WICAT Systems Incorporated

# Installation Instructions and Product Information

## UCC 4.3.0 (WMCS)

## C Compiler for WMCS

UCC62-0430-000

January 1987

Copyright 1987 by WICAT Systems Incorporated, Orem, Utah
All rights reserved
Printed in the United States of America

Receipt of this document must not be construed as any kind of commitment on the part of WICAT Systems Incorporated regarding delivery or ownership of items manufactured by WICAT Systems Incorporated.

This document is subject to change without notice.

First printing January 1987.

## Product Summary Chart

| | | |
|---|---|---|
| A. | Product name: | C Compiler |
| B. | Product code: | UCC |
| C. | Product version: | 4.3.0 |
| D. | Operating system: | WMCS 6.0.0 or later * |
| E. | Support class: | A |
| F. | Release date: | January 1987 |
| G. | Systems on which this product operates: | S150, S155, S160, S200, S220, S300, S1250, S1255, S1260, S2220, S3220, S1250H, S1255H, S1260H |
| H. | Terminals on which this product operates: | N/A |
| I. | Printers on which this product operates: | N/A |
| J. | Media on which this product is available: | 5 1/4" floppy diskette, 1/4" cartridge tape, 1/2" magnetic tape, 1/4" SCSI cartridge tape |
| K. | Prerequisite programs: | None |
| L. | Free disk-space: | 2000 kbytes |
| M. | Load time: | about 15 minutes |
| N. | Minimum system memory: | 1 Mb |

O. Publications:

| | |
|---|---|
| 188-370-101 A | The C Primer |
| 188-370-301 A | A C Reference Manual |
| 188-377-301 A | The WIMAC Programmer's Reference Manual |
| 188-370-302 A | The C Programmer's Implementation Reference Manual |
| UCC62-0430-000 | UCC 4.3.0 (WMCS) Release Notice (this document) |

P. Notes:

See Part 2 of this publication for additional steps required to complete the load procedure.

* 68881 floating-point coprocessor is only supported under WMCS 7.0.0 and later and only on 3220 systems.

Q. Product Summary:

This version of the C compiler contains support for the 68020 CPU and 68881 math co-processor found on the system 3220.

# Chapter 1

## How to Install the Program on Your Computer

**Preface** This chapter tells you how to install the program you received from WICAT Systems.

**Completion time** Item M of the Product Summary Chart lists the time needed to install this product. (You can take breaks during the installation procedure, however.)

**Prerequisites**

1. You should know how to operate your computer's tape or diskette drive. (A basic introduction to putting tapes and diskettes into their drives is forthcoming. For now, call WICAT Customer Service if you need assistance.)
2. Power to the computer and the main terminal must be on.
3. The WMCS operating system must be up and running.
4. You must be logged on as the SYSTEM user.

**Materials** The WICAT tape(s) or diskette(s) containing your new program.

Perform the following steps to make sure you have the right version of the operating system on your computer.

Step 1

Make sure you have satisfied all the prerequisites listed at the beginning of this chapter.

When you have logged on as the SYSTEM user, the following prompt appears on your terminal screen:

```
SYSTEM>
```

Step 2

Type the following:

```
mcsver
```

This tells the computer that you want to know what version of the WMCS operating system is running on it.

Step 3

Strike [RETRN]

This kind of report appears:

```
WMCS, version 6.1.1, 06-June-1986
```

NOTE: For versions 6.0.0 through 6.1.0 of WMCS, typing **mcsver** will not work. However, typing the old **version** command for these versions will always yield "6.0.0".

Step 4

Compare the version listed in the report with the version listed in item D of the Product Summary Chart.

If the version number in the report on your screen does not match the version number listed on the Product Summary Chart, the program you are installing will not run on your computer.

NOTE: If you do not have the correct version of the WMCS operating system, call WICAT Systems Customer Service.

Step 5

If the version of the WMCS running on your system is 6.1.0 or later (you determined this in step 3), type the following:

**sp**

If you have a version of the WMCS prior to 6.1.0, type the following:

**sp :kbytes**

This command tells the computer that you want to know how the space on the default disk is being used.

Step 6

Strike [RETRN]

The following kind of report appears on the screen:

```
Disk usage of __WICAT_DS0
Free =        7944.0 kbytes.
Used =       66269.0 kbytes.
Bad  =           1.0 kbytes.
Total=       74214.0 kbytes.
```

Step 7

Compare the number of kilobytes (kbytes) you have free on your disk with item L in the Product Summary Chart, which lists the number of kilobytes required to load the program.

The number of free kilobytes on your disk must be at least as great as the number listed on the Product Summary Chart.

NOTE: If your computer's disk does not have enough free space, and you have purged or deleted all unnecessary files, call WICAT Systems Customer Service.

Perform the following steps to make sure you have on your computer the correct versions of the other programs required by the program you are installing.

Step 8

Write down (in the blanks provided below) the three-letter product code and the version number of each program listed in item K of the Product Summary Chart.

If no prerequisite programs are listed in item K, go to step 12.

| Product code | Version number |
|---|---|
| _____ | _____ |
| _____ | _____ |
| _____ | _____ |

Now perform steps 9-10 for each program you listed above.

Step 9

Type the three-letter product code, followed immediately by **ver.**

For example, if you had written the product code WSR and the corresponding version number 6.2.1 in the foregoing chart, you would type the following:

```
wsrver
```

Step 10

Strike [RETRN]

This kind of report appears on the screen:

```
WISE Runtime, version 6.2.1, 15-Oct-86
```

The version number in the report must match the number you wrote down for the product in step 8.

NOTE: If you do not have the correct version of a program listed in step 8 (also in item K of the Product Summary Chart), call WICAT Systems Customer Service.

How to Install the Program on Your Computer

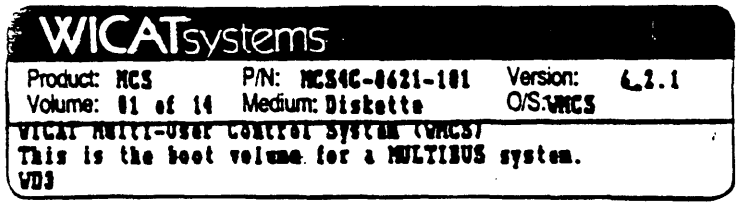
Step 11 | When you have the correct version of each prerequisite program installed on your computer, you are ready to install the new program you received from WICAT Systems.

Perform these steps to install (copy) the new program onto your computer's disk.

Step 12 | Make sure no one but you is logged on to the system.

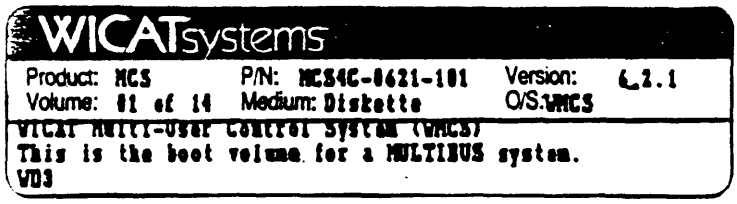
Step 13 | If the new program is contained on several diskettes or tapes, select the tape or diskette labeled vol. 1.

This is where you find the volume number on the label of a tape or diskette:




Step 14 | Look at the label on the tape or diskette to determine what type of drive it requires (you will need this information in step 18).

This is where you find this information on the label of a tape or diskette:




Step 15 | Put the tape or diskette into the appropriate drive. If you are loading 1/2" magnetic tape, be sure the tape is properly loaded and that the tape drive is "on-line." If you don't know how to operate the drive, call WICAT Systems Customer Service.

Step 16 | Type the following:

**load**

Step 17 | Strike [RETRN]

This prompt appears, asking for the type of drive from which you wish to load the product:

```
**************** SOFTWARE PRODUCT INSTALLATION UTILITY ****************

What device would you like to use?
```

| DEVICE | DESCRIPTION |
|---|---|
| dx0 | 5.25 inch Diskette |
| ct0 | Cartridge Tape |
| mt0 | Cipher Tape |
| st0 | SCSI Tape |
| other | (none of the above devices) |

```
Which DEVICE do you wish to load the product from (press [SPACE]
            to change the value or [RETRN] to accept the value): dx0
```

Step 18 | Look at the following chart to determine the name of the drive where you put the tape or diskette (you determined the type of media in step 14):

| TYPE OF MEDIA | NAME OF DRIVE |
|---|---|
| .5 Mag. Tape | _MT0 |
| Cartridge Tape | _CT0 |
| Diskette | _DX0 |
| SCSI Cart. Tape | _ST0 |

Step 19 | Strike the spacebar until the letters next to the colon at the end of the prompt change to the name of the drive where you put your tape or diskette.

Step 20

Strike [RETRN]

The following line appears on the screen:

```
Please enter the product identifier(s):
```

Step 21

Type the three-letter product code listed in item B of the Product Summary Chart.

For example, if the product code were WSA you would type the following:

```
wsa
```

Step 22

Strike [RETRN]

Lines like these (without highlighting) appear on the screen:

```
__WICAT_MT0 Mounted. Label is "WSA_6.2.1_vol1".
Is this the correct volume (Y or N)? >
```

The items enclosed in quotation marks (highlighted in the message above) are the label that is coded onto the tape or diskette.

With some computers the following lines also appear on the screen, immediately above the foregoing message:

```
LOAD      : Mounting __WICAT_MT0
CAUTION   : Status = -269
MESSAGE   : The specified device is write-protected.
```

Disregard these three lines. They simply inform you that the tape or diskette in the drive is protected from being written on or erased.

Step 23

Check the label listed on the screen to see if it matches the correct product code and product version number (listed in items B and C of the Product Summary Chart), and if the volume number is 1.

If the label is correct, go to the next step.

If the product code, product version number, or volume number is not correct, type **n** and then check the diskettes or tapes that you received from WICAT Systems to make sure you have the right diskette or tape in the drive. Once the correct volume is in the drive, return to step 22.

NOTE: If you cannot find a diskette or tape whose printed label matches the information in the Product Summary Chart, call WICAT Systems Customer Service.

Step 24

To tell the computer that the tape or diskette in the drive is the correct volume, type the following:

**y**

Lines like these appear on the screen:

********** Installing WSA from device _MT0 **********

What disk number do you wish to load the product on? (press [SPACE] to change the value or [RETRN] to accept the value): 0

Step 25

Strike the spacebar until the number after the colon changes to the number of the disk on which you wish to load the product.

NOTE: If your system has only 1 disk, it is disk 0.

Step 26 | Strike [RETRN]

Lines like the following begin to fill the screen:

```
__NODE_DS0/SYSEXE.USERS/WSABKUP.COM.1 restored as SYS$DISK/SYSEXE.USERS/
WSABKUP.COM
__NODE_DS0/SYSEXE.USERS/WSALOAD.COM.1 restored as SYS$DISK/SYSEXE.USERS/
WSALOAD.COM
```

The computer is now copying the program onto the disk.

If the program you are installing came to you on more than one diskette or tape, perform steps 27-33. If the new program came on only one diskette or tape, go to step 34.

Step 27 | If the copy of the new program came on several diskettes or tapes, the terminal beeps and displays lines like these each time you need to insert another diskette or tape:

```
Insert volume 2
Press RETURN when ready
```

Step 28 | When the foregoing message appears, remove the diskette or tape that is in the drive.

Step 29 | Take the next diskette or tape in the set and put it into the drive.

Step 30 | Strike [RETRN]

Lines like these appear on the screen:

```
_ WICAT_MT0 Mounted. Label is "WSA_6.2.1_vol2".
Is this the correct volume (Y or N)? >
```

Step 31 | If the volume number in the report on your screen matches the volume number you were prompted for in step 27, go on to step 32.

If the volume numbers do not match, type **n** and then check the diskettes or tapes that you received from WICAT Systems to make sure you have the right diskette or tape in the drive. Once the correct volume is in the drive, return to step 30.

Step 32 | To tell the computer that the tape or diskette in the drive is the correct volume, type the following:

**Y**

Step 33 | Repeat steps 27-32 until the following prompt appears on the screen:

Press [RETRN] to display verification results

Then go to step 34 to complete the installation procedure.

Perform these steps to complete the installation procedure.

Step 34 | When the computer has finished copying the program onto its disk, the following line appears on the screen:

Press [RETRN] to display verification results

Step 35 | When the foregoing line appears, strike [RETRN]

The following kind of information rolls down the screen, pushing the text above it off of the screen:

Verification Summary

| File Name | Verified | Size | Sum |
|---|---|---|---|
| WSADISK/AUTHOR/EXECUP.COM | yes | good | good |
| WSADISK/AUTHOR/FSTAT.COM | yes | good | good |
| WSADISK/AUTHOR/WSAUP.COM | yes | good | good |

Or, if you are installing one of WICAT Systems' educational programs, only the following lines appear:

Verification Summary

| File Name | Verified | Size | Sum |
|---|---|---|---|

In either case, the foregoing lines indicate that the computer is checking (verifying) the files that it copied onto its disk to determine whether they were copied correctly. The verification process may take several minutes.

When the computer has finished the verification, a line like the following appears on the screen:

86 files checked. 0 files failed.

Step 36 | Look at the number of files failed. This number must be zero.

NOTE: If any files failed, you must repeat the installation procedure, beginning with step 12. If any files fail on a second attempt, call WICAT Systems Customer Service.

Step 37 | After the files are verified, several messages could be displayed on the screen. These messages simply tell you what the computer is doing. For example, the following messages could appear:

```
FYI:  Setting file protection for WSA....
      Purging release directories....
```

Step 38 | When a line like the following appears on the screen, you are finished loading the program:

```
********** LOAD OF WSA 6.2.1 COMPLETED **********
```

Step 39 | Remove the tape or diskette from the drive.

Step 40 | Store the tape(s) or diskette(s) in a cool, dry place.

Step 41 | Look at item Q in the Product Summary Chart.

If item Q indicates that you must perform additional steps to complete the installation procedure, go to Chapter 2 now and follow the instructions for "Completing the Installation Procedure."

Otherwise, you have finished installing the program.

Chapter 2

Notes and Technical Information

**Completing the Load Procedure (3220 system only)**

The file SYS$DISK/SYSLIB/DEVICEUP.COM is configured to use the 68881 floating-point co-processor by default. However, loading C 4.3.0 will cause this default to be ignored in favor of the definition in the file SYS$DISK/SYSLIB.SGS/SGSUP.COM. The file appears like this:

```
!!@(#)WICAT Systems Inc., SGS setup
!============================================================
! SGSUP.COM - This file is executed by the APPLICUP.COM file
! each time the system is booted. Its purpose is to install
! all privileged SGS utilities and initialize floating-point.
! The contents of this file will be replaced with each release
! of the SGS. You should change this file to initialize any
! floating-point hardware you have on your system. Currently
! it sets up only the floating-point software emulation.
!============================================================
install sys$disk/sysexe.sgs/ll.exe :privilege=writephys;
fpmgr lib2 :add
```

This effectively changes the default floating-point to software. To retain use of 68881 floating-point, use the VEW editor (see the VEW User Reference Manual) to delete the "fpmgr" line from this file.

**Completing the Load Procedure (non-3220 systems)**

If floating-point hardware (FFP or SKY) is to be used the floating-point class handler must be loaded at boot time. To make sure it is loaded type the following command:

**sysprof**

This kind of display appears:

```
                              System Configuration
-------------------------------------------------------------------------------
System Model      : 156                          Boot Device Driver : SMD$156
Operating System  : KERNEL                       Current Year       : 1985

Class Handlers:     TTY          DISK            TAPE            KSAM
                  NOMATH1010     QUEUE         NOFPOINT          NETWORK

                              Device Configuration
-------------------------------------------------------------------------------
Device Name   : BQ0          Device Driver : QUE$156
Drive ID      : 0B0          Drive Type    :
     .
     .
     .
```

If the item **NOFPOINT** is listed in the "Class Handlers" section, use the arrow keys to position the cursor on that item and use the spacebar to change **NOFPOINT** to **FPOINT.** By specifying **FPOINT**, you tell the system to load the floating-point class handler when the system is booted.

Strike [ESC] [ESC], type **ex** to exit sysprof.

If floating-point hardware (FFP or SKY) is to be used, you must modify the file SYS$DISK/SYSLIB.SGS/SGSUP.COM so that the desired type of floating-point will be set up at boot time. (The file currently sets up software floating-point.) Use the VEW editor (see the VEW User Reference Manual) to change the file. The file appears like this:

```
!!@(#)WICAT Systems Inc., SGS setup
!============================================================
! SGSUP.COM - This file is  executed by the  APPLICUP.COM file
! each time the  system is booted.  Its purpose is to  install
! all  privileged SGS utilities and initialize floating-point.
! The contents of this file will be replaced with each release
! of the  SGS.  You should change this file to  initialize any
! floating-point  hardware you have on your system.  Currently
! it sets up only the floating-point software emulation.
!============================================================
install sys$disk/sysexe.sgs/ll.exe :privilege=writephys;
fpmgr lib2 :add
```

Change the last line of the file to read as follows:

```
        fpmgr skyl :add
```

OR

```
        fpmgr ffpl :add
```

depending upon whether you want SKY or FFP hardware floating-point as the default mode.

NOTE: If you have WMCS version 6.1.1 or later, delete this line from the file altogether. Then edit SYS$DISK/SYSLIB/DEVICEUP.COM, find the "fpmgr" line, and edit it to install the appropriate floating-point as shown above.

If your floating-point hardware board is already installed in your system, or if you are going to use software floating-point, type the following command:

```
        shutdown +0 :reboot
```

This step reboots the system so that UCC 4.3.0 is installed and ready to use. You have completed the load procedure.

If your floating-point hardware board is not installed you should call WICAT Customer Service for assistance.

When you are ready to install the board you must shutdown the system. To do this type:

```
shutdown +0
```

After the message "Shutdown complete." appears, turn off the system power switch. At this time the board can be installed.

Once the board is installed, turn on the system power and boot the system (see the WMCS System Manager Reference Manual). The floating-point hardware and UCC 4.3.0 are now installed and ready to use. You have completed the load procedure.

**Executing the Product with Sample Programs/Files**

There are example programs in directory /UCC/. They are EX1.C, EX2.C, EX3.C, and EX4.C. Copy them to a directory where you have write access. To compile and execute EX1.C type the following:

```
compile ex1.c
ex1
```

The program should produce the following output:

```
hello world
```

To compile and execute EX2.C type the following:

```
compile ex2.c
ex2 123456 123 yyy xxxx
```

The program should produce the following output:

```
123456 - 6
123 - 3
yyy - 3
xxxx - 4
```

To compile and execute EX3.C type the following:

```
compile ex3.c :float=lib2 :lib=libm
ex3
```

The program should produce the following output (timings may vary):

```
add time: 0,475
a: 0x40c3880000000000 1.0000000000e+04
mul time: 0,791
a: 0x40052460e89431d6 2.6427629633e+00
cmul time: 0,807
a: 0x4005a3756ed911e0 2.7048138294e+00
```

```
div time: 0,831
a: 0x3fd8379280b1c456 3.7839186256e-01
cdiv time: 0,839
a: 0x3fecf4cc6b4d94b0 9.0488263090e-01
sine time: 0,3028
a: 0x3fd5d0bdb48f1f91 3.4086554177e-01
sqrt time: 0,2032
a: 0x3ffc5bf891b4ef6a 1.7724538509e+00
log time: 0,14054
a: 0x3ff250d048e7a1bd 1.1447298858e+00
exp time: 0,11515
a: 0x403724046eb09338 2.3140692633e+01
```

If you have a 3220 system, recompile EX3.C with the following:

```
compile ex3.c :lib=libm881
ex3
```

The resulting executable should run **much** faster.

The program ex4.c is a self-replicating program. To compile, execute, and test EX4.C type the following:

```
compile ex4.c
ex4 > tmp
dumpdiff ex4.c tmp
```

There should be no differences reported by the DUMPDIFF utility.

## Enhancements and New Features

Version 4.3.0 of the C compiler now supports the 68881 math co-processor on 3220 systems. A new floating point preprocessor and math library were added as part of this support.

## The Linker (LL.EXE)

A typographical error in the default prolog/epilog files has been corrected. The spelling of a linker defined symbol has been corrected. This should not affect any user programs.

## The COMPILE Utility

The :target= switch has been added to allow specification of the target CPU type. The accepted values are "68000" and "68020". If you type :target=68020 COMPILE will attempt to take advantage of two features of the WICAT 68020 based systems. First, if :optimize was specified COMPILE will invoke the C optimizer pass with an option to suppress "stack probes". This usually produces smaller and faster binaries. Second, COMPILE will invoke the 68881 floating point preprocessor (A881.EXE) to generate co-processor instructions for floating point operations. This produces faster binaries for floating point intensive programs. If you type :target=68000, COMPILE will make no default assumptions about what hardware is available. This is useful for producing binaries which will run on either WICAT 68000 or 68020 based systems. Note that programs built with the :target=68020 switch will **not** run on systems other than the 3220.

For a complete description of COMPILE and all its options, please refer to the COMPILE manual page in Appendix B of this document.

## The 68881 Floating-point Preprocessor (A881.EXE)

This preprocessor generates 68881 co-processor instructions for use on 3220 systems. It is invoked by COMPILE if you are running on a 3220 system or the :target=68020 switch is specified.

## The 68881 Math Library (LIBM881.LIB)

A new version of the math library is included in this release for use on 3220 systems. SYS$DISK/COMLIB/LIBM881.LIB is functionally equivalent to the old math library but was built to use 68881 instructions whenever possible. This can make a dramatic speed difference in programs that use routines such as sin() and cos(). This library is **not** the default library and must be explicitly referenced via :lib=libm881 on the COMPILE command line (as in the EX3.C example above). Note that programs linked with the 68881 math library will **not** run on systems other than the 3220.

## Bug Fixes

| SAR # | Description |
|---|---|
| UCC0294 | COMPILE no longer displays a file version number of zero. It now gives the correct version number. |
| UCC0291<br>UCC0297 | LLLIB (and LLRAN) command lines are no longer artificially limited to about 3000 characters. The only limit is that which is imposed by the underlying version of the WMCS. For WMCS 7.0 and above it is essentially unbounded, for earlier versions, it is still about 3000 characters. |

**Notes and Warnings**

If you intend to install Pascal on your system, it must be loaded before C. If C has already been installed, you must backup C, load Pascal, and then restore C.

For information on getting the best floating-point performance, refer to the section on floating-point in the C Programmer's Implementation Reference Manual.

The C optimizer may require more than 1 Mb of memory on some programs. One way to reduce the amount of memory used is to reduce the size of functions/subroutines.

Programs that are compiled using UCC 4.3.0 will not execute on versions of WMCS prior to 5.1. If they are compiled with direct code for FFP, they will not execute on versions of WMCS prior to 6.0 (since FFP was first supported by WMCS version 6.0).

Programs that are compiled with the COMPILE :target=68020 switch or linked with the 68881 math library will not execute on systems other than the 3220 running WMCS 7.0.0 or above.

The /COMLIB/ directory contains utilities and libraries used by both Fortran and C. It is included with the release of either language. Because they share this directory, you must use the same version of both languages. You can not use F77 4.1 with UCC 4.3.0 or vice versa. If you have Fortran on your system, you must upgrade it to F77 version 4.3.0.

Programs which use the SKY floating-point hardware may return erroneous results if the system is a 150/155/160 with an ADEI cartridge tape controller and the ADEI controller has not been updated to revision 3.3 (effective 13 August 1984).

For the FFP to operate, it requires the following boards at the specified revision numbers in the System 2220:

| Board | Revision # | Part # |
|---|---|---|
| Cache CPU | rev. C.8 | 810-141-001 |
| FFP | rev. B.4 | 810-144-001 |
| PF Memory | rev. B.6 | 810-123-001 |

Known Bugs

LLLIB emits spurious warnings about duplicate entry points for the symbol 'fltused' followed by a final message about terminating with N warnings. [ UCC0290 ]

If a computation produces infinity as a result (as it would if an overflow occurred and that exception was masked), and the program tries to print the result, it will instead print the largest finite number (approximately 1.79769e+308).

Statements of the form:

<char,short> <op>= <double>;

may produce compiler loops. The work around is to rewrite as:

<char,short> = <char,short> <op> <double>

Functions with more than 32 Kb of local storage and containing complex expressions may produce errors of the form:

"cannot create a temporary, simplify expression or reduce the amount of automatic storage in the function"

# Appendix A

## Contents of the Release Volumes

These are the files on the release volumes:

| File designation | Description |
|---|---|
| /comlib/ | |
| ar.exe | archive utility (temporary, used by LLLIB) |
| comup.com | assigns logical names |
| copt.exe | C optimizer (C and FORTRAN 77) |
| libc.lib | C library |
| libc100.lib | C library (100 files, no floating-point, for COBOL) |
| libcnofp.lib | C library with no floating-point |
| libm.lib | math library |
| libm881.lib | math library for 68881 co-processor |
| ran.exe | library utility (temporary, used by LLLIB) |
| /sysexe.sgs/ | |
| a881.exe | preprocessor (68881 co-processor) |
| affpl.exe | preprocessor (FFP hardware) |
| alib2.exe | preprocessor (software emulation) |
| askyl.exe | preprocessor (SKY hardware) |
| compile.exe | compile driver utility |
| ll.exe | linking loader |
| llc.exe | linking loader compile utility |
| lllib.exe | linking loader packaged library utility |
| llran.exe | linking loader directory library utility |
| li.exe | utility to display symbols in image files |
| wimac.exe | assembler |
| /sysexe.users/ | |
| uccbkup.com | product backup command file |
| uccload.com | product installation command file |

| | |
|---|---|
| /syshlp.users/ | text help files |
| compile.hlp | |
| lllib.hlp | |
| llran.hlp | |
| li.hlp | |
| | |
| /syslib.sgs/ | |
| aaaaaa.llc | standard LL directives source |
| aaaaaa.wu | standard LL directives (object) |
| epilog.llc | user-definable LL directives source |
| epilog.wxn | user-definable LL directives (object) |
| prolog.llc | user-definable LL directives source |
| prolog.wxn | user-definable LL directives (object) |
| prolognfp.llc | LL directives for no floating-point (source) |
| prolognfp.wxn | LL directives for no floating-point (object) |
| sgsup.com | assigns logical names, installs floating-point |
| zzzzzz.llc | standard LL directives source |
| zzzzzz.wu | standard LL directives (object) |
| | |
| /ucc/ | |
| ex1.c | example #1 source |
| ex2.c | example #2 source |
| ex3.c | example #3 source |
| ex4.c | example #4 source |
| ccom.exe | C compiler |
| cpp.exe | C preprocessor |
| execup.com | assigns logical names |
| uccup.com | assigns logical names |
| ucc0420.cks | checksums for the files in this release |
| | |
| /ucc.include/ | header files |
| alarm.h | alarm signals |
| aout.h | a.out format |
| aouthdr.h | a.out header format |
| ar.h | archive |
| assert.h | assert macros |
| core.h | machine dependent core |
| ctype.h | isalpha, isupper, etc. macros |
| curses.h | terminal independence subroutines |
| dial.h | modem connection |
| dumpresto.h | dump/restore file utilities |
| errno.h | system error numbers |
| execargs.h | physical addresses of arguments |
| fatal.h | fatal errors |
| fcntl.h | file control (R_only, etc.) |
| filehdr.h | system file format |
| ftw.h | n/a vax |
| grp.h | group id |
| ioctl.h | I/O control functions |
| ldfcn.h | file access macros |

| | |
|---|---|
| linenum.h | line number for object routines |
| macros.h | misc system macros |
| malloc.h | malloc structures |
| math.h | trancendentals |
| memory.h | global declarations for memory routines |
| mnttab.h | mount tables |
| mon.h | profile monitor |
| nan.h | not a number definitions |
| nlist.h | symbol table entries |
| prof.h | profiling definitions |
| pwd.h | passwd entries |
| regexp.h | regular expression routines |
| reloc.h | relocation |
| scnhdr.h | common object file section headers |
| search.h | search hash, tree |
| setjmp.h | definition for jumps |
| sgtty.h | stty flag definition |
| signal.h | signal definitions |
| stand.h | standalone application definitions |
| stdio.h | standard I/O definitions |
| storclass.h | memory storage class definitions |
| string.h | global declaration string functions |
| symbol.h | symbol table entry definitions |
| syms.h | symbol definitions |
| term.h | terminal function definitions |
| termio.h | terminal I/O definitions |
| time.h | structure for time/date |
| tpdefs.h | n/a vax |
| unctrl.h | uncontrol macro |
| ustat.h | file system status structure |
| utmp.h | process accounting |
| values.h | system numeric definitions |
| varargs.h | variable argument list macros |
| | |
| /ucc.include.m68k/ | header files for MC68000 processor |
| machine.h | mc68000 machine dependencies |
| | |
| /ucc.include.sys/ | system header files |
| acct.h | system accounting |
| blkfwd.h | bad-block forwarding |
| bootblock.h | bootblock structures and macros |
| buf.h | buffer allocation |
| calclock.h | calendar clock definitions |
| callo.h | callout structure for OS interrupts |
| clock.h | clock definitions |
| conf.h | device configuration |
| context.h | context definitions |
| cpuid.h | CPU identification definitions |
| crtctl.h | cursor control codes |
| deviceid.h | peripheral device information |

| | |
|---|---|
| dir.h | directory file format |
| diskio.h | disk driver definitions |
| diskmap.h | partition descriptions |
| elog.h | error logging |
| erec.h | error record structure |
| err.h | error buffer areas |
| errno.h | system error numbers |
| fblk.h | free blocks |
| ffp.h | FFP floating-point definitions |
| file.h | file description structures |
| filsys.h | file system structure |
| fp.h | floating-point hardware |
| gd.h | generic disk information |
| ino.h | inode structure definition |
| inode.h | inode macro definitions |
| io.h | device specific parameters |
| iobuf.h | I/O buffer structures |
| ipc.h | inter process communication |
| lock.h | locking flags |
| locking.h | file locking |
| map.h | memory map |
| mmu.h | memory mapping |
| mount.h | mount structure |
| msg.h | inter process communication |
| mtio.h | mag tape I/O control |
| param.h | system parameters |
| proc.h | process control block |
| reg.h | register definitions |
| scat.h | scatter memory |
| seg.h | memory management definitions |
| sem.h | semaphore facility |
| shm.h | shared memory facility |
| signal.h | signal definitions |
| sky.h | SKY floating-point definitions |
| stat.h | file status structure (system) |
| sxt.h | multiplexed channels definitions |
| sysinclud.h | includes all system header files |
| sysinfo.h | system information definitions |
| sysmacros.h | system macros |
| systm.h | misc system global declarations |
| termio.h | terminal I/O definitions |
| text.h | text structure -- pure procedure |
| times.h | system times |
| tty.h | tty structures (clist) |
| types.h | system data types |
| user.h | user process block |
| utsname.h | system name information |
| var.h | misc system variables |
| wioctl.h | controller drivers |

| | |
|---|---|
| /ucc.include.wmcs/ | special header files for certain WMCS-specific applications |
| rtenv.h | miscellaneous definitions |
| stat.h | stat structure |
| stdio.h | standard I/O definitions (100 files - for COBOL) |
| timeb.h | timeb structure |

# Appendix B

# COMPILE Utility Description

COMPILE

## Functional Description

Use this command to compile C, FORTRAN, Pascal, and/or assembly language source files into object files, and to link object files together to produce an executable file.

## Command Line Syntax

Mnemonic compile

Required parameter File list

Switches

| | | | |
|---|---|---|---|
| File selection | :before= | :filesize= | :typeselect= |
| | :class= | :mod | :uic= |
| | :exclude= | :since= | |
| Suppress compilation | :assemble | :load | :process |
| | :fpreprocess | :preprocess | |
| General switches | :deletetemp | :optimize= | :subpass= |
| | :floatingpoint= | :output= | :target= |
| | :listing | :startup | :tempprefix= |
| | :listing= | :startup= | :verbose |
| | :log | :subopt= | :warnings |
| | :optimize | | |
| C switches | :define= | :include= | :undefine= |
| FORTRAN switches | :case | :maxhash= | :undefined |
| | :int= | :maxstno= | :warnf66 |
| | :maxctl= | :onetrip | :xref |
| | :maxequ= | :range | :xref= |
| | :maxext= | | |
| Pascal switches | :define= | :hexlist= | :octlist= |
| | :dispose | :include= | :overflow |
| | :hexlist | :octlist | :range |
| LL switches | :epilog= | :prolog= | :runtime= |
| | :libraries= | :reloc | :strip |
| | :prefix= | | |

## Parameters

| | | |
|---|---|---|
| File list | Function | Use this parameter to specify the list of files to be compiled and loaded together. |
| | Default | None. |
| | Syntax | Standard syntax for file lists. Wildcards are allowed. |

## Switches

| | | |
|---|---|---|
| :assemble | Function | Use this switch to specify that assembly language source files are to be assembled into object files. Assembly-language source files are tiles specified in the file list with a .S extension, plus any output files from the floating-point preprocessors. |
| | Default | :assemble, i.e., assembly-language files are assembled into object files. |
| | Syntax | Type **:noassemble** to suppress the assembler and the linker/loader phases. Files specified in the file list with a .S extension will be left untouched. Assembly-language output files from the floating-point preprocessors will be saved in the default directory with the same names as the corresponding source files, but with .S extensions. |
| :before= | Function | Use this switch to select only those files that were specified in the file list and were created/modified before the specified date and time. |
| | Default | Selects all files that were specified in the file list. |
| | Syntax | Type **:before=** followed by a date and/or time in the standard date and time syntax. |
| :case | Function | Use this switch with FORTRAN source files to allow upper- and lower-case names to be distinct. |
| | Default | :nocase, i.e., upper- and lower-case names are not distinct. |

| | | |
|---|---|---|
| | Syntax | Type **:case** to allow upper- and lower-case names to be distinct. |
| :class= | Function | Use this switch to select only those files that were specified in the file list and reside on the class(es) of devices given. |
| | Default | Selects all files that were specified in the file list. |
| | Syntax | Type **:class=** followed by a list of device designations, separated by commas, any one of which may contain wildcard characters. |
| :define= | Function | Use this switch with C source files to define macros for the C preprocessor, or with Pascal source files to define constants. C macro definitions given by this switch can be cancelled by the :undefine= switch. |
| | Default | No macros or constants are defined except some macros predefined by the C preprocessor itself. |
| | Syntax | Type **:define=** followed by a list of values, separated by commas. Each value must be in one of the following forms: |

| | |
|---|---|
| name=value | Assigns the value to the name |
| name | Assigns to name a value of 1 for C, true for Pascal |

| | | |
|---|---|---|
| :deletetemp | Function | Use this switch to cause temporary files to be deleted automatically. |
| | Default | :deletetemp, i.e., temporary files are automatically deleted. |
| | Syntax | Type **:nodeletetemp** to preserve temporary files. |
| :dispose | Function | Use this switch on Pascal source files to specify that calls to NEW are to be of the NEW/DISPOSE variety. |
| | Default | Calls to NEW are of the MARK/NEW/RELEASE variety. |
| | Syntax | Type **:dispose** to cause calls to NEW to be of the NEW/DISPOSE variety. |
| :epilog= | Function | Use this switch to specify a custom epilog file to be used for linking. |
| | Default | A standard epilog file is used. |

| | | |
|---|---|---|
| | Syntax | Type **:epilog=** followed by the name of the epilog file. If no directory is specified, library directories are searched. If no extension is given, the appropriate extension is appended. |
| :exclude= | Function | Use this switch to select only those files or devices that were specified in the file list and which do not match any of the files specified as the value of the :exclude= switch. |
| | Default | Selects all files or devices specified in the file list. |
| | Syntax | Type **:exclude=** followed by a list of file or device designations, separated by commas, any one of which may contain wildcard characters. |
| :filesize= | Function | Use this switch to select only those files that were specified in the file list and are of the specified size. |
| | Default | Selects all files that were specified in the file list. |
| | Syntax | Type **:filesize=** followed by a numeric range of files sizes in K. |
| :floatingpoint= | Function | Use this switch to cause the compilers to generate code for a specific kind of floating-point hardware and/or software. This switch also selects the floating-point preprocessor to be used. |
| | Default | On MC68000-based systems, :floatingpoint=lib, i.e., the generic floating-point library is used. On MC68020-based systems, :floatingpoint=881, i.e., the MC68881 floating-point coprocessor is used. |
| | Syntax | Type **:floatingpoint=** followed by one of the following: |

LIB — Current version of the generic floating-point library (same as LIB2).

LIB2 — Version 2 of the generic floating-point library.

SKY — Current version of the SKY floating-point board (same as SKY1).

SKY1 Version 1 of the SKY floating-point board.

FFP Current version of the FFP floating-point board (same as FFP1).

FFP1 Version 1 of the FFP floating-point board.

881 The MC68881 floating-point coprocessor.

NOFP No floating-point (produces smaller image files).

:fpreprocess

Function: Use this switch to specify that pseudo-assembly language source files are to be translated into actual assembly-language source files by a floating-point preprocessor. Pseudo-assembly language source files are files specified in the file list with a .K extension, plus any output files from compilers and the C optimizer. The floating-point preprocessor to be used is selected by the :floatingpoint= switch.

Default: :fpreprocess, i.e., pseudo-assembly language files are translated to actual assembly-language files.

Syntax: Type **:nofpreprocess** to suppress the floating-point preprocessor, assembler, and linker/loader phases. Files specified in the file list with a .K extension will be optimized if the C optimizer is selected but will otherwise be left untouched. Pseudo-assembly language output files from the compilers and the C optimizer will be left in the current default directory with the same names as the corresponding source files, but with .K extensions.

:hexlist

Function: Use this switch on Pascal source files to generate a hexadecimal assembly listing in the current default directory in files with the same name as the corresponding source files, but with .OLS extensions.

Default: :nohexlist, i.e., no hex assembly listing is generated.

Syntax: Type **:hexlist** to generate a hex assembly listing.

| | | |
|---|---|---|
| :hexlist= | Function | Use this switch on Pascal source files to specify a file in which a hexadecimal assembly listing is generated. |
| | Default | A hex assembly listing is not generated. |
| | Syntax | Type **:hexlist=** followed by a file designation. Wildcards are not allowed. |
| :include= | Function | Use this switch on C (or Pascal) source files to specify additional directories in which the C preprocessor (or the Pascal compiler) is to search for include files. Directories specified by the :include= switch are searched first, followed by predefined "standard" directories. |
| | Default | Only the predefined "standard" directories are searched. |
| | Syntax | Type **:include=** followed by a list of directory specifications, separated by commas. |
| :int= | Function | Use this switch on FORTRAN source files to change the default size of FORTRAN integers. |
| | Default | :int=4, i.e., FORTRAN integers are 4 bytes long. |
| | Syntax | Type **:int=** followed by one of the following:<br>2 Integers are 2 bytes long<br>4 Integers are 4 bytes long |
| :libraries= | Function | Use this switch to specify the names of library files to be used by the linker/loader in addition to the standard libraries. |
| | Default | Only the standard libraries are used. |
| | Syntax | Type **:libraries=** followed by a list of file names. Wildcards are not allowed. Device and directory names may not be given here but must instead be specified by the :prefix= switch. |
| :listing | Function | Use this switch on FORTRAN, Pascal, and assembly source files to generate a source listing in the current default directory in files with the same name as the corresponding source files, but with .PRN extensions. |
| | Default | :nolist, i.e., no source listing is generated. |
| | Syntax | Type **:list** to generate a source listing. |

| | | |
|---|---|---|
| :listing= | Function | Use this switch on FORTRAN, Pascal, and assembly source files to specify a file in which a source listing is generated. |
| | Default | A source listing is not generated. |
| | Syntax | Type **:listing=** followed by a file designation. Wildcards are not allowed. |
| :load | Function | Use this switch to specify that object files are to be linked to create an executable file. Object files are files specified in the file list with a .W, .WC, or .WP extension, plus any output files from the assembler and LLC. |
| | Default | :load, i.e., object files are linked to create an executable file. |
| | Syntax | Type **:noload** to suppress the linker/loader phase. Files specified in the file list with a .W, .WC, or .WP extension will be left untouched, and object files from the assembler and LLC will be left in the current default directory with the same names as the corresponding source files, but with .W extensions. |
| :log | Function | Use this switch to specify whether log messages are displayed. (Log messages are informational displays that indicate what the utility is doing.) |
| | Default | The value specified by the OPTION command. |
| | Syntax | Type **:log** or :nolog to override the default. |
| :maxctl= | Function | Use this switch with FORTRAN source files to specify the maximum level that IF and DO statements can be nested. |
| | Default | :maxctl=10, i.e., IF and DO statements can be nested 10 levels deep. |
| | Syntax | Type **:maxctl=** followed by a numeral indicating the maximum number of levels that IF and DO statements can be nested. |
| :maxequ= | Function | Use this switch with FORTRAN source files to specify the maximum number of equivalences allowed. |
| | Default | :maxequ=150, i.e., a maximum of 150 equivalences is allowed. |
| | Syntax | Type **:maxequ=** followed by a numeral indicating the maximum number of equivalences to be allowed. |

| | | |
|---|---|---|
| :maxext= | Function | Use this switch with FORTRAN source files to specify the maximum number of external symbols allowed. |
| | Default | :maxext=200, i.e., a maximum of 200 external symbols is allowed. |
| | Syntax | Type **:maxext=** followed by a numeral indicating the maximum number of external symbols to be allowed. |
| :maxhash= | Function | Use this switch with FORTRAN source files to specify the size of the FORTRAN compiler's symbol table. |
| | Default | :maxhash=401, i.e., the FORTRAN compiler's symbol table has room for 401 entries. |
| | Syntax | Type **:maxhash=** followed by a numeral indicating the size of the FORTRAN compiler's symbol table. |
| :maxstno= | Function | Use this switch with FORTRAN source files to specify the maximum number of statement numbers allowed. |
| | Default | :maxstno=401, i.e., a maximum of 401 statement numbers is allowed. |
| | Syntax | Type **:maxstno=** followed by a numeral indicating the maximum number of statement numbers to be allowed. |
| :mod | Function | Use this switch to specify that the modification date is to be used in all date and time considerations by the :before= or :since= switches. |
| | Default | :nomod, i.e., the creation date is used in all date and time considerations by the :before= or :since= switches. |
| | Syntax | Type **:mod** |
| :octlist | Function | Use this switch on Pascal source files to generate an octal assembly listing in the current default directory in files with the same name as the corresponding source files, but with .OLS extensions. |
| | Default | :nooctlist, i.e., no octal assembly listing is generated. |
| | Syntax | Type **:octlist** to generate an octal assembly listing. |
| :octlist= | Function | Use this switch on Pascal source files to specify a file in which an octal assembly listing is generated. |

| | | |
|---|---|---|
| | Default | A octal assembly listing is not generated. |
| | Syntax | Type **:octlist=** followed by a file designation. Wildcards are not allowed. |
| :onetrip | Function | Use this switch with FORTRAN source files to specify that DO loops are to be executed at least once. |
| | Default | :noonetrip, i.e., DO loops are not performed if the upper limit is less than the lower limit. |
| | Syntax | Type **:onetrip** to cause DO loops to be executed at least once even if the upper limit is less than the lower limit. |
| :optimize | Function | Use this switch to perform code optimizations where possible. This switch has the same effect as :optimize=F77,OPT and is included for convenience. |
| | Default | :nooptimize, i.e., no optimizations are performed. |
| | Syntax | Type **:optimize** to perform optimizations. |
| :optimize= | Function | Use this switch to perform specific code optimizations. |
| | Default | No optimizations are performed. |
| | Syntax | Type **:optimize=** followed by one or both of the following values (if both, separate with commas): |
| | | F77 For FORTRAN source files only, performs FORTRAN-specific optimizations. |
| | | OPT Performs optimizations on the pseudo-assembly language that is output from the compilers. |
| :output= | Function | Use this switch to name the output file of the compilation process. |
| | Default | If :preprocess, :nofpreprocess, :noassemble, or :noload is specified, the output is stored in the default directory in files with the same names as the corresponding source files specified in the file list, but with extensions of .I, .K, .S, or .W respectively. If the linker/loader phase is not suppressed, the resulting executable file is stored in the default directory with the same name as the first file specified in the file list, but with an extension of .EXE. |

| | | |
|---|---|---|
| | Syntax | Type **:output=** followed by a file name without a file extension. The correct extension (.I, .S, or .EXE, for example) is added automatically by the compiler. Wildcards are not allowed. |
| :overflow | Function | Use this switch with Pascal source files to specify that integer overflow checking is to be performed at runtime. |
| | Default | :nooverflow, i.e., integer overflow checking is not performed. |
| | Syntax | Type **:overflow** to cause integer overflow checking to be performed. |
| :prefix= | Function | Use this switch to specify additional directories in which the linker/loader is to search for libraries. Directories specified by the :prefix= switch are searched first, followed by predefined "standard" directories. |
| | Default | The linker/loader will search only the predefined "standard" directories. |
| | Syntax | Type **:prefix=** followed by a list of directory specifications, separated by commas. As a special case, a value of zero causes the predefined "standard" directories to not be searched. |
| :preprocess | Function | Use this switch to specify that the output from the preprocessors is to be left in the current default directory in files with the same name as the corresponding source files, but with .I extensions. All other phases of the compile are suppressed. |
| | Default | :nopreprocess, i.e., preprocessor output is sent to the compilers, and other phases of the compile are not suppressed. |
| | Syntax | Type **:preprocess** to send the output of the preprocessors to .I files. |
| :process | Function | Use this switch to specify that the output from the preprocessors is to be sent to standard output. All other phases of the compile are suppressed. |
| | Default | :noprocess, i.e., preprocessor output is not sent to standard output, and other phases of the compile are not suppressed. |
| | Syntax | Type **:process** to send the output of the preprocessors to standard output. |

| | | |
|---|---|---|
| :prolog= | Function | Use this switch to specify a custom prolog file to be used for linking. |
| | Default | A standard prolog file is used. |
| | Syntax | Type **:prolog=** followed by the name of the prolog file. If no directory is specified, library directories are searched. If no extension is given, the appropriate extension is appended. |
| :range | Function | Use this switch with FORTRAN and Pascal source files to specify that range-checking of array subscripts and Pascal subranges is to be performed at runtime. |
| | Default | :norange, i.e., runtime range-checking of subscripts is not performed. |
| | Syntax | Type **:range** to cause runtime range-checking of subscripts to be performed. |
| :reloc | Function | Use this switch to specify that relocation information is to be preserved in the executable file. |
| | Default | :noreloc, i.e., relocation information is not preserved. |
| | Syntax | Type **:reloc** to cause relocation information to be preserved in the executable file (useful for unmapped SYSTEM 150's). |
| :runtime= | Function | Use this switch to control which the language-specific runtime libraries the linker/loader uses. |
| | Default | The linker/loader automatically uses language-specific libraries whenever the corresponding language compiler is used. |
| | Syntax | Type **:runtime=** followed by a list of values, separated by commas. Valid values are: |

| Value | Library |
|---|---|
| C | C libraries |
| F | FORTRAN libraries |
| N | No language-specific libraries |
| P | Pascal libraries |

| | | |
|---|---|---|
| :since= | Function | Use this switch to select only those files specified in the file list and were created/ modified since the specified date and time. |
| | Default | Selects all files specified in the file list. |
| | Syntax | Type **:since=** followed by a date and/or time in the standard syntax. |

:startup
- Function: Use this switch to allow the startup parameter file given by the logical name SGS$START to be used.
- Default: :startup, i.e., the startup file will be used.
- Syntax: Type **:nostartup** to prevent the startup file from being used.

:startup=
- Function: Use this switch to specify the startup parameter file to be used.
- Default: The startup file given by the logical name SGS$START is used.
- Syntax: Type **:startup=** followed by the name of the startup file.

:strip
- Function: Use this switch to specify that all symbols are to be stripped from the executable file.
- Default: :strip, i.e., all symbols are stripped.
- Syntax: Type **:nostrip** to preserve symbols in the executable file.

:subpass=
- Function: Use this switch to specify substitute compiler passes. The COMPILE utility uses a three-step algorithm to determine the filename of each compiler pass. First, compiler passes specified by the :subpass= switch are used. Second, for passes not specified by the :subpass= switch, passes specified by logical names are used. Finally, for passes not specified by the :subpass switch or by logical names, the standard compiler passes are used.
- Default: If any of the following logical names are defined, then their definition is used as the filename of the corresponding compiler pass, otherwise the standard compiler passes are used:

| Name | Corresponding Compiler Pass |
|---|---|
| CPP | C preprocessor |
| F77 | FORTRAN compiler, pass 1 |
| F771 | FORTRAN compiler, pass 1 |
| F772 | FORTRAN compiler, pass 2 |
| F77X | FORTRAN cross reference |
| CC | C compiler |
| OPT | C optimizer |
| APP | Floating-point preprocessor |
| 881 | 68881 floating-point preprocessor |

| | |
|---|---|
| FFP | FFP floating-point preprocessor |
| LIB | LIB floating-point preprocessor |
| SKY | SKY floating-point preprocessor |
| NOFP | No floating-point preprocessor |
| PAS1 | Pascal compiler, pass 1 |
| PAS2 | Pascal compiler, pass 2 |
| M2W | Pascal compiler, MRL-to-W converter |
| AS | Assembler |
| LLC | Compiler for linker/loader |
| LL | Linker/loader |

Syntax — Type **:subpass=** followed by one or more values, separated by commas, in the form name:filename, where name is one of the names given in the above table, and filename is the filename (including its directory) of the corresponding substitute compiler pass.

:subopt=

Function — Use this switch to specify arbitrary command line arguments for individual compiler passes. (This switch is provided for maximum flexibility. Take care not to misuse it.)

Default — Only arguments put on the command line by COMPILE (possibly determined by regular COMPILE switches) are passed to the compiler passes.

Syntax — Type **:subopt=** followed by one or more values, separated by commas, where each value is of the form "name:string". The name must be one of the following: 881, cpp, f77, f771, f772, f77x, cc, opt, app, ffp, lib, sky, nofp, as, pas1, pas2, m2w, llc, or ll (see :subpass= for the meaning of each name). The string is the actual string to be placed on the named compiler pass's command line. If the string contains spaces, then the entire name:string value should be enclosed in double quotes.

:tempprefix=

Function — Use this switch to specify the directory in which temporary files are stored.

Default — If the logical name TMPDIR is defined, then the directory it specifies is used, otherwise the directory SYS$TMP/SYSTMP/ is used.

Syntax — Type **:tempprefix=** followed by a directory specification. Wildcards are not allowed.

:target=

Function — Use this switch to specify the CPU on which the program being compiled will run.

| | | |
|---|---|---|
| | Default | On MC68000-based systems, :target=68000. On MC68020-based systems, :target=68020. |
| | Syntax | Type **:target=** followed by either 68000 or 68020. |
| :typeselect= | Function | Use this switch to select only those files that were specified in the file list and are of the specified filetype. |
| | Default | Selects all files that were specified in the file list. |
| | Syntax | Type **:typeselect=** followed by a filetype list or range. |
| :uic= | Function | Use this switch to select only those files or devices that are specified in the file list and which are owned by the specified user or list of users. |
| | Default | Selects all files specified in the file list. |
| | Syntax | Type **:uic=** followed by a list of UIC's or usernames. |
| :undefine= | Function | Use this switch with C source files to cancel macro definitions for the C preprocessor given by the :define= switch. This switch can also be used to cancel the macros predefined by the C preprocessor itself. |
| | Default | No macros are cancelled. |
| | Syntax | Type **:undefine=** followed by a list of names, separated by commas, to be cancelled. |
| :undefined | Function | Use this switch with FORTRAN source files to specify that the default type of variables is undefined, rather than using the default FORTRAN rules. |
| | Default | :noundefined, i.e., the default type of variables is determined according the default FORTRAN rules. |
| | Syntax | Type **:undefined** to cause the default type of FORTRAN variables to be undefined. |
| :verbose | Function | Use this switch to display the command line for each compiler pass before it is executed. This switch also sets the :log switch. Additional information may be displayed as well, depending on the situation. |
| | Default | :noverbose, i.e., command lines for compiler passes are not displayed. |
| | Syntax | Type **:verbose** to display command lines for each compiler pass. |

| | | |
|---|---|---|
| :warnf66 | Function | Use this switch with FORTRAN source files to suppress extensions that enhance compatibility with FORTRAN 66. |
| | Default | :nowarnf66, i.e., extensions that enhance FORTRAN 66 compatibility are not suppressed. |
| | Syntax | Type **:warnf66** to suppress extensions that enhance FORTRAN 66 compatibility. |
| :warnings | Function | Use this switch to display warning messages generated by the compilers. |
| | Default | :warnings, i.e., warning messages are displayed. |
| | Syntax | Type **:nowarnings** to suppress warnings. |
| :xref | Function | Use this switch with FORTRAN source files to generate a symbol cross reference listing in the current default directory in files with the same name as the corresponding source files, but with .REF extensions. |
| | Default | :noxref, i.e., no cross reference is generated. |
| | Syntax | Type **:xref** to generate a cross reference listing. |
| :xref= | Function | Use this switch with FORTRAN source files to specify a file in which a symbol cross reference listing is generated. |
| | Default | A cross reference listing is not generated. |
| | Syntax | Type **:xref=** followed by a filename. Wildcards are not allowed. |

## Examples

> **compile main.c,routine1.c,routine2.c**

This command compiles the C source files named MAIN.C, ROUTINE1.C, and ROUTINE2.C in the default directory, producing the object files MAIN.W, ROUTINE1.W, and ROUTINE2.W, and the executable file MAIN.EXE.

> **compile *.f :prefix=/mylib/ :lib=matrix :float=sky :output=munge.exe**

Assume that the library file /MYLIB/MATRIX.LIB exists on the default device. This command compiles all of the FORTRAN source files with a .F extension in the default directory, producing a .W object file for each one. These object files are then linked together using the library file /MYLIB/MATRIX.LIB to produce an executable file named MUNGE.EXE. The program is targeted for the SKY floating-point board.

## Using Prompts

```
> compile
File list       > main.c,routine1.c,routine2.c
```

This is the same as the first example.

## Notes on Usage

The :optimize and :optimize= switches improve the ultimate efficiency of the program, but compilation usually takes longer.

Any of the five switches that suppress compilation phases may be specified at the same time. The switch that comes first in the following list is used, and any other switches in the list that are specified are ignored:

:process
:preprocess
:nofpreprocess
:noassemble
:noload

At first glance, the :runtime= switch might appear useless since language-specific libraries are automatically included whenever the corresponding language compiler is used. However, suppose you have previously compiled several FORTRAN source files into objects, and you now want to link these object files together to produce an executable file. COMPILE would only see several .W files and wouldn't know about their FORTRAN ancestry. You would have to use the :runtime= switch to explicitly tell COMPILE that these object files were produced from FORTRAN source files and require the use of the FORTRAN libraries.

In order to produce smaller executable files, the symbol table is normally stripped by the linker/loader. When debugging programs with WIBUG, however, it is highly recommended that you use :nostrip to preserve the symbols.

You may now customize COMPILE by using the new startup file feature. A startup file is a text file containing COMPILE command line switches just like a parameter file (see Chapter 8 in the WMCS User's Reference Manual). However, the startup file is designed for customizing, while the parameter file is designed to accommodate long command lines.

There are two ways to specify the name of the startup file. One is to set the logical name SGS$START to the name of the startup file (see Chapter 11 in the WMCS User's Reference Manual). The other is to use the new COMPILE switch :startup=<filename>. If both the logical name is set and the :startup= switch is specified, and they each reference a different file, the file specified by the switch is used. The default is that no startup file is used.

Switches given on the command line override switches in the startup file except for the switches :define=, :include=, :libraries=, :prefix=, :runtime=, :subopt=, :subpass=, and :undefine=. Since these switches accept comma-separated lists, the values from the startup file are appended to the values on the command line. For example, if the command were

```
compile :strip :include=/A/ test.f77
```

and the startup file contained

```
:nostrip
:tempprefix=_dsl/systmp/
:include=/B/
```

the command line would effectively be

```
compile :strip :tempprefix=_dsl/systmp/ :include=/A/,/B/ test.f77
```

---

Related CIP Commands

---

lllib
llran
li