Nov / Dec 2021

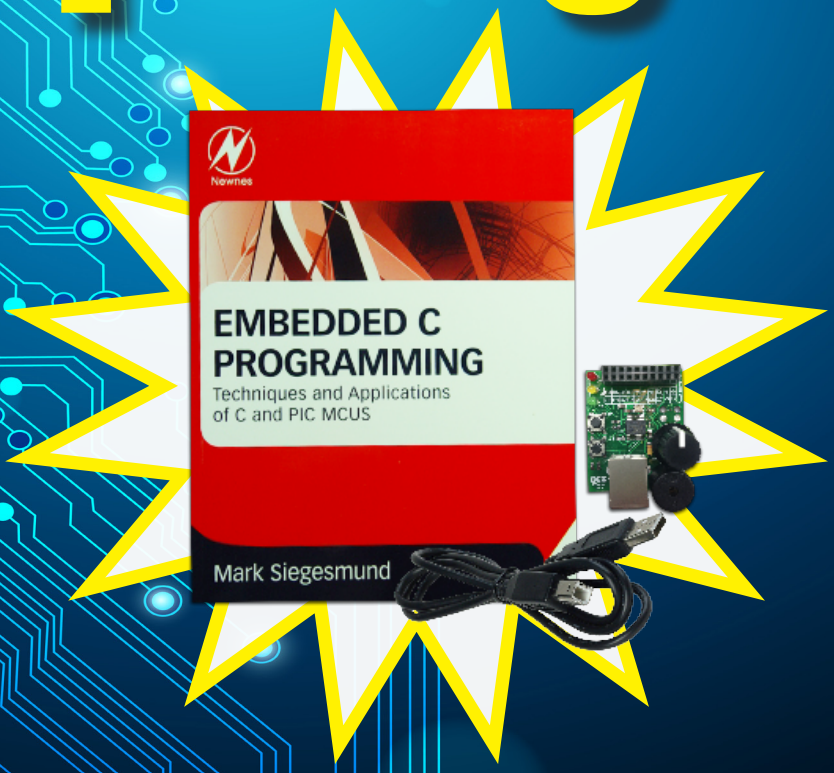
CCS Inc

<BITS & BYTES> Newsletter

www.ccsinfo.com
262-522-6500

Product Spotlight

INSIDE THIS ISSUE:



E3MINI BOARD AND BOOK BUNDLE

This book provides a hands-on introductory course on concepts of C programming using a PIC® microcontroller and the CCS C compiler. Through a project-based approach, this book provides an easy to understand method of learning the correct and efficient practices to program a PIC® microcontroller in the C language. Principles of C programming are introduced gradually, building on skill sets and knowledge. Early chapters emphasize the understanding of C language through experience and exercises, while the latter half of the book covers the PIC® microcontroller, its peripherals, and how to use those peripherals from within C in great detail.

support@ccsinfo.com sales@ccsinfo.com

A Way to Teach C in the Classroom

The Microchip PIC® Microcontrollers (MCUs) have become the most popular choice for new 8 bit embedded designs. Other features include running at a speed of up to 280 million instruction per second, a low cost of some under $1, and large number of interfaces like USB, Ethernet and analog signals are useful features of PIC® MCUs.

The C programming language, originally developed by AT&T Labs by authors Brian Kerninghan and Dennis Ritchie, known as K&R C, became an international standard by ANSI in 1989. The C programming language syntax is the core of many newer programming languages. In 2003 a new standard derived from C defined the C++ language. C++ has some complex language elements that make it impractical for use on a microcontroller as opposed to a desktop PC. C is now the most popular language for programming microcontrollers and it is important for students to learn.

Whether you are an instructor looking to teach embedded C programming in a classroom setting, or a hobbyist looking to learn a new skill, Embedded C Programming:Techniques and Applications of C and PIC® MCUs by Mark Siegesmund is a great resource. It provides hands-on introductory concepts of C programming using PIC® microcontrollers with the CCS C compiler. The book demonstrates programming methodology and tools used by professionals in the field of embedded design by using a step-by-step approach. The writing is reader-friendly and the principles of the C programming language are introduced gradually. Each chapter builds on concepts introduced in the previous chapter, forming a strong foundation in embedded C programming.

There are endless possibilities in the field and the book encourages readers to alter, expand, and customize code for use in their own projects. The book focuses on real-world applications, programming methodology and tools, and best-practice techniques. Some other key features include:

- Each chapter includes C code project examples, exercises, and quizzes
- Tables, graphs, charts, references, photographs, schematic diagrams, flow charts, and compiler compatibility notes
- Publisher support and resources for educators
- Free Single-chip IDE Compiler for PIC18FK50 included with book purchase

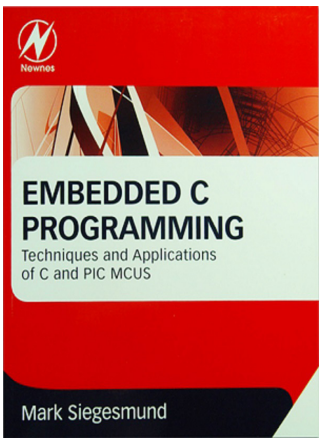


Table of Contents Include:	
C Overview and Program Structure	PIC® Microcontroller
Constants	Discrete Input and Output
Preprocessor Directives	Interrupts
Data Variables and Types	Timers/ Counters
Expressions and Operators	Advanced Timing
Statements	Analog Techniques
Functions	Internal Serial Busses
Arrays	External Serial Busses
Structures	Multitasking
Memory and Pointers	Inline Assembly
Built-in Functions	Debugging
Strings	
Function-Like Macros	
Conditional Compilation	

The hands-on exercises in the book have been tailored to be used with the CCS E3mini development board. This board uses the PIC18F14K50 microcontroller. The board has a bootloader, so no device programmer is required to reprogram the board. Included is a USB cable that can send data to the PC for running the programs. Everything is included that you need to get started!

Chapter 3 example of an exercise:

1. Write a program that turns the green LED on for 10 seconds, the yellow LED on for 3 seconds, and red LED on for 10 seconds.

...

Chapter 3 example of a quiz question:

(3) What happens if a #define uses its own identifier name in the text of the define?

(a) This is the only way to get that identifier in the code post-preprocessor
(b) An error will be flagged on the #define line
(c) An error will be flagged where the define is used
(d) When the identifier is used it is turned into white space
(e) The computer hangs because it replaces the identifier with itself forever

CCS also has a Sensors Explorers kit. This product includes the E3mini Board, bunch of sensors and a tutorial book with examples for each sensor.

- Human Touch
- Temperature
- Light
- Barometric Pressure
- Humidity
- Accelerometer
- Magnetic Field

- Ultrasonic Range
- Vibration
- Sound
- Rotary Encoder

Included Output Devices:

- Full Color LED
- Generic Relay
- Stepper Motor

Also available to add on for additional purchase:

- GPS Unit
- 7-Seg LED
- Keypad

Learn more about our books here: https://www.ccsinfo.com/books.php

CCS provides many development tools in the world of embedded software, with our specialty being in embedded C programming for PIC® MCUs. A variety of options for our famous C-Aware IDE compilers and Command Line compilers, custom engineering and consulting services, and personalized technical support, are just a few of the tools and resources we have to offer. Check out our website and dive deeper into the world of embedded C programming! **https://www.ccsinfo.com/**

Problems Buying Production IC's? Migrating to Newer Processors

Since the beginning Microchip has been great at having stock of all processors, even very old ones. That has changed over the last year as many of you have found out. If you are reading this then you probably are a CCS C compiler user. The good news is with most programs it is very easy to change from one chip to another.

For example if it is hard to buy the popular PIC16F887 part, you may be able to find the PIC16F747 and that might work for your application.

In many cases the only change you need to make is changing:

```
#include <16F887.h>
```

To:

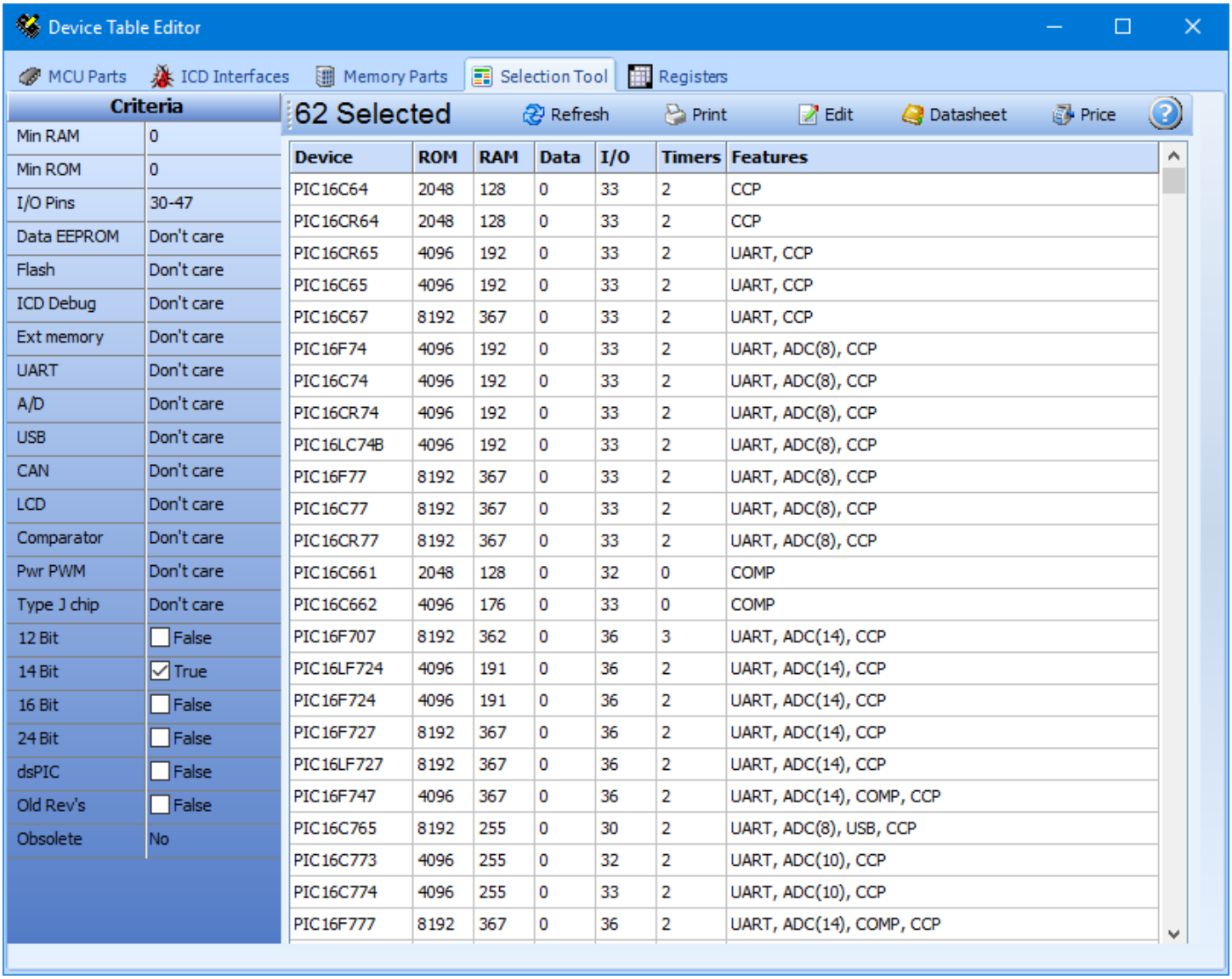
```
#include <16F747.h>
```

Then recompile and start up the production line.

Another bonus is the newer parts seem to have more features and they are a lower cost. For example a popular 8 pin part for many classic designs uses the PIC12F675 part. The newer PIC16F18313 part is 33% lower cost with double the memory.

If you have the IDE compiler there is an easy way to find parts that might work as an alternative. Use TOOLS > DEVICE EDITOR > SELECTION TOOL

Then on the right side select the families (14 bit shown here) and the pin count range (30-47 shown here) along with anything else you want to filter by. The table shows the essential characteristics including the modules (like UART) included in the part.



This works so well because of the compiler built in functions that are customized on the fly for the chip that is being compiled. For example the built in function to read the Analog to Digital converter, read_adc().

For the 12F675 you get:

```
.....................   adc_data = read_adc();
0010:  BSF    1F.1
0011:  BTFSC  1F.1
0012:  GOTO   011
0013:  MOVF   1E,W
0014:  MOVWF  25
```

For the 16F18313 you get this with the same C code:

```
.....................   adc_data = read_adc();
000B:  BSF    1D.1
000C:  BTFSC  1D.1
000D:  GOTO   00C
000E:  MOVF   1C,W
000F:  MOVLB  00
0010:  MOVWF  21
```

If you write directly to a register then you have have more trouble converting the code. For example on a PIC12F675 you might have a line like this:

```
#byte  PORT_A = 0x05
```

Then you might do this in your code:

```
data = PORT_A;
```

For a PIC16F18313 the port A address is 0x0C so you could do this:

```
#byte  PORT_A = 0x0C
```

Or better yet, to make the same code work on both chips:

```
#byte  PORT_A = getenv("SFR:PORTA")
```

This only works if the SFR names are the same in both chips. You should also consider switching to built-ins by removing the #byte all together and in your code doing this:

```
data = input_a();
```

You can do a similar thing for bit names like this:

```
#bit  TIMER_INT_FLAG = getenv("BIT:T1IF")
```

Be aware sometimes the bits, even with the same name, function differently on different chips. Taking some extra time to use built in functions can save you time in the long run.

Chips with the more advanced peripheral modules may have different options for the built in functions. It is usually easy to figure out the changes by looking at the device header in the section for the peripheral. The comparator is one module that changes a lot between chips. For example:

PIC12F675: `setup_comparator( A0_A1 );`
PIC16F18313: `setup_comparator( CP1_A1_A0 );`

Newer chips generally have new fuse settings. In general the compiler default fuses are good. You may want to review the fuses (VIEW > CONFIGURATION BITS) for the new part to make sure they are good.

If you use oscillator fuse settings we strongly recommend you remove them and use the #use delay() instead. That directive sets all the fuses and registers for the oscillator. These settings are very different between chips so using the #use delay() will make the code more portable. For example:

```
#use delay( crystal=8mhz, clock=32mhz )
```

This article gives an overview of how to implement a simple low pass and high pass filter for audio using dsPIC processors. The example code implements a tone control pot. Center does not filter, and when turned counter-clockwise. the bass is increased and clockwise increases the trebble.

Our example uses a structure for each sample point that is a complex number. It has a real and imaginary part to allow other areas of the program that use them both. For our use we only need the real part.

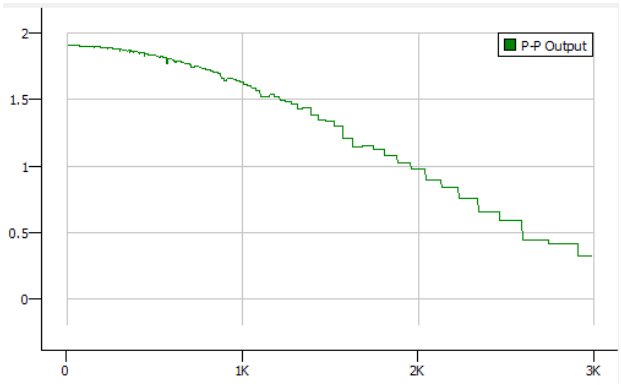
```
typedef struct
{
    signed int16   re;
    signed int16   im;
} Complex;
```

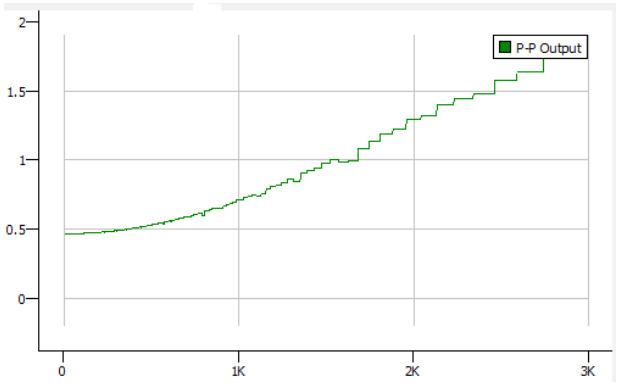
The pseudo-code for a 3 pole filter looks like this:

```
for i=0 to count-1
     window[top] = samples[i].re
     kp = 0

     acca = 0

     for j = top downto 0
       acca += K[kp++] * window[j]

     for j = 2 downto top+1
       acca += K[kp++] * window[j]

     samples[i++].re = acca

     if ++top==3
       top=0
```

The algorithm uses a set of coeficient constants that determine if the filter is low pass or high pass and what the cutoff frequency is. In our example we want the constants to change based on the pot setting. These constants are based on a 8khz sample rate.

Simulated performance (max CW and CCW):





The acca register is a built in accumulator in the processors with a DSP. The DSP engine has a bult in instruction to help with code like "`acca += K[kp++] * window[j].`" It looks like this in assembly:

```
mac W4*W5,A
```

Notice in the assembly language how C variables are referenced. Pay close attention to when pointers are being used as opposed to straight values.

```
void adjust_tone(Complex* samples, unsigned int16 size, int16 tone_pot) {
static int16 r[3];    // Window
static int16 top;     // Window ptr
       int16 k[3];    // coefficients

   if(((tone_pot)>(0x8000-NO_TONE_ADC_COUNTS)) &&
      ((tone_pot)<(0x8000+NO_TONE_ADC_COUNTS)))
         return;

   if(tone_pot<0x8000) {  // Low pass
      k[0] = (0x8000-tone_pot)/364;        // 8000          0
      k[0] = k[0]*k[0];                    //    0 (0.0)    1Fa4 (0.25)
      k[1] = 0x7FFF-((0x8000-tone_pot)/2); // 7fff (1.0)    3fff (0.5)
      k[2] = k[0];
   } else {               // High pass
      k[0] = (tone_pot-0x8000)/3;          // 8000          ffff
      k[0] = 0xf800-k[0];                  // F800 (-0.9)   CD56 (-.6)
      k[1] = 0x7fff;                      // 7fff (1.0)    7FFF (1.0)
      k[2] = k[0];
   }

   #asm
   mov  samples, w3       // w3 -> &source[0].re
   mov  size, w6          // w8 = size - 1
   dec  w6, w6
   do   w6, loop          // do the following loop "size" times
        sl  top,W0           // r[top] = source[i].re
        mov r,W7
        add W0,W7,W0
        mov [W3],[W0]
        mov k,W1             // W1=&K[0]
        clr a                // acca = 0
           sl  top,W0        // 1st inner for W0=&R[top] downto &R[0]
           mov r,W7
           add W0,W7,W0
           loop2:
           mov [W0--],W4         // acca += K[W1++] * R[W2]
           mov [W1++],W5
           mac W4*W5,A
           cp  W7,W0
           bra le,loop2
                                 // 2nd inner for W0=&R[2] downto &R[top+1]
           neg top,W0
           add W0,#2,W7
           bra z,loop3exit
           mov r+4,W0
           loop3:                          // acca += K[W1++] * R[W2]
           mov [W0--],W4
           mov [W1++],W5
           mac W4*W5,A
           dec w7,w7
           bra nz,loop3
           loop3exit:
        sac A,#0,W0          // source[i].re = acca
        mov W0,[W3++]
        inc top              // top = top +1
        mov top,W7           // if top>=3 (poles)
        cp  w7,#3
        bra lt,loop
        clr top                 // top=0
        loop:
        inc2  w3, w3         // w3 = W3 + 2 (next point)
   #endasm

}
```

About CCS: 30 Years in the Making

Embedded computer systems has been our passion since 1992. We were first to release a C compiler for the PIC® MCU, after switching internally from the 8051. The C compiler was designed from the bottom up around the special architecture of the PIC® MCU. The PCB compiler supported the original four Microchip parts and the PCM compiler supported the one mid-range part they offered.

Microchip continued to develop new chips with additional features, and so did CCS to support these devices. Now we support over 1000 parts from 12 bit to 24 bit opcodes. We make powerful features available for even the smallest devices. We created unique built-in functions that the user would be able to migrate to new devices with ease and less programming time. We deliver tools that do not force our users into buying bigger, and more expensive, chips every time you want to do something more advanced.

As a software compiler developer we have a unique position in the marketplace for hardware development tools that work seamlessly with the IDE compiler. CCS develops tools exclusively for the Microchip PIC® MCU. We do not produce a watered down product that is available for dozens of chip manufactures. We intentionally do not try to shoe-horn an existing compiler engine into simply working for a PIC® MCU.

CCS has a team of embedded engineers that develop the compiler and use it to produce real world applications in custom projects. We produce products that do work in the real world as opposed to products that should theoretically work. Our custom customers are on the cutting edge and need us to be ready for new protocols. This also improves the compiler so we can support these new technologies and have built-in functions and examples for our customers.
We are committed to producing a compiler that is easy to use for Software Engineers, Electronic Engineers and hobbyists alike. The emphasis is on using the rich set of built-in functions that operate across all PIC® MCU families.

Approximately 11 years ago, CCS took on another endeavor, in purchasing a company in the Amateur Radio field, called West Mountain Radio. This brand focused on radio accessories including interface units from the radio to a PC. CCS was able to augment some brilliant and very popular analog designs with microprocessors (PIC® MCU's of course) to take the product lines to the next level. For example CCS took a simple power distribution unit and added internet conductivity so the power distribution could be monitored and controlled from any web browser in the world.

As a company we continue to fully support and update the product lines from both companies. We appreciate all of our supportive customers and are always willing to listen to feedback.

Where in the World is Waukesha?
(Pronounced locally W-awe-ka-shaw)

Just a few short weeks ago, Waukesha, Wisconsin became international news and most of our customers are probably not aware our facility is in Waukesha Wisconsin. Some have been asking about recent events and the city. To the curious, we wanted to share a few words here even though it will not help with your C coding.

Waukesha is just a short distance away from the more known city of Milwaukee, but alike it is named from an Native American word for "fox". Other Native American cities in this area include Pewaukee - the home of professional football player JJ and TJ Watt, Oconomowoc and also in recent news,

Kenosha. Waukesha sits atop dozens of natural springs, nicknaming us Spring City. In fact our street address is Spring City Drive.

The one claim to fame the city has used to market itself, is as the home of Les Paul, the first to make an electric guitar. There is a major road renamed to Les Paul Parkway, a middle school was renamed, and there are colorful extra large guitars decorating the city.

Waukesha started out very industrious and within the past few decades, the local community has transitioned to more residential. A fairly sizable city, but with a home town feeling of coffee shops, bike paths and many new retail areas. Waukesha can boast by having 100+ year old Carroll University, as well as a large Missionary college.

So, why is everyone asking where is Waukesha? Just before Thanksgiving this past November, a man drove a red SUV through a annual Holiday Parade. People of all ages were involved and over 50 people were taken to the hospital for varying injuries. Unfortunately, five people died, including one eight-year old child. The driver was in fact, just released out on bail, again, for trying to drive over his girlfriend just the week before. The community came together and people put blue lights on their homes to show support for those that lost ones in this tragedy.

Waukesha has also had some infamous crime, including the two young girls that repeatedly stabbed a third called the “Slender Man Stabbings.” Both girls were sent to a mental hospital. One is now out on the way to college and the other is being evaluated every 6 months to figure out when she has been “cured.” However, in this case, the victim survived.

When we go to shows and people find out where we are, the ones who have heard of Waukesha often times know the airport as being on the way to Oshkosh (yet another Indian name). Pilots from all over the world come to Oshkosh at the end of July for the Experimental Aircraft Association Fly-In. This a huge event and makes the airport the busiest airport in the world for one week. This past July we had a booth at the show to show case DC Power and Battery analyzer equipment made by our subsidiary company, West Mountain Radio. This is a spectacular event with over 100,000 visitors a day. You can walk right up to hundreds of all types of planes, watch an air show along the mile long show runway, or even buy a plane. The view of the aircraft parking lot is amazing with uniqueness and ingenuity.

Waukesha may have some good and bad notoriety, but it is still an excellent place to work to provide quality products to our customers. So, where is Waukesha? The question is why not Waukesha?

E3mini Board and Book Bundle

Includes

Embedded C Programming Textbook

E3mini Prototyping Board

Single chip IDE compiler

sales@ccsinfo.com
262-522-6500 EXT 35
www.ccsinfo.com/NL1221

CCS Inc

C Compiler Savings

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!