MPLAB® Code Configurator & Core Independent Peripherals

Lab Manual

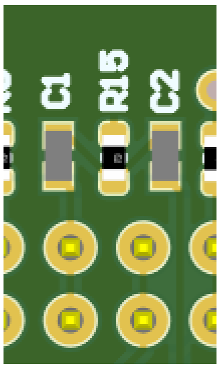
This page is intentionally left blank

Lab 2 – FSK Receiver with CIPs

Purpose

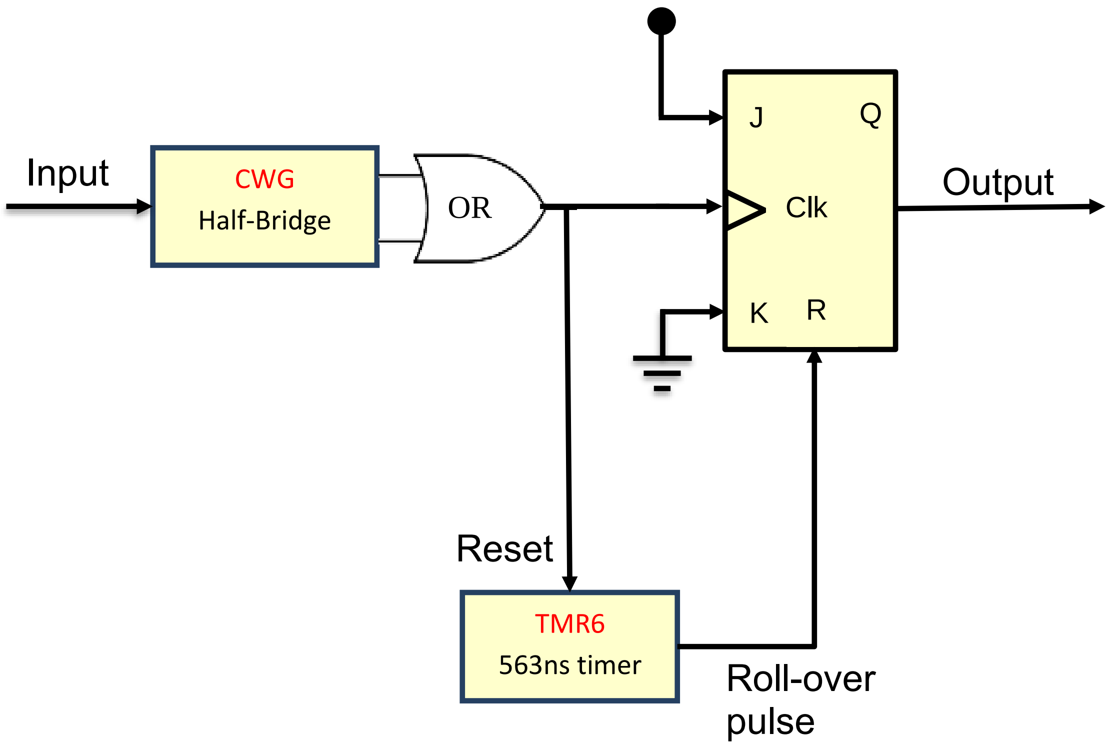
This lab demonstrates how to implement FSK receiver by configuring the core-independent peripherals (CIPs) on the PIC18F47K42. The CIPs used include Timer, Complementary Waveform Generator (CWG) and Configurable Logic Cell (CLC). Curiosity HPC board is used and the project is fully implemented with MPLAB® Code Configurator. The usage of CIPs means that the actual core of the MCU is not involved at all and can either be sleeping to save power or work on other tasks.

Hardware



Function	Pin	Setting
CLC1	RC5	Input
CWG	-	
CLC2	-	
CLC3	RD3	De-modulated output
TMR6	-	Roll-over pulse, external reset

Theory of operation



Theory

CLC1 is used to route the input signal from RC5 pin to the Complementary Waveform Generator (CWG). CWG follows the input signal, but is configured to have a delay of 515ns to 531ns before reacting. In simple FSK modulation we have two different input frequencies, in this case 800 kHz & 1 MHz. With 800 kHz signal, the input signal changes state every 625ns, so the CWG will have time to react to input signal and will thus have output. With 1 MHz input signal, the CWG won’t output anything due to configured delay. So in essence, the CWG determines whether we are receiving the higher or lower of the FSK frequencies, and outputs either a stable signal (high freq) or a continuous train of pulses (low freq). CLC2 is used to combine the CWGA and CWGB outputs so that we monitor both the high and low signal parts.

In the next step, we use the CWG output as an external reset signal to Timer 6. Timer 6 is configured to have a 563ns period, and to output a single pulse when it rolls over (i.e. 563ns has passed without it being reset). This means that when the output of CWG is stable (high frequency), the Timer 6 does not get reset, thus it rolls over and outputs a pulse. This output is connected to CLC3, which is configured as a J-K flip-flop with Reset. Timer 6 output is the reset input of the J-K, which means that the output of J-K is set to 0 when Timer 6 rolls over (= when the CWG output does not change = when the input signal has high frequency).

In addition to being connected to Timer 6, the CWG output is also connected to CLK input of the J-K flip-flop. When the input frequency is low, and the CWG output is a train of pulses, these pulses will clock the J-K flip-flop and will set the state of the flip-flop according to its J & K inputs. These are permanently fixed so that J = 1 and K = 0, which will set the output to 1.

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 (screenshots in Lab 1)

> <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 Curiosity board (**PIC18F47K42**). Please make sure the PIC18F47K42 is selected, and **not** the **LF** variant.

We will not use debug header for this lab, please select "None", and click **Next**

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

Choosing the compiler

From the Select Compiler list **select the latest XC8 compiler**. For this lab we use XC8 (2.00)

Saving the project and finalizing the settings

The **Project Name** for this lab is **FSK_RECEIVER**.
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 v3: Open/Close** from the main menu.

Setup your 64MHz internal clock and other fuses in the System resource:

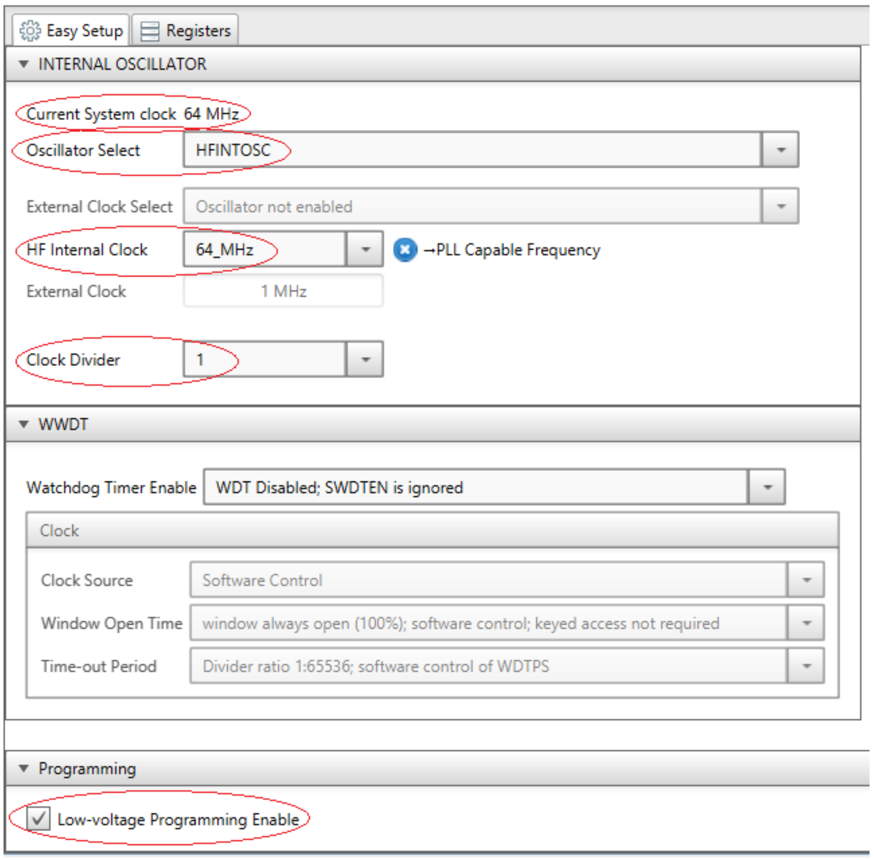
From **Project resources** tab, double click on ***System Module.***

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



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

Watchdog Timer Enable: WDT Disabled; SWDTEN is ignored

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

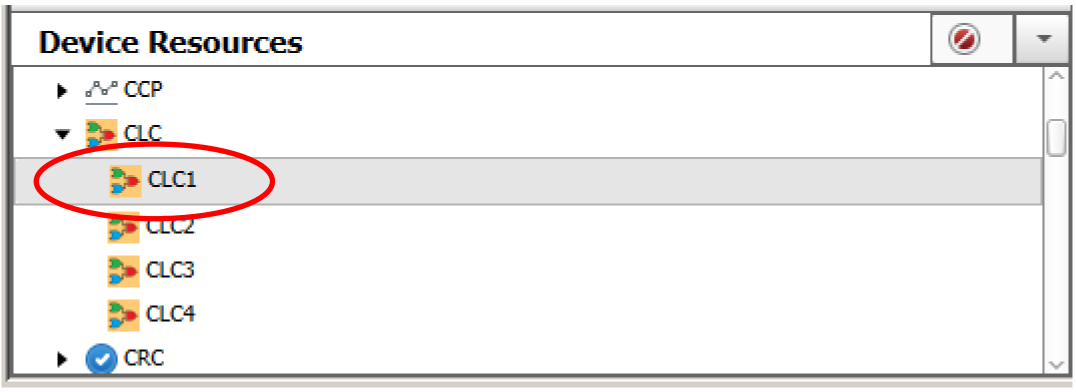
✓ Low-voltage Programming Enable

Add the CLC1 driver to the Project Resources

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

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

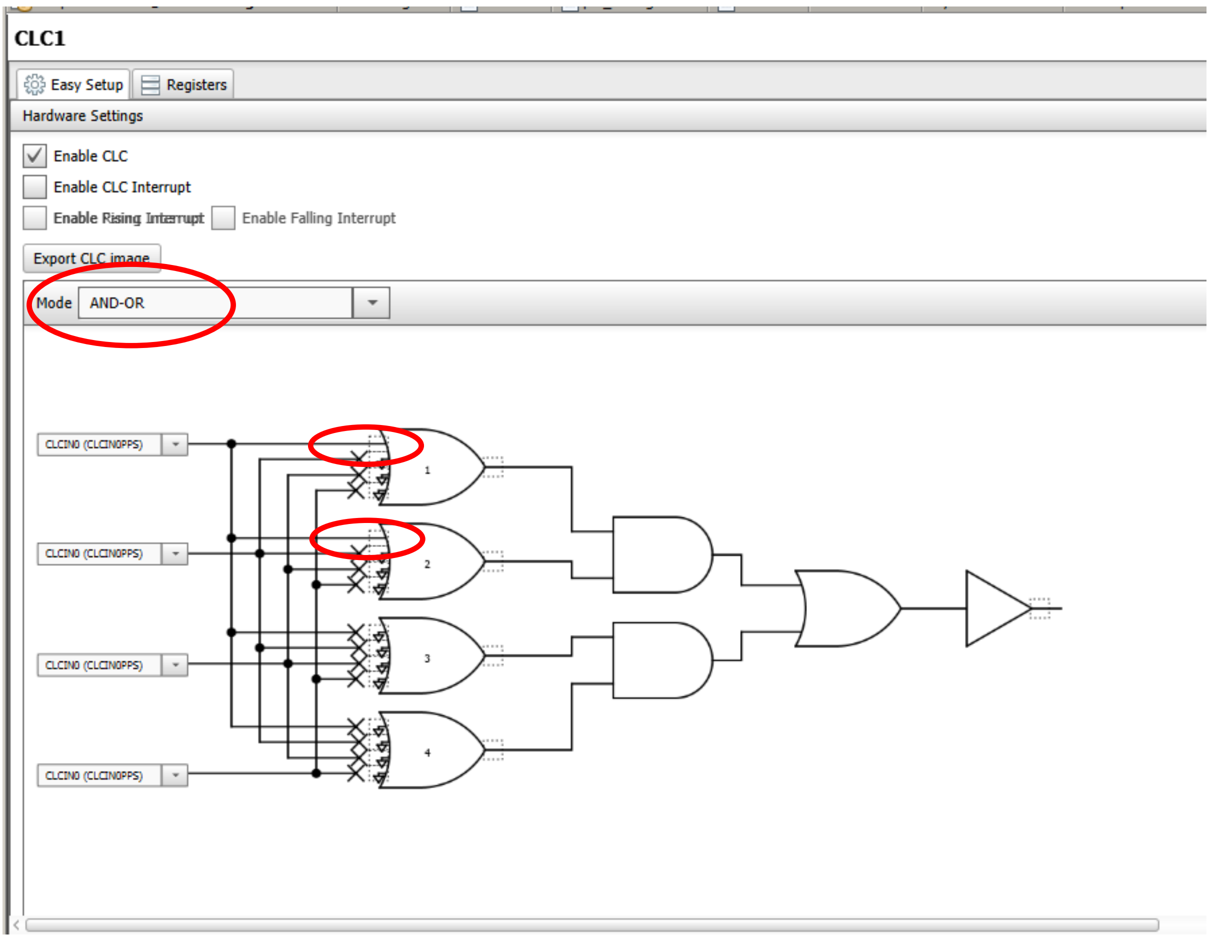
Now double click-on the **CLC1** entry to add it to the **Project Resources** list.



Configure **CLC1** to route input signal to **CWG**

The RC5 pin which is the FSK signal on the CIP board cannot be directly connected to CWG, so we will be using CLC1 block to re-route the signal in this step.

Select AND-OR as CLC1 Mode.
Configure one of the inputs as CLCIN0, and route this to OR gates 1 and 2.
Now the AND gate in the next state will have two similar inputs, and we have the input signal available to **CWG.**


CLC1

Hardware Settings

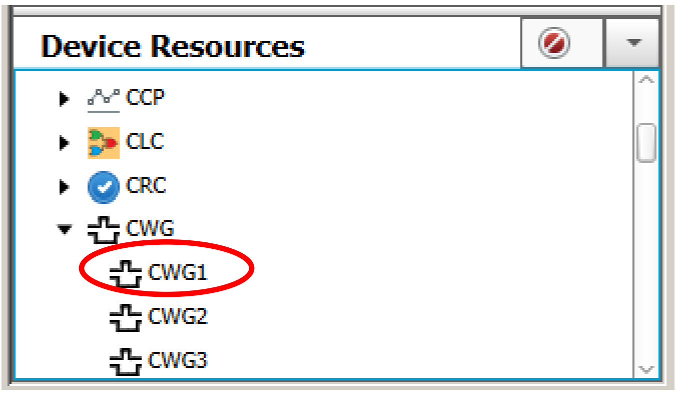
[✓] Enable CLC
[ ] Enable CLC Interrupt
[ ] Enable Rising Interrupt [ ] Enable Falling Interrupt

Mode: AND-OR

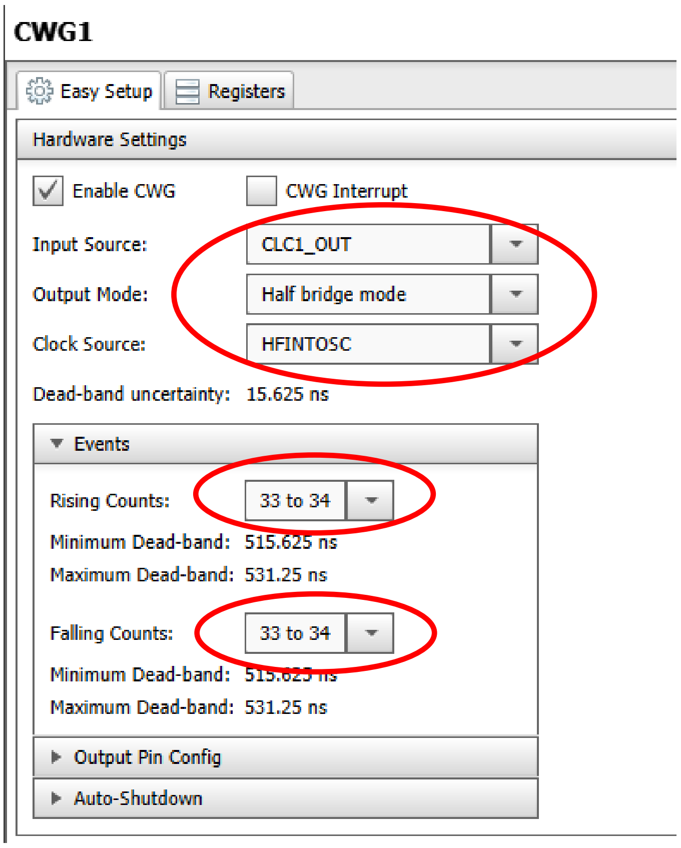
CLCIN0 (CLCIN0PPS)
CLCIN0 (CLCIN0PPS)
CLCIN0 (CLCIN0PPS)
CLCIN0 (CLCIN0PPS)

Add the CWG to project resources
Select the **Resource Management [MCC]** tab.

In the **Device Resources - Peripherals** area scroll-down to the **CWG** entry and expand the list. (**Hint**: You may need to click-on the '▶' symbol.)
Now double click-on the **CWG1** entry to add it to the **Project Resources** list.



Configure the **CWG**
Select **CLC1_OUT** as input source.
Select Output Mode as Half bridge mode.
Set Clock source to **HFINTOSC.**
Expand the '**Events**' menu
Set both **Rising Counts** and **Falling Counts** to **33 to 34** to get the desired 515-531ns delay for **CWG1**



Enable CWG [✓]
CWG Interrupt [ ]
Input Source: CLC1_OUT
Output Mode: Half bridge mode
Clock Source: HFINTOSC
Dead-band uncertainty: 15.625 ns

Rising Counts: 33 to 34
Minimum Dead-band: 515.625 ns
Maximum Dead-band: 531.25 ns

Falling Counts: 33 to 34
Minimum Dead-band: 515.625 ns
Maximum Dead-band: 531.25 ns

Add the CLC2 driver to the Project Resources

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

In the **Device Resources - Peripherals** area scroll-down to the **CLC** entry and expand the list.

(**Hint**: You may need to click-on the '▶' symbol.)

Now double click-on the **CLC1** entry to add it to the **Project Resources** list.

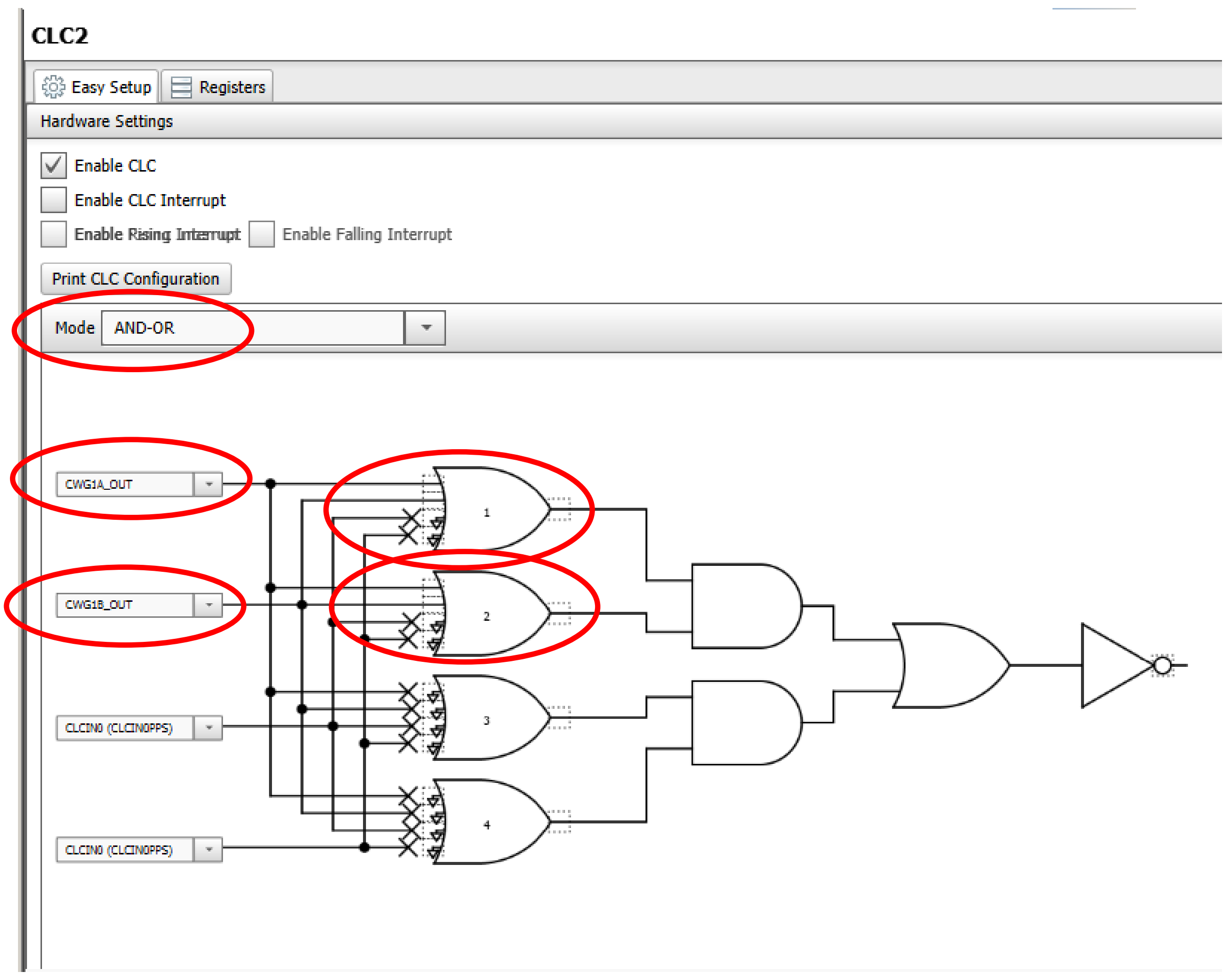
Configure the **CLC2**

Set the mode to **AND-OR**

Select **CWG1A** as input source 1.

Select **CWG1B** as input source 2.

Connect both inputs to OR gates 1 & 2.



CLC2

Hardware Settings

- [✓] Enable CLC
- [ ] Enable CLC Interrupt
- [ ] Enable Rising Interrupt
- [ ] Enable Falling Interrupt

Mode: AND-OR

Add the **TMR6** driver to the Project Resources

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

In the **Device Resources - Peripherals** area scroll-down to the **TMR** entry and expand the list.

(**Hint**: You may need to click-on the '▶' symbol.)

Now double click-on the **TMR6** entry to add it to the **Project Resources** list.

Configure the **TMR6**

Set the **Clock Source** to **FOSC/4**

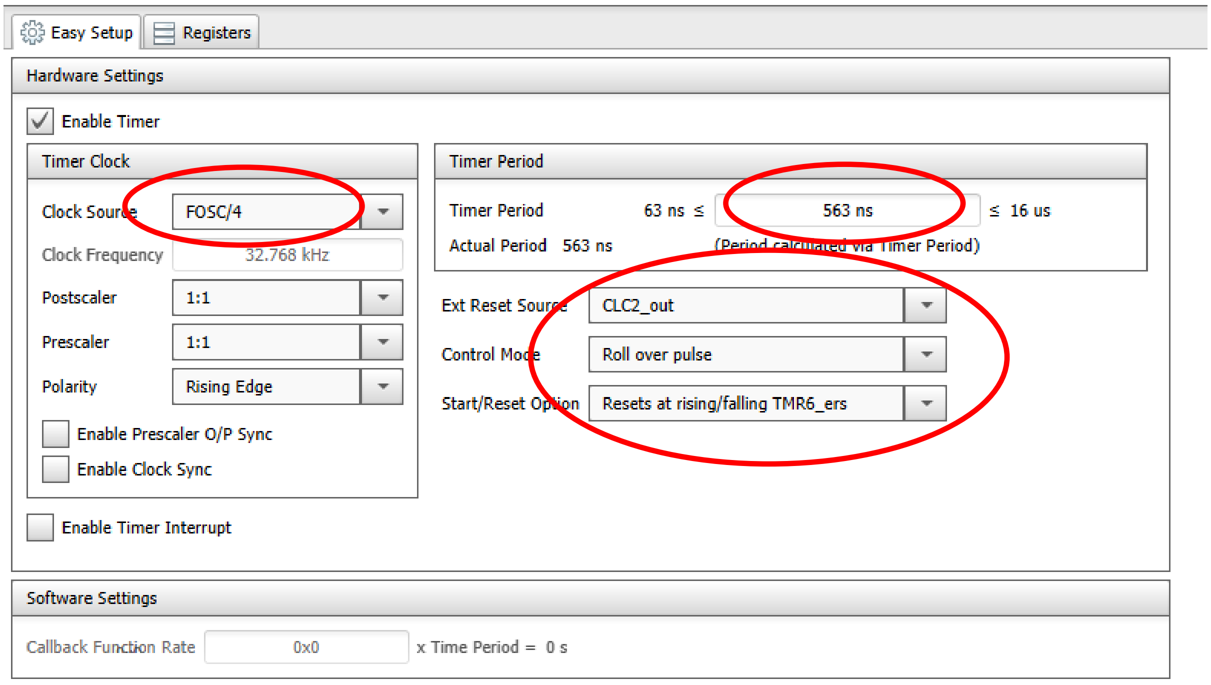
Set the **Timer Period**. With 800 kHz & 1MHz signals and the selected clock source, 625**ns** is suitable value.

Select **Ext Reset Source** as **CLC2_out**.

Select **Control Mode** as **Roll over pulse**.

Select **Start/Reset Option** as **Resets at rising/falling TMR6_ers.**

TMR6



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 ≤ 563 ns ≤ 16 us
Actual Period: 563 ns (Period calculated via Timer Period)
Ext Reset Source: CLC2_out
Control Mode: Roll over pulse
Start/Reset Option: Resets at rising/falling TMR6_ers
Enable Timer Interrupt: ☐
Callback Function Rate: 0x0 x Time Period = 0 s

Add the **CLC3** driver to the Project Resources

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

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

Now double click-on the **CLC3** entry to add it to the **Project Resources** list.

Configure the **CLC3**

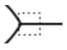
Set the mode to **JK flip-flop with R.**

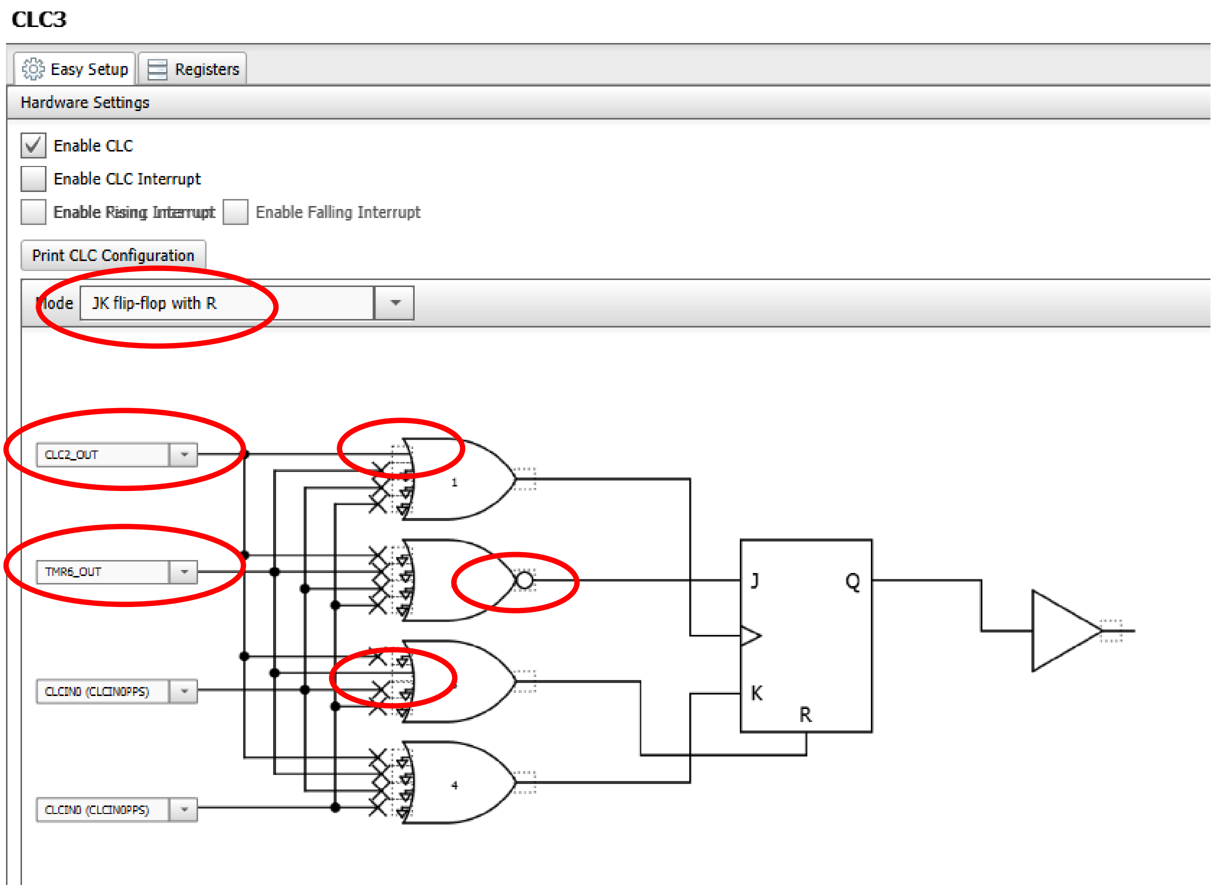
Select **CLC2_out** as input source 1.

Select **TMR6** as input source 2.

Connect input 1 (CLC2_OUT) to AND gate 1, that goes to CLK of the JK

Connect input 2 (TMR6) to AND gate 3, that goes to R of the JK

The J & K need to be permanently fixed to VDD and GND. To achieve this, we need to know that AND gate output without any input is 0 (GND). So just disable any input to AND gates 2 and 4, and negate the output of AND gate 2. You can change logic connections and straight/negated output by clicking on the dashed squares  on the wires


CLC3

Easy Setup | Registers

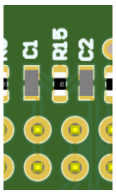
Hardware Settings

- [✓] Enable CLC
- [ ] Enable CLC Interrupt
- [ ] Enable Rising Interrupt
- [ ] Enable Falling Interrupt

Print CLC Configuration

Mode: JK flip-flop with R

Inputs: CLC2_OUT, TMR6_OUT, CLCIN0 (CLCIN0PPS), CLCIN0 (CLCIN0PPS)

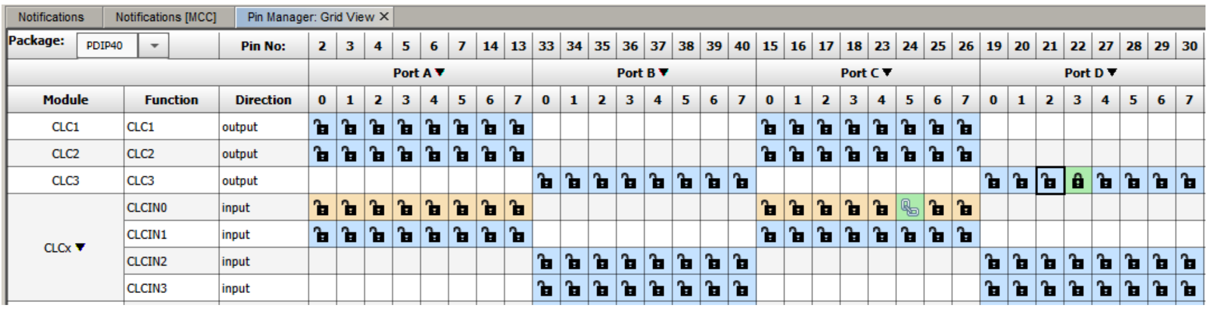


Assign the corresponding pins in the **Pin Manager Grid [MCC]** window

Select **RC5** as **CLC1 input**.

To view the output from the FSK demodulator, we can route the output from **CLC3** to some pin, for example **RD3.**

<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.


Notifications | Notifications [MCC] | Pin Manager: Grid View

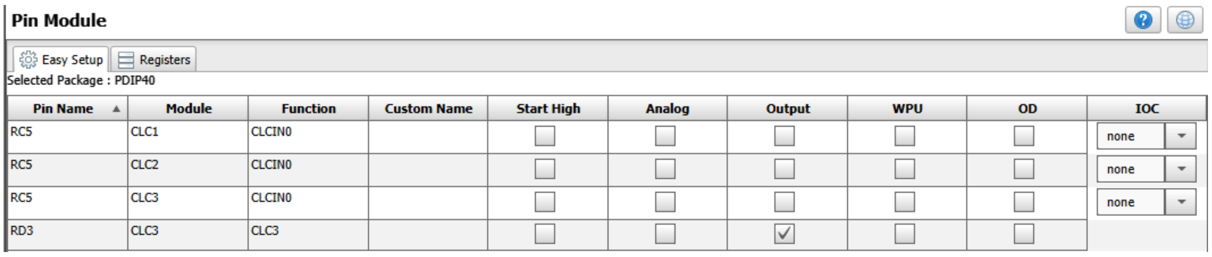
Package: PDIP40

		Pin No:	2	3	4	5	6	7	14	13	33	34	35	36	37	38	39	40	15	16	17	18	23	24	25	26	19	20	21	22	27	28	29	30
			Port A ▼								Port B ▼								Port C ▼								Port D ▼							
Module	Function	Direction	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
CLC1	CLC1	output	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓									🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓								
CLC2	CLC2	output	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓									🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓								
CLC3	CLC3	output									🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓									🔓	🔓	🔓	🔒	🔓	🔓	🔓	🔓
CLCx ▼	CLCIN0	input	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓									🔓	🔓	🔓	🔓	🔓	🔗	🔓	🔓								
	CLCIN1	input	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓									🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓								
	CLCIN2	input									🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓									🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓
	CLCIN3	input									🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓									🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓

Configure the Pin module

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

RD3 needs to be configured as digital output, and **RC5** as digital input. Tick or untick the corresponding boxes to ensure this setup. Note that RC5 / 'CLCIN0' is on the list several times due to it being a default input to CLC2 and CLC3 as well, but since it's not used in those blocks, we can ignore these lines.



Pin Module

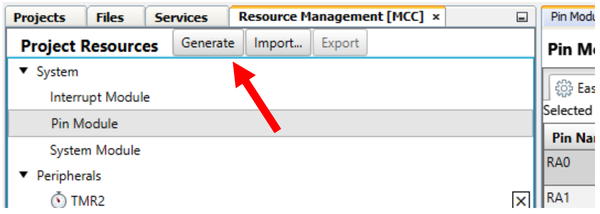
Easy Setup | Registers

Selected Package : PDIP40

Pin Name	Module	Function	Custom Name	Start High	Analog	Output	WPU	OD	IOC
RC5	CLC1	CLCIN0		☐	☐	☐	☐	☐	none
RC5	CLC2	CLCIN0		☐	☐	☐	☐	☐	none
RC5	CLC3	CLCIN0		☐	☐	☐	☐	☐	none
RD3	CLC3	CLC3		☐	☐	✓	☐	☐	

Generate the configuration code for the design

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



Make/build the design and program the target device

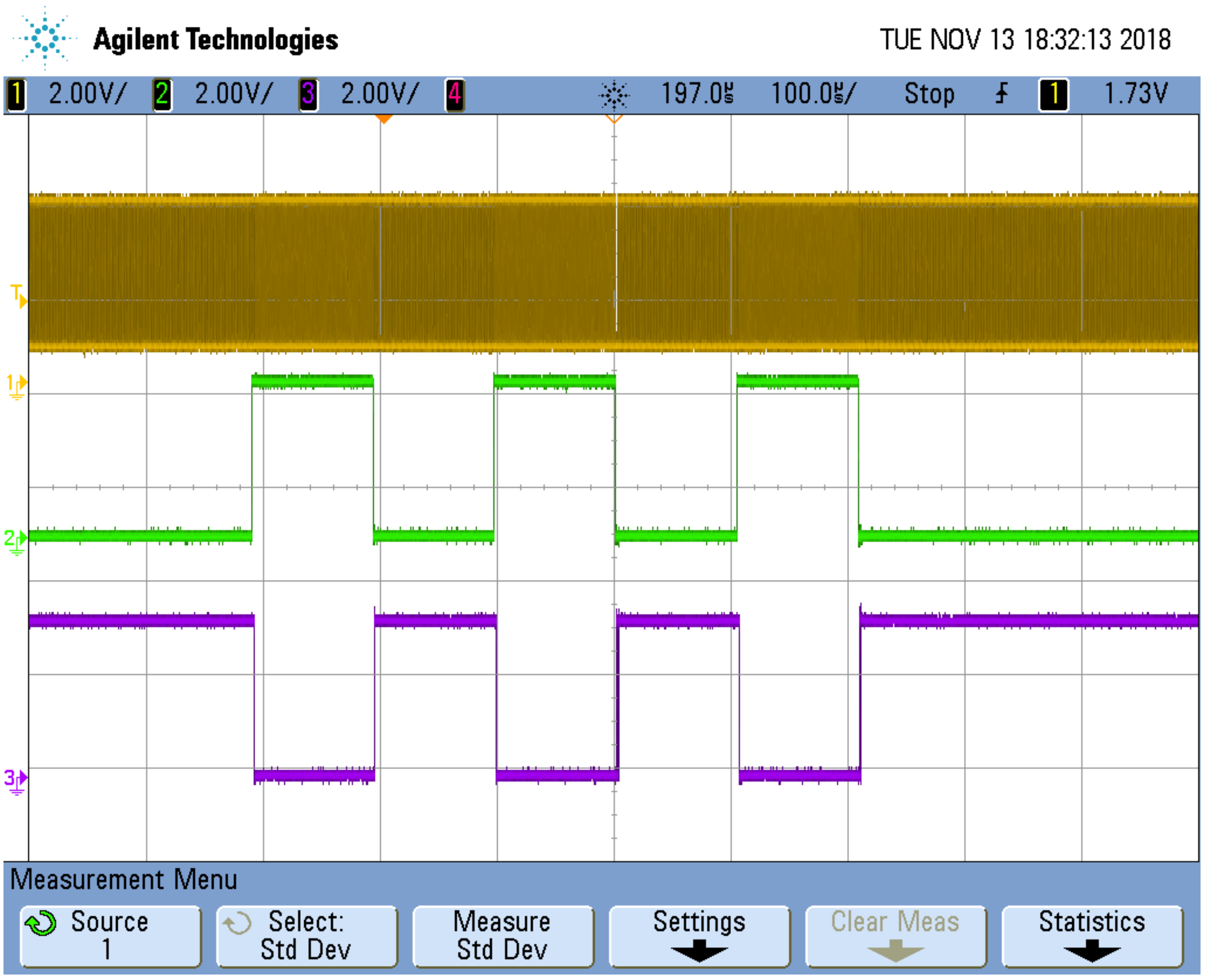
Select the **Make and Program Device** button or select **Run → Run Main Project** from the main menu.

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

Results

The FSK demodulator is now ready, and any FSK input to RC5 pin (with frequencies of 800 kHz and 1 MHz) will be demodulated and output to pin RD3.

To try this out in practice, you can combine the FSK RX & TX projects. This is easiest if you take the TX project as a starting point, and add the peripherals as described here in this lab manual. Remember to also change the pin configuration (RC5 as input, RD3 as output). Then you can observe the demodulated signal on the RD3 pin, and see it change when you press the S1 button.



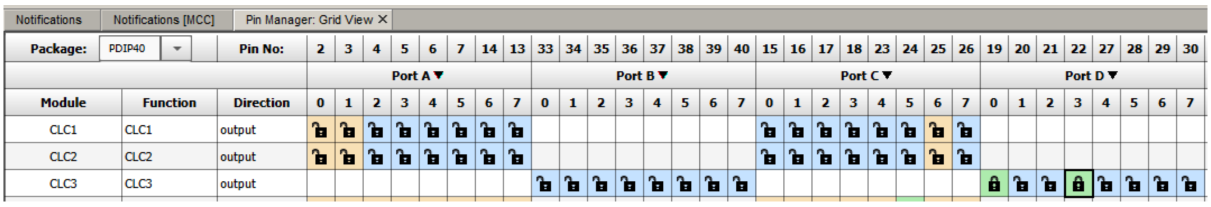
Yellow = FSK signal

Green = UART signal modulating the FSK

Purple = demodulated UART (inverted)

The Click Analyzer board can also be used to monitor the output. Please see the FSK TX training manual for instructions on how to setup the Click Analyzer and terminal on your PC.

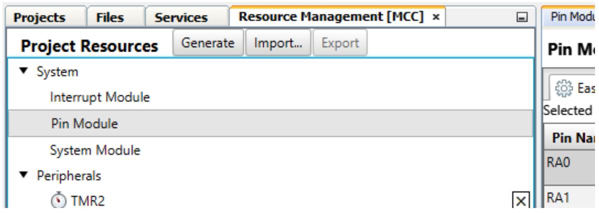
The Click Analyzer is connected to mikroBUS connector 1, and our output pin RD3 is not available there. Please go to MPLAB-X Pin Manager again, and direct the output from CLC3 to some pin that is available on mikroBUS 1 connector, RD0 for example (the output can be routed to multiple pins, so it’s not necessary to remove the output from our original output pin RD3).



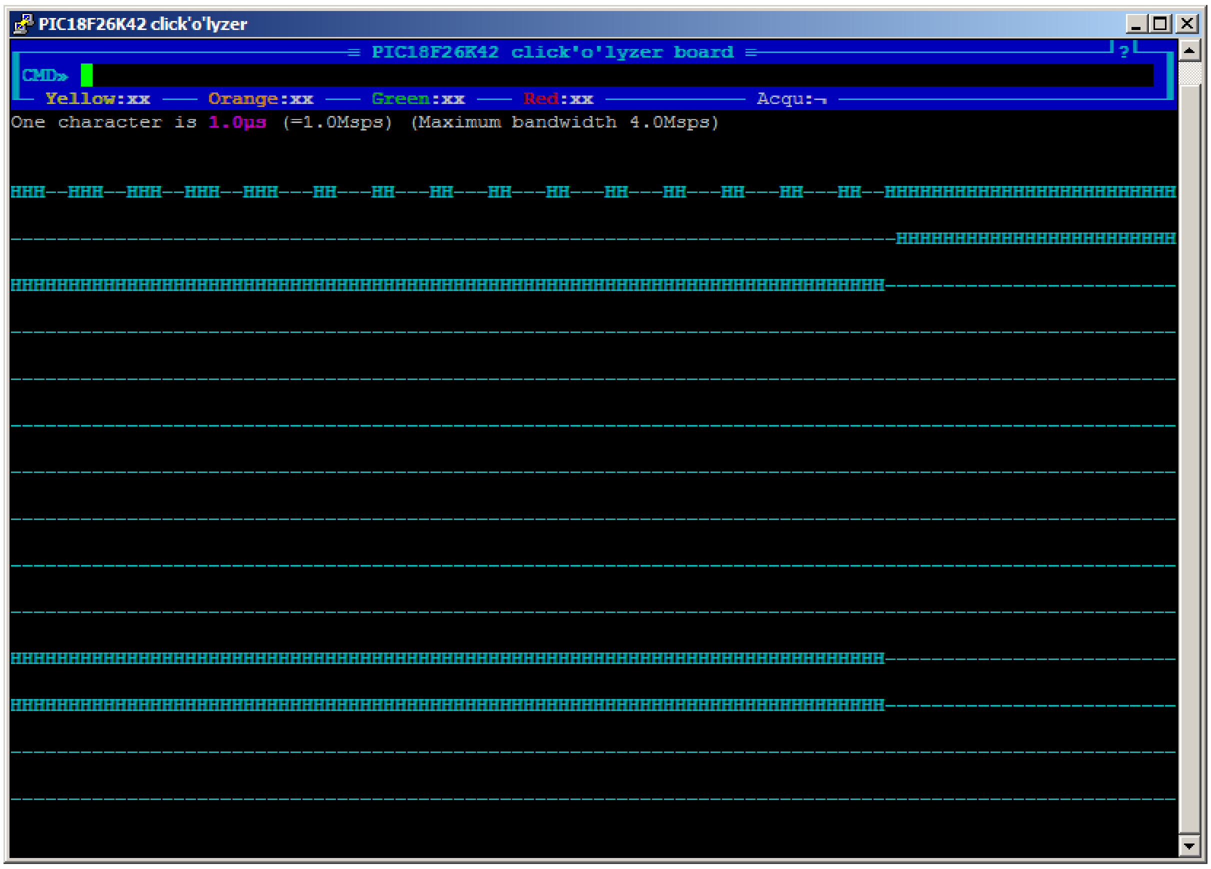
Notifications | Notifications [MCC] | Pin Manager: Grid View

Package:	PDIP40	Pin No:	2	3	4	5	6	7	14	13	33	34	35	36	37	38	39	40	15	16	17	18	23	24	25	26	19	20	21	22	27	28	29	30
			Port A								Port B								Port C								Port D							
Module	Function	Direction	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
CLC1	CLC1	output	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓									🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓								
CLC2	CLC2	output	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓									🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓								
CLC3	CLC3	output									🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓									🔒	🔓	🔓	🔒	🔓	🔓	🔓	🔓

Remember to press “Generate” button in MCC / Resource management / Project resources to apply the changes in the pin configuration. And click Make & Program button to update the firmware on Curiosity board.



Now with these changes we can see how the signals are behaving (type command LS FREQ=1M to sample the signals, you might have to do this several times to catch a moment where the demodulated signal is actually changing):



PIC18F26K42 click'o'lyzer

```
──────────────────────────────≡ PIC18F26K42 click'o'lyzer board ≡──────────────────────────────┘?└
CMD»
 Yellow:xx ── Orange:xx ── Green:xx ── Red:xx ──────────── Acqu:¬ ─────────────────────────────
One character is 1.0µs (=1.0Msps) (Maximum bandwidth 4.0Msps)


HHH--HHH--HHH--HHH--HHH---HH---HH---HH---HH---HH---HH---HH---HH---HH---HH--HHHHHHHHHHHHHHHHHHHHHHHHH

----------------------------------------------------------------------------HHHHHHHHHHHHHHHHHHHHHHHH

HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH-------------------------

----------------------------------------------------------------------------------------------------

----------------------------------------------------------------------------------------------------

----------------------------------------------------------------------------------------------------

----------------------------------------------------------------------------------------------------

----------------------------------------------------------------------------------------------------

----------------------------------------------------------------------------------------------------

----------------------------------------------------------------------------------------------------

HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH-------------------------

HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH-------------------------

----------------------------------------------------------------------------------------------------

----------------------------------------------------------------------------------------------------
```

In the picture above, line 11 is the FSK input (UART), line 1 is the modulated FSK signal, and line 2 (RD0) is the demodulated signal (inverted).

Note