October 2020

www.ccsinfo.com
262-522-6500

<Bits & Bytes> Newsletter

PRODUCT SPOTLIGHT

LOAD-n-GO

Our LOAD-n-GO Programmer is the ultimate Solution for Programming Productivity. Simply load the program into the Battery Powered device with a USB, connect to the target chip and program. IT'S THAT EASY!

LoRa – Low Power, Long Range RF Protocol

LoRa is low-power wide-area network protocol that can be used to periodically communicate sensor data. LoRa uses a proprietary spread spectrum modulation that is similar to and a derivative of Chip Spread Spectrum (CSS) modulation. This allows for long-range transmission, up to 15 km, with low power consumption. This makes it ideal for battery applications that only need to periodically transmit sensor data.

There are two types of networks that can be built using LoRa modulation. The first is a peer-to-peer or point-to-point network. This is the simplest network that can be created only requiring a minimum of two devices with LoRa radios to develop. In this type of network, each device's radio must set to use the same LoRa settings frequency, coding rate, spreading factor, etc. This type a network is good for small networks with only a few devices that periodically need to transmit data to each other.

The second network type of network described in this article using LoRa modulation is a LoRaWAN network. LoRaWAN is a upper level network protocol, developed by the LoRa Alliance, that uses LoRa modulation for creating a cloud-based network. In LoRaWAN networks end-devices periodically send messages to a gateway, which then forwards to a network server, which forwards it to an application server. Any response is then forwarded back in the reverse order. In order for a device to communicate on a LoRaWAN network it must first join that network, additionally for security purposes the data being sent and received on a LoRaWAN network is encrypted twice. First, the application payload is encrypted with an application key that is only known by the device and application server. Second, the network payload, which includes the application payload, is encrypted with a network key that is only know by the device and the network. This ensures that the data being sent by the end-device to the application is secure.

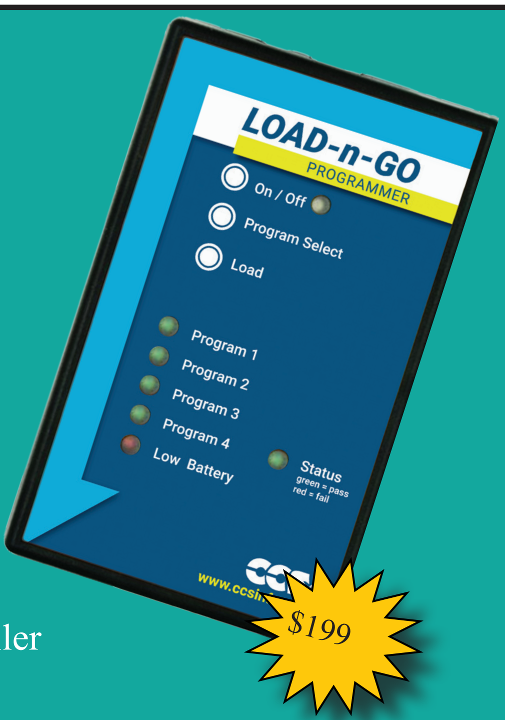
When developing a LoRaWAN network it requires at least one end-device with a LoRa radio, a gateway, a network server and an application server. However, it is possible to purchase or build a gateway that have the network and application servers built-in to it. Additionally there are several service providers, The Things Network for example, that have public gateways operating in many areas of the world that can be used to receive messages and forward them to your application server. See thethingsnetwork.org for more info about their service. This allows building a LoRaWAN network quickly without investing a lot of time and money in developing and maintaining the network infrastructure. However these services are not required, it is possible to build a private network from the ground up using your own gateways and servers.

Devices that communicate using LoRa modulation requires a LoRa radio, for that purpose Microchip developed the RN2903 module. The RN2903 modules is designed to communicate using LoRa modulation using the US frequency band, additionally the RN2903 module has a built-in stack for connecting to and communicating on a LoRaWAN network.

CCS created a driver for the RN2903 module to assist in development, rn2903.c , for communicating with and controlling the module. The rn2903.c driver is a low level driver that uses serial messages to communicate with and control the RN2903 module. It also has specific functions for controlling RN2903 module's radio, which are used for doing peer-to-peer, point-to-point, communication, and for controlling the RN2903 module's LoRaWAN stack, which is used for connecting to and communicating on a LoRaWAN network. In addition to the rn2903.c driver, CCS also created two additional drivers, lora.c and lorawan.c. The lora.c driver was developed by CCS to use the rn2903.c driver to do peer-to-peer, point-to-point, communication. This is designed so that two or more PIC® MCU devices with RN2903 modules can communicate with eachother. The lorawan.c driver was developed by CCS to use the rn2903.c driver to connect a PIC® MCU device with an RN2903 module to a LoRaWAN network. All three of these driver come with the the current version of the CCS C Compiler. CCS will be releasing a LoRa development kit in the near future.

LOAD-n-GO Programmer

Quick ICSP
Field Programming!



* ICD & ICSP functionality with CCS C-Aware IDE Compiler
* Supports all Microchip PIC® MCU or Flash Devices
* Standalone Programmer or Power & Connects with USB

Order Today!

SENT- Hot New Wired Protocol

The SENT (Single Edge Nibble Transmission) protocol is a relatively new protocol used in the automotive industry for transmitting sensor data to a controller. The SENT protocol is a one-way asynchronous voltage interface which requires three wires, a signal line, a supply voltage line and a ground line. SENT uses pulse width modulation to encode 4 bits (1 nibble) of data per symbol.

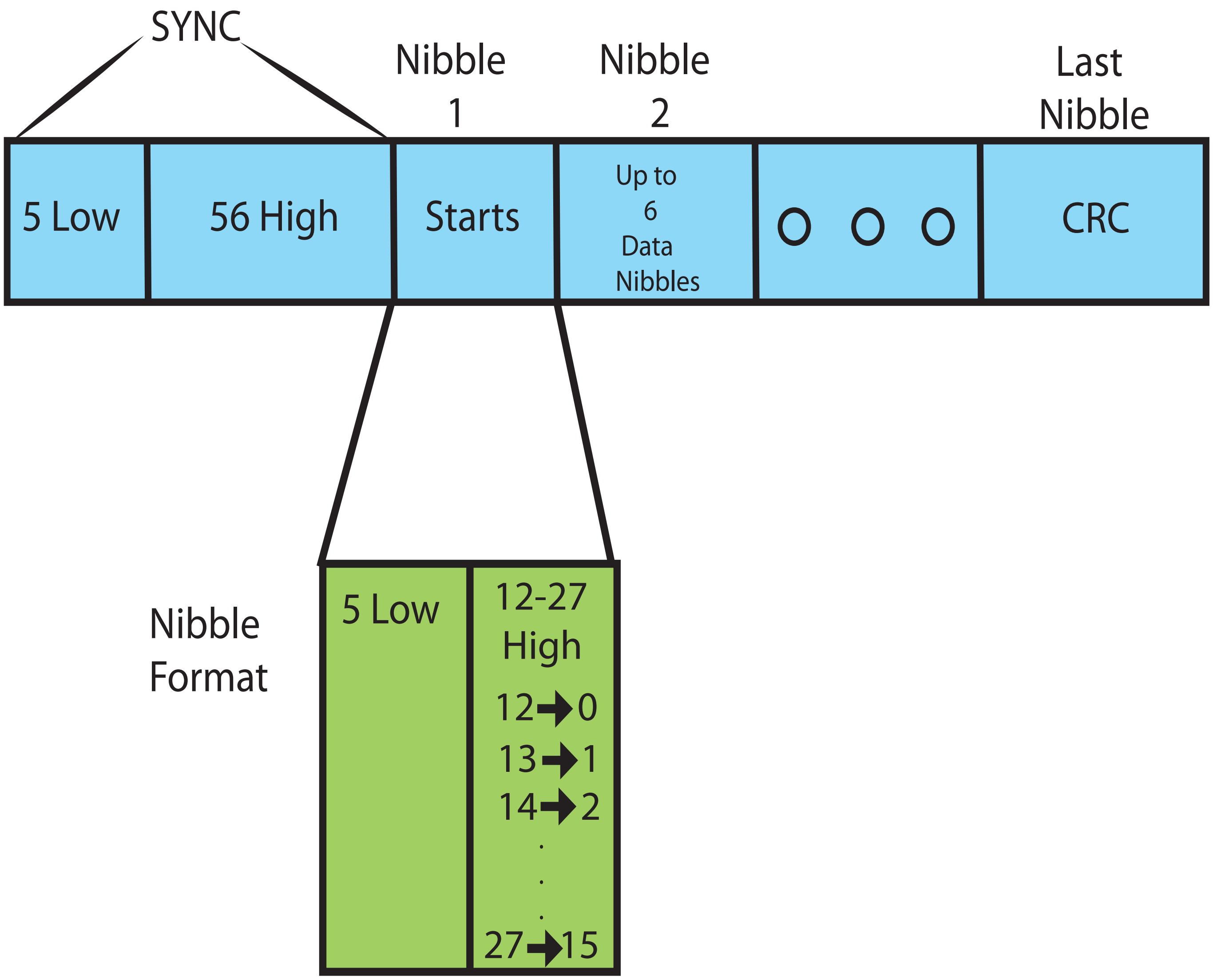
The basic unit of time in SENT is a tick, which can be between 3-90us. Each SENT message contains a sync period, made up of 5 low ticks and 56 high ticks, followed by up to 8 nibbles of data. Each data nibble is transmitted with a fixed-width low period of 5 ticks, followed by a variable-length high period from 12 to 27 ticks representing a nibbles value from 0 to 15. The data portion of the SENT message is make up of a status nibble, 1 to 6 nibbles of sensor measurements and a CRC nibble. Additionally, an option pause pulse can be used to compensate for variable message lengths.

Some of Microchip's newer PIC® microcontrollers, the dsPIC33EV256GM106 family for example, comes with a build-in SENT peripheral for transmitting or receiving SENT messages. The CCS C Compiler has build-in functions for setting up and using the SENT peripheral For these devices. The devices that have a SENT peripheral, the following functions will be available in the compiler for setting up and using the SENT peripheral: setup_sent(), sent_putd(), sent_getd() and sent_status().

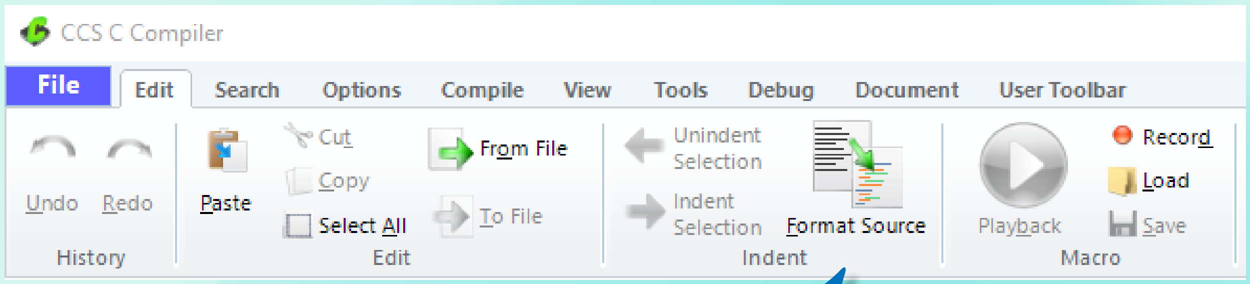
The setup_sent() function is used to setup the SENT peripheral and has a parameter for setting up as an asynchronous SENT transmitter, synchronous SENT transmitter or a SENT receiver and the number of data nibbles to sent or receive. Other settings include whether to use the pause pulse and

The sent_putd() function is used to load the data nibbles to transmit when the peripheral is setup as a SENT transmitter, and the sent_getd() function is used to retrieve the data nibbles when the peripheral is setup as a SENT receiver. Finally, the sent_status() function is used to get the status of the SENT peripheral, for example what nibble it is currently transmitting or receiving.

The CCS C Compiler also provides two examples ex_sent_transmitter.c and ex_sent_receiver.c showing how use the SENT peripheral and built-in functions. ex_sent_transmitter.c is an example demonstrating how to setup and use the SENT peripheral as an asynchronous SENT transmitter, and ex_sent_receiver.c is an example show how to setup and use the SENT peripheral as a SENT receiver.



CCS COMPILER FEATURE FOCUS



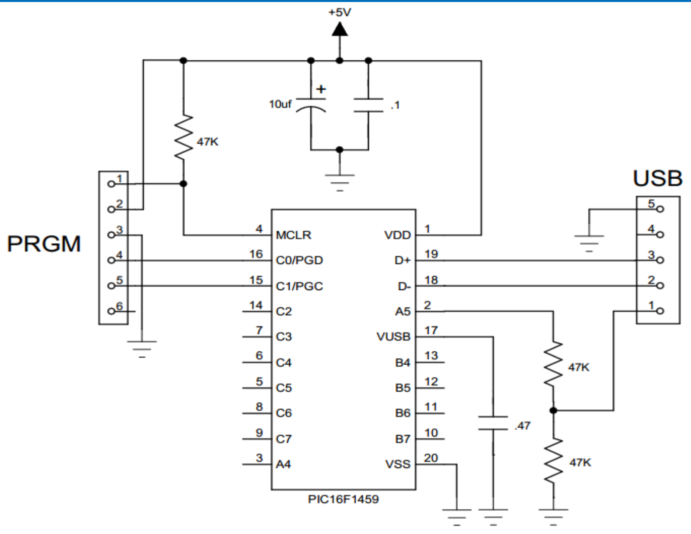
Forgot to indent your code? Use the Format Source Button!

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!

Easy USB

The CCS C Compiler from the beginning has made it easy to communicate over an RS232 like port. Many of our users have used this extensively, not only for communicating with serial devices but also for diagnostics and debugging. Now, PC's and many other devices have replaced their RS232 ports with USB ports. The USB hardware and software is much more complex than RS232. This has discouraged many from migrating over or they have used a crutch like an external chip to do the USB magic (like FTDI). This article shows how to easily add a USB port to your PIC^R MCU application.

HARDWARE
Microchip has a number of chips with built in USB. For example, the PIC16F1459 ($1.38/100) or PIC18F14K50 ($1.79/100). Here is an example schematic:
(note: pin numbers apply to PDIP, SOIC, and SSOP packages)



Shown here is a USBmicro style connector. The more common type B connector is the same pinout except there is no pin 5 and 4 is the ground. The D+ and D- pins are for the data and sometimes there will be a 27 ohm series resistor and/or zener protection diodes on those pins. When using a peripheral device, pin 1 will supply 5 volts (up to a half amp). The board can be powered from that 5 volts; however, in this schematic it is to simply detect if the USB cable is plugged in. Although users can do that from the data lines, it is easier to detect the 5 volts like this. Sometimes users will put series coils on the 5 volts and ground to reduce noise.

The Vusb on this chip simply needs a cap for an internal voltage regulator. Some chips have a dedicated Vbus pin for the 5 volts detect. Careful with the pin names, as they are not consistent between chips.

HOST SOFTWARE

The USB bus has a number of protocols that can be used, each with many options and configurations. HID (human Input device) is used for keyboards and mice and is easy to use on any OS because the drivers are built in. There is a data limit however, (like 8 bytes per ms) that makes it impractical for many applications. CDC is a protocol designed to emulate an RS232 port. The Windows drivers will create a virtual COM port for a CDC USB device so the PC application can use COM1... just like an RS232 port.

When a Windows 10 sees a CDC device, it automatically installs a driver for it. For older versions of Windows, users need a short .inf file to describe their device and include the device VID (vendor ID) and PID (product ID). Examples .inf files are in the CCS C Compiler examples directory. Every USB device is supposed to have a unique VID/PID and serial number. If users have two of the same devices plugged in, the VID/PID will match but they are differentiated by serial number. To get a VID users need to register with the USB standards organization.

USB is point to point and one point is a host and the other a device. Hub's can also be involved but that is beyond our concern for this article. The host initiates all activity. For example, a PC is a host and it will poll every device about once a millisecond and transfer any data needing transferring. When the device is first plugged into a port, there is a handshaking that takes place that involves the device identifying itself along with the protocol it will use and various parameters. For example, how much current it expects to draw off the bus. This handshake is called enumeration. In Windows users know it happened by the beep.

Some PIC24 parts have USB hardware that can also be a host. For example, this might be used to read a USB flash drive.

PIC^R MCU SOFTWARE
The CCS C Compiler has several example programs for CDC as well as the required drivers. The following are the key declarations you need for a USB CDC program:

```
#include <16F1459.h>
#use delay(internal=48MHz,USB_FULL,ACT=USB)

#define USB_CON_SENSE_PIN  PIN_A5   // Connected to USB +5V

#define USB_STRINGS_OVERWRITTEN

#define USB_CONFIG_VID 0x2405  // CCS VID
#define USB_CONFIG_PID 0x8001

#define USB_DESC_STRING_TYPE 3
#define USB_STRING(x)  (sizeof(_UNICODE(x))+2),  \
                 USB_DESC_STRING_TYPE,_UNICODE(x)
#define USB_ENGLISH_STRING 4,USB_DESC_STRING_
TYPE,0x09,0x04
            //Microsoft Defined for US-English
```

Incoming and outgoing data is buffered and transfer is controlled by the host. The program typically only needs to know if enumerated. To manage all this the USB task needs to be periodically called to do housekeeping. The following is an example main program that simply sends an ADC reading to the host every second:

```
void main(void) {
   setup_adc(ADC_CLOCK_INTERNAL);
   setup_adc_ports(sAN10);  // B4
   set_adc_channel(10);

   usb_init();

   while(TRUE)
   {
      if(usb_enumerated())
```

To see the data start up a terminal program on the PC and select the COM port. In the CCS PCW IDE use TOOLS > Serial Port Monitor. The CDC driver has many more options for more complex programs, however the above function calls plus usb_cdc_getc() not shown here will be all most users need.

Summary
That is all there is to it. Find a suitable chip, add a USB connector and slip some of the above into the program. For high speed transfers there is a USB bulk mode, however the driver interface is more complex and there is no standard. Several companies have free generic USB bulk drivers that can be used, however the customer will need to install those on the target PC. The CDC protocol can transfer data at the RS232 equivalent of 250K baud, twice the speed of RS232 and it may be faster than many PIC's can produce the data.

COVID-19 RESPONSE

During this time of global uncertainty and change, we want to assure you that we are taking every precaution to ensure that we can safely support our customers during this time.

Despite these challenges, CCS staff is continuing to provide technical support, as well as processing orders. It is essential customers have the tools they need to provide the development of existing or new products that may be necessary in the fight of Covid-19.

Many of our existing customers are having to work from home and we want to remind everyone of our Software Licensing Agreement. We pre-register all compilers in a user's name. You can install your compiler on your home PC and laptops. If you do not have access to the registration files and installer, contact customer service for assistance.

Most importantly, as we work together in this unique and rapidly changing environment, we do so with confidence that we will overcome this challenge. Until then, we hold our enduring commitment to the health and well-being of our employees and customers.

Please let us know how we can help you. Stay healthy.

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!