APPENDIX B

MODIFYING COMPILER OPERATION

This chapter tells you how to modify compiler operation by making changes to the standard prototype file. It also explains how to create your own "programmable options" which you can use to modify compiler operation from the **c** compiler driver command line.

The Prototype File

The contents of the standard prototype file **c.pro** appears below.

```
#  WHITESMITHS/COSMIC C ENVIRONMENT PROTOTYPE FILE
#  CXDOS86 HD64180/Z80 -- MS/PC-DOS Cross to HD64180/Z80
#  STANDARD ANSI C ENVIRONMENT PROTOTYPE FILE

c:cpp80     -o (o) -x  \
            {lincl?+lincl}  \
            {proto?-d_PROTO} \
            {std?+std} \
            {listc?-err} \
            {listcs?-err}  \
            -i |/c/hdrs80/  \
            {dl1?+lincl:{dl2?+lincl}} \
            {xdebug?+xdebug} \
            (i)
1:cp180     -o (o) -m {schar?:-u} \
            {std?+std}  \
            {listcs?-err}  \
            {listc? -err > (r).err} \
            {nostrict?-strict} {strict?+strict} \
            {xdebug?+xdebug} \
            {dl1?-dl:{dl2?-dl}} \
            {sprec?-sp} \
            (i)
 {listc?echo \"\#error cp1(V3.32) (r).c\:0\" >> (r).err}
 {listc?lm -o (r).tmp -err -lt (r).err}
 {listc?pr33 -err -t (r).c (r).tmp > (r).lst}
 {listc?del (r).err (r).tmp}
2:cp280     -o (o) -x4 \
            {64180?-h64180} \
           {nobss? -bss} \
           {listcs?+list -err} \
           {dl1?-dl1:{dl2?-dl2}} \
           {rev?-rev} \
           {sp?-sp} \
           {verbose?-v} \
           (i)
3:cp380    -o (o) -e -r30 (i)
 {listcs?lm -o (r).tmp -err -lt -c \";\" (o)}
 {listcs?mv (r).tmp (o)}
s:xz80     -o (o) \
           {listcs?+l >(r).ls} \
           (i)
 {listcs?lm    -o (r).asl -err -lt (r).ls}
 {listcs?pr33  -err (r).asl >(r).ls}
 {listcs?del   (r).asl}
o::lnk80    -o (r).80 {prom?+h}\
           {map?+map=(r).map} \
           +text -b0x1000 \
           +data -b0x8000 \
           /c/lib/Crts{prom?rom}.80 (i) \
           {float?/c/lib/lib{sprec?f:d}.{64180?1:z}80} \
           /c/lib/libi.{64180?1:z}80 \
           /c/lib/libm.{64180?1:z}80 \
           +def __romdata=__text__ +def __memory=__bss__
 {prom?toprom -o (r).prm (r).80}
 {prom?del (r).80}
 {prom?ren (r).prm (r).80}
80:
```

The command line

```
c hello.c lenstr.o
```

in combination with the above prototype file directs the **c** compiler driver to execute the following commands:

```
cpp80 -o ctempc.c0 -x -i |/c/hdrs/ hello.c
cp180 -o ctempc.c1 -m -u ctempc.c0
cp280 -o ctempc.c0 -x4 ctempc.c1
cp380 -o ctempc.c1 -e -r30 ctempc.c0
xz80 -o hello.o ctempc.c1
echo +h -o ctempc.c0 > (r).lnk
echo +text -b0x1000 +data -b0x8000 >>(r).lnk
echo /c/lib/crts.80 hello.o lenstr.o  >> (r).lnk
echo /c/lib/libi.80 /c/lib/libm.80 \
   +def __memory=__bss__ +def __romdata=__text__ >> (r).lnk
 lnk80 < (r).lnk
```

The compiler driver compiles **hello.c** and links it with **lenstr.o**, along with the libraries that the prototype file specifies. The executable program has the default name **xeq.80**. These are the default commands that **c** executes if you do not modify the compilation process by specifying command line options to c or by editing the prototype file.

Changing the Combination of Options Passed to Programs

To change the combination of options that the compiler driver will pass to a program, edit the line of the prototype file that invokes that program. Always add options (either standard or programmable) before the sequence **(i)** that specifies the name of the file that the program will use as input. If you specify an invalid option or combination of options, compilation will not proceed beyond the step where the error occurs.

Changing the Default Location of Libraries

To change the name of the directory that the linker will search for library modules to satisfy references within your program, edit the string specifying the host system directory where the C library routines **(libd, libf, libi,** and **libm)** and the runtime startup module **(crts)** reside. This string appears before the names of the libraries and the runtime startup file on the line of the prototype file that invokes the linker. The line that invokes the linker begins with the label "**o::**".

Changing Input File Name Conventions

To change the suffix of the input file that a pass of the compiler will accept by default, edit the "label" that begins each line of the prototype file that runs a program. The "label" is the part of the line that precedes the first colon ':' character. For example, to change the default file name that the **xz80** assembler accepts as input from <file>.s to <file>.as, edit the first character of the line of the prototype file that invokes **xz80**. Change the string "**s:**" to "**as:**".

Creating Your Own Programmable Options

To create a new programmable option, you must first define the behavior of the new option in terms of existing options or parameters that the appropriate compiler passes or other programs accept. You then write a sequence enclosed in braces to perform the behavior. For example, assume that **xxx** represents a new programmable option. To invoke its behavior from the compiler driver command line, include **-dxxx** in the option list.

From within the prototype file, add a syntactically correct brace-enclosed sequence before the name of the input file on the appropriate line or lines. This sequence can take one of three basic forms:

{xxx?abc} - which becomes the string **abc** when the compiler driver reads the prototype file and **-dxxx** is present on the command line. **abc** may be any string that does not contain an unquoted colon.

{xxx?:def} - which directs **c** to ignore **def** when it reads the prototype file if **-dxxx** is present on the command line.

{xxx?abc:def} - which expands **abc** if the option is present on the command line, and **def** if it is not.

It is also possible to nest programmable option expansion sequences. You can use a nested sequence to obtain logical OR and logical AND constructs when testing for combinations of programmable options. For example, the sequence **{a?{b?AB:A-}:{b?-B:--}}** expands to:

AB	if both **-da** and **-db** appear on the command line
A-	if **-da** appears but not **-db**
-B	if **-db** appears but not **-da**
--	if neither programmable option appears

APPENDIX B

MODIFYING COMPILER OPERATION

This chapter tells you how to modify compiler operation by making changes to the standard prototype file. It also explains how to create your own "programmable options" which you can use to modify compiler operation from the c compiler driver command line.

The Prototype File

The contents of the standard prototype file **c.pro** appears below.

```
#  COSMIC C ENVIRONMENT PROTOTYPE FILE
#  CXIDR86 HD64180/Z80 -- XENIX Cross to HD64180/Z80

c:/usr/bin/cpp80   -o (o) -x {xdebug?+xdebug} {lincl?+lincl}  \
            {std?+std} {proto?-d_PROTO} \
            {listcs?-err} {listc?-err} \
            -i "|/usr/include/wsl/" \
            {dl1?+lincl:{dl2?+lincl}} \
            (i)

1:/usr/bin/cp180   -o (o) -m {schar?:-u} {std?+std}  \
            {nostrict?-strict} {strict?+strict} \
            {dl1?-dl:{dl2?-dl}} \
            {xdebug?+xdebug} {sprec?-sp} \
            {listc?-err} {listcs?-err}  \
            (i)
  {listc?echo \"\#error cp1(V3.32) (r).c\:0\" >>(r).err}
  {listc?lm -o (r).tmp -err -lt (r).err}
  {listc?pr -err -t (r).c (r).tmp >(r).lst}
  {listc?rm (r).err (r).tmp}

2:/usr/bin/cp280   -o (o) -x4 {64180?-h64180} \
            {nobss?-bss} {listcs?+list -err} \
            {dl1?-dl1:{dl2?-dl2}} \
            {verbose?-v} \
            {sp?-sp} {rev?-rev} \
            (i)

3:/usr/bin/cp380   -o (o) -e -r30 (i)
```

```
 {listcs?lm    -o (r).tmp -err -lt -c \";\" (o)}
 {listcs?mv    (r).tmp (o)}

s:/usr/bin/x80 -o (o) {listcs?+l >(r).ls} (i)
 {listcs?lm    -o (r).asl -err -lt (r).ls}
 {listcs?pr.33 -err (r).asl >(r).ls}
 {listcs?/bin/rm (r).asl}

o::/usr/bin/lnk80 -o (o) {prom?+h} \
       {map?+map=(r).map} \
       +text -b0x1000 \
       +data -b0x8000 \
       /lib/Crts{prom?rom}.80 (i) \
       {float?/lib/lib{sprec?f:d}.{64180?1:z}.80} \
       /lib/libi.{64180?1:z}80 \
       /lib/libm.{64180?1:z}80 \
       +def __romdata=__text__ +def __memory=__bss__
 {prom?toprom -o (r).prm (o);/bin/mv (r).prm (o)}
80:
```

The command line

```
c hello.c lenstr.o
```

in combination with the above prototype file directs the **c** compiler driver to execute the following commands:

```
/usr/bin/cpp80 -o /tmp/t0c150 -x -i |/usr/include/wsl/ hello.c
/usr/bin/cp180 -o /tmp/t0c151 -m -u /tmp/t0c150
/usr/bin/cp280 -o -x4 /tmp/t0c150 /tmp/t0c151
/usr/bin/cp380 -o -e -r30 /tmp/t0c151 /tmp/t0c150
/usr/bin/x80 -o hello.o /tmp/t0c151
hello:
   /usr/bin/lnk80 -o hello.80 +text -b0x1000 +data -b0x8000 \
   /lib/crts.80 hello.o lenstr.o /lib/libi.z80 /lib/libm.z80 \
   +def __romdata=__text__ +def __memory=__bss__
```

The compiler driver compiles **hello.c** and links it with **lenstr.o**, along with the libraries that the prototype file specifies. The executable program has the default name **xeq.80**. These are the default commands that **c** executes if you do not modify the compilation process by specifying command line options to **c** or by editing the prototype file.

Changing the Combination of Options Passed to Programs

To change the combination of options that the compiler driver will pass to a program, edit the line of the prototype file that invokes that program. Always add options (either standard or programmable) before the sequence **(i)** that specifies the name of the file that the program will use as input. If you specify an invalid option or combination of options, compilation will not proceed beyond the step where the error occurs.

Changing the Default Location of Libraries

To change the name of the directory that the linker will search for library modules to satisfy references within your program, edit the string specifying the host system directory where the C library routines (**libd, libf, libi,** and **libm)** and the runtime startup module (**crts)** reside. This string appears before the names of the libraries and the runtime startup file on the line of the prototype file that invokes the linker. The line that invokes the linker begins with the label "**o::**".

Changing Input File Name Conventions

To change the suffix of the input file that a pass of the compiler will accept by default, edit the "label" that begins each line of the prototype file that runs a program. The "label" is the part of the line that precedes the first colon ':' character. For example, to change the default file name that the **x80** assembler accepts as input from <ins>&lt;file&gt;.s</ins> to <ins>&lt;file&gt;.as,</ins> edit the first character of the line of the prototype file that invokes **x80.** Change the string "**s:**" to "**as:**".

Creating Your Own Programmable Options

To create a new programmable option, you must first define the behavior of the new option in terms of existing options or parameters that the appropriate compiler passes or other programs accept. You then write a sequence enclosed in braces to perform the behavior. For example, assume that **xxx** represents a new programmable option. To invoke its behavior from the compiler driver command line, include **-dxxx** in the option list.

From within the prototype file, add a syntactically correct brace-enclosed sequence before the name of the input file on the appropriate line or lines. This sequence can take one of three basic forms:

{xxx?abc} - which becomes the string **abc** when the compiler driver reads the prototype file and **-dxxx** is present on the command line. **abc** may be any string that does not contain an unquoted colon.

{xxx?:def} - which directs **c** to ignore **def** when it reads the prototype file if **-dxxx** is present on the command line.

{xxx?abc:def} - which expands **abc** if the option is present on the command line, and **def** if it is not.

It is also possible to nest programmable option expansion sequences. You can use a nested sequence to obtain logical OR and logical AND constructs when testing for combinations of programmable options. For example, the sequence **{a?{b?AB:A-}:{b?-B:--}}** expands to:

AB	if both **-da** and **-db** appear on the command line
A-	if **-da** appears but not **-db**
-B	if **-db** appears but not **-da**
--	if neither programmable option appears