Part Four

Experts–PL/M
Installation Guide

Table of Contents

1. Introduction

This portion of the Experts–PL/M™ manual describes the method of installing the 80/DS development system on various common machines and operating systems. These include:

- The VAX under VMS
- The Tektronix 8560 under TNIX
- Most other UNIX systems
- The IBM PC under PC-DOS

A separate chapter is devoted to each.

All of the installation discusions in this part are intended to be read and followed by experienced users who are familiar with the general problems of software installation on the particular machine.

2. Installing on VAX/VMS

This chapter discusses the method of installing an 80/DS delivery tape on a VAX under the VMS operating system. The discussion assumes that the delivery is made on a 9-track magnetic tape. If some other medium is used, the same general method should apply. Any special instructions will be found on an installation memorandum packed with the delivery.

This discussion assumes that the installation is being performed by an experienced VMS systems programmer.

2.1 SUPPORTED OPERATING ENVIRONMENT

This version of 80/DS requires a Digital Equipment Corporation VAX or Micro-VAX and the VMS operating system, release 3.6 or later.

2.2 RESTORING THE TAPE

The delivery tape is in standard VMS *backup* format. It may be restored by

```
$ alloc mta tape
$ mount/foreign/den=1600 tape
$ backup/rew/log tape:80ds.bkp [.80ds...]
```

This creates a subdirectory of the current directory called *80ds* which, itself, contains three subdirectories: *80pc*, *80as*, *80rl*.

2.3 DEFINING LOGICAL NAMES

Logical names must be defined to locate each of the major components of 80/DS. Suppose that restoring the tape (Section 2.2) created the *80ds* directory as a subdirectory of *sys$disk:[tools]*. The following definitions should be made:

```
$ define sys$80as sys$disk:[tools.80ds.80as]
$ define sys$80pc sys$disk:[tools.80ds.80pc]
$ define sys$80rl sys$disk:[tools.80ds.80rl]
```

If only a few people will be using 80/DS, these definitions may be placed in those person's LOGIN.COM files. If many people will be using 80/DS, the definitions should be made

system-wide.

2.4 INSTALLING THE NATIVE COMMANDS

The 80/PC, 80/AS, 80/LINK, 80/LOC, and 80/MAP are all intended to be installed as VMS native commands and executed using standard DCL syntax. Each of these commands has an associated CLD file which defines the invocation syntax to the DCL processor and a HELP file which can provide on-line help.

These commands are installed by the *SET COMMAND* DCL command. To install 80/AS, use

```
$ set command sys$80as:80as
```

Help on the use of 80/AS may be obtained by

```
$ help/lib=sys$80as:80as 80as
```

To install 80/PC, use

```
$ set command sys$80pc:80pc
```

Help on the use of 80/PC may be obtained by

```
$ help/lib=sys$80pc:80pc 80pc
```

To install the native 80/RL commands, use

```
$ set command sys$80rl:80link
$ set command sys$80rl:80loc
$ set command sys$80rl:80map
```

Help on the use of 80/MAP, for example, may be obtained by

```
$ help/lib=sys$80rl:80rl 80map
```

If only a few people will be using 80/DS, these definitions may be placed in those person's LOGIN.COM files. If many people will be using 80/DS, the definitions should be made system-wide by modifying the system DCL command tables.

2.5 INSTALLING THE FOREIGN COMMANDS

The remaining 80/RL tools are installed as foreign commands and are invoked using the UNIX-style syntax described in Part Three of this manual. The foreign command definitions are

```
$ crobj80 :== $sys$80rl:80crobj
$ dsobj80 :== $sys$80rl:80dsobj
$ hex80 :== $sys$80rl:80hex
$ thex80 :== $sys$80rl:80thex
$ libcr80 :== $sys$80rl:80libcr
$ libls80 :== $sys$80rl:80libls
$ strip80 :== $sys$80rl:80strip
```

If only a few people will be using 80/DS, these definitions may be placed in those person's LOGIN.COM files. If many people will be using 80/DS, the definitions should be made system-wide.

2.5.1 Use of Command Files

Most of these 80/RL programs take their input from SYS$INPUT and place their output on SYS$OUTPUT. These may be redirected by appropriate redefinition.

Note that when SYS$OUTPUT is redirected to a file, the file is opened for update and positioned to the beginning. Thus, if a program's output is redirected to an existing file that is longer than the output, there will be garbage at the end. Therefore, a command file like the following (for 80/DSOBJ) may be useful:

```
$ delete 'p1'.dso;*
$ define/user sys$output 'p1'.dso
$ dsobj80 'p1'.q80
```

3. Installing on Tektronix 856x

This chapter discusses the method of installing an 80/DS delivery diskette on a Tektronix 856x under the TNIX operating system.

This discussion assumes that the installation is being performed by an experienced TNIX user or systems programmer.

3.1 SUPPORTED OPERATING ENVIRONMENT

This version of 80/DS requires a Tektronix 8560, 8561, or 8562 computer and the TNIX operating system, Version 2 or later.

3.2 RESTORING THE DISKETTE

The delivery diskette is in standard TNIX *fbr* format. To install it, log in as *root* and enter

```
install
```

If questions appear on the screen during installation, answer them appropriately.

The result of the installation will be to place the drivers for 80/AS and 80/PC, as well as all the 80/RL tools, into the */usr/bin* directory. The individual phases of 80/AS and 80/PC will be placed in */usr/lib/80ds*.

3.3 SELECTING 80/PC FINAL OUTPUT DEFAULT

As delivered, the 80/PC compiler will produce object modules in Intel standard object module format by default. To change this default so that Tektronix LAS is produced instead, do

```
cp /usr/lib/80ds/80ptfo /usr/lib/80ds/80pfo
```

To restore Intel object as the default, do

```
cp /usr/lib/80ds/80pifo /usr/lib/80ds/80pfo
```

3.4 TAILORING WITH ENVIRONMENT VARIABLES

The 80/DS Development System can be extensively tailored by using environment variables. So as not to clutter the environment, this method is best when only a few simple changes are desired. The methods described in Section 3.5 are prefered when extensive changes are to be made.

3.4.1 Global Tailoring Changes

Many parts of 80/DS use temporary files (by default located in /usr/tmp) and many need to find the location of the directory (by default */usr/lib/80ds*) where various phases reside. The following environment variables may be used to change these locations:

A80DSTMP If this variable is defined, its contents will be used as a prefix for all temporary file names. For example, if it contains "/tmp/", all temporary files will be created in the /tmp directory.

A80DSLIB If this variable is defined, its contents will be used as a prefix for all phase names. For example, if it contains "/u3/tools/dev/", 80/PC will look in the */u3/tools/dev* directory for each of the compiler phases.

3.4.2 Local Tailoring Changes

Environment variables may be used to provide arguments to specific parts of 80/DS. The contents of the variable "*AnameHEAD*" will be processed as arguments to the "*name*" program before the arguments on the invocation line are processed. After processing the invocation line arguments, the contents of the variable "*AnameTAIL*" are processed as arguments.

For example, the definition

```
A80PCHEAD="-p58 -Xs.plm -Xo.obj -Xl.lis"
export A80PCHEAD
```

will cause all 80/PC compilations to use a listing page depth of 58 lines, a source suffix of ".plm", an object suffix of ".obj", and a listing suffix of ".lis".

The definition

```
A80LINKTAIL=/usr/lib/80ds/pl.lib
export A80LINKTAIL
```

will cause the file "/usr/lib/80ds/pl.lib" to be linked in after all other files. Presumably, this is a library to resolve remaining external references.

3.4.3 Specifying Maximum Number of Arguments

Each 80/DS component has been configured to process a reasonable number of arguments. Sometimes, however, this predetermined maximum may not be enough. If the variable "*AnameMAXA*" is defined, its contents are taken as a decimal integer giving the maximum number of arguments to allow for the "*name*" program. If the variable "*A80DSMAXA*" is defined, its contents are taken as a decimal integer giving the maximum number of arguments to allow for all 80/DS programs.

Note that these variables cannot be used to increase the maximum beyond any limits which may be imposed by the operating system.

3.5 TAILORING WITH AN INITIALIZATION FILE

Very detailed tailoring for each 80/DS component may be performed by use of a file named *80ds.ini*. When any 80/DS command is executed, this file is searched for in the following places (in order): the current directory and in */usr/lib/80ds*. If there is an *A80DSINI* environment variable, its contents are used as the complete path to the file instead.

If it is found, it is opened and read, searching for lines which begin with the name of the command (e.g. "80pc") followed by a single colon. The remainder of each such line is processed just as if it were a set of invocation options appearing before the options on the invocation line. When the last such line is processed, the actual invocation options are processed. Finally, the file is searched for lines beginning with the command name followed by *two* colons and these lines are processed as invocation options.

Lines are processed in order of appearance, and any *double-colon* lines for a given command must follow all *single-colon* lines for that command. The special command name "*" matches any command.

3.5.1 Examples

The *80ds.ini* line

```
80pc: -p58 -Xs.plm -Xo.obj -Xl.lis
```

will establish a page depth of 58 lines, source suffix of ".plm", object suffix of ".obj", and listing suffix of ".lis" for all 80/PC invocations.

The *80ds.ini* line

```
80link:: /usr/lib/80ds/pl.lib
```

will cause the file "/usr/lib/80ds/pl.lib" to be linked in after all other source files. Presumably, this file is a library to resolve remaining external references.

The temporary directory will be changed to "/scratch" by the line

```
*: -Xt/scratch/
```

4. Installing on UNIX Systems

This chapter discusses installing the 80/DS Development System on various UNIX systems. Only binary installations are discussed in this chapter. If you are installing a UNIX source version of 80/DS, see Appendix A.

This discussion assumes that the installation is being performed by an experienced UNIX user or systems programmer.

4.1 SUPPORTED OPERATING ENVIRONMENT

Generally, the UNIX version of 80/DS requires a 16-bit or 32-bit, byte-addressing machine which runs a true Version 7, System III, or System V UNIX or XENIX operating system. Only the most common of these are supported in binary form.

4.2 BINARY INSTALLATION

The delivery medium for a binary installation will normally consist of a 9-track magnetic tape. In some cases, however, it may be one or more diskettes or other special types of recording media.

In general, installation is performed by logging in as root, mounting the delivery tape, and entering

```
cd /
tar xv
```

If other methods are required, a memorandum describing them will accompany the delivery.

The result of the installation will be to place the drivers for 80/AS and 80/PC, as well as all the 80/RL tools, into the */usr/bin* directory. The individual phases of 80/AS and 80/PC will be placed in */usr/lib/80ds*.

4.3 SELECTING 80/PC FINAL OUTPUT DEFAULT

As delivered, the 80/PC compiler will produce object modules in Intel standard object module format by default. In some versions, this may be changed to produce Tektronix LAS instead by

```
cp /usr/lib/80ds/80ptfo /usr/lib/80ds/80pfo
```

To restore Intel object as the default, do

```
cp /usr/lib/80ds/80pifo /usr/lib/80ds/80pfo
```

4.4 TAILORING WITH ENVIRONMENT VARIABLES

The 80/DS Development System can be extensively tailored by using environment variables. So as not to clutter the environment, this method is best when only a few simple changes are desired. The methods described in Section 4.5 are prefered when extensive changes are to be made.

4.4.1 Global Tailoring Changes

Many parts of 80/DS use temporary files (by default located in /usr/tmp) and many need to find the location of the directory (by default */usr/lib/80ds*) where various phases reside. The following environment variables may be used to change these locations:

A80DSTMP If this variable is defined, its contents will be used as a prefix for all temporary file names. For example, if it contains "/tmp/", all temporary files will be created in the */tmp* directory.

A80DSLIB If this variable is defined, its contents will be used as a prefix for all phase names. For example, if it contains "/u3/tools/dev/", 80/PC will look in the */u3/tools/dev* directory for each of the compiler phases.

4.4.2 Local Tailoring Changes

Environment variables may be used to provide arguments to specific parts of 80/DS. The contents of the variable "*AnameHEAD*" will be processed as arguments to the "*name*" program before the arguments on the invocation line are processed. After processing the invocation line arguments, the contents of the variable "*AnameTAIL*" are processed as arguments.

For example, the definition

```
A80PCHEAD="-p58 -Xs.plm -Xo.obj -Xl.lis"
export A80PCHEAD
```

will cause all 80/PC compilations to use a listing page depth of 58 lines, a source suffix of ".plm", an object suffix of ".obj", and a listing suffix of ".lis".

The definition

```
A80LINKTAIL=/usr/lib/80ds/pl.lib
export A80LINKTAIL
```

will cause the file "/usr/lib/80ds/pl.lib" to be linked in after all other files. Presumably, this is a library to resolve remaining external references.

4.4.3 Specifying Maximum Number of Arguments

Each 80/DS component has been configured to process a reasonable number of arguments. Sometimes, however, this predetermined maximum may not be enough. If the variable *"AnameMAXA"* is defined, its contents are taken as a decimal integer giving the maximum number of arguments to allow for the *"name"* program. If the variable *"A80DSMAXA"* is defined, its contents are taken as a decimal integer giving the maximum number of arguments to allow for all 80/DS programs.

Note that these variables cannot be used to increase the maximum beyond any limits which may be imposed by the operating system.

4.5 TAILORING WITH AN INITIALIZATION FILE

Very detailed tailoring for each 80/DS component may be performed by use of a file named *80ds.ini*. When any 80/DS command is executed, this file is searched for in the following places (in order): the current directory and in */usr/lib/80ds*. If there is an *A80DSINI* environment variable, its contents are used as the complete path to the file instead.

If it is found, it is opened and read, searching for lines which begin with the name of the command (e.g. "80pc") followed by a single colon. The remainder of each such line is processed just as if it were a set of invocation options appearing before the options on the invocation line. When the last such line is processed, the actual invocation options are processed. Finally, the file is searched for lines beginning with the command name followed by *two* colons and these lines are processed as invocation options.

Lines are processed in order of appearance, and any *double-colon* lines for a given command must follow all *single-colon* lines for that command. The special command name "*" matches any command.

4.5.1 Examples

The *80ds.ini* line

```
80pc: -p58 -Xs.plm -Xo.obj -Xl.lis
```

will establish a page depth of 58 lines, source suffix of ".plm", object suffix of ".obj", and listing suffix of ".lis" for all 80/PC invocations.

The *80ds.ini* line

```
80link:: /usr/lib/80ds/pl.lib
```

will cause the file "/usr/lib/80ds/pl.lib" to be linked in after all other source files. Presumably, this file is a library to resolve remaining external references.

The *80ds.ini* line

```
*: -Xt/tmp/
```

would cause the "/tmp" directory to be used for all temporary files.

5. Installing on PC-DOS

This chapter discusses the method of installing the 80/DS Development System on an IBM PC computer under the PC-DOS operating system.

This discussion assumes that the installation is being performed by someone who is familiar with installing system software under DOS.

5.1 SUPPORTED OPERATING ENVIRONMENT

This version of 80/DS requires an IBM PC-XT or PC-AT or equivalent computer and the PC-DOS or MS-DOS operating system, Version 2.1 or later.

5.2 RESTORING THE DISKETTE

The 80/DS delivery consists of one or more diskettes in standard DOS copy format. By default, the various components reside in the "\80ds.lib" directory which must be created by

```
mkdir \80ds.lib
```

To restore the diskettes, first set the current directory to be 80ds.lib by

```
cd \80ds.lib
```

and then restore the diskettes by performing

```
copy a:*.* .
```

for each.

5.3 MAKING THE TEMPORARY DIRECTORY

Many of the 80/DS components need to create temporary files while operating. These are created in the directory "\tmp". If this directory does not exist, it should be created by

```
mkdir \tmp
```

5.4 INSTALLING 80/DS IN THE SEARCH PATH

The 80/DS commands (e.g., 80pc, 80dsobj) should be in a directory which is in your search path. The simplest way to do this is just to place the "80ds.lib" directory in the search path. If your AUTOEXEC.BAT file does not have a *path* statement, add

```
path c:\80ds.lib
```

to the file. If there already is a *path* statement, add ";c:\80ds.lib" to the end of the statement. It might, for example, then look like

```
path c:\bin;c:\80ds.lib
```

A slightly more complex, but more efficient, method is to place only the command names in a directory that is in the search path. Many systems have, for example, a "\bin" directory to contain commands and this is a good place to move the 80/DS commands. In particular, these are the files

80as.exe	80libcr.exe	80map.exe
80crobj.exe	80libls.exe	80pc.exe
80dsobj.exe	80link.exe	80strip.exe
80hex.exe	80loc.exe	

They may be moved to the "\bin" directory, leaving only the subphases in the "\80ds.lib" directory.

5.5 SELECTING 80/PC FINAL OUTPUT DEFAULT

As delivered, the 80/PC compiler will produce object modules in Intel standard object module format by default. With some versions, Tektronix LAS may be produced instead, by

```
copy \80ds.lib\80ptfo.exe \80ds.lib\80pfo.exe
```

To restore Intel object as the default, do

```
copy \80ds.lib\80pifo.exe \80ds.lib\80pfo.exe
```

5.6 TAILORING WITH ENVIRONMENT VARIABLES

The 80/DS Development System can be extensively tailored by using environment variables. So as not to clutter the environment, this method is best when only a few simple changes are desired. The methods described in Section 5.7 are prefered when extensive changes are to be made.

5.6.1 Global Tailoring Changes

Many parts of 80/DS use temporary files (by default located in \tmp) and many need to find the location of the directory (by default *\80ds.lib*) where various phases reside. The following environment variables may be used to change these locations:

A80DSTMP If this variable is defined, its contents will be used as a prefix for all temporary file names. For example, if it contains "d:\tmp\", all temporary files will be created in the \tmp directory on device "d:".

A80DSLIB If this variable is defined, its contents will be used as a prefix for all phase names. For example, if it contains "d:\tools\", 80/PC will look in the \ *tools*\ directory on device "d:" for each of the compiler phases.

5.6.2 Local Tailoring Changes

Environment variables may be used to provide arguments to specific parts of 80/DS. The contents of the variable "*AnameHEAD*" will be processed as arguments to the "*name*" program before the arguments on the invocation line are processed. After processing the invocation line arguments, the contents of the variable "*AnameTAIL*" are processed as arguments.

For example, the definition

```
set A80PCHEAD=-p58 -Xs.plm -Xo.obj -Xl.lis
```

will cause all 80/PC compilations to use a listing page depth of 58 lines, a source suffix of ".plm", an object suffix of ".obj", and a listing suffix of ".lis".

The definition

```
A80LINKTAIL=c:\80ds.lib\pl.lib
```

will cause the file "c:\80ds.lib\pl.lib" to be linked in after all other files. Presumably, this is a library to resolve remaining external references.

5.6.3 Specifying Maximum Number of Arguments

Each 80/DS component has been configured to process a reasonable number of arguments. Sometimes, however, this predetermined maximum may not be enough. If the variable "*AnameMAXA*" is defined, its contents are taken as a decimal integer giving the maximum number of arguments to allow for the "*name*" program. If the variable "*A80DSMAXA*" is defined, its contents are taken as a decimal integer giving the maximum number of arguments to allow for all 80/DS programs.

Note that these variables cannot be used to increase the maximum beyond any limits which may be imposed by the operating system.

5.7 TAILORING WITH AN INITIALIZATION FILE

Very detailed tailoring for each 80/DS component may be performed by use of a file named *80ds.ini*. When any 80/DS command is executed, this file is searched for in the following places (in order): the current directory, *\80ds.lib*, *c:\80ds.lib*, **, and *c:*. If there is an *A80DSINI* environment variable, its contents are used as the complete path to the file instead.

If it is found, it is opened and read, searching for lines which begin with the name of the command (e.g. "80pc") followed by a single colon. The remainder of each such line is processed just as if it were a set of invocation options appearing before the options on the invocation line. When the last such line is processed, the actual invocation options are processed. Finally, the file is searched for lines beginning with the command name followed by *two* colons and these lines are processed as invocation options.

Lines are processed in order of appearance, and any *double-colon* lines for a given command must follow all *single-colon* lines for that command. The special command name "*" matches any command.

5.7.1 Examples

The *80ds.ini* line

```
80pc: -p58 -Xs.plm -Xo.obj -Xl.lis
```

will establish a page depth of 58 lines, source suffix of ".plm", object suffix of ".obj", and listing suffix of ".lis" for all 80/PC invocations.

The *80ds.ini* line

```
80link:: c:\80ds.lib\pl.lib
```

will cause the file "c:\80ds.lib\pl.lib" to be linked in after all other source files. Presumably, this file is a library to resolve remaining external references.

The temporary directory will be changed to "d:\tmp\" by the line

```
*: -Xtd:\tmp\
```

A. Source Installation on UNIX Systems

This appendix discusses installation of a source version of 80/DS on a UNIX system. This is a complicated process and should only be performed by an experienced UNIX systems programmer.

This chapter only discusses the actual building and installing of the 80/DS components. Chapter 4 should be read for a discussion of various post-installation tailoring options.

A.1 SUPPORTED OPERATING ENVIRONMENT

Generally, the UNIX version of 80/DS requires a 16-bit or 32-bit, byte-addressing machine which runs a true Version 7, System III, or System V UNIX or XENIX operating system. Because of the lack of compatibility among the various systems, installing of a source version of 80/DS may require changes to the programs in order to make them operate correctly. Such changes are the responsibility of the installer.

A.2 RESTORING THE DELIVERY TAPES

The 80/DS package is distributed in a manner intended to be very portable. It does not depend upon the availability of any particular tape archiving program. This chapter describes the format of the delivery tapes and discusses the steps required for restoring such tapes.

A.2.1 Tape Format

Each piece of 80/DS software is distributed on one reel of 9-track, standard magnetic tape containing several files:

1. A small C source program, named "sarin", which may be used to extract the source files from the archives contained in the other tape files;
2. One or more archive files containing the source.

A.2.2 Restoring the Tape

A distribution in the form of an industry-standard magnetic tape will have the format:

```
source for sarin utility
<EOF>
source archive #1
<EOF>
source archive #2
<EOF>
.
.
.
<EOF>
<EOF>
```

All data blocks on the tape are 512 bytes long. A tape will have one or more source archives.

The files may be restored by a shell sequence such as:

```
(dd of=sarin.c;
dd of=sarc1;
dd of=sarc2;
.
.
.
) </dev/rmt0
```

where */dev/rmt0* is the name of the raw interface for the tape drive on which the delivery tape is mounted.

A.2.3 Compiling the Sarin Utility

The sarin utility may be used to extract the source files from the archive. It should be compiled by

```
cc sarin.c -o sarin
```

which will leave an executable version of the utility in the file "sarin".

Because all tape blocks are 512 bytes long, it is possible that the last block of the sarin.c file will contain one or more trailing ASCII nulls. This should not cause any problems. However, if the C compiler complains about illegal characters which appear to be at the end of the file, the shell sequence

```
ed sarin.c
w
q
```

should remove the trailing nulls.

A.2.4 Extracting Source Files From the Archives

Once the sarin utility has been compiled, it may be used to process the archives.

A.2.4.1 Structure of the Source Archives

The archives consist of a sequence of lines, each terminated by a newline. Lines whose first three characters are "$" are *control lines*. The possible control lines are:

```
{$}C commentary
```

which causes the commentary to be displayed when the archive is processed;

```
{$}D directory-name
```

which causes the named directory to be created by "mkdir";

```
{$}F file-name
```

which causes the named file to be created and opened;

```
{$}E
```

which causes the file named in the last "$F" control line to be closed; and

```
{$}Z
```

which terminates the archive. Any line which is not a control line is source; it is written to the file named in the last "$F" control line.

A.2.4.2 Processing the Source Archives

The commands

```
sarin -t <sarc1
sarin -t <sarc2
.
.
.
```

will display a listing of the names of all archived files and directories. The listing will be written on the standard error file.

Actual extraction of the source files may be performed by the commands

```
sarin <sarc1
sarin <sarc2
.
.
.
```

These will create the necessary directories and subdirectories and restore all source files. A progress log will also be displayed on the standard error file.

During extraction, warning messages will be issued if any of the directories or subdirectories already exist, but extraction will proceed using the existing directories.

A.3 INSTALLING THE 80/PC COMPILER

This chapter discusses the procedures to produce a working 80/PC from the delivery tape.

A.3.1 Restoring the 80/PC Delivery Tape

The 80/PC delivery tape has two archive files. The restoration process is described in Section A.2.

Restoring the archives will create the following directory structure:

```
80pc.d                  (root directory for 80pc)
    common.d:           (directory for common C source)
    lib.d:              (directory for 80pl.lib source)
    p1.d:               (directory for p1 source)
    p2.d:               (directory for p2 source)
    pc.d:               (directory for driver source)
    pcg.d:              (directory for pcg source)
    pfo.d:              (directory for pfo source)
    pjo.d:              (directory for pjo source)
    pp.d:               (directory for pp source)
    psym.d              (directory for psym source)
    ptfo.d              (directory for ptfo source)
    pxrf.d              (directory for pxrf source)
```

A.3.2 Modifying the 80/PC Shell Scripts

All examples in this chapter assume that you have already restored the delivery tape and extracted the source files. It is also assumed that your working directory is "80pc.d". The following shell scripts are provided:

pcdefs.sh	which defines things for the other shell scripts
pccompile.sh	which compiles everything
pclink.sh	which links everything
pcmklib.sh	which makes the library
pcinstall.sh	which installs everything
pcprint.sh	which prints everything

Before proceeding with the generation and installation, examine these shell scripts and

make any required changes as indicated by their comments and the following descriptions.

A.3.2.1 Modifying pcdefs.sh

This shell script sets up definitions of shell variables used in other shell scripts.

A.3.2.1.1 Installation Directory for the Driver

The shell variable *I* is set to the directory where the 80/PC driver is installed. As distributed, the shell variable is defined as

```
I=/usr/bin
```

A.3.2.1.2 Installation Directory for the Compiler Phases

The shell variable *P* is set to the directory where the 80/PC driver expects to find its phases. The library will also be installed there. As distributed, the shell variable is defined as

```
P=/usr/lib/80pc
```

This corresponds to the default definition of PDIR in some of the C source code. If the definition of *P* is changed, then PDIR must also change. This is done through the C compilation flags.

A.3.2.1.3 C Compilation Flags

The shell variable C is set to the flags to be used for C compilations. As distributed, the shell variable is defined as

```
C='-I../common.d'
```

This sets the search path for include files.

A.3.2.1.3.1 Temporary File Directory

TDIR is defined in some source modules to be "/usr/tmp/". This is the name of the directory in which 80/PC should create temporary files. Some UNIX systems, notably PWB, do not normally have the "/usr/tmp" directory. For such systems, TDIR should be redefined – usually to "/tmp/". This is done by adding

```
-DTDIR="/tmp/"
```

to the definition of C.

A.3.2.1.3.2 Phase Installation Directory

PDIR is defined in some source modules to be "/usr/lib/80pc/". This is the name of the directory in which the 80/PC driver expects to find its phases. If you wish to change this for some reason, PDIR should be redefined. This is done by adding

```
-DPDIR="/new/dir"
```

to the definition of *C*. Note that the shell variable *P* must also be redefined.

A.3.2.1.3.3 Treatment of Warning Messages

When compiling with some C compilers, a number of warning messages will be issued. These messages relate to different interpretations of the proper way to use the C language in certain circumstances; they can be ignored. These warnings can be suppressed by adding *-w* to the definition of *C*.

A.3.2.1.3.4 C Compiler Optimization

Most C compilers can provide processing to attempt to produce more optimal code; this may be specified by adding -O to the definition of *C*. It is possible that this will produce a smaller, faster 80/PC. Considering the problems frequently encountered with the use of such optimizers, we do not recommend this unless you have first successfully installed and tested 80/PC without optimization.

A.3.2.1.3.5 Other C Compiler Options

Any other desired C compiler options of local interest (usually none) may be specified by adding to the definition of *C*.

A.3.2.1.4 Specifying Linker Options

The shell variable *L* is set to the flags to be used by the linker. As distributed, the shell variable is defined as

```
L='-i'
```

A.3.2.1.4.1 Split I/D Linking

When the *-i* option is used, the linker produces an executable image (text file) using separate code and data spaces. This is advisable for best performance of 80/PC except on the VAX under Berkeley UNIX. Berkeley UNIX uses a demand paging mechanism and the split i/d concept is not relevant to that environment. When generating 80/PC under Berkeley UNIX, remove the "-i" from the definition of *L*.

A.3.2.1.4.2 Other Linker Options

Any other desired linker options of local interest (usually none) may be specified by adding to the definition of *L*.

A.3.2.2 Modifying pccompile.sh

The local name of the C compiler should replace "cc" if necessary.

A.3.2.3 Modifying pclink.sh

The local name of the C compiler should replace "cc" if necessary.

A.3.2.4 Modifying pcmklib.sh

If you have the 80/RL tools, this script may be used to create *80pl.lib*, the 80/PC runtime library. It should require no changes.

If you do not have the 80/RL tools, it is assumed that you have an assembler for the Intel 8080/8085 assembly language and something that will create a library in a format compatible with your linker. It will be necessary to create your own shell script. To create the library, first assemble all files lib.d/*.asm, then run your library creation program using the resulting object files. The library produced may be named "80pl.lib" or anything else convenient.

A.3.2.5 Modifying pcinstall.sh

This shell script, among other things, installs the 80/PC manual pages (provided as nroff source). This should be checked for compatibility with your installation's standards. The 80/PC runtime library is also installed by this shell script. If you are not using the 80/RL tools, then this should be modified to suit your program linkage environment.

A.3.2.6 Modifying pcprint.sh

This shell script uses the "pr" program. If there is a more appropriate local routine, that may be used instead.

A.3.3 Using the 80/PC Shell Scripts

The following sections describe how to generate 80/PC by running the shell scripts that have been examined and, if necessary, modified.

A.3.3.1 Compiling the Source

The command

```
sh -v pccompile.sh
```

compiles all of 80/PC.

A.3.3.2 Linking the Object

Next, the object is linked to make the executable modules by running pclink.sh.

A.3.3.3 Building the Runtime Library

The 80/PC runtime library, "80pl.lib", may now be created. If you have the 80/RL tools, then pcmklib.sh may be used. It is necessary that both *80/CROBJ* and *80/LIBCR* be installed when this script is run.

If you don't have the 80/RL tools, use whatever replacement for pcmklib.sh you wrote earlier to produce the runtime library.

A.3.3.4 Installing 80/PC

80/PC is next installed by running pcinstall.sh. This installs the driver, all the phases and the manual pages according to the directory assignments made in pcdefs.sh. It should be run while logged on as the owner of these directories. It also installs the 80/PC runtime library if the 80/RL tools are being used.

A.3.3.5 Listing 80/PC

A listing of the source of 80/PC may be produced by:

```
sh -v pcprint.sh >pcprint
```

A.4 INSTALLING THE 80/AS ASSEMBLER

This chapter discusses the procedures to produce a working 80/AS from the delivery tape.

A.4.1 Restoring the 80/AS Delivery Tape

The 80/AS delivery tape has one archive file. The restoration process is described in Section A.2.

Restoring the archive will create the following directory structure:

```
80as.d                    (root directory for 80/AS)
```

A.4.2 Modifying the 80/AS Source Files

There are some changes that may be made to the source to customize 80/AS.

A.4.2.1 Modifying custom.h

This file makes several defines that may be changed.

A.4.2.1.1 Temporary File Directory

TDIR is defined to be "/usr/tmp/". This is the name of the directory in which 80/AS should create temporary files. Some UNIX systems, notably PWB, do not normally have the "/usr/tmp" directory. For such systems, TDIR should be redefined – usually to "/tmp".

A.4.2.1.2 Listing Width

WIDTH is the width of the 80/AS listing. This MUST NOT be set less than 72 or greater than 158.

A.4.2.1.3 Listing Page Length

L_PER_P is the total number of lines to be printed on a page, including a 6 line header and a 3 line trailer. DO NOT set L_PER_P between 1 and 9 inclusive. A setting of 0 means that no pagination will be performed; only the header for the first page is generated.

A.4.2.1.4 Form Feeds in Listing

FSW controls whether a form feed (CTRL-L) or multiple blank lines are output at the end of each page of the listing. Setting FSW to YES will cause a form feed to be generated; setting FSW to NO will generate blank lines. Unless the usual output device does not recognize form feed, the switch should probably be set to YES.

A.4.3 Modifying the 80/AS Shell Scripts

All examples in this chapter assume that you have already restored the delivery tape and extracted the source files. It is also assumed that your working directory is "80as.d".

The following shell scripts are provided:

asdefs.sh	which defines things for the other shell scripts
ascompile.sh	which compiles everything
aslink.sh	which links everything
asinstall.sh	which installs everything
asprint.sh	which prints everything

Before proceeding with the generation and installation, examine these shell scripts and make any required changes as indicated by their comments and the following descriptions.

A.4.3.1 Modifying asdefs.sh

This shell script sets up definitions of shell variables used in other shell scripts.

A.4.3.1.1 Installation Directory

The shell variable *I* is set to the directory where 80/AS is to be installed. As distributed, the shell variable is defined as

```
I=/usr/bin
```

A.4.3.1.2 C Compilation Flags

The shell variable *C* is set to the flags to be used for C compilations. As distributed, the shell variable is defined as

```
C=
```

Thus no compilation flags are set.

A.4.3.1.2.1 Treatment of Warning Messages

When compiling with some C compilers, a number of warning messages will be issued. These messages relate to different interpretations of the proper way to use the C language in certain circumstances; they can be ignored. These warnings can be suppressed by adding *-w* to the definition of C.

A.4.3.1.2.2 C Compiler Optimization

Most C compilers can provide processing to attempt to produce more optimal code; this may be specified by adding *-O* to the definition of C. It is possible that this will produce a smaller, faster 80/AS. Considering the problems frequently encountered with the use of such optimizers, we do not recommend this unless you have first successfully installed and tested 80/AS without optimization.

A.4.3.1.2.3 Other C Compiler Options

Any other desired C compiler options of local interest (usually none) may be specified by adding to the definition of C.

A.4.3.1.3 Specifying Linker Options

The shell variable L is set to the flags to be used by the linker. As distributed, the shell variable is defined as

```
L='-i'
```

A.4.3.1.3.1 Split I/D Linking

When the *-i* option is used, the linker produces an executable image (text file) using separate code and data spaces. This is required for proper operation of 80/AS except on the VAX under Berkeley UNIX. Berkeley UNIX uses a demand paging mechanism and the split i/d concept is not relevant to that environment. When generating 80/AS under Berkeley UNIX, remove the "-i" from the definition of L.

A.4.3.1.3.2 Other Linker Options

Any other desired linker options of local interest (usually none) may be specified by adding to the definition of L.

A.4.3.2 Modifying ascompile.sh

The local name of the C compiler should replace "cc" if necessary.

A.4.3.3 Modifying aslink.sh

The local name of the C compiler should replace "cc" if necessary.

A.4.3.4 Modifying asinstall.sh

This shell script, among other things, installs the 80/AS manual pages (provided as nroff source). This should be checked for compatibility with your installation's standards.

A.4.3.5 Modifying asprint.sh

This shell script uses the "pr" program. If there is a more appropriate local routine, that may be used instead.

A.4.4 Using the 80/AS Shell Scripts

The following sections describe how to generate 80/AS by running the shell scripts that have been examined and, if necessary, modified.

A.4.4.1 Compiling the Source

The command

```
sh -v ascompile.sh
```

compiles all of 80/AS.

A.4.4.2 Linking the Object

Next, the object is linked to make the executable modules by running aslink.sh.

A.4.4.3 Installing 80/AS

80/AS is next installed by running asinstall.sh. This installs 80/AS and the manual pages according to the directory assignments made in asdefs.sh. It should be run while logged on as the owner of these directories.

A.4.4.4 Listing 80/AS

A listing of the source of 80/AS may be produced by:

```
sh -v asprint.sh >asprint
```

A.5 INSTALLING THE 80/RL TOOLS

This chapter discusses the procedures to produce working versions of the 80/RL tools from the delivery tape.

A.5.1 Restoring the 80/RL Delivery Tape

The 80/RL delivery tape has one archive file. The restoration process is described in Section A.2.

Restoring the archive will create the following directory structure:

```
80rl.d                    (root directory for 80/RL)
```

A.5.2 Modifying the 80/RL Shell Scripts

All examples in this chapter assume that you have already restored the delivery tape and extracted the source files. It is also assumed that your working directory is "80rl.d". The following shell scripts are provided:

rldefs.sh	which defines things for the other shell scripts
rlcompile.sh	which compiles everything
rllink.sh	which links everything
rlinstall.sh	which installs everything
rlman.sh	which is used by rlinstall.sh to install the manual pages
rlprint.sh	which prints everything

Before proceeding with the generation and installation, examine these shell scripts and make any required changes as indicated by their comments and the following descriptions.

A.5.2.1 Modifying rldefs.sh

This shell script sets up definitions of shell variables used in other shell scripts.

A.5.2.1.1 Installation Directory

The shell variable *I* is set to the directory where the 80/RL tools are to be installed. As distributed, the shell variable is defined as

```
I=/usr/bin
```

A.5.2.1.2 C Compilation Flags

The shell variable *C* is set to the flags to be used for C compilations. As distributed, the shell variable is defined as

```
C=
```

Thus no compilation flags are set.

A.5.2.1.2.1 Temporary File Directory

TDIR is defined in some source modules to be "/usr/tmp/". This is the name of the directory in which the 80/RL tools should create temporary files. Some UNIX systems, notably PWB, do not normally have the "/usr/tmp" directory. For such systems, TDIR should be redefined – usually to "/tmp/". This is done by adding '-DTDIR="/tmp/"' to the definition of *C*.

A.5.2.1.2.2 Treatment of Warning Messages

When compiling with some C compilers, a number of warning messages will be issued. These messages relate to different interpretations of the proper way to use the C language in certain circumstances; they can be ignored. These warnings can be suppressed by adding *-w* to the definition of *C*.

A.5.2.1.2.3 C Compiler Optimization

Most C compilers can provide processing to attempt to produce more optimal code; this may be specified by adding -O to the definition of C. It is possible that this will produce smaller, faster 80/RL tools. Considering the problems frequently encountered with the use of such optimizers, we do not recommend this unless you have first successfully installed and tested the 80/RL tools without optimization.

A.5.2.1.2.4 Other C Compiler Options

Any other desired C compiler options of local interest (usually none) may be specified by adding to the definition of *C*.

A.5.2.1.3 Specifying Linker Options

The shell variable *L* is set to the flags to be used by the linker. As distributed, the shell variable is defined as

```
L='-i'
```

A.5.2.1.3.1 Split I/D Linking

When the *-i* option is used, the linker produces an executable image (text file) using separate code and data spaces. While not required, this is desirable for best performance of the 80/RL tools except on the VAX under Berkeley UNIX. Berkeley UNIX uses a demand paging mechanism and the split i/d concept is not relevant to that environment. When generating the 80/RL tools under Berkeley UNIX, remove the "-i" from the definition of *L*.

A.5.2.1.3.2 Other Linker Options

Any other desired linker options of local interest (usually none) may be specified by adding to the definition of *L*.

A.5.2.2 Modifying rlcompile.sh

The local name of the C compiler should replace "cc" if necessary.

A.5.2.3 Modifying rllink.sh

The local name of the C compiler should replace "cc" if necessary.

A.5.2.4 Modifying rlinstall.sh

This shell script should need no changes.

A.5.2.5 Modifying rlman.sh

This shell script is used by "rlinstall.sh" to install the 80/RL manual pages (provided as nroff source). It should be checked for compatibility with your installation's standards.

A.5.2.6 Modifying rlprint.sh

This shell script uses the "pr" program. If there is a more appropriate local routine, that may be used instead.

A.5.3 Using the 80/RL Shell Scripts

The following sections describe how to generate the 80/RL tools by running the shell scripts that have been examined and, if necessary, modified.

A.5.3.1 Compiling the Source

The command

```
sh -v rlcompile.sh
```

compiles all of the 80/RL tools.

A.5.3.2 Linking the Object

Next, the object is linked to make the executable modules by running rllink.sh.

A.5.3.3 Installing the 80/RL Tools

The 80/RL tools are next installed by running rlinstall.sh. This installs all the tools and the manual pages according to the directory assignments made in rldefs.sh. It should be run while logged on as the owner of these directories. The shell script "rlman.sh" is invoked to install the manual pages.

A.5.3.4 Listing the 80/RL Tools

A listing of the source of the 80/RL tools may be produced by:

```
sh -v rlprint.sh >rlprint
```

Index