AN2805

Robust Debouncing with Core-Independent Peripherals

Introduction

Author: Qubo Hu, Microchip Technology Inc.

Typical button debouncing approaches are often not sufficiently robust, and with slow setup and response, which may lead to ghost or non-detected stimuli. This Tips and Tricks document presents two more robust and quick debouncing solutions, which equally applies to rotary encoders, buttons, switches, keypads, knobs, and the like. By using an internal timer to provide customized slower clock and Configurable Logic Cells (CLCs), it is possible to filter out various kinds of button generated noise. One solution requires two CLCs per button, which on PIC18F Q10 family devices with eight CLCs can support the debouncing of four buttons. Another solution requires three CLCs per button, which is more robust and addresses a timing corner case in the 2-CLCs solution. The debouncing is performed in hardware peripherals with no CPU execution required, after a system initialization which contains the timer and CLCs configuration. Both solutions require two CLC clock cycles as this gives a very fast debouncing.

An example creation in MPLAB and MCC configuration is described in this document. The example code for replicating the results is also available from MPLAB Xpress code example: https://mplabxpress.microchip.com/mplabcloud/example/details/799.

Table of Contents

1. Background

When physically rotating an encoder or pressing a button, two pieces of metal come into contact with each other. The two pieces of metal can make and break contact intermittently before they are in full contact. When the input signal crosses the digitization threshold, it could result in multiple pulses of varying duration in the digitized signal, as illustrated in Figure 1-1. Because of the high sample rate of the input signal line for the microcontroller to process, multiple pulses on the input signal line can result in one press/release to be registered as multiple triggers. Ideally, short period pulses of the digitized input signal are just noise, and these noisy pulses should be filtered out. Only a stable pulse with a long enough period, which means the input signal is stable for a certain time, should trigger debounce. This method is called “debouncing”.

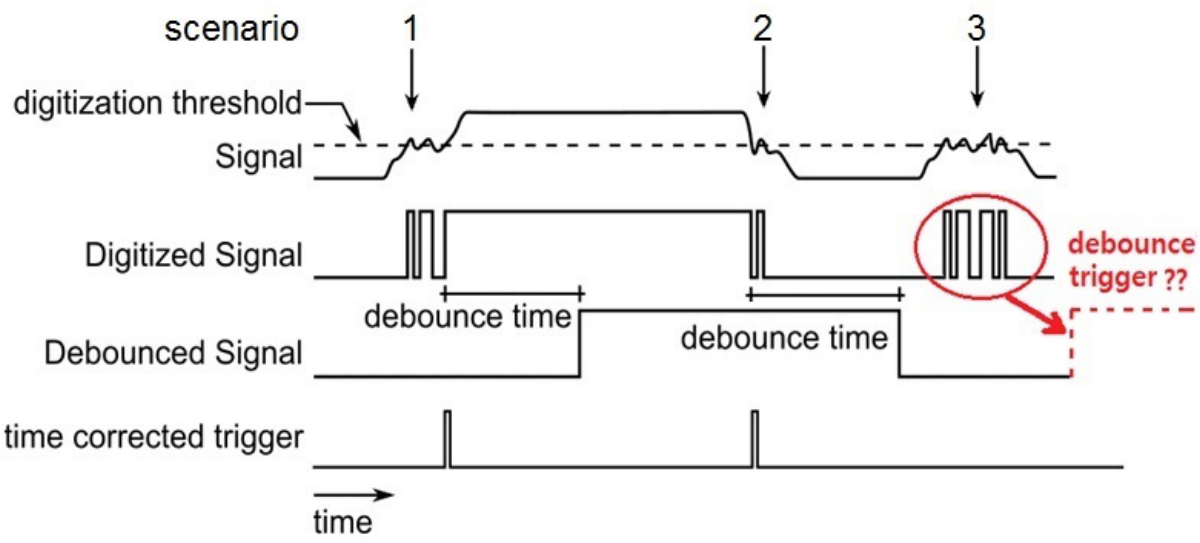
Figure 1-1 also illustrates a number of scenarios associated with signal debouncing. In this figure:

- Scenario 1 refers to the case when the encoder is rotated to ON
- Scenario 2 refers to the case when the encoder is rotated to OFF
- Scenario 3 refers to the case when the encoder is mechanically loose or halfway rotated, making the input signal very unstable

Debouncing is a type of low-pass filter. The debounce time is the delay between the occurrence of the trigger and the registration of the trigger. This debounce time influences the accuracy of the timestamps. For scenarios 1 and 2, the edge of a trigger indicating a stable pulse should be registered after the debounce time, named debounce signal. For scenario 3, many short period pulses can occur in a very short period on the digitized signal. The user should, in advance, define if these short period pulses are noisy pulses to be ignored, or if it could be considered as a valid and stable pulse in a certain condition.

In this document, a rotary encoder has been used as an example for demonstration purposes. However, the same technique applies to also buttons, switches, keypads, and knobs.

Figure 1-1. Debouncing Illustration



1.1 Existing Solutions

For debouncing of slow signals, the simplest hardware solution is to debounce with an external resistive-capacitive (RC) filter component to filter out quick pulse changes in order to keep clean edges for the

microcontroller to process. This will require more onboard components, resulting in extra cost and size. On the other hand, software debounce solutions range from the simple to sophisticated algorithms. The simplest debouncing strategy is to read the debounce input signal by using a delay period, for example, every 50 ms up to 500 ms, and set a flag indicating the input's state. A read during the initial bounce period returns a zero or a one indicating the switch's indeterminate state. One downside is high power consumption since the device cannot be put to sleep while waiting for a delay.

This delay can be handled with a timer that interrupts the CPU at a regular rate, and it can partly release the CPU waiting for the delay. One downside of all these approaches is the slow response since the delay period has to be long enough to avoid multiple pulses recognized at one debouncing. There are enhanced approaches which have been developed to reduce the debounce time. For example, there is a so-called pattern-based debouncer method which takes the overall pattern of the switch's voltage output over a relatively long period of time into account. The intention is to improve the overestimating of the bounce timeout. It keeps track of which state the button is currently in and detects the debouncing at a fine-grain time period but requires some more code to realize the function.

Code Free Switch Debounce using TMR2 with HLT proposes a code-free solution, in which the very first switch activation is used to start the timer counting, ignoring any subsequent bouncing. Once the timer count reaches a predetermined value, the timer peripheral will produce a signaling event that can be used to indicate that a valid switch activation has been detected. It is performed on hardware with no code requirements other than setting up the CIPs and Timer 2. This approach is code-free and fast. The limitation is that it is the first pulse which is used to generate the debounced signal, with no identification whether it is a stable pulse or a noise pulse.

2. Robust Debouncing with CLCs

As far as is known, none of the existing solutions work as expected for the three scenarios shown in Figure 1-1. Some existing solutions rely on detecting the first input signal pulse, without checking whether it is a noise pulse or not. Or it waits for a long, fixed delayed period (slow debounce time) and rechecking if the pulse is still valid. Debouncing relying on the first pulse could result in a debounce signal at an incorrect time or even incorrect debouncing for a scenario 1 case. It often has slow debounce time while waiting for a fixed, delayed period and requires the CPU's intervention. For scenario 3, existing solutions could result in an unpredictable debounce signal as the number of pulses varies and the pulse periods vary.

This document introduces two solutions using timer and CLCs. The first proposed solution requires one timer and two CLCs per button and is named two CLCs (2-CLCs) debouncing solution. It works, in practice, even for the extreme example with various noisy pulse occurrence. An enhanced three CLCs (3-CLCs) solution is hence introduced and will be discussed in the solution description. The user has the option to choose between these two solutions based on their requirements.

2.1 2-CLCs Debouncing Solution

Figure 2-1 presents the logic diagram for the 2-CLCs solution. Four 2-input D flip-flops represent two groups of CLCs, CLC3 and CLC1, CLC4 and CLC2 respectively. They correspond to two input buttons, RC5 and RB2, respectively. The input signal of CLC3 is logical AND with the CLC3 output signal to serve as input to CLC1. The timer6 signal serves as clock for all CLCs. The timer can be reused among all the buttons if their debouncing scan rates are in the same range. The debouncing output will light the LEDs (D2, D3, D4, and D5) for demonstration purpose on the MPLAB Curiosity HPC board. PIC18F Q10 family devices contain up to eight CLCs which means four buttons can be supported. The debouncing is performed with hardware peripherals and no code is required other than configuring the timers and CLCs in MPLAB Code Configurator (MCC) in MPLAB X. It does not require external hardware filtering and hence avoids extra space usage and cost. The configuration is explained in the 3. Project Creation and Configuration chapter.

Figure 2-1. Two CLCs Debouncing Solution Logic Diagram

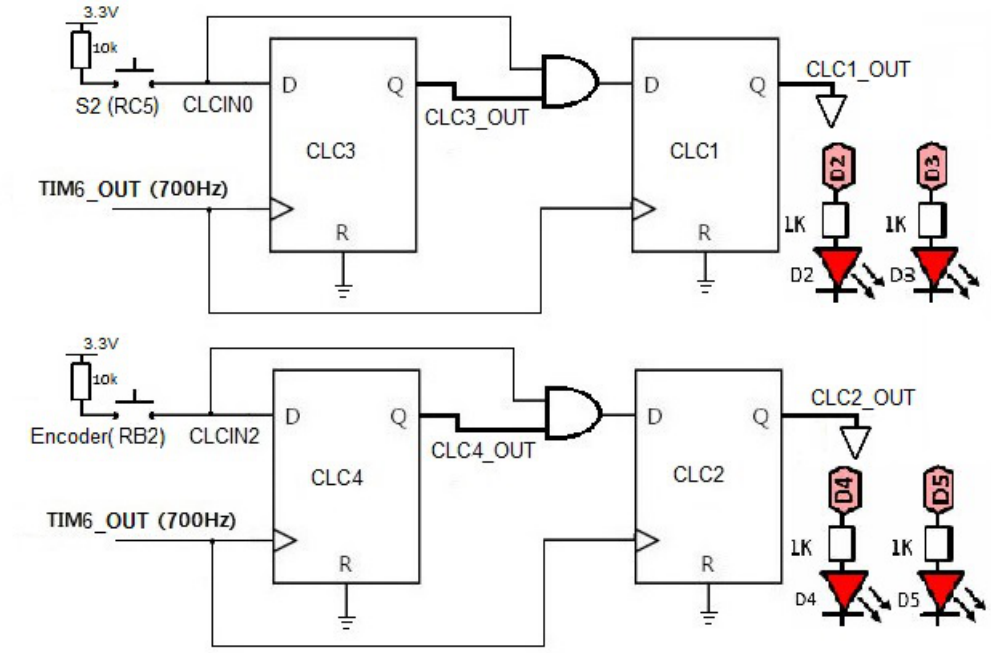
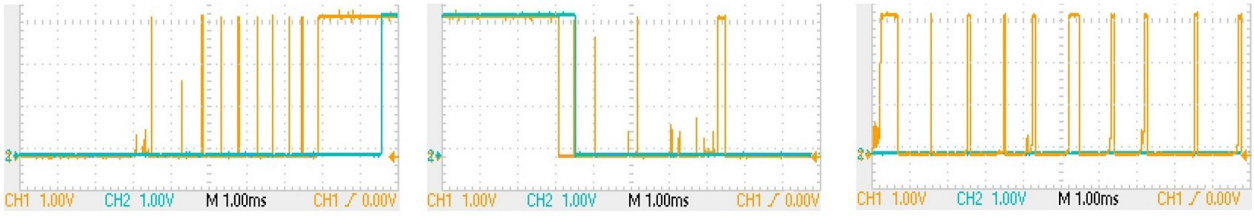


Figure 2-2.a, .b, and .c illustrate the oscilloscope trace of three scenarios when the rotary encoder is switched ON, OFF, and partly switched. In the oscilloscope trace, CH1, the yellow colored signal illustrates the digitized input signal. CH2, the green colored signal, is the debounce signal, which is equivalent to the CLC2 output, CLC2_OUT, in Figure 2-1. As shown, when the encoder gets switched there are many short noisy pulses before it gets stabilized. These noisy pulses normally have a period of maximum 500 µs for the example rotary encoder. In order to filter out these noisy pulses, it is crucial to make sure that the CLC clock period is much larger than the maximum noisy pulse period. This ensures that no noise pulses are recognized as ghost debouncing in two consecutive clocks. A 1.5 ms clock period is chosen in this case, which is three times the maximum noisy period, which means around 700 Hz clock frequency for the timer to overflow. The CLC clock can be configured at any frequency depending on the requirement.

Figure 2-2. Oscilloscope Images of Debouncing Scenarios



a. noise pulses when encoder rotated ON

b. noise pulses when encoder rotated OFF

c. noise pulses when encoder half rotated

For the 2-CLCs solution, the output signal will be logic HIGH when the input signal is held HIGH consecutively for two clock cycles. Since the noisy pulses in Figure 2-2 all have periods shorter than 500 µs which is smaller than the CLC period (1.5 ms), one noise pulse can get the 1st CLC output signal to be logic HIGH but will not lead the 2nd CLC output signal to be HIGH, and hence no debouncing occurs. Note that the input signal must be low for only one clock cycle for the debounce output signal to go low, as the output debouncing signal is ANDed of the input signal and the 1st CLC output. This is the case for the debouncing signal shown in Figure 2-2.b.

In the extreme case, when the rotary encoder is half switched, the two pieces of metal comes into partly contact for a longer period. The input signal gets very noisy and many noise pulses occur continuously as shown in Figure 2-2.c. In this case, the chance that the input signal is read HIGH at the two consecutive CLC clock cycles increases and debouncing would occur. However, there are still not any debouncing monitored in the oscilloscope for this extreme case. The reason why no debouncing occurs might be due to that the noisy pulses have a much shorter period (<300 µs) than the chosen CLC clock period (1.5 ms). Another important reason is that there are no two consecutive noisy pulses which are caught at the CLC clock edge. The chance still exists though it has not been seen in experiments.

2.1.1 Low-Power Mode

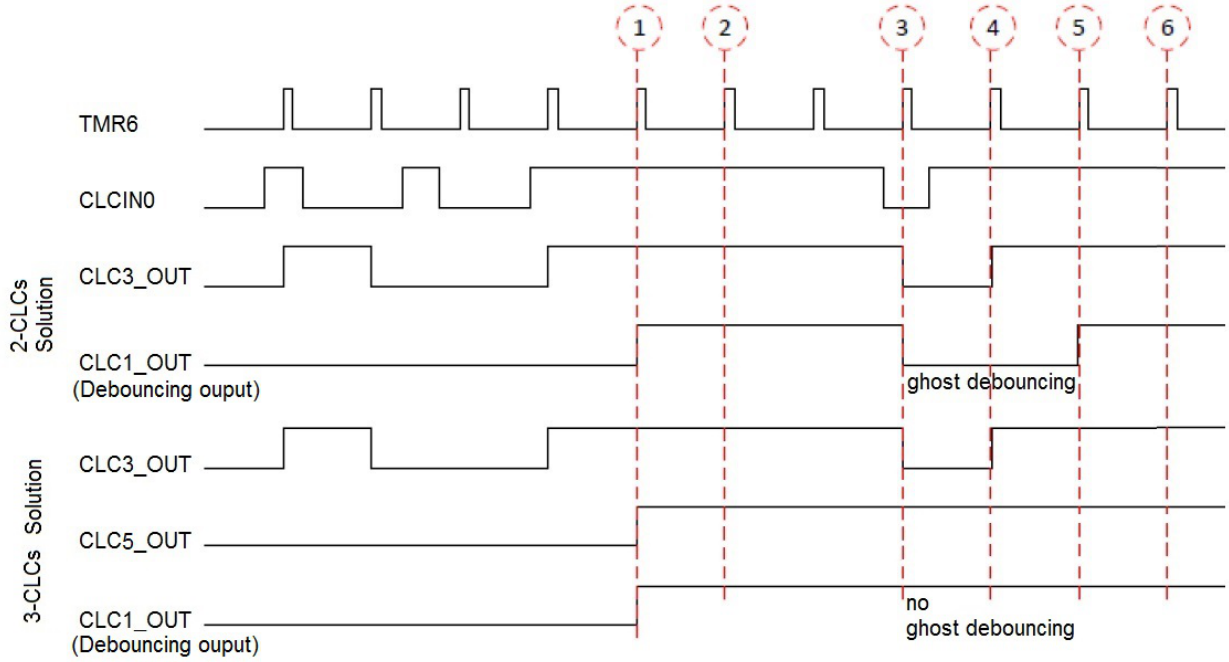
As mentioned, this is a pure hardware solution with no extra code required. Since both the timer and the CLCs can run in Sleep mode, the device can be configured to run in Sleep mode in order to suspend the CPU to minimize the power consumption. This can be done by inserting the following line of code in the *main()* function of the *main.c* file, after the system initialization function:

```
SLEEP();
```

2.2 3-CLCs Debouncing Solution

As discussed, the 2-CLCs solution is asymmetrical. The input signal needs to be logic HIGH for two clock cycles for the debouncing output to be HIGH and to be logic LOW for just one clock cycle for the debouncing output to be LOW. This is normally not a problem with asymmetrical debouncing time. However, when the input signal is holding HIGH while it has a LOW glitch which gets identified at the CLC clock edge, this will cause the output debouncing signal to go LOW for two clock cycles and ghost debouncing will occur. This is illustrated with the timing diagram of the 2-CLCs solution in Figure 2-3. The ghost debouncing scenario is illustrated at the marked 3rd clock edge where both CLCs outputs are LOW. This scenario has not occurred in experiments but the possibility exists. When this scenario happens, it is recommended to implement the 3-CLCs solution which avoids the occurrence of ghost debouncing.

Figure 2-3. Timing Diagram for the 2-CLCs Solution and 3-CLCs Solution

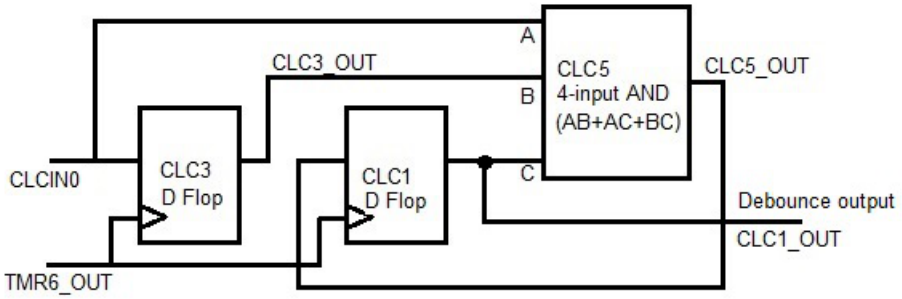


The logic diagram of the 3-CLCs solution is shown in Figure 2-4. The 1st CLC (CLC3) has the same input as before (see Figure 2-1). The 3rd CLC (CLC5) is chosen as a 4-input AND logic. The input signal and the first two CLC output signals are paired ANDed and then ORed together as

$$CLC5\_OUT = CLCIN0 * CLC3\_OUT + CLCIN0 * CLC1\_OUT + CLC3\_OUT * CLC1\_OUT$$

The output signal of the 3rd CLC provides the input signal to the 2nd CLC (CLC1). The 2nd CLC outputs the same debouncing signal as before. The 3rd CLC ensures no single LOW value on any of its three inputs will cause the debouncing output to go LOW. Figure 2-3 also shows the timing diagram of the 3-CLCs solution where the ghost debouncing does not exist anymore, which would occur in the 2-CLCs solution.

Figure 2-4. 3-CLCs Debouncing Solution Logic Diagram



The PIC18F Q10 family with eight CLCs can only support debouncing on two buttons with the 3-CLCs solution. On the other hand, it can support four buttons with the 2-CLCs solution. Since the 2-CLCs solution is normally sufficient even for the extreme case as shown in Figure 2-2.c, the user can decide which solution to use or to use a combination of the two solutions based on their application requirement and also hardware resources.

2.3 Further Discussions

So far we have shown how the two solutions work. Experiments show both works for the typical three scenarios as shown in Figure 2-2. Both provide a very fast reaction with the debounce time at 3 ms as demonstrated, while existing solutions may require up to 20-200 ms debounce time. In contrast, no existing solutions cover all the three scenarios with no unexpected debouncing occurring.

When a series of noisy pulses occur consecutively as shown in Figure 2-2.c, there is still a probability that debouncing occurs if noisy pulses get caught at the two consecutive rising CLC clock edges. The user needs to define whether it should be considered as a stable input pulse with valid debouncing output generated or not. If it is not considered as valid debouncing, the suggestion on how to reduce the possibility is to use more stages of CLCs and/or to extend the CLC clock period. For example, introducing another stage of CLC means further check of noise pulse occurrence on the input signal at the third consecutive clock cycle and hence reduce the possibility for unexpected debouncing to be recognized. For 2-CLCs solution, another stage of CLC means 3-CLCs are required. For the 3-CLCs solution, it means four CLCs are required. On the other hand, it also helps when the CLC clock period gets prolonged. When the CLC clock period gets prolonged, more noise pulses will get filtered out during one clock period and fewer noise pulses would occur at the next CLC clock cycle.

This document describes debouncing of up to four buttons with the 2-CLCs solution on the PIC18F Q10 family devices with eight CLCs available. Or it can support debouncing of three buttons by choosing two 3-CLCs solutions and one 2-CLCs solution. It can also be adapted to support fewer buttons with more CLCs per button if preferred, depending on the user's requirement.

So far we have discussed how to use a timer to provide a configurable clock frequency to CLCs. An alternative is to use the main clock at a lower frequency as the CLC clock frequency if it satisfies the user's requirement. The timer will be saved if the main clock is slow enough for the CLC and this save can further reduce power consumption.

3. Project Creation and Configuration

This section describes how the example project is created and configured in MPLAB for the 2-CLCs solution. The configuration of the 3-CLCs solution will be described later in this document.

3.1 Project Overview

Project overview using the MCU:

- CPU clock: 1 MHz
- Peripherals used:
  - Timer6:
    - Timer6 is configured to overflow with 1.5 ms timer period and is used as CLC clock
  - CLC1, CLC2, CLC3, CLC4, and CLC5:
    - CLC3 and CLC1 are used together for onboard button S2 debouncing control
    - CLC4 and CLC2 are used together for external rotary encoder debouncing control
    - CLC5 is used to extend the 2-CLCs solution (CLC3 and CLC1) to 3-CLCs solution
  - GPIO:
    - pin RC5: CLCIN0 input signal, connected to the onboard S2 button by default
    - pin RB2: CLCIN2 input signal, manually connect it to the external rotary encoder
    - pin RA4: D2 LED output
    - pin RA5: D3 LED output
    - pin RA6: D4 LED output
    - pin RA7: D5 LED output

3.2 Hardware Prerequisites

Hardware used in this application:

- A Curiosity High Pin Count (HPC) Development Board (DM164136)
  - http://www.microchip.com/developmenttools/ProductDetails/PartNo/DM164136
- A PIC18F45Q10 40-lead microcontroller (the PDIP package to fit the Curiosity HPC)
  - https://www.microchip.com/wwwproducts/en/PIC18F45Q10
- An external rotary encoder (the encoder on the XMEGA-E5 Xplained board is used)
  - https://www.microchip.com/DevelopmentTools/ProductDetails/PartNO/ATXMEGAE5-XPLD
- Two Micro-USB cables
- Two male-to-female wires

3.3 Software Prerequisites

Software used in this application:

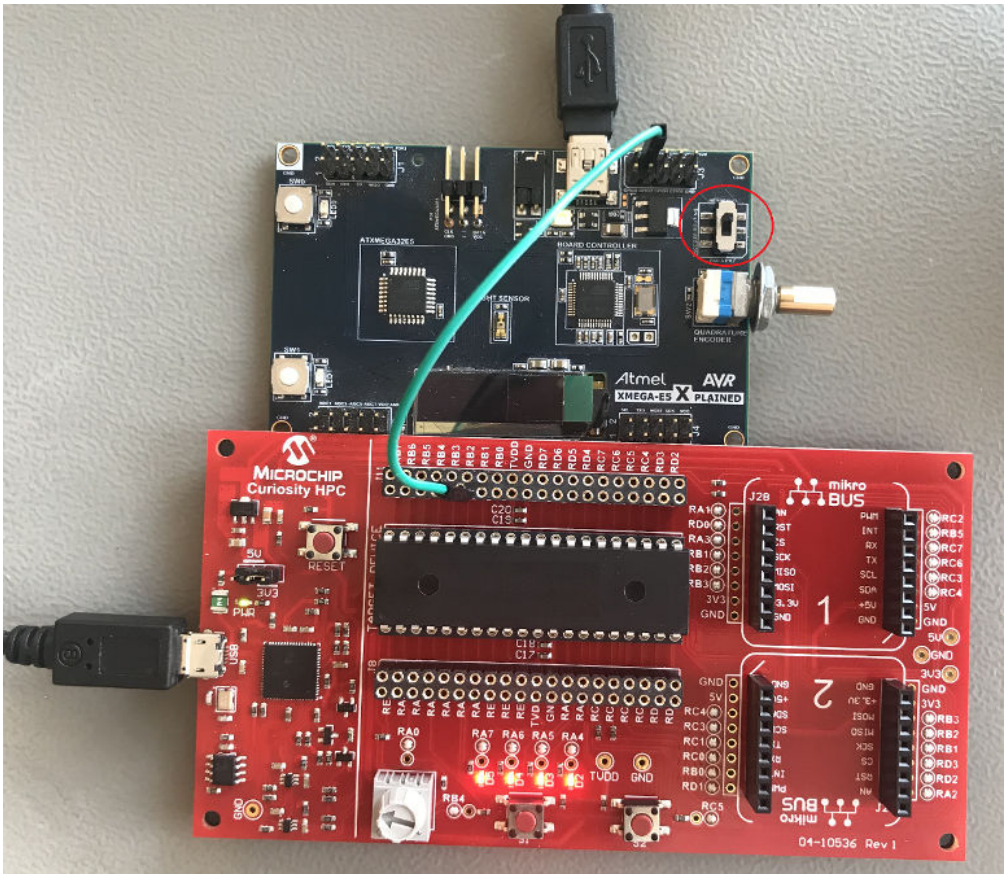
- MPLAB ® X IDE v.5.15
  - http://www.microchip.com/mplab/mplab-x-ide
- MPLAB XC8 C Compiler v.2.0
  - http://www.microchip.com/mplab/compilers

- MPLAB Code Configurator (MCC) v3.75 plug-in in MPLAB

3.4 Hardware Setup

- The Curiosity High Pin Count (HPC) Development Board (DM164136) is used as the test platform. Switch S2 is used to test button debouncing. An external rotary encoder from the XMEGA-E5 Xplained board is used to test encoder debouncing. Onboard LEDs D2, D3, D4, and D5 are used to detect light respectively when debouncing on S2 and external encoder occurs.
- The XMEGA-E5 Xplained board uses the USB power supply. The encoder pin, *J3→GPIO3* on the XMEGA-E5 Xplained board, is connected to the debouncing input pin RB2 of the HPC board. Also, make sure the switch is set to route the encoder to J3 in the *Header J3* as highlighted in the red circle in Figure 3-1.
- The HPC uses the USB power supply, configured to 3.3V V_{CC} by a jumper

Figure 3-1. HW Setup: Curiosity HPC with External Rotary Encoder Connected



The table below shows the external encoder pin connection from the HPC board to the XMEGA-E5 Xplained board.

Table 3-1. HW Connection of HPC with External Rotary Encoder

HPC	XMEGA-E5 Xplained board
RB2 pin	GPIO3 pin

3.5 2-CLCs Solution Configuration in MPLAB

This section describes how the example project is created and configured in MPLAB for the 2-CLCs solution. Refer to the block diagram in Figure 2-1 for the configuration.

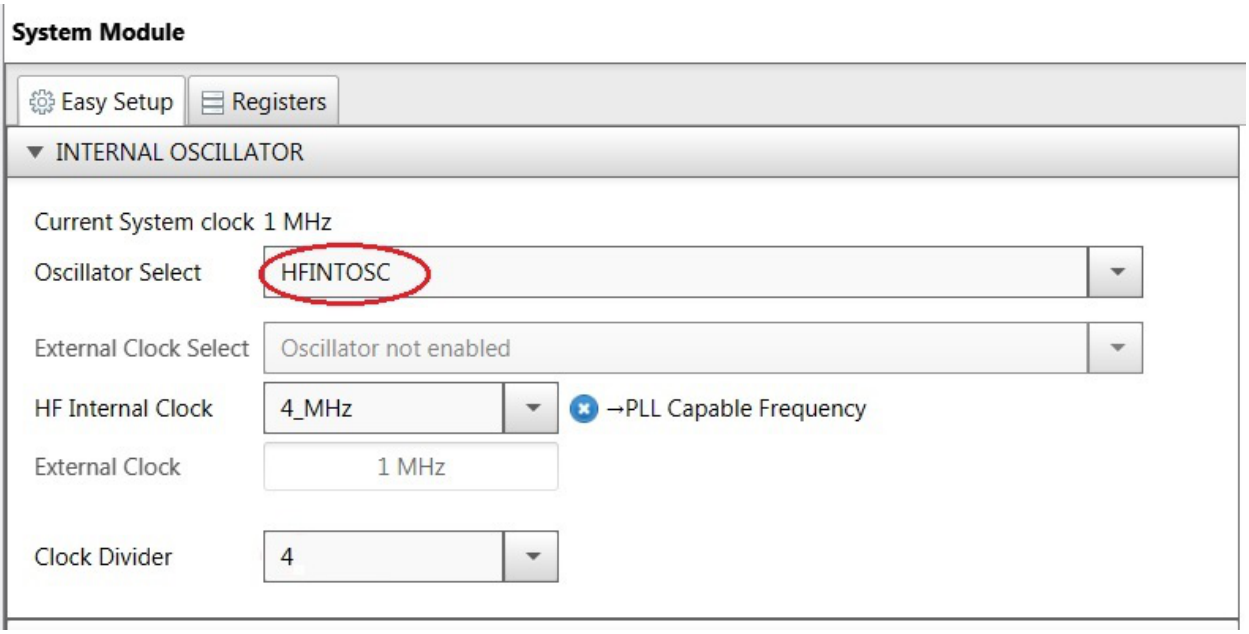
In MPLAB X IDE, create a new project with *PIC18* as the selected *Device Family* and *PIC18F45Q10* as the *Device*. Select *Hardware Tools* as *Curiosity in Microchip Starter Kits*. Select *Compiler* as *XC8* (v2.0 was used).

When the project is created, click on the *MCC* logo on the top menu bar for code configuration. As described, one timer and four CLCs will be used on two debouncing inputs. On the left side, double-click *TMR6, CLC1, CLC2, CLC3*, and *CLC4* from the *Device Resources* section to add them to *Project Resource* section.

3.5.1 Clock

Configure the system clock to 1 MHz under *Project Resource* → *System* → *System Module*. Select *HFINTOSC* for the *Oscillator Select* option as highlighted in the red circle in Figure 3-2. Make sure *4* is selected in the *Clock Divider* option. This will configure the system clock to 1 MHz.

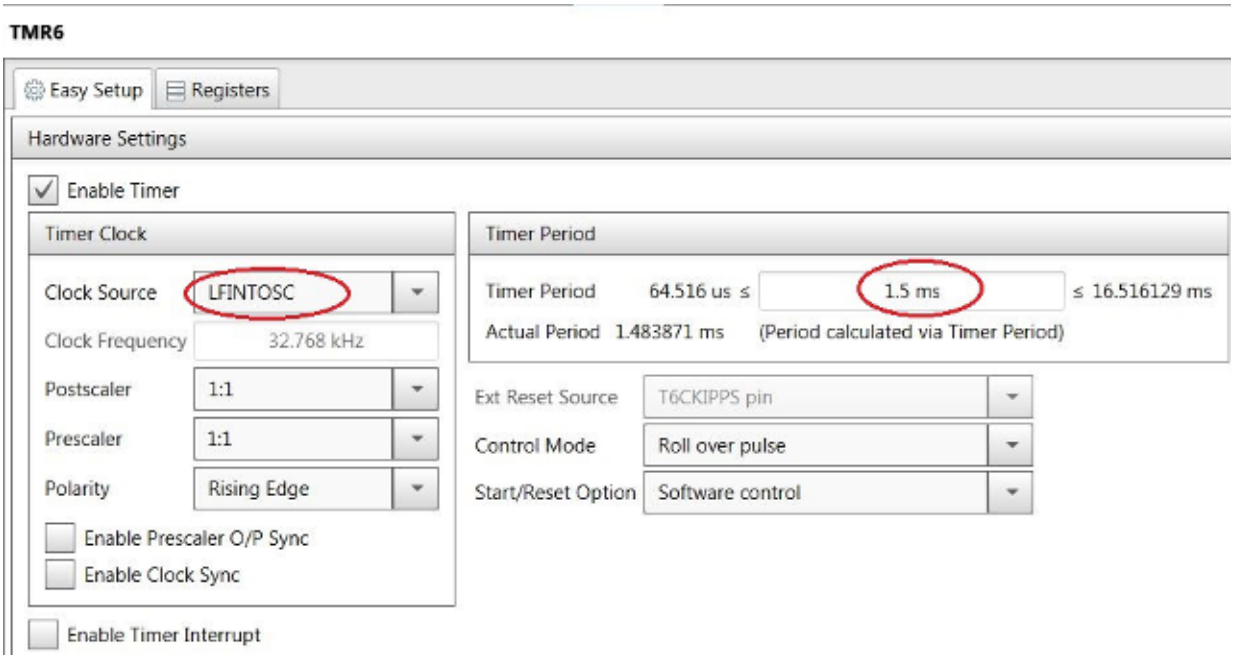
Figure 3-2. System Module at 1 MHz System Clock



3.5.2 Timer

Configure *TMR6* to get a 1.5 ms *Timer Period*. Select *LTINTOSC* as the clock source and type in *1.5 ms* in the *Timer Period* field, as shown in Figure 3-3.

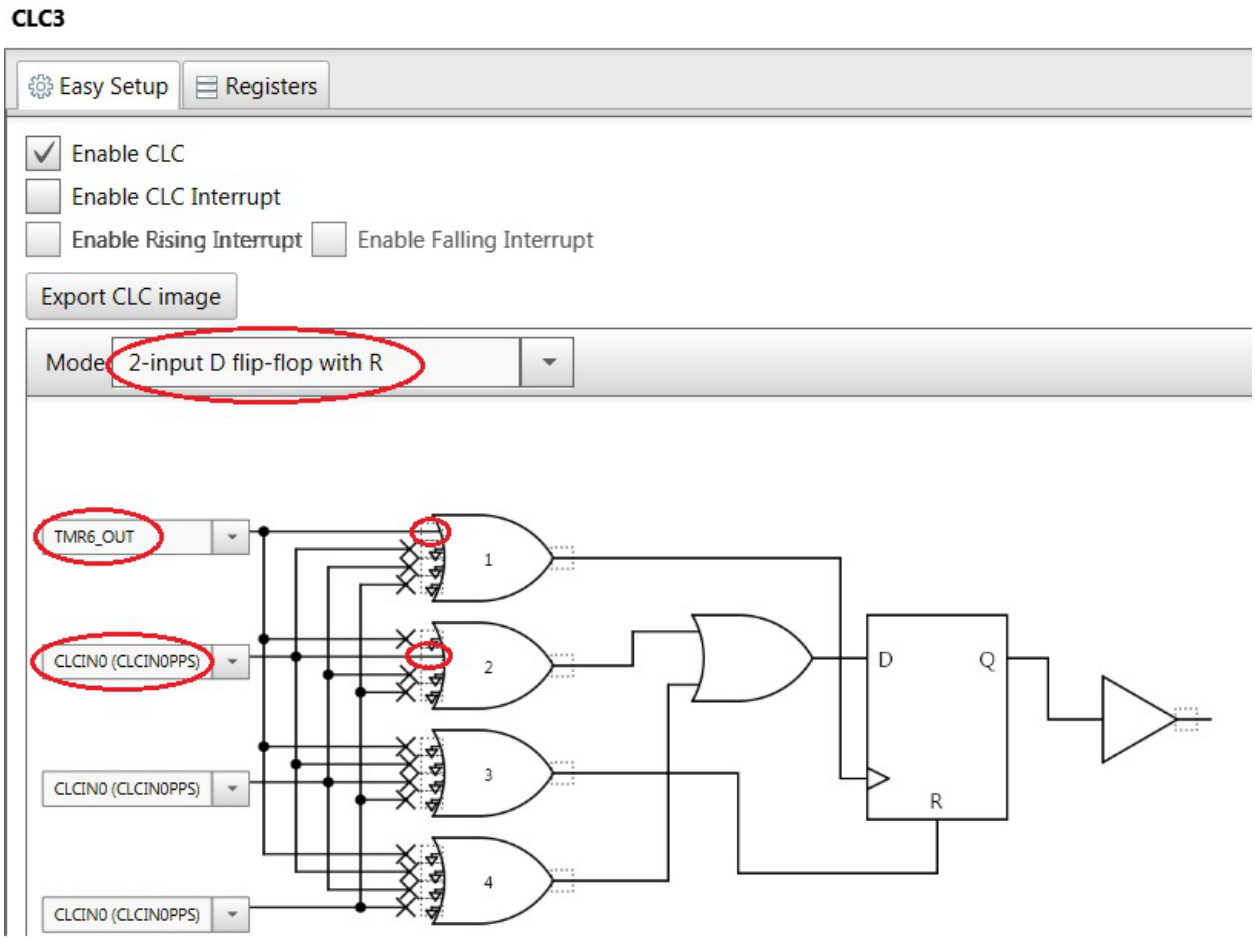
Figure 3-3. TMR6 Overflow at 1.5 ms Frequency



3.5.3 CLC3

Configure *CLC3* as shown in Figure 3-4. Select *2-input D flip-flop with R* in the *Mode* field. Select Timer6 overflow signal *TMR6_OUT* as the clock to the D Flip-Flop register. Select *CLCIN0* as the input signal. Connect these two signals to the CLC logic gates *1* and *2* respectively as highlighted in the red circle in Figure 3-4.

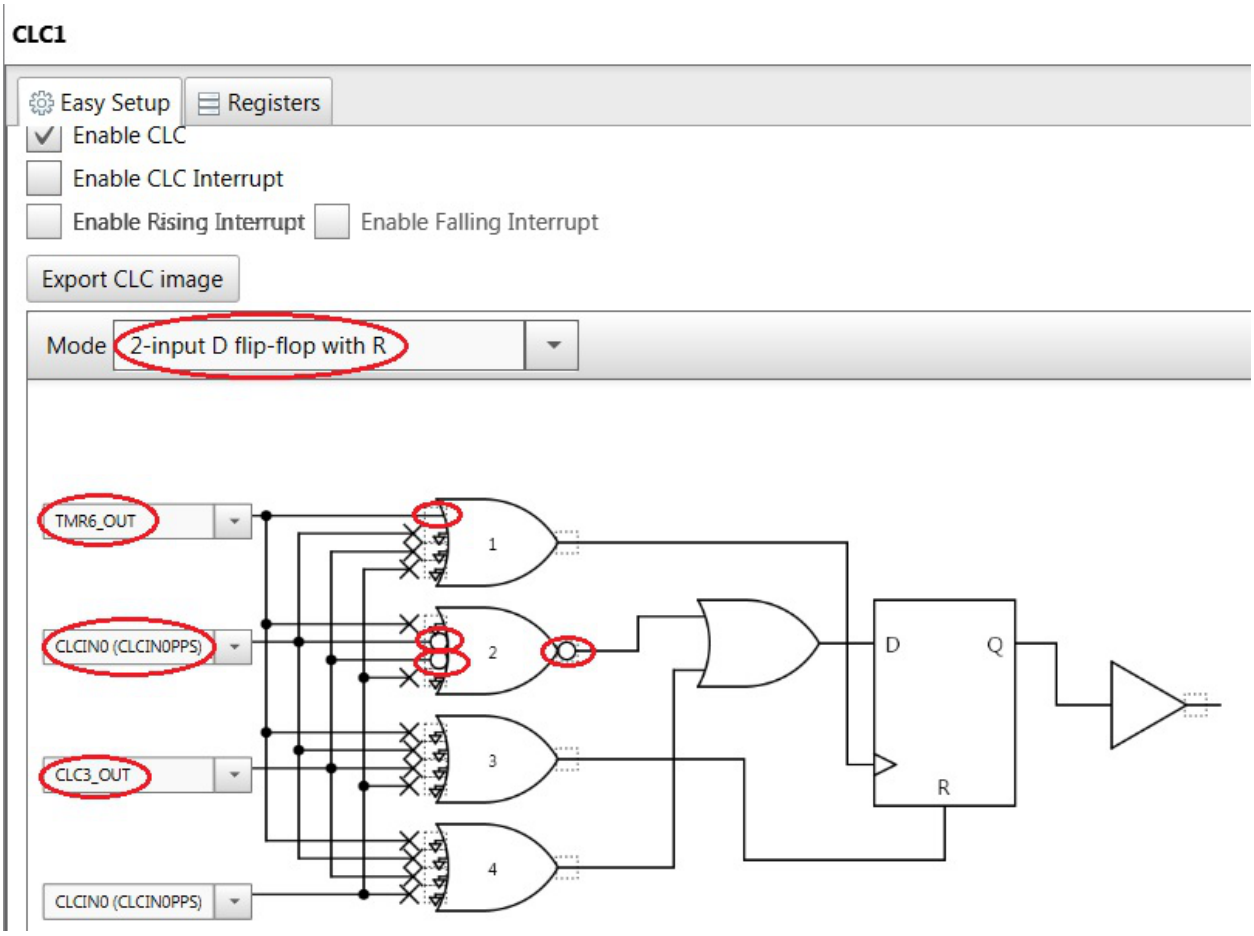
Figure 3-4. CLC3 Configuration



3.5.4 CLC1

Configure CLC1 as shown in Figure 3-5. Select *2-input D flip-flop with R* in the *Mode* field. Select *TMR6_OUT* as the clock signal to the D Flip-Flop register. Select *CLC3_OUT* (the output signal of *CLC3*) and logical AND-it with the default *CLCIN0* (the input signal of *CLC3*) as an input signal to *CLC1*. Connect the *TMR6_OUT* signal to the CLC logic gate *1* as highlighted in the red circle shown in Figure 3-5. Connect *CLC3_OUT* and *CLCIN0* to the CLC logic gates *2* and invert both the input and output signals of the CLC logic gates *2* as highlighted in the red circles.

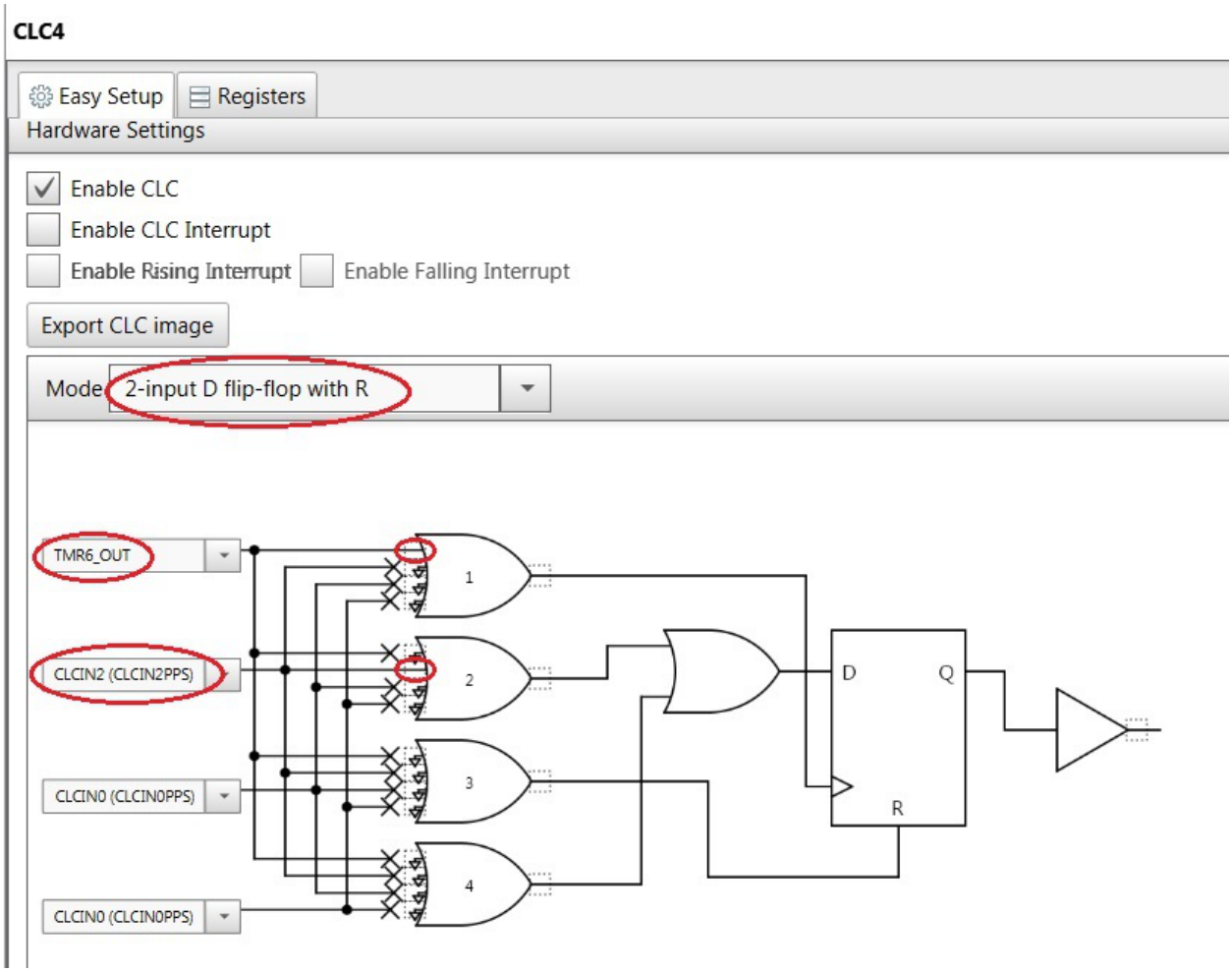
Figure 3-5. CLC1 Configuration



3.5.5 CLC4

Configure *CLC4* as shown in Figure 3-6. Select *2-input D flip-flop with R* in the *Mode* field. Select the *TMR6_OUT* overflow signal as the clock to the D Flip-Flop register and select *CLCIN2* as the input signal. Connect these two signals to the CLC logic gates *1* and *2* respectively as highlighted in the red circles in Figure 3-6.

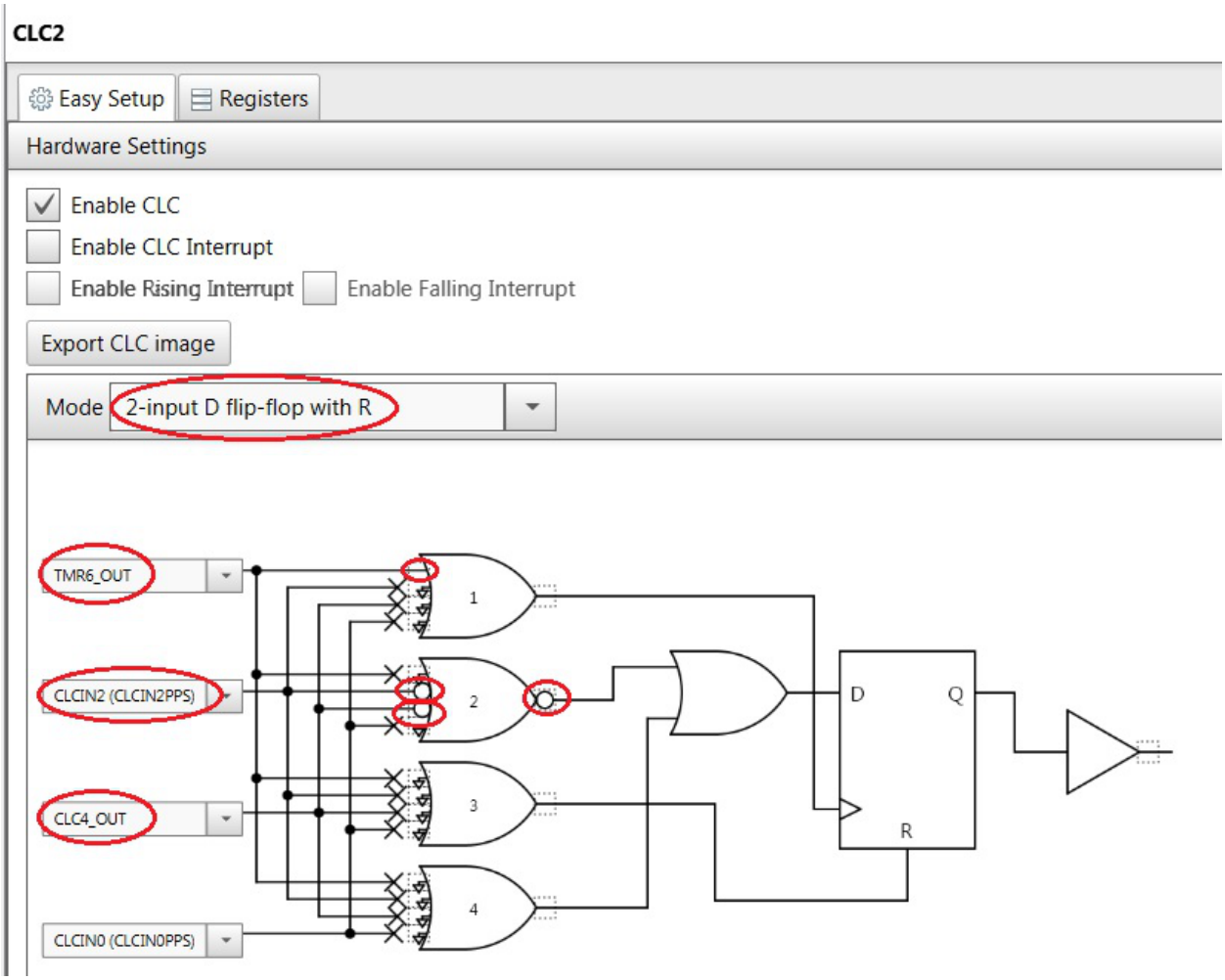
Figure 3-6. CLC4 Configuration



3.5.6 CLC2

Configure CLC2 as shown in Figure 3-7. Select *2-input D flip-flop with R* in the *Mode* field. Select *TMR6_OUT* as the clock to the D Flip-Flop register. Select *CLC4_OUT* (the output signal of *CLC4*) and logical AND-it with the default *CLCIN2* (the input signal of *CLC4*) as an input signal to the *CLC2*. Connect the *TMR6_OUT* signal to the CLC logic gate *1* as highlighted in the red circle shown in Figure 3-7. Connect *CLC4_OUT* and *CLCIN2* to the CLC logic gates *2* and invert both the input and output signals of the CLC logic gates *2* as highlighted in the red circles.

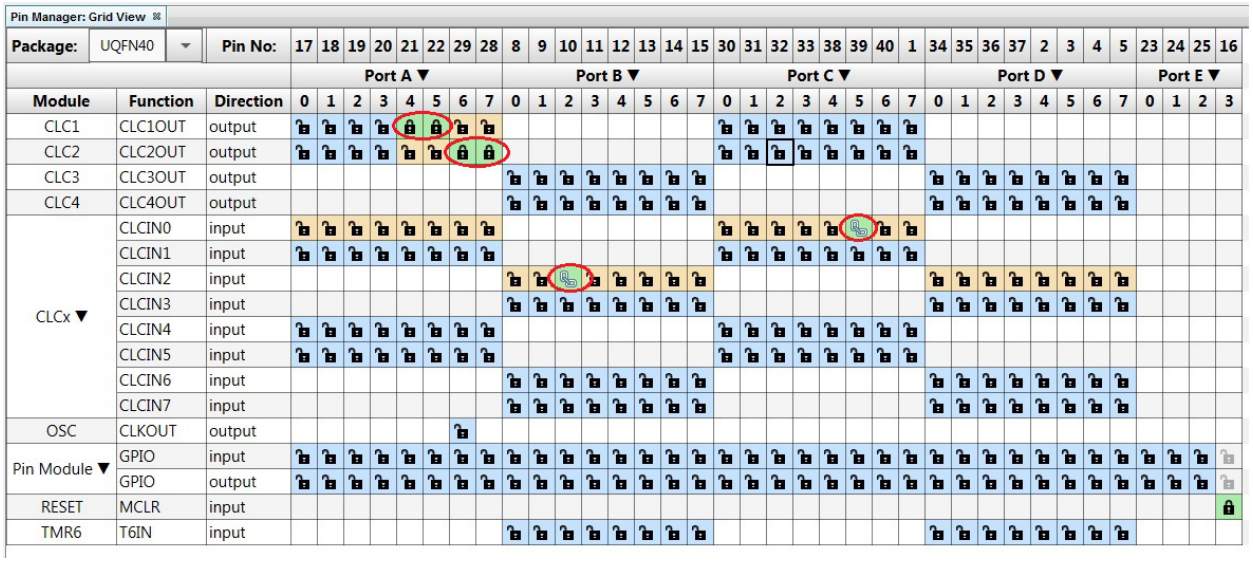
Figure 3-7. CLC2 Configuration



3.5.7 PINS

Click *Pin Manager: Grid View* to configure the input/output pins. Click *RC5* and *RB2* as input pins of *CLCIN0* and *CLCIN2* respectively, as highlighted in the red circles in Figure 3-8. Configure *CLC1* output to pins *RA4* and *RA5* as highlighted in the red circle. Configure *CLC2* output to pins *RA6* and *RA7* as highlighted in the red circle. Pins *RA4*, *RD5*, *RA6*, and *RA7* are connected to the HPC board LEDs D2, D3, D4, and D5 respectively. *RB2* will be manually connected to the external rotary button, while *RC5* is already connected to the onboard button *S2*. The connection between the pins and the CLCs has been illustrated in Figure 2-1. The *CLC1* output has been used to light the LEDs *D2* and *D3*. The *CLC2* output has been used to light the LEDs *D4* and *D5*.

Figure 3-8. Pin Manager: Grid View



3.6 Generate Code and Program Device

Click *Generate* in the upper left sub-window in MPLAB to generate code. The generated code is the final code since this is a software-free solution.

As already described, since both the timer and CLCs can run in sleep mode, one can set the device in sleep mode in order to minimize the power consumption. This is done by inserting the following line of code after the system initialization in the *main()* function of the *main.c* file. However, this is optional.

```
SLEEP();
```

Build the project with short-cut key *F11* and make sure no error occurs. Then click the *Make and Program Device Main Project* icon on the top toolbar in MPLAB to program the device. Now it is ready to detect the onboard LEDs *D2* and *D3* toggling when the onboard *S2* button is pressed/released. The onboard LEDs *D4* and *D5* should toggle when the external rotary encoder connected to the input pin *RB2* gets rotated. It is also recommended to use an oscilloscope to monitor how the debouncing is performed when the noise pulses are filtered out.

The user should now be able to reproduce the solution by following the above configurations.

3.7 3-CLCs Solution Configuration in MPLAB

This section describes how the existing project is reconfigured in MCC to the 3-CLCs Solution as shown in Figure 2-4. As shown, the first stage CLC, *CLC3*, is unchanged. The third CLC, *CLC5*, needs to be configured and the second CLC, *CLC1*, needs to be reconfigured with modification.

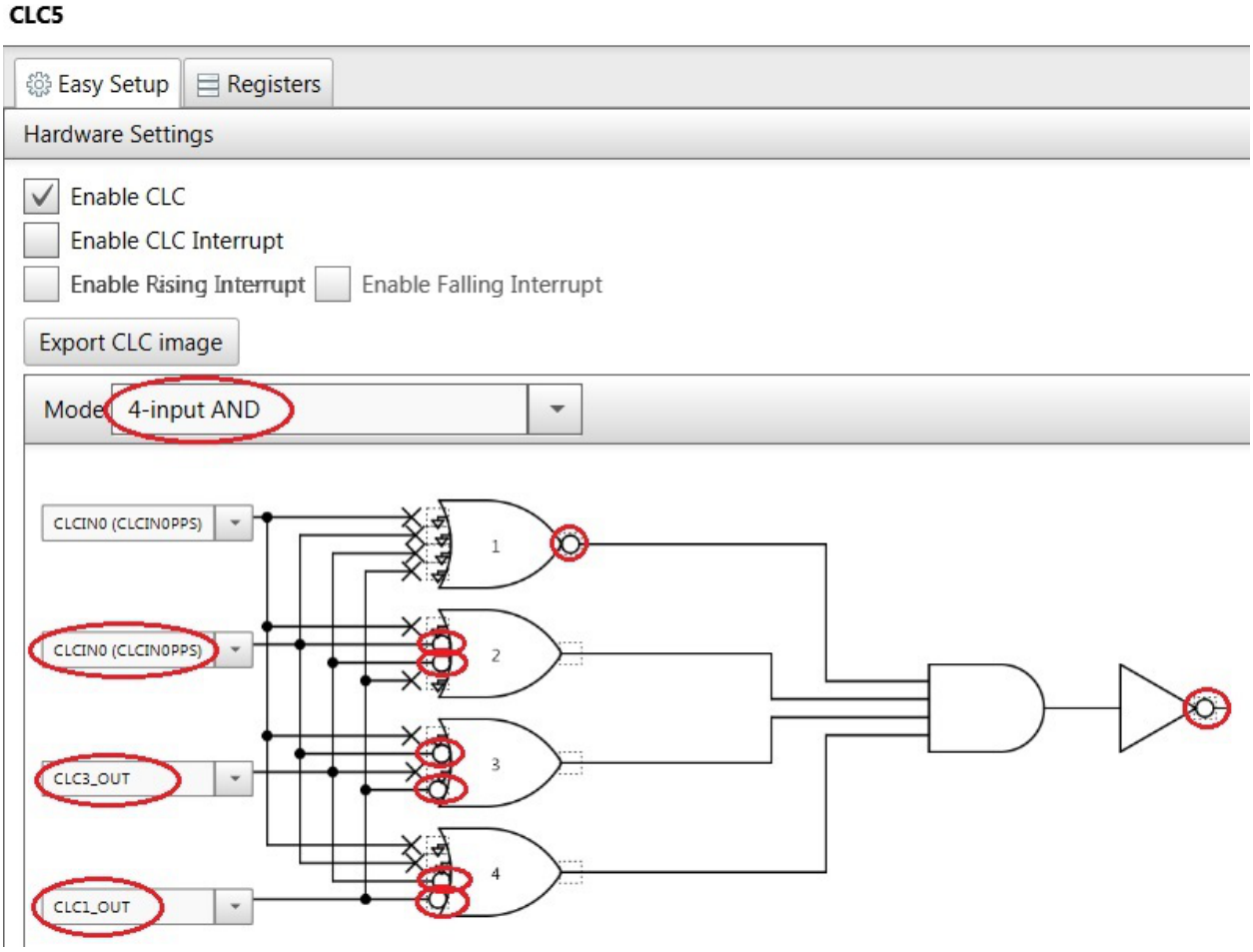
In MPLAB X IDE, click on the *MCC* logo on the top menu bar for code reconfiguration. On the left side, double-click *CLC5* from the *Device Resources* section to add them to the *Project Resource* section.

3.7.1 CLC5

Configure CLC5 as shown in Figure 3-9. Select 4-input AND in the Mode field. Select the three input signals as highlighted in the red circles, the default *CLCIN0*, *CLC3_OUT*, and *CLC1_OUT*. The CLC logic gate 1 has output NOT as highlighted in the red circle. Make sure logic gate 2, 3, and 4 has the correct

two inputs connected and also inverted. The four signals are then logically ANDed with the value inverted as output. The CLC5 logic functions as pair ANDed of the three inputs and then ORed.

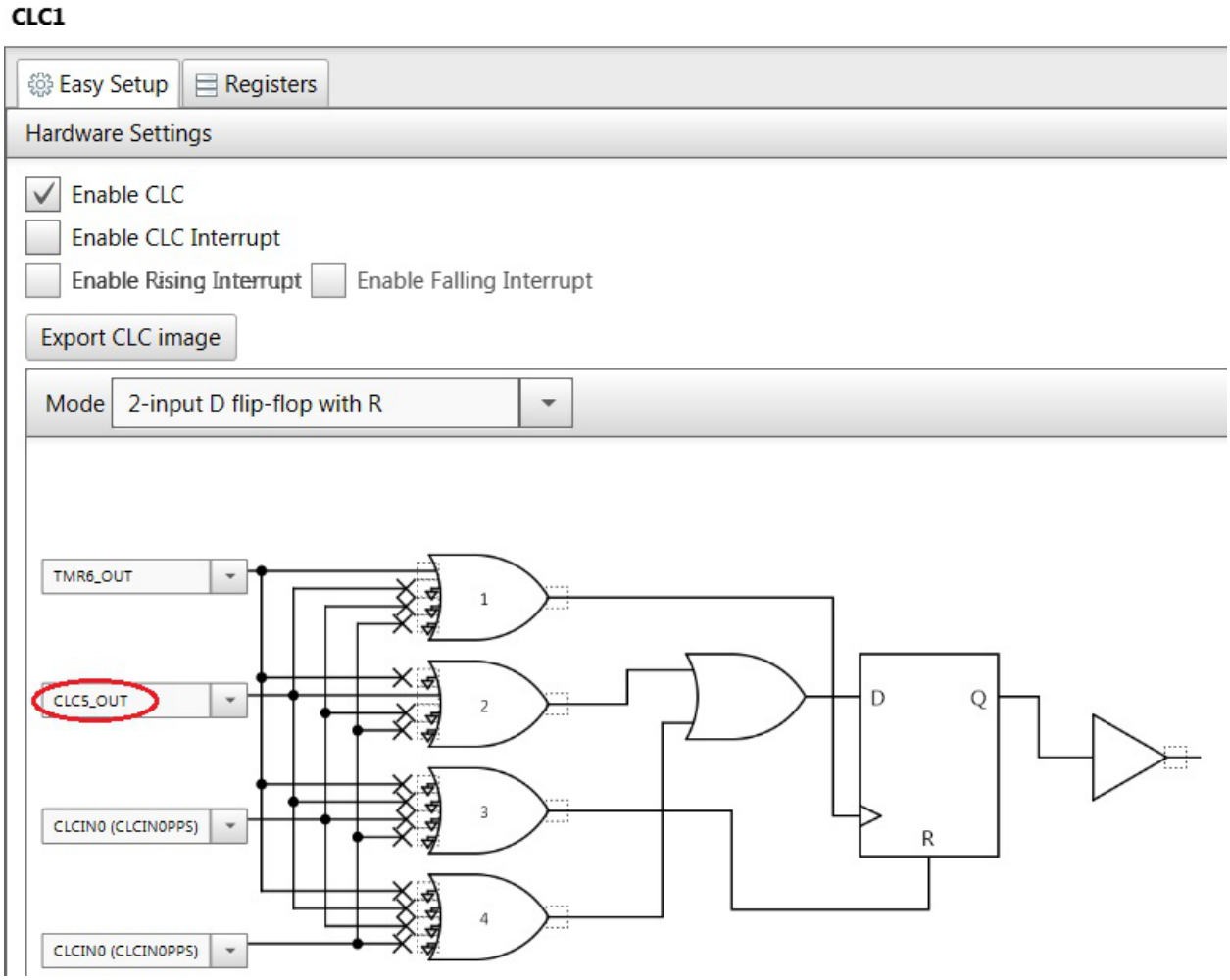
Figure 3-9. CLC5 Configuration



3.7.2 CLC1

Reconfigure *CLC1* as shown in Figure 3-10. Deselect all the invert before and after on logic 2. Deselect *CLC3_OUT* as input and select *CLC5_OUT* as the only input signal to logic 2, as highlighted in the red circle.

Figure 3-10. CLC1 ReConfiguration



Click *Generate* in the upper left sub-window in MPLAB to regenerate the code. The generated code is the code for the 3-CLCs solution. Rebuild the project with short-cut key *SHIFT+F11* and then click the *Make and Program Device Main Project* icon on the top toolbar in MPLAB to reprogram the device. The 3-CLCs solution is now applied to the connected external rotary encoder.

4. Revision History

Doc. Rev.	Date	Comments
A	06/2019	Initial document release

The Microchip Website

Microchip provides online support via our website at http://www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to http://www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the web site at: http://www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Trademarks

ISBN: 978-1-5224-3644-7

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit http://www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: http://www.microchip.com/support Web Address: http://www.microchip.com **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **Chicago** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **Dallas** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **Detroit** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4450-2828 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-72400 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-72884388 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820