - Lab11-6_NVM_Fuse_ReadAux1

Categories:

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

Debugger SNAP

Manage Configurations...

Manage Network Tools...

Options for Snap

Memory to Program

Option categories:

Memories to Program

Manual select memories and range

Auto select memories and ranges

Manually select memories and ranges

Configuration Memory

ID

Program Memory

Program Memory Range(s)(hex)

0-3ffff

Check ID

Preserve Program Memory

Preserve Program Memory Range(s)(hex)

Option Description

OK

Cancel

Apply

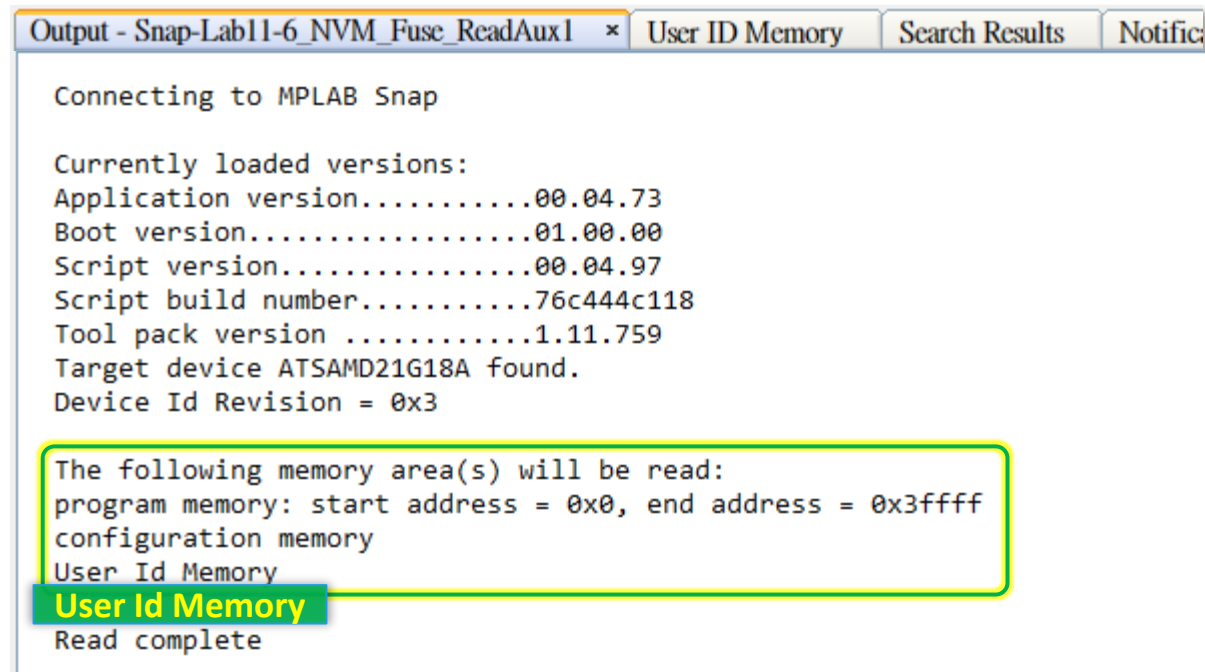
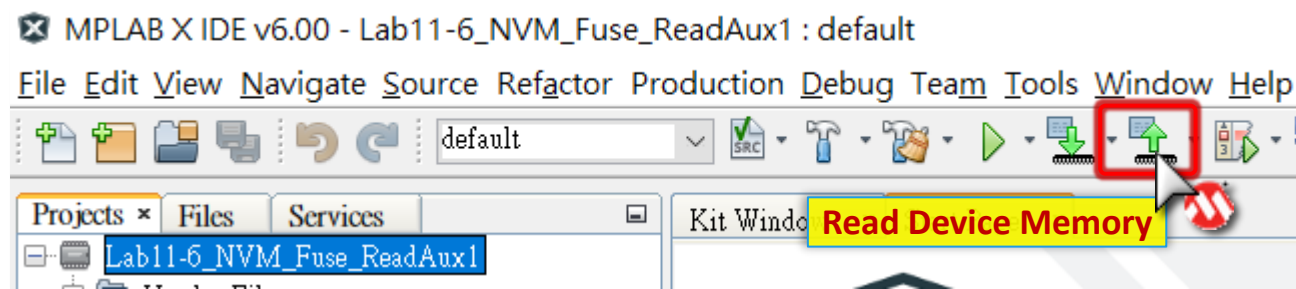
Unlock

Help

User ID Memory in NVM AUX0 (User Row)



Read User ID Memory in MPLAB X IDE



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

Updated after Read

Memory User ID Memory v

User ID Memory in NVM AUX0 (User Row)



Modify and write User ID Memory in MPLAB X IDE

User ID Memory × Output - Snap-Lab11-6_NVM_Fuse_ReadAux1 Search Results Notifications

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

Memory User ID Memory Format Standard

Try Click to Modify

(Lab11-6_NVM_Fuse_ReadAux1) - Memory access restricted

[Debug Build Required For Editing.](#)

Debug Build Required For Editing.

25

User ID Memory in NVM AUX0 (User Row)



Modify and write User ID Memory in MPLAB X IDE

The screenshot shows the MPLAB X IDE interface. The 'User ID Memory' window is open, displaying a table of memory addresses and User IDs. The address 0080_4008 is highlighted, and the User ID 0x12345678 is entered. A red box highlights the 'Build for Debugging' button in the toolbar. A green box highlights the 'Click to Modify' button. A yellow box highlights the 'Build for Debugging' button in the context menu.

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

Memory: User ID Memory Format: Standard

User ID Memory in NVM AUX0 (User Row)



Modify and write User ID Memory in MPLAB X IDE

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

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

Memory: User ID Memory Format: Standard

Program Device for Debugging (Project Lab11-6_NVM_Fuse_ReadAux1)

Program Device for Production (Project Lab11-6_NVM_Fuse_ReadAux1)

Erase Device Memory (Project Lab11-6_NVM_Fuse_ReadAux1)

Programmer To Go PICkit3/PICkit4 (Project Lab11-6_NVM_Fuse_ReadAux1)

User ID Memory in NVM AUX0 (User Row)

NVMCTRL User Row Interface function

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

bool NVMCTRL_USER_ROW_RowErase(uint32_t address)

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

bool NVMCTRL_USER_ROW_PageWrite(uint32_t *data, const uint32_t address)

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

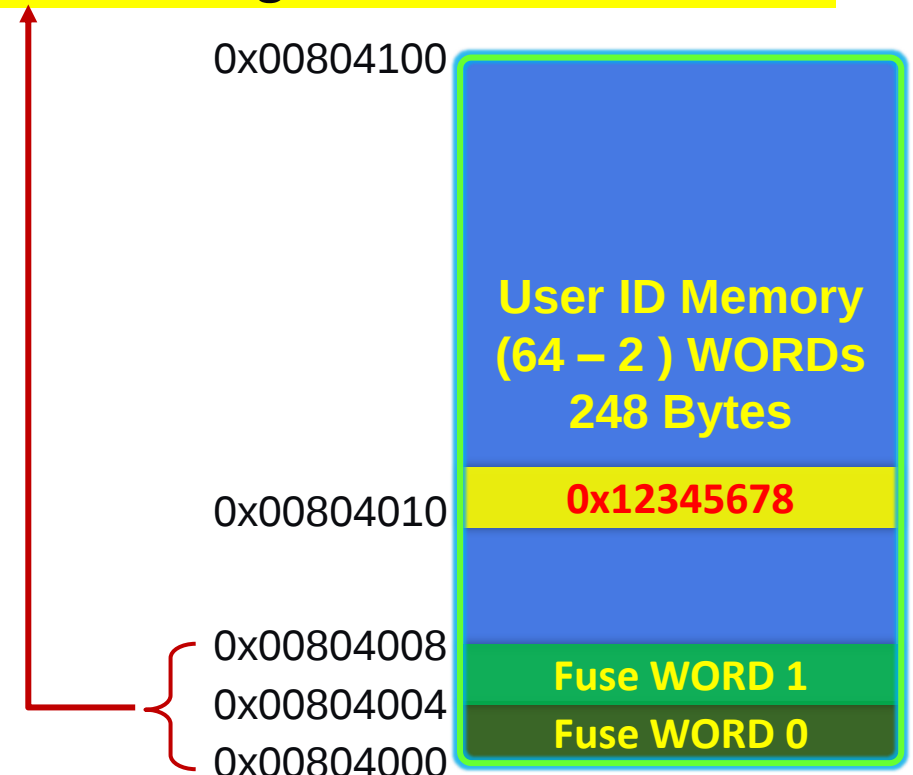
void NVMCTRL_SecurityBitSet(void)

Set the Security Bit by software.

⚙️ Lab11-7 Write NVM AUX0 (User Row)

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

Let's go !



⚙️ Lab11-7 Write NVM AUX0 (User Row)

Step1

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

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

✂ Lab11-7 Write NVM AUX0 (User Row)

Step2

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

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

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

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

⚙️ Lab11-7 Write NVM AUX0 (User Row)

Step3

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

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

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

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

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

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

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

✖ Lab11-7 Write NVM AUX0 (User Row)

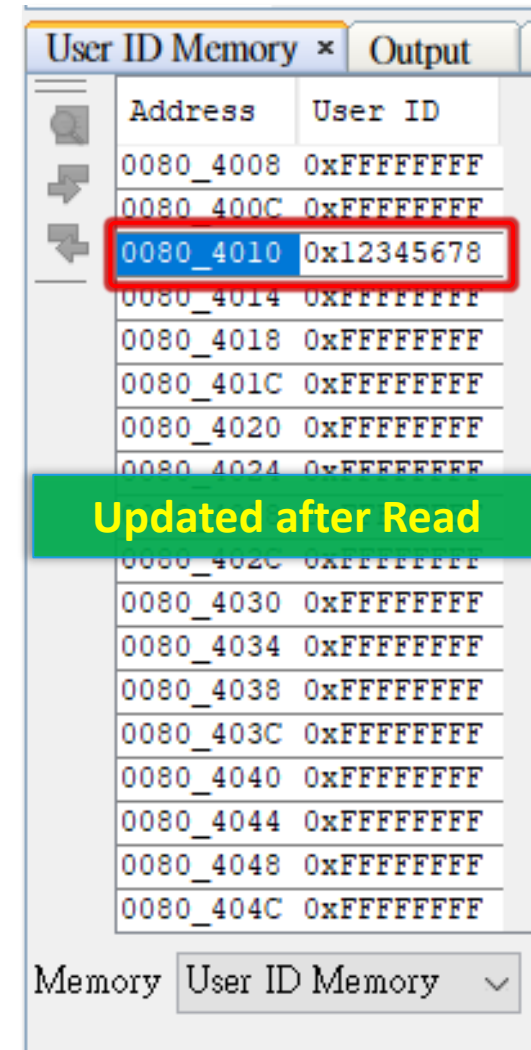
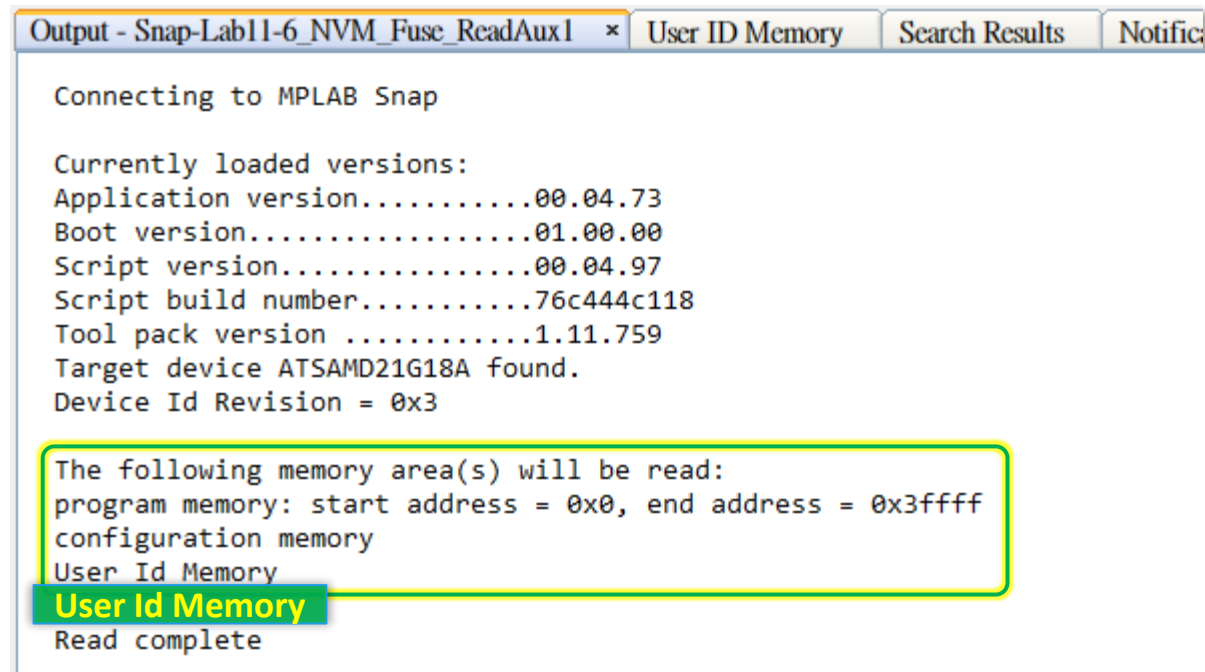
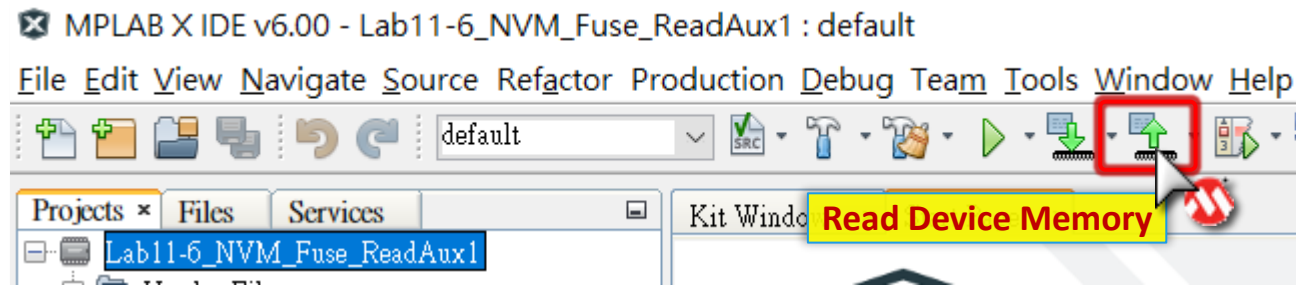
Result

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

⚙️ Lab11-7 Write NVM AUX0 (User Row)



Read User ID Memory in MPLAB X IDE



⚙️ Lab11-7 Write NVM AUX0 (User Row)

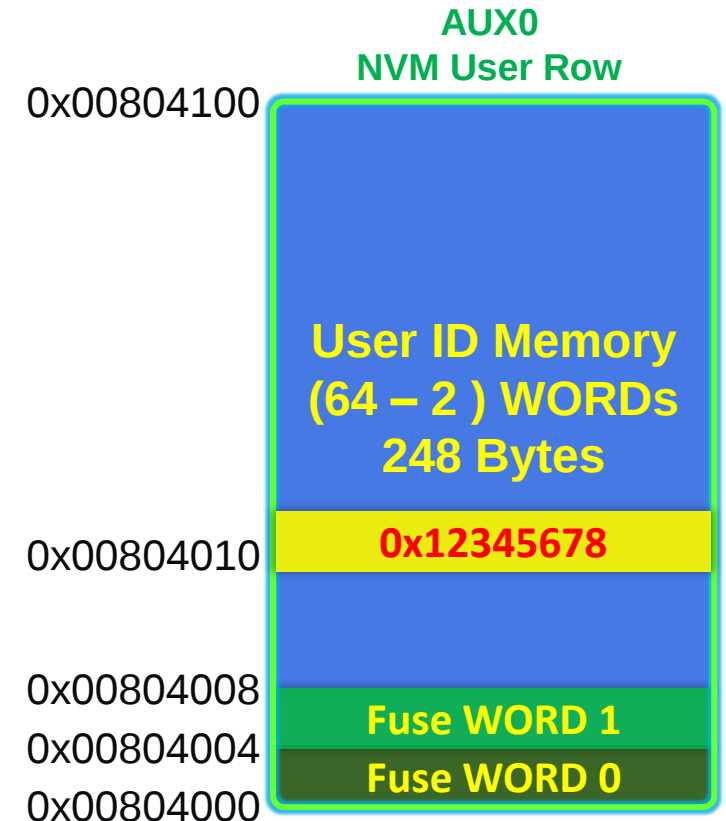
Read User ID Memory in Microchip Studio



Document.userpage

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

Microchip Studio IDE to store Fuse and User ID Memory together



NVM Security Bit

- **Security Bit**

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

- **Chip Erase**

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

- **Programming**

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

- **NVM Auxiliary space**

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

- **Debug Operation**

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

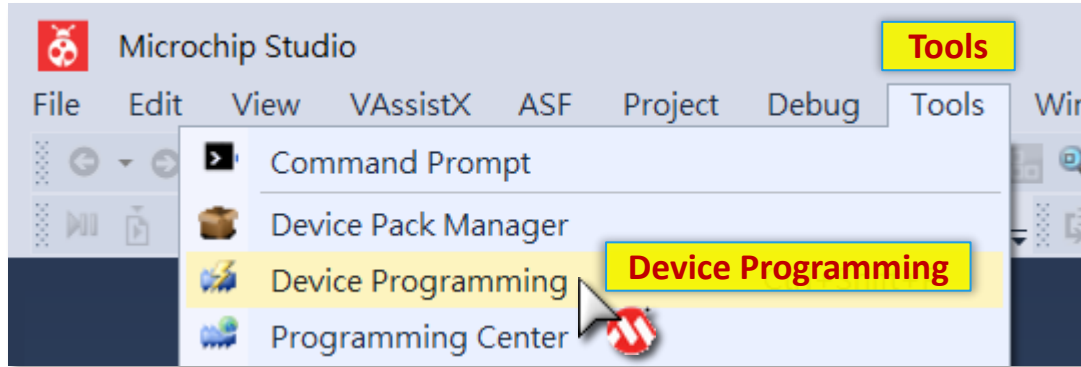
Lab11-8 Code Protect by NVM Security Bit

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

Let's go !

⚙️ Lab11-8 Code Protect by NVM Security Bit

Use Microchip Studio to enable Security Bit



Open Tools

► Device Programming

Tool : MPLAB Snap

Device : ATSAM21G18A

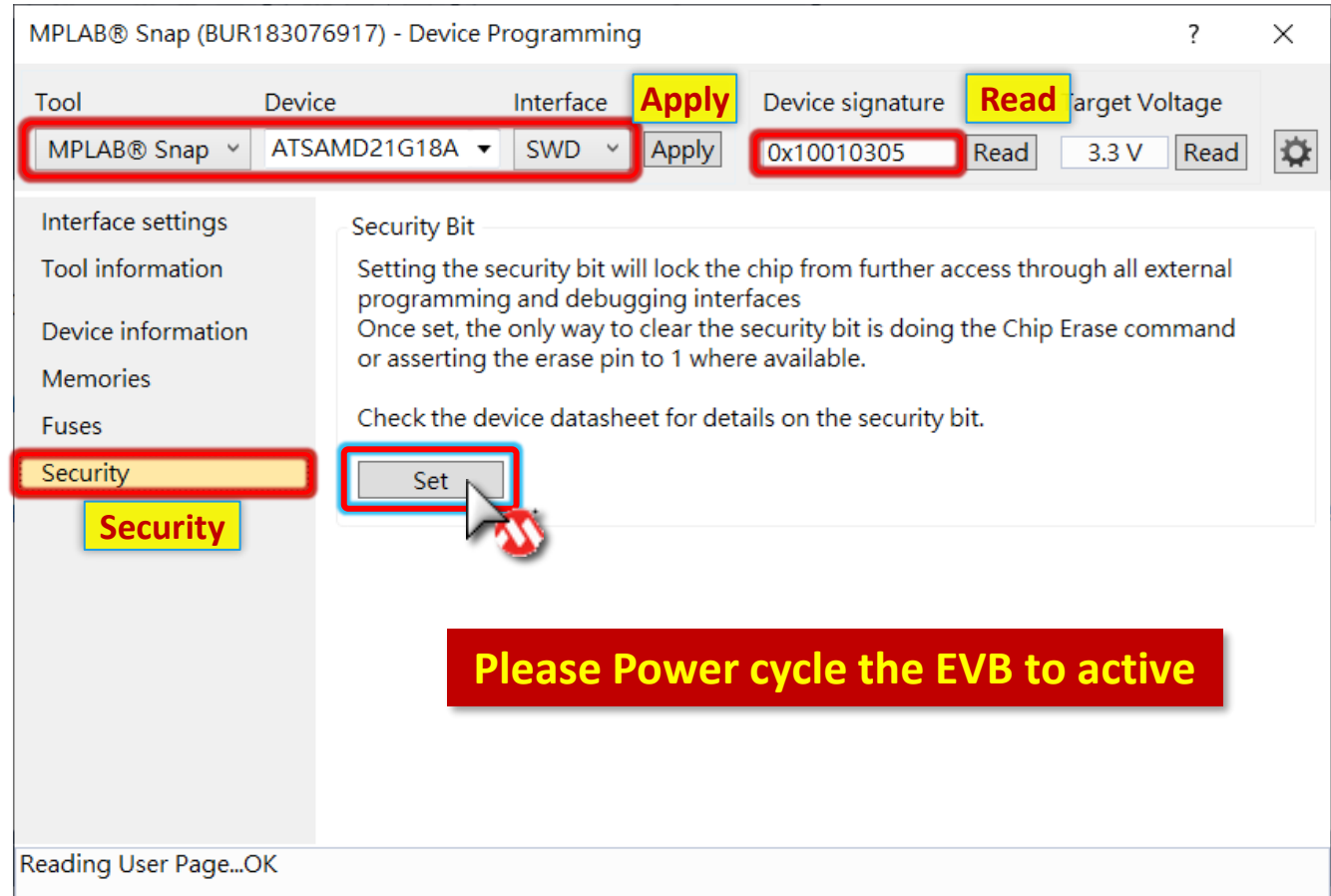
Interface : SWD

► [Apply]

► [Read]

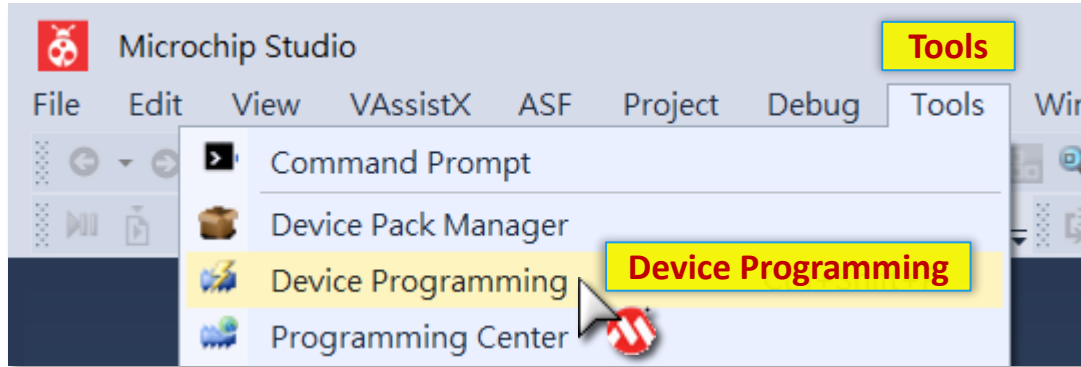
Device signature

► [Security]



⚙️ Lab11-8 Code Protect by NVM Security Bit

Use Microchip Studio to Read Device signature after enable Security Bit



Power cycle the EVB

► **Tools**

► **Device Programming**

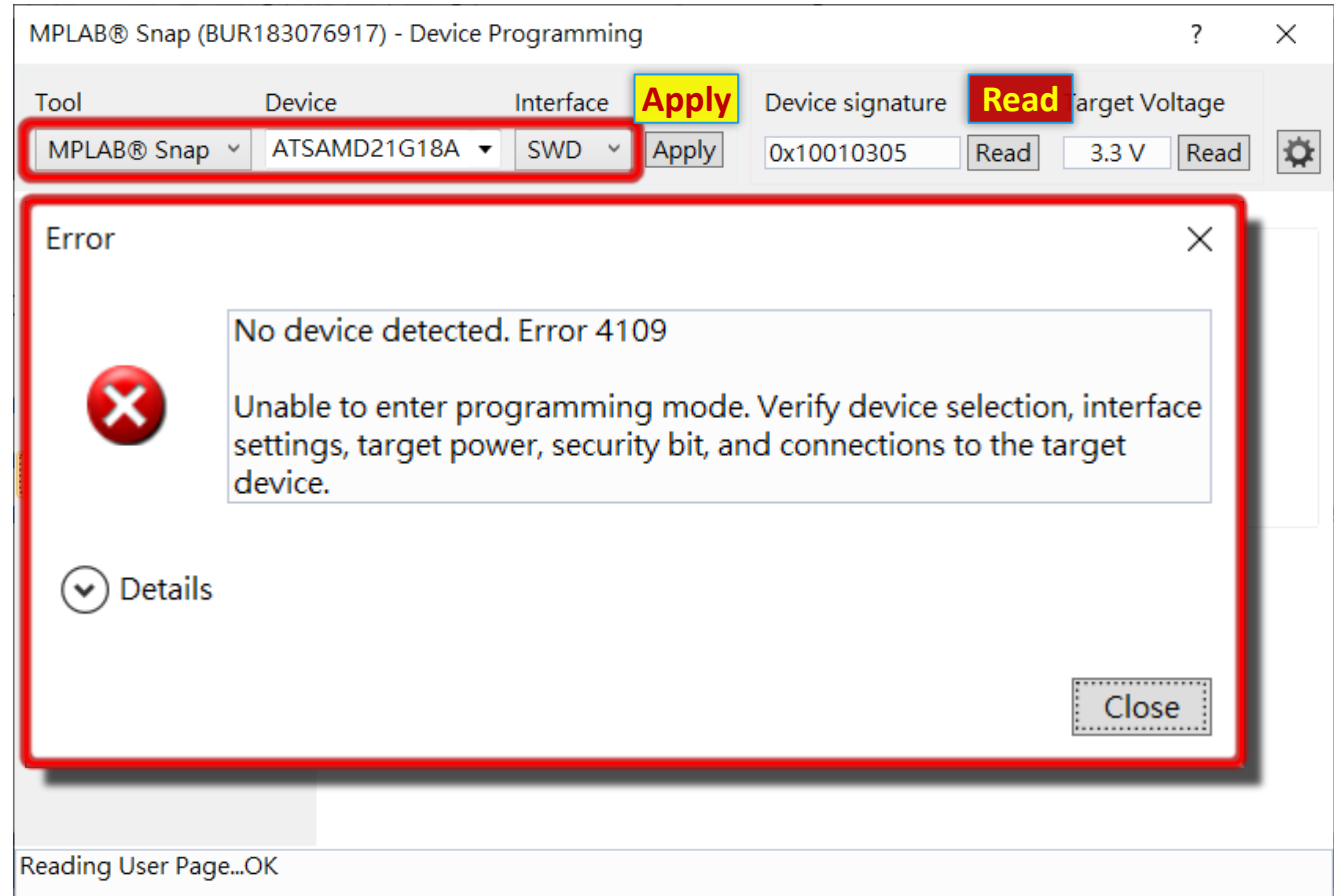
Tool : MPLAB Snap

Device : ATSAM21G18A

Interface : SWD

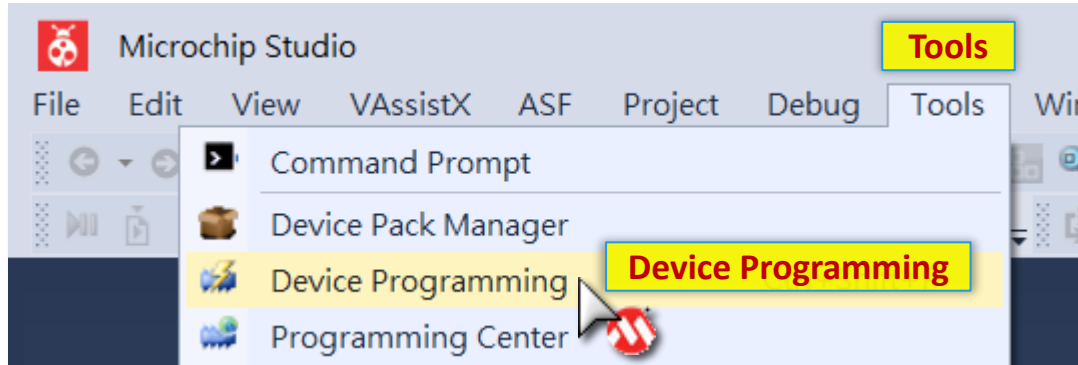
► **[Apply]**

► **[Read]**



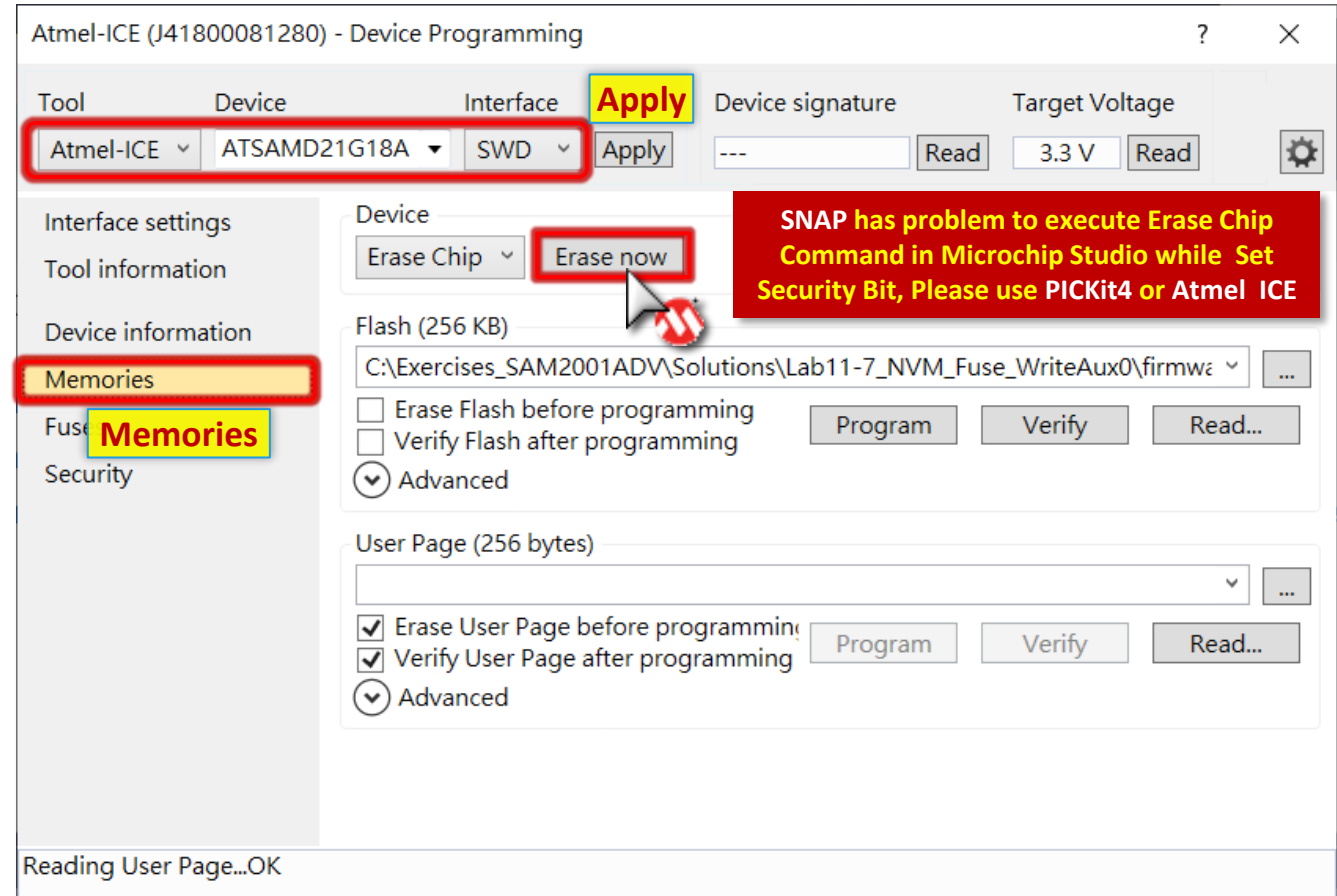
⚙️ Lab11-8 Code Protect by NVM Security Bit

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



Power cycle the EVB

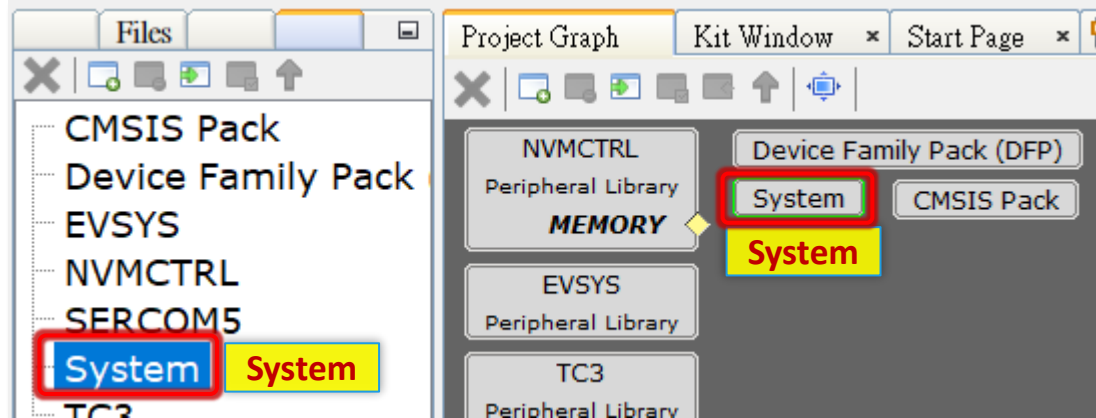
- ▶ **Tools**
- ▶ **Device Programming**
 - Tool : **PICKit4 or Atmel ICE**
 - Device : **ATSAMD21G18A**
 - Interface : **SWD**
- ▶ **[Apply]**
- ▶ **[Read]**
- ▶ **[Memory]**



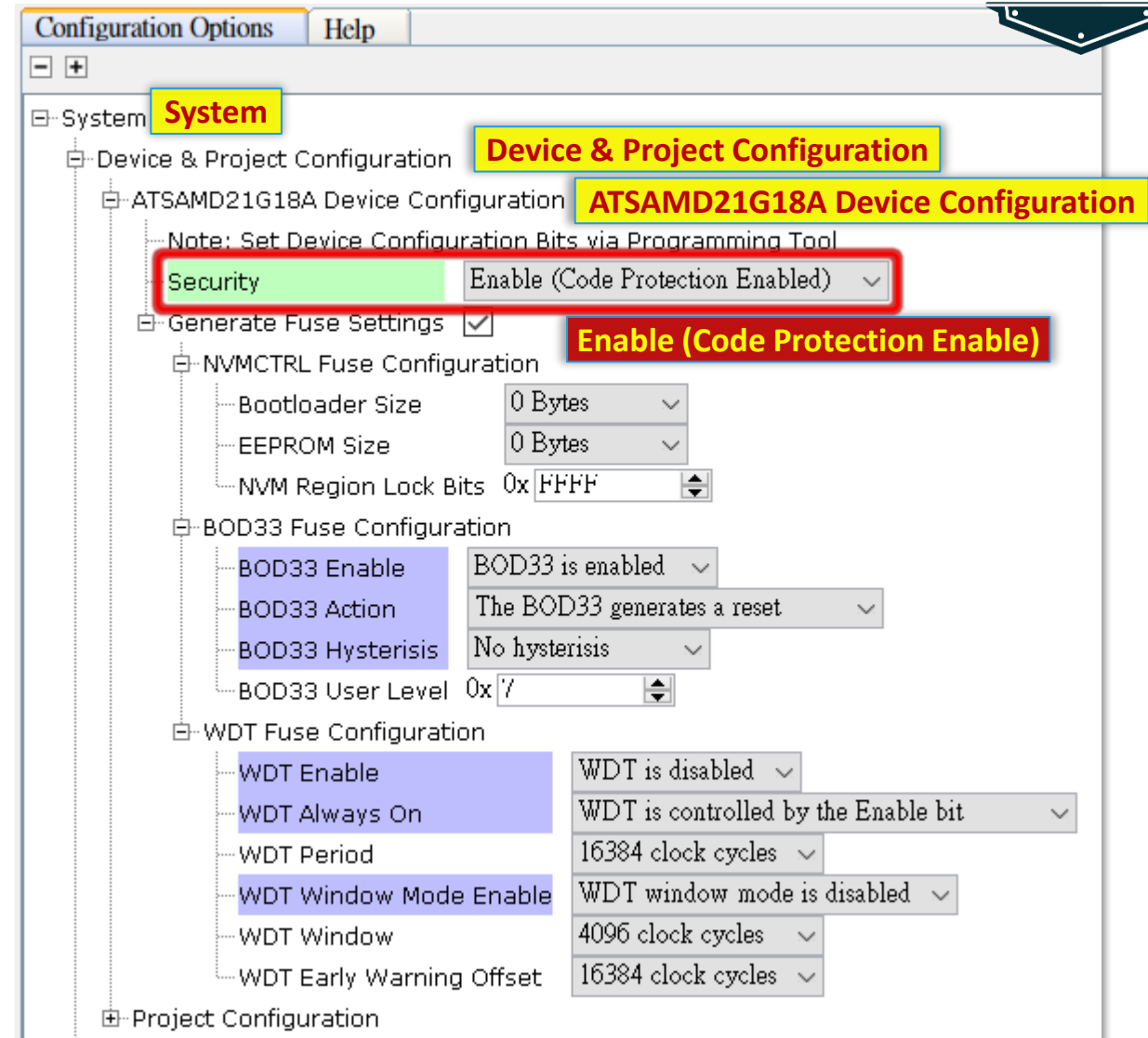
⚙️ Lab11-8 Code Protect by NVM Security Bit



Enable Code Protect in MPLAB Harmony



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

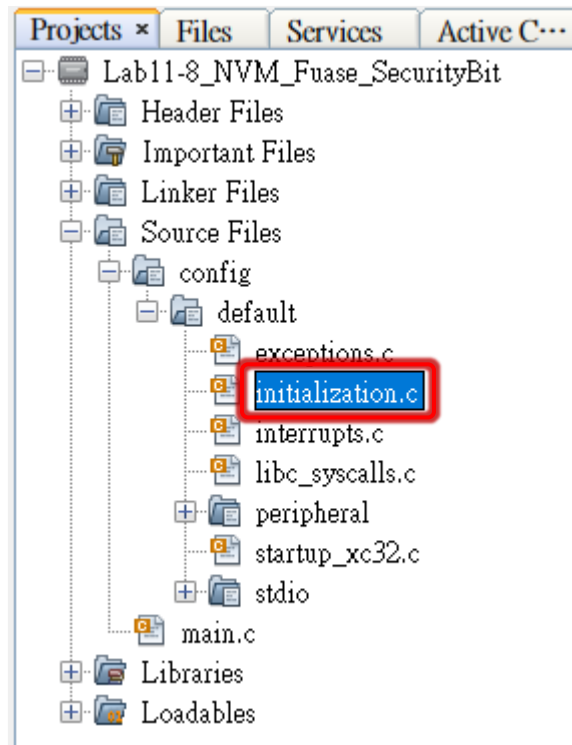


⚙️ Lab11-8 Code Protect by NVM Security Bit



Enable Code Protect in MPLAB Harmony

- Security Bit will be configured in initialization.c



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

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

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

Security Bit

✖ Lab11-8 Code Protect by NVM Security Bit

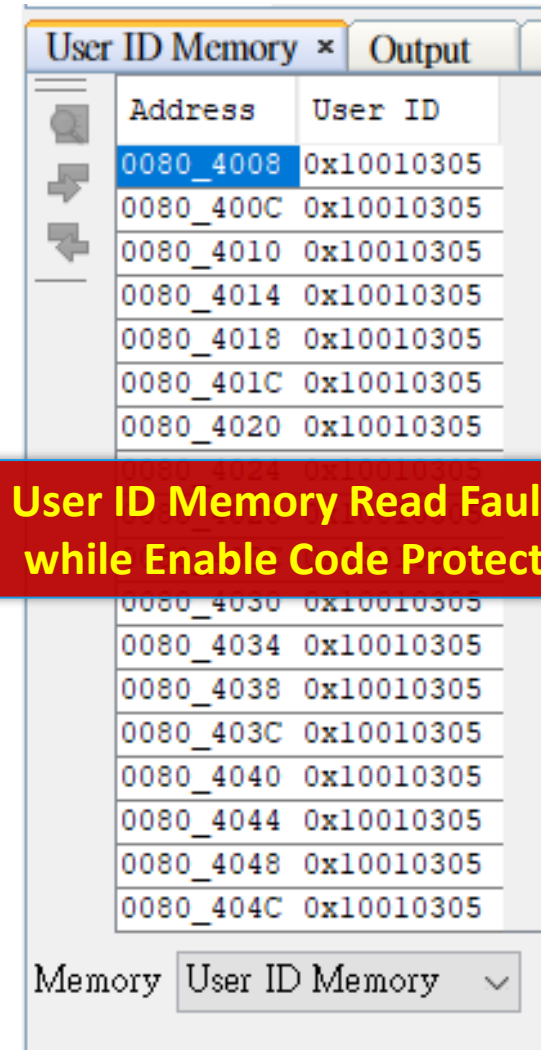
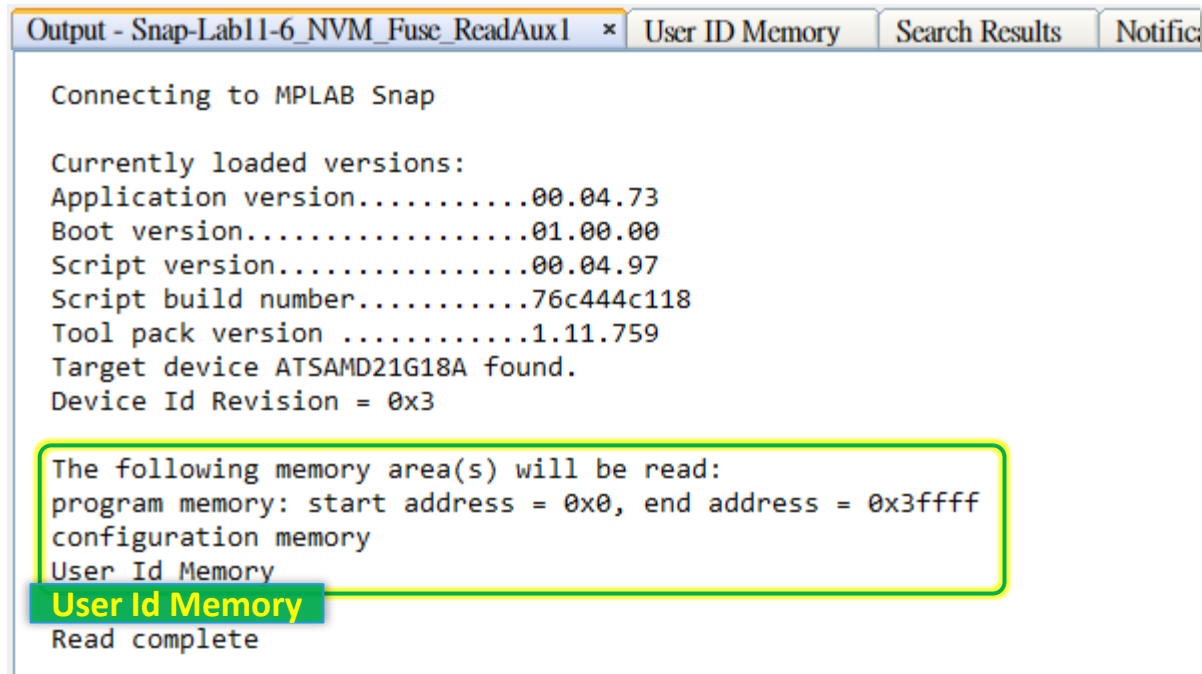
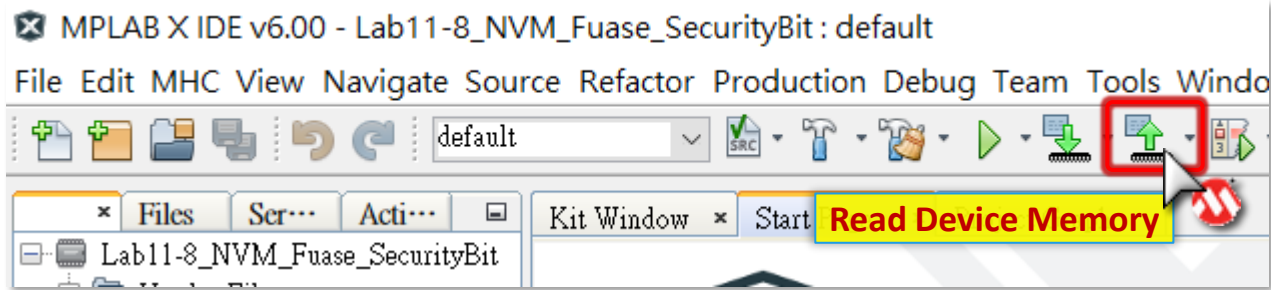
Result

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

⚙️ Lab11-8 Code Protect by NVM Security Bit



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



⚙️ Lab11-8 Code Protect by NVM Security Bit



Read Execution Memory in MPLAB X IDE after execute Erase Chip

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

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

default

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

Lab11-8_NVM_Fuase_SecurityBit

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

Erase Device Memory

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

Connecting to MPLAB Snap

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

Erasing...
Erase successful

Erase to un-seal Security Bit

Execution Memory x Output Search Results Notifications Console New

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

Execution Memory Read Successful after Erase Chip

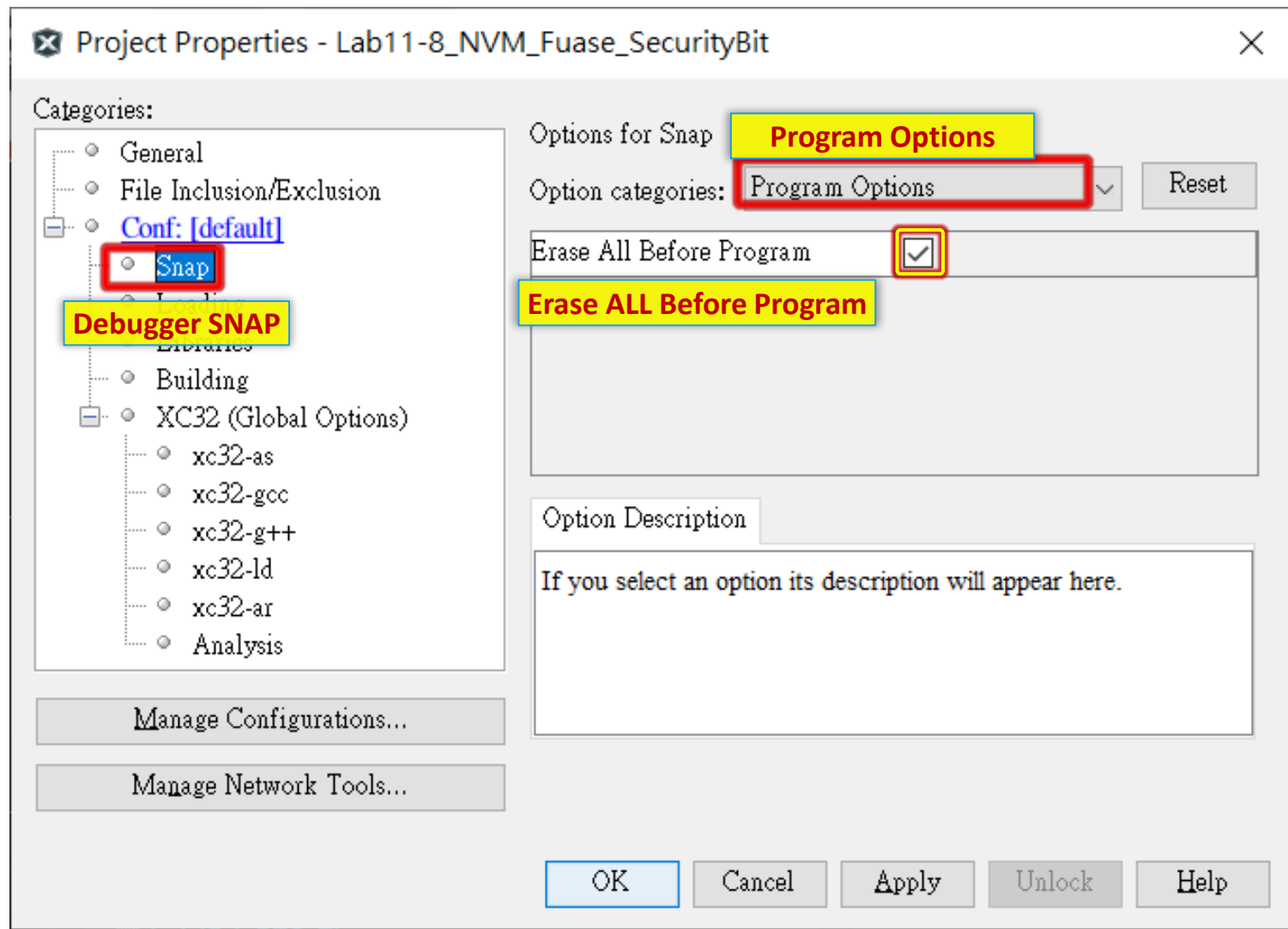
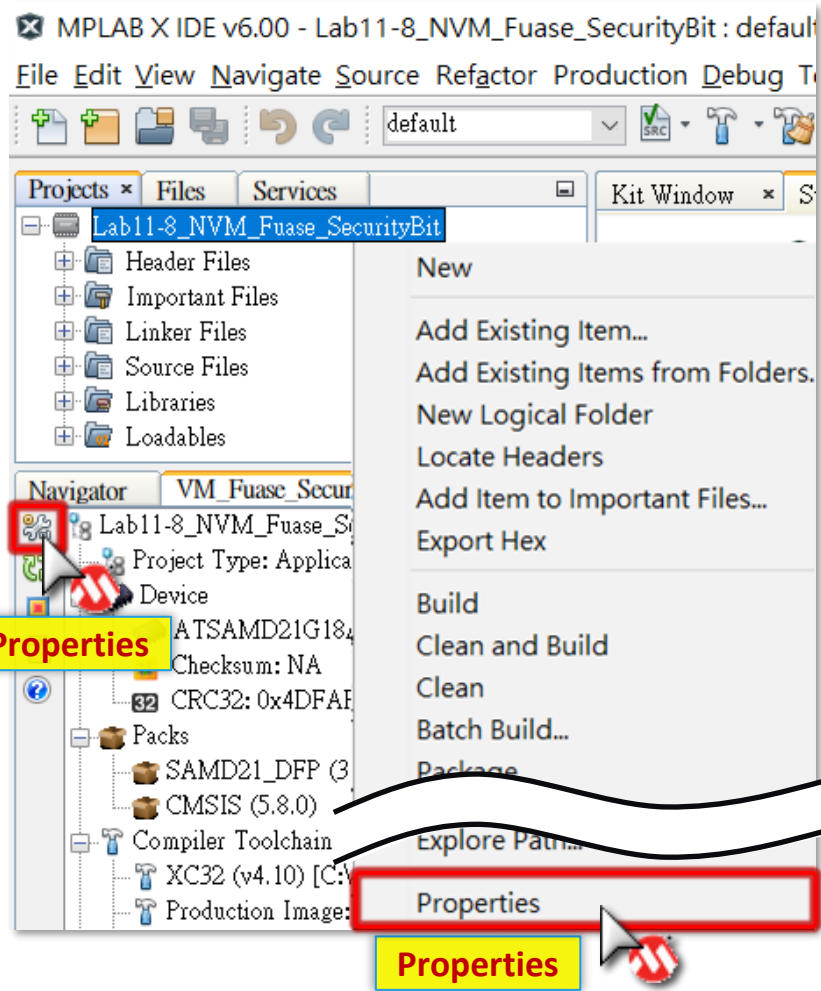
Memory Execution Mem... Format Data

Execution Memory

⚙️ Lab11-8 Code Protect by NVM Security Bit



Harmony New project will enable Erase All Before Program option



⚙️ Lab11-8 Code Protect by NVM Security Bit

Step1 : Implement a software Security Bit configure function

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

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

    if( NVMCTRL_GetSecurityBitStatus() )
        return;

    NVMCTRL_SecurityBitSet();

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

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

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

Lab11-8 Code Protect by NVM Security Bit

Step2 : Implement Self NVM Flash Read and Write function

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

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

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

    while( NVMCTRL_IsBusy() ) {}

    NVMCTRL_Read( ReadWord, 16, SELF_RW_ADDRESS );

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

⚙️ Lab11-8 Code Protect by NVM Security Bit

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

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

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

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

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

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

Lab11-8 Code Protect by NVM Security Bit

Step4 : Execute Self NVM Flash Read and Write function

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

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

```
#define SELF_RW_ADDRESS 0x3000  
void NVM_Self_ReadWrite() {...}
```

```
int main ( void )  
{  
    ...  
    NVM_Read_UQSN( NVM_UQSN_Word );  
    NVM_Read_AUX0( NVM_AUX0_Word );  
    NVM_Read_AUX1( NVM_AUX1_Word );  
    NVM_AUX0_Word[4] = 0x12345678;  
    NVM_Write_AUX0( NVM_AUX0_Word );  
    NVM_Read_AUX0( NVM_AUX0_Word );
```

```
// TODO 11-8.02  
NVM_Code_Protect();  
NVM_Self_ReadWrite();
```

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

Lab11-8 Code Protect by NVM Security Bit

Result

```
COM3 - Tera Term VT
File Edit Setup Control Window Help
1st boot after program

Erase NVM-AUX0 User Row :
- success
Write NVM-AUX0 User Row :
- Page 0 : success
- Page 1 : success
- Page 2 : success
- Page 3 : success
done

NVM-AUX0 User Row : (64 Words)
- Word 0 and Word 1
0x00804000 : 0xD800C777
0x00804004 : 0xFFFF005C
- User ID Memory
0x00804008 : 0xFFFFFFFF 0xFFFFFFFF 0x12345678 0xFFFFFFFF
0x00804018 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804028 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804038 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804048 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804058 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804068 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804078 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804088 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804098 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040A8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040B8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040C8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040D8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040E8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040F8 : 0xFFFFFFFF 0xFFFFFFFF
- done

Security Bit set successfully, please cycle power to active!

Self Read (4 Words)
0x00003000 : 0xFFFFFFFF
0x00003004 : 0xFFFFFFFF
0x00003008 : 0xFFFFFFFF
0x0000300C : 0xFFFFFFFF
Erase Row at address 0x00003000
Write Page at address 0x00003000
done
```

```
COM3 - Tera Term VT
File Edit Setup Control Window Help
2nd boot after power cycle

Security Bit has been set!
NVM-AUX0 User Row : (64 Words)
- Word 0 and Word 1
0x00804000 : 0xD800C777
0x00804004 : 0xFFFF005C
- User ID Memory
0x00804008 : 0xFFFFFFFF 0xFFFFFFFF 0x12345678 0xFFFFFFFF
0x00804018 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804028 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804038 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804048 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804058 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804068 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804078 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804088 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x00804098 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040A8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040B8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040C8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040D8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040E8 : 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
0x008040F8 : 0xFFFFFFFF 0xFFFFFFFF
- done

Self Read (4 Words)
0x00003000 : 0xEEEEEEEE
0x00003004 : 0xDDDDDDDD
0x00003008 : 0CCCCCCCC
0x0000300C : 0BBBBBBBB
Self Flash Write successfully
While Security Bit enabled

Erase Row at address 0x00003000
Write Page at address 0x00003000
done
```

Microchip Trademarks

Overview

Microchip Technology Incorporated's products are marketed and sold under one or more of the following trademarks belonging to Microchip or one of Microchip's subsidiaries. Questions regarding use of these trademarks should be addressed to the Microchip's Legal Department via email: legal.department@microchip.com.

The following are registered trademarks of Microchip in the U.S.A. and other countries:

The Microchip name and logo, the Microchip logo, Adaptec, AnyRate, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Klear, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PackeTime, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA

The following are registered trademarks of Microchip in the U.S.A:

AgileSwitch, APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, Flashtec, Hyper Speed Control, HyperLight Load, IntelliMOS, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, TrueTime, WinPath, and ZL

The following are registered trademarks of Microchip in other countries: BeaconThings, GestIC, Silicon Storage Technology

The following are trademarks of Microchip in the U.S.A. and other countries:

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, Augmented Switching, BlueSky, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, Espresso T1S, EtherGREEN, IdealBridge, In-Circuit Serial Programming, ICSP, INICnet, Intelligent Paralleling, Inter-Chip Connectivity, JitterBlocker, maxCrypto, maxView, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICKit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, RTAX, RTG4, SAM-ICE, Serial Quad I/O, simpleMAP, SimpliPHY, SmartBuffer, SMART-I.S., storClad, SQL, SuperSwitcher, SuperSwitcher II, Switchtec, SynchroPHY, Total Endurance, TSHARC, USBCheck, VariSense, VectorBlox, VeriPHY, ViewSpan, WiperLock, XpressConnect, and ZENA

The following is a service mark of Microchip in the U.S.A.: SQTP

The following are logos of Microchip in the U.S.A. and other countries: The Adaptec logo, Frequency on Demand, Silicon Storage Technology, Symmcom, and Trusted Time

The following is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries: GestIC

