MICROCHIP

Regional Training Centers

SAM1001 v1.00

Introduction to ATSAMD21 M0+ Microcontroller

Authors: Libra Chien
Adam Syu
Microchip Technology Inc.

Class Objectives

- **When you finish this class you will be able to:**
  - Be familiar with the Studio 7 & SAMD21.
  - Understand how to configure and use the ATSAMD21G18A peripherals.

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Agenda

- ATSAMD21 M0+ Introduction
- Development Tools Introduction
- Getting Started First Project
- PORT I/O Architecture
- TC Architecture
- NVIC Architecture
- Clock System Architecture
- PINMUX & SERCOM - UART Architecture
- High Speed ADC Architecture & Event System
- TCC – PWM Architecture

© 2008 Microchip Technology Incorporated. All Rights Reserved.

ATSAMD21 M0+ Introduction

© 2008 Microchip Technology Incorporated. All Rights Reserved.

ARM’s history

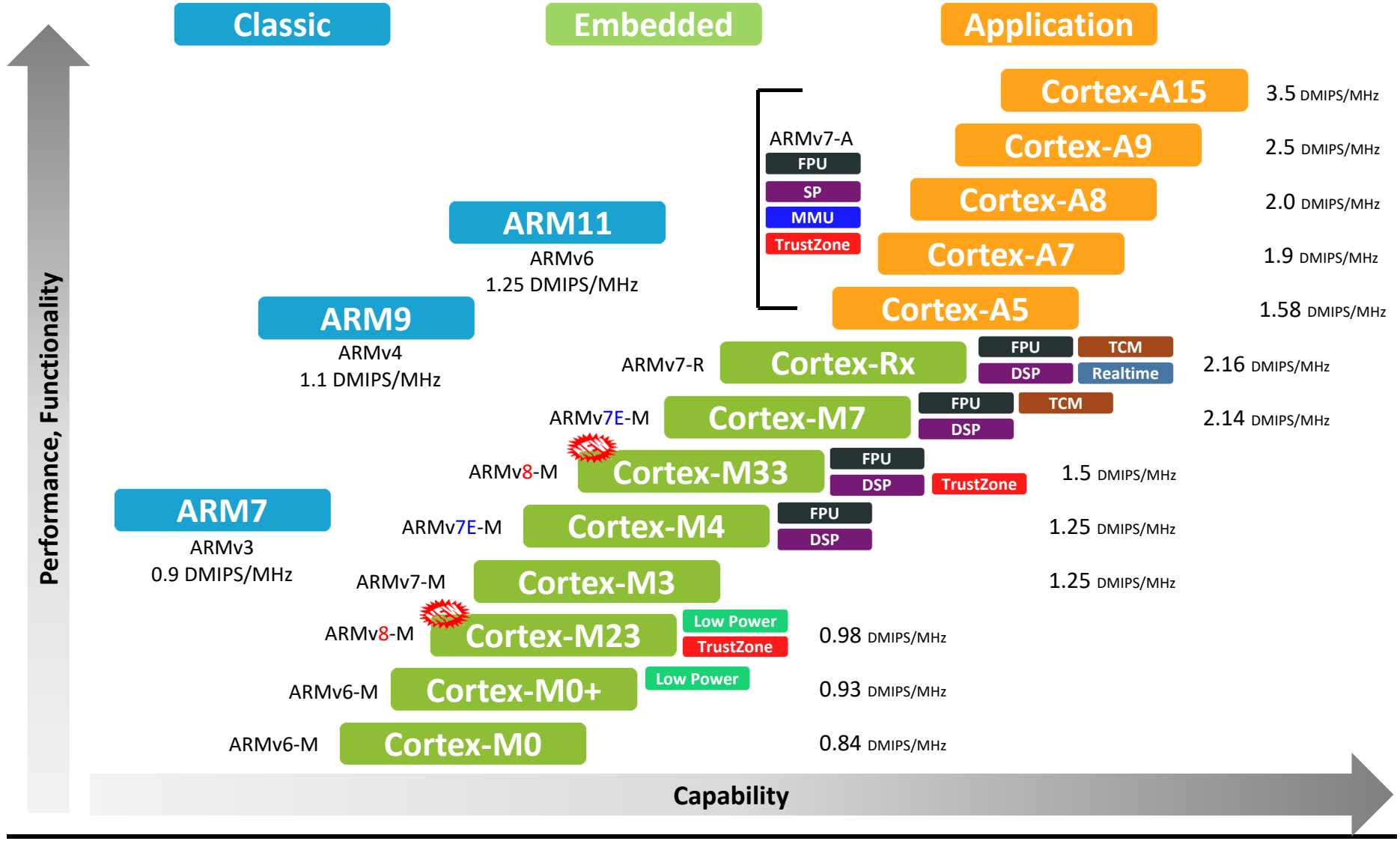
- ARM (Advanced RISC Machine) 安謀, 進階精簡指令集機器。是一個32位元精簡指令集（RISC）處理器架構, ARM處理器非常適用於嵌入式系統以及行動通訊領域,主要設計目標為低成本、高效能、低耗電的特性。至2009年為止, ARM架構處理器佔市面上所有32位元嵌入式RISC處理器90%的比例。
- ARM處理器從可攜式裝置（PDA、行動電話、多媒體播放器、掌上型電玩和計算機）到電腦週邊設備（硬碟、桌上型路由器），甚至在飛彈的彈載電腦等軍用設施中都有他的存在。
- 2016年7月18日, 日本軟銀集團斥資3.3萬億日元（約合311億美元）將設計ARM的公司ARM Holdings收購。

© 2008 Microchip Technology Incorporated. All Rights Reserved.

ARM in Partnership

© 2008 Microchip Technology Incorporated. All Rights Reserved.

ARM core MCU/MPU brief



© 2008 Microchip Technology Incorporated. All Rights Reserved.

ARM core MCU/MPU success

Performance, Functionality

3DS

Raspberry Pi B+

iPhone1

ARM11

PS-Vista

Cortex-A15

Cortex-A9

Cortex-A8

Cortex-A7

Cortex-A5

HMI

Amazon Echo

Smart Phone

ARM9

digital dashboard

Cortex-Rx

Cortex-M7

Medical

Microsoft Surface
Google Chromebook

WINCE PND

ARM7

Cortex-M33

Cortex-M4

Amazon Dash
(IoT Clients)

Feature
phone

iPod

Cortex-M3

Cortex-M23

Fitbit

Cortex-M0+

Cortex-M0

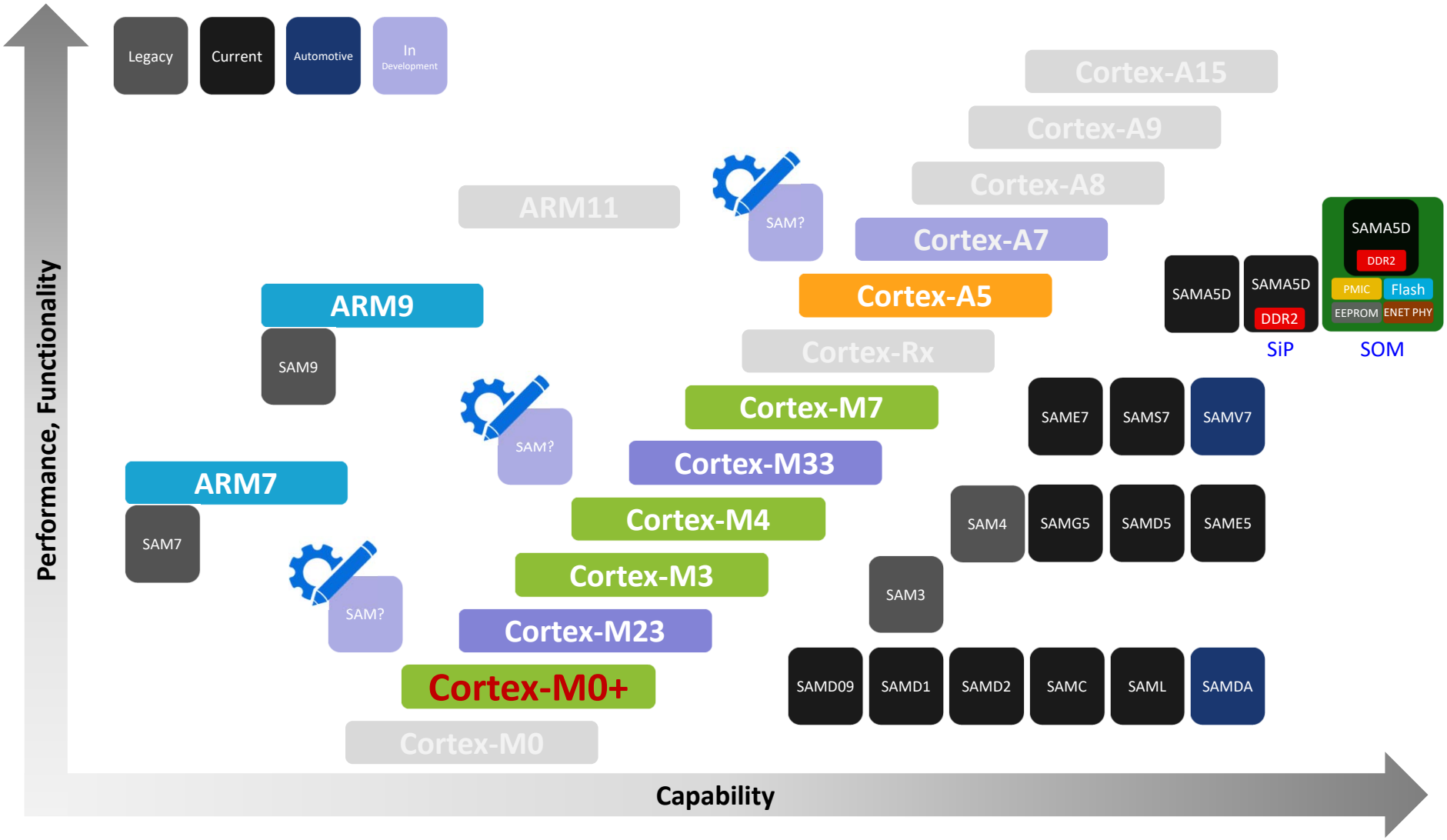
Smart door locker

Mi Band

Arduino M0 Pro

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Microchip ARM Core Products



© 2008 Microchip Technology Incorporated. All Rights Reserved.

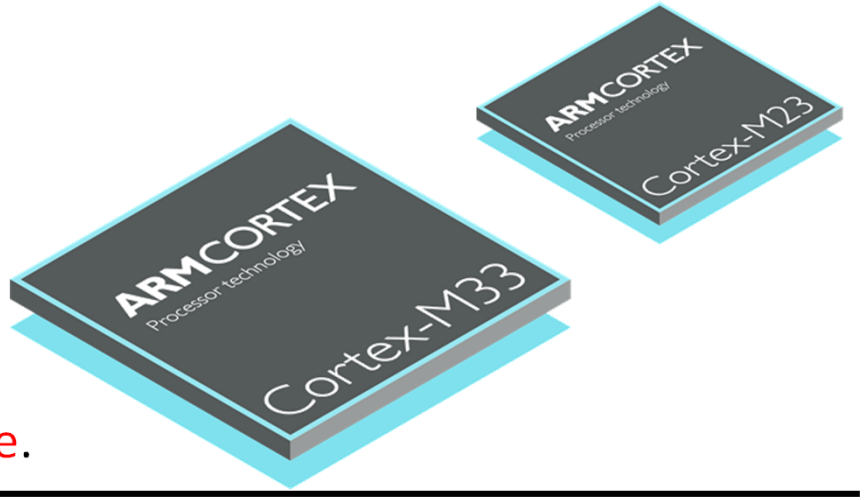
Microchip Cortex-M MCU Series

- **Microchip 32 Bits (ARM core) MCU包括以下系列:**
- **Cortex-M0+**
  - SAMD09/SAMD1x
    - Low pin count, 高性價比.
  - SAMD2x
    - 擁有豐富的外設周邊.
  - SAMC2x
    - 支援5V工作電壓及CAN bus.
  - SAML2x
    - 極低功耗及豐富的外設周邊.
  - SAMDAx
    - 車規認證.
- **Cortex-M3, Cortex-M4 (Legacy)**
  - SAM3x, SAM4x.

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Microchip Cortex-M MCU Series (cont.)

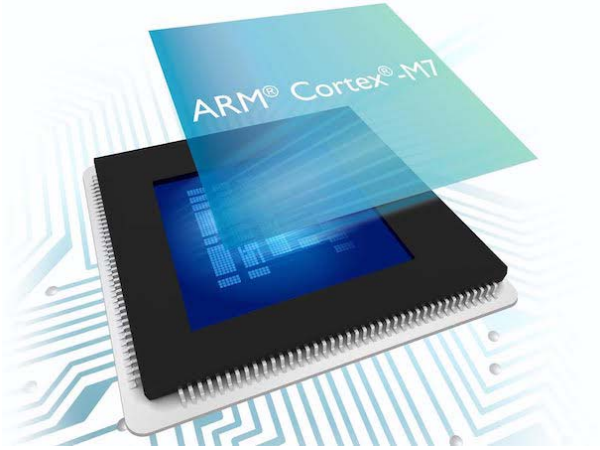
- **Cortex-M4 (New generation)**
  - SAMG5x
    - 極小包裝, 高性價比.
  - SAMD5x
    - 支援浮點運算, 硬體加密, USB ,低功耗.
  - SAME5x
    - 支援浮點運算,硬體加密, USB, Ethernet, CAN-FD,低功耗.
- **Cortex-M23 (Coming soon)**
  - SAML1x
    - 極低功耗, 支援TrustZone.
- **Cortex-M33 (Coming soon)**
  - SAMD6x
    - 高效能低功耗, 支援TrustZone.



© 2008 Microchip Technology Incorporated. All Rights Reserved.

Microchip Cortex-M MCU Series (cont.)

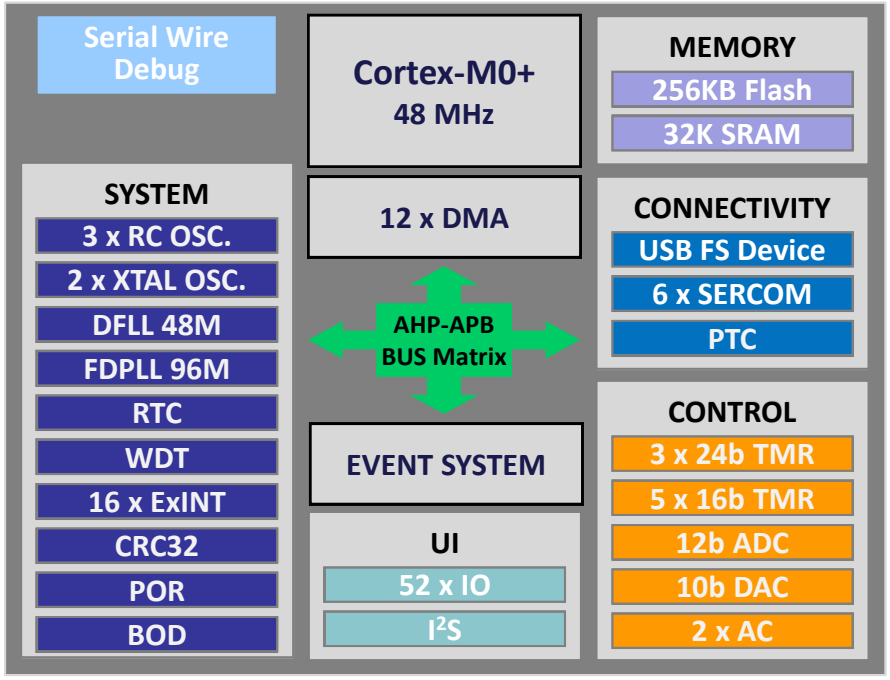
- **Cortex-M7**
  - SAMS7x
    - 高效能, 豐富的外設周邊.
  - SAME7x
    - Ethernet支援, CAN-FD支援.
  - SAMV7x
    - Ethernet支援, CAN-FD支援, 車規支援.



© 2008 Microchip Technology Incorporated. All Rights Reserved.

SAMD21 Block Diagram

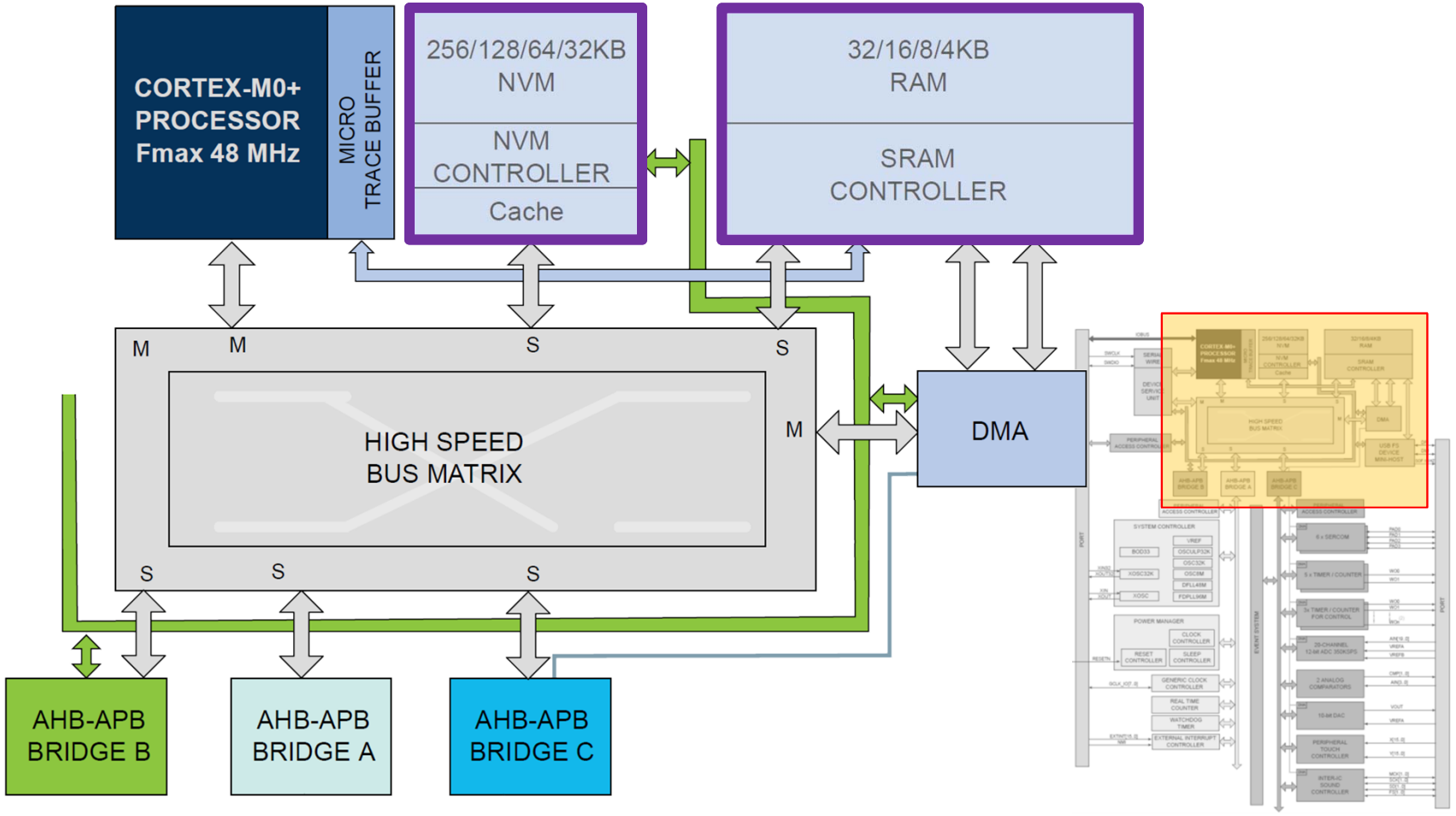
- The SAM D21 is a series of low-power microcontrollers using the 32bits ARM® Cortex® M0+ processor (Cortexv6-M)。
- Maximum frequency of 48MHz and reach 2.46 CoreMark®/ MHz。
- All devices include intelligent and flexible peripherals, Event System for inter-peripheral signaling, and support for capacitive touch button, slider and wheel user interfaces。



© 2008 Microchip Technology Incorporated. All Rights Reserved.

Architecture

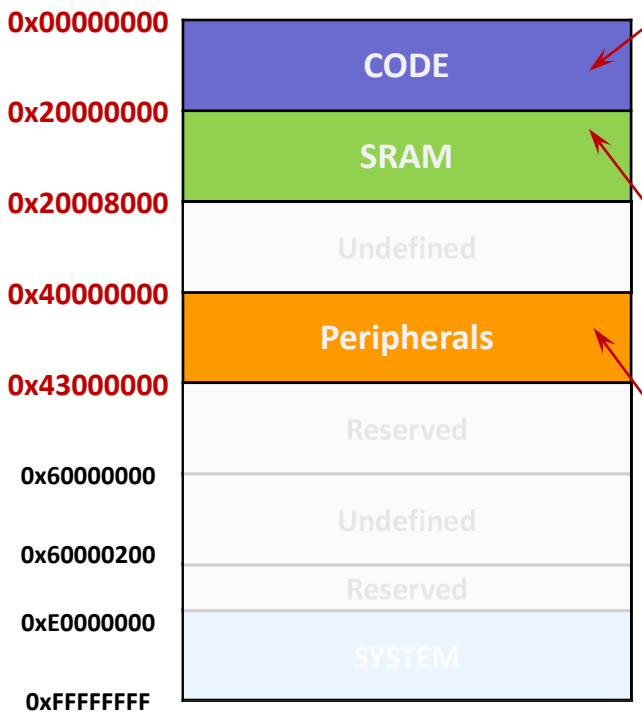
- Harvard, RISC architecture



© 2008 Microchip Technology Incorporated. All Rights Reserved.

SAMD21 Memory

- Up to 256KB Flash and 32KB SRAM



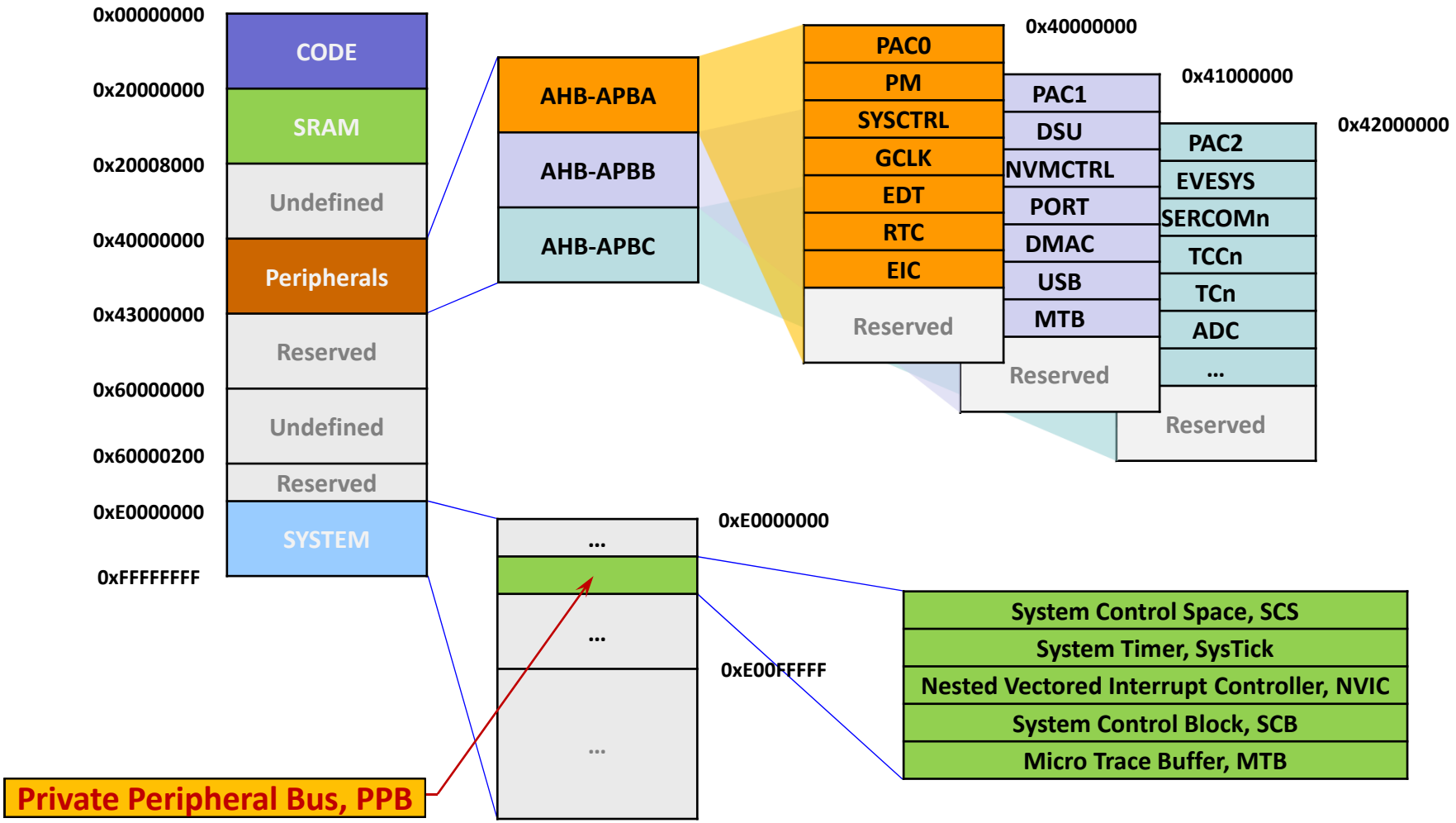
Program Memory
The actual location of the application code.
for example,
main () is placed in this area.
The actual size depend on part number.

Data Memory
This area is used for storing data.
The actual size depend on part number.

Peripherals Function Registers
Reserved for the device
This area include control and status bits for the CPU and peripherals on the device.

© 2008 Microchip Technology Incorporated. All Rights Reserved.

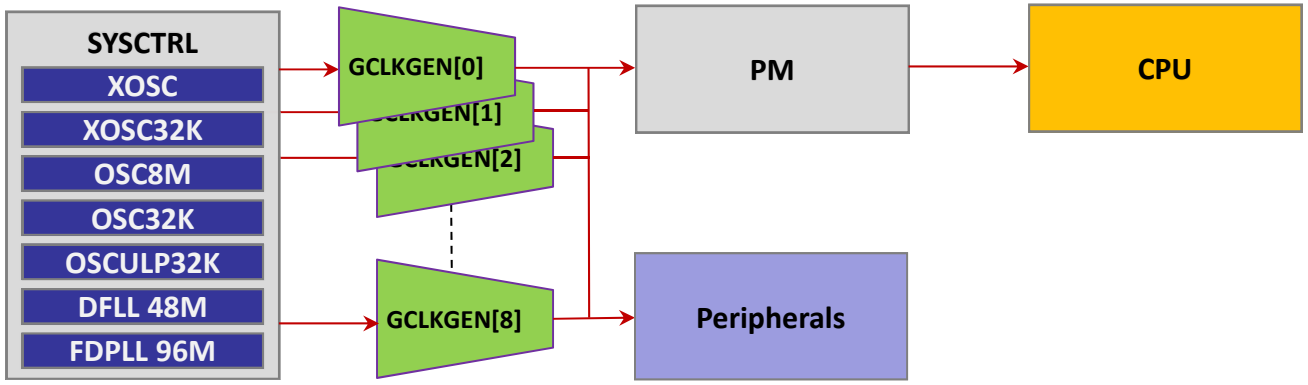
SAMD21 Memory Mapping



© 2008 Microchip Technology Incorporated. All Rights Reserved.

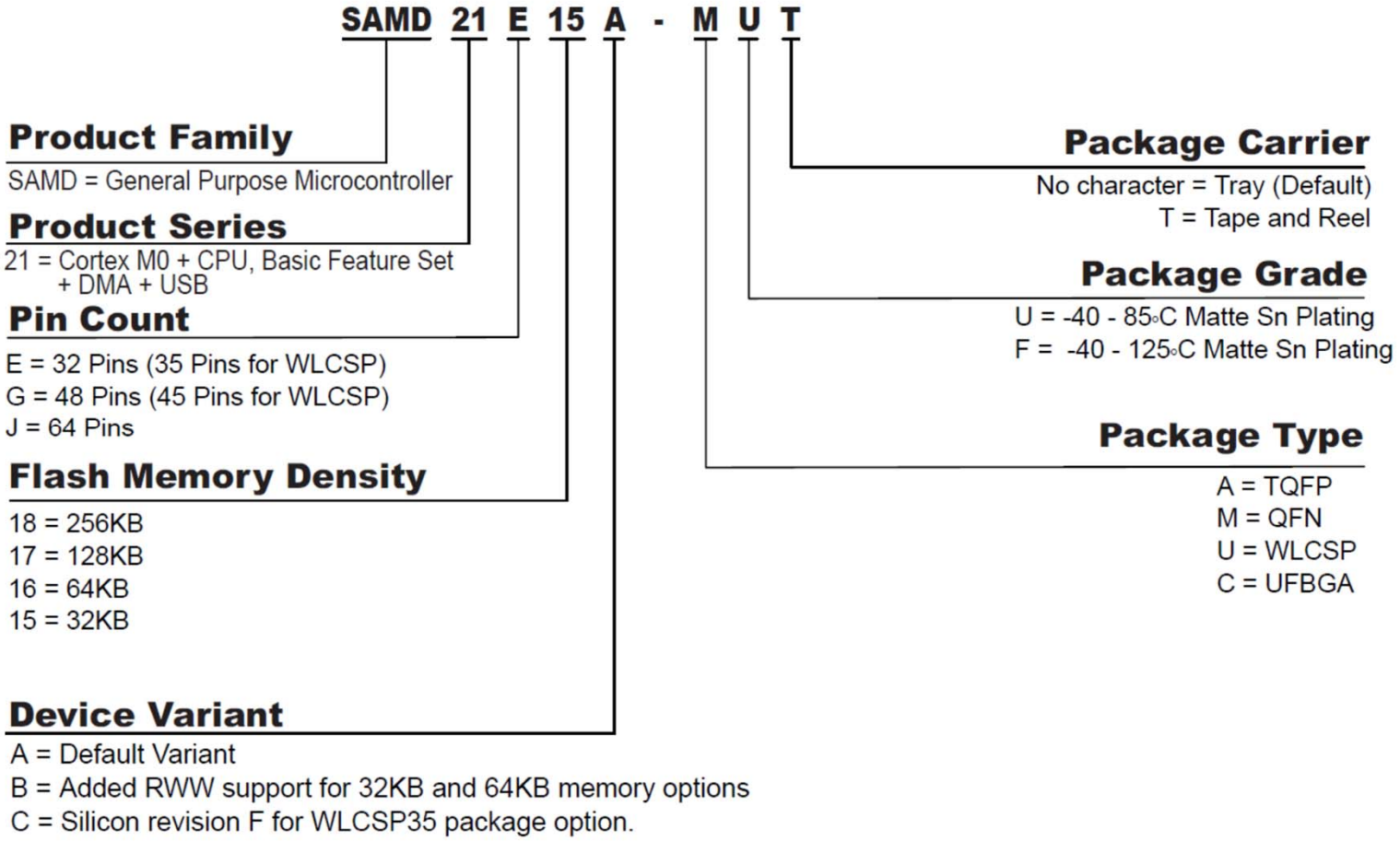
Clock System

- SYSCTRL provides clock sources to the Generic Clock Controller。The available clock sources are XOSC, XOSC32K, OSC32K, OSCULP32K, OSC8M,DFLL48M and FDPLL96M。
- The bus clock can be enabled and disabled in the Power Manager(PM) include CPU Clock (GCLK_MAIN)。



© 2008 Microchip Technology Incorporated. All Rights Reserved.

Part Number



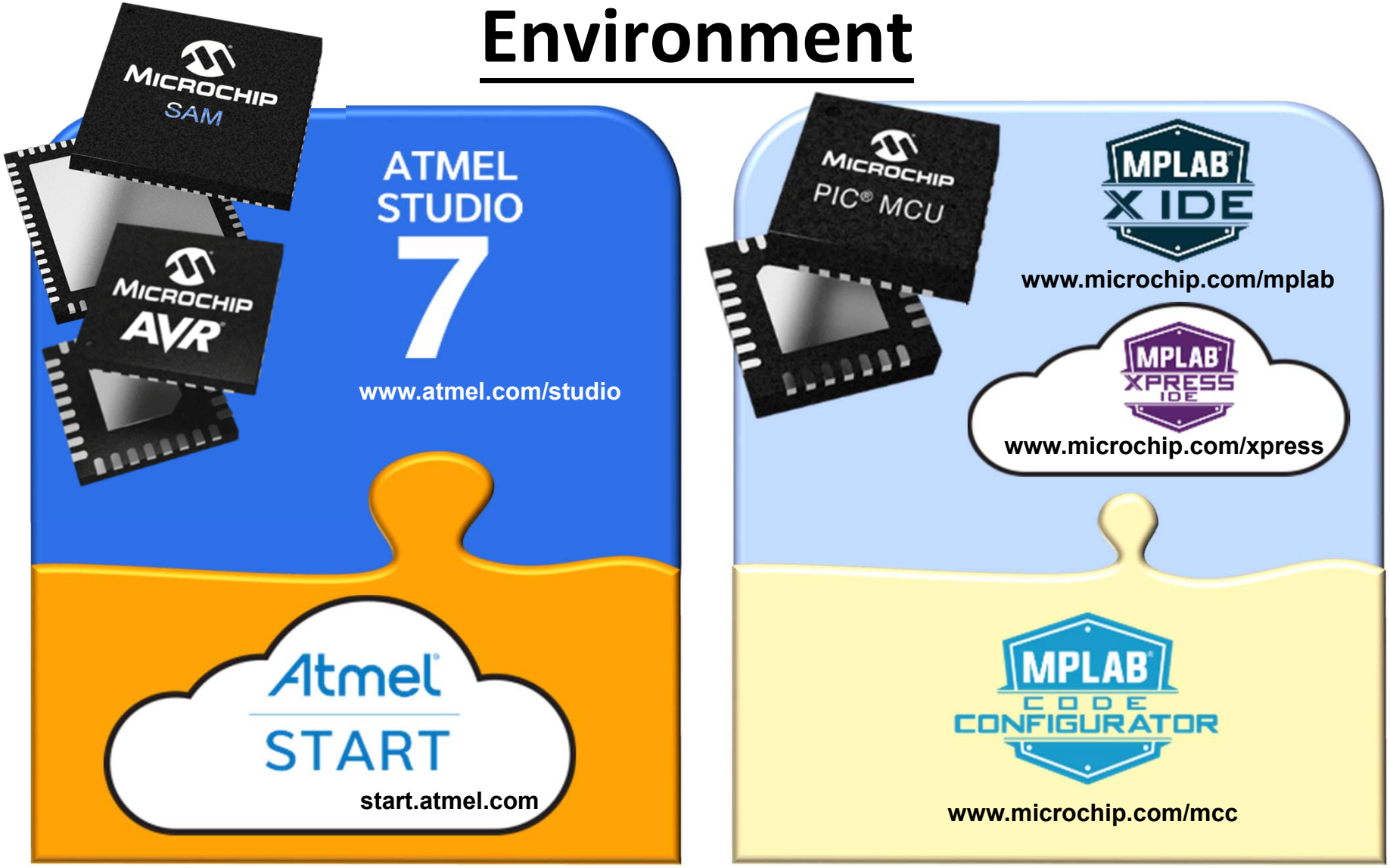
© 2008 Microchip Technology Incorporated. All Rights Reserved.

Development Tools Introduction

- IDE, Compiler, ASF & Hardware Tools

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Integrated Development Environment

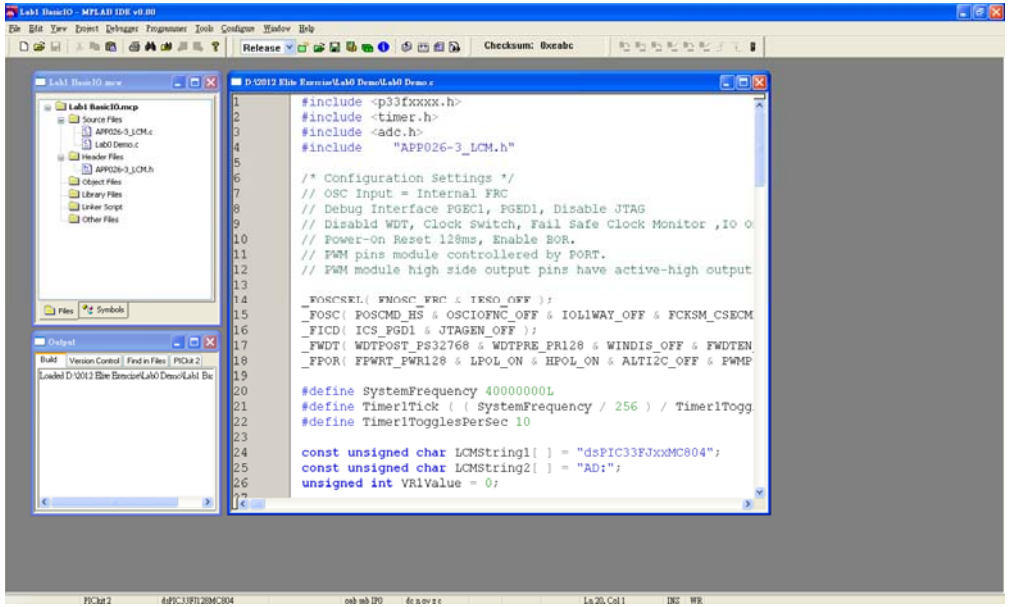


© 2008 Microchip Technology Incorporated. All Rights Reserved.

PIC Series Development

MPLAB® IDE

- Microchip提供的整合式開發環境, 支援全系列8-Bits, 16-Bits及32-Bits的MCU。所有的MCU都可透過相同的環境開發。最終版本是v8.92(不再更新, 以MPLAB X IDE取代)。
- 可整合各式的組譯器/編譯器(MPLAB C, Hi-TECH C, etc.), 開發工具(PICkit3, ICD3, Real ICE, etc..)。

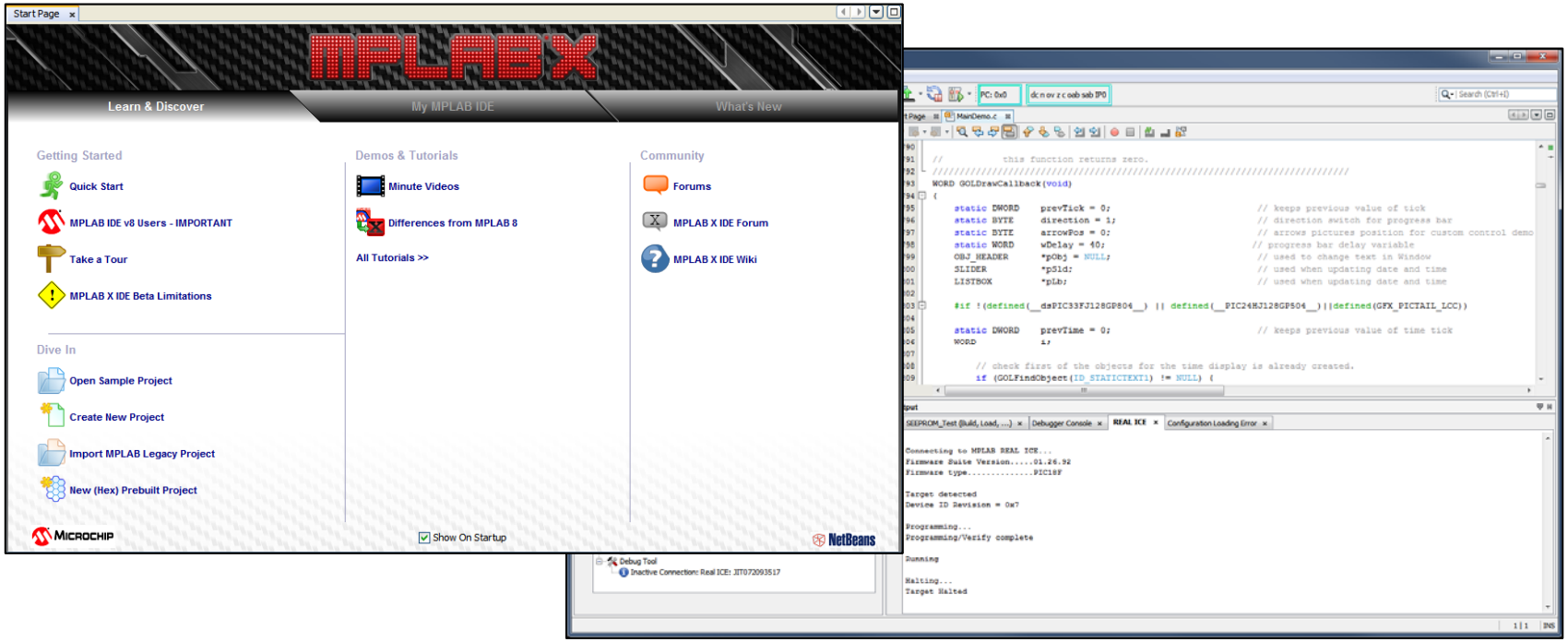


© 2008 Microchip Technology Incorporated. All Rights Reserved.

PIC Series Development

MPLAB® X IDE

- New generation integrated Development Tools, Support PIC Series MCU, Provide Plug-in function to extend more advance function. Java Based, Cross platform, Current version is v4.10。

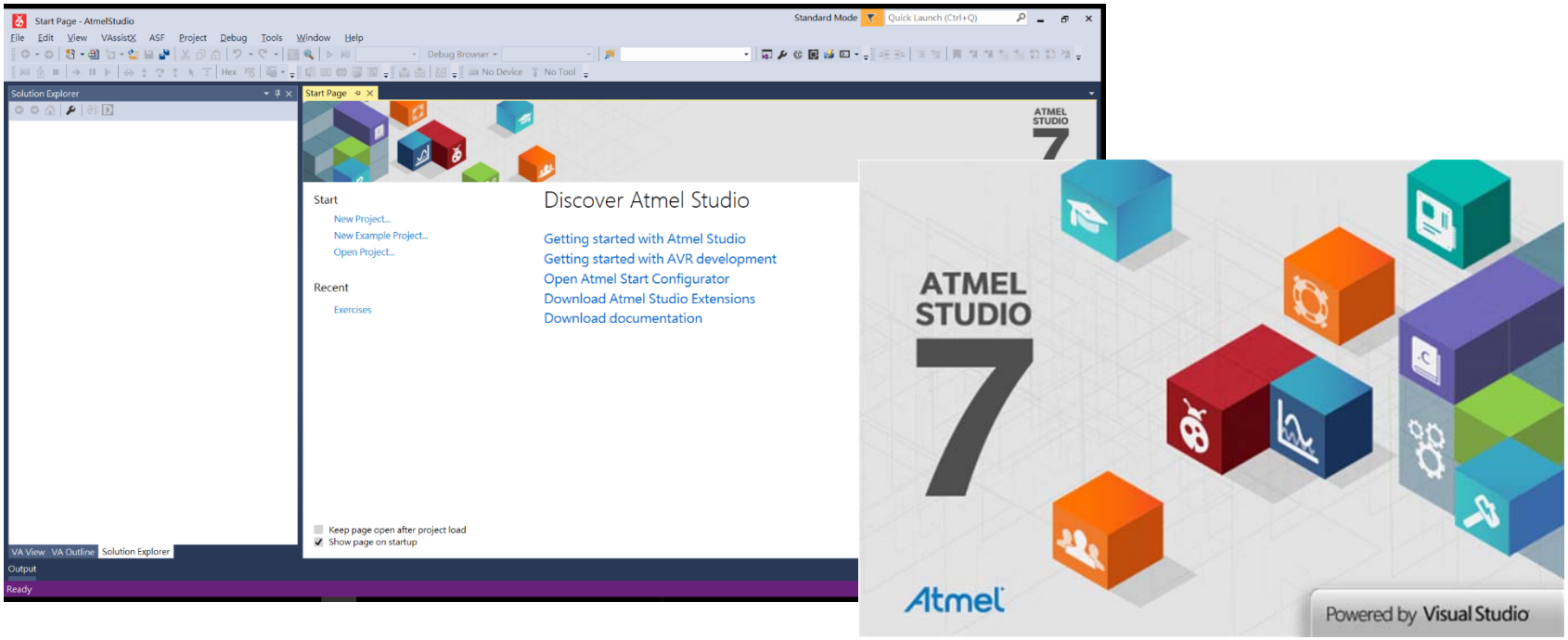


© 2008 Microchip Technology Incorporated. All Rights Reserved.

AVR/SAM Series Development

Atmel Studio 7

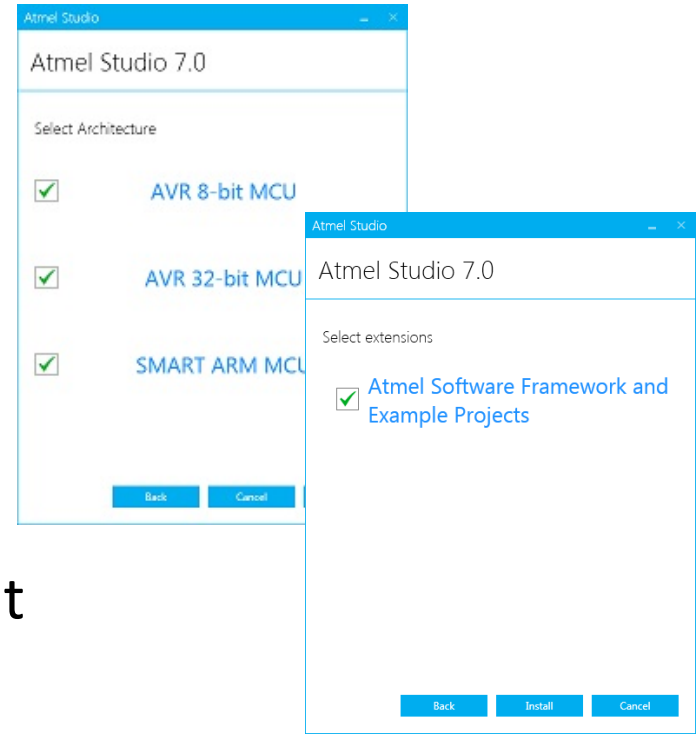
- New generation integrated Development Tools, Support AVR, SAM Series MCU/MPU, Current version is v7.0.1645。
- The GCC compilers already include support AVR/SAM series.



© 2008 Microchip Technology Incorporated. All Rights Reserved.

Customized Installer

- **Provide Web or offline installer**
  - **Web installer**
    A 2.5 MB boots trapper, download and install only selected components.
  - **offline installer**
    908MB for version 7.0.1645.
- **Greater granularity about what you choose to install**
  - Grouping based on SAM, AVR8 and AVR32 architectures
  - ASF is also an optional component



© 2008 Microchip Technology Incorporated. All Rights Reserved.

Tools Installation

- **You can download development tools from below link.**
  http://www.microchip.com.tw/Data_CD/

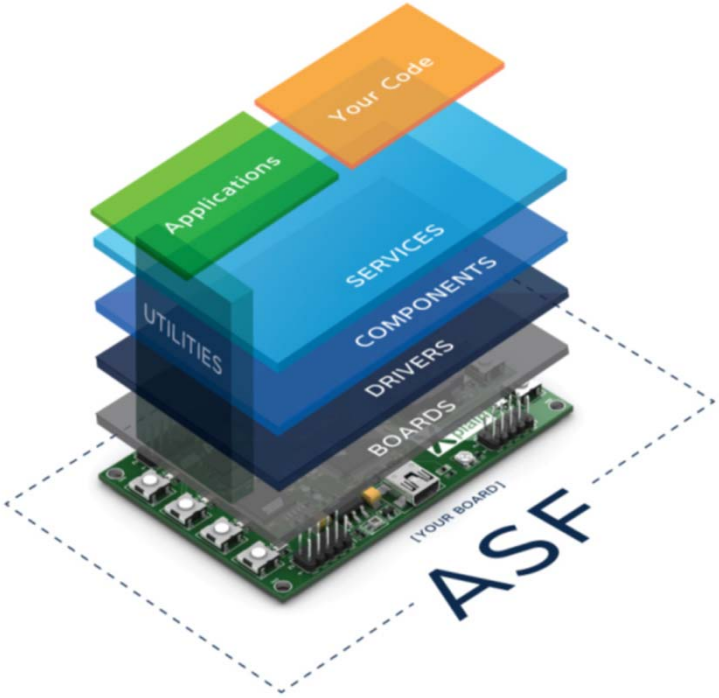
開發軟體, 編譯器	
MPLAB® X IDE	v4.05 Windows(Local) Windows Version Linux Version Mac Version *Detail Info.*
MPLAB® IDE	v8.92(Local)
MPLAB® XC8	v1.44(Local) *Peripheral libraries not include
• Part Support Patch Files	v1.44(Local)
• Peripheral Libraries	v2.00RC3(Local)
	v1.34(Local)
MPLAB® XC16	v1.33(Local) *Peripheral libraries not include
• Part Support Patch Files	v1.33(Local)
• Peripheral Libraries	v2.00(Local)
	v1.24(Local)
MPLAB® XC32	v1.44(local) *Peripheral libraries not include
• Part Support Patch Files	v1.44(Local)
• Peripheral Libraries	v1.0.0.0(local)
	v1.34(Local)
MPLAB® C18 Lite	v3.47(Local)
MPLAB® C30 Lite	v3.31(Local)
MPLAB® C32 Lite	v2.02a(Local)
HI-TECH C for PIC10/12/16	v9.83(Local)
HI-TECH C for PIC18	v9.80(Local)
Atmel Studio	v7.0 Build 1645(Local)
• Hot Fix KB2999226 (Win7)	x64 x86
• Hot Fix KB2978092 (Win7)	x64 x86
AVR 8-bit Toolchain	v3.5.4.91(Local)
ARM GNU Toolchain	v5.3.1.27(Local)
Advanced Software Framework (ASF)	v3.36.0.58(Local)

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Advance Embedded Software

Feature rich software libraries for Atmel MCUs

- **Advance Software Framework**
  - Integrated in Studio 7, 3rd party editions available.
  - More than 4000 ready-to-run project examples
  - Most Atmel MCU devices
- **SoftPack**
  - Drivers, software services and libraries, mainly for MPU devices.
  - GCC & IAR compilers supported
- **Libraries & Appnotes**
  - SDKs for Wireless, Crypto etc..
  - Linux4SAM distribution for ARM-based MPUs

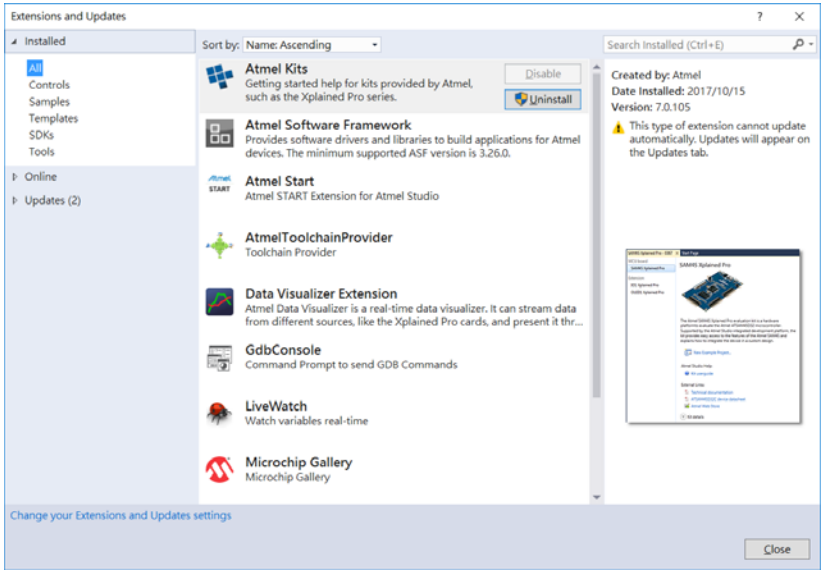


© 2008 Microchip Technology Incorporated. All Rights Reserved.

Microchip Gallery

App store for Atmel Studio extensions

- **Integrated into Studio 7**
  - Productivity enhancing extensions for Studio 7
  - Free, evaluation, paid for extensions available
- **Makes development faster and easier**
  - Software examples, board support, Qtouch configuration tools…
  - Tool and software vendors Visual Micro, Somnium, Micrium, Percepio, and others
- **Easy Access, Easy to Use**
  - From Studio through Extension Manager
  - Web interface: gallery.microchip.com



© 2008 Microchip Technology Incorporated. All Rights Reserved.

Programmer/Debugger

Atmel-ICE

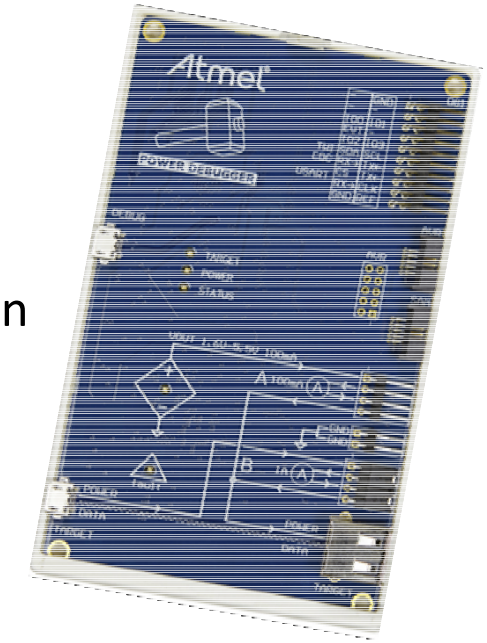
- **Supports programming and debugging all Cortex-M based SAM, AVR and UC3 based parts**
  - Programming/ Debugging AVR/SAM MCU/CPU over JTAG, SD, PDI, SPI, TPI and aWire etc.
  - Target operating voltage range 1.62V to 5.5V
  - ITM serial trace support on Cortex-M based SAM MCUs
- **Supported in Atmel Studio, IAR, Keil and more**
  - Any IDE that supports the CMSIS-DAP protocol can support Atmel-ICE
- **3 different editions**
  - USD 50 - Low cost PCBA only
  - USD 90 - Basic edition with most common cables
  - USD 130 - Including all cables and adapters

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Programmer/Debugger

Power Debugger

- **Simultaneous debug & power measurement**
  - Debugging over CMSIS-DAP
  - Extended to support SWO trace, same as Atmel-ICE
- **Two independent channels for power measurements**
  - Channel A for high accuracy, lower currents
  - Channel B for higher currents with lower resolution
- **CDC virtual COM port**
- **DGI for streaming application & power data**
  - Open specification, docs provided
- **Provides target power, 1.6 – 5.5V, up to 100mAStatus LEDs for debugging & target supply**



© 2008 Microchip Technology Incorporated. All Rights Reserved.

Getting Started First Project

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab0 – First Project

- Try to create a first project use Studio 7.
- Learn how to use edit, build, project manager, debug and program functions.

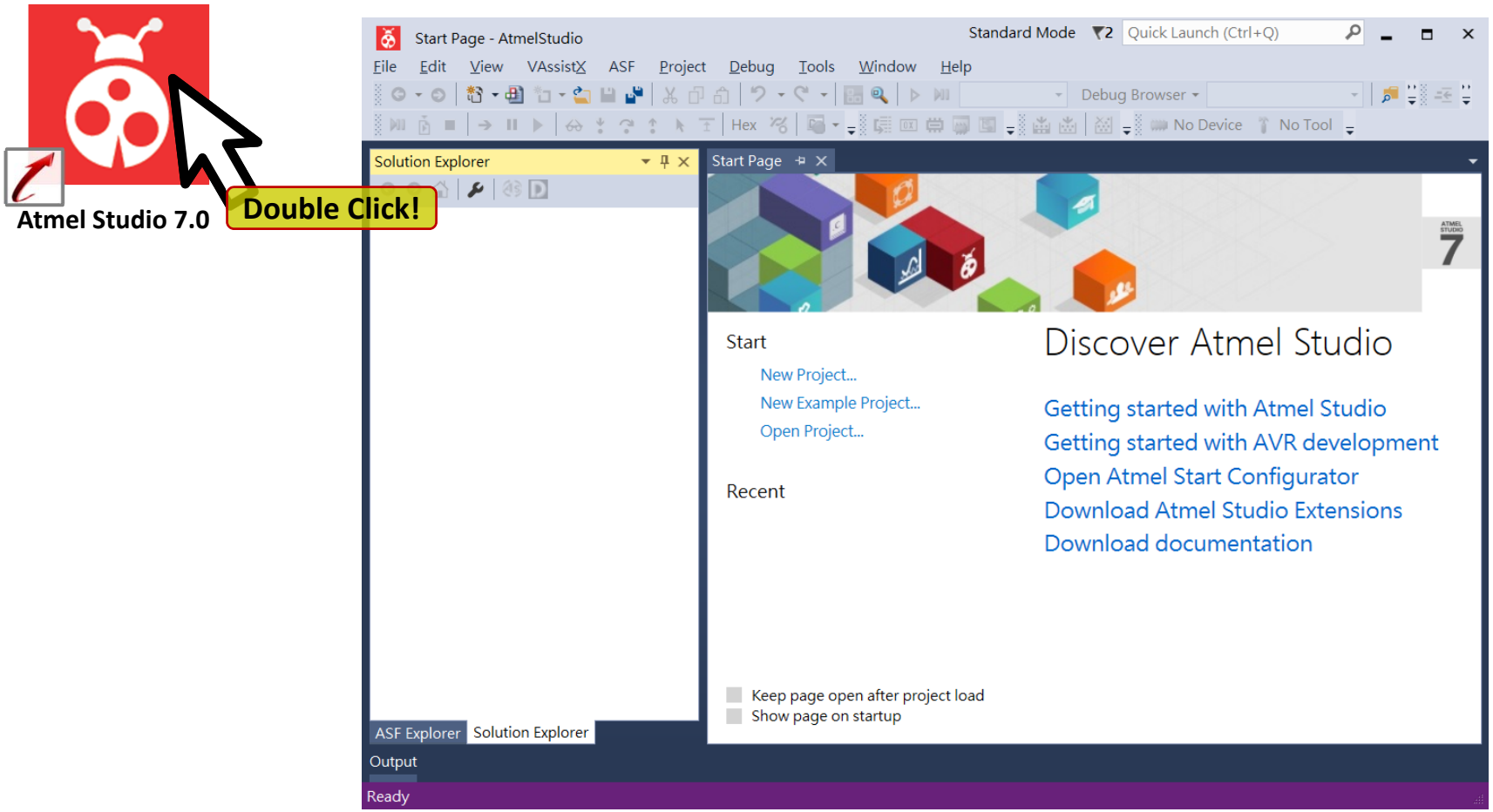
- How to Start ?

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab0 - Create Project

Step 1

- Open Atmel Studio 7, first.

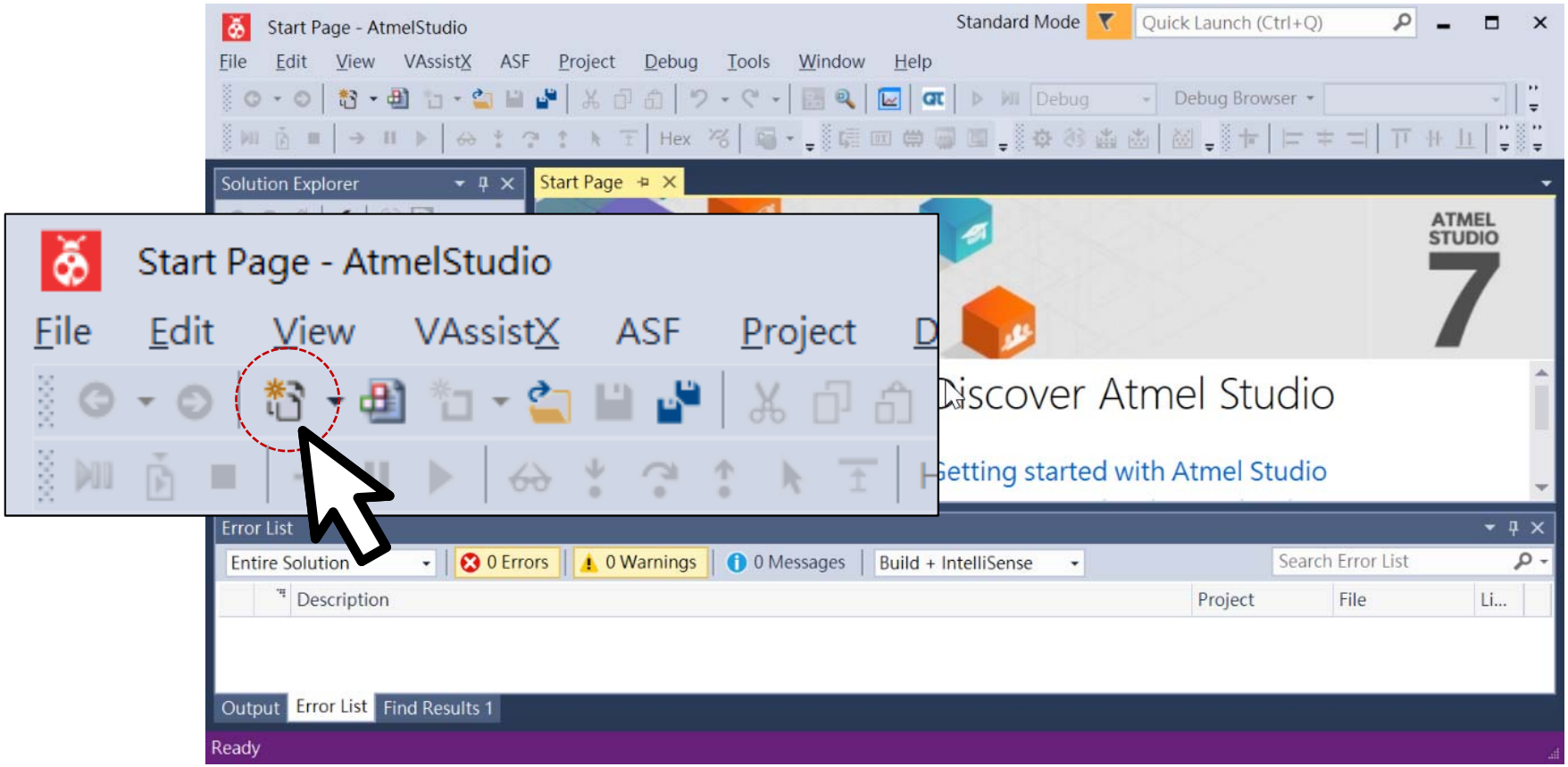


© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab0 - Create Project

Step 2

- Select **File ▸ New ▸ Project**
  or Click **icon** 

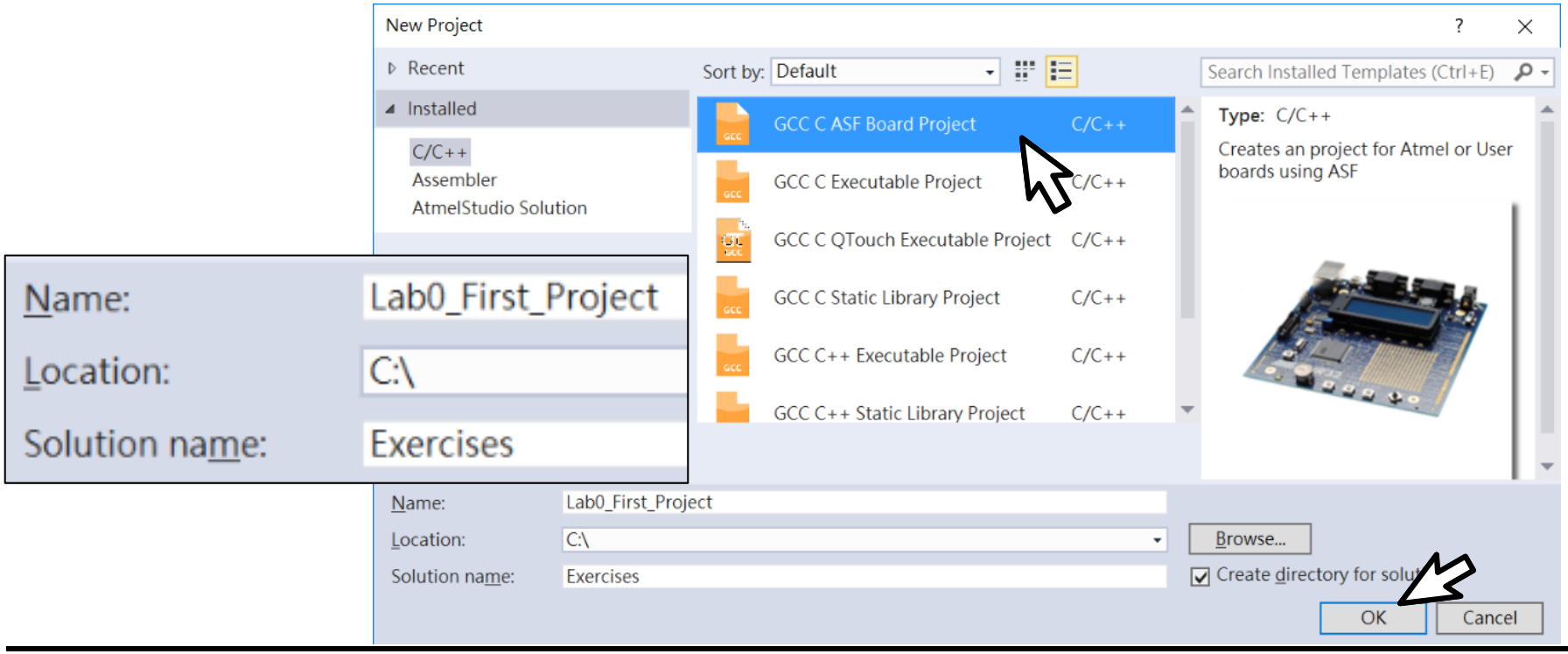


© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab0 - Create Project

Step 3

- Select **GCC ASF Board Project**
- Project Name **Lab0_First_Project**
- Location **C:**
- Solution name **Exercises**

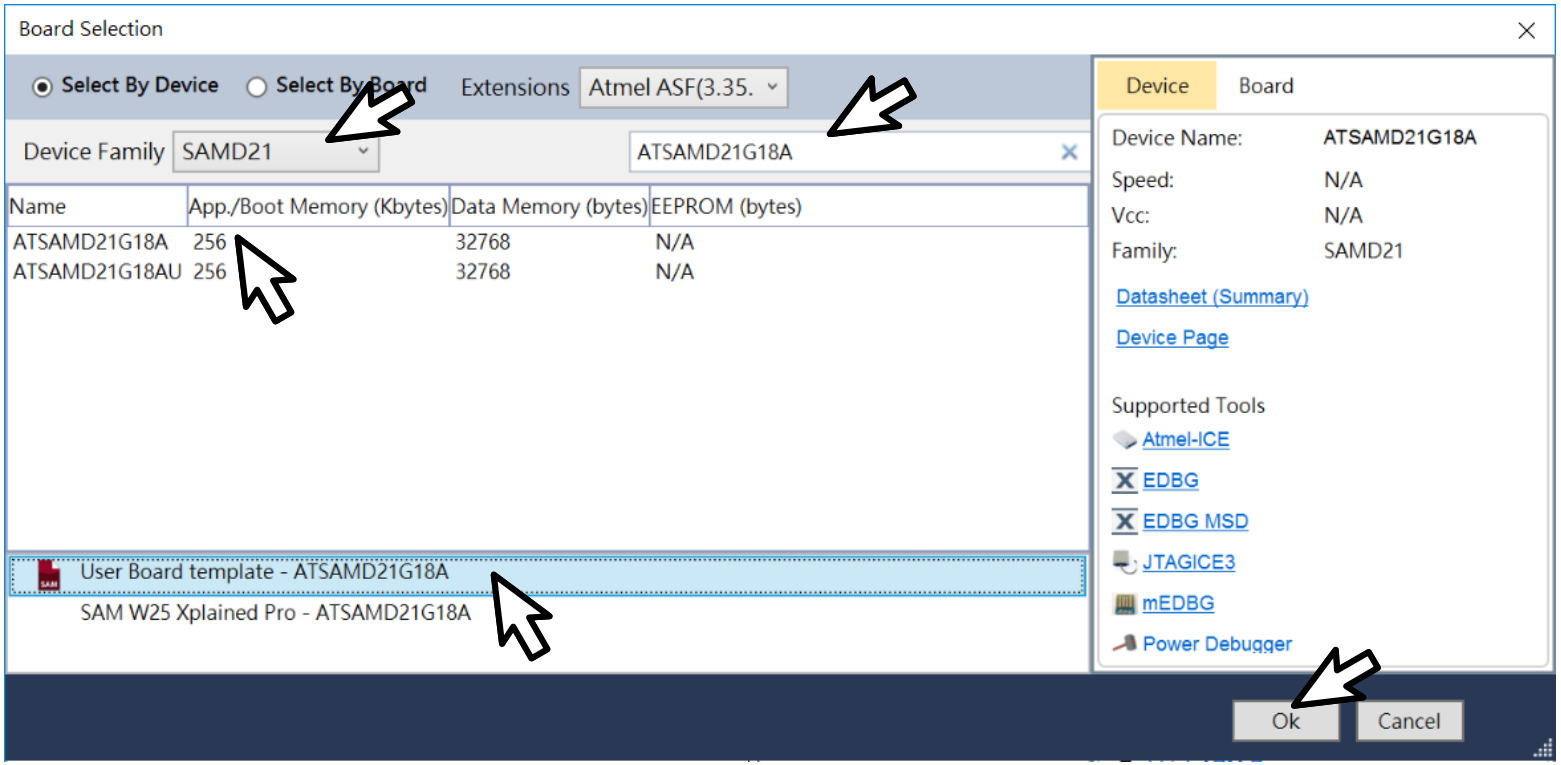


© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab0 - Create Project

Step 4

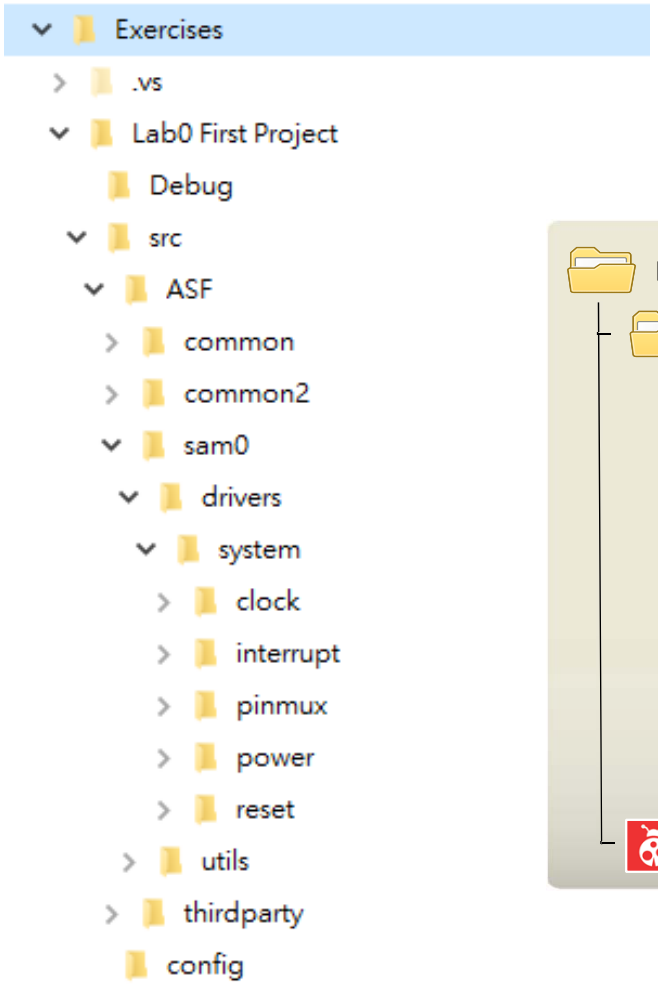
- Use **Device Family** to filter **SAMD21** or
- Search **ATSAMD21G18A**
- Select **User Board Template - ATSAMD21G18A**



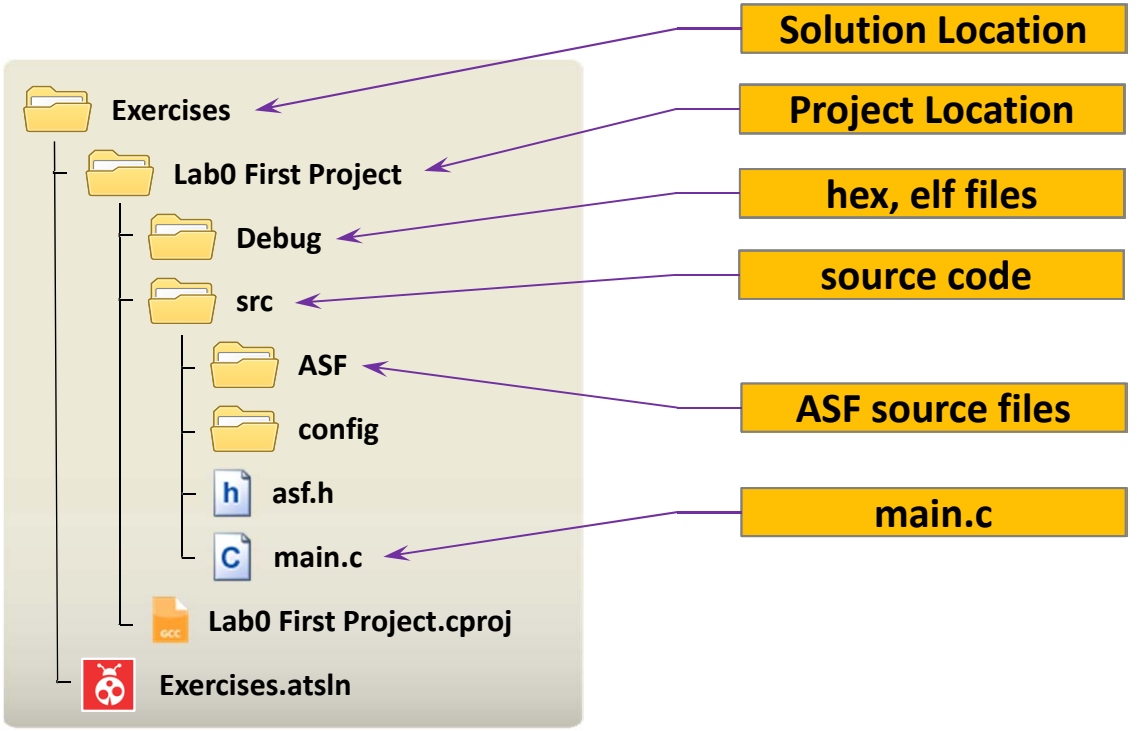
© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab0 - Create Project

Step 5



Directory Architecture

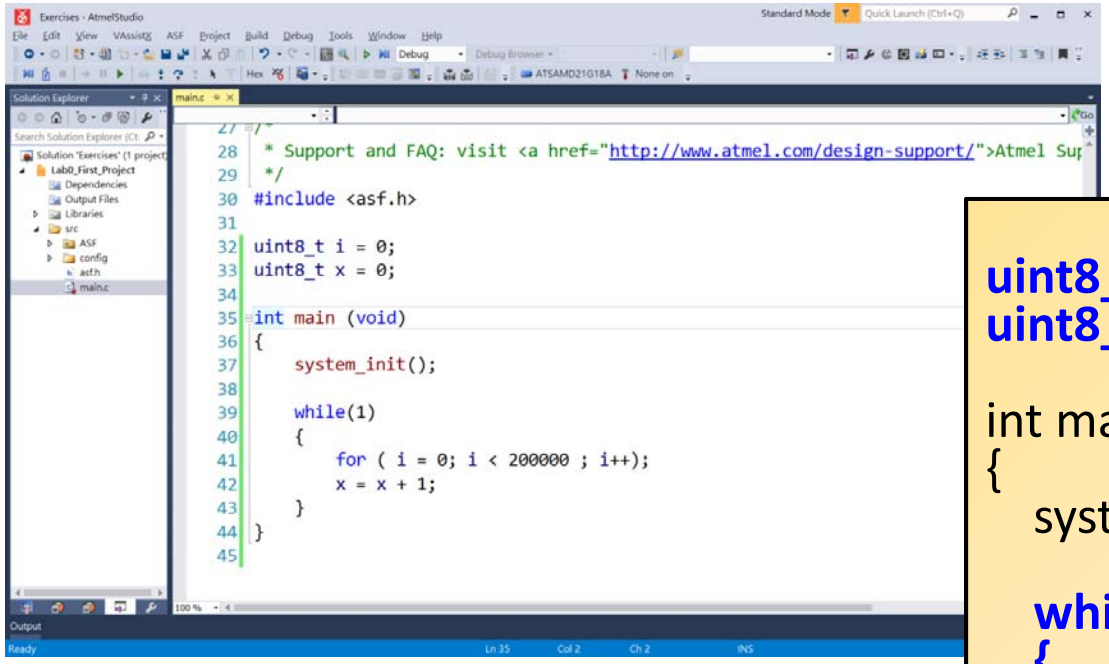


© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab0 - Create Project

Step 6

- Click **main.c** to view and edit it



Add below code to main()

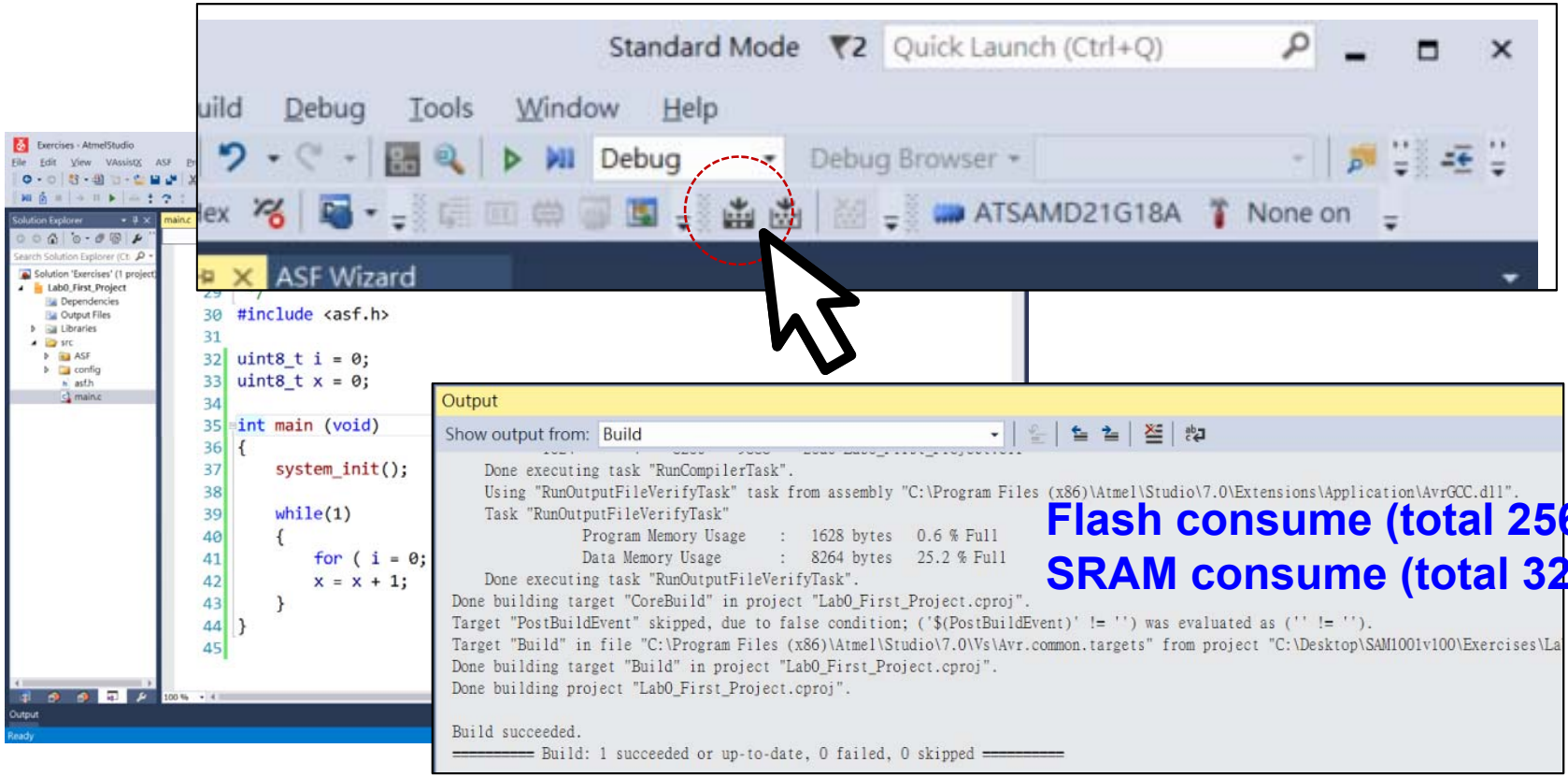
```
uint8_t i = 0;
uint8_t x = 0;

int main (void)
{
   system_init();

   while(1)
   {
      for ( i = 0; i < 200000 ; i++);
      x = x + 1;
   }
}
```

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab0 - Create Project

Step 7

- Click Icon  or press F7 to build your project.
- All message show at output windows.



© 2008 Microchip Technology Incorporated. All Rights Reserved.

Build FAILED !

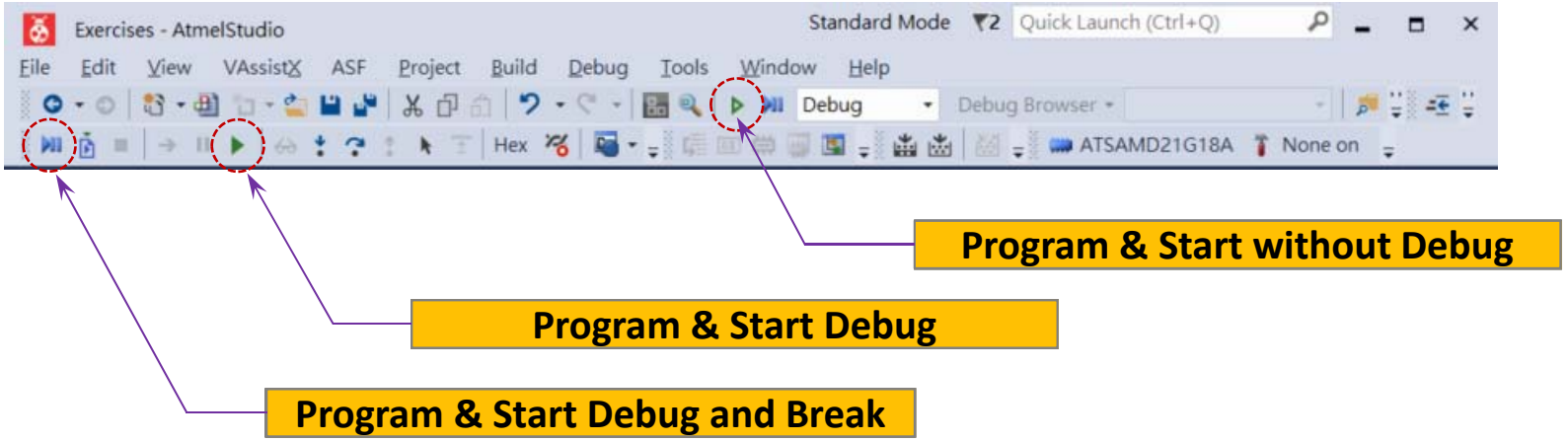
- **Build FAILED表示程式有誤,Output視窗會提示錯誤訊息。可使用滑鼠雙擊錯誤訊息, 會跳到程式中有錯誤的地方。接著觀察, 冷靜的判斷, 找出錯誤。**
- **常見的錯誤:**
  - **大小寫不一致:**
    C 語言中大小寫不同就代表不一樣的名稱。
  - **變數未宣告, 或重複宣告:**
    變數一定要先被宣告才能使用 ,且僅能宣告一次。若要使用在其他檔案所宣告的變數, 則需加上*extern*修飾詞。
  - **敘述(Statement)區塊不完整:**
    少了敘述結尾的分號(;)或Statement Block的大括弧({ , }) 。
  - **引數或參數列不對稱:**
    呼叫函式時的參數數量或型別不正確。
    Ex:void Func(int, int, int); 呼叫時要給3個整數型態的參數。

© 2008 Microchip Technology Incorporated. All Rights Reserved.

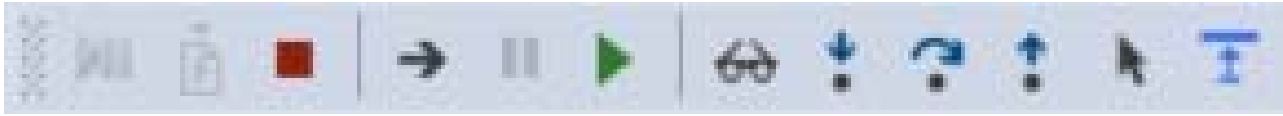
Lab0 - Create Project

Step 8

- Click Icon to program firmware to target board.



- Debug function enable if start at debug mode.

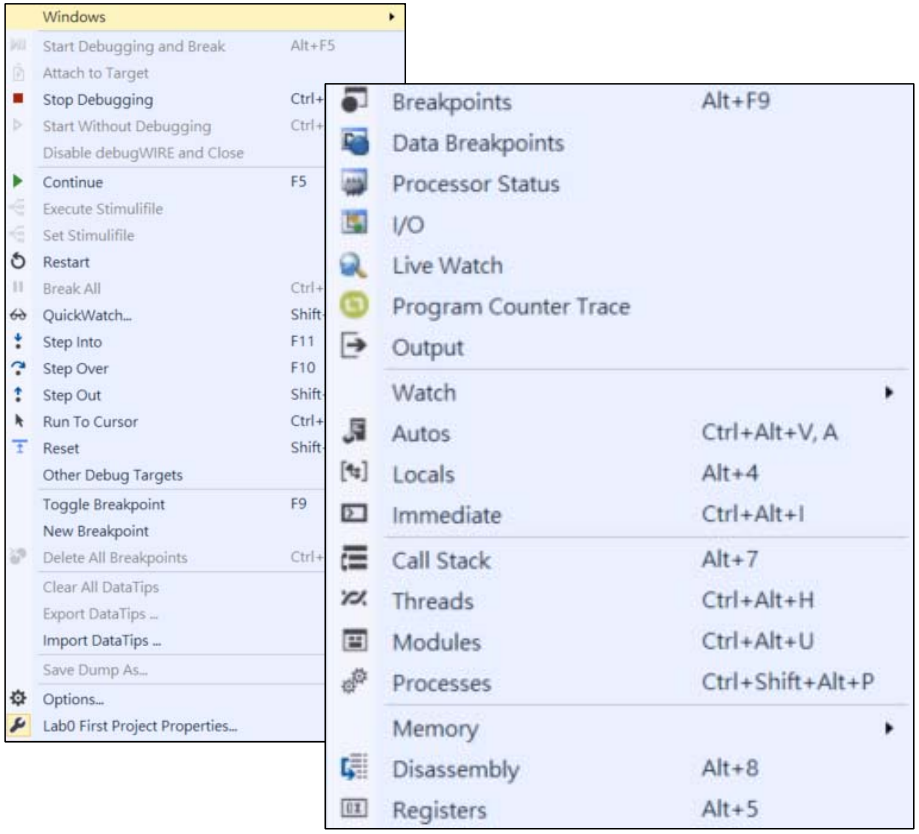


© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab0 - Create Project

Step 8

- **Studio 7 provide a lot of debug window**
- Select **Debug ▸ Windows**

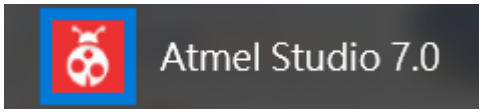


© 2008 Microchip Technology Incorporated. All Rights Reserved.

EVB setup

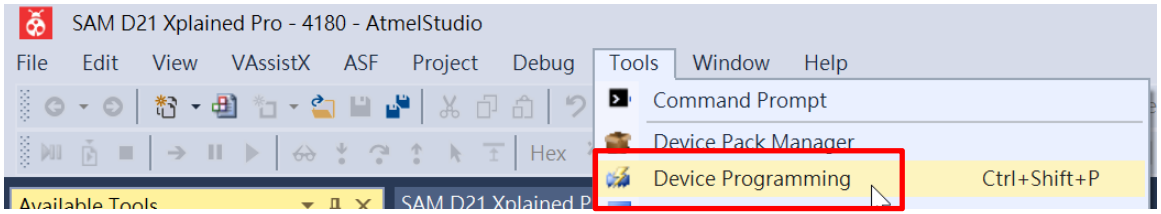
Check to see if your EVB boards have the latest embedded debug (EDBG) firmware version:

Step 1: Run Studio 7

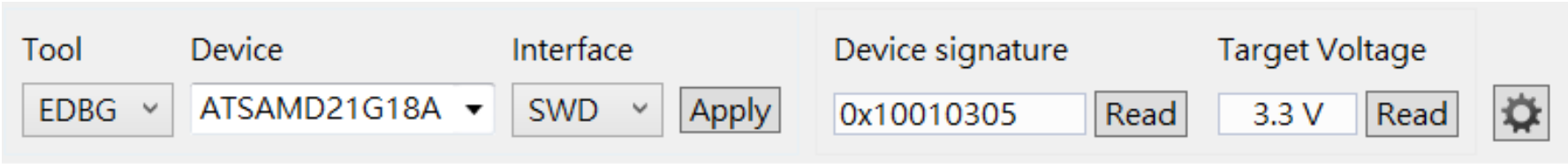


Step 2: Connect USB to APP-ESS17-2
Program USB port

Step 3: Select Tools ► Device Programming



Step 4: Use the dropdown menu and select as below.

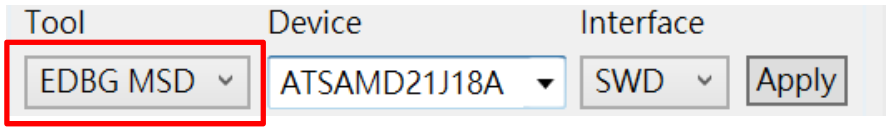


Step 5: Click Apply and Read to show Device signature.

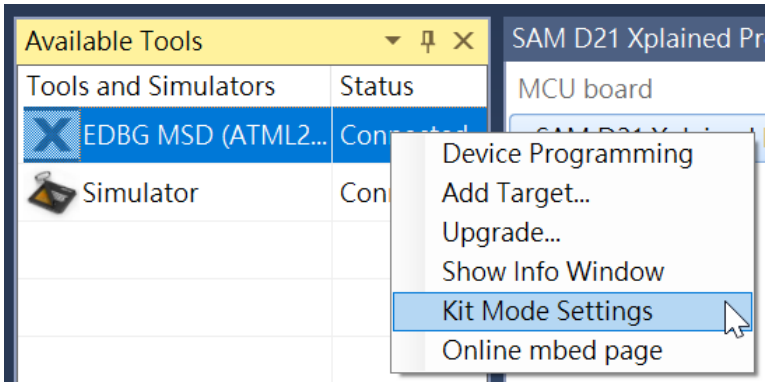
© 2008 Microchip Technology Incorporated. All Rights Reserved.

EVB setup

if your EVB boards to shows "EDBG MSD" in Tool as below
Please go through below steps for correct !!

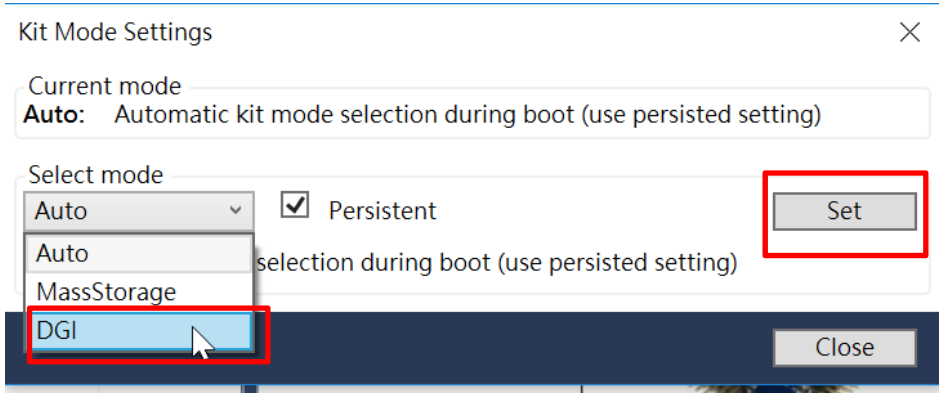


Step 1: Select View ▸ Available Atmel Tools



Step 2: Right click on EDBG MSG and select Kit Mode Settings

Step 3: Use the dropdown menu and select mode to DGI

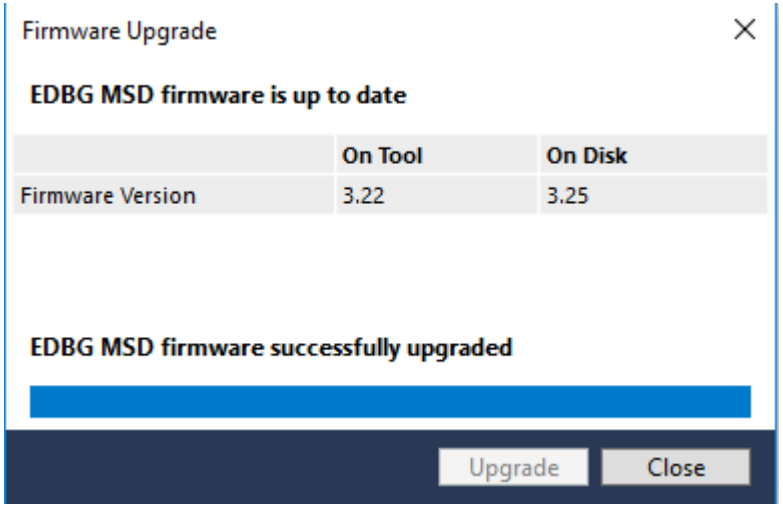


Step 4: Click Set

© 2008 Microchip Technology Incorporated. All Rights Reserved.

EVB setup

- If the EDBG firmware need to be updated, the window shown below will appear :

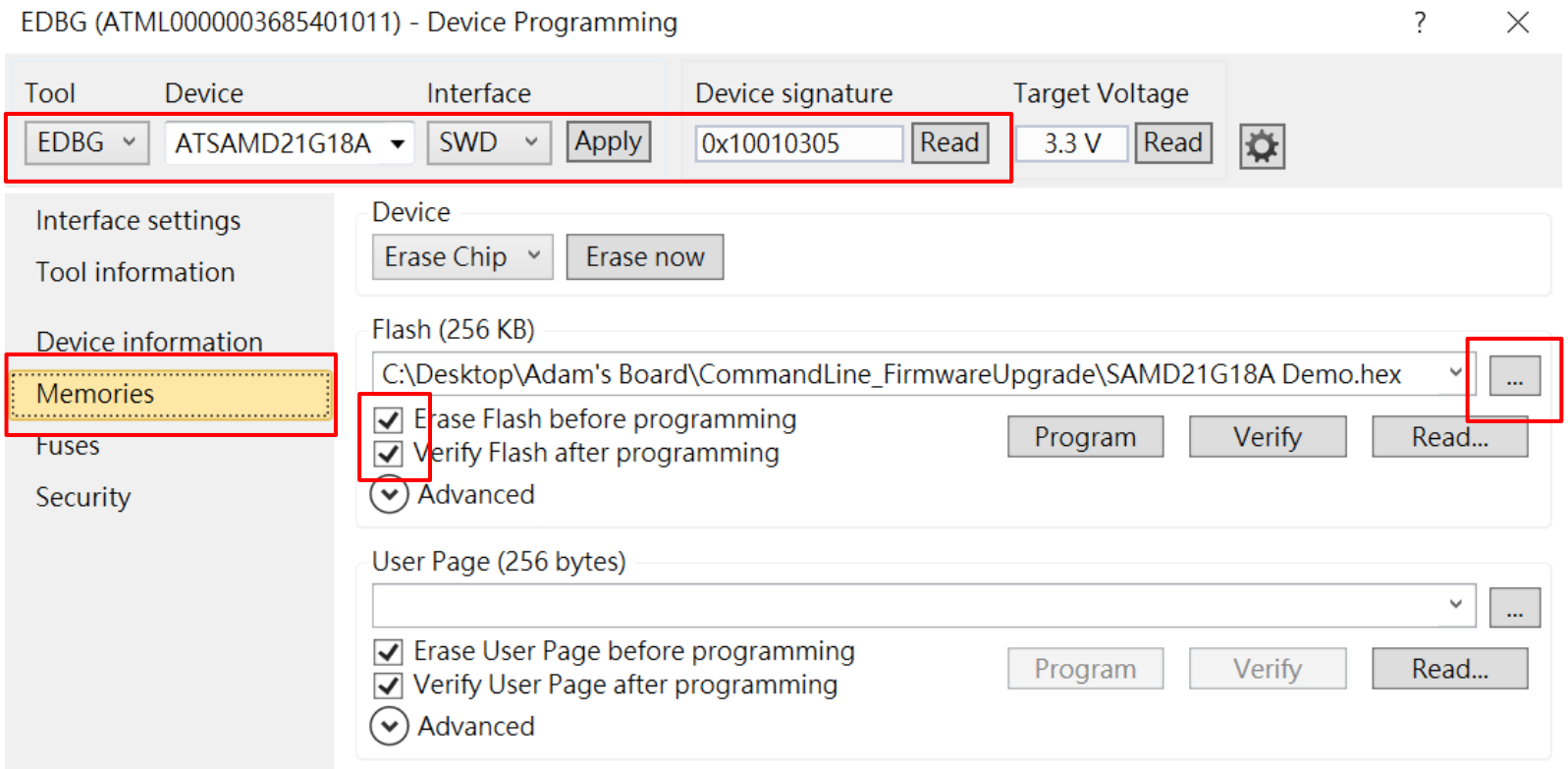


- If an upgrade is required, press “Upgrade”
- If not, the device signature will fill in with the device’s unique serial# and you can close the window.

© 2008 Microchip Technology Incorporated. All Rights Reserved.

EVB setup

- Program Pre-build ready binary or HEX file to EVB.
- Select Tools ▸ Device Programming
- Select correct Tool/Device/Interface ▸ Apply ▸ Read
- Select Memories

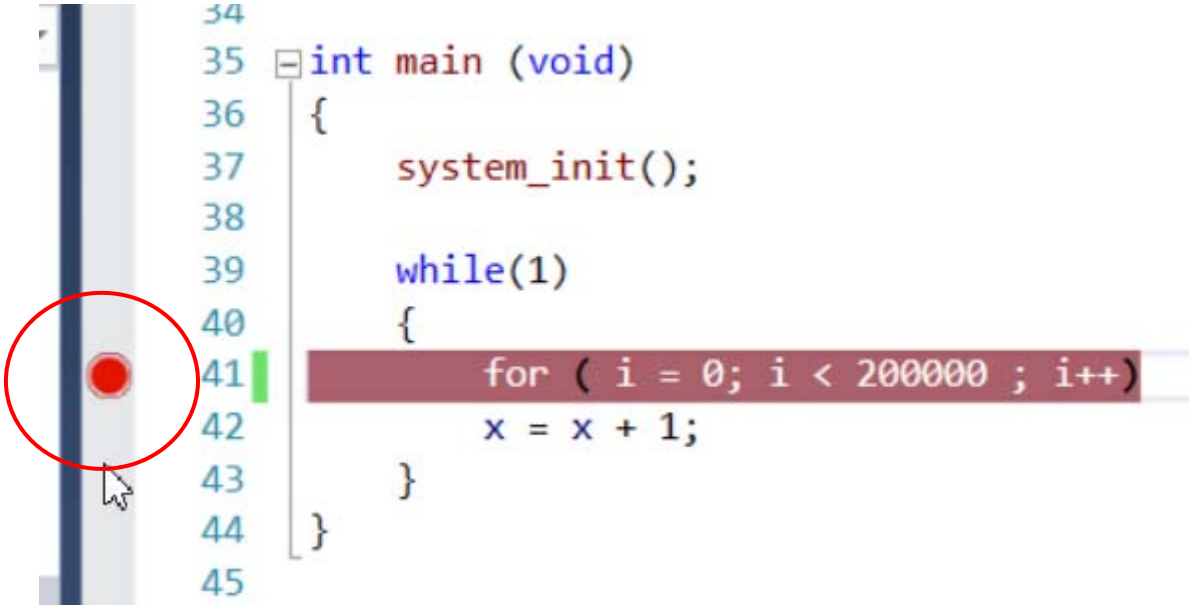


© 2008 Microchip Technology Incorporated. All Rights Reserved.

Debug Windows

Set Break point

- Click **Gary bar** of line for add a Break Point  or
- Press **F9** for add a Break Point

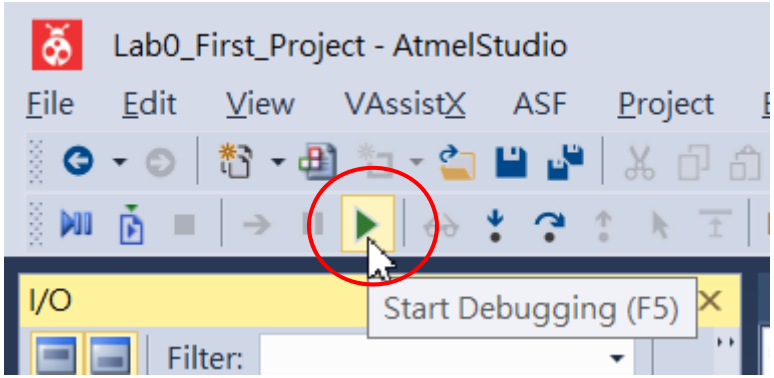


© 2008 Microchip Technology Incorporated. All Rights Reserved.

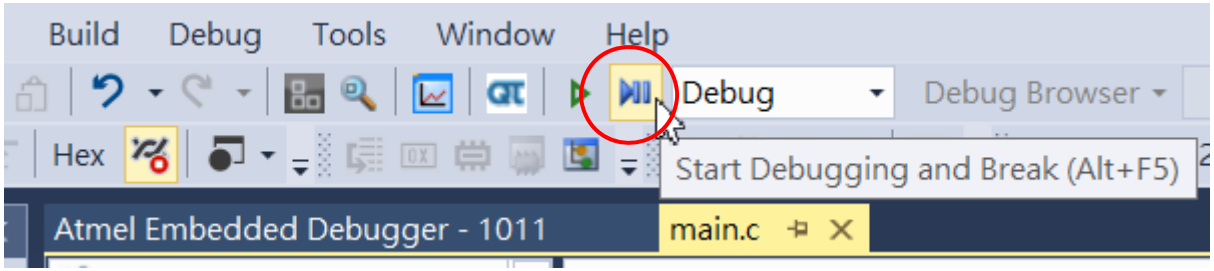
Debug Windows

Start Debug

- Click on icon  press **F5** for start Debug and stop at first Break Point.



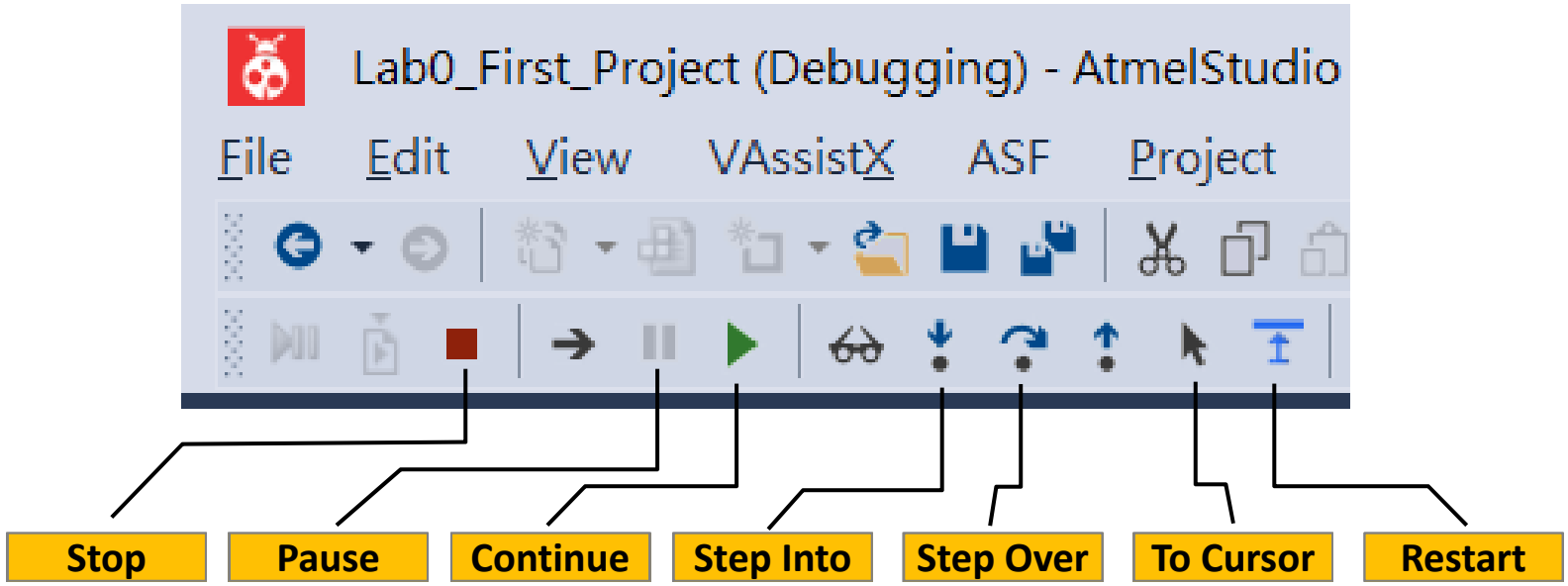
- Click on icon  or press **Alt-F5** for start Debug and stop at main() function.



© 2008 Microchip Technology Incorporated. All Rights Reserved.

Debug Windows

Stop/Pause/Continue/Step/Restart Debug

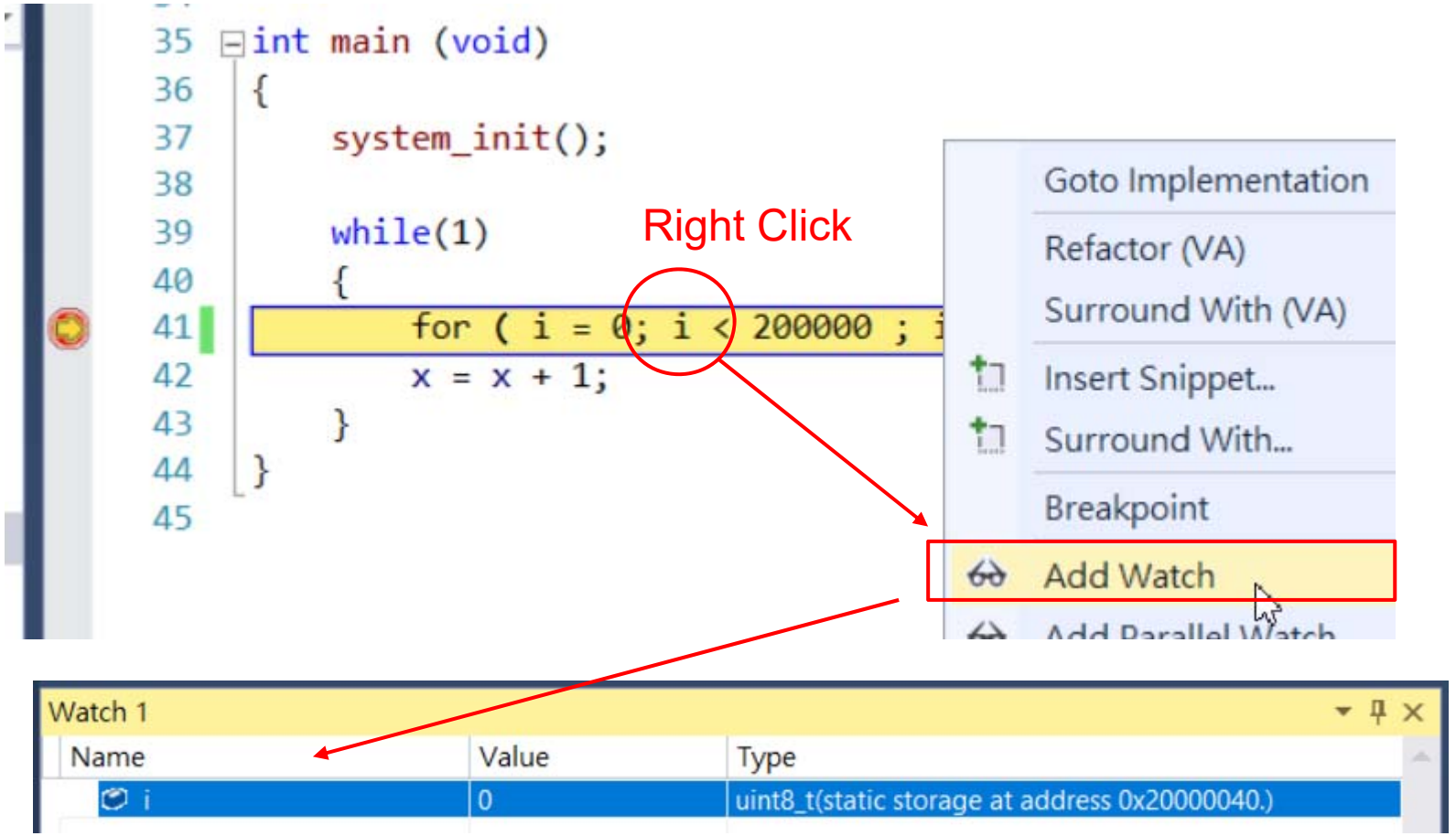


© 2008 Microchip Technology Incorporated. All Rights Reserved.

Debug Windows

Add Watch

- Right Click variable and select to **Add Watch**

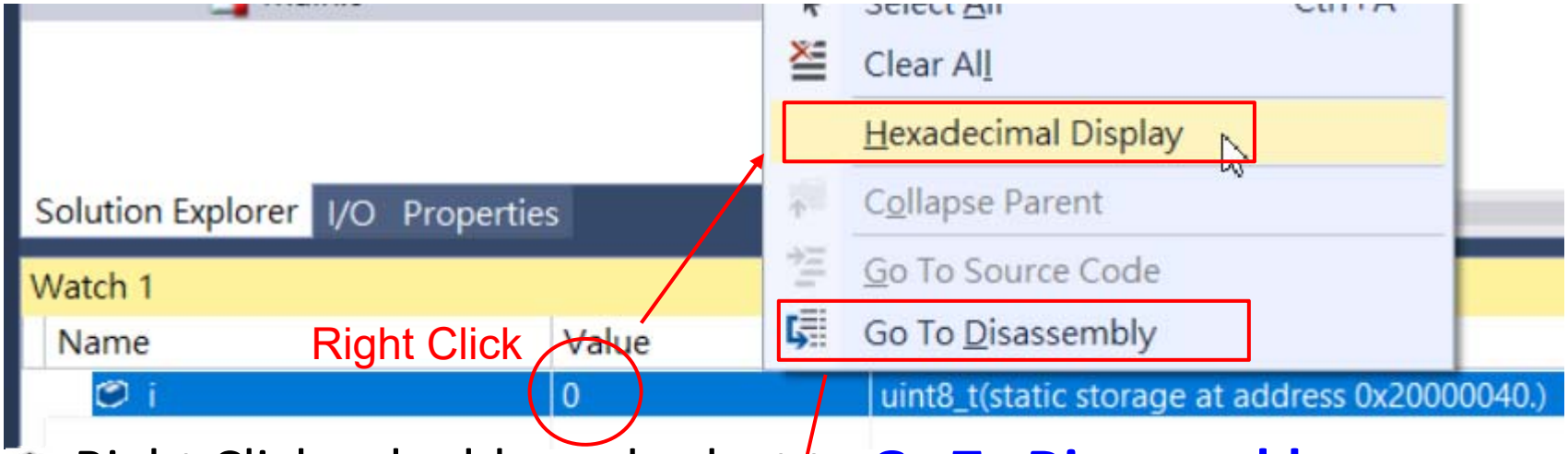


© 2008 Microchip Technology Incorporated. All Rights Reserved.

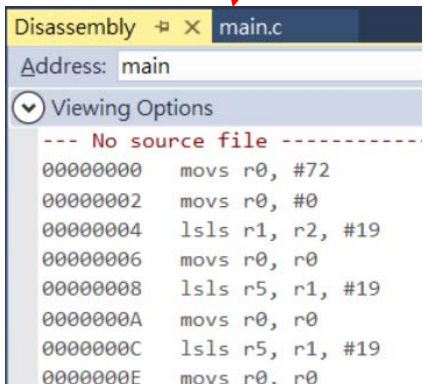
Debug Windows

Change Watch to Hex type or Assemble view

- Right Click variable and select to **Hexadecimal Display**



- Right Click valuable and select to **Go To Disassembly**



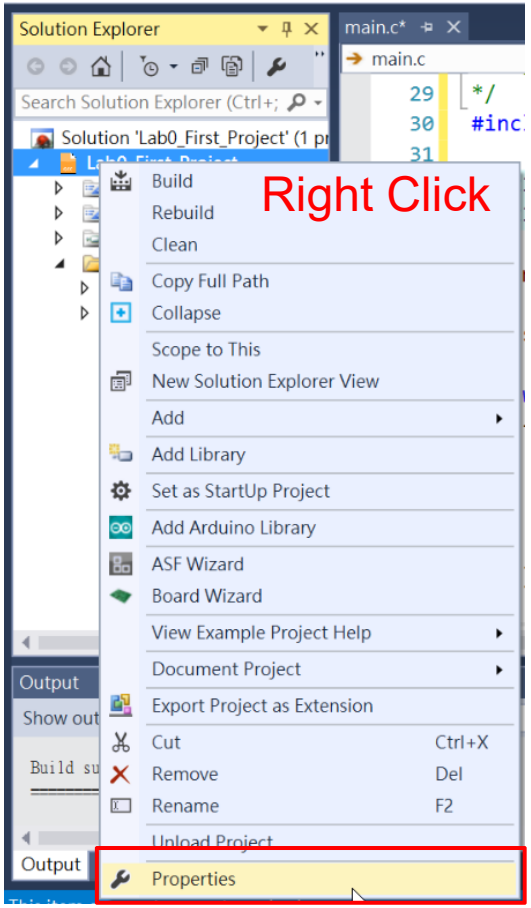
© 2008 Microchip Technology Incorporated. All Rights Reserved.

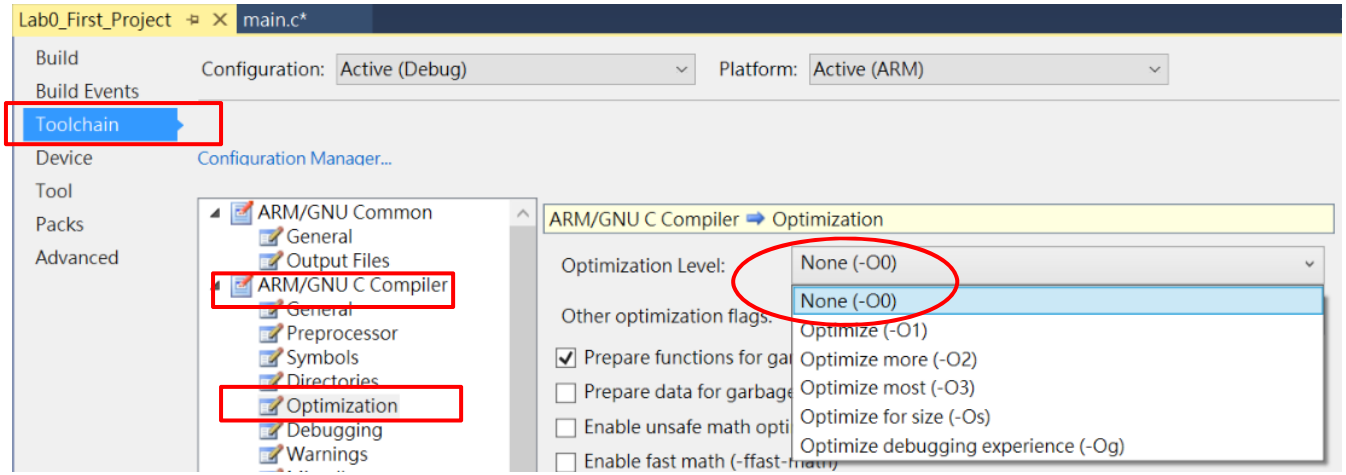
Debug Windows

Compiler Optimization option while debug

- Right click project and select **Properties**
- In **Toolchain ▸ ARM/GNU C Compiler ▸ Optimization**

Select to None (-O0)



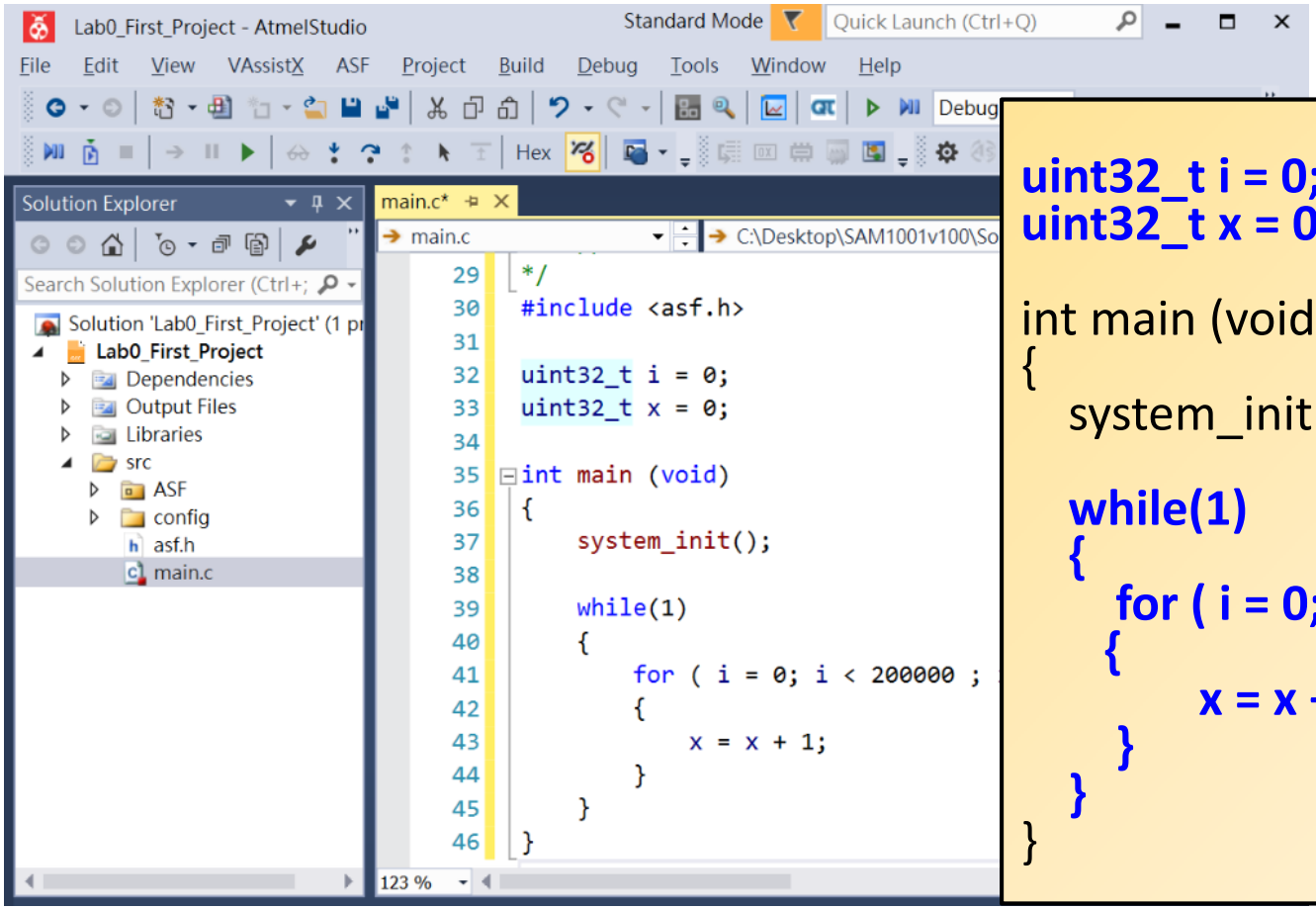


© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab0 - Create Project

Step 9

- Modify **main.c** for debug trace



Modify code in main()

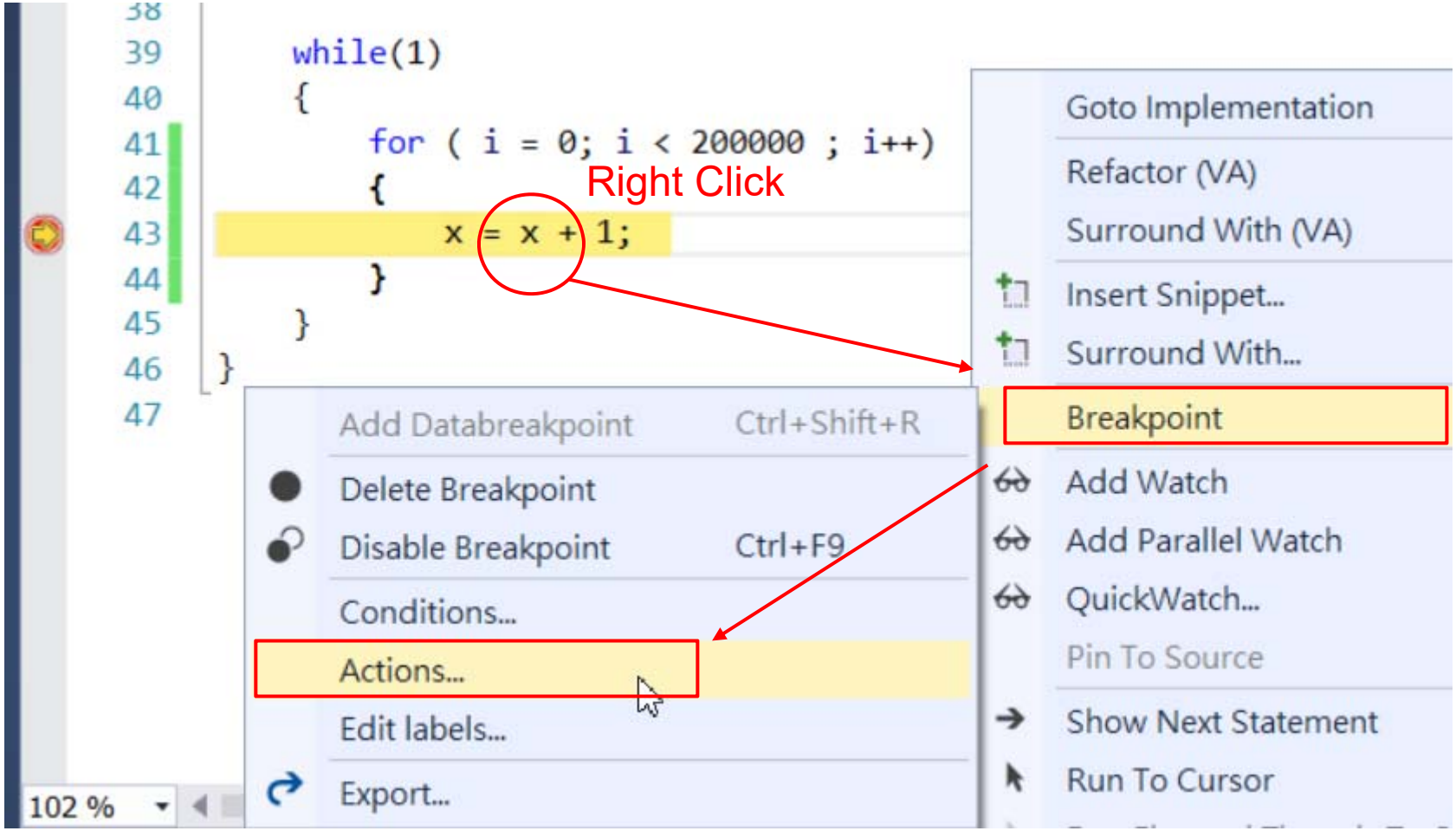
```
uint32_t i = 0;
uint32_t x = 0;

int main (void)
{
   system_init();

   while(1)
   {
      for ( i = 0; i < 200000 ; i++)
      {
         x = x + 1;
      }
   }
}
```

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Debug Windows

Breakpoint : Actions

- Right Click variable and select to **Breakpoint ▸ Action**

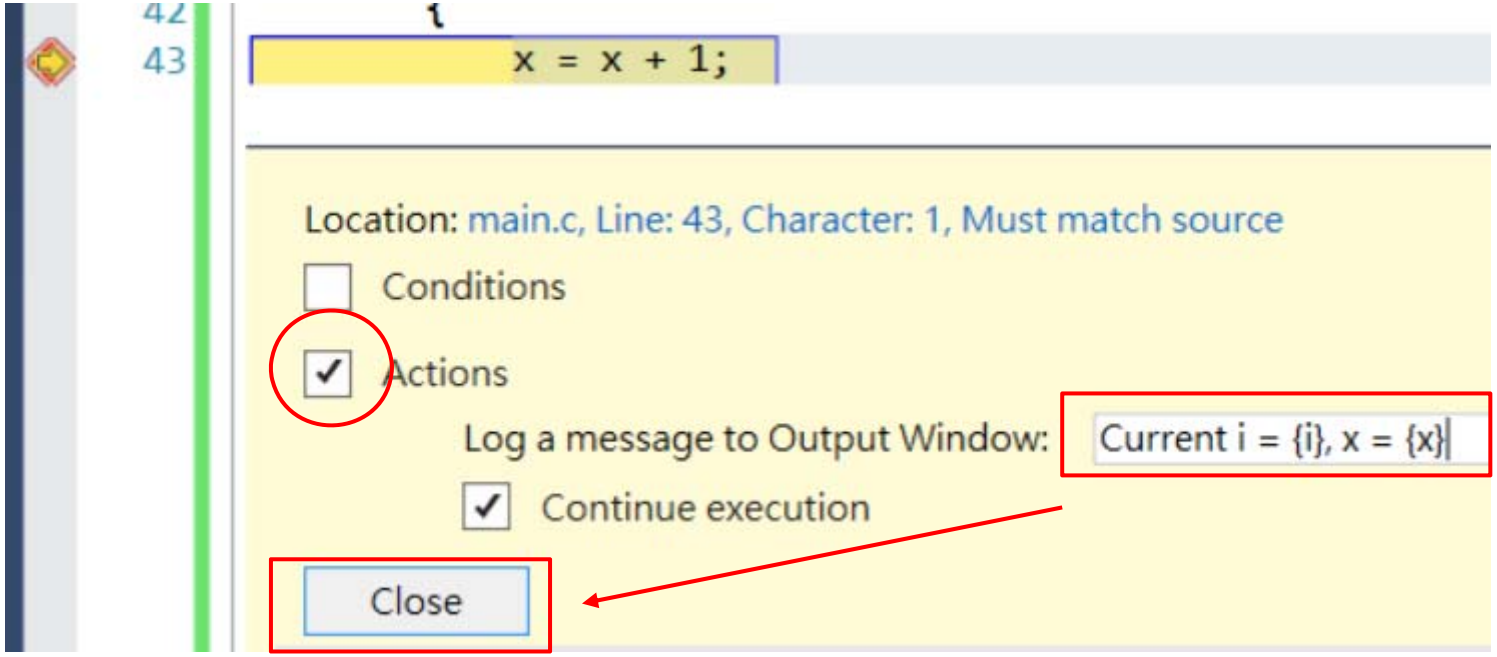


© 2008 Microchip Technology Incorporated. All Rights Reserved.

Debug Windows

Breakpoint : Actions

- Format : Output string { variable }
- Ex. Current i = {i}, x = {x}



© 2008 Microchip Technology Incorporated. All Rights Reserved.

Debug Windows

Breakpoint : Actions

- Break Point will change to  and Action debug will show in Output window.

```
39      while(1)
40      {
41          for ( i = 0; i < 200000 ; i++)
42          {
43              x = x + 1;
44          }
45      }
46  }
```

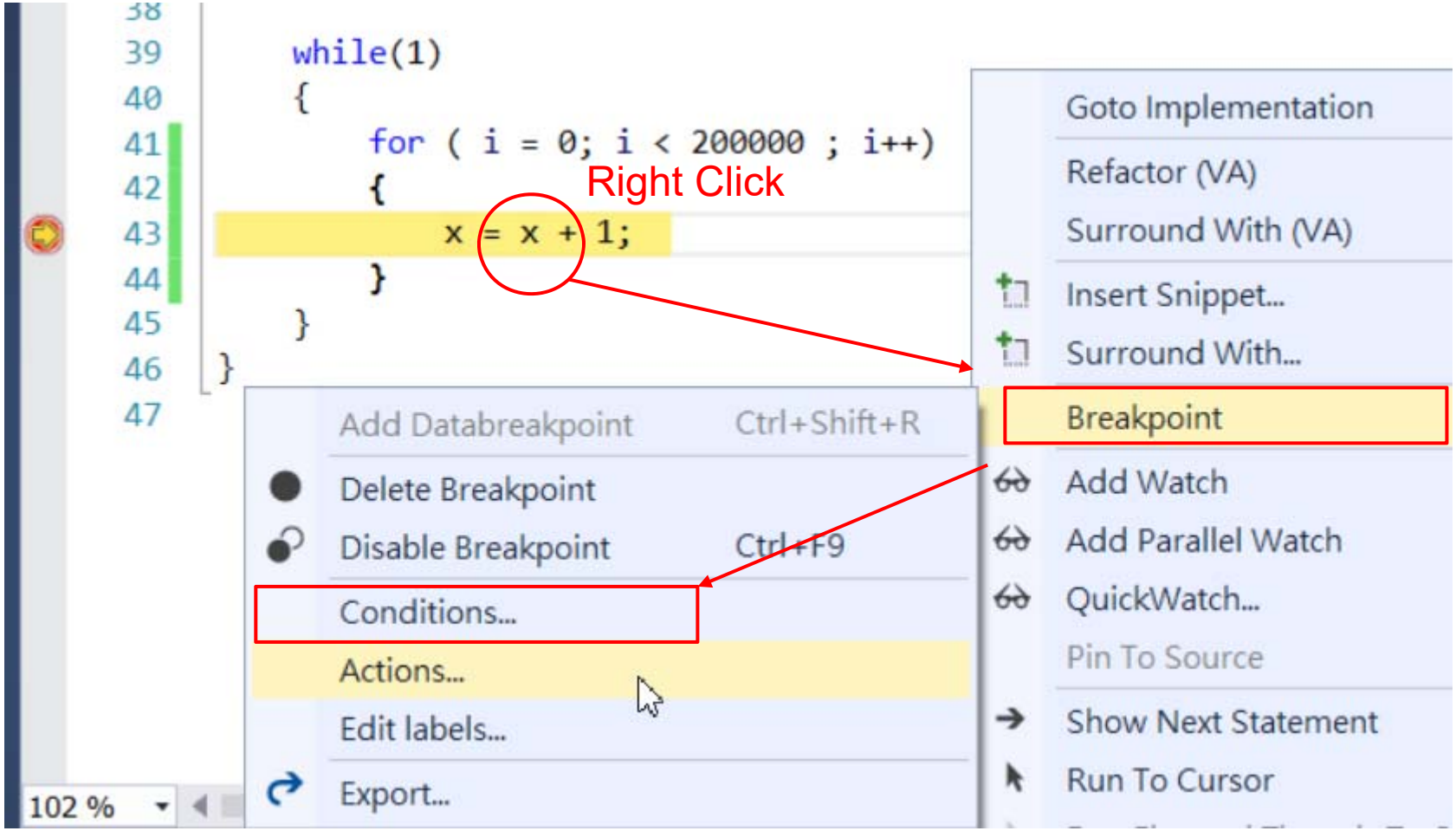
```
Output
Show output from: Debug
Current i = 0, x = 0
Current i = 1, x = 1
Current i = 2, x = 2
Current i = 3, x = 3
```

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Debug Windows

Breakpoint : Conditions

- Right Click variable and select to **Breakpoint ▸ Conditions**

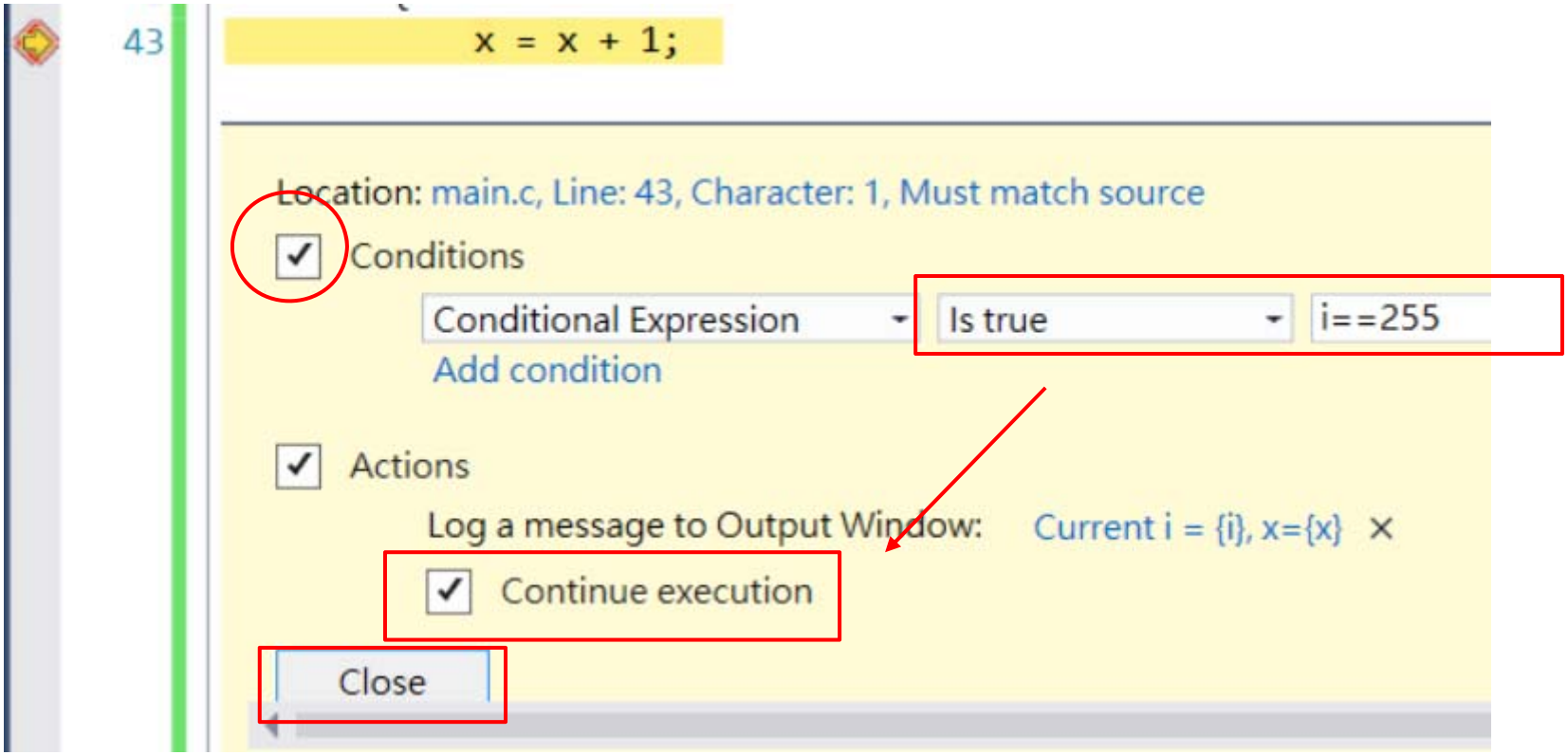


© 2008 Microchip Technology Incorporated. All Rights Reserved.

Debug Windows

Breakpoint : Conditions

- **Condition Expression to use C expression :**
- **Ex. i == 255**

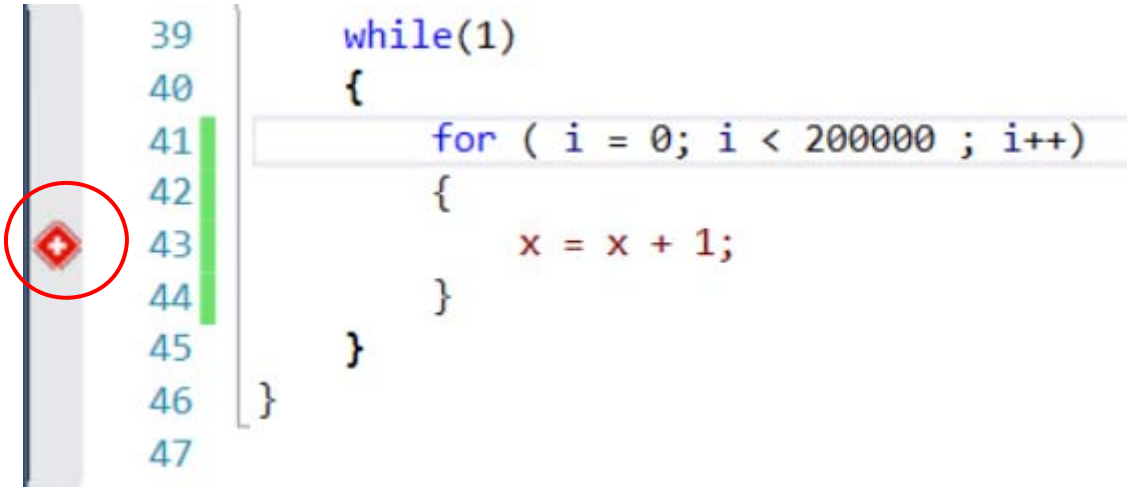


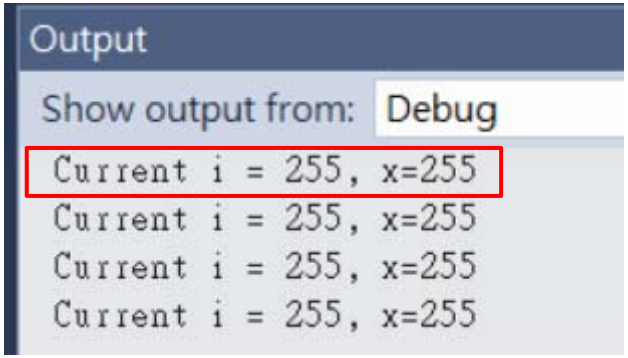
© 2008 Microchip Technology Incorporated. All Rights Reserved.

Debug Windows

Breakpoint : Conditions

- Break Point will change to  and Action debug will show in Output window.





© 2008 Microchip Technology Incorporated. All Rights Reserved.

Debug Windows

Watch to element of array though a pointer

- Add a pointer to Watch window will only shows first element of array, please use Watch format for whole array.
  Ex. *(uint8_t(*)[100])px

```
32  uint8_t i = 0;
33  uint8_t x[100];
34
35  int main (void)
36  {
37      uint8_t *px;
38
39      system_init();
40
41      px=x;
42
43      while(1)
44      {
45          for ( i = 0; i < 100 ; i++)
46          {
47              px[i] = i;
48          }
49      }
50  }
```

Watch 1

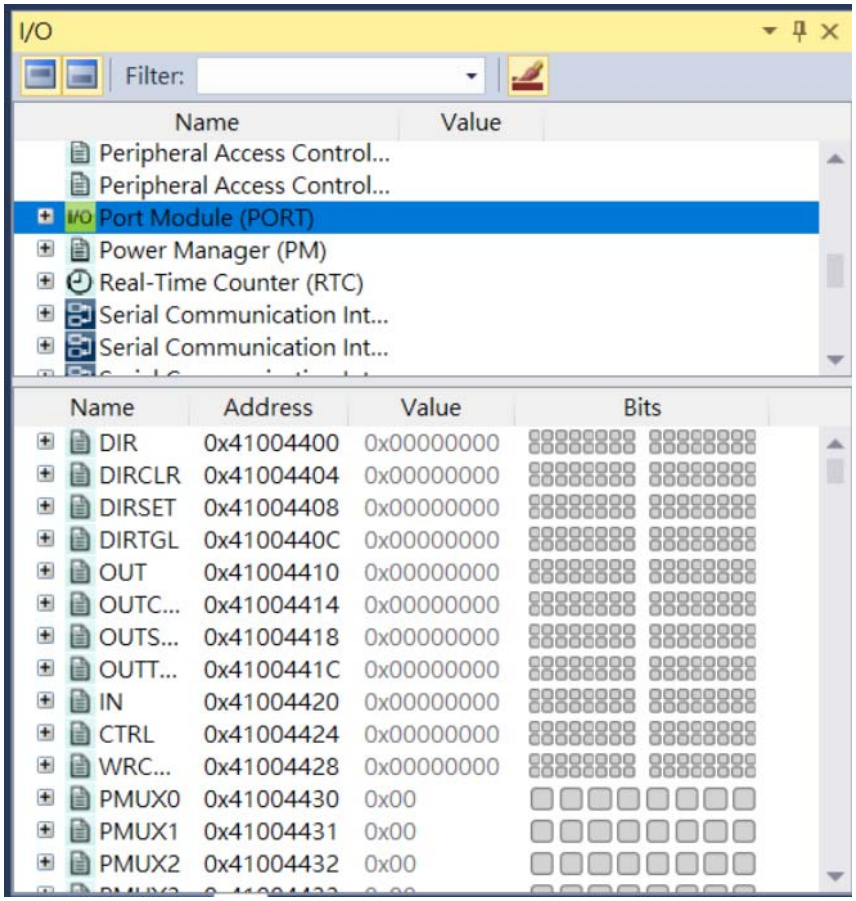
Name	Value	Type
x	[100]	uint8_t [100]
px	0x20000044 <x> ""	uint8_t *
*px	0	uint8_t
*(uint8_t (*)[100])px	[100]	uint8_t [100]
0	0	uint8_t
1	1	uint8_t
2	2	uint8_t
3	3	uint8_t
4	4	uint8_t
5	5	uint8_t

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Debug Windows

I/O view

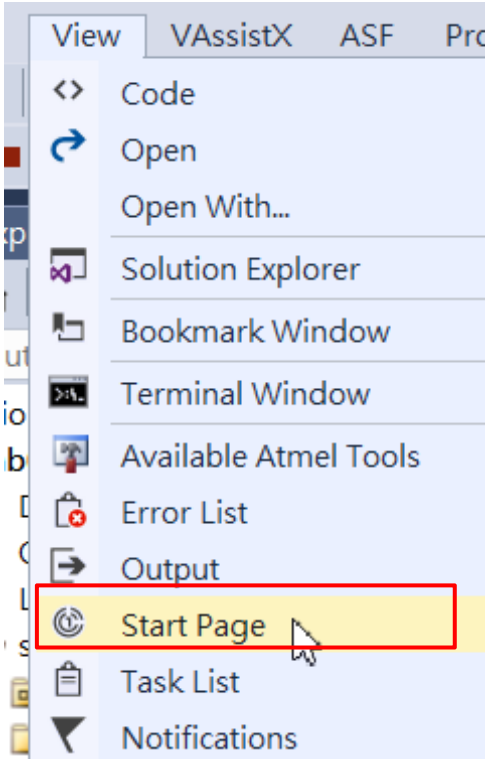
- Select **Debug ▸ Windows ▸ I/O**

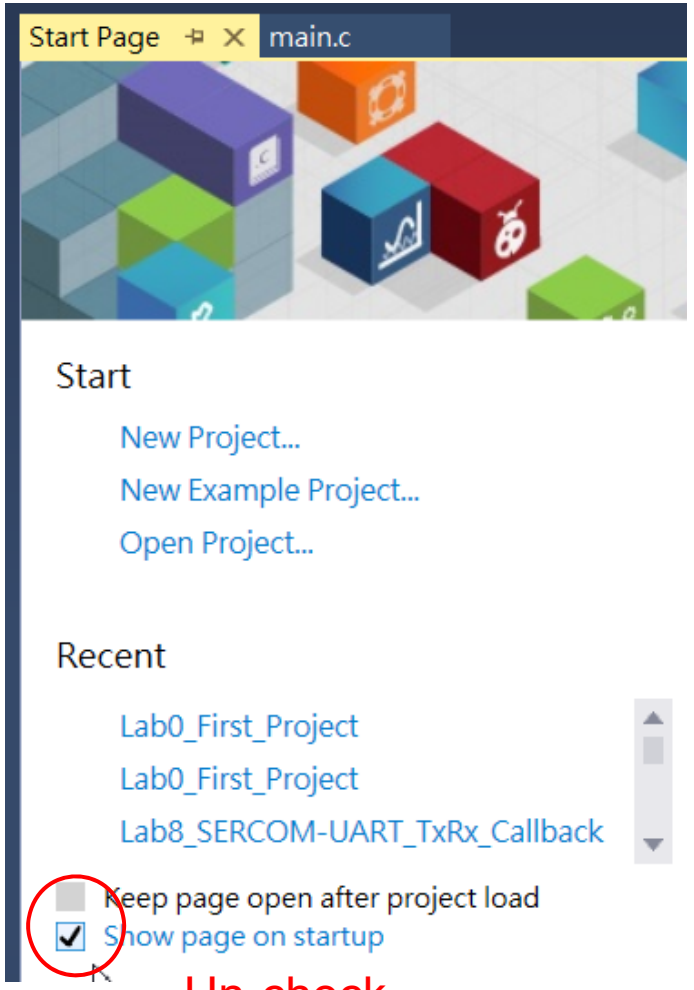


© 2008 Microchip Technology Incorporated. All Rights Reserved.

Studio 7 tips-1

- Disable Start Page of Studio 7.
- Select **View ▸ Start Page**





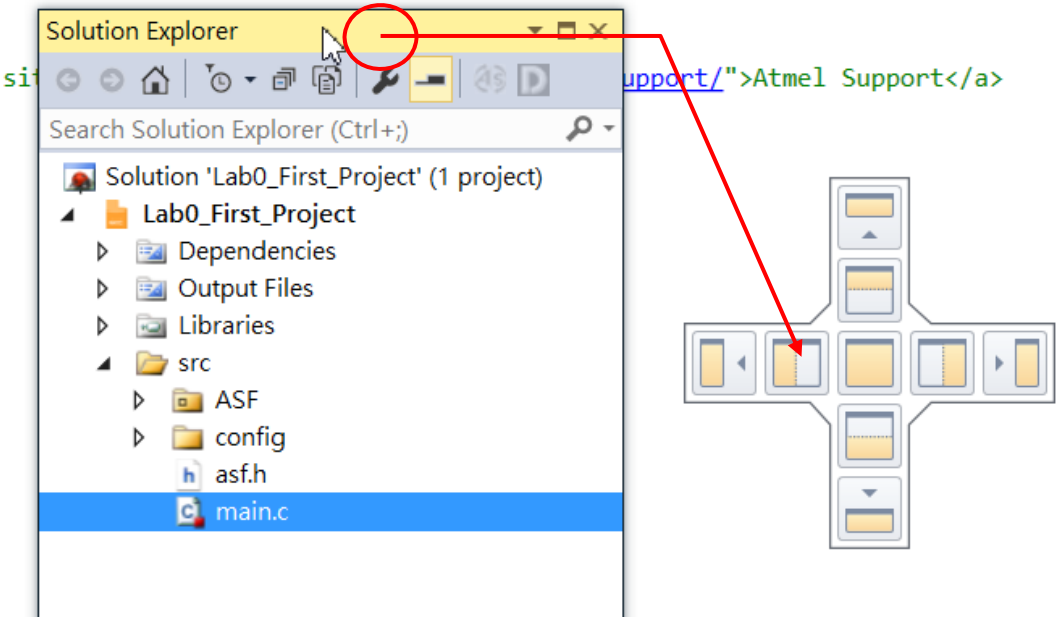
Un-check

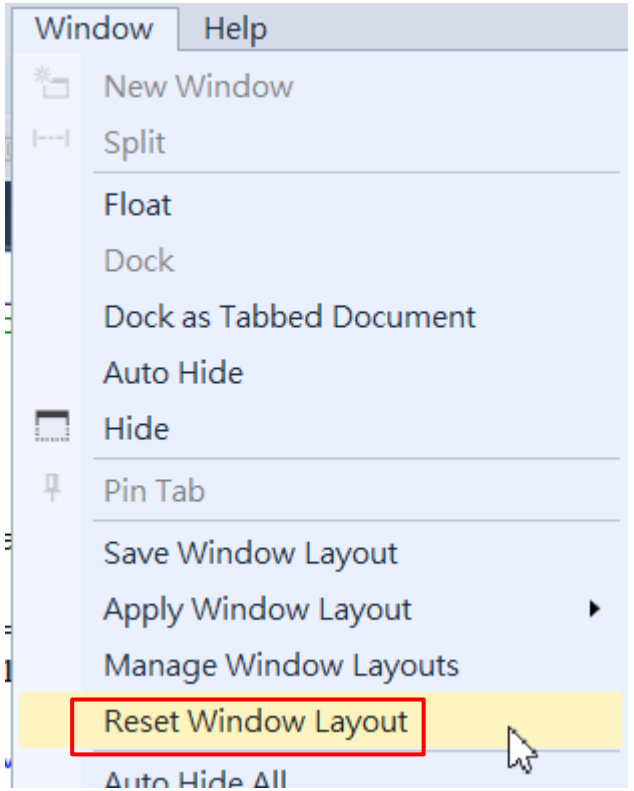
© 2008 Microchip Technology Incorporated. All Rights Reserved.

Studio 7 tips-2

- Customize Window dock
- Restore to default, Select **Window ▸ Reset Window Layout**

Click hold and drag-drop

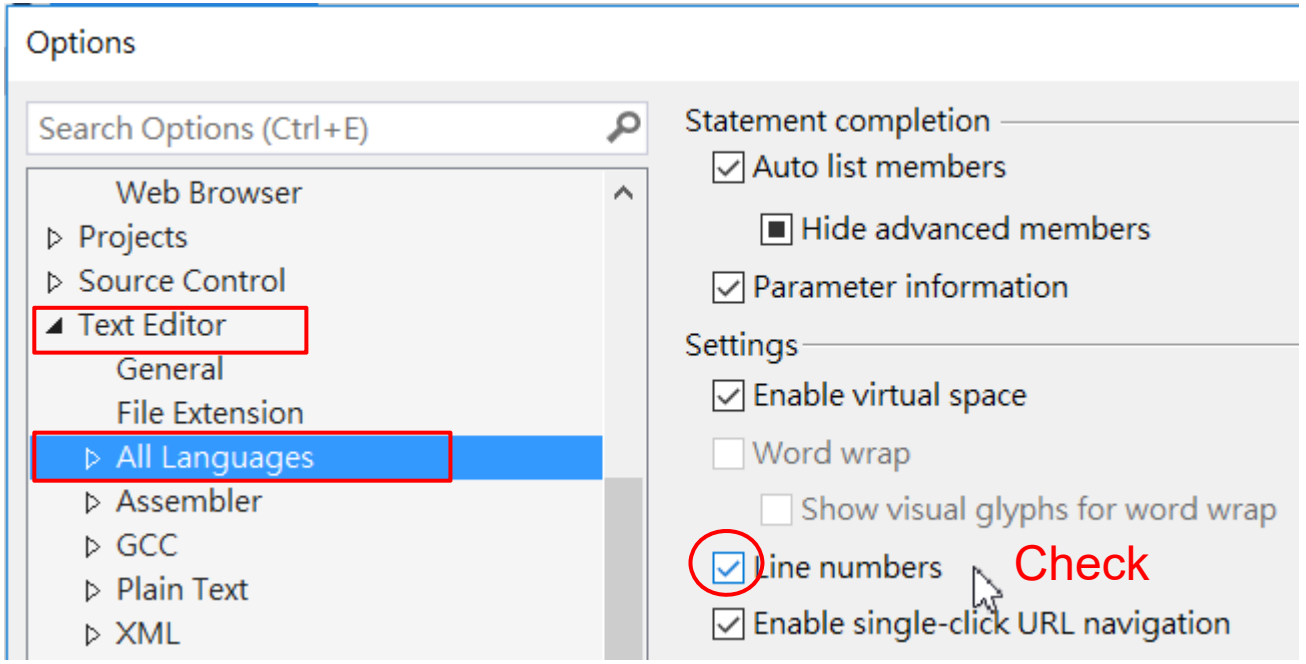


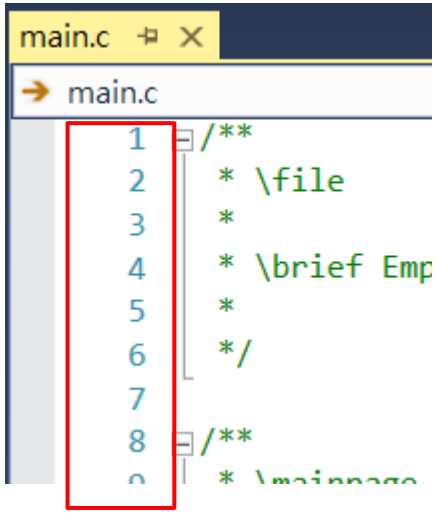


© 2008 Microchip Technology Incorporated. All Rights Reserved.

Studio 7 tips-3

- Enable Line number
- Select **Tools ▸ Options**

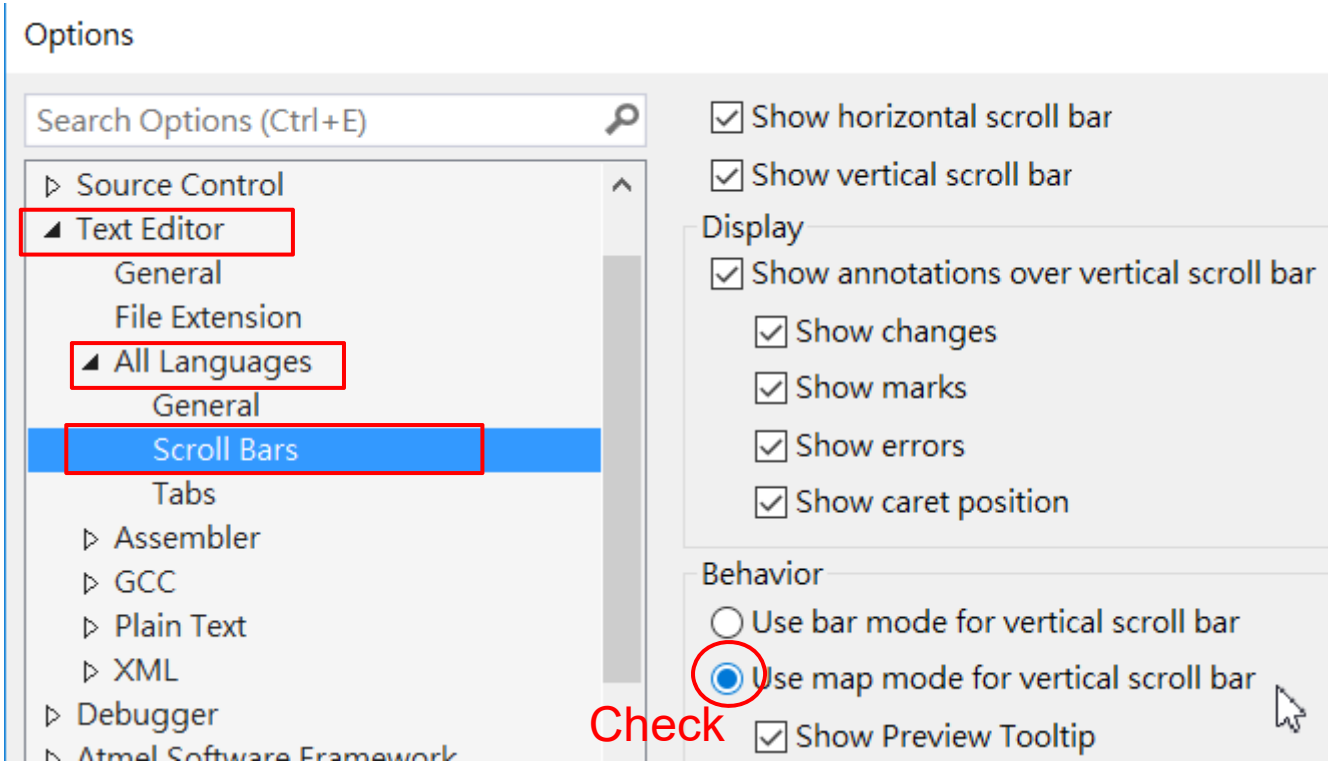


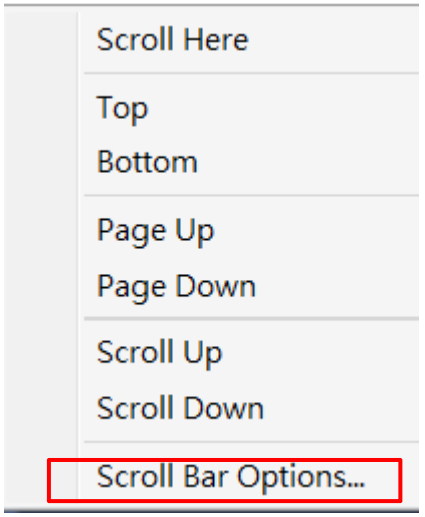


© 2008 Microchip Technology Incorporated. All Rights Reserved.

Studio 7 tips-4

- Enable MAP mode of vertical scroll bar
- Select **Tools ▸ Options** **or** **Right click of scroll bar**

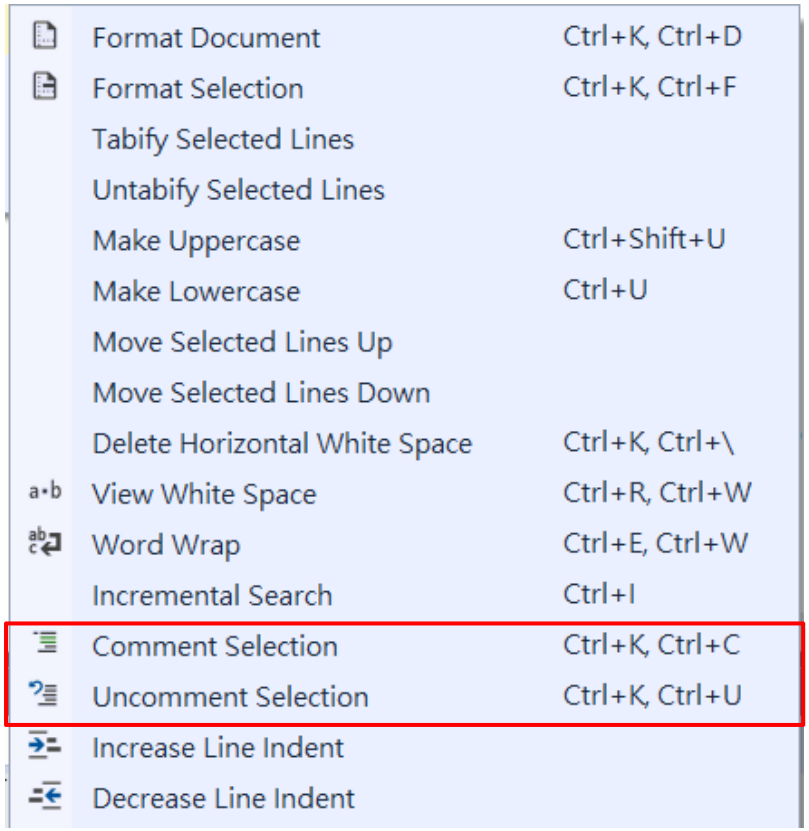




© 2008 Microchip Technology Incorporated. All Rights Reserved.

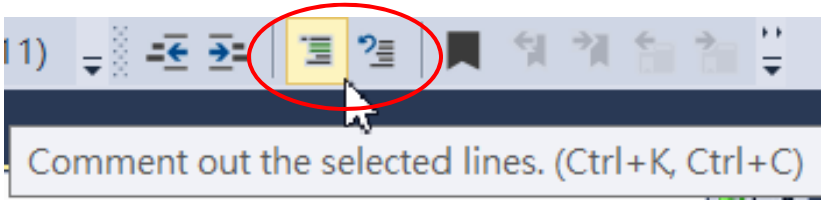
Studio 7 tips-5

- Comment/Uncomment selected code segment.
- Select **Edit ▸ Advanced**



- Hotkey
  - Comment : Ctrl-K ▸ Ctrl-C
  - Uncomment : Ctrl-K ▸ Ctrl-U

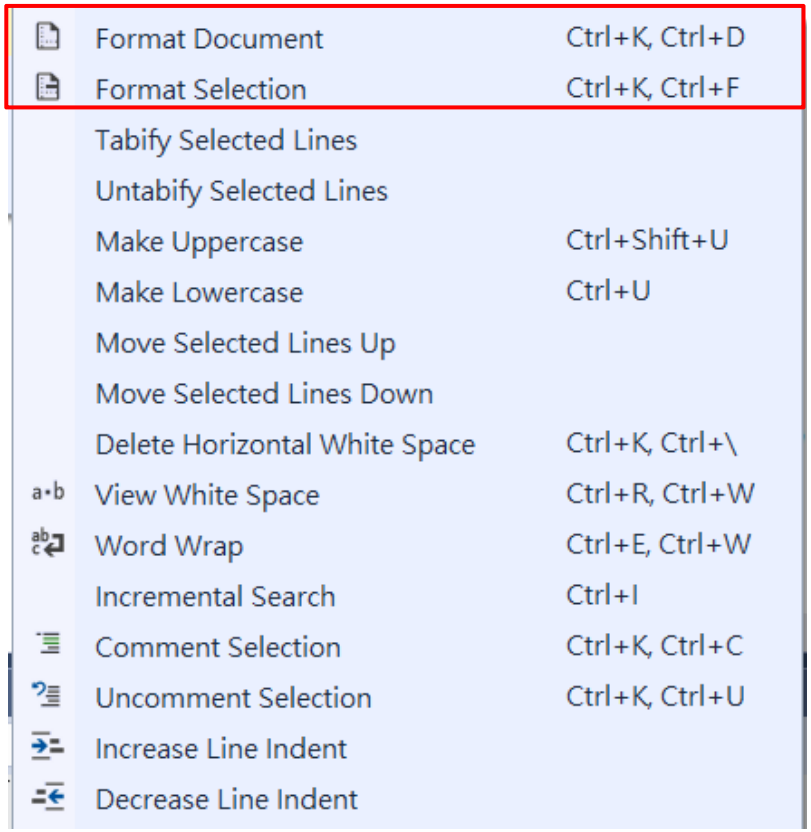
- Tool Bar



© 2008 Microchip Technology Incorporated. All Rights Reserved.

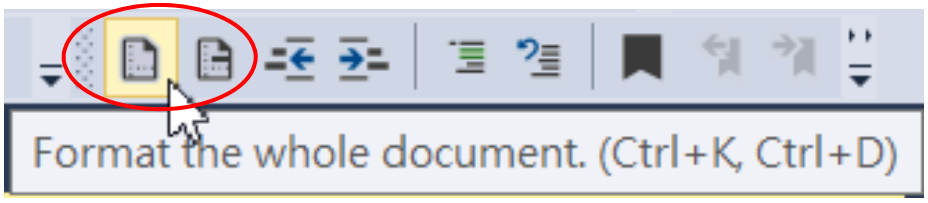
Studio 7 tips-6

- Format Document/Selected code segment.
- Select **Edit ▸ Advanced**



- Hotkey
  - Whole Document : Ctrl-K ▸ Ctrl-D
  - Selection : Ctrl-K ▸ Ctrl-F

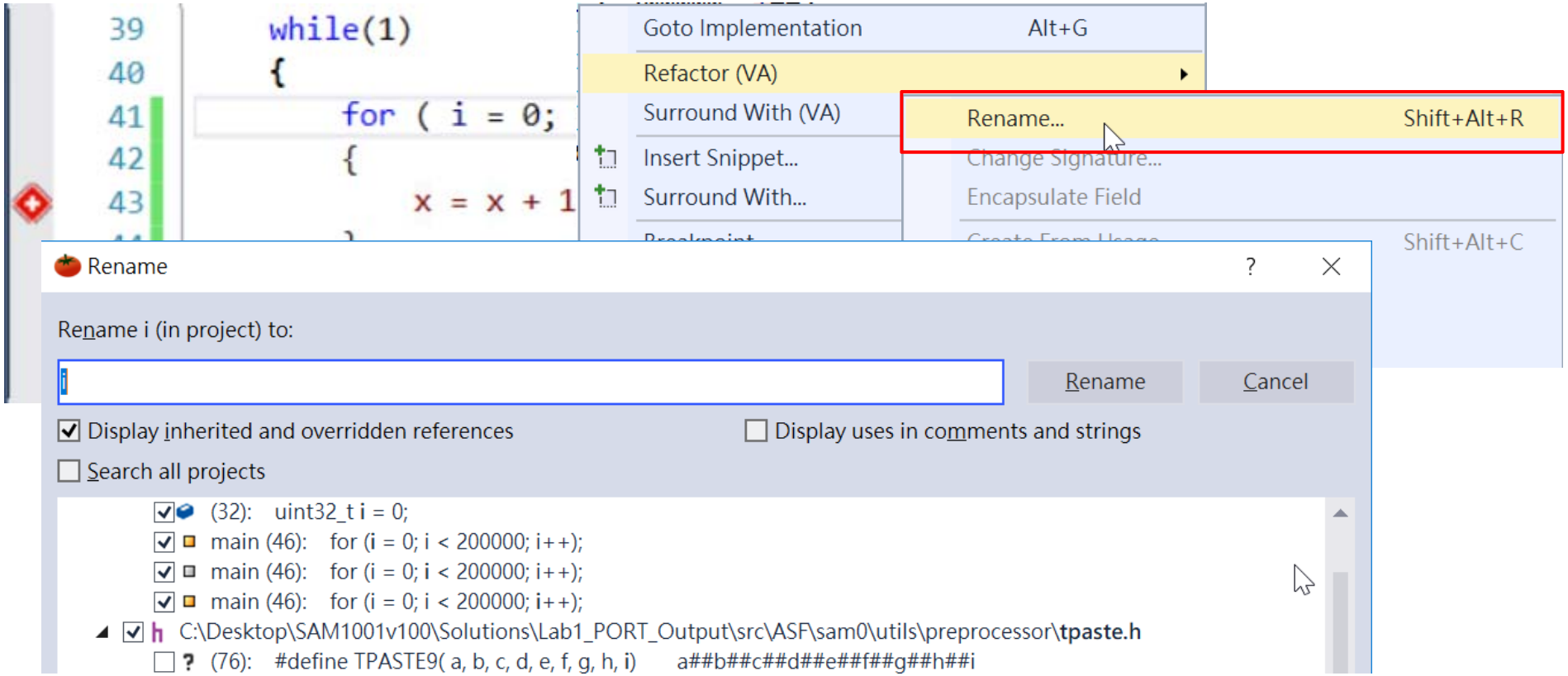
- Tool Bar (Need customized add)



© 2008 Microchip Technology Incorporated. All Rights Reserved.

Studio 7 tips-7

- Visual Assist - Rename.
- Right Click variable and select to **Refactor (VA) ▸ Rename**



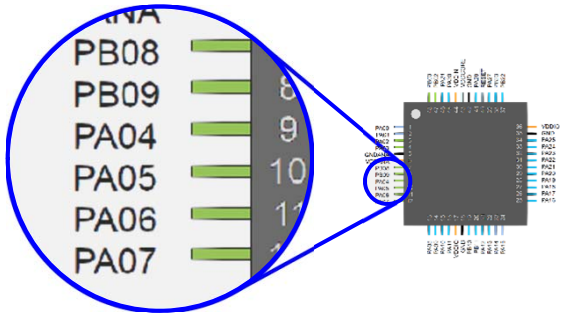
© 2008 Microchip Technology Incorporated. All Rights Reserved.

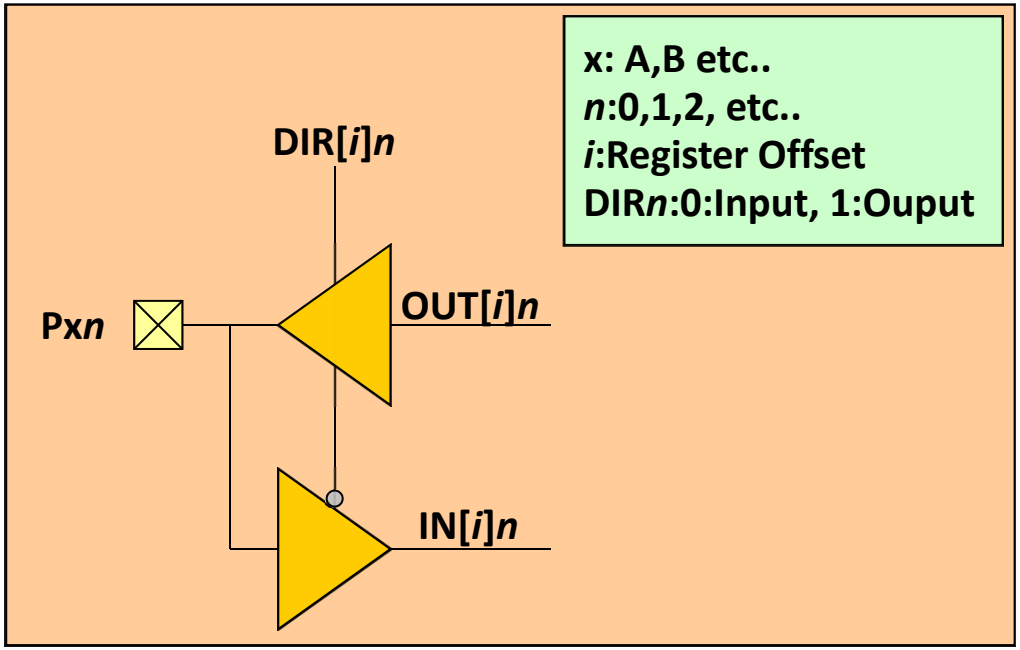
PORT I/O Architecture

© 2008 Microchip Technology Incorporated. All Rights Reserved.

PORT Block Diagram

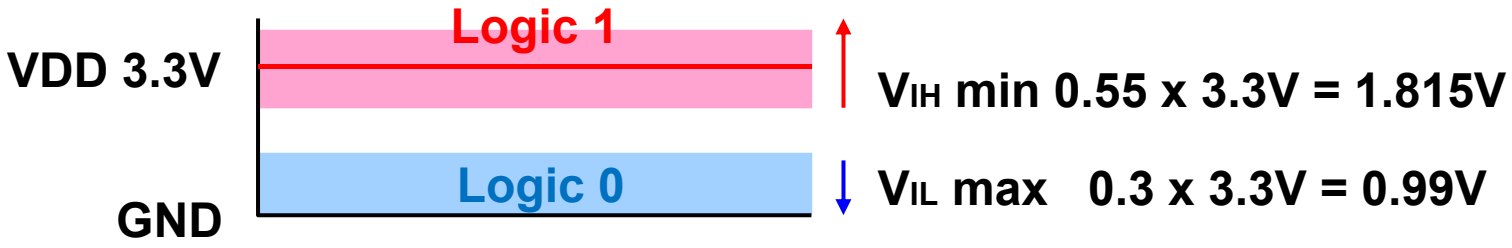
- Please refer to the block diagram, this is the concept for programming model. The real and detail block diagram please refer to the following slide.
- DIR: use to determine in or out. 0 for input, 1 for output.
- OUT: set output drive value for Output pins.
- IN: reaction logical level from Input pins.





© 2008 Microchip Technology Incorporated. All Rights Reserved.

PORT Block Diagram



37.8.1 Normal I/O Pins

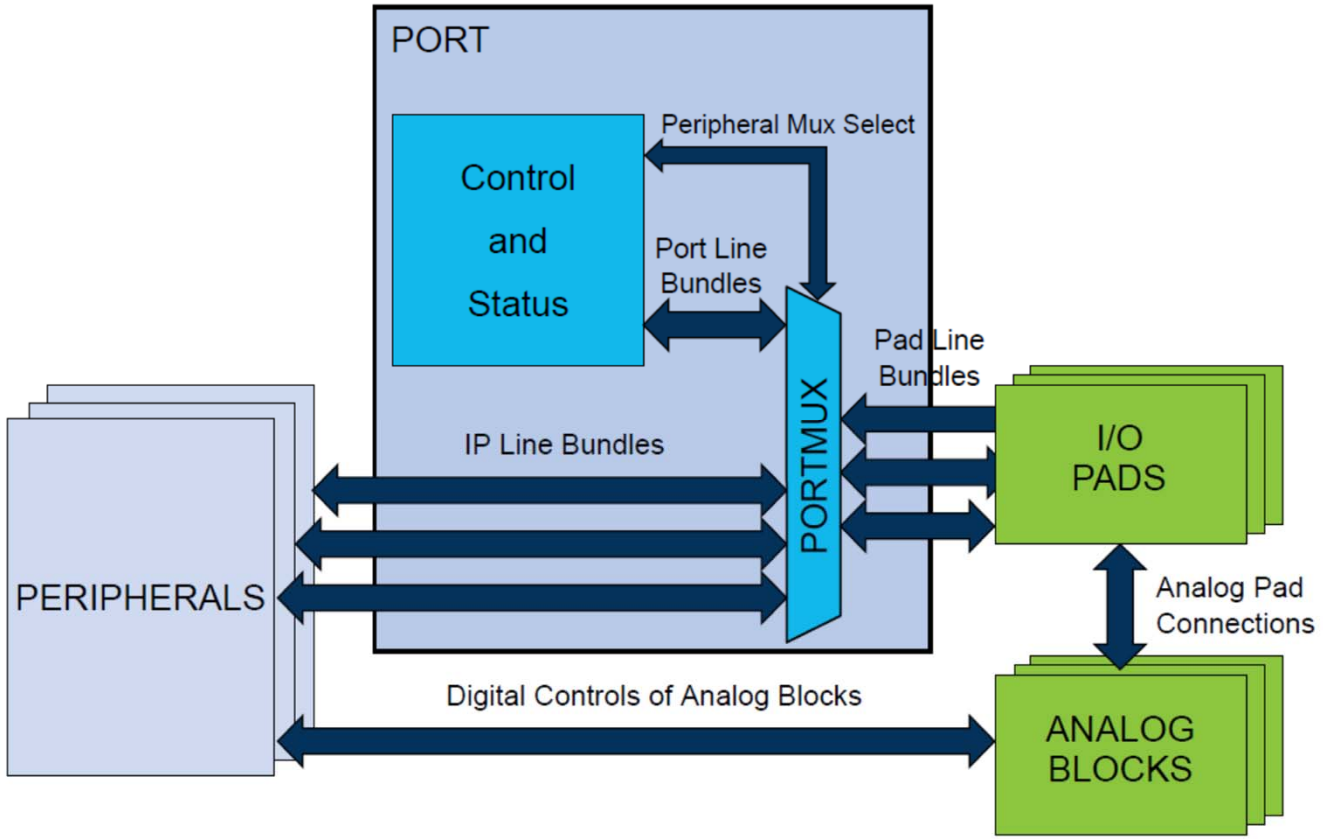
Table 37-14. Normal I/O Pins Characteristics

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
R_{PULL}	Pull-up - Pull-down resistance	All pins except for PA24 and PA25	20	40	60	kΩ
V_{IL}	Input low-level voltage	V_{DD}=1.62V-2.7V	-	-	0.25*V_{DD}	V
		V_{DD}=2.7V-3.63V	-	-	0.3*V_{DD}	
V_{IH}	Input high-level voltage	V_{DD}=1.62V-2.7V	0.7*V_{DD}	-	-	
		V_{DD}=2.7V-3.63V	0.55*V_{DD}	-	-	
V_{OL}	Output low-level voltage	V_{DD}>1.6V, I_{OL} maxI	-	0.1*V_{DD}	0.2*V_{DD}	
V_{OH}	Output high-level voltage	V_{DD}>1.6V, I_{OH} maxII	0.8*V_{DD}	0.9*V_{DD}	-	

© 2008 Microchip Technology Incorporated. All Rights Reserved.

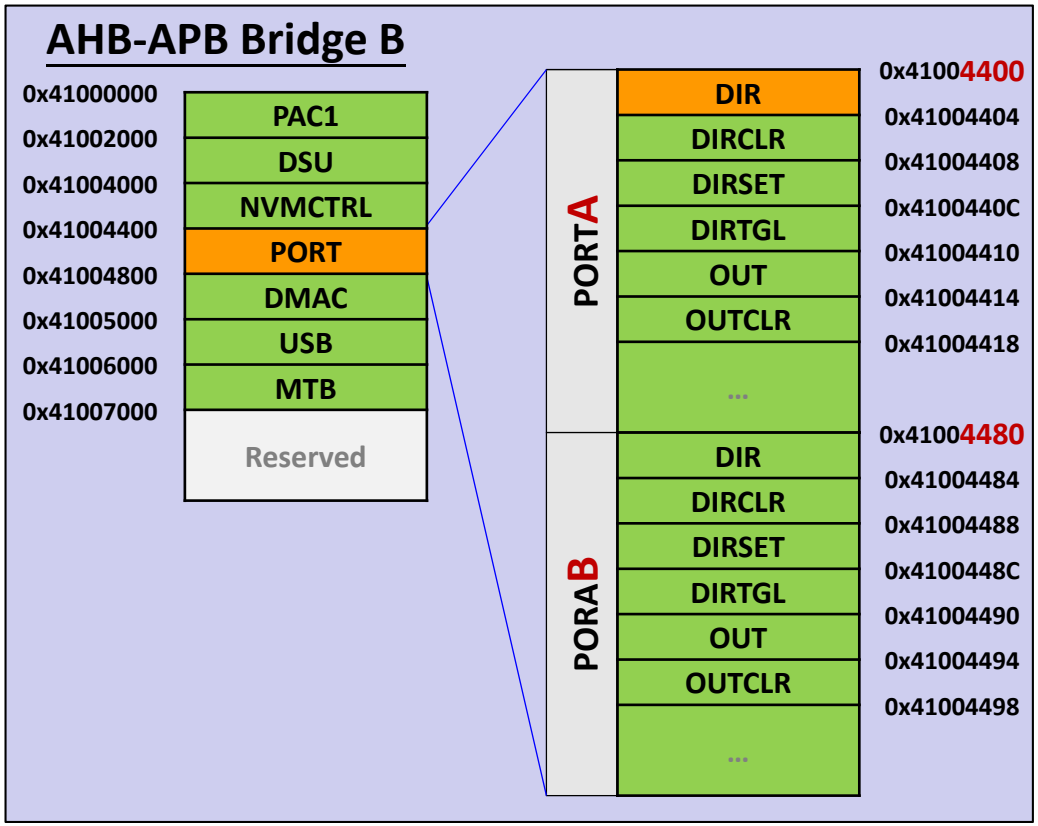
PORT Block Diagram

- **SAMD21G18A PORT Block Diagram**



© 2008 Microchip Technology Incorporated. All Rights Reserved.

Register Access



Register Summary from datasheet

Offset	Name	Bit Pos.
0x00	DIR	7:0
0x01		15:8
0x02		23:16
0x03		31:24
0x04	DIRCLR	7:0
0x05		15:8
0x06		23:16
0x07		31:24
0x08	DIRSET	7:0
0x09		15:8
0x0A		23:16
0x0B		31:24
0x0C	DIRTGL	7:0
0x0D		15:8
0x0E		23:16
0x0F		31:24
0x10	OUT	7:0
0x11		15:8
0x12		23:16
0x13		31:24
0x14	OUTCLR	7:0
0x15		15:8
0x16		23:16
0x17		31:24

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Register Access

- There are two way to access Register:
  - Access by Pointer

```
*(RwReg *)(0x41004400U) = 0x55; /* 0x41004400U : PORTA DIR */
*(RwReg *)(0x41004480U) = 0x55; /* 0x41004480U : PORTB DIR */
```

  - Access by Pre-defined struct

```
PORT->Group[0].DIR.bit.DIR = 0x55; /* Group[0] : PORTA Register Set */
PORT->Group[1].DIR.bit.DIR = 0x55; /* Group[1] : PORTB Register Set */
```

samd21g18a.h

```
#define PORT ( (Port *) 0x41004400UL )
```

port.h

```
typedef struct
{
   PortGroup Group[2];
}Port;
```

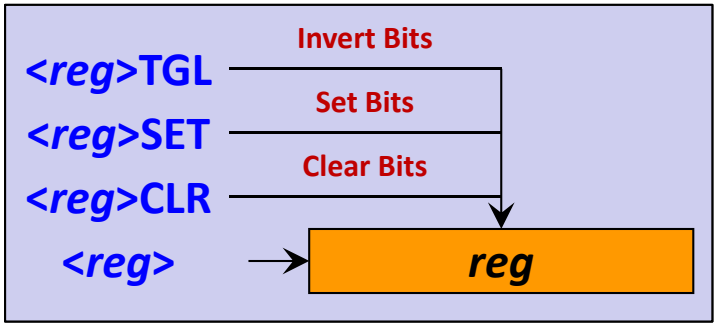
port.h

```
typedef struct
{
   __IO PORT_DIR_Type DIR;
   __IO PORT_DIRCLR_Type DIRCLR;
   __IO PORT_DIRSET_Type DIRSET;
   __IO PORT_DIRTGL_Type DIRTGL;
   ...
} PortGroup;
```

© 2008 Microchip Technology Incorporated. All Rights Reserved.

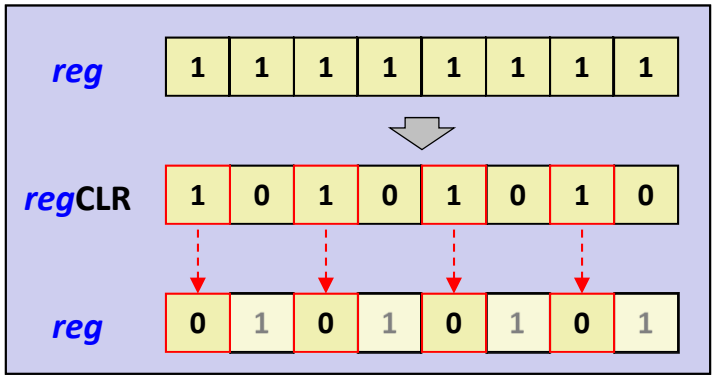
Atomic Bits Manipulation

- **Some register provide <*reg*>CLR, <*reg*>SET , <*reg*>TGL register for atomic bits manipulation.**
- **This register manipulated without doing a read-modify-write operation by using below**
  - <*reg*> Toggle (<*reg*>TGL)
  - <*reg*> Clear (<*reg*>CLR)
  - <*reg*> Set (<*reg*>SET)
- **Setting the manipulation register bits corresponding to the base register will set the base bits to 1, clear to 0 or toggle.**

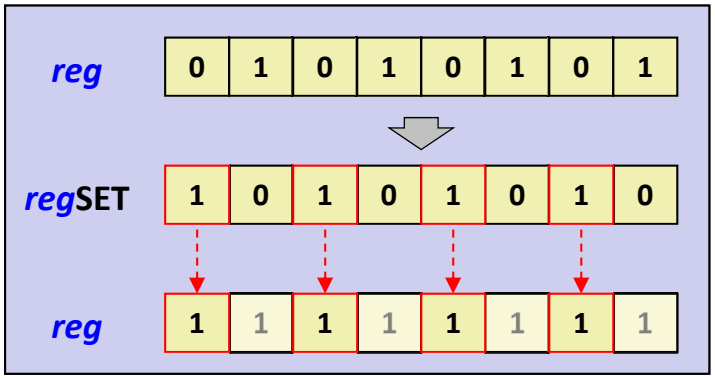


© 2008 Microchip Technology Incorporated. All Rights Reserved.

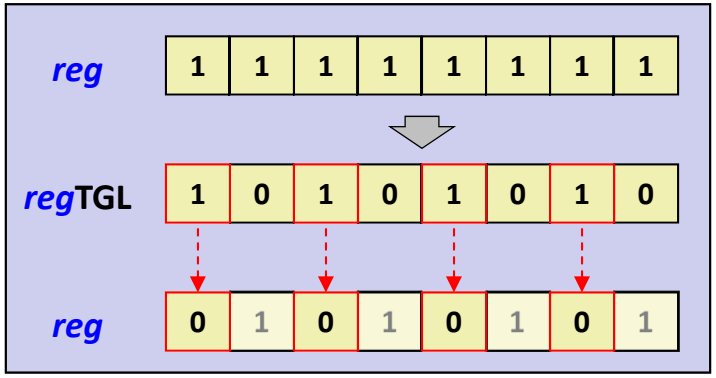
Atomic Bits Manipulation



*reg*CLR : corresponding bits clear



*reg*SET : corresponding bits set



*reg*TGL : corresponding bits toggle

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab1 PORT Output

- Try to control LED1(PA27) continues toggle (period around 500ms ~ 1S).
- You can use register pointer or pre-defined struct to access PORT.

- How to start ?

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab1 PORT Output

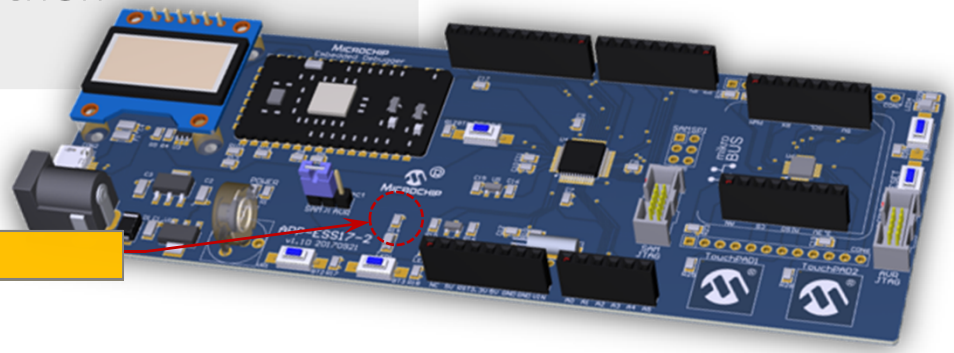
Step

a **Add below code segment to your main loop**

```
uint32_t i = 0;
int main (void)
{
   system_init();
   *(RwReg *)(0x41004400U + 0x08U) = 0x08000000;

   while(1)
   {
      for ( i = 0; i < 200000 ; i++);
      *(RwReg *)(0x41004400U + 0x1CU) = 0x08000000;
   }
}
```

Period Control

b **Program firmware to target board then observe result.**

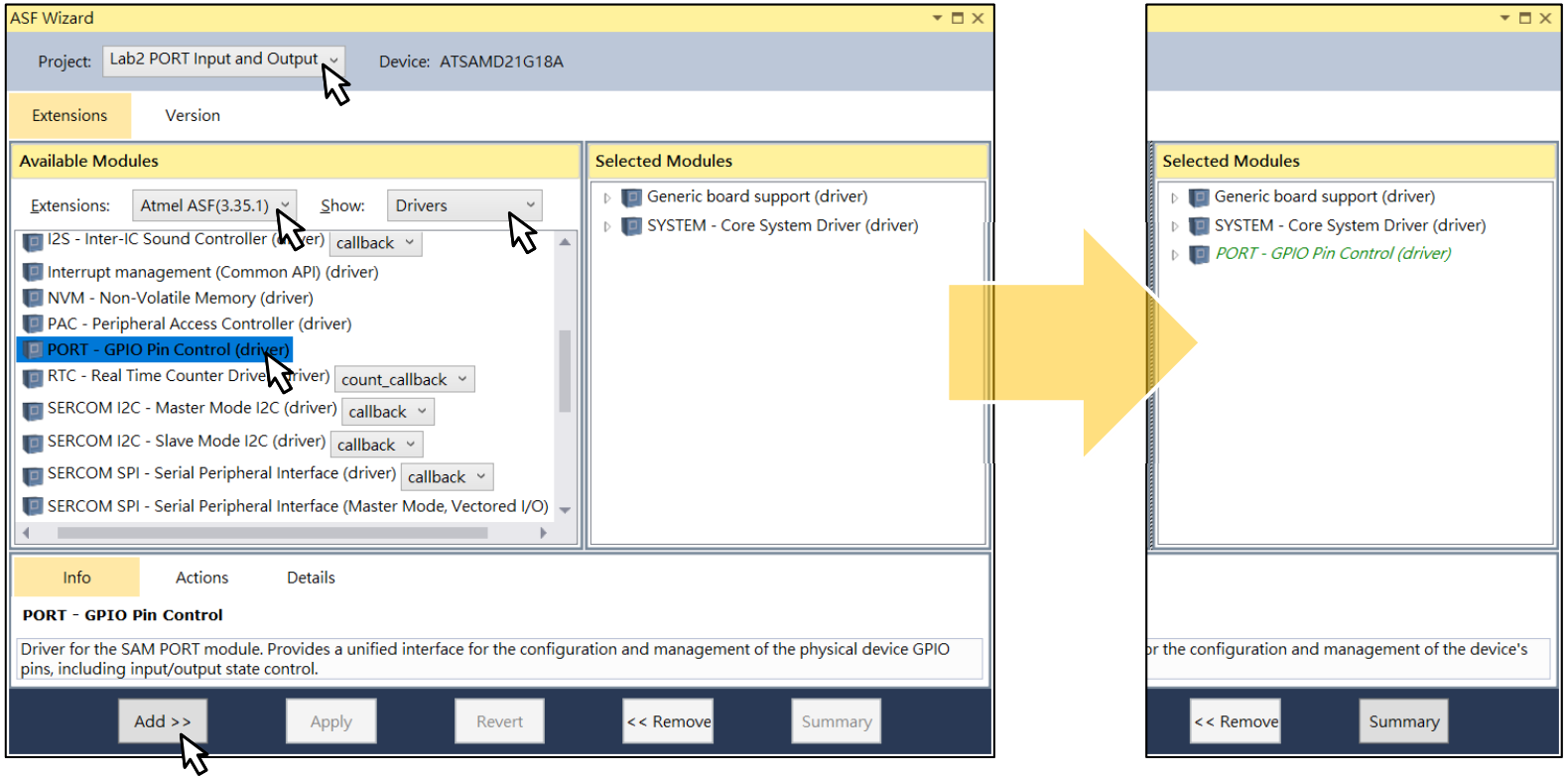


LED1 Toggle !

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Add PORT Function from ASF

- ASF provide more advanced function. Easy to use and understanding. Let me try to use ASF to my project.
- Select **ASF ▸ ASF Wizard**, Add **PORT – GPIO …** to project.



© 2008 Microchip Technology Incorporated. All Rights Reserved.

ASF PORT Driver Functions

- /* Initializes a Port pin configuration structure to defaults.*/
  void port_get_config_defaults(struct port_config *const config)

- /* Writes a Port pin configuration to the hardware module. */
  void port_pin_set_config
  (const uint8_t gpio_pin,const struct port_config *const config)

- /* Sets the state of a port pin that is configured as an output. */
  void port_pin_set_output_level
  (const uint8_t gpio_pin,const bool level)

- /* Toggles the state of a port pin that is configured as an output. */
  void port_pin_toggle_output_level(const uint8_t gpio_pin)

- /* Retrieves the state of a port pin that is configured as an input. */
  bool port_pin_get_input_level(const uint8_t gpio_pin)

© 2008 Microchip Technology Incorporated. All Rights Reserved.

ASF PORT Code Example

PORT Output Example

```
uint32_t i = 0;

struct port_config PORT_Config;
port_get_config_defaults(&PORT_Config);
PORT_Config.direction = PORT_PIN_DIR_OUTPUT;
port_pin_set_config(PIN_PA27, &PORT_Config);

while(1)
{
   for ( i = 0; i < 200000 ; i++);
   port_pin_toggle_output_level(PIN_PA27);
}
```

PORT Input & Output Example

```
struct port_config PORT_Config;
port_get_config_defaults(&PORT_Config);
PORT_Config.direction = PORT_PIN_DIR_OUTPUT;
port_pin_set_config(PIN_PA27, &PORT_Config);
PORT_Config.direction = PORT_PIN_DIR_INPUT;
port_pin_set_config(PIN_PA14, &PORT_Config);

while(1)
{
   port_pin_set_output_level
       (PIN_PA20, port_pin_get_input_level(PIN_PA14));
}
```

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab2 PORT Input and Output

- Base On Lab1, Try to add LED2(PB03) continues toggle (Period around 500ms ~ 1S).
- Get BT1(PA14) status then use to control LED3(PA17).
  Pressed -> LED3 Light
  Released -> LED3 Dark

- Let's go!

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab2 PORT Input and Output

Step

a. **Add below code segment to your main loop**

```
uint32_t i = 0;
struct port_config PORT_Config;
int main (void)
{
    system_init();
    port_get_config_defaults(&PORT_Config);
    PORT_Config.direction = PORT_PIN_DIR_OUTPUT;
    port_pin_set_config(LED1, &PORT_Config);
    ...
    PORT_Config.direction = PORT_PIN_DIR_INPUT;
    port_pin_set_config(BT1, &PORT_Config);
    ...
    while(1)
    {
        if (i++ > 200000)
        {
            port_pin_toggle_output_level(LED1);
            ...
        }
        port_pin_set_output_level(LED1, port_pin_get_input_level(BT1));
    }
}
```

b. **Program firmware to target board then observe result.**

© 2008 Microchip Technology Incorporated. All Rights Reserved.

TC Architecture

- TC : Timer/Counter

© 2008 Microchip Technology Incorporated. All Rights Reserved.

SAMD21 Timer/Counter

- The SAMD21 Timer/Counter, TC module consists 3 function Counter, Compare and Capture.
- The counter can be set to count events, or it can be configured to count clock pulses. The counter, together with the compare/capture channels, can be configured to timestamp input events, allowing capture of frequency and pulse width.
- It can also perform waveform generation, such as frequency generation and pulse-width modulation (PWM).
- TC/TCC naming rule
  - TCC0, TCC1, TCC2 and TC3, TC4, TC5, TC6, TC7
  - TC6 and TC7 are not supported on the SAM D21E and G devices

© 2008 Microchip Technology Incorporated. All Rights Reserved.

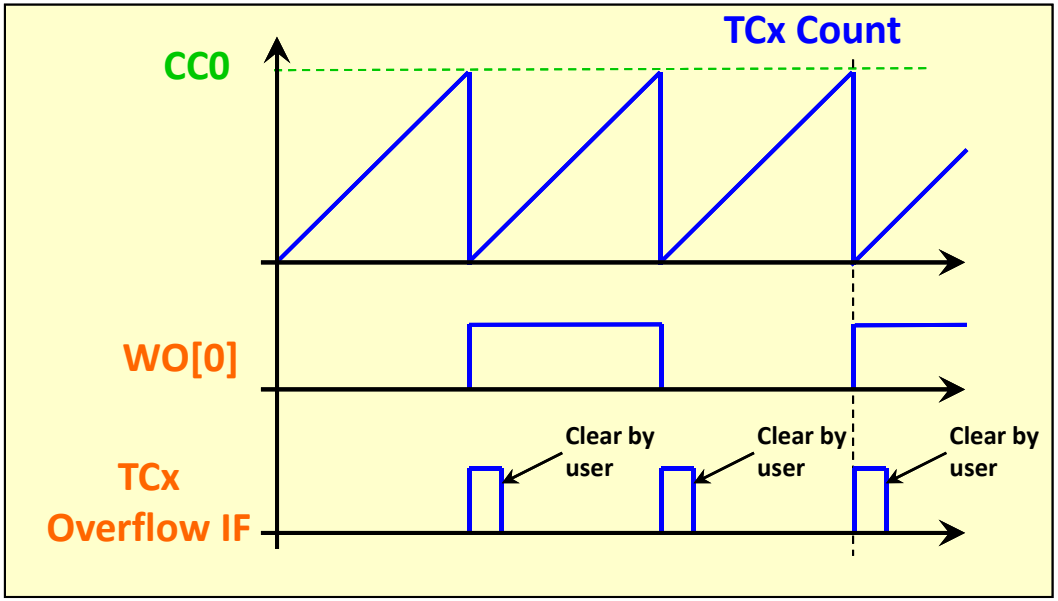
Timer/Counter

- **TC provide 4 difference for Timer, Counter and Compare operation mode**
  - **Normal Frequency Generation (NFRQ)**
    toggled on each compare match
  - **Match Frequency Generation (MFRQ)**
    toggles on each update condition
  - **Normal Pulse-Width Modulation Operation (NPWM)**
    toggles on each match & update condition
  - **Match Pulse-Width Modulation Operation (MPWM)**
    toggles on each match & update condition, Count direction reverse.

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Match Frequency Generation (MFRQ) Mode

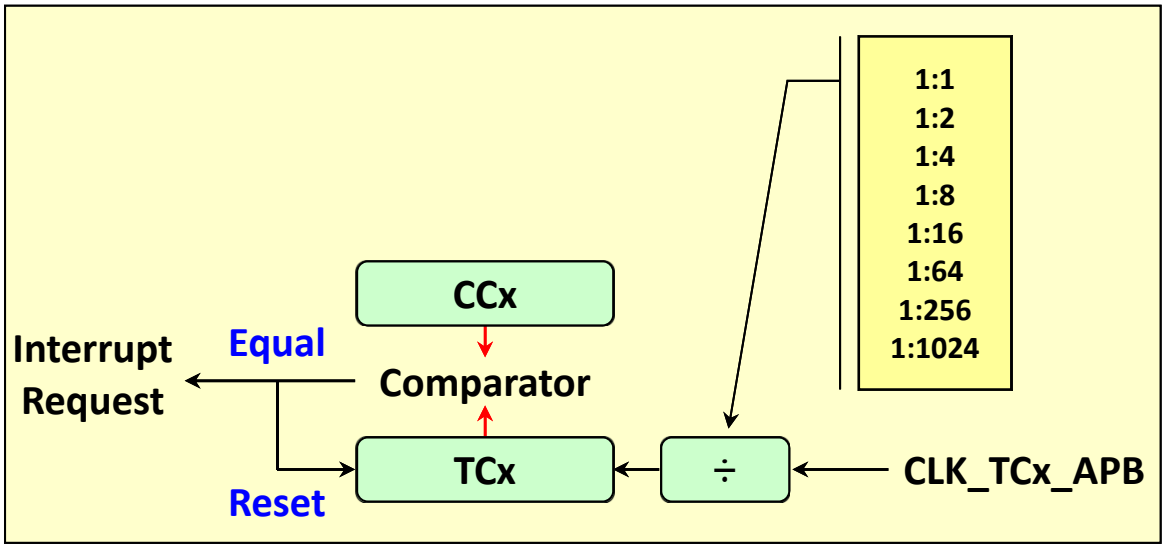
- For Match Frequency Generation, the period time (T) is controlled by the CC0 register instead of PER. WO[0] toggles on each update condition.



© 2008 Microchip Technology Incorporated. All Rights Reserved.

SAMD21 Timer/Counter

- Please refer to the block diagram, Just a concept. The real and detail block diagram please refer to the following slide.
- Clock into TCx after prescaler. Comparator continuously compares the values of TCx and CCx. Asserts a into the an interrupt request when equal, and resets TCx to zero.
- You can fill a value to CCx to determine period you want.



Match Frequency Generation (MFRQ), 16Bit, Up count

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Prescaler

- TC is 16 Bits counter , counting range from 0 to 65,535. If you need more than the count. You must adjust prescaler to expand the count range.
- For example:
  if the clock in is 0.25uS, to reach a 500mS period, the CC0 must be set to 2,000,000 (2,000,000 * 0.25uS = 500mS).
  However, this value has exceeded the acceptable range of CC0.
  At this point you can set the prescaler to reduce the count value to expand the count range.

$$\left(2^n - 1\right) \le CC0 = \frac{Peroid}{Clock \times Scale}$$

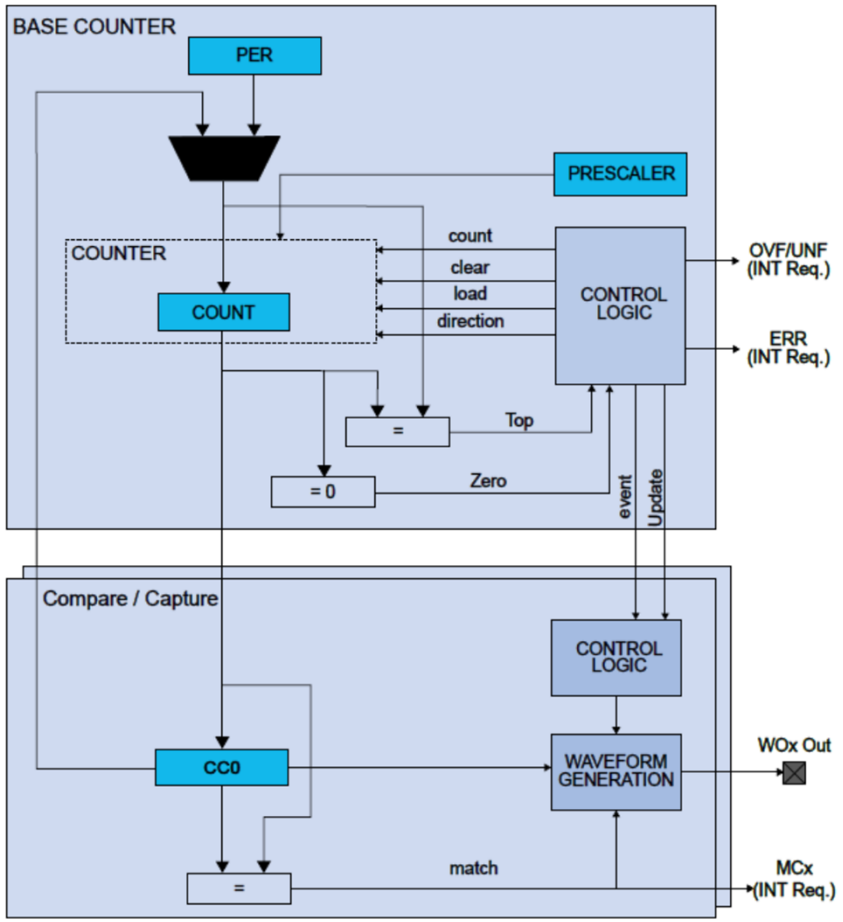
✗ $CC0 = \frac{500mS}{0.25uS \times 1}, CC0 > 65535$

✓ $CC0 = \frac{500mS}{0.25uS \times 64}, CC0 < 65535$

© 2008 Microchip Technology Incorporated. All Rights Reserved.

TC Block Diagram

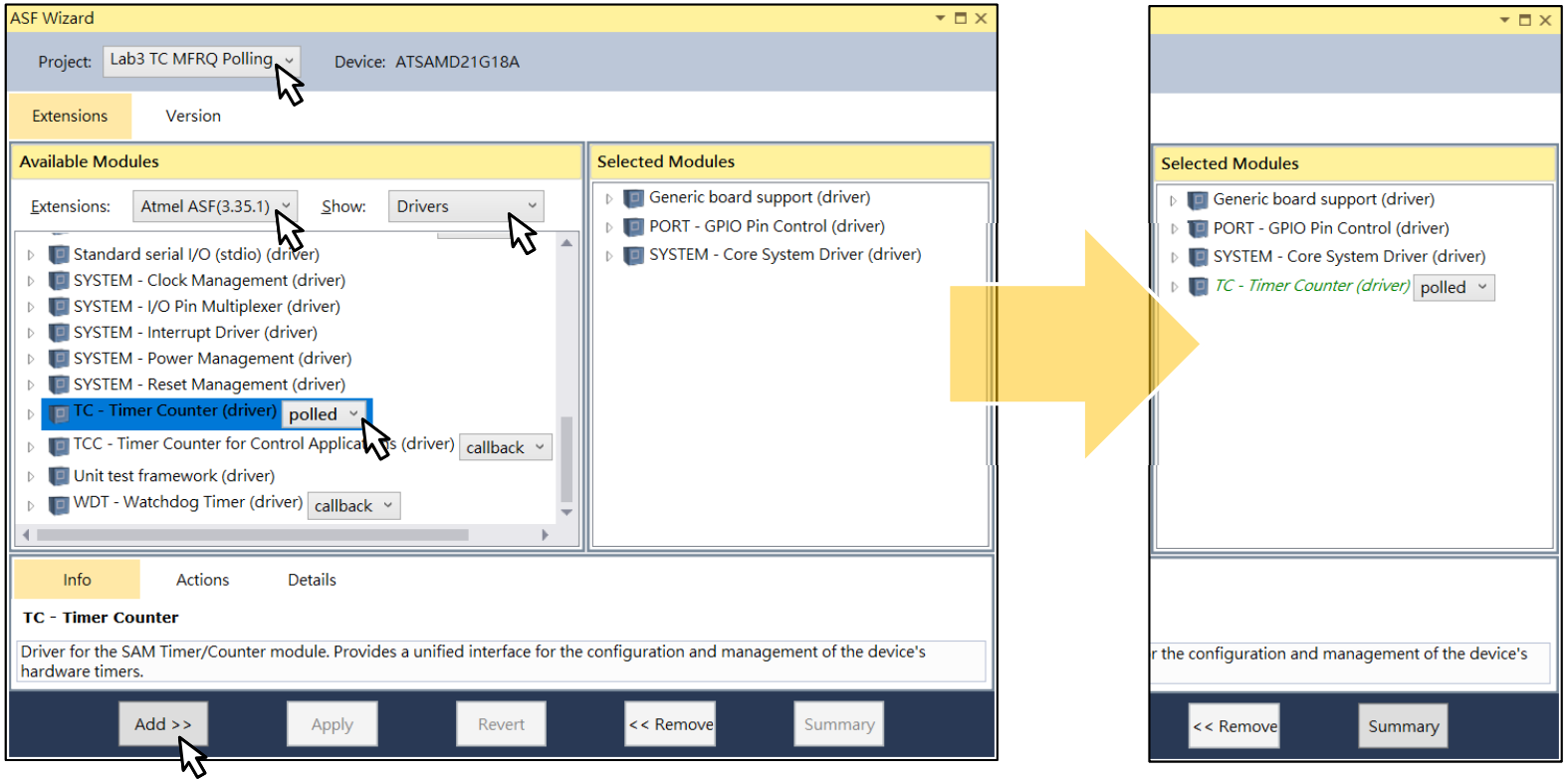
- **SAMD21G18A TC Block Diagram**



© 2008 Microchip Technology Incorporated. All Rights Reserved.

Add TC Function from ASF

- ASF provide more advanced function. Easy to use and understanding. Let me try to use ASF to my project.
- Select **ASF ▸ ASF Wizard**, Add **TC … polled** to project.



© 2008 Microchip Technology Incorporated. All Rights Reserved.

ASF TC Driver Functions

- /* Initializes config with predefined default values. */
  void tc_get_config_defaults(struct tc_config *const config)

- /* Initializes a hardware TC module instance. */
  enum status_code tc_init
  (struct tc_module *const module_inst, Tc *const hw,
  const struct tc_config *const config)

- /* Enable the TC module. */
  void tc_enable(const struct tc_module *const module_inst)

© 2008 Microchip Technology Incorporated. All Rights Reserved.

TC Interrupt Event

- **The TC has the following interrupt sources:**
  - **Overflow/Underflow (OVF)**
    Counter overflow/underflow
  - **Match or Capture Channel x (MCx)**
    Compare match or capture
  - **Capture Overflow Error (ERR)**
    New capture interrupt
    occurs while interrupt flag is set.
  - **Synchronization Ready (SYNCRDY)**
    Read/Write synchronization ready.
- **How to get and polling TC interrupt flag status ?**

```
if (REG_TCx_INTFLAG & TC_INTFLAG_OVF)
{
   REG_TCx_INTFLAG  = TC_INTFLAG_OVF;
   // ….
}
```

Or

```
if (TCx->COUNT16.INTFLAG.bit.OVF)
{
   TCx->COUNT16.INTFLAG.bit.OVF = 1;
   // ….
}
```

Or

```
if (tc_get_status(&TCx_Instance) &
                TC_STATUS_COUNT_OVERFLOW)
{
   tc_clear_status(&TCx_Instance,
   TC_STATUS_COUNT_OVERFLOW);
   // ….
}
```

© 2008 Microchip Technology Incorporated. All Rights Reserved.

ASF TC Code Example

TC MFRQ Example

```
struct tc_module TC_Instance;
struct tc_config TC_Config;

// PORT initial segment
…

tc_get_config_defaults(&TC_Config);
TC_Config.wave_generation = TC_WAVE_GENERATION_MATCH_FREQ;
TC_Config.counter_size = TC_COUNTER_SIZE_16BIT; /* 16 Bits, Up Count */
TC_Config.clock_prescaler = TC_CLOCK_PRESCALER_DIV256;
TC_Config.counter_16_bit.compare_capture_channel[0] = (8000000 / 256 / 2); /* 8M / 256 / 2 */
tc_init(&TC_Instance, TC3, &TC_Config);
tc_enable(&TC_Instance);

while(1)
{
   if (TC3->COUNT16.INTFLAG.bit.OVF)
   {
      TC3->COUNT16.INTFLAG.bit.OVF = 1;
      port_pin_toggle_output_level(LED1);
   }
}
```

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Waveform Output

- TC support waveform output through WO[x] pin depend on different mode.
- Setup PINMUX module to determine which signal connect to pin. Talk about detail at below section.
- At this moment. Please just append below code at your initial code segment. Then you can measure the signal at PA14 if you want.

```
TC_Config.pwm_channel[0].enabled = true;
TC_Config.pwm_channel[0].pin_out = PIN_PA14E_TC3_WO0;
TC_Config.pwm_channel[0].pin_mux = MUX_PA14E_TC3_WO0;
```

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab3 TC MFRQ Polling

- Try to use TC3 to replace software delay to control LED1(PA27) continues toggle.
- The TC3 setup to MFRQ, 16 bit mode, Timer period is 100mS.

- Let's go!

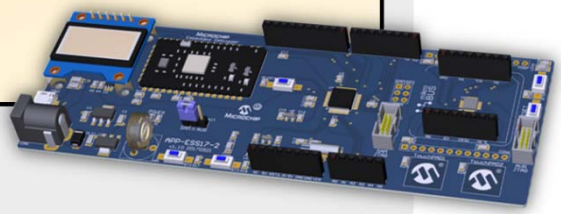
© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab3 TC MFRQ Polling

Step

a **Add below code segment to your main loop**

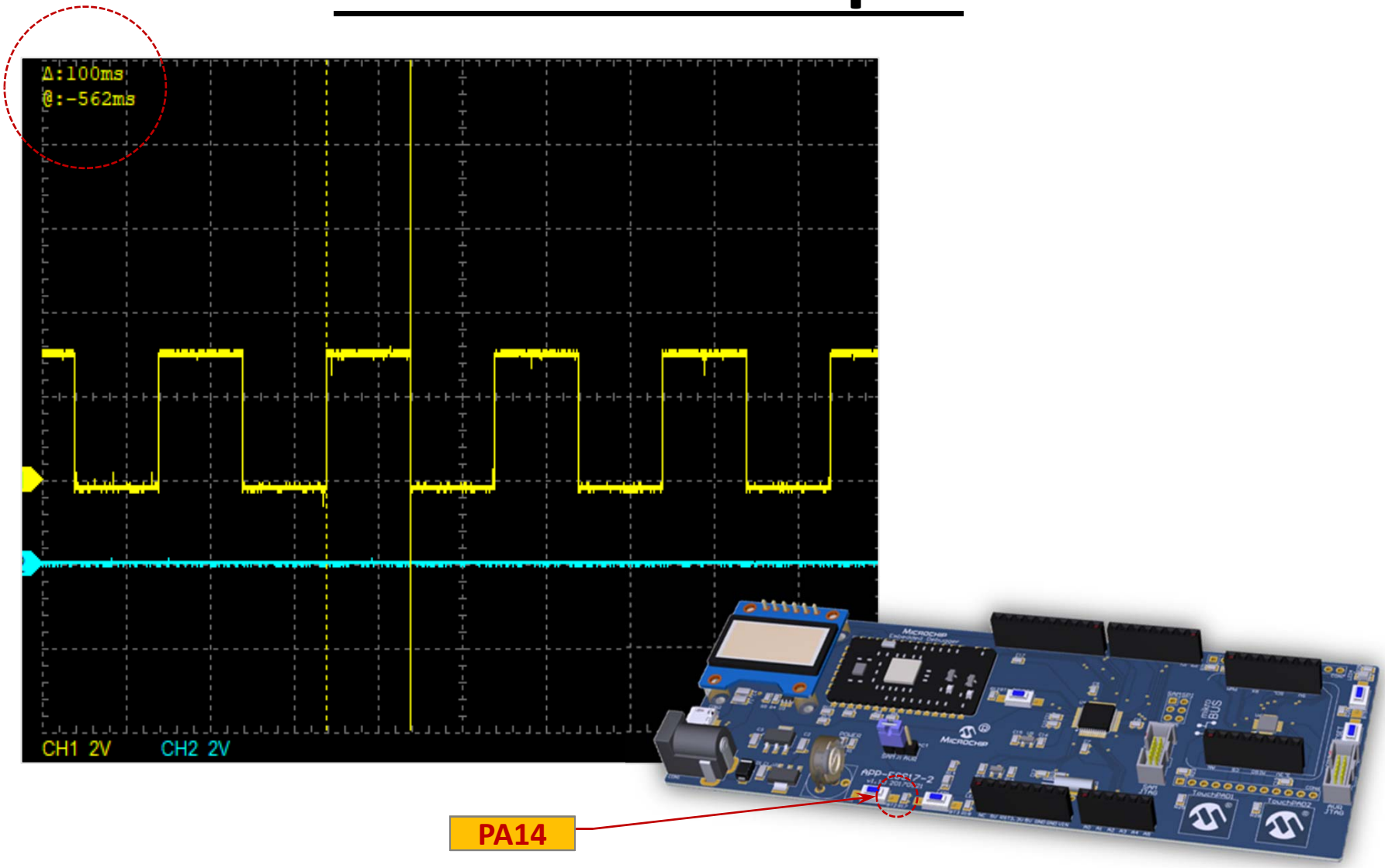
```
struct tc_module TC3_Instance;
struct tc_config TC_Config;

int main (void)
{
    ...
    tc_get_config_defaults(&TC_Config);
    TC_Config.wave_generation = TC_WAVE_GENERATION_MATCH_FREQ;
    TC_Config.counter_size = TC_COUNTER_SIZE_16BIT;
    TC_Config.clock_prescaler = TC_CLOCK_PRESCALER_DIV256;
    TC_Config.counter_16_bit.compare_capture_channel[0] = (8000000 / 256 / 10);
    tc_init(&TC3_Instance, TC3, &TC_Config);
    tc_enable(&TC3_Instance);

    while(1)
    {
        if (TC3->COUNT16.INTFLAG.bit.OVF)
        {
            TC3->COUNT16.INTFLAG.bit.OVF = 1;
            port_pin_toggle_output_level(LED1);
        }
}
```



b **Program firmware to target board then observe result.**

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Waveform Output



© 2008 Microchip Technology Incorporated. All Rights Reserved.

NVIC Architecture

- NVIC : Nested Vector Interrupt Controller Architecture

© 2008 Microchip Technology Incorporated. All Rights Reserved.

What's Interrupt ?

- In general, processor are executed in sequence according to the program's flow.
- However, if some peripherals or event needs immediate attention. An interrupt flag set to high-priority condition then requiring the interruption of the current program the processor is executing.
- Processor will jump to execute the interrupt service routine (ISR, Interrupt Service Routine) to service peripherals or events.

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Which events need Interrupt ?

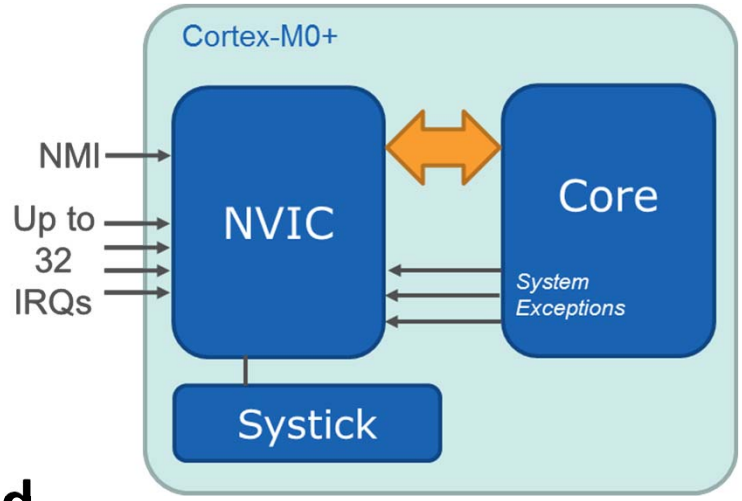
- **For General (maskable)**
  - **Time critical event**
    As soon as possible, when event occurred.
  - **Unpredicted event**
    Polling too waste computing time.

- **For Emergency (non maskable)**
  - Stack Overflow/Underflow
  - Math Error
  - Address Error
  - etc..

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Nested Vector Interrupt Controller

- Nested Vector Interrupt Controller, NVIC provides an interface between interrupt sources (peripherals and external pins) and the core.
- The priority (Level 0 ~ 3) for each interrupt source is programmable.
- Each of the interrupt lines is connected to one peripheral instance. Each peripheral can have one or more interrupt flags.



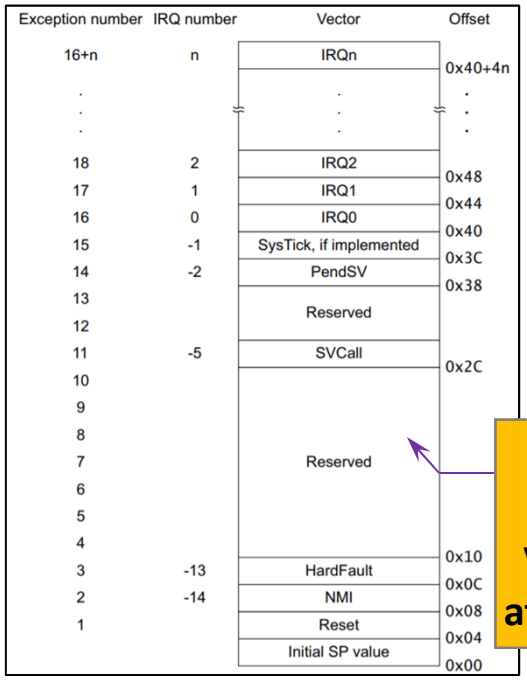
- An interrupt request is generated from the peripheral when the interrupt flag is set and the corresponding interrupt is enabled.

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Natural Priority

- If two pending interrupts share the same priority, priority is given to the interrupt with the lowest exception number

  (lowest interrupt vector address).



Each peripheral has an assigned. Vector table is based at address 0x00000000

Peripheral Source	NVIC Line
EIC NMI – External Interrupt Controller	NMI
PM – Power Manager	0
SYSCTRL – System Control	1
WDT – Watchdog Timer	2
RTC – Real Time Counter	3
EIC – External Interrupt Controller	4
NVMCTRL – Non-Volatile Memory Controller	5
DMAC - Direct Memory Access Controller	6
USB - Universal Serial Bus	7
EVSYS – Event System	8
SERCOM0 – Serial Communication Interface 0	9
SERCOM1 – Serial Communication Interface 1	10
SERCOM2 – Serial Communication Interface 2	11
SERCOM3 – Serial Communication Interface 3	12
SERCOM4 – Serial Communication Interface 4	13
SERCOM5 – Serial Communication Interface 5	14
TCC0 – Timer Counter for Control 0	15
TCC1 – Timer Counter for Control 1	16
TCC2 – Timer Counter for Control 2	17
TC3 – Timer Counter 3	18
TC4 – Timer Counter 4	19
TC5 – Timer Counter 5	20
TC6 – Timer Counter 6	21
TC7 – Timer Counter 7	22
ADC – Analog-to-Digital Converter	23
AC – Analog Comparator	24
DAC – Digital-to-Analog Converter	25
PTC – Peripheral Touch Controller	26
I2S - Inter IC Sound	27

© 2008 Microchip Technology Incorporated. All Rights Reserved.

SAMD21G18A IVT Define

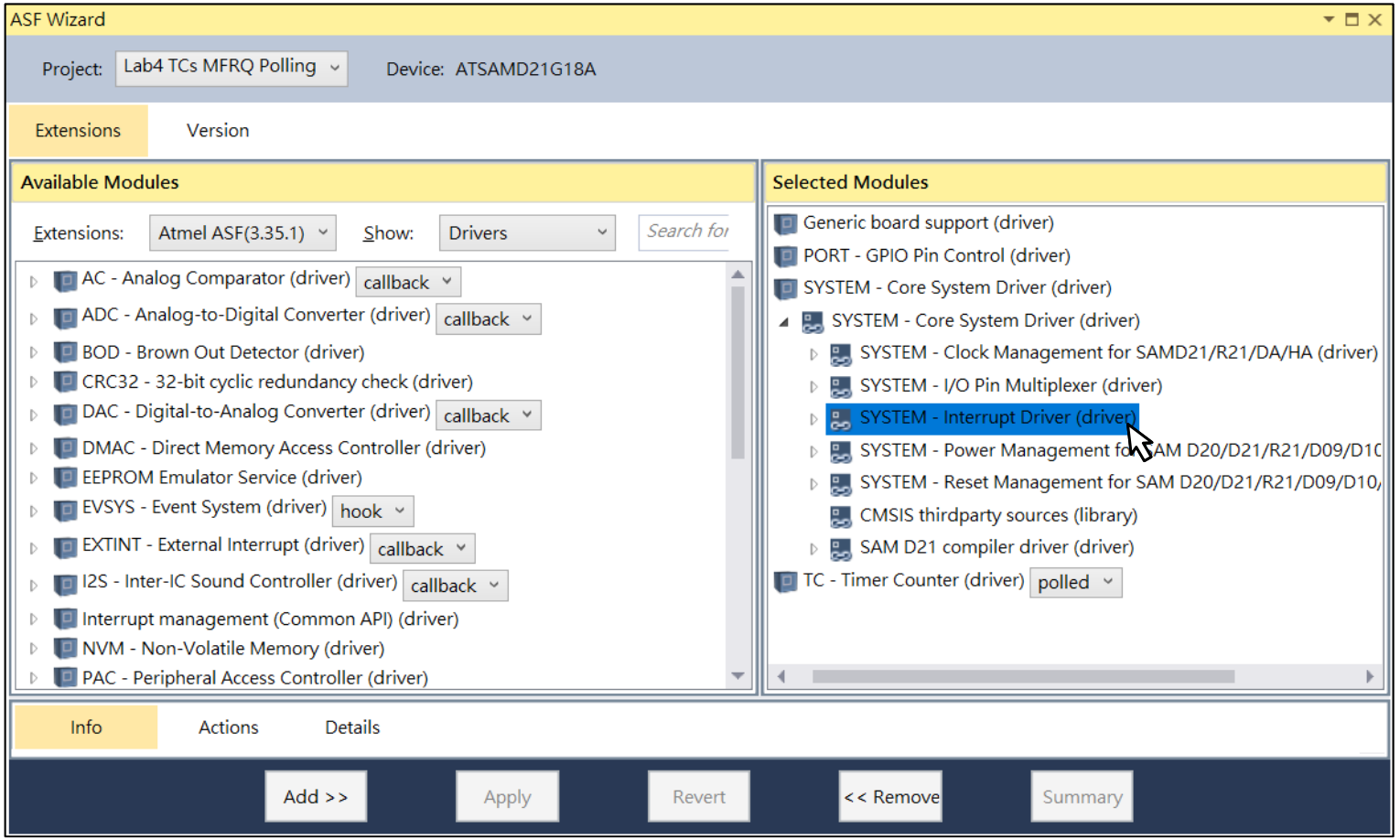
startup_samd21.c

```
__attribute__ ((section(".vectors"))) const DeviceVectors exception_table =
{
        /* Configure Initial Stack Pointer, using linker-generated symbols */
        .pvStack = (void*) (&_estack),
        .pfnReset_Handler = (void*) Reset_Handler,
        .pfnNMI_Handler = (void*) NMI_Handler,
        .pfnHardFault_Handler = (void*) HardFault_Handler,
        .pvReservedM12 = (void*) (0UL), /* Reserved */
        …
        .pfnSVC_Handler = (void*) SVC_Handler,
        …
        .pfnPendSV_Handler = (void*) PendSV_Handler,
        .pfnSysTick_Handler = (void*) SysTick_Handler,
        /* Configurable interrupts */
        .pfnPM_Handler = (void*) PM_Handler, /*  0 Power Manager */
        .pfnSYSCTRL_Handler = (void*) SYSCTRL_Handler, /*  1 System Control */
        .pfnWDT_Handler = (void*) WDT_Handler, /*  2 Watchdog Timer */
        …
        .pfnTCC0_Handler = (void*) TCC0_Handler, /* 15 Timer Counter Control 0 */
        .pfnTCC1_Handler = (void*) TCC1_Handler, /* 16 Timer Counter Control 1 */
        ....
}
```

© 2008 Microchip Technology Incorporated. All Rights Reserved.

NVIC Function from ASF

- **NVIC function include at standard new project package.**



© 2008 Microchip Technology Incorporated. All Rights Reserved.

ASF NVIC Driver Functions

- /* Enters/Leave a critical section. */
  void system_interrupt_enter_critical_section(void)
  void system_interrupt_leave_critical_section(void)

- /* global interrupts are enabled/disable. global interrupts status check */
  void system_interrupt_enable_global(void)
  void system_interrupt_disable_global(void)
  bool system_interrupt_is_global_enabled(void)

- /* global interrupts are enabled/disable. global interrupts status check */
  void system_interrupt_enable
  (const enum system_interrupt_vector vector)
  void system_interrupt_disable
  (const enum system_interrupt_vector vector)
  bool system_interrupt_is_enabled
  (const enum system_interrupt_vector vector)

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Hands-on Lab

No!

© 2008 Microchip Technology Incorporated. All Rights Reserved.

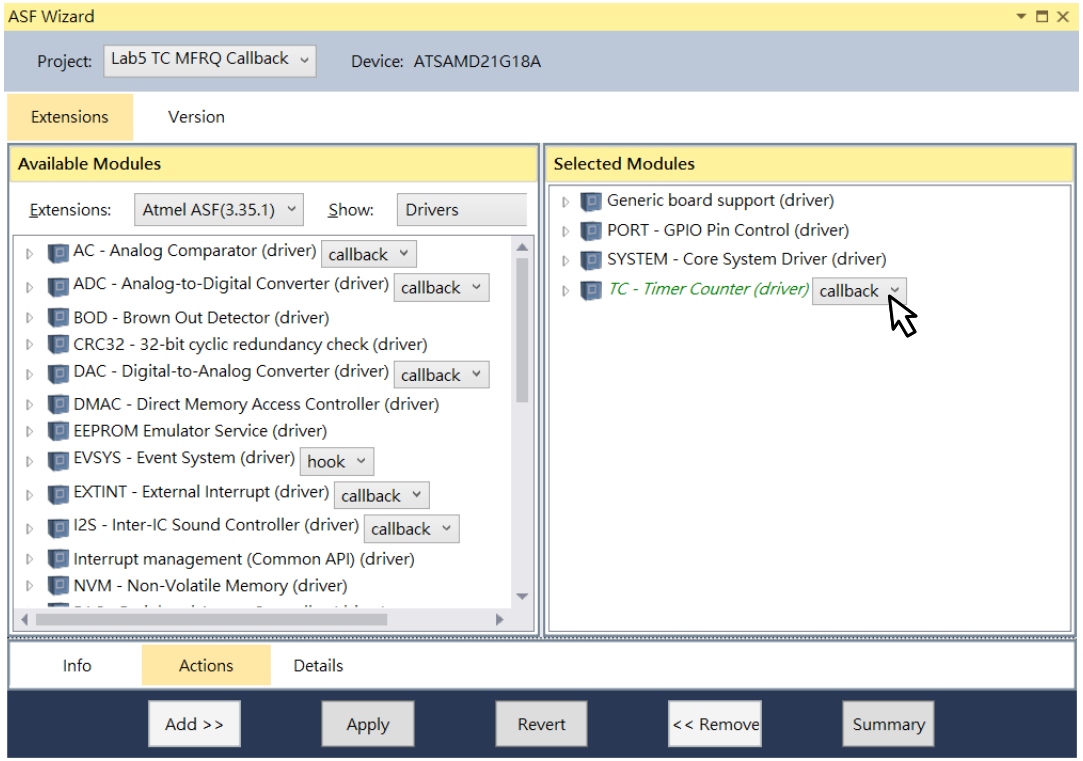
Callback Function

- Most ASF module provide the call “callback” mode driver.
- All flag and event handle by ASF automatically, user focus at application only. It’s more power and easy to understand than operating a NVIC function directly.
- The callback mode provide hook function to hook up your callback function to special interrupt source and event.
- For Example,
  you can hook LED toggle function at TC overflow event.
  toggle function executing automatically, when TC overflow event occurred.

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Change TC mode to callback

- **ASF provide more advanced function. Easy to use and understanding. Let me try to use ASF to my project.**
- Select **ASF ▸ ASF Wizard**, **Change to TC … callback**.



© 2008 Microchip Technology Incorporated. All Rights Reserved.

TC Driver Callback Functions

- /* Register callback functions with the driver */

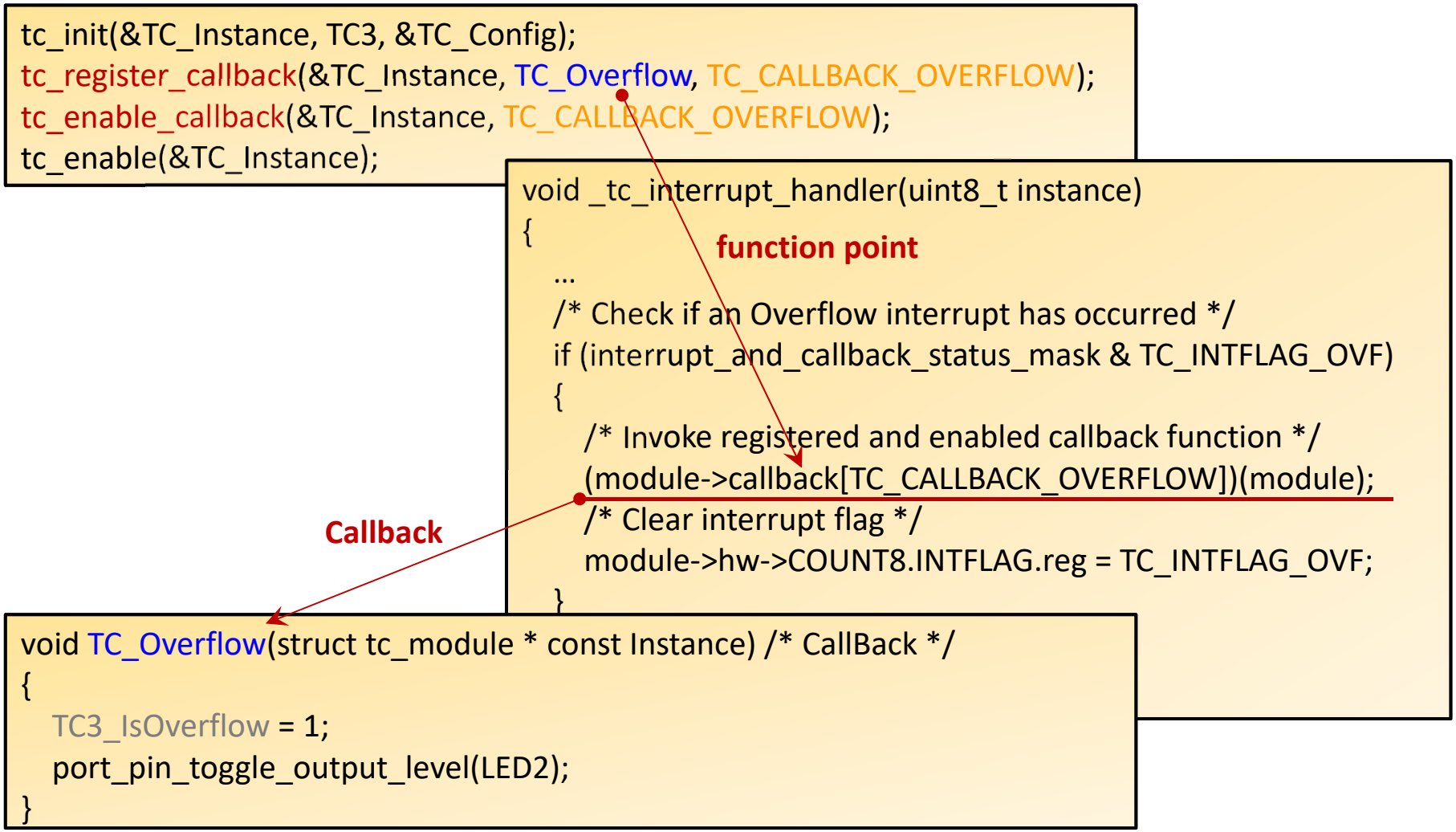
```
enum status_code tc_register_callback
                ( struct tc_module *const module,
                  tc_callback_t callback_func,
                  const enum tc_callback callback_type )
```

- /* Enable/Disable callback. */

```
static inline void tc_enable_callback
                                (struct tc_module *const module,
                                 const enum tc_callback callback_type)
static inline void tc_disable_callback
                                (struct tc_module *const module,
                                 const enum tc_callback callback_type)
```

© 2008 Microchip Technology Incorporated. All Rights Reserved.

TC Callback Hook Example



© 2008 Microchip Technology Incorporated. All Rights Reserved.

volatile Qualification

- If the ISR and the main program share the same variables, then. You must add the volatile qualification when this variable is declared.

  For Example : volatile uint8_t EventFlag = 0;

Sadm21g18a.h

```
/** \brief TC_COUNT16 hardware registers */
#if !(defined(__ASSEMBLY__) || defined(__IAR_SYSTEMS_A
typedef struct { /* 16-bit Counter Mode */
  __IO TC_CTRLA_Type              CTRLA;       /**< \br
  __IO TC_READREQ_Type            READREQ;     /**< \br
  __IO TC_CTRLBCLR_Type           CTRLBCLR;    /**< \br
  __IO TC_CTRLBSET_Type           CTRLBSET;    /**< \br
  __IO TC_CTRLC_Type              CTRLC;       /**< \br
       RoReg8                     Reserved1[0x1];
  __IO TC_DBGCTRL_Type            DBGCTRL;     /**< \br
       RoReg8                     Reserved2[0x1];
  __IO TC_EVCTRL_Type             EVCTRL;      /**< \br
  __IO TC_INTENCLR_Type           INTENCLR;    /**< \br
  __IO TC_INTENSET_Type           INTENSET;    /**< \br
  __IO TC_INTFLAG_Type            INTFLAG;     /**< \br
  __I  TC_STATUS_Type             STATUS;      /**< \br
  __IO TC_COUNT16_COUNT_Type      COUNT;       /**< \br
       RoReg8                     Reserved3[0x6];
                                  2];          /**< \br

                                  defined(__IAR_SYS
```

Core-cm0plus.h

```
#ifdef __cplusplus
  #define   __I     volatile
#else
  #define   __I     volatile const
#endif
#define     __O     volatile
#define     __IO    volatile
```

Optimization for general variables the results correct !	Optimization for peripherals function registers the results incorrect !
B=A; C=B; Optimized C=A;	TMR1=A; C=TMR1; Optimized C=A; !

* All peripherals function registers add volatile qualification already.

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab4 TC MFRQ Callback

- Try to modify code style to callback style. Please change TC3, LED1 toggle function to callback style.
- First, Delete LED1 toggle code segment at main loop.
- Create a callback function for LED toggle, then hook up to TC3 overflow event.
- The TC3 setup no need any change, just hook up callback function and enable callback function.

- Let's go!

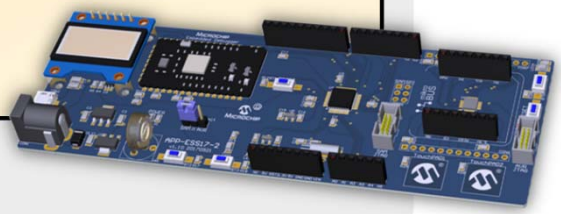
© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab4 TC MFRQ Callback

Step

a **Add below code segment to your main loop**

```
void TC3_Overflow(struct tc_module * const);

struct tc_module TC3_Instance;
struct tc_config TC_Config;

volatile uint8_t TC3_IsOverflow = 0;

int main (void)
{
   ...
   tc_init(&TC3_Instance, TC3, &TC_Config);
   tc_register_callback(&TC3_Instance, TC3_Overflow, TC_CALLBACK_OVERFLOW);
   tc_enable_callback(&TC3_Instance, TC_CALLBACK_OVERFLOW);
   tc_enable(&TC3_Instance);

   while(1)
   {
      if(TC3_IsOverflow)
         TC3_IsOverflow = 0; port_pin_toggle_output_level(LED1);
   }
}

void TC3_Overflow(struct tc_module * const)
{
   TC3_IsOverflow = 1;
}
```



b **Program firmware to target board then observe result.**

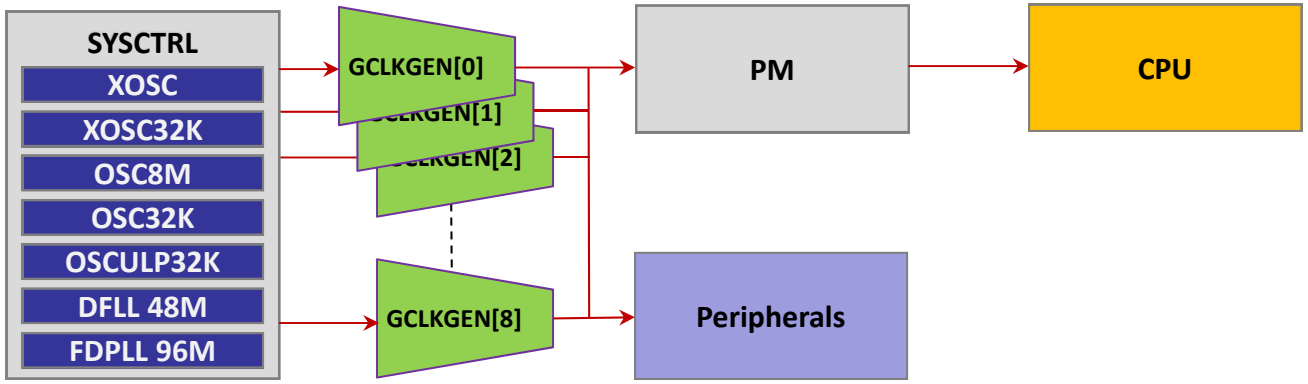
© 2008 Microchip Technology Incorporated. All Rights Reserved.

Clock System Architecture

© 2008 Microchip Technology Incorporated. All Rights Reserved.

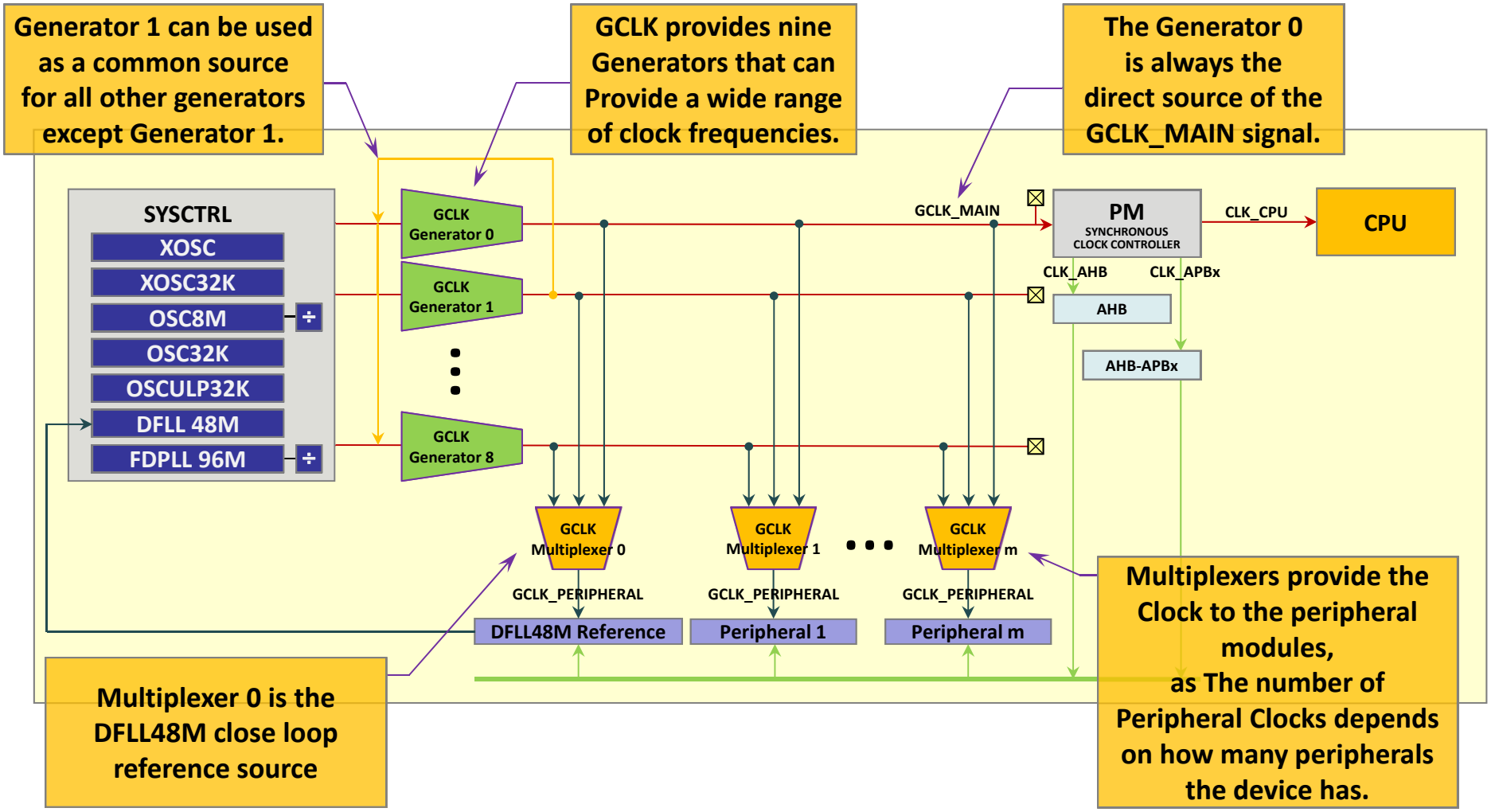
Clock System

- Review previously section, SAMD21's clock system provides clock sources to the Generic Clock Controller。The available clock sources are XOSC, XOSC32K, OSC32K, OSCULP32K, OSC8M,DFLL48M and FDPLL96M。
- The bus clock can be enabled and disabled in the Power Manager(PM) include CPU Clock(GCLK_MAIN)。



© 2008 Microchip Technology Incorporated. All Rights Reserved.

Clock System Block Diagram

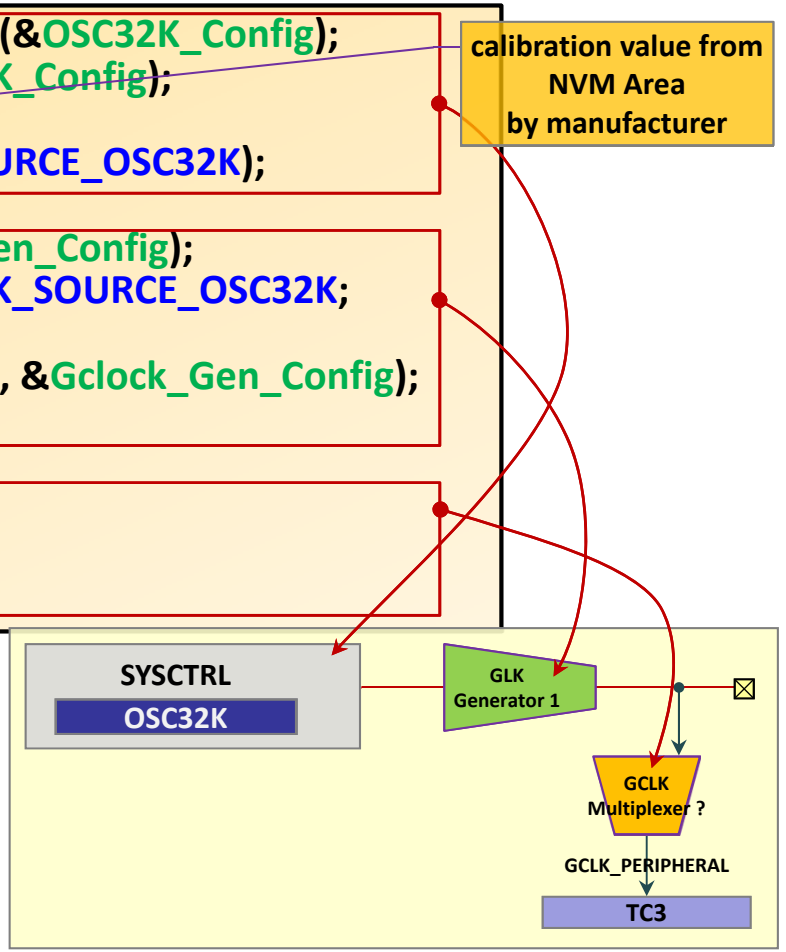


© 2008 Microchip Technology Incorporated. All Rights Reserved.

Clock Setting Use ASF Function

- For Example : Internal 32k RC -> Generator 1 -> TC3

```
system_clock_source_osc32k_get_config_defaults(&OSC32K_Config);
system_clock_source_osc32k_set_config(&OSC32K_Config);
SYSCTRL->OSC32K.bit.CALIB = 0x2C;
system_clock_source_enable(SYSTEM_CLOCK_SOURCE_OSC32K);

system_gclk_gen_get_config_defaults(&Gclock_Gen_Config);
Gclock_Gen_Config.source_clock = SYSTEM_CLOCK_SOURCE_OSC32K;
Gclock_Gen_Config.output_enable = true;
system_gclk_gen_set_config(GCLK_GENERATOR_1, &Gclock_Gen_Config);
system_gclk_gen_enable(GCLK_GENERATOR_1);

tc_get_config_defaults(&TC_Config);
TC_Config.clock_source = GCLK_GENERATOR_1;
tc_init(&TC_Instance, TC3, &TC_Config);
```

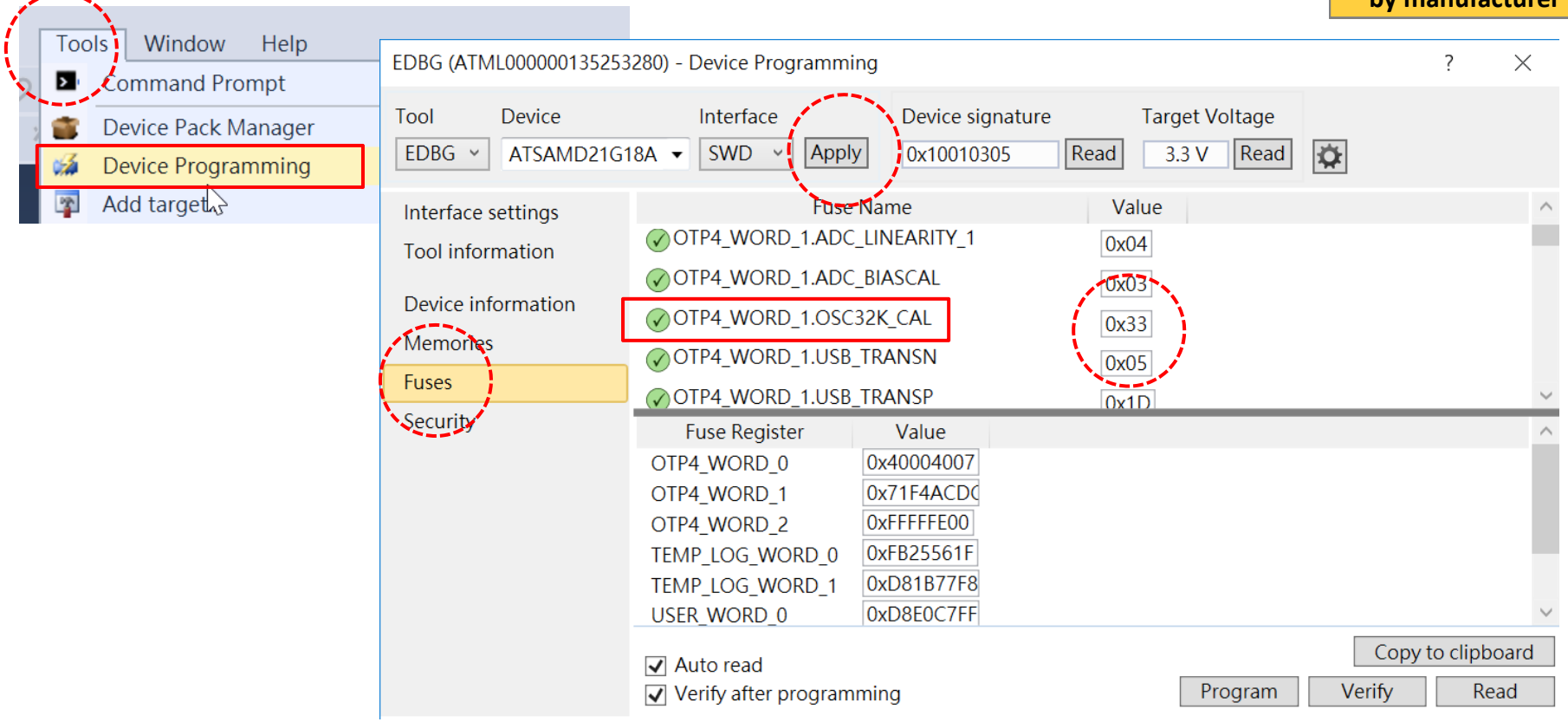


© 2008 Microchip Technology Incorporated. All Rights Reserved.

Clock Setting Use ASF Function

```
system_clock_source_osc32k_get_config_defaults(&OSC32K_Config);
system_clock_source_osc32k_set_config(&OSC32K_Config);
SYSCTRL->OSC32K.bit.CALIB = 0x2C;
system_clock_source_enable(SYSTEM_CLOCK_SOURCE_OSC32K);
```

calibration value from NVM Area by manufacturer

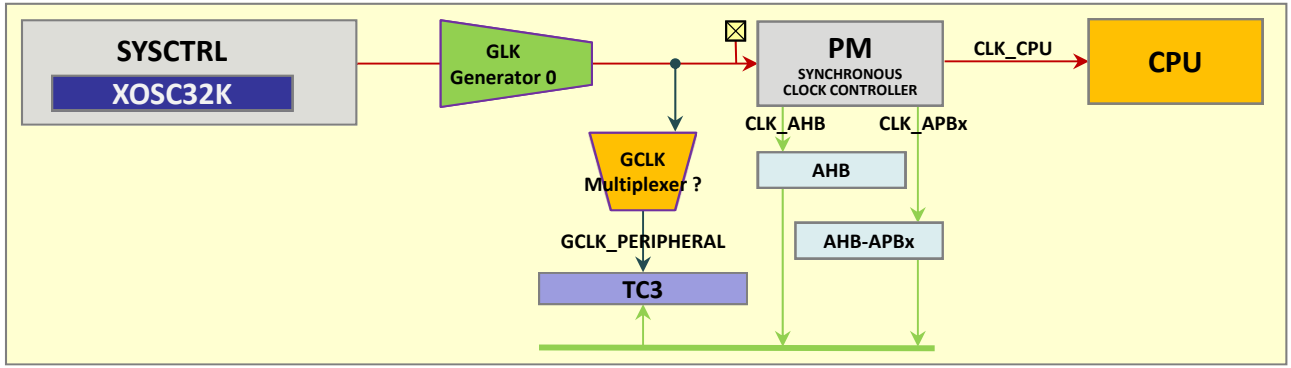


© 2008 Microchip Technology Incorporated. All Rights Reserved.

Clock Setting Use ASF Function

- For Example : External 32.768K -> Generator 0 -> PM & TC3

```
system_clock_source_xosc32k_get_config_defaults(&XOSC32K_Config);
system_clock_source_xosc32k_set_config(&XOSC32K_Config);
system_clock_source_enable(SYSTEM_CLOCK_SOURCE_XOSC32K);

system_gclk_gen_get_config_defaults(&Gclock_Gen_Config);
Gclock_Gen_Config.source_clock = SYSTEM_CLOCK_SOURCE_XOSC32K;
Gclock_Gen_Config.output_enable = true;
system_gclk_gen_set_config(GCLK_GENERATOR_0, &Gclock_Gen_Config);
system_gclk_gen_enable(GCLK_GENERATOR_0);

tc_get_config_defaults(&TC_Config);
TC_Config.clock_source = GCLK_GENERATOR_0;
tc_init(&TC_Instance, TC3, &TC_Config);
```



© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab5 System Clock XOSC32K

- Try to change main clock from default source to external 32.768K.
- Setting XOSC32K then enable it, first. Then Connect XOSC32K to Generator 0. Connect TC3 to Generator 0.

- Let's go!

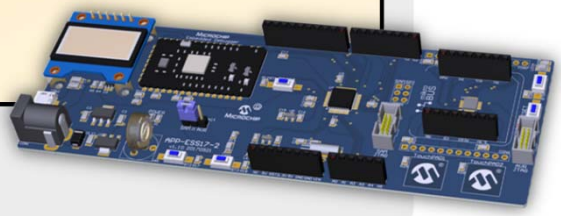
© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab5 System Clock XOSC32

Step1

a **Add below code segment to your main loop**

```
...
int main (void)
{
    ...
    system_clock_source_xosc32k_get_config_defaults(&XOSC32K_Config);
    system_clock_source_xosc32k_set_config(&XOSC32K_Config);
    system_clock_source_enable(SYSTEM_CLOCK_SOURCE_XOSC32K);

    system_gclk_gen_get_config_defaults(&Gclock_Gen_Config);
    Gclock_Gen_Config.source_clock = SYSTEM_CLOCK_SOURCE_XOSC32K;
    Gclock_Gen_Config.output_enable = true;
    system_gclk_gen_set_config(GCLK_GENERATOR_0, &Gclock_Gen_Config);
    system_gclk_gen_enable(GCLK_GENERATOR_0);
    ...
    TC_Config clock_source = GCLK_GENERATOR_0;
    tc_init(&TC3_Instance, TC3, &TC_Config);
    ...
    while(1);
}
```



b **Program firmware to target board then observe result.**

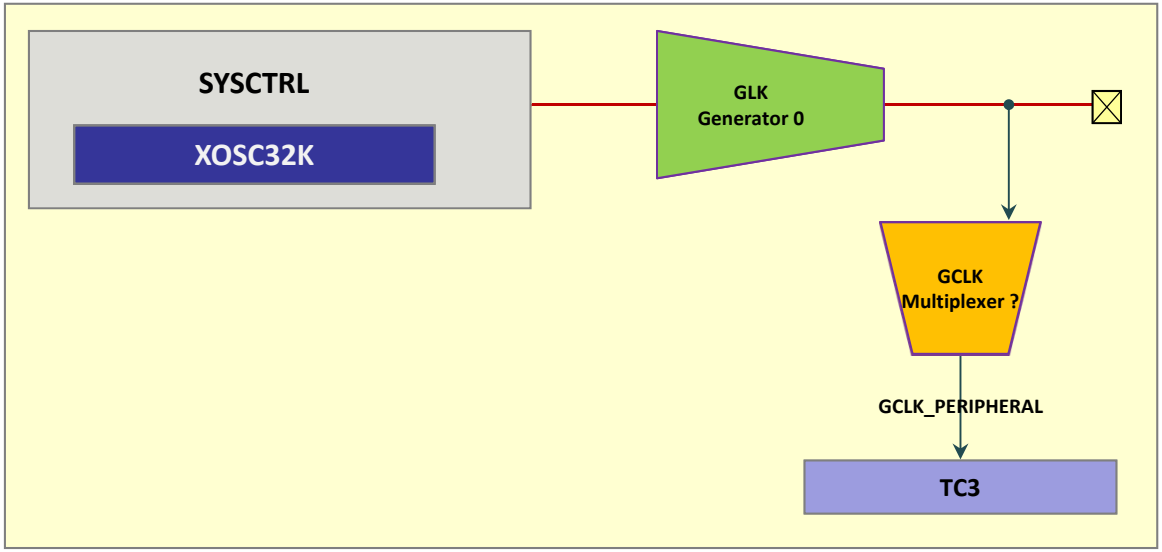
© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab5 System Clock XOSC32

Step2

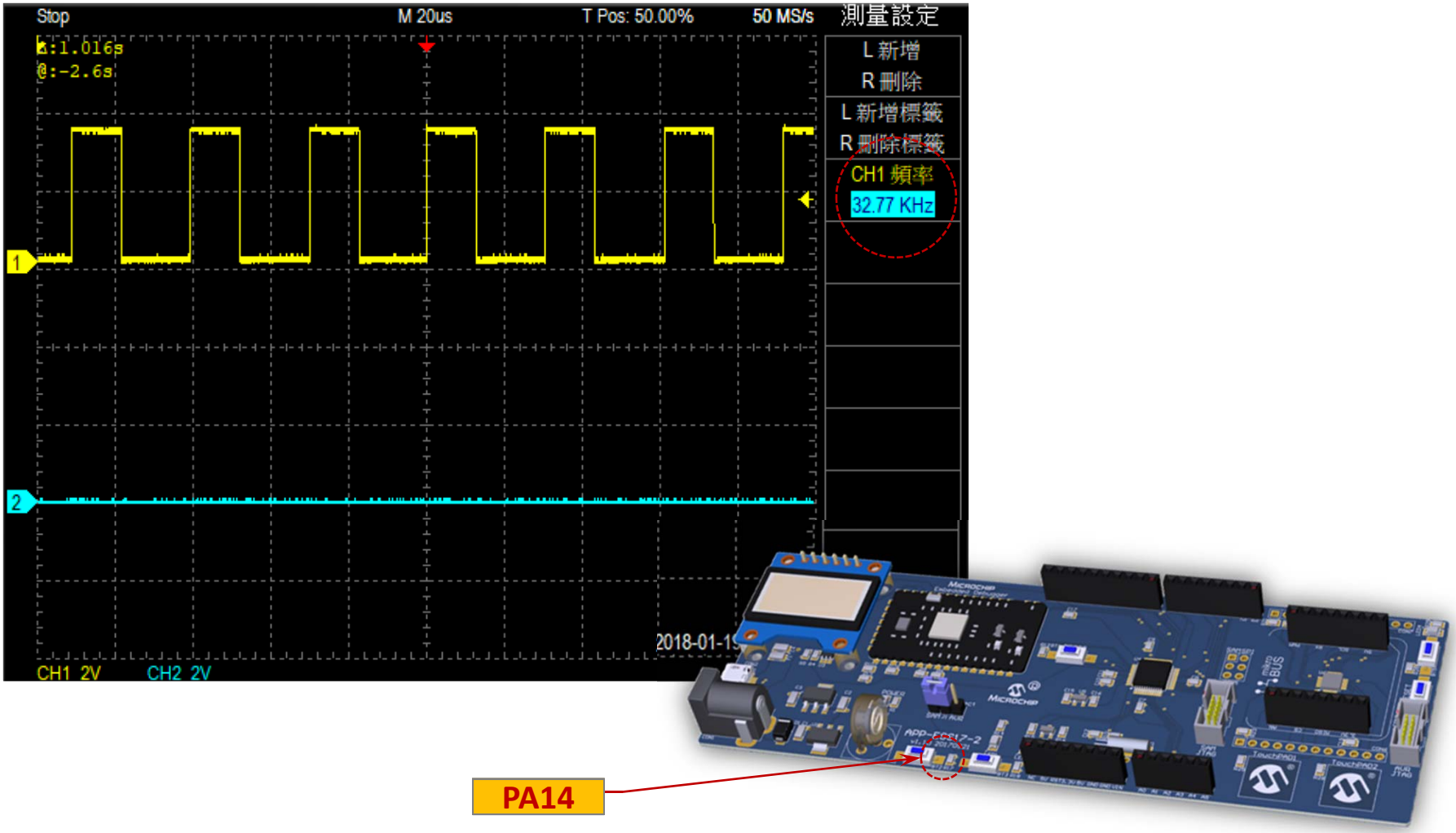
- **Modify PINMUX code segment to below. Than you can measure the Generator 0 at PA14.**

```
struct system_pinmux_config PINMUX_Config;

system_pinmux_get_config_defaults(&PINMUX_Config);
PINMUX_Config.mux_position = MUX_PA14H_GCLK_IO0;
system_pinmux_pin_set_config(PIN_PA14H_GCLK_IO0, &PINMUX_Config);
```



© 2008 Microchip Technology Incorporated. All Rights Reserved.

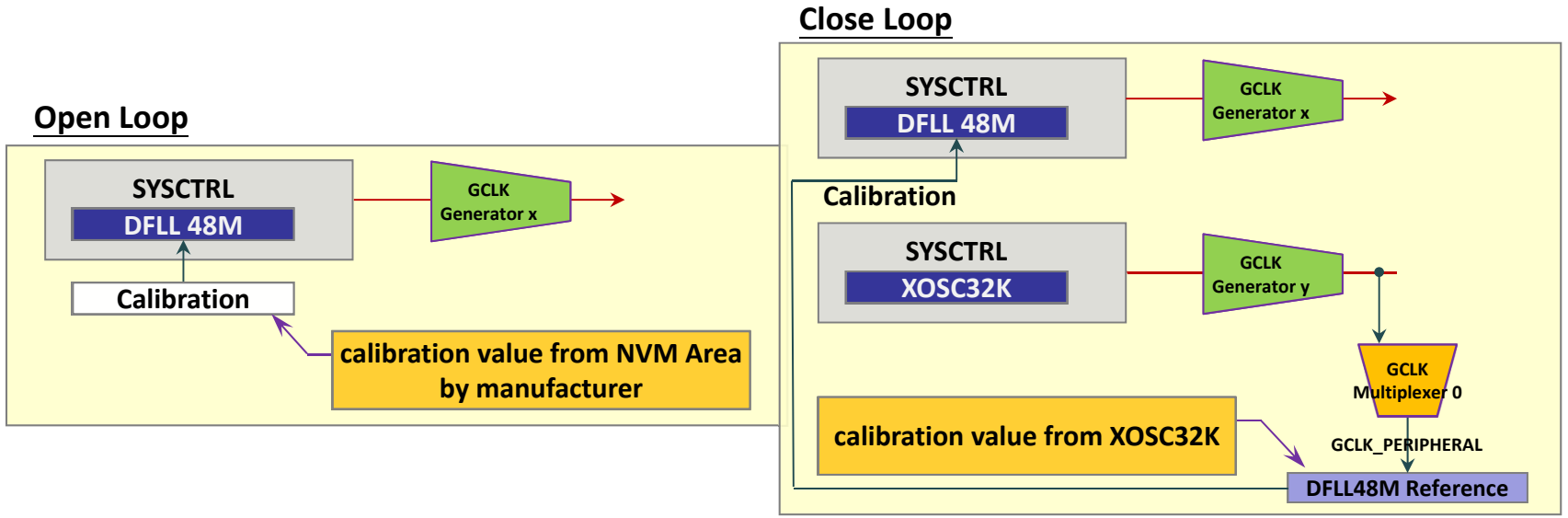
Clock Output



© 2008 Microchip Technology Incorporated. All Rights Reserved.

Clock source DFLL48M

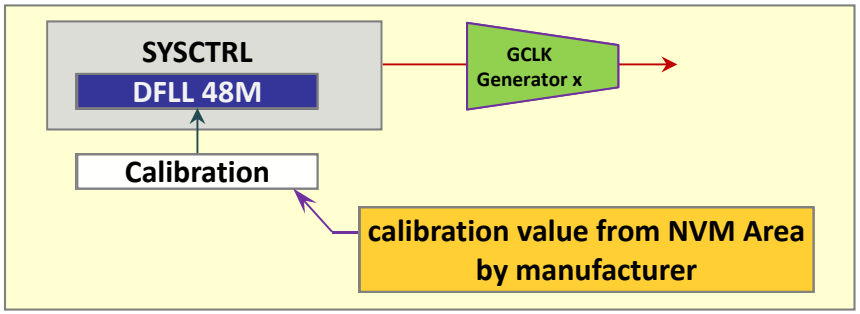
- There are 2 different mode for DFLL48M clock source calibration.
  - Open Loop : Clock turning from fix calibration value by manufacturer.
  - Close Loop : Clock turning from external 32.786k crystal.

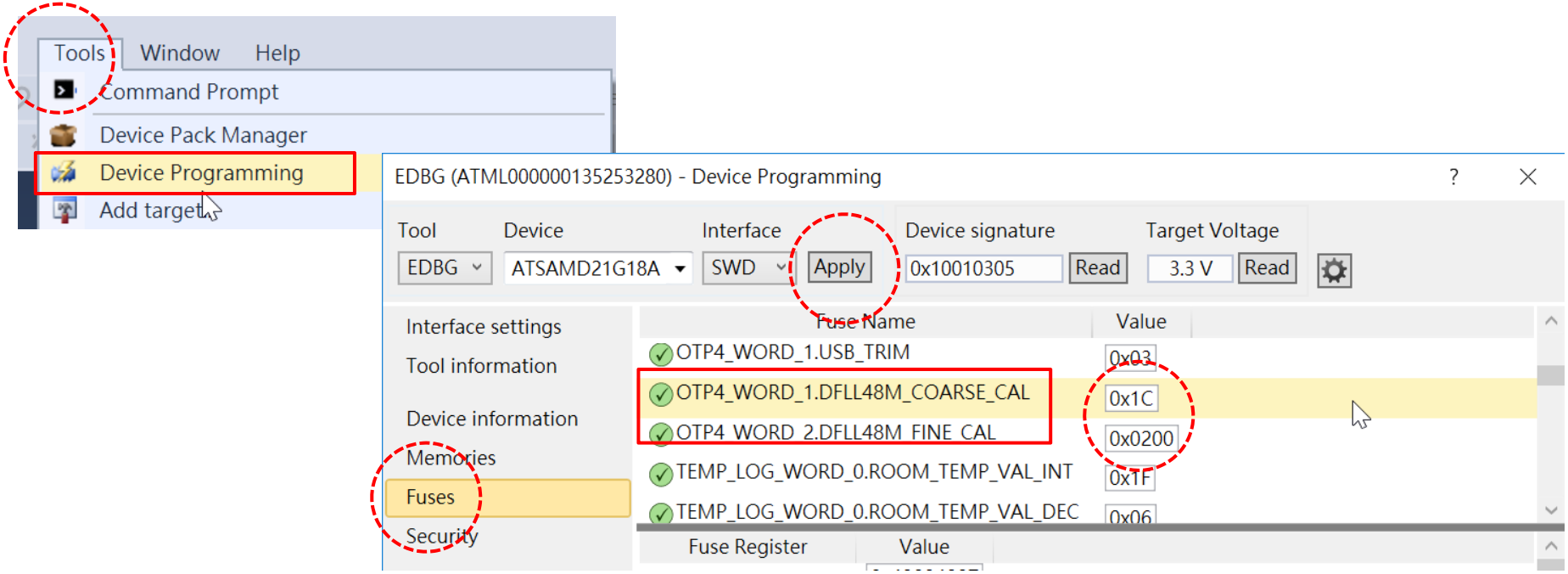


© 2008 Microchip Technology Incorporated. All Rights Reserved.

Clock source DFLL48M

Open Loop





© 2008 Microchip Technology Incorporated. All Rights Reserved.

Clock Setting Use Predefined Header

- ASF also provide predefined header file call conf_clock.h. It's locate at \src\config\. User can modify header file depend on needs directly, don't need any source code.

```
/* SYSTEM_CLOCK_SOURCE_OSC8M configuration - Internal 8MHz oscillator */
#define CONF_CLOCK_OSC8M_PRESCALER SYSTEM_OSC8M_DIV_1
...
/* SYSTEM_CLOCK_SOURCE_XOSC configuration - External clock/oscillator */
...
/* SYSTEM_CLOCK_SOURCE_XOSC32K configuration - External 32KHz crystal/clock oscillator */
#define CONF_CLOCK_XOSC32K_ENABLE false
...
/* SYS
...
```

Clock Source

```
/* Configure GCLK generator 0 (Main Clock) */
# define CONF_CLOCK_GCLK_0_ENABLE true
# define CONF_CLOCK_GCLK_0_RUN_IN_STANDBY false
# define CONF_CLOCK_GCLK_0_CLOCK_SOURCE SYSTEM_CLOCK_SOURCE_OSC8M
# define CONF_CLOCK_GCLK_0_PRESCALER 1
# define CONF_CLOCK_GCLK_0_OUTPUT_ENABLE false

/* Configure GCLK generator 1 */
...
```

Generator Setting

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Flash Access Time and Wait State

- The SAMD21 main flash Random Read time around 40nS. This is mean you must adjust wait state to suitable value when main clock over 24MHz(2.7~3.63V). The actually value show below, differ value depend on system supply

V_{DD} range	NVM Wait States	Maximum Operating Frequency	Units
1.62V to 2.7V	0	14	MHz
	1	28	
	2	42	
	3	48	
2.7V to 3.63V	0	24	
	1	48	

```
/* System clock bus configuration */
# define CONF_CLOCK_CPU_CLOCK_FAILURE_DETECT false
# define CONF_CLOCK_FLASH_WAIT_STATES 0 or 1 or 2 or 3
# define CONF_CLOCK_CPU_DIVIDER SYSTEM_MAIN_CLOCK_DIV_1
# define CONF_CLOCK_APBA_DIVIDER SYSTEM_MAIN_CLOCK_DIV_1
# define CONF_CLOCK_APBB_DIVIDER SYSTEM_MAIN_CLOCK_DIV_1
# define CONF_CLOCK_APBC_DIVIDER SYSTEM_MAIN_CLOCK_DIV_1
```

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab6 System Clock DFLL48M

Close Loop

- Try to change main clock from to DFLL48M Close Loop.
- Find DFLL48M clock source then enable it, assign DFLL48M as source of Generator 0.
- Find XOSC32K clock source then enable it, assign XOSC32K as source of Generator 1.
- Setup Generator 1 output as DFLL48M close loop reference .
- Setup DFLL48M multiply factor (48M/32.768K).
- Adjustment wait state to 1.

- Let’s go!

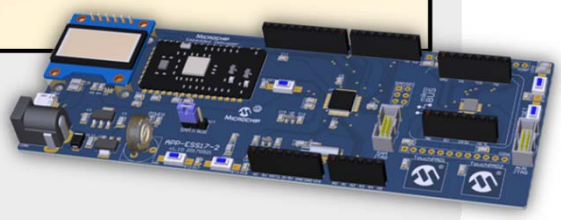
© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab6 System Clock DFLL48M

Step

a **modify below line at conf_clock.h**

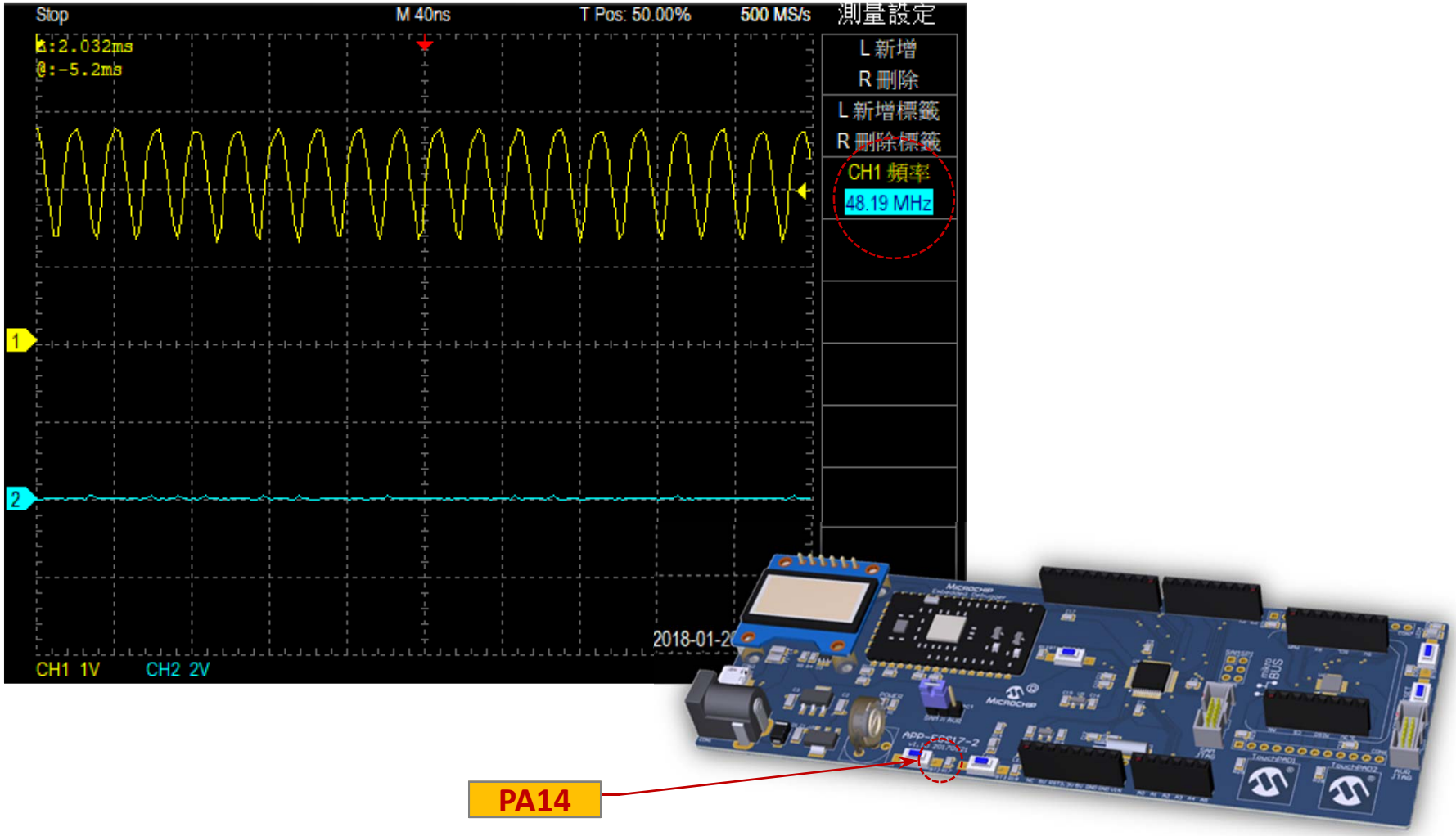
```
// Line 91 /* SYSTEM_CLOCK_SOURCE_DFLL configuration – Digital Frequency Locked Loop */
# define CONF_CLOCK_DFLL_ENABLE true
# define CONF_CLOCK_DFLL_LOOP_MODE SYSTEM_CLOCK_DFLL_LOOP_MODE_CLOSED
// Line 134 /* Configure GCLK generator 0 (Main Clock) */
# define CONF_CLOCK_GCLK_0_ENABLE true
# define CONF_CLOCK_GCLK_0_CLOCK_SOURCE SYSTEM_CLOCK_SOURCE _DFLL
# define CONF_CLOCK_GCLK_0_OUTPUT_ENABLE true
// Line 74 /* SYSTEM_CLOCK_SOURCE_XOSC32K configuration –
                                        External 32KHz crystal/clock oscillator */
# define CONF_CLOCK_XOSC32K_ENABLE true
// Line 141 /* Configure GCLK generator 1 */
# define CONF_CLOCK_GCLK_1_ENABLE true
# define CONF_CLOCK_GCLK_1_CLOCK_SOURCE SYSTEM_CLOCK_SOURCE_XOSC32K
// Line 99 /* DFLL closed loop mode configuration */
# define CONF_CLOCK_DFLL_SOURCE_GCLK_GENERATOR GCLK_GENERATOR_1
# define CONF_CLOCK_DFLL_MULTIPLY_FACTOR (48000000 / 32768)
// Line 51 /* System clock bus configuration */
# define CONF_CLOCK_FLASH_WAIT_STATES 1
```



b **Program firmware to target board then observe result.**

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Clock Output

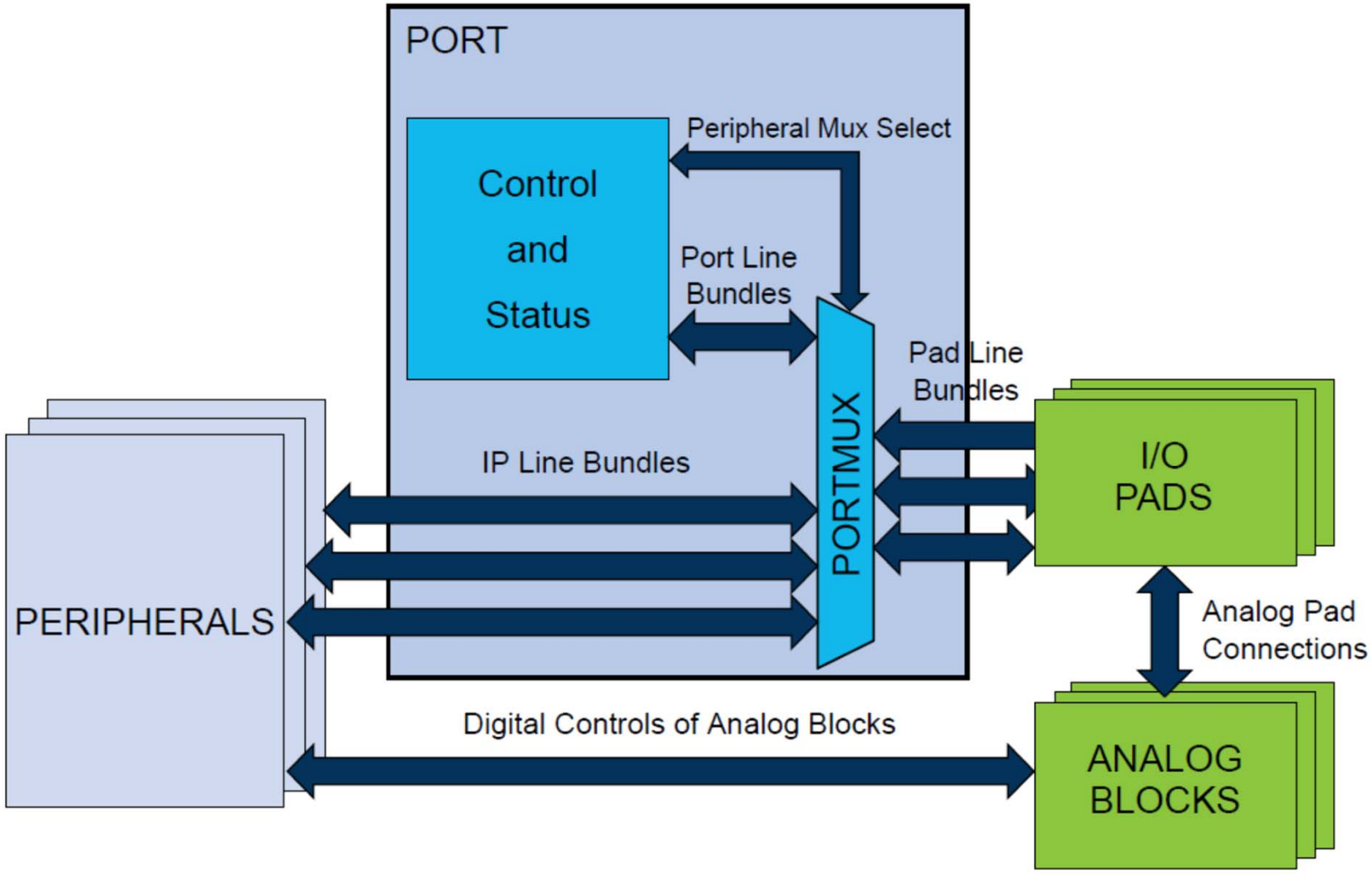


© 2008 Microchip Technology Incorporated. All Rights Reserved.

PINMUX &
SERCOM - UART Architecture

© 2008 Microchip Technology Incorporated. All Rights Reserved.

What's PINMUX ?



© 2008 Microchip Technology Incorporated. All Rights Reserved.

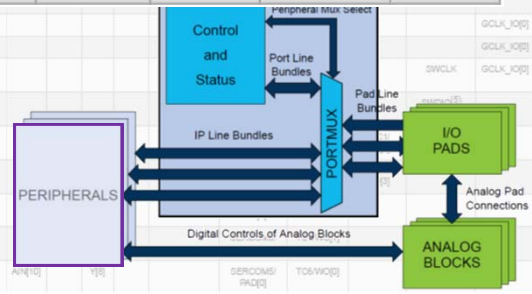
What's PINMUX ?

- Each pin is by default controlled by the PORT as a general purpose I/O and alternatively. It can be assigned to one of the peripheral functions group.

Function Groups

MUX			0	1					2	3	4	5	6	7
Pin(1)			I/O Pin	Supply	A	B(2)(3)					C	D	E	F	G	H
SAMD21E	SAMD21G	SAMD21J			EIC	REF	ADC	AC	PTC	DAC	SERCOM(2)(3)	SERCOM-ALT	TC(4) /TCC	TCC	COM	AC/ GCLK
15	23	31	PA14	VDDIO	EXTINT[14]						SERCOM2/ PAD[2]	SERCOM4/ PAD[2]	TC3/WO[0]	TCC0/ WO[4]		GCLK_IO[0]
16	24	32	PA15	VDDIO	EXTINT[15]						SERCOM2/ PAD[3]	SERCOM4/ PAD[3]	TC3/WO[1]	TCC0/ WO[5]		GCLK_IO[1]
	37	49	PB22	VDDIO	EXTINT[6]							SERCOM5/ PAD[2]	TC7/WO[0]			GCLK_IO[0]
	38	50	PB23	VDDIO	EXTINT[7]							SERCOM5/ PAD[3]	TC7/WO[1]			GCLK_IO[1]

Pin Name



© 2008 Microchip Technology Incorporated. All Rights Reserved.

ASF PINMUX Functions

- /* Initializes a Port pin configuration structure to defaults. */

```
void system_pinmux_get_config_defaults(
                              struct system_pinmux_config *const config)
```

- /* Writes a Port pin configuration to the hardware module. */

```
void system_pinmux_pin_set_config(
                          const uint8_t gpio_pin,
                          const struct system_pinmux_config *const config);
```

```
struct system_pinmux_config
{
   uint8_t mux_position;
   enum system_pinmux_pin_dir direction;
   enum system_pinmux_pin_pull input_pull;
   bool powersave;
};
```

© 2008 Microchip Technology Incorporated. All Rights Reserved.

PINMUX for GPIO

main.c (Lab2)

```
port_get_config_defaults(&PORT_Config);
PORT_Config.direction = PORT_PIN_DIR_OUTPUT;
port_pin_set_config(LED1, &PORT_Config);
```



```
struct system_pinmux_config PINMUX_Config;
system_pinmux_get_config_defaults(&PINMUX_Config);
PINMUX_Config.mux_position = SYSTEM_PINMUX_GPIO;
PINMUX_Config.direction = SYSTEM_PINMUX_PIN_DIR_OUT;
system_pinmux_pin_set_config(PIN_PA20, &PINMUX_Config);
```

pinmux.h

```
#define SYSTEM_PINMUX_GPIO (1 << 7)
```

samd21G18A.h

```
#define PIN_PA20 20
```

© 2008 Microchip Technology Incorporated. All Rights Reserved.

PINMUX for TC

MUX	0	1					2	3	4	5	6	7
Pin(1)			I/O Pin	Supply	A	B(2)(3)				C	D	E	F	G	H
SAMD21E	SAMD21G	SAMD21J			EIC	REF	ADC	AC	PTC	DAC	SERCOM(2)(3)	SERCOM-ALT	TC(4) /TCC	TCC	COM	AC/ GCLK
15	23	31	PA14	VDDIO	EXTINT[14]						SERCOM2/ PAD[2]	SERCOM4/ PAD[2]	TC3/WO[0]	TCC0/ WO[4]		GCLK_IO[0]

main.c (Lab3~4)

```
TC_Config.pwm_channel[0].enabled = true;
TC_Config.pwm_channel[0].pin_out = PIN_PA14E_TC3_WO0;
TC_Config.pwm_channel[0].pin_mux = MUX_PA14E_TC3_WO0;
```

samd21G18A.h

```
#define PIN_PA14E_TC3_WO0 14L
#define MUX_PA14E_TC3_WO0 4L
```

tc.c

```
if (config->pwm_channel[0].enabled)
{
    system_pinmux_get_config_defaults(&pin_config);
    pin_config.mux_position = config->pwm_channel[0].pin_mux;
    pin_config.direction = SYSTEM_PINMUX_PIN_DIR_OUTPUT;
    system_pinmux_pin_set_config(
                        config->pwm_channel[0].pin_out, &pin_config);
}
```

© 2008 Microchip Technology Incorporated. All Rights Reserved.

PINMUX for GCLK output

	MUX	0	1					2	3	4	5	6	7
Pin[1]			I/O Pin	Supply	A	B[2][3]				C	D	E	F	G	H
SAMD21E	SAMD21G	SAMD21J			EIC	REF	ADC	AC	PTC	DAC	SERCOM[2][3]	SERCOM-ALT	TC[4] /TCC	TCC	COM	AC/ GCLK
15	23	31	PA14	VDDIO	EXTINT[14]						SERCOM2/ PAD[2]	SERCOM4/ PAD[2]	TC3/WO[0]	TCC0/ WO[4]		GCLK_IO[0]

main.c (Lab5~6)

```
system_pinmux_get_config_defaults(&PINMUX_Config);
PINMUX_Config.mux_position = MUX_PA14H_GCLK_IO0;
system_pinmux_pin_set_config(PIN_PA14H_GCLK_IO0, &PINMUX_Config);
```

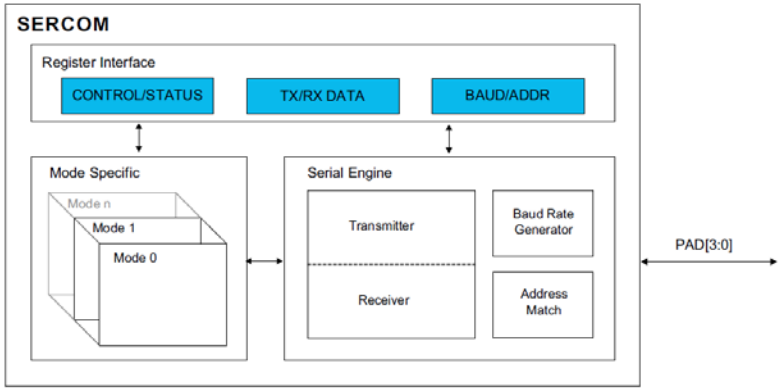
samd21G18A.h

```
#define PIN_PA14H_GCLK_IO0 14L
#define MUX_PA14H_GCLK_IO0 7L
```

© 2008 Microchip Technology Incorporated. All Rights Reserved.

SERCOM Introduction

- Serial Communication Interface, SERCOM be configured to support a number of modes :
  - I2C
  - SPI
  - USART : LIN, RS232, RS485, IrDA
- There are up to six instances of the serial communication interface SERCOM peripheral.



- The SERCOM serial engine consists of a transmitter and receiver, baud-rate generator and address matching functionality. It can use the internal generic clock or an external clock to operate in all sleep.

© 2008 Microchip Technology Incorporated. All Rights Reserved.

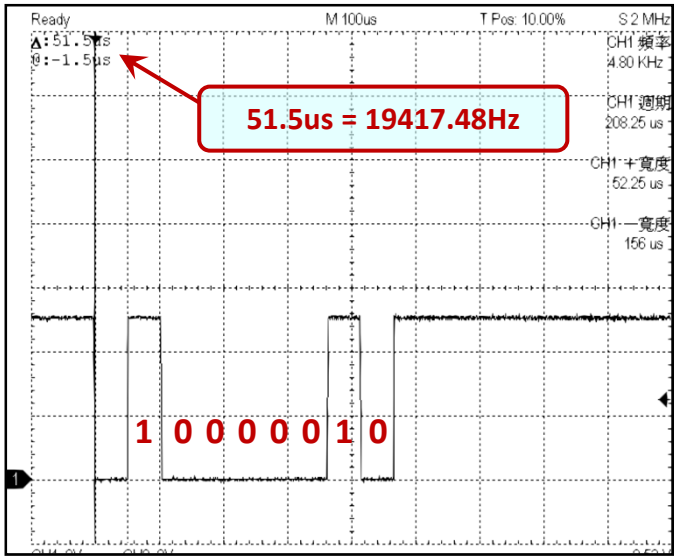
SERCOM Introduction

- **For I2C**
  - Fast Mode (400kHz)
  - Master or slave operation
  - SMBus compatible
  - Wake on address match
- **For SPI**
  - Up to 24Mb/s
  - Supports all four SPI modes
  - Full-duplex operation
  - Single data direction mode frees unused MISO or MOSI pin
- **For USART**
  - Up to 24Mb/s
  - Half-/Full-duplex operation
  - Sync and Async operation

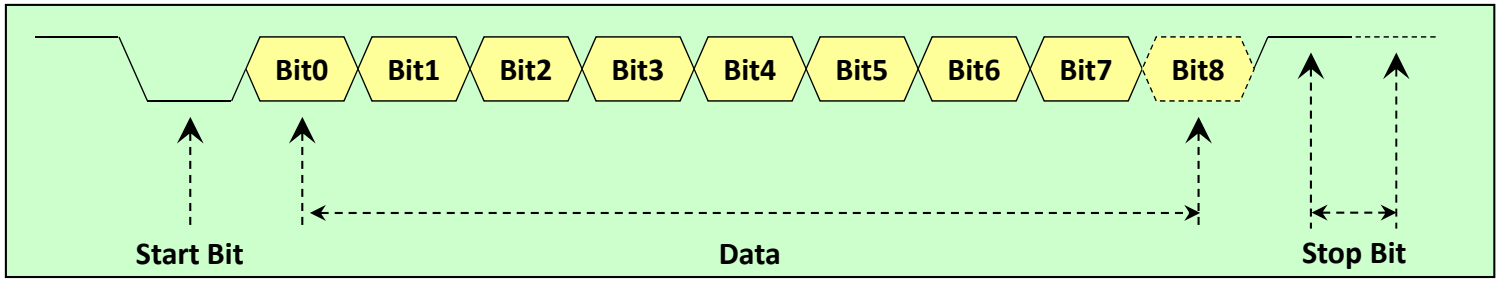
© 2008 Microchip Technology Incorporated. All Rights Reserved.

What's UART

- UART : Universal Asynchronous Receiver Transmitter.
- The module data exchange through serial communication.
- The data formatting include 1 start bit 5 to 9 data bits and 1 or 2 stop bits.



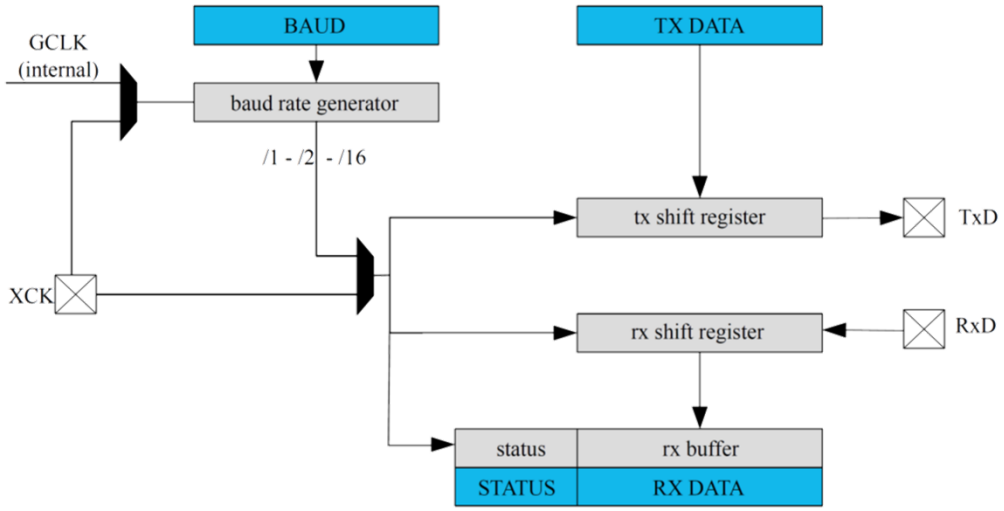
UART Transmit Example :
'A'(0x41), 19200 bps



© 2008 Microchip Technology Incorporated. All Rights Reserved.

SERCOM - USART

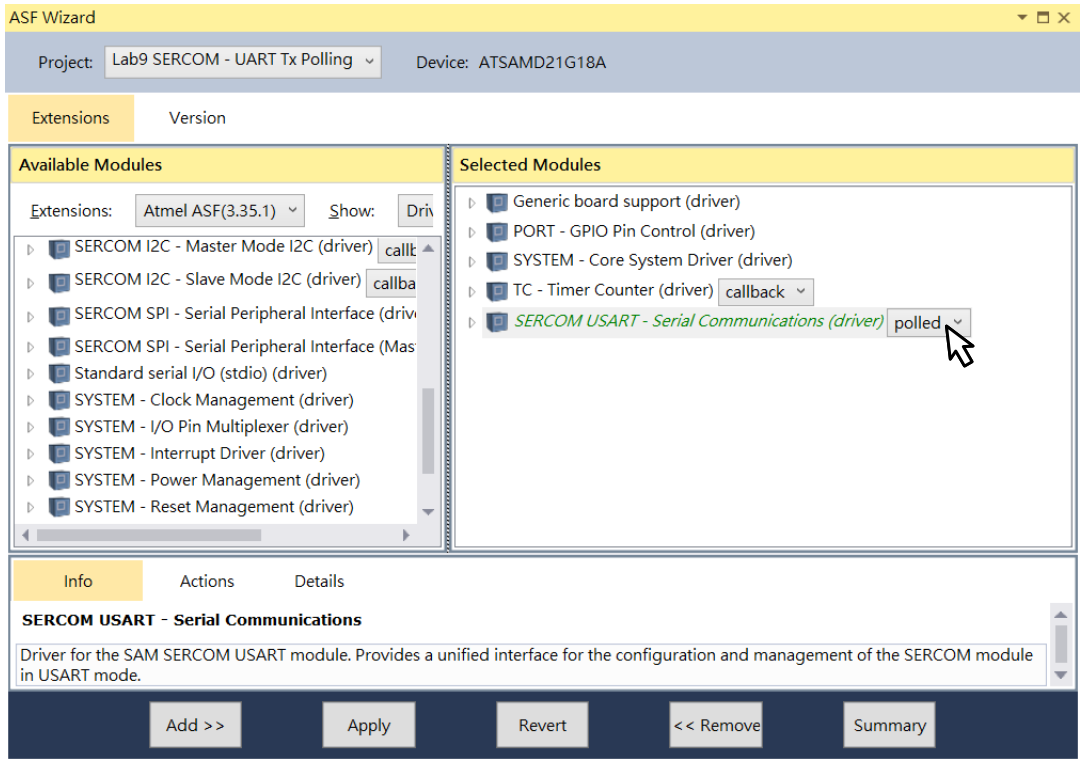
- Supports serial frames with 5, 6, 7, 8 or 9 data bits and 1 or 2 stop bits, parity check, RTS and CTS flow control.
- Full-duplex operation, Support RS-232, RS-485, IrDA and LIN Bus.
- Buffer overflow and frame error detection.
- Internal or external clock source for asynchronous and synchronous operation.



© 2008 Microchip Technology Incorporated. All Rights Reserved.

Add SERCOM Function from ASF

- Select **ASF ▸ ASF Wizard**, **Add SERCOM USART … polled to project.**



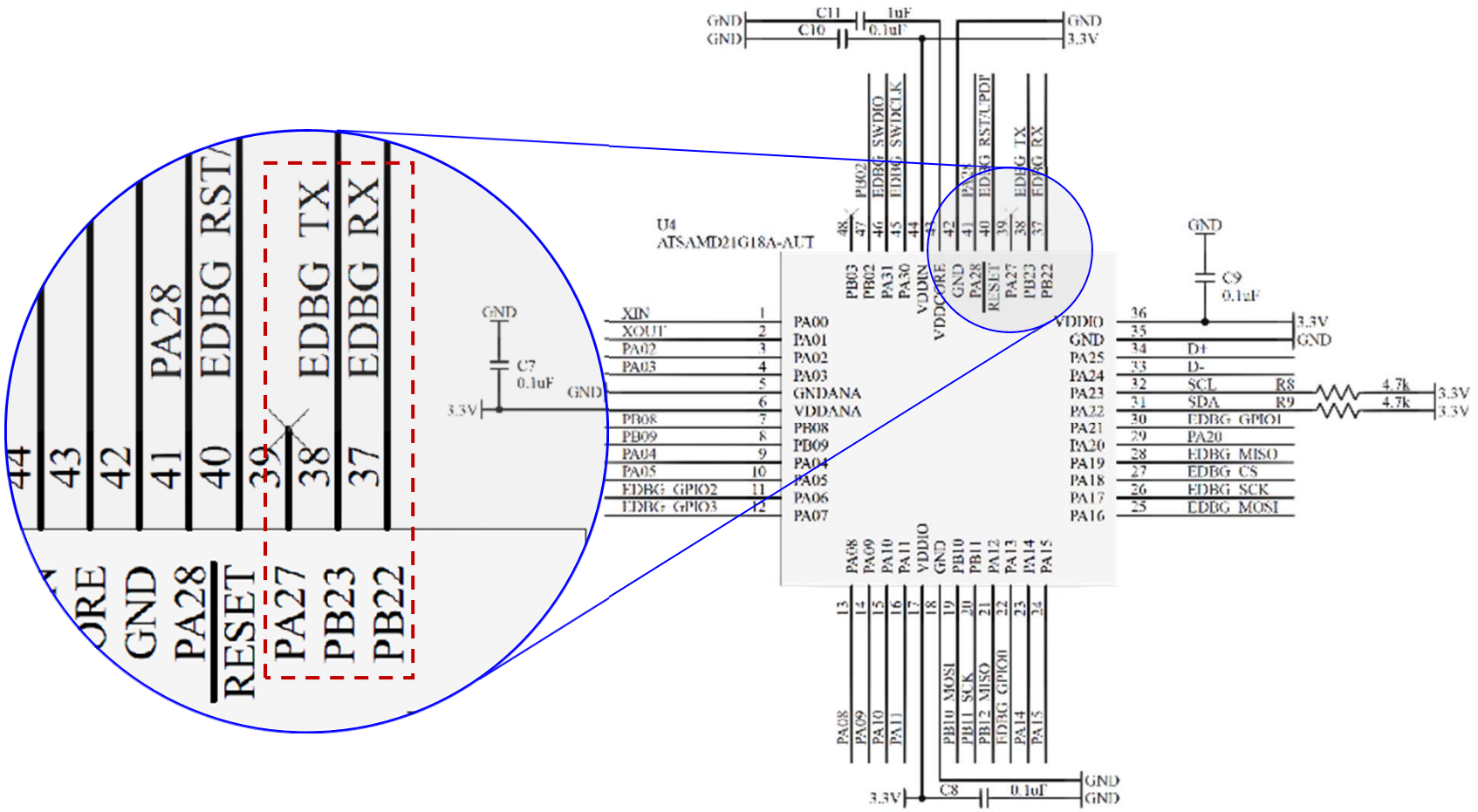
© 2008 Microchip Technology Incorporated. All Rights Reserved.

ASF USART Driver Functions

- /* Initializes the device to predefined defaults. */
  void usart_get_config_defaults(struct usart_config *const config)
- /* Initializes the device. */
  enum status_code usart_init
  (struct usart_module *const module,Sercom *const hw,
  const struct usart_config *const config)
- /* Enable the module. */
  void usart_enable(const struct usart_module *const module)
- /* Read/Write Function. */
  enum status_code usart_write_wait
  (struct usart_module *const module,const uint16_t tx_data)
  enum status_code usart_read_wait
  (struct usart_module *const module,uint16_t *const rx_data)
  enum status_code usart_write_buffer_wait
  (struct usart_module *const module,
  const uint8_t * tx_data,uint16_t length)
  enum status_code usart_read_buffer_wait
  (struct usart_module *const module,
  uint8_t * rx_data,uint16_t length)

© 2008 Microchip Technology Incorporated. All Rights Reserved.

APP-ESS17-2 SERCOM Schematic



© 2008 Microchip Technology Incorporated. All Rights Reserved.

ASF USART Tx Polling Code Example

```
struct usart_module USART_Instance;
struct usart_config USART_Config;
…

usart_get_config_defaults(&USART_Config);
USART_Config.baudrate = 9600;
USART_Config.mux_setting = USART_RX_3_TX_2_XCK_3;
USART_Config.pinmux_pad0 = PINMUX_UNUSED;
USART_Config.pinmux_pad1 = PINMUX_UNUSED;
USART_Config.pinmux_pad2 = PINMUX_PB22D_SERCOM5_PAD2;
USART_Config.pinmux_pad3 = PINMUX_PB23D_SERCOM5_PAD3;
usart_init(&USART_Instance, SERCOM5, &USART_Config);

usart_enable(&USART_Instance);

while(1)
{
   usart_write_buffer_wait(&USART_Instance, “Hello/r/n”, 7);
}
```

© 2008 Microchip Technology Incorporated. All Rights Reserved.

SERCOM PINMUX for USART

```
USART_Config.pinmux_pad2 = PINMUX_PB22D_SERCOM5_PAD2;
USART_Config.pinmux_pad3 = PINMUX_PB23D_SERCOM5_PAD3;
```

MUX	0	1	2	3	4	5	6	7

Pin(1)			I/O Pin	Supply	A	B(2)(3)					C	D	E	F	G	H
SAMD21E	SAMD21G	SAMD21J			EIC	REF	ADC	AC	PTC	DAC	SERCOM(2)(3)	SERCOM-ALT	TC(4) /TCC	TCC	COM	AC/ GCLK
15	23	31	PA14	VDDIO	EXTINT[14]						SERCOM2/ PAD[2]	SERCOM4/ PAD[2]	TC3/WO[0]	TCC0/ WO[4]		GCLK_IO[0]
16	24	32	PA15	VDDIO	EXTINT[15]						SERCOM2/ PAD[3]	SERCOM4/ PAD[3]	TC3/WO[1]	TCC0/ WO[5]		GCLK_IO[1]
	37	49	PB22	VDDIO	EXTINT[6]							SERCOM5/ PAD[2]	TC7/WO[0]			GCLK_IO[0]
	38	50	PB23	VDDIO	EXTINT[7]							SERCOM5/ PAD[3]	TC7/WO[1]			GCLK_IO[1]

samd21G18a.h

```
#define PIN_PB22D_SERCOM5_PAD2 54L /* 32(PORTA) + 22 = 54 */
#define MUX_PB22D_SERCOM5_PAD2 3L
#define PINMUX_PB22D_SERCOM5_PAD2
                ((PIN_PB22D_SERCOM5_PAD2 << 16) | MUX_PB22D_SERCOM5_PAD2)
#define PIN_PB23D_SERCOM5_PAD3 55L /* 32(PORTA) + 23 = 55 */
#define MUX_PB23D_SERCOM5_PAD3 3L
#define PINMUX_PB23D_SERCOM5_PAD3
                ((PIN_PB22D_SERCOM5_PAD3 << 16) | MUX_PB22D_SERCOM5_PAD3)
```

© 2008 Microchip Technology Incorporated. All Rights Reserved.

SERCOM PINMUX for USART

```
USART_Config.mux_setting = USART_RX_3_TX_2_XCK_3;
```

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0	RUNSTDBY			MODE[2:0]			ENABLE	SWRST
0x01		15:8	SAMPR[2:0]							IBON
0x02		23:16	SAMPA[1:0]		RXPO[1:0]				TXPO[1:0]	
0x03		31:24		DORD	CPOL	CMODE	FORM[3:0]			

RXPO[1:0]	Name	Description
0x0	PAD[0]	SERCOM PAD[0] is used for data reception
0x1	PAD[1]	SERCOM PAD[1] is used for data reception
0x2	PAD[2]	SERCOM PAD[2] is used for data reception
0x3	PAD[3]	SERCOM PAD[3] is used for data reception

TXPO	TxD Pin Location	XCK Pin Location (When Applicable)	RTS	CTS
0x0	SERCOM PAD[0]	SERCOM PAD[1]	N/A	N/A
0x1	SERCOM PAD[2]	SERCOM PAD[3]	N/A	N/A
0x2	SERCOM PAD[0]	N/A	SERCOM PAD[2]	SERCOM PAD[3]
0x3	Reserved			

Pin(1)			I/O Pin	Supply	A	B(2)(3)					C	D	E	F	G	H
SAMD21E	SAMD21G	SAMD21J			EIC	REF	ADC	AC	PTC	DAC	SERCOM(2)(3)	SERCOM-ALT	TC(4) /TCC	TCC	COM	AC/ GCLK
15	23	31	PA14	VDDIO	EXTINT[14]						SERCOM2/ PAD[2]	SERCOM4/ PAD[2]	TC3/WO[0]	TCC0/ WO[4]		GCLK_IO[0]
16	24	32	PA15	VDDIO	EXTINT[15]						SERCOM2/ PAD[3]	SERCOM4/ PAD[3]	TC3/WO[1]	TCC0/ WO[5]		GCLK_IO[1]
	37	49	PB22	VDDIO	EXTINT[6]							SERCOM5/ PAD[2]	TC7/WO[0]	Tx : 0x01		GCLK_IO[0]
	38	50	PB23	VDDIO	EXTINT[7]							SERCOM5/ PAD[3]	TC7/WO[1]	Rx : 0x03		GCLK_IO[1]

© 2008 Microchip Technology Incorporated. All Rights Reserved.

SERCOM PINMUX for USART

```
USART_Config.mux_setting = USART_RX_3_TX_2_XCK_3;
```

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0	RUNSTDBY			MODE[2:0]			ENABLE	SWRST
0x01		15:8	SAMPR[2:0]							IBON
0x02		23:16	SAMPA[1:0]		RXPO[1:0]				TXPO[1:0]	
0x03		31:24		DORD	CPOL	CMODE	FORM[3:0]			

Rx : 0x03

Tx : 0x01

usart.h

```
USART_RX_3_TX_2_XCK_3 =
        (SERCOM_USART_CTRLA_RXPO(3) | SERCOM_USART_CTRLA_TXPO(1))
```

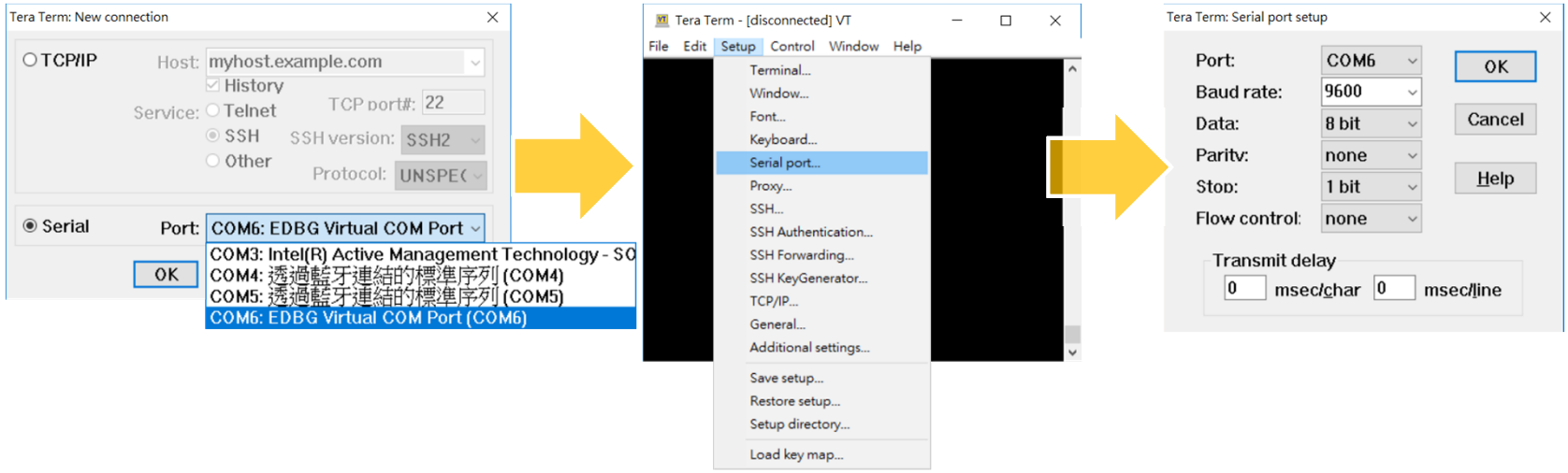
sercom.h

```
#define SERCOM_USART_CTRLA_RXPO_Pos 20
#define SERCOM_USART_CTRLA_RXPO_Msk (0x3ul << SERCOM_USART_CTRLA_RXPO_Pos)

#define SERCOM_USART_CTRLA_RXPO(value)
    (SERCOM_USART_CTRLA_RXPO_Msk & ((value) << SERCOM_USART_CTRLA_RXPO_Pos))

#define SERCOM_USART_CTRLA_TXPO_Pos ….
```

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Terminal Software

- Tera Term is a terminal software. We receive and transmit through Tera Term. (http://ttssh2.sourceforge.jp/index.html.en)
- Select COMx EDBG Virtual COM Port
- Set 9600 , N , 8 , 1.



© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab7 SERCOM - UART Tx Polling

- Try to add SERCOM - UART function to project.
- UART set to 9600, N, 8 , 1, No flow control.
- Send string from UART to PC one time per second continuously. (Hello World ??)

- How to start ?

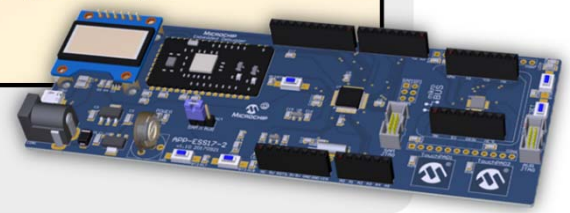
© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab7 SERCOM - UART Tx Polling

Step

a **Add below code segment to your main loop**

```
struct usart_module USART5_Instance;
struct usart_config USART_Config;

usart_get_config_defaults(&USART_Config);
USART_Config.baudrate = 9600;
USART_Config.mux_setting = USART_RX_3_TX_2_XCK_3;
USART_Config.pinmux_pad0 = PINMUX_UNUSED;
USART_Config.pinmux_pad1 = PINMUX_UNUSED;
USART_Config.pinmux_pad2 = PINMUX_PB22D_SERCOM5_PAD2;
USART_Config.pinmux_pad3 = PINMUX_PB23D_SERCOM5_PAD3;
usart_init(&USART5_Instance, SERCOM5, &USART5_Config);
usart_enable(&USART5_Instance);

while(1)
{
   if(TC3_IsOverflow)
   {
      TC3_IsOverflow = 0;
      ...
      usart_write_buffer_wait(&USART5_Instance, "Hello World!!\r\n", 15);
   }
}
```

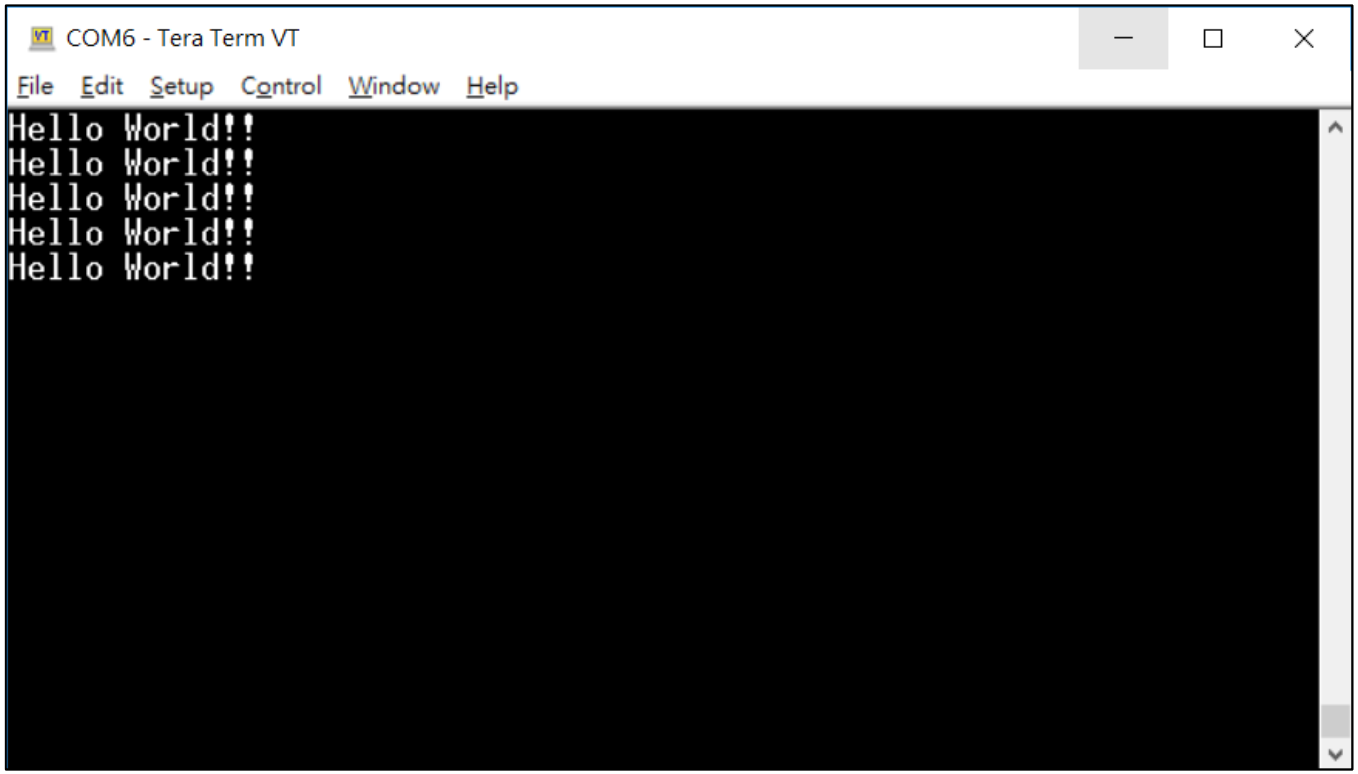


b **Program firmware to target board then observe result.**

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab7 SERCOM - UART Tx Polling

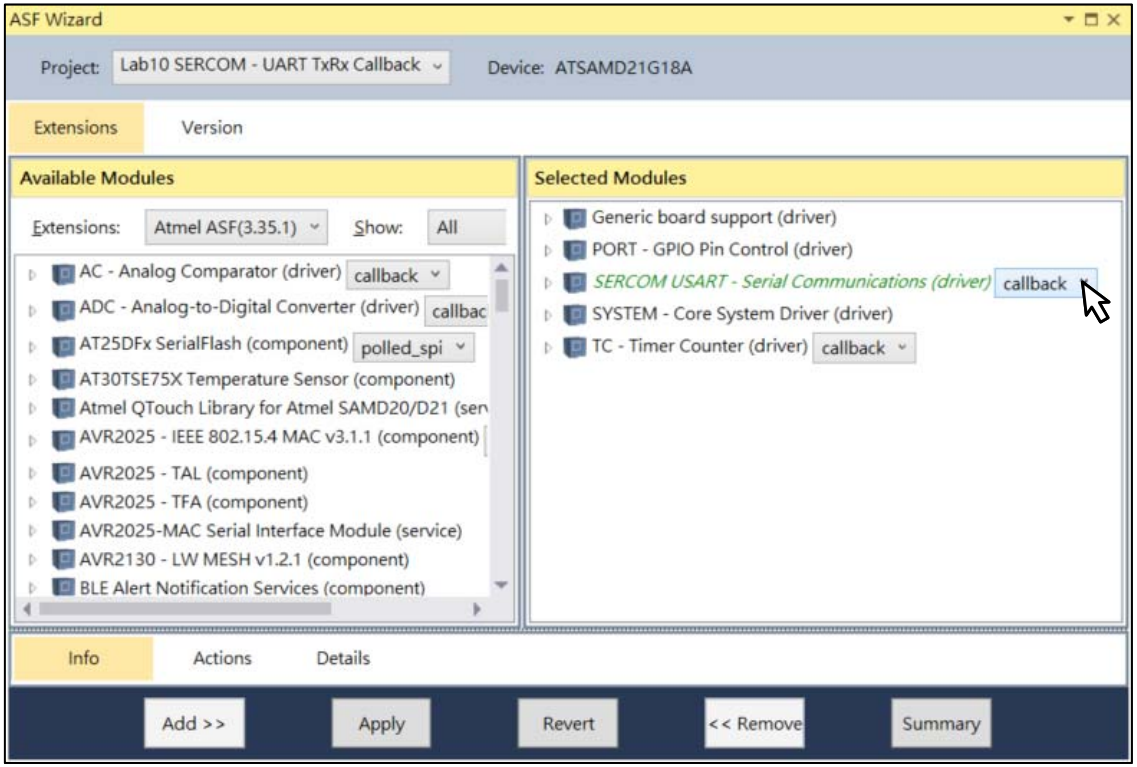
Result



© 2008 Microchip Technology Incorporated. All Rights Reserved.

Callback Function

- ASF SERCOM driver provide callback function, also.
- Please change ASF driver from polling to callback at ASF wizard.



© 2008 Microchip Technology Incorporated. All Rights Reserved.

ASF USART Driver Callback Functions

- /* Register callback functions with the driver */
  void usart_register_callback(struct usart_module *const module,
  usart_callback_t callback_func, enum usart_callback callback_type)
- /* Enable callback. */
  void usart_enable_callback(struct usart_module *const module,
  enum usart_callback callback_type)
- /* Asynchronous Read/Write Function. */
  enum status_code usart_write_job
  (struct usart_module *const module,const uint16_t tx_data)
  enum status_code usart_read_ job
  (struct usart_module *const module,uint16_t *const rx_data)
  enum status_code usart_write_buffer_ job
  (struct usart_module *const module,
  const uint8_t * tx_data,uint16_t length)
  enum status_code usart_read_buffer_ job
  (struct usart_module *const module,
  uint8_t * rx_data,uint16_t length)
- /* Status check. */
  enum status_code usart_get_job_status(
  struct usart_module *const module,
  enum usart_transceiver_type transceiver_type)

Asynchronous means is sets up the driver to read/write the data given.
If registered and enabled, a callback function will be called when the receive/transmit is completed.

© 2008 Microchip Technology Incorporated. All Rights Reserved.

UART Callback Example

UART Initial (Hook callback function)

```
usart_register_callback(&USART5_Instance, USART5_Receive,
                                          USART_CALLBACK_BUFFER_RECEIVED);
usart_enable_callback(&USART5_Instance, USART_CALLBACK_BUFFER_RECEIVED);
usart_enable(&USART5_Instance);

void USART5_Receive(struct usart_module * const Instance) /* CallBack */
{
   USART5_IsReceived = 1;
}
```

```
usart_read_job(&USART5_Instance, &USART5_ReceiveData);
while(1)
{
   if (USART5_IsReceived)
   {
      USART5_IsReceived = 0;
      printf("%c",USART5_ReceiveData); // Dummy Code
      usart_read_job(&USART5_Instance, &USART5_ReceiveData);
   }
}
```

One-shot event!

UART Read byte

© 2008 Microchip Technology Incorporated. All Rights Reserved.

About String Function

- UART 所處理的資料大多是字串或文字資料, 使用C所提供的字串處理函式將能更方便的進行“資料” <-> “字串”的轉換。常用的轉換函式有sprintf( ), strlen( )。
- int sprintf(char *s, const char *format, ...);
  將資料依據指定格式轉換後的字串存入指定的buffer中。
  * 必需 include <stdio.h>
  size_t strlen(const char *s);
  計算字串(string)的長度(不包含字串結尾的的0x00(‘\0’))。
  * 必需 include <string.h>
- For Example

```
uint8_t USARTRTxBuffer[100];
uint16_t USART5_ReceiveData;
sprintf((char *) USARTRTxBuffer, "Received Data : %1c\r\n", USART5_ReceiveData);
usart_write_buffer_job(&USART5_Instance, USARTRTxBuffer, strlen((char *)USARTRTxBuffer));
```

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab8 SERCOM - UART TxRx Callback

- Try to change SERCOM - UART to Callback mode driver.
- Modify polling transmit function to asynchronous function.
- Add new received function base on callback mode.
- Try to receive character from PC and response to PC after received. Like this “Received Data : …” .
- UART setting same as lab9

- **Let’s go!**

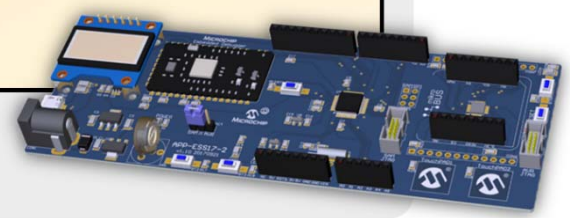
© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab8 SERCOM - UART TxRx Callback

Step1

a **Add below code segment for initial**

```
void USART5_Receive(struct usart_module * const);

uint8_t USARTRTxBuffer[100];
uint16_t USART5_ReceiveData;
uint8_t USART5_IsReceived = 0;

int main (void)
{
   usart_register_callback(&USART5_Instance, USART5_Receive,
                                                USART_CALLBACK_BUFFER_RECEIVED);
   usart_enable_callback(&USART5_Instance, USART_CALLBACK_BUFFER_RECEIVED);
   usart_enable(&USART5_Instance);

   usart_read_job(&USART5_Instance, &USART5_ReceiveData);
   …

}

void USART5_Receive(struct usart_module * const Instance) /* CallBack */
{
   USART5_IsReceived = 1;
}
```



© 2008 Microchip Technology Incorporated. All Rights Reserved.

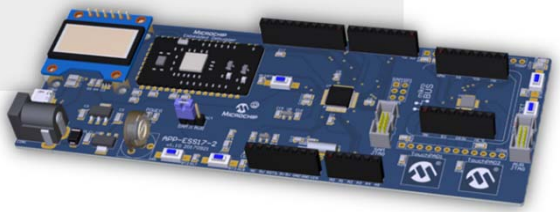
Lab8 SERCOM - UART TxRx Callback

Step2

a **Add below code segment to your main loop**

```
int main (void)
{
   while(1)
   {
      if (USART5_IsReceived)
      {
         USART5_IsReceived = 0;
         sprintf((char *) USARTRTxBuffer, "Received Data : %1c\r\n",
                                                    USART5_ReceiveData);

         usart_write_buffer_job(&USART5_Instance,
                                    USARTRTxBuffer, strlen((char *) USARTRTxBuffer));

         usart_read_job(&USART5_Instance, &USART5_ReceiveData);
      }
   }
}
```

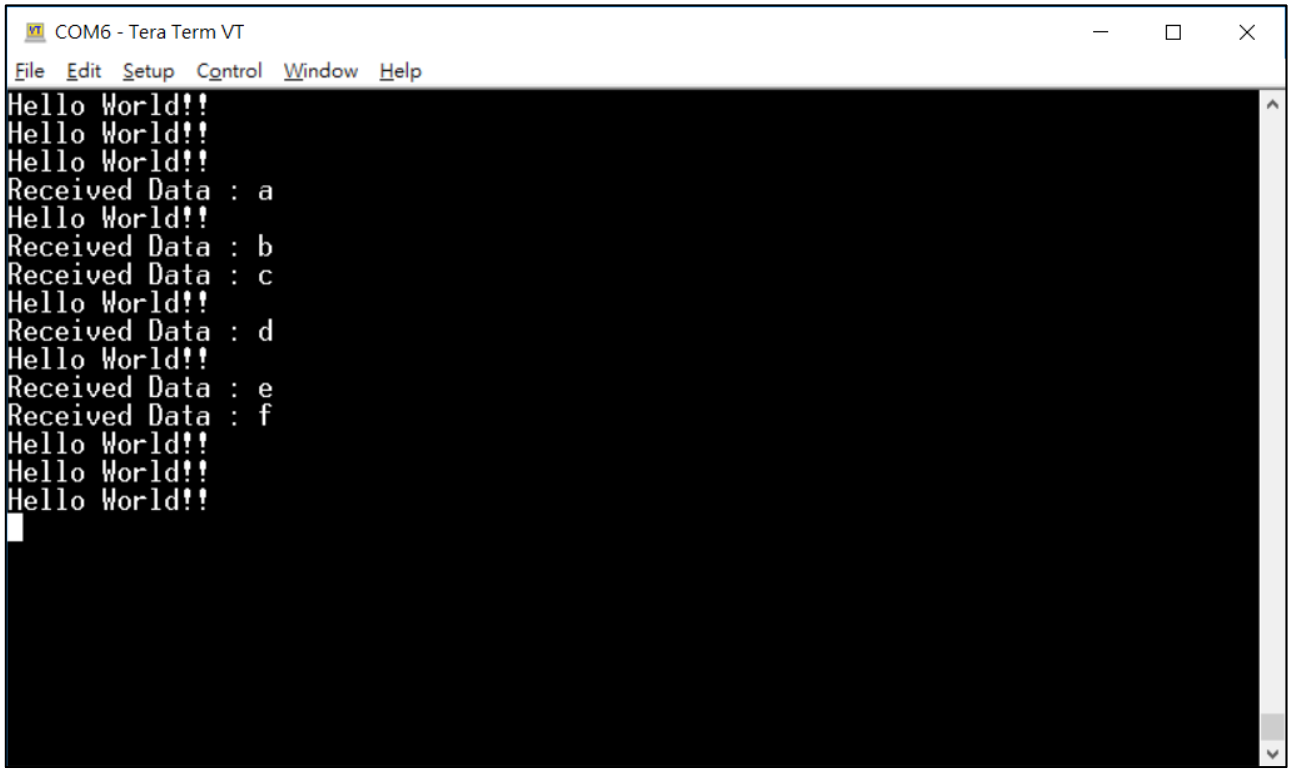
b **Program firmware to target board then observe result.**



© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab8 SERCOM - UART TxRx Callback

Result



© 2008 Microchip Technology Incorporated. All Rights Reserved.

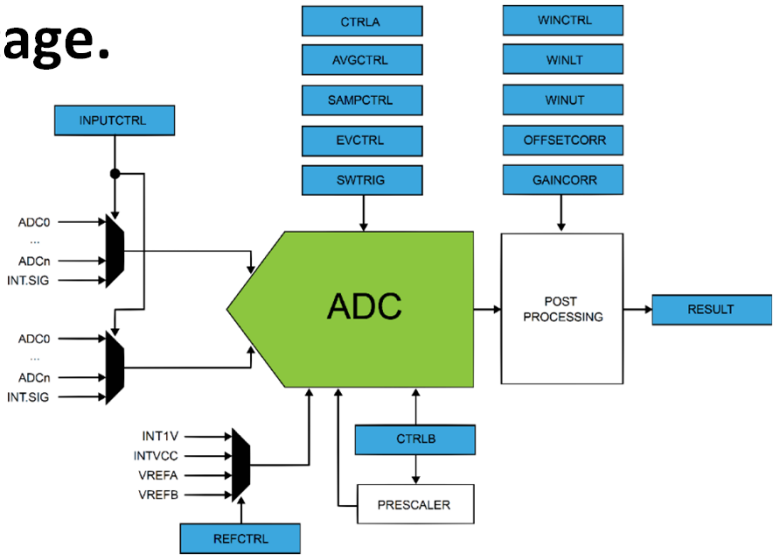
High Speed ADC Architecture & Event System

© 2008 Microchip Technology Incorporated. All Rights Reserved.

ADC Block Diagram

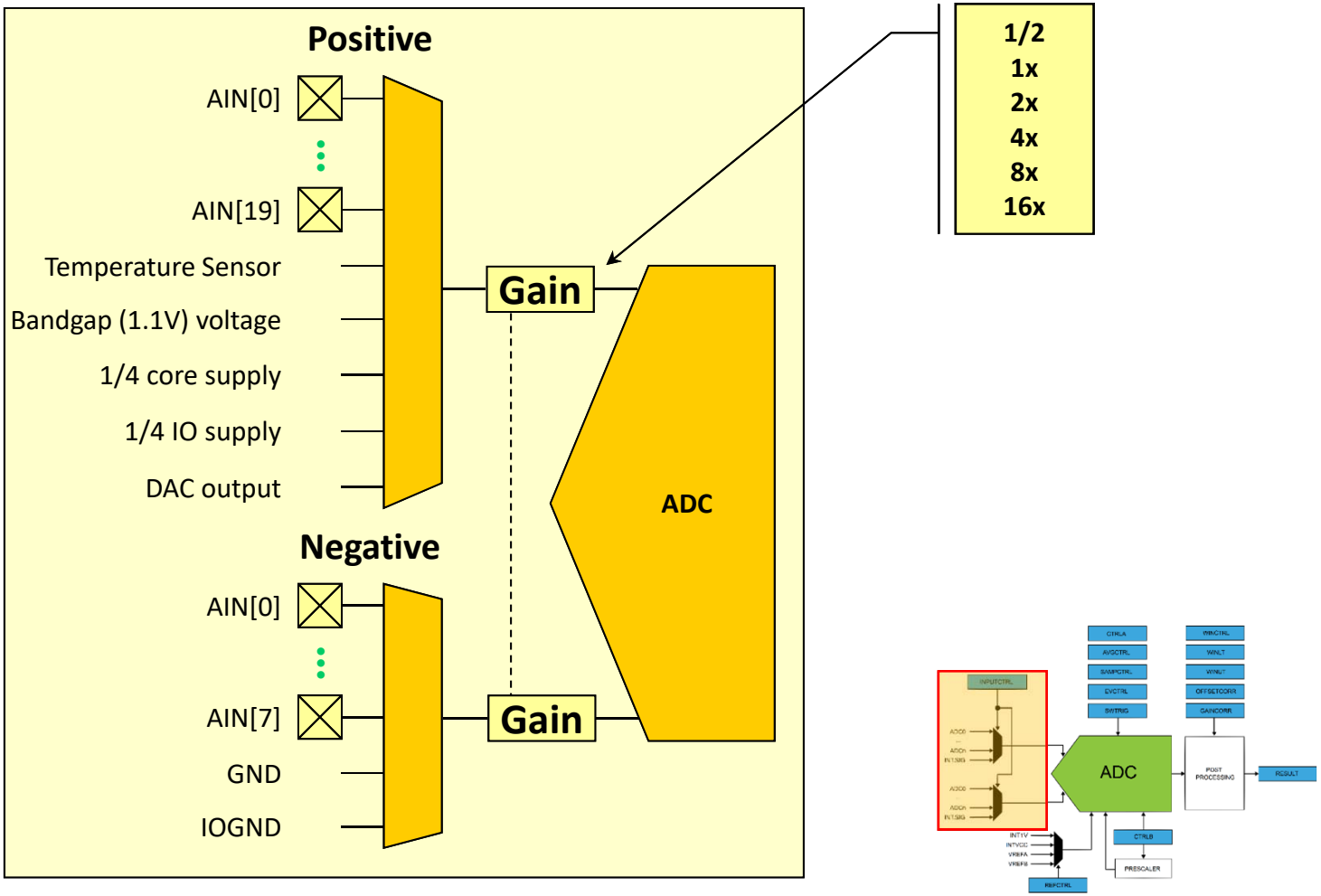
- Up to 32 analog input, 8, 10 or 12 bits resolution.
- Differential and single-ended inputs.
- 1/2x to 16x gain. Up to 350ksps.
- Support Event-triggered and pin scan.
- Support averaging and oversampling.
- External or Internal reference voltage.
- Oversample accumulation average and window monitor feature.

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



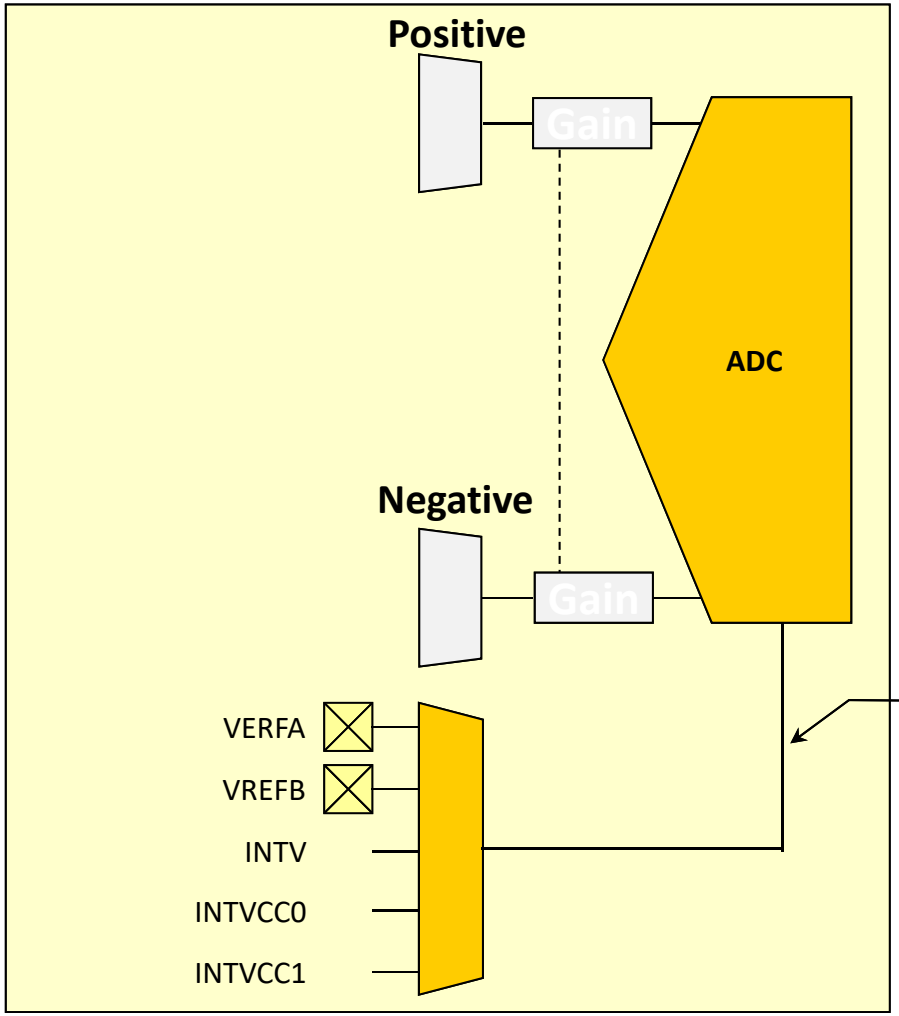
© 2008 Microchip Technology Incorporated. All Rights Reserved.

ADC Channel Selection & Gain



© 2008 Microchip Technology Incorporated. All Rights Reserved.

ADC Reference Voltage Selection



REFSEL[3:0]	Name	Description
0x0	INT1V	1.0V voltage reference
0x1	INTVCC0	1/1.48 VDDANA
0x2	INTVCC1	1/2 VDDANA (only for VDDANA > 2.0V)
0x3	VREFA	External reference
0x4	VREFB	External reference
0x5-0xF		Reserved

Conversion range:

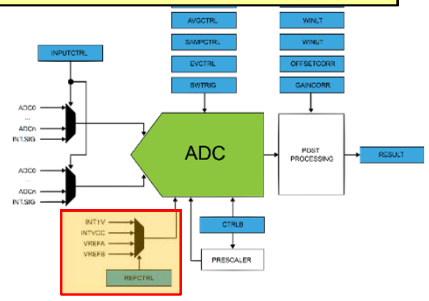
– VREF(A/B) [1.0V to (VDDANA - 0.6V)]

– Single-ended mode

[0V to +VREF/GAIN]

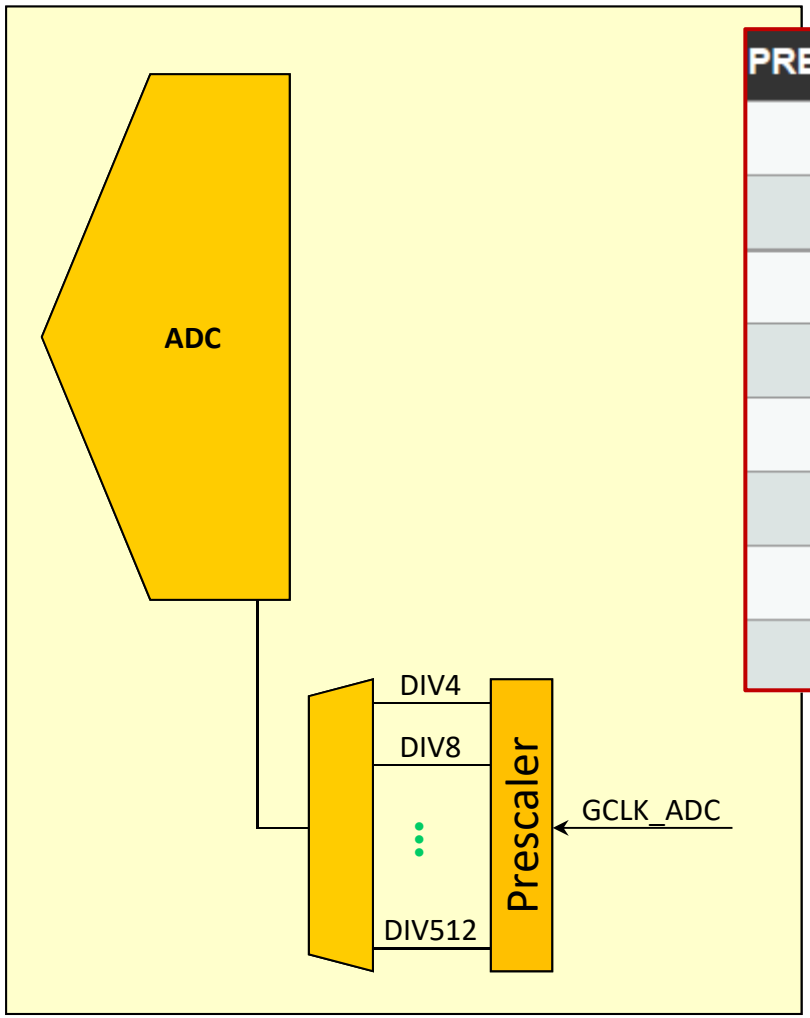
– Differential mode

[-VREF/GAIN to +VREF/GAIN]

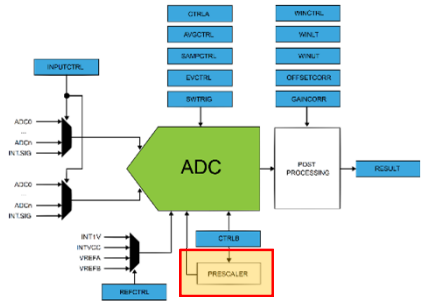


© 2008 Microchip Technology Incorporated. All Rights Reserved.

ADC Prescaler

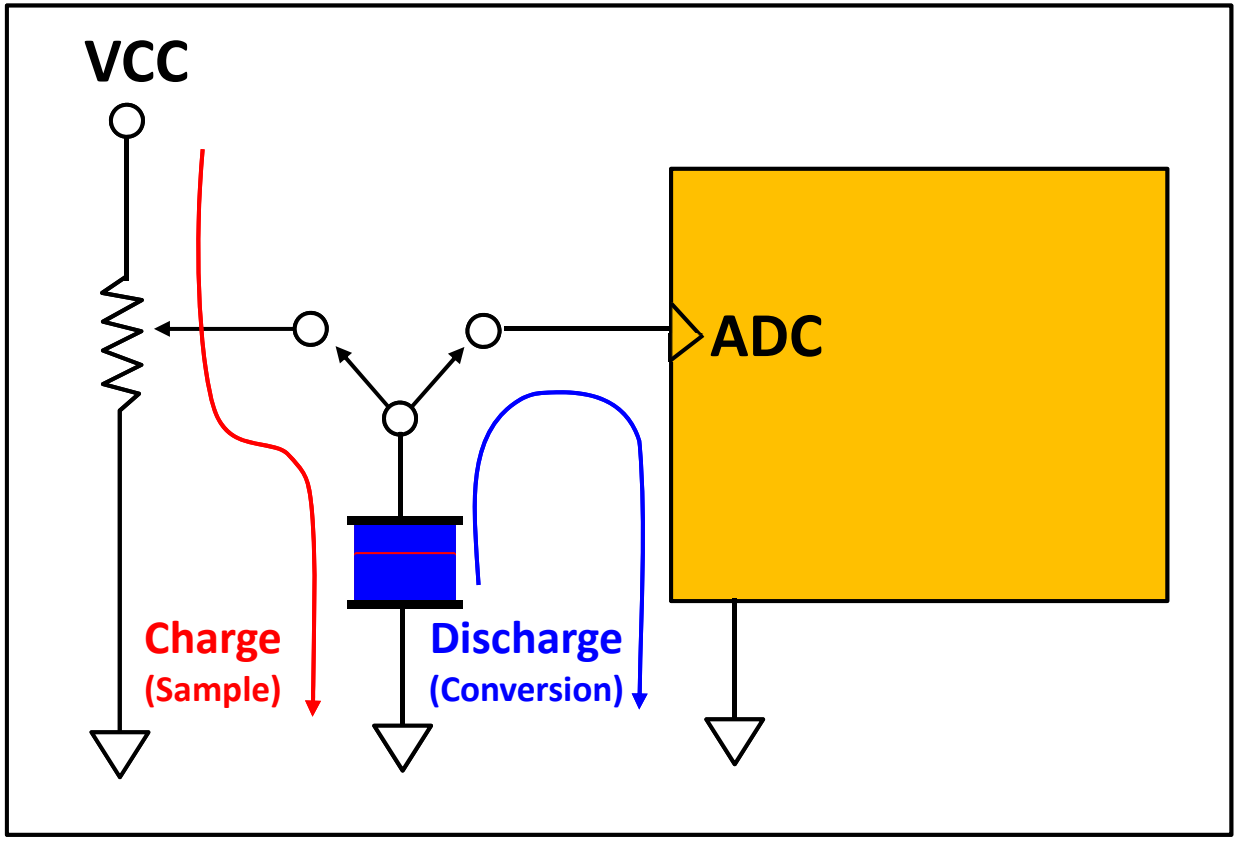


PRESCALER[2:0]	Name
0x0	DIV4
0x1	DIV8
0x2	DIV16
0x3	DIV32
0x4	DIV64
0x5	DIV128
0x6	DIV256
0x7	DIV512



© 2008 Microchip Technology Incorporated. All Rights Reserved.

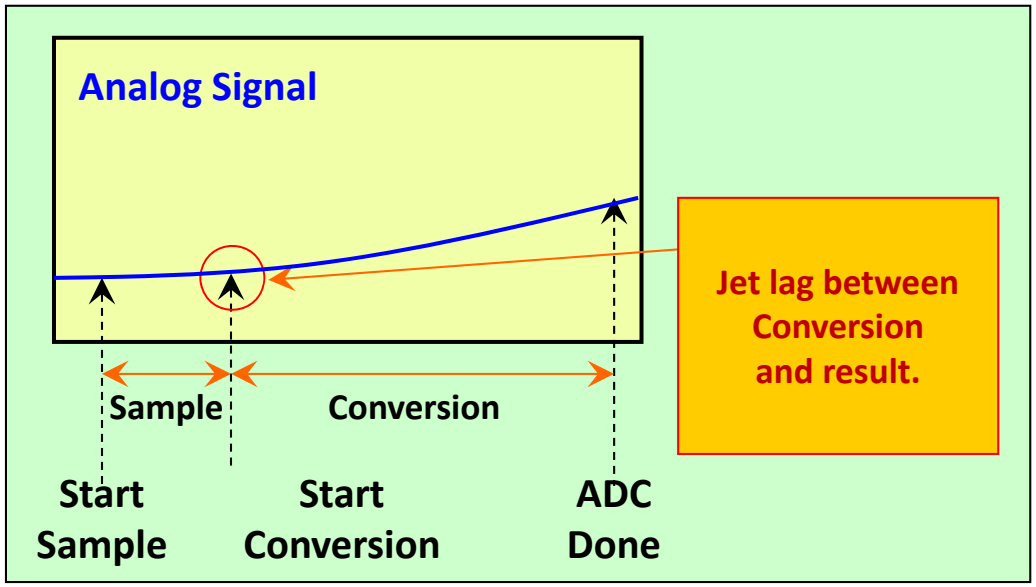
Sample and Conversion Sequence



© 2008 Microchip Technology Incorporated. All Rights Reserved.

Sample and Conversion Sequence

- The ADC sample and conversion time are requirements from the specification.
- The sample time must greater than 1 edge and conversion time need 13 edges (12 Bits). The maximize ADC clock frequency is 2.1MHz.



© 2008 Microchip Technology Incorporated. All Rights Reserved.

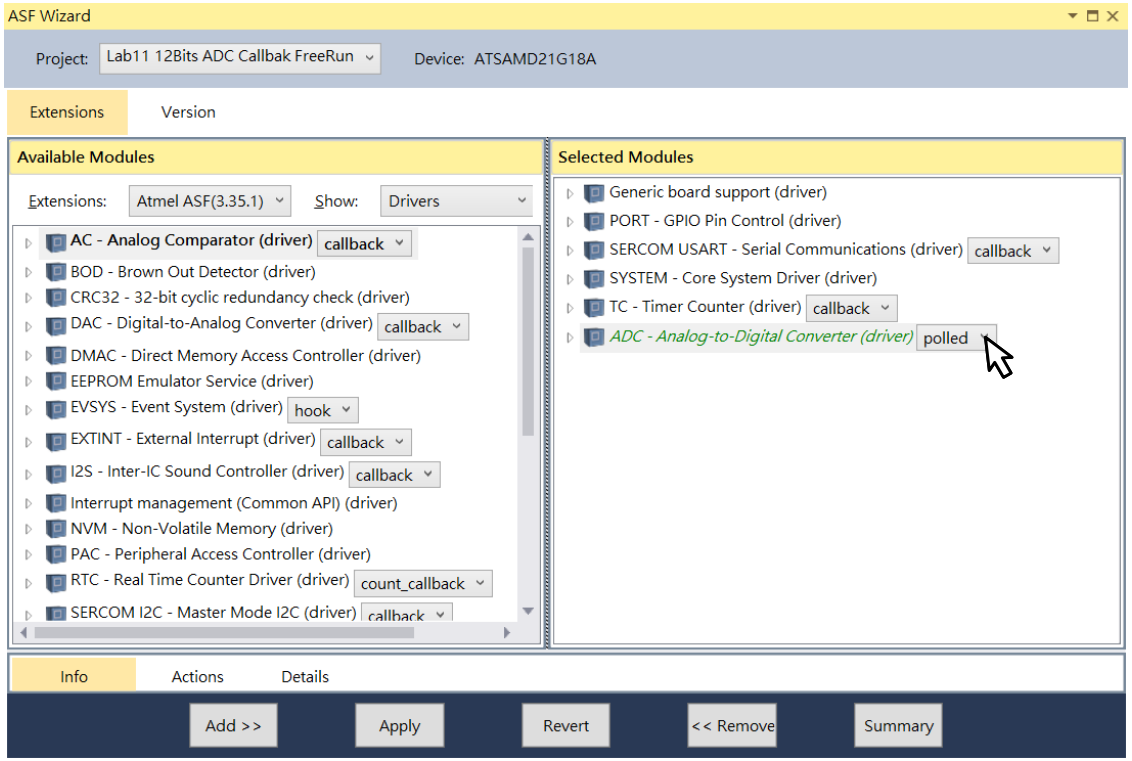
ADC Trigger Source

- **There are three way to be started data conversion.**
  - **Software**
    setting the Start bit SWTRIG.START.
  - **Free Running**
    A free-running mode can be used to continuously convert an input channel. When using free-running mode the first conversion must be started, while subsequent conversions will start automatically at the end of previous conversions.
  - **Automatically**
    Through Event System.

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Add SERCOM Function from ASF

- Select **ASF ▸ ASF Wizard**, Add **ADC .. polled** to project.



© 2008 Microchip Technology Incorporated. All Rights Reserved.

ASF ADC Driver Functions

- /* Initializes an ADC configuration structure to defaults. */
  void adc_get_config_defaults(struct adc_config *const config)
- /* Initializes the ADC. */
  enum status_code adc_init(struct adc_module *const module_inst,
  Adc * hw, struct adc_config * config)
- /* Enable the module. */
  enum status_code adc_enable
  (struct adc_module *const module_inst)
- /* Software Starts an ADC conversion.*/
  void adc_start_conversion(struct adc_module *const module_inst)
- /* Reads the ADC result.*/
  enum status_code adc_read
  (struct adc_module *const module_inst,uint16_t * result)
- /* Sets positive/negative ADC input pin.*/
  void adc_set_positive_input(struct adc_module *const module_inst,
  const enum adc_positive_input positive_input)
  void adc_set_negative_input(struct adc_module *const
  module_inst,const enum adc_negative_input negative_input)

© 2008 Microchip Technology Incorporated. All Rights Reserved.

ASF ADC Polling, Software Trigger Code Example

```
struct adc_module ADC_Instance;
struct adc_config ADC_Config;
uint16_t ADC_Result;

adc_get_config_defaults(&ADC_Config);
ADC_Config.reference = ADC_REFERENCE_INTVCC1; /* 1/2 VDDA */
ADC_Config.clock_prescaler = ADC_CLOCK_PRESCALER_DIV512;
ADC_Config.positive_input = ADC_POSITIVE_INPUT_PIN5;
adc_init(&ADC_Instance, ADC, &ADC_Config);
adc_enable(&ADC_Instance);

while(1)
{
   adc_start_conversion(& ADC_Instance);
   while (adc_read(&ADC_Instance, &ADC_Result) == STATUS_BUSY);
   sprintf((char *) USARTRTxBuffer, "ADC Value : %4d\r\n", ADC_Result);
   usart_write_buffer_job(&USART5_Instance, USARTRTxBuffer, strlen((char *) USARTRTxBuffer));
}
```

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab9 12Bits ADC Polling SoftTrigger

- Try to add ADC function to project.
- Initial ADC and get the quantify value form VR1 one time per second ,then use UART send it to PC.
- ADC set to software trigger, 12 bits, Single-End mode.

- Let's go!

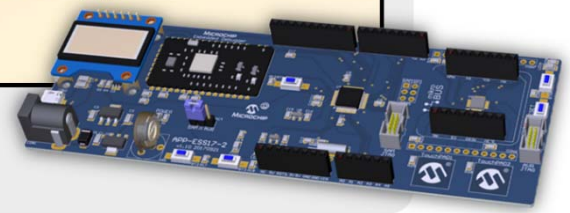
© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab9 12Bits ADC Polling SoftTrigger

Step

a **Add below code segment to your project**

```
struct adc_module ADC_Instance;
struct adc_config ADC_Config;
uint16_t ADC_Result;

adc_get_config_defaults(&ADC_Config);
ADC_Config.reference = ADC_REFERENCE_INTVCC1; /* 1/2 VDDA */
ADC_Config.clock_prescaler = ADC_CLOCK_PRESCALER_DIV512;
ADC_Config.positive_input = ADC_POSITIVE_INPUT_PIN5;
while (adc_init(&ADC_Instance, ADC, &ADC_Config) != STATUS_OK);
adc_enable(&ADC_Instance);

while(1)
{
   if(TC3_IsOverflow)
   {
      adc_start_conversion(& ADC_Instance);
      while (adc_read(&ADC_Instance, &ADC_Result) == STATUS_BUSY);
      sprintf((char *) USARTRTxBuffer, "ADC Value : %4d\r\n", ADC_Result);
      usart_write_buffer_job(&USART5_Instance,
                             USARTRTxBuffer, strlen((char *) USARTRTxBuffer));
   }
}
```



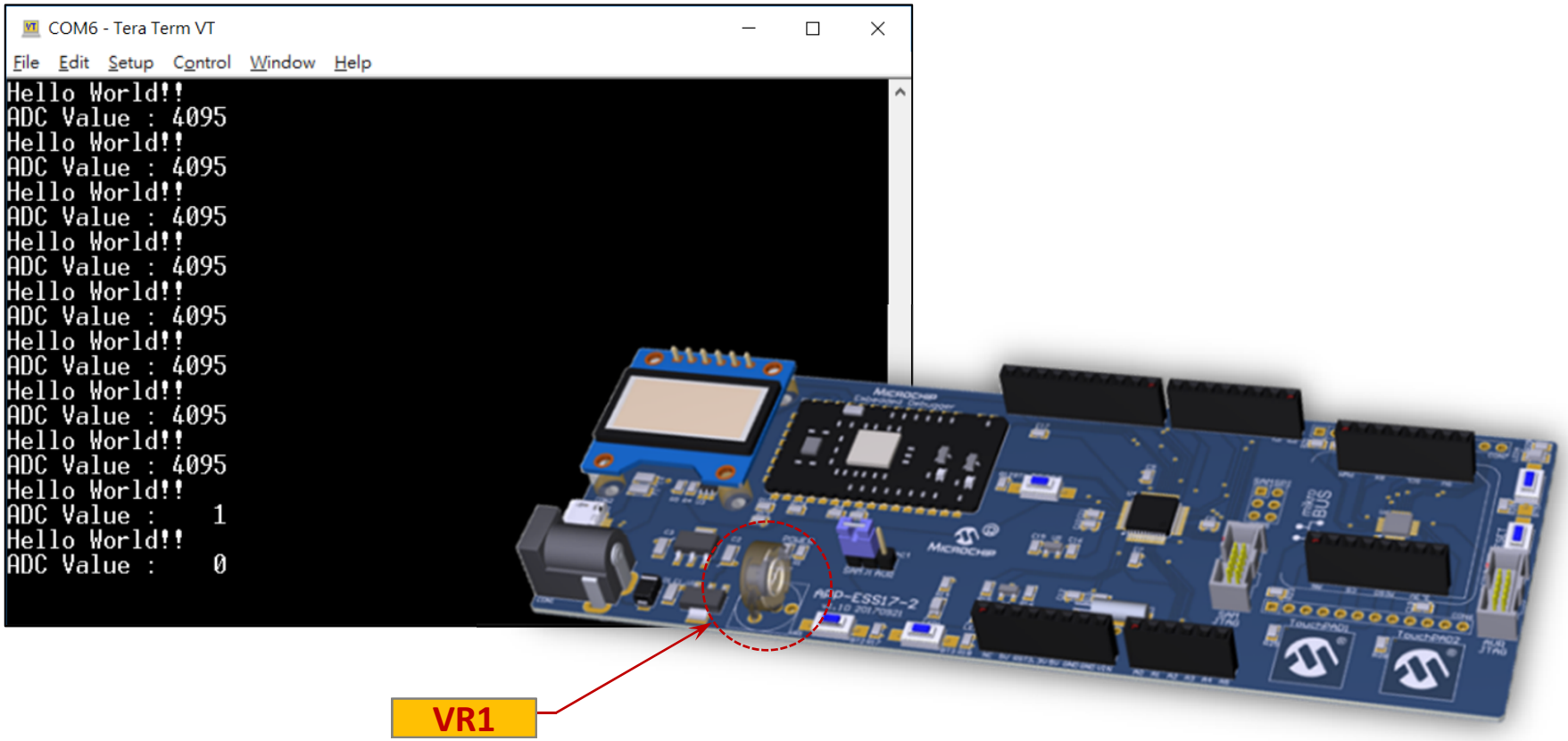
b **Program firmware to target board then observe result.**

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab9 12Bits ADC Polling SoftTrigger

Result

- Update the “Hello …” string and VR1 value one time per second at Terminal.



© 2008 Microchip Technology Incorporated. All Rights Reserved.

Pin Scan

- The ADC support analog channel switch call Pin Scan function.
- Pin scan switch analog channel sequential after a conversion automatically.
- Start Pin : MUXPOS + INPUTOFFSET
- End Pin : MUXPOS + INPUTOFFSET + INPUTSCAN
- For Example : Pin Scan form AIN[5] to AIN[10].

```
ADC_Config.positive_input = ADC_POSITIVE_INPUT_PIN5;
while (adc_init(&ADC_Instance, ADC, &ADC_Config) != STATUS_OK);
adc_set_pin_scan_mode(&ADC_Instance, 6, 0);
```

INPUTSCAN

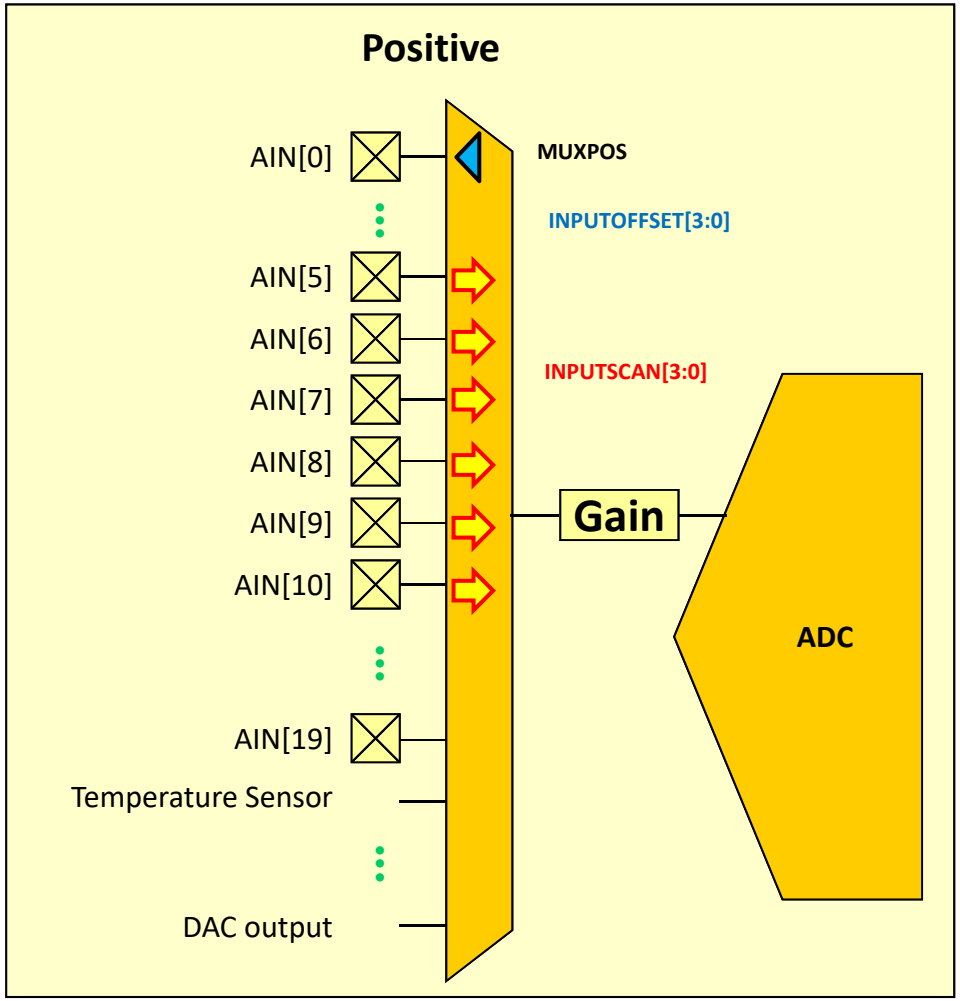
INPUTOFFSET

Or

```
ADC_Config.positive_input =
                    ADC_POSITIVE_INPUT_PIN5;
ADC_Config.pin_scan.offset_start_scan = 0;
ADC_Config.pin_scan.inputs_to_scan = 6;
while (adc_init(&ADC_Instance, ADC,
                    &ADC_Config) != STATUS_OK);
```

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Pin Scan



Start Pin :

MUXPOS + INPUTOFFSET

Ex.

MUXPOS=AIN[0]

INPUTOFFSET = 5

Start Pin = AIN[0+5]

or

MUXPOS=AIN[5]

INPUTOFFSET = 0

Start Pin = AIN[5+0]

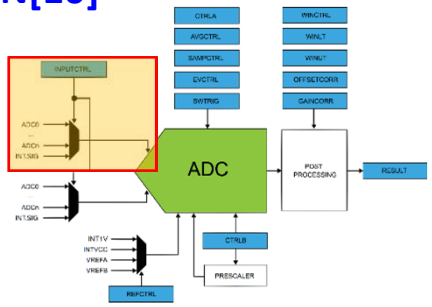
End Pin :

Start Pin + INPUTSCAN

Ex.

INPUTSCAN = 6

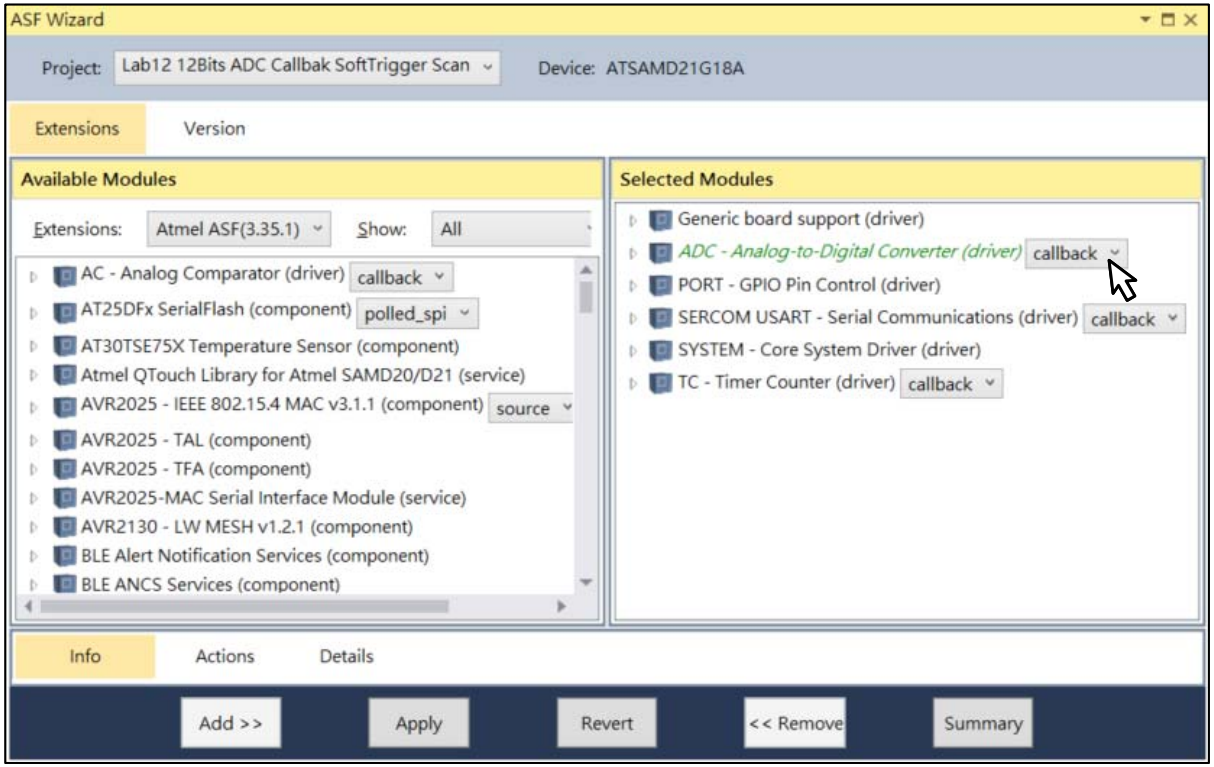
AIN[5] ~ AIN[10]



© 2008 Microchip Technology Incorporated. All Rights Reserved.

Callback Function

- ASF ADC driver provide callback function, also.
- Please change ASF driver from polling to callback at ASF wizard.



© 2008 Microchip Technology Incorporated. All Rights Reserved.

ASF ADC Driver Callback Functions

- /* Register callback functions with the driver */
```
void adc_register_callback(struct adc_module *const module,
                           callback_t callback_func,
                           enum adc_callback callback_type)
```

- /* Enable callback. */
```
void adc_enable_callback(struct adc_module *const module,
                         enum adc_callback callback_type)
```

- /* Read multiple samples from ADC. */
```
enum status_code adc_read_buffer_job
                 (struct adc_module *const module_inst,
                  uint16_t * buffer,uint16_t samples)
```

© 2008 Microchip Technology Incorporated. All Rights Reserved.

ASF ADC Callback, Pin Scan, Software Trigger Code Example

ADC Initial (Hook callback & set pin scan function)

```
uint16_t ADC_Result[6];
...
ADC_Config.pin_scan.offset_start_scan = 0; /* Start : 5 + 0 = 5 */
ADC_Config.pin_scan.inputs_to_scan = 6; /* Scan 6 channel AIN[5] ~ AIN[10] */
while (adc_init(&ADC_Instance, ADC, &ADC_Config) != STATUS_OK);
adc_register_callback(&ADC_Instance, ADC_Complete,
                                          ADC_CALLBACK_READ_BUFFER);
adc_enable_callback(&ADC_Instance, ADC_CALLBACK_READ_BUFFER);
void ADC_Complete(struct adc_module * const Instance) /* CallBack */
{
   ADC_IsCompleted = 1;
}
```

```
while(1)
{
   if(TC3_IsOverflow)
      adc_read_buffer_job(&ADC_Instance, ADC_Result, 6);
   if (ADC_IsCompleted)
   {
      ADC_IsCompleted = 0;
      VR1 = ADC_Result[0];
      MCP9700 = ADC_Result[5];
   }
}
```

ADC Read Array

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab10 12Bits ADC
Callback Scan SoftTrigger

- Change ADC polling to callback mode, enable pin scan function.
- get the quantify value form VR1 and temperature sensor one time per second ,then use UART send it to PC.
- ADC set same Lab9.

- Let's go!

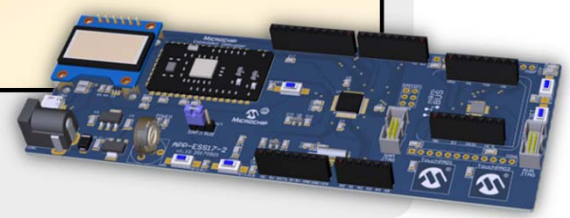
© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab10 12Bits ADC

Callback Scan SoftTrigger Step1

a Add below code segment for initial

```
uint16_t ADC_Result[6];
volatile uint8_t ADC_IsCompleted = 0;

int main (void)
{
    …

    ADC_Config.pin_scan.offset_start_scan = 0; /* Start : 5 + 0 = 5 */
    ADC_Config.pin_scan.inputs_to_scan = 6; /* Scan 6 channel AIN[5] ~ AIN[10] */
    while (adc_init(&ADC_Instance, ADC, &ADC_Config) != STATUS_OK);

    adc_register_callback(&ADC_Instance, ADC_Complete,
                                                ADC_CALLBACK_READ_BUFFER);
    adc_enable_callback(&ADC_Instance, ADC_CALLBACK_READ_BUFFER);
    …

}

void ADC_Complete(struct adc_module * const Instance) /* CallBack */
{
    ADC_IsCompleted = 1;
}
```



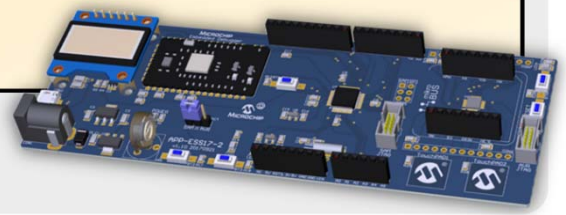
© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab10 12Bits ADC

Callback Scan SoftTrigger Step2

a **Add below code segment to your main loop**

```
int main (void)
{
   while(1)
   {
      if(TC3_IsOverflow)
      {
         TC3_IsOverflow = 0;
         adc_read_buffer_job(&ADC_Instance, ADC_Result, 6);
      }

      if (ADC_IsCompleted)
      {
         ADC_IsCompleted = 0;
         sprintf((char *) USARTRTxBuffer, "VR1 Value : %4d \r\n", ADC_Result[0]);
         usart_write_buffer_job(&USART5_Instance,
                                USARTRTxBuffer, strlen((char *) USARTRTxBuffer));
         sprintf((char *) USARTRTxBuffer, "Temperature Value: %4d\r\n", ADC_Result[5]);
         usart_write_buffer_job(&USART5_Instance,
                                USARTRTxBuffer, strlen((char *) USARTRTxBuffer));
      }
   }
}
```



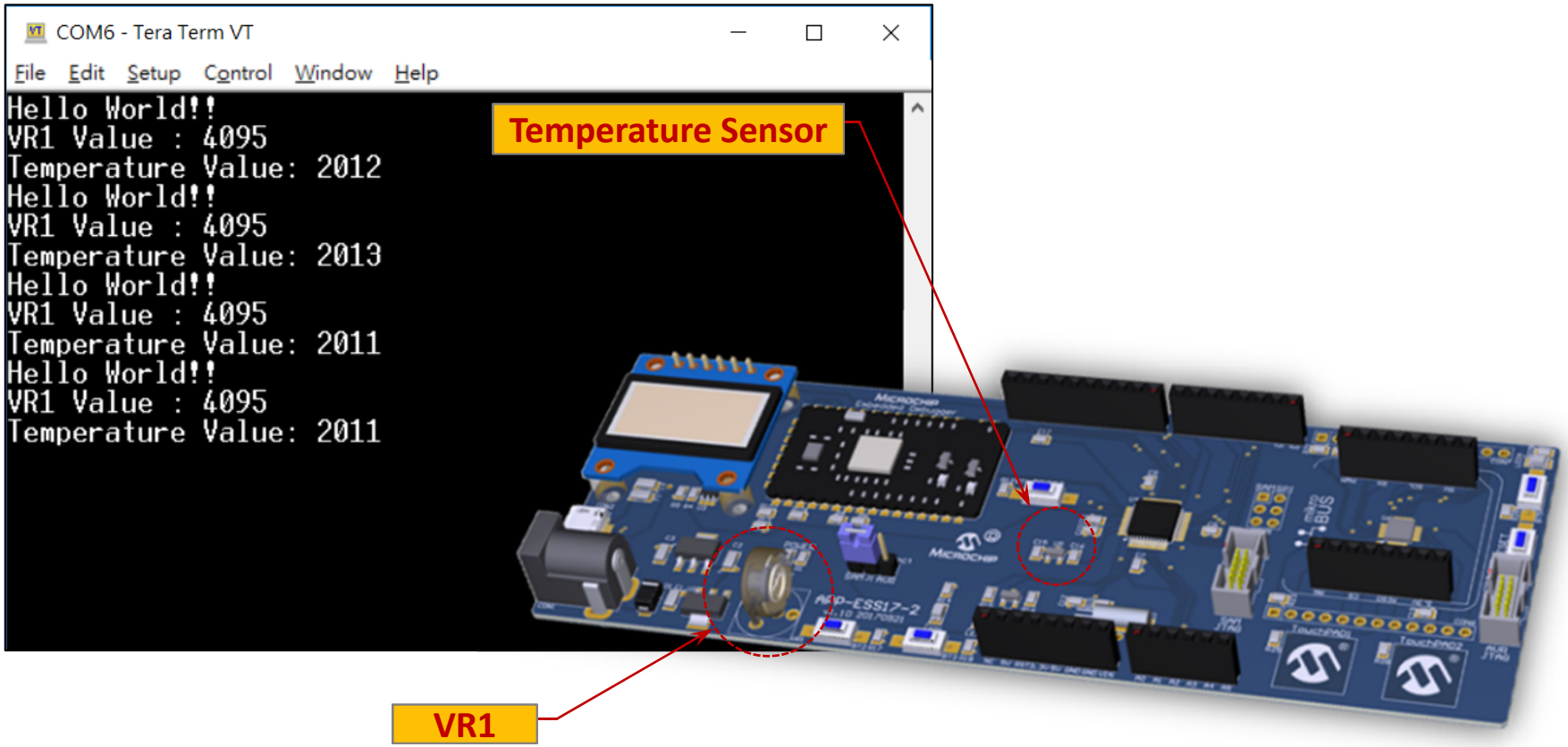
b **Program firmware to target board then observe result.**

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab10 12Bits ADC Polling SoftTrigger

Result

- Update the “Hello …” string VR1 and temperature sensor value one time per second at Terminal.



© 2008 Microchip Technology Incorporated. All Rights Reserved.

Accumulation and Averaging

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

$$\frac{\text{Sample 1} + \text{Sample 2} + \ldots\ldots\ldots + \text{Sample m}}{m}$$

Accumulation

Averaging

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Accumulation and Averaging

Example :

Number of Accumulated Samples : ∑ 256 (2^8 samples)
Intermediate Result precision : 20 bits (12bits + 8 = 20 bits overflow)
Number of Automatic Right Shift : 4 bits (20bits >> 4 = 16bits)
Adjust Division Factor : /16 (2^4)
Finial Result Precision : 12 bits (16bit >> 4 = 12 bits)

Number of Accumulated Samples	AVGCTRL. SAMPLENUM	Intermediate Result Precision	Number of Automatic Right Shifts	Automatic Division Factor	AVGCTRL. ADJRES	Adjust Division Factor	Total Number of Right Shifts	Final Result Precision
1	0x0	12 bits	0	0	0x0	1		12 bits
2	0x1	13 bits	0	0	0x1	2	1	12 bits
4	0x2	14 bits	0	0	0x2	4	2	12 bits
8	0x3	15 bits	0	0	0x3	8	3	12 bits
16	0x4	16 bits	0	0	0x4	16	4	12 bits
32	0x5	17 bits	1	2	0x4	16	5	12 bits
64	0x6	18 bits	2	4	0x4	16	6	12 bits
128	0x7	19 bits	3	8	0x4	16	7	12 bits
256	0x8	20 bits	4	16	0x4	16	8	12 bits
512	0x9	21 bits	5	32	0x4	16	9	12 bits
1024	0xA	22 bits	6	64	0x4	16	10	12 bits
Reserved	0xB –0xF							

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

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Accumulation and Averaging

Code example

Example :

Number of Accumulated Samples : ∑ 256 (2^8 samples)
Intermediate Result precision : 20 bits (12bits + 8 = 20 bits overflow)
Number of Automatic Right Shift : 4 bits (20bits >> 4 = 16bits)
Adjust Division Factor : /16 (2^4)
Finial Result Precision : 12 bits (16bit >> 4 = 12 bits)

```
adc_get_config_defaults(&ADC_Config);
ADC_Config. resolution = ADC_RESOLUTION_CUSTOM;
ADC_Config. accumulate_samples = ADC_ACCUMULATE_SAMPLES_256;
ADC_Config. divide_result = ADC_DIVIDE_RESULT_16;
```

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

```
enum adc_resolution {
……..
/** ADC 16-bit result register for use with averaging. When using this mode
  * the ADC result register will be set to 16-bit wide, and the number of
  * samples to accumulate and the division factor is configured by the
  * \ref adc_config.accumulate_samples and \ref adc_config.divide_result
  * members in the configuration struct.
  */
ADC_RESOLUTION_CUSTOM,
};
```

© 2008 Microchip Technology Incorporated. All Rights Reserved.

No Lab Exercises for Accumulation and Averaging

- Please practice yourself from Lab10
- Adjust TC3 compare value in MyTC_init() for shorten ADC conversion interval.

```
TC_Config.counter_16_bit.compare_capture_channel[0] = (48000000 / 1024 / 100);
```

- In main(), Commenting out “Hello world!!” output

```
// sprintf((char *) USARTRTxBuffer, "Hello World!!\r\n");
// usart_write_buffer_job(&USART5_Instance, USARTRTxBuffer, strlen(USARTRTxBuffer));
```

- Modify ADC result output and without newline ‘\n’ character for renew at the same line.

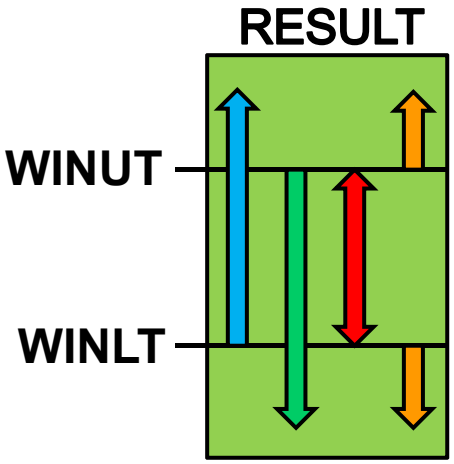
```
sprintf((char *) USARTRTxBuffer, "VR1 Value : %4d ", ADC_Result[0]);
sprintf((char *) USARTRTxBuffer, "Temperature Value: %4d\r", ADC_Result[5]);
```

- Let’s go!

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Window Monitor

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

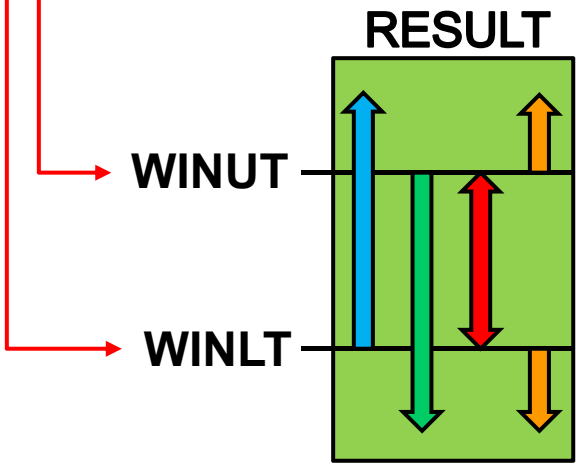


WINMODE[2:0]	Name	Description
0x0	DISABLE	No window mode (default)
0x1	MODE1	Mode 1: RESULT > WINLT
0x2	MODE2	Mode 2: RESULT < WINUT
0x3	MODE3	Mode 3: WINLT < RESULT < WINUT
0x4	MODE4	Mode 4: !(WINLT < RESULT < WINUT)
0x5-0x7		Reserved

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Window Monitor
Code example

```
adc_get_config_defaults(&ADC_Config);
ADC_Config. window_mode = ADC_WINDOW_MODE_BETWEEN;
ADC_Config. window_lower_value= 1000;
ADC_Config. window_upper_value= 3000;
….
adc_register_callback(&ADC_Instance, ADC_Window, ADC_CALLBACK_WINDOW);
adc_enable_callback(&ADC_Instance, ADC_CALLBACK_WINDOW);
```



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

© 2008 Microchip Technology Incorporated. All Rights Reserved.

No Lab Exercises for Window Monitor

- Please practice yourself from Lab10
- Add below code segment to Lab10.

```
adc_get_config_defaults(&ADC_Config);
ADC_Config. window_mode = ADC_WINDOW_MODE_BETWEEN;
ADC_Config. window_lower_value= 1000;
ADC_Config. window_upper_value= 3000;
….
adc_register_callback(&ADC_Instance, ADC_Window, ADC_CALLBACK_WINDOW);
adc_enable_callback(&ADC_Instance, ADC_CALLBACK_WINDOW);
```

- Add a callback function ADC_Window()

```
volatile uint8_t ADC05_InWindow = 0;
volatile uint8_t ADC10_InWindow = 0;
void ADC_Window(struct adc_module * const Instance) /* CallBack */
{
      if( Instance->remaining_conversions==5 )    ADC05_InWindow = 1;
      if( Instance->remaining_conversions==0 )    ADC10_InWindow = 1;
}
```

© 2008 Microchip Technology Incorporated. All Rights Reserved.

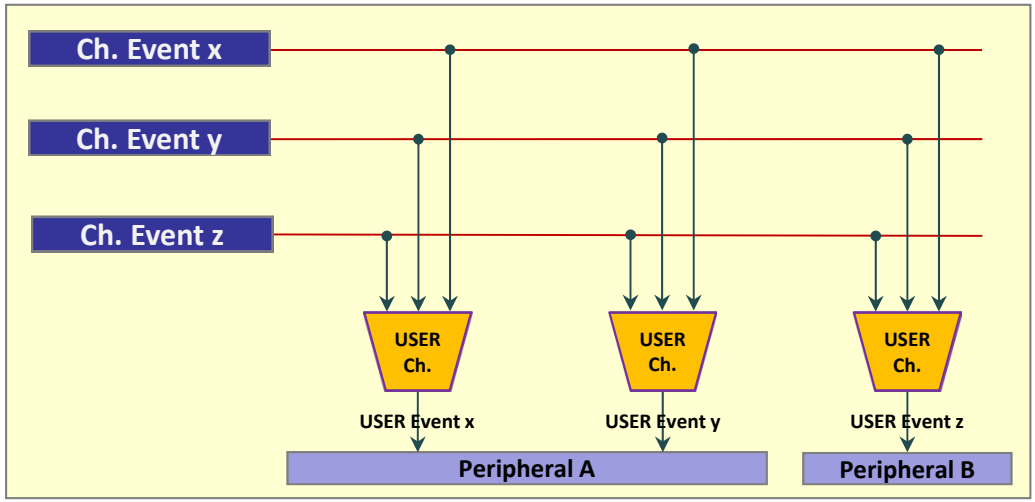
Event System

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

© 2008 Microchip Technology Incorporated. All Rights Reserved.

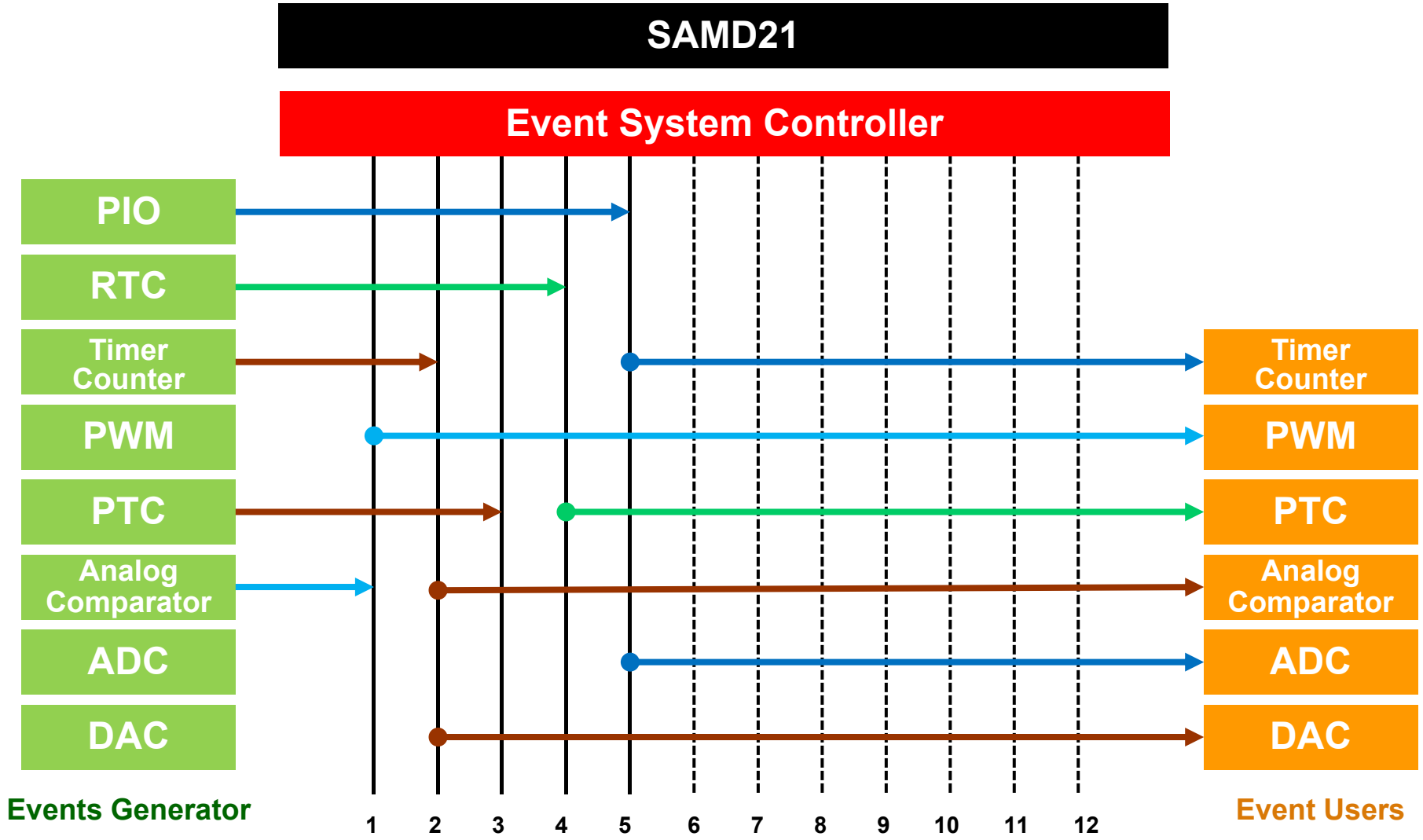
Event System

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



© 2008 Microchip Technology Incorporated. All Rights Reserved.

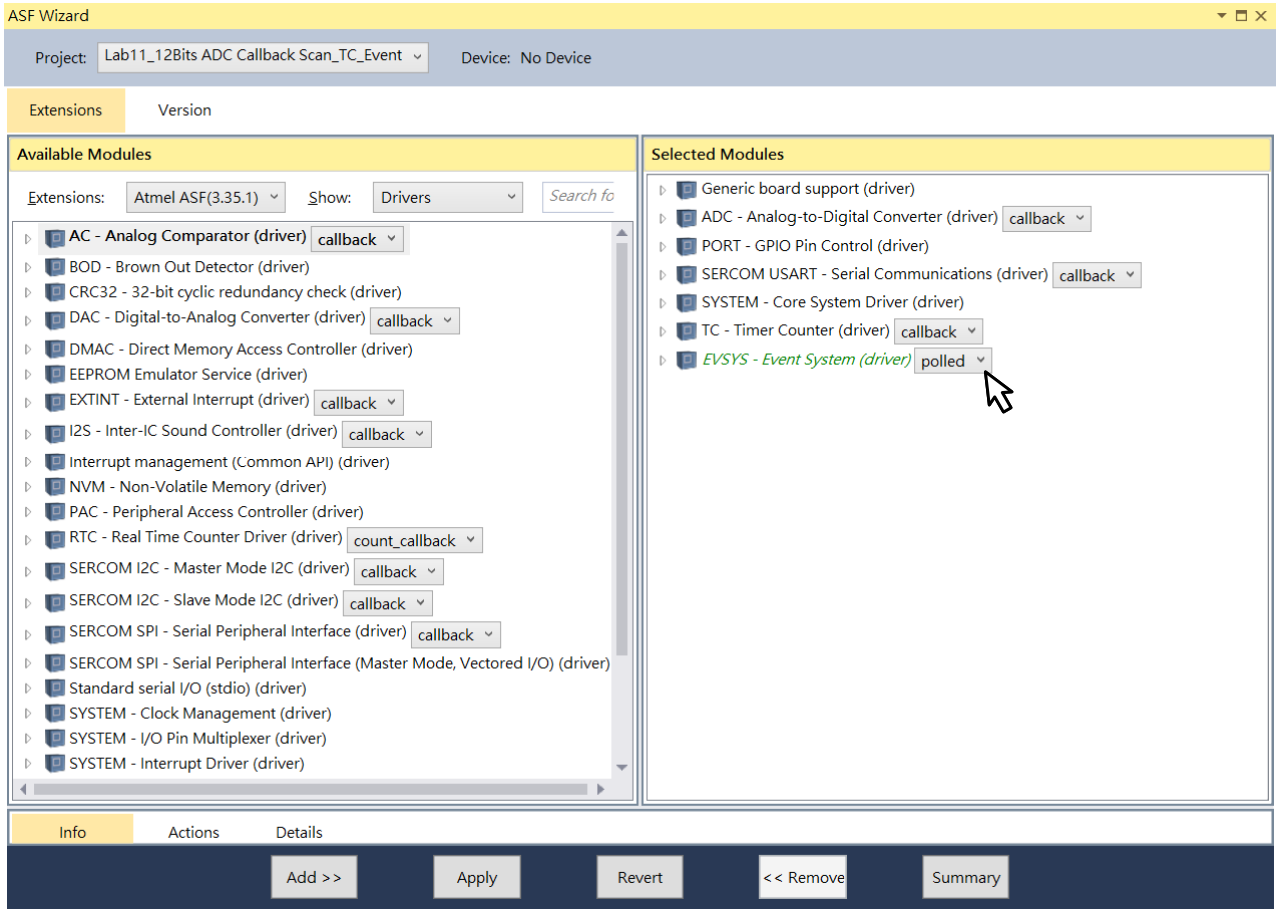
Event System



© 2008 Microchip Technology Incorporated. All Rights Reserved.

Add EVSYS Function from ASF

- Select **ASF ▸ ASF Wizard**, Add **EVSYS .. polled** to project.



© 2008 Microchip Technology Incorporated. All Rights Reserved.

ASF EVSYS Driver polled Functions

- /* brief Initializes an event configurations struct to defaults. */
  void events_get_config_defaults(struct events_config *config)

- /* Allocate an event channel and set configuration */
  enum status_code events_allocate(
  struct events_resource * resource,
  struct events_config * config)

- /* Attach user to the event channel. */
  enum status_code events_attach_user(
  struct events_resource * resource,
  uint8_t user_id)

© 2008 Microchip Technology Incorporated. All Rights Reserved.

ASF EVSYS polled, TC3 Overflow trigger ADC Start Conversion.

Event System Initial

```
struct events_resource EVENTS1_Instance;
struct events_config EVENTS_Config;
...
events_get_config_defaults(&EVENTS_Config);
EVENTS_Config.generator = EVSYS_ID_GEN_TC3_OVF;
EVENTS_Config.edge_detect = EVENTS_EDGE_DETECT_NONE;
EVENTS_Config.path = EVENTS_PATH_ASYNCHRONOUS;
EVENTS_Config.clock_source = GCLK_GENERATOR_0;
events_allocate(& EVENTS1_Instance, &EVENTS_Config);
events_attach_user(& EVENTS1_Instance, EVSYS_ID_USER_ADC_START);
```

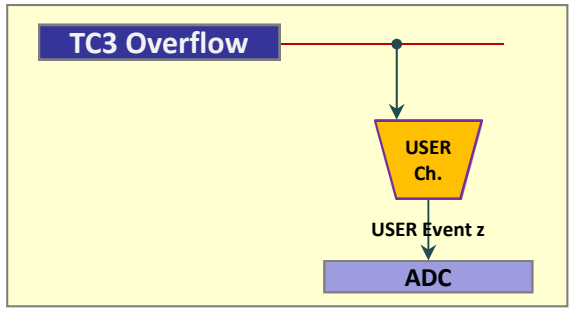
Channel allocate by system automatically.

GENERATOR Setting (TC3, Overflow)

```
TC3_Config_Events.generate_event_on_overflow = true;
tc_enable_events(&TC3_Instance, &TC3_Config_Events);
```

USER Setting (ADC, Start Conversion)

```
ADC_Config.event_action = ADC_EVENT_ACTION_START_CONV;
```



© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab11 12Bits ADC Callback Scan TC Event

- use TC overflow event to replace software trigger ADC.
- Other same as Lab10.

- Let's go!

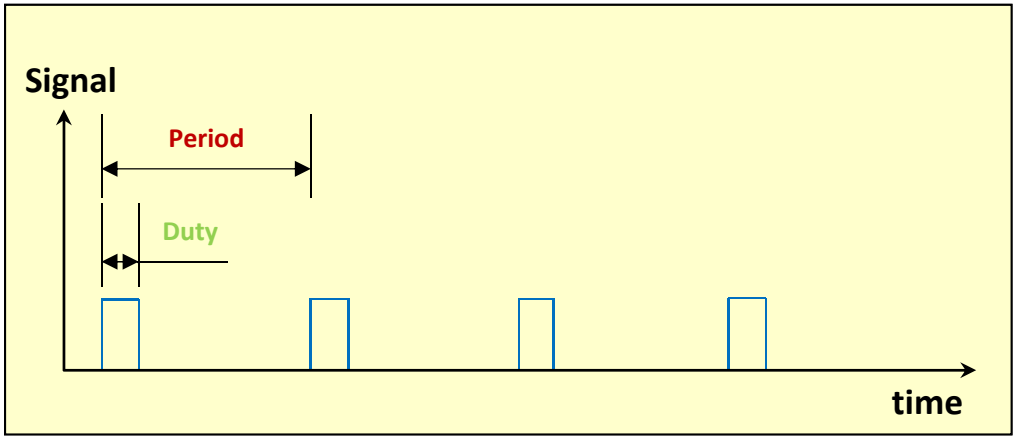
© 2008 Microchip Technology Incorporated. All Rights Reserved.

TCC – PWM Architecture

© 2008 Microchip Technology Incorporated. All Rights Reserved.

What's PWM

- The Pulse-width modulation (PWM) Waveform is a modulation technique used to encode a message into a pulsing signal.
- Its main use is to allow the control of the power supplied to electrical devices, ex. motor control, ballast, LED, H-bridge, power converters, and other types of power control applications.
- There have two important terms
  - **Period :**
    The single square wave duration.
  - **Duty :**
    The proportion of active time to the regular interval



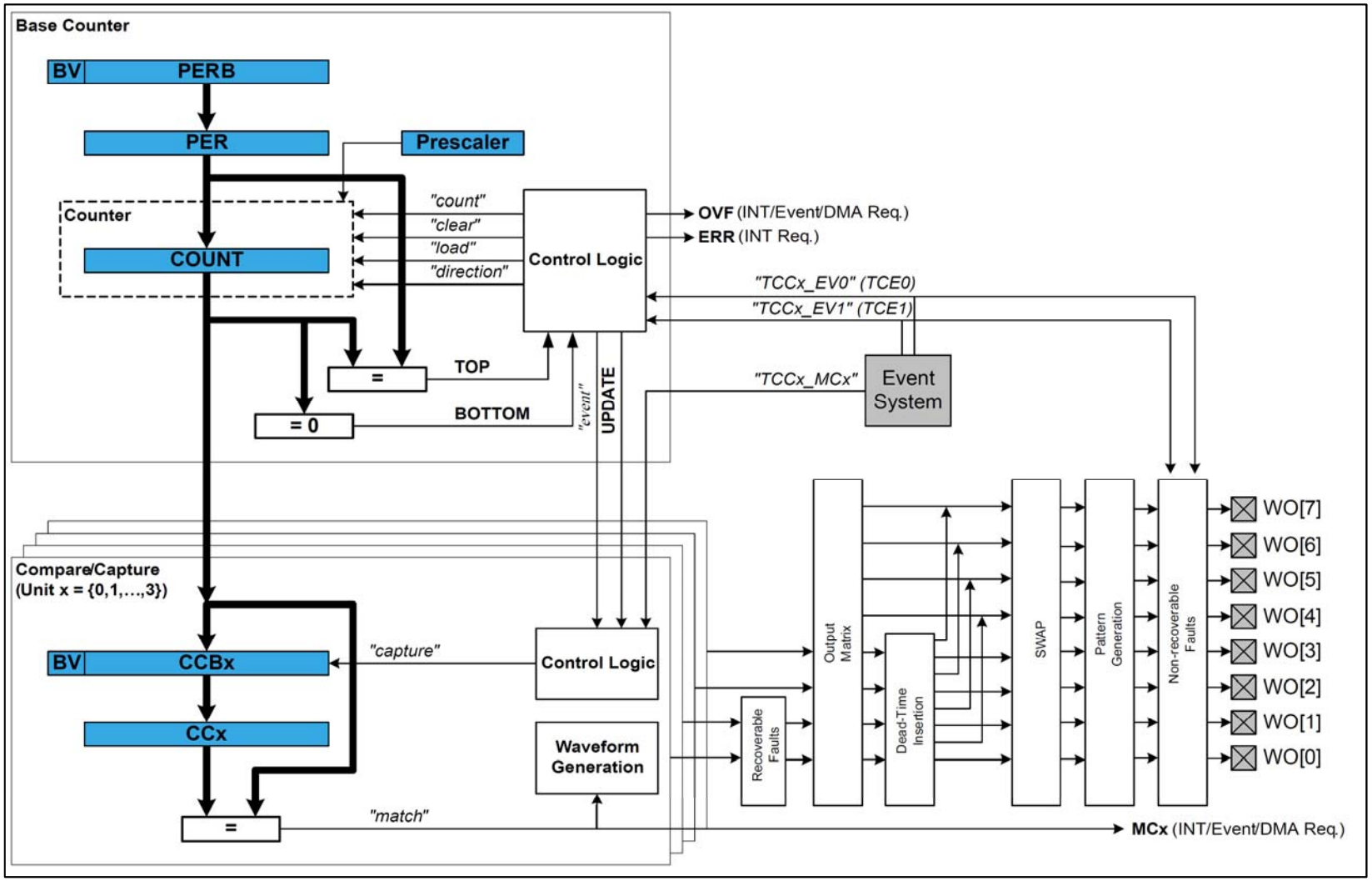
© 2008 Microchip Technology Incorporated. All Rights Reserved.

SAMD21 TCC

- **Timer/Counter for Control, TCC is the TC Enhanced version.**
- **TCC provide below feature**
  - Up to four compare/capture channels
  - Waveform (PWM, Frequency generation)
  - Input capture (Event, Frequency, PWM Capture)
  - Fault protection for safe disabling
  - 3 input Event, 3 Event
  - Support Interrupt, DMA, Event

© 2008 Microchip Technology Incorporated. All Rights Reserved.

TCC Block Diagram



© 2008 Microchip Technology Incorporated. All Rights Reserved.

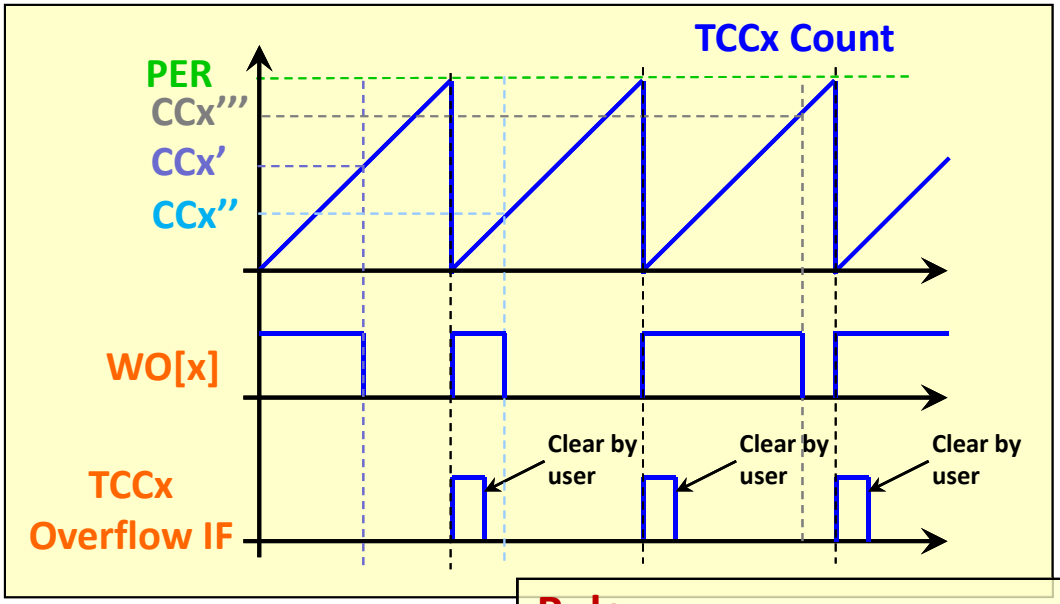
TCCx, Channel and WO[x]

- **TCCx : Timer Counter for Control peripheral**
  - TCC0, TCC1, TCC2
- **Channel : Compare Channel CCx**
  - TCC0 : 4 channel
  - TCC1 : 2 channel
  - TCC2 : 2 channel
- **WO[x] : Waveform Output Pins**
  - TCC0 : 8 pins ➔ WO[0] ~ WO[7] ➔ Channel [0, 1, 2, 3, **0, 1, 2, 3**]
  - TCC1 : 4 pins ➔ WO[0] ~ WO[3] ➔ Channel [0, 1, **0, 1**]
  - TCC2 : 2 pins ➔ WO[0] ~ WO[1] ➔ Channel [0, 1]

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Normal Pulse-Width Modulation (NPWM) Mode

- For Normal Pulse-Width Modulation , the period time (T) is controlled by the PER register. WO[x] is set at start and cleared on compare match.



Rule

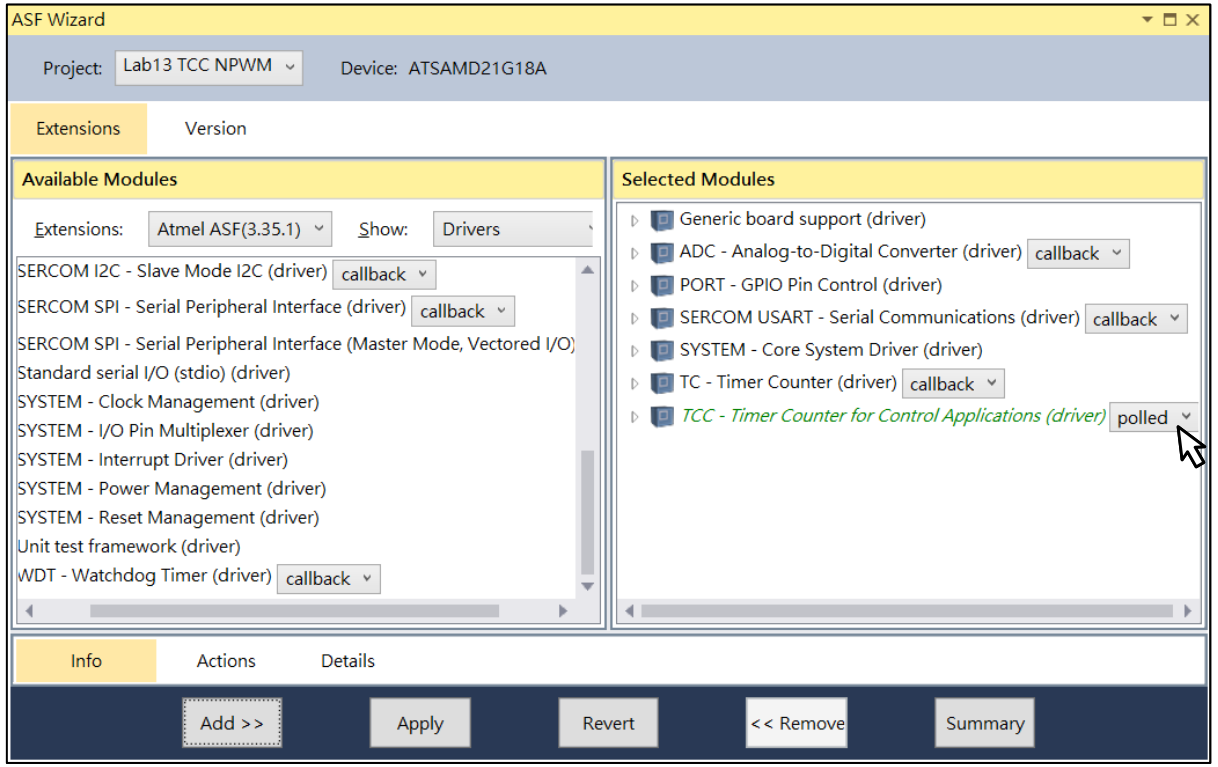
1.TCCx Counter == CCx -> WO[x] Clear to low

2.TCCx Counter == PER -> WO[x] Set to high

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Add TCC Function from ASF

- Select **ASF ▸ ASF Wizard**, Add **TCC .. polled** to project.



© 2008 Microchip Technology Incorporated. All Rights Reserved.

ASF TCC – NPWM Code Example

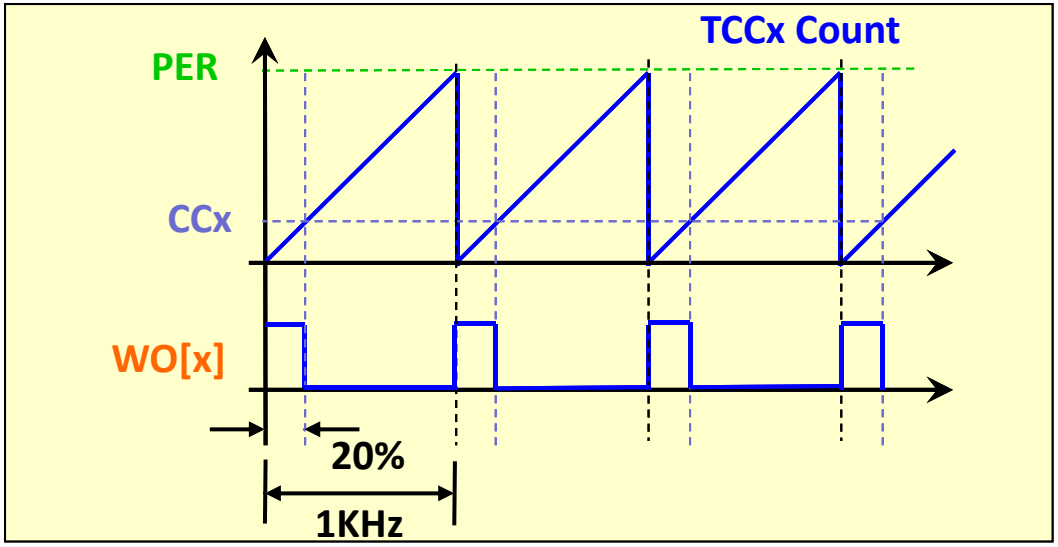
```
struct tcc_module TCC_Instance;
struct tcc_config TCC_Config;

tcc_get_config_defaults(&TCC_Config, TCC2);
TCC_Config.compare.wave_generation = TCC_WAVE_GENERATION_SINGLE_SLOPE_PWM;
TCC_Config.counter.clock_prescaler = TCC_CLOCK_PRESCALER_DIV1024;
TCC_Config.counter.period = 1000;
TCC_Config.compare.match[1] = 500;
TCC_Config.pins.enable_wave_out_pin[1] = true;
TCC_Config.pins.wave_out_pin[1] = PIN_PA17E_TCC2_WO1;
TCC_Config.pins.wave_out_pin_mux[1] = MUX_PA17E_TCC2_WO1;
while (tcc_init(&TCC_Instance, TCC2, &TCC_Config) != STATUS_OK);

tcc_enable(&TCC_Instance);

while(1);
```

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab12 TCC NPWM

- Add TCC NPWM Mode function to your project.
- Try to set TCC2 to output PWM waveform to LED3.
- PWM frequency is 1KHz, 20% Duty.

- Let's go!



© 2008 Microchip Technology Incorporated. All Rights Reserved.

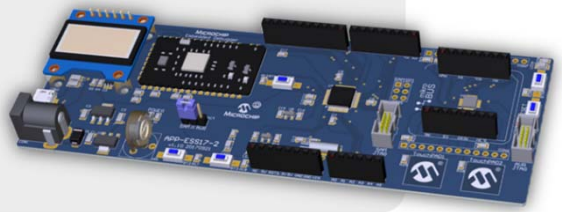
Lab12 TCC NPWM

Step

a **Add below code segment for initial**

```
struct tcc_module TCC2_Instance;
struct tcc_config TCC2_Config;

int main (void)
{
   tcc_get_config_defaults(& TCC2_Config, TCC2);
   TCC2_Config.compare.wave_generation =
                                        TCC_WAVE_GENERATION_SINGLE_SLOPE_PWM;
   TCC2_Config.counter.clock_prescaler = TCC_CLOCK_PRESCALER_DIV1024;
   TCC2_Config.counter.period = (48000000 / 1024 / 1000);
   TCC2_Config.compare.match[1] = (48000000 / 1024 / 1000) / 5;
   TCC2_Config.pins.enable_wave_out_pin[1] = true;
   TCC2_Config.pins.wave_out_pin[1] = PIN_PA17E_TCC2_WO1;
   TCC2_Config.pins.wave_out_pin_mux[1] =MUX_PA17E_TCC2_WO1;
   while (tcc_init(&TCC2_Instance, TCC2, & TCC2_Config) != STATUS_OK);    …

   while(1);
}
```

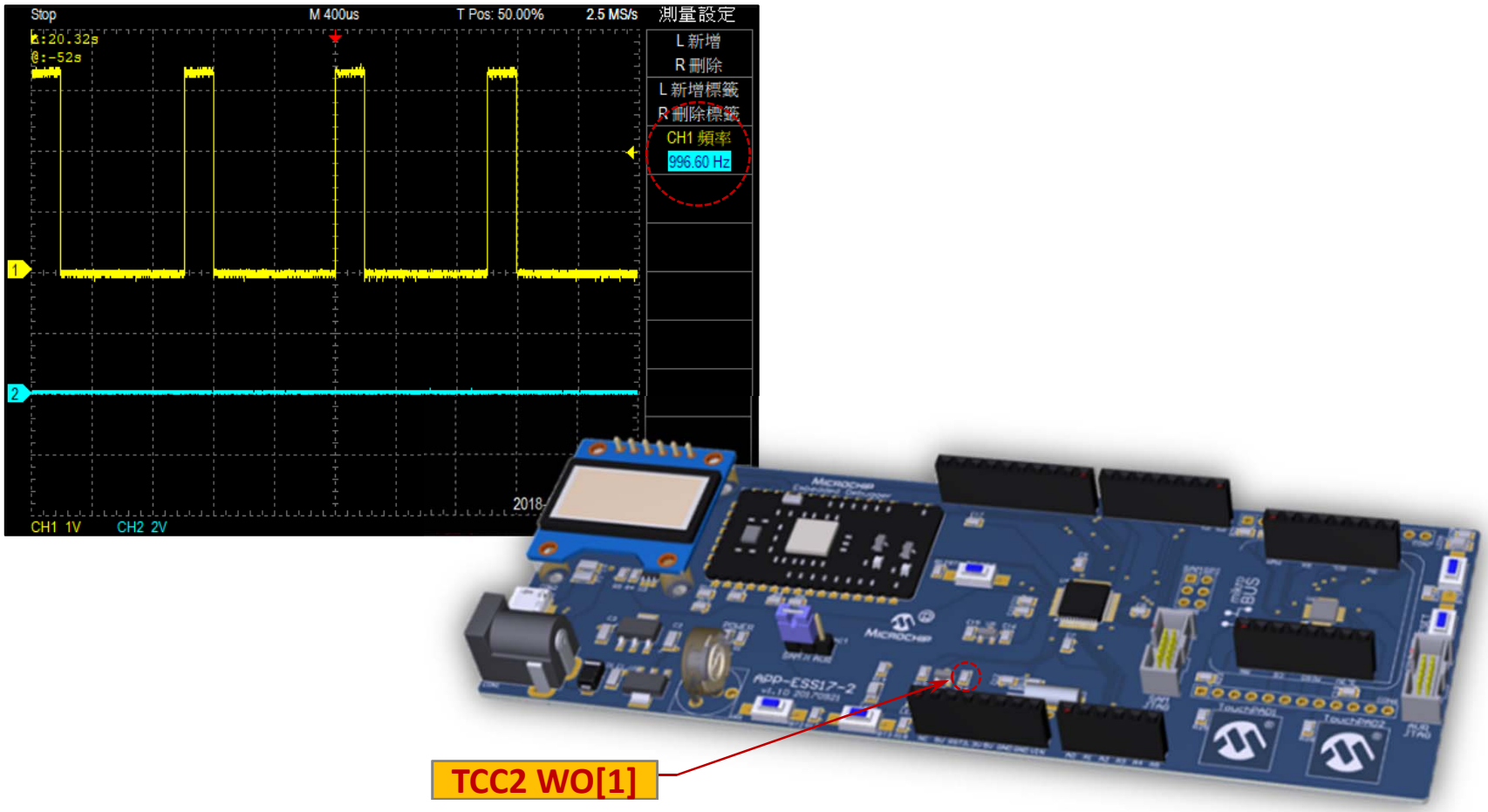
b **Program firmware to target board then observe result.**



© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab12 TCC NPWM

Result



© 2008 Microchip Technology Incorporated. All Rights Reserved.

ASF TCC - NPWM Functions

- /* Sets a TCC module compare value. */

```
enum status_code tcc_set_compare_value
                (const struct tcc_module *const module_inst,
                 const enum tcc_match_capture_channel channel_index,
                 const uint32_t compare)
```

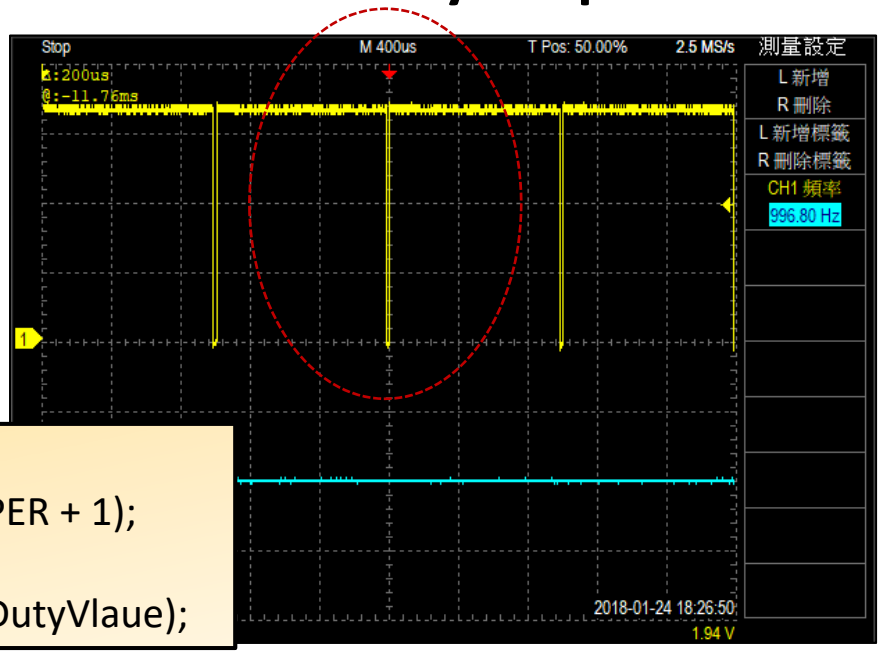
- /* Set the timer TOP/PERIOD value.. */

```
enum status_code tcc_set_top_value
                (const struct tcc_module *const module_inst,
                 const uint32_t top_value)
```

© 2008 Microchip Technology Incorporated. All Rights Reserved.

100% Duty Issue

- How to generate 100% duty PWM ?
  - Rule Review
    1.TCCx Counter = CCx -> Output Set to low
    2.TCCx Counter = PER -> Output Set to high
- This means CCx = PER must be set for 100% duty output.
  Which rule be executed ?
  - Result :
    Glitch !
- To prevent this issue,
  set CCx > PER for 100% duty.




```
if( DutyVlaue >= PER ) /* 100% duty */
   tcc_set_compare_value(&TCC_Instance, 1, PER + 1);
else
   tcc_set_compare_value(&TCC_Instance, 1, DutyVlaue);
```

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Lab13 TCC NPWM ADC DutyAdj

- **Try to use VR1 to control PWM duty.**
  - VR Value 0 -> Duty 0%
  - VR Value 4095 -> Duty 100%

```
if (ADC_Result[0] >= 4095)
   tcc_set_compare_value(&TCC2_Instance, 1, TCC2_PWM_Peroid + 1);
else
   tcc_set_compare_value(&TCC2_Instance, 1, ADC_Result[0] * TCC2_PWM_Peroid / 4095);
```

© 2008 Microchip Technology Incorporated. All Rights Reserved.

Trademarks

The Microchip name and logo, the Microchip logo, dsPIC, KeeLoq, KeeLoq logo, MPLAB, PIC,PICmicro, PICSTART, PIC32 logo, rfPIC and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

FilterLab, Hampshire, HI-TECH C, Linear Active Thermistor, MXDEV, MXLAB, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, Application Maestro, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, dsSPEAK, ECAN, ECONOMONITOR, FanSense, HI-TIDE, In-Circuit Serial Programming, ICSP, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, mTouch, Octopus, Omniscient Code Generation, PICC, PICC-18, PICDEM, PICDEM.net, PICkit, PICtail, REAL ICE, rfLAB, Select Mode, Total Endurance, TSHARC, UniWinDriver, WiperLock and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

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

© 2010, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

Thank You !!

© 2008 Microchip Technology Incorporated. All Rights Reserved.