ANSI COMPLIANCE MATRIX
ANS/ISO 9899-1990

CCS C Compilers for PIC® MCU and dsPIC® DSC V4.1xx

PO Box 2452
Brookfield, WI 53008 USA
262-522-6500
ccs@ccsinfo.com
http://www.ccsinfo.com

●	Full Compliance
◗	Complies except for noted item
❍	Not implemented

Part 5 Environment:

5.1.1.1 Program structure	●	
5.1.1.2 Translation phases	●	
5.1.1.3 Diagnostics	●	
5.1.2 Execution environment	●	
5.1.2.1 Freestanding environment	●	The processor is put to sleep when the program terminates.
5.1.2.2 Hosted environment	❍	This does not apply to an embedded target.
5.1.2.3 Program execution	◗	Reentrancy is not permitted.
5.2.1 Character sets	●	
5.2.1.1 Trigraph sequences	●	
5.2.1.2 Multibyte characters	❍	This optional paragraph applies to non-western languages. No support is provided for such languages.
5.2.2 Character display semantics	●	
5.2.3 Signals and interrupts	◗	Reentrancy permitted on some PIC architectures, not enabled by default.
5.2.4.1 Translation limits	●	

Part 6 Language:

6.1 Lexical elements	●	
6.1.1 Keywords	●	
6.1.2 Identifiers	●	
6.1.2.1 Scopes of identifiers	●	
6.1.2.2 Linkage of identifiers	●	
6.1.2.3 Name spaces of identifiers	●	
6.1.2.4 Storage duration of objects	●	
6.1.2.5 Types	◗	Pointers to ROM are not supported on all architectures. Long double is not supported. On 8-bit architectures, double is the same as float. On 8-bit architectures, all data types other than float are unsigned by default.
6.1.2.6 Compatible types and composite types	●	
6.1.3 Constants	●	
6.1.3.1 Floating constants	●	
6.1.3.2 Integer constants	●	
6.1.3.3 Enumeration constants	●	
6.1.3.4 Character constants	●	
6.1.4 String literals	●	
6.1.5 Operators	●	
6.1.6 Punctuators	●	
6.1.7 Header names	●	
6.1.8 Preprocessing numbers	●	
6.1.9 Comments	●	
6.2 Conversions	●	
6.2.1.1 Characters and integers	●	
6.2.1.2 Signed and unsigned integers	●	
6.2.1.3 Floating and integral	●	
6.2.1.4 Floating types	●	
6.2.1.5 Usual arithmetic conversions	●	
6.2.2.1 Lvalues and function designators	●	
6.2.2.2 Void	●	
6.2.2.3 Pointers	●	
6.3 Expressions	●	
6.3.1 Primary expressions	●	
6.3.2 Postfix operators	●	
6.3.2.1 Array subscripting	●	
6.3.2.2 Function calls	◗	See 5.2.3 for information about recursive/reentrant functions.

6.3.2.3 Structure and union members	●	
6.3.2.4 Postfix increment and decrement	●	
6.3.3 Unary operators	●	
6.3.3.1 Prefix increment and decrement	●	
6.3.3.2 Address and indirection operators	◗	Pointers to bits are not permitted. Pointers to ROM are not supported on all architectures.
6.3.3.3 Unary arithmetic operators	●	
6.3.3.4 The size of operator	●	
6.3.4 Cast operators	●	
6.3.5 Multiplicative operators	●	
6.3.6 Additive operators	●	
6.3.7 Bitwise shift operators	●	
6.3.8 Relational operators	●	
6.3.9 Equality operators	●	
6.3.10 Bitwise AND operator	●	
6.3.11 Bitwise exclusive OR operator	●	
6.3.12 Bitwise inclusive OR operator	●	
6.3.13 Logical AND operator	●	
6.3.14 Logical OR operator	●	
6.3.15 Conditional operator	●	
6.3.16 Assignment operator	●	
6.3.16.1 Simple assignment	●	
6.3.16.2 Compound assignment	●	
6.3.17 Comma operator	●	
6.4 Constant expression	●	
6.5 Declarations	●	
6.5.1 Storage-class specifiers	●	
6.5.2 Type specifiers	◗	See note on 6.1.2.5
6.5.2.1 Structure and union specifiers	●	
6.5.2.2 Enumeration specifiers	●	
6.5.2.3 Tags	●	
6.5.3 Type qualifiers	●	
6.5.4 Declarators	●	
6.5.4.1 Pointer declarators	◗	Pointers to ROM are not supported on all architectures.
6.5.4.2 Array declarators	●	
6.5.4.3 Function declarators	●	
6.5.5 Type names	◗	See note on 6.1.2.5
6.5.6 Type definitions	●	
6.5.7 Initialization	●	

6.6 Statements	●	
6.6.1 Labeled statements	●	
6.6.2 Compound statement, or block	●	
6.6.3 Expression and null statements	●	
6.6.4 Selection statements	●	
6.6.4.1 The if statement	●	
6.6.4.2 The switch statement	●	
6.6.5 Iteration statements	●	
6.6.5.1 The while statement	●	
6.6.5.2 The do statement	●	
6.6.5.3 The for statement	●	
6.6.6 Jump statements	●	
6.6.6.1 The goto statement	●	
6.6.6.2 The continue statement	●	
6.6.6.3 The break statement	●	
6.6.6.4 The return statement	●	
6.7 External definitions	●	
6.7.1 Function definitions	●	
6.7.2 External object definitions	●	
6.8 Preprocessing directives	●	
6.8.1 Conditional inclusion	●	
6.8.2 Source file inclusion	●	
6.8.3 Macro replacement	●	
6.8.3.1 Argument substitution	●	
6.8.3.2 The # operator	●	
6.8.3.3 The ## operator	●	
6.8.3.4 Rescanning & further replacement	●	
6.8.3.5 Scope of macro definitions	●	
6.8.4 Line control	●	
6.8.5 Error directive	●	
6.8.6 Pragma directive	●	
6.8.7 Null directive	●	
6.8.8 Predefined macro names	◗	__STDC__ is not #defined since the CCS implementation is not 100% compliant.
6.9 Future language directions	●	

Part 7 Library:

7.1.1 Definition of terms	●	
7.1.2 Standard headers	◗	See notes on 7.9.6
7.1.3 Reserved identifiers	●	
7.1.4 Errors <errno.h>	●	
7.1.5 Limits <float.h> and <limits.h>	●	
7.1.6 Common definitions <stddef.h>	●	

7.1.7 Use of library functions	●	
7.2 Diagnostics <assert.h>	●	
7.2.1.1 The assert macro	●	
7.3 Character handling <ctype.h>	●	
7.3.1 Character-testing functions	●	
7.3.1.1 The isalnum function	●	
7.3.1.2 The isalpha function	●	
7.3.1.3 The iscntrl function	●	
7.3.1.4 The isdigit function	●	
7.3.1.5 The isgraph function	●	
7.3.1.6 The islower function	●	
7.3.1.7 The isprint function	●	
7.3.1.8 The ispunct function	●	
7.3.1.9 The isspace function	●	
7.3.1.10 The isupper function	●	
7.3.1.11 The isxdigit function	●	
7.3.2 Character mapping function	●	
7.3.2.1 The tolower function	●	
7.3.2.2 The toupper function	●	
7.4 Localization <locale.h>	●	Provided for compatibility, however nothing special is done.
7.5 Mathematics <math.h>	●	
7.5.1 Treatment of error condition	●	
7.5.2 Trigonometric functions	●	
7.5.2.1 The acos function	●	
7.5.2.2 The asin function	●	
7.5.2.3 The atan function	●	
7.5.2.4 The atan2 function	●	
7.5.2.5 The cos function	●	
7.5.2.6 The sin function	●	
7.5.2.7 The tan function	●	
7.5.3.1 The cosh function	●	
7.5.3.1 The sinh function	●	
7.5.3.1 The tanh function	●	
7.5.4.1 The exp function	●	
7.5.4.2 The frexp function	●	
7.5.4.3 The ldexp function	●	
7.5.4.4 The log function	●	
7.5.4.5 The log10 function	●	
7.5.4.6 The modf function	●	
7.5.5.1 The pow function	●	
7.5.5.2 The sqrt function	●	
7.5.6.1 The ceil function	●	
7.5.6.2 The fabs function	●	
7.5.6.3 The floor function	●	
7.5.6.4 The fmod function	●	
7.6 Non local jumps <setjmp.h>	◗	The stack is not cleaned up on 12 and 14-bit parts, since these parts do not allow access to the stack.
7.6.1.1 The setjmp function	●	
7.6.1.2 The longjmp function	●	
7.7 Signal handling <signal.h>	●	
7.7.1.1 The signal function	●	
7.7.2.1 The raise function	●	
7.8 Variable arguments <stdarg.h>	●	
7.9 Input/output <stdio.h>	◗	A file system library is included that works with MMC/SD cards, and can be adapted to other storage mediums. The file system library does not follow standard C conventions, but is comparable. RS232 character I/O functions are provided for all compilers and is built into the compiler.
7.9.1 Introduction	◗	See note on 7.9
7.9.2 Streams	●	
7.9.3 Files 7.9.4 Operations on files 7.9.5 File access functions	◗	See note on 7.9.1 above.
7.9.6 Formatted input/output functions	◗	The following functions are absent: vfprintf, vprintf, vsprintf
7.9.6.1 The fprintf function	◗	In order to provide efficient implementation, there are some limitations imposed. For example, the formatting string must be a constant string known at compile time.
7.9.6.2 The fscanf function	❍	
7.9.6.3 The printf function	●	See note on 7.9.6.1
7.9.6.4 The scanf function	❍	
7.9.6.5 The sprintf function	●	
7.9.6.6 The sscanf function	❍	
7.9.7 Character input/output function	●	
7.9.7.1 The fgetc function	●	
7.9.7.2 The fgets function	●	
7.9.7.3 The fputc function	●	
7.9.7.4 The fputs function	●	
7.9.7.5 The getc function	●	
7.9.7.6 The getchar function	●	
7.9.7.7 The gets function	●	
7.9.7.8 The putc function	●	

7.9.7.9 The putchar function	●	
7.9.7.10 The puts function	●	
7.9.7.11 The ungetc function 7.9.8 Direct input/output functions 7.9.9 File positioning functions 7.9.10 Error-handling functions	❍	See note on 7.9.1 above.
7.10 General utilities <stdlib.h>	●	
7.10.1.1 The atof function	●	
7.10.1.2 The atoi function	●	
7.10.1.3 The atol function	●	
7.10.1.4 The strtod function	●	
7.10.1.5 The strtol function	●	
7.10.1.6 The strtoul function	●	
7.10.2.1 The rand function	●	
7.10.2.2 The srand function	●	
7.10.3.1 The calloc function	●	
7.10.3.2 The free function	●	
7.10.3.3 The malloc function	●	
7.10.3.4 The realloc function	●	
7.10.4.1 The abort function	●	
7.10.4.2 The atexit function	●	
7.10.4.3 The exit function	●	
7.10.4.4 The getenv function	●	
7.10.4.5 The system function	●	
7.10.5.1 The bsearch function	●	
7.10.5.2 The qsort function	●	The algorithm provided is the shell-metzner algorithm, not the quick sort algorithm.
7.10.6.1 The abs function	●	
7.10.6.2 The div function	●	
7.10.6.3 The labs function	●	
7.10.7.1 The mblen function	●	
7.10.7.2 The mbtowc function	●	
7.10.7.3 The wctomb function	●	
7.10.8.1 The mbstowcs function	●	
7.11 String handling	●	

<string.h>		
7.11.1 String function conventions	●	
7.11.2 Copying functions	●	
7.11.2.1 The memcpy function	●	
7.11.2.2 The memmove function	●	
7.11.2.3 The strcpy function	●	
7.11.2.4 The strncpy function	●	
7.11.3 Concatenation functions	●	
7.11.3.1 The strcat function	●	
7.11.3.2 The stncat function	●	
7.11.4 Comparison functions	●	
7.11.4.1 The memcmp function	●	
7.11.4.2 The strcmp function	●	
7.11.4.3 The strcoll function	●	
7.11.4.4 The strncmp function	●	
7.11.4.5 The strxfrm function	●	
7.11.5 Search functions	●	
7.11.5.1 The memchr function	●	
7.11.5.2 The strchr function	●	
7.11.5.3 The strcspn function	●	
7.11.5.4 The strpbrk function	●	
7.11.5.5 The strrchr function	●	
7.11.5.6 The strspn function	●	
7.11.5.7 The strstr function	●	
7.11.5.8 The strtok function	●	
7.11.6.1 The memset function	●	
7.11.6.2 The stderror function	●	
7.11.6.3 The strlen function	●	
7.12 Date and time <time.h>	●	
7.13 Future library directions	●	