MPLAB® Code Configurator & Core Independent Peripherals

Lab Manual

This page is intentionally left blank

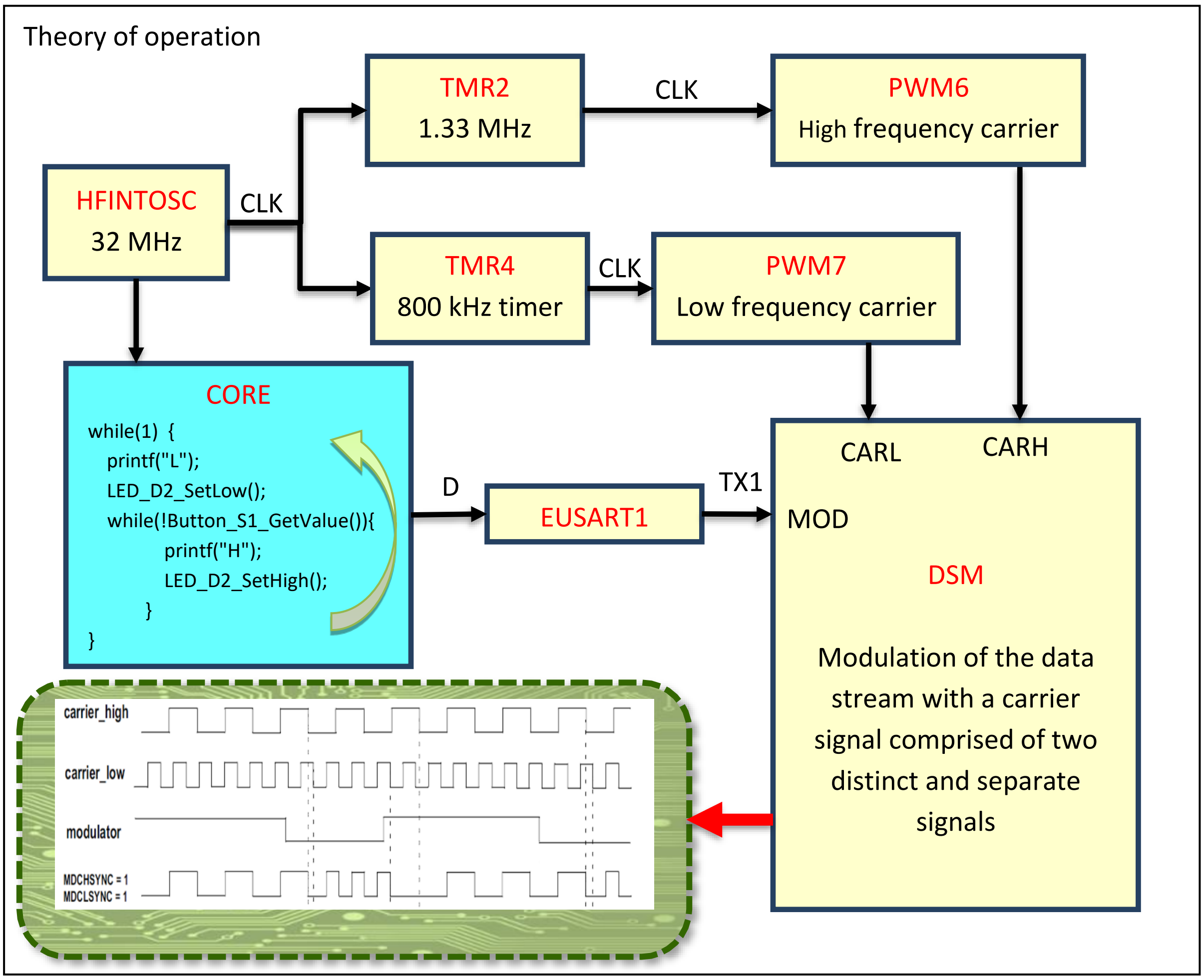
Lab 1 – FSK Transmitter with CIPs

Purpose

Demonstrate how to configure an FSK transmitter using different peripherals present on the PIC16F18446 such as timers, PWMs, DSM and UART. Curiosity HPC board is used and the project is fully implemented with MPLAB® Code Configurator.

Hardware

Function	Pin	Setting
DSM1	RA0 (16)	Output
DSM1	RA1 (15)	Output-Analyser
TX1	RC6 (5)	Output-Analyser
Button_S1	RC2 (11)	Input
LED_D2	RA2 (14)	Output



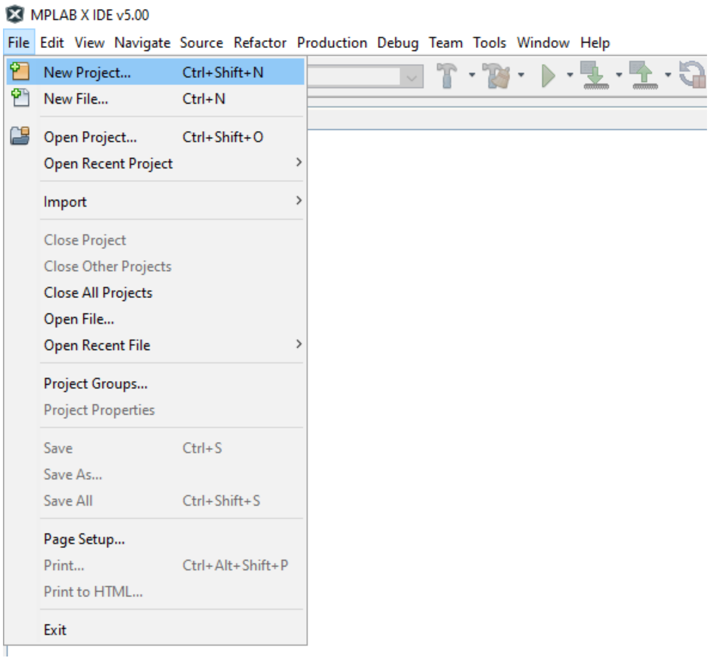
Procedure

Connect the PIC16F18446-CNANO board (DM164144) to the PC.

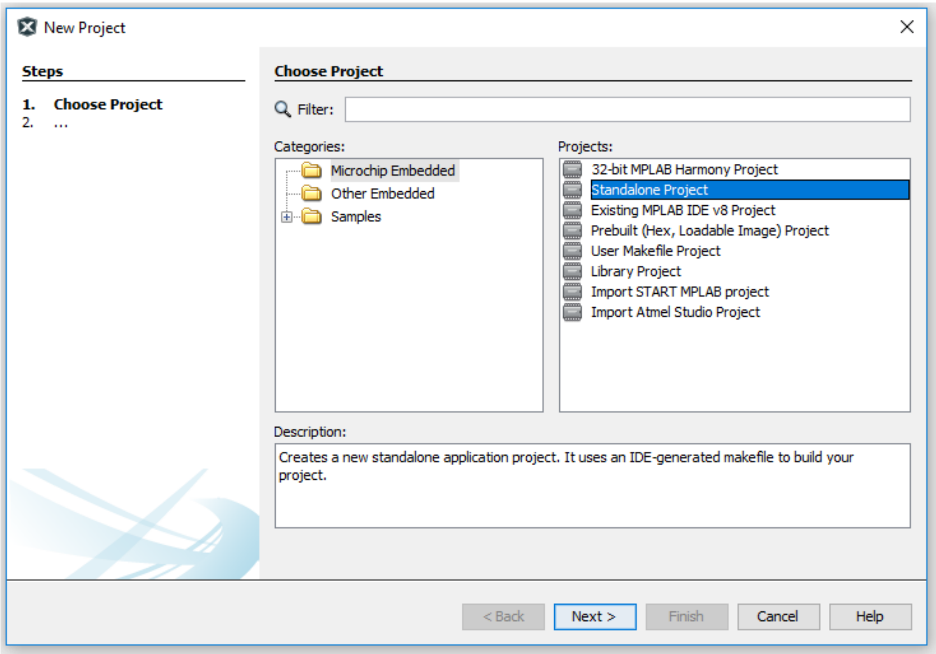
> Depending on your computer, there might be some time needed for driver installation. The debugger on-board uses HID, and the operating system has a driver for it. (Windows, Mac, Linux)

Create a new project using PIC16F18446 on CNANO

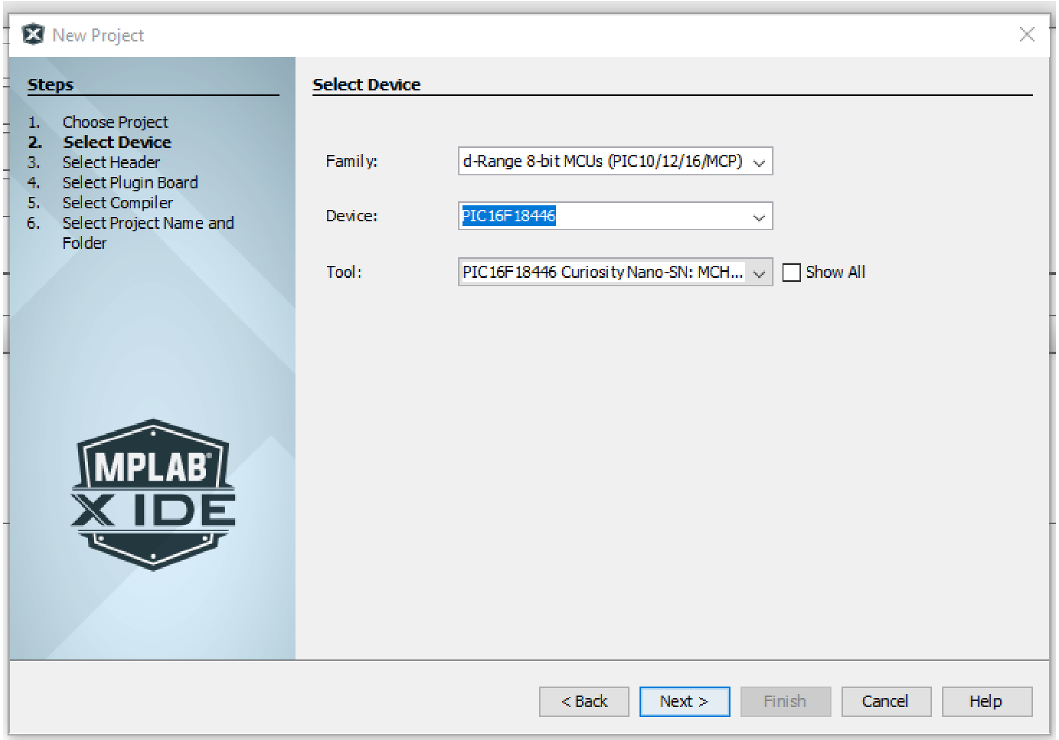
> <u>Create a new project in the MPLAB® X IDE:</u>
>
> Start MPLAB X, and then go to **File → New Project…**



> In *"New Project"* dialog navigate to **Microchip Embedded** Category and select **Standalone Project**

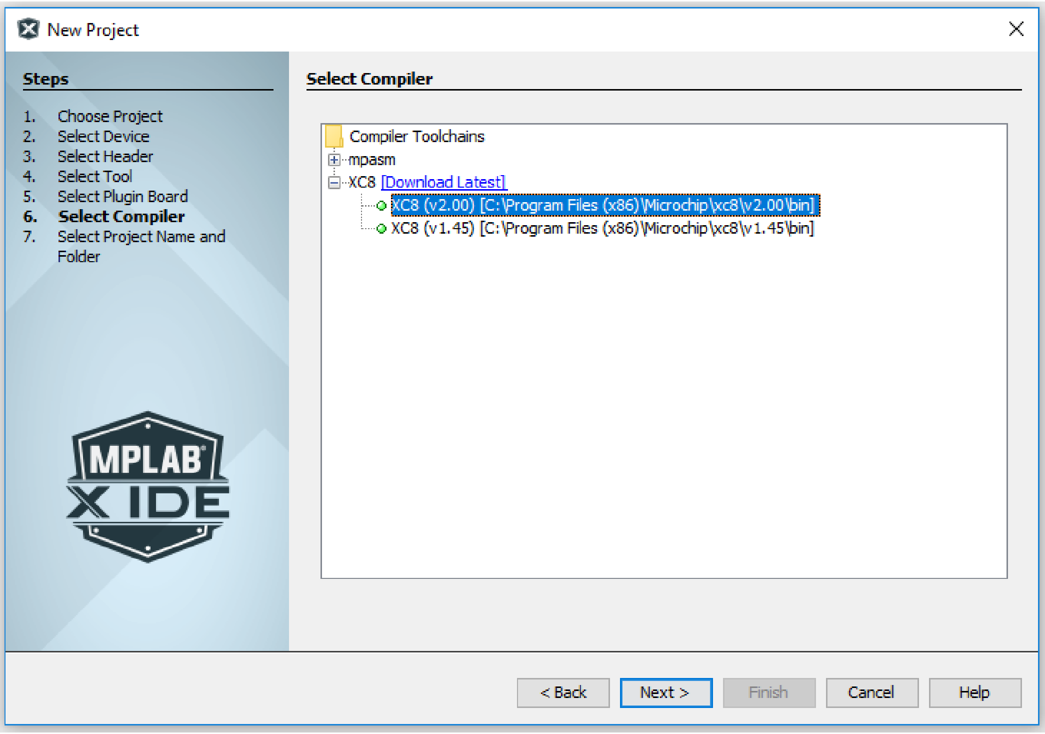


Choose the right HW settings. **Select** from the list the device on the CNANO board (**PIC16F18446**). Please make sure the PIC16F18446 is selected, and **not** the **LF** variant. From the **Tool** list we will select **Curiosity Nano.**



Choosing the compiler

From the Select Compiler list **select the latest XC8 compiler**. For this lab we use XC8 (v2.20)

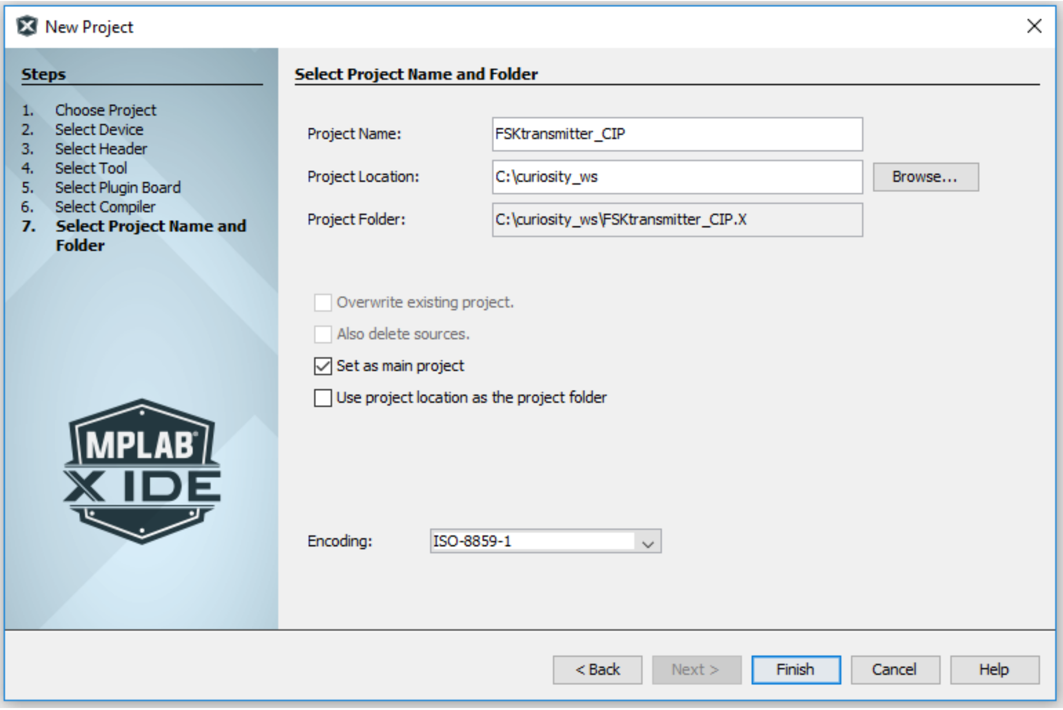


Saving the project and finalizing the settings

The **Project Name** for this lab is **FSKtransmitter_CIP**.

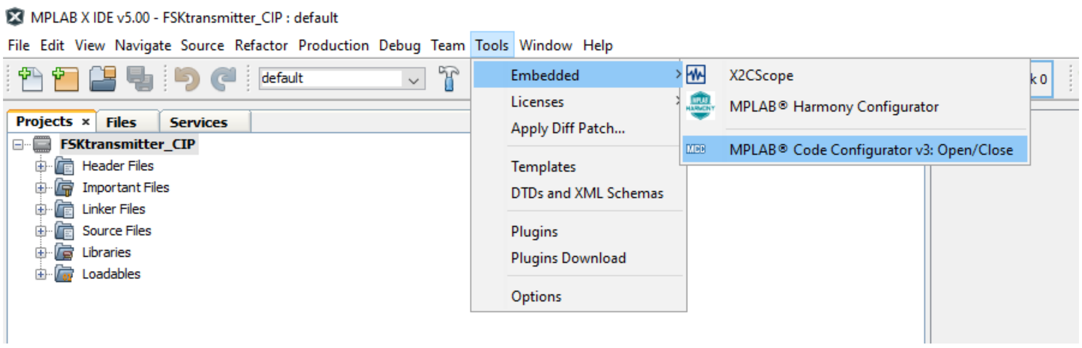
For the **Project Location** we will use **c:\curiosity_ws**

Dismiss the dialog with ***Finish***

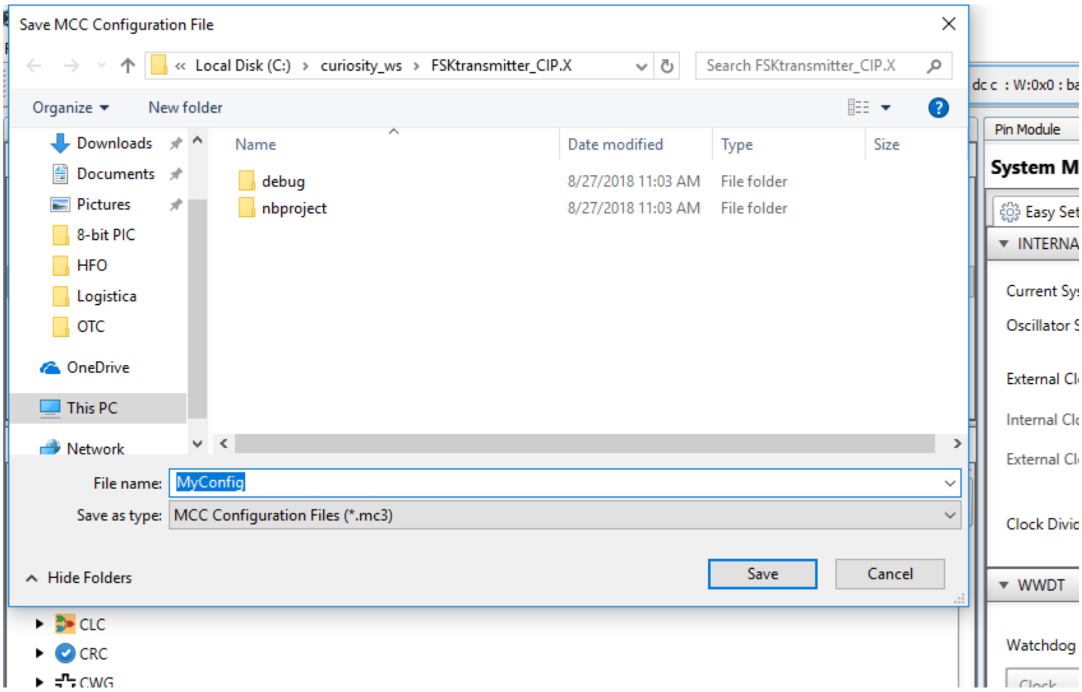


Start the MPLAB® Code Configurator tool

Select **Tools → Embedded → MPLAB® Code Configurator v4: Open/Close** from the main menu.



Save the MCC configuration file in the same project location


Save MCC Configuration File
Local Disk (C:) › curiosity_ws › FSKtransmitter_CIP.X
Search FSKtransmitter_CIP.X
Organize
New folder
Downloads
Documents
Pictures
8-bit PIC
HFO
Logistica
OTC
OneDrive
This PC
Network
Name
Date modified
Type
Size
debug
8/27/2018 11:03 AM
File folder
nbproject
8/27/2018 11:03 AM
File folder
File name: MyConfig
Save as type: MCC Configuration Files (*.mc3)
Hide Folders
Save
Cancel
CLC
CRC
CWG
Pin Module
System M
Easy Set
INTERNA
Current Sy
Oscillator S
External Cl
Internal Cl
External Cl
Clock Divic
WWDT
Watchdog

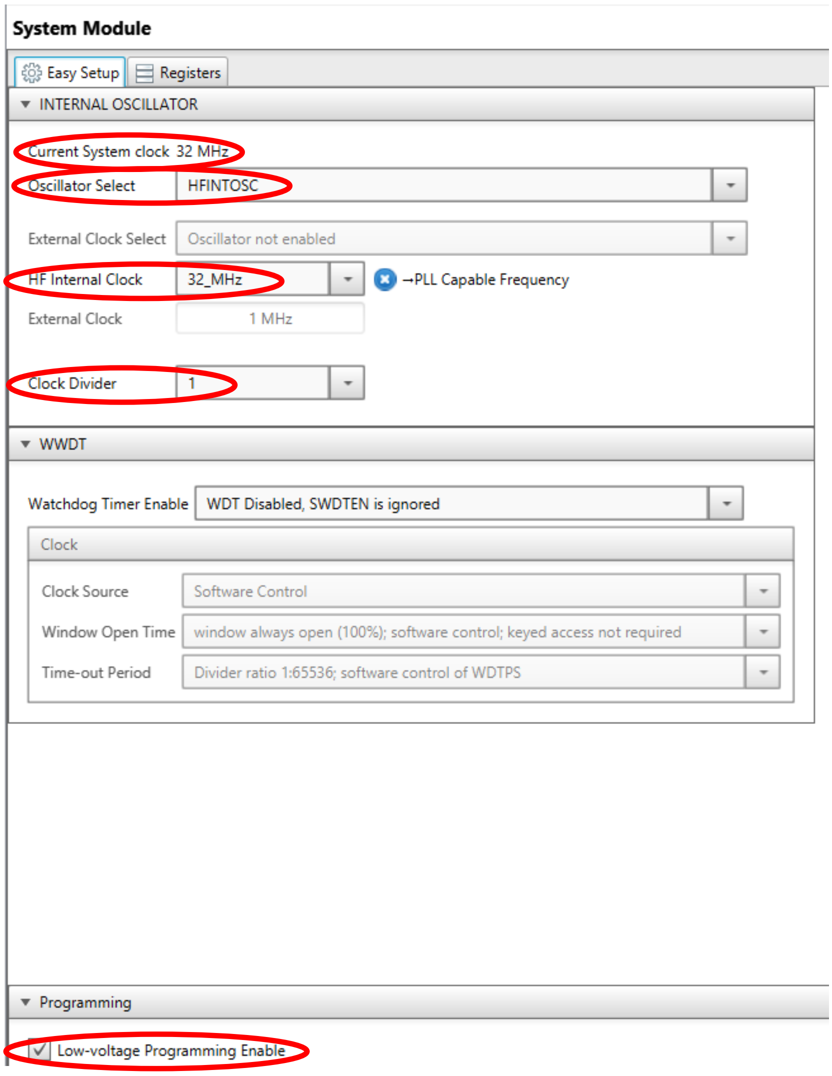
Setup the internal clock and other fuses in the System resource:

From **Project resources** tab, double click on ***System Module.***

From the **Oscillator Select** settings, make sure to select ***HFINTOSC***

HF Internal Clock should be set to 32_***MHz*** setting and the **Clock Divider** to ***1***. This setup stablishes the system clock to 32MHz for the CPU.

As the debugger on board (PKOB) uses **Low Voltage Program** method to program the MCU, ensure that **Low-voltage programming Enable** box is checked:

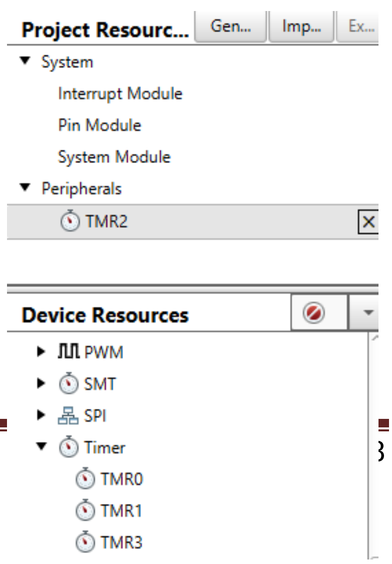


Add the TMR2 driver to the Project Resources

Select the **Resource Management [MCC]** tab.

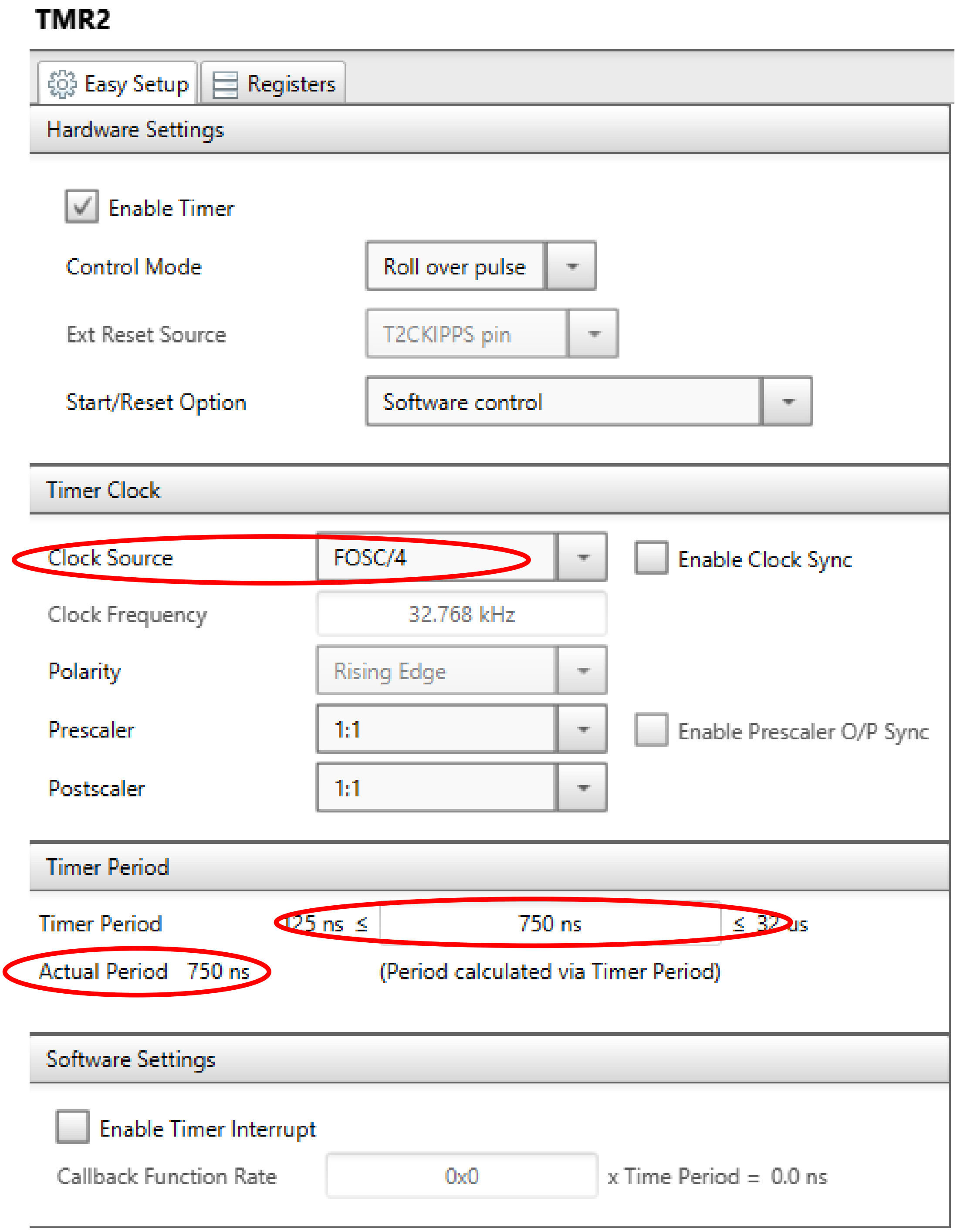
In the **Device Resources - Peripherals** area scroll-down to the **Timer** entry and expand the list. (**Hint**: You may need to click-on the '▶' symbol.)

Now double click-on the **TMR2** entry to add it to the **Project Resources** list.



Configure TMR2 for a period of 1 µs

Ensure that **TMR2** is highlighted in the **Project Resources** window.
Tick ☑ **Enable Timer.** Select **Clock Source FOSC/4, Postscaler 1:1 Prescaler 1:1**
Set **Timer period** to **750ns** with the current settings.
Control mode setting is Roll over pulse, for continuous timer operation. Make sure **Start/Reset Option** is **"Software Control"** This will inhibit hardware reset of timer.

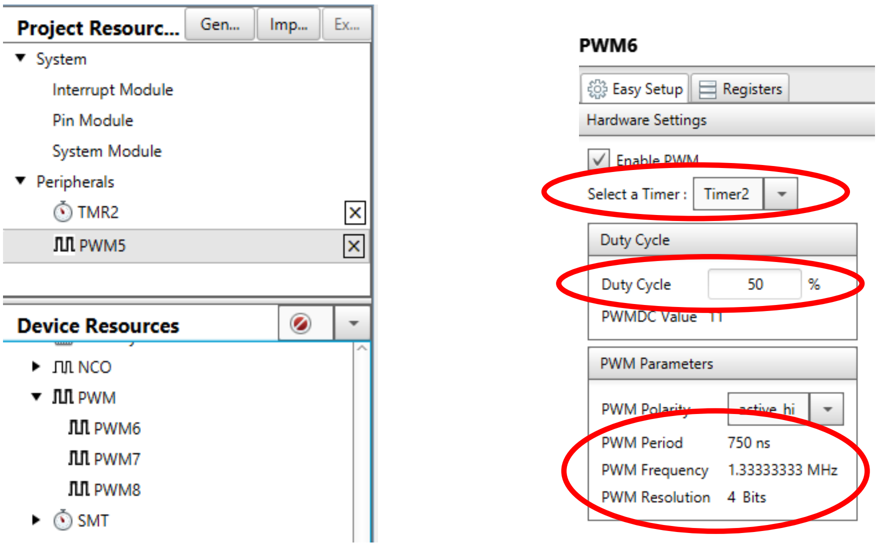


Configure PWM6 with TMR2 for the high frequency carrier signal

Add the PWM6 driver to the **Project Resources** list by double click on the **PWM6** entry, located under the **PWM** list in the **Device Resources - Peripherals**

Ensure that **PWM6** is highlighted in the **Project Resources** window and tick ☑ **Enable PWM.**

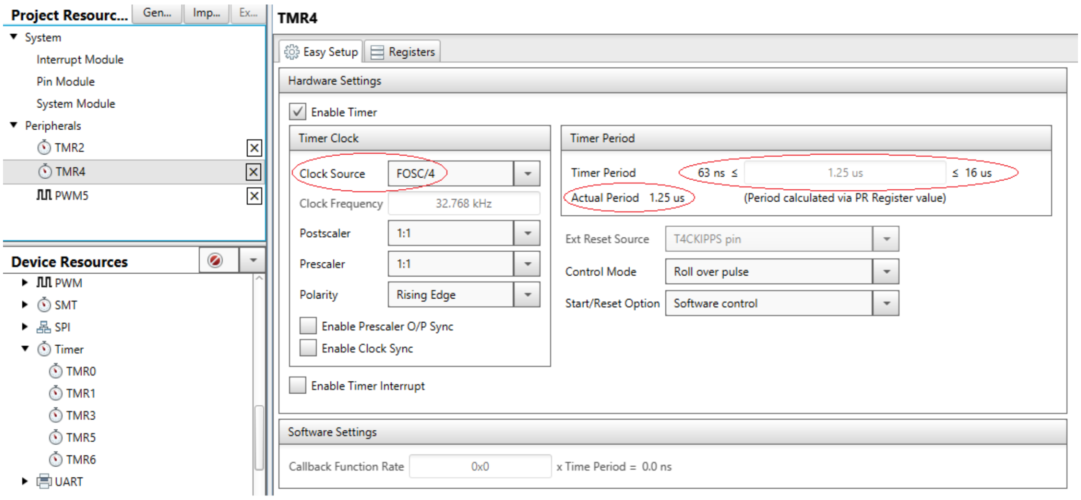
Select **Timer2** as the timer and the **Duty Cycle** to **50%.** These settings generate a square signal with the frequency stablished in Timer2 (1.33MHz).



Configure TMR4 and PWM7 for the low frequency carrier signal

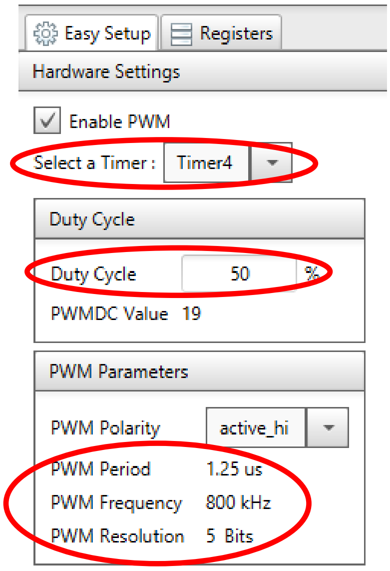
Follow the same steps as before to configure a square signal with a frequency of 800kHz.

Add and configure **TMR4** with **FOSC/4** as a **clock source** and a period of **1.25µs**



Then, add and configure PWM7 with **Timer4** as the source and the **Duty Cycle** to **50%.**

PWM7

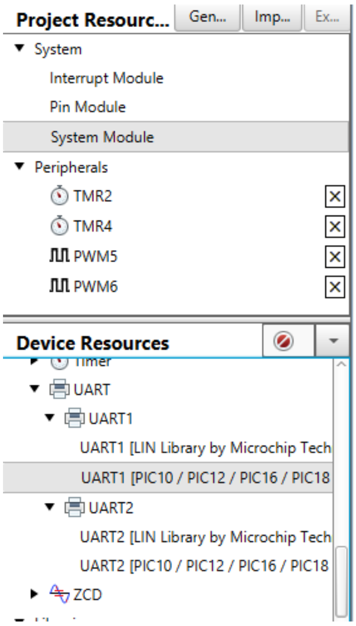


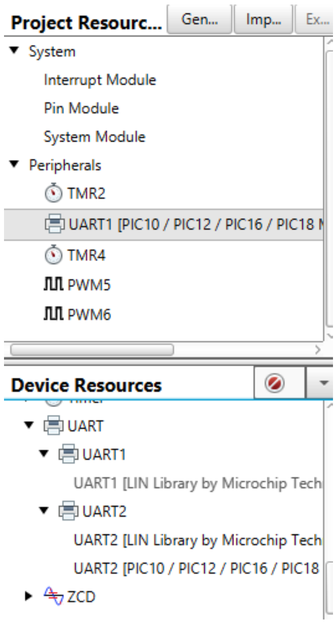
Add the ESUART1 driver to the Project Resources

Select the **Resource Management [MCC]** tab.

In the **Device Resources** area scroll-down to the **ESUART** entry and expand the list. (**Hint**: You may need to click-on the '▶' symbol.)

Now double click-on the **ESUART1** entry to add it to the **Project Resources** list.





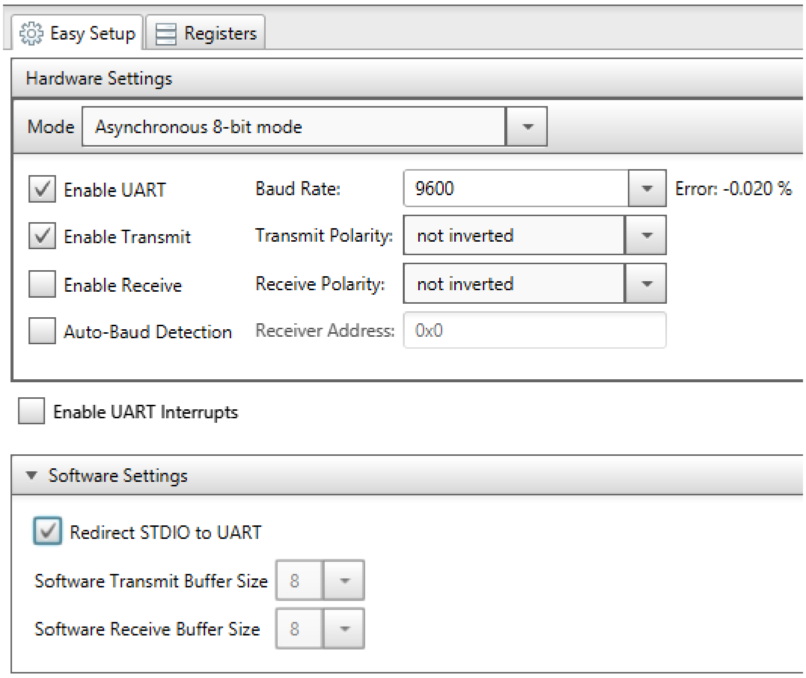
Configure ESUART1 to transmit the data stream

Ensure that **ESUART1** is highlighted in the **Project Resources** window.

Tick ☑ **Enable UART** and ☑ **Enable Transmit.** Select a **Baud Rate** of **9600**

For simplicity, tick ☑ **Redirect STDIO to UART** that will allow us using the ***printf()*** syntaxis

UART1

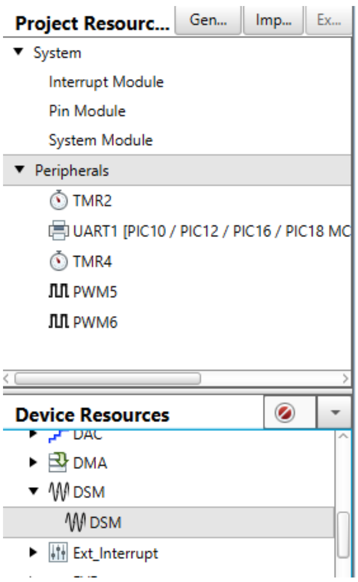


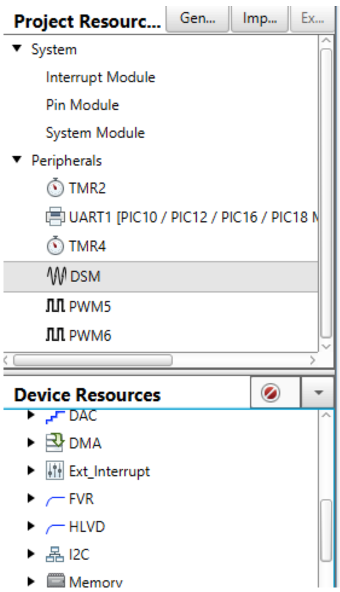
Add the Data Signal Modulator (DSM) module driver to the Project Resources

Select the **Resource Management [MCC]** tab.

In the **Device Resources** area scroll-down to the **DSM** entry and expand the list. (**Hint**: You may need to click-on the '▶' symbol.)

Now double click-on the **DSM** entry to add it to the **Project Resources** list.





Configure the DSM to modulate the data stream coming from the UART with the carrier signal composed of PWM6 (1.33MHz) and PWM7 (800kHz)

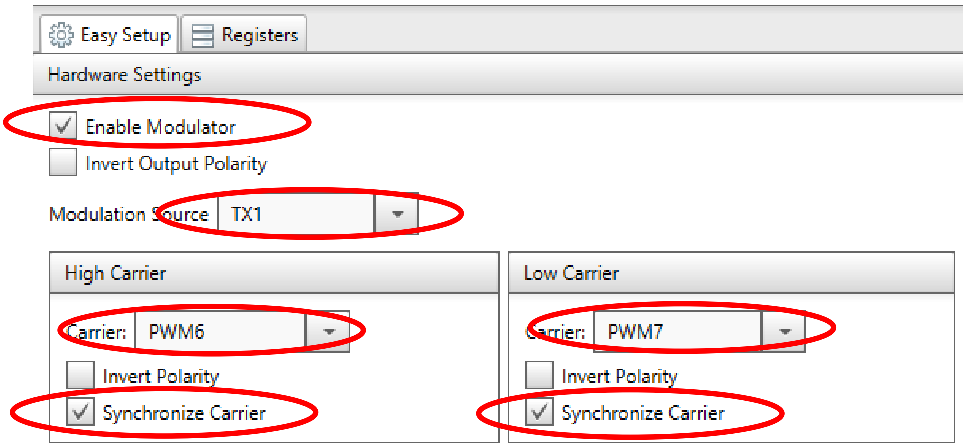
Ensure that **DSM** is highlighted in the **Project Resources** window.

Tick ☑ **Enable Modulator.** Select the TX1 as the Modulation Source.

Select **PWM6** as the **high carrier** signal and enable the **synchronization**

Select **PWM7** as the l**ow carrier** signal and enable the **synchronization**

DSM



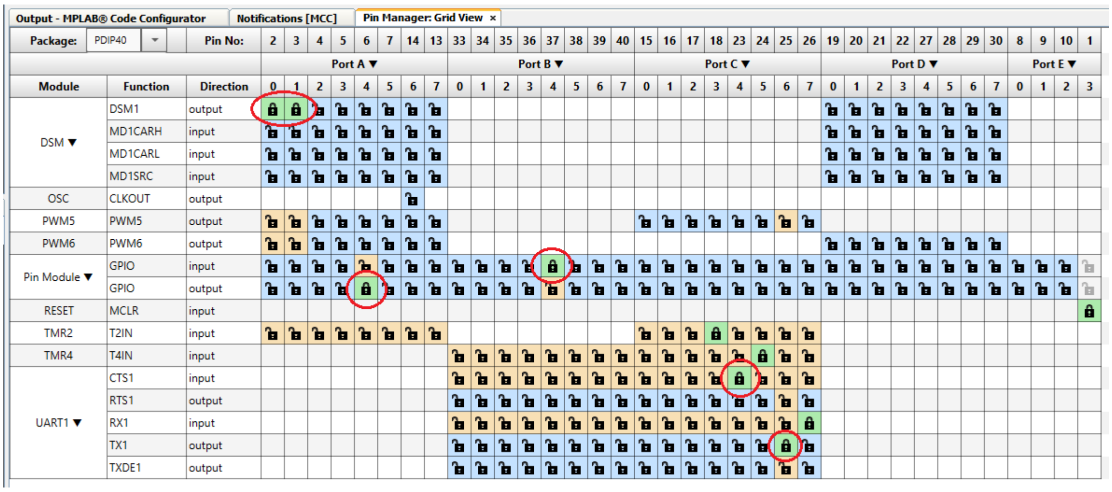
Assign the corresponding pins in the **Pin Manager Grid [MCC]** window

Select **RA0** and **RA1** for **DSM1 output**. Output in **RA1** is selected for visualizing the results.

Select the pin **RC6** for the **UART1 TX1 output**, also for visualizing the laboratory results.

We are also going to configure the Button S1 of the CNANO by selecting **RC2** as a **GPIO input**, as well as the LED D2 selecting **RA2** as a **GPIO output**.

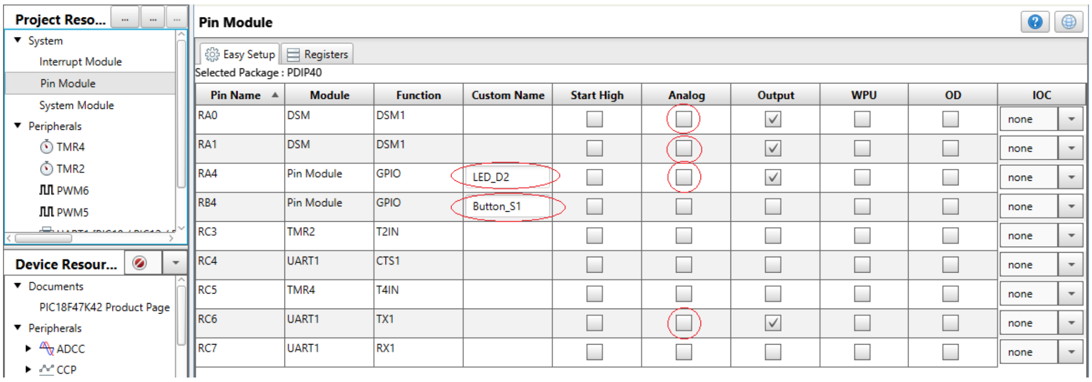
<u>**Note**</u> there are other pins selected by default. In this laboratory we are not using those pins and can be deselected or leave them as configured by default.



Configure the Pin module

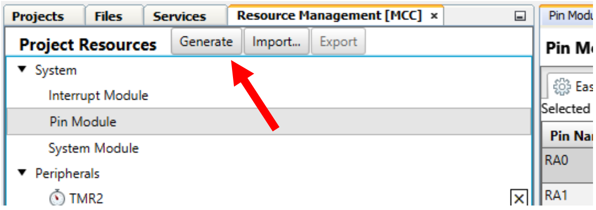
From **Project resources** tab, click on ***Pin Module.***

RA0, **RA1**, **RC6** and **RA2** need to be configured as digital outputs, tick or untick the corresponding boxes to ensure this setup. For code simplicity, change the names of **RA4** and **RC2** to **LED_D2** and **Button_S1** respectively.



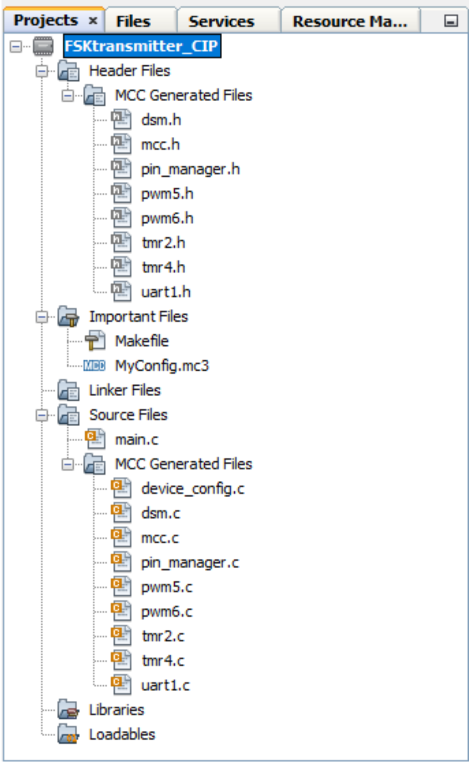
Generate the configuration code for the design

Click-on the **Generate** button in the **Resource Management [MCC]** tab.



Write the data stream to transmit

Coming back to the **Projects** tab and opening the different folders, it is possible to observe the different files generated by the MCC with the selected configuration.



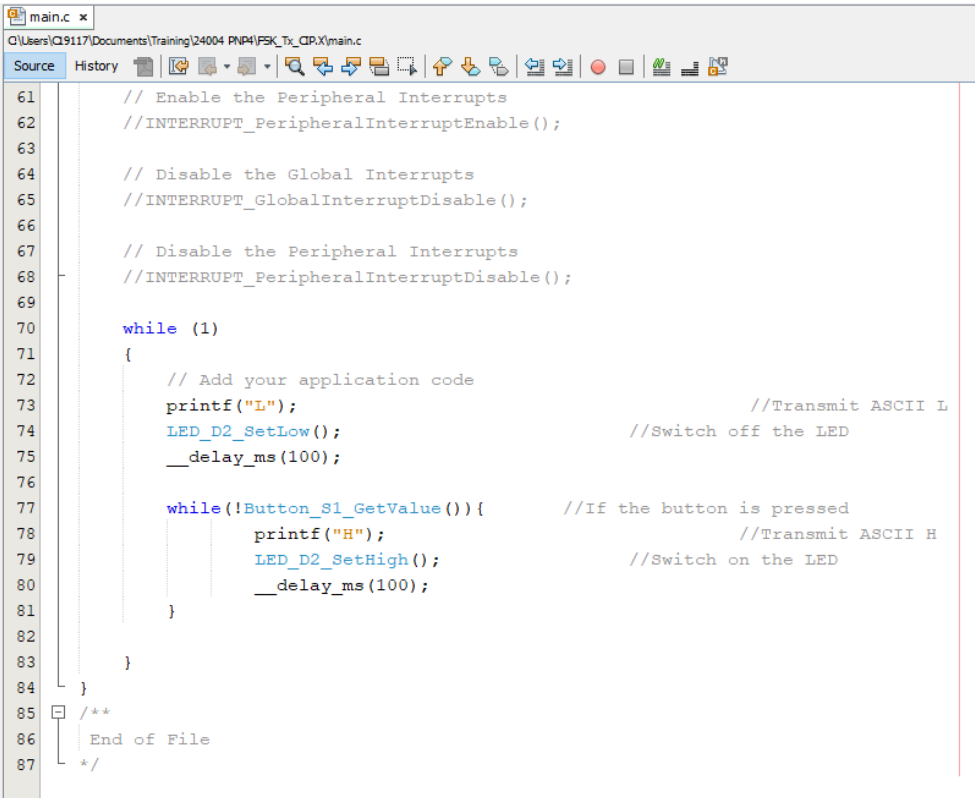
Under the ***Source Files*** folder, double click on ***main.c.***

Scroll down so the ***main()*** function is visible in the editor window.

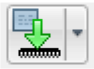
In this laboratory, the transmitter will be continuously sending the ASCII character “L” unless the button is pressed. When the button is pressed, the ASCII character “H” will be transmitted and the LED will be switched on. To obtain the desired behaviour, insert the following code inside ***while (1)***

```
printf("L");                     //Transmit ASCII L
LED_D2_SetLow();                 //Switch off the LED
__delay_ms(100);

while(!Button_S1_GetValue()){    //If the button is pressed
        printf("H");             //Transmit ASCII H
        LED_D2_SetHigh();        //Switch on the LED
        __delay_ms(100);
}
```



Make/build the design and program the target device

Select the **Make and Program Device** button or select **Run → Run Main Project** from the main menu. 

Note: If you receive a warning message about the selection of a 5V programmable voltage, simply click on the **OK** button.

Results

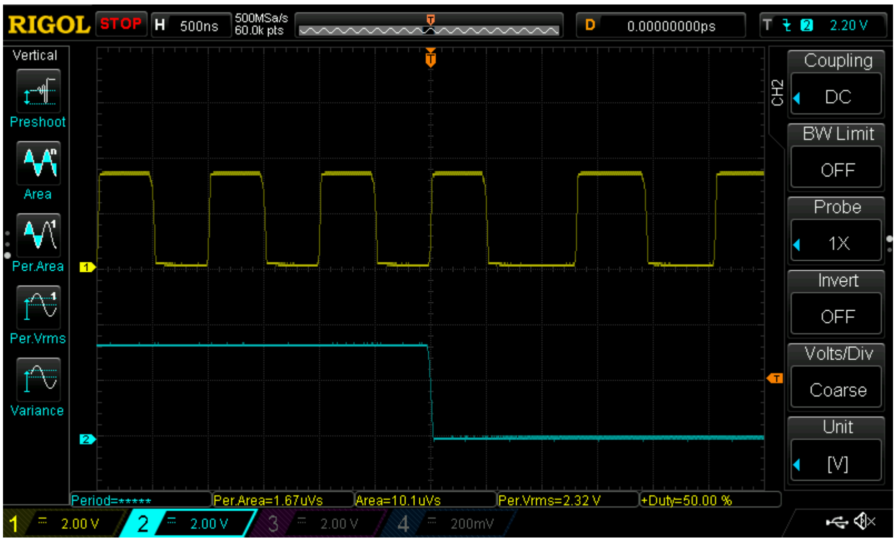
With the previous configuration, we have obtained a FSK transmitter on the pin RA1. The CPU tells the UART the data stream to be transmitted, which is modulated by the DSM with frequencies of 1.33MHz for the high state and 800kHz for the low state.

The data stream also changes depending on the button state:

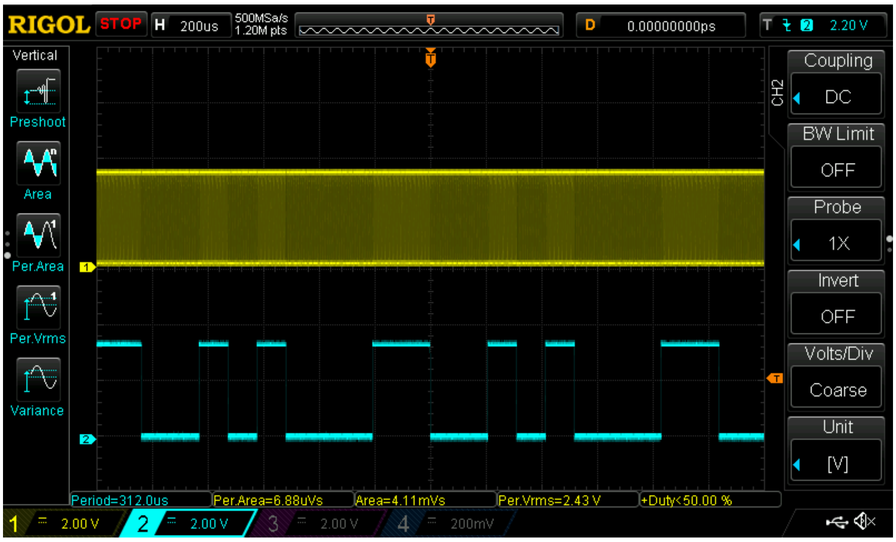
- When is not pressed, the ASCII character "L" is sent continuously
- When pressed, the ASCII character "H" instead and the LED is switched on to better visualize what character is being sent.

The following images present this behaviour.

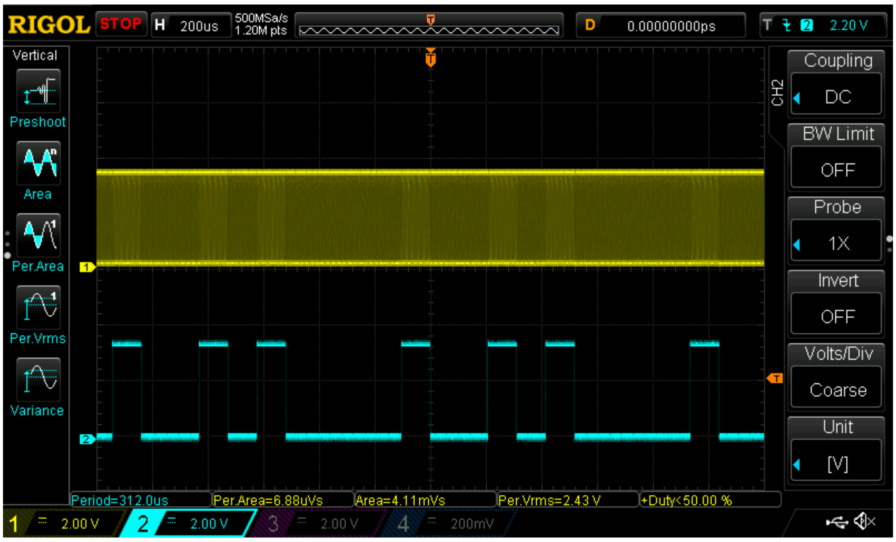
Change in the output frequency between 1.33MHz and 800kHz depending on the UART state



Button not pressed, ASCII character "L" sent. LED is OFF



Button not pressed, ASCII character “H” sent. LED is ON.



Note