

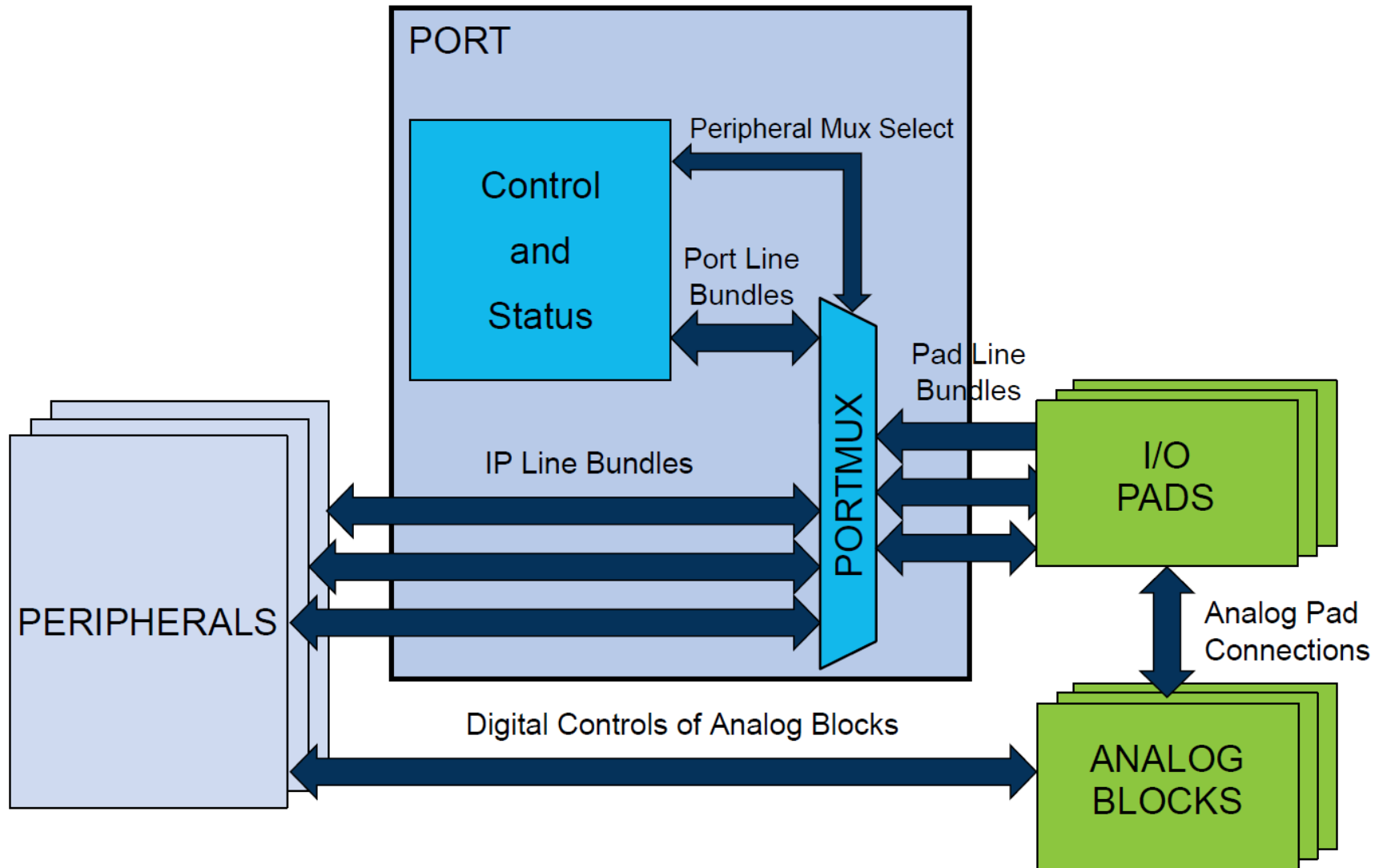


MICROCHIP

Regional Training Centers

**Section 9
PINMUX &
SERCOM - UART Architecture**

What's PINMUX ?



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 (A ~ H)

PMUX			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]
19	27	37	PA18	VDDIO	EXTINT[2]				X[6]		SERCOM1/ PAD[2]	SERCOM3/ PAD[2]	TC3/WO[0]	TCC0/ WO[2]		AC/CMP[0]
20	28	38	PA19	VDDIO	EXTINT[3]				X[7]		SERCOM1/ PAD[3]	SERCOM3/ PAD[3]	TC3/WO[1]	TCC0/ WO[3]	I2S/SD[0]	AC/CMP[1]
21	31	43	PA22	VDDIO	EXTINT[6]				X[10]		SERCOM3/ PAD[0]	SERCOM5/ PAD[0]	TC4/WO[0]	TCC0/ WO[4]		GCLK_IO[6]
22	32	44	PA23	VDDIO	EXTINT[7]				X[11]		SERCOM3/ PAD[1]	SERCOM5/ PAD[1]	TC4/WO[1]	TCC0/ WO[5]	USB/SOF 1kHz	GCLK_IO[7]

Function Groups (A ~ H)

Pin Name

PINMUX PLIB Analysis

main() @ main.c

SYS_Initialize (NULL);

SYS_Initialize() @ initialization.c

PORT_Initialize();

PORT_Initialize() @ plib_port.c

```
PORT_REGS->GROUP[0].PORT_DIR = 0x120200;
PORT_REGS->GROUP[0].PORT_OUT = 0x200;
PORT_REGS->GROUP[0].PORT_PINCFG[14] = 0x3;
PORT_REGS->GROUP[0].PORT_PINCFG[15] = 0x3;
PORT_REGS->GROUP[0].PORT_PINCFG[18] = 0x1;
PORT_REGS->GROUP[0].PORT_PINCFG[19] = 0x1;
PORT_REGS->GROUP[0].PORT_PINCFG[22] = 0x1;
PORT_REGS->GROUP[0].PORT_PINCFG[23] = 0x1;
```

```
PORT_REGS->GROUP[0].PORT_PMUX[7] = 0x77;
PORT_REGS->GROUP[0].PORT_PMUX[9] = 0x44;
PORT_REGS->GROUP[0].PORT_PMUX[11] = 0x44;
```

Offset	Name	7	6	5	4	3	2	1	0
0x30	PMUX0	PMUXO[3:0]				PMUXE[3:0]			
	PMUX	PMUXO (ODD)				PMUXE (EVEN)			
0x3F	PMUX15	PMUXO[3:0]				PMUXE[3:0]			
0x40	PINCFG0		DRVSTR				PULLEN	INEN	PMUXEN
	PINCFG								
0x5F	PINCFG31		DRVSTR				PULLEN	INEN	PMUXEN

INEN PMUXEN

PA14 (GCLK_IO0)

PA15 (GCLK_IO1)

PA18 (TC3_WO0)

PA19 (TC3_WO1)

PA22 (TC4_WO0)

PA23 (TC4_WO1)

INEN = 1 (Button)

INEN = 1 (Button)

PMUX[7] : PA15 , PA14

PMUX[9] : PA19 , PA18

PMUX[11] : PA23 , PA22

PINMUX Configuration Example

MPLAB Harmony Configurator - default*

File Generate Tools Utilities Window

Framework: D:\HarmonyFramework_3.3.0\

Active Components

- CMSIS Pack
- Device Family Pack (DFP)
- EVSYS
- NVMCTRL
- System
- TC3
- TC4

Pin Settings

Order: Pins Table View Easy View

Pin Number	Pin ID	Custom Name	Function	Mode	Direction	Latch	Pull Up	Pull Down
16	PA11		Available	Digital	High I...	Low		
17	VDDIO			Digital	High I...	Low		
18	GNDIO			Digital	High I...	Low		
19	PB10		Available	Digital	High I...	Low		
20	PB11		Available	Digital	High I...	Low		
21	PA12		Available	Digital	High I...	Low		
22	PA13		Available	Digital	High I...	Low		
23	PA14	GCLK_IO0	GCLK_IO0	Digital	In	n/a		
24	PA15	GCLK_IO1	GCLK_IO1	Digital	In	n/a		
25	PA16		Available	Digital	High I...	Low		
26	PA17	LED3	GPIO	Digital	Out	Low		
27	PA18	TC3_WO0	TC3_WO0	Digital	High I...	n/a		
28	PA19	TC3_WO1	TC3_WO1	Digital	High I...	n/a		
29	PA20	LED1	GPIO	Digital	Out	Low		
30	PA21		Available	Digital	High I...	Low		
31	PA22	TC4_WO0	TC4_WO0	Digital	High I...	n/a		
32	PA23	TC4_WO1	TC4_WO1	Digital	High I...	n/a		
33	PA24		Available	Digital	High I...	Low		
34	PA25		Available	Digital	High I...	Low		
35	GNDIO			Digital	High I...	Low		
36	VDDIO			Digital	High I...	Low		
37	PB22		Available	Digital	High I...	Low		
38	PB23		Available	Digital	High I...	Low		
39	PA27		Available	Digital	High I...	Low		

Pin Diagram

Unavailable Available Assigned

484746454443424140393837

ATSAMD21G18A

131415161718192021222324

Pin Table

Package: TQFP48

Module	Function	PA00	PA01	PA02	PA03	GNDAN	VDDAN	PB08	PB09	PA04	PA05	PA06	PA07	PA08	LED2	PA10	PA11	VDDIO	GNDIO	PB10	PB11	PA12	PA13	GCLK_I	GCLK_I	PA16	LED3	TC3_W0	TC3_W0	LED1	PA21	TC4_W0	TC4_W0	PA24	PA25	GNDIO	VDDIO	PB22	PB23	PA27	RESET	PA28	GNDIO	VDDIO	VDDIO	PA30	PA31	PB02	PB03		
GPIO	GPIO																																																		
	I2S_FS0																																																		
	I2S_MCK0																																																		

Welcome to the MPLAB Harmony Configurator!

PINMUX PLIB Analysis

			PMUX		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	PA14 (7)	1	PA14	VDDIO	EXTINT[14]						SERCOM2/ PAD[2]	SERCOM4/ PAD[2]	TC3/WO[0]	GCLK_IO[0]		GCLK_IO[0]
16	PA15 (7)	2	PA15	VDDIO	EXTINT[15]						SERCOM2/ PAD[3]	SERCOM4/ PAD[3]	TC3/WO[1]	GCLK_IO[1]		GCLK_IO[1]
19	PA18 (4)	7	PA18	VDDIO	EXTINT[2]				X[6]		TC3/WO[0]		TC3/WO[0]	TCC0/ WO[2]		AC/CMP[0]
20	PA19 (4)	8	PA19	VDDIO	EXTINT[3]				X[7]		TC3/WO[1]		TC3/WO[1]	TCC0/ WO[3]	I2S/SD[0]	AC/CMP[1]
21	PA22 (4)	3	PA22	VDDIO	EXTINT[6]				X[10]		TC4/WO[0]		TC4/WO[0]	TCC0/ WO[4]		GCLK_IO[6]
22	PA23 (4)	4	PA23	VDDIO	EXTINT[7]				X[11]		TC4/WO[1]		TC4/WO[1]	TCC0/ WO[5]	USB/SOF 1kHz	GCLK_IO[7]

PORT_REGS->GROUP[0].PORT_PMUX[7] = 0x77;
 PORT_REGS->GROUP[0].PORT_PMUX[9] = 0x44;
 PORT_REGS->GROUP[0].PORT_PMUX[11] = 0x44;

PA15 (7) , PA14 (7)

PA19 (4) , PA18 (4)

PA23 (4) , PA22 (4)

SERCOM Introduction

- Serial Communication Interface, SERCOM be configured to support a number of modes :

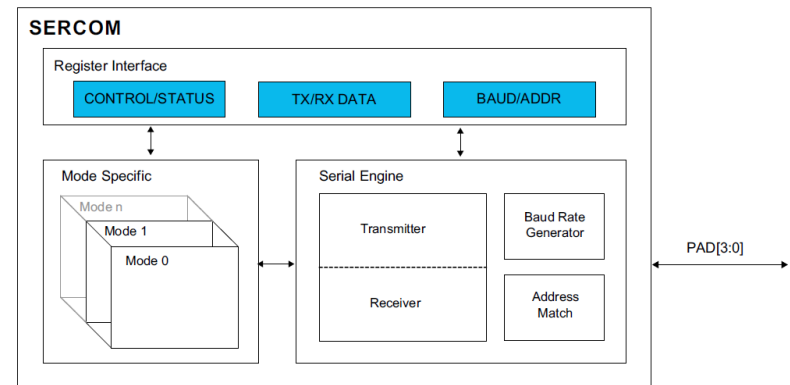
❏ I2C

❏ SPI

❏ USART : 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.



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

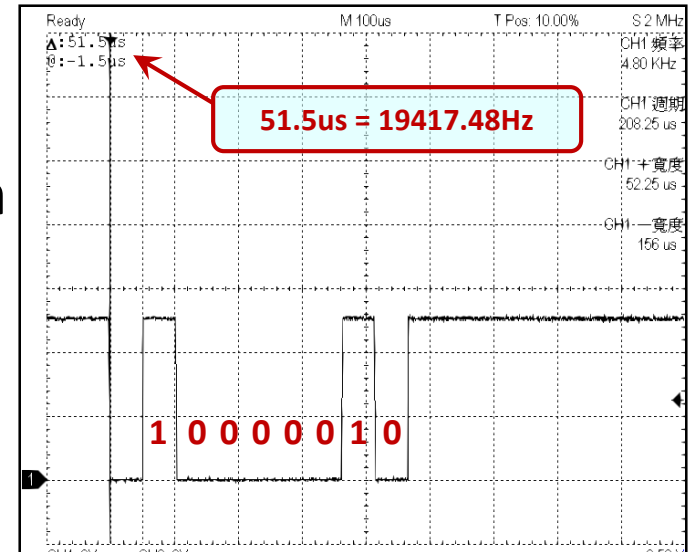
■ For USART

- ◆ Up to 24Mb/s
- ◆ Half-/Full-duplex operation
- ◆ Sync and Async operation

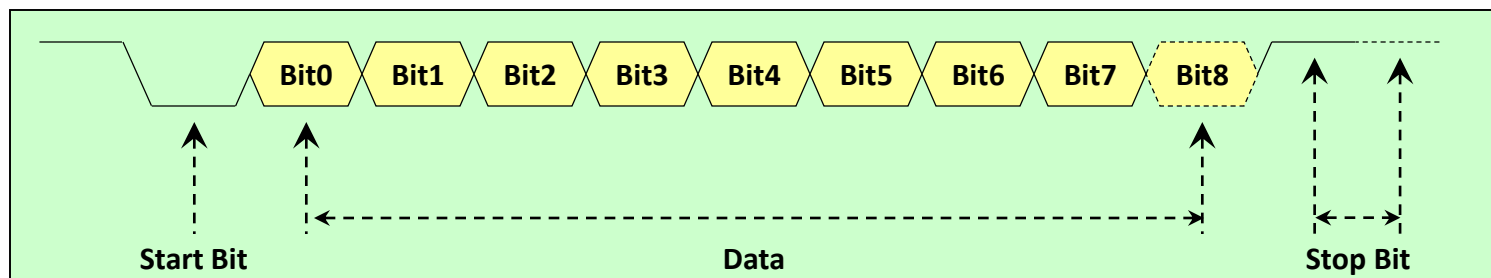


What's UART ?

- **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.

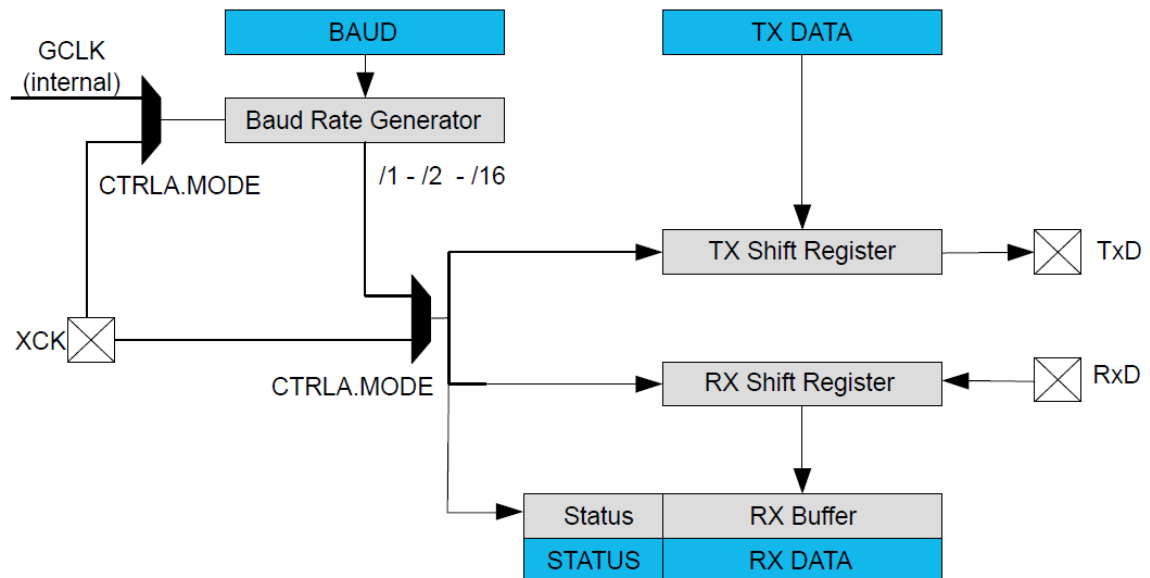


UART Transmit Example :
'A'(0x41), 19200 bps



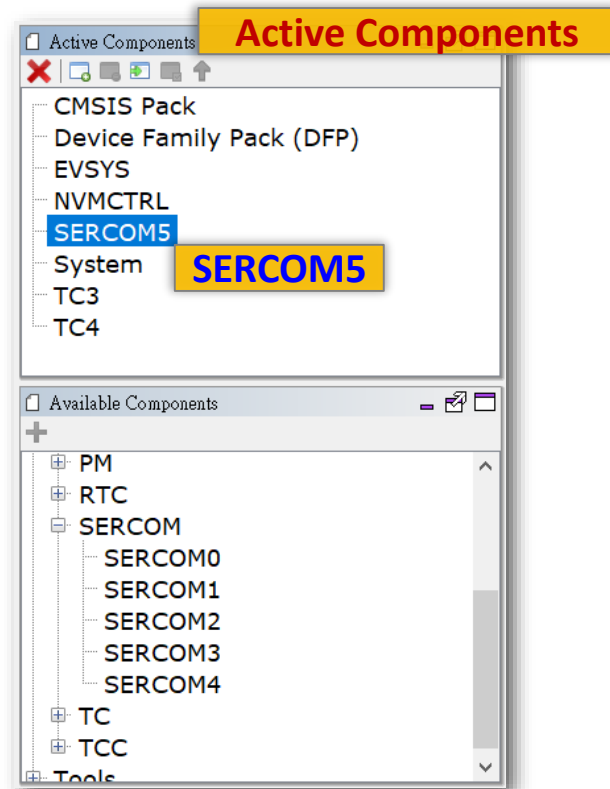
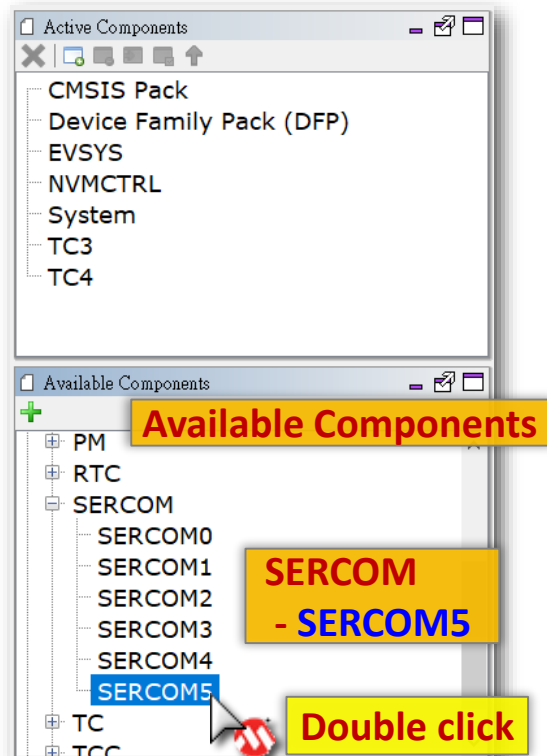
SERCOM - USART

- 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 and IrDA.
- Buffer overflow and frame error detection.
- Internal or external clock source for asynchronous and synchronous operation.



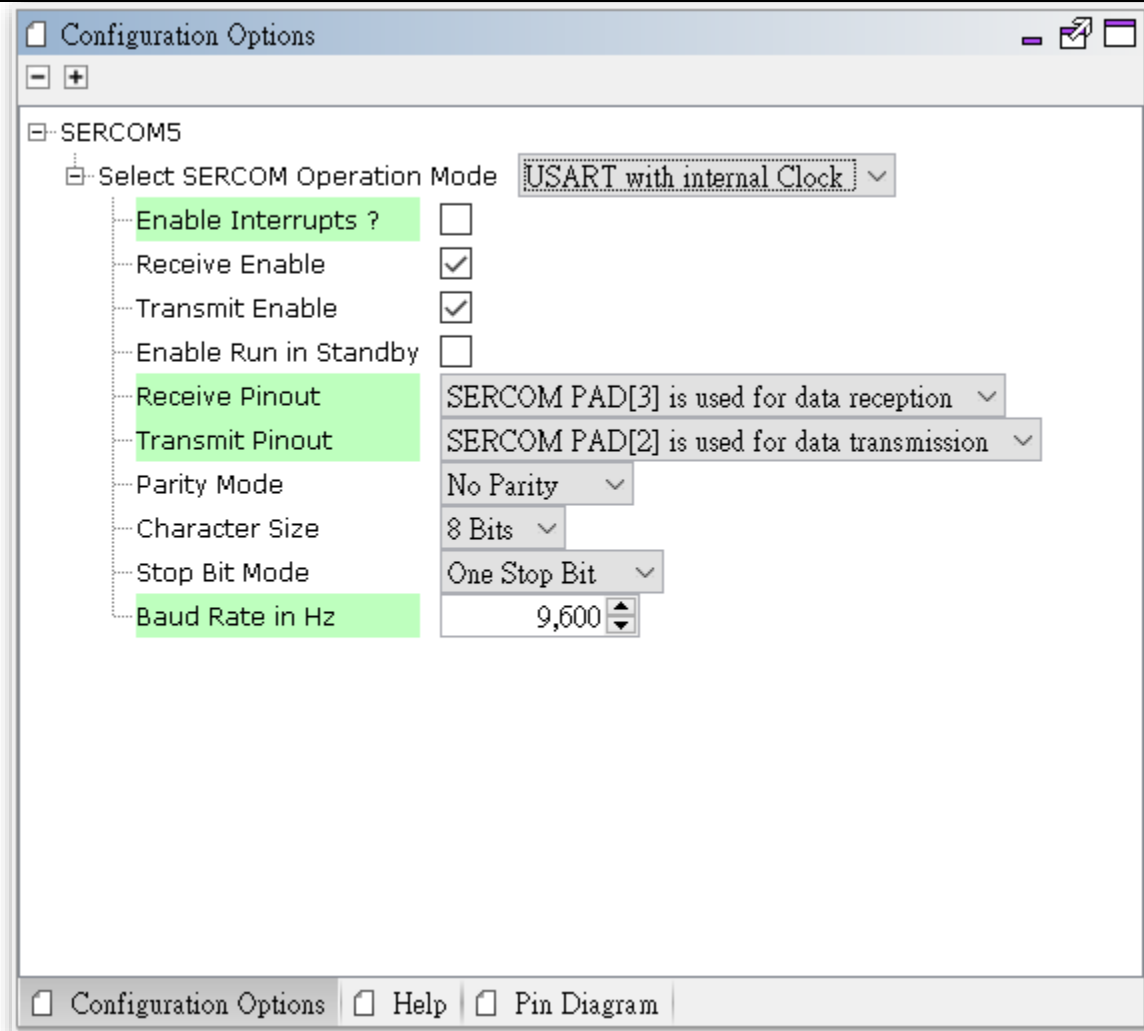
Add SERCOM Function using MHC

- Find **SERCOM** component in Available Components window.
- Double click **SERCOM5** to add to Active Components window.

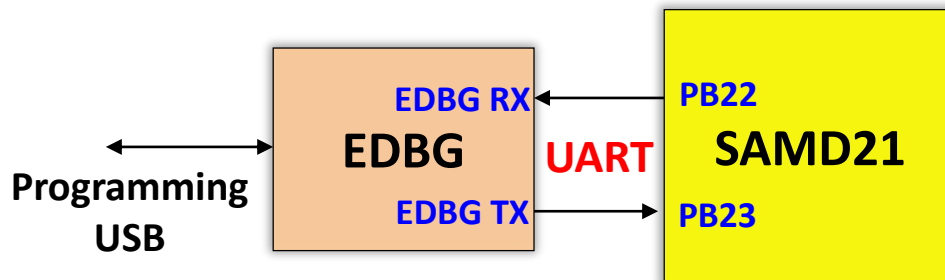
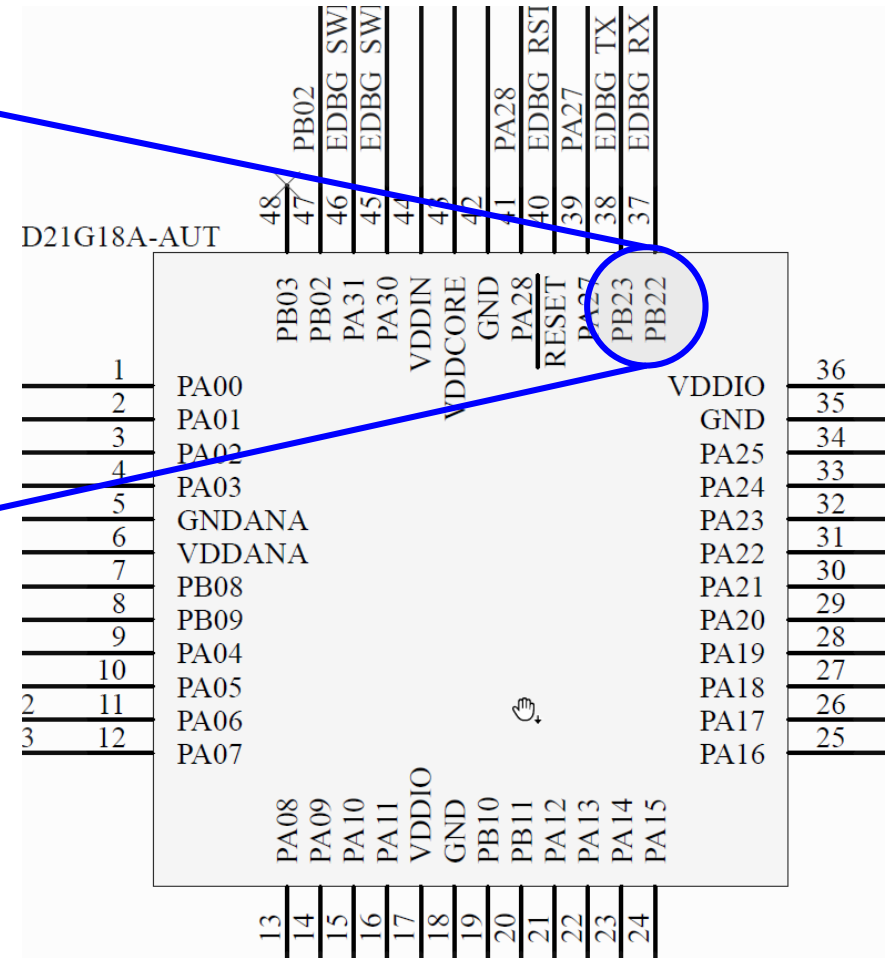
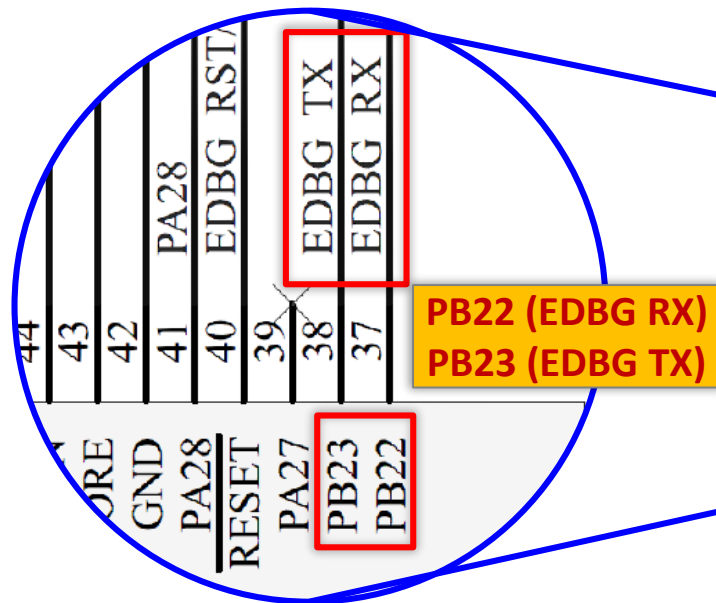


SERCOM USART Polling

Configuration Options Example



APP045 SERCOM USART Schematic



SERCOM USART PINMUX

Configuration Example

- SERCOM allow limited alternate pad index assignment.
- For example : PB22 as SERCOM5_PAD2
PB23 as SERCOM5_PAD3

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
	37	49	PB22	VDDIO	EXTINT[6]							SERCOM5/ PAD[2]	TC7A_WO[0]	TCC3/ WO[0]		GCLK_IO[0]
	38		PB23	VDDIO	EXTINT[7]							SERCOM5/ PAD[3]	TC7A_WO[1]	TCC3/ WO[1]		GCLK_IO[1]

Pin Table

Package: TQFP48

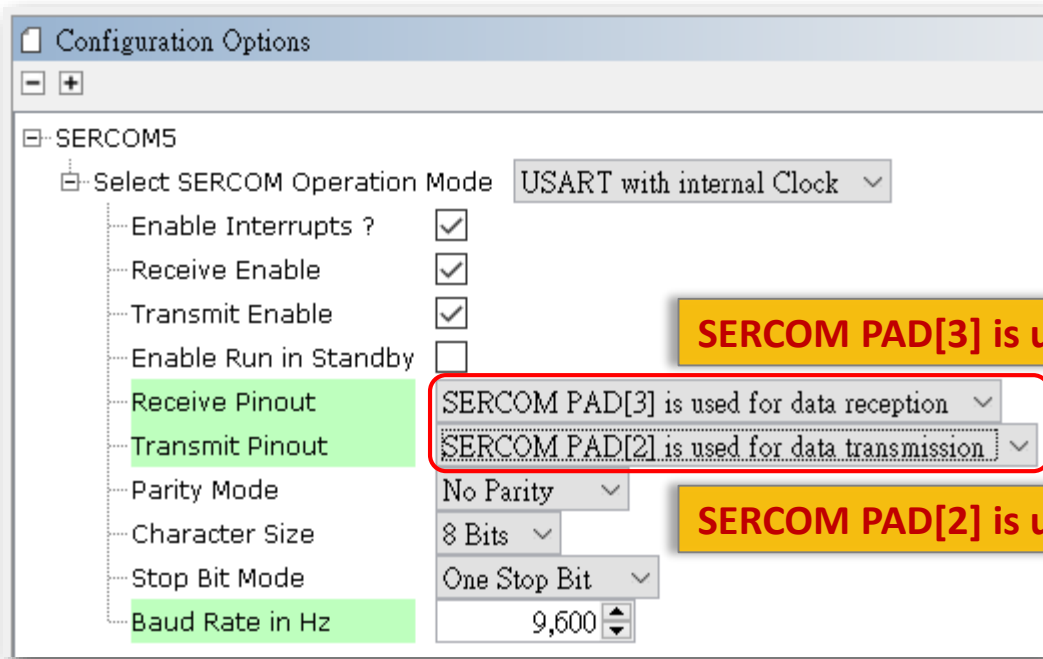
Module	Function	PA13	GCLK_I	GCLK_I	PA16	LED3	TC3_WC	TC3_WC	LED1	PA21	TC4_WC	TC4_WC	PA24	PA25	GNDIO	VDDIO	SERCOM	SERCOM	PA27
	SERCOM4_PAD3																		
	SERCOM5_PAD0																		
	SERCOM5_PAD1																		
SERCOM5	SERCOM5_PAD2																		
	SERCOM5_PAD3																		
	SYSCTRL_XIN																		

PB22 (SERCOM5_PAD2)
PB23 (SERCOM5_PAD3)

SERCOM USART PINMUX

Configuration Example

- Remember arrange pads function in SERCOM module.
- For example : SERCOM5 PAD[2] as USART **TXD** (**PB22**)
SERCOM5 PAD[3] as USART **RXD** (**PB23**)



SERCOM PAD[3] is used for data reception

SERCOM PAD[2] is used for data transmission

SERCOM USART Polling

Interface Routines

```
/** SERCOM USART Interface Routines**/  
void SERCOM5_USART_Initialize( void );  
bool SERCOM5_USART_SerialSetup( USART_SERIAL_SETUP * serialSetup,  
                                uint32_t clkFrequency );  
bool SERCOM5_USART_Write( void *buffer, const size_t size );  
bool SERCOM5_USART_TransmitterIsReady( void );  
bool SERCOM5_USART_TransmitComplete( void );  
void SERCOM5_USART_WriteByte( int data );  
bool SERCOM5_USART_Read( void *buffer, const size_t size );  
bool SERCOM5_USART_ReceiverIsReady( void );  
int SERCOM5_USART_ReadByte( void );  
USART_ERROR SERCOM5_USART_ErrorGet( void );  
uint32_t SERCOM5_USART_FrequencyGet( void );
```

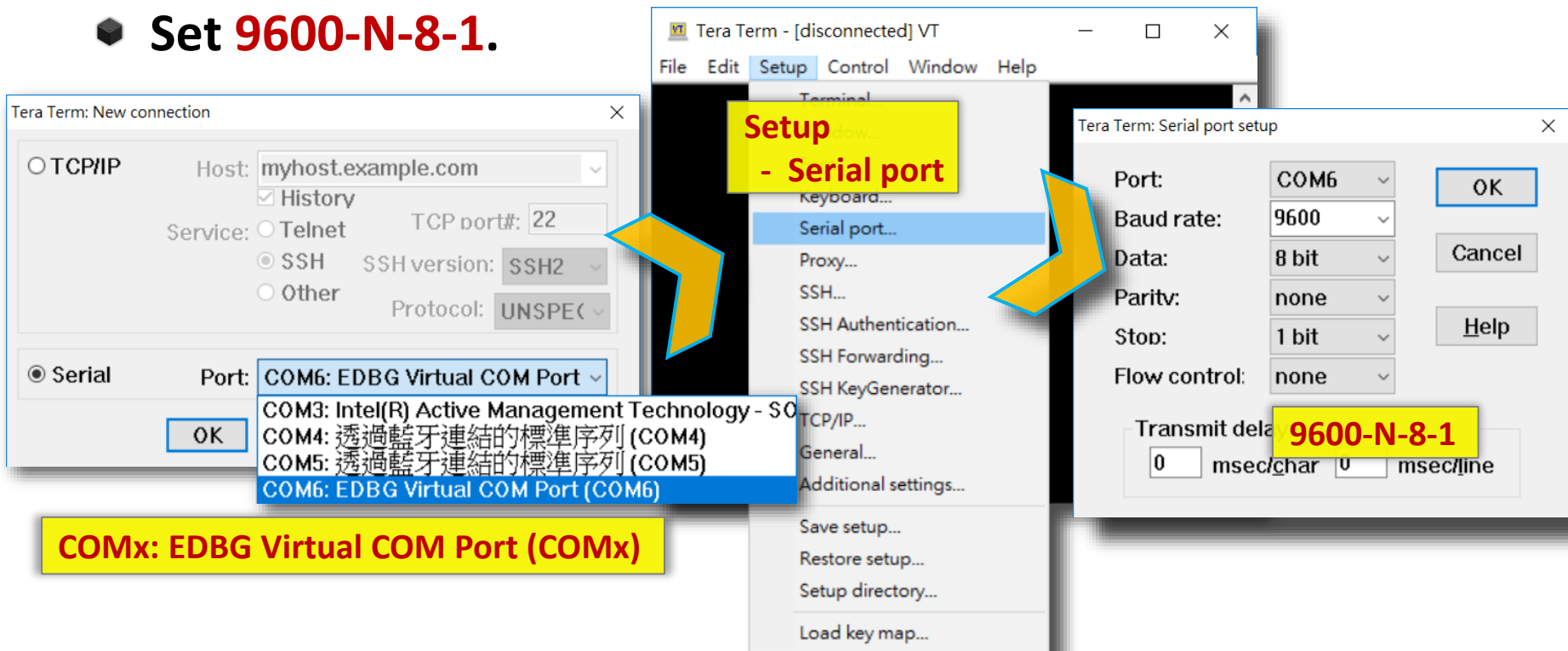

SERCOM USART Tx Polling

Code Example

```
...  
  
int main (void)  
{  
    SYS_Initialize ( NULL );  
    ...  
  
    while(1)  
    {  
        ...  
        SERCOM5_USART_Write( "Hello world!\r\n", 14 );  
    }  
}
```

Terminal Console Software

- ❖ Tera Term is a terminal console software.
(<http://ttssh2.sourceforge.jp/index.html.en>)
- ❖ Select COMx EDBG Virtual COM Port
- ❖ Set **9600-N-8-1**.



Lab9 SERCOM - UART Tx Polling

- ❖ Try to add SERCOM-USART function to project.
- ❖ UART set to **9600-N-8-1**, No flow control.
- ❖ Send string from UART to PC of **one second interval** repeated. (Hello World!)
- ❖ Character **\r** for carriage **R**eturn (0x0D)
- ❖ Character **\n** for **N**ew line (0x0A)
- ❖ **How to start ?**

Lab9 SERCOM - UART Tx Polling

Step 1

Configuration Options

SERCOM5

Select SERCOM Operation Mode: USART with internal Clock

Enable Interrupts ? ☐ **Uncheck !!**

Receive Enable ☒ **Receive Enable**

Transmit Enable ☒ **Transmit Enable**

Enable Run in Standby ☐

Receive Pinout: SERCOM PAD[3] is used for data reception **SERCOM PAD[3]**

Transmit Pinout: SERCOM PAD[2] is used for data transmission **SERCOM PAD[2]**

Parity Mode: No Parity **No Parity**

Character Size: 8 Bits **8 Bits**

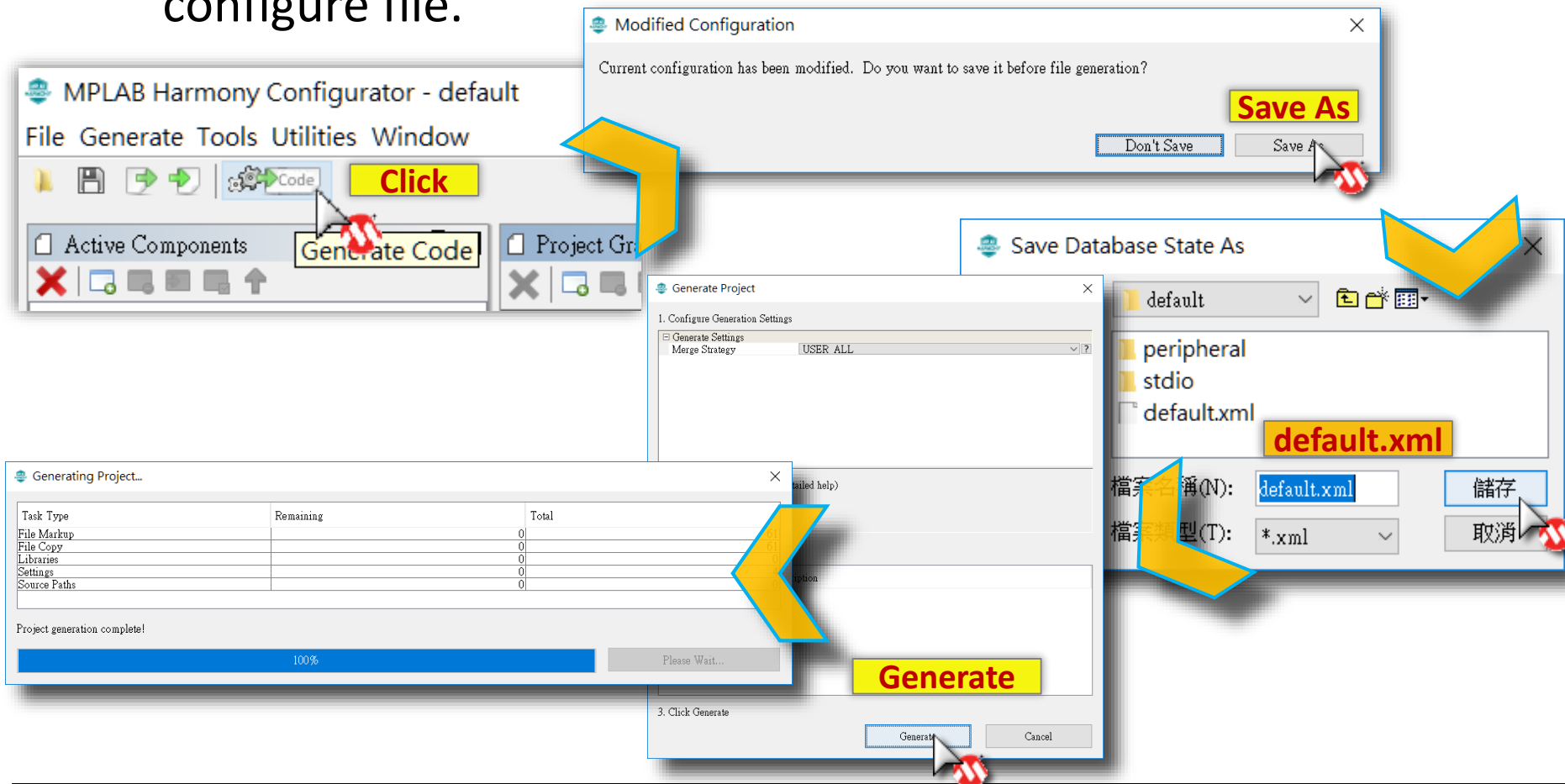
Stop Bit Mode: One Stop Bit **One Stop Bit**

Baud Rate in Hz: 9,600 **9,600**

Lab9 SERCOM - UART Tx Polling

Step 3

- Click  to Generate Code and save changes to MHC configure file.



The screenshot illustrates the process of generating code in the MPLAB Harmony Configurator. The main window shows the 'Generate Code' button in the 'Tools' menu, which is highlighted with a red circle and a yellow arrow pointing to the 'Generate Code' dialog box. The dialog box has a 'Generate' button, also highlighted with a red circle and a yellow arrow. A 'Modified Configuration' dialog box is shown, asking if the user wants to save the configuration before file generation, with a 'Save As' button highlighted. The 'Save Database State As' dialog box is also shown, with the 'default.xml' file selected and the '儲存' (Save) button highlighted. The 'Generating Project...' dialog box shows the progress of the generation process, with a '100%' completion bar and a 'Please Wait...' button. The 'Generate Project' dialog box shows the 'Generate Settings' section with 'Merge Strategy' set to 'USER ALL'.

MPLAB Harmony Configurator - default

File Generate Tools Utilities Window

Active Components

Project Generator

Modified Configuration

Current configuration has been modified. Do you want to save it before file generation?

Save As

Don't Save

Save

Save Database State As

default

peripheral

stdio

default.xml

default.xml

*.xml

儲存

取消

Generating Project...

Task Type	Remaining	Total
File Markup	0	0
File Copy	0	0
Libraries	0	0
Settings	0	0
Source Paths	0	0

Project generation complete!

100%

Please Wait...

Generate

Cancel

Lab9 SERCOM - UART Tx Polling

Step 4

a Add code segment to your main loop

```
void TC3_Overflow(TC_COMPARE_STATUS status, uintptr_t context)
{
    if( status & TC_INTFLAG_OVF_Msk )
    {
        LED1_Toggle();

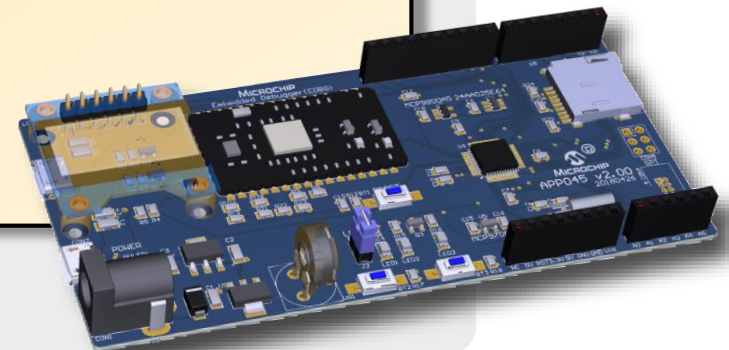
        // TODO 9.01
        SERCOM5_USART_Write( "Hello world!\r\n", 14 );
    }
}

int main (void)
{
    SYS_Initialize ( NULL );

    ...

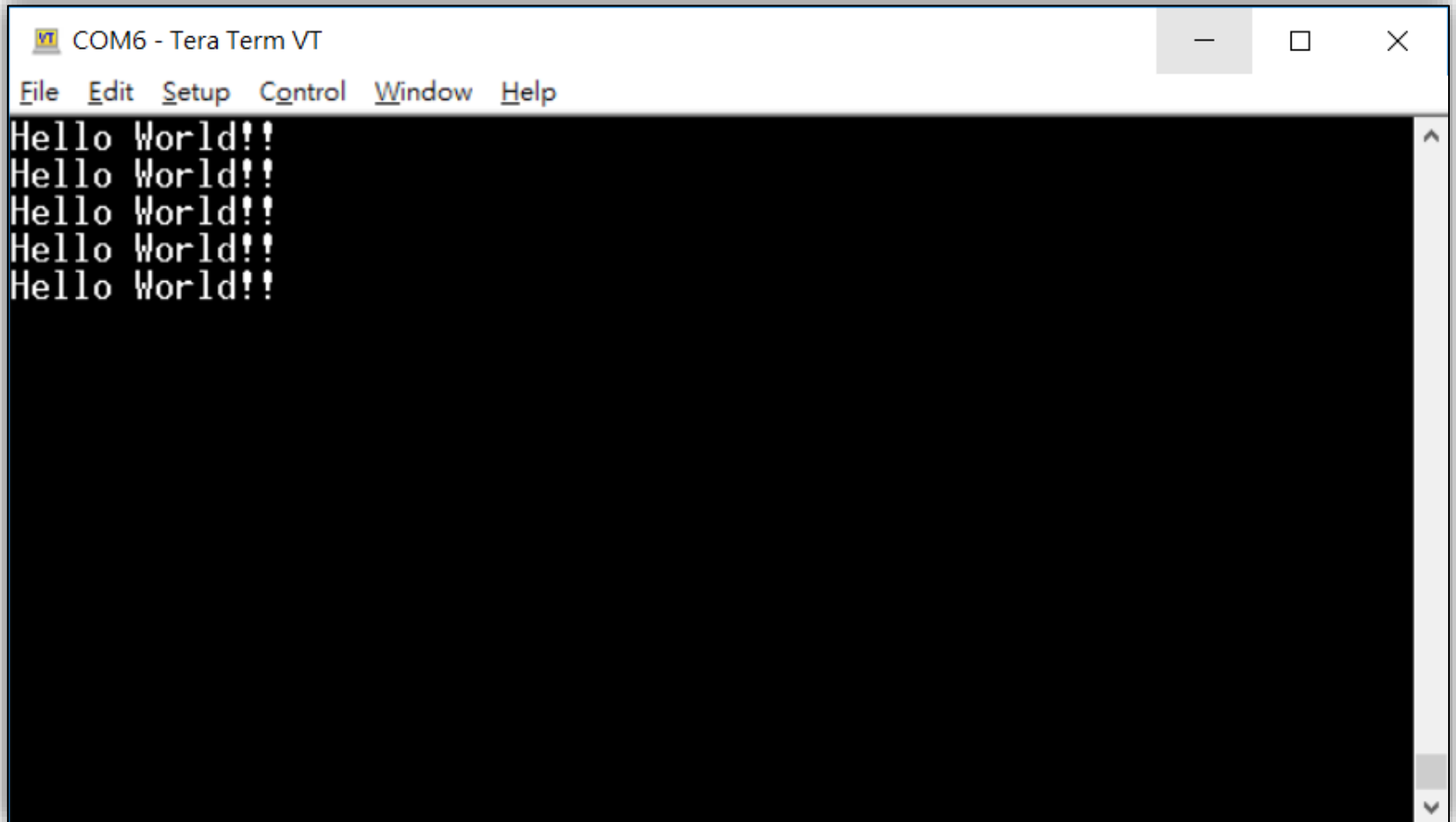
}
```

b Program firmware to target board then observe result.



Lab9 SERCOM - UART Tx Polling

Result

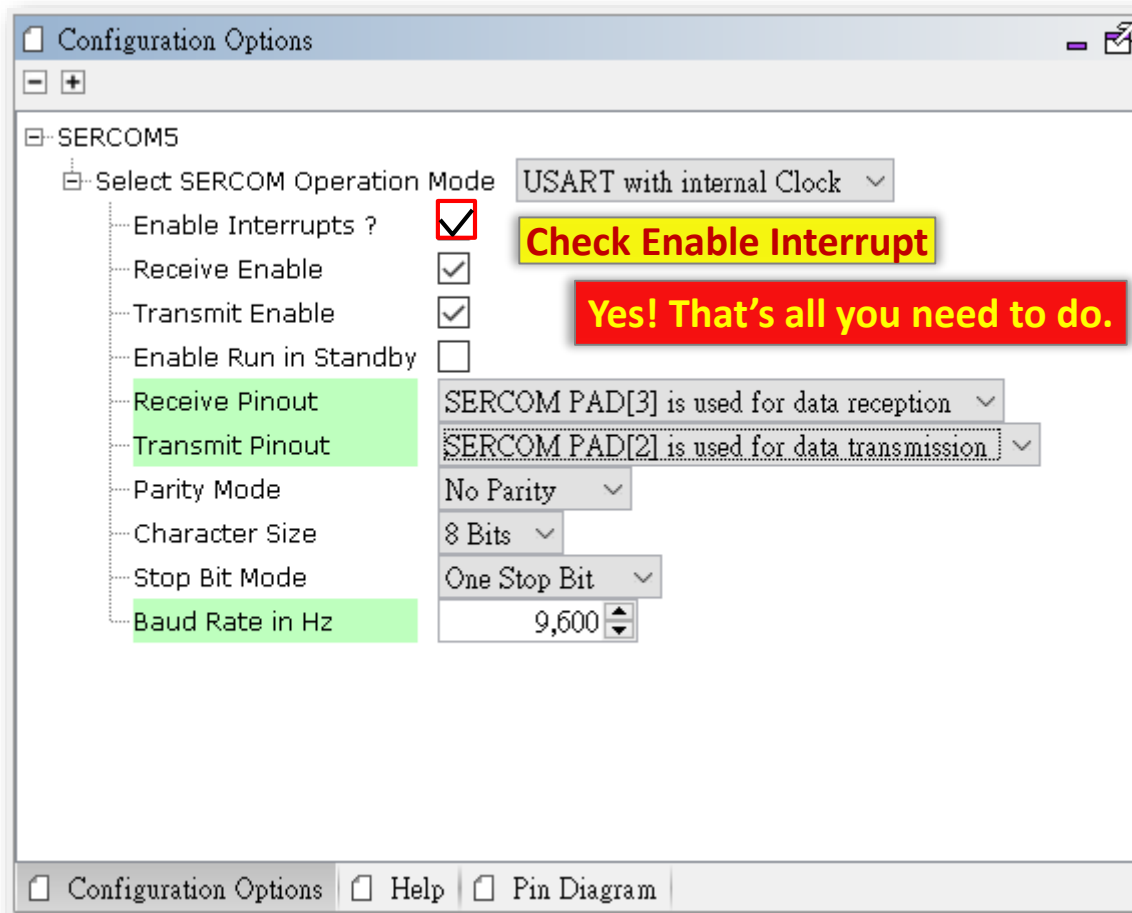


The image shows a screenshot of a Tera Term VT window titled "COM6 - Tera Term VT". The window has a menu bar with "File", "Edit", "Setup", "Control", "Window", and "Help". The main display area is black with white text. It shows five lines of "Hello World!!" text, indicating successful UART transmission. A vertical scrollbar is visible on the right side of the window.

```
COM6 - Tera Term VT
File Edit Setup Control Window Help
Hello World!!
Hello World!!
Hello World!!
Hello World!!
Hello World!!
```


SERCOM USART Interrupt Configuration Options Example

- SERCOM provide callback(interrupt) mode, also.



SERCOM USART Interrupt Callback

Interface Routines

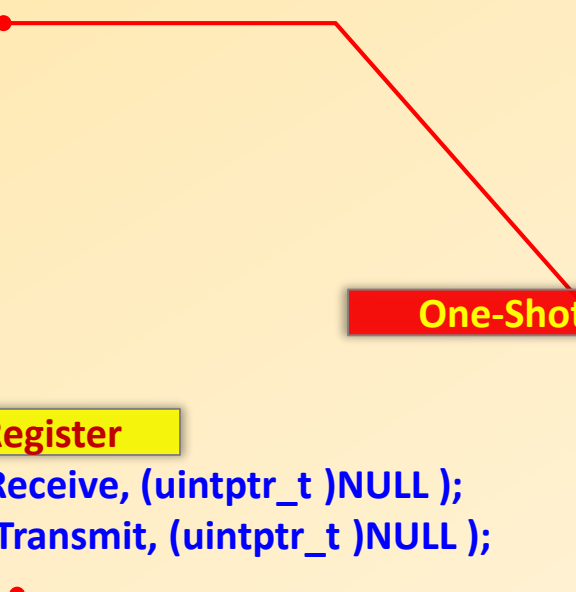
```
/** SERCOM USART Interface Routines */  
void SERCOM5_USART_Initialize( void );  
bool SERCOM5_USART_SerialSetup( USART_SERIAL_SETUP * serialSetup,  
                                uint32_t clkFrequency );  
bool SERCOM5_USART_Write( void *buffer, const size_t size );  
bool SERCOM5_USART_WritelsBusy( void );  
size_t SERCOM5_USART_WriteCountGet( void );  
void SERCOM5_USART_WriteCallbackRegister( SERCOM_USART_CALLBACK callback,  
                                           uintptr_t context );  
bool SERCOM5_USART_Read( void *buffer, const size_t size );  
bool SERCOM5_USART_ReadlsBusy( void );  
size_t SERCOM5_USART_ReadCountGet( void );  
void SERCOM5_USART_ReadCallbackRegister( SERCOM_USART_CALLBACK callback,  
                                           uintptr_t context );  
USART_ERROR SERCOM5_USART_ErrorGet( void );  
uint32_t SERCOM5_USART_FrequencyGet( void );
```

USART TxRx Interrupt Code Example

```
uint8_t USART5_ReceiveData[1];
void USART5_Receive( uintptr_t context )
{
    SERCOM5_USART_Write( USART5_ReceiveData, 1 );
    SERCOM5_USART_Read( USART5_ReceiveData, 1 );
}

void USART5_Transmit( uintptr_t context ) { }

int main (void)
{
    SYS_Initialize ( NULL );
    ...
    SERCOM5_USART_ReadCallbackRegister( USART5_Receive, (uintptr_t )NULL );
    SERCOM5_USART_WriteCallbackRegister( USART5_Transmit, (uintptr_t )NULL );
    SERCOM5_USART_Read( USART5_ReceiveData, 1 );
    while(1)
    { ...
        SERCOM5_USART_Write( "Hello world!\r\n", 14 );
    }
}
```



One-Shot Event

Callback Register

About String Functions

- 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printf(), strlen()。
- `int sprintf(char *s, const char *format, ...);`
將資料依據指定格式轉換後存入指定的buffer字串中。
* 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Null('\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);  
SERCOM5_USART_Write( USARTRTxBuffer, strlen((char *)USARTRTxBuffer));
```

Lab10 SERCOM - UART TxRx

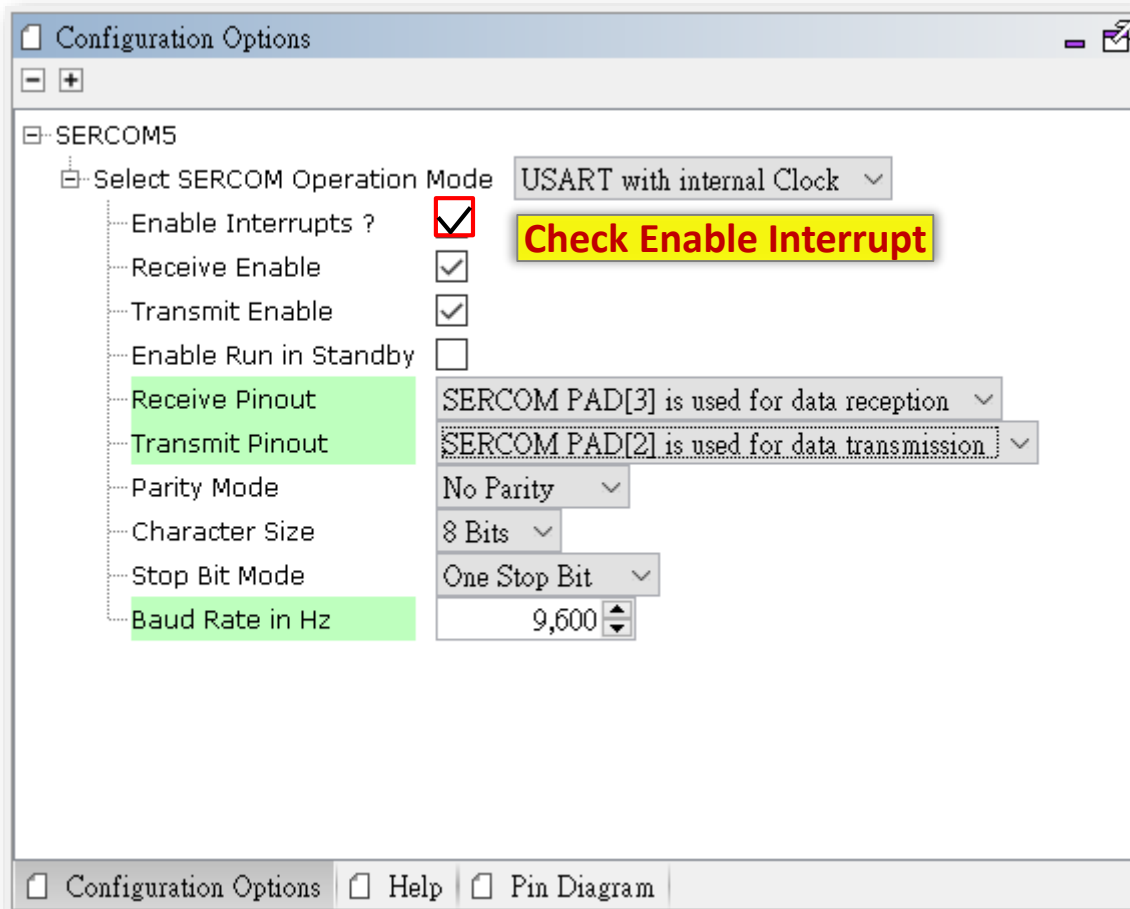
Interrupt Callback

- ❖ Try to enable SERCOM-USART **Interrupt** Callback driver.
- ❖ Modify polling transmit function to asynchronous function.
- ❖ Add new USART receive function base on interrupt callback function.
- ❖ Try to receive character from PC and loopback to TeraTerm as string format “**Received Data : x**” .
- ❖ SERCOM - UART setting same as Lab9

❖ **Let's go!**

Lab10 SERCOM - UART TxRx

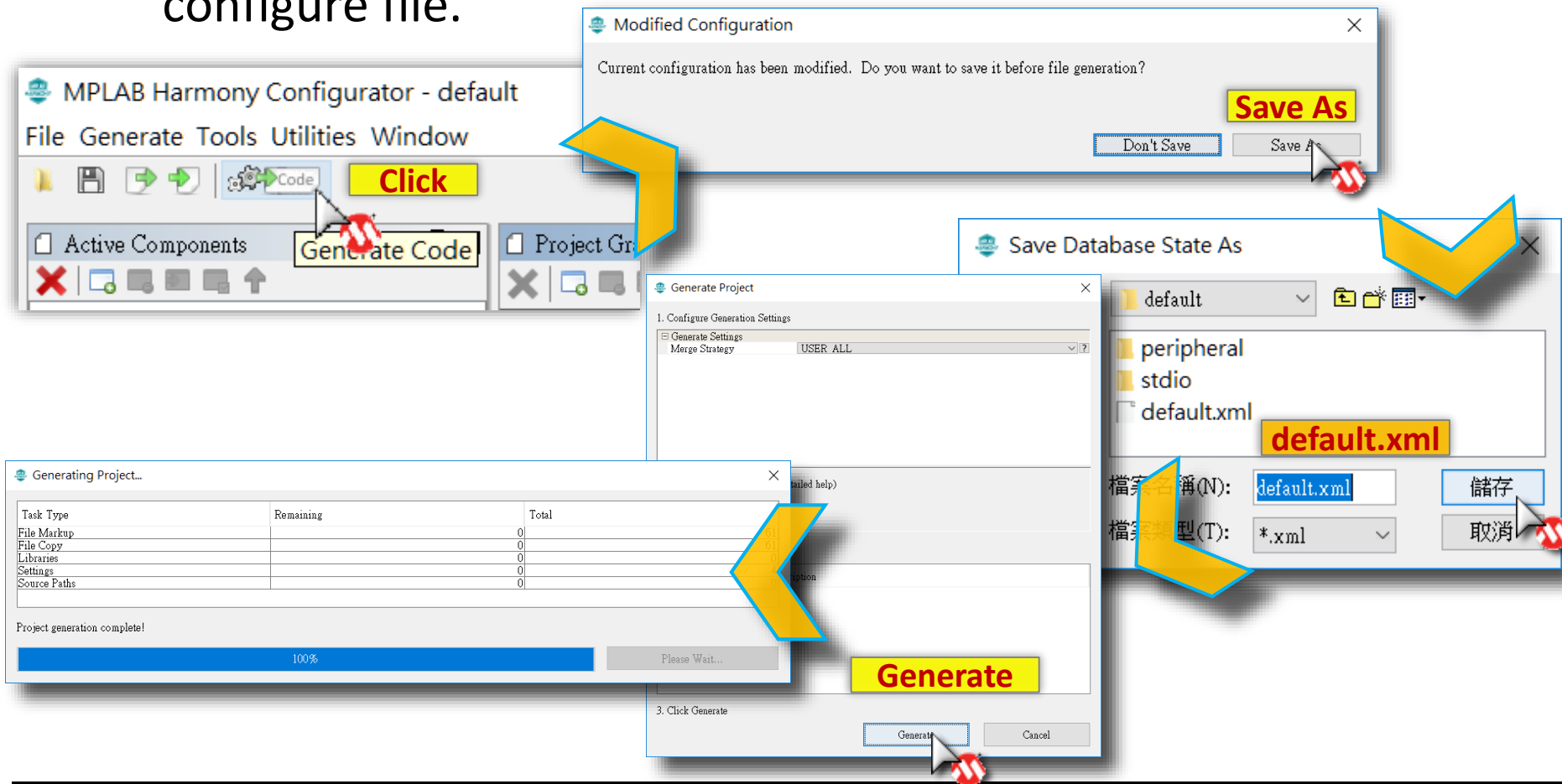
Interrupt Callback Step 1



Lab10 SERCOM - UART TxRx

Interrupt Callback step 2

- Click  **Code** to Generate Code and save changes to MHC configure file.



The image illustrates the process of generating code in the MPLAB Harmony Configurator. The main window shows the 'Generate Code' button, which is highlighted with a red circle and a yellow arrow. A yellow arrow points from the 'Generate Code' button to the 'Modified Configuration' dialog box, which asks 'Current configuration has been modified. Do you want to save it before file generation?'. The 'Save As' button is highlighted with a red circle and a yellow arrow. A yellow arrow points from the 'Save As' button to the 'Save Database State As' dialog box, which shows the 'default.xml' file selected. A yellow arrow points from the 'Save Database State As' dialog box to the 'Generate Project' dialog box, which shows the 'Generate' button highlighted with a red circle and a yellow arrow. A yellow arrow points from the 'Generate Project' dialog box to the 'Generating Project...' progress window, which shows the progress of the generation process.

MPLAB Harmony Configurator - default

File Generate Tools Utilities Window

Active Components

Project Generator

Click

Generate Code

Modified Configuration

Current configuration has been modified. Do you want to save it before file generation?

Save As

Don't Save Save

Save Database State As

default

peripheral

stdio

default.xml

default.xml

檔案名稱(N): default.xml

檔案類型(T): *.xml

儲存

取消

Generate Project

1. Configure Generation Settings

Generate Settings

Merge Strategy USER ALL

Generate

Generating Project...

Task Type	Remaining	Total
File Markup	0	0
File Copy	0	0
Libraries	0	0
Settings	0	0
Source Paths	0	0

Project generation complete!

100%

Please Wait...

Lab10 SERCOM - UART TxRx

Interrupt Callback Step 3

a

Create UART TxRx Callback function, related buffer and

```
// TODO 10.01
#include <stdio.h>
#include <string.h>

// TODO 10.02
uint8_t USARTRTxBuffer[100];
uint8_t USART5_ReceiveData[1];
volatile uint8_t USART5_IsReceived = 0;
volatile uint8_t USART5_IsTransmitted = 1;

void USART5_Transmit( uintptr_t context )
{
    USART5_IsTransmitted = 1;
}

void USART5_Receive( uintptr_t context )
{
    USART5_IsReceived = 1;
}

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


Lab10 SERCOM - UART TxRx

Interrupt Callback Step 4

a Add code segment to your main loop

```
int main (void)
{
    ...
    // TODO 10.03
    SERCOM5_USART_ReadCallbackRegister( USART5_Receive, (uintptr_t )NULL );
    SERCOM5_USART_WriteCallbackRegister( USART5_Transmit, (uintptr_t )NULL );
    SERCOM5_USART_Read( USART5_ReceiveData, 1 );

    while(1)
    {
        ...
        // TODO 10.04
        if( USART5_IsReceived )
        {
            USART5_IsReceived = 0;
            sprintf((char *) USARTRTxBuffer,
                    "\r\nReceived Data : %1c\r\n", USART5_ReceiveData[0] );
            SERCOM5_USART_Write( USARTRTxBuffer, strlen((char*)USARTRTxBuffer) );
            while( SERCOM5_USART_WriteIsBusy() );
            SERCOM5_USART_Read( USART5_ReceiveData, 1 );
        }
    }
}
```

Lab10 SERCOM - UART TxRx

Interrupt Callback Step 5

a Modify TC3 Callback for UART Tx

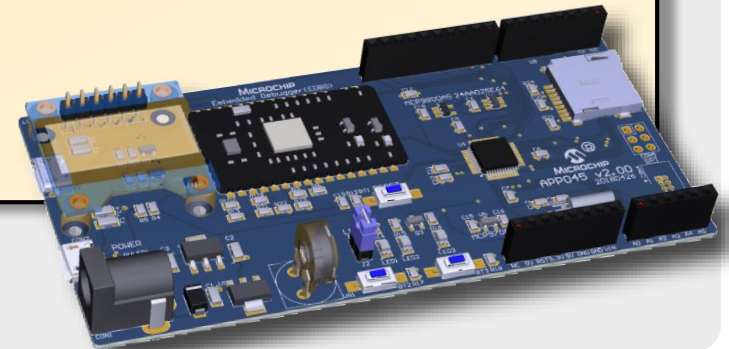
```
void TC3_Overflow(TC_COMPARE_STATUS status, uintptr_t context)
{
    if( status & TC_INTFLAG_OVF_Msk )
    {
        LED1_Toggle();

        // TODO 10.05
        sprintf((char *) USARTRTxBuffer, "Hello world!\r\n" );
        SERCOM5_USART_Write( USARTRTxBuffer, strlen((char*)USARTRTxBuffer) );
    }
}

int main (void)
{
    SYS_Initialize ( NULL );

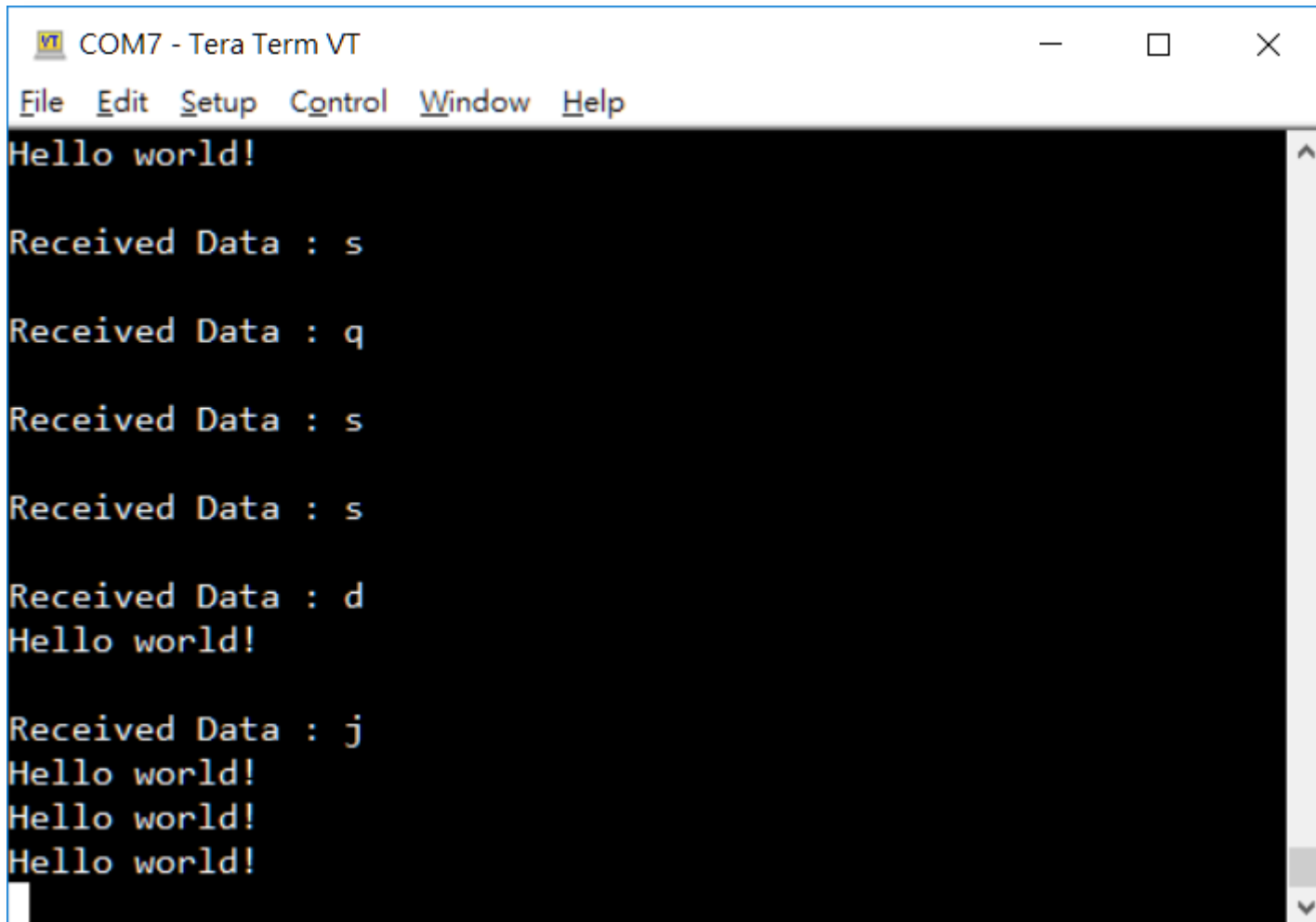
    ...
}
```

b Program firmware to target board then observe result.



Lab10 SERCOM - UART TxRx

Interrupt Callback Result



The screenshot shows a Tera Term VT window titled 'COM7 - Tera Term VT'. The window has a menu bar with 'File', 'Edit', 'Setup', 'Control', 'Window', and 'Help'. The main text area displays the following sequence of messages:

```
Hello world!  
  
Received Data : s  
  
Received Data : q  
  
Received Data : s  
  
Received Data : s  
  
Received Data : d  
Hello world!  
  
Received Data : j  
Hello world!  
Hello world!  
Hello world!
```

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

Lab11 SERCOM - UART TxRx

Interrupt Callback with STDIO

- Try add **STDIO** interface for SERCOM-USART
- After done this, you could use **printf()** function to instead of USART interface routines.

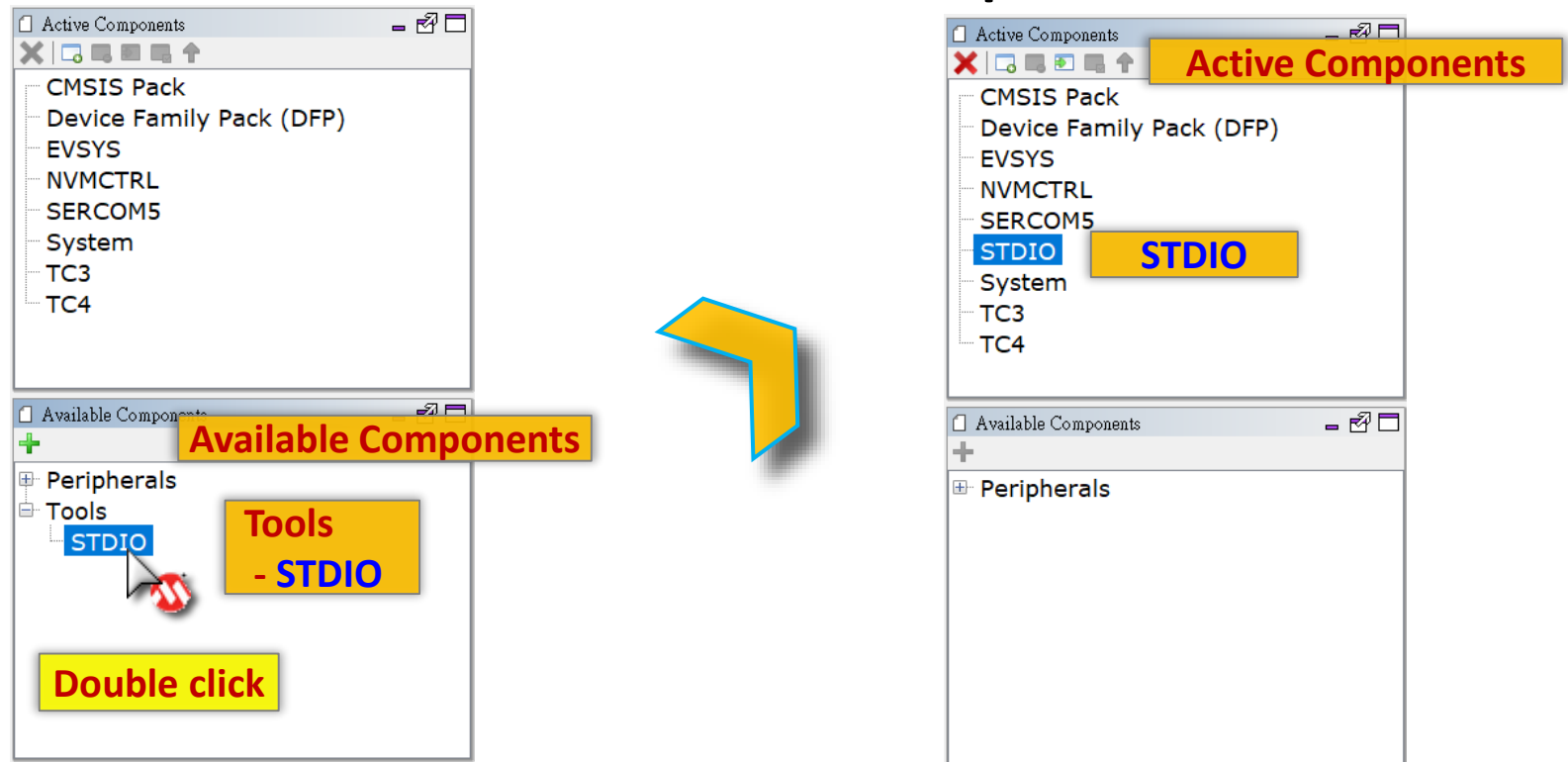
 **Let's go!**

Lab11 SERCOM - UART TxRx

Interrupt Callback with STDIO Step 1

Add STDIO Function using MHC

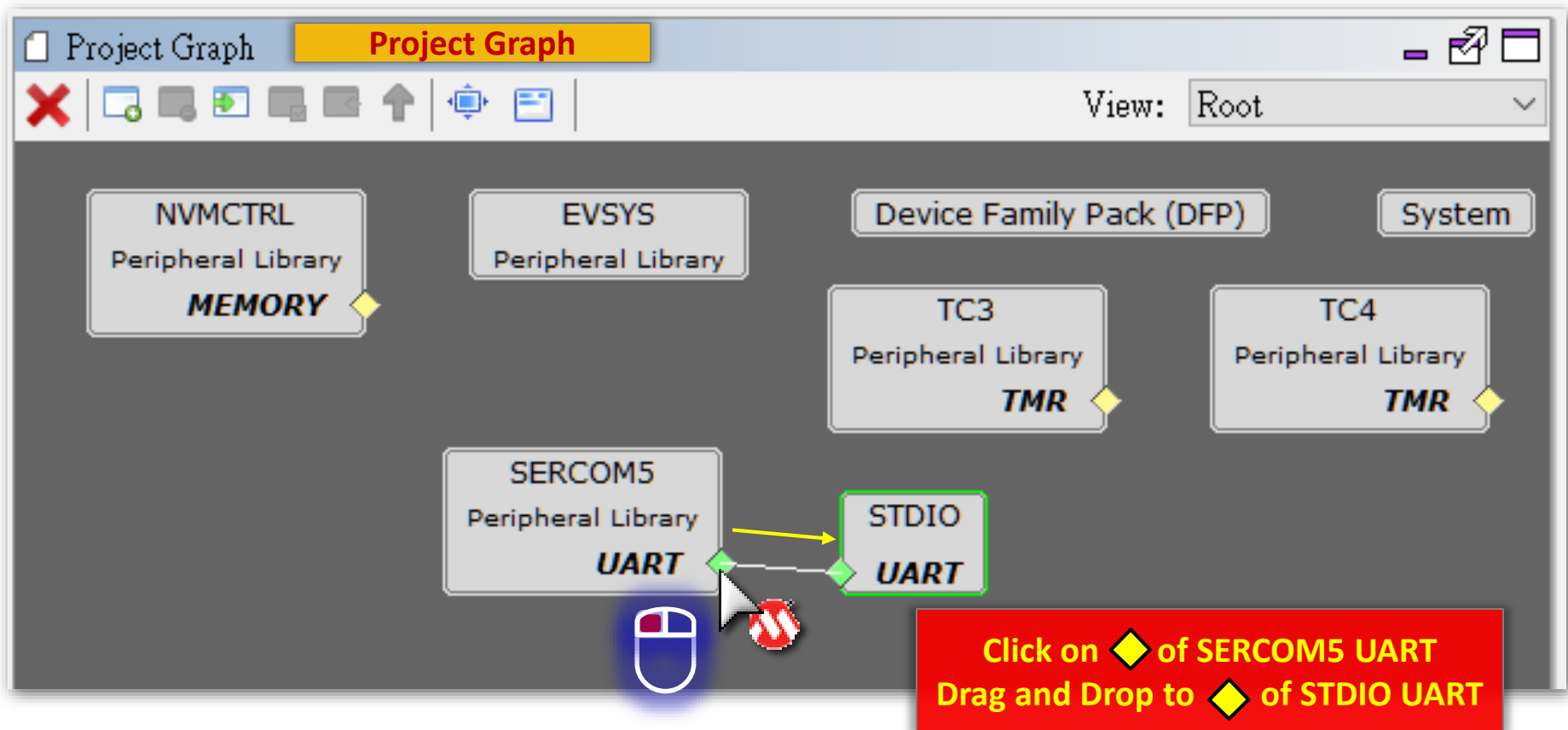
- Find **Tools** ► **STDIO** component in Available Components window.
- Double click **STDIO** to add to Active Components window.



Lab11 SERCOM - UART TxRx


Interrupt Callback with STDIO Step 2

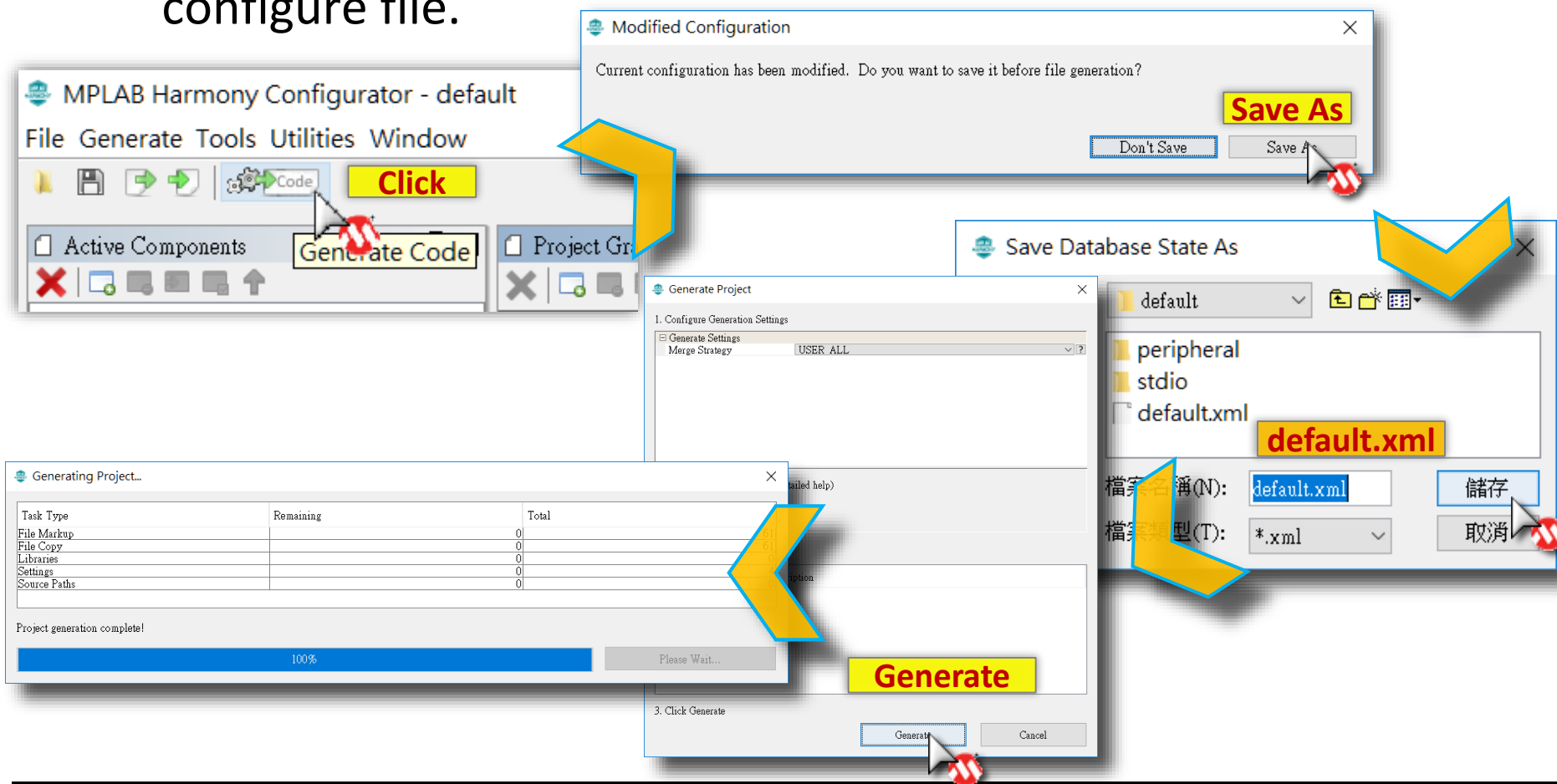
Project Graph Connection



Lab11 SERCOM - UART TxRx

Interrupt Callback with STDIO Step 3

- Click  **Code** to Generate Code and save changes to MHC configure file.



Lab11 SERCOM - UART TxRx

Interrupt Callback with STDIO Step 4

a Add software flag control of TC3 and TC4.

```
// TODO 11.01
volatile uint8_t TC3_IsOverflow = 0;
volatile uint8_t TC4_IsOverflow = 0;

...
void TC3_Overflow(TC_COMPARE_STATUS status, uintptr_t context) /* Callback */
{
    if( status & TC_INTFLAG_OVF_Msk )
    {
        // TODO 11.02
        TC3_IsOverflow = 1;
    }
}

void TC4_Overflow(TC_COMPARE_STATUS status, uintptr_t context) /* Callback */
{
    if( status & TC_INTFLAG_OVF_Msk )
    {
        // TODO 11.03
        TC4_IsOverflow = 1;
    }
}
```


Lab11 SERCOM - UART TxRx

Interrupt Callback with STDIO Step 5

a Add code segment to your main while loop.

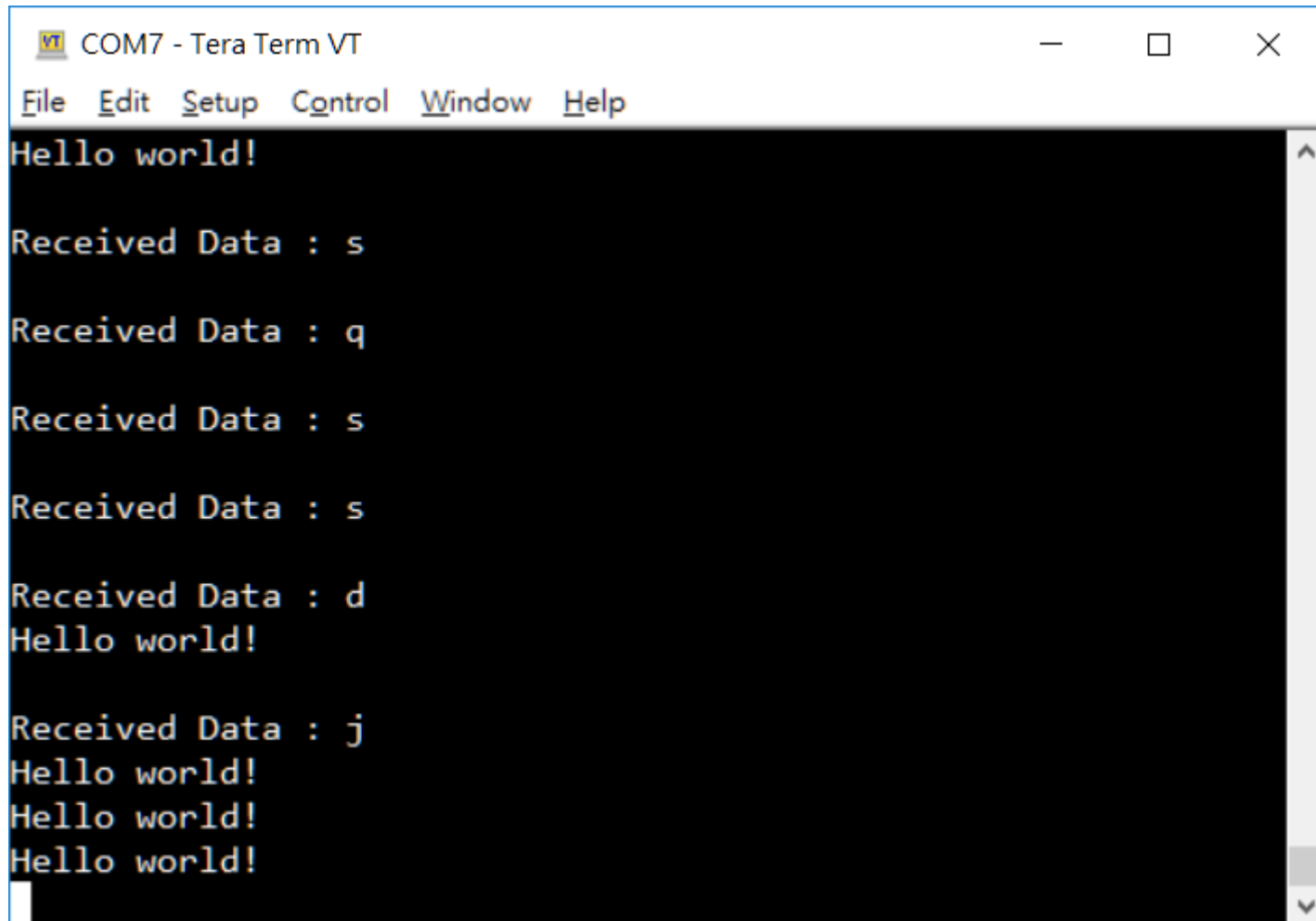
```
while(1)
{
    // TODO 11.04
    if (TC3_IsOverflow)
    {
        TC3_IsOverflow = 0;
        LED1_Toggle();
        printf( "Hello World!!\r\n" );
    }

    if (TC4_IsOverflow)
    {
        TC4_IsOverflow = 0;
        LED2_Toggle();
    }

    if( USART5_IsReceived )
    {
        // TODO 11.05
        printf( "\r\nReceived Data : %1c\r\n", USART5_ReceiveData[0] );
        ...
    }
}
```

Lab11 SERCOM - UART TxRx

Interrupt Callback with STDIO Result



The screenshot shows a Tera Term VT window titled "COM7 - Tera Term VT". The window has a menu bar with "File", "Edit", "Setup", "Control", "Window", and "Help". The main text area displays the following output:

```
Hello world!  
  
Received Data : s  
  
Received Data : q  
  
Received Data : s  
  
Received Data : s  
  
Received Data : d  
Hello world!  
  
Received Data : j  
Hello world!  
Hello world!  
Hello world!
```

Bonus Lab

- Input “LED1” in TeraTerm will toggle LED1
- Input “LED2” in TeraTerm will toggle LED2
- Input “LED3” in TeraTerm will toggle LED3

■ **Let's go!**