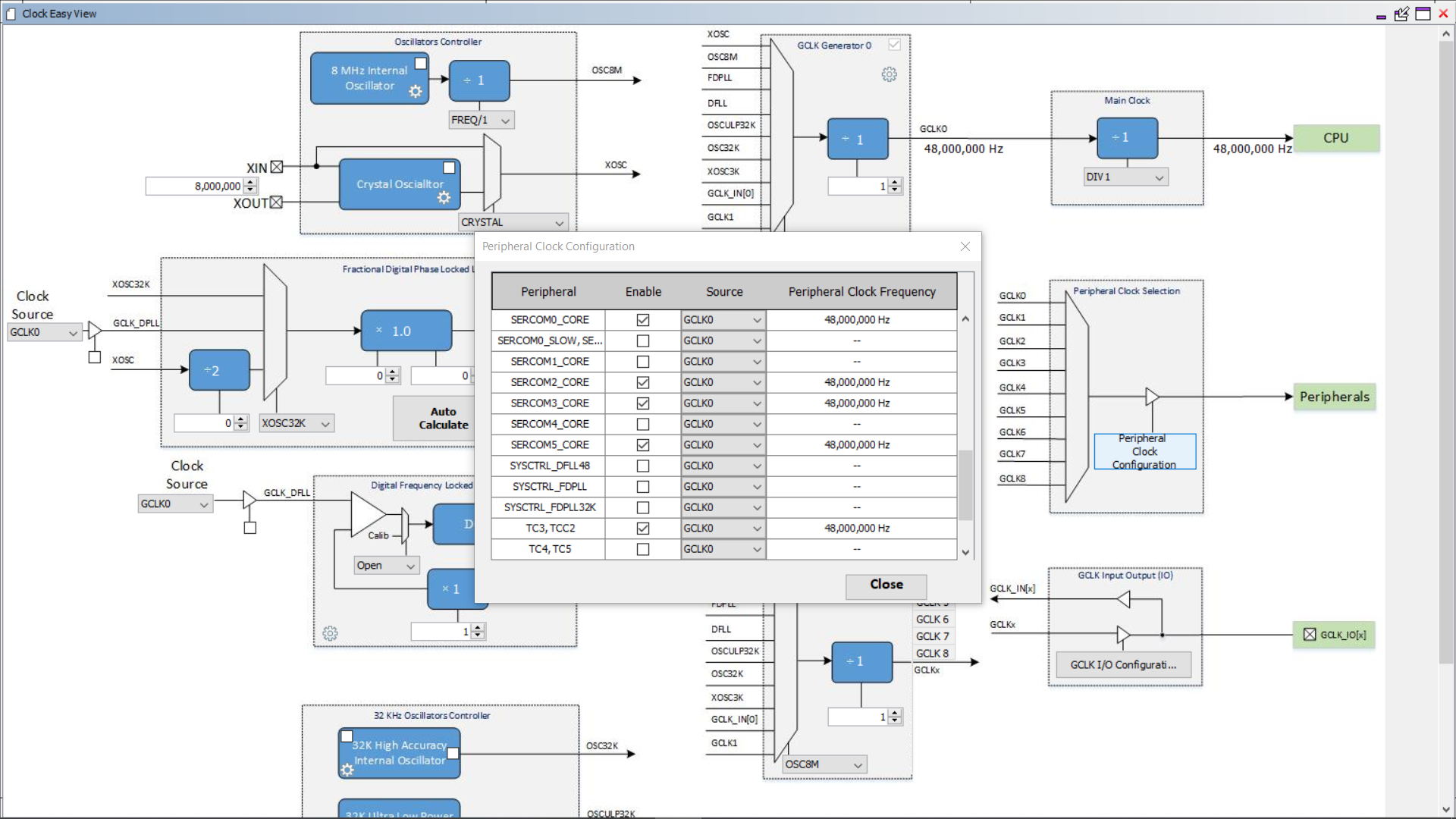
**UART**

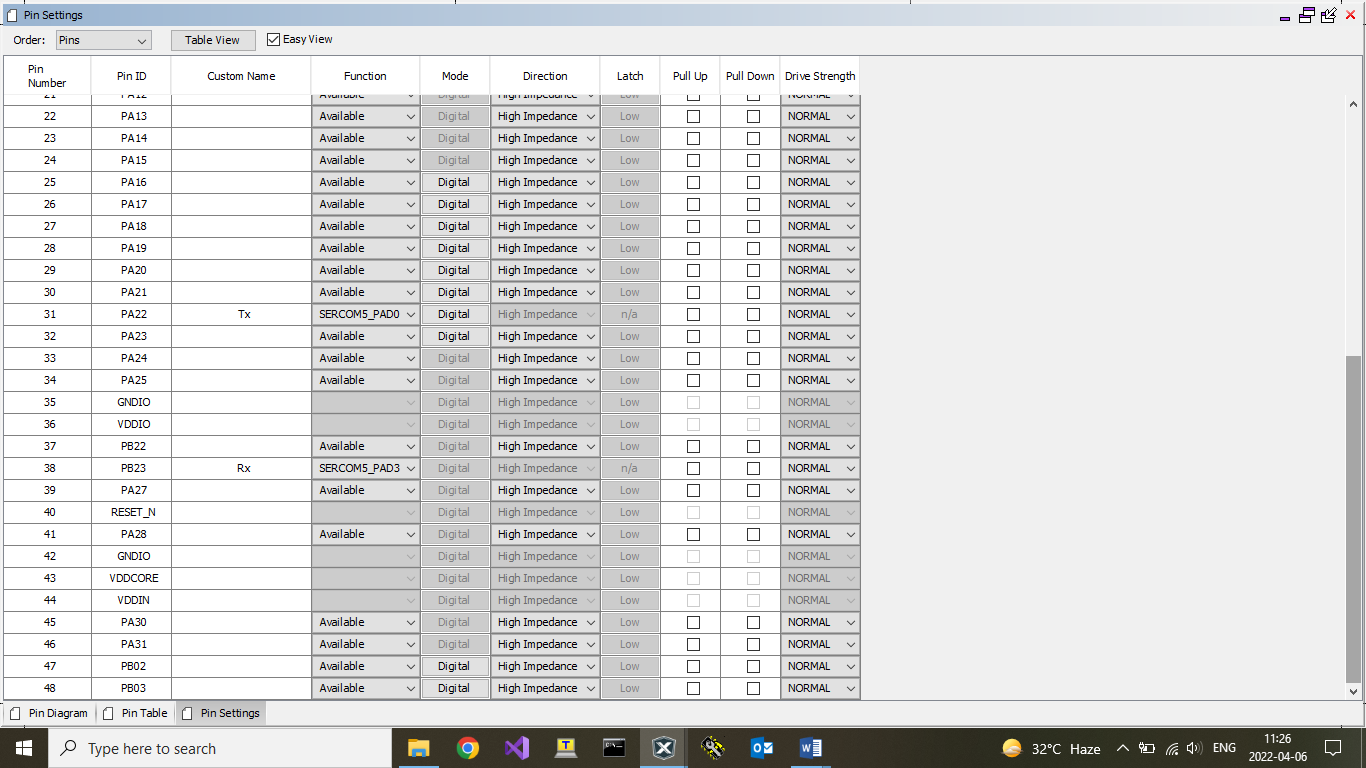
**UART – Polling Method**

Step 1: Create a MPLAB Project and Open MHC (Refer to Creating First project).

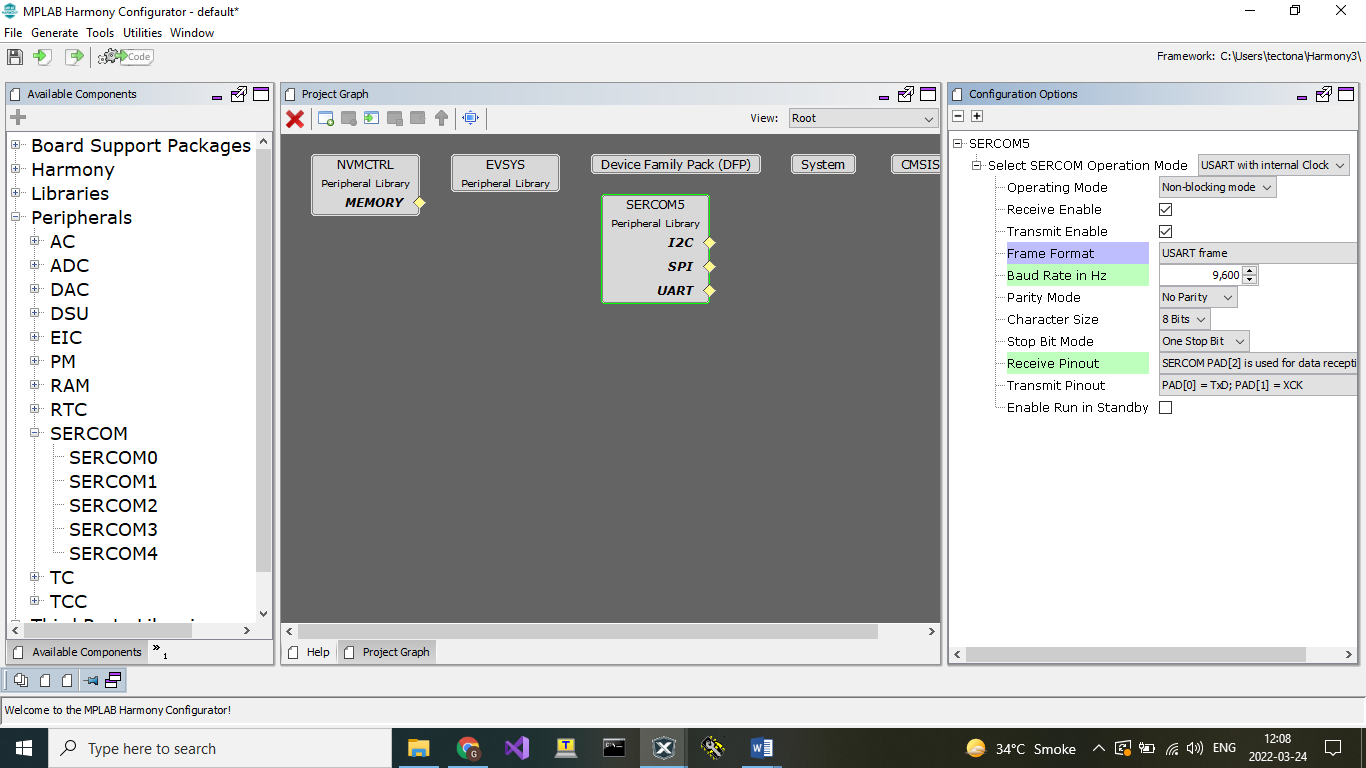
Step 2: Clock Configuration:



Step 3: Set PA22 as Transmit and PB22 as Receive. Change the Custom name as you want.



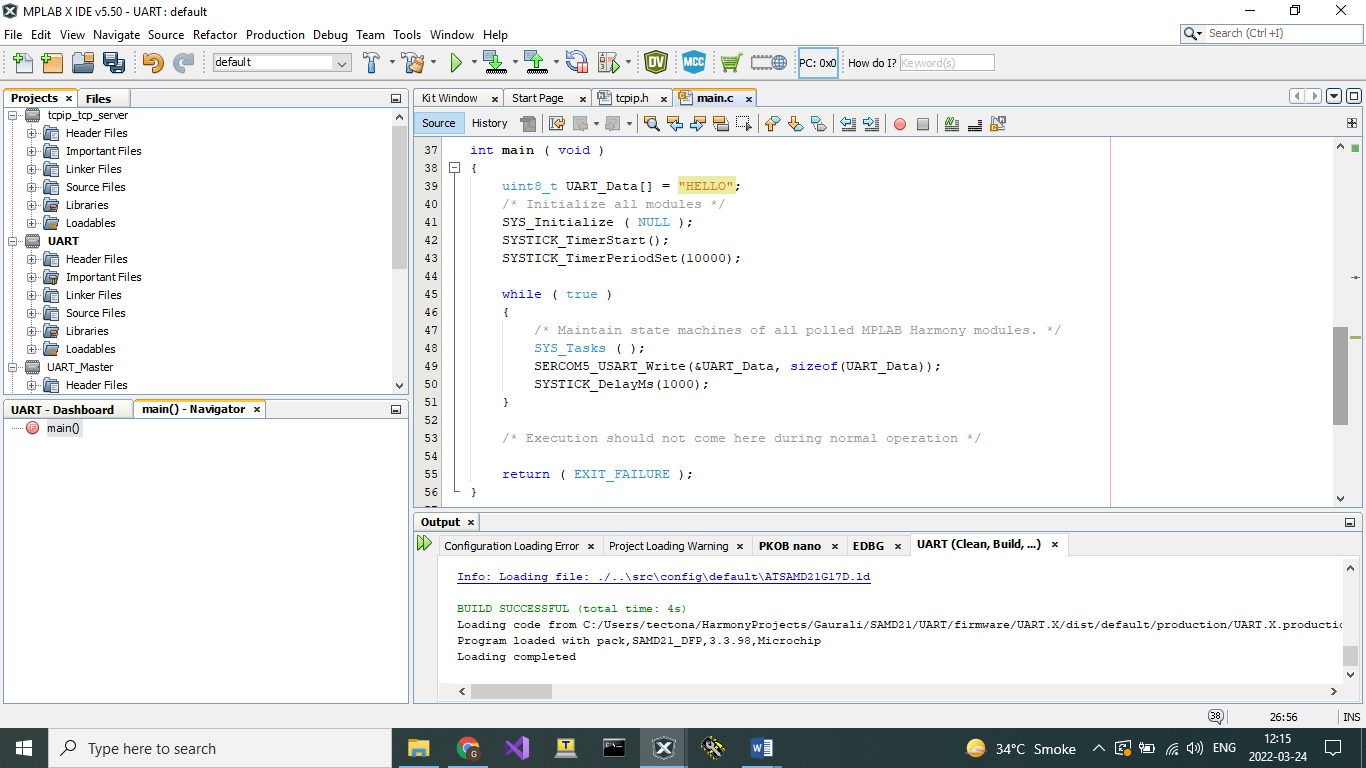
Step 4: Click on Peripherals, Select SERCOM5. Set the configurations as shown below.

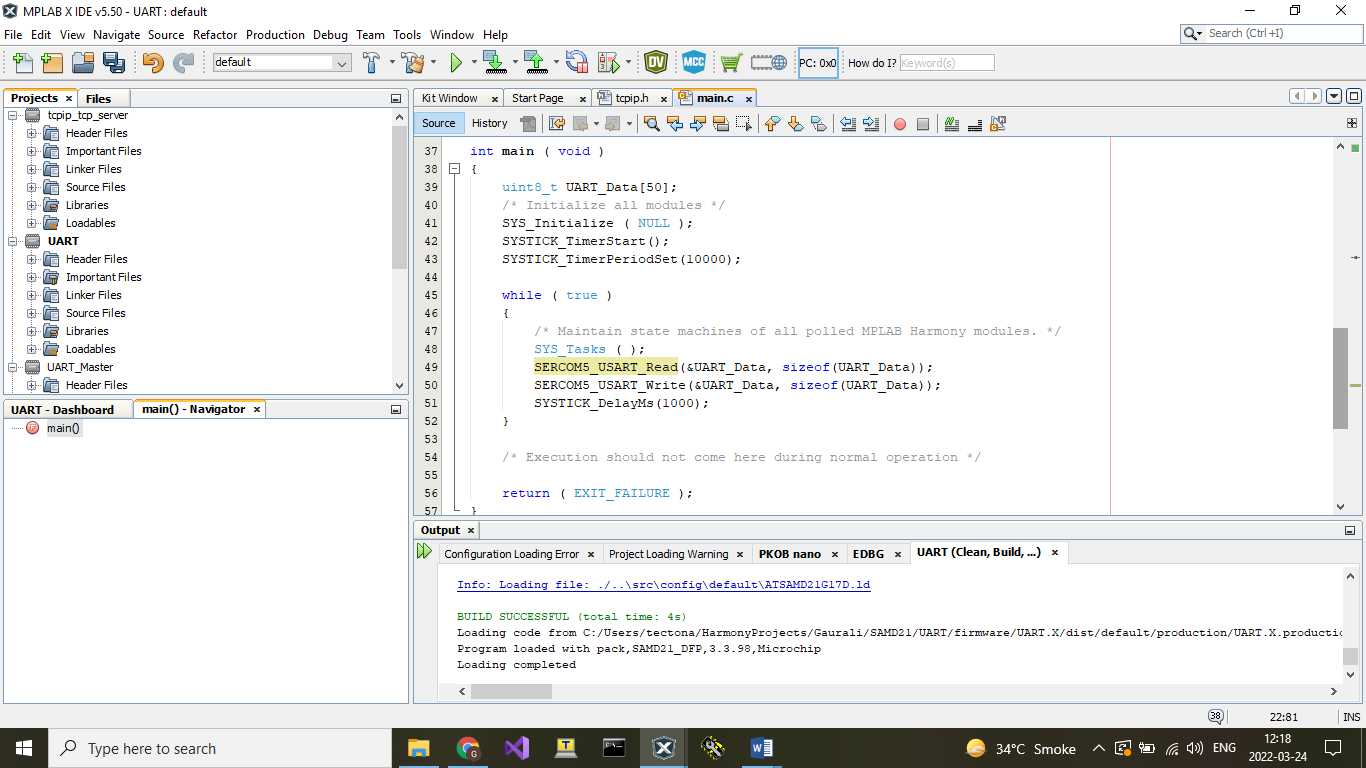


Step 5: Click on Generate Code.

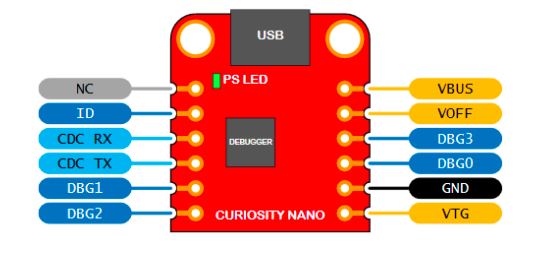
You should be able to see MHC generated files under Project->Source Files->config->default->peripheral->usart->plib\_sercom5\_usart.c.

Code:





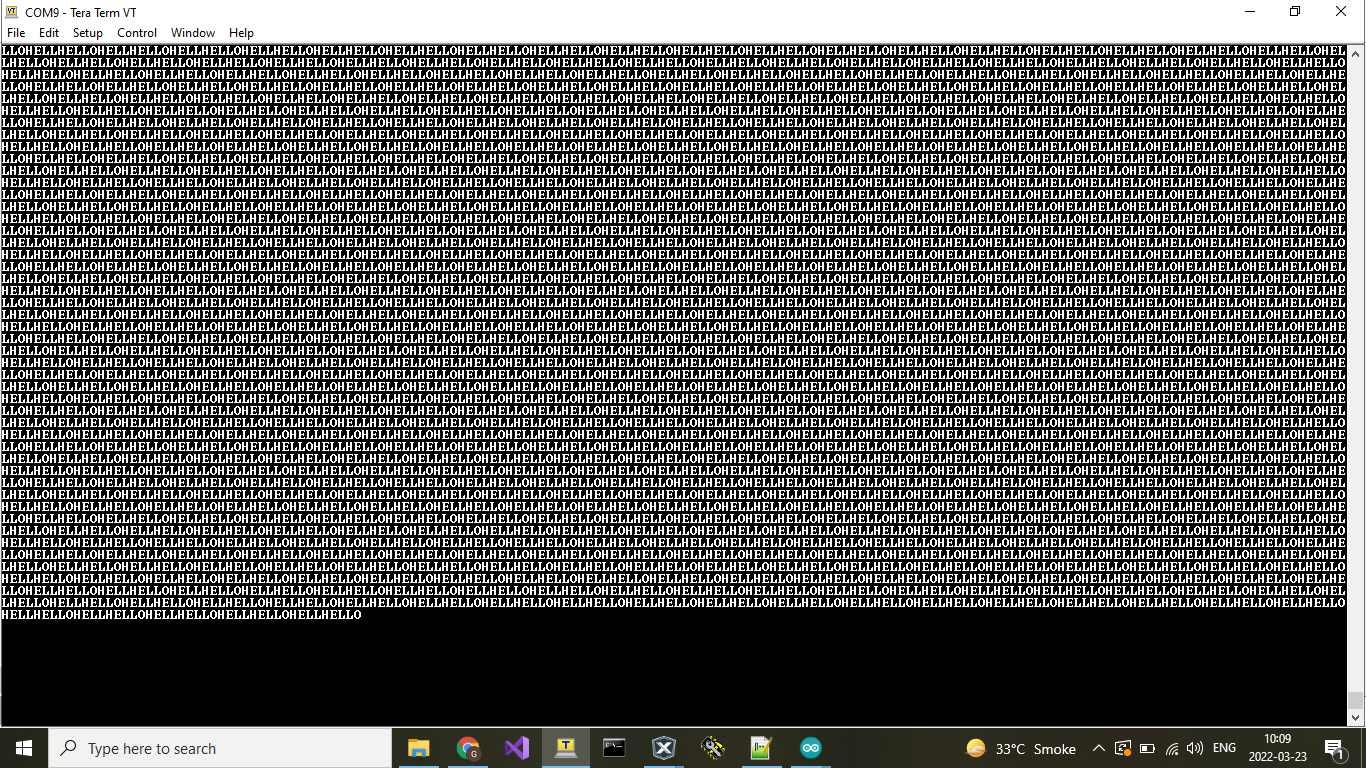
PA22 and PB22 are CDC Rx (UART Tx) and CDC Tx (UART Rx) respectively.



Connect Tx pin of one board with Rx pin of the other and vice versa.

You can see the output on TeraTerm (if SAMD21 is used to read) or on Arduino serial com (if Arduino is used to read).

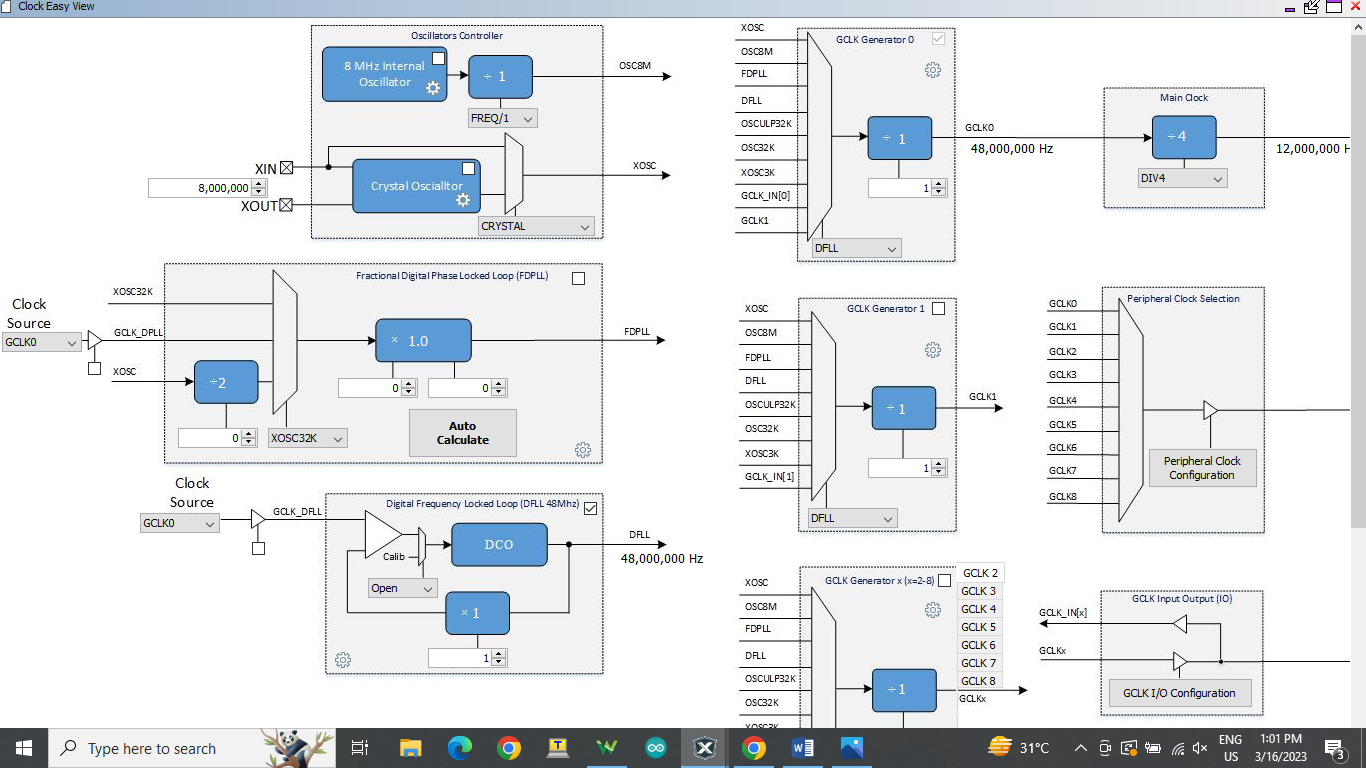
Output:



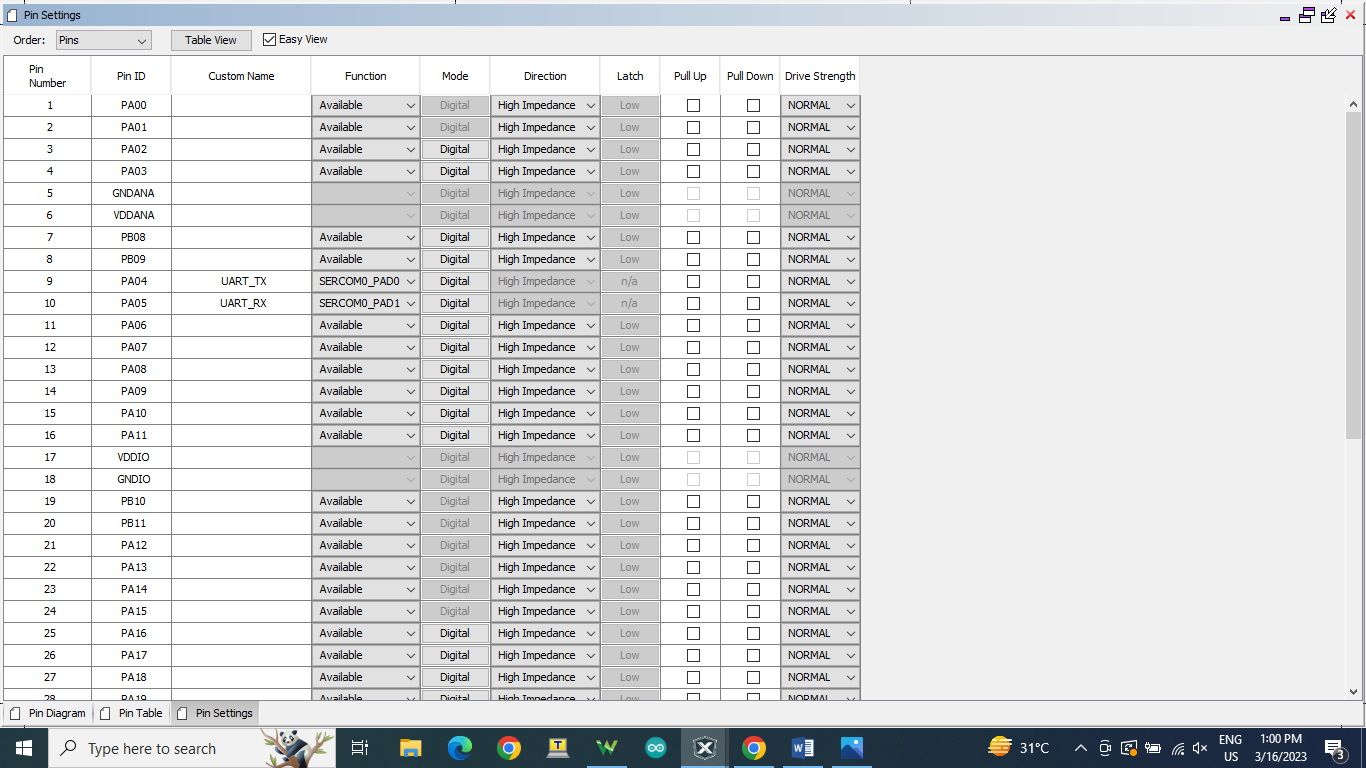
**UART –Interrupt Method**

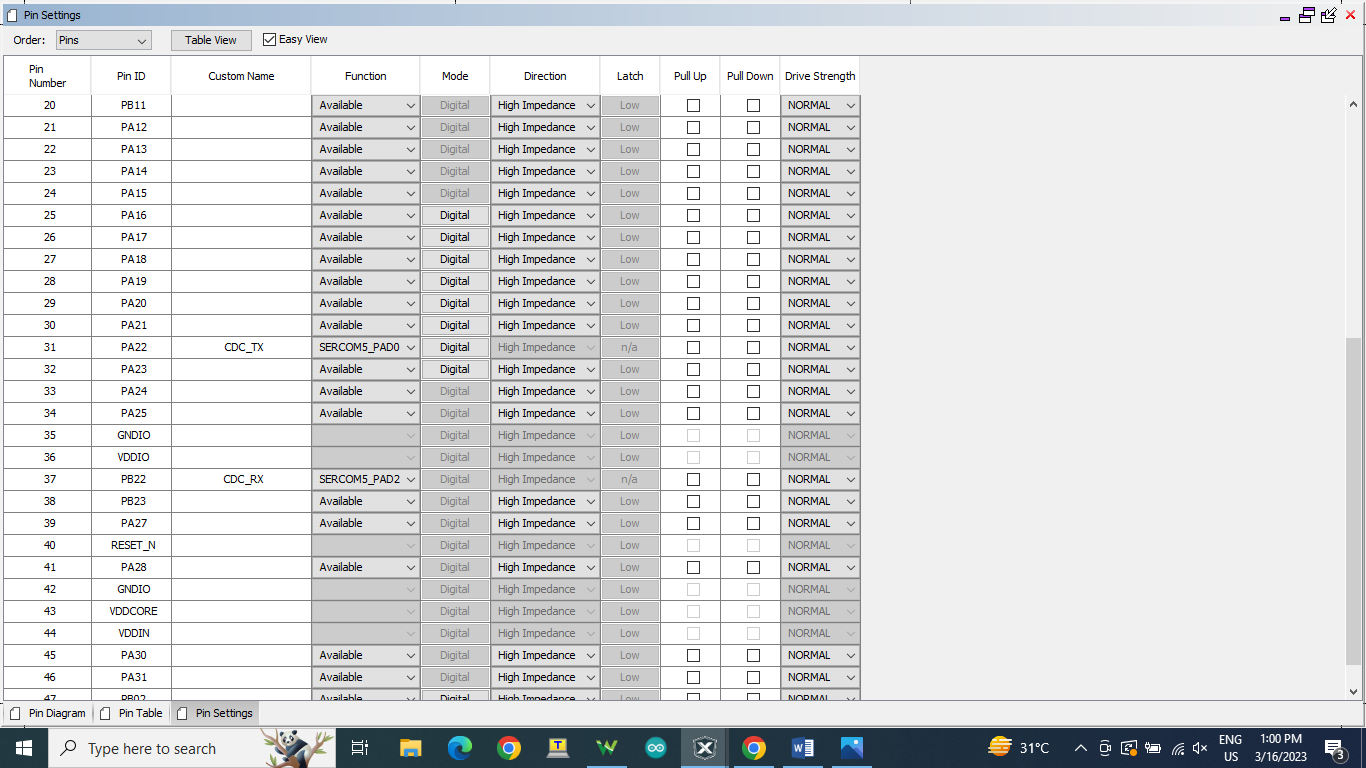
Step 1: Create a MPLAB Project and Open MHC (Refer to Creating First project).

Step 2: Clock Configuration:

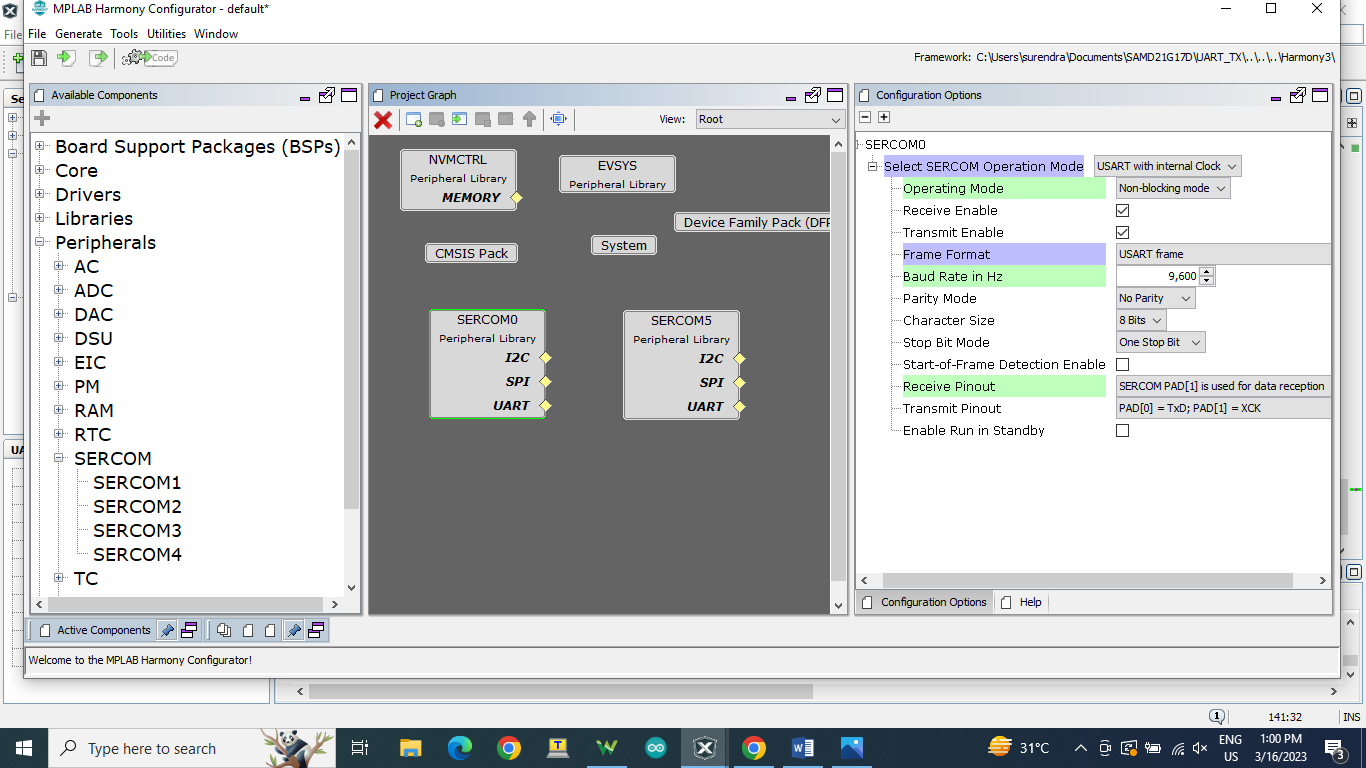


Step 3 : Set the Pin configuration as shown below.

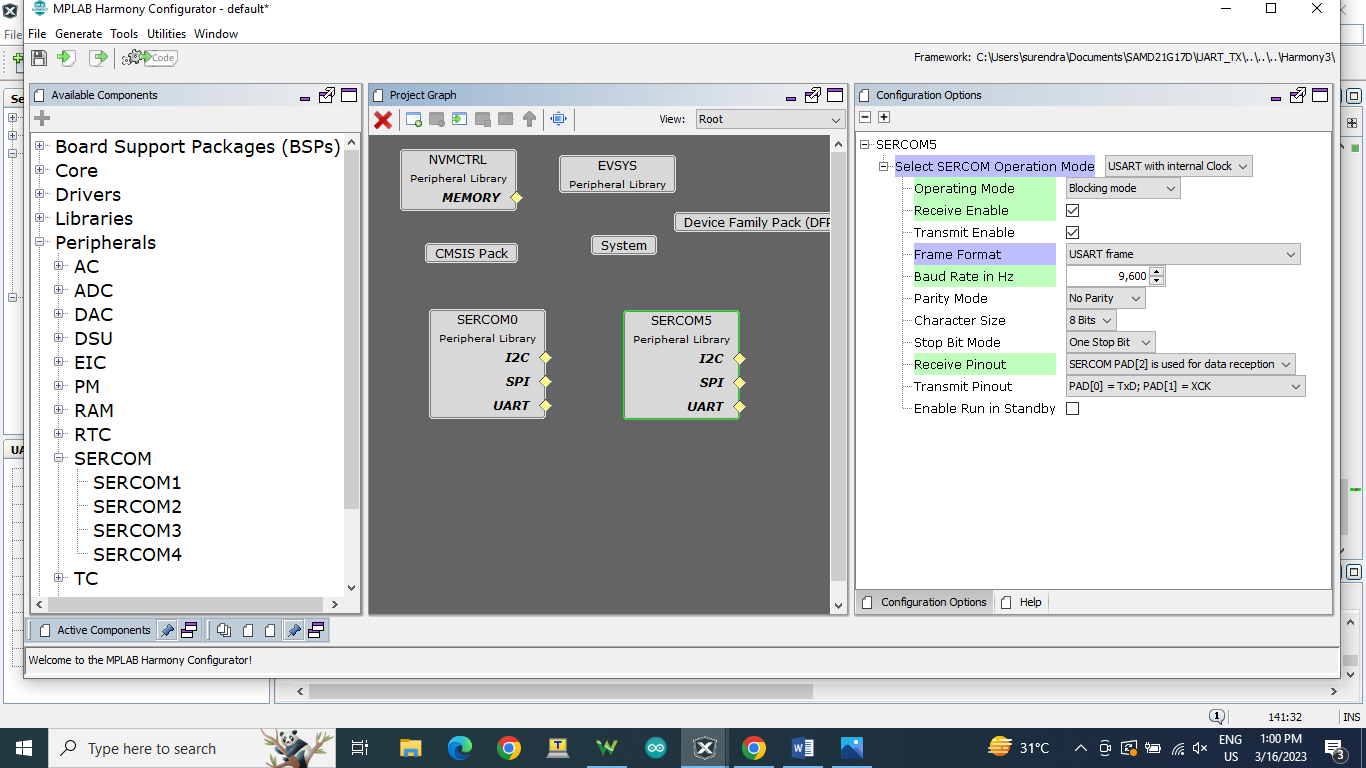




Step 4 : Set the configuration of Peripherals -> SERCOM->SERCOM0 as shown below.

****

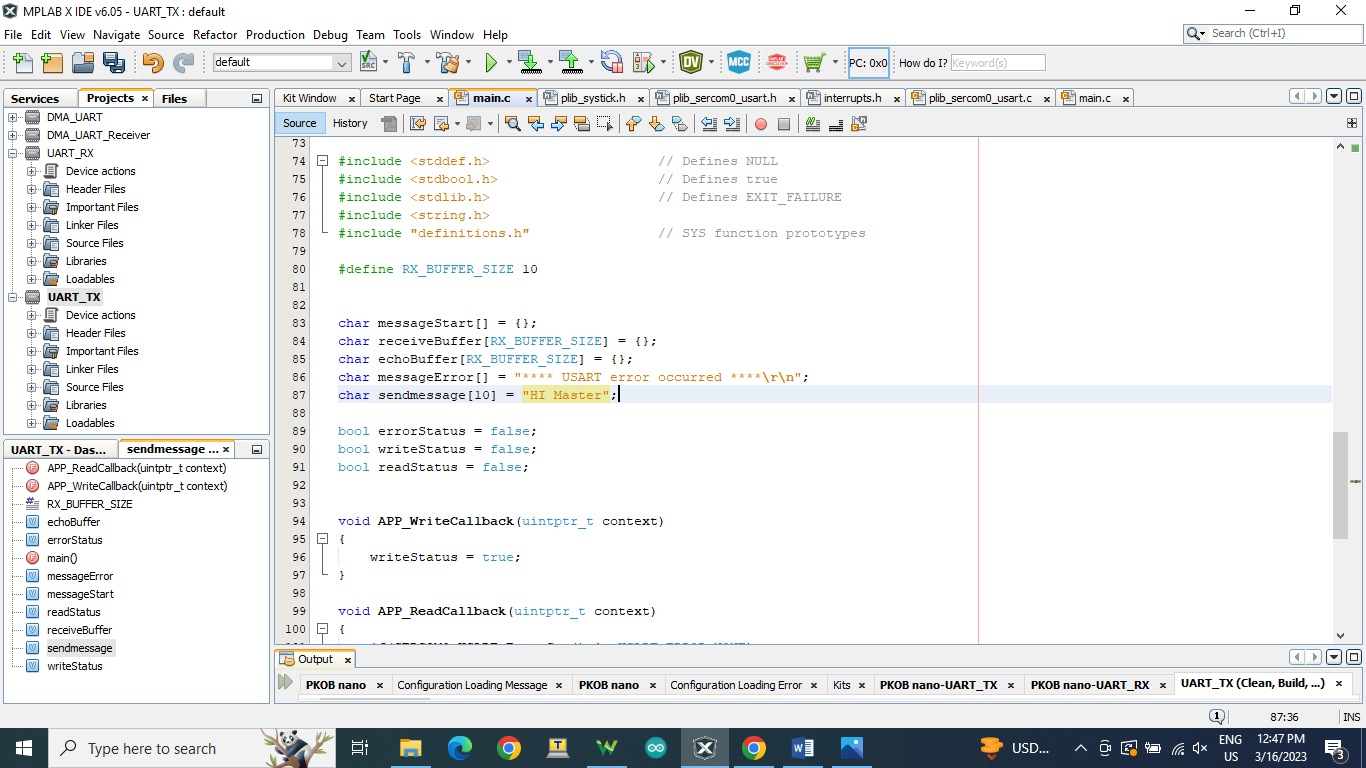
Step 5 : Set the configuration of Peripherals -> SERCOM-> SERCOM5 as shown below.

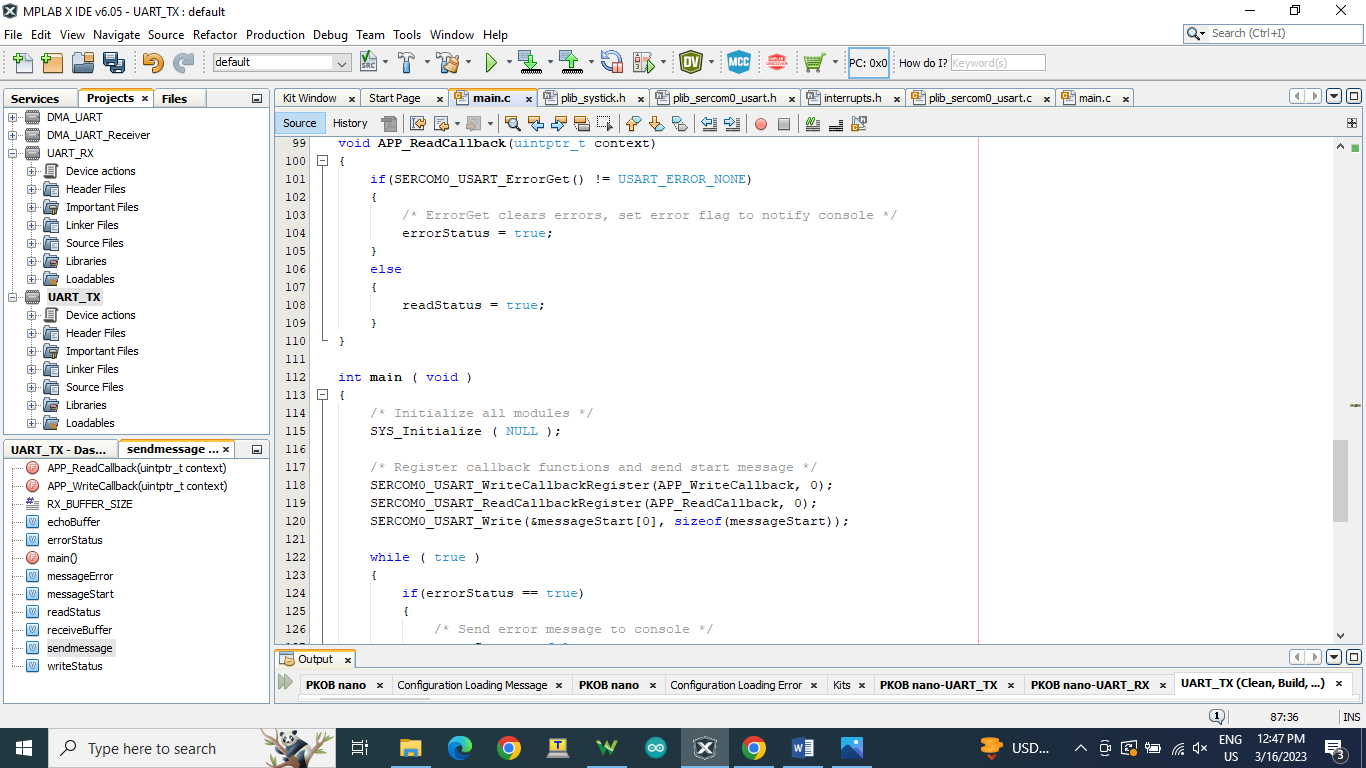
****

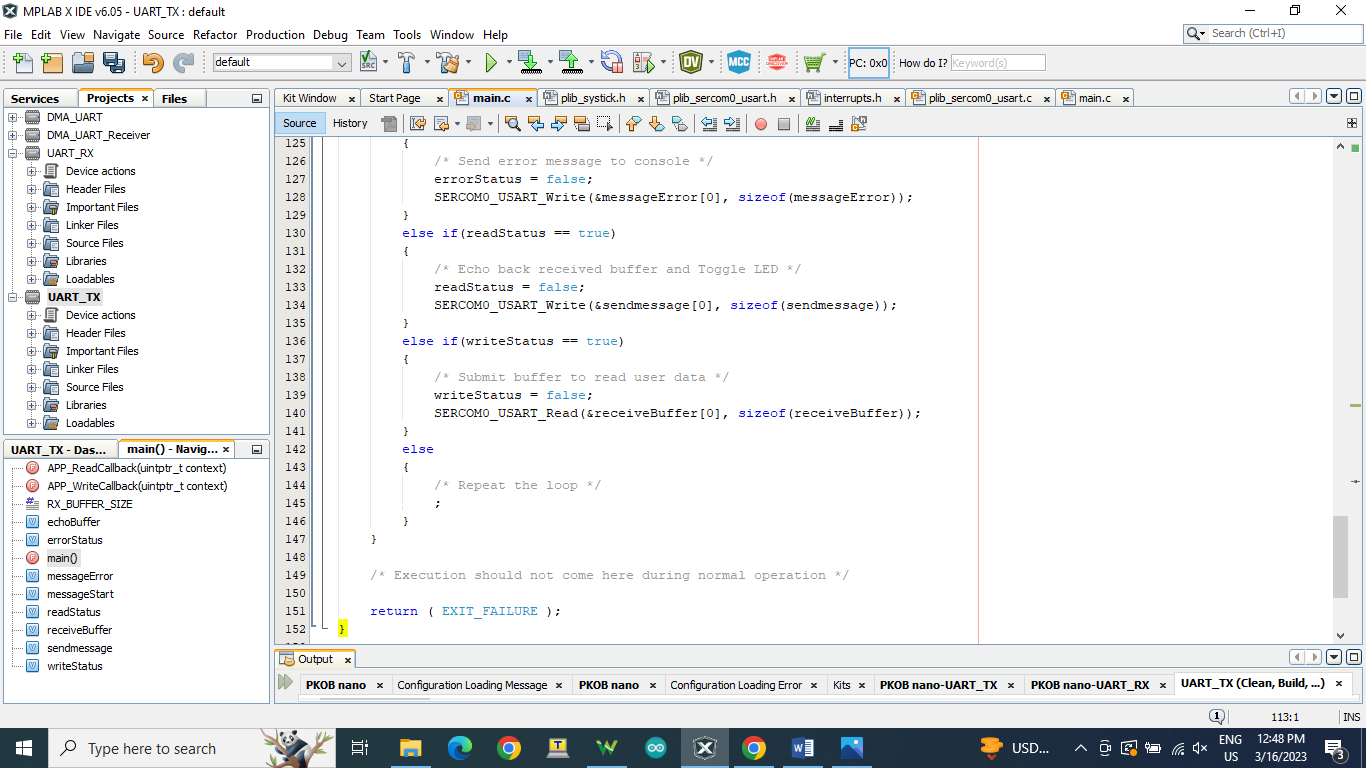
Step 6 : Click on Generate Code.

Step 7 : You should be able to see MHC generated files under Project->Source Files- >config->default->peripheral->usart->plib\_sercom0\_usart.c.

Also open main.c file for coding.







Step 8 : Sample code for UART .

#include <stddef.h> // Defines NULL

#include <stdbool.h> // Defines true

#include <stdlib.h> // Defines EXIT\_FAILURE

#include <string.h>

#include "definitions.h" // SYS function prototypes

#define RX\_BUFFER\_SIZE 10

char messageStart[] = {};

char receiveBuffer[RX\_BUFFER\_SIZE] = {};

char echoBuffer[RX\_BUFFER\_SIZE] = {};

char messageError[] = "\*\*\*\* USART error occurred \*\*\*\*\r\n";

char sendmessage[10] = "HI Master";

bool errorStatus = false;

bool writeStatus = false;

bool readStatus = false;

void APP\_WriteCallback(uintptr\_t context)

{

writeStatus = true;

}

void APP\_ReadCallback(uintptr\_t context)

{

if(SERCOM0\_USART\_ErrorGet() != USART\_ERROR\_NONE)

{

/\* ErrorGet clears errors, set error flag to notify console \*/

errorStatus = true;

}

else

{

readStatus = true;

}

}

int main ( void )

{

/\* Initialize all modules \*/

SYS\_Initialize ( NULL );

/\* Register callback functions and send start message \*/

SERCOM0\_USART\_WriteCallbackRegister(APP\_WriteCallback, 0);

SERCOM0\_USART\_ReadCallbackRegister(APP\_ReadCallback, 0);

SERCOM0\_USART\_Write(&messageStart[0], sizeof(messageStart));

while ( true )

{

if(errorStatus == true)

{

/\* Send error message to console \*/

errorStatus = false;

SERCOM0\_USART\_Write(&messageError[0], sizeof(messageError));

}

else if(readStatus == true)

{

/\* Echo back received buffer and Toggle LED \*/

readStatus = false;

SERCOM0\_USART\_Write(&sendmessage[0], sizeof(sendmessage));

}

else if(writeStatus == true)

{

/\* Submit buffer to read user data \*/

writeStatus = false;

SERCOM0\_USART\_Read(&receiveBuffer[0], sizeof(receiveBuffer));

SERCOM5\_USART\_Write(&receiveBuffer[0], sizeof(receiveBuffer));

}

else

{

/\* Repeat the loop \*/

;

}

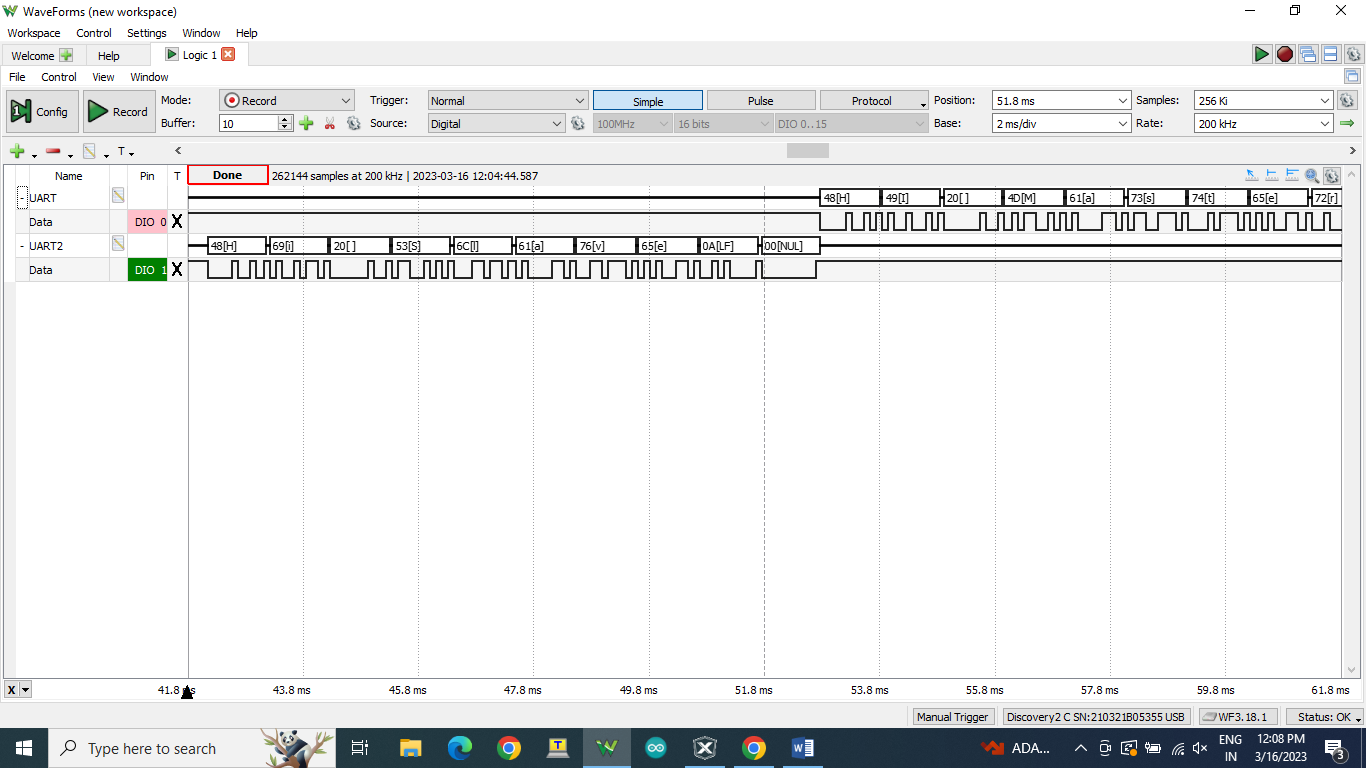
}

/\* Execution should not come here during normal operation \*/

return ( EXIT\_FAILURE );

}

Output : View output in logic analyzer or Virtual com port.

****