READ THIS FIRST

INTELLEC® SERIES III MICROCOMPUTER DEVELOPMENT SYSTEM PRODUCT OVERVIEW

Order Number: 121575-003

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

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 to identify Intel products:

BXP	int$_e$l	iSBC	MULTICHANNEL
CREDIT	Intelevision	iSBX	MULTIMODULE
i	int$_e$ligent Identifier	iSXM	Plug-A-Bubble
I^{2}ICE	int$_e$ligent Programming	Library Manager	PROMPT
ICE	Intellec	MCS	RMX/80
iCS	Intellink	Megachassis	RUPI
i$_m$	iOSP	MICROMAINFRAME	System 2000
iMMX	iPDS	MULTIBUS	UPI
Insite	iRMX		

REV.	REVISION HISTORY	DATE
-001	Original issue.	9/80
-002	Adds information to support CREF86 and V4.2 of ISIS-II; updates references to related publications and filenames.	11/81
-003	Adds information to support Resident Processor Card configuration, updates references to related publications, and describes new product packaging.	9/82

INTRODUCTION

When you receive your new Intellec Series III Microcomputer Development System or Model 557 Resident Processor Card package, you should read this manual first. It provides the following information:

- An overview of the Intellec Series III development solution, including basic mainframe hardware, fundamental software, and optional hardware and software packages.
- A list of the items you will find in the Model 225 system and the Model 557 Resident Processor Card package, so that you may check for all these items when you unpack the products.
- A description of the library of technical manuals provided for the Series III. This library includes all the manuals supplied with the basic system, plus the additional manuals provided with the optional packages.
- A glossary of terms used in the manuals of the Series III publications library.

This manual is primarily a guide to the Series III publications library. After reading this manual, you should be able to pick out the particular technical manual containing a given item of information, or instructions for performing a given task. The manual is also a useful reference to the vocabulary of terms used by other manuals within the publications library.

CONTENTS

ILLUSTRATIONS

CHAPTER 1
THE INTELLEC® SERIES III DEVELOPMENT SOLUTION

The Intellec Series III Microcomputer Development System consists of an iMDX 225 mainframe box with an iMDX 557 Resident Process Card installed. The iMDX 225 (Series II) and the iMDX 557 (16-bit upgrade package) are each shipped with the appropriate software and technical manuals.

The package containing this manual also contains the RPC-86 Resident Processor Card, fundamental software, and technical manuals for the Series III. If you have purchased a new Series III system, you will also receive, in a separate carton, an Intellec Model 225 Development System mainframe in which to install the resident processor card. If you are upgrading your present Intellec system and you have performed all the necessary upgrade preparation on your present system, this package contains all the additional hardware, software, and manuals you need.

The Development Solution

The Intellec Series III, along with additional optional hardware and software packages available from Intel, is a complete development solution for your microcomputer applications based on Intel microprocessors. The Series III is designed especially to aid in developing iAPX 86, 88 applications—that is, applications based on the 8086, 8088, 8087, 8089, 186, and 286 microprocessors. The Series III also supports processors of the MCS®-85, MCS®-80, MCS®-51, and MCS®-48 Families, and the 2920 Signal Processor.

Figure 1-1 outlines the process of developing a microcomputer-based design. The diagram shows how Intel development system tools aid you at all stages of the development process. Note the importance of top-down, modular design, which enables your team to develop hardware and software modules that work together. With the aid of Intel high-level programming languages and ICE™ In-Circuit Emulators, you can test, debug, and integrate your system in stages, adding individual hardware and software modules as you complete them.

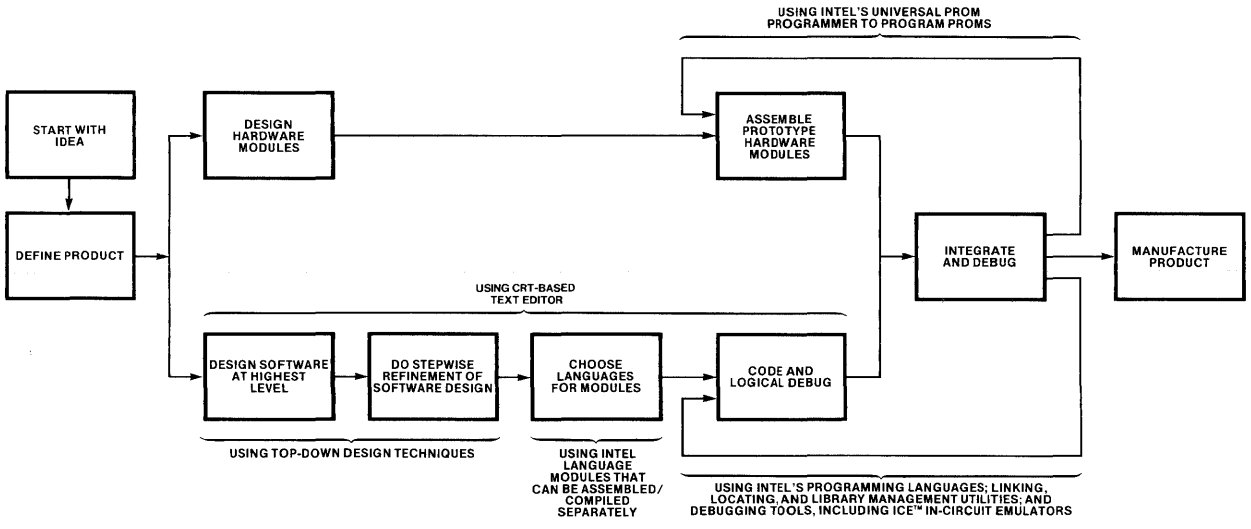


Figure 1-1. Microcomputer System Development Process

121575-1

Overview of Series III System

Mainframe Hardware

The Series III mainframe includes two host CPU's—an 8086 and an 8085A—to provide enhanced performance and two native execution environments. Thus the Series III is both an 8086-based development system and an 8085-based development system. The system includes 224K bytes of iAPX 86, 88 user memory, a 2000-character CRT, detachable full ASCII keyboard with cursor controls and upper/lower-case capability, and a 250K-byte integral single-density floppy disk drive. Built-in interfaces are provided for two serial I/O channels, a high-speed paper tape reader/punch, a line printer or teletype, and the Intel Universal PROM Programmer. The Series III operating system extends the user interface of previous Intellec development systems, is software-compatible with them, and has a superset of their capabilities, thus providing an easy upgrade path for users.

Figure 1-2 shows the Series III system as it appears when installed. Figure 1-3 shows a system with an external dual-drive flexible disk subsystem; figure 1-4 shows a system with a Winchester disk subsystem.

Supplied with the system is fundamental software including the Series III operating system, 8086/87/88/186 Macro Assembler and other program development software, plus a publications library containing all the technical manuals you need to use the Series III hardware and fundamental software. The system software is listed in the following section; the manuals are described in Chapter 2.

Figure 1-2. Series III System 121575-2

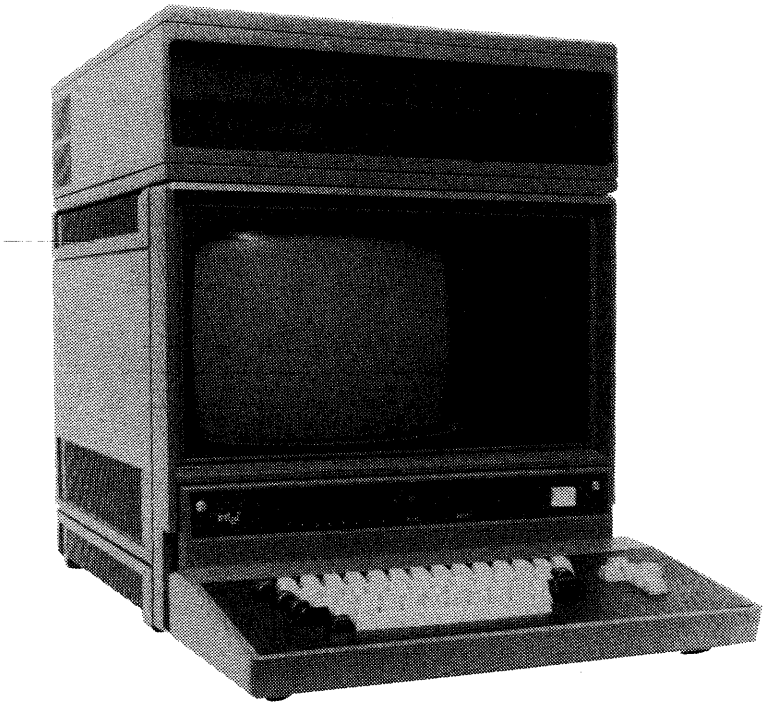


Figure 1-3. System with External Flexible Disk Drive Subsystem 121575-3

Figure 1-4. System with Winchester Disk Subsystem 121575-4

System Firmware and Software

The following firmware and software are supplied with the Series III system:

iMDX 225

The iMDX 225 provides

- ROM-resident 8085A-based Monitor providing low-level system services and debugging for MCS-80/85 programs
- ISIS-II operating system software
- CREDIT™ CRT-based Text Editor
- Utility programs for MCS-80/85 applications (LINK, LOCATE, LIB, OBJHEX, HEXOBJ)
- 8080/8085 Floating-Point Arithmetic Library (FPAL)

IMDX 557

The iMDX 557 provides

- ROM-resident 8086-based software applications debugger (DEBUG-86) for iAPX and iAPX 88 programs
- Intellec Series III confidence test software
- RUN operating system extension for program execution in the 8086 environment
- iAPX 86-resident 8086/87/88/186 Macro Assembler (ASM86)
- iAPX 86-resident software development utilities for iAPX 86, 88 applications (LINK86, LOC86, LIB86, OH86, CREF86)
- Run-time support and interface libraries for iAPX 86, 88 applications, including operating system interfaces, full support for the 8087 Numeric Data Processor, and the full 8087 Emulator software
- MCS-86 Assembly Language Converter (CONV86) for converting programs from 8080/8085 Macro Assembly Language to 8086/87/88/186 Macro Assembly Language
- ALTER 8086-based, menu-driven, full screen text editor

Optional Packages

The Series III system and its software and manuals provide the foundation for microcomputer system development using the Series III. To help you build on this foundation, Intel provides additional hardware and software packages from which you can choose according to your individual needs.

Since the Series III is upward compatible with the Series II, all Intel hardware and software available for the Series II will also run on a Series III.

Add-On Disk Storage

For many applications, additional disk storage may be desirable. You may choose to add one or more external double-density flexible disk drive subsystems, each of which contains two drives (0.5 megabyte total storage capacity per disk; 2.5 megabytes maximum in one system). Or you may choose a Winchester disk subsystem, with a formatted capacity of 22 megabytes. The Winchester disk subsystem is the recommended configuration for iAPX 86 and iAPX 88 microsystem development.

Expansion Chassis

For users installing additional controller, memory, or in-circuit emulator boards, the Model 201 expansion chassis provides four additional card slots and supporting power supply, fans, and cable assemblies.

High-Level Languages for iAPX 86 and iAPX 88 Software Development

You can greatly reduce your system design and maintenance time by using a high-level language for software development. The high-level language compilers produce code for the 8086 and 8088 processors, and all run in the 8086 development environment for higher performance. The PL/M-86 compiler also produces code for the 186 processor.

For systems programming, PL/M-86 is recommended. (If you are producing code for the 286 processor, PL/M-286 is recommended.) This language, especially designed by Intel for microprocessors, combines a high-level block structure with hardware-level features, such as memory address and bit manipulation.

For applications programming, Intel provides Pascal-86 and FORTRAN-86. Pascal-86 is a standard Pascal with extensions for microprocessor port I/O, interrupt processing, and compilation of separate modules. FORTRAN-86, a superset of ANSI standard FORTRAN with features tailored to microprocessor software development, is well-suited for applications that require complex mathematical expressions.

All Intel iAPX 86, 88 compilers and assemblers produce compatible object code, so that modules in assembly language and modules in high-level languages can be linked together.

Macro Assembler for the 8089 I/O Processor

For users designing iAPX 86 or iAPX 88 applications using the 8089 I/O Processor, the 8089 Macro Assembler is available. This assembler produces object modules that can be linked to modules written in 8086/87/88/186 Macro Assembly Language or in any of the high-level languages for iAPX 86, 88 systems. The 8089 Macro Assembler runs on a Series III in 8085 execution mode.

In-Circuit Emulators for iAPX 86 and iAPX 88 Applications

In-circuit emulator (ICE) modules enable you to develop and debug your microcomputer system hardware and software together, saving considerable development cost and time. Using these tools, you can add individual hardware and software modules to your system as you complete them, substituting in-circuit emulator resources for the modules not yet prototyped.

For iAPX 86 microsystems, use the ICE-86A Emulator; for iAPX 88 microsystems, use the ICE-88A Emulator. If your iAPX 86 system includes an 8089 I/O Processor, use the RBF-89 Real-Time Breakpoint Facility provided with the ICE-86A Emulator.

Software and In-Circuit Emulators for MCS®-80 and MCS®-85 Applications

Intel provides full support for MCS-80 and MCS-85 microcomputer system development on the Series III. The Series III's 8085 execution environment allows you to run and debug your programs on the development system, and also to use all the Intel

development aids available for MCS-80/85 microsystems. These development aids include the 8080/8085 Macro Assembler (ASM80), the PL/M-80 and FORTRAN-80 Compilers, the BASIC-80 Interpreter, the Pascal-80 and iCIS-COBOL Compiler/Run-Time Systems, the 8080/8085 Fundamental Support Package applications routines, and the ICE-80 and ICE-85 In-Circuit Emulators.

Software and In-Circuit Emulators for MCS®-51 and MCS®-48 Family Applications

You can also develop MCS-51 and MCS-48 Family microcomputer applications on the Series III. Development tools for these applications include the MCS-48 and UPI-41 Macro Assembler, the MCS-51 Macro Assembler, the MCS-51 Linker and Locator, and the ICE-49, ICE-41A, and ICE-51 In-Circuit Emulators.

Software for 2920 Signal Processor Applications

The development tools available for applications using the 2920 Signal Processor run on the Series III. These tools include the 2920 Assembler (AS2920), the 2920 Simulator (SM2920), and the 2920 Signal Processing Applications Software Compiler (SPAS20).

Universal PROM Programmer

The iUP 200/201 Universal PROM Programmer is provided to program and verify Intel programmable ROMs (PROMs). Programming and verification are initiated from the Series III system console using the iPPS software.

Mainframe Link for Distributed Development

The Series III supports the Mainframe Link hardware/software package to allow transfer of data between an Intellec development system and most large mainframes and minicomputers. The Mainframe Link allows integration of Intellec system and user mainframe system resources.

Future Products

The list of add-on products just given is by no means complete. Additional systems products will continue to become available to support Intel microprocessors and microcomputer systems.

Contents of the iMDX 225 and iMDX 557

The following sections identify the individual items included in each product. When you unpack, check to be sure all the items are there. If any item is missing, contact your Intel representative.

Before installing your system, read Chapter 2. This chapter gives a complete description of the library of manuals for the Series III, some of which you will use during installation.

iMDX 225

The iMDX 225 consists of the following:

- Model 225 chassis
- Detachable keyboard
- Software diskettes and manuals

iMDX 557

The iMDX 557 consists of the following:

- RPC-86 Resident Processor Card with RAM multimodule board
- Software diskettes and manuals

Software Diskettes

All software is supplied in both single density and double density to ensure that you have the diskette format that fits your environment.

iMDX 225

The iMDX 225 contains the following diskettes:

- One single-density and one double-density diskette labeled "ISIS-II Operating System"
- One single-density and one double-density diskette labeled "CREDIT ISIS-II CRT-Based Editor"
- One single-density and one double-density diskette labeled "ASM-80"
- One single-density and one double-density diskette labeled "Diagnostic Confidence Test"

iMDX 557

- One double-density diskette labeled "Series III Diskette"
- One single-density diskette labeled "Resident 8086/87/88/186 Macro Assembler"
- One single-density diskette labeled "Resident 8086/8088 Utilities and Numerics Libraries"
- One single-density and one double-density diskette labeled "ALTER, 8086-Based Editor"
- One single-density and one double-density diskette labeled "Series III Diagnostic Confidence Test"

NOTE

The double-density diskette labeled "Series III Diskette" contains the same files that appear on the two single-density diskettes labeled "Resident 8086/87/88/186 Macro Assembler" and "Resident 8086/8088 Utilities and Numerics Libraries."

NOTE

Many of the diskette files listed below have the Invisible attribute, so if you make directory listings to check the contents of the diskettes, you should use the ISIS-II DIR command with the Invisible (I) switch set. (See the *Intellec Series III Microcomputer Development System Console Operating Instructions.*)

The diskettes labeled "ISIS-II Operating System" contain the following files:

ISIS.DIR	DIR	LINK
ISIS.MAP	FIXMAP	LINK.OVL
ISIS.T0	FORMAT	LOCATE
ISIS.LAB	HDCOPY	LIB
ISIS.BIN	IDISK	HEXOBJ
ISIS.CLI	RENAME	OBJHEX
ISIS.OV0	SUBMIT	FPAL.LIB
ATTRIB	EDIT	PLM80.LIB
COPY	SYSTEM.LIB	VERS
DELETE		

ISIS.DIR, ISIS.MAP, ISIS.T0, and ISIS.LAB are the basic disk format files recognized by the ISIS-II software, which is the foundation for the Series III operating system. Files with these names appear on every disk used under ISIS-II. ISIS.BIN and ISIS.CLI are the basic ISIS-II operating system files. ATTRIB, COPY, DELETE, DIR, FIXMAP, FORMAT, HDCOPY, IDISK, RENAME, VERS, and SUBMIT are ISIS-II system command files. EDIT is the Intellec 800 and Series II line-oriented text editor; ALTER is the recommended editor for use on the Series III.

SYSTEM.LIB is a library containing ISIS-II system routines you can call from your MCS-80/85 programs. LINK, LINK.OVL, LOCATE, LIB, HEXOBJ, and OBJHEX are the MCS-80/85 utilities used for MCS-80 and MCS-85 software development; these programs are needed by users of 8080/8085 Macro Assembly Language, PL/M-80, and FORTRAN-80. FPAL.LIB is the 8080/8085 Floating-Point Arithmetic Library, a run-time library of arithmetic routines you can call from your MCS-80/85 programs. PLM80.LIB is a library provided to support the routines in SYSTEM.LIB and FPAL.LIB.

The ISIS-II operating system software, ISIS-II system commands, and MCS-80/85 Family utilities programs run in the Series III's 8085 execution mode.

NOTE

On a Series III system, you must use version 4.2 or later of the ISIS-II software. For hard disk systems, ISIS-II version 4.2 or later must be present in drive 0 *and* in drive 4.

The diskettes labeled "CREDIT ISIS-II CRT-Based Text Editor" contain the following files:

ISIS.DIR	CREDIT	VT100.MAC
ISIS.MAP	CREDIT.HLP	1510T.MAC
ISIS.T0	ADDS.MAC	1510E.MAC
ISIS.LAB	MICROB.MAC	LEAR.MAC
	VT52.MAC	

ISIS.DIR, ISIS.MAP, ISIS.T0, and ISIS.LAB are the basic format files that appear on every disk. The remaining files are all part of the CREDIT CRT-Based Text Editor software.

The diskettes labeled "ALTER Text Editor" contain the following files:

ISIS.DIR	ALTER.86	VT100.MAC
ISIS.MAP	ADDS.MAC	1510T.MAC
ISIS.T0	MICROB.MAC	1510E.MAC
ISIS.LAB	VT52.MAC	LEAR.MAC

ISIS.DIR,ISIS.MAP,ISIS.T0, and ISIS.LAB are the basic format files that appear on every disk. The remaining files are all part of the ALTER Text Editor software.

The single-density diskette labeled "Resident 8086/87/88/186 Macro Assembler" contains the following files:

ISIS.DIR	RUN.OV1
ISIS.MAP	SMALL.LIB
ISIS.T0	COMPAC.LIB
ISIS.LAB	LARGE.LIB
RUN	ASM86.86
RUN.OV0	

ISIS.DIR, ISIS.MAP, ISIS.T0, and ISIS.LAB are the basic disk format files. RUN, RUN.OV0, and RUN.OV1 are the files for the RUN portion of the operating system, which controls execution in the 8086 environment of the Series III. SMALL.LIB, COMPAC.LIB, and LARGE.LIB are operating system interface libraries. ASM86.86 is the resident 8086/87/88/186 Macro Assembler.

The single-density diskette labeled "Resident 8086/8088 Utilities and Numerics Libraries" contains the following files:

ISIS.DIR	LOC86.86	E8087.LIB
ISIS.MAP	LIB86.86	8087.LIB
ISIS.T0	CREF86.86	CONV86
ISIS.LAB	OH86.86	
LINK86.86	E8087	

ISIS.DIR, ISIS.MAP, ISIS.T0, and ISIS.LAB are the basic disk format files. LINK86.86, LOC86.86, LIB86.86, CREF86.86, and OH86 are the resident iAPX 86, 88 Family utilities for software development; these programs are needed by users of the 8086/87/88/186 Macro Assembler, PL/M-86, Pascal-86, and FORTRAN-86. E8087 is the 8087 Emulator, provided for run-time numerics support in systems that do not contain an 8087 Numeric Data Processor. E8087.LIB and 8087.LIB are interface libraries for the 8087 Emulator and the 8087 processor. CONV86 is the MCS-86 Assembly Language Converter.

CONV86 runs on the Series III in 8085 execution mode. The 8086/87/88/186 Macro Assembler and the iAPX 86, 88 Family utilities run in 8086 execution mode. The RUN command is invoked from the 8085 execution mode to transfer control into the 8086 mode.

The double-density diskette labeled "Series III Diskette" contains the files listed for both the single-density diskettes just described—that is, the diskettes labeled "Resident 8086/87/88/186 Macro Assembler" and "Resident 8086/8088 Utilities and Numerics Libraries."

The diskettes labeled "Series III Diagnostic Confidence Test" contain the following files:

ISIS.DIR	DELETE	CONFID.OV0
ISIS.MAP	DIR	CONFID.OV1
ISIS.T0	RUN	CONFID.OV2

ISIS.LAB	RUN.OV0	CONFID.OV3
ISIS.BIN	RUN.OV1	CONFID.OV4
ISIS.CLI	CONF	CONFID.OV5
ATTRIB	CONF.OVE	ECON86.86
COPY	CONFID	

These files contain all the software you need to check out your system after installation, whether you are installing a new Series III or upgrading your existing system. Some of the files on this diskette are duplicates of files on other diskettes. ISIS.DIR, ISIS.MAP, ISIS.T0, and ISIS.LAB are the basic disk format files. ISIS.BIN, ISIS.CLI, ATTRIB, COPY, DELETE, and DIR are identical to the ISIS-II operating system files and system command files with those names on the "ISIS-II Operating System" diskette. RUN, RUN.OV0, and RUN.OV1 are the same as the files by those names on the single-density "Resident 8086/87/88/186 Macro Assembler" diskette and the double density "Series III Diskette." The remaining files are confidence test programs for checking out the Intellec Mainframe and the RPC-86 Resident Processor Card.

Technical Manuals

The iMDX 225 system and the iMDX 557 upgrade package are each shipped with the appropriate technical manuals.

iMDX 225

The iMDX 225 Literature Kit consists of the following manuals:

- *A Guide to Intellec Microcomputer Development Systems*, 9800558
- *ISIS-II User's Guide*, 9800306
- *ISIS-II Pocket Reference*, 9800841
- *MCS-80/85 Family User's Manual*, 121506
- *Intellec Series Model 22X/23X Installation Manual*, 9800559
- *Series II Hardware Reference Manual*, 9800556
- *Series II Hardware Interface Manual*, 9800555
- *8080/8085 Assembly Language Programming Manual*, 9800301
- *MCS-85 Assembly Language Pocket Reference*, 9800438
- *ISIS-II 8080/8085 Macro Assembly User's Manual*, 9088292
- *8080/8085 Floating Point Arithmetic Library User's Manual*, 9088452
- *ISIS Command Line Interpreter Pocket Supplement*, 122011
- *Series II/III CRT and Keyboard Interface Manual*, 122029
- *Intellec Series II Monitor Listing*, 9800605
- *ISIS-II CREDIT CRT-Based Text Editor User's Guide*, 9800902
- *ISIS-II CREDIT CRT-Based Text Editor Pocket Reference*, 9800903
- *Intellec Double Density DOS Hardware Reference Manual*, 9800422

iMDX 557

The iMDX 557 Literature Kit consists of the following manuals:

- *iAPX 86,88 User's Manual*, 210201
- *The iAPX 88 Book*, 210200
- *A Guide to Intellec Series III Microcomputer Development Systems*, 121632

- *Intellec Series III Microcomputer Development System Product Overview*, 121575
- *Model 557 Resident Processor Card Installation and Checkout Manual*, 122015
- *Intellec Series III Microcomputer Development System Console Operating Instructions*, 121609
- *Intellec Series III Microcomputer Development System Pocket Reference*, 121610
- *Intellec Series III Microcomputer Development System Programmer's Reference Manual*, 121618
- *ALTER Text Editor User's Guide*, 121956
- *ALTER Text Editor Pocket Reference*, 121957
- *MCS-80/85 Utilities User's Guide for 8080/8085-Based Development Systems*, 121617
- *iAPX 86,88 Family Utilities User's Guide*, 121616
- *iAPX 86,88 Family Utilities Pocket Reference*, 121669
- *An Introduction to ASM86*, 121689
- *ASM86 Language Reference Manual*, 121703
- *ASM86 Macro Assembler Operating Instructions for 8086-Based Development Systems*, 121628
- *ASM86 Macro Assembler Pocket Reference*, 121674
- *MCS-86 Assembly Language Converter Operating Instructions for ISIS-II Users*, 9800642

CHAPTER 2
THE SERIES III PUBLICATIONS LIBRARY

The Intellec Series III publications library is an integrated set of technical manuals to support the Series III development system and its associated options. The fundamental set of manuals is provided in the Model 557 package. When you buy optional hardware or software packages, you receive additional manuals that fit into the overall publications library scheme.

This chapter will familiarize you with the structure of the publications library, so that when you need to look up a given item of information or to find instructions for performing a given task using your Series III system, you will know which manual to consult.

Copies of the technical manuals described here are shipped with the Intel products they support—the Series III or optional packages. Additional copies of all manuals may be obtained by contacting your Intel representative or the Literature Department at Intel. The Literature Department also distributes other Intel literature, such as application notes, magazine articles describing Intel products, and brochures on new products.

Figure 2-1 is a tree diagram showing the manuals in the Series III publications library and their relationship to each other. The manuals outlined with a solid line are supplied with the Model 557 package. The manuals outlined with dotted lines are provided with optional packages. Only a few of these additional manuals are shown here; if all of them were shown, the diagram would cover several pages. Refer to the current *Literature Guide*, for an up-to-date list of available publications.

Each manual in the library corresponds to a different user activity. Each contains the information needed for performing that activity and is written at the level of a typical user performing the activity. For instance, the *Intellec Series III Microcomputer Development System Console Operating Instructions* is written for a new Series III user who may not be familiar with Intel systems or the ISIS-II operating system. In contrast, the *ASM86 Macro Assembler Operating Instructions for 8086-Based Development Systems* is written for a user who is assembling programs coded in 8086/87/88/186 Macro Assembly Language. The level of this manual assumes that the user has a basic knowledge of assembly language programming, and that he is already familiar with the iAPX 86, 88 microsystem family, the Series III operating system, and the 8086/87/88/186 Macro Assembly Language.

In figure 2-1, the manuals are grouped according to activity categories. The following sections describe these categories and the individual manuals supporting each kind of activity.

An Introduction to the Series III

The 7" × 9" manual entitled *A Guide to Intellec Series III Microcomputer Development Systems* introduces the Series III software, especially the software related to the 8086 execution environment. By using an example of a typical micro-application development process, this publication serves as a tutorial on general software design, the Series III operating system, CREDIT, Pascal-86, PL/M-86, DEBUG-86, and ICE-88.

Microsystem Design Using the iAPX 86, 88 Family of Microprocessors

Two of the 7" × 9" manuals provided with the basic Series III—the *iAPX 86, 88 User's Manual* and the *iAPX 88 Book*—give a complete description of the iAPX 86, 88 Family of microprocessors and how they are configured into systems. These are the manuals to consult if you are evaluating the iAPX 86, 88 Family to determine their suitability for your design. Later, once you have chosen the iAPX 86, 88 Family, these manuals will aid you in doing your hardware design.

These manuals assume a basic knowledge of microprocessor hardware. They describe the architecture, instruction sets, and capabilities of the 8086, 8087, 8088, and 8089 processors; define the configuration of iAPX 86 and iAPX 88 microcomputer systems; give hardware reference information and device specifications; outline the available software and systems support; and provide programming examples and application notes. The *iAPX 86, 88 User's Manual* covers the 8086 and 8088 CPU's, the 8089 I/O Processor, and the 8087 Numeric Data Processor. The *iAPX 88 Book* provides additional details and examples for the 8088 CPU, including benchmarks, data sheets, and considerations related to multiple and coprocessor environments.

Hardware Installation and Checkout

When you are ready to install your new Series III upgrade kit, consult the *Model 557 Resident Processor Card Kit Installation Manual.*

If you need to do hardware troubleshooting, consult the *iMDX 557 Upgrade Kit Schematic Drawings.* This manual consists of schematic drawings with no accompanying instructions and assumes you have detailed hardware knowledge.

If you have purchased a Winchester disk subsystem, follow the instructions in the *Winchester Peripheral Chassis Insstallation and Checkout Manual* for information on installing the subsystem. To operate your hard disk after installation, follow the instructions in the *Winchester Peripheral Chassis ISIS-II (W) Supplement.*

Installation and checkout information for add-on hardware products other than the hard disk are covered in the technical manuals supplied with these products.

Systems Operation and Systems Programming

Once your system has been installed and you are ready to operate it, you should first consult the *Intellec Series III Microcomputer Development System Console Operating Instructions.* This manual tells you how to start up the system, use the flexible disk and hard disk units, and give commands from the console to the operating system, the 8085-resident monitor, and the DEBUG-86 applications debugger. In other words, it describes the system environment that a human operator sees. You need to become familiar with the information in this manual before you use any of the manuals described in the following paragraphs and sections.

An important feature of the Intellec Series III system is the ALTER Text Editor, which executes in 8086 mode and can be invoked from the console. ALTER is a full-screen, menu driven text editor that provides easy command entry. To use the ALTER Text Editor, refer to the *ALTER Text Editor User's Guide.*

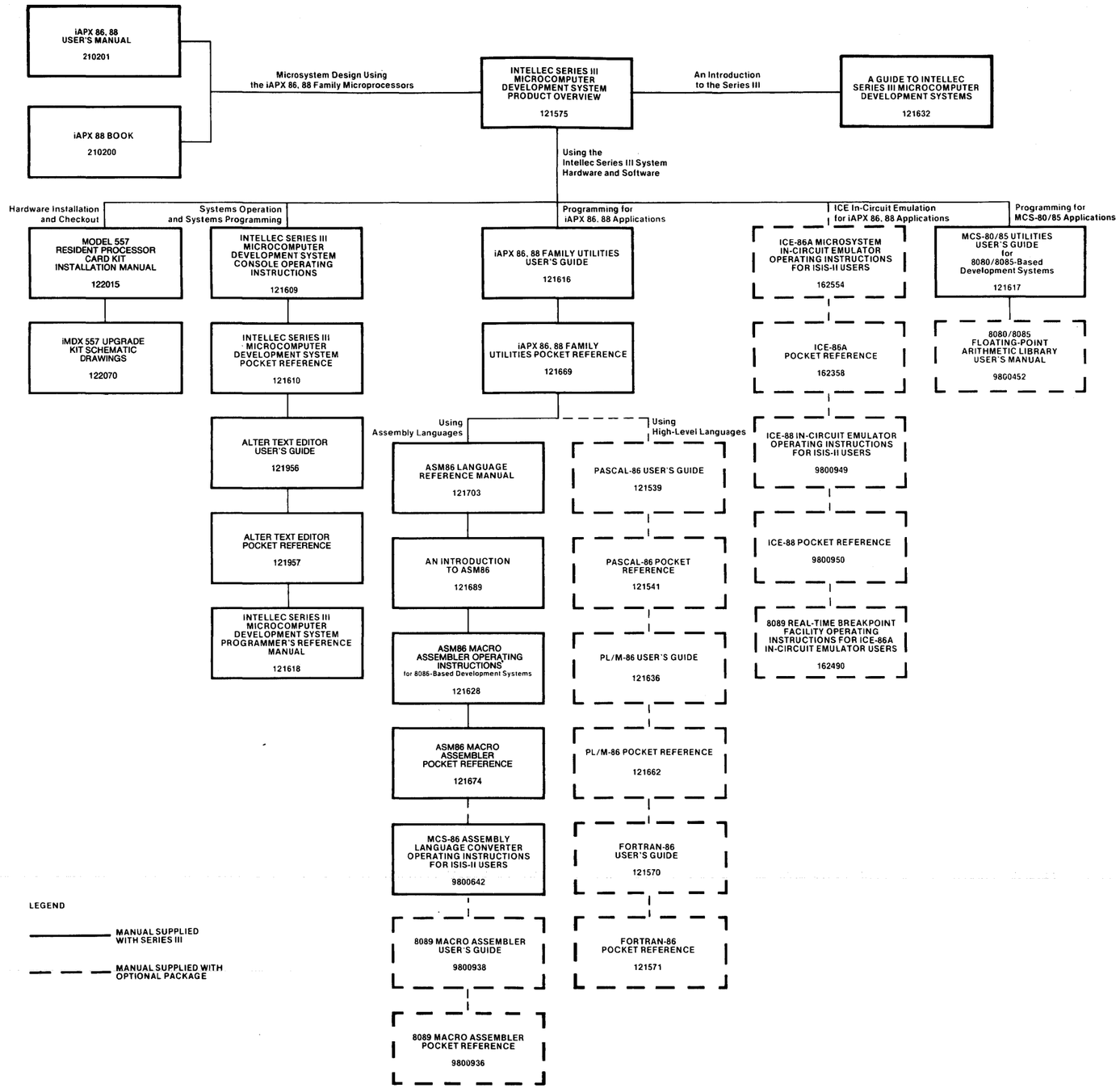


Figure 2-1. The Intellec® Series III Publications Library

121575-5

Once you have become familiar with system console operation and the ALTER Text Editor, you will find two other publications useful: the *Intellec Series III Microcomputer Development System Pocket Reference* and the *ALTER Text Editor Pocket Reference*. These contain quick summaries of the information in the corresponding manuals. You can keep these in your pocket or tape them to your system to remind you of details such as the exact form of console commands.

The Series III operating environment also makes available two sets of operating system services directly callable from programs. One set of services supports programs written for iAPX 86, 88 systems, and the other serves MCS-80/85 programs. If you are writing programs to run on the Series III and wish to use these routines (known as *system calls*), use the *Intellec Series III Microcomputer Development System Programmer's Reference Manual*. This manual describes the system environment that a computer program sees.

Programming for iAPX 86, 88 Applications

Intel provides a wide selection of programming languages for you to use in writing software for your iAPX 86 and iAPX 88 applications. Thus you can choose the languages best suited to your needs. The 8086 and 8088 processors have the same instruction set, so each translator produces code that will run on either an iAPX 86 or an iAPX 88 microcomputer system.

All Intel language translators for iAPX 86 and 88 systems produce compatible object code, so you can link together modules written in different languages. For instance, you might want to code your application in Pascal-86, then re-code the most time-critical or space-critical modules in assembly language. Then you can link all the modules together using the iAPX 86, 88 Family utilities.

For instructions on using the common linking and locating facilities after you have assembled or compiled your programs, consult the *iAPX 86, 88 Family Utilities User's Guide*. Once you have become familiar with this manual, you can use the *iAPX 86, 88 Family Utilities Pocket Reference* for quick reference.

Each language has its own set of manuals for you to use when writing programs and assembling or compiling them. For most of the languages, the reference information is combined into one manual, which contains both a definition of the language and instructions for operating the assembler or compiler and linking in any run-time support libraries. The information is in one manual because there is often much overlap between the information you need when writing programs and the information you need when compiling them. In the case of the 8086/87/88/186 Macro Assembler, the information is divided into three manuals, because the large size of a combined manual would have made it difficult to use.

Using Assembly Languages

The resident 8086/87/88/186 Macro Assembler produces object code for the 8086, 8088, and 186 processors, and also for the 8087 Numeric Coprocessor or 8087 Emulator. This assembler is provided with the Series III system, and runs on the Series III in 8086 mode for enhanced performance. When you are programming in 8086/87/88/186 Macro Assembly Language (ASM86), consult the *ASM86 Language Reference Manual* for complete reference information, after becoming familiar with the basic concepts provided in *An Introduction to ASM86*. Once you have become familiar with the information in these manuals, you can use the *ASM86 Macro Assembler Pocket Reference* for quick reference. When you are ready to add assembler controls and assemble your program, read the *ASM86 Macro Assembler Operating Instructions for 8086-Based Development Systems* for instructions.

If you are converting existing 8080/8085 Macro Assembly Language programs to 8086/87/88/186 Macro Assembly Language, refer to the *MCS-86 Assembly Language Converter Operating Instructions for ISIS-II Users* for instructions.

The 8089 Macro Assembler, a purchasable option, produces program modules for the 8089 I/O Processor. It runs on the Series III in 8085 mode. All the information you need when you are programming, adding controls, and operating the assembler is given in the *8089 Macro Assembler User's Guide.* Once you are familiar with the information in this manual, you can use the *8089 Macro Assembler Pocket Reference* for quick reference.

Using High-Level Languages

The high-level language compilers produce code for the 8086 and 8088 processors and contain run-time floating-point arithmetic support that will either emulate or produce code for the 8087 Numeric Data Processor. (The PL/M-86 compiler also produces code for the 186 processor.) The compilers run on the Series III in 8086 mode for higher performance.

If you are using Pascal-86, consult the *Pascal-86 User's Guide* for the information you need to write programs, add controls, operate the compiler, and perform any custom run-time interfacing you desire. The *Pascal-86 Pocket Reference* supplies quick reference information for you when you have become familiar with the information in the user's guide.

If you are using PL/M-86, read the *PL/M-86 User's Guide* for reference information when programming, adding controls, and operating the compiler. When you have become familiar with this manual, you can use the *PL/M-86 Pocket Reference* for quick reference.

If you are using FORTRAN-86, use the *FORTRAN-86 User's Guide* for programming, control, and compiler instructions. Once you are familiar with the information in this manual, consult the *FORTRAN-86 Pocket Reference* for quick reference.

ICE™ In-Circuit Emulation for iAPX 86, 88 Applications

When you are using the ICE-86A In-Circuit Emulator to develop and debug iAPX 86 applications, consult the *ICE-86A In-Circuit Emulator Operating Instructions for ISIS-II Users* for complete instructions, and use the *ICE-86A Pocket Reference* for quick reference information once you are familiar with the operating instructions. When using the ICE-88 in-circuit emulator for iAPX 88 applications, see the *ICE-88 In-Circuit Emulator Operating Instructions for ISIS-II Users* for complete instructions, and the *ICE-88 Pocket Reference* for quick reference. If you are developing and debugging iAPX 86 applications that include an 8089 I/O Processor, using the RBF-89 Real-Time Breakpoint Facility in conjunction with an ICE-86 Emulator, refer to the appropriate in-circuit emulator manual or pocket reference and to the *8089 Real-Time Breakpoint Facility Operating Instructions for ICE-86A In-Circuit Emulator Users.*

Programming for MCS®-80/85 Applications

The basic Series III software includes two packages useful to programmers writing software for MCS-80 and MCS-85 applications. These packages are the MCS-80/85 utilities for linking, locating, library management, and object module format conversion, and the 8080/8085 Floating Point Arithmetic Library (FPAL).

To use the MCS-80/85 utilities, see the *MCS-80/85 Utilities User's Guide for 8080/8085-Based Development Systems*. When you are writing and linking programs that call modules from FPAL, refer to the *8080/8085 Floating-Point Arithmetic Library User's Manual*.

The MCS-80/85 language translators, including the 8080/8085 Macro Assembler (ASM80), PL/M-80, FORTRAN-80, BASIC-80, Pascal-80, and iCIS-COBOL, are all purchasable options. Each of these products is shipped with appropriate user manuals; to save space, these manuals are not listed here. When you are writing programs and translating them, consult the manuals shipped with the translator you are using. When linking and locating translated modules, collecting them into libraries, or converting object module formats, see the MCS-80/85 utilities manuals just mentioned.

When using the ICE-85 and ICE-80 In-Circuit Emulators and the 8080/8085 Fundamental Support Package, consult the user manuals supplied with these products.

Using Other Products

In the interest of space, figure 2-1 does not show the user manuals for many other products you may purchase for your Series III, such as external flexible disk drive subsystems, the Universal PROM Programmer, and support packages for the MCS-51, MCS-48, and 2920 processors. You will receive the appropriate user manuals with each product you purchase.

CHAPTER 3
GLOSSARY OF SERIES III TERMS

This glossary defines terms used in manuals in the Series III publications library. Included are microprocessor terms, system-related terms, terms related to program code, terms pertaining to floating-point arithmetic, and terms describing physical devices.

Some terms you may encounter in the manuals of the Series III library are specific to a particular item of hardware or software or are used in a different way depending upon the item or product being described. For instance, *directive* has a different meaning for assembly-language programmers than it does for Pascal programmers. Since this is a system-wide glossary, such specific definitions are not given here. If you cannot find a particular term in this glossary or if the definition given here is too general, consult the manual in which the term appears.

absolute object module: an object module that contains all the necessary references to physical addresses, so that it can be executed by the target microprocessor.

absolute object module format: the bit-pattern format in which absolute object modules are encoded when they are produced by a locater such as LOC86 (for iAPX 86, 88 object modules) or LOCATE (for MCS-80/85 object modules). This is not an ASCII format, so linked and located object modules must be converted to hexadecimal format before they can be transferred to paper tape.

applications debugger: see *debugger*.

assembler: a program that translates assembly language source code into object code.

assembly language: a programming language whose instructions are closely related to the instruction set of the target processor. Often a single assembly language instruction represents a single machine instruction.

attribute: a disk file characteristic, specified in the directory containing the file, that can be either present or absent (i.e., set or reset). The four attributes of ISIS-II disk files are invisible, write-protect, format, and system.

base address: in an iAPX 86 or iAPX 88 microcomputer system, the 16-bit portion of an address that determines the location of the segment being addressed.

bound object module: an iAPX 86, 88 object module that can be loaded and executed on a Series III using the RUN loader. It can also be translated into an absolute object module by the LOC86 locater.

buffer: an area of memory reserved for expediting input and output, generally to or from disk.

byte bucket: a pseudo-device designated for input or output data that is to be discarded, that is, not saved or displayed. Its device name is :BB:.

caution: in a manual, a section of text introduced by the symbol

giving instructions necessary to avoid possible damage to equipment or loss of stored information.

class: in an iAPX 86 or iAPX 88 microcomputer system, a collection of logical segments that share common characteristics and can be referred to at locate time by specifying only the class name.

command: an instruction given to an interactive program from the console. If the command is the name of a program, it may also be referred to as an invocation line.

command tail: in a command, the portion that follows the program or command name and specifies particular options or arguments for the command.

compiler: a program that translates high-level language source code into object code.

connection: in the 8086-based environment of the Series III, a word specifying a device or file to be used by programs at run time.

console: the primary device used for interactive input and/or output in a system. For the Series III, the standard console input device is the keyboard, and the standard console output device is the CRT screen.

control: an instruction to an assembler, compiler, linker, locater, or other program that processes programs. It is given either in the invocation line for the assembler, compiler, linker, or locater, or (for an assembler or compiler) in a control line within the source program being processed.

control character: a single-character command given at the console keyboard by pressing the CNTL key and another key simultaneously. Control characters perform such functions as displaying the contents of the line editing buffer, deleting the contents of the line editing buffer, and stopping the console output.

control line: a line, within the text of a source program, that contains controls for the assembler or compiler rather than constructs that are part of the source language.

debugger: a program, generally interactive, that aids a programmer in debugging programs by providing facilities such as breakpoints, single-stepping, and access to the contents of variables. Its features are more sophisticated than those of a monitor.

default value: the input parameter or control value that is assumed by a program (such as an operating system command or compiler) if no value is explicitly given.

device name: a character string, recognized by the operating system, that identifies a physical input or output device (or a pseudo-device such as the byte bucket). It consists of four characters: a colon, followed by two alphanumeric characters, followed by another colon. Examples of device names are :F1:, :CI:, :LP:, :BB:, :HP:, and :R1:.

directory: a table, present on a disk, that lists all the files on the disk, giving their filenames, lengths, locations, and attributes. (When a directory is listed using the DIR command, the locations are not displayed.)

disk: a generic term denoting either a flexible disk (diskette) or a hard disk platter.

diskette: a single-density or double-density flexible disk.

driver: a routine that transfers data or performs other communication with an external device.

error: 1. a mistake in a source program that is severe enough to prevent generation of an object module. 2. a run-time condition that may cause the output of a program to be wrong, due to a logical mistake in the source program or due to invalid input.

exception: a run-time condition that may cause the output of a program to be wrong, due to a logical mistake in the source program or due to invalid input; also called a run-time error. The use of the term "exception" implies that in some cases, a routine can be called to handle the situation, and then processing can continue normally.

exception handler: a routine that is invoked when a run-time exception condition occurs. This routine performs processing to take care of the exception; depending on the application, it may send a message to the console and return control to the operating system, perform a fixup and allow the program to continue processing, or perform some other action.

exponent: in the internal representation of a real (floating-point) value, the part that designates (in binary) the base-2 exponent of the number.

extension: the optional part of a filename, consisting of one to three alphanumeric characters. If present, it must be preceded by a period.

external: said of a variable, procedure, function, or other source program entity that is referenced in one module, but is declared in another module.

external reference: in a source program module or an object module, a reference to a symbol or location in another module.

fatal error: an anomaly in the environment of an operating system, assembler, compiler, or other program that makes it impossible for the program to continue its processing.

file: a collection of information that may be read or written by an operating system command, assembler, compiler, or any other program. It may be an entire physical device, as in the case of a line printer, or it may be one of many files on a device, as on a disk.

filename: a character string, recognized by the operating system, that identifies a disk file. It consists of a name of one to six alphanumeric characters, followed by an optional period and extension of one to three alphanumeric characters. Examples of filenames are PROGA, PROGA.SRC, and ISIS.BIN.

floating-point number or *floating-point value:* see *real number* or *real value*.

general control: an assembler or compiler control that may appear either in the assembler or compiler invocation line or in a control line anywhere in the source program text, and that may be changed later in the source program.

group: in an iAPX 86 or iAPX 88 microcomputer system, a collection of logical segments that must be located within a 64K-byte range; that is, within a memory segment.

hexadecimal format: a bit-pattern format, used for data storage and transfer, which consists completely of ASCII codes. Absolute object modules must be converted into this format before they can be transferred to paper tape.

high-level language: a programming language whose instructions, or statements, are relatively independent of the instruction set of the target processor. One high-level language statement generally represents a sequence of machine or assembly language instructions.

iAPX: a prefix referring to Intel microcomputer systems built around a family of processors. Two iAPX processor families are currently defined:

iAPX 86: 8086 CPU based system
iAPX 88: 8088 CPU based system

With additional suffix information, configuration options within each iAPX system can be identified; for example:

iAPX 86/10: CPU alone (8086)
iAPX 86/11: CPU + I/O processor (8086 + 8089)
iAPX 88/20: CPU + math extension (8088 + 8087)
iAPX 88/21: CPU + math extension + I/O processor (8088 + 8087 + 8089)

The number preceding the slash identifies the CPU processor. The number following the slash identifies the microsystem configuration.

This nomenclature is intended as an addition to, rather than a replacement for, Intel's current part numbers. These new series-level descriptions are used to define the functional capabilities provided by specific configurations of the processors in the iAPX 86, 88 Family. The hardware used to implement each functional configuration is still described by referring to the individual parts involved. For instance, we speak of the 8086/8087/8088 Macro Assembler and the 8089 Macro Assembler, identifying each assembler by naming the processors for which it produces code. In contrast, LINK86, LOC86, LIB86, CREF86, and OH86 are referred to as the iAPX 86, 88 Family utilities, since these programs process object code for all iAPX 86, 88 systems.

In-Circuit Emulator: a module, composed of hardware, firmware, and software, that aids the designer of a microcomputer system in developing and debugging hardware and software together. It allows him to add individual hardware and software modules to the system as he completes them, substituting in-circuit emulator resources for the modules not yet prototyped. In-circuit emulators, such as the ICE-86A and ICE-88 emulators, run on Intellec development systems such as the Series III.

INCLUDE file: a separate file of source code that is inserted or included in the source input to an assembler or compiler, by means of an INCLUDE control in the source text.

interpreter: a program that directly executes high-level language source code or intermediate code, so that the source code need not be translated into object code to run or debug the program. An example is the BASIC-80 interpreter, which executes BASIC-80 source code. See also *compiler*.

interrupt: a signal, sent when an external event occurs, that causes a microprocessor CPU to stop what it is doing, perform a special routine to handle the interrupt, and then resume the interrupted processing where it left off.

interrupt handler: a routine that is invoked when an interrupt occurs. It performs whatever processing is necessary to take care of the event that caused the interrupt.

invocation line: the line of text used to invoke an assembler, compiler, linker, locater, or other program. An invocation line can also be called a command, though the latter term is generally used in discussing simpler programs.

librarian: see *library utility*.

library: a file that contains object modules and is created and maintained by a library utility such as LIB or LIB86.

library utility: a program, such as LIB or LIB86, that creates and maintains collections of object modules.

link: to combine together several assembled or compiled object modules to prepare them for locating and executing.

linker: a program, such as LINK or LINK86, that links object modules.

listing file: a file containing printed output produced by an assembler, compiler, linker, locater, or other program.

load: to place a program, in the form of an object module, in random-access memory so that it is ready to run.

loader: a program that loads object modules into memory so that they are ready to run. The kind of object module accepted for loading depends on the loader; for instance, the RUN loader accepts bound object modules, but some other loaders require absolute object modules.

load-time locatable (LTL) code: object code that can be located at load time on a Series III using the RUN loader, and that may refer to segment base addresses to access other segments in memory. Position-independent code (PIC) is always load-time locatable, but load-time locatable code is not always position-independent.

locate: to bind the code in linked object modules to memory addresses so that it may be executed.

locater: a program, such as LOCATE or LOC86, that locates object modules.

logical segment: a piece of an object module that is acted on by the programs that link and locate the object module.

LTL: see *load-time locatable (LTL) code*.

macro: a shorthand notation for a source text string. Macros are commonly used in assembly language programs, where they allow the programmer to specify a sequence of instructions by using one macro call.

macro assembler: an assembler that translates assembly-language programs containing macros. It expands all macros in the source code before translating the source code into object code.

memory segment: see *segment*.

Microsystem 80: a generic term referring to all microprocessors and associated peripheral chips marketed by Intel, except the 4004, 4040, 8008, 3000, and 2920.

model of segmentation: the scheme used by an iAPX 86, 88 high-level language translator, such as the PL/M-86 compiler, to divide program modules into logical segments, classes, and groups. The model of segmentation determines how many segments of code and data are allowed; whether some of these segments are combined by the translator into one 64K-byte group; and whether jumps, calls, and/or external references in the object code are in long or short form.

module: a separately assembled or compiled program unit.

monitor: a ROM-resident program that provides rudimentary system interaction and debugging capabilities such as program loading, I/O device configuration, access to memory locations, and execution with breakpoints.

NaN: in real (floating-point) number representation, one of several bit patterns that are not actual numbers. (NaN stands for "Not a Number.") The presence of NaN's in a computation generally indicates a real arithmetic exception condition.

non-system disk: a disk containing only those system files necessary for maintaining the directory of files on that disk, leaving more space for user files than is available on a system disk.

nonterminal symbol: in the syntax notation or other specification of a programming language or command language, a term (such as *label*, *filename*, or *expression*) standing for an item that must be filled in by the user.

normal zero: in real (floating-point) number representation, the real value whose exponent is of minimum value and whose significand is all zeros.

normalized: said of a real value if it is a normal zero or if its leading significant bit is one.

object code: program code that has been processed by an assembler or compiler, and that may have been processed further by a linker and/or locater.

object file: a file containing object code.

object module: a section of object code output by an assembler, compiler, linker, or locater. See also *absolute object module*, *relocatable object module*, and *bound object module*.

offset: in an iAPX 86 or iAPX 88 microcomputer system, the 16-bit portion of the address that gives the position of the addressed location within the segment specified by the base address.

operating system: a collection of programs that provide the functional (as opposed to physical) environment in which other programs do their work. The operating system is the functional environment in which assemblers, compilers, linkers, locaters, in-circuit emulation software, other Intel software, and user programs run on the Series III.

overlay: a section of code and/or data that is not present in memory at all times during a program run, but is loaded dynamically into memory when needed. Overlays save memory space at run time and permit programs to be larger than the available memory.

pathname: a character string, recognized by the operating system, that completely identifies a physical file (or a pseudo-file such as the byte bucket). It may be a device name such as :CI:, :LP:, or :BB: (for a non-disk file), a device name-filename combination such as :F1:PROGA.SRC (for a disk file), or a filename alone such as PROGB.SRC (for a disk file on the default device :F0:).

PIC: see *position-independent code*.

port: an arrangement of circuitry on a microprocessor that allows a byte or word of data to be input from, or output to, an external device.

position-independent code (PIC): object code that can be located at load time on a Series III using the RUN loader, and that does not refer to segment base addresses to access other segments in memory. Position-independent code is always load-time locatable (LTL), but load-time locatable code is not always position-independent.

primary control: an assembler or compiler control that must appear either in the assembler or compiler invocation line or in a control line preceding the first non-control line in the source program text, and that serves as a global control affecting the entire assembly or compilation. Compare with *general control.*

prompt: a single character or sequence of characters displayed on the console by an interactive program, such as the Series III operating system, to indicate that the program is ready to accept a command.

public: said of a variable, procedure, function, or other source program entity that some or all other modules may freely reference; also said of a symbol (such as an identifier) standing for such an entity.

real number or *real value:* a number internally represented by a sign bit, exponent, and significand, as in scientific notation. The exponent allows for the representation of extremely small and extremely large numbers.

reentrant: said of a section of program code that can be interrupted, then called again from an interrupt handler or another task before the first invocation is finished.

relocatable object module: an object module that contains no references to physical memory addresses. It can be translated into an absolute object module by a locater such as LOC86 (for iAPX 86, 88 object modules) or LOCATE (for MCS-80/85 object modules).

relocation: see *locate*.

root: in a program containing overlays, the section of code and data that is always present in memory. It usually consists of the main program module, frequently used routines, and the routine that loads the overlays.

segment: in an iAPX 86 or iAPX 88 microcomputer system, an area of memory that starts at a 16-byte boundary, consists of up to 64K contiguous bytes, and can be addressed without changing the base register used.

semantics: the set of rules for determining, given a syntactically acceptable program or command, what that program or command means — that is, what actions it will cause the processor to take.

sign bit: in the internal representation of a real value, the bit that indicates the sign of the number.

significand: in the internal representation of a real value, the part that designates (in binary form) the significant bits of the number.

source code: code written in an assembly language or in a high-level language such as PL/M or Pascal.

source file: a file containing source code.

stack: 1. a section of 8080, 8085, 8086, or 8088 memory that is accessed last in, first out, by means of special instructions that "push" data onto the stack or "pop" data off it. The stack is accessed by a processor register called the Stack Pointer. 2. a last-in, first-out memory structure similar to a microprocessor stack, but created and maintained by software.

stack pointer: see *stack*.

syntax: the set of rules defining what sequences of symbols make up acceptable programs in a given programming language or acceptable commands in a command language.

syntax notation: a symbolic notation used to define the syntax of a programming language or a command language.

system call: a call to an operating system procedure, such as CONSOL, from within a program.

system disk: a disk containing all the operating system software files.

terminal symbol: in the syntax notation or other specification of a programming language or command language, a symbol (such as a keyword, letter symbol, or punctuation symbol) that is to be used or entered verbatim.

text editor: a program, generally interactive, that allows a user to create and modify files containing text (ASCII characters grouped into lines).

translator: a program that translates program source code; that is, an assembler, compiler, or interpreter.

unsatisfied external reference: in a relocatable or bound object module, an external reference whose location is not yet determined. All external references must be satisfied before an absolute object module can be produced.

warning: 1. in the output from an assembler, compiler, linker, locater, or other program, an anomaly that may point to a problem, but that will not necessarily affect the validity of the program's output. A warning is less serious than an error. 2. in a manual, a section of text introduced by the symbol

giving instructions necessary to avoid possible injury to persons.

wild card designation: in an ISIS-II file control command, the use of an * or ? character in place of one or more characters in the name or extension part of a filename. The * character matches any number of characters, and the ? character matches any single character, when the system searches a directory for a filename.

workfile: a temporary file created by an assembler, compiler, or other program for its own internal use and deleted when the program finishes.

8080, 8085, 8086, or *8088 memory:* memory addressable by the 8080, 8085, 8086, or 8088 processor, respectively.

REQUEST FOR READER'S COMMENTS

Intel's Technical Publications Departments attempt to provide publications that meet the needs of all Intel product users. This form lets you participate directly in the publication process. Your comments will help us correct and improve our publications. Please take a few minutes to respond.

Please restrict your comments to the usability, accuracy, readability, organization, and completeness of this publication. If you have any comments on the product that this publication describes, please contact your Intel representative. If you wish to order publications, contact the Intel Literature Department (see page ii of this manual).

1. Please describe any errors you found in this publication (include page number).

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

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

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

5. Please rate this publication on a scale of 1 to 5 (5 being the best rating). _____

NAME _____ DATE _____

TITLE _____

COMPANY NAME/DEPARTMENT _____

ADDRESS _____

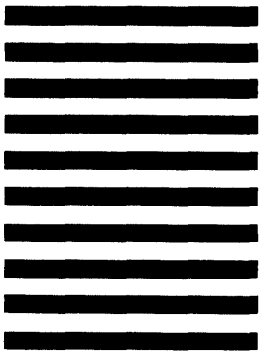
CITY _____ STATE (COUNTRY) _____ ZIP CODE _____

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 U.S.A.



BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1040 SANTA CLARA, CA

POSTAGE WILL BE PAID BY ADDRESSEE

Intel Corporation
Attn: Technical Publications M/S 6-2000
3065 Bowers Avenue
Santa Clara, CA 95051

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

Printed in U.S.A.