July 2021

CCS Inc

www.ccsinfo.com
262-522-6500

<BITS & BYTES> Newsletter

INSIDE THIS ISSUE:

Product Spotlight

LONG RANGE RF DEVELOPMENT KIT

The Long Range RF Development Boards can have a 9 mile RF link with a PIC® MCU and LoRaWAN®. The included module is compatible with LoRaWAN® networks. This development kit includes the powerful PCWH Integrated Development Environment with compiler support for Microchip's PIC® PIC10, PIC12, PIC16 and PIC18 families and an ICD-U64 in-circuit programmer/debugger that supports C-aware real time debugging. Long range communications drivers are included with the compiler.

support@ccsinfo.com

LoRaWAN® is a mark used under license from the LoRa Alliance®. Use of the LoRa Alliance® and LoRa Alliance® Member marks is pursuant to license from the LoRa Alliance®

sales@ccsinfo.com

The Easy Way to Configure Oscillators
By: Mark Siegesmund

Easily setup your PIC® MCU clock with #use delay(). This article takes a deeper look at how to set the various oscillator fuses and oscillator registers to get the clock speed specified in #use delay(). Projects get a jump start with #use delay() accepting simple keywords: CRYSTAL, OSCILLATOR, INTERNAL and RC in addition to, or in place of, the keyword CLOCK to setup the clock.

By using one of these keywords with the clock, speed causes the compiler to set the appropriate oscillator related configuration fuses and registers to automatically achieve the specified speed with the specified clock source. The following #use delay() examples demonstrate how to set up the PIC® MCU to run at 8MHz from an external crystal and at 4MHz from the internal oscillator:

```
#use delay(clock=8MHz, crystal)
#use delay(crystal=8MHz)        //same as above line

#use delay(clock=4MHz, internal)
#use delay(internal=4MHz)       //same as above line
```

Devices that have a PLL, the #use delay() has an easy mechanism for setting up the PIC® MCU to run from use of it. This is easily done by setting the CLOCK option to the PLL clock speed and the CRYSTAL or OSCILLATOR option to the speed of the external clock source. For example, the following #use delay() line will set the clock to run at 32 MHz from an 8 MHz external oscillator using the PLL:

```
#use delay(oscillator=8MHz, clock=32MHz)
```

Devices with an USB peripheral, CCS has added the keywords USB, USB_FULL (same as USB) and USB_ LOW. By using one of these keywords in #use delay() it causes the compiler to set the device's configuration fuses for the USB clock to be setup for either FULL or LOW speed. For example, the following line will set up the PIC® MCU to run at 48MHz from a 20MHz crystal and setup the USB clock for FULL speed:

```
#use delay(crystal=20MHz, clock=48MHz,  USB_FULL)
```

By default, the compiler turns off the clock output that is available for some oscillator modes. If your application requires the clock output turned on, do something like this:

```
#use delay(internal=4MHz, clock_out)
```

When using the #use delay() to configure your oscillator it is not recommended to include oscillator related #fuses in your code or the use of setup_oscillator() unless you need to change the oscillator run-time.

Capture Event Timers with the CCS C Compiler
By: CCS Staff

The Capture Compare (CCP) peripheral of the PIC16 and PIC18 PIC® MCUs and the Input Capture (IC) peripheral of the dsPIC and PIC24 PIC® MCUs allow the MCU to capture and hold a timer value when a specific input pin reaches a user defined state. This peripheral is useful for measuring the duration between events, or determining the frequency or duty cycle of an incoming signal. This peripheral operates independent of the MCU operation, meaning it can trap the timer on the input signal without the developer having to block code execution waiting for the input signal to happen.

Version 5 of the CCS C Compiler has added a #use capture() library that makes it easy to use the CCP or IC peripheral. The API for this library provides an easy to use method for using and configuring the peripheral and timers, while also being portable from one PIC® MCU device to another. Here is an example of its use:

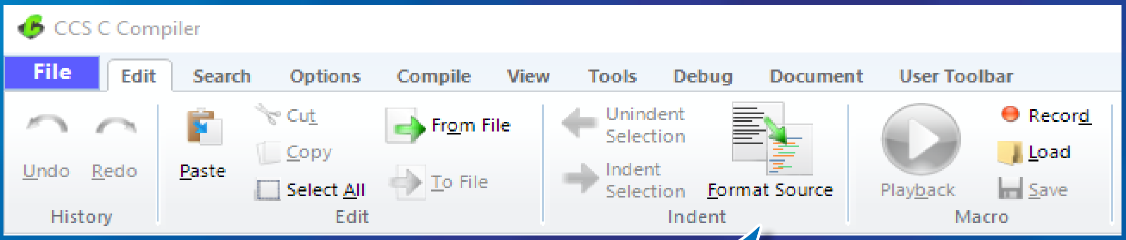
```
#use capture(CCP1, input=PIN_C2, capture_rising, stream=CAPTURE1)

unsigned int16 WaitForEvent(void)
{
while(!get_capture_event(CAPTURE1));
return(get_capture_time(CAPTURE1));
}
```

#use capture() configures a CCP or IC peripheral for use, and by using the STREAM option several peripherals can be configured with several instances of #use capture(). #use capture() also provides a method of configuring the timer peripheral to use and the tick rate of the timer. Several other configuration options exist, see the compiler help manual for documentation.

get_capture_event() returns TRUE if the CCP or IC peripheral has trapped an event that can be read, and get_capture_time() returns the time the event happened.

The CCS C Compiler provides an example of using this library, ex_use_caputure.c which can be found in the examples directory of the CCS C Compiler. This example can be used on any PCM, PCH or PCD version 5 CCS C Compiler.

CCS COMPILER FEATURE FOCUS



Forgot to indent your code? Use the Format Source Button!

Our Automatic Source code formatting is a great tool in the CCS C Compiler! The tool is intelligent enough to look over your code and indent it in an easy-to-read format!

USB Development Kit

Includes

- USB Prototyping Board
- In-Circuit Debugger/ Programmer
- Exercise Tutorial
- 9V AC Adapters and Cables

sales@ccsinfo.com
262-522-6500 EXT 35
www.ccsinfo.com/NL521

Summer is for C Compiler Savings

$25 Off a Full Compiler or Compiler Maintenance

Use Code:Summer2021

More than 25 years experience in software, firmware and hardware design and over 500 custom embedded C design projects using a Microchip PIC® MCU device. We are a recognized Microchip Third-Party Partner.

www.ccsinfo.com

Follow Us!