INTELLEC® SERIES IV MICROCOMPUTER DEVELOPMENT SYSTEM OVERVIEW

READ THIS FIRST

Order Number: 121752-001

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

A868/ 183/ 3K DD

REV.	REVISION HISTORY	DATE
-001	Original issue.	10/82

PREFACE

This manual describes the Intellec Series IV Microcomputer Development System and the optional hardware and software packages available from Intel. This manual also describes the Series IV publications library and provides a glossary of terms used throughout the library.

This manual contains two chapters and a glossary:

- Chapter 1, "Intellec Series IV Development Solution," describes the Series IV system and its associated options, and the packaging of the Series IV.
- Chapter 2, "Series IV Publications Library," describes the library of manuals for the Series IV and its associated options.
- "Glossary" describes the vocabulary used throughout the publications library.

CONTENTS

FIGURES

CHAPTER 1
INTELLEC® SERIES IV DEVELOPMENT SOLUTION

Introduction

This chapter describes the Series IV development system and the optional hardware and software packages available from Intel.

Development systems from Intel are hosts for a unique combination of hardware and software tools that are designed to increase your development productivity and lessen your time to market. 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. Top-down, modular design enables you to develop hardware and software modules that work together. With the aid of Intel high-level programming languages, assemblers, symbolic debuggers, 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.

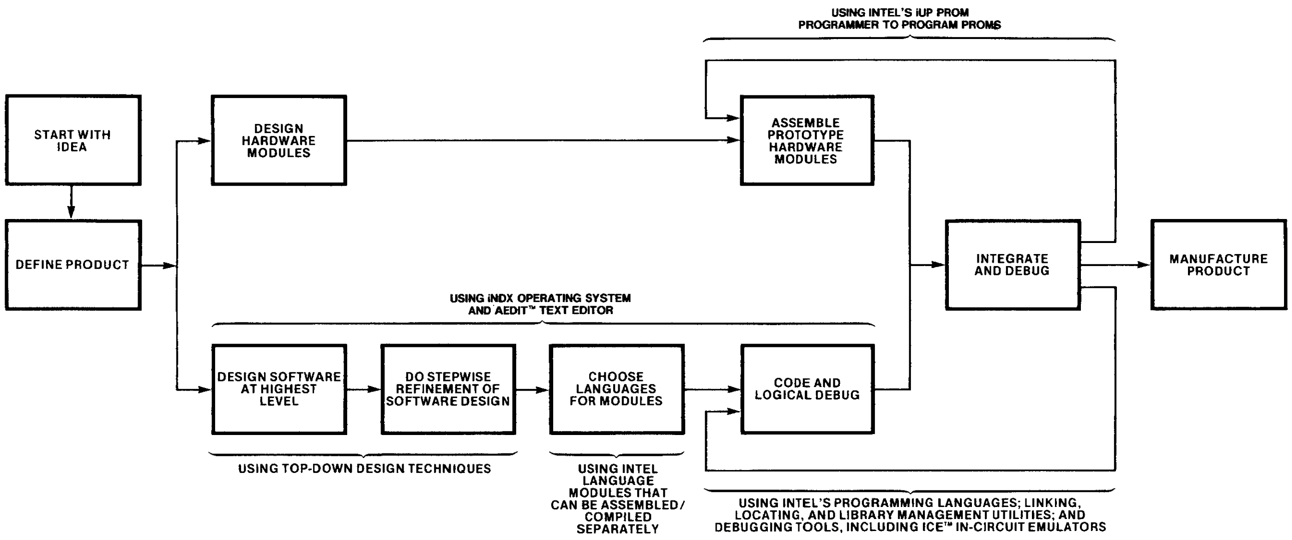
USING INTEL'S iUP PROM PROGRAMMER TO PROGRAM PROMS
START WITH IDEA
DEFINE PRODUCT
DESIGN HARDWARE MODULES
ASSEMBLE PROTOTYPE HARDWARE MODULES
INTEGRATE AND DEBUG
MANUFACTURE PRODUCT
USING iNDX OPERATING SYSTEM AND AEDIT™ TEXT EDITOR
DESIGN SOFTWARE AT HIGHEST LEVEL
DO STEPWISE REFINEMENT OF SOFTWARE DESIGN
CHOOSE LANGUAGES FOR MODULES
CODE AND LOGICAL DEBUG
USING TOP-DOWN DESIGN TECHNIQUES
USING INTEL LANGUAGE MODULES THAT CAN BE ASSEMBLED/ COMPILED SEPARATELY
USING INTEL'S PROGRAMMING LANGUAGES; LINKING, LOCATING, AND LIBRARY MANAGEMENT UTILITIES; AND DEBUGGING TOOLS, INCLUDING ICE™ IN-CIRCUIT EMULATORS

Figure 1-1. Microcomputer System Development Process 121752-1

The Intellec Series IV is a complete development solution for your microcomputer applications based on the following Intel microprocessors:

- The iAPX 86,88 family (applications based on the 8086, 8088, 8087, and 8089 microprocessors)
- The MCS®-80/85 family (applications based on the 8080 and 8085 processors)
- The MCS®-48/51 family (applications based on the 8048 and 8051 processors)
- The iAPX 432
- The iAPX 286
- The iAPX 186
- The 2920 Signal Processor

Intel eliminates the risk of development system obsolescence by guaranteeing that new products are compatible with past products and upgradable to future products. Intel provides a spectrum of tools, ranging from standalone systems to networks of specialized workstations.

The Series IV extends the user interface of previous Intellec development systems and is software compatible with them. Thus, the Series IV provides an easy upgrade path for users.

With the addition of an Ethernet communications board set and associated cables, the Series IV standalone development system can become a workstation in the Network Development System-II (NDS-II). For more information on the NDS-II, see the *NDS-II Network Development System Overview.*

Series IV Overview

The Series IV is both an 8086/8088-based development system and an 8080/8085-based development system. The host execution mode is 8086/8088, which runs under the iNDX operating system. To execute an 8080/8085 program, you would enter 8085 execution mode by invoking the ISIS-IV utility. ISIS-IV is a subsystem of iNDX and provides the 8080/8085 interface on the Series IV; its user interface is compatible with previous ISIS-based development systems.

The human interface of the Series IV (i.e., all user interaction with the system) is designed for both novices and experts. You interface with the system by entering commands at the keyboard. Furthermore, you do not have to remember the command syntax because you are guided by the command Syntax Guide. If you do not understand the syntax of a command, you can invoke the HELP facility for a description of the command. The HELP facility and Syntax Guide are described in more detail later in this chapter.

iNDX Operating System

The iNDX operating system employs a hierarchical file system and allows both foreground and background processing. The Syntax Guide assists you in entering commands and the Help Text provides a brief description of each command.

Hierarchical File System

The iNDX file system is structured hierarchically and provides file sharing and protection features. The hierarchical structure allows logical grouping of data. The structure resembles an inverted tree. The root of the system is called the Logical System Root (the first tier of figure 1-2). The System Root logically "connects" the

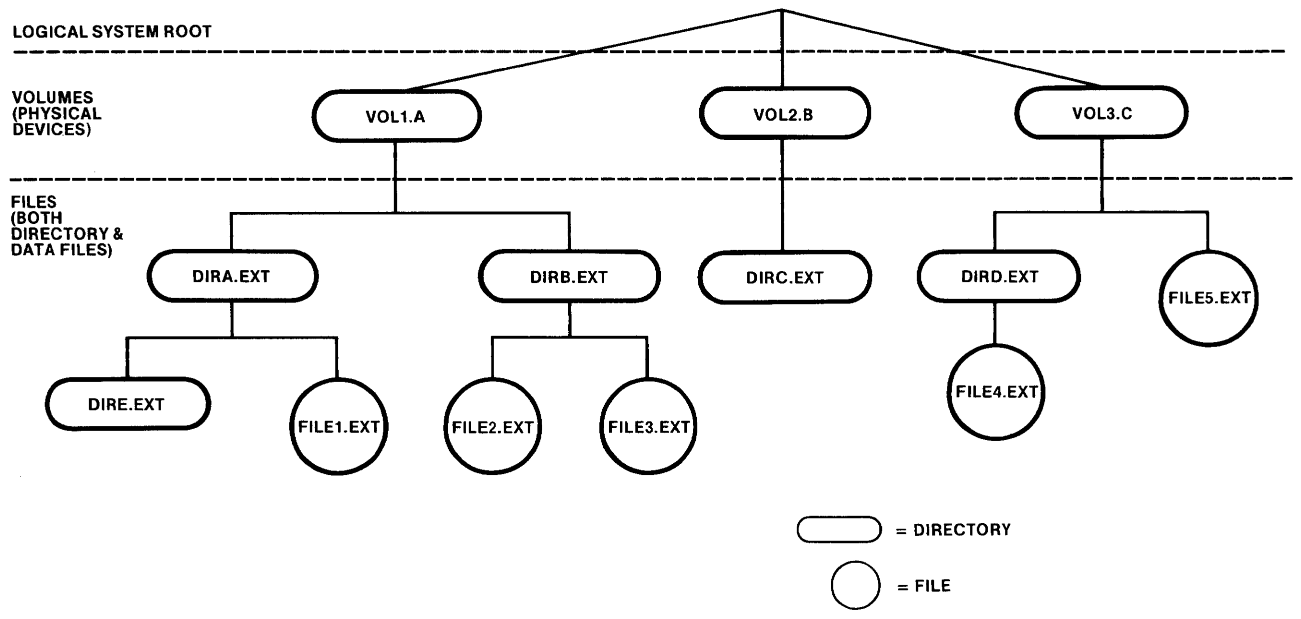

Figure 1-2. Hierarchical File Structure 121752-2

volumes within the file system. Each volume (the second tier of figure 1-2) corresponds on a one-to-one basis to a physical mass storage device. Volumes are further divided into files. Files (the third tier of figure 1-2) can be either directory files or data files. Directory files contain references to further directory files and data files. Data files contain only data.

You do not need to know the physical location of your files to address them. Each file can be addressed by a pathname, which is a character string recognized by the operating system.

The iNDX file system provides file protection features in the form of access rights. Thus, if the Series IV is being used as a single terminal, multi-user system, files can be protected from problems such as accidental addressing and destruction.

Foreground/Background Processing

Foreground/background processing greatly improves your productivity. While one program is executing in the background, you can be executing another program in the foreground.

All user interactive jobs are foreground jobs. The foreground job is explicitly activated when you log on to the system and ends when you log off. Background jobs are executed simultaneously with foreground jobs. They differ only in the fact that background jobs are not capable of user interaction. (Therefore, an interactive program such as an editor or debugger can only be run in the foreground.)

Syntax Guide

The Syntax Guide is part of the Command Line Interpreter (CLI). The Command Line Interpreter is the program with which you directly interact; it provides the following facilities: command line editing, menu display, and keyword completion.

There are two distinct command line editing modes: single and batch. Single is the normal mode in which the CLI accepts a command from the keyboard. Batch is an extension of single command mode, which allows the editing of an entire command file with the Syntax Guide.

The iNDX operating system presents commands, options, and executable files as menu entries on the screen. Up to eight entries can be displayed at one time. These entries are accessed by pressing the associated soft keys labeled F0-F7 on the Series IV keyboard. The first (leftmost) entry is associated with F0, the second with F1, etc.

These entries can also be accessed by typing the capital letters of each entry as it is displayed on the screen. For example, you need only type AE for the editing program AEDIT. The Syntax Guide automatically inserts the remaining letters of the keyword into the command text. This feature is called keyword completion.

Help Text

For each menu entry, the accompanying Help Text is obtained by holding down the SHIFT key while pressing the appropriate soft key. The Help Text describes a given command, option, or executable file.

AEDIT Text Editor

An important feature of the Series IV system is the AEDIT text editor. AEDIT runs under iNDX and can be invoked from the console. AEDIT allows you to

- Display and scroll text on the screen
- Move to any position in the text file or to any point on the screen instantly
- Correct typing mistakes as you type
- Rewrite text by typing new letters over old ones
- Make insertions and deletions easily
- Find any string of characters and substitute another string
- Move or copy sections of text within a file or to another file
- Create macros to execute several commands at once, thereby simplifying repetitive editing tasks
- Scan listing files while editing your primary file
- Indent text automatically
- View lines over 80 characters long

DEBUG-88

Also included with the Series IV system is DEBUG-88—a software applications debugger for iAPX 86,88 programs. The DEBUG-88 command set enables you to

- Initialize DEBUG-88 and load your program's symbolic names and line numbers in the DEBUG-88 symbol table and clear all break registers
- Set starting and stopping points for execution of your program
- Execute your program in either continuous or single step mode

- Set, display, and alter 8086/8088 registers, flags, memory locations, and stack content
- Display the contents of memory locations as disassembled instructions
- Search for and display user defined program labels and line numbers

Series IV Configurations

The Series IV development system is available in two configurations (the iMDX 44X and the iMDX 43X), so you can choose the one most appropriate for your application needs. Each system includes a 2000 character CRT and a detachable, full ASCII keyboard with cursor controls and upper/lowercase capability. Built-in interfaces are provided for a serial I/O channel, a line printer, and the Universal PROM programmer (iUP 200/201). Figure 1-3 shows the Series IV system.

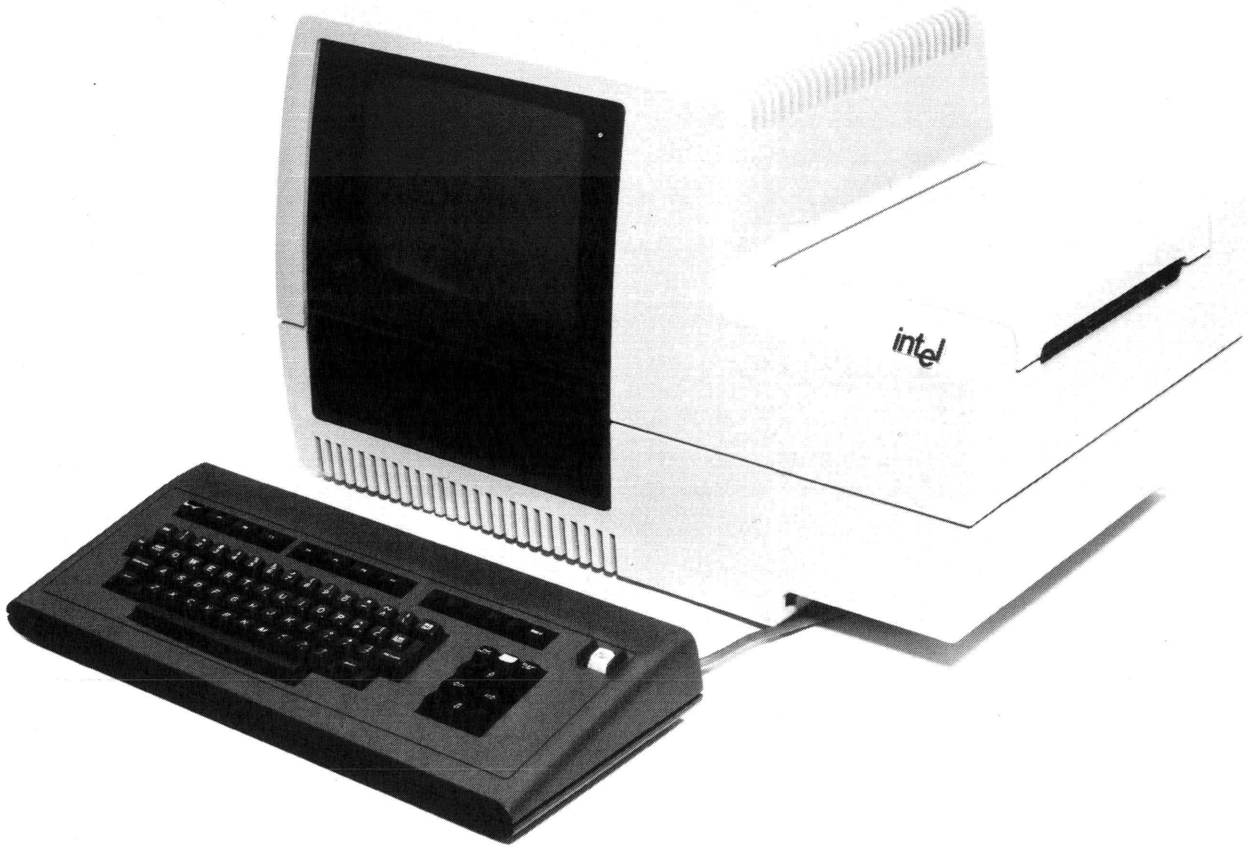

Figure 1-3. Series IV System 121752-3

iMDX 44X

The iMDX 44X high-performance system contains an 8085, 8088, and 8086 CPU with a minimum of 256K of user memory. The iMDX 44X provides both foreground and background processing. Furthermore, this system provides the option of two 5¼ inch flexible disk drives each with 640K bytes of storage capacity or one 5¼ inch flexible disk drive and one 5¼ inch Winchester disk drive with 10 megabytes of storage capacity, for a total storage capacity of 10640K bytes.

If your iMDX 44X system has two integral flexible disk drives, you can also purchase a 32 megabyte (formatted capacity) peripheral Winchester disk subsystem for added storage.

iMDX 43X

The iMDX 43X contains both an 8085 and an 8088 CPU with a minimum of 192K bytes of user memory. This system provides both foreground and background processing. The iMDX 43X also provides the option of two 5¼ inch flexible disk drives or one 5¼ inch flexible disk drive and one 5¼ inch Winchester disk drive. (This system also supports the optional Winchester disk subsystem.)

System Firmware and Software

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

- iNDX operating system
- ISIS-IV subsystem
- AEDIT, 8086-based text editor
- CREDIT, 8085-based text editor
- DEBUG-88 software applications debugger
- Intellec Series IV confidence test
- ASM86 Macro Assembler
- LINK86, LOC86, LIB86, OH86 software development utilities
- 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
- CONV86 Assembly Language Converter for converting programs from 8080/8085 Macro Assembly Language to ASM86 Macro Assembly Language
- 8085 monitor providing low-level system services and debugging for MCS-80/85 programs
- Utility programs for MCS-80/85 applications (LINK, LOCATE, LIB, OBJHEX, HEXOBJ)
- 8080/8085 Floating-Point Arithmetic Library (FPAL)

Optional Packages

Intel provides a host of additional hardware and software packages to aid in the microcomputer system development provided by the Series IV.

Since the Series IV is upward compatible with the Model 800, Series II, and Series III, most Intel hardware and software available for these systems will also run on a Series IV. And, with the addition of a communications board set and cables, all Intellec models can become workstations in the Network Development System-II.

Add-On Disk Storage

For many applications, additional disk storage may be desirable. Intel provides an optional 32 megabyte (formatted capacity) peripheral Winchester disk subsystem for added storage. (Systems with an integral Winchester disk will support the optional Winchester disk subsystem.)

Intel's Model 740 hard disk subsystem (5440 drive) is also compatible with the Series IV.

Software and Emulators for iAPX 86,88 Product Development

Software support for your product development is extensive. Intel provides a wide selection of programming languages for iAPX 86,88 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.

Intel's compilers and assemblers produce compatible linkable and relocatable object modules. You can code applications in the most appropriate language, then link object modules from different languages to form the final program. For example, you might want to code your application in Pascal-86, recode the most time-critical or space-critical modules in assembly language, and then link all the modules together using the iAPX 86,88 Utilities.

Assembly Languages

ASM86 is Intel's powerful "high-level" Macro Assembler. This assembler produces object code for the 8086, 8088, and 186 processors, and for the 8087 numeric coprocessor or 8087 emulator. This assembler is provided with the Series IV system, and runs on the Series IV in 8086/8088 mode for enhanced performance.

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 ASM86 assembly language or in any of the high-level languages for iAPX 86,88 systems. The 8089 Macro Assembler runs on the Series IV in 8085 execution mode.

High-Level Languages

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

PL/M-86 is a structured high-level language designed by Intel. It allows you to program at a level closer to your microprocessor hardware by combining high-level block structure with hardware-level features such as indirect addressing, bit manipulation, and direct I/O. PL/M-86 is generally more suitable for systems programming.

Pascal-86 is a high-level, block-structured language that conforms to the ISO Draft Proposal for standard Pascal. It also has extensions for microprocessor port I/O, interrupt processing, and separate compilation of program modules. Pascal-86 is ideally suited to applications programming such as data processing.

FORTRAN-86 is a superset of ANSI FORTRAN 77, with extensions such as port I/O and interrupt processing to aid in microprocessor development. It is also compatible with Intel's FORTRAN-80. FORTRAN-86 has a rich set of arithmetic operations that make it best suited to scientific and numeric applications.

High-Level Language Debugger

PSCOPE is an interactive, symbolic debugger for Pascal-86 and PL/M-86 programs. Operations are performed on source statements, procedure entry points, labels, and variables, as opposed to machine instructions and code or data addresses.

PSCOPE improves productivity in the debug phase of development and produces more reliable software. PSCOPE allows you to perform extensive tests and consistency checks on your program, and it automates much of the testing.

In-Circuit Emulators

In-Circuit Emulator (ICE) modules enable you to develop and debug your microcomputer system hardware and software concurrently, 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.

The ICE-86A emulator—an enhanced version of the ICE-86 emulator—is provided for iAPX 86 microsystems. The ICE-86A emulator incorporates the 8089 Real-Time Breakpoint Facility. This facility was formerly a separate product for debugging systems that included an 8089 processor.

The ICE-88A emulator is provided for iAPX 88 microsystems. The ICE-88A emulator does not incorporate the 8089 breakpoint facility, but it is still available as a separate product for use with systems that include an 8089 processor.

Software and Emulators for MCS®-80/85 Product Development

Series IV software includes two packages useful to programmers writing software for MCS-80 and MCS-85 applications: the MCS-80/85 Utilities and the 8080/8085 Floating-Point Arithmetic Library (FPAL).

The MCS-80/85 Utilities provide linking, locating, library management, and format conversion of your object modules. The Floating-Point Arithmetic Library extends the arithmetic capabilities of your programs. You can incorporate various floating-point operations using simple procedure calls.

Assembly Languages

ASM 8080/8085 is the Macro Assembler for products based on Intel's 8080 and 8085 microprocessors.

High-Level Languages

PL/M-80 is a structured, high-level programming language designed by Intel specifically for use on the 8080/8085 microprocessors. It is ideally suited to systems and applications programming.

Pascal-80 is a powerful programming tool for the 8080/8085 microprocessors. It provides such features as separate compilation of modules, port I/O, and string manipulation.

FORTRAN-80 is based on ANSI FORTRAN 77, with extensions for microprocessor development. It allows efficient arithmetic calculations using Intel's numerics facilities.

BASIC-80 allows you to apply the computational and I/O capabilities of microprocessors to a wide range of business, numeric analysis, and data processing applications.

iCIS-COBOL is based on standard ANSI COBOL and contains several extensions that are specifically oriented to the microcomputer environment. COBOL is best suited to commercial and administrative data processing.

In-Circuit Emulators

The ICE-85b emulator is provided for MCS-80/85 microsystems. Intel also provides MULTI-ICE, which performs asynchronous debugging of a multiprocessor design by operating two ICE hardware modules from one Intellec system.

Software and Emulators for MCS®-48/51 Product Development

You can also develop MCS-51 and MCS-48 family microcomputer applications on the Series IV. Development tools for these applications include the MCS-48 and UPI-41 Assembler, the MCS-51 Macro Assembler, 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 will also run on the Series IV. These tools include the 2920 Assembler (AS2920), the 2920 Simulator (SM2920), and the 2920 Signal Processing Applications Software Compiler (SPAS20).

Universal PROM Programmer (iUP 200/201)

The iUP 200/201 Universal PROM Programmer is provided to program and verify all Intel programmable ROMs (PROMs). In addition, the iUP 200/201 programs the PROM memory portions of the 8751, 8748, and 8749 microcomputers, the 8741 UPI processor, the 8755 PROM and I/O chip, and the 2920 Signal Processor. Programming and verification are initiated from the Series IV console and are controlled by the Intel PROM Programming Software (iPPS) supplied with the iUP 200/201.

Mainframe Link for Distributed Development

The Series IV supports a 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.

iAPX 432 Micromainframe System

The iAPX 432 micromainframe system is a set of processors that share access to a common pool of memory and to each other. The iAPX 432 32-bit General Data Processor (iAPX 432 GDP) is the first processor whose architecture supports software transparent, multiprocessor operation.

Designed for large information management applications, the new iAPX 432 computer technology is an integration of hardware and software, using an object-oriented methodology. This design significantly reduces the life-cycle costs of complex computer applications.

iAPX 286 Processor

The iAPX 286 processor is code compatible with the 8086. The iAPX 286 allows up to a six-times performance increase over the 8086 and offers unique on-chip memory management and memory protection features to increase system throughput and reliability.

iAPX 186 Processor

The iAPX 186 is a highly integrated, enhanced processor with twice the performance of the 8086. The 186 integrates the common system functions of the 8086, so code compatibility is maintained.

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.

Packaging of the Series IV

The Series IV system comes complete with a System-to-System Link; software diskettes, which include the iNDX operating system disk and all operating system extensions; and the following technical manuals:

- *Intellec Series IV Installation and Checkout Manual*, order number 121757
- *Intellec Series IV Microcomputer Development System Overview*, order number 121752
- *Intellec Series IV Operating and Programming Guide*, order number 121753
- *Intellec Series IV Pocket Reference*, order number 121760
- *Intellec Series IV ISIS-IV User's Guide*, order number 121880
- *Intellec Series IV ISIS-IV Pocket Reference*, order number 121890
- *AEDIT Text Editor User's Guide*, order number 121756
- *AEDIT Text Editor Pocket Reference*, order number 121767
- *ISIS CREDIT CRT-Based Text Editor User's Guide*, order number 9800902
- *CREDIT CRT-Based Text Editor Pocket Reference*, order number 9800903
- *DEBUG-88 User's Guide*, order number 121758
- *iAPX 88 Book*, order number 210200
- *iAPX 86,88 User's Manual*, order number 210201

- *iAPX 86, 88 Family Utilities User's Guide*, order number 121616
- *MCS-80/85 Family User's Manual*, order number 121506
- *MCS-80/85 Utilities User's Guide for 8080/8085-Based Development Systems*, order number 121617
- *8080/8085 Floating-Point Arithmetic Library User's Manual*, order number 9800452
- *An Introduction to ASM86*, order number 121689
- *ASM86 Macro Assembler Operating Instructions for 8086-Based Systems*, order number 121628
- *ASM86 Language Reference Manual*, order number 121703
- *ASM86 Macro Assembler Pocket Reference*, order number 121674

CHAPTER 2 SERIES IV PUBLICATIONS LIBRARY

Introduction

This chapter describes the structure of the Series IV publications library—an integrated set of technical manuals that describe the Series IV development system and its associated options.

The manuals are grouped into different categories, which include

- Hardware installation and checkout manuals, such as the *Intellec Series IV Installation and Checkout Manual*
- Systems operation and programming manuals, such as the *Intellec Series IV Operating and Programming Guide*
- Processor overview manuals, such as the *iAPX 86,88 User's Manual*
- Language programming manuals, such as the *Pascal-86 User's Guide*

Each programming language has its own set of manuals for writing programs and assembling or compiling them. For some of the languages, the reference information is combined into one manual. These manuals contain language reference information, compiler or assembler operating instructions, and instructions for linking in any run-time support libraries. For other languages, the information is divided into two manuals. In these manuals, the language reference information is separate from the compiler or assembler operating instructions and linking instructions.

Several manuals have corresponding pocket references. Pocket references contain a brief summary of the information in the corresponding manual. For example, the *AEDIT Text Editor Pocket Reference* contains information on how to invoke AEDIT and a list of the editing and macro commands.

Each manual corresponds to a different user activity and is written at the level of a typical user performing that activity. For example, the *Intellec Series IV Operating and Programming Guide* is written for a new Series IV user who may not be familiar with Intel systems. In contrast, the *ASM86 Macro Assembler Operating Instructions for 8086-Based Systems* is written for a user who is assembling programs coded in ASM86 Macro Assembly language. This manual assumes that you have a basic knowledge of assembly language programming, and that you are already familiar with the iAPX 86,88 microsystem family, the Series IV operating system, and the ASM86 Macro Assembly language.

Figure 2-1 is a tree diagram showing the manuals in the Series IV publications library and their relationship to each other. The shaded manuals make up the literature kit that is supplied with the Series IV system. The non-shaded manuals are provided with optional packages. (Because of limited space, not all of the manuals supplied with optional packages are shown.)

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, data sheets, magazine articles describing Intel products, and brochures on new products.

Hardware Installation and Checkout

If you have purchased a new Series IV, consult the *Series IV Installation and Checkout Manual* for information on installing and maintaining the Series IV system.

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

System Operation and System Programming

Once your system has been installed and you are ready to operate it, consult the *Series IV Operating and Programming Guide.* This manual provides the following information necessary to Series IV operators and programmers:

- It describes how to start up the system, use the flexible disk and Winchester disk units, and give commands from the console to the operating system. In other words, it describes the system environment that a human operator sees.
- It also describes the operating system services that are directly accessible from other programs. These services, also known as system calls, allow you to write programs that run under the iNDX operating system (8088/8086 execution mode).

If you are writing 8080/8085-based programs, refer to the *ISIS-IV User's Guide.* This manual describes the 8080/8085 execution mode of the Series IV. It includes information on the 8085-resident Monitor and the ISIS-IV subsystem calls.

Both the *Series IV Operating and Programming Guide* and the *ISIS-IV User's Guide* have corresponding pocket references for quick reference.

Read these manuals before you use any of the manuals described in the following sections.

Text Editing

An important feature of the Series IV system is the AEDIT text editor, which runs under iNDX and can be invoked from the console. When you are ready to insert text into your system, consult the *AEDIT Text Editor User's Guide.* This manual describes how to invoke AEDIT and describes its many features with detailed coverage of editing commands and macro commands. For quick reference, consult the *AEDIT Text Editor Pocket Reference.*

If you are editing text using the CREDIT text editor under ISIS-IV, see the *CREDIT Text Editor User's Guide.*

Debugging

Once you know how to operate your system and have written programs using the AEDIT text editor, you will want to debug these programs. The *DEBUG-88 User's Guide* describes how to invoke DEBUG-88 and gives a detailed description of the symbolic debugging capabilities of DEBUG-88.

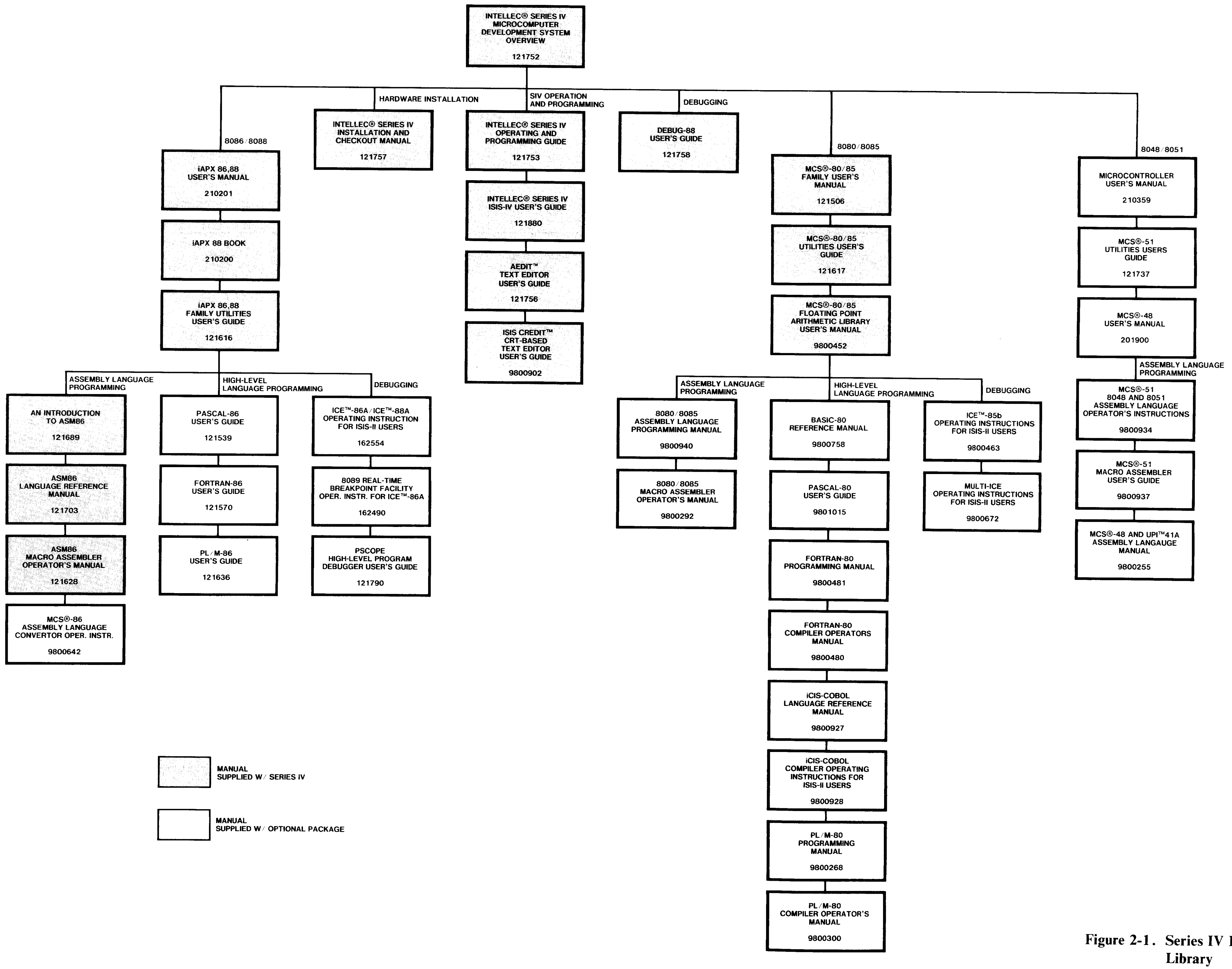
INTELLEC® SERIES IV MICROCOMPUTER DEVELOPMENT SYSTEM OVERVIEW 121752
8086/8088
HARDWARE INSTALLATION
SIV OPERATION AND PROGRAMMING
DEBUGGING
8080/8085
8048/8051
iAPX 86,88 USER'S MANUAL 210201
iAPX 88 BOOK 210200
iAPX 86,88 FAMILY UTILITIES USER'S GUIDE 121616
INTELLEC® SERIES IV INSTALLATION AND CHECKOUT MANUAL 121757
INTELLEC® SERIES IV OPERATING AND PROGRAMMING GUIDE 121753
INTELLEC® SERIES IV ISIS-IV USER'S GUIDE 121880
AEDIT™ TEXT EDITOR USER'S GUIDE 121756
ISIS CREDIT™ CRT-BASED TEXT EDITOR USER'S GUIDE 9800902
DEBUG-88 USER'S GUIDE 121758
MCS®-80/85 FAMILY USER'S MANUAL 121506
MCS®-80/85 UTILITIES USER'S GUIDE 121617
MCS®-80/85 FLOATING POINT ARITHMETIC LIBRARY USER'S MANUAL 9800452
MICROCONTROLLER USER'S MANUAL 210359
MCS®-51 UTILITIES USERS GUIDE 121737
MCS®-48 USER'S MANUAL 201900
ASSEMBLY LANGUAGE PROGRAMMING
HIGH-LEVEL LANGUAGE PROGRAMMING
DEBUGGING
AN INTRODUCTION TO ASM86 121689
ASM86 LANGUAGE REFERENCE MANUAL 121703
ASM86 MACRO ASSEMBLER OPERATOR'S MANUAL 121628
MCS®-86 ASSEMBLY LANGUAGE CONVERTOR OPER. INSTR. 9800642
PASCAL-86 USER'S GUIDE 121539
FORTRAN-86 USER'S GUIDE 121570
PL/M-86 USER'S GUIDE 121636
ICE™-86A/ICE™-88A OPERATING INSTRUCTION FOR ISIS-II USERS 162554
8089 REAL-TIME BREAKPOINT FACILITY OPER. INSTR. FOR ICE™-86A 162490
PSCOPE HIGH-LEVEL PROGRAM DEBUGGER USER'S GUIDE 121790
ASSEMBLY LANGUAGE PROGRAMMING
HIGH-LEVEL LANGUAGE PROGRAMMING
DEBUGGING
8080/8085 ASSEMBLY LANGUAGE PROGRAMMING MANUAL 9800940
8080/8085 MACRO ASSEMBLER OPERATOR'S MANUAL 9800292
BASIC-80 REFERENCE MANUAL 9800758
PASCAL-80 USER'S GUIDE 9801015
FORTRAN-80 PROGRAMMING MANUAL 9800481
FORTRAN-80 COMPILER OPERATORS MANUAL 9800480
iCIS-COBOL LANGUAGE REFERENCE MANUAL 9800927
iCIS-COBOL COMPILER OPERATING INSTRUCTIONS FOR ISIS-II USERS 9800928
PL/M-80 PROGRAMMING MANUAL 9800268
PL/M-80 COMPILER OPERATOR'S MANUAL 9800300
ICE™-85b OPERATING INSTRUCTIONS FOR ISIS-II USERS 9800463
MULTI-ICE OPERATING INSTRUCTIONS FOR ISIS-II USERS 9800672
ASSEMBLY LANGUAGE PROGRAMMING
MCS®-51 8048 AND 8051 ASSEMBLY LANGUAGE OPERATOR'S INSTRUCTIONS 9800934
MCS®-51 MACRO ASSEMBLER USER'S GUIDE 9800937
MCS®-48 AND UPI™41A ASSEMBLY LANGAUGE MANUAL 9800255
MANUAL SUPPLIED W/ SERIES IV
MANUAL SUPPLIED W/ OPTIONAL PACKAGE

Figure 2-1. Series IV Publications Library

121752-4

The iAPX 88,86 Family of Microprocessors

The *iAPX 86,88 User's Manual* describes the iAPX 86,88 family of microprocessors and how they are configured into systems. Consult this manual if you are evaluating the iAPX 86,88 family to determine their suitability for your design. It is the definitive source of information on the iAPX components, with moderately detailed coverage of software topics. This manual is your reference source during system design and implementation.

This manual assumes a basic knowledge of microprocessor hardware and provides the following information:

- Describes the architecture, instruction sets, and capabilities of the 8086 and 8088 CPU's, the 8089 I/O processor, and the 8087 numeric data processor
- Defines the configuration of iAPX 86,88 microcomputer systems
- Gives hardware reference information and device specifications
- Outlines the available software and systems support
- Provides programming examples and application notes

The *iAPX 88 Book* describes the 8088 processor—the processor designed for 8-bit applications requiring high performance and low cost. This book gives detailed descriptions of the 8088 CPU, programming architecture, and hardware design using the 8088 CPU. Examples of system design are also provided.

Programming

For instructions on using the common linking and locating facilities after you have assembled or compiled your programs, refer to the *iAPX 86,88 Family Utilities User's Guide*. For quick reference, see the *iAPX 86, 88 Family Utilities Pocket Reference.*

Assembly Languages

When you are programming in ASM86 Macro Assembly language, consult the following manuals:

- *An Introduction to ASM86* is designed to teach the fundamentals of constructing ASM86 source modules and provide the conceptual background needed to write ASM86 code. This manual provides numerous drawings, examples, and syntax diagrams.
- The *ASM86 Language Reference Manual* provides detailed reference information to supplement the Introduction manual. For quick reference, refer to the *ASM86 Macro Assembler Pocket Reference.* When you are ready to add assembler controls and assemble your program, consult the *ASM86 Macro Assembler Operating Instructions for 8086-Based Systems.*
- If you are converting existing 8080/8085 Macro Assembly Language programs to 8086/8087/8088 Macro Assembly Language, consult the *MCS-86 Assembly Language Converter Operating Instructions for ISIS-II Users.*
- If you are programming, adding controls, and operating the 8089 Macro Assembler, consult the *8089 Macro Assembler User's Guide.* For quick reference, see the *8089 Macro Assembler Pocket Reference.*

High-Level Languages

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. For quick reference, see the *Pascal-86 Pocket Reference.*

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. For quick reference, see the *PL/M-86 Pocket Reference.*

If you are using FORTRAN-86, consult the *FORTRAN-86 User's Guide* for reference information when programming, adding controls, and operating the compiler. For quick reference, see the *FORTRAN-86 Pocket Reference.*

High-Level Language Debugger

For information on symbolic debugging of Pascal-86 and PL/M-86 programs using the PSCOPE high-level language debugger, consult the *PSCOPE High-Level Language Debugger User's Guide.* For quick reference, see the *PSCOPE Pocket Reference.*

ICE™ In-Circuit Emulation

When you are using the ICE-86A In-Circuit Emulator to develop and debug iAPX 86 applications, consult the *ICE-86A/ICE-88A In-Circuit Emulator Operating Instructions for ISIS-II Users.* For quick reference, use the *ICE-86A/88A Pocket Reference.*

When using the ICE-88A In-Circuit Emulator for iAPX 88 applications, consult the *ICE-86A/ICE-88A In-Circuit Emulator Operating Instructions for ISIS-II Users.* For quick reference, consult the *ICE-86A/88A Pocket Reference.*

If you are using the 8089 Real-Time Breakpoint Facility in conjunction with the ICE-88A emulator, refer to the *8089 Real-Time Breakpoint Facility Operating Instructions for ICE-86A In-Circuit Emulator Users.*

The MCS®-80/85 Family of Microprocessors

The *MCS-80/85 Family User's Manual* describes the MCS-80/85 family of microprocessors and how they are configured into systems. This is the definitive source of information on the MCS-80/85 components.

Programming

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 the 8080/8085 Floating-Point Arithmetic Library (FPAL), consult the *8080/8085 Floating-Point Arithmetic Library User's Manual.*

Assembly Languages

If you are using the 8080/8085 Macro Assembler, consult the *ISIS-II 8080/8085 Macro Assembler Operator's Manual* for information on operating procedures, controls, list file formats, PL/M linkage conventions and error messages. This manual is a companion to the *8080/8085 Assembly Language Programming Manual*, which contains full details on the assembly language and assembler directives.

High-Level Languages

If you are using PL/M-80, consult the *ISIS-II PL/M-80 Compiler Operator's Manual*. This manual describes the PL/M compiler running under the ISIS operating system and the steps needed to execute compiled programs. This manual is a companion to the *PL/M Programming Manual*, which describes the PL/M-80 language as implemented by the PL/M-80 compiler.

If you are using FORTRAN-80, consult the *ISIS-II FORTRAN-80 Compiler Operator's Manual*. This manual describes the operating procedures for the ISIS-II FORTRAN-80 compiler and run-time libraries. This manual is a companion to the *FORTRAN-80 Programming Manual*, which describes the FORTRAN-80 language as implemented by the FORTRAN compiler.

If you are using BASIC-80, consult the *BASIC-80 Reference Manual*. This manual describes the features and conventions of BASIC-80 as implemented on the Series IV system under ISIS.

If you are using Pascal-80, refer to the *Pascal-80 User's Guide*. This manual describes the Pascal-80 system, the language features and operating instructions, and provides example programs.

If you are using iCIS-COBOL, consult the *iCIS-COBOL Compiler Operating Instructions for ISIS-II Users* for information on the operating procedures for the iCIS-COBOL compiler running under ISIS. This manual is a companion to the *iCIS-COBOL Language Reference Manual*, which describes the iCIS-COBOL language as implemented on the iCIS-COBOL compiler.

ICE™ In-Circuit Emulation

When using the ICE-85b In-Circuit Emulator and the 8080/8085 Fundamental Support Package, see the *ICE-85b Operating Instructions for ISIS-II Users.*

When using the Multi-ICE enhancement to Intel's ICE-85b emulator, consult the *Multi-ICE Operating Instructions for ISIS-II Users.*

The MCS®-48/51 Family of Microprocessors

The *Microcontroller User's Manual* provides complete information on the architecture, instruction sets, and memory of the MCS-48 and MCS-51 families. Application examples are also included.

Programming

When writing programs for the 8048 microprocessors, consult the following manuals:

- *MCS-48 User's Manual*
- *MCS-48 & UPI 41A Assembly Language Manual*

When writing programs for the 8051 microprocessor, consult the following manuals:

- *MCS-51 Utilities User's Guide*
- *MCS-51 Macro Assembler User's Guide*
- *MCS-51 8048 & 8051 Assembly Language Operator's Instructions*

ICE™ In-Circuit Emulation

When using the ICE-49 emulator, consult the *ICE-49 In-Circuit Emulator Operating Instructions for ISIS-II Users.*

When using the ICE-51 emulator, consult the *ICE-51 In-Circuit Emulator Operating Instructions for ISIS-II Users.*

iAPX 432 Micromainframe System

The *Intel 432 System Summary* is the first publication anyone interested in the iAPX 432 should read. It introduces an array of topics related to the iAPX 432, which are described in detail in the following reference manuals:

- *Introduction to the iAPX 432 Architecture*
- *iAPX 432 Object Primer*
- *Introduction to the Intel 432 Cross Development System*
- *Reference Manual for the Ada Programming Language*
- *Reference Manual for the Intel 432 Extensions to Ada*
- *Intel 432 CDS VAX Host User's Guide*
- *iMAX 432 Reference Manual*
- *System 432/600 System Reference Manual*
- *System 432/600 Diagnostic Software User's Guide*
- *Mainframe Link for Distributed Development User's Guide*
- *Intel 432 CDS Workstation User's Guide*
- *iAPX 432 General Data Processor Architecture Reference Manual* (Advanced Partial Issue)
- *iAPX 432 Interface Processor Architecture Reference Manual*
- *Intel 432 CDS/Ada Support Packages User's Guide*
- *Asynchronous Communication Link User's Guide*

iAPX 286 Processor

The architecture of the iAPX 286 processor is described in the following manuals:

- *Introduction to the iAPX 286*
- *iAPX 286/10 High Performance Microprocessor with Memory Management and Protection*
- *iAPX 286 Programmer's Reference Manual*

A set of four User's Guides describes the iAPX 286 Utilities, Builder, Simulator, and Monitor:

- *iAPX 286 Utilities User's Guide*
- *iAPX 286 System Builder User's Guide*
- *iAPX 286 Simulator User's Guide*
- *iAPX 286 Monitor User's Guide*

In addition, the following language manuals are written specifically for the 286:

- *ASM286 Language Reference Manual*
- *ASM286 Macro Assembly Operating Instructions*
- *PL/M-286 User's Guide*

Using Other Products

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

GLOSSARY

This glossary defines terms used in the Series IV publications library. These terms pertain to microprocessors, network systems, program code, file descriptors, floating-point arithmetic, and physical devices.

absolute object module: An object module that contains references to physical addresses, so that it can be directly 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 locator such as LOC86.

access rights: Protection attributes associated with a file. It defines what rights (read, write, delete for data files; list, add-entry, delete for directory files) users (Owner/World/Superuser) have to a file.

add-entry access: Access right pertaining to a directory file. Specified user can create a directory or a file within a directory.

AEDIT: 8086/8088-based, screen-oriented text editor with menu style command prompts.

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: An ISIS file characteristic (e.g., invisible, write-protect, format, system), specified in the directory containing the file, that can be either present or absent (i.e., set or reset).

background job: Execution of commands separate and independent from the foreground interactive job. Background jobs do not involve console interaction.

batch: A list of commands that are processed sequentially (and without user interaction) by a computer.

bound object module: See *load time locatable module.*

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. Its device name is :BB:.

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

CAUTION

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

CLI: See *Command Line Interpreter*

command: An instruction given to an interactive program from the console.

Command Line Interpreter (CLI): The human interface program with which the user directly interacts. The CLI is responsible for breaking the commands entered at the terminal into appropriate segments, and then interpreting them as correct instructions to the system.

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

Communication Controller Board Set: Board set located in the workstation that allows the workstation to communicate with the Network Manager.

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

connection: A word specifying a device or file to be used by programs at run time.

console: The primary device used for interactive input to and/or output from a system. (The part of the computer system used for communication between computer operator and CPU.) The Series IV console consists of a CRT and a keyboard.

control character: A single-character command given at the console keyboard by pressing the CONTROL key and another key simultaneously.

CPIO (Central Processor and I/O): 8088-based I/O and CPU board.

CREDIT: 8080/8085-based text editor that runs under ISIS.

data file: A file other than a directory file. A data file cannot contain another data file or a directory file. See also *directory file.*

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.

DEBUG-88: Interactive software applications debugger for iAPX 86,88 programs.

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

delete access: Access right pertaining to a directory file or data file. Specified user can delete the directory or the file.

directory: A logical collection of files stored on a disk.

directory file: A file whose contents represent a directory. A directory file may contain data files or other directory files (a logical collection of files stored on a disk).

directory listing: A table that lists all the files in a directory giving their filenames and other attributes. When a directory is listed using the DIR command, the locations are not displayed.

disk: A generic term denoting either a flexible disk or a hard disk where directory and data files can be stored.

display access: Access right pertaining to directory files. Specified user can display the directory.

driver: A routine that transfers data or performs other communications with an external device.

Ethernet: (Registered trademark of Xerox Corporation.) The physical and data link layers of the local area network architecture that provides a means of transmitting high-speed data exchange between computers and other digital devices within a moderate-sized geographic area.

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 processing can continue.

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.

EXPORT: Command that submits a user-defined job to a job queue at the Network Resource Manager (NRM) for remote execution.

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

fatal error: An error 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. An iNDX filename is the final path component of a pathname. See *pathname.*

flexible data entry: The ability to reenter any parameter while preserving all intervening entries.

foreground job: Job created at LOGON that user interacts with.

granularity: The number of bytes in a sector.

Help Text: Describes a given command option or executable file that is invoked by holding the SHIFT key while pressing the appropriate soft key.

hierarchical file structure: File structure resembling an inverted tree that allows logical grouping of data. The structure consists of a hierarchy of protected directory and data files stored in volumes.

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.

IEU (ISIS Execution Unit): Board providing an ISIS environment on which current 8080/8085 software will execute.

IMPORT: Command that allows a workstation to receive and execute jobs from other workstations.

In-Circuit Emulator (ICE): A module, composed of hardware, firmware, and software, that aids in the development and debugging of hardware and software. It allows you to add individual hardware and software modules to the system, substituting In-Circuit Emulator resources for the modules not yet prototyped.

iNDX (Intel Network Distributed Executive): The Series IV operating system.

interactive program: A program that communicates with the user while it is running.

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. 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 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, locator, or other program. An invocation line can also be called a command, though the latter term is generally used in discussing simpler programs.

ISIS-IV: Subsystem of iNDX, which provides 8080/8085 execution environment on the Series IV.

job: A list of commands grouped and run together in order to perform a specific function.

librarian: See *library utility.*

library: A file that contains object modules and is created and maintained by a library utility.

library utility: A program that creates and maintains collections of object modules.

link: To combine several assembled or compiled object modules, which can then be bound for execution.

linker: A program that combines object modules.

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

LNAME: See *logical name.*

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.

load time locatable module: An object module that contains information that the loader can process. The LTL module contains references to logical addresses, which are then made absolute by the Loader, so that the module can be executed on the target microprocessor.

locate: To assign absolute addresses to code in linked object modules so that it may be executed on a target system. (Located files cannot be executed on the Series IV).

locator: A program that locates object modules by binding them to memory addresses. (The locator assigns actual (physical) addresses to a program.)

logical: Symbolic—as opposed to physical—representation of an object, device, connection, etc.

logical device name: A label (:CO:, :LP:) assigned to a physical device or directory.

logical name: An arbitrary character string that can be associated with a directory. Thus, you can reference a long pathname by its much shorter logical name.

LOGON: Command that identifies you to the system, allowing you to begin a job.

LOGOFF: Command that terminates your current foreground job and logs you off the system.

LTL module: See *load time locatable module.*

macro: A group of often used instructions treated as a single entity. Macros allow you 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.*

menu: List of commands, options, and executable files that appears at the bottom of the Series IV screen. These commands, options, and executable files are executed by pressing the associated soft key (or the capital letters within the menu field).

message line: Display area used to present messages to the user so that the text area is not disturbed.

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

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.

MON-85: Program monitor used with 8080/8085 programs.

NDS-II: See *Network Development System-II.*

Network Development System-II (NDS-II): An Ethernet-based local networking system, allowing interconnection of individual development systems for shared network facilities, a distributed file system, printer spooling, distributed job control, etc.

Network Resource Manager (NRM): The resource control station on the NDS-II network. The NRM maintains the shared files, acts as print server, and performs network job control functions.

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.

NRM: See *Network Resource Manager.*

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 locator.

object file: A file containing object code.

object module: A section of object code output by an assembler, compiler, linker, or locator. See also *absolute object module*, *relocatable object module*, and *load time locatable module.*

operating environment: Pertains to foreground, background, or remote jobs. When a job is run in the background (background job) or exported (remote job), it does not inherit the logical names or file connections of the foreground job.

operating system: A collection of resource allocation 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, locators, in-circuit emulators, other Intel software, and user programs run.

OS status field: Used for display of job control messages, including name of background job and fatal error messages.

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.

packet: A data transmission of a specified length.

parameter: A variable that is usually given a constant value for a specific program to run.

password: A 1- to 14-character string made up of the set of printable ASCII characters (hex codes 021-07EH) except /. The password, combined with the user ID, is used to verify that the person logging on to the system has been identified correctly to the network and is a valid user.

pathname: A character string recognized by the operating system. A pathname completely identifies a 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), or a combination of names such as /SYSTEM.VOL/MYDIR/MYFILE (for a network file), or a filename alone such as PROGB.SRC.

physical device: An I/O device or storage medium (e.g., hard disk, line printer, disk drive, etc.).

physical device name: A label (WD1, :LP:, etc.) assigned to a device.

primitive: Descriptive of lowest level of machine instruction; lowest level of language translator.

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

private workstation: A workstation logged on to the network. The private workstation services its own jobs exclusively. It can also send jobs to the NRM and have them executed by other workstations in the network.

prompt: A single character or sequence of characters displayed on the console by an interactive program, indicating that the program is ready to accept a command. The iNDX prompt is a right angle bracket (>); the ISIS prompt is a hyphen (-).

prompt line: Used for display of menu entries.

public workstation: A workstation logged onto the network that services network job queues. It accepts jobs from the NRM for execution, but it cannot send jobs to the NRM.

query: Interactive questioning process between the operating system and the user. For example, if you are deleting a file and you request QUERY, iNDX will query you before deleting the file.

read access: Access right pertaining to a data file. Specified user can read file.

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 locator.

relocation: See *locate.*

remote file: Files and directories that reside on the network hard disk file system (on the shared hard disk).

remote job: Job executed at a remote public workstation; i.e., at another workstation in the network.

root: 1. 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. 2. The root directory of a volume.

scrolling field: Text area of the screen, which consists of lines 1-23.

segment: A piece of an object module.

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.

single command mode: When commands are entered from the keyboard one at a time (as opposed to batch mode).

soft key: A function key whose meaning changes, depending on which menu prompt line is displayed.

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.

spooling: Procedure of temporarily storing data on disk files until the CPU is ready for additional processing of the information.

SPU (Slave Processor Unit): A board providing a high speed 8086 environment for the Series IV.

standalone system: The terminal and the support software that make up the "box" used for program development; the standalone system is not part of any other system or network.

string: 1. A sequence of bytes, the first of which is the length byte and contains the number of bytes in the sequence (not including the length byte). A length of zero specifies a null string. 2. A sequence of ASCII characters.

SUBMIT: Command used to execute a command file in the foreground.

subtree: All the data and directory files that descend from a single parent directory.

Superuser: The user designated as the network administrator. Certain commands (for system administration functions) can be entered only by the Superuser at the NRM terminal.

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 Guide: Program used to help the user construct command lines.

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

SYSGEN: See *System Generation.*

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

system directory: A directory containing all the operating system software files.

system disk: Disk that contains the operating system files.

System Generation: An interactive system configuration process whereby you define or modify your network configuration, peripheral device configuration, and system parameters, and create the operating system software for the NRM and workstations.

system root: The symbolic connection point for all volumes of a hierarchical file system.

system bootstrap: The process of loading the appropriate operating system.

terminal: The physical hardware component that makes up the "box" containing a keyboard and a display device.

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 you to create and modify files containing text (ASCII characters grouped into lines).

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

tree structure: See *Hierarchical File System.*

UDI: See *Universal Development Interface.*

Universal Development Interface (UDI): An interface to the operating system that provides a standard set of support primitives. Thus, implementation of language translators, ICE products, etc., can proceed independently of the implementation of the operating systems that host them.

user: A person who is able to access system resources by logging on to a system with a valid username and password.

user ID: A unique 16-bit number that is assigned by the Superuser to a user for access checking and accounting purposes. The user ID, combined with a password, is used to verify that the person logging on to the network is a valid user.

username: The name that identifies a system user to the operating system. Syntactically it consists of 1 to 14 characters from the set of printable ASCII characters (hex codes 021H-07EH), except /.

utility: A software tool (program) that aids in the development of programs. For example, LINK allows you to link program modules, LOCATE allows you to locate program modules, etc.

volume: 1. Logical collection of directory files that reside on a storage medium. 2. A physical object containing files, such as a disk pack or reel of tape.

warning: 1. In the output from an assembler, compiler, linker, locator, or other program, a warning may point to a problem, but 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

WARNING

giving instructions necessary to avoid possible injury to persons.

wild card designation: In a file control command, the use of an * or ? character in place of one or more characters. The * character matches any number of characters and the ? character matches any single character when the system searches a directory for a filename. In an ISIS filename, the designation may appear in the name or the extension. In an iNDX filename, the designation may appear only in the final component of the pathname.

Winchester: A disk device based on Winchester technology.

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

workstation: A hardware box and the support software comprising a user station for program development in the network environment.

INDEX

Within this index, *f*, or *ff* after a page number means *and the following page (or pages).*

intel®

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

Printed in U.S.A.