GUIDE TO USING iRMX 86™ LANGUAGES

Order Number: 143907-001

Intel Corporation, 3065 Bowers Avenue, Santa Clara, California 95051

REV.	REVISION HISTORY	PRINT DATE
-001	Original Issue	9/81

Additional copies of this manual or other Intel literature may be obtained from:

Literature Department
Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051

The information in this document is subject to change without notice.

Intel Corporation makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Intel Corporation assumes no responsibility for any errors that may appear in this document. Intel Corporation makes no commitment to update nor to keep current the information contained in this document.

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied.

The following are trademarks of Intel Corporation and its affiliates and may be used only to identify Intel products:

BXP	Intelevision	Micromap
CREDIT	Intellec	Multibus
i	iRMX	Multimodule
ICE	iSBC	Plug-A-Bubble
iCS	iSBX	PROMPT
im	Library Manager	Promware
INSITE	MCS	RMX/80
Intel	Megachassis	System 2000
Intel	Micromainframe	UPI
		μScope

and the combination of ICE, iCS, iRMX, iSBC, iSBX, MCS, or RMX and a numerical suffix.

A604/482/ 2K DD

PREFACE

The iRMX 86 languages are a group of language products and utilities that run under the iRMX 86 Operating System as Human Interface commands. This manual provides a general description of each product and refers you to other manuals where you can find more detailed information about the products. This manual also contains information, not found in the other manuals, which describes how to use the language products in an iRMX 86 environment.

This manual is not intended to be a comprehensive reference manual; it provides summary information and refers you to other manuals for most of the detailed reference information. However, you should read this manual before using the language products and utilities on an iRMX 86-based system. This manual provides additional information needed by the iRMX 86 language user and identifies portions of the language and utilities manuals that do not apply to the iRMX 86 language user.

READER LEVEL

This manual is intended for application programmers who are already familiar with:

- The notions of program translation, linking, and locating
- The assembler, compiler, and utilities, as described in the Intel language and utilities manuals
- The iRMX 86 Operating System, especially the Human Interface

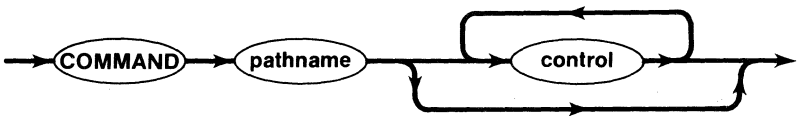
NOTATIONAL CONVENTIONS

This manual uses the following conventions to illustrate syntax.

UPPERCASE	Uppercase information must be entered or coded exactly as shown. You can, however, enter this information in either uppercase or lowercase.
lowercase	Lowercase fields contain variable information. You must enter the appropriate value or symbol for variable fields.
[ ]	Fields within brackets are optional.

...	The elipsis indicates that the preceding syntactic item can be repeated an indefinite number of times. It is often used within brackets following a comma [,...] to indicate that the preceding item can be repeated, but each repetition must be separated by a comma.
underscore	In examples of dialog at the terminal, user input is underscored to distinguish it from system output.

Also, this manual uses the "railroad track" schematic to illustrate the syntax of commands that invoke the language and utility products. This schematic consists of what looks like an aerial view of a model railroad setup, with syntactic elements scattered along the track. To interpret the command syntax, you start at the left side of the schematic, follow the track through all the syntactic elements you desire (sharp turns and backing up are not allowed), and exit at the right side of the schematic. The syntactic elements that you encounter comprise a valid command. For example, a command that consists of a command name, a pathname, and any number of optional controls would have the following schematic representation:

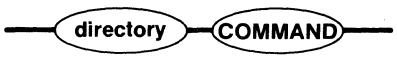


You could enter this command in any of the following forms:

```
COMMAND pathname
COMMAND pathname control
COMMAND pathname control control ...
```

The arrows indicate the possible flow through the tracks; they are omitted in the remainder of this manual.

Syntactic elements that appear close together, such as:



must be entered without spaces or other characters separating them. Syntactic elements that appear farther appart must be entered with spaces separating them. Syntactic elements that can be entered more than once must also be separated with spaces.

RELATED PUBLICATIONS

The following manuals provide additional information that may be helpful to users of this manual.

Manual	Number
8086/8087/8088 Macro Assembly Language Reference Manual for 8086-Based Development Systems	121627
8086/8087/8088 Macro Assembler Operating Instructions for 8086-Based Development Systems	121628
PL/M-86 User's Guide for 8086-Based Development Systems	121636
Pascal-86 User's Guide	121539
FORTRAN-86 User's Guide	121570
iAPX 86,88 Family Utilities User's Guide for 8086-Based Development Systems	121616
Run-Time Support Manual for iAPX 86,88 Applications	121776
iRMX 86™ Nucleus Reference Manual	9803122
iRMX 86™ Basic I/O System Reference Manual	9803123
iRMX 86™ Extended I/O System Reference Manual	143308
iRMX 86™ Loader Reference Manual	143318
iRMX 86™ System Programmer's Reference Manual	142721
iRMX 86™ Human Interface Reference Manual	9803202
EDIT Reference Manual	143587

CONTENTS

CONTENTS (continued)

TABLES

CHAPTER 1. INTRODUCTION

The iRMX 86 languages and utilities are a group of products that provide the iRMX 86 user with full program-development capability. The iRMX 86 languages and utilities include:

EDIT	A powerful text editor.
ASM86	The 8086/8087/8088 macro assembler.
PLM86	The PL/M-86 compiler.
PASC86	The Pascal-86 compiler.
FORT86	The FORTRAN-86 compiler.
LINK86	The 8086 Linker, which combines individually-compiled object modules into a single, relocatable object module.
LOC86	The 8086 Locater, which assigns absolute addresses to relocatable object modules.
LIB86	The 8086 Librarian, which creates and maintains object module libraries.
OH86	A program which converts absolute object modules to hexadecimal format.

All of these products run on iRMX 86 systems, and they are totally compatible with the corresponding language and utility products available with Intel Series III development systems. The products that generate object code (the assembler and the compilers) all generate modules in a standard 8086 object module format that is compatible with UDI (the Universal Development system Interface). Thus programs developed with the iRMX 86 language products can run on any system that supports UDI, as long as the programs make only UDI calls.

iRMX 86 LANGUAGES AND UDI

The Universal Development system Interface (UDI) is a set of routines that provides a standard method for applications to request operating system services. Instead of requesting services directly from the operating system (such as by making iRMX 86 or ISIS-II system calls), an application program can call standard UDI routines to obtain the services. These UDI calls are the same, regardless of the operating system on which the application runs. (The RUN-TIME SUPPORT MANUAL FOR iAPX 86, 88 APPLICATIONS describes these UDI routines in detail.) By using this UDI interface, an application program can be ported from one operating system to another without changing the source code.

Each operating system that supports UDI supplies separate sets of UDI routines, in the form of UDI libraries. Each library translates the individual UDI calls into specific operating system calls. Thus, in order to run an application program on a Series III development system, you would link that program to a Series III UDI library. To run the same program on an iRMX 86-based system, you would instead link the program to an iRMX 86 UDI library.

The iRMX 86 language products conform to the UDI standard. With the exception of EDIT, which is not currently available with the Series III, all iRMX language products are compatible with their Series III counterparts. Therefore you can develop any portion of your code on either system. Regardless of where it was developed, the code can run on an iRMX 86 application system as long as you link it to an iRMX 86 UDI library.

iRMX 86 UDI LIBRARIES

Three UDI libraries are delivered as part of the iRMX 86 Operating System. These are interface libraries that you can link with your programs to allow them to run in an iRMX 86 environment. You should use the library that corresponds to the model of segmentation for your program. The libraries include:

Library	Model of Segmentation
URXLRG.LIB	LARGE or MEDIUM
URXCOM.LIB	COMPACT
URXSML.LIB	SMALL

The language you use when writing your programs and the activities which those programs perform determine whether you need to link the programs to a UDI library. If you write programs in assembly language or PL/M-86, you do not have to link your programs to a UDI library unless the programs invoke specific UDI calls. This is because your assembly language and PL/M-86 programs cannot access operating system services without invoking specific operating system calls (either UDI calls or

iRMX 86 calls). However, if you write your programs in Pascal-86 or FORTRAN-86, you may have to link your program to a UDI library, even if you made no explicit UDI calls. Both Pascal-86 and FORTRAN-86 have formatted I/O features, which, if used, require you to link your programs to UDI libraries.

INSTALLING THE LANGUAGES

The iRMX 86 language products reside on four diskettes. Table 1-1 lists the pathnames of the products and their corresponding diskettes.

Table 1-1. Release Diskettes

RELEASE DISKETTE	PATHNAME
iRMX 86 Assembler Diskette	ASM86
iRMX 86 Language Utilities Diskette	LINK86 LOC86 LIB86 OH86 EDIT
iRMX 86 Pascal Diskette	PASC86
iRMX 86 PL/M Diskette	PLM86
iRMX 86 FORTRAN Diskette	FORT86

To install any of the products on your iRMX 86-based system, you must copy the corresponding file from its diskette to a file on one of your iRMX 86 secondary storage devices. (It is recommended that you place the product on a file of the same name in directory SYSTEM.) This process requires you to be familiar with the iRMX 86 file-naming conventions and the Human Interface. If you need more information on these subjects, refer to the iRMX 86 HUMAN INTERFACE REFERENCE MANUAL. The steps required to install a language product include:

1. Place the release diskette into one of your diskette drives (this procedure assumes drive F1, an iSBC 204 drive).

2. If you have not already attached drive F1, do so by entering the following Human Interface command:

   ATTACHDEVICE F1 AS :F1:

   This command attaches the drive with physical name F1 and associates it with logical name :F1:. The physical name "F1" implies that the system expects your diskette to have been formatted with a 128-byte volume granularity, the granularity of the release diskette.

3. Copy the product from the release diskette to a file on one of your secondary storage devices by entering the following Human Interface command (this procedure assumes that you want to place the product on the drive associated with your default prefix):

   COPY :F1:product TO SYSTEM/product

   where product is the name of the iRMX 86 language product.

Note that if you try to copy all your language products to files on a single diskette, you will run out of space. If you have a diskette-based system, you will have to store your language products on multiple diskettes.

USING THE LANGUAGE PRODUCTS IN AN iRMX 86 ENVIRONMENT

After you have installed the language products on your iRMX 86-based system, you can invoke them by specifying, at your Human Interface terminal, the pathnames of the products followed by any necessary parameters. Chapters 2 through 9 describe in more detail the invocation lines for all products except EDIT. The EDIT REFERENCE MANUAL describes the EDIT invocation line.

Chapters 2 through 9 also contain summaries of the controls or commands associated with the products. Each chapter refers you to additional manuals where you can find detailed reference information. However, you should be aware of some additional information, not described in the language and utility manuals, that will help you to use these language products in an iRMX 86 environment. The following sections describe the basic differences between using the language products on an Intel development system, such as the Series III, and using the products on an iRMX 86-based system.

SYSTEM HARDWARE

Most of the language and utility manuals state that you must have a Series III development system in order to run the products. However, you can run the iRMX 86 language products on your iRMX 86 system. You should disregard all references to Series III hardware in the language and utilities manuals.

PRODUCT INVOCATION

The language and utility manuals state that you must use the RUN command before invoking the products. The RUN command applies to Series III development systems only. When you invoke the products on your iRMX 86-based system, you do not enter the RUN command. Therefore, disregard all references to RUN.

FILE NAMES AND DEVICE NAMES

The language and utility manuals describe the Series III file- and device-naming conventions. When you use the products on your iRMX 86-based system, you should specify iRMX 86 logical names and pathnames for devices and files. Refer to the iRMX 86 HUMAN INTERFACE REFERENCE MANUAL for more information on iRMX 86 logical names and pathnames.

Most of the language products produce output files. If you do not explicitly specify an output file, the product will place the output on a file whose pathname is a modification of the input pathname. The language and utility manuals describe this as placing the output in a file with the same pathname as the input file, but with a different extension. In an iRMX 86-based system, the extension portion of the pathname consists of all characters after the last period in the last pathname element. For example, in the file:

```
PROG/TEST1.SOURCE
```

the extension consists of the characters "SOURCE." If you compile this program with the following statement:

```
PLM86 PROG/TEST1.SOURCE
```

the compiler, by default, places the output in the following file:

```
PROG/TEST1.OBJ
```

If the last element of the pathname does not contain a period (even if some of the directory names in the pathname do contain periods), the language products add an appropriate extension to the input pathname to create an output file. For example, the following file:

```
PROG/IN.PRG/TEST
```

has no extension. If you compile this program with the following command:

```
PLM86 PROG/IN.PRG/TEST
```

the compiler, by default, places the output in a file whose pathname is:

```
PROG/IN.PRG/TEST.OBJ
```

USING THE REST OF THIS MANUAL

Chapters 2 through 9 of this manual describe the individual language products (except EDIT). Each chapter discusses a single product and includes the following information:

- A short discription of the product
- A description of the invocation line for the product
- A short list of the product's controls or commands
- A reference to other manuals where you will find more detailed information about the product
- Additional information about the product that affects the way you use it in an iRMX 86 environment

This manual does not contain a separate chapter for EDIT. The EDIT REFERENCE MANUAL provides the complete description of how to use EDIT in an iRMX 86 environment.

CHAPTER 2. 8086/8087/8088 MACRO ASSEMBLER

The 8086/8087/8088 Macro Assembler is a software development tool that assembles programs written in the 8086/8087/8088 Macro Assembly Language. This assembly language allows you to invoke 8086/8087/8088 machine instructions which let you use all the hardware functions of the 8086, 8087, and 8088 processors.

WRITING ASSEMBLY LANGUAGE PROGRAMS

The 8086/8087/8088 MACRO ASSEMBLY LANGUAGE REFERENCE MANUAL FOR 8086-BASED DEVELOPMENT SYSTEMS is the primary reference source for the assembly language. You should refer to this manual for information on data types, registers, the instruction set, codemacros, and the macro processing language.

INVOKING OPERATING SYSTEM CALLS

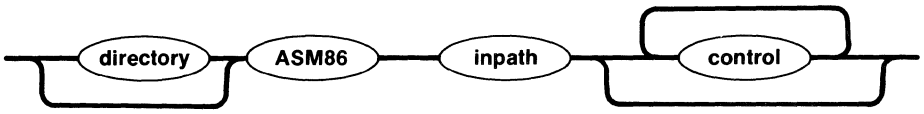
Assembly language programs can call the operating system directly in two ways: they can call UDI procedures or they can invoke iRMX 86 system calls. For information about UDI procedures, refer to the RUN-TIME SUPPORT MANUAL FOR iAPX 86,88 APPLICATIONS. For information about iRMX 86 system calls, refer to the iRMX 86 reference manuals listed in the preface.

USING THE ASSEMBLER

The 8086/8087/8088 MACRO ASSEMBLER OPERATING INSTRUCTIONS FOR 8086-BASED DEVELOPMENT SYSTEMS is the primary source of information for the assembler. However, that manual is written specifically for users of Series III development systems. Therefore, some of the information in that manual does not accurately reflect how to use the assembler in an iRMX 86 environment. Chapter 1 describes most of the differences between using the assembler on a Series III and using it on an iRMX 86-based system. The following sections provide the additional information that you need to operate the assembler in an iRMX 86 environment. When the information in the following sections conflicts with the information in the assembler operating instructions manual, ignore the information in the assembler operating instructions manual.

INVOKING THE ASSEMBLER

To invoke the 8086/8087/8088 macro assembler, enter the following command at your Human Interface terminal:



where:

directory — Portion of the pathname that identifies the device and directories which contain ASM86. You can omit the device designation if the device corresponds to the default prefix for your system. Otherwise, enter the device's logical name, as specified in the last ATTACHDEVICE Human Interface command. Refer to the iRMX 86 HUMAN INTERFACE REFERENCE MANUAL for more information about file and device names.

inpath — Pathname of the file containing containing assembly language source code. Refer to the iRMX 86 HUMAN INTERFACE REFERENCE MANUAL for more information about pathnames.

control — Controls give the assembler information that it needs to define files, identify devices, and produce the desired kind of object code. The next section contains a list of assembler controls. You can enter any number of controls in a single invocation of the assembler. If you do not specify a particular control, the assembler assumes the default for that control.

As with any Human Interface command, you can continue the assembler invocation on additional lines by entering the continuation character (&) after any parameter (as the last character in a line). However, the Human Interface restricts a command to contain no more than 255 characters, including punctuation, embedded blanks, continuation characters, non-executable comments, and carriage returns.

ASSEMBLER CONTROLS

The 8086/8087/8088 MACRO ASSEMBLER OPERATING INSTRUCTIONS FOR 8086-BASED DEVELOPMENT SYSTEMS is the primary reference source for assembler controls. However, Table 2-1 provides a summary of all assembler controls. The following information applies to this table:

- Brackets ([]) denote optional parts of controls. The description lists the default condition if you omit the optional part.
- Controls preceded by the * character are default controls. Unless you explicitly specify otherwise, these controls are in effect.
- Unless otherwise stated, you can enter the controls on the assembler invocation line or include them in the source file. However, some of the controls have little utility unless they are included in the source file.

Table 2-1. 8086/8087/8088 Macro Assembler Controls Summary

CONTROL	DESCRIPTION
DEBUG DB	Places local symbol information in the object file for symbolic debugging.
*NODEBUG NODB	Does not place local symbol information in the object file.
EJECT EJ	Indicates a page break location.
ERRORPRINT[(pathname)] EP	Generates a list that summarizes the errors encountered in assembly. The default file is :CO:.
*NOERRORPRINT NOEP	Does not generate a list of the errors encountered.
* Default control	

Table 2-1. 8086/8087/8088 Macro Assembler Controls Summary (continued)

CONTROL	DESCRIPTION
GEN GE	Provides a full listing of all macro calls at all levels, as well as the macro expansions, in the listing file.
NOGEN NOGE	Provides a listing of source file lines only. The bodies of macros are not printed, except those macro expansion lines that contain errors.
*GENONLY GO	Provides a listing of the text of the fully expanded source file.
INCLUDE(pathname) IC	Includes the specified file as input to the assembler.
*LIST LI	Directs the assembler to include listing lines.
NOLIST NOLI	Directs the assembler to suppress printing of listing lines. Error messages, with appropriate line numbers and lines, are printed, however.
*MACRO MR	Directs the assembler to process macros.
NOMACRO NOMR	Allows the user who has included no macro calls in the source text to save assembly time.
*OBJECT[(pathname)] OJ	Generates object code and writes that code to the specified file. If you omit the pathname parameter, the default object file has the same pathname as the source file (with an extension of OBJ) and resides on the same device as the source file.
NOOBJECT NOOJ	Does not generate object code.
* Default control	

Table 2-1. 8086/8087/8088 Macro Assembler Controls Summary (continued)

CONTROL	DESCRIPTION
*PAGELENGTH(n) PL	Indicates the number of lines to be contained on each page of the listing file. Default is 60 lines.
*PAGEWIDTH(n) PW	Sets the maximum number of characters allowed on each line of the listing file. Default is 120 characters/line.
*PAGING PI	Formats the listing file into numbered pages with headers at each page break.
NOPAGING NOPI	Does not format the listing file into numbered pages.
*PRINT[(pathname)] PR	Generates a listing file on the specified file or device. If you specify neither PRINT nor NOPRINT on the command line, the listing file has the same pathname as the source file (but with the extension LST) and resides on the same device as the source file.
NOPRINT NOPR	Suppresses the listing file. The result is that NOPAGING, NOSYMBOLS, and NOXREF are implied.
SAVE ... RESTORE	These controls allow the settings of certain general controls to be saved on a stack (before an INCLUDE control switches the input source to another file) and then restored after the INCLUDE.
SYMBOLS SB	Produces a symbol table at the end of the listing file.
*NOSYMBOLS NOSB	Omits the symbol table from the end of the listing file.
*TITLE(name) TT	Gives a page or set of pages a title. The default title is the module name specified with the NAME directive.
* Default control	

Table 2-1. 8086/8087/8088 Macro Assembler Controls Summary (continued)

CONTROL	DESCRIPTION
*WORKFILES(:lognm1:, :lognm2:) WF	Assigns the temporary assembler-generated files to the devices or directories corresponding to the specified logical names. Refer to the iRMX 86 HUMAN INTERFACE REFERENCE MANUAL for a description of logical names. The default is placement of the temporary files on the :WORK: directory.
XREF XR	Provides a symbol list with a cross-reference of the lines where user-defined symbols are defined, referenced, and purged. XREF overrides NOSYMBOLS.
*NOXREF NOXR	Does not produce cross-referencing information.
* Default control	

ERROR MESSAGES

If the assembler returns an ASM86 I/O ERROR message, it also returns an iRMX 86 condition code. To interpret this condition code, refer to the iRMX 86 EXTENDED I/O SYSTEM REFERENCE MANUAL.

The 8086/8087/8088 MACRO ASSEMBLER OPERATING INSTRUCTIONS FOR 8086-BASED DEVELOPMENT SYSTEMS provides complete descriptions for the remaining error messages.

EXAMPLE

Suppose the assembler resides in file SYSTEM/ASM86 on drive :F0: (where :F0: is the default prefix) and your assembly language source program resides in file PROG/TEST1.SRC on drive :F1:. The following command assembles that program:

```
-ASM86 :F1:PROG/TEST1.SRC
iRMX 86 8086/8087/8088 MACRO ASSEMBLER  V1.0

ASSEMBLY COMPLETE, NO ERRORS FOUND
-
```

The assembler places the object code in file PROG/TEST1.OBJ on drive :F1:.

CHAPTER 3. PL/M-86 COMPILER

The PL/M-86 compiler is a software development tool that compiles programs written in the PL/M-86 language. This language is a high-level language designed for both system and application programming.

WRITING PL/M-86 PROGRAMS

The PL/M-86 USER'S GUIDE FOR 8086-BASED DEVELOPMENT SYSTEMS is the primary reference for information concerning the PL/M-86 language. You should refer to this manual for information on the language elements and for information on how to write PL/M-86 programs. However, the PL/M-86 user's guide contains some information that is either incomplete or does not apply to programs that run in an iRMX 86 environment. This information includes:

- INPUT AND OUTPUT

  PL/M-86 does not provide formatted I/O capabilities like those of FORTRAN or PASCAL, but it does provide built-in procedures (such as INPUT and OUTPUT) which perform I/O functions. The PL/M-86 user's guide discusses these procedures. However, in order to use these functions, you must specify the correct input and output port numbers. The iRMX 86 Basic and Extended I/O Systems also provide you with I/O capabilities and a complete file system. Refer to the iRMX 86 BASIC I/O SYSTEM REFERENCE MANUAL and the iRMX 86 EXTENDED I/O SYSTEM REFERENCE MANUAL for more information.

- FLOATING-POINT ARITHMETIC

  The PL/M-86 user's guide states that you can use the REAL math facility if your system contains an 8087 Numeric Data Processor or if you link your program to an 8087 emulator. However, the iRMX 86 Operating System does not support the use of the emulator. If you intend to use the REAL math facility in a multitasking environment, ensure that your iRMX 86 hardware system contains an 8087 Numeric Data Processor.

- INTERRUPT PROCESSING

  The PL/M-86 user's guide contains an appendix which describes run-time interrupt processing. The information in this appendix is intended for users who run PL/M-86 programs in a non-iRMX 86 environment. You should disregard this appendix unless you are writing programs that will run in a non-iRMX 86 environment. To set up interrupt handlers and tasks for an iRMX 86-based system, refer to the iRMX 86 NUCLEUS REFERENCE MANUAL.

MAKING OPERATING SYSTEM CALLS

PL/M-86 programs can call the operating system directly in two ways: they can call UDI procedures or they can invoke iRMX 86 system calls. For information about UDI procedures, refer to the RUN-TIME SUPPORT MANUAL FOR iAPX 86,88 APPLICATIONS. For information about iRMX 86 system calls, refer to the iRMX 86 reference manuals listed in the preface.

USING THE COMPILER

The PL/M-86 USER'S GUIDE FOR 8086-BASED DEVELOPMENT SYSTEMS is the primary reference source for information on how to use the PL/M-86 compiler. However, the chapters of that manual that discuss compiler invocation and compiler controls are written specifically for users of Series III development systems. Therefore, some of the information in those chapters does not accurately reflect how to use the compiler in an iRMX 86 environment. Chapter 1 describes most of the differences between using the compiler on a Series III and using it in an iRMX 86 environment. However, the following item also applies:

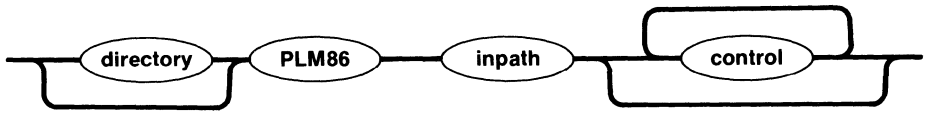
- INTERMODULE CROSS-REFERENCE INFORMATION

  The PL/M-86 user's guide describes a program called IXREF that produces an intermodule cross-reference listing. This program does not currently run on an iRMX 86-based system. Therefore, you should ignore all references to the IXREF program.

The following sections provide the additional information you need to operate the PL/M-86 compiler in an iRMX 86 environment. When the information in the following sections conflicts with the information in the PL/M-86 user's guide, ignore the information in the user's guide.

INVOKING THE PL/M-86 COMPILER

To invoke the PL/M-86 compiler, enter the following command at your Human Interface terminal:



where:

directory	Portion of the pathname that identifies the device and directories which contain PLM86. You can omit the device designation if the device corresponds to the default prefix for your system. Otherwise, enter the device's logical name, as specified in the last ATTACHDEVICE Human Interface command. Refer to the iRMX 86 HUMAN INTERFACE REFERENCE MANUAL for more information about file and device names.
inpath	Pathname of the file containing PL/M-86 source code. Refer to the iRMX 86 HUMAN INTERFACE REFERENCE MANUAL for more information about pathnames.
control	Controls give the compiler information that it needs to define files, identify devices, and produce the desired kind of object code. The next section contains a list of compiler controls. You can enter any number of controls in a single invocation of the compiler. If you do not specify a particular control, the compiler assumes the default for that control.

As with any Human Interface command, you can continue the compiler invocation on additional lines by entering the continuation character (&) after any parameter (as the last character in a line). However, the Human Interface restricts a command to contain no more than 255 characters, including punctuation, embedded blanks, continuation characters, non-executable comments, and carriage returns.

COMPILER CONTROLS

The PL/M-86 USER'S GUIDE FOR 8086-BASED DEVELOPMENT SYSTEMS is the primary reference source for the PL/M-86 compiler controls. However, Table 3-1 provides a summary of all compiler controls. The following information applies to this table:

- Brackets ([]) denote optional parts of controls. The description lists the default condition if you omit the optional part.
- Controls preceded by the * character are default controls. Unless you specify otherwise, these controls are in effect.
- Unless otherwise stated, you can enter the controls on the compiler invocation line or include them in the source file. However, some of the controls have little utility unless they are included in the source file.

Table 3-1. PL/M-86 Compiler Controls Summary

CONTROL	DESCRIPTION
CODE	Directs the compiler to display the generated object code, in standard assembly language format. This code is interleaved with the program code on the listing file.
*NOCODE	Directs the compiler to suppress the listing of the generated object code.
*COND	Causes the compiler to display on the listing file all text within IF blocks, even if the IF blocks are not compiled.
NOCOND	Causes the compiler to suppress the listing of IF blocks, if those IF blocks are not compiled.
DEBUG	Places local symbol information in the object file for symbolic debugging.
*NODEBUG	Does not place local symbol information in the object file.
* Default control	

Table 3-1. PL/M-86 Compiler Controls Summary (continued)

CONTROL	DESCRIPTION
EJECT	Indicates a page break location. The line containing the EJECT control begins a new page.

IF
ELSE
ELSIF
ENDIF
	Provide conditional compilation capabilities. These controls cannot be used on the compiler invocation line.

INCLUDE(pathname)	Includes the specified file as input to the compiler.
*INTVECTOR	Creates an interrupt vector consisting of a 4-byte entry for each interrupt procedure in the module.
NOINTVECTOR	Does not generate an interrupt vector.
IXREF[(pathname)]	Writes an intermediate intermodule cross-reference listing to the specified file. The default intermediate cross-reference file has the same pathname as the source file but with the extension IXI.
*NOIXREF	Does not generate an intermediate intermodule cross-reference file.
*LEFTMARGIN(column)	Specifies the left margin of the source input. If not specified, the default is LEFTMARGIN(1).
*LIST	Directs the compiler to include listing lines in the listing file.
NOLIST	Directs the compiler to suppress printing of listing lines. Error messages, with appropriate line numbers and lines, are printed, however.
* Default control	

Table 3-1. PL/M-86 Compiler Controls Summary (continued)

CONTROL	DESCRIPTION
*OBJECT[(pathname)]	Generates object code and writes that code to the specified iRMX 86 file. The default object file has the same pathname as the source file (but with an extension of OBJ) and resides on the same device as the source file.
NOOBJECT	Does not generate object code.
*OPTIMIZE(n)	Governs the kind of optimization performed in generating object code. The n can range from 0 to 3. If you omit this control, OPTIMIZE(1) is the default.
OVERFLOW	Detects overflow conditions in signed (INTEGER) arithmetic.
*NOOVERFLOW	Does not check for overflow conditions.
*PAGELENGTH(n)	Indicates the number of lines to be contained on each page of the listing file. Default is 60 lines/page.
*PAGEWIDTH(n)	Sets the maximum number of characters allowed on each line of the listing file. Default is 120 characters/line.
*PAGING	Formats the listing file into numbered pages with headers at each page break.
NOPAGING	Does not format the listing file into numbered pages. The symbol table is separated from the source file by four lines.
* Default control	

Table 3-1. PL/M-86 Compiler Controls Summary (continued)

CONTROL	DESCRIPTION
*PRINT[(pathname)]	Generates a listing file on the specified file or device. If you specify neither PRINT nor NOPRINT on the command line, the listing file has the same pathname as the source file (but with an extension of LST) and resides on the same device as the source file.
NOPRINT	Suppresses the listing file.
*RAM	Places constants within the DATA segment for all segmentation models except LARGE, in which constants are placed in the CODE segment.
ROM	Places constants in the CODE segment.
SAVE ... RESTORE	These controls allow the settings of certain general controls to be saved on a stack before an INCLUDE control switches the input source to another file and then restored after the INCLUDE.
SET(switch assignment)	Sets values for compiler switches.
RESET(switch list)	Sets all switches in the switch list to "false" (0).
*SMALL COMPACT MEDIUM LARGE	Specifies the memory size requirements of the program being compiled.
SUBTITLE('name')	Causes the specified subtitle to appear on all pages until another SUBTITLE control appears.
SYMBOLS	Produces a symbol table at the end of the listing file.
*NOSYMBOLS	Omits the symbol table from the end of the listing file.
* Default control	

Table 2-1. PL/M-86 Compiler Controls Summary (continued)

CONTROL	DESCRIPTION
*TITLE('name')	Places the specified name on the title line of each page of listed output. If not specified, the default name is the module name.
*TYPE	Places information about the types of symbols into the object module.
NOTYPE	Does not place type definitions into the object module.
*WORKFILES(:lognm1:, :lognm2:) WF	Assigns the temporary assembler-generated files to the devices or directories corresponding to the specified logical names. Refer to the iRMX 86 HUMAN INTERFACE REFERENCE MANUAL for a description of logical names. The default is placement of the temporary files on the :WORK: directory.
XREF	Provides, in the listing file, a symbol list with a cross-reference of the lines where user-defined symbols are defined, referenced, and purged.
*NOXREF	Does not produce cross-referencing information.
* Default control	

ERROR MESSAGES

If the compiler returns a PL/M-86 I/O error message, it also returns an iRMX 86 exception code. To interpret this exception code, refer to the iRMX 86 EXTENDED I/O SYSTEM REFERENCE MANUAL.

The PL/M-86 USER'S GUIDE FOR 8086-BASED DEVELOPMENT SYSTEMS provides complete descriptions of the remaining error messages.

EXAMPLE

Suppose the PL/M-86 compiler resides in file PLM86 on device :F1: and a PL/M-86 source program, DEVELOP/SOURCE/TEST2, resides on device :F2:. The following command compiles the source program:

```
-:F1:PLM86 :F2:DEVELOP/SOURCE/TEST2  &
**          OBJECT(:F2:DEVELOP/OBJECT/TEST2) LARGE
iRMX 86 PL/M-86 COMPILER V1.0
PL/M-86 COMPILATION COMPLETE.     0 WARNINGS,     0 ERRORS
-
```

The compiler reads the input from file TEST2 in directory DEVELOP/SOURCE and places the object code in file TEST2 in directory DEVELOP/OBJECT.

CHAPTER 4. PASCAL-86 COMPILER

The Pascal-86 compiler is a software development tool that compiles programs written in the Pascal-86 language. This language is a higher-level language than PL/M-86, and therefore well suited to application programming.

WRITING PASCAL-86 PROGRAMS

The PASCAL-86 USER'S GUIDE is the primary reference for information concerning the Pascal-86 language. You should refer to this manual for information on the language elements and for information on how to write Pascal-86 programs. However, the PASCAL-86 USER'S GUIDE contains some information that is either incomplete or does not apply to programs that run in an iRMX 86 environment. This information includes:

- FLOATING-POINT ARITHMETIC

  The PASCAL-86 USER'S GUIDE states that your programs can use floating-point arithmetic if the hardware system on which you run your program contains an 8087 Numeric Data Processor or if you link your program to an 8087 emulator. However, the iRMX 86 Operating System does not support the use of the emulator. If you intend to use floating-point arithmetic in an iRMX 86 multitasking environment, ensure that your hardware system contains an 8087 Numeric Data Processor.

- INPUT AND OUTPUT

  Pascal-86 provides both formatted I/O capabilities (such as READLN and WRITELN) and port I/O procedures (such as INBYT and OUTBYT). Parts of the PASCAL-86 USER'S GUIDE may lead you to believe that you can't use the formatted I/O facilities if you intend to run your programs in an iRMX 86 environment. This is not correct. Programs that run in an iRMX 86 environment can use either method of I/O, although to use port I/O you must specify the correct port addresses.

- INTERRUPT CONTROL PROCEDURES

  The PASCAL-86 USER'S GUIDE describes several procedures that aid in interrupt processing. Since the iRMX 86 Operating System implements its own form of interrupt processing (refer to the iRMX 86 NUCLEUS REFERENCE MANUAL), Pascal-86 programs that run in an iRMX 86 environment must not use these Pascal-86 interrupt control procedures.

- RUN-TIME INTERFACE

  The PASCAL-86 USER'S GUIDE contains an appendix describing the run-time interface. This appendix states that you must provide your own interface procedures if your programs run in any environment other than that of the Series III. However, this is not true for the iRMX 86 environment. You do not need to develop your own run-time interface as long as you link your programs to the UDI interface library URXLRG.LIB. Therefore, when running your programs in an iRMX 86 environment, you can ignore the run-time interface appendix.

INVOKING OPERATING SYSTEM CALLS

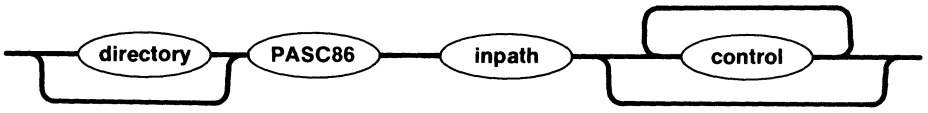
Pascal-86 programs cannot directly call the operating system, either by calling UDI procedures or by invoking iRMX 86 system calls. Currently, the only way to access the iRMX 86 Operating System is through Pascal-86 built-in procedures such as READLN and WRITELN.

USING THE COMPILER

The PASCAL-86 USER'S GUIDE is the primary reference source for information on how to use the PASCAL-86 compiler. However, the chapters of that manual that discuss compiler invocation and compiler controls are written specifically for users of Series III development systems. Therefore, some of the information in those chapters does not accurately reflect how to use the compiler in an iRMX 86 environment. Chapter 1 describes most of the differences between using the assembler on a Series III and using it on an iRMX 86-based system. The following sections provide the additional information you need to operate the Pascal-86 compiler in an iRMX 86 environment. When the information in the following sections conflicts with the information in the PASCAL-86 USER'S GUIDE, ignore the information in the user's guide.

INVOKING THE PASCAL-86 COMPILER

To invoke the Pascal-86 compiler, enter the following command at your Human Interface terminal:



where:

directory	Portion of the pathname that identifies the device and directories which contain PASC86. You can omit the device designation if the device corresponds to the default prefix for your system. Otherwise, enter the device's logical name, as specified in the last ATTACHDEVICE Human Interface command. Refer to the iRMX 86 HUMAN INTERFACE REFERENCE MANUAL for more information about file and device names.
inpath	Pathname of the file containing Pascal-86 source code. Refer to the iRMX 86 HUMAN INTERFACE REFERENCE MANUAL for more information about pathnames.
control	Controls give the compiler information that it needs to define files, identify devices, and produce the desired kind of object code. The next section contains a list of compiler controls. You can enter any number of controls in a single invocation of the compiler. If you do not specify a particular control, the compiler assumes the default for that control.

As with any Human Interface command, you can continue the compiler invocation on additional lines by entering the continuation character (&) after any parameter (as the last character in a line). However, the Human Interface restricts a command to contain no more than 255 characters, including punctuation, embedded blanks, continuation characters, non-executable comments, and carriage returns.

COMPILER CONTROLS

The PASCAL-86 USER'S GUIDE is the primary reference source for the Pascal-86 compiler controls. However, Table 4-1 provides a summary of all compiler controls. The following information applies to this table:

- Brackets ([]) denote optional parts of controls. The description lists the default condition if you omit the optional part.
- Controls preceded by the * character are default controls. Unless you specify otherwise, these controls are in effect.
- Unless otherwise stated, you can enter the controls on the compiler invocation line or include them in the source file. However, some of the controls have little utility unless they are included in the source file.

Table 4-1. Pascal-86 Compiler Controls Summary

CONTROL	DESCRIPTION
CHECK	Checks for invalid references, overflow, and out-of-range assignments and subscripts during compilation and run time.
*NOCHECK	Does not do any checking.
CODE	Lists the approximate assembly code on the list file.
*NOCODE	Suppresses the listing of assembly code.
DEBUG	Generates debug records in the object module.
*NODEBUG	Does not generate debug records.
EJECT	Forces the start of a new page of printed output.
*ERRORPRINT[(pathname)]	Writes all compiler-generated error messages to the specified file. If you omit this control (or the pathname), the compiler writes the information to :CO:.
NOERRORPRINT	Does not write compiler-generated error messages.
*EXTENSIONS	Allows Intel extensions to standard Pascal.
NOEXTENSIONS	Issues an extension warning whenever the source program contains any Intel extensions to standard Pascal.
INCLUDE(pathname)	Includes the specified file as input to the compiler.
INTERRUPT(proc[=n][,...])	Designates procedures as interrupt procedures and generates an interrupt vector.
* Default condition	

Table 4-1. Pascal-86 Compiler Controls Summary (continued)

CONTROL	DESCRIPTION
*LIST	Lists source lines in the listing file.
NOLIST	Suppresses the listing of source lines.
*OBJECT[(pathname]) NOOBJECT	Generates object code and writes that code to the specified iRMX 86 file. The default object file has the same pathname as the source file (but with an extension of OBJ) and resides on the same device as the source file. Does not generate an object file.
*PRINT[(pathname)] NOPRINT	Generates a listing file on the specified file or device. If you specify neither PRINT nor NOPRINT on the command line, the listing file has the same pathname as the source file (but with an extension of LST) and resides on the same device as the source file. Suppresses the listing file.
SUBTITLE('name')	Causes the specified subtitle to appear on all pages until another SUBTITLE control appears.
*TITLE('name')	Places the specified name on the title line of each page of listed output. If you do not specify this control, the default name is the module name.
*TYPE NOTYPE	Places information about the types of symbols into the object module. Does not place type definitions into the object module.
*XREF NOXREF	Provides, in the listing file, a cross-reference listing of source program identifiers. Does not produce cross-referencing information.
* Default control	

ERROR MESSAGES

If the Pascal-86 compiler returns a fatal error whose number is in the range 9000-9002 or 9006-9017, it also returns an iRMX 86 exception code. To interpret this exception code, refer to the iRMX 86 EXTENDED I/O SYSTEM REFERENCE MANUAL.

EXAMPLE

Suppose the Pascal-86 compiler resides in file PASC86 on drive :F0: (the default prefix for your system) and a Pascal-86 source program, PROG/TEST3.SRC resides on drive :F1:. The following command compiles that program:

```
-PASC86 :F1:PROG/TEST3.SRC
iRMX 86 Pascal-86 V1.0
PARSE(0), ANALYSE(0), NOXREF, OBJECT

    COMPILATION OF TEST3 COMPLETED, 0 ERROR DETECTED,
    END OF Pascal-86 COMPILATION.
```

Pascal-86 places the object code in file PROG/TEST3.OBJ on drive :F1:.

LINKING PASCAL-86 PROGRAMS

The PASCAL-86 USER'S GUIDE describes the run-time support libraries that you need to link with your programs in order for those programs to run on a Series III development system. With two exceptions, this is the same set of libraries that you need in order to run your programs on an iRMX 86-based system. The exceptions are:

- The modules E8087.LIB and E8087 provide support for the 8087 emulator. This emulator is not supported in an iRMX 86 environment.

- The module LARGE.LIB, which is described in the manual, is the system services (UDI) library for the Series III environment. To obtain system services from an iRMX 86 environment, link your programs to the iRMX 86 UDI library (URXLRG.LIB) instead of to LARGE.LIB. You do not need to provide any other special libraries.

The remainder of the libraries provide the same functions as listed in the PASCAL-86 USER'S GUIDE. Therefore, when linking your Pascal-86 programs, include some or all of the following libraries (in the order listed here):

CEL.LIB	The floating-point built-in function library. You must link your program to this library if the program calls any floating-point functions.

P86RN0.LIB P86RN1.LIB	Formatting and I/O libraries which are required for any run-time I/O support. If your programs do not perform any I/O, you should link your program to RTNULL.LIB to resolve external references.
P86RN2.LIB P86RN3.LIB	Default logical record system libraries. If your programs perform I/O and run in an iRMX 86 environment, you should link the programs to these libraries.
8087.LIB	The 8087 Numeric Data Processor support library. If you require floating-point arithmetic, you should include the 8087 Numeric Data Processor in your hardware system and link your program to this library. If your program does not perform any floating-point arithmetic, you should link it to 87NULL.LIB to resolve external references.
URXLRG.LIB	The iRMX 86 UDI library. If you plan to run your program in an iRMX 86 environment, you should link it to this library.

For example, the following command links a Pascal-86 program (INCHES.OBJ), which does not use floating-point arithmetic, to the libraries it needs for execution in an iRMX 86 environment.

```
-:LINK86  :F1:INCHES.OBJ,  &
**        :F1:P86RN0.LIB,  &
**        :F1:P86RN1.LIB,  &
**        :F1:P86RN2.LIB,  &
**        :F1:P86RN3.LIB,  &
**        :F1:87NULL.LIB,  &
**        :F1:URXLRG.LIB   &
**  TO    :F1:INCHES  BIND  MEMPOOL(+2000H)

iRMX 86 8086 LINKER, V1.0
-
```

CHAPTER 5. FORTRAN-86 COMPILER

The FORTRAN-86 compiler is a software development tool that compiles programs written in the FORTRAN-86 language. This language is a superset of the FORTRAN 77 subset defined by the American National Standards Institute (ANSI).

WRITING FORTRAN-86 PROGRAMS

The FORTRAN-86 USER'S GUIDE is the primary reference for information concerning the FORTRAN-86 language. You should refer to this manual for information on the language elements and for information on how to write FORTRAN-86 programs. However, the FORTRAN-86 USER'S GUIDE contains some information that is either incomplete or does not apply to programs that run in an iRMX 86 environment. This information includes:

- INPUT/OUTPUT STATEMENTS

  The FORTRAN-86 USER'S GUIDE states that the console input device and the console output device are preconnected for units 5 and 6 respectively in a Series III system. This is also true for an iRMX 86 environment.

  You cannot use the FORTRAN statements BACKSPACE and ENDFILE to manipulate iRMX 86 physical files, such as :CI:, :CO:, line printers, or other such files. FORTRAN-86 returns a run-time error in such cases.

  You cannot use the OPEN statement to open an iRMX 86 physical file for direct access. Physical files are by definition sequential files and must be opened for sequential access only. Attempts to open physical files for direct access result in run-time errors.

  If you use the OPEN statement to open an iRMX 86 file and specify a status of UNKNOWN or specify no status at all, the iRMX 86 Operating System will open the file with a mode appropriate to the file. That is, it will open iRMX 86 physical files for reading if they correspond to input devices (such as :CI:), for writing if they correspond to output devices (such as :CO: and line printers), and for both reading and writing if they correspond to I/O devices such as disk drives. The Operating System will open named files for both reading and writing. Refer to the description of the S$OPEN system call in the iRMX 86 EXTENDED I/O SYSTEM REFERENCE MANUAL for more information about access modes.

- FLOATING-POINT ARITHMETIC

  The FORTRAN-86 USER'S GUIDE states that you can use the floating-point data types (REAL, DOUBLE PRECISION, and TEMPREAL) and perform floating-point operations if your system contains an 8087 Numeric Data Processor or if you link your program to an 8087 emulator. However, the iRMX 86 Operating System does not support the use of the emulator. If you intend to use floating-point arithmetic in a multitasking environment, ensure that your iRMX 86 hardware system contains an 8087 Numeric Data Processor.

- INTERRUPT PROCESSING

  The FORTRAN-86 USER'S GUIDE contains an appendix which describes run-time interrupt processing. The information in this appendix is intended for users who run FORTRAN-86 programs in a non-iRMX 86 environment. You should disregard the sections that discuss interrupt procedures unless you are writing programs that will run in a non-iRMX 86 environment. To set up interrupt handlers and tasks for an iRMX 86-based system, refer to the iRMX 86 NUCLEUS REFERENCE MANUAL.

INVOKING OPERATING SYSTEM CALLS

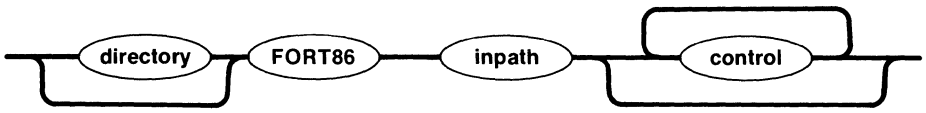
FORTRAN-86 programs cannot directly call the operating system, either by calling UDI procedures or by invoking iRMX 86 system calls. Currently, the only way to access the iRMX 86 Operating System is through FORTRAN-86 statements such as OPEN, CLOSE, BACKSPACE, REWIND, ENDFILE, READ, WRITE, and PRINT.

USING THE COMPILER

The FORTRAN-86 USER'S GUIDE is the primary reference source for information on how to use the FORTRAN-86 compiler. However, the chapters of that manual that discuss compiler invocation and compiler controls are written specifically for users of Series III development systems. Therefore, some of the information in those chapters does not accurately reflect how to use the compiler in an iRMX 86 environment. Chapter 1 describes most of the differences between using the compiler on a Series III and using it on an iRMX 86-based system. The following sections provide the additional information you need to operate the FORTRAN-86 compiler in an iRMX 86 environment. When the information in the following sections conflicts with the information in the FORTRAN-86 USER'S GUIDE, disregard the information in the user's guide.

INVOKING THE FORTRAN-86 COMPILER

To invoke the FORTRAN-86 compiler, enter the following command at your Human Interface terminal:



where:

directory	Portion of the pathname that identifies the device and directories which contain FORT86. You can omit the device designation if the device corresponds to the default prefix for your system. Otherwise, enter the device's logical name, as specified in the last ATTACHDEVICE Human Interface command. Refer to the iRMX 86 HUMAN INTERFACE REFERENCE MANUAL for more information about file and device names.
inpath	Pathname of the file containing FORTRAN-86 source code. Refer to the iRMX 86 HUMAN INTERFACE REFERENCE MANUAL for more information about pathnames.
control	Controls give the compiler information that it needs to define files, identify devices, and produce the desired kind of object code. The next section contains a list of compiler controls. You can enter any number of controls in a single invocation of the compiler. If you do not specify a particular control, the compiler assumes the default for that control.

As with any Human Interface command, you can continue the compiler invocation on additional lines by entering the continuation character (&) after any parameter (as the last character in a line). However, the Human Interface restricts a command to contain no more than 255 characters, including punctuation, embedded blanks, continuation characters, non-executable comments, and carriage returns.

COMPILER CONTROLS

The FORTRAN-86 USER'S GUIDE is the primary reference source for the FORTRAN-86 compiler controls. However, Table 5-1 provides a summary of all compiler controls. The following information applies to this table:

- Brackets ([]) denote optional parts of controls. The description lists the default condition if you omit the optional part.
- Controls preceded by the * character are default controls. Unless you specify otherwise, these controls are in effect.
- Unless otherwise stated, you can enter the controls on the compiler invocation line or include them in the source file. However, some of the controls have little utility unless they are included in the source file.

Table 5-1. FORTRAN-86 Compiler Controls Summary

CONTROL	DESCRIPTION
CODE CO	Lists pseudo-assembly code on the list file.
*NOCODE NOCO	Suppresses the listing of pseudo-assembly code.
DEBUG DB	Generates debug records in the object module.
*NODEBUG NODB	Does not generate debug records.
DO66	Assumes that all DO-loops in the program conform to the ANSI 1966 standard (DO-loops must perform at least one iteration during execution).
*DO77	Assumes that all DO-loops in the program conform to the ANSI 1977 standard (zero iteration DO-loops are permitted).
* Default control	

Table 5-1. FORTRAN-86 Compiler Controls Summary (continued)

CONTROL	DESCRIPTION
EJECT[(number)] EJ	Forces the start of a new page of printed output.
ERRORLIMIT(number) EL *NOERRORLIMIT NOEL	Terminates compilation prematurely after the compiler detects the specified number of errors. Allows compilation to continue until the end of the program, regardless of the number of errors the compiler encounters.
EXCEPTION	Compiles the subroutine as an exception handling procedure.
FREEFORM FF *NOFREEFORM NOFF	Accepts programs written in nonstandard input format. Accepts programs only if they are written in standard format.
IGNORE(control[,...]) IN	Ignores the specified general controls during compilation.
INCLUDE(pathname) IC	Includes the specified file as input to the compiler.
INTERRUPT(proc[=n] [,...]) IT	Designates procedures as interrupt procedures.
*LIST LI NOLIST NOLI	Lists source lines in the listing file. Suppresses the listing of source lines until the occurrence of the next LIST control.
* Default control	

Table 5-1. FORTRAN-86 Compiler Controls Summary (continued)

CONTROL	DESCRIPTION
*OBJECT[(pathname)] OJ	Generates object code and writes that code to the specified iRMX 86 file. The default object file has the same pathname as the source file (but with an extension of OBJ) and resides on the same device as the source file.
NOOBJECT NOOJ	Does not generate an object file.
*PAGELENGTH(n) PL	Indicates the number of lines to be contained on each page of the listing file. Default is 60 lines.
*PAGEWIDTH(n) PW	Sets the maximum number of characters allowed on each line of the listing file. Default is 120 characters/line.
*PRINT[(pathname)] PR	Generates a listing file on the specified file or device. If you specify neither PRINT nor NOPRINT on the command line, the listing file has the same pathname as the source file (but with an extension of LST) and resides on the same device as the source file.
NOPRINT NOPR	Suppresses the listing file.
REENTRANT RE	Indicates that a particular subroutine or function can call itself.
STORAGE(INTEGER intlen[, LOGICAL* loglen]) STORAGE(LOGICAL* loglen[, INTEGER* intlen]) ST	Specifies the default lengths, in bytes, applied to INTEGER and/or LOGICAL data items. Default is STORAGE(INTEGER*2,LOGICAL*1).
* Default control	

Table 5-1. FORTRAN-86 Compiler Controls Summary (continued)

CONTROL	DESCRIPTION
SUBTITLE('name') ST	Causes the specified subtitle to appear on all pages until another SUBTITLE control appears.
SYMBOLS SB *NOSYMBOLS NOSB	Produces a symbol table at the end of the listing file. Omits the symbol table from the end of the listing file.
*TITLE('name') TT	Places the specified name on the title line of each page of listed output. If not specified, the default name is the module name.
*TYPE TY NOTYPE NOTY	Places information about the types of symbols into the object module. Does not place type definitions into the object module.
*XREF XR NOXREF NOXR	Provides a symbol-table listing of source program identifiers. Does not produce symbol table information.
* Default control	

LINKING FORTRAN-86 PROGRAMS

The FORTRAN-86 USER'S GUIDE describes the run-time support libraries that you need to link with your programs in order for those programs to run on a Series III development system. With two exceptions, this is the same set of libraries that you need in order to run your programs on an iRMX 86-based system. The exceptions are:

- The modules E8087.LIB and E8087 provide support for the 8087 emulator. This emulator is not supported in an iRMX 86 multitasking environment.

- The module LARGE.LIB, which is described in the user's guide, is the system service (UDI) library for the Series III environment. To obtain system-service support for an iRMX 86 environment, link your programs to the iRMX 86 UDI library (URXLRG.LIB) instead of to LARGE.LIB. You do not need to provide any other special libraries.

The remainder of the libraries provide the same functions as listed in the FORTRAN-86 USER'S GUIDE. Therefore, when linking your FORTRAN-86 programs, include some or all of the following libraries (in the order shown here):

CEL.LIB	The floating-point intrinsic function library. You must link your program to this library if the program calls any floating-point functions.
F86RN0.LIB F86RN1.LIB F86RN2.LIB	Formatting and I/O libraries which are required for any run-time I/O support. If your programs do not perform I/O, you should link your program to RTNULL.LIB to resolve external references.
F86RN3.LIB F86RN4.LIB	Default logical record system libraries. If your programs perform I/O and run in an iRMX 86 environment, you should link the programs to these libraries.
8087.LIB	The 8087 Numeric Data Processor support library. If you require floating-point arithmetic, you should include the 8087 Numeric Data Processor in your hardware system and link your program to this library. If your program does not perform any floating-point arithmetic, you should link it to 87NULL.LIB to resolve external references.
URXLRG.LIB	The iRMX 86 UDI library. If you plan to run your program in an iRMX 86 environment, you should link it to this library.

For example, the following command links a FORTRAN-86 program (PROG/MULT3.OBJ), which uses floating-point arithmetic, to the libraries it needs for execution in an iRMX 86 environment.

```
-LINK86   :F1:PROG/MULT3.OBJ, &
**        :F1:CEL.LIB,        &
**        :F1:F86RN0.LIB,     &
**        :F1:F86RN1.LIB,     &
**        :F1:F86RN2.LIB,     &
**        :F1:F86RN3.LIB,     &
**        :F1:F86RN4.LIB,     &
**        :F1:8087.LIB,       &
**        :F1:URXLRG.LIB      &
**  TO    :F1:PROG/MULT3  BIND  MEMPOOL(+2000H)
iRMX 86 8086 LINKER,  V1.0
-
```

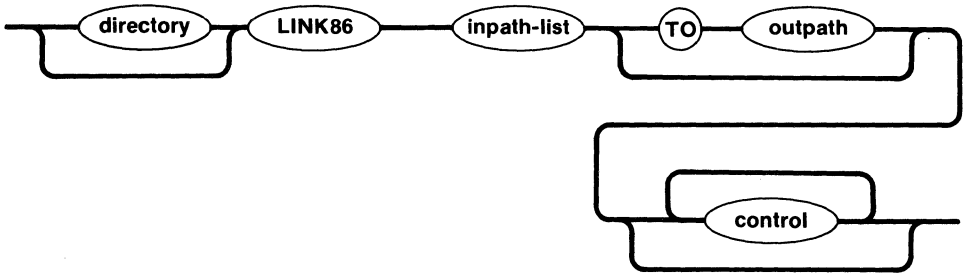
CHAPTER 6. LINK86

LINK86 combines 8086 object modules (produced by the assembler and compilers) and resolves references between independently translated modules. If the BIND and MEMPOOL controls are specified, the resulting module can be run in an iRMX 86 environment without using LOC86 to assign absolute addresses.

The iAPX 86,88 FAMILY UTILITIES USER'S GUIDE FOR 8086-BASED DEVELOPMENT SYSTEMS is the primary reference source for information on LINK86. You should refer to this manual for detailed descriptions of each LINK86 control. However, that manual is written specifically for users of Series III development systems. Therefore, some of the information in the manual does not accurately reflect how to use LINK86 in an iRMX 86 environment. Chapter 1 describes most of the differences between using LINK86 on a Series III and using it on an iRMX 86-based system. The following sections provide the additional information you need to operate LINK86 in an iRMX 86 environment. When the information in the following sections conflicts with the information in the family utilities manual, disregard the information in the family utilities manual.

INVOKING LINK86

To invoke LINK86, enter the following command at your Human Interface terminal:



where:

directory — Portion of the pathname that identifies the device and directories which contain LINK86. You can omit the device designation if the device corresponds to the default prefix for your system. Otherwise, enter the device's logical name, as specified in the last ATTACHDEVICE Human Interface command. Refer to the iRMX 86 HUMAN INTERFACE REFERENCE MANUAL for more information about file and device names.

inpath-list	Pathnames, separated by commas, of the files and libraries which are to be linked together. Refer to the iRMX 86 HUMAN INTERFACE REFERENCE MANUAL for more information about pathnames.
outpath	Pathname of the file to receive the linked output module. If you omit both the outpath parameter and the BIND control, LINK86 places the output module in a file that has the same pathname as the first element in inpath-list, but has an extension of LNK. If you omit the outpath parameter but include the BIND control, LINK86 places the output module in a file that has the same pathname as the first element in inpath-list, but has no extension.
control	Controls give LINK86 information that it needs to combine modules and generate output. The next section contains a list of LINK86 controls. You can enter any number of controls in a single invocation of LINK86. If you do not specify a particular control, LINK86 assumes the default for that control.

As with any Human Interface command, you can continue the LINK86 command on additional lines by entering the continuation character (&) after any parameter (as the last character in a line). However, the Human Interface restricts a command to contain no more than 255 characters, including punctuation, embedded blanks, continuation characters, non-executable comments, and carriage returns.

LINK86 CONTROLS

The iAPX 86,88 FAMILY UTILITIES USER'S GUIDE FOR 8086-BASED DEVELOPMENT SYSTEMS is the primary reference source for the LINK86 controls. However, Table 6-1 provides a summary of all controls. The following information applies to this table:

- Brackets ([]) denote optional parts of controls. The description lists the default condition if you omit the optional part.
- Controls preceded by the * character are default controls. Unless you specify otherwise, these controls are in effect.

Table 6-1. LINK86 Controls Summary

CONTROL	DESCRIPTION
BIND BI	Combines input modules into a load-time-locatable (LTL) module which can be loaded and executed in an iRMX 86 environment.
*NOBIND NOBI	Does not produce an LTL module.
*COMMENTS CM	Includes object file comment records in the output file.
NOCOMMENTS NOCM	Removes all but nonpurgable comment records from the output file.
*LINES LI	Includes line number information in the output file.
NOLINES NOLI	Removes line number information from the output file.
*MAP MA	Produces a link map and inserts it in the PRINT file.
NOMAP NOMA	Inhibits production of a link map.
MEMPOOL(minsize[,maxsize]) MP	Specifies the dynamic memory requirements of the program. If your programs run in an iRMX 86 environment, you should use this control to specify the amount of dynamic memory needed to create iRMX 86 objects.
NAME(module name) NA	Assigns the specified module name to the output module.
* Default Control	

Table 6-1. LINK86 Controls Summary (continued)

CONTROL	DESCRIPTION
OBJECTCONTROLS(controls) OC	Applies the specified controls to the object file only, instead of to both the object and print files. Valid controls include: LINES/NOLINES COMMENTS/NOCOMMENTS SYMBOLS/NOSYMBOLS PUBLICS[EXCEPT]/NOPUBLICS[EXCEPT] TYPE/NOTYPE PURGE/NOPURGE
ORDER(group name(segment name [class name overlay name] [,...]) OD	Specifies partial or complete order for the segments in one or more groups.
OVERLAY[(overlay name)] OV *NOOVERLAY NOOV	Combines all input modules into a single overlay module. Does not create an overlay module.
*PRINT[(pathname)] PR NOPRINT NOPR	Directs the link map and other diagnostic information to the specified file. If omitted, the listing file has the same pathname as the input file (but with an extension of MP1). Suppresses the map file.
* Default Control	

Table 6-1. LINK86 Controls Summary (continued)

CONTROL	DESCRIPTION
PRINTCONTROLS(controls) PC	Applies the specified controls to the print file only, instead of to both the object and print files. Valid controls include: LINES/NOLINES COMMENTS/NOCOMMENTS SYMBOLS/NOSYMBOLS PUBLICS[EXCEPT]/NOPUBLICS[EXCEPT] TYPE/NOTYPE PURGE/NOPURGE
*PUBLICS [EXCEPT(public symbol [,...])] PL [EC] NOPUBLICS [EXCEPT(public symbol [,...])] NOPL [EC]	Includes the public symbol records in the output file. If not specified, all public symbol records are included. Omits public symbol records from the output file.
PUBLICSONLY(pathname[,...]) PO	Includes only the absolute public symbol records of the argument files in the output file.
PURGE PU *NOPURGE NOPU	Removes all debug and public records from the output file. Includes all debug and public records in the output file.
RENAMEGROUPS(group name1 TO group name 2[,...]) RG	Changes the group names assigned by the translator.
SEGSIZE(segment name[class name overlay name](minsize [,maxsize]) [,...]) SS	Specifies the minimum memory space needed for any segment.
* Default Control	

Table 6-1. LINK86 Controls Summary (continued)

CONTROL	DESCRIPTION
*SYMBOLS SB NOSYMBOLS NOSB	Includes all local symbol records in the output file. Omits local symbol records from the output file.
*SYMBOLCOLUMNS(n) SC	Specifies the number of columns to be used when producing the symbol table for the object module. The default is SYMBOLCOLUMNS(2).
*TYPE TY NOTYPE NOTY	Performs type checking on the object file. Does not perform type checking.
* Default Controls	

ERROR MESSAGES

If LINK86 returns an error message whose number is in the range 1-4, it also returns an iRMX 86 exception code. To interpret this exception code, refer to the iRMX 86 EXTENDED I/O SYSTEM REFERENCE MANUAL.

USING OVERLAYS IN AN iRMX 86 ENVIRONMENT

If your assembly language or PL/M-86 programs use overlays and use UDI calls to load the overlays (the DQ$OVERLAY procedure), you should take care to ensure that you link the UDI library to your program correctly. The family utilities manual contains an example of linking an overlay program. This example lists a two-step link process, as follows:

1. Link the root and each of the overlays separately, specifying the OVERLAY control, but not the BIND control, in each LINK86 command.

2. Link all the output modules together in one module, specifying the BIND control, but not the OVERLAY control.

This is the same process that you should use when linking your iRMX 86 overlay programs. However, you must ensure that you link the entire UDI library to the root portion of the program and not to any of the overlays. To do this, use the INCLUDE control to include the UDI externals file (UDI.EXT) with the assembly or compilation of the root portion of the program. By including this file with the root, you make external references to all UDI routines from that root. Then when you link the root to the UDI library, LINK86 pulls in all of the UDI routines, not just the ones called in the root. Since you are linking the UDI library to the root only, this prevents you from having unsatisfied externals when you link the root to the overlays.

For example, suppose your program consists of three files, ROOT.OBJ, OV1A.OBJ, and OV2A.OBJ, the root and overlay files, respectively. You have compiled these program modules with the PL/M-86 compiler and included the UDI externals file UDI.EXT with the compilation of the root. Assuming that LINK86 resides on the default logical device in directory SYSTEM and that the object files reside on device :F1: in directory PROG, the following LINK86 commands will link the overlay program and produce an executable module. This happens in two steps.

1. The first three LINK86 commands separately link the root and overlay portions of the program. The root portion of the program is linked to the UDI library.

```
-LINK86 :F1:PROG/ROOT.OBJ,      &
        :F1:PROG/URXLRG.LIB OVERLAY
iRMX 86 8086 LINKER, V1.0

-LINK86 :F1:PROG/OV1A.OBJ OVERLAY(OVERLAY1)
iRMX 86 8086 LINKER, V1.0

-LINK86 :F1:PROG/OV2A.OBJ OVERLAY(OVERLAY2)
iRMX 86 8086 LINKER, V1.0
```

2. The next LINK86 command links together in one module all the output modules produced in the first step.

```
-LINK86 :F1:PROG/ROOT.LNK,      &
        :F1:PROG/OV1A.LNK,      &
        :F1:PROG/OV2A.LNK       &
     TO :F1:PROGRAM1  BIND  MEMPOOL(+2000H)
iRMX 86 8086 LINKER,  V1.0
-
```

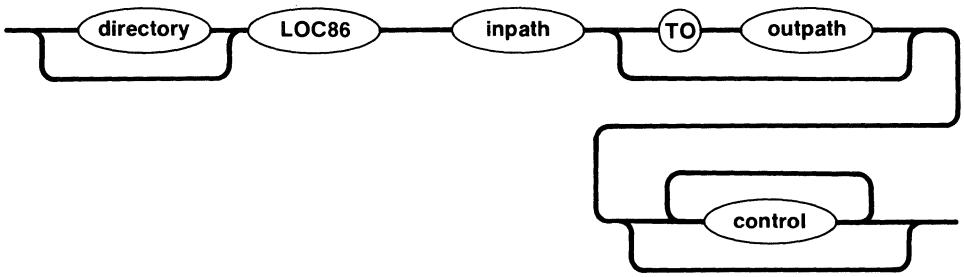
CHAPTER 7. LOC86

LOC86 changes relocatable object modules into absolute object modules. It takes a single object module as input and generates a print file and a located object file.

The iAPX 86,88 FAMILY UTILITIES USER'S GUIDE FOR 8086-BASED DEVELOPMENT SYSTEMS is the primary reference source for information on LOC86. You should refer to that manual for detailed descriptions of each LOC86 control. However, that manual is written specifically for users of Series III development systems. Therefore, some of the information in the manual does not accurately reflect how to use LOC86 in an iRMX 86 environment. Chapter 1 describes most of the differences between using LOC86 on a Series III and using it on an iRMX 86-based system. The following sections provide the additional information you need to operate LOC86 in an iRMX 86 environment. When the information in the following sections conflicts with the information in the family utilities manual, disregard the information in the family utilities manual.

INVOKING LOC86

To invoke LOC86, enter the following command at your Human Interface terminal:



where:

directory — Portion of the pathname that identifies the device and directories which contain LOC86. You can omit the device designation if the device corresponds to the default prefix for your system. Otherwise, enter the device's logical name, as specified in the last ATTACHDEVICE Human Interface command. Refer to the iRMX 86 HUMAN INTERFACE REFERENCE MANUAL for more information about file and device names.

inpath	Pathname of the file to be located. Refer to the iRMX 86 HUMAN INTERFACE REFERENCE MANUAL for more information about pathnames.
outpath	Pathname of the file to receive the located output module. If you omit the outpath parameter, LOC86 places the output module in a file that has the same pathname as the input file, but has no extension.
control	Controls give LOC86 information that it needs to assign addresses and generate output. The next section contains a list of LOC86 controls. You can enter any number of controls in a single invocation of LOC86. If you do not specify a particular control, LOC86 assumes the default for that control.

As with any Human Interface command, you can continue the LOC86 command on additional lines by entering the continuation character (&) after any parameter (as the last character in a line). However, the Human Interface restricts a command to contain no more than 255 characters, including punctuation, embedded blanks, continuation characters, non-executable comments, and carriage returns.

LOC86 CONTROLS

The iAPX 86,88 FAMILY UTILITIES USER'S GUIDE FOR 8086-BASED DEVELOPMENT SYSTEMS is the primary reference source for the LOC86 controls. However, Table 7-1 provides a summary of all controls. The following information applies to this table:

- Brackets ([]) denote optional parts of controls. The description lists the default condition if you omit the optional part.
- Controls preceded by the * character are default controls. Unless you specify otherwise, these controls are in effect.

Table 7-1. LOC86 Controls Summary

CONTROL	DESCRIPTION
ADDRESSES(SEGMENTS(segment [class[overlay]] (address)[,...]) CLASSES(class(address) [,...]), GROUPS(group(address) [,...])) AD(SM) AD(CS) AD(GR)	Overrides the LOC86 default address assignment algorithm and assigns absolute addresses.
BOOTSTRAP BS	Places the code for a long jump to the module's start address at location OFFFFOH.
*COMMENTS CM	Includes object file comment records in the output file.
NOCOMMENTS NOCM	Removes all but nonpurgable comment records from the output file.
INITCODE[(address)] IC	Includes code to initialize the segment registers.
*LINES LI	Includes line number information in the output file.
NOLINES NOLI	Removes line number information from the output file.
*MAP MA	Produces a link map and inserts it in the PRINT file.
NOMAP NOMA	Inhibits production of a link map.
NAME(module name) NA	Assigns the specified module name to the output module.
* Default Control	

Table 7-1. LOC86 Controls Summary (continued)

CONTROL	DESCRIPTION
OBJECTCONTROLS(controls) OC	Causes the specified controls to be applied to the object file only, instead of to both the object and print files. Valid controls include: LINES/NOLINES COMMENTS/NOCOMMENTS SYMBOLS/NOSYMBOLS PUBLICS/NOPUBLICS PURGE/NOPURGE
ORDER(SEGMENTS(segment[class [overlay]] [,...]), CLASSES(class[(segment [,...]) [,...])) OD	Specifies partial or complete order for the segments in one or more groups.
*PRINT[(pathname)] PR	Directs the locate map and other diagnostic information to the specified file. If omitted, the listing file has the same pathname as the input file (but with an extension of MP2).
NOPRINT NOPR	Suppresses the map file.
PRINTCONTROLS(controls) PC	Causes the specified controls to be applied to the print file only, instead of to both the object and print files. Valid controls include: LINES/NOLINES COMMENTS/NOCOMMENTS SYMBOLS/NOSYMBOLS PUBLICS/NOPUBLICS PURGE/NOPURGE
* Default Control	

Table 7-1. LOC86 Controls Summary (continued)

CONTROL	DESCRIPTION
*PUBLICS PL	Includes the public symbol records in the output file. If not specified, all public symbol records are included.
NOPUBLICS NOPL	Omits public symbol records from the output file.
PURGE PU	Removes all debug or public records from the output file.
*NOPURGE NOPU	Includes all debug or public records in the output file.
RESERVE(address1 TO address2 [,...] RS	Prevents LOC86 from locating segments in the specified areas of memory.
SEGSIZE(segment name[class name [overlay name]](size) SS	Specifies the memory space used by a segment.
START(public symbol) START(paragraph,offset) ST	Specifies the start address of the program.
*SYMBOLS SB	Includes all local symbol records in the output file.
NOSYMBOLS NOSB	Omits local symbol records from the output file.
*SYMBOLCOLUMNS(n) SC	Specifies the number of columns to be used when producing the symbol table for the object module. The default is SYMBOLCOLUMNS(2).
* Default Controls	

ERROR MESSAGES

If LOC86 returns an I/O error message (error number 1), it also returns an iRMX 86 exception code. To interpret this exception code, refer to the iRMX 86 EXTENDED I/O SYSTEM REFERENCE MANUAL.

EXAMPLE

The following command assigns absolute addresses to the module PROG/TEST3.LNK and places the located module on file PROG/TEST3. It orders the classes and assigns addresses starting with the CODE class. It also generates a map file on file PROG/TEST3.MP2.

```
-LOC86    PROG/TEST3.LNK  TO  PROG/TEST3                &
**        ORDER (CLASSES (CODE, DATA, STACK, MEMORY))   &
**        ADDRESSES (CLASSES (CODE (3C00H)))            &
**        MAP  PRINT (PROG/TEST3.MP2)
```

CHAPTER 8. LIB86

LIB86 allows you to create, modify, and examine library files. It is an interactive program which you enter by specifying a LIB86 invocation line. Then, you enter individual LIB86 commands to manipulate the library files.

The iAPX 86,88 FAMILY UTILITIES USER'S GUIDE FOR 8086-BASED DEVELOPMENT SYSTEMS is the primary reference source for information on LIB86. You should refer to that manual for detailed descriptions of the LIB86 commands. However, that manual is written specifically for users of Series III development systems. Therefore, some of the information in the manual does not accurately reflect how to use LIB86 in an iRMX 86 environment. Chapter 1 describes most of the differences between using LIB86 on a Series III and using it on an iRMX 86-based system. The following sections provide the additional information you need to operate LIB86 in an iRMX 86 environment. When the information in the following sections conflicts with the information in the family utilities manual, disregard the information in the family utilities manual.

INVOKING LIB86

To invoke LIB86, enter the following command at your Human Interface terminal:

where:

directory — Portion of the pathname that identifies the device and directories which contain LIB86. You can omit the device designation if the device corresponds to the default prefix for your system. Otherwise, enter the device's logical name, as specified in the last ATTACHDEVICE Human Interface command. Refer to the iRMX 86 HUMAN INTERFACE REFERENCE MANUAL for more information about file and device names.

comment	Any comment you wish to include on the invocation line. This comment is ignored by LIB86.

LIB86 COMMANDS

After you enter the invocation line, LIB86 responds by displaying an asterisk (*). You can then enter any of the LIB86 commands. The iAPX 86,88 FAMILY UTILITIES USER'S GUIDE FOR 8086-BASED DEVELOPMENT SYSTEMS is the primary reference source for the LIB86 commands. However, Table 8-1 provides a summary of all commands. The following information applies to this table:

- Brackets ([]) denote optional parts of commands. The description lists the default condition if you omit the optional part.

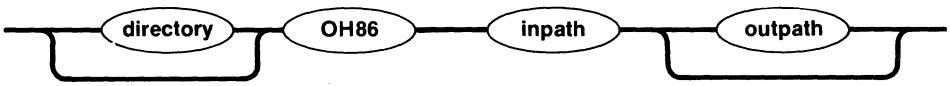
Table 8-1. LIB86 Command Summary

COMMAND	DESCRIPTION
ADD inpath[(module [,...]) [,...] TO libpath A	Adds modules from the files specified by the inpath parameters to the library specified by the libpath parameter.
CREATE pathname C	Creates the specified file as a library.
DELETE pathname(module [,...]) D	Deletes modules from the specified library file.
EXIT E	Terminates the LIB86 session and returns control to the Human Interface.
LIST pathname(module [,...]) [,...] [TO pathname] [PUBLICS] L [P]	Lists modules contained in the specified file and optionally lists all publics.

CHAPTER 9. OH86

OH86 converts 8086 absolute object modules to 8086 hexadecimal format. The iAPX 86,88 FAMILY UTILITIES USER'S GUIDE FOR 8086-BASED DEVELOPMENT SYSTEMS is the primary reference source for information on OH86. However, that manual is written specifically for users of Series III development systems. Therefore, some of the information does not accurately reflect how to use OH86 in an iRMX 86 environment. Chapter 1 describes most of the differences between using OH86 on a Series III and using it on an iRMX 86-based system. The following paragraphs provide the additional information you need to operate OH86 in an iRMX 86 environment. When the information in the following sections conflicts with the information in the family utilities manual, disregard the information in the family utilities manual.

To invoke OH86, enter the following command at your Human Interface terminal:



where:

directory — Portion of the pathname that identifies the device and directories which contain OH86. You can omit the device designation if the device corresponds to the default prefix for your system. Otherwise, enter the device's logical name, as specified in the last ATTACHDEVICE Human Interface command. Refer to the iRMX 86 HUMAN INTERFACE REFERENCE MANUAL for more information about file and device names.

inpath — Pathname of the file which contains an 8086 absolute object module. Refer to the iRMX 86 HUMAN INTERFACE REFERENCE MANUAL for more information about pathnames.

outpath — Pathname of the file to receive the 8086 hexadecimal format module. If you omit the outpath parameter, OH86 places the output module in a file that has the same pathname as the input file, but has the extension HEX.

APPENDIX A. MEMORY REQUIREMENTS

Table A-1 lists the memory requirements for the iRMX 86 language products. This table assumes that you store your language products on secondary storage devices and that you load and run them with the Human Interface. The Total column indicates the minimum amount of free space (RAM not reserved for the operating system or other programs) that your iRMX 86 system must contain when running the language products. The other columns divide this minimum memory into code, data, and dynamic memory.

Table A-1. Memory Requirements

LANGUAGE PRODUCT	CODE	STATIC DATA	DYNAMIC MEMORY	TOTAL
ASM86 V1.0	31.5K	64K	19.1K	114.6K
PLM86 V1.0	30.1K	64K	17.9K	112K
LINK86 V1.0	28K	64K	18.6K	110.6K
LOC86 V1.0	32K	64K	12K	108K
LIB86 V1.0	12.3K	64K	12K	88.3K
OH86 V1.0	7.2K	64K	12K	83.2K
EDIT V1.0	12K	64K	4.9K	80.9K

INDEX

REQUEST FOR READER'S COMMENTS

Intel Corporation attempts to provide documents that meet the needs of all Intel product users. This form lets you participate directly in the documentation process.

Please restrict your comments to the usability, accuracy, readability, organization, and completeness of this document.

1. Please specify by page any errors you found in this manual.

2. Does the document cover the information you expected or required? Please make suggestions for improvement.

3. Is this the right type of document for your needs? Is it at the right level? What other types of documents are needed?

4. Did you have any difficulty understanding descriptions or wording? Where?

5. Please rate this document on a scale of 1 to 10 with 10 being the best rating. _____

NAME _____ DATE _____

TITLE _____

COMPANY NAME/DEPARTMENT _____

ADDRESS _____

CITY _____ STATE _____ ZIP CODE _____

Please check here if you require a written reply. ☐

WE'D LIKE YOUR COMMENTS . . .

This document is one of a series describing Intel products. Your comments on the back of this form will help us produce better manuals. Each reply will be carefully reviewed by the responsible person. All comments and suggestions become the property of Intel Corporation.

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 79 BEAVERTON, OR

POSTAGE WILL BE PAID BY ADDRESSEE

Intel Corporation
5200 N.E. Elam Young Pkwy.
Hillsboro, Oregon 97123

O.M.S. Technical Publications

INTEL CORPORATION, 3065 Bowers Avenue, Santa Clara, California 95051 (408) **987-8080**

Printed in U.S.A.