MPLAB® Code Configurator & Core Independent Peripherals

Lab Manual

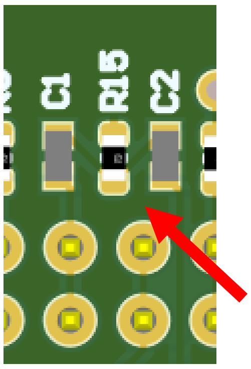
This page is intentionally left blank

Lab 1 – FSK Transmitter with CIPs

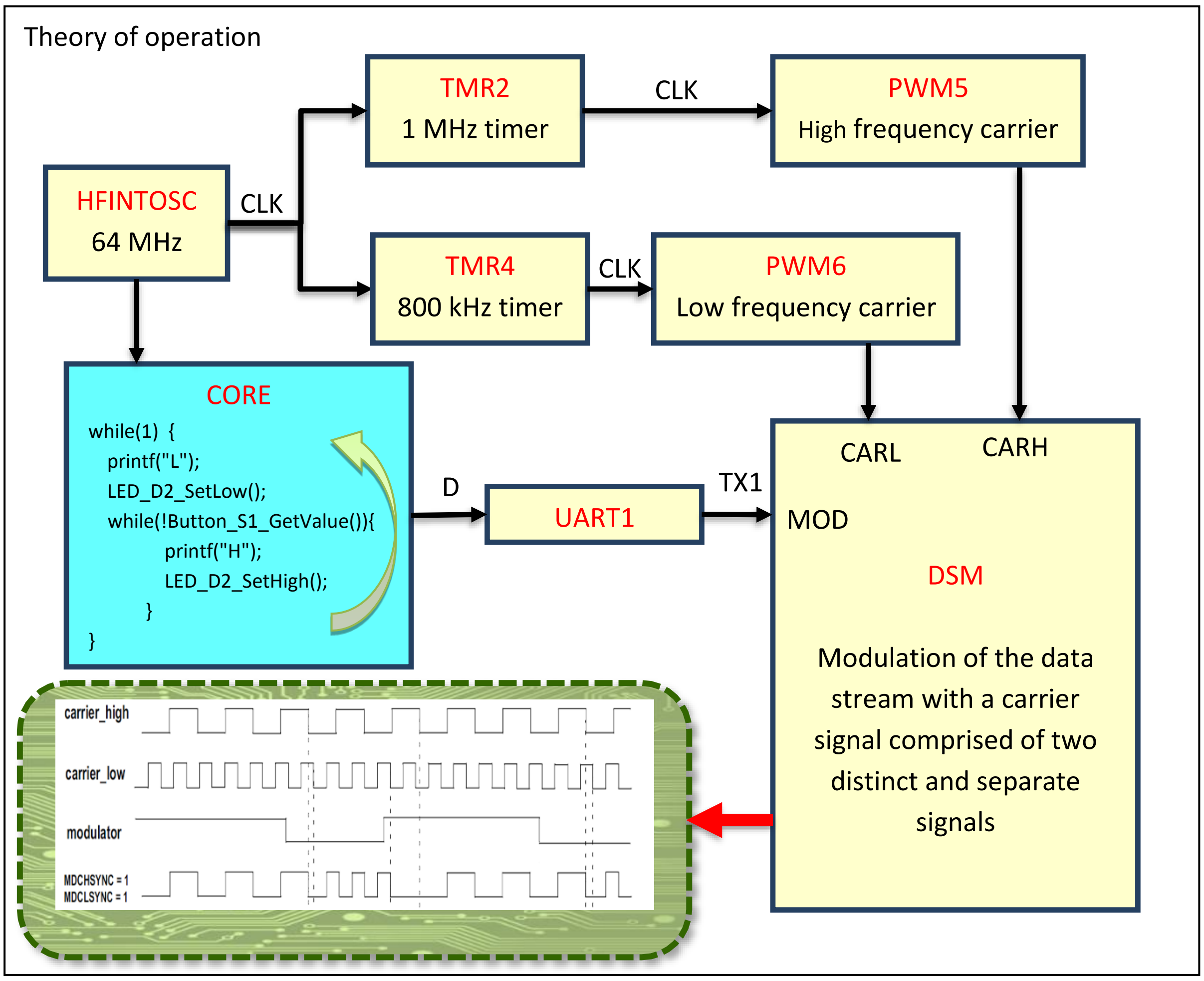
Purpose

Demonstrate how to configure an FSK transmitter using different peripherals present on the PIC18F47K42 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 (2)	Output
DSM1	RA1 (3)	Output-Analyser
TX1	RC6 (25)	Output-Analyser
Button_S1	RB4 (37)	Input
LED_D2	RA4 (6)	Output



```
while(1) {
   printf("L");
   LED_D2_SetLow();
   while(!Button_S1_GetValue()){
         printf("H");
         LED_D2_SetHigh();
      }
}
```

Procedure

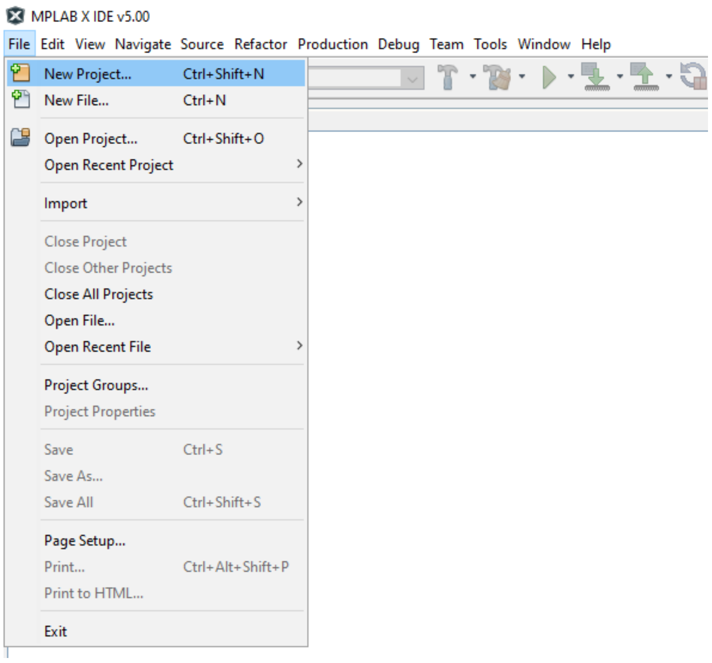
Connect the Curiosity HPC board (DM164136) to the PC.

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)

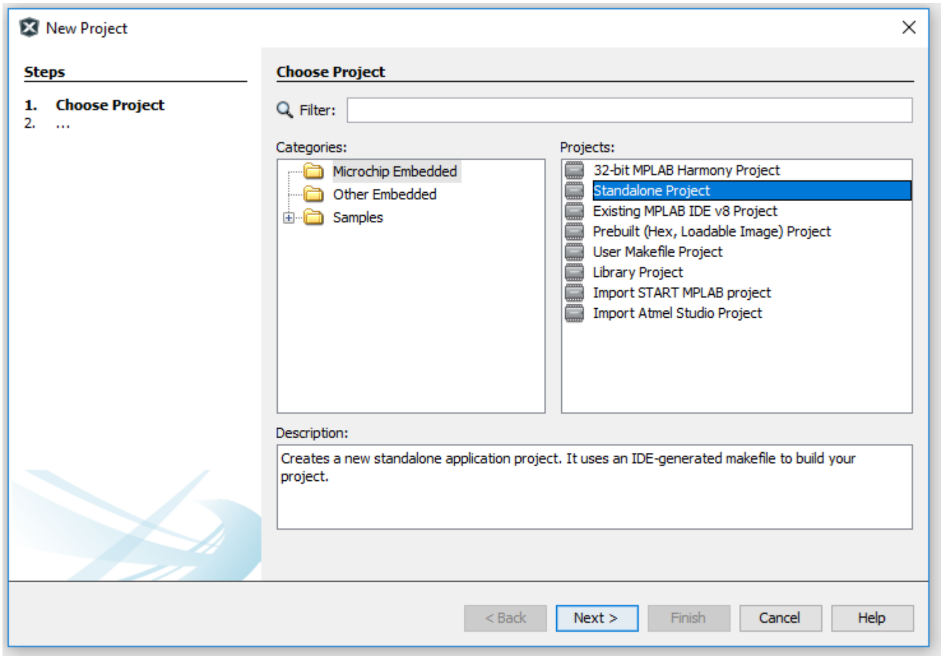
Create a new project using PIC18F47K42 on Curiosity Board HPC

<u>Create a new project in the MPLAB® X IDE:</u>

Start MPLAB X, and then go to **File → New Project…**

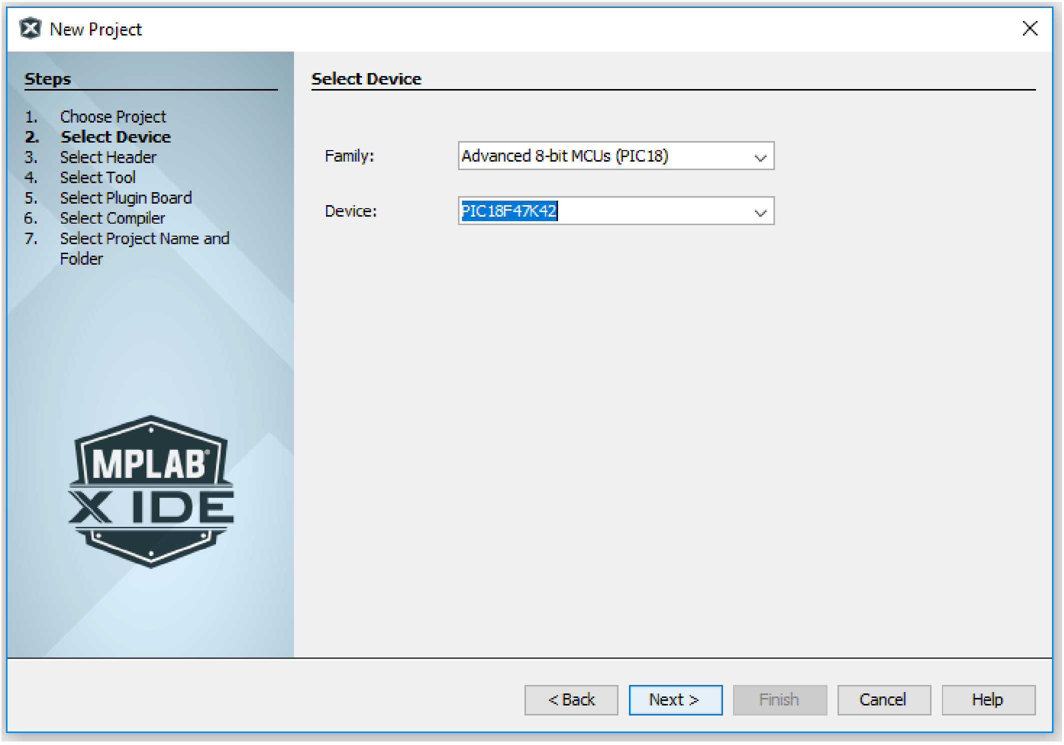


In *"New Project"* dialog navigate to **Microchip Embedded** Category and select **Standalone Project**



Creates a new standalone application project. It uses an IDE-generated makefile to build your project.

Choose the right HW settings. **Select** from the list the device on the Curiosity board (**PIC18F47K42**). Please make sure the PIC18F47K42 is selected, and **not** the **LF** variant.


New Project

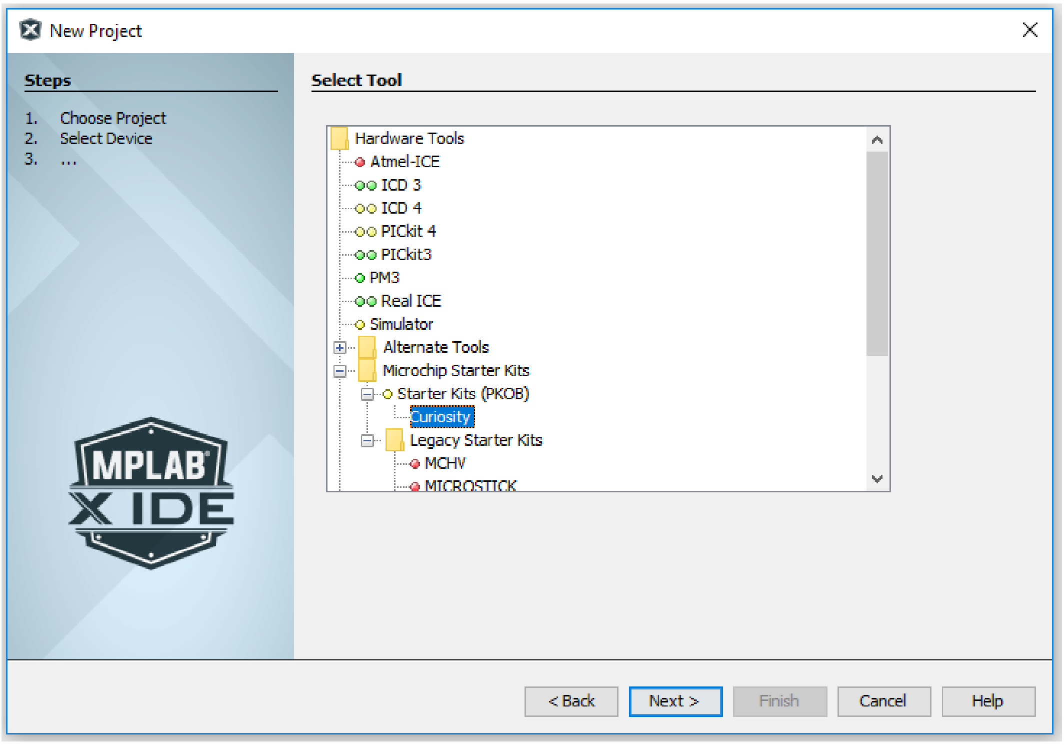
Steps
1. Choose Project
2. **Select Device**
3. Select Header
4. Select Tool
5. Select Plugin Board
6. Select Compiler
7. Select Project Name and Folder

Select Device

Family: Advanced 8-bit MCUs (PIC18)

Device: PIC18F47K42

From the **Select Tool** list we will select **Curiosity**


New Project

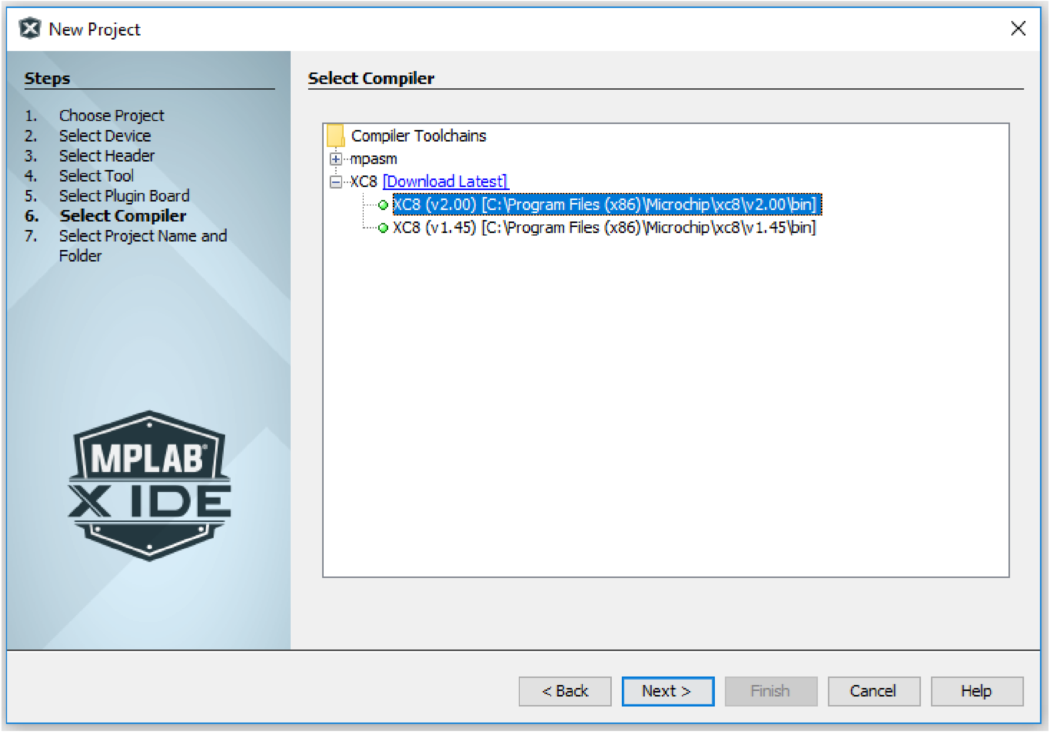
Steps
1. Choose Project
2. Select Device
3. ...

Select Tool

- Hardware Tools
  - Atmel-ICE
  - ICD 3
  - ICD 4
  - PICkit 4
  - PICkit3
  - PM3
  - Real ICE
  - Simulator
- Alternate Tools
- Microchip Starter Kits
  - Starter Kits (PKOB)
    - Curiosity
  - Legacy Starter Kits
    - MCHV
    - MICROSTICK

Choosing the compiler

From the Select Compiler list **select the latest XC8 compiler**. For this lab we use XC8 (v2.00)


New Project

Steps
1. Choose Project
2. Select Device
3. Select Header
4. Select Tool
5. Select Plugin Board
6. **Select Compiler**
7. Select Project Name and Folder

Select Compiler

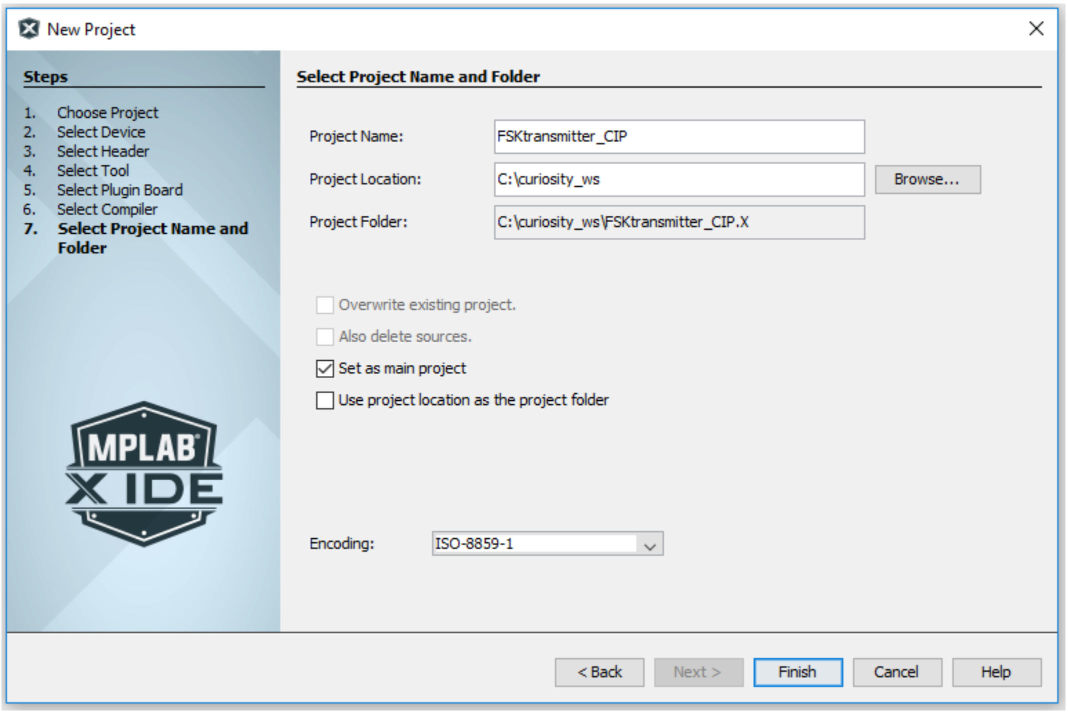
Compiler Toolchains
- mpasm
- XC8 [Download Latest]
  - XC8 (v2.00) [C:\Program Files (x86)\Microchip\xc8\v2.00\bin]
  - XC8 (v1.45) [C:\Program Files (x86)\Microchip\xc8\v1.45\bin]

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***


New Project

Steps
1. Choose Project
2. Select Device
3. Select Header
4. Select Tool
5. Select Plugin Board
6. Select Compiler
7. **Select Project Name and Folder**

Select Project Name and Folder

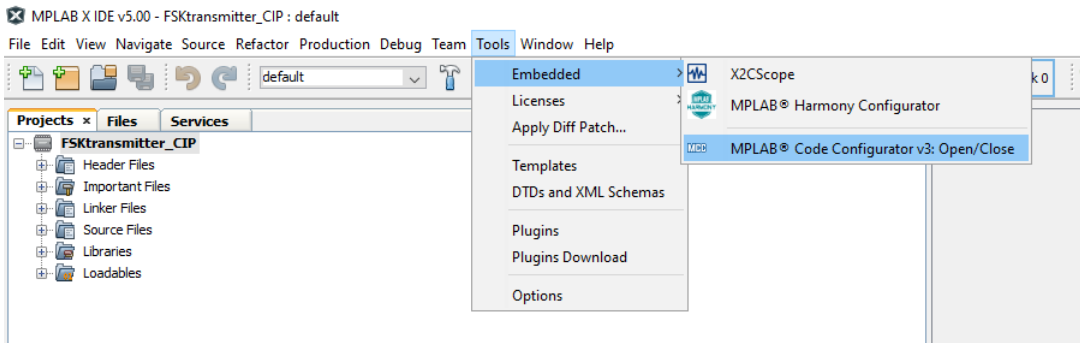
Project Name:	FSKtransmitter_CIP
Project Location:	C:\curiosity_ws
Project Folder:	C:\curiosity_ws\FSKtransmitter_CIP.X

- [ ] Overwrite existing project.
- [ ] Also delete sources.
- [x] Set as main project
- [ ] Use project location as the project folder

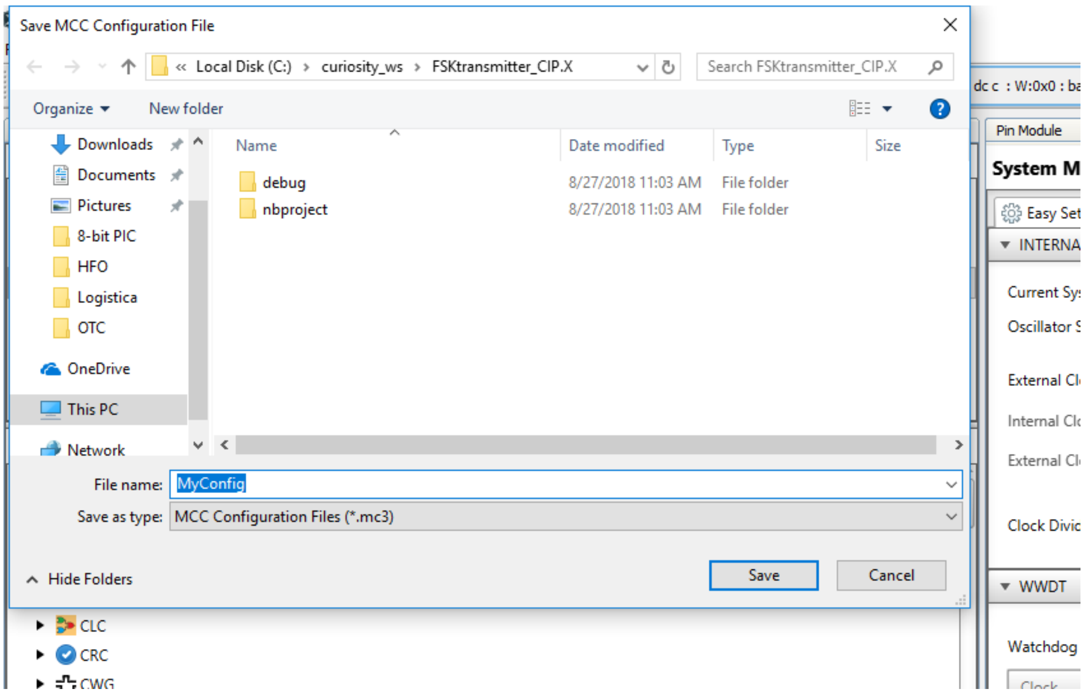
Encoding: ISO-8859-1

Start the MPLAB® Code Configurator tool

Select **Tools → Embedded → MPLAB® Code Configurator v3: Open/Close** from the main menu.



Save the MCC configuration file in the same project location



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)

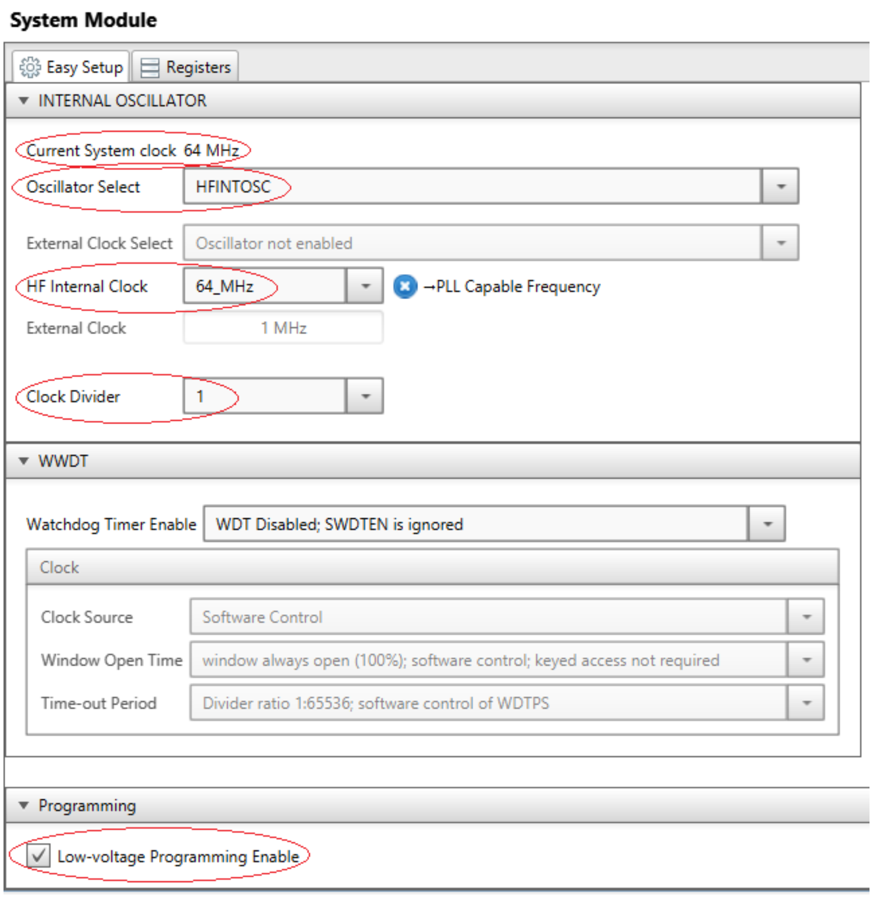
Setup the internal clock and other fuses in the System resource:

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 ***64_MHz*** setting and the **Clock Divider** to ***1***. This setup stablishes the system clock to 64MHz 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:


System Module

Easy Setup | Registers

INTERNAL OSCILLATOR

Current System clock 64 MHz

Oscillator Select: HFINTOSC

External Clock Select: Oscillator not enabled

HF Internal Clock: 64_MHz →PLL Capable Frequency

External Clock: 1 MHz

Clock Divider: 1

WWDT

Watchdog Timer Enable: WDT Disabled; SWDTEN is ignored

Clock

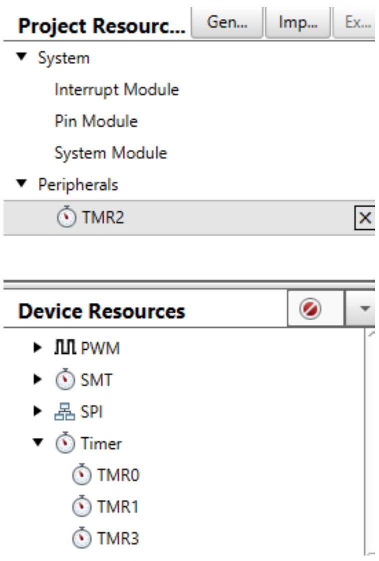
Clock Source: Software Control

Window Open Time: window always open (100%); software control; keyed access not required

Time-out Period: Divider ratio 1:65536; software control of WDTPS

Programming

[✓] Low-voltage Programming Enable



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

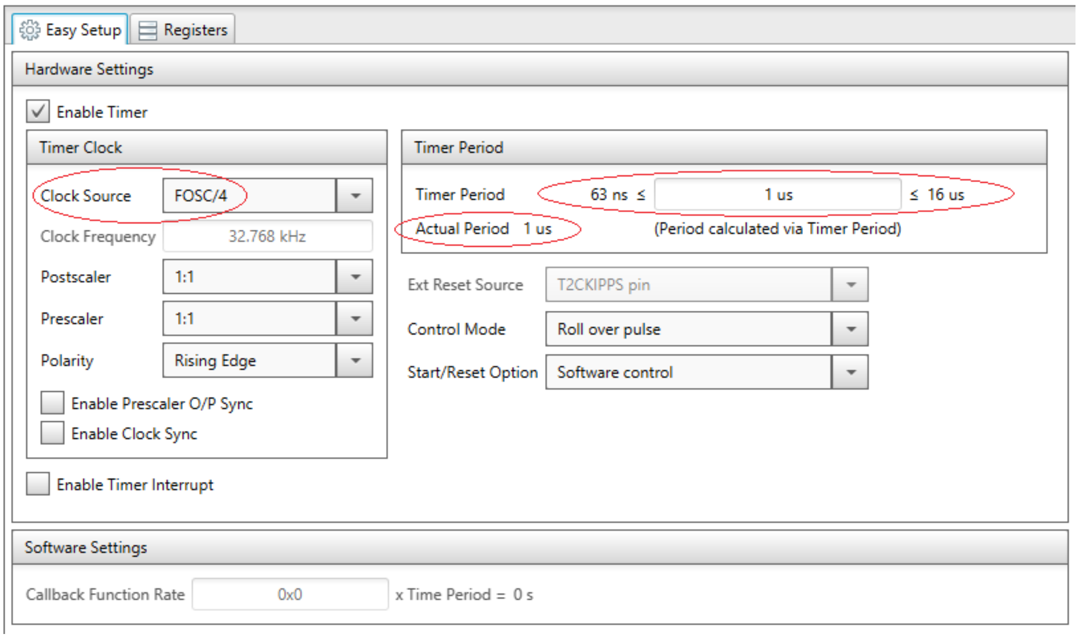
Ensure that **TMR2** is highlighted in the **Project Resources** window.

Tick ☑ **Enable Timer.** Select **Clock Source FOSC/4, Postscaler 1:1 Prescaler 1:1**

Set **Timer period** to **1µs** 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.

TMR2



☑ Enable Timer

Clock Source	FOSC/4
Clock Frequency	32.768 kHz
Postscaler	1:1
Prescaler	1:1
Polarity	Rising Edge

☐ Enable Prescaler O/P Sync
☐ Enable Clock Sync

Timer Period 63 ns ≤ 1 us ≤ 16 us
Actual Period 1 us (Period calculated via Timer Period)

Ext Reset Source	T2CKIPPS pin
Control Mode	Roll over pulse
Start/Reset Option	Software control

☐ Enable Timer Interrupt

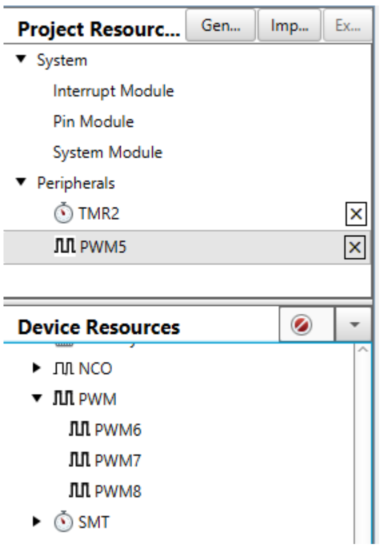
Callback Function Rate 0x0 x Time Period = 0 s

Configure PWM5 with TMR2 for the high frequency carrier signal

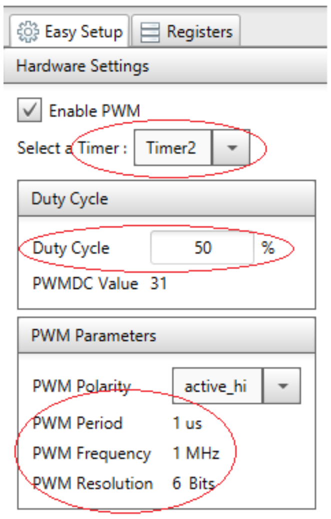
Add the PWM5 driver to the **Project Resources** list by double click on the **PWM5** entry, located under the **PWM** list in the **Device Resources - Peripherals**

Ensure that **PWM5** 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 (1MHz).



PWM5



☑ Enable PWM

Select a Timer : Timer2

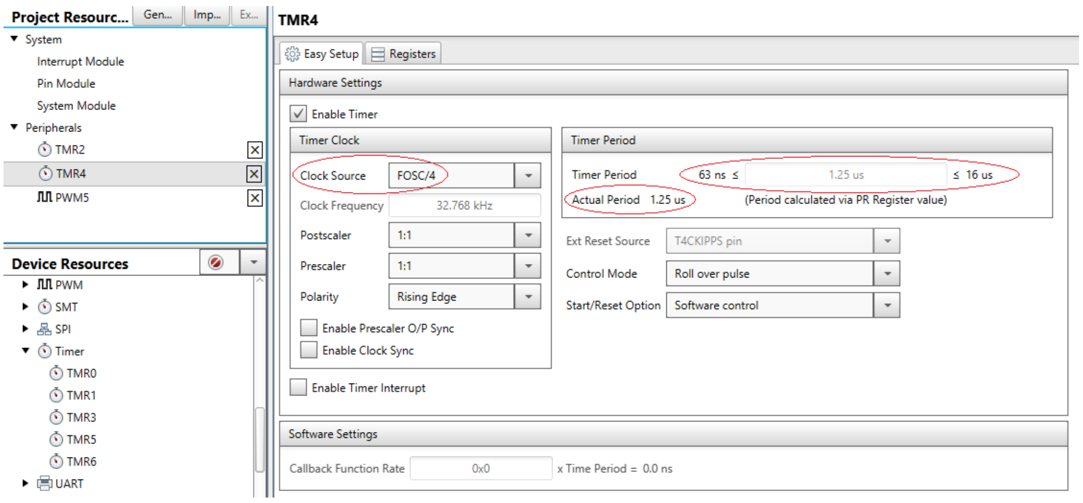
Duty Cycle	50 %
PWMDC Value	31

PWM Polarity	active_hi
PWM Period	1 us
PWM Frequency	1 MHz
PWM Resolution	6 Bits

Configure TMR4 and PWM6 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**



TMR4

Hardware Settings

☑ Enable Timer

Timer Clock

Clock Source	FOSC/4
Clock Frequency	32.768 kHz
Postscaler	1:1
Prescaler	1:1
Polarity	Rising Edge

☐ Enable Prescaler O/P Sync
☐ Enable Clock Sync

☐ Enable Timer Interrupt

Timer Period

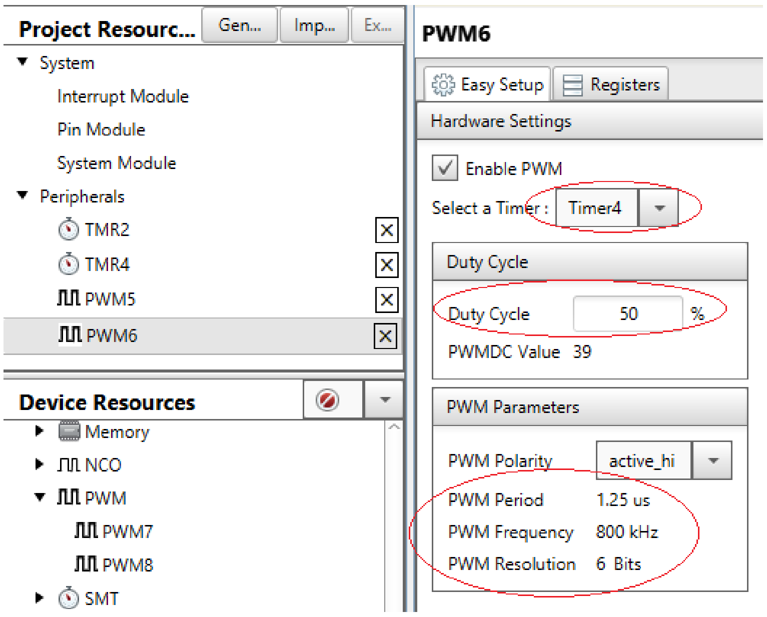
Timer Period 63 ns ≤ 1.25 us ≤ 16 us
Actual Period 1.25 us (Period calculated via PR Register value)

Ext Reset Source	T4CKIPPS pin
Control Mode	Roll over pulse
Start/Reset Option	Software control

Software Settings

Callback Function Rate 0x0 x Time Period = 0.0 ns

Then, add and configure PWM6 with **Timer4** as the source and the **Duty Cycle** to **50%.**



PWM6

Hardware Settings

☑ Enable PWM

Select a Timer : Timer4

Duty Cycle

Duty Cycle 50 %
PWMDC Value 39

PWM Parameters

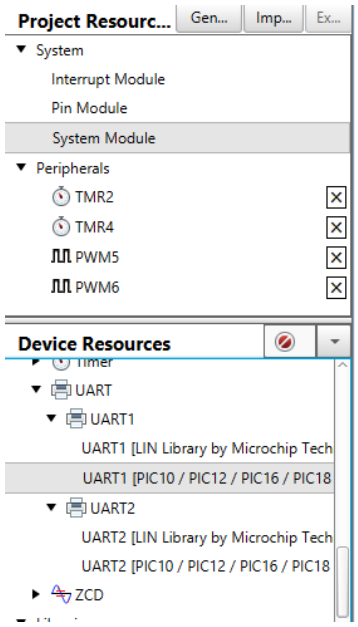
PWM Polarity	active_hi
PWM Period	1.25 us
PWM Frequency	800 kHz
PWM Resolution	6 Bits

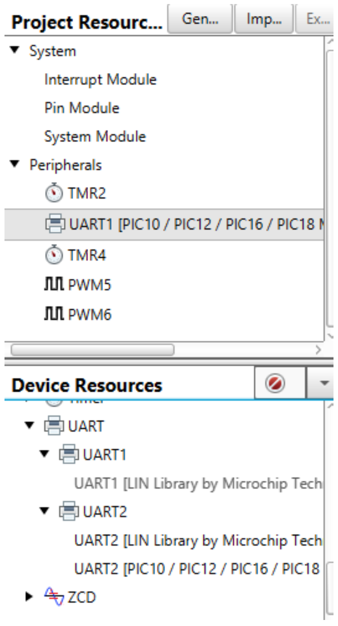
Add the UART1 driver to the Project Resources

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

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

Now double click-on the **UART1** entry to add it to the **Project Resources** list.


Project Resourc...
Gen...
Imp...
Ex...
System
Interrupt Module
Pin Module
System Module
Peripherals
TMR2
TMR4
PWM5
PWM6
Device Resources
UART
UART1
UART1 [LIN Library by Microchip Tech
UART1 [PIC10 / PIC12 / PIC16 / PIC18
UART2
UART2 [LIN Library by Microchip Tech
UART2 [PIC10 / PIC12 / PIC16 / PIC18
ZCD


Project Resourc...
Gen...
Imp...
Ex...
System
Interrupt Module
Pin Module
System Module
Peripherals
TMR2
UART1 [PIC10 / PIC12 / PIC16 / PIC18 M
TMR4
PWM5
PWM6
Device Resources
UART
UART1
UART1 [LIN Library by Microchip Tech
UART2
UART2 [LIN Library by Microchip Tech
UART2 [PIC10 / PIC12 / PIC16 / PIC18
ZCD

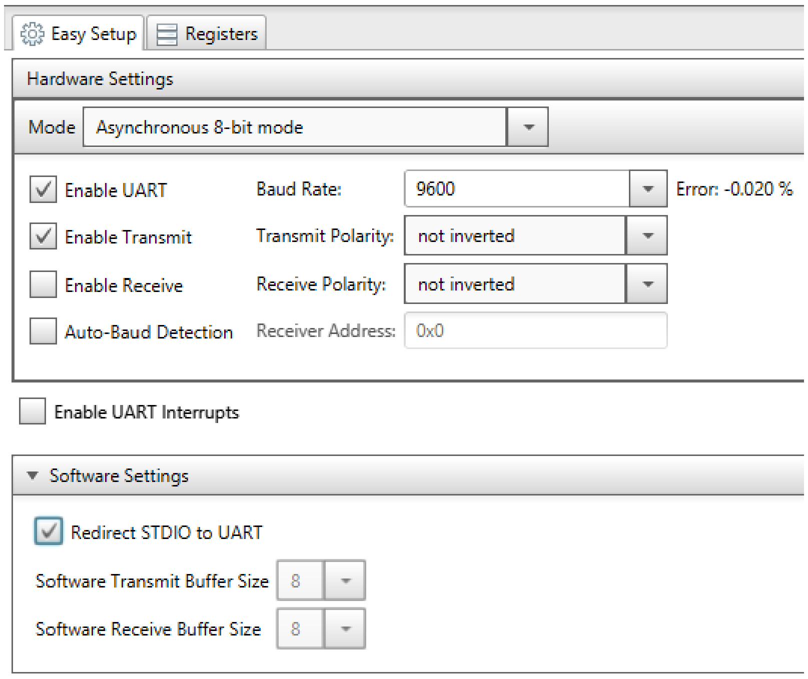
Configure UART1 to transmit the data stream

Ensure that **UART1** 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



Mode: Asynchronous 8-bit mode

☑ Enable UART	Baud Rate:	9600	Error: -0.020 %
☑ Enable Transmit	Transmit Polarity:	not inverted	
☐ Enable Receive	Receive Polarity:	not inverted	
☐ Auto-Baud Detection	Receiver Address:	0x0	

☐ Enable UART Interrupts

☑ Redirect STDIO to UART

Software Transmit Buffer Size: 8

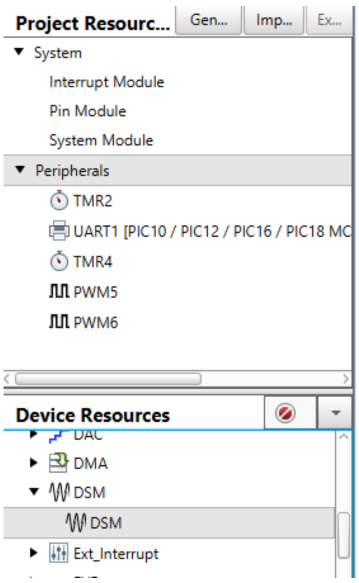
Software Receive Buffer Size: 8

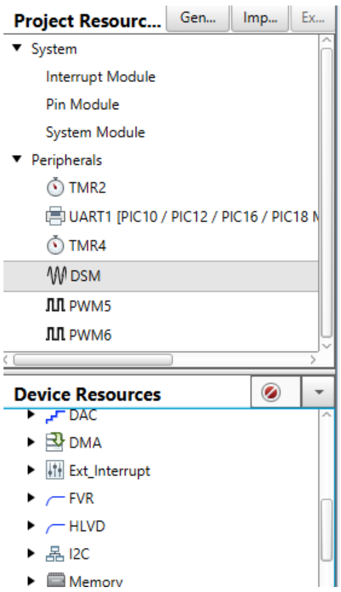
Add the Data Signal Modulator (DSM) module driver to the Project Resources

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 PWM5 (1MHz) and PWM6 (800kHz)

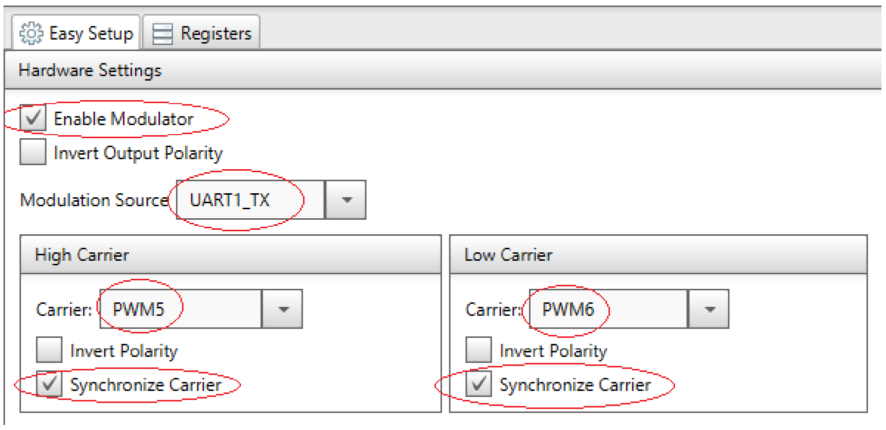
Ensure that **DSM** is highlighted in the **Project Resources** window.

Tick ☑ **Enable Modulator.** Select the UART1_TX as the Modulation Source.

Select **PWM5** as the **high carrier** signal and enable the **synchronization**

Select **PWM6** as the l**ow carrier** signal and enable the **synchronization**

DSM



Hardware Settings

☑ Enable Modulator

☐ Invert Output Polarity

Modulation Source: UART1_TX

High Carrier

Carrier: PWM5

☐ Invert Polarity

☑ Synchronize Carrier

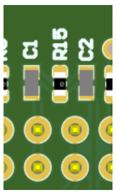
Low Carrier

Carrier: PWM6

☐ Invert Polarity

☑ Synchronize Carrier

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



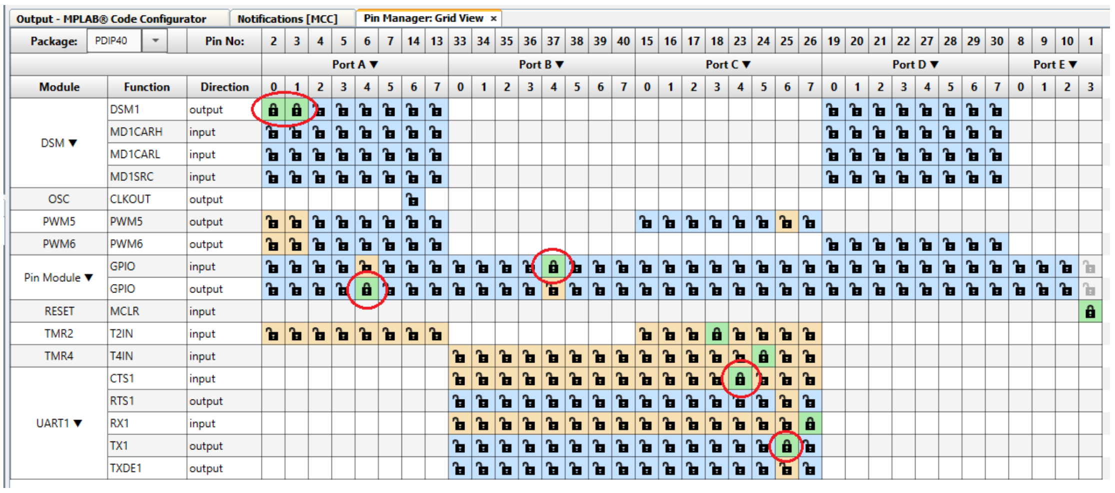
Select **RA0** and **RA1** for **DSM1 output**. **RA0** connects to the CIP board, transmitting the modulated signal trough a low pass filter composed of the capacitor C2 and the resistance R15. Output in **RA1** is selected for visualizing the results.

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

UART CTS1 needs to be changed to another available pin such as **RC4** as it needs to be configured as an input.

We are also going to configure the Button S1 of the Curiosity by selecting RB4 as a **GPIO input**, as well as the LED D2 selecting **RA4** 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.

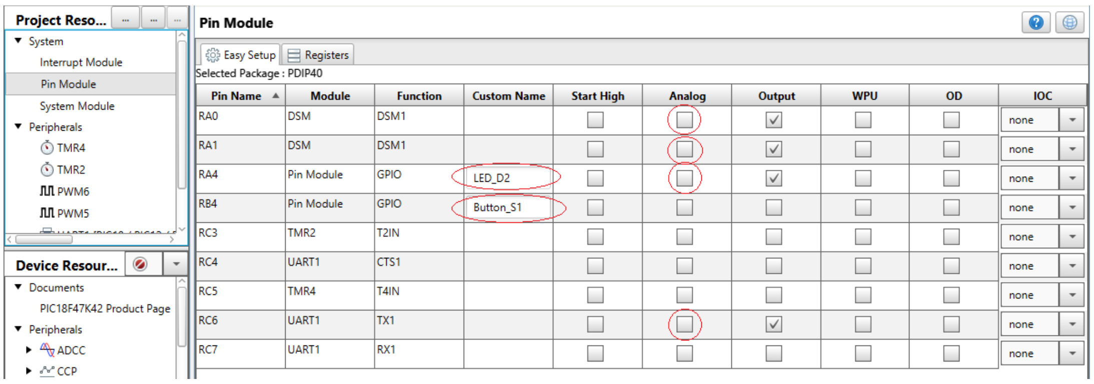


Module	Function	Direction
DSM ▼	DSM1	output
	MD1CARH	input
	MD1CARL	input
	MD1SRC	input
OSC	CLKOUT	output
PWM5	PWM5	output
PWM6	PWM6	output
Pin Module ▼	GPIO	input
	GPIO	output
RESET	MCLR	input
TMR2	T2IN	input
TMR4	T4IN	input
UART1 ▼	CTS1	input
	RTS1	output
	RX1	input
	TX1	output
	TXDE1	output

Configure the Pin module

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

RA0, **RA1**, **RC6** and **RA4** 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 **RB4** to **LED_D2** and **Button_S1** respectively.

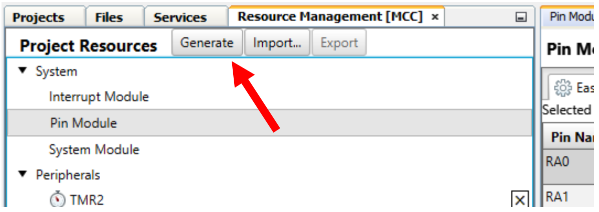


Selected Package : PDIP40

Pin Name ▲	Module	Function	Custom Name	Start High	Analog	Output	WPU	OD	IOC
RA0	DSM	DSM1		☐	☐	☑	☐	☐	none
RA1	DSM	DSM1		☐	☐	☑	☐	☐	none
RA4	Pin Module	GPIO	LED_D2	☐	☐	☑	☐	☐	none
RB4	Pin Module	GPIO	Button_S1	☐	☐	☐	☐	☐	none
RC3	TMR2	T2IN		☐	☐	☐	☐	☐	none
RC4	UART1	CTS1		☐	☐	☐	☐	☐	none
RC5	TMR4	T4IN		☐	☐	☐	☐	☐	none
RC6	UART1	TX1		☐	☐	☑	☐	☐	none
RC7	UART1	RX1		☐	☐	☐	☐	☐	none

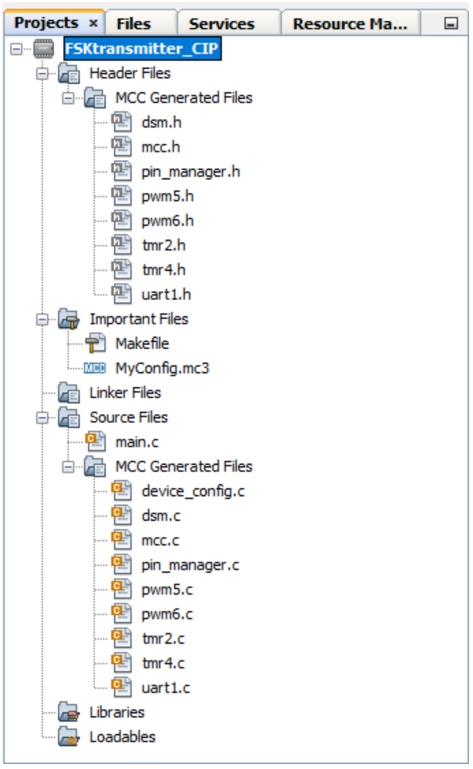
Generate the configuration code for the design

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.



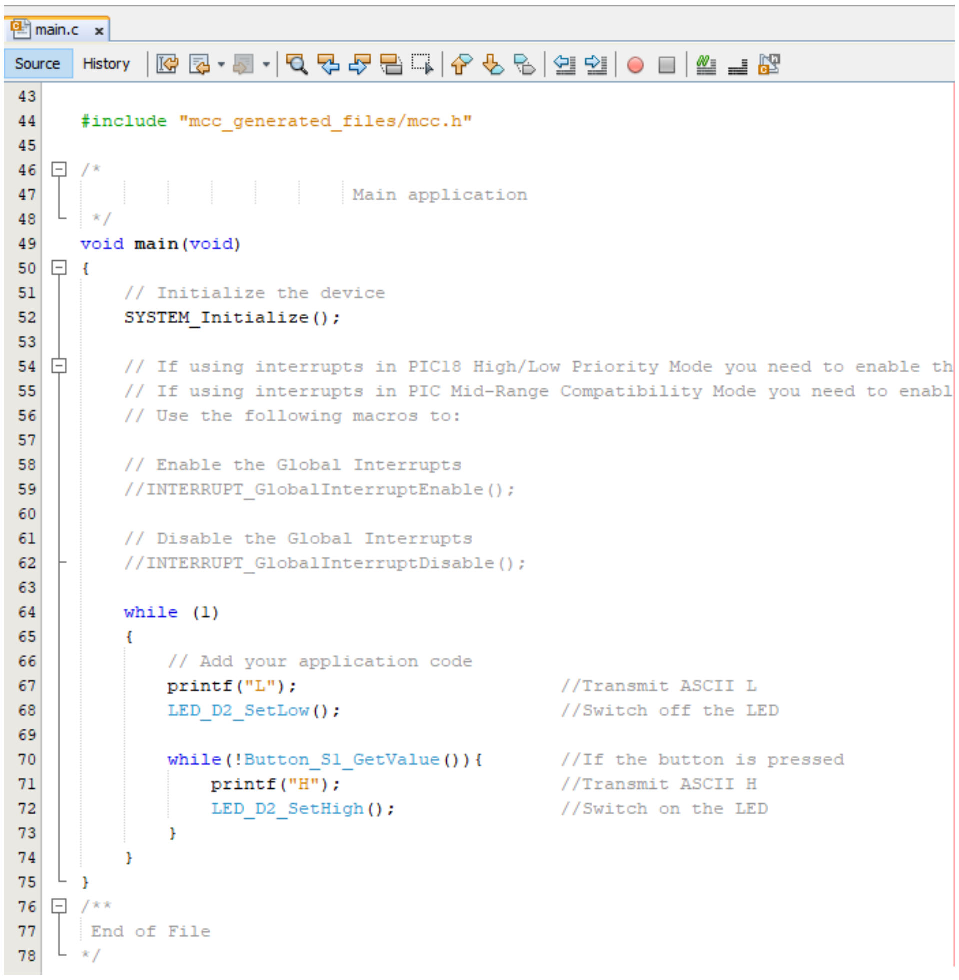
Under the ***Source Files*** folder, double click on ***main.c.***

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

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



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

/*
                         Main application
 */
void main(void)
{
    // Initialize the device
    SYSTEM_Initialize();

    // If using interrupts in PIC18 High/Low Priority Mode you need to enable th
    // If using interrupts in PIC Mid-Range Compatibility Mode you need to enabl
    // Use the following macros to:

    // Enable the Global Interrupts
    //INTERRUPT_GlobalInterruptEnable();

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

    while (1)
    {
        // Add your application code
        printf("L");                        //Transmit ASCII L
        LED_D2_SetLow();                    //Switch off the LED

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

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

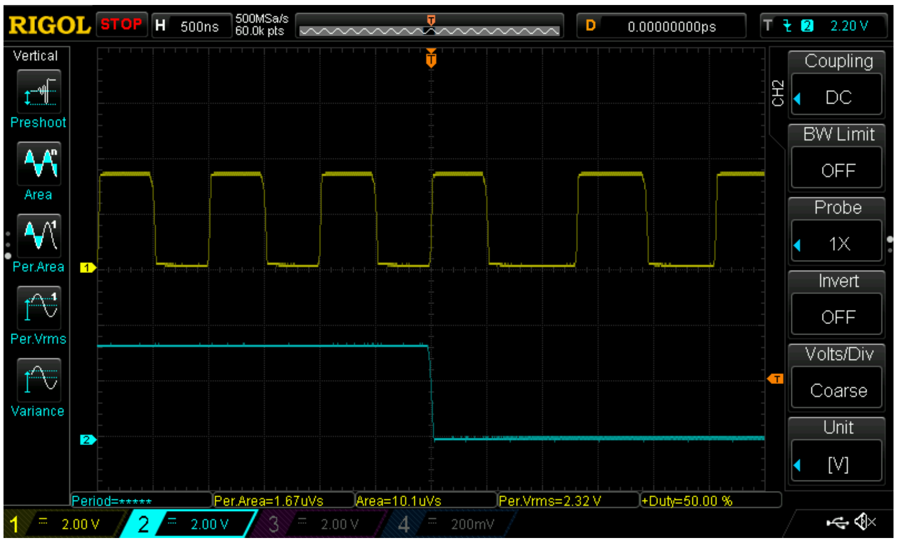
With the previous configuration, we have obtained a FSK transmitter in the pin RA0, connected to a low pass filter. The CPU tells the UART the data stream to be transmitted, which is modulated by the DSM with frequencies of 1MHz for the high state and 800kHz for the low state.

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 1MHz and 800kHz depending on the UART state


RIGOL STOP H 500ns 500MSa/s 60.0k pts D 0.00000000ps T 2 2.20 V

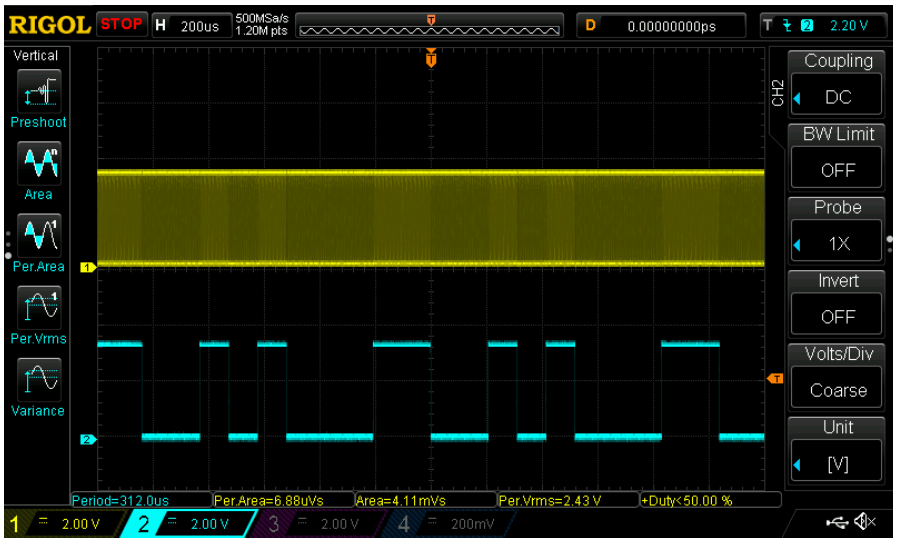
Vertical: Preshoot, Area, Per.Area, Per.Vrms, Variance

CH2: Coupling DC, BW Limit OFF, Probe 1X, Invert OFF, Volts/Div Coarse, Unit [V]

Period=***** Per.Area=1.67uVs Area=10.1uVs Per.Vrms=2.32 V +Duty=50.00 %

1 2.00 V 2 2.00 V 3 2.00 V 4 200mV

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


RIGOL STOP H 200us 500MSa/s 1.20M pts D 0.00000000ps T 2 2.20 V

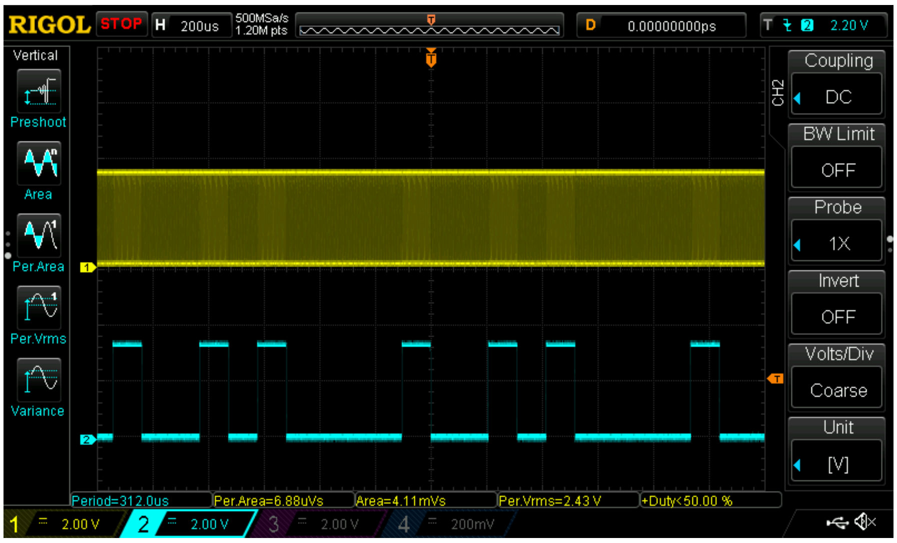
Vertical: Preshoot, Area, Per.Area, Per.Vrms, Variance

CH2: Coupling DC, BW Limit OFF, Probe 1X, Invert OFF, Volts/Div Coarse, Unit [V]

Period=312.0us Per.Area=6.88uVs Area=4.11mVs Per.Vrms=2.43 V +Duty<50.00 %

1 2.00 V 2 2.00 V 3 2.00 V 4 200mV

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


RIGOL STOP H 200us 500MSa/s 1.20M pts D 0.00000000ps T 2 2.20 V

Vertical: Preshoot, Area, Per.Area, Per.Vrms, Variance

CH2: Coupling DC, BW Limit OFF, Probe 1X, Invert OFF, Volts/Div Coarse, Unit [V]

Period=312.0us Per.Area=6.88uVs Area=4.11mVs Per.Vrms=2.43 V +Duty<50.00 %

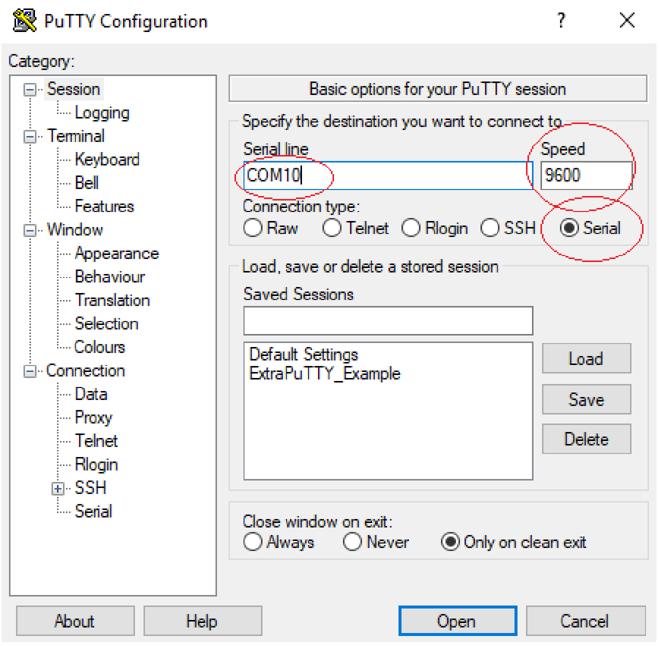
1 2.00 V 2 2.00 V 3 2.00 V 4 200mV

Finally, we are going to use the Click Analyser to observe the original signal and the corresponding modulated signal in the transmission. The device will be connected to the **MikroBUS 1** socket in the Curiosity HPC.

A free terminal emulator called **PuTTY** will be used, you can find this software online by downloading it from the official **PuTTY** website.

Install and open **PuTTY**

Set the ***Connection type*** field to serial and set your **COM port**, for example in my machine is connected to COM port 10. Set your **baud rate** to **9600**.



PuTTY Configuration

Category: Session, Logging, Terminal, Keyboard, Bell, Features, Window, Appearance, Behaviour, Translation, Selection, Colours, Connection, Data, Proxy, Telnet, Rlogin, SSH, Serial

Basic options for your PuTTY session

Specify the destination you want to connect to

Serial line: COM10

Speed: 9600

Connection type: Raw / Telnet / Rlogin / SSH / [✓ Serial]

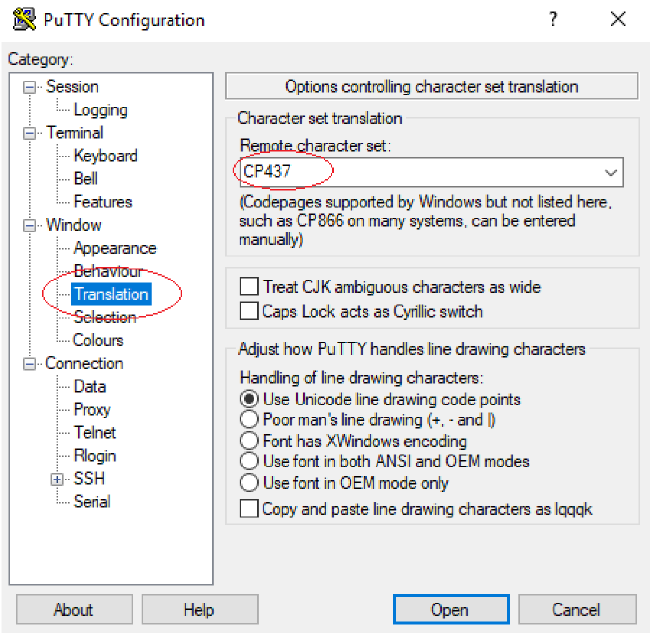
Load, save or delete a stored session

Saved Sessions

Default Settings
ExtraPuTTY_Example

Close window on exit: Always / Never / [✓ Only on clean exit]

Navigate the ***Category*** pane on the left and find a tab labelled ***Translation***. Click on it and change the ***Remote character set*** to **CP437** as the analyser board sends all information in this format. Once these settings have been changed, open the terminal.



PuTTY Configuration

Category: Session, Logging, Terminal, Keyboard, Bell, Features, Window, Appearance, Behaviour, Translation, Selection, Colours, Connection, Data, Proxy, Telnet, Rlogin, SSH, Serial

Options controlling character set translation

Character set translation

Remote character set: CP437

(Codepages supported by Windows but not listed here, such as CP866 on many systems, can be entered manually)

☐ Treat CJK ambiguous characters as wide
☐ Caps Lock acts as Cyrillic switch

Adjust how PuTTY handles line drawing characters

Handling of line drawing characters:
[✓ Use Unicode line drawing code points]
Poor man's line drawing (+, - and |)
Font has XWindows encoding
Use font in both ANSI and OEM modes
Use font in OEM mode only

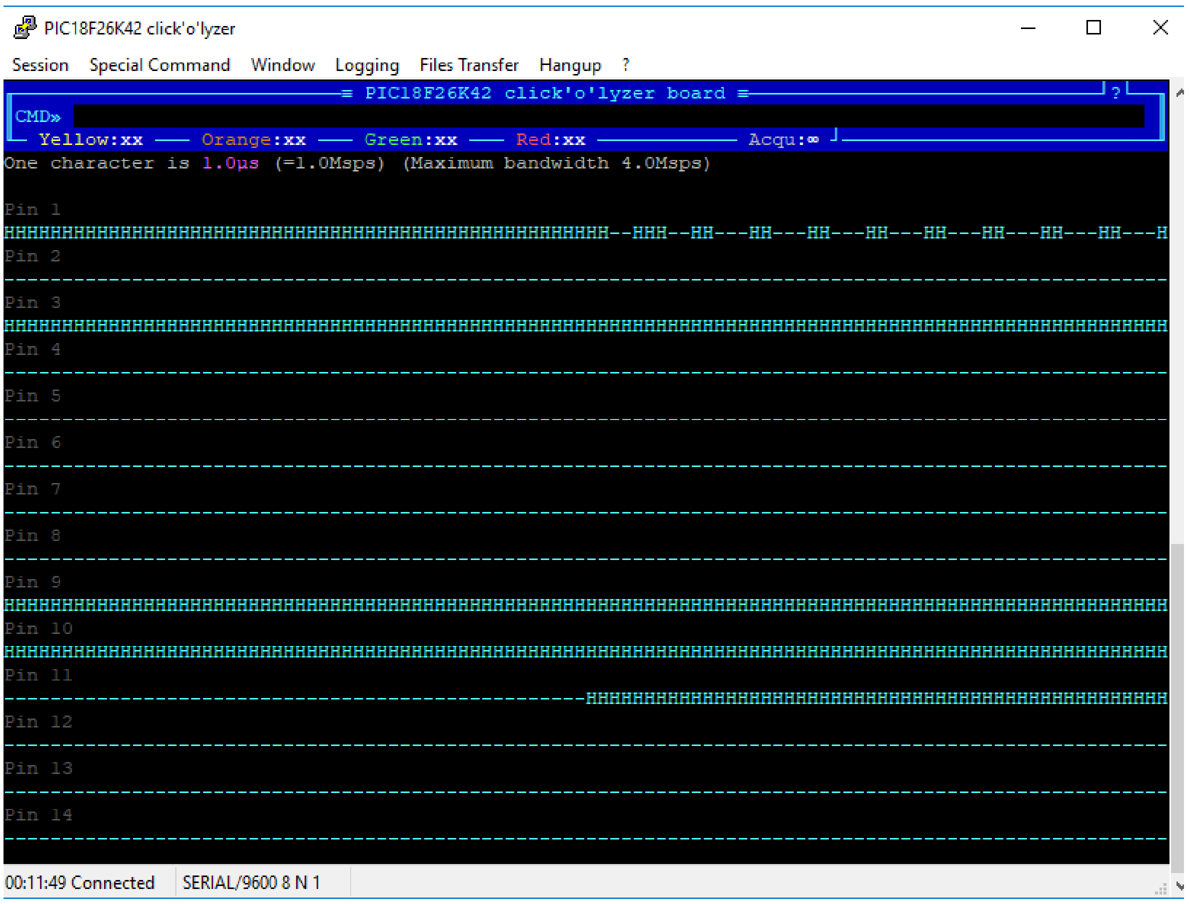
☐ Copy and paste line drawing characters as lqqqk

We will use the Logic Scope capability of the click, that reads the voltage and show a HHHHH as an output when your logic value is high and ------- when your logic value is low. The 14 pins of the ***MikroBUS*** will be shown. During the project, the following connections were implemented for visualization:

- Original signal to transmit: **UART** connected to **RC6**, corresponding to **pin 11** in the ***mikroBUS*** and the **Click Analyser**
- Modulated signal that is transmitted: **DSM** connected to **RA1**, corresponding to **pin 1**

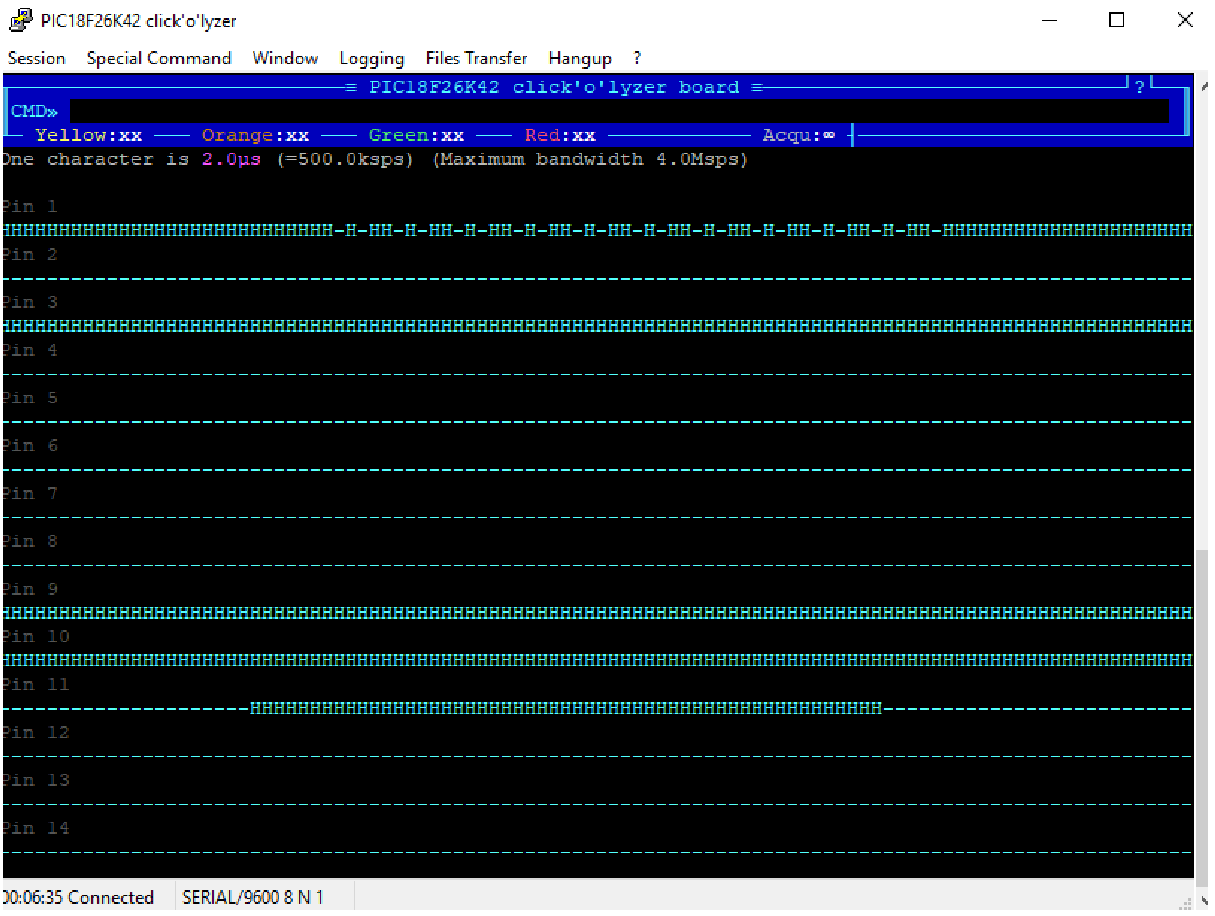
Type the command **LS FREQ=1M** in the command prompt and a single reading of the 14 pins will be shown. The sampling frequency specified in the command can be changed to any value the range of kHz or MHz. To visualize continuous readings of the input pins, click on the ***Acquisition*** button, the symbol will be changed to an infinity sign. Type again the command to apply the changes.

Focusing on pin 1 and pin 11, we can observe how the frequency on pin 1 changes depending on the state of pin 11, demonstrating the modulation of the signal carried out with the FSK transmitter.



```
 Yellow:xx —— Orange:xx —— Green:xx —— Red:xx ———————— Acqu:∞
One character is 1.0µs (=1.0Msps) (Maximum bandwidth 4.0Msps)

Pin 1
HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH--HHH--HH---HH---HH---HH---HH---HH---HH---HH---H
Pin 2
------------------------------------------------------------------------------------------------------
Pin 3
HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
Pin 4
------------------------------------------------------------------------------------------------------
Pin 5
------------------------------------------------------------------------------------------------------
Pin 6
------------------------------------------------------------------------------------------------------
Pin 7
------------------------------------------------------------------------------------------------------
Pin 8
------------------------------------------------------------------------------------------------------
Pin 9
HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
Pin 10
HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
Pin 11
---------------------------------------------------HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
Pin 12
------------------------------------------------------------------------------------------------------
Pin 13
------------------------------------------------------------------------------------------------------
Pin 14
------------------------------------------------------------------------------------------------------
```


PIC18F26K42 click'o'lyzer
Session Special Command Window Logging Files Transfer Hangup ?

```
═══════════════════════════════ PIC18F26K42 click'o'lyzer board ═══════════════════════════════ ?
CMD»
─ Yellow:xx ── Orange:xx ── Green:xx ── Red:xx ──────── Acqu:∞ ┤
One character is 2.0µs (=500.0ksps) (Maximum bandwidth 4.0Msps)

Pin 1
HHHHHHHHHHHHHHHHHHHHHHHHHHHH-H-HH-H-HH-H-HH-H-HH-H-HH-H-HH-H-HH-H-HH-H-HH-H-HH-HHHHHHHHHHHHHHHHHHHHH
Pin 2
----------------------------------------------------------------------------------------------------
Pin 3
HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
Pin 4
----------------------------------------------------------------------------------------------------
Pin 5
----------------------------------------------------------------------------------------------------
Pin 6
----------------------------------------------------------------------------------------------------
Pin 7
----------------------------------------------------------------------------------------------------
Pin 8
----------------------------------------------------------------------------------------------------
Pin 9
HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
Pin 10
HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
Pin 11
---------------------HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH--------------------------
Pin 12
----------------------------------------------------------------------------------------------------
Pin 13
----------------------------------------------------------------------------------------------------
Pin 14
----------------------------------------------------------------------------------------------------
```

00:06:35 Connected   SERIAL/9600 8 N 1

Note