November 2020

www.ccsinfo.com
262-522-6500

<Bits & Bytes> Newsletter

PRODUCT SPOTLIGHT

Rapid USB

Experience the Power of a PIC® MCU in your USB Port! Imagine the possibilities

INSIDE THIS ISSUE:

Advanced Code Profiling

The CCS C Compilers have a unique feature called Data Streaming, where an ICD unit is used as a TTL to USB translator. This can do printf()'s and getc()'s through the programming pins to the PC. Many of our users have been using data streaming not only for debugging, but for diagnostics, factory test and calibration.

Using this same interface, the compilers have the ability to inject code to send data out this port at specific points in the code. This information can include a timestamp, as well as, text data. This forms the infrastructure of the code profiling feature in the IDE.

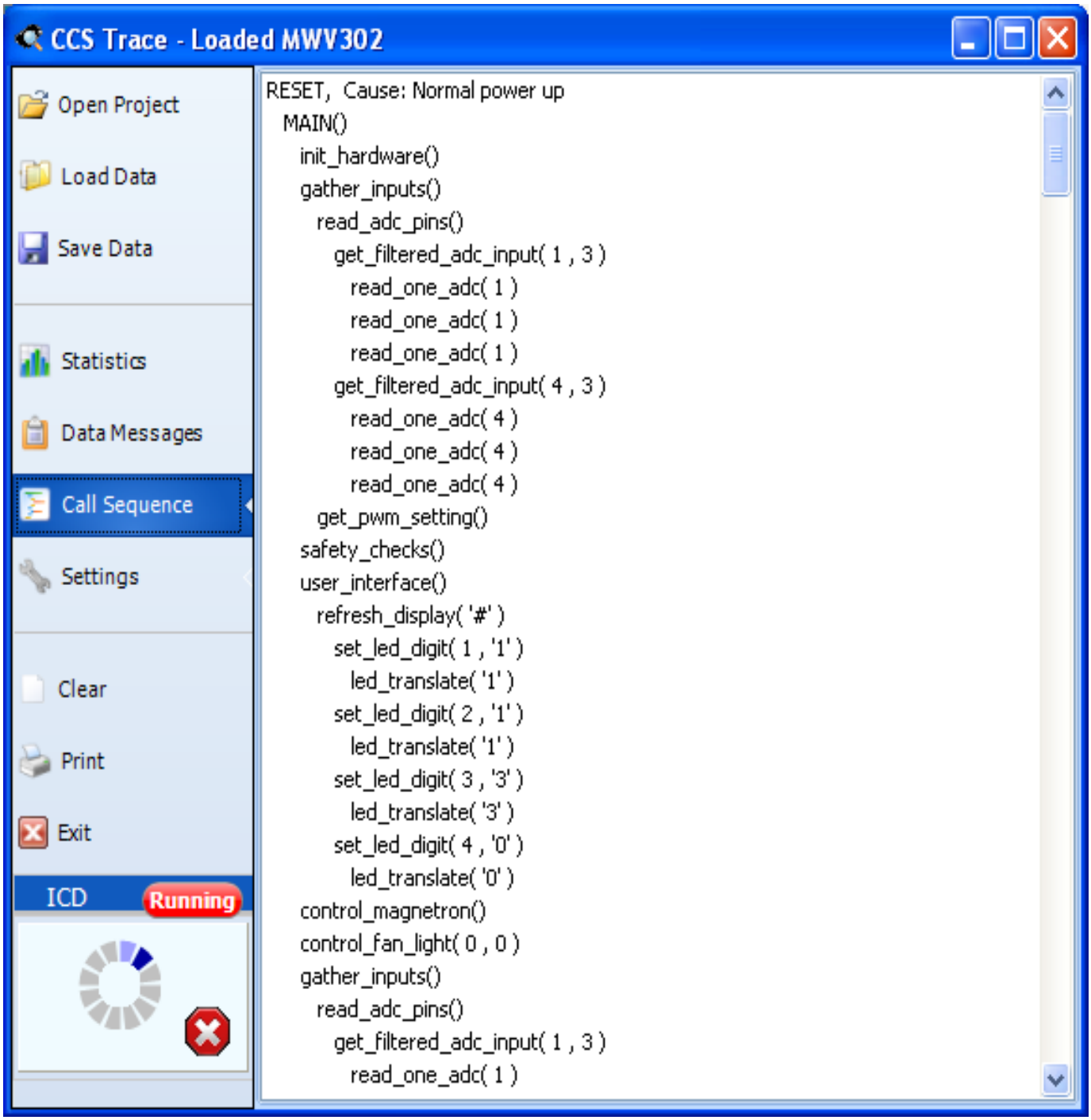
For example, in the code one needs to only add these lines:

```
#use profile( ICD)
#profile  functions
```

and the compiler inserts a code to transmit a tag at the start and end of every function in the program. When the profile tool is then run in the IDE, the following occurs:

	Count	Min	Ave	Max
MAIN()				
init_hardware()	1	143ms	143ms	143ms
gather_inputs()	30	39.7ms	39.9ms	40.0ms
read_adc_pins()	30	18.9ms	19.0ms	19.2ms
get_filtered_adc_input	120	8286us	8331us	8565us
read_one_adc	360	1364us	1079us	1643us
get_pwm_setting()	30	784us	784us	784us
safety_checks()	30	45.7ms	44.2ms	45.9ms
check_ram	30	2098us	360us	2377us
check_over_voltage	30	10.3ms	10.4ms	10.6ms
check_over_current	30	10.3ms	10.4ms	10.6ms
check_idle_conditions()	15	784us	784us	784us
check_rom_crc	30	2624us	2644us	2904us
check_data_EE_crc	30	2363us	2372us	2642us
check_end_of_life()	30	784us	784us	784us
user_interface()	29	37.6ms	37.8ms	37.9ms
refresh_display	29	20.5ms	18.8ms	20.8ms
set_led_digit	117	3956us	3972us	56.6ms
led_translate	117	1310us	2219us	54.0ms
control_magnetron()	29	10.7ms	10.8ms	11.0ms
control_fan_light	29	7839us	7887us	8118us
check_running_conditions()	15	784us	784us	784us

The function tags inserted can optionally include the actual parameters and in the software to get a sequence of events as shown here:



The compiler also allows user defined areas of code to be timed. The user can specify a start and stop event and give the timer a name. A profileout() call is used with text starts with START, followed by something and then another profileout with STOP, and the same something will cause a timer to be created in the software. For example:

```
profileout("Start interpolation algorithm");
y2=((x2-x1)*(y3-y1))/(x3-x1)+y1;
profileout("Stop interpolation algorithm");
```

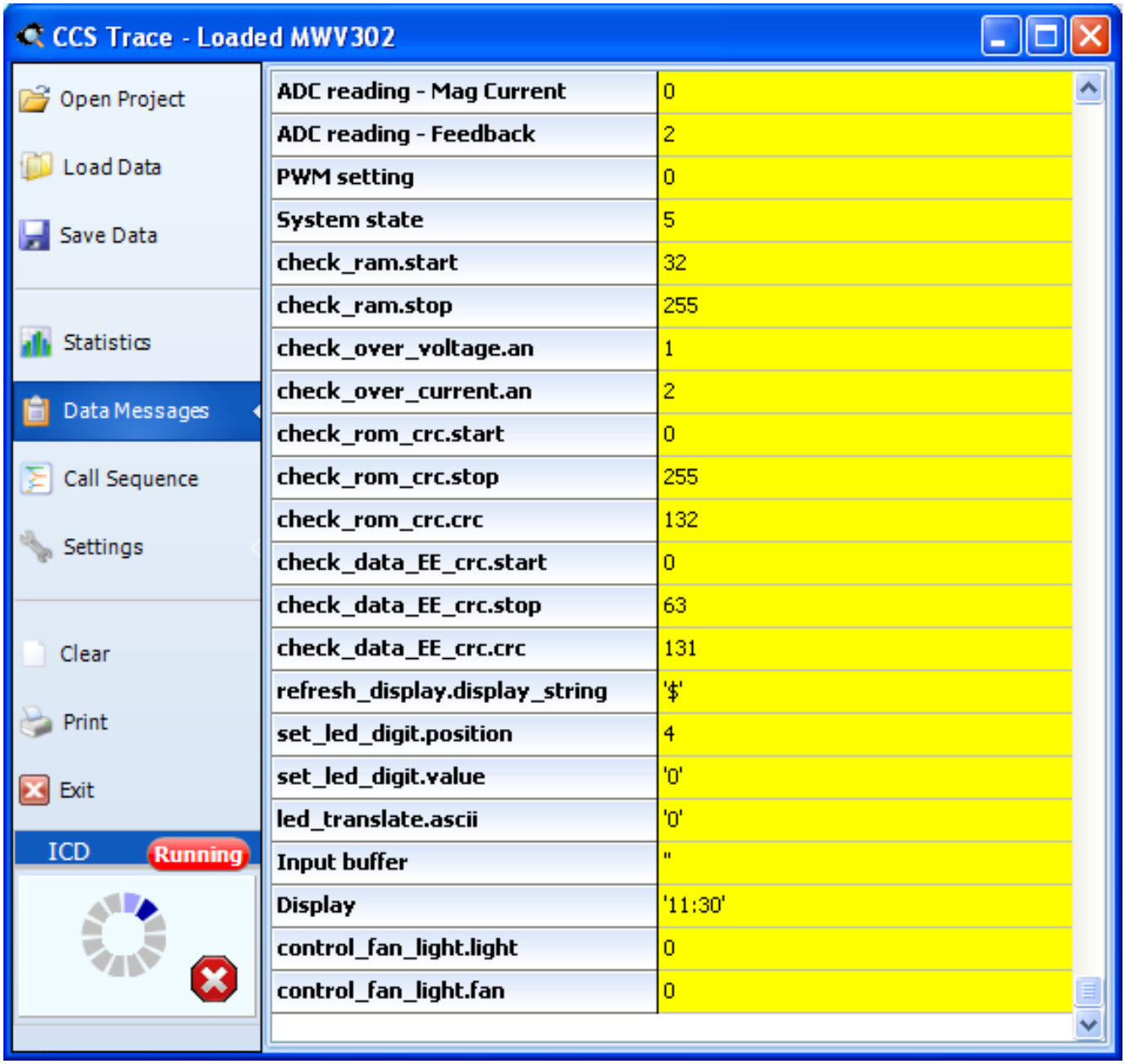
Notice the 4th line down:

	Count	Min	Ave	Max
MAIN()				
init_hardware()	1	143ms	143ms	143ms
gather_inputs()	27	41.4ms	41.6ms	41.7ms
interpolation algorithm	27	929us	940us	1208us
read_adc_pins()	27	18.9ms	19.0ms	19.2ms
get_filtered_adc_input	108	8286us	8328us	8565us
read_one_adc	324	1364us	1047us	54.0ms
get_pwm_setting()	27	784us	784us	784us
safety_checks()	27	45.7ms	44.0ms	45.9ms
check_ram	27	2098us	2109us	2377us
check_over_voltage	27	10.3ms	10.4ms	10.6ms
check_over_current	27	10.3ms	10.4ms	10.6ms
check_idle_conditions()	14	784us	784us	784us
check_rom_crc	27	2624us	693us	2904us
check_data_EE_crc	27	2363us	2374us	2642us
check_end_of_life()	27	784us	784us	784us
user_interface()	27	37.6ms	37.8ms	37.9ms
refresh_display	27	20.5ms	20.6ms	20.8ms
set_led_digit	108	3956us	4462us	56.6ms
led_translate	108	1310us	1804us	54.0ms
control_magnetron()	27	10.7ms	8896us	11.0ms
control_fan_light	27	7839us	7880us	8118us

profileout() can also be used to output the values of variables real-time. For example:

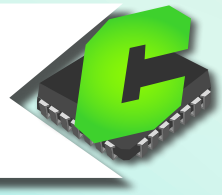
```
profileout("value=", value);        // Sends a variable and a title for the variable
profileout( value );                // Sends a variable and the title is the variable name
```

An example screen showing the profileout() data:

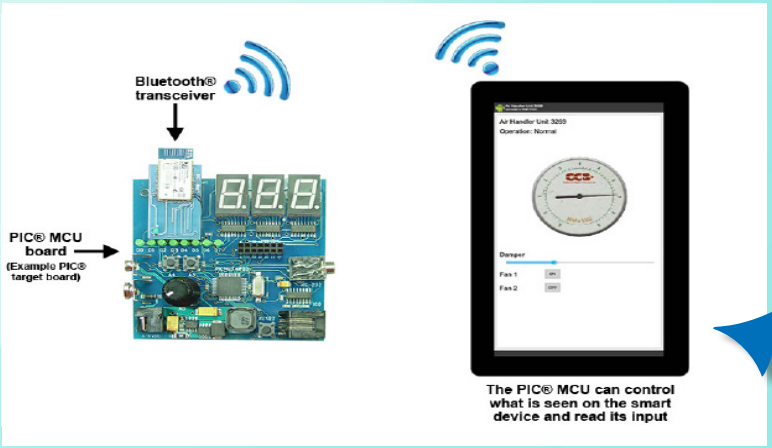


The compiler can also be set up to insert tags at every branch in the program or between specific points to help with full path testing. If users own an IDE compiler and a CCS ICD-U64 or ICD-U80, this is a feature that can help users a great deal and is very easy to get going.

CCS Inc COMPILER FEATURE FOCUS



Do you know the CCS C Compiler has an entire library of functions used to interface your PIC to Bluetooth?



Use the EZApp library to quickly create a wireless sensor or controller on a PIC® MCU that may be viewed and displayed on a mobile device using Bluetooth® included in CCS IDE Compilers. Drivers, examples and development boards for the Microchip RN-4020 Bluetooth® module.

Bootloaders for Field Up-Gradable Programs

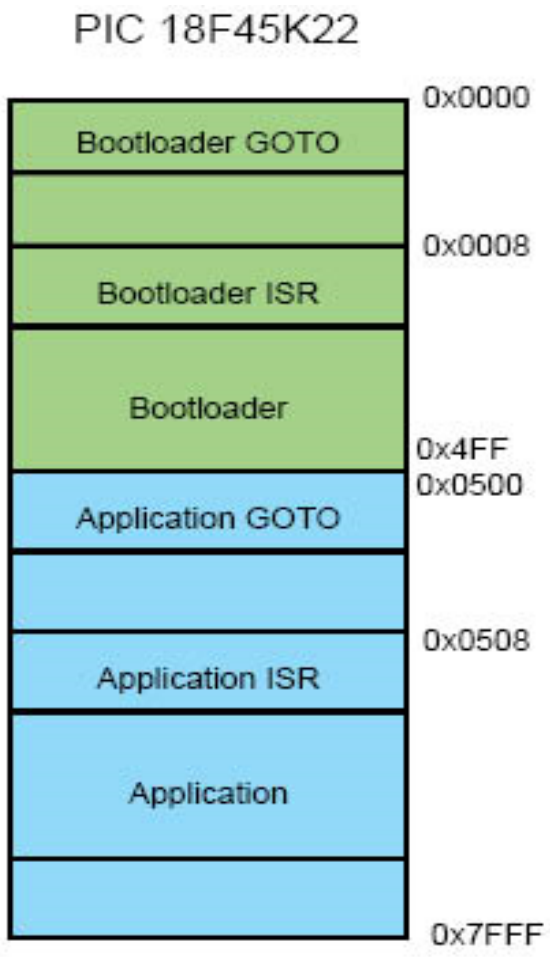
One of the most difficult things to deal with is upgrading a products firmware to fix a bug for products that are already in the field. It can be expensive and time consuming to do a recall of the products or send technicians to update the firmware. One option is to add a bootloader to the product. By using a bootloader it is possible to update a products firmware automatically or by the end user. One of the easiest type of bootloader to implement is a serial bootloader.

A serial bootloader uses a serial connection, RS232 for example, to transfer the new firmware from a PC to the product, which is then programmed onto the product by a small program that runs on the device. To aid in quickly developing a serial bootloader, the CCS C Compiler has bootloader code that can be included in your project, as well has a PC program that can be used to transfer the firmware to product.

The CCS C Compiler provides the following bootloader examples, ex_bootloader.c and ex_pcd_bootloader.c. The first is an example of a serial bootloader for PIC16 and PIC18 devices, PCM and PCH compilers, and the second is an example of a serial bootloader for PIC24, dsPIC30 and dsPIC33 devices, PCD compiler. Both are an example of a standalone bootloader. Standalone bootloaders are small programs that run on the device that are responsible for both receiving the firmware and for programming it onto the device. In general, standalone bootloaders do not require the application for them to work. The size of a serial bootloader program depends on the device they are being used on, for example the CCS serial bootloader for PIC18 devices use 1280 instructions or 2560 bytes of ROM and always remains at the same location in ROM. Some PIC® MCUs allow you to specially code protect the bootloader area in ROM. Additionally the CCS C Compiler provides the following bootloader applications, ex_bootload.c and ex_pcd_bootload.c. Both are examples of applications that can be bootloaded onto a device using the

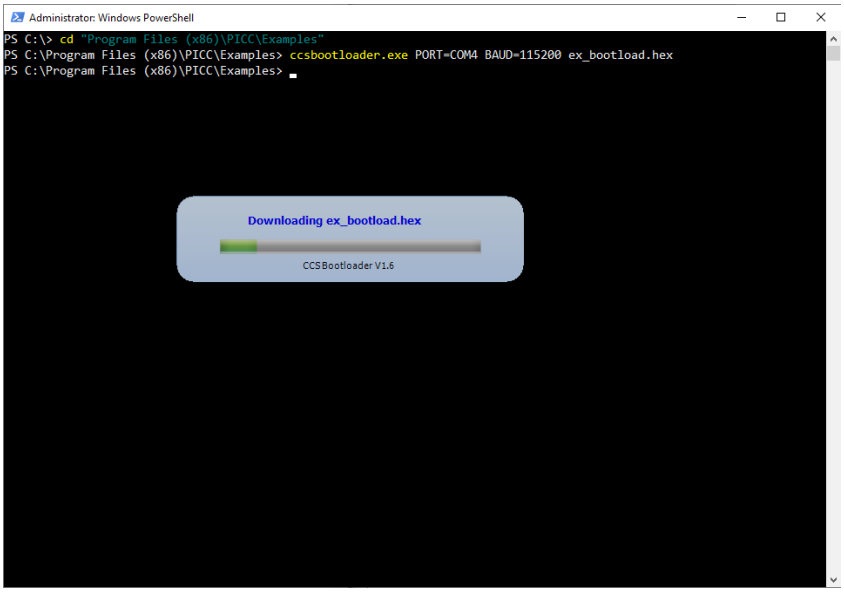
ex_bootloader.c and ex_pcd_bootloader.c bootloaders. The key difference between a standard application and one that can be bootloaded is that the bootloadable application reserves an area of ROM for the bootloader. Frequently that area includes the reset and interrupt vectors so the application will use an alternate area that the bootloader can link to. In general #including the same bootloader.h file that the bootloader uses is all that needs to be done to build an application that is compatible with the bootloader.

Here is a memory map for a low memory bootloader:



A key consideration for bootloaders is deciding when to bootload. The bootloader program starts when the chip starts. If there is no application program in memory then it goes into bootload mode. That is the easy case. For reloading, a button could be used, for example hold that button down, power up and the bootloader sees the button down and starts the loading process. The application itself could trigger a bootload by writing a value to EEPROM and then resetting, the bootloader would see the special value and could force a bootload.

Finally CCS provides a PC program, CCS Bootloader, that can be used to transfer firmware (a .hex file) from a PC to a device that is running a CCS C Compiler bootloader. The CCS Booloader program is a command line utility that may be distributed as part of the user's end product.



It should be noted that the CCS IDE new project Wizard has an option to create a bootloader for you.

CCS has done bootloaders that work over USB, I2C, CAN, SD cards, USB Flash sticks, TCP/IP and HTTP. Contact us if you need help with your bootloader.

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

CCS Inc

Follow Us!