List of COMPLETE EXAMPLE PROGRAMS
(in the EXAMPLES directory)

EX_14KAD.C	An analog to digital program with calibration for the PIC14000
EX_1920.C	Uses a Dallas DS1920 button to read temperature
EX_8PIN.C	Demonstrates the use of 8 pin PICs with their special I/O requirements
EX_92LCD.C	Uses a PIC16C92x chip to directly drive LCD glass
EX_AD12.C	Shows how to use an external 12 bit A/D converter
EX_ADMM.C	A/D Conversion example showing min and max analog readings
EX_CCP1S.C	Generates a precision pulse using the PIC CCP module
EX_CCPMP.C	Uses the PIC CCP module to measure a pulse width
EX_COMP.C	Uses the analog comparator and voltage reference available on some PICs
EX_CRC.C	Calculates CRC on a message showing the fast and powerful bit operations
EX_CUST.C	Change the nature of the compiler using special preprocessor directives
EX_FIXED.C	Shows fixed point numbers
EX_DPOT.C	Controls an external digital POT
EX_DTMF.C	Generates DTMF tones
EX_ENCOD.C	Interfaces to an optical encoder to determine direction and speed
EX_EXPIO.C	Uses simple logic chips to add I/O ports to the PIC
EX_EXTEE.C	Reads and writes to an external EEPROM
EX_FLOAT.C	Shows how to use basic floating point
EX_FREQC.C	A 50 mhz frequency counter
EX_GLINT.C	Shows how to define a custom global interrupt handler for fast interrupts
EX_ICD.C	Shows a simple program for use with Microchips ICD debugger
EX_INTEE.C	Reads and writes to the PIC internal EEPROM
EX_LCDKB.C	Displays data to an LCD module and reads data from keypad
EX_LCDTH.C	Shows current, min and max temperature on an LCD
EX_LED.C	Drives a two digit 7 segment LED
EX_LOAD.C	Serial boot loader program for chips like the 16F877
EX_MACRO.C	Shows how powerful advanced macros can be in C
EX_MOUSE.C	Shows how to implement a standard PC mouse on a PIC
EX_MXRAM.C	Shows how to use all the RAM on parts will problem memory allocation
EX_PATG.C	Generates 8 square waves of different frequencies
EX_PBUSM.C	Generic PIC to PIC message transfer program over one wire
EX_PBUSR.C	Implements a PIC to PIC shared RAM over one wire
EX_PBUTT.C	Shows how to use the B port change interrupt to detect pushbuttons
EX_PGEN.C	Generates pulses with period and duty switch selectable
EX_PLL.C	Interfaces to an external frequency synthesizer to tune a radio
EX_PSP.C	Uses the PIC PSP to implement a printer parallel to serial converter
EX_PULSE.C	Measures a pulse width using timer0
EX_PWM.C	Uses the PIC CCP module to generate a pulse stream
EX_REACT.C	Times the reaction time of a relay closing using the CCP module
EX_RMSDB.C	Calculates the RMS voltage and dB level of an AC signal
EX_RTC.C	Sets and reads an external Real Time Clock using RS232
EX_RTCLK.C	Sets and reads an external Real Time Clock using an LCD and keypad
EX_SINE.C	Generates a sine wave using a D/A converter
EX_SISR.C	Shows how to do RS232 serial interrupts
EX_SLAVE.C	Simulates an I2C serial EEPROM showing the PIC slave mode
EX_SPEED.C	Calculates the speed of an external object like a model car
EX_SPI.C	Communicates with a serial EEPROM using the H/W SPI module
EX_SQW.C	Simple Square wave generator
EX_SRAM.C	Reads and writes to an external serial RAM
EX_STEP.C	Drives a stepper motor via RS232 commands and an analog input
EX_STR.C	Shows how to use basic C string handling functions
EX_STWT.C	A stop Watch program that shows how to user a timer interrupt
EX_TANK.C	Uses trig functions to calculate the liquid in an odd shaped tank
EX_TEMP.C	Displays (via RS232) the temperature from a digital sensor
EX_TGETC.C	Demonstrates how to timeout of waiting for RS232 data
EX_TONES.C	Shows how to generate tones by playing "Happy Birthday"
EX_TOUCH.C	Reads the serial number from a Dallas touch device
EX_USB.C	Implements a USB device on the PIC16C765
EX_VOICE.C	Self learning text to voice program
EX_WAKUP.C	Shows how to put a chip into sleep mode and wake it up
EX_WDT.C	Shows how to use the PIC watch dog timer
EX_WDT18.C	Shows how to use the PIC18 watch dog timer
EX_X10.C	Communicates with a TW523 unit to read and send power line X10 codes

CUSTOM COMPUTER SERVICES, INC.

CCS Inc.
PO Box 2452
Brookfield, WI 53008
http://www.ccsinfo.com

Sales:
262.797.0455 x35 • sales@ccsinfo.com
Technical Support:
262.797.0455 x32 • support@ccsinfo.com

List of INCLUDE FILES
(in the DRIVERS directory)

14KCAL.C	Calibration functions for the PIC14000 A/D converter
2401.C	Serial EEPROM functions
2402.C	Serial EEPROM functions
2404.C	Serial EEPROM functions
2408.C	Serial EEPROM functions
24128.C	Serial EEPROM functions
2416.C	Serial EEPROM functions
24256.C	Serial EEPROM functions
2432.C	Serial EEPROM functions
2465.C	Serial EEPROM functions
25160.C	Serial EEPROM functions
25320.C	Serial EEPROM functions
25640.C	Serial EEPROM functions
25C080.C	Serial EEPROM functions
68HC68R1.C	Serial RAM functions
68HC68R2.C	Serial RAM functions
74165.C	Expanded input functions
74595.C	Expanded output functions
9346.C	Serial EEPROM functions
9356.C	Serial EEPROM functions
9356SPI.C	Serial EEPROM functions (uses H/W SPI)
9366.C	Serial EEPROM functions
AD7715.C	A/D Converter functions
AD8400.C	Digital POT functions
ADS8320.C	A/D Converter functions
ASSERT.H	Standard C error reporting
AT25256.C	Serial EEPROM functions
CE51X.C	Functions to access the 12CE51x EEPROM
CE62X.C	Functions to access the 12CE62x EEPROM
CE67X.C	Functions to access the 12CE67x EEPROM
CTYPE.H	Definitions for various character handling functions
DS1302.C	Real time clock functions
DS1621.C	Temperature functions
DS1868.C	Digital POT functions
ERRNO.H	Standard C error handling for math errors
FLOAT.H*	Standard C float constants
FLOATEE.C	Functions to read/write floats to an EEPROM
INPUT.C	Functions to read strings and numbers via RS232
ISD4003.C	Functions for the ISD4003 voice record/playback chip
KBD.C	Functions to read a keypad
LCD.C	LCD module functions
LIMITS.H	Standard C definitions for numeric limits
LOADER.C	A simple RS232 program loader
LOCALE.H	Standard C functions for local language support
LTC1298.C	12 Bit A/D converter functions
MATH.H	Various standard trig functions
MAX517.C	D/A converter functions
MCP3208.C	A/D converter functions
NJU6355.C	Real time clock functions
PCF8570.C	Serial RAM functions
SETJMP.H	Standard C functions for doing jumps outside functions
SIGNAL.H	Standard C signal functions
STDDEF.H	Standard C definitions
STDIO.H	Not much here - Provided for standard C compatibility
STDLIB.H	String to number functions
STRING.H	Various standard string functions
TONES.C	Functions to generate tones
TOUCH.C	Functions to read/write to Dallas touch devices
X10.C	Functions to read/write X10 codes

QUICK START with CCS C Compiler

How to start programming PICmicro® MCUs

Why C on a PICmicro® MCU?

The widespread use of C on microcontrollers has gained popularity because writing in C takes a fraction of the time it takes to write the same code in Assembly. C is a well known language and is the language of choice by embedded programmers. Thus, using C saves a developer countless hours.

TO BEGIN...

To start development, you will need at least an IBM compatible PC running Windows 95, 98, NT, ME, XP or Linux. Depending upon the specific chip your project requires, you will also need to purchase one of the C compilers from CCS, Inc. The compiler will actually allow you to write in easy to read, high level C instructions, and it will convert those instructions into machine language outputting a HEX file.

Why the CCS C Compiler?

Embedded C designers find the CCS PICmicro® MCU C compiler very user friendly, and an exceptionally cost-effective tool for all embedded C projects. The CCS C compiler offers an extensive list of built-in functions, device drivers, example programs, and included libraries. Since CCS offers engineering services for custom software, firmware, and hardware design, we relate to other developers and always continue to try to accommodate their special engineering needs. This is demonstrated by continuous updates to the compiler as new capabilities and application examples, as well as chips become available. These advantages only begin to explain why programmers find the compiler so easy to use. We strive to make our compiler the most efficient C compiler on the market.

CCS C Compiler Packages

PCW, PCWH, or PCWHD

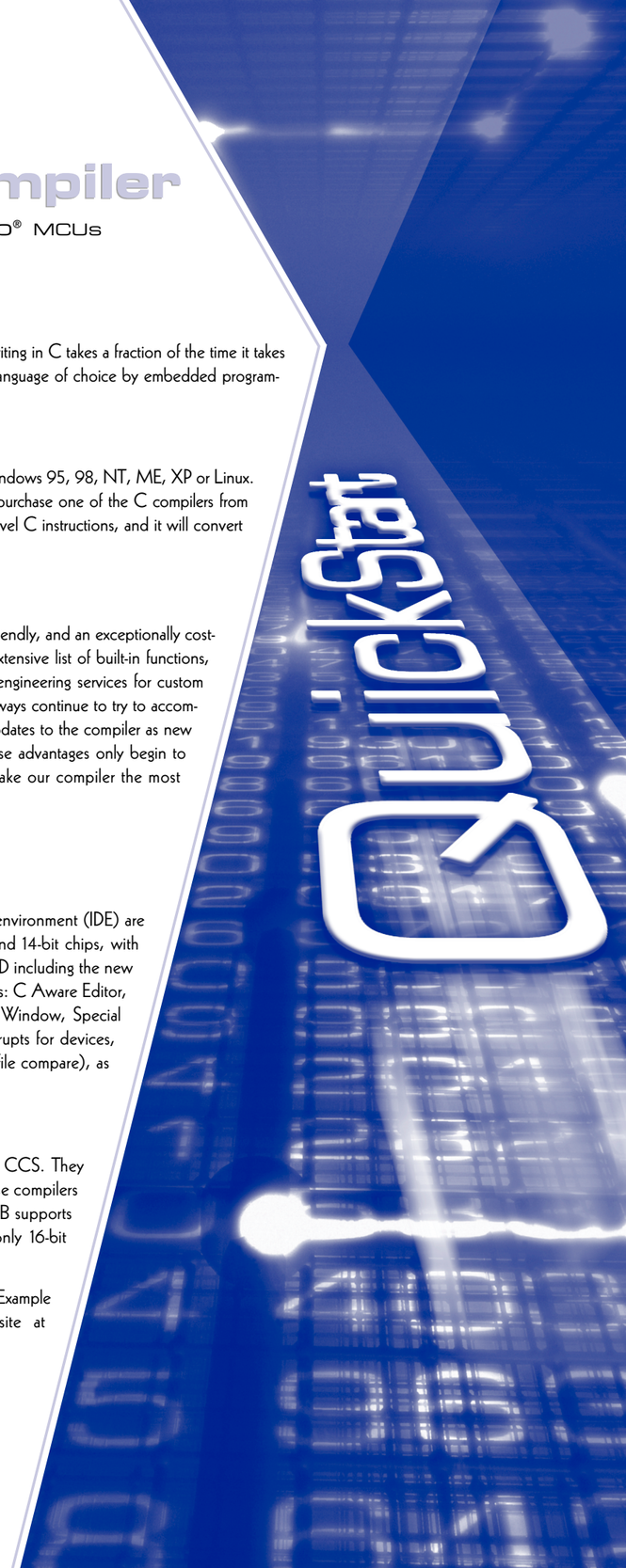
The three CCS C compiler packages that integrate a powerful development environment (IDE) are the PCW, PCWH, and the PCWHD. These compilers support both 12 and 14-bit chips, with the PCWH additionally including PIC18XXX MCU support, and the PCWHD including the new dsPIC™ chip support. All the compiler packages offer several unique features: C Aware Editor, New Project Wizard, Device Selector/Editor, Extra Optimization, Statistics Window, Special Viewers (such as quick and easy access to data sheets, valid fuses and interrupts for devices, a HEX file disassembler, COD file interpreter, and an advanced source/list file compare), as well as a Serial Port Utility.

PCB, PCM, or PCH

PCB, PCM, and PCH are the command line PICmicro® C compilers from CCS. They require you to provide your own editor, such as Microchip's MPLAB®. These compilers are attractively priced for hobbyists and low throughput users. Note that PCB supports only 12-bit chips, PCM supports only 14-bit chips, and PCH supports only 16-bit PIC18XXX chips.

All of the CCS C compiler versions come complete with Built-in Functions, Example Programs, Device Drivers, and a MPLAB® Interface. Visit our website at http://www.ccsinfo.com/demo.html for a free demo of our compiler.

CUSTOM COMPUTER SERVICES, INC.



Now that you have figured out which CCS compiler will meet your project's needs, you can evaluate your hardware options.

HARDWARE TOOLS:

Device Programmers

WARP-13

WARP-13 is a very good, low cost device programmer that supports PICmicro® chips. It is operated by easy to use Windows software, and is MPLAB® compatible. (CCS sells the WARP-13 for $99.)

PICSTART® Plus

Microchip's PICSTART® Plus is a development kit providing a highly flexible microcontroller design for all PICmicro® MCU devices. It operates only with Windows® under MPLAB® environment. It is higher priced than the WARP-13, however Microchip provides the earliest support for new chips.

MPLAB® ICD

MPLAB® ICD is primarily used as an In-Circuit Debugger for Microchips PIC16F87X Flash MCU line, but can also be used to program PIC16F87X chips. The ICSP™ (In-Circuit Serial Programming) protocol allows cost-effective, in-circuit Flash programming and debugging under the MPLAB® IDE.

Others

There is a large number of chip and EEPROM programmers available. As a general guideline, be sure to check what chips the programmers work with, as well as how dedicated the company is to keeping up with new chips as they are released.

Debuggers/Emulators

MPLAB® Simulator

MPLAB® is a free IDE for use with Microchip developing. One component it provides is a simulator that will imitate a running PICmicro® MCU on a PC. You can single step through C code or Assembly code and view variables. This is a dry run only—it does not actually talk to your hardware parts.

In-Circuit Emulator

The emulator is a hardware device that plugs into your target board (or a prototyping board) and emulates a PICmicro® chip controlled by a PC. You can download the program to the emulator, single step through C code or Assembly code, and view variables. The emulator runs in real time and communicates with all the hardware just like a real PICmicro® MCU.

Emulators are made by a number of companies and most have equivalent capabilities. This is the most effective debugging option as well as the most expensive. (The emulator we sell at CCS is the ICEPIC™ emulator which typically costs $515. It works in either MPLAB® or with its own proprietary software.)

ICD (MPLAB® ICD)

Some PICmicro® chips, but not all, have built-in hardware to aid in the debugging process. These chips can dedicate some of the ROM and use some I/O lines to perform a software/hardware combination debugging environment. Many of the features of the emulator are available, however some system resources are tied up. A low cost ICD module that connects the PC to the target hardware or prototyping board is required. (CCS sells the MPLAB® ICD for $75.)

Software

You do not necessarily need a hardware debugging solution to help debug your problems. Any two port pins on a PICmicro® MCU can be used by the C compiler to generate an RS232 link, which then allows you to use `printf` to output memory values and processor conditions to a terminal PC. Another option is to toggle a port pin, high or low, when entering or leaving specific algorithms, letting you know if your code is getting to the proper functions. These ports could then be either viewed on an oscilloscope or tied to an LED.

Test Platform

Target Platform

If ready, your target hardware can be used for testing. If using an emulator, make sure the socket is right (emulators usually have DIP sockets by default). If using the ICD try to dedicate the required pins to a debug socket. This socket may also be used for in circuit re-programming, which is useful in itself for updating firmware on units which have already been out in the field.

Prototyping Board

CCS (and a number of other companies) sell a prototyping board with the basic PICmicro® MCU hardware, some LEDs, buttons, RS232 drivers, and other common interfaces. This board may be used to start testing software functions before your target hardware is complete. It is a fast way to start testing code. (CCS sells a Software Prototyping Board for $145.)

PICmicro® MCUs

It is generally a good idea to use a PIC16F877 for debugging. This chip is easy to reprogram because it is Flash and it has the features of most of the other PICmicro® devices. However, there are instances when the PIC16F877 is not a viable solution and another chip should be used. If emulating is done during the debugging process, a chip is not needed. If you are using your own target hardware for debugging, then you will need the corresponding target chip. Also, if a non-Flash part is used for development, a windowed version can be erased with a UV eraser and then reprogrammed for your needs.

Keep in mind your ROM and RAM requirements when choosing a PICmicro® MCU for your platform. See the list of devices the CCS C compiler supports, and notice the opcode column: a 12-bit opcode requires the PCB or PCW compiler, a 14-bit opcode requires the PCM or PCW compiler, a 16-bit opcode requires the PCH compiler or PCH add-on to the PCW, SXC requires the SXC compiler, and a dsPIC™ requires the PCDSP or PCWHD compiler.

To develop your hardware design, choose one of many device programmers to burn the program onto the chip. You will then need to debug your code with either an emulator, simulator, or debugger. Both of these will allow you to step through the program as you watch the microcontroller carry out the program code. If your target platform is not yet ready, a special prototyping board may be the simplest solution. CCS offers a prototyping board that has the basic PICmicro® MCU hardware and other common interfaces. Finally, when choosing your target processor (PICmicro® MCU), be sure to keep in mind your ROM and RAM requirements for the project.

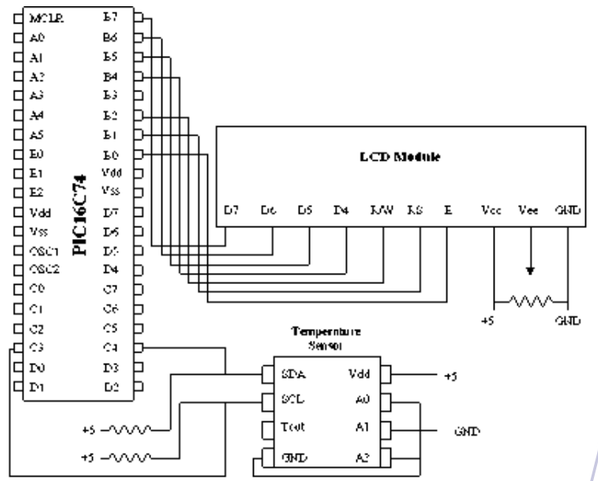
LCD Thermometer Example Application

```
#include <16C74.H>
#fuses HS,NOWDT,NOPROTECT
#use delay(clock=20000000)
#include <lcd.c>
#include <ds1621.c>
#define RESET_BUTTON    PIN_C0

int current_temp, max_temp, min_temp;

void reset_temp() {
     current_temp = read_temp();
     min_temp=current_temp;
     max_temp=current_temp;
}

main() {
      init_temp();
      lcd_init();
      delay_ms(6);
      reset_temp();
      while(TRUE)  {
           current_temp = read_temp();
           if(input(RESET_BUTTON)==0)
                 reset_temp();
           else if(current_temp>max_temp)
                 max_temp=current_temp;
           else if(current_temp<min_temp)
                 min_temp=current_temp;
           printf(lcd_putc,"\fCurrent Temp: %U F\nMin: %U Max: %U",
                 current_temp,min_temp,max_temp);
           delay_ms(500);
      }
}
```



The example application shown here interfaces with a DS1621 temperature sensor. The current temperature is displayed on an LCD, along with the minimum and maximum temperatures recorded since the last reset. The PICmicro® MCU device and the DS1621 sensor communicate with each other via I²C serial communication. The code shown is the only software the user needs to write. This example shows how easy it is to use the large library of supplied drivers to create simple but powerful applications. The compiler toolkit is designed to dramatically reduce software development time. This, along with many other examples, is included with the CCS C compiler.

RECOMMENDED BOOKS and RESOURCES

There are several books and references which can be of great use to new users of PICmicro® MCUs, Microchip's PIC line of microcontrollers, and to embedded systems programming in general. A few of these books are available via CCS directly, and a link has been provided on our website to Amazon.com for the books we do not provide.

PIC C - An introduction to programming the Microchip PIC in C — by Nigel Gardner

PIC C is aimed towards those who do not know C. The basics of C are explained from the perspective of writing programs for the PICmicro® MCU. Although there are many C starter books available, this book details programming specific to the PICmicro® MCU rather than the common PC target. (PICmicro programming is different even though the language is the same.) This book includes examples that work with the CCS C compiler.

The C Programming Language by Brian W. Kerninghan and Dennis M. Ritchie

Dennis Ritchie is the original creator of C and this is the book that introduced C to the world. The CCS C compiler used **The C Programming Language** as the reference for the design of the compiler. This is a short but sweet look at C with all the details of the language simply explained. This is a great reference book for all C programmers.

PCB, PCM, PCH, and PCW Compiler Reference Manual by CCS (Included with each compiler)

The Reference Manual is a great resource that accompanies the CCS compiler. Although it does not provide the C reference information described above, it does list the supported C features. It also offers detail on all of the new PICmicro® MCU specific features and functions. This is sufficient for those who know C and the PICmicro® MCU. If you do not know C we recommend purchasing one of the above books or their equivalents.

Microchip Data Sheets

The **Data Sheets** are included on the PCW CD ROM or can be downloaded from Microchip's WWW site. Hard copies may also be obtained from local Microchip offices. The data sheets cover the details of how the PICmicro® MCU works, how to wire it and exactly how the special features in the chip work (like the timers and A/D converter). The built-in functions of the compiler shelter most users from needing all these details, however it will help you to fully understand the part.

Easy PIC'n by David Benson

Easy PIC'n is a very popular book that covers the information in the Microchip data sheets in an easier to read format, for the beginner. This may be a good book for those who have not used microcontrollers before.

Programming Embedded Systems in C and C++ by Michael Barr

Programming Embedded Systems in C and C++ is a more advanced book not specific to the PICmicro® MCU or the CCS C compiler. It has some general principals for embedded programming.

The Standard C Library by P. J. Plauger

The Standard C Library defines all the standard C functions in good detail. Many of these functions do not apply to C on a PICmicro® MCU, but for those that do, this is a good reference to use. In general, most users should not need this book as the CCS Compiler Reference Manual and The C Programming Language book cover the essentials.

TCP/IP Lean: Web Servers for Embedded Systems by Jeremy Bentham

TCP/IP Lean: Web Servers for Embedded Systems is a guide to implementing the TCP/IP stack on embedded systems, as well as embedded web servers. Not only did the author use PIC16XXX microcontrollers, but also the CCS PICmicro® MCU C compiler.