



**DesignNews**

# IoT Device Prototyping with Microchip Curiosity Development Boards

**Day 2:**

**Curiosity Firmware Development Using the Microchip XC8 Compiler**

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.

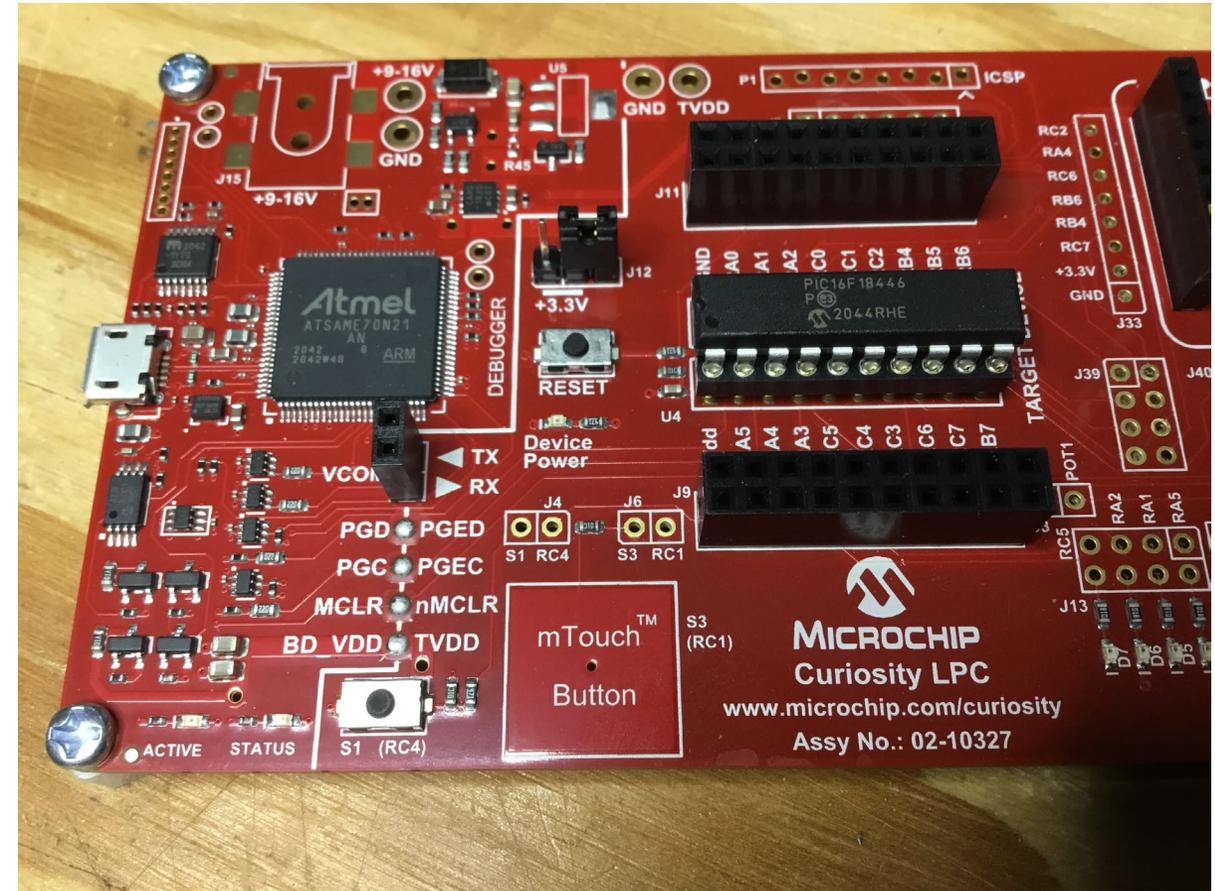
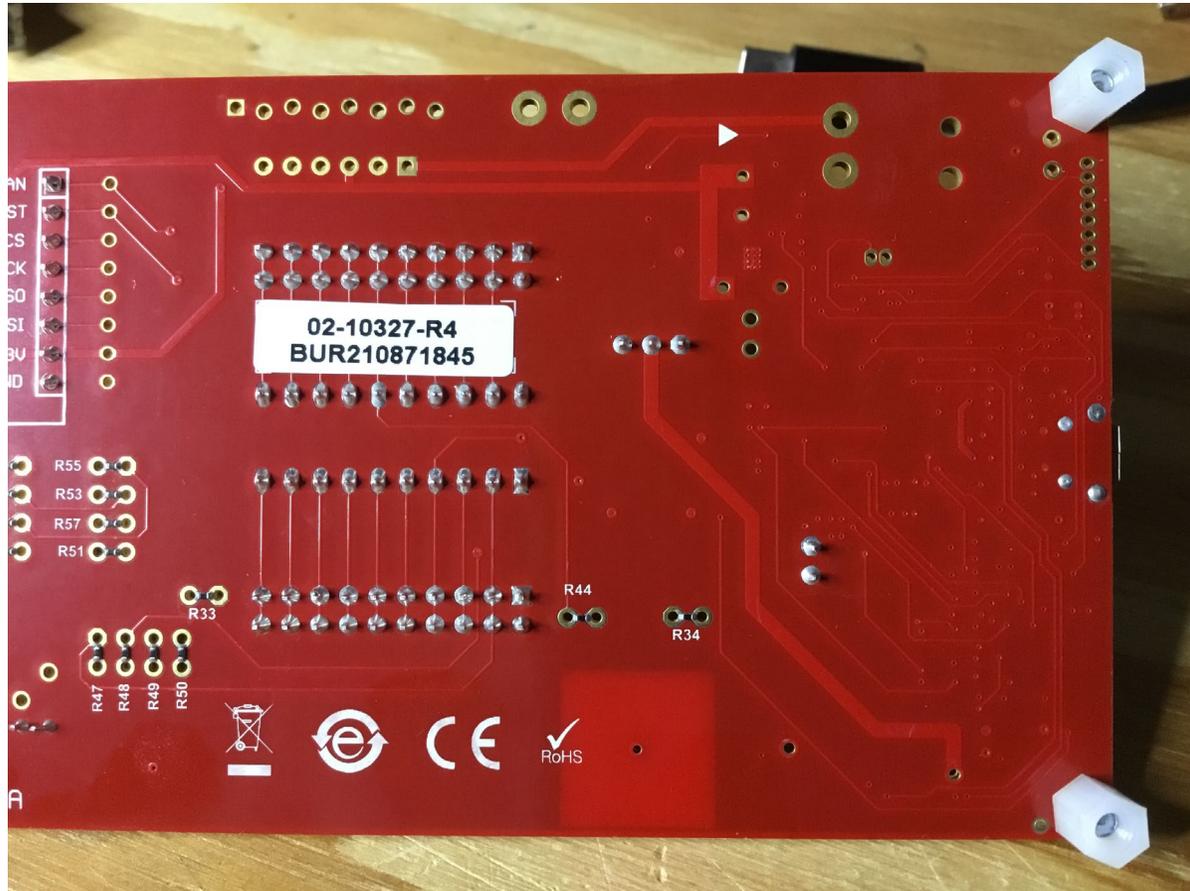
# AGENDA

- **Coding a PIC16F1619 REYAX RYLR890 Driver**
- **Coding a PIC16F18446 REYAX RYLR890 Driver**





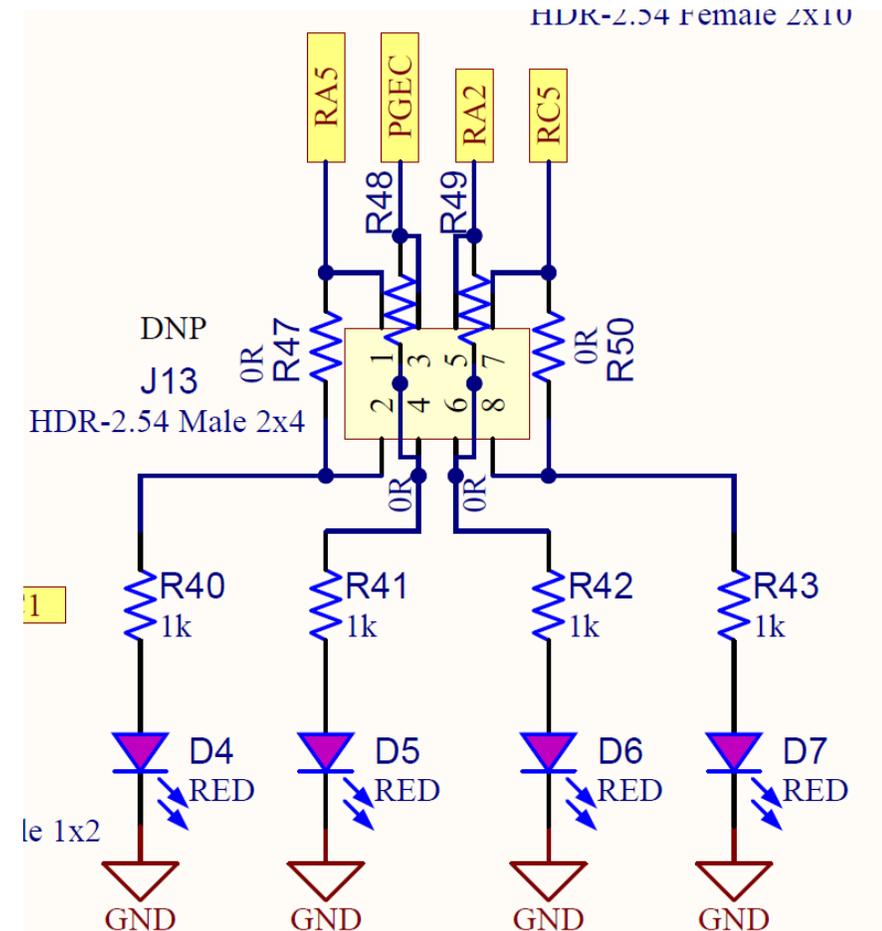
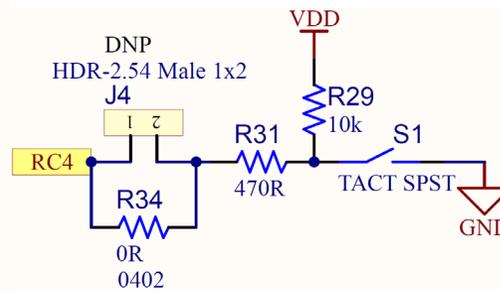
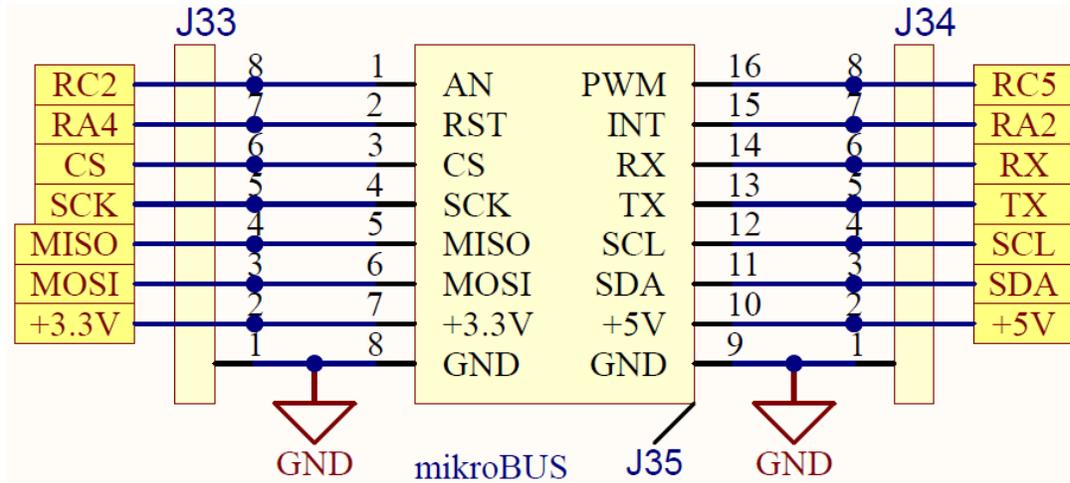
### Curiosity LPC



# Create the PIC16F1619 MPLAB X Project

The screenshot shows the 'New Project' dialog in MPLAB X IDE, divided into two panes: 'Select Device' and 'Select Compiler'. The 'Steps' pane on the left lists the following steps: 1. Choose Project, 2. Select Device, 3. Select Header, 4. Select Plugin Board, 5. Select Compiler, and 6. Select Project Name and Folder. The 'Select Device' pane has the following fields: Family: 'id-Range 8-bit MCUs (PIC10/12/16/MCP)', Device: 'PIC16F18446', and Tool: 'No Tool'. A dropdown menu for the Tool field is open, showing options: 'No Tool', 'Simulator', 'Starter Kits (PKOB)-SN: BUR.153472455', and 'Curiosity/Starter Kits (PKOB4)-SN: BUR.210871845'. The 'Select Compiler' pane shows a tree view of 'Compiler Toolchains' with 'XC8 (v2.36) [C:\Program Files\Microchip\xc8\v2.36\bin]' selected. Red arrows point to the 'Device' field with the text 'CHOOSE TARGET PIC', to the 'Curiosity/Starter Kits' option with the text 'NEW LPC', and to the 'ORIGINAL' text with an arrow pointing to the 'No Tool' option. The MPLAB X IDE logo is visible in the bottom left corner. Navigation buttons '< Back', 'Next >', 'Finish', 'Cancel', and 'Help' are present at the bottom of both panes.

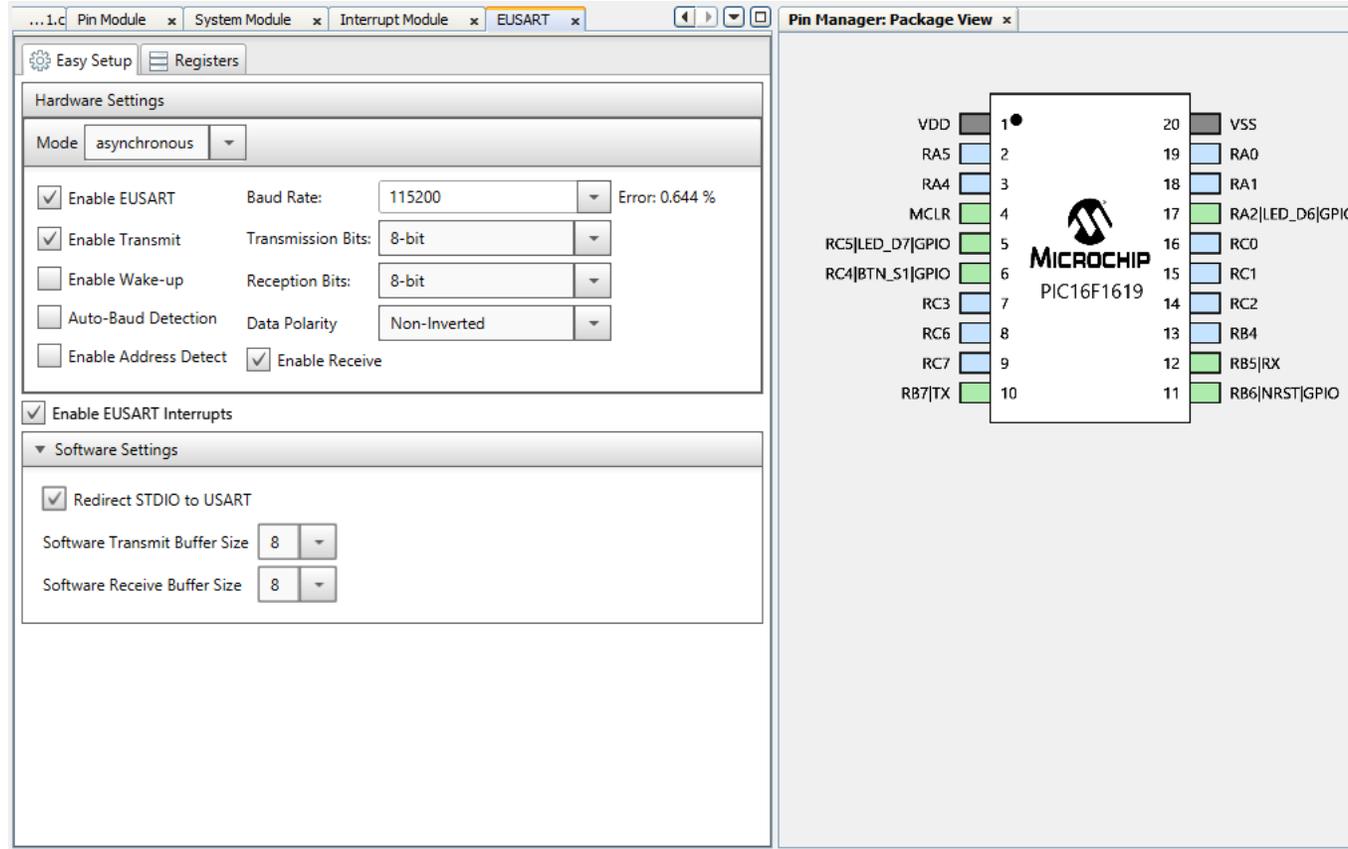
## Curiosity Pin and LED Layouts



J39 HDR-2.54 Male 2x4

J40 HDR-2.54 Male 2x4

# MCC Configuration – EUSART



The screenshot shows the MCC interface for configuring the EUSART module. The left pane displays the configuration settings, and the right pane shows the Pin Manager Package View for the PIC16F1619.

**Hardware Settings:**

- Mode: asynchronous
- Enable EUSART:  Baud Rate: 115200 Error: 0.644 %
- Enable Transmit:  Transmission Bits: 8-bit
- Enable Wake-up:  Reception Bits: 8-bit
- Auto-Baud Detection:  Data Polarity: Non-Inverted
- Enable Address Detect:  Enable Receive:

**Software Settings:**

- Enable EUSART Interrupts:
- Redirect STDIO to USART:
- Software Transmit Buffer Size: 8
- Software Receive Buffer Size: 8

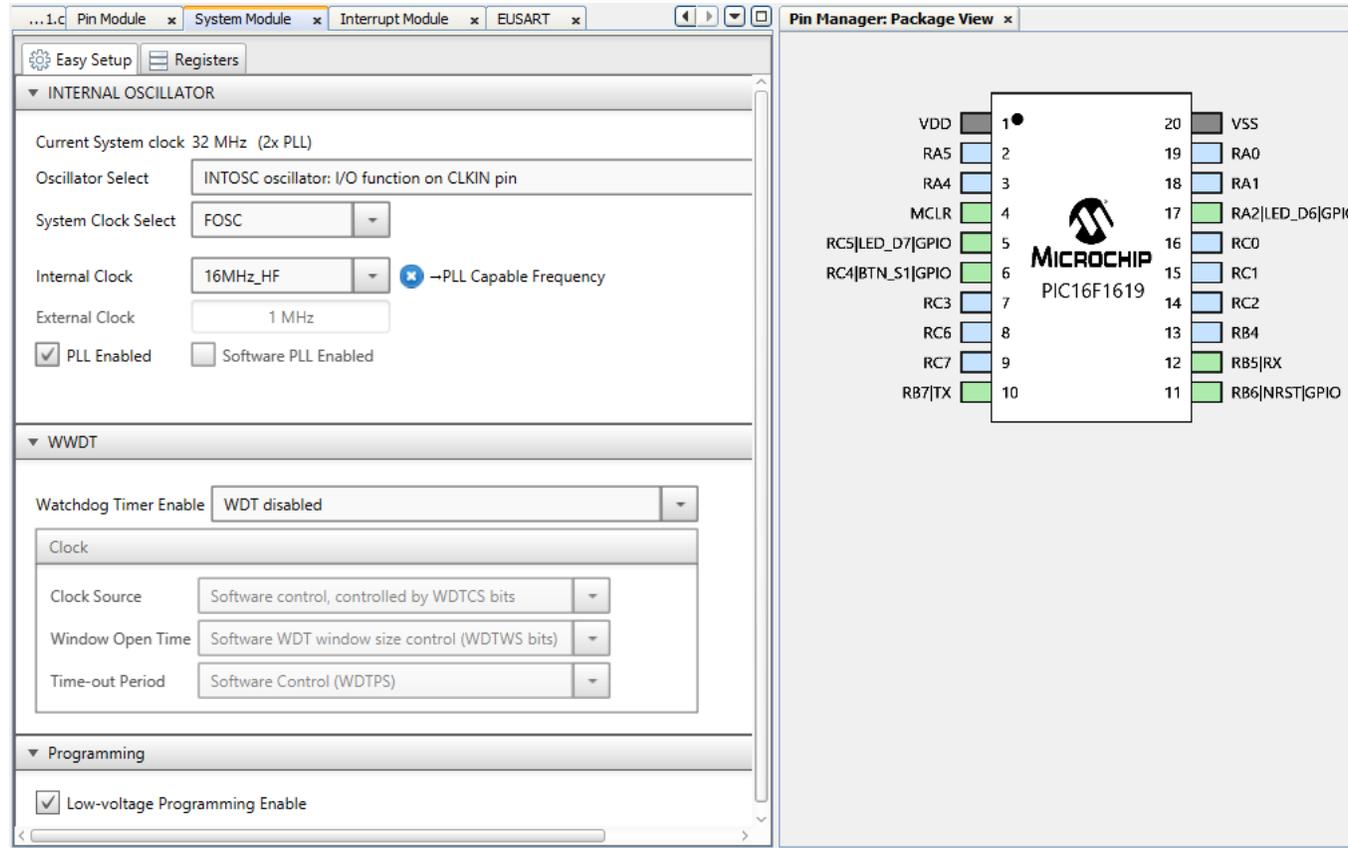
**Pin Manager Package View:**

The package view shows the PIC16F1619 chip with pins 1 through 20. The pins are color-coded and labeled as follows:

- 1: VDD
- 2: RA5
- 3: RA4
- 4: MCLR
- 5: RC5|LED\_D7|GPIO
- 6: RC4|BTN\_S1|GPIO
- 7: RC3
- 8: RC6
- 9: RC7
- 10: RB7|TX
- 11: RB6|NRST|GPIO
- 12: RB5|RX
- 13: RB4
- 14: RC2
- 15: RC1
- 16: RC0
- 17: RA2|LED\_D6|GPIO
- 18: RA1
- 19: RA0
- 20: VSS

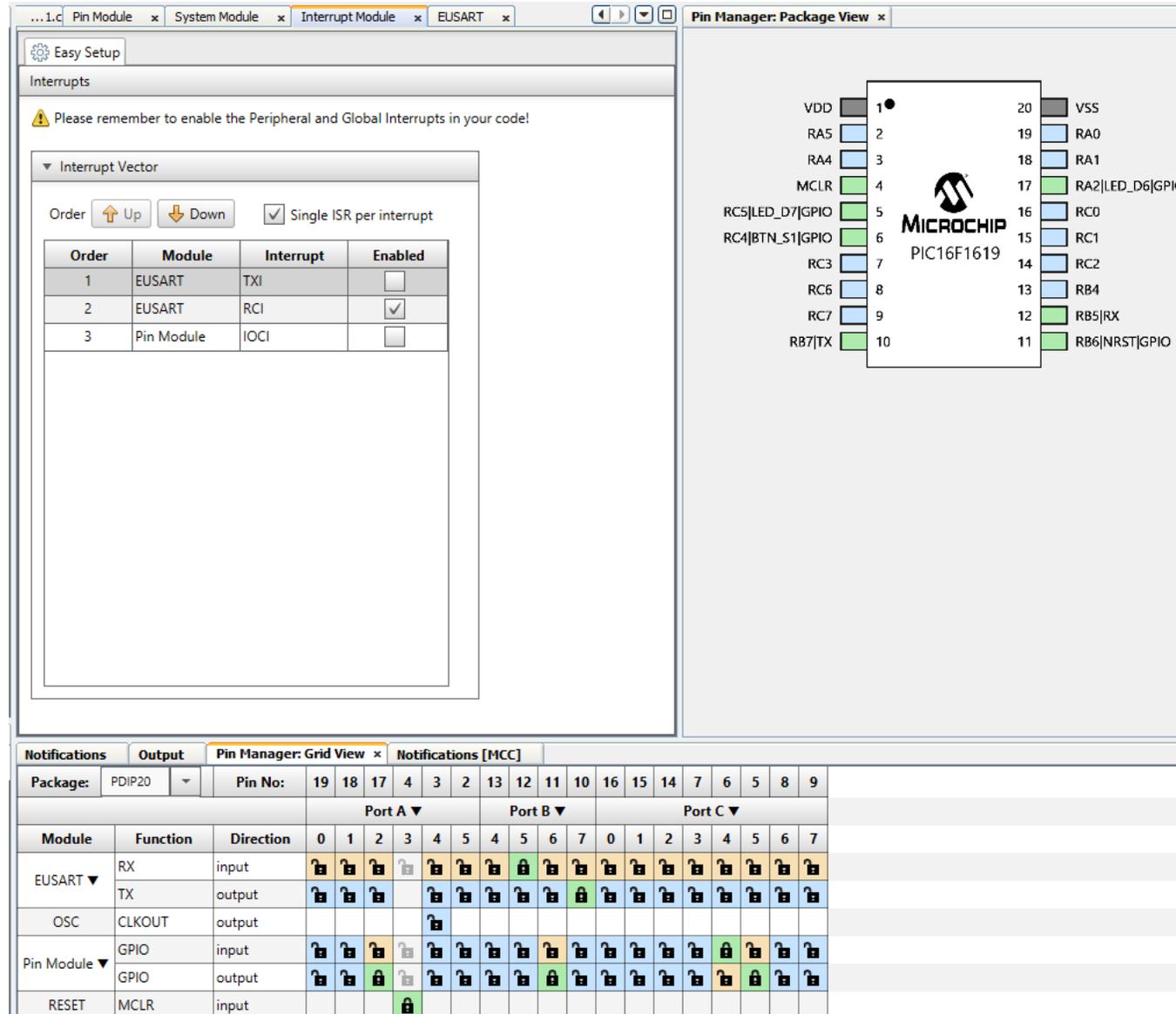
Package:	PDIP20	Pin No:	19	18	17	4	3	2	13	12	11	10	16	15	14	7	6	5	8	9
			Port A				Port B				Port C									
Module	Function	Direction	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1
EUSART	RX	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
	TX	output	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
OSC	CLKOUT	output					🔒													
Pin Module	GPIO	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
	GPIO	output	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
RESET	MCLR	input			🔒															

# MCC Configuration – Clock/Programming Mode



Notifications			Output			Pin Manager: Grid View			Notifications [MCC]											
Package:	PDIP20	Pin No:	19	18	17	4	3	2	13	12	11	10	16	15	14	7	6	5	8	9
			Port A					Port B					Port C							
Module	Function	Direction	0	1	2	3	4	5	4	5	6	7	0	1	2	3	4	5	6	7
EUSART	RX	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
	TX	output	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
OSC	CLKOUT	output					🔒													
Pin Module	GPIO	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
	GPIO	output	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
RESET	MCLR	input			🔒															

# MCC Configuration – EUSART Interrupts



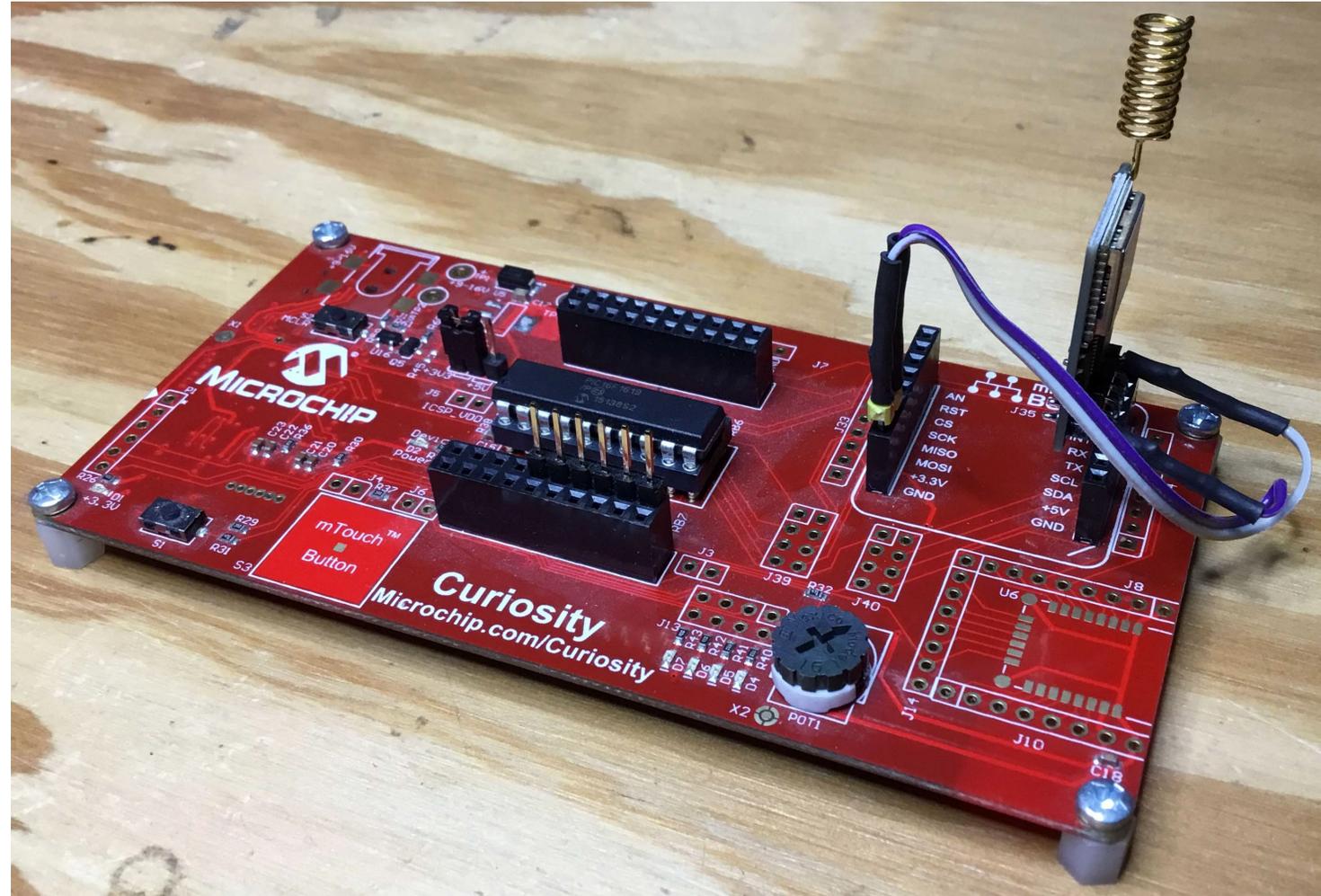
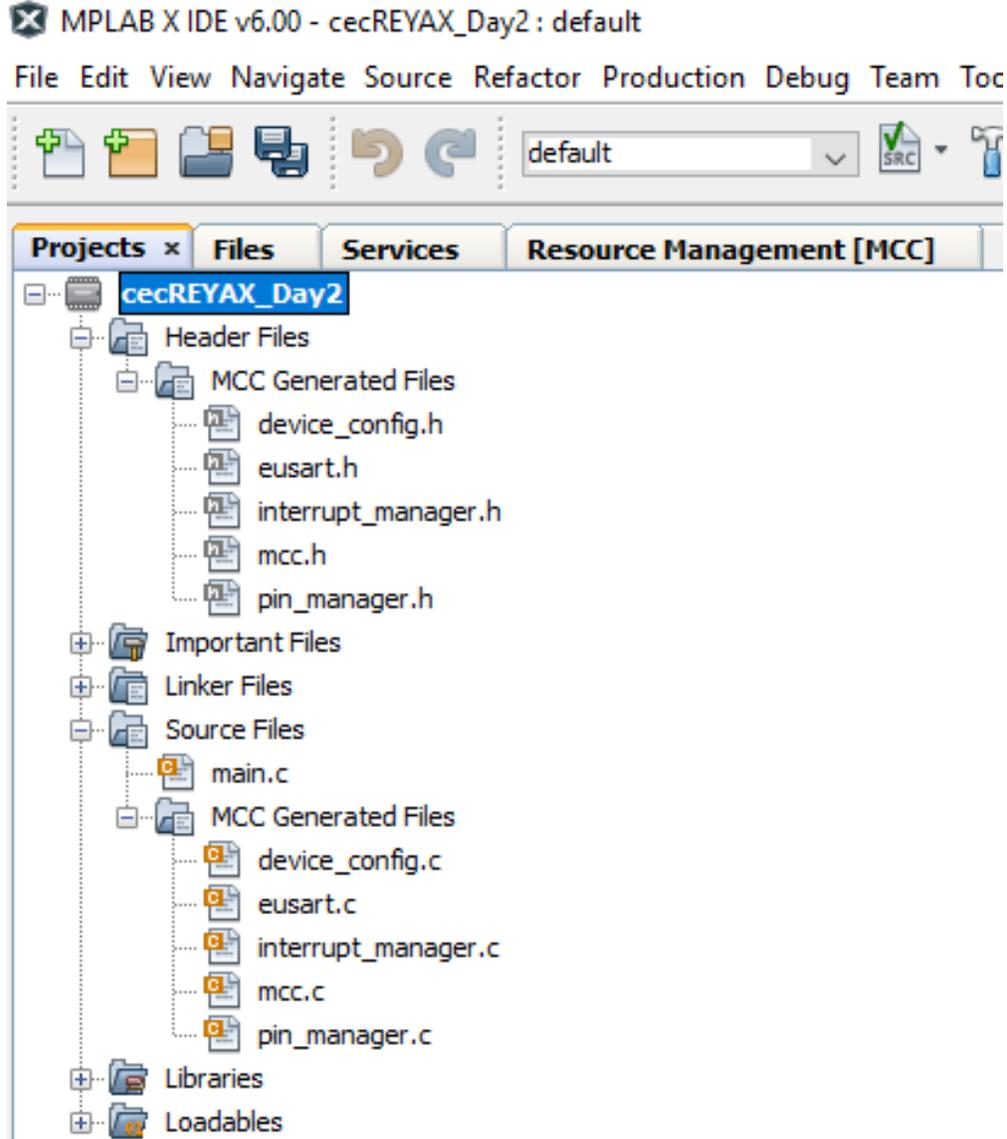
The screenshot shows the MCC interface with the following components:

- Interrupts Section:**
  - Warning: Please remember to enable the Peripheral and Global Interrupts in your code!
  - Interrupt Vector Table:**

Order	Module	Interrupt	Enabled
1	EUSART	TXI	<input type="checkbox"/>
2	EUSART	RCI	<input checked="" type="checkbox"/>
3	Pin Module	IOCI	<input type="checkbox"/>
- Pin Manager: Package View:** Shows a diagram of the PIC16F1619 chip with pins 1-20 and their functions:
  - 1: VDD, 2: RA5, 3: RA4, 4: MCLR, 5: RC5|LED\_D7|GPIO, 6: RC4|BTN\_S1|GPIO, 7: RC3, 8: RC6, 9: RC7, 10: RB7|TX, 11: RB6|NRST|GPIO, 12: RB5|RX, 13: RB4, 14: RC2, 15: RC1, 16: RC0, 17: RA2|LED\_D6|GPIO, 18: RA1, 19: RA0, 20: VSS.
- Pin Manager: Grid View:** A table showing pin configurations for Port A, Port B, and Port C.
 

Module	Function	Direction	Port A							Port B							Port C										
			0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	
EUSART	RX	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
	TX	output	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
OSC	CLKOUT	output					🔒																				
Pin Module	GPIO	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
	GPIO	output	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
RESET	MCLR	input				🔒																					

## MCC-Generated Project Tree



# EUSART/EUSART Interrupt Handler Verification

```
#include "mcc_generated_files/mcc.h"
uint8_t rxBite;
void main(void)
{
    // initialize the device
    SYSTEM_Initialize();

    // When using interrupts, you need to set the Global and Peripheral Interrupt Enable bits
    // Use the following macros to:

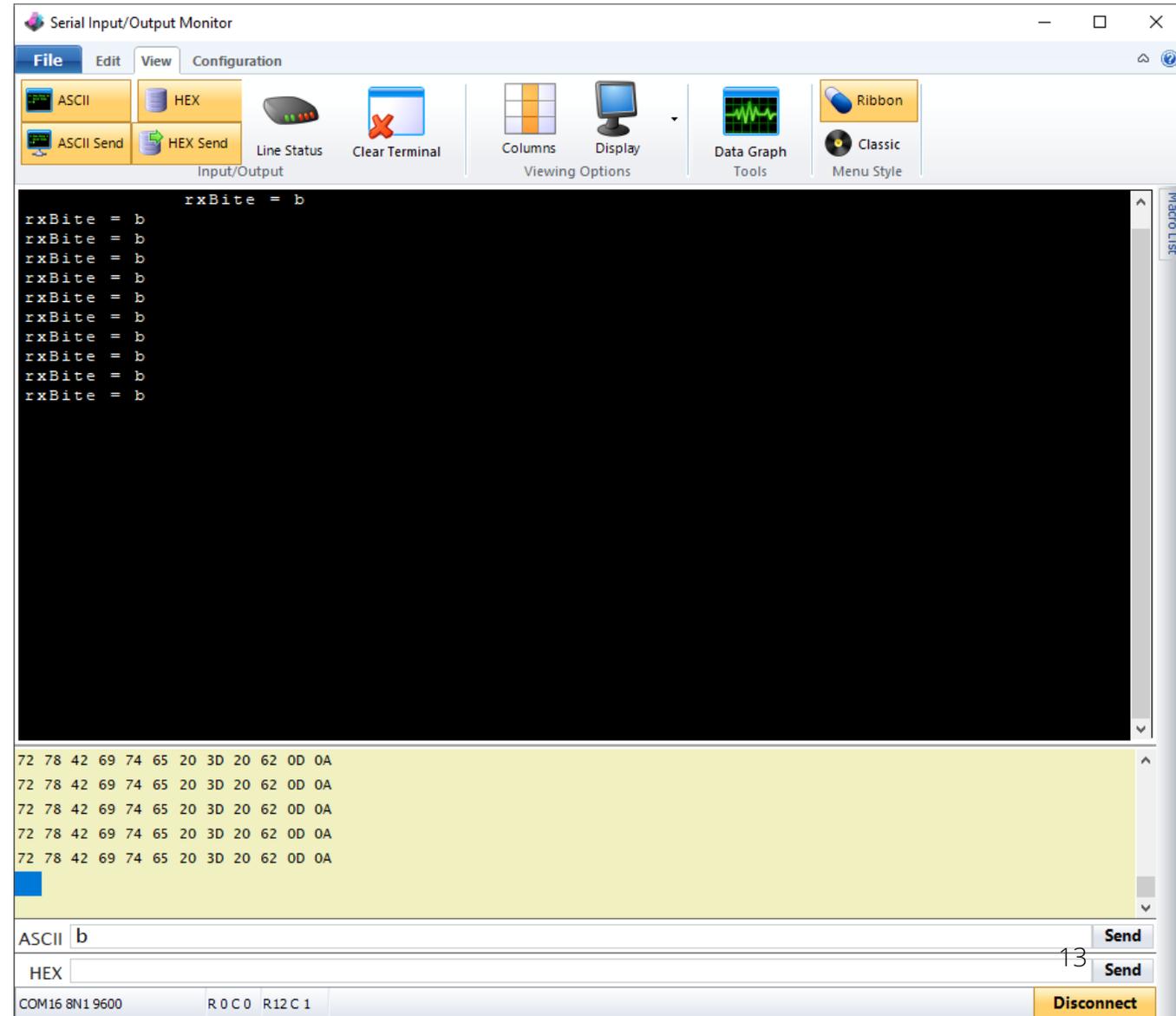
    // Enable the Global Interrupts
    INTERRUPT_GlobalInterruptEnable();

    // Enable the Peripheral Interrupts
    INTERRUPT_PeripheralInterruptEnable();

    // Disable the Global Interrupts
    //INTERRUPT_GlobalInterruptDisable();

    // Disable the Peripheral Interrupts
    //INTERRUPT_PeripheralInterruptDisable();

    while (1)
    {
        do{
            rxBite = EUSART_Read();
            printf("rxBite = %c\r\n",rxBite);
        }while(EUSART_is_rx_ready());
    }
}
```



Serial Input/Output Monitor

File Edit View Configuration

ASCII HEX Line Status Clear Terminal Columns Display Data Graph Ribbon Classic

Input/Output Viewing Options Tools Menu Style

```
rxBite = b
```

72	78	42	69	74	65	20	3D	20	62	0D	0A
72	78	42	69	74	65	20	3D	20	62	0D	0A
72	78	42	69	74	65	20	3D	20	62	0D	0A
72	78	42	69	74	65	20	3D	20	62	0D	0A
72	78	42	69	74	65	20	3D	20	62	0D	0A

ASCII b Send

HEX 13 Send

COM16 8N1 9600 R 0 C 0 R 12 C 1 Disconnect

## EUSART Support Code

```
#include "mcc_generated_files/mcc.h"
```

```
uint8_t rxBuf[64];
uint8_t indx;
uint8_t loraState;
uint8_t txBite;
```

```
#define EUSART_RX_BUFFER_SIZE 64
extern volatile uint8_t
eusartRxBuffer[EUSART_RX_BUFFER_SIZE];
```

```
enum
{
    sWakeUp,
    sTx,
    sRx
};
```

```
void main(void)
{
    SYSTEM_Initialize();
```

```
    INTERRUPT_GlobalInterruptEnable();
    INTERRUPT_PeripheralInterruptEnable();
```

```
    loraState = sWakeUp;
```

### eusart.c

```
#define EUSART_TX_BUFFER_SIZE 64
#define EUSART_RX_BUFFER_SIZE 64
```

```
/**
 * Section: Global Variables
 */
```

```
volatile uint8_t eusartRxHead = 0;
volatile uint8_t eusartRxTail = 0;
volatile uint8_t
eusartRxBuffer[EUSART_RX_BUFFER_SIZE];
volatile eusart_status_t
eusartRxStatusBuffer[EUSART_RX_BUFFER_SIZE];
volatile uint8_t eusartRxCount;
volatile eusart_status_t eusartRxLastError;
```

## sWakeUp State

```
while (1)
{
    /* State = sWakeUp
    if(loraState == sWakeUp)
    {
        indx = 0;

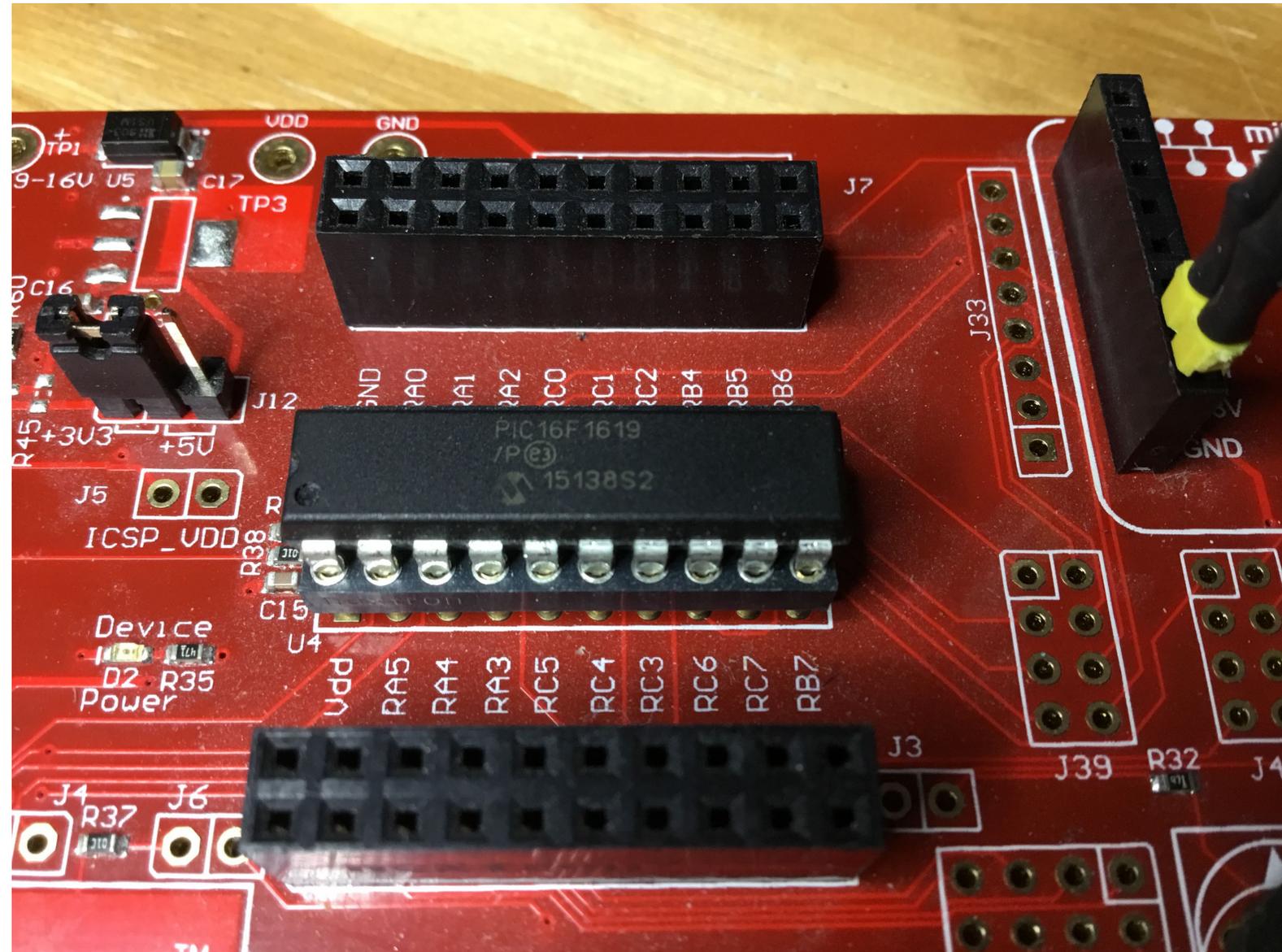
        NRST_SetLow();
        __delay_ms(10);
        NRST_SetHigh();

        while(EUSART_is_rx_ready() == false);
        __delay_ms(100);
        do{
            rxBuf[indx++] = EUSART_Read();
        }while(EUSART_is_rx_ready());

        indx = 0;
        memset(rxBuf,0x00,sizeof(rxBuf));
        printf("AT\r\n");

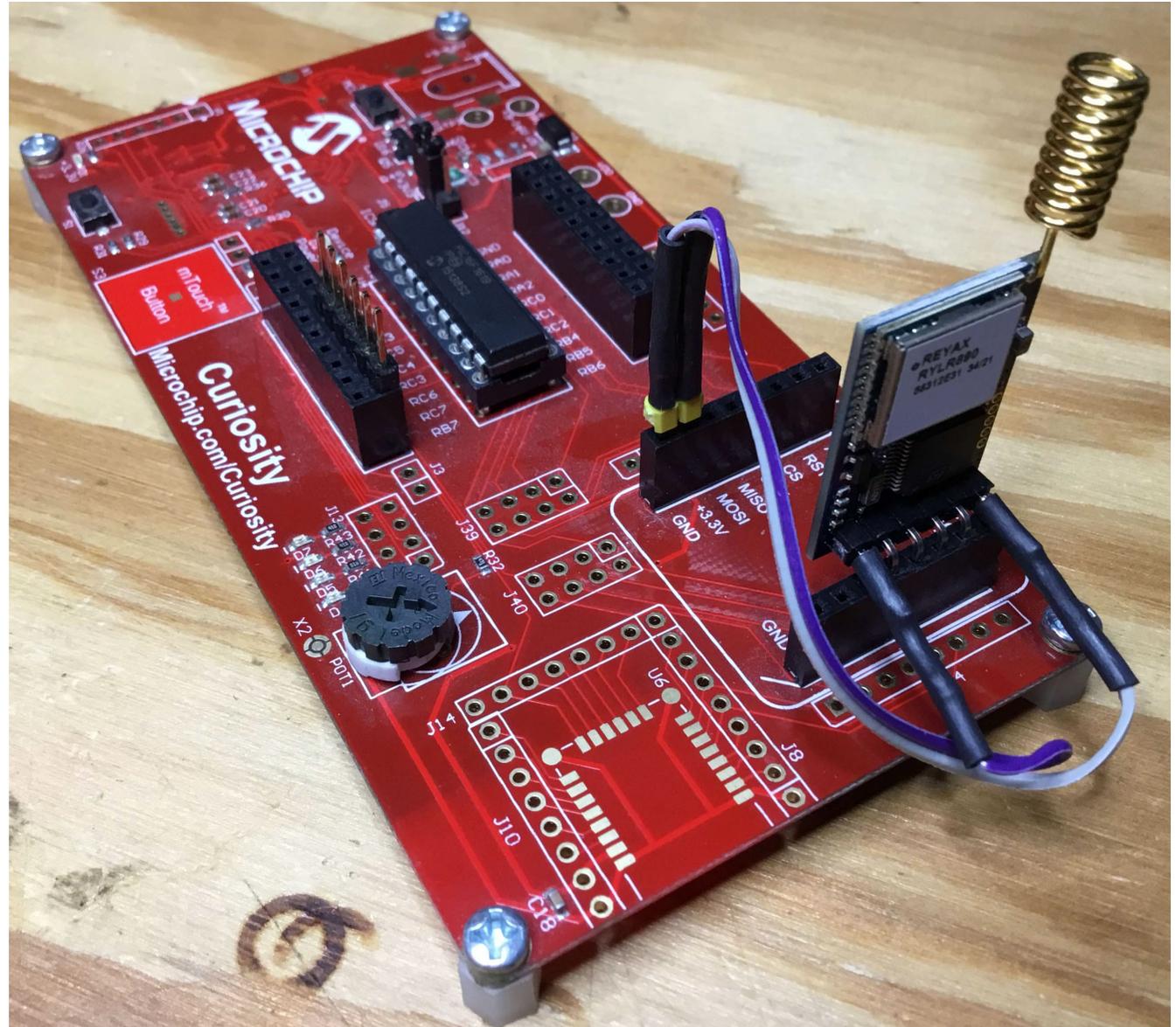
        while(EUSART_is_rx_ready() == false);
        __delay_ms(100);
        do{
            rxBuf[indx++] = EUSART_Read();
        }while(EUSART_is_rx_ready());

        indx -= 3;
        if(rxBuf[indx] == 'K')
        {
            loraState = sTx;
        }
    }
}
```

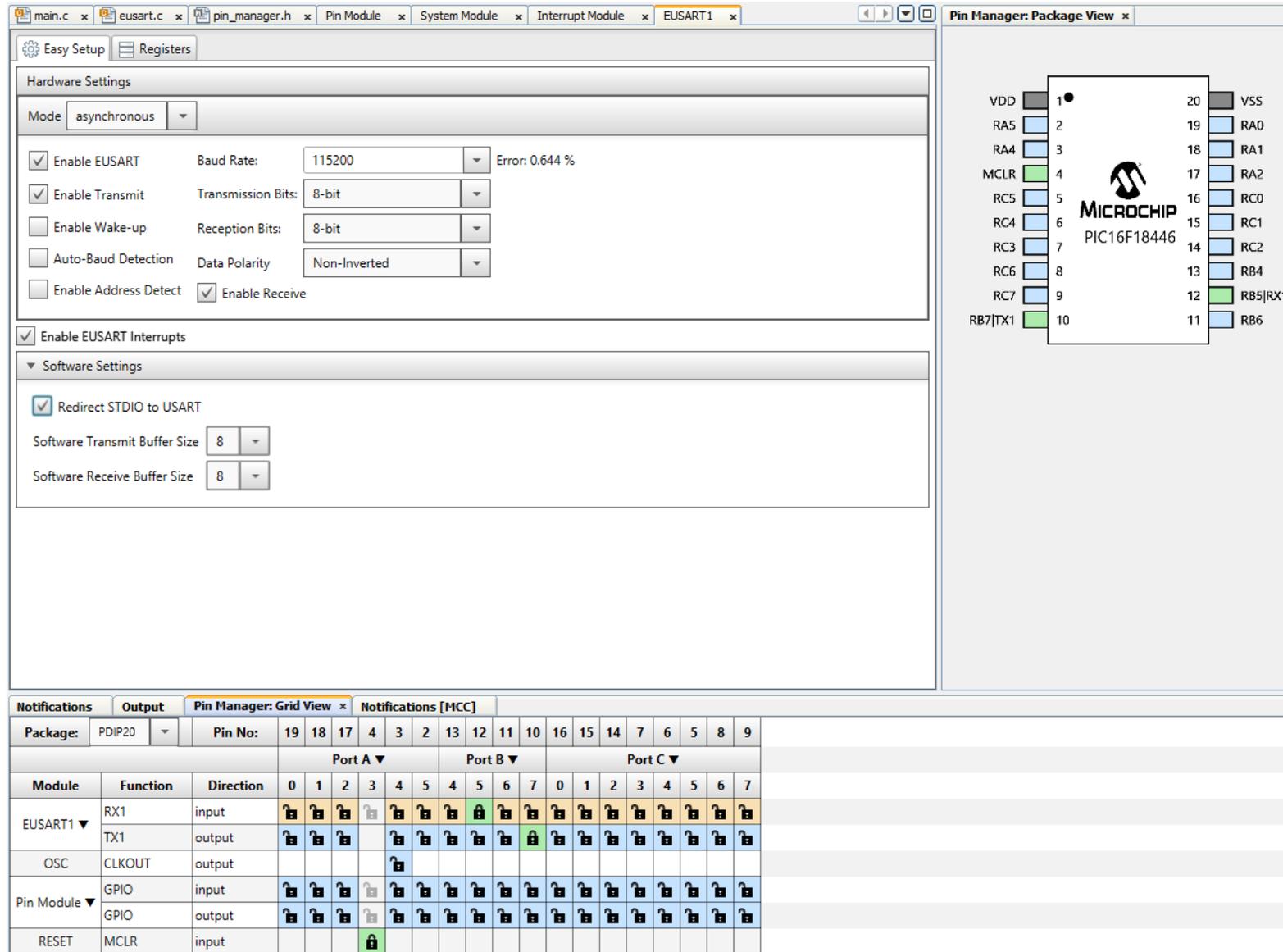


## sTx State

```
/* State = sTx
   if(loraState == sTx)
   {
       txBite = 0x42;
       printf("AT+SEND=0,1,%c\r\n",txBite);
       __delay_ms(1000);
       LED_D6_Toggle();
   }
   }//while(1))
} //main
```



# MCC Configuration – EUSART



**Hardware Settings**

Mode: asynchronous

Enable EUSART    Baud Rate: 115200    Error: 0.644 %

Enable Transmit    Transmission Bits: 8-bit

Enable Wake-up    Reception Bits: 8-bit

Auto-Baud Detection    Data Polarity: Non-Inverted

Enable Address Detect     Enable Receive

Enable EUSART Interrupts

**Software Settings**

Redirect STDIO to USART

Software Transmit Buffer Size: 8

Software Receive Buffer Size: 8

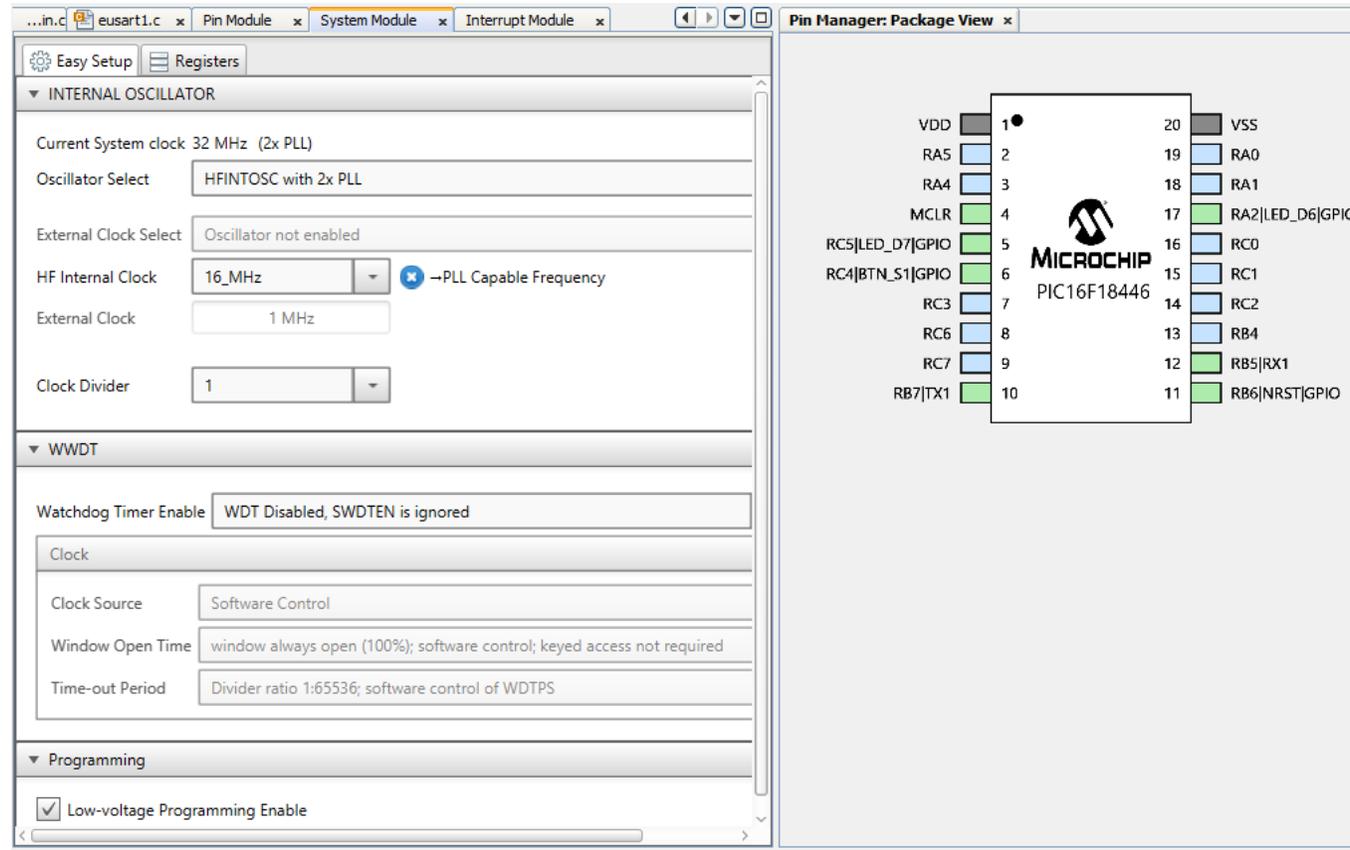
**Pin Manager: Package View**

VDD 1, VSS 20, RA5 2, RA0 19, RA4 3, RA1 18, MCLR 4, RA2 17, RC5 5, RC0 16, RC4 6, RC1 15, RC3 7, RC2 14, RC6 8, RB4 13, RC7 9, RB5|RX1 12, RB7|TX1 10, RB6 11

**Pin Manager: Grid View**

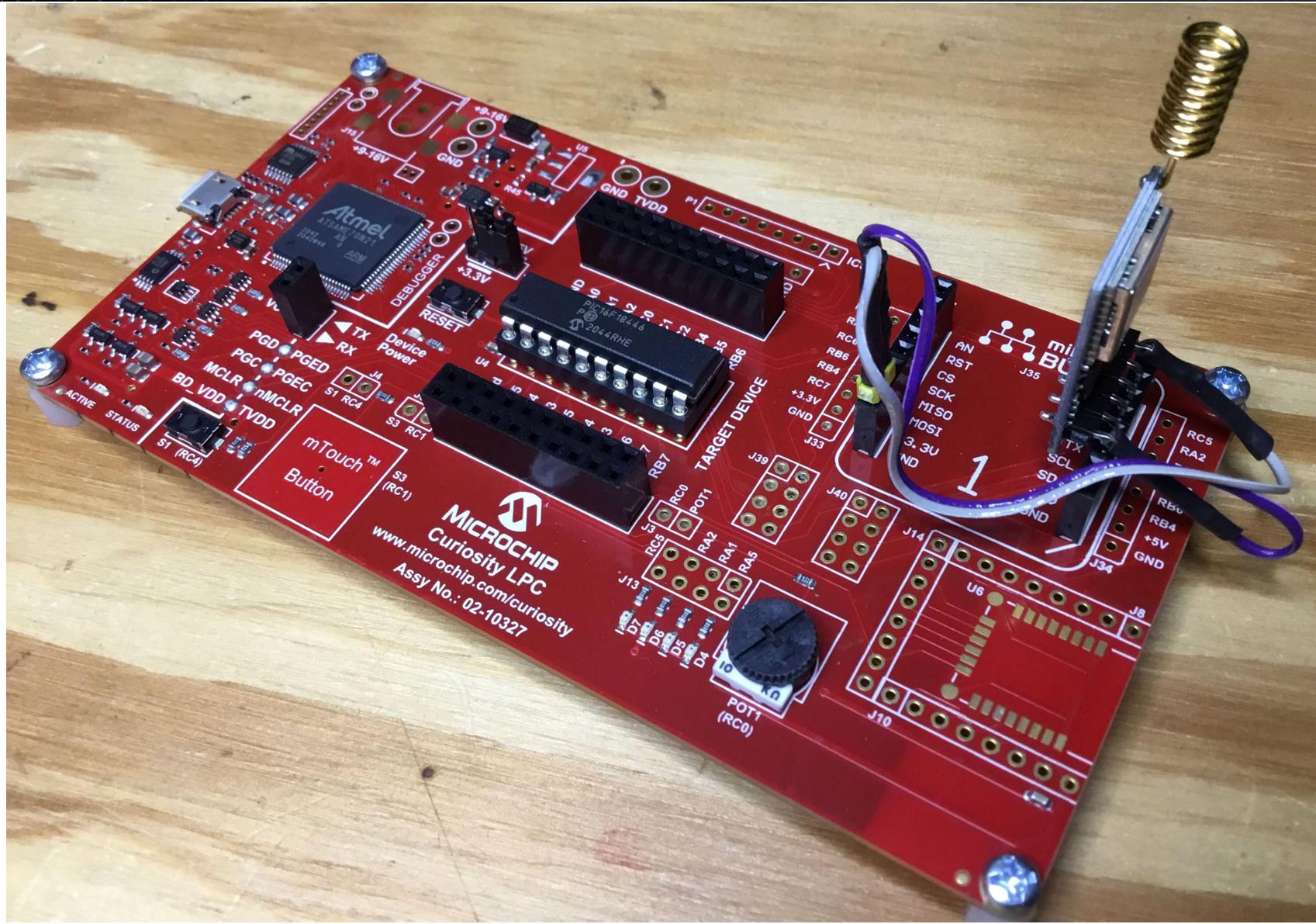
Package:	PDIP20	Pin No:	19	18	17	4	3	2	13	12	11	10	16	15	14	7	6	5	8	9
			Port A ▼					Port B ▼					Port C ▼							
Module	Function	Direction	0	1	2	3	4	5	4	5	6	7	0	1	2	3	4	5	6	7
EUSART1 ▼	RX1	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
	TX1	output	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
OSC	CLKOUT	output					🔒													
Pin Module ▼	GPIO	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
	GPIO	output	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
RESET	MCLR	input				🔒														

# MCC Configuration – PIC16F18446

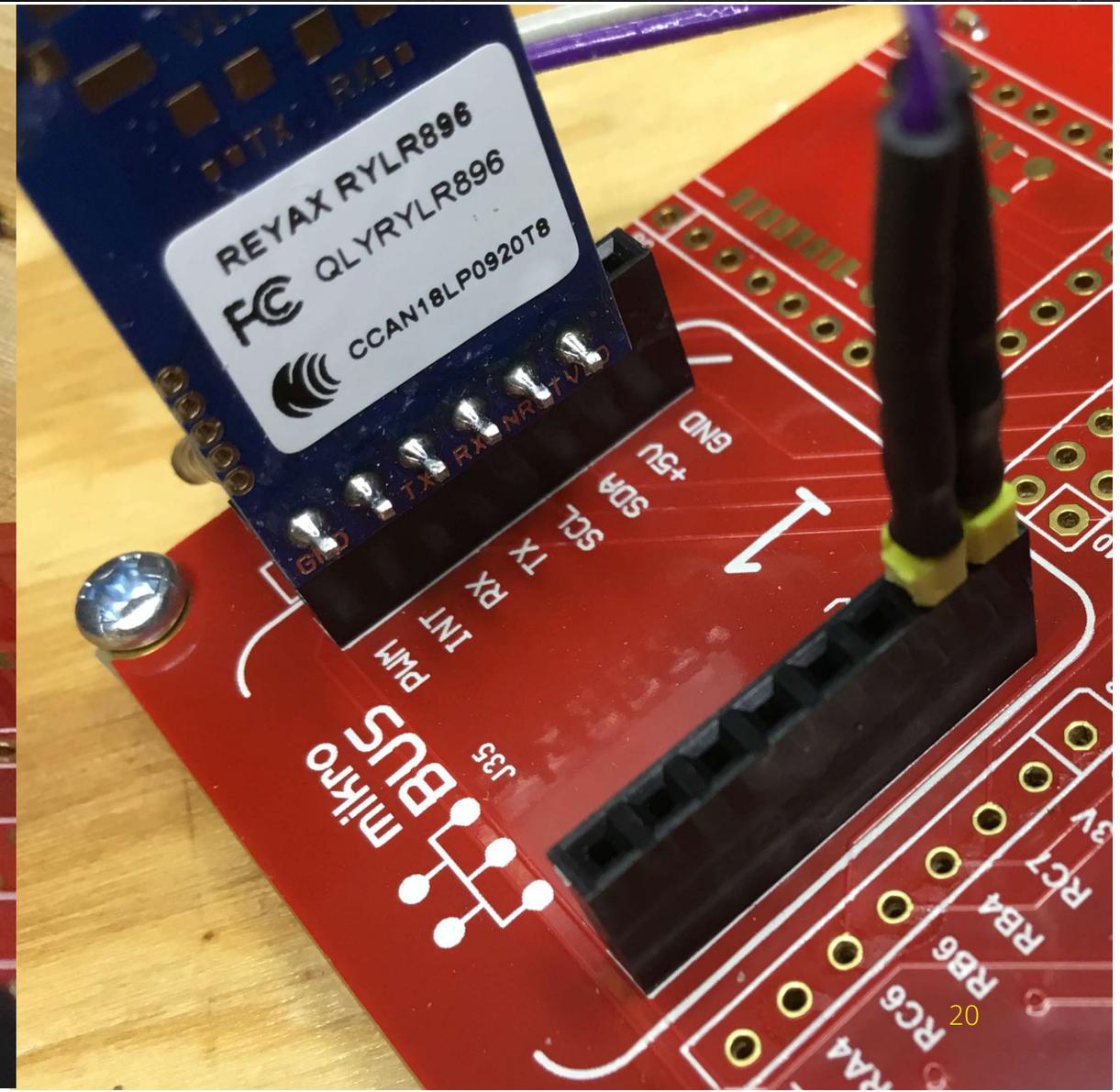
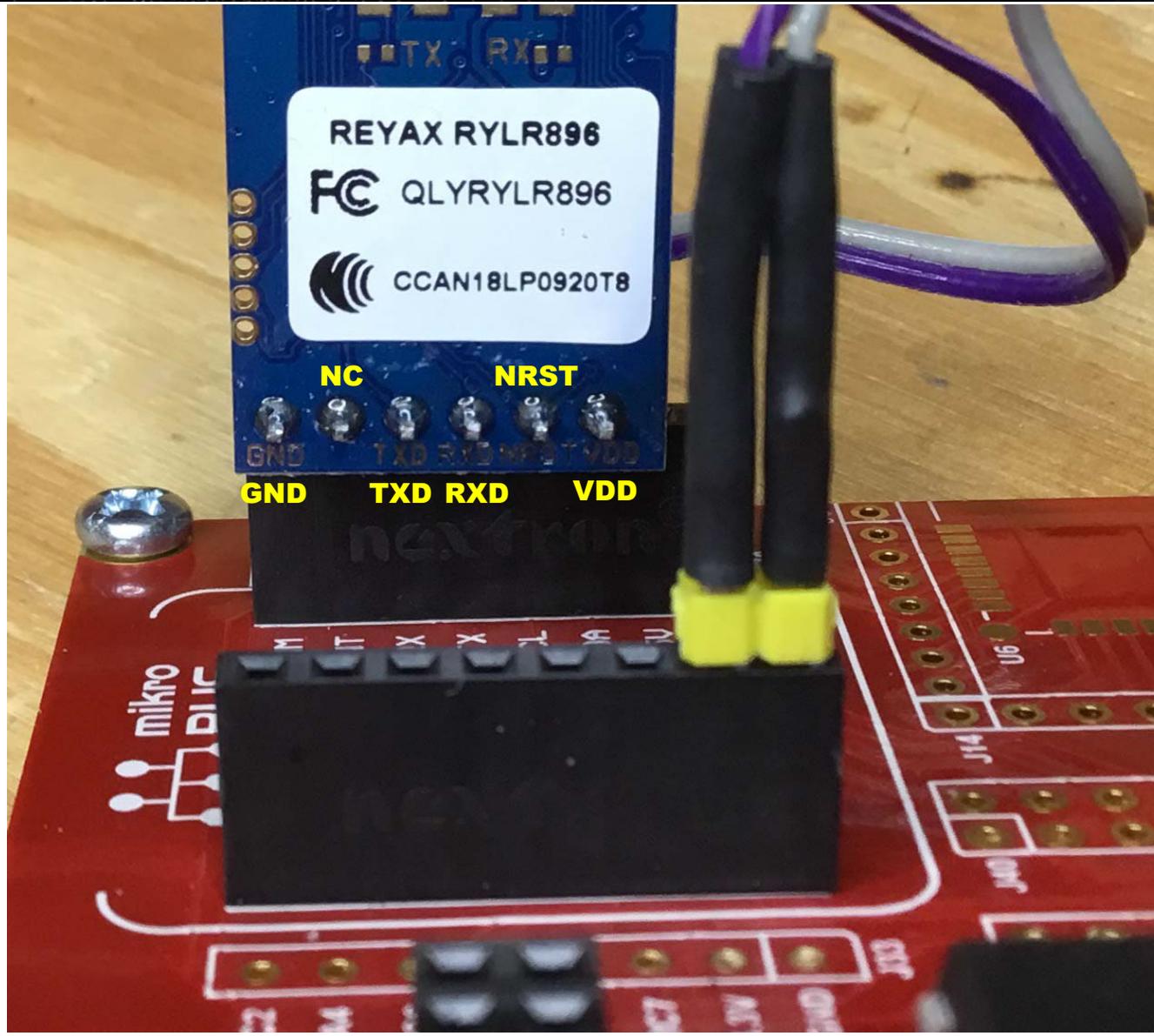


Notifications			Output			Pin Manager: Grid View			Notifications [MCC]											
Package:	PDIP20	Pin No:	19	18	17	4	3	2	13	12	11	10	16	15	14	7	6	5	8	9
Module	Function	Direction	Port A ▼					Port B ▼					Port C ▼							
			0	1	2	3	4	5	4	5	6	7	0	1	2	3	4	5	6	7
EUSART1 ▼	RX1	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
	TX1	output	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
OSC	CLKOUT	output					🔒													
Pin Module ▼	GPIO	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
	GPIO	output	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
RESET	MCLR	input				🔒														

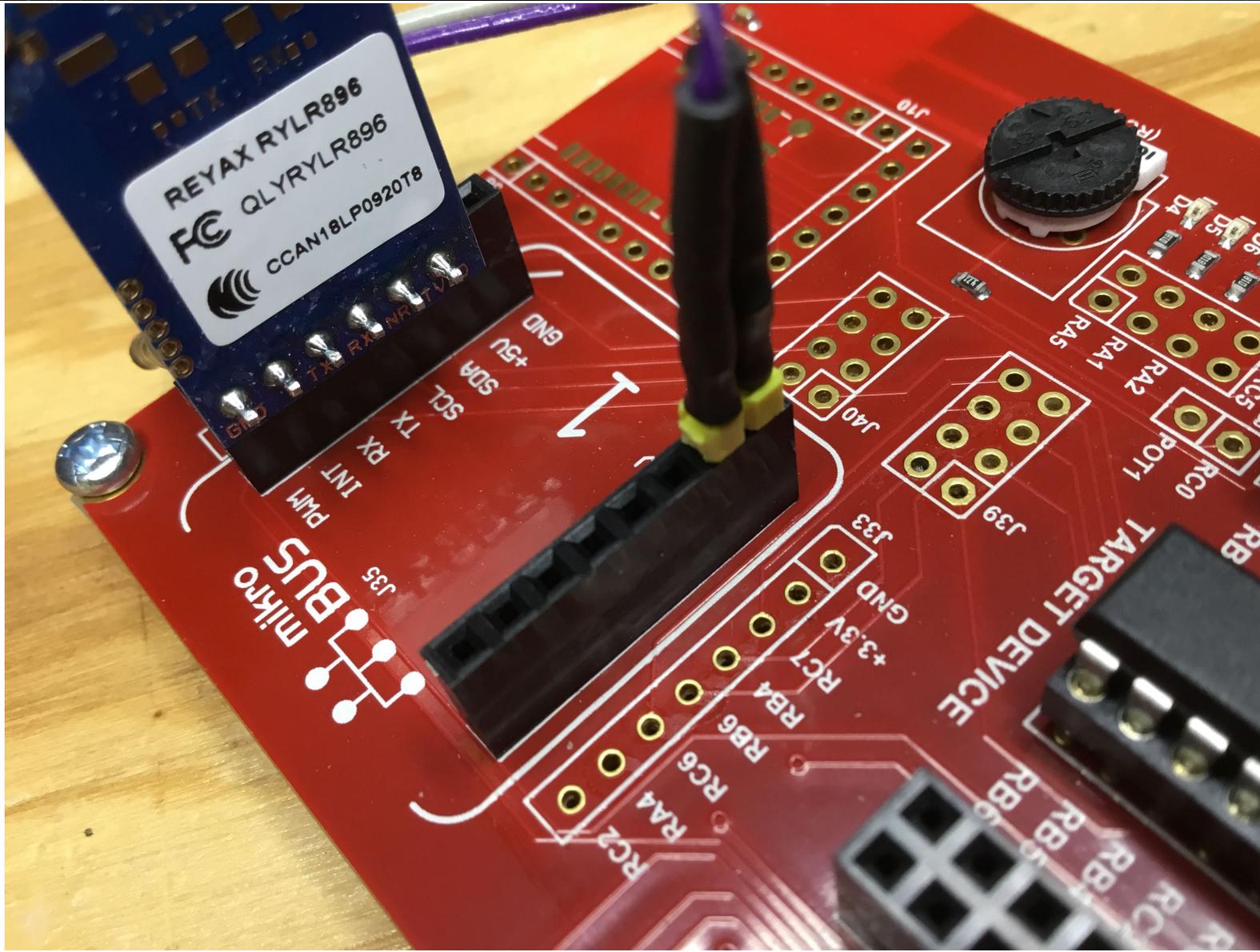
## MCC Configuration – PIC16F18446

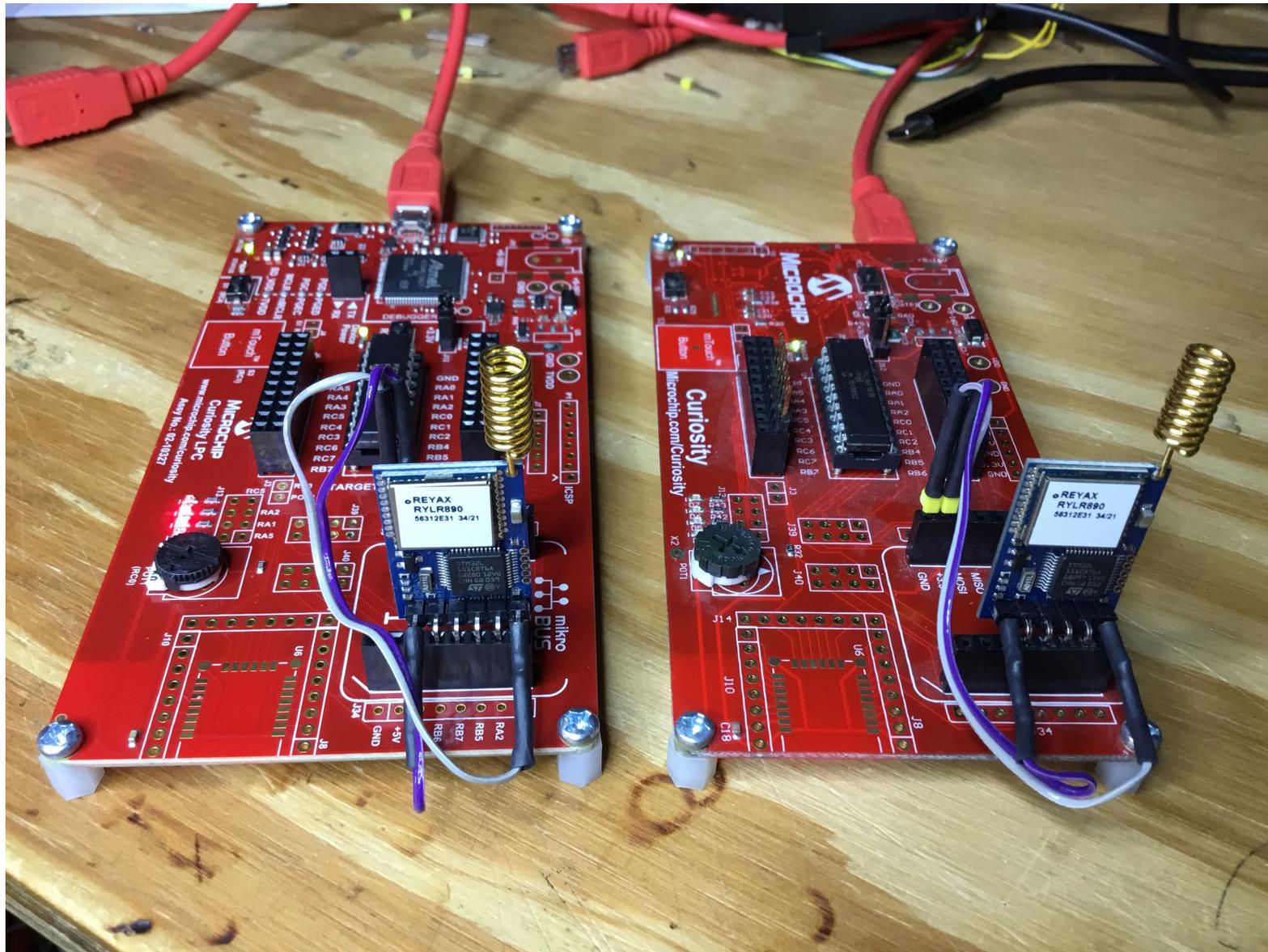


# MCC Configuration – PIC16F18446



## MCC Configuration – PIC16F18446



**PIC16F18446 – PIC16F1619**

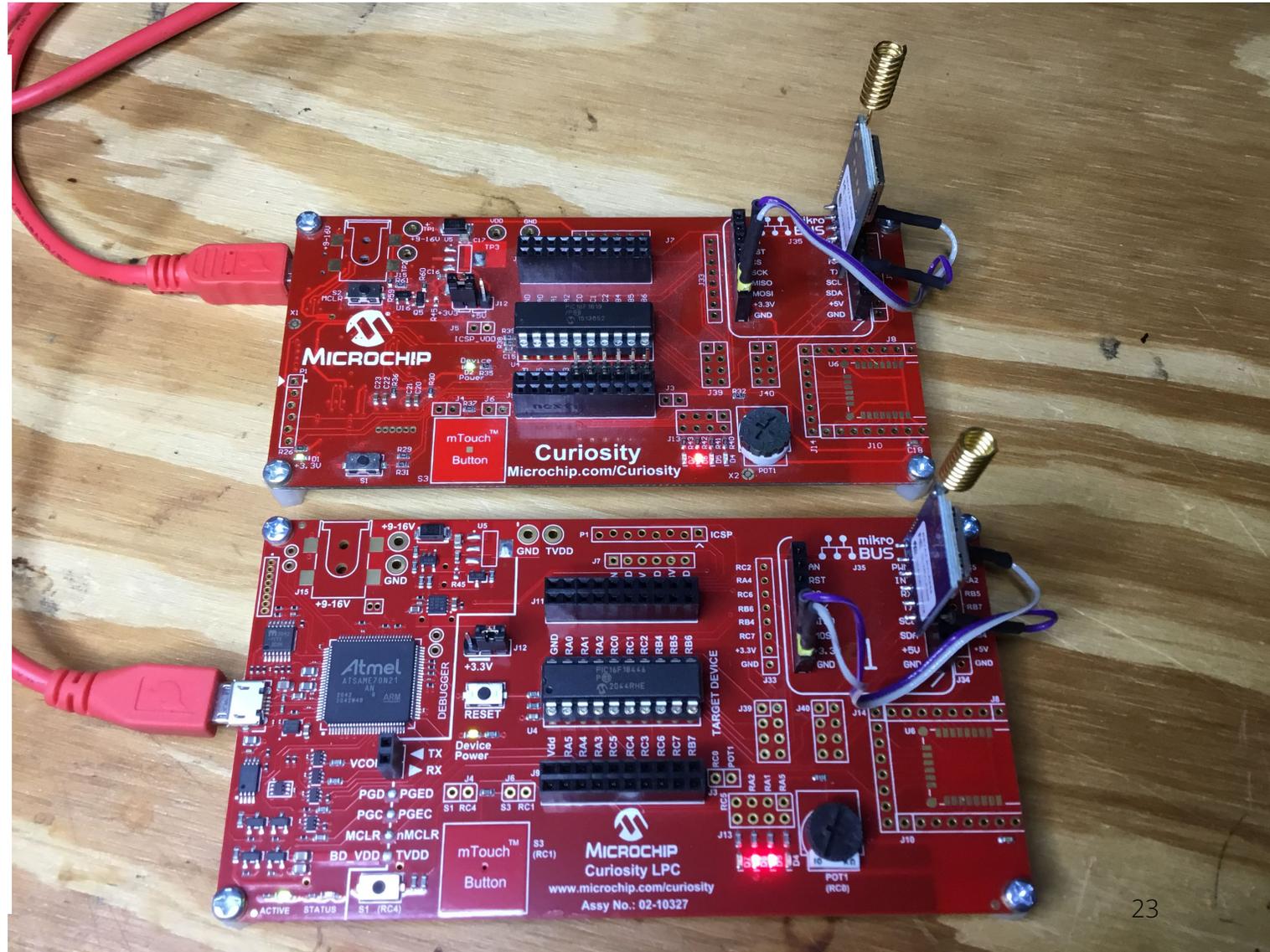
## PIC16F1619 – PIC16F18446

### 11. AT+SEND Send data to the appointment Address

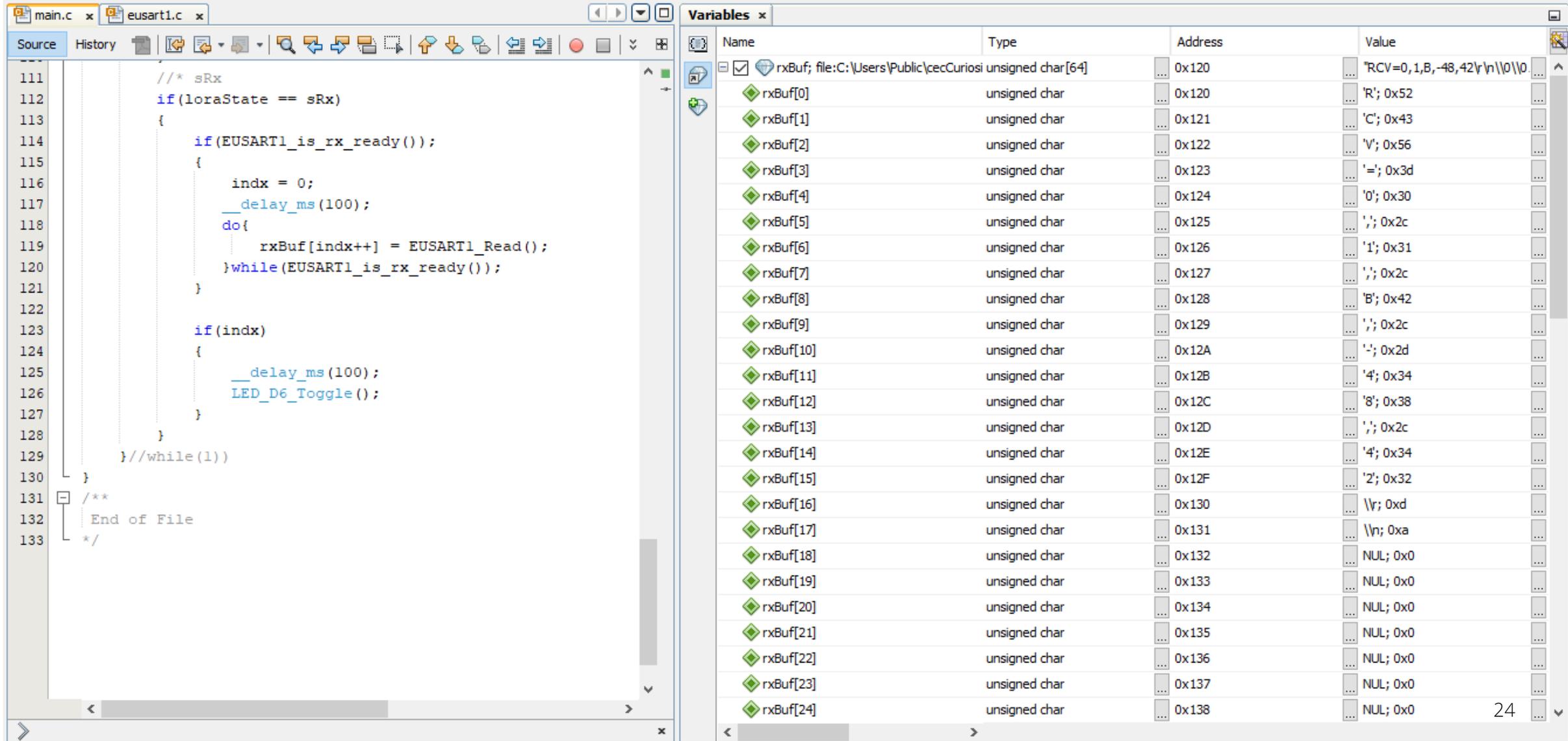
Syntax	Response
AT+SEND=<Address>,<Payload Length>,<Data>	+OK
<Address>0~65535, When the <Address> is 0, it will send data to all address (From 0 to 65535.)	
<Payload Length>Maximum 240bytes	
<Data>ASCII Format	
Example : Send HELLO string to the Address 50, AT+SEND=50,5,HELLO	
AT+SEND?	+SEND=50,5,HELLO

### 12. +RCV Show the received data

Syntax	Response
+RCV=<Address>,<Length>,<Data>,<RSSI>,<SNR>	
<Address>Transmitter Address ID	
<Length>Data Length	
<Data>Data	
<RSSI> Received Signal Strength Indicator	
<SNR> Signal-to-noise ratio	
Example: Module received the ID Address 50 send 5 bytes data, Content is HELLO string ,RSSI is -99dBm, SNR is 40, It will show as below. +RCV=50,5,HELLO,-99,40	



# MCC Configuration – PIC16F18446



The screenshot displays an IDE with two windows. The left window shows the source code for `eusart1.c`, and the right window shows the Variables window.

```
111  /** sRx
112  if(loraState == sRx)
113  {
114      if(EUSART1_is_rx_ready());
115      {
116          indx = 0;
117          __delay_ms(100);
118          do{
119              rxBuf[indx++] = EUSART1_Read();
120          }while(EUSART1_is_rx_ready());
121          }
122
123      if(indx)
124      {
125          __delay_ms(100);
126          LED_D6_Toggle();
127      }
128  }
129  } //while(1))
130  }
131  /**
132  End of File
133  */
```

The Variables window shows the following data:

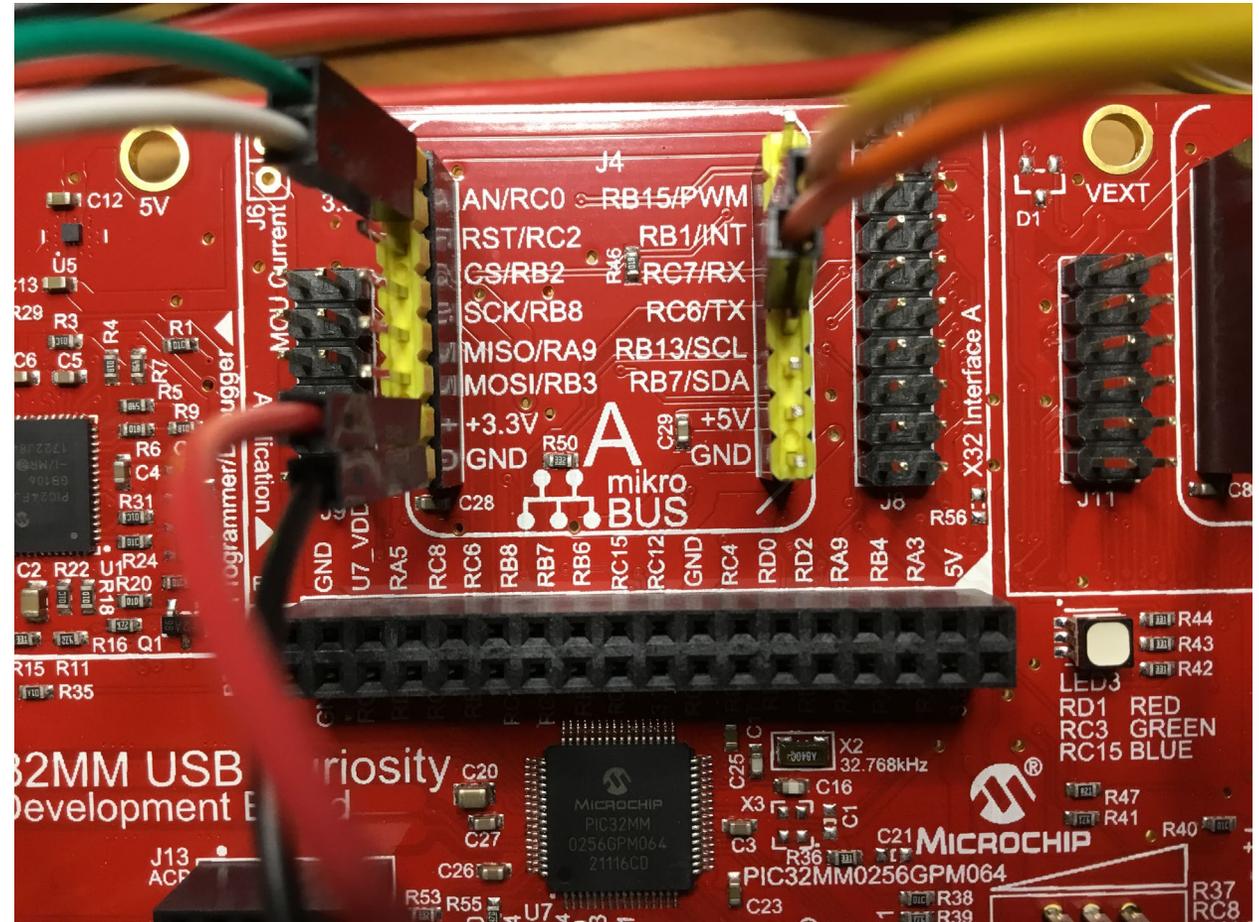
Name	Type	Address	Value
rxBuf; file:C:\Users\Public\cecCuriosi	unsigned char [64]	0x120	"RCV=0,1,B,-48,42\r\n\\0\\0
rxBuf[0]	unsigned char	0x120	'R'; 0x52
rxBuf[1]	unsigned char	0x121	'C'; 0x43
rxBuf[2]	unsigned char	0x122	'V'; 0x56
rxBuf[3]	unsigned char	0x123	'='; 0x3d
rxBuf[4]	unsigned char	0x124	'0'; 0x30
rxBuf[5]	unsigned char	0x125	';'; 0x2c
rxBuf[6]	unsigned char	0x126	'1'; 0x31
rxBuf[7]	unsigned char	0x127	';'; 0x2c
rxBuf[8]	unsigned char	0x128	'B'; 0x42
rxBuf[9]	unsigned char	0x129	';'; 0x2c
rxBuf[10]	unsigned char	0x12A	'-'; 0x2d
rxBuf[11]	unsigned char	0x12B	'4'; 0x34
rxBuf[12]	unsigned char	0x12C	'8'; 0x38
rxBuf[13]	unsigned char	0x12D	';'; 0x2c
rxBuf[14]	unsigned char	0x12E	'4'; 0x34
rxBuf[15]	unsigned char	0x12F	'2'; 0x32
rxBuf[16]	unsigned char	0x130	\r; 0xd
rxBuf[17]	unsigned char	0x131	\n; 0xa
rxBuf[18]	unsigned char	0x132	NUL; 0x0
rxBuf[19]	unsigned char	0x133	NUL; 0x0
rxBuf[20]	unsigned char	0x134	NUL; 0x0
rxBuf[21]	unsigned char	0x135	NUL; 0x0
rxBuf[22]	unsigned char	0x136	NUL; 0x0
rxBuf[23]	unsigned char	0x137	NUL; 0x0
rxBuf[24]	unsigned char	0x138	NUL; 0x0

MORE TO COME..

# Thank you for attending!!!

Please consider the resources below:

- [microchip.com](http://microchip.com)
- **REYAX LoRa AT COMMAND GUIDE**
- [lemosint.com](http://lemosint.com)





**DesignNews**

Thank You

Sponsored by

