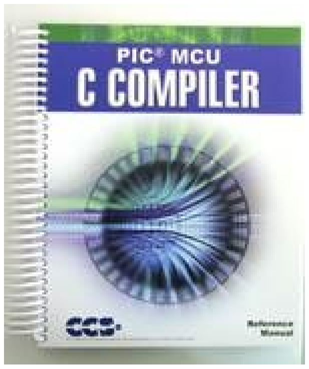

CCS C Compiler

Functional changes from version 2 to version 3:

- Support for DOS only systems and Windows 3.1/DOS only systems has been terminated. The PCB and PCM IDE's have been discontinued. CCSC.EXE is now a 32 bit application and now has a full set of command line options. PCW and CCSC fully operate on Windows 95,98,ME,NT4, and 2000. Customers who buy or have PCB or PCM must use MPLAB as an IDE or their own editor.

- All DLL files used by the compilers are in a DLL directory directly under the directory with the EXE files. All .DLL files in the same directory as the EXE are no longer used and may be deleted. The PC.DEF file and PCINCLUDES environment variable (in AUTOEXEC.BAT) are also no longer used.

- CCSC now has the following command format:

CCSC Valid Options:			
options	*cfilename*	*options*	*cfilename*
+FB	Select PCB (12 bit)	-D	Do not create debug file
+FM	Select PCM (14 bit)	+DS	Standard .COD format debug file
+FH	Select PCH (PIC18)	+DM	.MAP format debug file
+F7	Select PC7 (PIC17)	+DC	Expanded .COD format debug file
+FS	Select PCS (SX)	+Yx	Optimization level x (0-9)
+ES	Standard error file	+T	Create call tree (.TRE)
+EO	Old error file format	+A	Create stats file (.STA)
-J	Do not create PJT file	-M	Do not create symbol file

CCSC Valid Options: (The xxx in the following are optional. If included, it sets the file extension.)			
options	*cfilename*	*options*	*cfilename*
+LNxxx	Normal list file	+O8xxx	8 bit Intel HEX output file
+LSxxx	MPASM format list file	+OWxxx	16 bit Intel HEX output file
+LOxxx	Old MPASM list file	+OBxxx	Binary output file
+LYxxx	Symbolic list file	-O	Do not create object file
-L	Do not create list file		
+P	Keep compile status window up after compile		
+Pxx	Keep status window up for xx seconds after compile		
+PN	Keep status window up only if there are no errors		
+PE	Keep status window up only if there are errors		
+Z	Keep scratch and debug files on disk after compile		
I="..."	Set include directory search path, for example: I="c:\picc\examples;c:\picc\myincludes" If no I= appears on the command line the .PJT file will be used to supply the include file paths.		
#xxx="yyy"	Set a global #define for id xxx with a value of yyy, example: #debug="true"		
+STDOUT	Outputs errors to STDOUT (for use with third party editors)		
+SETUP	Install CCSC into MPLAB (no compile is done)		
+V	Show compiler version (no compile is done)		
+Q	Show all valid devices in database (no compile is done)		

- PCW has a new "Global Defines" window to allow compile time #defines to be entered, saved in a separate file and loaded.

- A new preprocessor directive #LOCATE has been added. #LOCATE works like #BYTE however in addition it prevents C from using the area. For example:
  ```
  float x;
  #locate x=0x50
  ```
  Will locate the float variable at 50-53 and C will not use this memory for other variables automatically located.

- New byte wide I/O functions have been added that follow the normal rules for maintaining the TRIS register (unlike memory mapped port access). Examples:
  ```
      OUTPUT_B(3);
      X = INPUT_A();
  ```

- New built-in function to restart the processor:

  ```
  RESET_CPU();
  ```

  Will jump to location 0 on 12 and 14 bit parts and also reset the registers to power-up state on the PIC18.

- PCW IDE changes:
  View Data Sheet and View Valid Fuses have moved to the View menu
  View|Valid Interrupts has been added to show a chips valid interrupts

- The device .h file format has been updated. Information about device specific functions available is now in the .h file. The following defines supported for compatibility with the old C71 compiler are no longer in the .h file. Use the newer names:

  ```
  RTCC_ZERO use INT_RTCC
  EXT_INT   use INT_EXT
  ADC_DONE  use INT_AD
  ```

- #DEVICE must now be the first non-comment line in the program. Some pre-processor directives may appear before this like #include and #define. Previous versions of the compiler did not require a #device (it defaulted to a '71 in PCM).

- The function EXT_INT_EDGE now accepts two parameter. The first parameter is the external interrupt number (0,1 or 2) for the PIC18. If used it has no effect on 14 bit parts. If there is only one parameter the function works as it used to on external interrupt 0.

- The functions SETUP_COUNTERS(), SET_RTCC() and GET_RTCC are still accepted by the compiler however newer functions will provide better compatibility across families. The newer functions are: SETUP_TIMER0(), SETUP_WDT(), SET_TIMER0() and GET_TIMER0(). Note that the PIC18 SETUP_WDT() only allows the WDT to be turned on or off, the time is set with #FUSES. The other PICs are the opposite. The TIMER0 is 16 bit on the PIC18 by default but can be setup to 8 bit.

- For the PIC18 the READ/WRITE_PROGRAM_EEPROM accept a byte address and the read returns a byte result. The WRITE will only work if a programming voltage is applied to the chip.

- The previously undocumented function ISAMOUNG is documented in this version. ISAMOUNG returns true if a character is one of the characters in a constant string. For example:

  ```
  if( ISAMOUNG( x, "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ") )
     ...
  ```

- Quoted strings now accept inserted hex numbers via \x. For example:

  ```
  printf("Hi There\x0D\x0A");
  ```

- By default the compiler treats SHORT as one bit, INT as 8 bits and LONG as 16 bits. The traditional C convention is to have INT defined as the most efficient size for the target processor. This is why it is 8 bits on the PIC. In order to help with code compatibility a new directive has been added that will allow these types to be changed. #TYPE can redefine these keywords. For example:
  #TYPE SHORT=8, INT=16, LONG=32
  Note that the commas are optional. Since #TYPE may render some sizes inaccessible (like a one bit int in the above) four new keywords have been added to represent the four ints: INT1, INT8, INT16 and INT32. Be warned CCS example programs and include files may not work right if you use #TYPE in your program.

- The ABS, LABS and FABS functions were implemented as simple C functions in stdlib.h. ABS() is now built into the compiler and accepts any type as a parameter. It returns the same type as the argument. For compatibility LABS and FABS are #defined to ABS in stdlib.h.

- The debug file (.COD) has been expanded from the Microchip definition to to include information to permit advanced debugging. This includes all C data types and better tracking information. CCS now provides a .DLL that may be used by debuggers to use this information. For example it is possible to have a watch expression like: TABLE[I].LAST-TABLE[I].FIRST
  The new .DLL knows how to evaluate the expression and provide a properly formatted string back to the debugger. Arrays and structures are also properly formatted. The stack frame may be viewed along with the parameters passed at each level and function return values may be traced. CCS is working with emulator manufactures to make use of these new capabilities.

- The PCW help filename has been changed from PCW.HLP to CCSC.HLP. The same help file is used for PCB, PCM, and PCH.

- The PCW context sensitive help has been enhanced. Placing the cursor on any C keyword, preprocessor directive or built in function and pressing F1 will bring up help on that item. In addition pressing F1 when a red error message is being shown will bring up additional help with that error.

- The PCW IDE now allows the toolbar to be customized.

- A number of internal compiler limits have been increased. For example the limitation of a macro expansion being less than 256 characters is now 32768.

Changes not documented in the May-2001 manual:

- If @filename appears on the CCSC command line command line options will be read from the specified file. Parameters may appear on multiple lines in the file.

- If the file CCSC.INI exists in the same directory as CCSC.EXE then command line parameters are read from that file before they are processed on the command line.

- printf() now accepts both 16 and 32 bit variables for %LU and %LD

- #define macros now accept the ANSI operators # and ##. In summary the #idx will be replaced with "paramx" and idx##idy is replaced with paramxparamy and is expanded if a new macro name is formed.

- #elif is now supported. For example:
  ```
  #if __device__==71
  #define adc_pin PIN_A0
  #elif __device__==74
  #define adc_pin PIN_A2
  #elif __device__==874
  #define adc_pin PIN_E0
  #else
  #define adc_pin PIN_A1
  #endif
  ```

- #IF directives now support: defined(id)
  This expression evaluates to 1 if id is a preprocessor id and 0 otherwise. For example:
  ```
  #ifdef __PCB__
  #include <16c54.h>
  #elif defined(__PCM__)
  #include <16c74.h>
  #elif defined(__PCH__)
  #include <18c658.h>
  #endif
  ```

- printf() now supports %s to insert a constant or variable string.

- Three functions have been added to make manipulation of bytes within variables easier.

  i8 = MAKE8(var,offset)
  Extracts the byte at offset from var.
  Same as: i8 = (((var >> (offset*8)) & 0xff)
  except it is done with a single byte move.

  i16 = MAKE16(varhigh,varlow)
  Makes a 16 bit number out of two 8 bit numbers.
  If either parameter is 16 or 32 bits only the lsb is used.
  Same as: i16 = (int16)(varhigh&0xff)*0x100+(varlow&0xff)
  except it is done with two byte moves.

```
i32 = MAKE32(var1,var2,var3,var4)
   Makes a 32 bit number out of any combination
   of 8 and 16 bit numbers. Note that the number
   of parameters may be 1 to 4.  The msb is first.
   If the total bits provided is less than 32 then
   zeros are added at the msb.
```

- #asm now accepts a parameter to prevent the compiler from doing automatic bank switching and optimization in the ASM code. For example:
  ```
   #byte x = 0xa0

   #asm
     clrf  x
   #endasm
  ```
  Will generate a bank switch to bank 1 and a clear of location 20 in that bank.
  ```
   #asm ASIS
     clrf  x
   #endasm
  ```
  Will just generate the clear of location 20 and ignore the bank.

- The READ_ADC() function always defaults to a 8 bit result. This may be different from previous versions where the defualt was different depending on the chip. Use #DEVICE ADC=xx to specify the desired resolution.

- Constant strings are now concatinated when they appear separated by white space. For example:
  ```
      printf("hi" "there");  is the same as printf("hithere");
  ```

Changes not documented in the July-2001 manual:

- Arrays may be defined with [] in many cases. For example:
  ```
    const char id[] = {"Hi There"};    // Same as [9]
    int x[];                          // Same as *x
    int x[] = {1,2,3};                // Same as [3]
  ```

- The #USE I2C option NOFORCE_SW is still accepted however the new FORCE_HW is what will appear in future documentation.

- #INT_???? directives now allow a NOCLEAR option to prevent the compiler from clearing the interrupt. For example:
  ```
   #INT_RTCC NOCLEAR
   isr() {
     ...
  }
  ```