



**DesignNews**

# IoT Device Prototyping with Microchip Curiosity Development Boards

**Day 5:**

Coding the PIC32CM MC Curiosity Nano Evaluation Kit Using Harmony v3

Sponsored by



## Webinar Logistics

- Turn on your system sound to hear the streaming presentation.
- If you have technical problems, click “Help” or submit a question asking for assistance.
- Participate in ‘Attendee Chat’ by maximizing the chat widget in your dock.



## Fred Eady

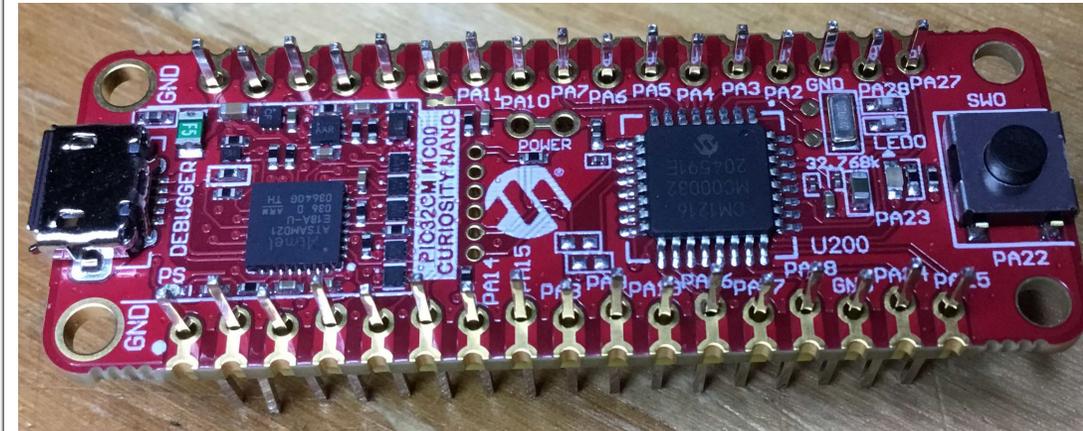
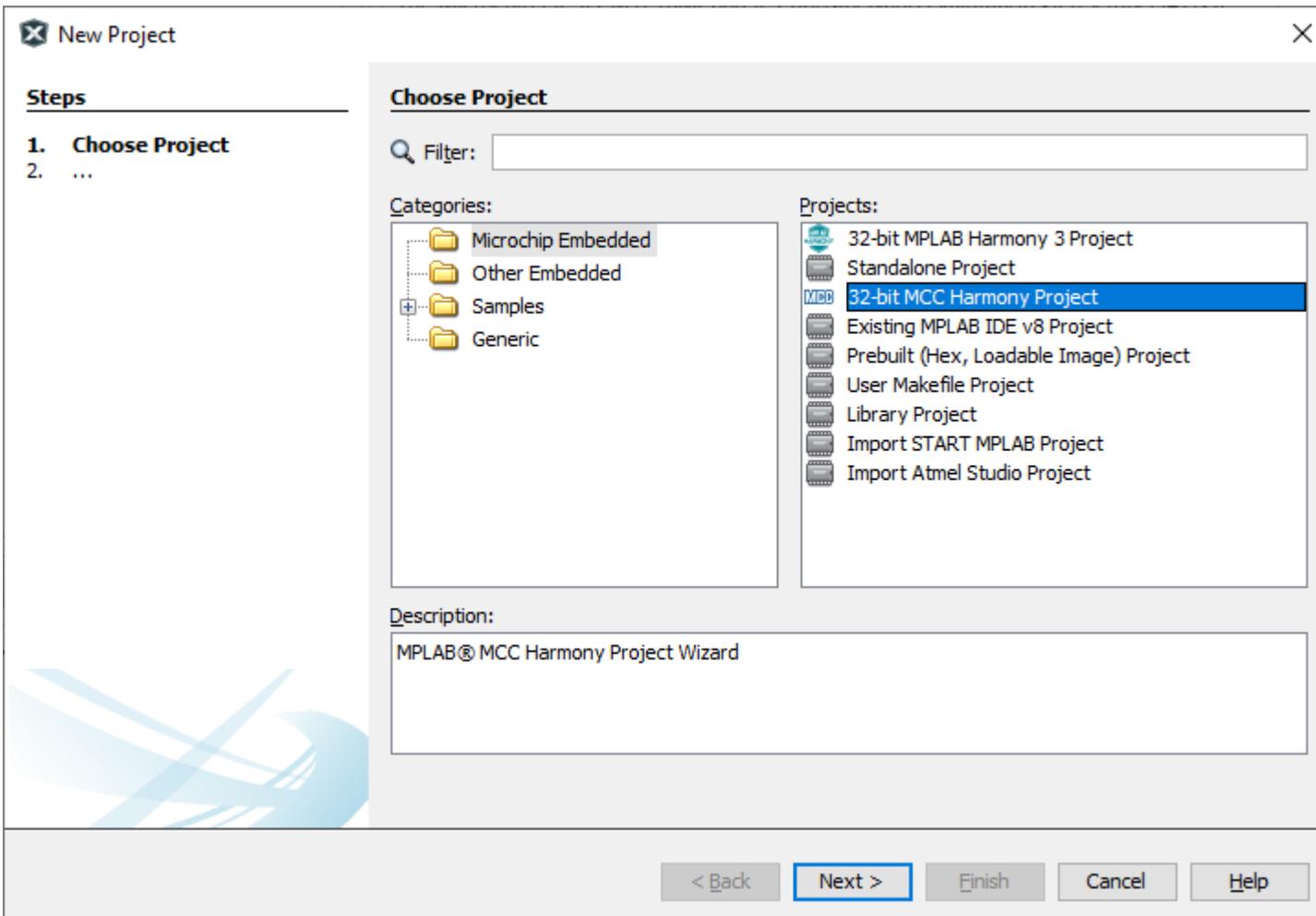
Visit 'Lecturer Profile' in your console for more details.

# AGENDA

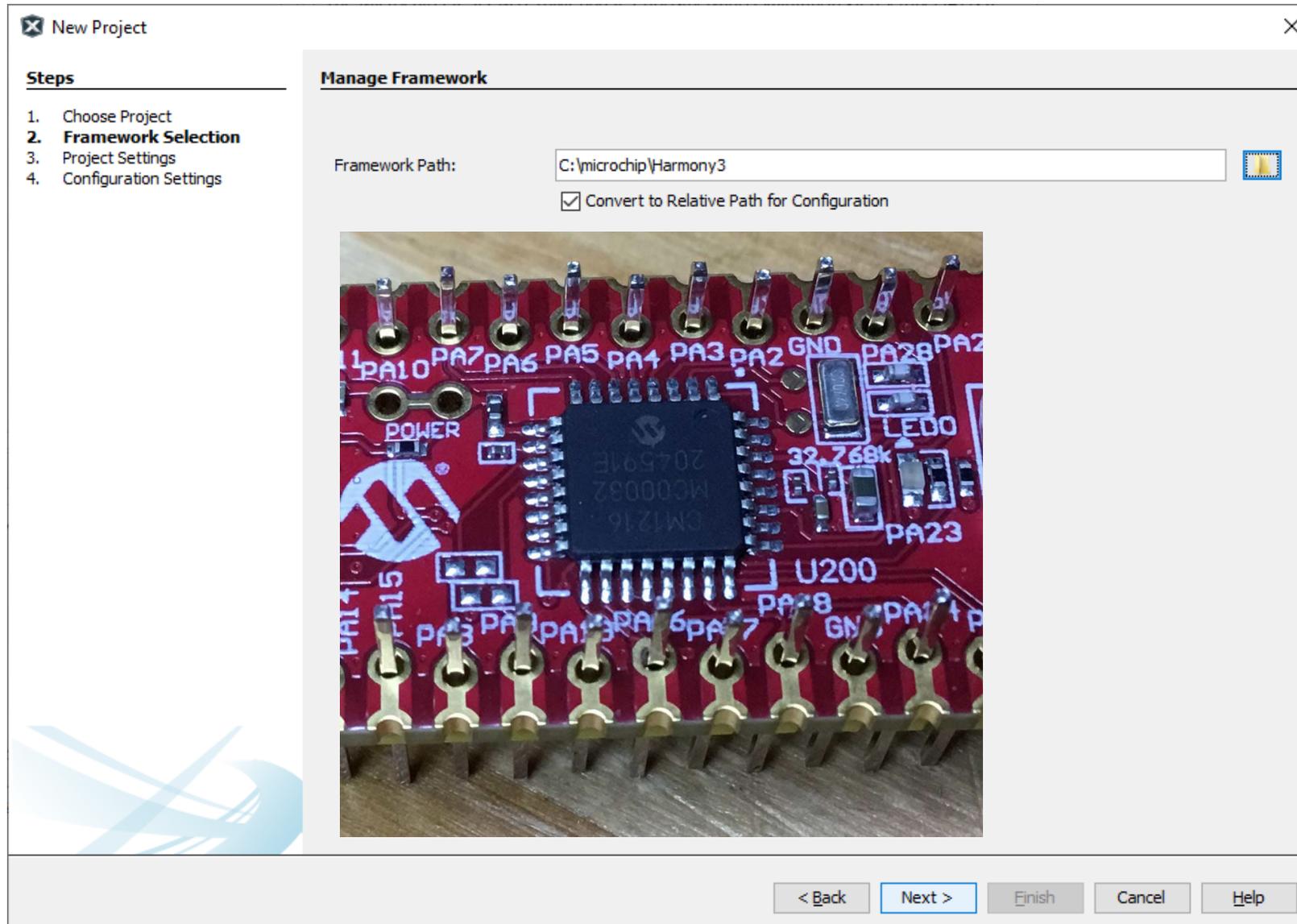
- **Blink an LED the Harmony v3 Way**
- **I SPI**
- **Coding a Harmony v3-Generated USB CDC Port**



## Create a 32-bit MCC Harmony Project



## Define the Framework Path



# Specify the PIC32CM1216MC00032

New Project

**Steps**

1. Choose Project
2. Framework Selection
3. Project Settings
- 4. Configuration Settings**

**Configuration Settings**

Name:

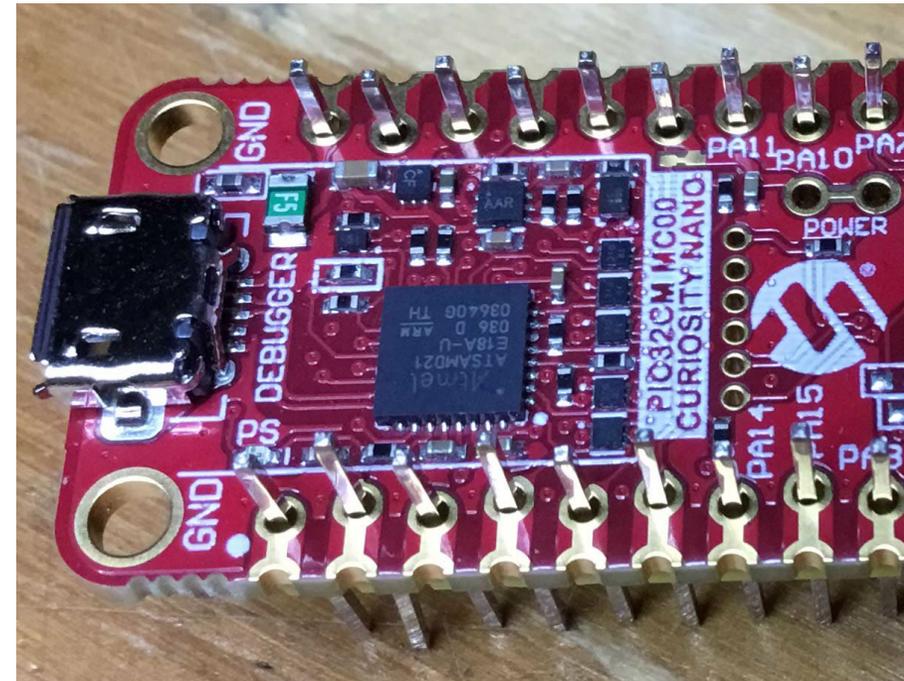
Device Family:  Target Device:

Device Filter:

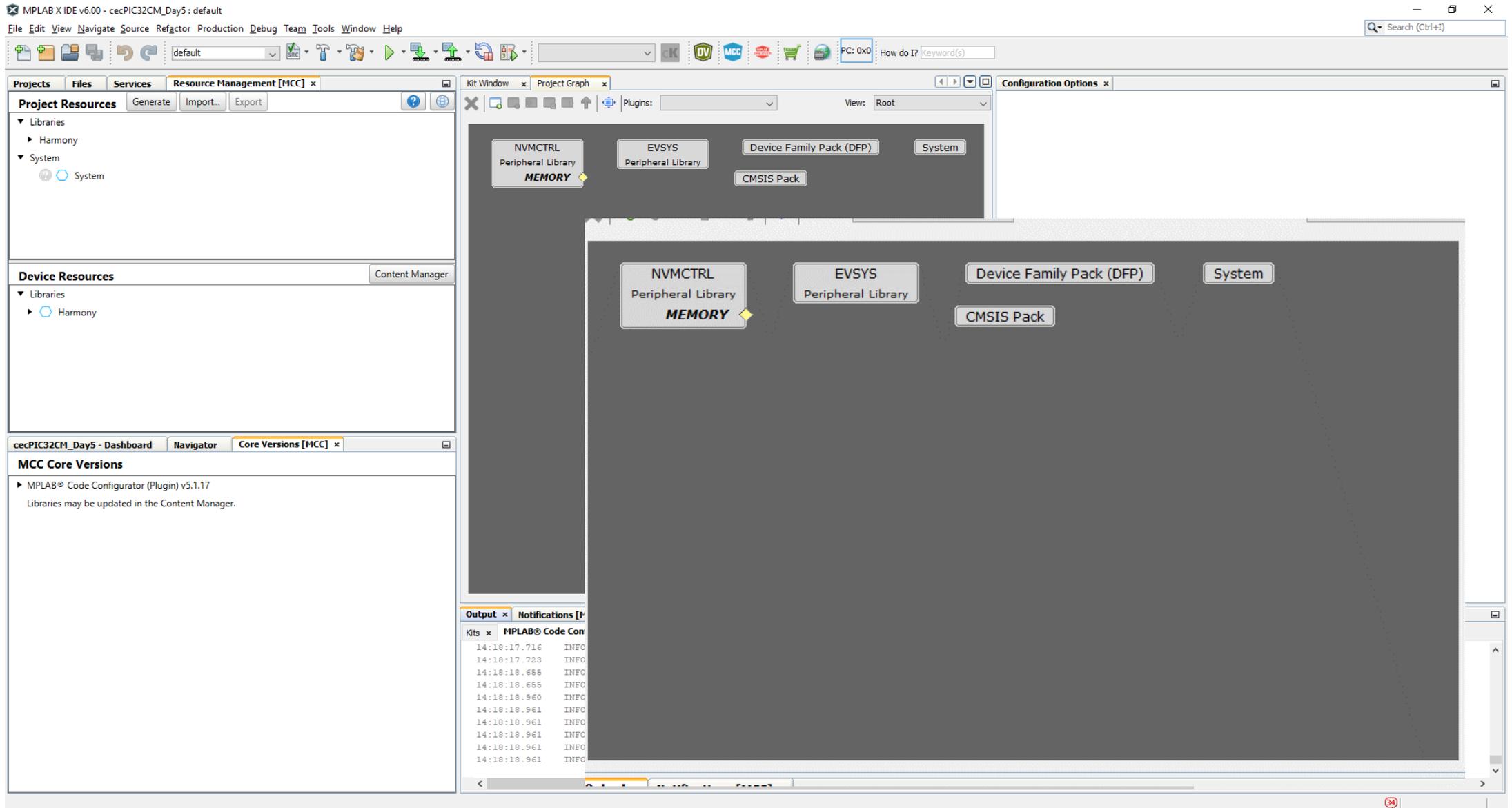
Show Visual Help

Install device pack

< Back   Next >   **Finish**   Cancel   Help



# Project Graph



The screenshot shows the MPLAB X IDE interface with the Project Graph window open. The graph displays a hierarchical structure of project components:

- Kit Window:** Shows the root of the project graph with components: NVMCTRL Peripheral Library (MEMORY), EVSYS Peripheral Library, Device Family Pack (DFP), CMSIS Pack, and System.
- Project Graph:** Shows a similar structure, highlighting the NVMCTRL Peripheral Library (MEMORY) component.

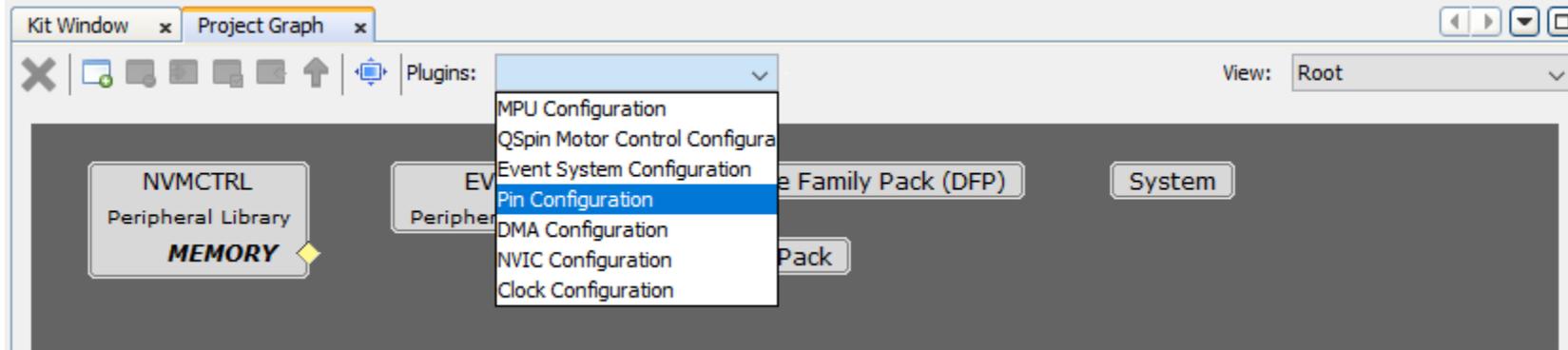
On the left side of the IDE, the Resource Management [MCC] window is visible, showing Project Resources (Libraries, System) and Device Resources (Harmony). Below it, the MCC Core Versions window shows the MPLAB® Code Configurator (Plugin) v5.1.17.

At the bottom, the Output window displays a log of information messages:

```

Output x Notifications [P
Kits x MPLAB® Code Con
14:18:17.716 INFO
14:18:17.723 INFO
14:18:18.655 INFO
14:18:18.655 INFO
14:18:18.960 INFO
14:18:18.961 INFO
  
```

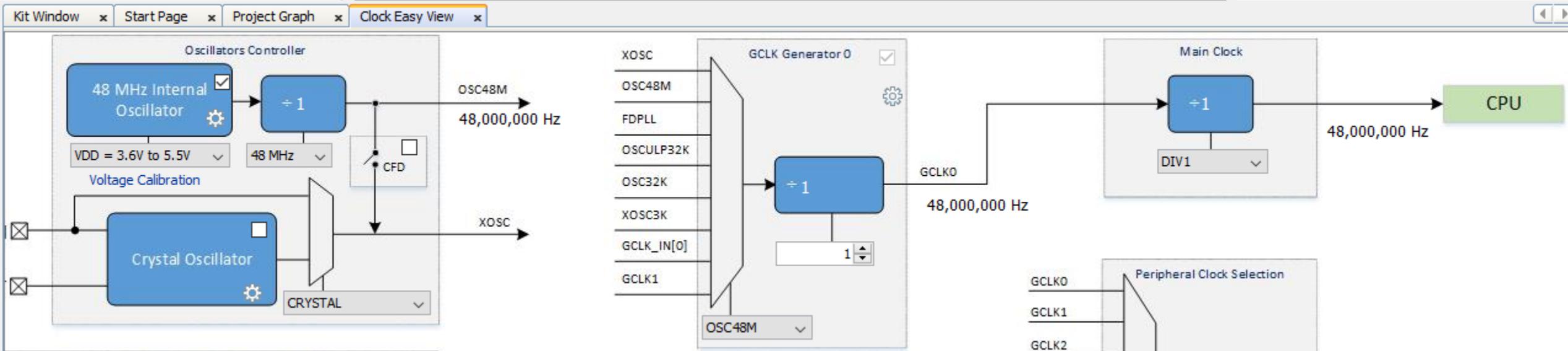
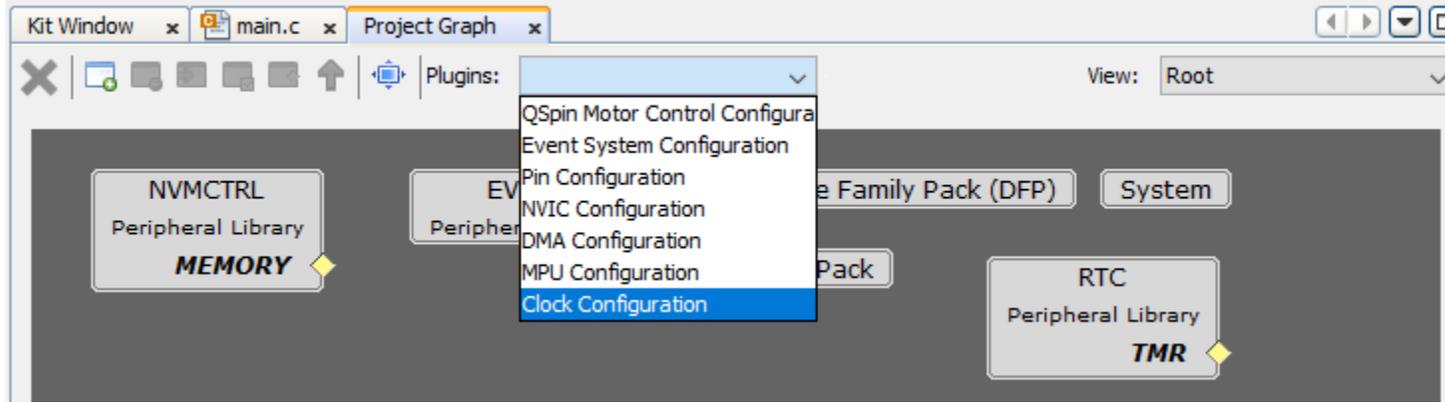
## Add the LED to the Project



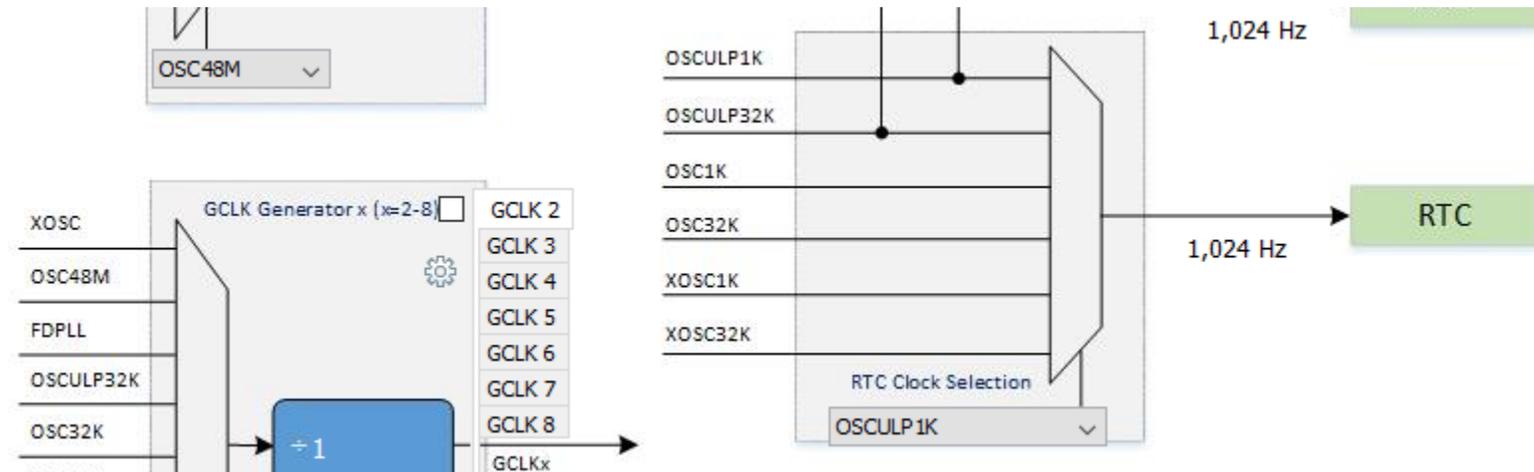
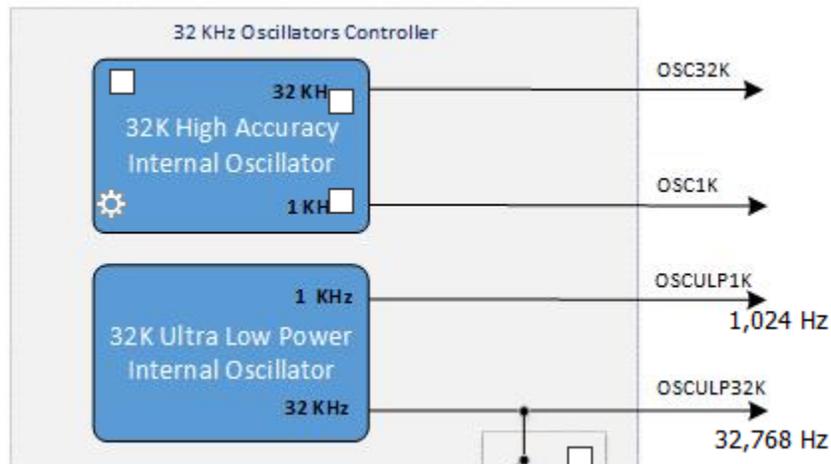
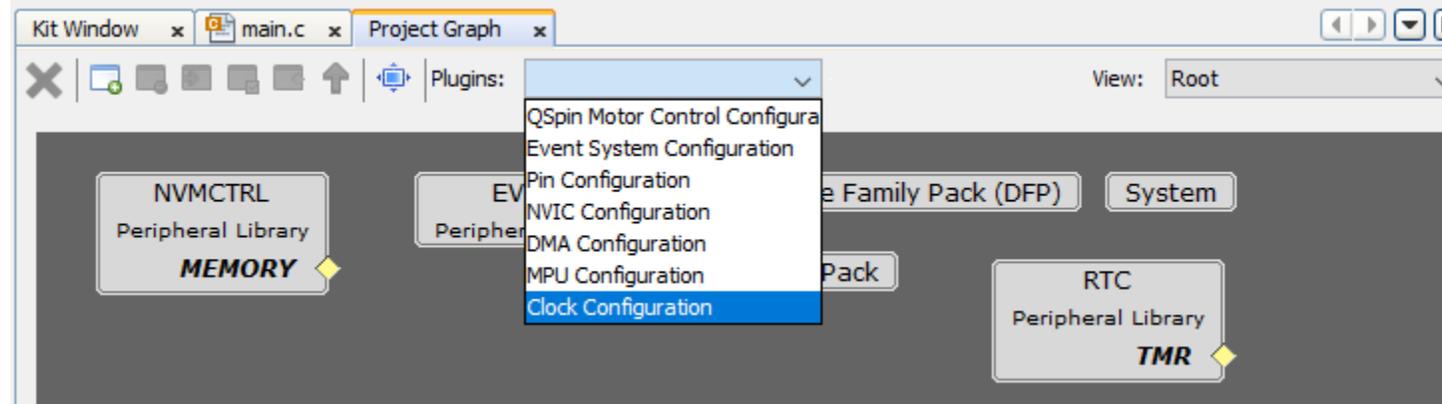
Order	Pin ID	Custom Name	Function	Mode	Direction	Latch	Pull Up	Pull Down	Drive Strength
13	PA10		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
14	PA11		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
15	PA14		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
16	PA15		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
17	PA16		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
18	PA17		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
19	PA18		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
20	PA19		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
21	PA22		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
22	PA23		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
23	PA24		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
24	PA25		CCL_IN7	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
25	PA27		EIC_EXTINT7	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
26	RESET_N		GCLK_IO7	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
27	PA28		GPIO	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
28	GNDIO		SERCOM3_PAD1	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
29	VDDCORE		TC0_WO1	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
30	VDDIN		TCC0_WO5	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
31	PA30		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
32	PA31		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL

Order	Pin ID	Custom Name	Function	Mode	Direction	Latch	Pull Up	Pull Down	Drive Strength
13	PA10		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
14	PA11		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
15	PA14		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
16	PA15		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
17	PA16		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
18	PA17		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
19	PA18		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
20	PA19		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
21	PA22		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
22	PA23	LED_PA23	GPIO	Digital	In/Out	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
23	PA24		Available	Digital	In	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
24	PA25		Available	Digital	Out	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
25	PA27		Available	Digital	In/Out	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
26	RESET_N		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
27	PA28		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
28	GNDIO		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
29	VDDCORE		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
30	VDDIN		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
31	PA30		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
32	PA31		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL

## Check the Main Clock Configuration and Add the RTC



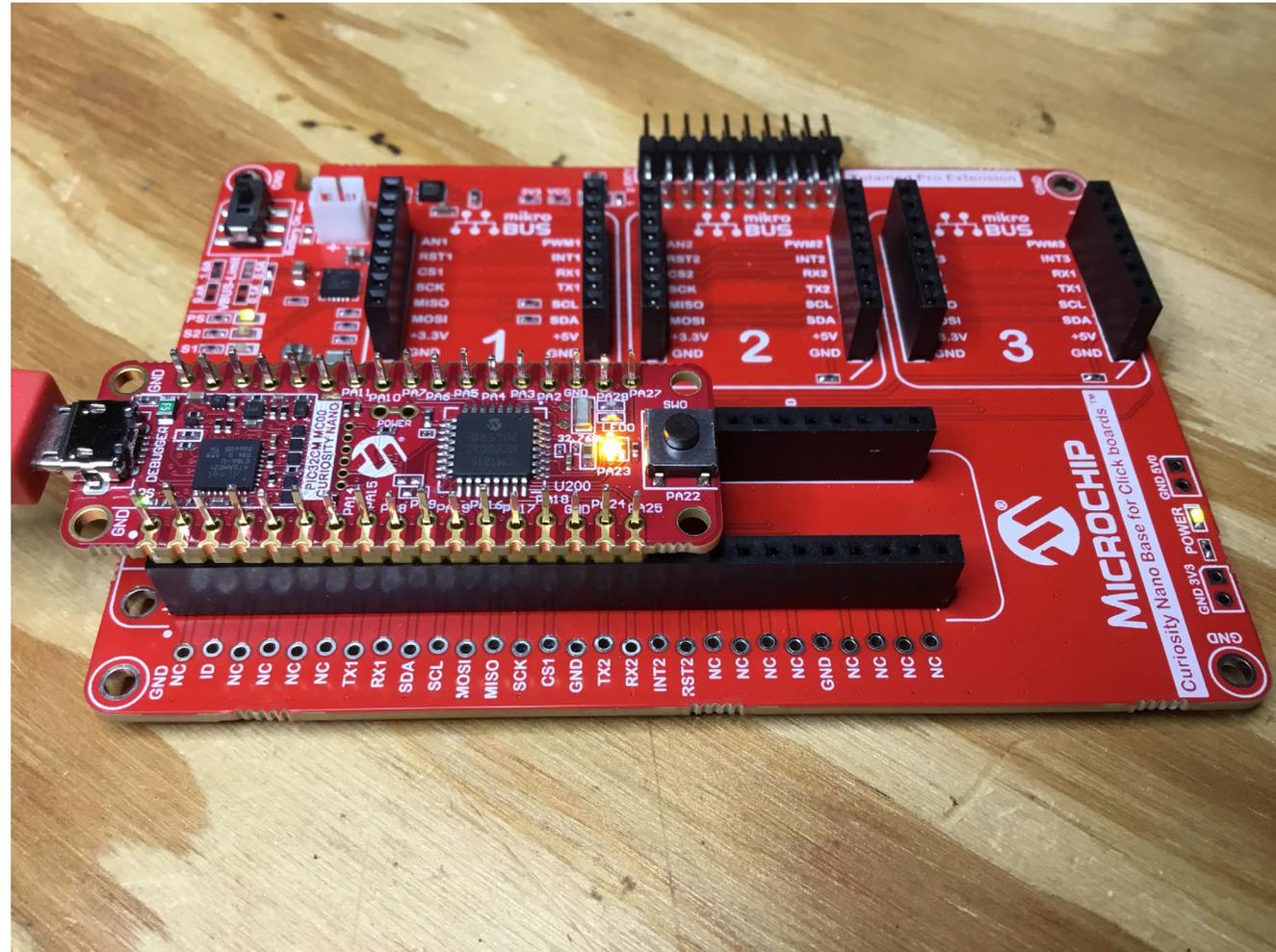
## Check the RTC Clock Configuration



## Configure the RTC Interrupt

**Configuration Options** x

- RTC
  - Hardware Settings
    - Generate Frequency Correction API
    - RTC Count Sync Enable
  - RTC Operation Mode **32-bit Counter with Single 32-bit Compare**
  - RTC MODE 0 Configuration
    - Enable Interrupts ? 
      - Periodic Interval 0 Interrupt Enable
      - Periodic Interval 1 Interrupt Enable
      - Periodic Interval 2 Interrupt Enable
      - Periodic Interval 3 Interrupt Enable
      - Periodic Interval 4 Interrupt Enable
      - Periodic Interval 5 Interrupt Enable
      - Periodic Interval 6 Interrupt Enable
      - Periodic Interval 7 Interrupt Enable
      - Compare 0 Interrupt Enable**
      - Overflow Interrupt Enable
    - RTC Prescaler **DIV1**
    - Compare Value** 0x 200
    - Clear on compare Match**
  - RTC EVENTS configuration



## Generate and Run the Code

```

#include <stddef.h>           // Defines NULL
#include <stdbool.h>         // Defines true
#include <stdlib.h>          // Defines EXIT_FAILURE
#include "definitions.h"     // SYS function prototypes

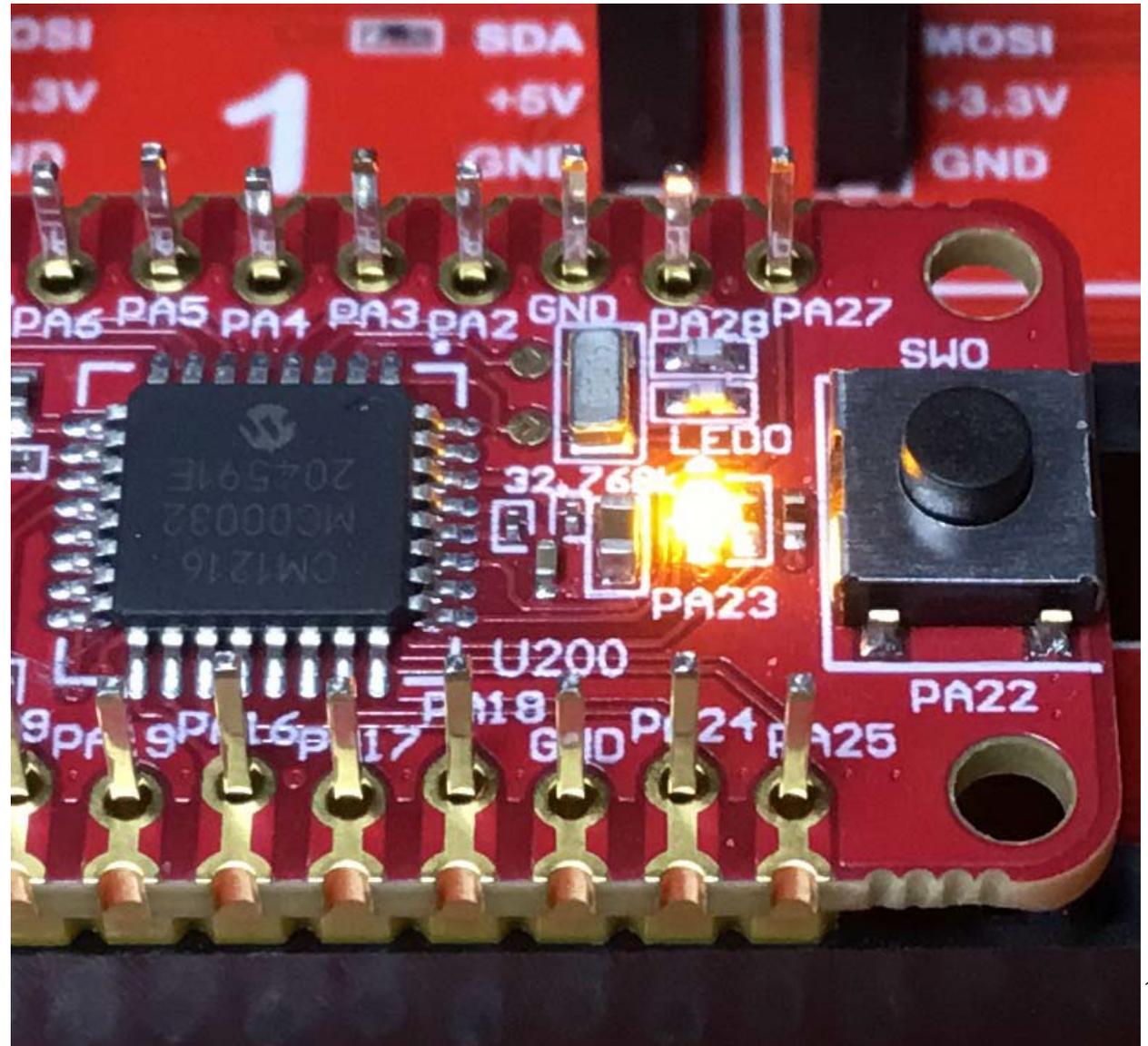
static volatile bool rtcTick= false;
static void rtcEventHandler (RTC_TIMER32_INT_MASK intCause, uintptr_t context)
{
    if (intCause & RTC_MODE0_INTENSET_CMP0_Msk)
    {
        rtcTick  = true;
    }
}

int main ( void )
{
    /* Initialize all modules */
    SYS_Initialize ( NULL );
    RTC_Timer32CallbackRegister(rtcEventHandler, 0);
    RTC_Timer32Start();
    while ( true )
    {
        if (rtcTick == true)
        {
            rtcTick= false;
            LED_PA23_Toggle();
        }
    }

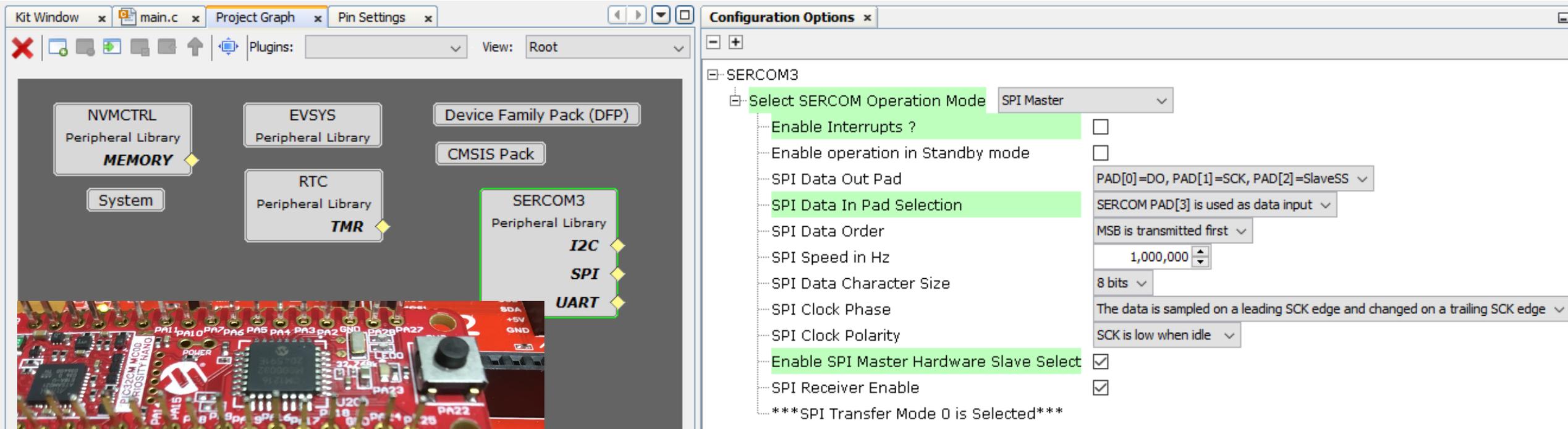
    /* Execution should not come here during normal operation */

    return ( EXIT_FAILURE );
}

```



## Add the SPI Peripheral



Kit Window x main.c x Project Graph x Pin Settings x

View: Root

Configuration Options x

SERC0M3

- Select SERCOM Operation Mode: SPI Master
- Enable Interrupts ?
- Enable operation in Standby mode
- SPI Data Out Pad: PAD[0]=DO, PAD[1]=SCK, PAD[2]=SlaveSS
- SPI Data In Pad Selection: SERCOM PAD[3] is used as data input
- SPI Data Order: MSB is transmitted first
- SPI Speed in Hz: 1,000,000
- SPI Data Character Size: 8 bits
- SPI Clock Phase: The data is sampled on a leading SCK edge and changed on a trailing SCK edge
- SPI Clock Polarity: SCK is low when idle
- Enable SPI Master Hardware Slave Select
- SPI Receiver Enable
- \*\*\*SPI Transfer Mode 0 is Selected\*\*\*

Project Graph:

- NVMCTRL Peripheral Library
- EVSYST Peripheral Library
- Device Family Pack (DFP)
- CMSIS Pack
- System
- RTC Peripheral Library
- SERC0M3 Peripheral Library
- I2C
- SPI
- UART
- TMR

Hardware Board:

PIC32CM MC001 CURIOUSITY NANO

PA1 PA10 PA7 PA6 PA5 PA4 PA3 PA2 GND PA28 PA27

POWER

SDA +5V GND

PA1 PA15

U2C

PA23

PA22

NC ID NC NC NC NC TX1 RX1 SDA SCL MOSI MISO SCK CS1 GND TX2 RX2 INT2 RST2 NC NC NC NC

## Add the SPI Peripheral and SPI Application Code

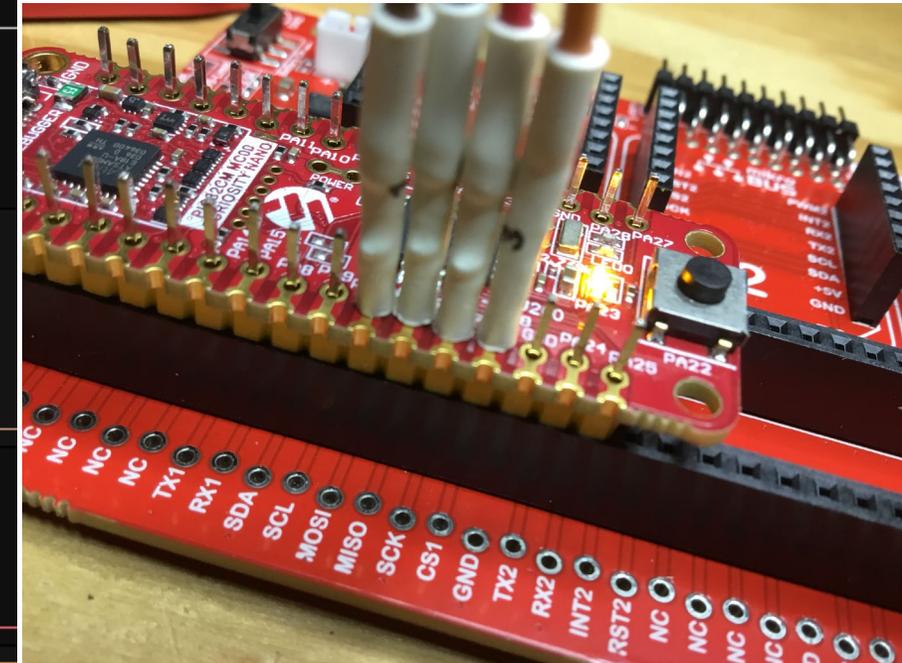
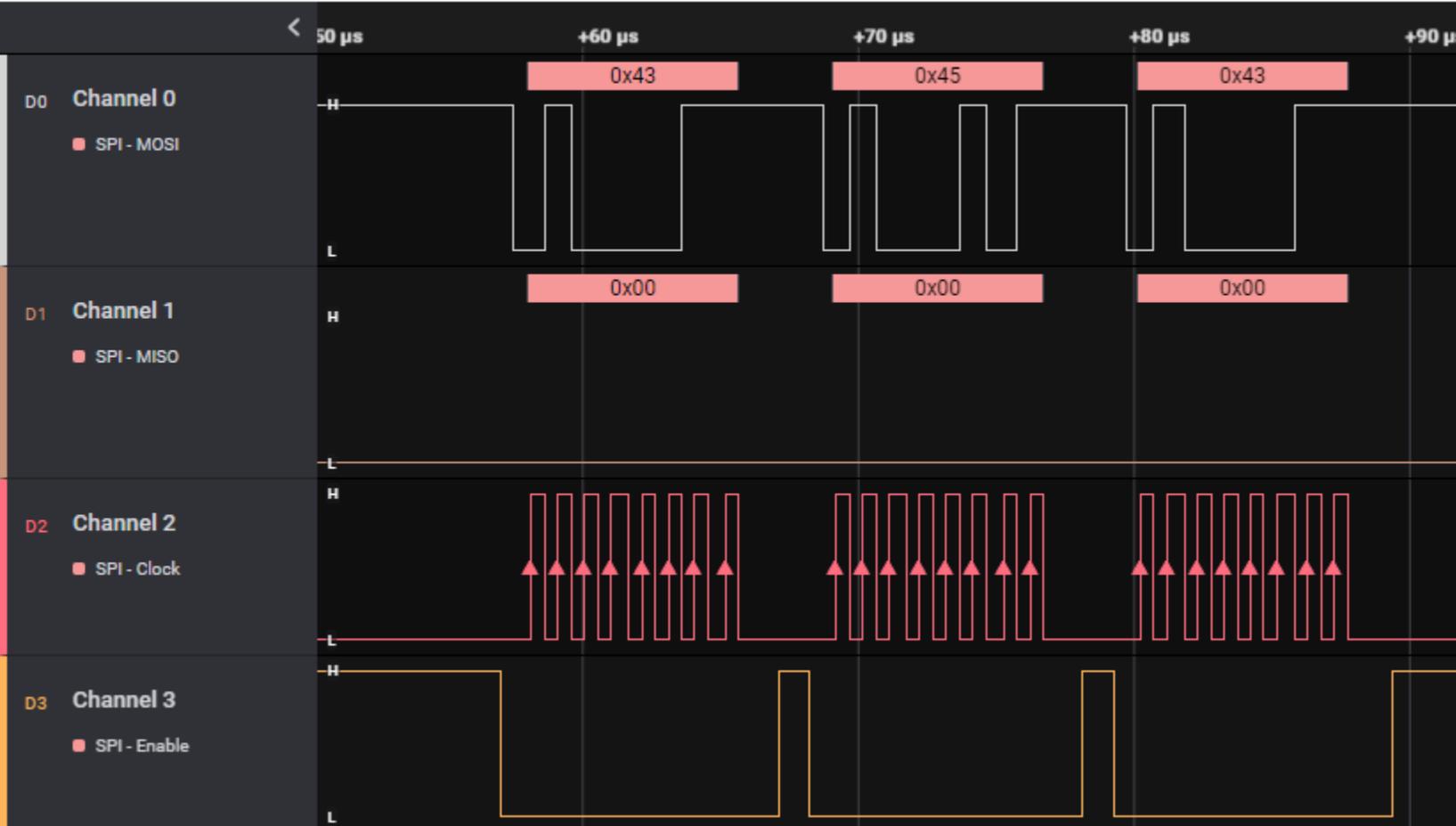
Pin Number	Pin ID	Custom Name	Function	Mode	Direction	Latch	Pull Up	Pull Down	Drive Strength
4	PA03		Available	Analog	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
5	PA04		Available	Analog	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
6	PA05		Available	Analog	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
7	PA06		Available	Analog	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
8	PA07		Available	Analog	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
9	VDDANA			Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
10	GNDIO			Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
11	PA08		Available	Analog	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
12	PA09		Available	Analog	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
13	PA10		Available	Analog	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
14	PA11		Available	Analog	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
15	PA14		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
16	PA15		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
17	PA16		SERCOM3_PAD0	Digital	High Impedance	n/a	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
18	PA17		SERCOM3_PAD1	Digital	High Impedance	n/a	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
19	PA18		SERCOM3_PAD2	Digital	High Impedance	n/a	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
20	PA19		SERCOM3_PAD3	Digital	High Impedance	n/a	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
21	PA22		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
22	PA23	LED_PA23	GPIO	Digital	Out	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
23	PA24		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
24	PA25		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL

```
uint8_t spiTxBuf[8];
int main ( void )
{
    /* Initialize all modules */
    SYS_Initialize ( NULL );
    spiTxBuf[0] = 0x43;
    spiTxBuf[1] = 0x45;
    spiTxBuf[2] = 0x43;
    RTC_Timer32CallbackRegister(rtcEventHandler, 0);
    RTC_Timer32Start();
    while ( true )
    {
        if (rtcTick == true)
        {
            rtcTick= false;
            SERCOM3_SPI_Write(spiTxBuf, 0x03 );
            LED_PA23_Toggle();
        }
    }
}
```

# Hook Up and Execute the SPI Peripheral Code

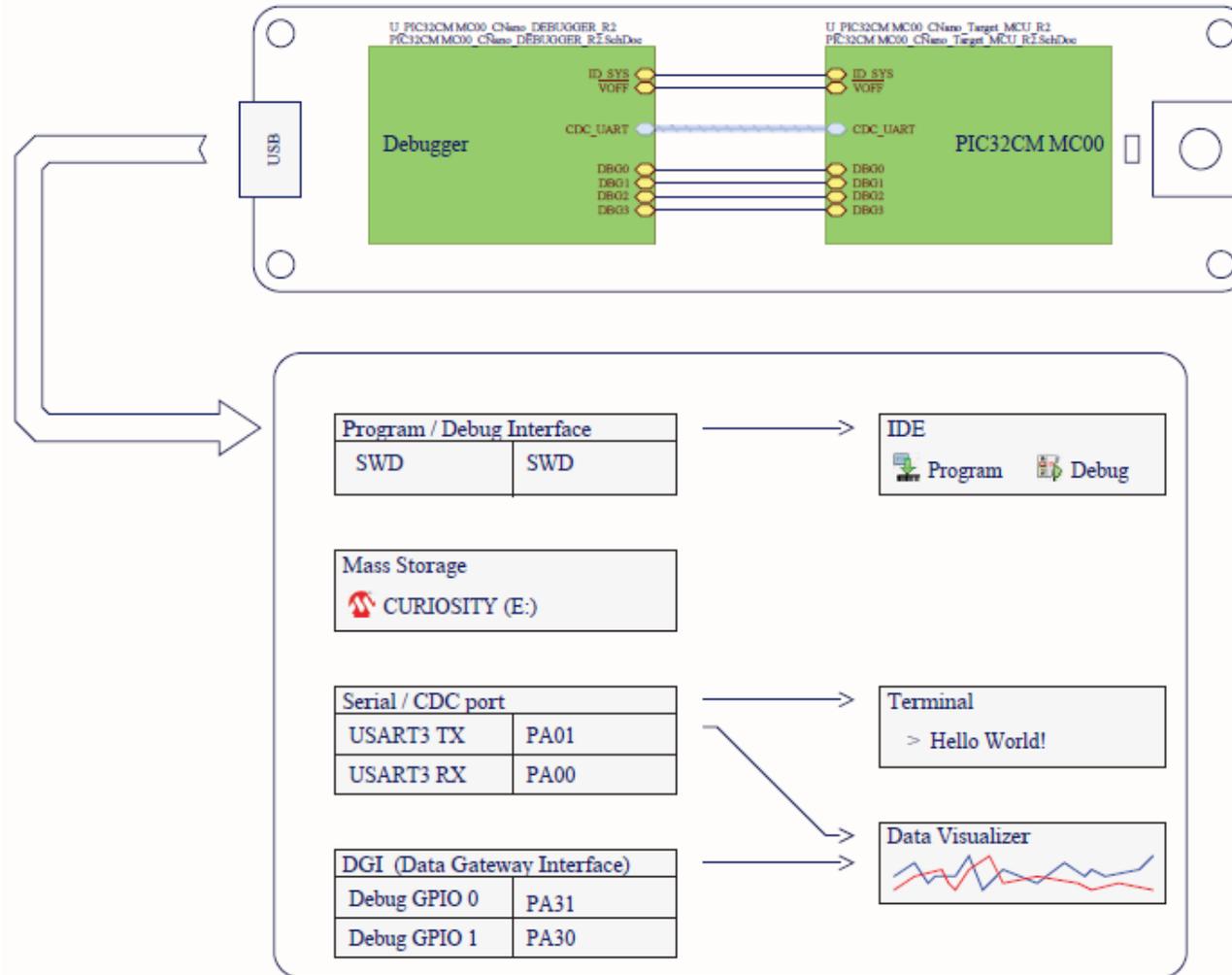
Logic 2 [Logic Pro 16 - Connected] [Session 0]

File Edit Capture Measure View Help



## Hook Up and Execute the SPI Peripheral Code

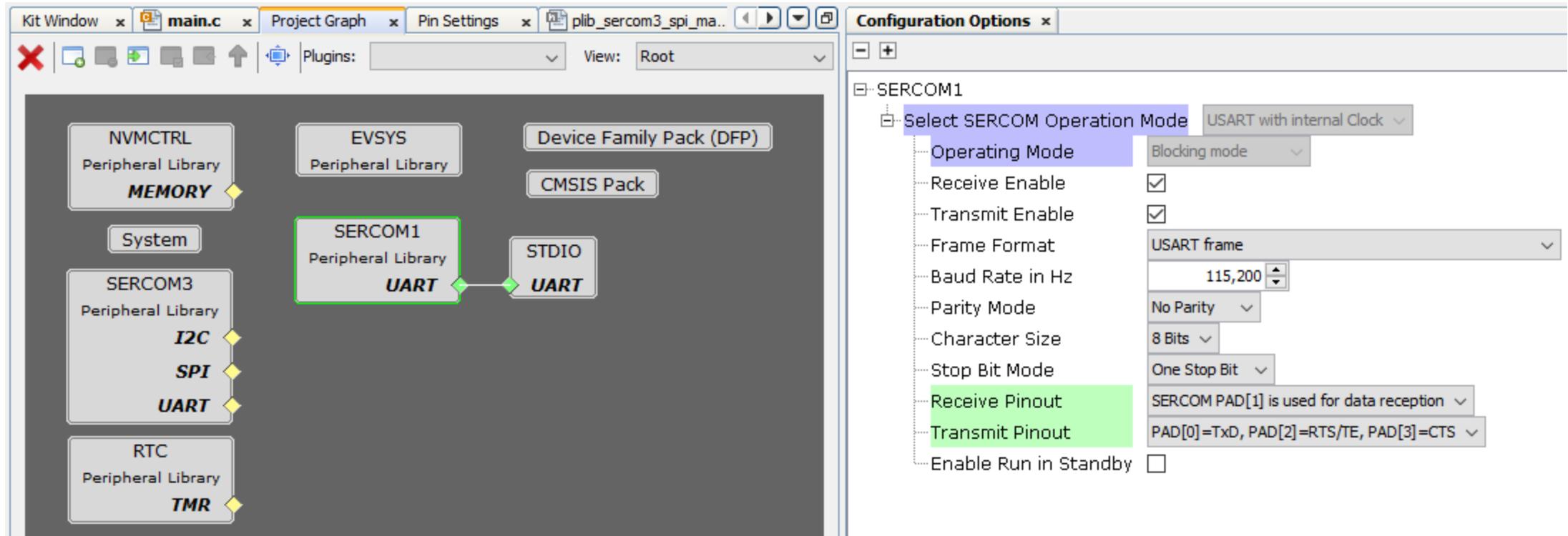
# PIC32CM MC00 Curiosity Nano



## Add UART Peripheral

The screenshot shows the Microchip Harmony IDE interface. On the left, the 'Peripheral Library' pane displays various components. The 'SERCOM1' peripheral library is highlighted with a green border, and its 'UART' sub-component is also highlighted. A context menu is open over the 'UART' component, showing options: 'Consumers', 'STDIO (stdio)', 'X2C Model (X2C Model)', and 'X2CScope (X2CScope)'. The 'STDIO (stdio)' option is selected. On the right, the 'Configuration Options' pane shows the configuration for 'SERCOM1'. The 'Select SERCOM Operation Mode' is set to 'USART with internal Clock'. Other visible options include 'Operating Mode' (Blocking mode), 'Receive Enable' (checked), 'Transmit Enable' (checked), 'Frame Format' (USART frame), 'Baud Rate in Hz' (115,200), 'Parity Mode' (No Parity), 'Character Size' (8 Bits), and 'Stop Bit Mode' (One Stop Bit). The 'Receive Pinout' and 'Transmit Pinout' options are also visible and highlighted in green.

## Attach UART Peripheral to STUDIO



The screenshot displays the Microchip Studio IDE interface. The top window shows the Project Graph with the following components:

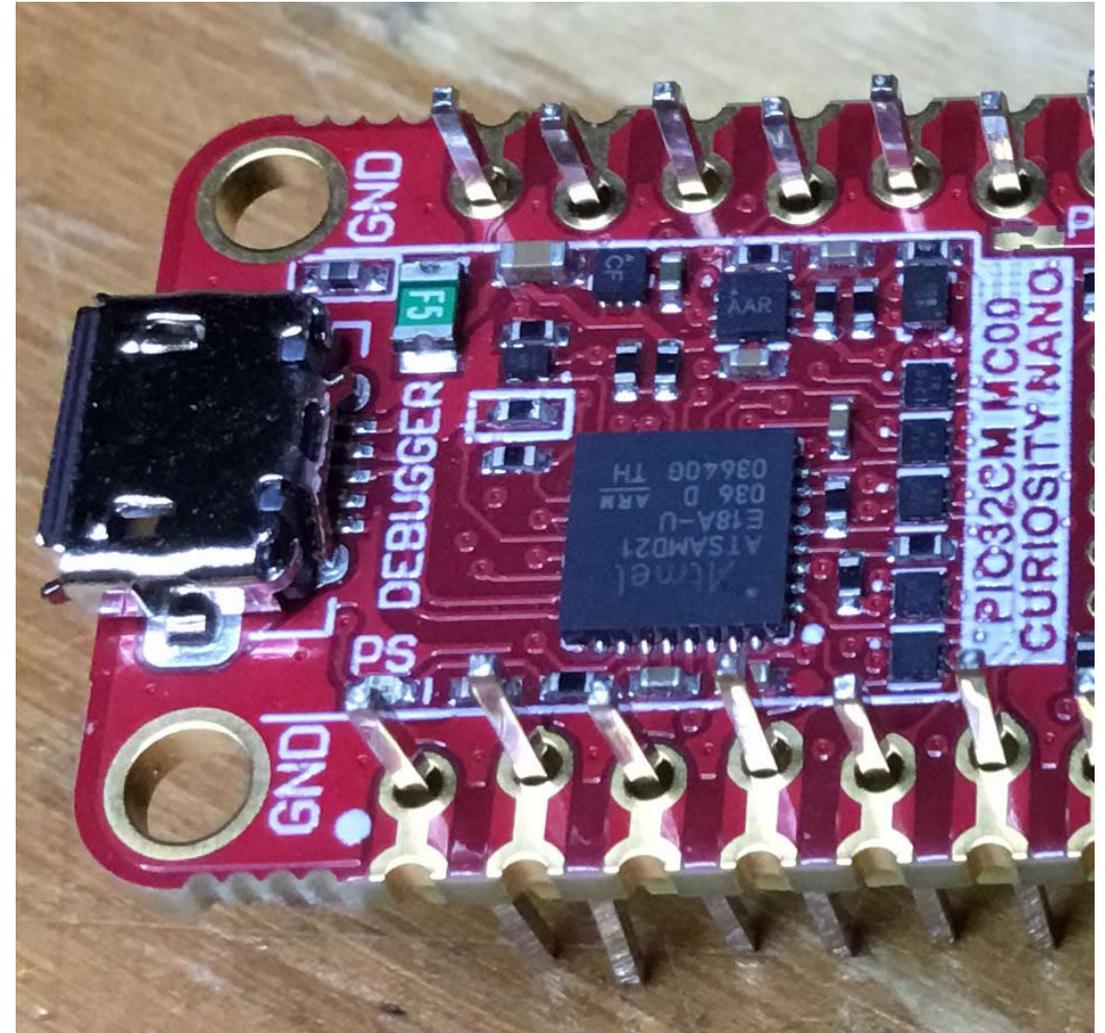
- NVMCTRL Peripheral Library (MEMORY)
- EVSYS Peripheral Library
- Device Family Pack (DFP)
- CMSIS Pack
- System
- SERCOM3 Peripheral Library (I2C, SPI, UART)
- SERCOM1 Peripheral Library (UART)
- STDIO (UART)
- RTC Peripheral Library (TMR)

The SERCOM1 and STDIO components are connected by a bidirectional arrow, indicating their association. The right-hand pane shows the Configuration Options for SERCOM1:

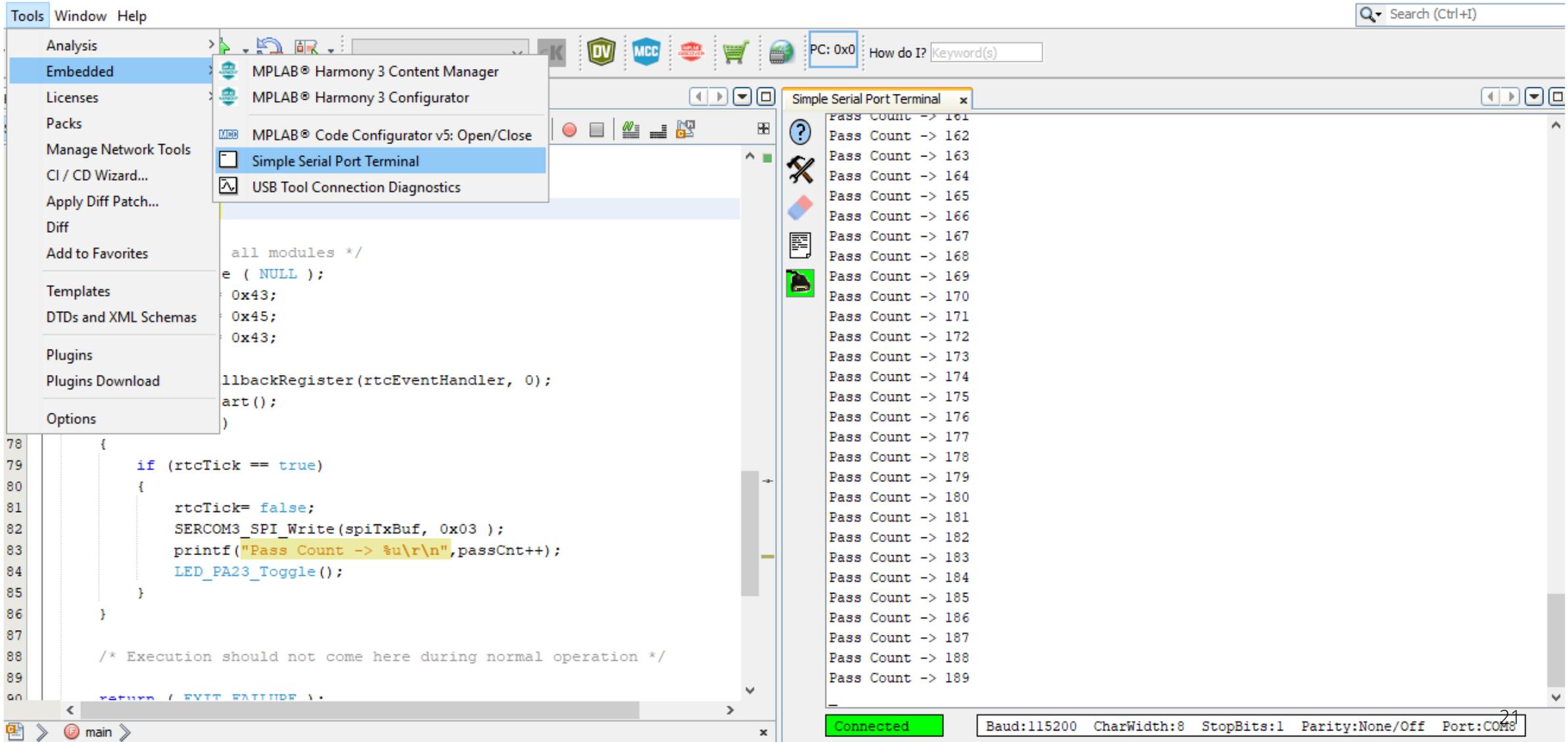
- Select SERCOM Operation Mode: USART with internal Clock
- Operating Mode: Blocking mode
- Receive Enable:
- Transmit Enable:
- Frame Format: USART frame
- Baud Rate in Hz: 115,200
- Parity Mode: No Parity
- Character Size: 8 Bits
- Stop Bit Mode: One Stop Bit
- Receive Pinout: SERCOM PAD[1] is used for data reception
- Transmit Pinout: PAD[0]=TxD, PAD[2]=RTS/TE, PAD[3]=CTS
- Enable Run in Standby:

## Add Pass Count Code

```
uint8_t spiTxBuf[8];
uint16_t passCnt;
int main ( void )
{
    /* Initialize all modules */
    SYS_Initialize ( NULL );
    spiTxBuf[0] = 0x43;
    spiTxBuf[1] = 0x45;
    spiTxBuf[2] = 0x43;
    passCnt = 0;
    RTC_Timer32CallbackRegister(rtcEventHandler, 0);
    RTC_Timer32Start();
    while ( true )
    {
        if (rtcTick == true)
        {
            rtcTick= false;
            SERCOM3_SPI_Write(spiTxBuf, 0x03 );
            printf("Pass Count -> %u\r\n",passCnt++);
            LED_PA23_Toggle();
        }
    }
}
```



# Activate Serial Port Terminal and Execute the App



The screenshot displays the MPLAB Harmony IDE interface. On the left, the 'Tools' menu is open, with 'Simple Serial Port Terminal' selected. The main editor shows the source code for a pass counter application. The code includes a loop that prints the pass count and toggles an LED. On the right, the 'Simple Serial Port Terminal' window shows the output of the application, displaying 'Pass Count -> 161' through 'Pass Count -> 189'. The terminal window also shows connection parameters: Baud:115200, CharWidth:8, StopBits:1, Parity:None/Off, Port:COM8.

```
all modules */
e ( NULL );
0x43;
0x45;
0x43;

llbackRegister(rtcEventHandler, 0);
art();
)

78 {
79     if (rtcTick == true)
80     {
81         rtcTick= false;
82         SERCOM3_SPI_Write(spiTxBuf, 0x03 );
83         printf("Pass Count -> %u\r\n",passCnt++);
84         LED_PA23_Toggle();
85     }
86 }
87
88 /* Execution should not come here during normal operation */
89
90 return ( EXIT_FAILURE );
```

Pass Count -> 161  
Pass Count -> 162  
Pass Count -> 163  
Pass Count -> 164  
Pass Count -> 165  
Pass Count -> 166  
Pass Count -> 167  
Pass Count -> 168  
Pass Count -> 169  
Pass Count -> 170  
Pass Count -> 171  
Pass Count -> 172  
Pass Count -> 173  
Pass Count -> 174  
Pass Count -> 175  
Pass Count -> 176  
Pass Count -> 177  
Pass Count -> 178  
Pass Count -> 179  
Pass Count -> 180  
Pass Count -> 181  
Pass Count -> 182  
Pass Count -> 183  
Pass Count -> 184  
Pass Count -> 185  
Pass Count -> 186  
Pass Count -> 187  
Pass Count -> 188  
Pass Count -> 189

Connected Baud:115200 CharWidth:8 StopBits:1 Parity:None/Off Port:COM8

**MORE TO COME..**

# Thank you for attending!!!

Please consider the resources below:

- [microchip.com](https://www.microchip.com)





Thank You

Sponsored by

